Vysokoškolská skripta

VŠE

FAKULTA INFORMATIKY A STATISTIKY

DATOVÉ MODELOVÁNÍ A NÁVRH RELAČNÍ DATABÁZE

SBÍRKA ŘEŠENÝCH ÚLOH

Ing. Dušan Chlapek, Ph.D.

Ing. Jan Kučera, Ph.D.

RNDr. Helena Palovská, Ph.D.

Vysoká škola ekonomická v Praze

Datové modelování a návrh relační databáze: Sbírka řešených úloh

Dušan Chlapek

Jan Kučera

Helena Palovská

OECONOMICA
Nakladatelství VŠE

2019

Autoři

Ing. Dušan Chlapek, Ph.D.
Vysoká škola ekonomická v Praze, Katedra informačních technologií

Ing. Jan Kučera, Ph.D.
Vysoká škola ekonomická v Praze, Katedra informačních technologií

RNDr. Helena Palovská, Ph.D.
Vysoká škola ekonomická v Praze, Katedra informačních technologií

Oponenti

prof. Ing. Václav Řepa, CSc.

Ing. Jan Kaltoun

Poděkování

Tato publikace byla vytvořena za podpory projektu Interní rozvojové soutěže Vysoké školy ekonomické v Praze *Elektronické nástroje pro podporu výuky předmětu 4IT218 – Databáze* (IRS/F4/32/2019).

ISBN 978-80-245-2331-6

Obsah

Úvod

Tento učební text je určen primárně pro studenty předmětu 4IT218 – Databáze a obsahuje sadu řešených příkladů, které usnadní pochopení činností a pravidel spojených s analýzou a návrhem struktury relační databáze. Učební text doplňuje přednášky a umožní studentům lépe porozumět vykládané látce a současně umožní ověřit a procvičit si získané vědomosti. Část textu a řešených příkladů věnovaná konceptuálnímu modelování může být využita v řadě dalších předmětů zaměřených na návrh datových skladů a BI řešení.

Při řešení příkladů jsou použity konkrétní nástroje dostupné pro studenty Vysoké školy ekonomické v Praze. Je třeba zdůraznit, že se jedná pouze o příklady nástrojů a že cílem není studenty učit pracovat s konkrétními nástroji, ale naučit je principy, které jsou dostupné i v řadě dalších programových nástrojů. Proto jsou pro návrh konceptuálního datového modelu a fyzického datového modelu použity alternativně dva nástroje: Sybase Power Designer (dále jen Power Designer) a Oracle Data Modeler (dále jen Data Modeler), přičemž druhý z uvedených nástrojů je volně dostupný. Použití dvou alternativních nástrojů tak čtenáři ukáže mírně odlišné přístupy používané tvůrci konkrétních programových nástrojů pro návrh databáze a současně umožní snazší orientaci v případných dalších nástrojích zaměřených na stejnou činnost, tj. na návrh struktury databáze.

V textu použité nástroje pro návrh datových modelů umožňují generovat SQL kód pro různé databázové platformy, jako jsou například Oracle, Microsoft SQL Server, DB/2, Informix, MySQL, případně i různých verzí. V učebním textu je uveden a komentován SQL kód generovaný do prostředí databázové platformy Oracle, který je používán v rámci výuky předmětu 4IT218 – Databáze. Současně čtenář může použít v textu vytvořené konceptuální datové modely pro vygenerování SQL kódu i pro další, v konkrétním nástroji podporované, databázové platformy.

Jak již bylo uvedeno, tak fyzický návrh databáze a SQL kód pro vytvoření databázových objektů v rámci řešení jednotlivých úloh je v tomto učebním textu přizpůsoben pro databázovou platformu Oracle. Kromě toho, že při transformaci konceptuálního datového modelu na fyzický datový model v nástrojích Power Designer a Data Modeler byla použita transformační pravidla přizpůsobená pro tuto databázovou platformu, byly při návrhu uplatňovány i následující zásady:

- V databázovém prostředí Oracle není podporován datový typ logické hodnoty (boolean), a proto je tento datový typ reprezentován jako sloupec s datovým typem `VARCHAR2(1)`, tj. obsahující textový řetězec o délce jednoho bytu. Pomocí omezení typu `CHECK` je pak omezeno, že se ve sloupci mohou vyskytovat pouze hodnoty „A“ (značící „ano“) a „N“ (značící „ne“).
- V databázovém prostředí Oracle je datový typ `CHAR` (textový řetězec pevné délky) reprezentován jako `VARCHAR2` (textový řetězec proměnlivé délky), přičemž textové řetězce, jejichž skutečná délka nedosahuje stanovené délky textového řetězce, jsou na stanovenou délku doplněny prázdnými znaky (Oracle, 2015). Z tohoto důvodu není datový typ `CHAR` využíván, a pro textové řetězce je využíván pouze datový typ `VARCHAR2`. Pro zjednodušení práce v nástrojích Power Designer a Data Modeler byla tato zásada uplatněna již na úrovni konceptuálního datového modelu, aby při jeho transformaci na fyzický datový model již byl volen datový typ `VARCHAR2` automaticky.

V tomto učebním textu jsou uvedena i další doporučení pro práci s daty, zejména pro použití vybraných atributů. U některých příkladů jsou uvedeny alternativní způsoby řešení spolu s vysvětlením jejich výhod a nevýhod.

Přejeme čtenáři, aby mu tento učební text pomohl proniknout do problematiky návrhu databází. A pevně věříme, že znalosti a zkušenosti získané v průběhu studia i díky těmto řešeným příkladům bude moci čtenář využít i ve svém odborném životě.

V Praze dne 30. září 2019

Tým autorů

1 Datové modelování a návrh databáze

V současné době stále rychleji roste objem vytvářených dat a tím roste i význam schopností data[1] ukládat a zpracovávat. Prostředí a způsobů ukládání dat je celá řada. Existují prostředí, která umožní ukládat pouze dobře strukturovaná data, ale i prostředí, která umožňují kombinovat strukturovaný tvar dat s méně strukturovanou formou, až po taková, která pracují s daty z prvního pohledu zcela nestrukturovanými. Tato prostředí se označují pojmem databáze, který označuje integrovanou počítačově zpracovávanou množinu perzistentních dat. Pro práci s databází se využívají programy a služby, tzv. systémy řízení báze dat (dále také SŘBD)[2], které umožňují zejména vytvářet databázi a definovat[3] (i rušit a měnit) v databázi používané databázové objekty[4]. SŘBD umožní s daty pracovat, tj. provádět operace vkládání dat, aktualizace dat, výběry dat a rušení dat. SŘBD také umožní spravovat jednotlivé databáze, řídit přístup uživatelů k datům a případně optimalizovat prováděná zpracování. Databázové systémy jsou složeny ze SŘBD a jednotlivých databází. Jednotlivé databázové systémy mohou pracovat s daty organizovanými dle vymezených logických modelů: například existují relační databázové systémy, hierarchické databázové systémy, grafové databázové systémy, bezschématové systémy. Databázové systémy jsou často členěny podle typu zpracování na databázové systémy zaměřené na zpracování velkého počtu relativně malých transakcí[5] (OLTP systémy) a na databázové systémy zaměřené na zpracování menšího počtu, zato relativně složitých dotazů (OLAP systémy). Ať již jsou data ukládána v jakémkoliv databázovém systému a zpracovávána jakýmkoliv typem zpracování, nesnižuje to nikterak potřebu poznat, pochopit a orientovat se v datech, která potřebujeme zpracovávat. Proto se vytváří popis nezávislý na použitém typu zpracování, technologii či konkrétním databázovém systému. Takový popis má podobu modelu, pro nějž požíváme označení konceptuální schéma nebo konceptuální datový

[1] Data pro potřeby tohoto textu chápeme jako formalizované a fyzicky zaznamenané znalosti, poznatky zkušenosti, výsledky pozorování činností, procesů, jevů a prvků.
[2] V angličtině označované jako Database Management System.
[3] Nejčastěji k práci s databázovými objekty slouží příkazy `CREATE` (vytváření objektu), `ALTER` (změna struktury objektu, `DROP` (zrušení databázového objektu. Například při rušení konkrétní relační tabulky (tedy databázového objektu typu Table) dojde k odstranění všech dat i celé tabulky ze struktury databáze.
[4] Databázovými objekty mohou být například relační tabulky (tables), constraints, pohledy (views), triggery, indexy.
[5] Pojem transakce je odvozen z bankovnictví. V databázovém prostředí je definován jako logická posloupnost operací, která je promítána do databáze jako celek. Transakce slouží k zajištění konzistence databáze a k řízení víceuživatelského přístupu k datům. Transakce mají řadu vlastností, ale pro jejich výklad odkazujeme k přednáškám.

model. Konceptuální schéma lze do značné míry chápat jako ontologii, která umožňuje poznat, popsat a sdílet informace o struktuře navrhovaného či používaného obsahu databáze různorodou komunitou lidí – od uživatelů budoucího informačního systému, přes datové architekty, datové experty, analytiky, až po softwarové inženýry, designery, vývojáře či testery.

Návrh databáze probíhá v několika úrovních, které dále stručně připomeneme.

1.1 Konceptuální úroveň návrhu

Konceptuální úroveň návrhu databáze slouží jako:

- nástroj pro poznání světa, který potřebujeme v databázi zachytit;
- prostředek komunikace mezi členy řešitelského týmu a diskuse s uživateli budoucí aplikace budované nad navrhovaným databázovým systémem;
- podklad pro návrh struktury databáze;
- forma dokumentace existující databáze.

Návrh na konceptuální úrovni je zcela nezatížený cílovým implementačním prostředím (databázovým systémem). Díky tomu, že se jedná o technologicky a implementačně nezávislý pohled na zkoumaný výsek světa, pro který chceme navrhovat databázi, je to výchozí pevný bod, který lze využít pro návrh v různých typech a verzích databázových systémů. Tedy při návrhu na konceptuální úrovni ještě nepotřebujeme vědět, zda vlastní realizace bude v relačním databázovém systému či jiném typu databázového systému, a můžeme se plně soustředit na poznání výseče světa, pro kterou budeme databázový systém navrhovat.

Poznámka: Na konceptuální úrovni nemodelujeme cizí klíče, jež se používají pro vyjádření vazeb v relačních databázových systémech. Jednak bychom tím porušili žádoucí nezávislost na typu implementačního prostředí, protože v ne-relačních databázových systémech mohou být vztahy v databázi zaznamenávány jiným způsobem, za druhé bychom tím narušili účel konceptuální analýzy a návrhu, jež mají zachycovat podstatné skutečnosti platné v modelované výseči světa. Protože cizí klíče jsou realizací vztahů mezi entitní množinami, jejich modelování na konceptuální úrovni by také znamenalo redundantní vyjádření vztahů. Na konceptuální úrovni návrhu modelujeme tedy pouze vztahy mezi entitními množinami, a nikoliv cizí klíče. Nástroje, které používáme v tomto textu, v dalších úrovních návrhu databáze pro

cílovou relační databázovou platformu cizí klíče pro realizaci vztahů mezi entitními množinami automaticky vytvoří.

V tomto učebním textu budou pro konceptuální úroveň návrhu databáze vytvářeny:

1. **Konceptuální schéma**, tj. model vytvářený za účelem poznání analyzované výseče světa, pro který zamýšlíme databázi navrhnout. Tento model bude obsahovat základní entitní množiny, vztahy mezi entitními množinami a atributy entitních množin a příp. vztahů.
2. **Konceptuální model dat**, tj. model vytvářený za účelem detailní specifikace navrhované databáze. Obsahuje již všechny entitní množiny, vztahy a atributy a jejich datové typy[6], které potřebujeme v databázi zachytit. Navíc mohou být v konceptuálním modelu dat vztahy M:N a n-ární vztahy z konceptuálního schématu transformovány do entitních množin.

Řada návrhových nástrojů umožňuje na základě konceptuálního schématu nebo konceptuálního modelu dat automaticky generovat další úrovně modelů. Nezávislost návrhu na konceptuální úrovni na cílovém implementačním prostředí tak usnadňuje přenos databázového řešení mezi různými databázovými platformami.

Na konceptuální úrovni návrhu budeme rozlišovat:

- entitní množiny,
- vztahy mezi entitními množinami,
- atributy entitních množin a vztahů.

Dále uvedený popis je stručným doplňkem k přednášené problematice.

1.1.1 Entitní množiny

Entitní množina reprezentuje množinu entit. Entita[7] je rozlišitelný a identifikovatelný objekt světa, který popisujeme. Entitou je např. zaměstnanec Karel Novák, nebo oddělení mzdové účtárny. Při návrhu nicméně nepracujeme na úrovni entit, ale pracujeme na úrovni entitních množin, např. modelujeme entitní množiny *Zaměstnanec*, *Oddělení*. Entitní množiny se pojmenovávají v jednotném čísle.

[6] Vymezením obecných datových typů nezávisle na programovacím jazyku nebo platformě se zabývá norma ISO/IEC 11404.
[7] Také často označovaná jako objekt.

Každá entitní množina musí mít identifikátor, tj. minimální množinu vlastností, které zajistí jednoznačnou identifikaci entity v rámci dané entitní množiny. Identifikátor entitní množiny může být tvořen vlastními atributy, nebo může být entitní množina identifikačně závislá (slabá) a na její identifikaci se pak podílí jiná entitní množina, se kterou je slabá entita v určitém vztahu.

Entitní množiny mohou být popisovány v rámci ISA[8] hierarchie (vztahu nadtyp/podtyp), také označované jako generalizace/specializace. Například entitní množina *Zaměstnanec* může být specializována na entitní množiny *Akademický pracovník* a *Administrativní zaměstnanec*. Takovýto způsob popisu používáme, pokud u specializovaných podmnožin entit existují specifické atributy anebo podmnožiny entit vstupují samostatně do dalších vztahů s jinými entitními množinami. V řadě nástrojů je generalizace/specializace vyjadřována tzv. dědičností[9].

Generalizace/specializace může být (viz např. Teorey a kol., 2011):

- **Výlučná či nevýlučná.** Výlučná – každá entita z entitní množiny *Zaměstnanec* může být buď *Akademickým pracovníkem*, nebo *Administrativním zaměstnancem*. Nevýlučná – jedna entita z entitní množiny *Zaměstnanec* může být současně *Akademickým pracovníkem* i *Administrativním zaměstnancem*.
- **S úplnou či částečnou dekompozicí do podtypů.** V případě úplné dekompozice je každá entita z entitní množiny *Zaměstnanec* zařazena do nějakého podtypu. V případě částečné dekompozice může existovat entita z entitní množiny *Zaměstnanec*, která není zařazena ani do jednoho podtypu, tj. může existovat zaměstnanec, který není ani akademickým pracovníkem ani administrativním zaměstnancem.

Vedle vztahu typu generalizace/specializace mohou být entitní množiny ve vztahu, který má povahu celek-část. Teorey a kol. (2011) označují situaci, kdy jedna entitní

[8] Z anglického "is a".
[9] Používat pojem dědičnost pro označení vztahů typu nadtyp/podtyp ale může být zavádějící, jelikož pojem dědičnost odkazuje k mechanismu známému z objektově orientovaných programovacích jazyků, přičemž ve všech takovýchto jazycích nemusí mít aplikace dědičnosti stejný význam jako má vztah typu nadtyp/podtyp, resp. generalizace/specializace v konceptuálním modelování. Viz např. (Cook a kol. 1990).

množina představuje část jiné entitní množiny pojmem agregace, přičemž agregaci chápou jako určitou formu vztahu mezi nadtypem a podtypem.

V jazyce Unified Modeling Language (UML) jsou pak rozlišovány vztahy agregace a kompozice. V případě agregace může část existovat nezávisle na celku. V případě kompozice část nezávisle na celku existovat nemůže. (Kanisová a Müller, 2006)

Entitní množinu, která by měla obsahovat pouze jedinou entitu (například organizaci, pro níž se databáze navrhuje), nemá obvykle v konceptuálním schématu smysl modelovat.

1.1.2 Vztahy mezi entitními množinami

Jednotlivé entitní množiny mohou být propojeny vztahy. Rozlišujeme binární vztahy, tj. vztahy mezi maximálně dvěma entitními množinami, a n-ární (polyární) vztahy, které propojují více než dvě entitní množiny. U binárních vztahů rozlišujeme kardinalitu a povinnost či volitelnost vazeb. Blíže viz kapitola 2.1.

1.1.3 Atributy

Entitní množiny musí, a vztahy mohou, obsahovat atribut. Atribut je vlastnost či charakteristika entit v množině nebo vztahu mezi entitami. Například entity v množině *Zaměstnanec* mohou být charakterizovány atributy *Osobní číslo*, *Jméno zaměstnance*, *Datum narození*, *Pohlaví*, *Titul před jménem*, *Titul za jménem*. Rozlišujeme povinné a volitelné atributy: povinné atributy musí být uvedeny u každé entity v dané entitní množině; volitelné atributy nikoli, tj. v dané entitní množině mohou existovat entity, u nichž onen atribut uveden být nemusí, třeba proto, že neexistuje, nebo není znám.

1.2 Fyzická úroveň návrhu

V tomto učebním textu bude v návaznosti na konceptuální schéma a konceptuální datový model uváděn **fyzický datový model**, který již zohledňuje konkrétní možnosti konkrétního databázového systému (v našem případě databázové platformy Oracle). Pro daný fyzický datový model pak bude vždy uveden **skript v jazyce SQL**, který navrhovanou databázi umožní v prostředí databázového systému Oracle vytvořit.

Na fyzické úrovni již budou zohledňovány technologicky a implementačně závislé typy objektů, tj. zejména objekty typu:

- **Table**, relační tabulka, která je základní strukturou pro ukládání dat v relačním prostředí. Tabulky vyjadřují entitní množiny a případně binární vztahy M:N anebo n-ární vztahy mezi entitními množinami, kde n > 2. Tabulka obsahuje sloupce, které vznikly z atributů entitní množiny nebo vztahu. Sloupce jsou vždy určitého datového typu a mohou mít nastavena integritní omezení, která se týkají jejich hodnot:
    - **povinnost uvádět hodnotu** v daném sloupci (`NOT NULL`),
    - **unikátnost hodnot** v jednom nebo více sloupcích (`UNIQUE`),
    - **primární klíč**, který vzniká z identifikátoru entitní množiny z konceptuální úrovně návrhu a případně z cizích klíčů pro relační tabulky, které vznikly z transformace M:N vztahů nebo n-árních vztahů z konceptuální úrovně návrhu; primární klíč může být tvořen jedním i více sloupci;
    - pro vyjádření vztahů z konceptuální úrovně návrhu se na fyzické úrovni používá **cizí klíč**, který je tvořen jedním či více sloupci, které odkazují na primární klíč či jiný unikátní identifikátor do stejné nebo jiné tabulky; graficky se ve fyzickém datovém modelu zobrazuje jako **reference** (viz kapitoly 2.3.1 a 2.3.2);
    - další omezení týkající se obsahu hodnot sloupce, která se vyjadřují v jazyce SQL s použitým klauzule `CHECK`.
- **View**, pohled, je dynamická odvozená relační tabulka, která umožní průhled do relačních tabulek přímo anebo prostřednictvím jiných view. View je definováno dotazem.
- **Materialized view**, což je typ databázového objektu, který obsahuje uložené výsledky dotazu. Pro srovnání, (nematerializovaný) pohled představuje uložený dotaz, tj. určitý předpis reprezentace dat uložených v jedné nebo více tabulkách, ale na rozdíl od materializovaného pohledu neobsahuje uložená data. (Oracle, © 2019a)

- **Sequence** je typ databázového objektu, který se používá pro nastavení pravidel pro automatické pořadové číslování. Nejčastěji se používá pro generování hodnot sloupce, který tvoří umělý[10] primární klíč.
- **Index**, je databázový objekt, který umožňuje rychlejší přístup k datům uloženým v databázi. Definuje se nad jedním či více sloupci.
- **Integritní omezení** – jsou pojmenovaná pravidla a podmínky, které zajišťují udržení konzistence dat v databázi, tj. toho, že data nabývají pouze takových hodnot, které jsou přípustné pro svět v databázi zachycovaný. Rozlišují se:
    - **Entitní integrita**, která je vyjadřována tím, že každá relační tabulka musí mít primární klíč.
    - **Doménová integrita** – zajištění, aby hodnota sloupce byla v souladu s množinou přípustných hodnot. V příkladech je řešíme zejména s využitím klauzule `CHECK`.
    - **Referenční integrita**, která je řešena referencemi a umožňuje zajistit konzistenci vztahů vyjádřených v relačních tabulkách cizími klíči. V kapitole 2 dále doplníme uvedené vysvětlení používaných pojmů na konceptuální i fyzické úrovni návrhu.

[10] Umělý primární klíč je primární klíč, který není tvořen z atributů entitní množiny, která byla zdrojem pro návrh tabulky, nemá vlastní význam. Je doplněn pouze pro odlišení jednoho řádku v relační tabulce od ostatních řádků. Nejčastěji se pro umělé primární klíče používají generované numerické identifikátory.

2 Notace modelů použitých v této publikaci

2.1 Notace konceptuálního schématu

Pro vytváření modelů na konceptuální úrovni návrhu databáze jsou využívány různé typy modelů, resp. notace. Z hlediska vývoje konceptuálního modelování uvádějí Teorey a kol. (2011) následující tři významné typy modelů používaných při návrhu databáze na konceptuální úrovni, které vznikly v průběhu času:

- digramy struktury dat (Data Structure Diagrams) Charlese Bachmana (představeny v roce 1969),
- entitně-relační modely (ER modely) Petera Chena (představeny v roce 1976),
- diagramy jazyka UML (Unified Modeling Language, představeny v roce 1997).

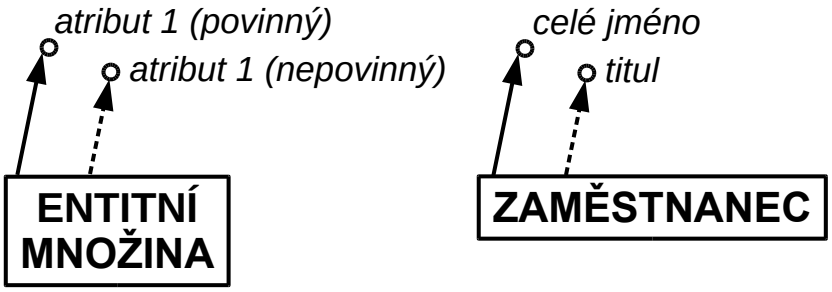
V diagramu struktury dat (Bachman, 1969) jsou entitní množiny znázorněny pomocí obdélníku a šipkami jsou pak znázorněny vztahy mezi entitními množinami. ER model (Chen, 1976) využívá obdélník pro reprezentaci entitní množiny a kosočtverec společně se spojnicemi pro reprezentaci vztahů mezi entitními množinami. ER model umožňuje také vyjádřit atributy entitních množin.

Jazyk UML obsahuje celou řadu modelů, které se liší svým účelem – viz (Object Management Group, © 1997-2019). Teorey a kol. (2011) uvádějí, že při návrhu databází jsou využívány zejména diagram tříd (class diagram) a diagram aktivit (aktivity diagram). V úvodu kapitoly 1 bylo uvedeno, že konceptuální schéma lze chápat také jako ontologii. V této souvislosti je vhodné uvést, že pro konceptuální modelování založené na ontologii existuje jazyk OntoUML, který je rozšířením jazyka UML (OntoUML Community © 2017).

V dalších částech této kapitoly je pak popsána notace konceptuálního schématu používaná v těchto skriptech, která je inspirována ER modelem Petera Chena.

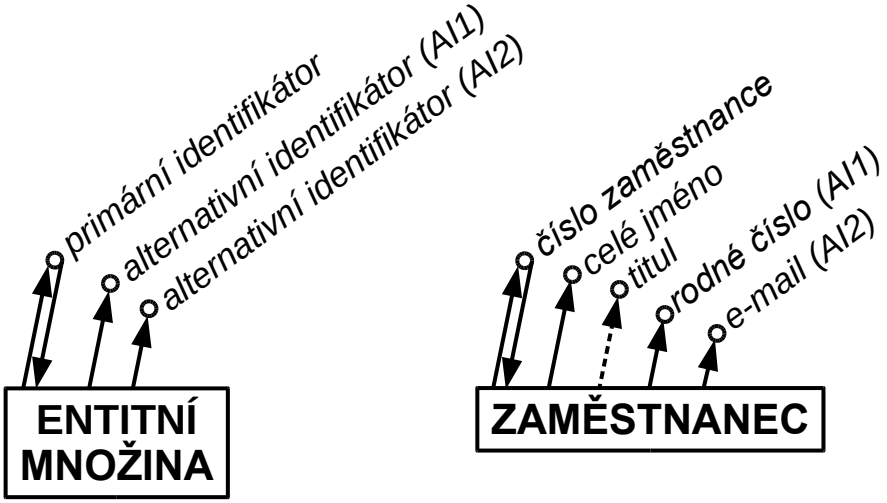
2.1.1 Entitní množiny, jejich atributy a identifikátory

Entitní množiny jsou značeny obdélníkem, uvnitř kterého je uveden název entitní množiny (obrázek 1). Název entitní množiny je tvořen nejčastěji podstatnými jmény v jednotném čísle. Atributy entitní množiny jsou pak zapisovány mimo obdélník. Atribut je značen kolečkem. Název atributu se zapisuje mimo kolečko, ale těsně u něj. Povinné atributy jsou značeny šipkou s plnou čarou vedoucí od symbolu entitní množiny, k symbolu atributu. U nepovinných atributů je čára šipky přerušovaná.



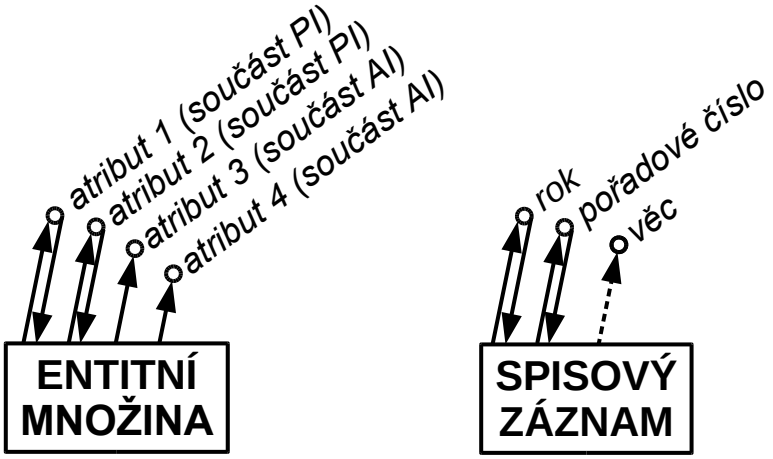
Obrázek 1: Značení entitní množiny (konceptuální schéma), zdroj: autoři

Šipka vedoucí od symbolu atributu k entitní množině vyjadřuje, že atribut představuje primární identifikátor entitní množiny. Pokud je primární identifikátor tvořen více atributy, je šipka od atributu k entitní množině uvedena u každého z atributů, které tento identifikátor tvoří. Alternativní identifikátory jsou značeny symbolem `(AI)` uvedeným za názvem atributu. Pokud existuje vícero alternativních identifikátorů, obsahuje symbol `(AI)` číslovku pro odlišení jednotlivých alternativních identifikátorů. Obrázek 2 ukazuje značení jednoduchých identifikátorů.



Obrázek 2: Jednoduchý primární a alternativní identifikátor (konceptuální schéma), zdroj: autoři

Na obrázku 3 je znázorněno značení složeného primárního a alternativního identifikátoru.



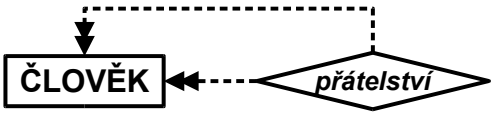
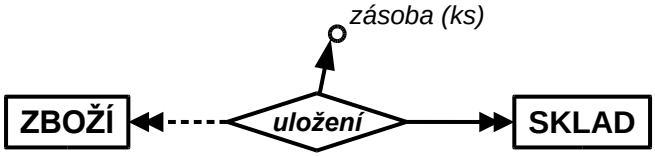
Obrázek 3: Složený primární a alternativní identifikátor (konceptuální schéma), zdroj: autoři

2.1.2 Vztahy mezi entitními množinami

Binární vztahy

Binární vztahy jsou značeny kosočtvercem, uvnitř kterého je uveden název vztahu. Názvy vztahů jsou nejčastěji tvořeny slovesy nebo podstatnými jmény slovesnými. Od kosočtverce vybíhají šipky k entitním množinám, které jsou vztahem spojeny. Kardinalita a parcialita vztahů je interpretována podle symbolů na konci vztahu ve směru od výchozí entitní množiny, pro kterou kardinalitu a parcialitu interpretujeme, k cílové entitní množině. Povinnost daného směru vztahu je značena plnou čarou šipky, nepovinnost pak přerušovanou čarou. Pokud je vztah v určitém směru jednonásobný, je v daném směru uvedena na konci jednoduchá šipka. Pokud je vztah v určitém směru vícenásobný, je v daném směru uvedena na konci dvojitá šipka. V tabulce 1 jsou uvedeny příklady značení vztahů s různou kardinalitou a parcialitou.

Tabulka 1: Příklady binárních vztahů (konceptuální schéma), zdroj: autoři

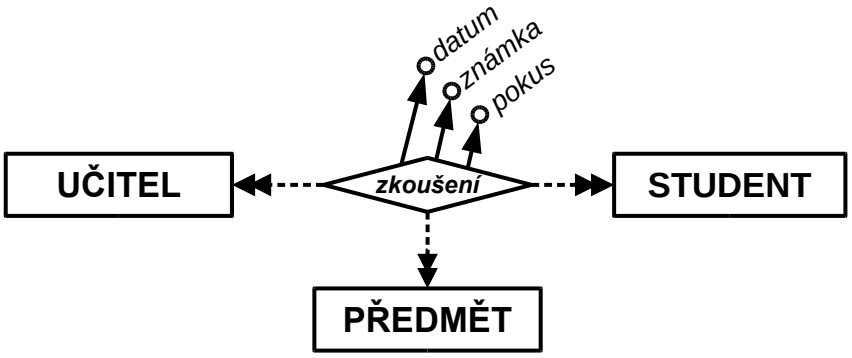
Typ vztahu	Příklady značení
Vztah 1:1	 Manžel musí mít za manželku právě jednu ženu, manželka musí mít za manžela právě jednoho muže.
Vztah 1:N	 Dítě má právě jednu matku; žena, která je matkou, porodila jedno nebo více dětí.
Vztah 1:N	 Zaměstnanec je zařazen právě do jednoho oddělení, v oddělení může být zařazeno více zaměstnanců, ale může být také neobsazeno.
Vztah M:N	 Člověk může mít více přátel a může být přítelem více jiných lidí.
Vztah M:N	 Nahrávku nahrál alespoň jeden interpret, interpret mohl nahrát vícero nahrávek, ale nemusel nahrát žádnou nahrávku.
Vztah M:N s atributem	 Zboží musí být uloženo alespoň na jednom skladu, na skladu může být uloženo vícero zboží. Mohou existovat i sklady, ve kterých není uloženo žádné zboží. Je-li zboží uloženo na nějakém skladu, je uvedena zásoba (počet kusů) toho zboží na tom skladu.

N-ární vztahy

V konceptuálním schématu jsou n-ární vztahy značeny stejně jako vztahy binární a mohou mít také atributy. V případě n-árních vztahů ze symbolu kosočtverce vychází tolik šipek, mezi kolika entitními množinami vztah existuje.

Na obrázku 4 je znázorněn vztah mezi třemi entitními množinami (ternární vztah), který vyjadřuje skutečnost, že zkoušení je vztahem mezi učitelem, studentem a předmětem. Současně lze z uvedeného ternárního vztahu odvodit následující skutečnosti:

- jeden učitel mohl zkoušet více žáků z více předmětů;
- žák mohl být zkoušen více učiteli v rámci různých předmětů;
- v rámci předmětu mohlo být vícero žáků zkoušeno vícero učiteli.



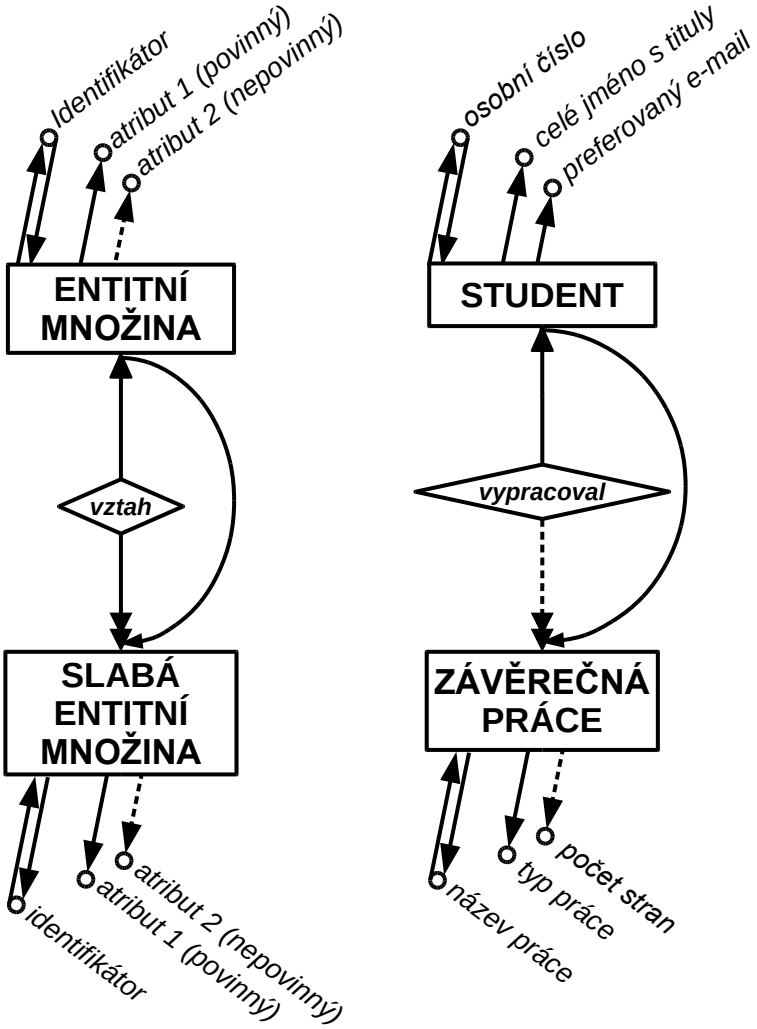
Obrázek 4: Ternární vztah (konceptuální schéma), zdroj: autoři

Ale současně je třeba upozornit, že uvedený příklad na obrázku 4 neumožní úplně přesně vyjádřit, skutečnost, že učitel může vyzkoušet z jednoho předmětu studenta opakovaně. K tomu by již bylo vhodnější modelovat vztah zkoušení spíše jako entitní množinu *Zkouška*, která by pak měla tři binární vztahy k původním třem entitním množinám. Entitní množina by pak měla mít složený identifikátor tvořený externími identifikátory od všech tří původních entitních množin a ještě navíc by součástí složeného identifikátoru bylo datum zkoušky.

2.1.3 Identifikační závislost

V případě identifikační závislosti slabých entitních množin[11] je externí identifikace značena šipkou směřující ke slabé entitní množině, tj. k té entitní množině, která je identifikačně závislá na jiné entitní množině. Nositelem identifikační závislosti je vztah mezi entitními množinami, proto je šipka umístěna u příslušného vztahu. Primární identifikátor slabých entit ale nemusí být vždy dán pouze vztahem k jiné entitní množině, ale může zahrnovat i vlastní atributy slabé entitní množiny. Značení identifikační závislosti je ilustrováno na obrázku 5.

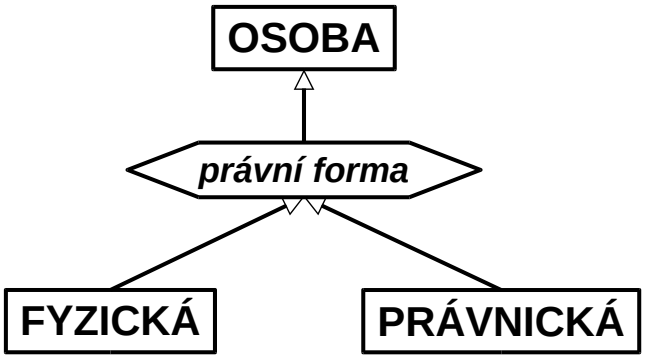
[11] Identifikačně slabá entitní množina je taková entitní množina, jejíž atributy nestačí pro vytvoření identifikátoru. Takové entitní množiny potřebují pro svou identifikaci atribut či atributy z jiných entitních množin.



Obrázek 5: Identifikační závislosti (konceptuální schéma), zdroj: autoři

2.1.4 Vztahy typu generalizace/specializace

V konceptuálním datovém modelu je vztah typu generalizace/specializace značen šestiúhelníkem, uvnitř kterého je uveden název vztahu. Dále jsou vedeny šipky od entitních množin představujících podtypy k symbolu generalizace/specializace. Od tohoto symbolu je pak vedena šipka k entitní množině představující nadtyp. Na rozdíl od šipek binárních nebo n-árních vztahů nejsou šipky u vztahu generalizace/specializace plné, nýbrž prázdné. Značení vztahu generalizace/specializace je znázorněno na obrázku 6.



Obrázek 6: Vztah typu generalizace/specializace (konceptuální schéma), zdroj: autoři

Notace konceptuálního schématu neobsahuje speciální symboly pro rozlišení výlučné/nevýlučné generalizace/specializace, resp. tohoto typu vztahu s úplnou, nebo částečnou dekompozicí do podtypů (viz kapitola 1.1.1). V konceptuálním schématu tak jsou všechny varianty tohoto typu vztahů vyjádřeny stejně. V rámci popisu notace konceptuálního datového modelu je dále ukázáno, že v nástroji Power Designer je možné tyto varianty vztahů typu generalizace/specializace rozlišit s využitím symbolů notace E/R + Merise; v nástroji Data Modeler je pak možné tyto varianty rozlišit v rámci konfigurace příslušného vztahu, kterým se realizuje rozdělení nadtypu do podtypů. Povaha vztahu má význam pro návrh databáze a je tedy důležité ji určit. Nicméně na úrovni konceptuálního schématu postačí rozpoznat, že vztah mezi entitními množinami má povahu vztahu generalizace/specializace. Bližší určení povahy tohoto vztahu je předmětem modelování a návrhu databáze na úrovni konceptuálního datového modelu.

2.2 Notace konceptuálního datového modelu

2.2.1 Notace modelu CDM v nástroji Power Designer

Nástroj Power Designer umožňuje volit vícero notací konceptuálního datového modelu (v nástroji označovaném jako Conceptual Data Model, zkráceně CDM). V těchto skriptech je použita zde popsaná notace v nástroji označená jako E/R + Merise.

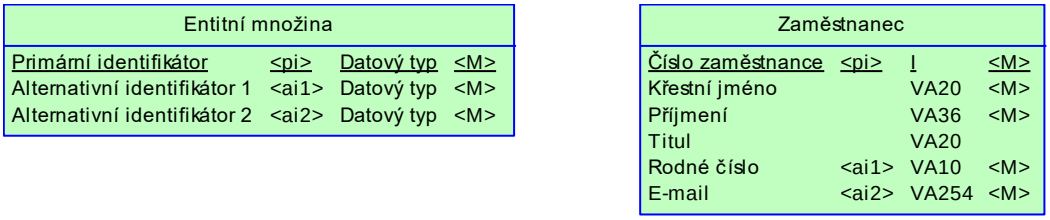
Entitní množiny, jejich atributy a identifikátory

Entitní množiny jsou značeny obdélníkem (obrázek 7), který v horní části obsahuje název entitní množiny, pod kterým následuje výčet atributů entitní množiny. U atributu je vedle jeho názvu uvedena i zkratka datového typu. Symbol ‹M› v řádku

atributu entitní množiny znamená povinný atribut. Atributy, u kterých tento symbol chybí, jsou nepovinné, tj. jejich hodnota může být neurčená (prázdná, NULL).

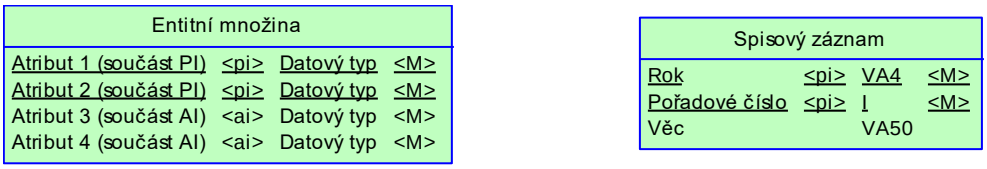
Obrázek 7: Značení entitní množiny (notace E/R + Merise), zdroj: autoři

Symboly <pi> a <ai> označují primární, resp. alternativní identifikátory. Primární identifikátor je navíc podtržen. Pokud je primární nebo alternativní identifikátor tvořen více atributy, je příslušný symbol uveden u každého z atributů, které identifikátor tvoří. Pokud existuje vícero alternativních identifikátorů, obsahuje symbol <ai> číslovku pro odlišení jednotlivých alternativních identifikátorů. Obrázek 8 ukazuje značení jednoduchých identifikátorů.



Obrázek 8: Jednoduchý primární a alternativní identifikátor (notace E/R + Merise), zdroj: autoři

Na obrázku 9 je znázorněno značení složeného primárního a alternativního identifikátoru.



Obrázek 9: Složený primární a alternativní identifikátor (notace E/R + Merise), zdroj: autoři

Přehled datových typů, které lze použít v konceptuálním datovém modelu nástroje Power Designer lze nalézt v dokumentaci (SAP, n.d.b). Tabulka 2 obsahuje přehled

datových typů a jejich zkratek, které jsou použity v konceptuálních modelech uvedených v těchto skriptech.

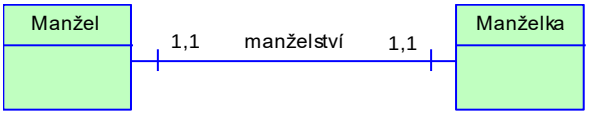
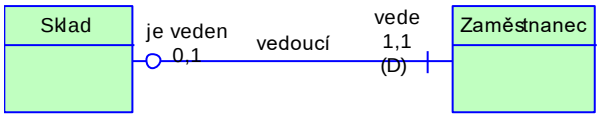
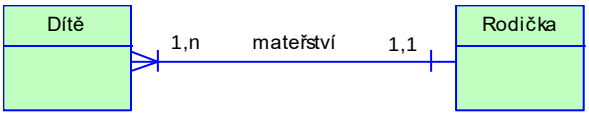
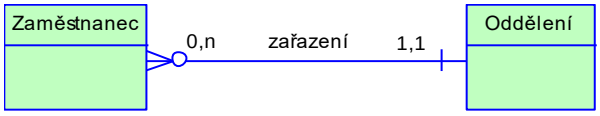
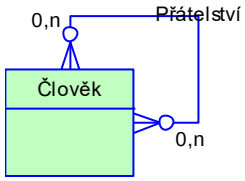
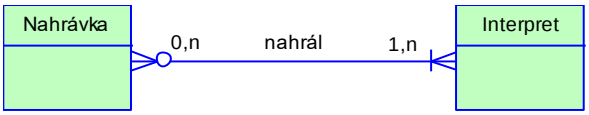
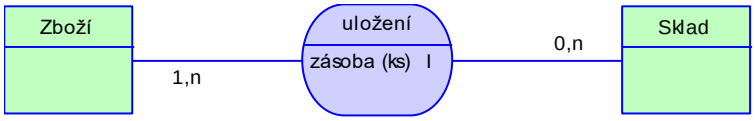
Tabulka 2: Datové typy použité v konceptuálních datových modelech, nástroj Power Designer, zdroj: zpracováno s využitím (SAP, n.d.b)

Název datového typu	Zkratka	Význam
Boolean	BL	Logická hodnota
Date	D	Kalendářní datum
Float	F	Desetinné číslo o délce 32 bitů
Integer	I	Celé číslo o délce 32 bitů
Money	MN (počet cifer, přesnost)	Číslo s pevně určeným počtem desetinných míst
Number	N (počet cifer, přesnost)	Číslo s pevně určeným počtem desetinných míst
Text	TXT	Textový řetězec; určeno pro dlouhé texty, které svojí délkou přesahují omezení datového typu Variable Characters
Variable Characters	VA (velikost)	Textový řetězec s určenou maximální délkou

Binární vztahy

Binární vztahy jsou v nástroji Power Designer značeny čárou spojující příslušné entitní množiny, u které může být umístěn titulek vyjadřující název vztahu. Povinnost vztahu je značena čárou kolmou k čáře vztahu; nepovinnost vztahu je značena prázdným kolečkem. Kardinalita a parcialita vztahů je interpretována podle symbolů na konci vztahu ve směru od výchozí entitní množiny, pro kterou kardinalitu a parcialitu interpretujeme, k cílové entitní množině. Vztah, který je v určitém směru vícenásobný, je v daném směru zakončen vraní nohou, případně třemi souběžným čarami v případě, že vztah je nositelem externí identifikace. V tabulce 3 jsou uvedeny příklady značení vztahů s různou kardinalitou a parcialitou.

Tabulka 3: Příklady binárních vztahů (notace E/R + Merise), zdroj: autoři

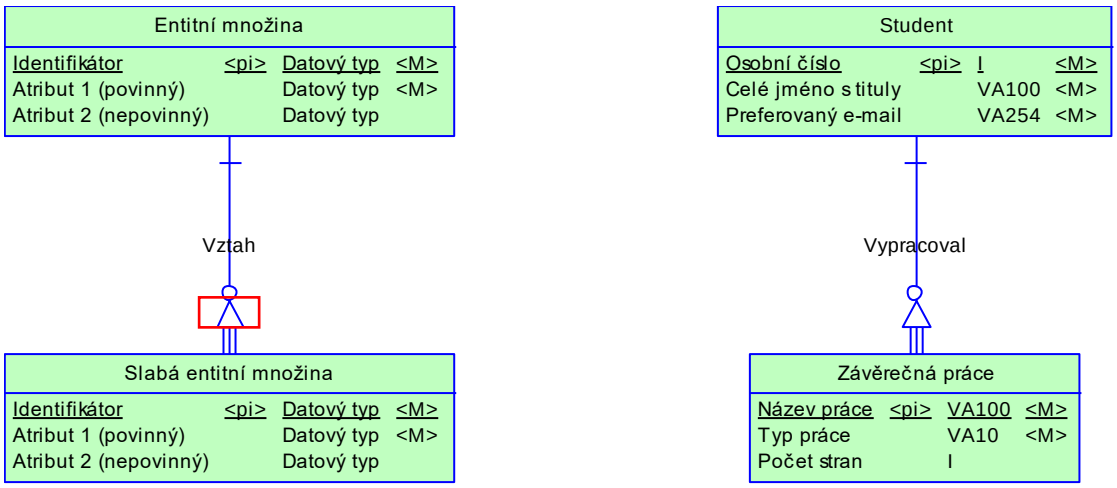
Typ vztahu	Příklady značení
Vztah 1:1	 Manžel musí mít právě jednu manželku, manželka musí mít právě jednoho manžela.
Vztah 1:1 s dominantní rolí	 Sklad je veden právě jedním zaměstnancem, zaměstnanec může vést nejvýše jeden sklad.
Vztah 1:N	 Dítě má právě jednu matku; žena, která je matkou, porodila jedno nebo více dětí.
Vztah 1:N	 Zaměstnanec je zařazen právě do jednoho oddělení, v oddělení může být zařazeno více zaměstnanců, ale může být také neobsazeno.
Vztah M:N	 Člověk může mít více přátel a může být přítelem více jiných lidí.
Vztah M:N	 Nahrávku nahrál alespoň jeden interpret, interpret mohl nahrát vícero nahrávek, ale nemusel nahrát žádnou nahrávku.
Vztah M:N s atributem	 Zboží musí být uloženo alespoň na jednom skladu, na skladu může být uloženo vícero zboží. Mohou existovat i sklady, ve kterých není uloženo žádné zboží. Je-li zboží uloženo na nějakém skladu, je uvedena zásoba (počet kusů) toho zboží na tom skladu.

Vztah, který má jeden nebo více atributů, nelze v nástroji Power Designer realizovat pomocí standardního elementu vztahu, ale při použití notace E/R + Merise je třeba

využít element asociace. Povšimněte si, že kardinalita a parcialita vztahu M:N s atributem je na rozdíl od vztahu bez atributu značena na straně u výchozí entitní množiny a nikoli symbolem na konci vztahu ve směru od výchozí entitní množiny k cílové entitní množině. Tj. oproti vztahům bez tributu je značení kardinality a parciality vztahů s atributy převrácené.

Identifikační závislost

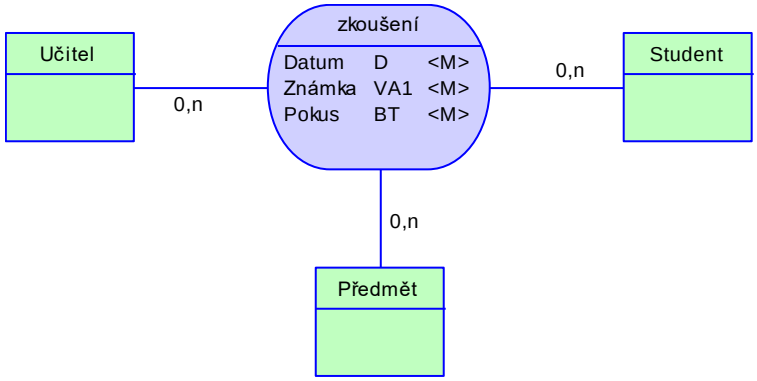
V případě identifikační závislosti slabých entitních množin se externí identifikace v nástroji Power Designer nastavuje v rámci příslušného vztahu. V konceptuálním datovém modelu je pak externí identifikace značena symbolem trojúhelníku, který je v příslušném symbolu vztahu umístěn na straně slabé entitní množiny (obrázek 10).



Obrázek 10: Identifikační závislosti (notace E/R + Merise), zdroj: autoři

N-ární vztahy

V nástroji Power Designer při použití notace E/R + Merise se pro vyjádření n-árních vztahů používá element asociace (stejný element, jako v případě binárních vztahů s atributy). N-ární vztahy mohou, ale nemusí mít atributy. Příklad značení ternárního vztahu je uveden na obrázku 11.

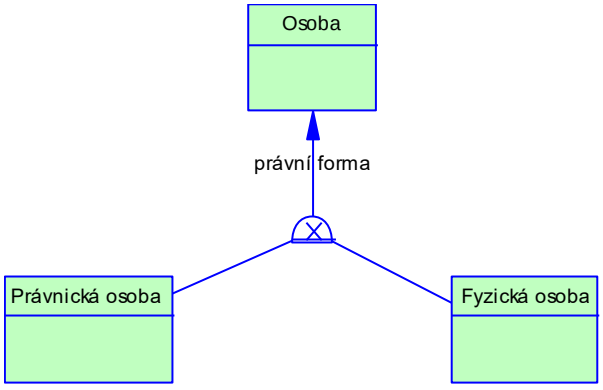


Obrázek 11: Ternární vztah (notace E/R + Merise), zdroj: autoři

Vztahy typu generalizace/specializace

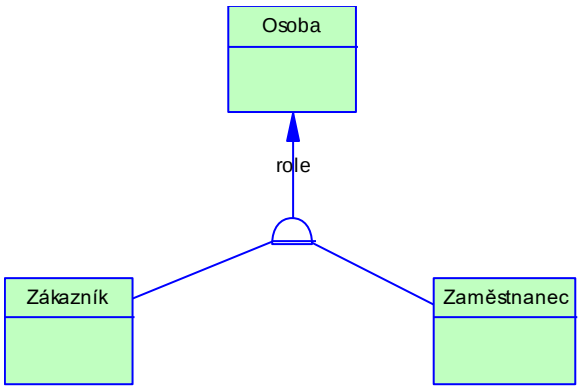
Vztahy typu generalizace/specializace jsou v nástroji Power Designer vyjádřeny pomocí elementu, který se v nástroji nazývá dědičnost (inheritance). Tomuto elementu je přiřazen symbol půloblouku, přičemž prázdný půloblouk znamená nevýlučný vztah generalizace/specializace a půloblouk s křížkem uvnitř značí výlučný vztah. Půloblouk s dvojitou spodní čárou značí úplný vztah generalizace/specializace, půloblouk s jednoduchou spodní čárou pak značí neúplný vztah. K výlučnosti/nevýlučnosti a úplnosti/neúplnosti vztahů typu generalizace/specializace viz kapitola 1.1.1.

Na obrázku 12 je znázorněn příklad výlučného a zároveň úplného vztahu typu generalizace/specializace. Konkrétně se jedná o situaci, kdy podle § 18 zák. č. 89/2012 Sb., občanský zákoník, jsou osoby buď fyzické, nebo právnické. Osoba nemůže mít obě formy zároveň a současně nemůže existovat osoba, která by neměla ani jednu z těchto právních forem.



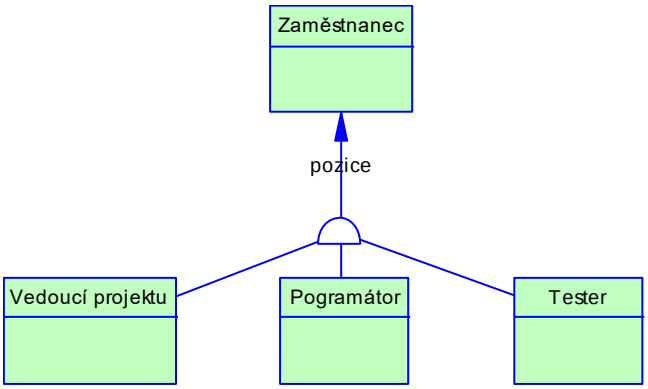
Obrázek 12: Výlučný a úplný vztah generalizace/specializace (notace E/R + Merise), zdroj: autoři

Obrázek 13 zachycuje příklad nevýlučného úplného vztahu. Konkrétně rozeznáváme osoby, které jsou zákazníkem, zaměstnancem, nebo obojím současně, a nemůže existovat osoba, která by nevystupovala alespoň v jedné z těchto rolí.



Obrázek 13: Nevýlučný a úplný vztah generalizace/specializace (notace E/R + Merise), zdroj: autoři

Na obrázku 14 je zachycen příklad nevýlučného a neúplného vztahu generalizace/specializace. Zaměstnanec může být vedoucím projektu, programátorem nebo testerem, přičemž může pracovat i na více pozicích současně. Zároveň mohou existovat zaměstnanci, kteří nejsou ani vedoucím projektu, ani programátorem, ani testerem.



Obrázek 14: Nevýlučný a neúplný vztah generalizace/specializace (notace E/R + Merise), zdroj: autoři

2.2.2 Notace modelu Logical v nástroji Data Modeler

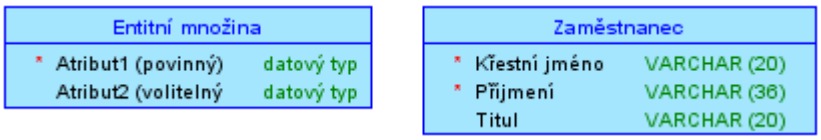
V nástroji Data Modeler odpovídá konceptuálnímu datovému modelu model s názvem Logical. Tento název na první pohled evokuje logický datový model. Logický datový model by ale měl být modelem v termínech určité třídy technologického prostředí,

přičemž s ohledem na zaměření těchto skript na návrh relačních databází by logický model měl být relačním databázovým modelem. Protože je ale model Logical nástroje Data Modeler tvořen mimo jiné entitami a umožňuje modelovat vztahy M:N, odpovídá konceptuálnímu datovému modelu, jak byl vymezen v kapitole 1.

Data Modeler umožňuje vytvářet konceptuální datový model ve třech notacích: Barkerova notace, Bachmanova notace a notace Information Engineering. V těchto skriptech bude dále v rámci konceptuálních datových modelů vytvořených v nástroji Data Modeler používána notace Information Engineering s ohledem na to, že má určité shodné rysy s notací E/R + Merise používanou v nástroji Power Designer.

Entitní množiny, jejich atributy a identifikátory

Entitní množiny jsou značeny obdélníkem (obrázek 15), který v horní části obsahuje název entitní množiny, pod kterým následuje výčet atributů entitní množiny. U atributu je vedle jeho názvu uveden i datový typ. Symbol hvězdičky (*) v řádku atributu entitní množiny znamená povinný atribut. Atributy, u kterých tento symbol chybí, jsou nepovinné, tj. jejich hodnota může být neurčená (prázdná, `NULL`).

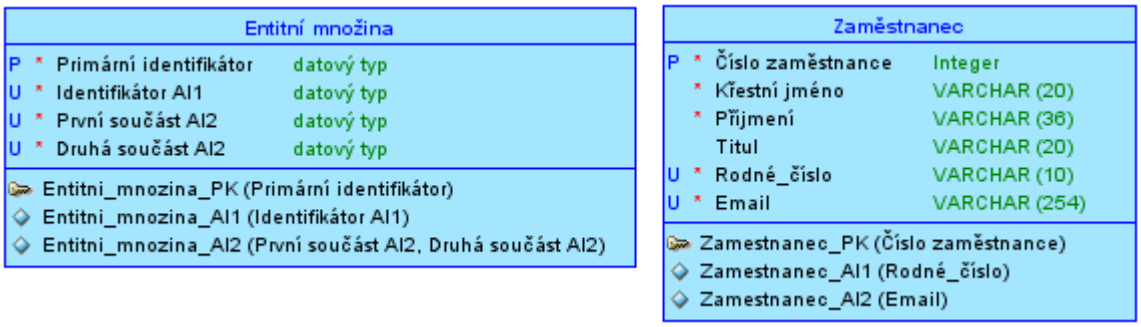


Obrázek 15: Značení entitní množiny (notace Information Engineering), zdroj: autoři

V nástroji Data Modeler jsou datové typy vybírány ze seznamu logických datových typů. Řešení úloh v nástrojích Power Designer a Data Modeler jsou shodná, tj. v obou nástrojích jsou voleny stejné datové typy. Přehled datových typů použitých v rámci řešení úloh v těchto skriptech uvádí tabulka 2 na straně 23.

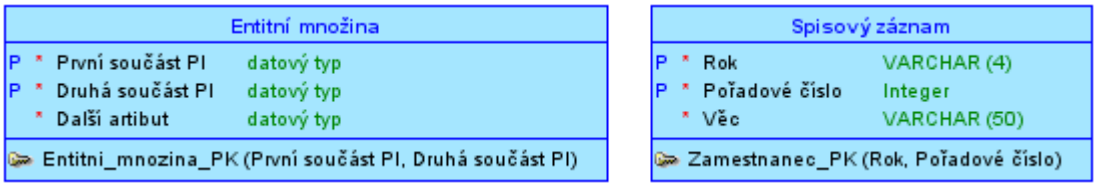
Symboly `P` a `U` označují primární, resp. alternativní identifikátory (atributy, jejichž hodnota je napříč výskyty entitní množiny unikátní). Pokud je primární nebo alternativní identifikátor tvořen více atributy, je příslušný symbol uveden u každého z atributů, které identifikátor tvoří. Obrázek 16 ukazuje značení jednoduchého primárního identifikátoru a dvou alternativních identifikátorů, z nichž jeden je složený.

Jak je vidět, symboly U neodlišují, zda se jedná o více jednoduchých či nějaké složené alternativní identifikátory. Potřebnou informaci zjistíme ze spodní části obdélníku.[12]



Obrázek 16: Jednoduchý primární a dva alternativní identifikátory, jeden z nich je složený (notace Information Engineering), zdroj: autoři

Na obrázku 17 je znázorněno značení složeného primárního identifikátoru. Tady nedorozumění nehrozí, primární identifikátor je jenom jeden, ať již složený, či nikoli.



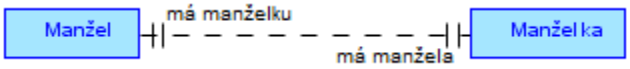
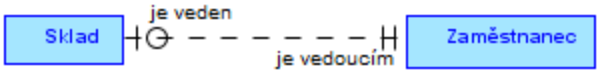
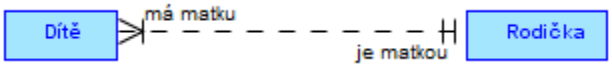
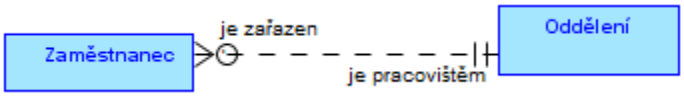
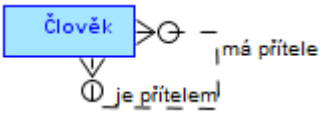
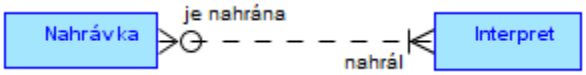
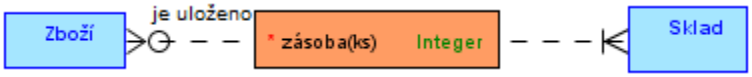
Obrázek 17: Složený primární identifikátor (notace Information Engineering), zdroj: autoři

Binární vztahy

Binární vztahy jsou v nástroji Data Modeler značeny čárou spojující příslušné entitní množiny. Povinnost vztahu je značena čárou kolmou k čáře vztahu; nepovinnost vztahu je značena prázdným kolečkem. Kardinalita a parcialita vztahů je interpretována podle symbolů na konci vztahu ve směru od výchozí entitní množiny, pro kterou kardinalitu a parcialitu interpretujeme, k cílové entitní množině. Vztah, který je v určitém směru jednonásobný, je v daném směru zakončen čárou kolmou k čáře vztahu (stejně jako povinnost vztahu, kardinalitu vyjadřuje čára blíže k entitní množině). Vztah, který je v určitém směru vícenásobný, je v daném směru zakončen vraní nohou. V tabulce 4 jsou uvedeny příklady značení vztahů s různou kardinalitou a parcialitou.

[12] Pokud zobrazení spodního prostoru entitní množiny nepřináší pro pochopení modelu dodatečnou informaci, je tento prostor v řešeních úloh pro větší přehlednost skryt.

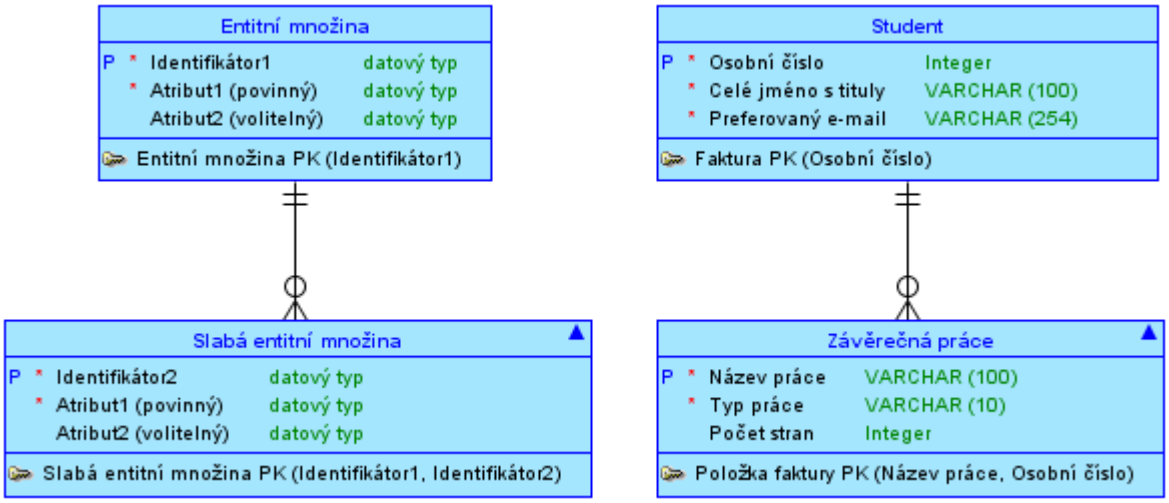
Tabulka 4: Příklady binárních vztahů (notace Information Engineering), zdroj: autoři

Typ vztahu	Příklady značení
Vztah 1:1	 Manžel musí mít právě jednu manželku, manželka musí mít právě jednoho manžela.
Vztah 1:1 s dominantní rolí	 Sklad je veden právě jedním zaměstnancem, zaměstnanec může vést nejvýše jeden sklad.
Vztah 1:N	 Dítě má právě jednu matku; žena, která je matkou, porodila jedno nebo více dětí.
Vztah 1:N	 Zaměstnanec je zařazen právě do jednoho oddělení, oddělení může být pracovištěm více zaměstnanců, ale může být také neobsazeno.
Vztah M:N	 Člověk může mít více přátel a může být přítelem více jiných lidí.
Vztah M:N	 Nahrávku nahrál alespoň jeden interpret, interpret mohl nahrát vícero nahrávek, ale nemusel nahrát žádnou nahrávku.
Vztah M:N s atributem	 Zboží musí být uloženo alespoň na jednom skladu, na skladu může být uloženo vícero zboží. Mohou existovat i sklady, ve kterých není uloženo žádné zboží. Je-li zboží uloženo na nějakém skladu, je uvedena zásoba (počet kusů) toho zboží na tom skladu.

Na rozdíl od nástroje Power Designer lze v nástroji Data Modeler přidat atributy vztahu přímo k elementu vztahu. Tj. v nástroji Data Modeler není pro reprezentaci vztahu s atributem zapotřebí zvláštního elementu a nemění se tak ani způsob interpretace kardinality a parciality vztahů s atributy.

Identifikační závislost

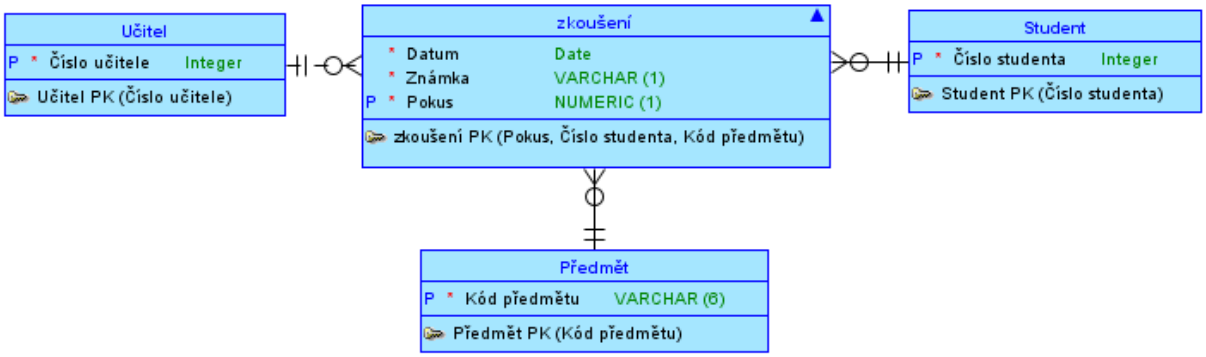
V případě identifikační závislosti slabých entitních množin se externí identifikace v nástroji Data Modeler nastavuje v rámci příslušného vztahu. V konceptuálním datovém modelu je pak externí identifikace značena plnou čárou příslušného vztahu (Obrázek 18.) Ve výchozím nastavení při použití notace Information Engineeting nástroj Data Modeler v modelu Logical pro vztah vždy zobrazuje u entit i příslušný cizí klíč. Aby ale použití modelu Logical odpovídalo vymezení konceptuálního datového modelu dle kapitoly 1.1, jsou dále v těchto skriptech cizí klíče v modelu Logical skryty.



Obrázek 18: Externí identifikace (notace Information Engineering), zdroj: autoři

N-ární vztahy

Data Modeler nepodporuje modelování n-árních vztahů. Ty je tak třeba rozložit na entitní množinu a více vztahů 1:N. Na obrázku 19 je znázorněn rozklad ternárního vztahu, který je znázorněn konceptuálním schématem na obrázku 11.

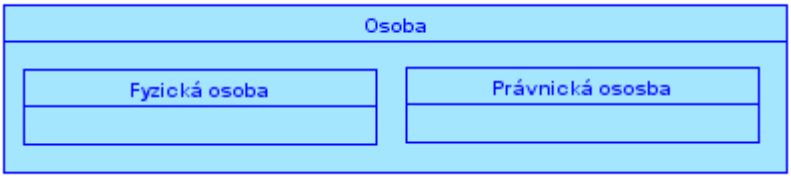


Obrázek 19: Rozklad ternárního vztahu (notace Information Engineering), zdroj: autoři

Vztahy typu generalizace/specializace

Vztahy typu generalizace/specializace jsou v nástroji Data Modeler vyjádřeny pomocí vztahu označovaného jako podtyp-nadtyp (subtype-supertype)[13]. Entitní množiny, které představují podtypy, jsou znázorněny uvnitř entitní množiny, která představuje nadtyp. Úplnost a výlučnost vztahu typu generalizace/specializace lze nastavit v rámci konfigurace podtypů v entitní množině představující nadtyp. Úplnost a výlučnost ale nemá v notaci Information Engineering nástroje Data Modeler specifické symboly.

Na obrázku 20 je znázorněn příklad vztahu typu generalizace/specializace: osoby jsou buď fyzické, nebo právnické. Příklad na obrázku 20 je shodný s příkladem na obrázku 12. Pokud by v nástroji Data Modeler byly realizovány příklady uvedené na obrázcích 13 a 14, znázornění vztahu typu generalizace/specializace by se nelišilo od znázornění tohoto typu vztahu na obrázku 20. Proto zde již varianty modelů na obrázcích 13 a 14 zpracované v nástroji Data Modeler nejsou uvedeny.



Obrázek 20: Vztah generalizace/specializace (notace Information Engineering), zdroj: autoři

2.3 Notace fyzického datového modelu

2.3.1 Notace modelu PDM v nástroji Power Designer

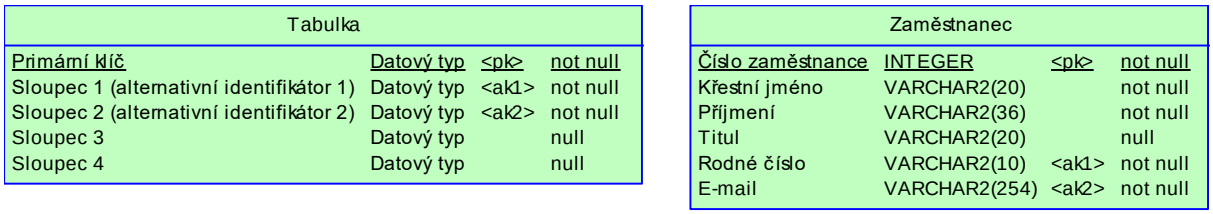
Fyzický datový model je v nástroji Power Designer označován jako Physical Data Model, zkráceně PDM.

Tabulky

Tabulky jsou značeny obdélníkem (obrázek 21), který v horní části obsahuje název tabulky, pod kterým následuje výčet sloupců. Pro každý sloupec je uveden název sloupce a jeho datový typ. Vysvětlení datových typů lze nalézt v dokumentaci cílového SŘBD, kterým je Oracle (Oracle, 2013a). Kromě datového typu mohou být u sloupců uvedeny symboly <pk>, <ak> nebo <fk> označující primární, alternativní, potažmo cizí

[13] Nadtyp se volí v rámci dialogu konfigurace entitní množiny představující podtyp.

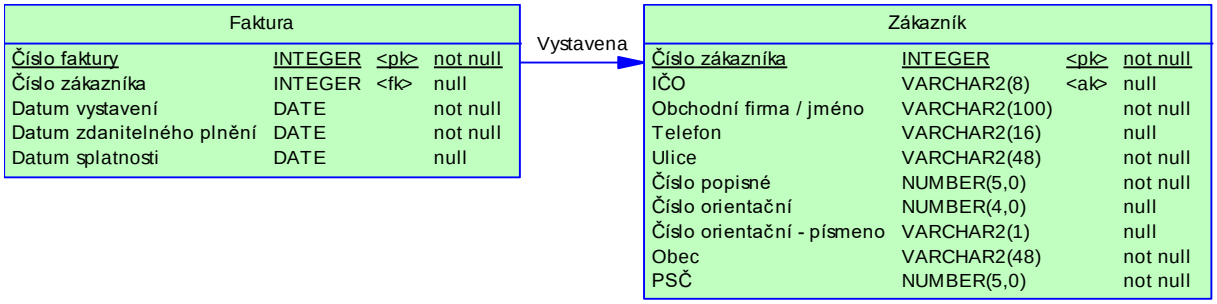
klíč. Tabulky vzniklé transformací elementu asociace z konceptuálního datového modelu mají ve výchozím nastavení barvu shodnou s barvou tohoto symbolu.



Obrázek 21: Značení tabulky (Power Designer), zdroj: autoři

Reference

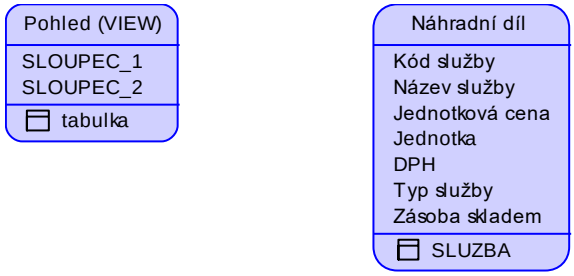
Reference je značena šipkou vedoucí od tabulky, která obsahuje cizí klíč, k tabulce, která obsahuje příslušný primární klíč – viz obrázek 22. Vzhledem ke snaze zarovnat modely tak, aby na stránce byly, pokud možno, co nejčitelnější, nemusí umístění šipek vždy korespondovat s tím, jaké sloupce představující cizí a primární klíč referenci tvoří.



Obrázek 22: Značení reference (Power Designer), zdroj: autoři

Pohledy

Pohled (`VIEW`, viz obrázek 23) je značen rovnoběžníkem se zaoblenými rohy, který v horní části obsahuje název pohledu, pod kterým následuje výčet sloupců. Ve spodní části symbolu pohledu je pak uvedena tabulka nebo tabulky z klauzule `FROM` příkazu `SELECT`, s jehož pomocí je pohled vytvořen.

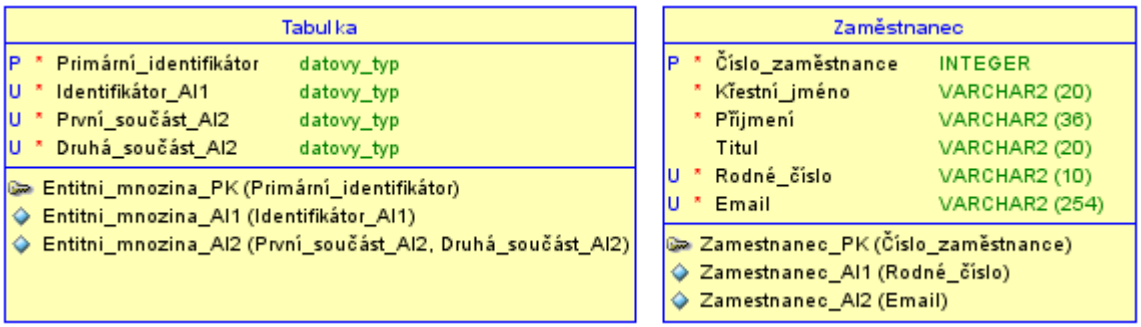


Obrázek 23: Značení pohledu / VIEW (Power Designer), zdroj: autoři

2.3.2 Notace modelu Relational v nástroji Data Modeler

Tabulky

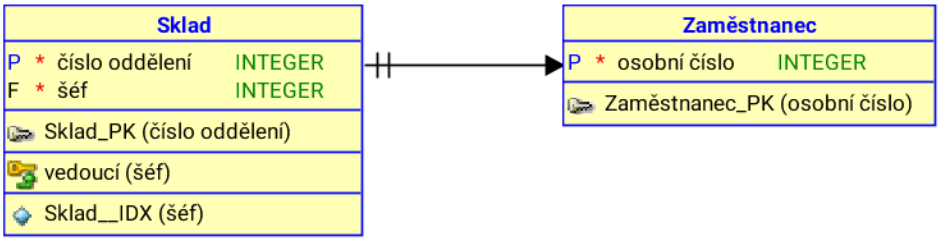
Tabulky jsou značeny obdélníkem (obrázek 24), který v horní části obsahuje název tabulky, pod kterým následuje výčet sloupců. Pro každý sloupec je uveden název sloupce a jeho datový typ. Pod výčtem sloupců je uveden název primárního klíče tabulky, cizích klíčů v tabulce, a případně také název omezení typu `UNIQUE`. Sloupce představující primární klíč, cizí klíč nebo sloupec s unikátními hodnotami jsou označeny symboly `P`, `F` a `U`.



Obrázek 24: Značení tabulky (Data Modeler), zdroj: autoři

Reference

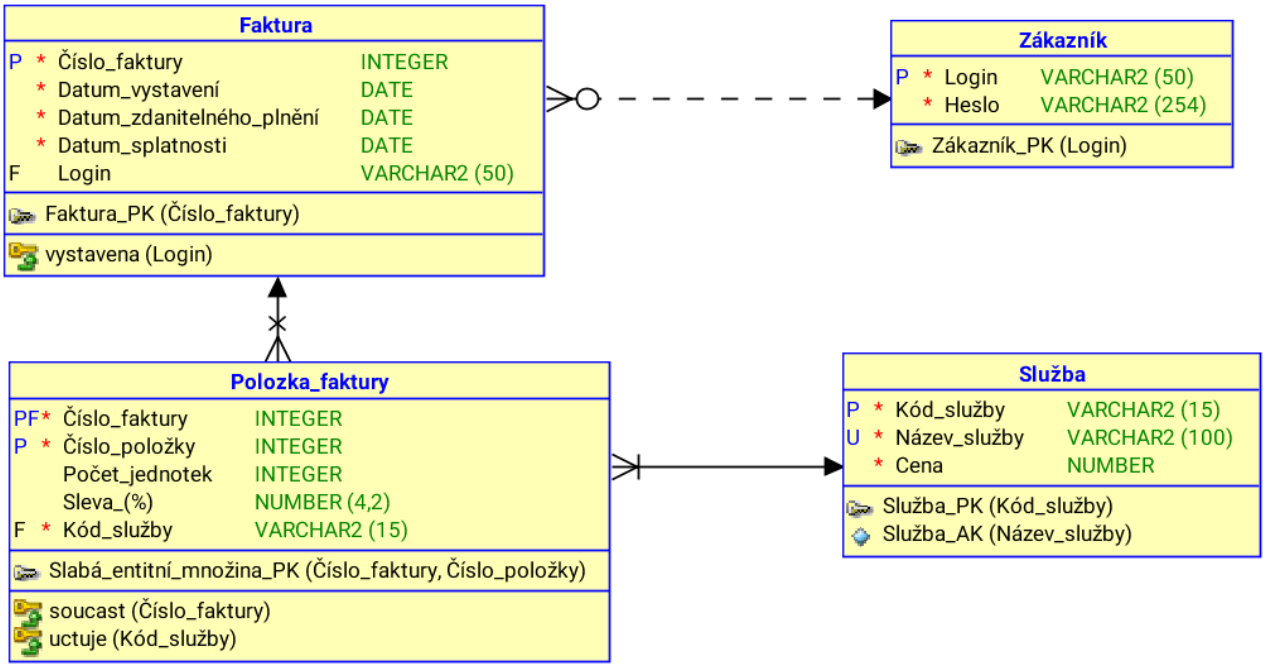
Reference je značena šipkou vedoucí od tabulky, která obsahuje cizí klíč, k tabulce, která obsahuje příslušný primární klíč. Vzhledem ke snaze zarovnat modely tak, aby na stránce byly, pokud možno, co nejčitelnější, umístění šipek obvykle nekoresponduje s tím, jaké sloupce představující cizí a primární klíč referenci tvoří.



Obrázek 25: Značení reference, příklad 1, (Data Modeler), zdroj: autoři

Konec šipky na straně tabulky obsahující cizí klíč je zakončen čárou kolmou k čáře šipky, nebo vraní nohou. Plná čára šipky značí povinný vztah, tj. hodnota cizího klíče nesmí být `NULL`. Přerušovaná čára šipky značí nepovinný vztah, tj. hodnota cizího klíče

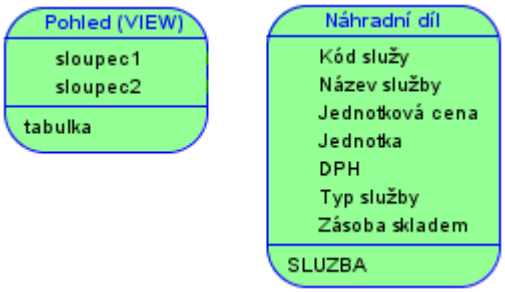
může být prázdná. Čára kolmá k čáře šipky značí referenci, která je transformací vztahu 1:1, tj. sloupec či sloupce cizího klíče musí obsahovat unikátní hodnoty (obrázek 25). Vraní noha značí referenci, která je transformací vztahu 1:N (obrázek 26). Čára kolmá k čáře šipky také značí referenční integrity typu `RESTRICT`, křížek přes čáru šipky pak značí referenční integritu typu `CASCADE`.



Obrázek 26: Značení reference, příklad 2, (Data Modeler), zdroj: autoři

Pohledy

Pohled (`VIEW`, viz obrázek 27) je značen rovnoběžníkem se zaoblenými rohy, který v horní části obsahuje název pohledu, pod kterým následuje výčet sloupců. Ve spodní části symbolu pohledu je pak uvedena tabulka nebo tabulky z klauzule `FROM` příkazu `SELECT`, s jehož pomocí je pohled vytvořen.



Obrázek 27: Značení pohledu / VIEW (Data Modeler), zdroj: autoři

3 Vysvětlení struktury řešených úloh

Řešení úloh popsaných v následujících kapitolách má následující strukturu:

- **Účel úlohy:** Stručné vysvětlení, jaké jevy či problémy řešené při datovém modelování a návrhu databáze úloha ilustruje.
- **Popis zvolené výseče světa (zadání úlohy):** V této části je uveden text, který představuje zadání řešené úlohy. Text popisuje určitou výseč světa, která je relevantní z hlediska návrhu datové základny. Text zejména popisuje, co je předmětem zájmu a mělo by to být evidováno. Nicméně pro pochopení zadání obsahuje text i popis relevantního kontextu, případně i upřesnění, co není třeba evidovat, tj. co leží mimo hranice zadání úlohy.
- **Konceptuální schéma:** V této části je uvedeno konceptuální schéma v notaci, která je popsána v kapitole 2.1.
- **Konceptuální datový model:** V této části je uveden konceptuální datový model vycházející z konceptuálního schématu vytvořený v nástrojích Power Designer a Data Modeler.
- **Fyzický datový model:** V této části je uveden fyzický datový model pro SŘBD Oracle. Obdobně jako v případě konceptuálního datového modelu je v této části uveden fyzický datový model vypracovaný jak v nástroji Power Designer, tak v nástroji Data Modeler.
- **Komentář k návrhu:** V této části je uveden komentář, který vysvětluje provedený návrh. Komentář nemá za cíl být vyčerpávajícím popisem všech prezentovaných modelů, ale jeho cílem je zejména poukázat na klíčové aspekty jednotlivých úloh a ozřejmit jejich řešení.
- **Příkazy pro vytvoření relačních tabulek a dalších objektů:** V této části je uveden **SQL skript** pro realizaci fyzického datového modelu v prostředí cílového databázového systému.

4 Evidence zásob

4.1 Účel úlohy

Tato úloha ilustruje zejména následující problémy/jevy:

- Binární vztahy mezi entitními množinami, kardinalita a parcialita vztahů, atributy vztahů.
- Dominantní směr vztahu v případě vztahu 1:1.
- Dekompozice složených atributů při přechodu z konceptuálního schématu na konceptuální datový model.

4.2 Popis zvolené výseče světa – zadání

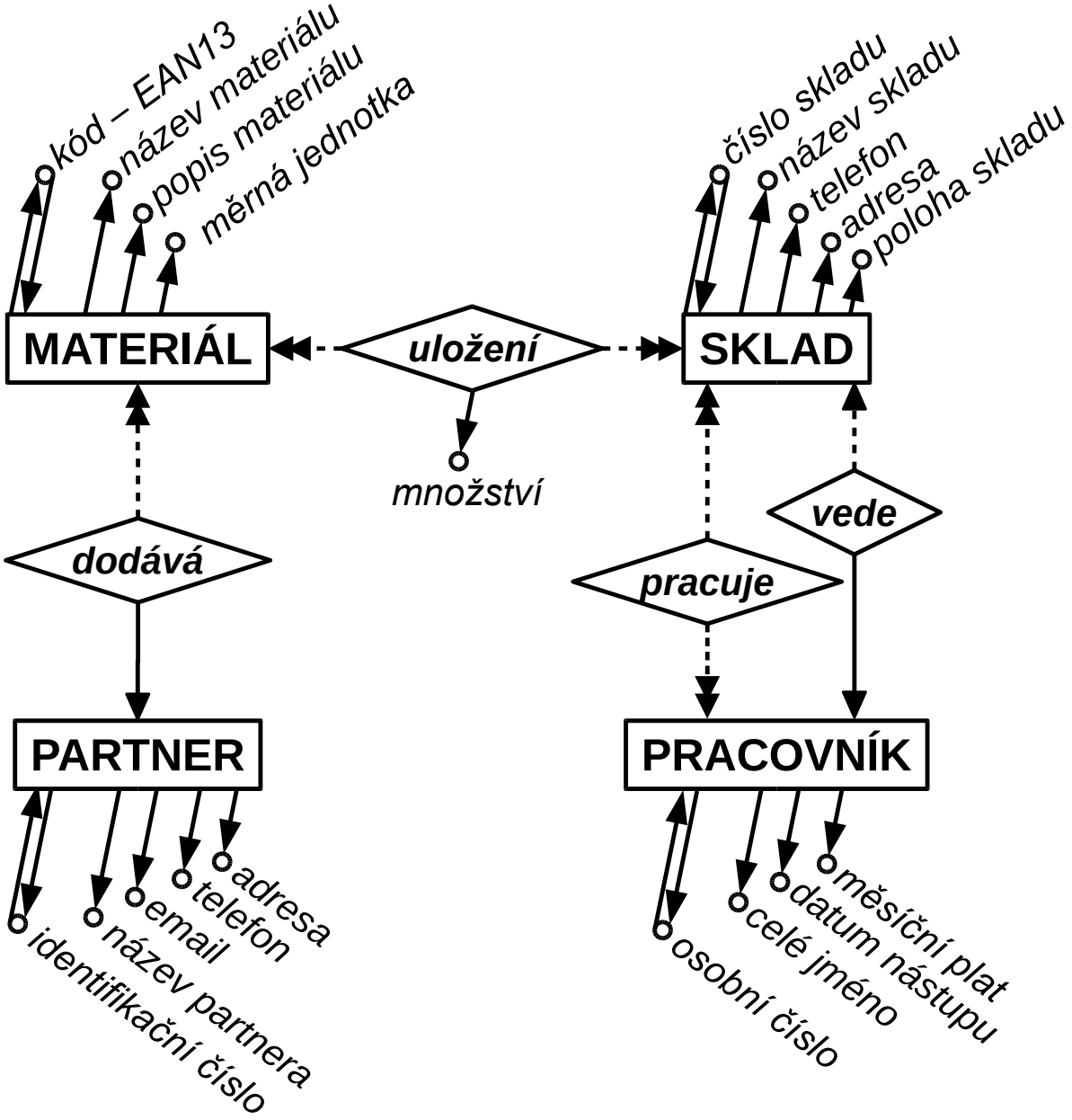
Mezinárodní společnost (založená 16. 8. 2001) potřebuje pro svou činnost cca 4 000 druhů materiálů a každý druh materiálu může být uložen na více skladech. Je nutno mít informace o stavu jednotlivých druhů materiálů na jednotlivých skladech a o celkovém množství materiálu za jednotlivé druhy. Jednotlivé druhy materiálu jsou identifikovány pomocí kódu EAN-13. Dále je o druhu materiálu třeba evidovat název materiálu, popis materiálu a měrnou jednotku z hlediska skladování. Tj. zda je materiál skladován na kusy (např. trubky), nebo např. na tuny (písek). O každém evidovaném druhu materiálu je vyžadováno evidovat všechny tyto údaje, přičemž popis materiálu může představovat i dlouhý text, např. dokument formátovaný pomocí HTML.

U každého druhu materiálu je třeba evidovat, který z obchodních partnerů společnosti je hlavním dodavatelem materiálu. O jednotlivých obchodních partnerech je třeba vždy evidovat identifikační číslo, název, adresu, jedno telefonní číslo a jednu e-mailovou adresu. Společnost chce v databázi evidovat i záznamy o partnerech, kteří dodávali nějaké druhy materiálu v minulosti, ale aktuálně žádný druh materiálu nedodávají. To umožní kontaktovat obchodního partnera v budoucnu, pokud opět bude poptávka po jím dodávaném druhu materiálu.

Skladů je celkem osm, ne každý ale musí být vždy skladem pro nějaký druh materiálu. Přestože se společnost snaží kapacity skladů vytěžovat, občas se stává, že v určitém skladu není uložen žádný druh materiálů. Každému skladu je přiřazeno unikátní číslo skladu a unikátní název. U skladů je vždy třeba evidovat adresu skladu, kontaktní telefon a souřadnice polohy skladu (souřadnice jsou předávány např. řidičům, kteří zaváží materiál na sklady, aby je mohli využít ve své satelitní navigaci).

V každém skladu pracuje obvykle od pěti do deseti pracovníků. Tj. může jich ve skladu pracovat i více než deset, přičemž tato situace je považována za nadstav vůči standardnímu počtu pracovníků ve skladu, nebo méně než pět, přičemž tato situace je považována za podstav. Ne všichni pracovníci společnosti jsou ale přímo zařazeni pro práci v některém ze skladů. I tyto pracovníky je třeba evidovat. Jeden pracovník může pracovat i ve více skladech – někteří pracovníci pracují např. ve dvou skladech. Každý sklad má svého vedoucího, nicméně ne každý pracovník je vedoucím nějakého skladu. Platí pravidlo, že pracovník může být vedoucím pouze jednoho skladu. Je zapotřebí evidovat údaje o pracovnících i vedoucích jednotlivých skladů. Každému pracovníkovi je při nástupu přiděleno osobní číslo. Dále je o každém pracovníkovi požadováno vždy evidovat následující údaje: celé jméno pracovníka, datum nástupu a výše měsíčního platu. Datum nástupu pracovníka nemůže předcházet datu založení společnosti.

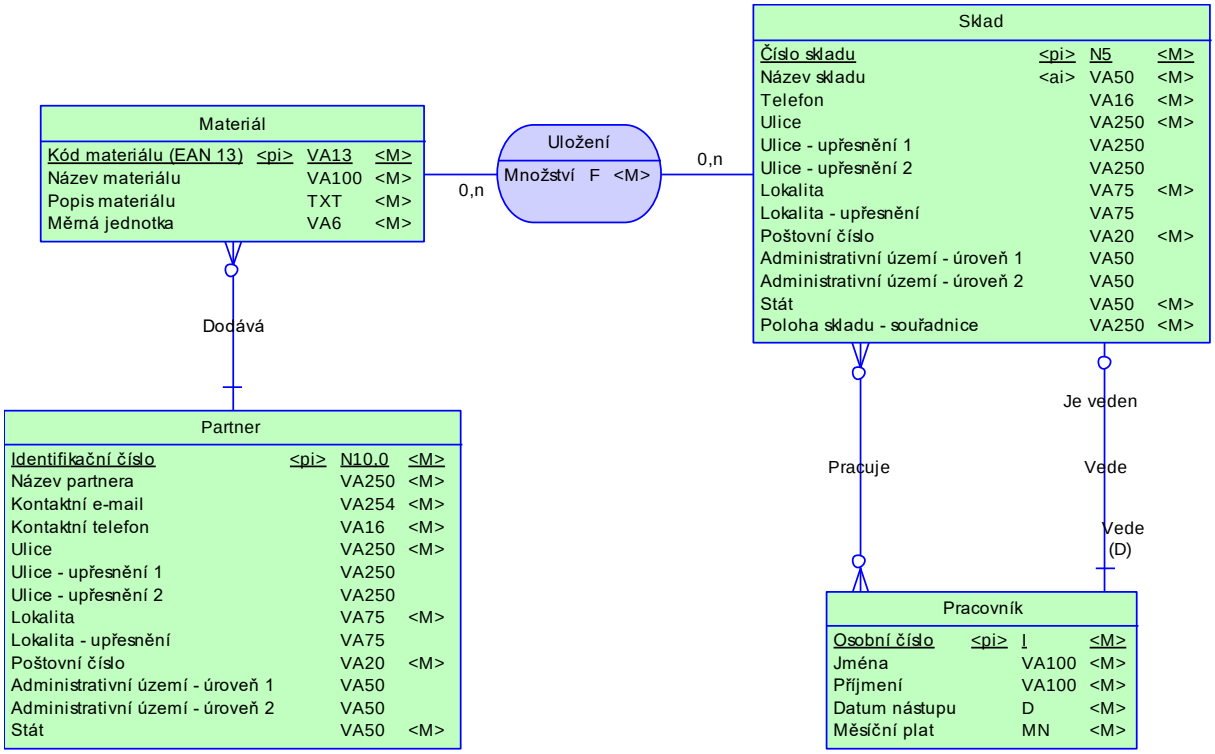
4.3 Konceptuální schéma



Obrázek 28: Evidence zásob – konceptuální schéma, zdroj: autoři

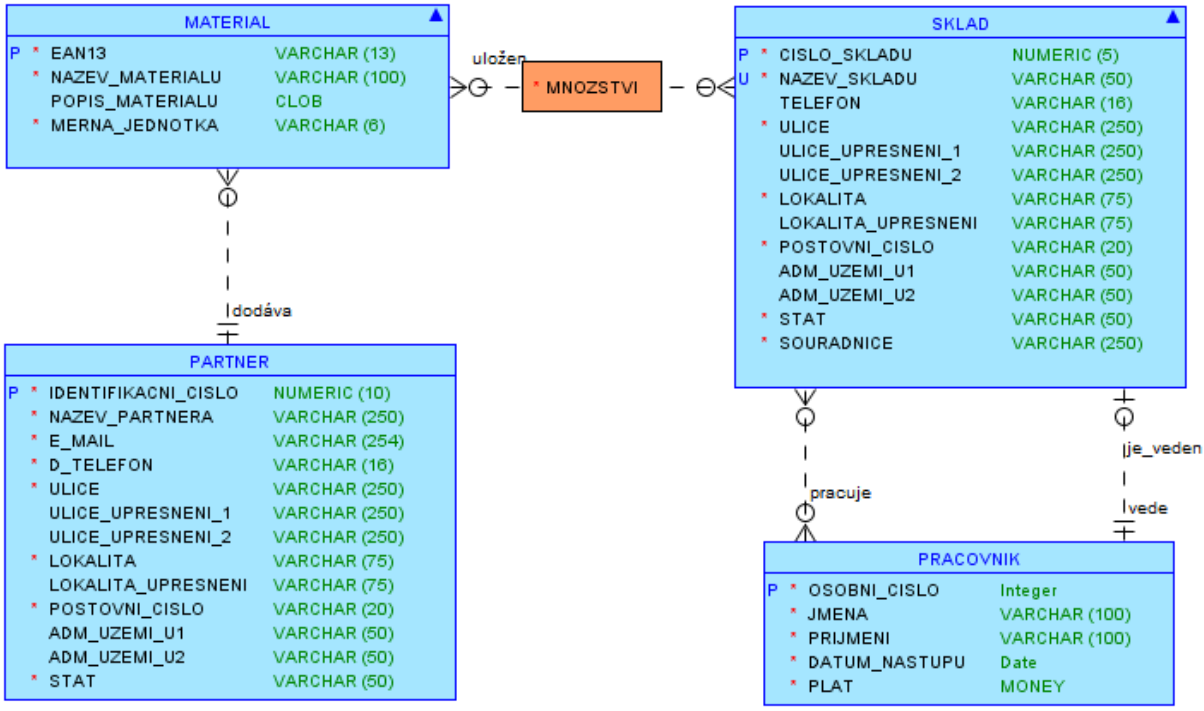
4.4 Konceptuální datový model

4.4.1 KDM v nástroji Power Designer



Obrázek 29: Evidence zásob – konceptuální datový model (Power Designer), zdroj: autoři

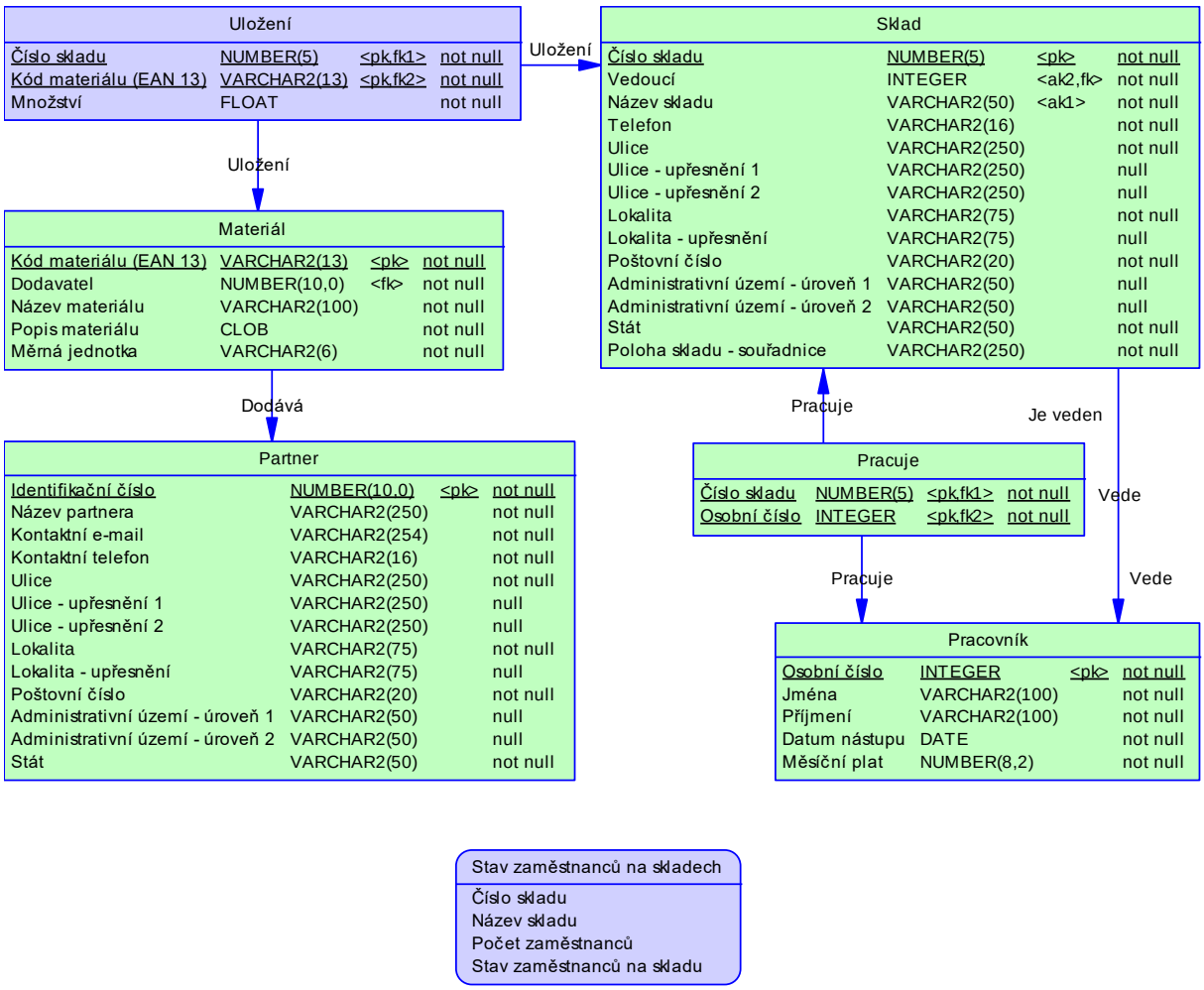
4.4.2 KDM v nástroji Data Modeler



Obrázek 30: Evidence zásob – konceptuální datový model (Data Modeler), zdroj: autoři

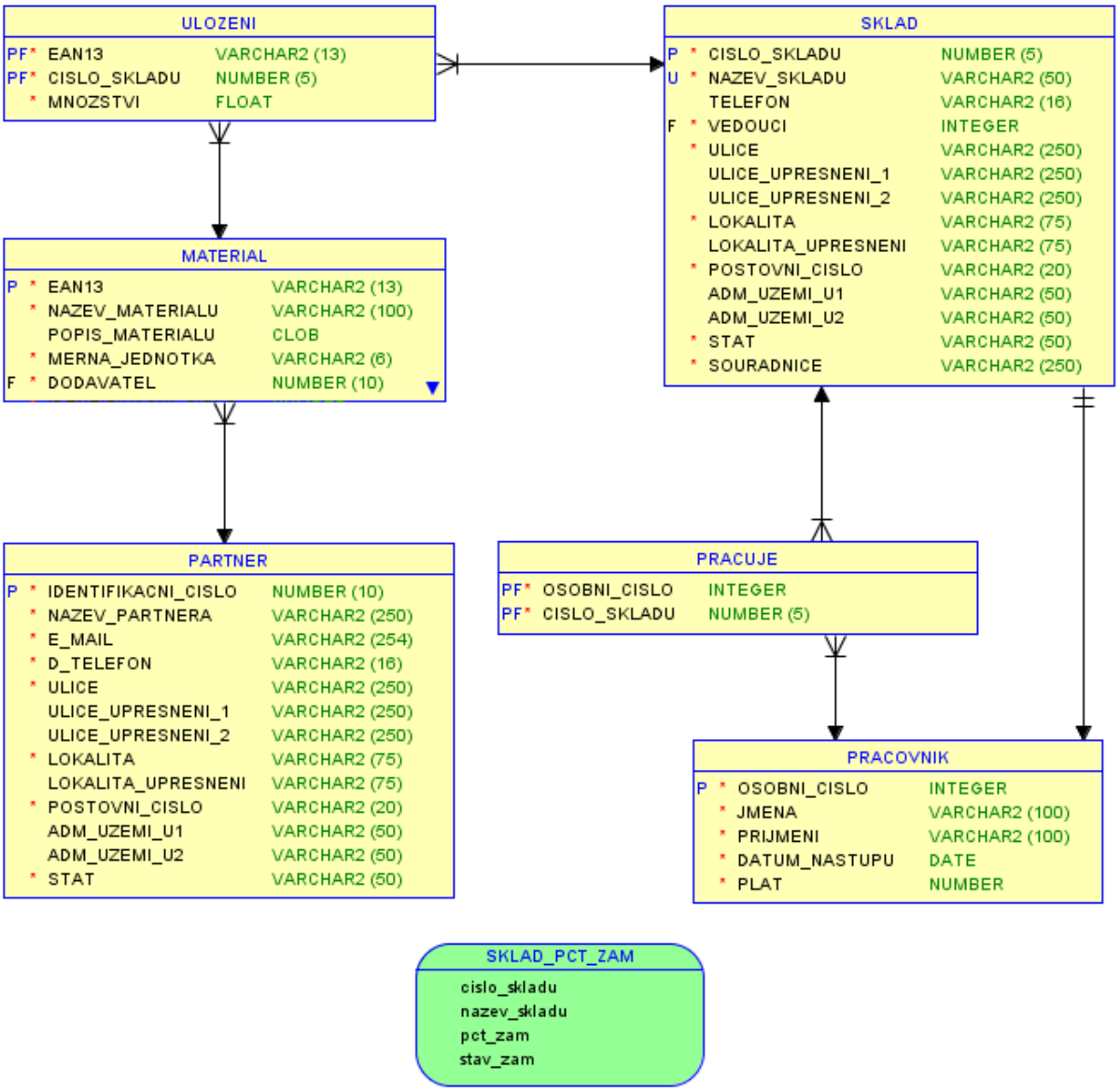
4.5 Fyzický datový model

4.5.1 FDM v nástroji Power Designer



Obrázek 31: Evidence zásob – fyzický datový model (Power Designer), zdroj: autoři

4.5.2 FDM v nástroji Data Modeler



Obrázek 32: Evidence zásob – fyzický datový model (Data Modeler), zdroj: autoři

4.6 Komentář k návrhu

Na základě zadání byly identifikovány čtyři entitní množiny, o kterých je třeba evidovat údaje: *materiál*, *sklad*, *partner* a *pracovník*. Atributy těchto entitních množin vyplývají taktéž ze zadání úlohy. Pracovník ve skladu a vedoucí skladu nepředstavují samostatné entitní množiny, protože se v obou případech jedná o osoby, které společnost, pro kterou je databáze navrhována, zaměstnává. Označení *pracovník ve skladu* a *vedoucí skladu* pouze vyjadřuje, jakou pozici daná osoba ve skladu zastává. Z tohoto důvodu nastávají mezi pracovníkem a skladem dva vztahy: *pracuje* ve skladu a *vede* sklad. Dle podmínek stanovených v zadání se tyto dva vztahy liší svojí

kardinalitou a parcialitou. Databáze je navrhována pro potřeby jedné konkrétní společnosti, která ale sama o sobě nepředstavuje předmět zájmu z hlediska toho, jaká data bude třeba v databázi ukládat. Z tohoto důvodu společnost provozující sklady nevystupuje v modelech jako entitní množina.

V konceptuálním schématu jsou zastoupeny všechny typy vztahů z hlediska kardinality, tj. vztah 1:1 (*vede*), 1:N (*dodává*) i M:N (*pracuje*, *uložen*). Vztah *uložen* má atribut *množství* vyjadřující množství určitého druhu *materiálu* na určitém *skladu*. Aby atribut *množství* mohl toto vyjádřit, nemůže být ani atributem *skladu*, ani atributem *materiálu*. Pokud by byl atribut *množství* atributem *skladu*, mohl by vyjadřovat celkové množství všech druhů *materiálu* na *skladu*, ze kterého by ale množství jednotlivých druhů *materiálu* na daném *skladu* nebylo možné odvodit. Vzhledem k tomu, že různé druhy materiálu mohou mít různou měrnou jednotku, není ani možné množství materiálů s různou měrnou jednotkou sčítat. Obdobně by tomu bylo v případě, že by *množství* bylo atributem *materiálu*. Odpadl by sice problém se sčítáním množství v různých měrných jednotkách, nicméně hodnota atributu *množství*, pokud by to byl atribut *materiálu*, by vyjadřovala celkové množství materiálu bez ohledu na to, na kterém skladu je materiál *uložen*, a množství materiálu uloženého na jednotlivých skladech by nebylo možné z celkového množství odvodit.

V konceptuálním datovém modelu byly složené atributy z konceptuálního schématu, kterými v této úloze jsou *celé jméno* pracovníka a *adresa* skladu a partnera, rozděleny na atributy představující jejich jednotlivé elementy.

O reprezentaci jmen osob v relačních databázích pojednává blíže kapitola 12.1. Při řešení této úlohy byla zvolena varianta, která umožňuje částečné vyjádření sémantiky jednotlivých částí jména pracovníka, proto je zaveden samostatný atribut pro křestní jména a příjmení pracovníka. Pokud by pracovník měl více křestních jmen, do databáze by je bylo nutné zapsat do sloupce pro křestní jméno jako jeden textový řetězec bez rozlišení jejich přesnějšího významu.

O reprezentaci adres v relačních databázích blíže pojednává kapitola 12.2. Při řešení této úlohy byl zvolen způsob reprezentace adresy umožňující částečné vyjádření významu jednotlivých elementů adresy. Protože společnost působí mezinárodně, bylo zvoleno řešení, které umožní zapsat adresy v různých státech. Konkrétně je povinně vyžadováno zadat ulici, nebo jiný nejmenší územní celek v dané zemi používaný

v adrese namísto názvu ulice, lokalitu (např. město/obec), poštovní číslo a stát. Volitelně lze uložit upřesnění ulice, lokality a další dvě úrovně administrativního členění používaného v dané zemi, díky čemuž lze uložit např. označení státu ve federativně uspořádané zemi.

Reprezentace e-mailové adresy, telefonu a polohy skladu odpovídá řešením doporučeným v kapitolách 12.3-12.5. V případě popisu materiálu byl zvolen datový typ `CLOB` (v konceptuálním datovém modelu nástroje Power Designer na obrázku 29 byl z tohoto důvodu zvolen datový typ `text`) a to z toho důvodu, že se dle zadání očekává, že popis materiálu může být i delší text formátovaný např. pomocí HTML.

Vazba *vede* ve směru od *pracovníka* ke *skladu* vyjadřující, že pracovník vede sklad, byla označena jako dominantní. V případě vztahu 1:1 je vždy třeba rozhodnout, do které ze dvou tabulek propojených takovýmto vztahem, bude vhodnější na fyzické úrovni návrhu umístit cizí klíč. Je to jedna ze situací, kdy na konceptuální úrovni musíme nastavit parametry tak, aby bylo možné na fyzické úrovni návrhu co nejpřesněji v databázi zachytit uvedený vztah. V případě tabulek *sklad* a *pracovník* propojených vtahem *vede*, je vhodnější umístit cizí klíč do tabulky *sklad*, protože sklad musí mít stanoveného vedoucího, kdežto ne každý pracovník musí být vedoucím nějakého skladu. Umístění cizího klíče do tabulky *sklad* (sloupec *vedoucí*) umožní, aby pro tento sloupec bylo deklarováno omezení `NOT NULL`, což zajistí, že každý sklad bude muset mít určeného vedoucího. Pro sloupec *vedoucí* v tabulce *sklad* bylo zároveň deklarováno omezení `UNIQUE`, díky čemuž je zajištěno, že jeden pracovník může být vedoucím nejvýše jednoho skladu. Pokud by toto omezení deklarováno nebylo, bylo by možné do sloupce *vedoucí* uvést osobní číslo jednoho a téhož vedoucího vícekrát. Takovýto fyzický návrh databáze by ale neodpovídal vztahu 1:1, nýbrž vztahu 1:N.

Určení dominantní role vztahu v konceptuálním datovém modelu tak upřesňuje, která z entitních množin má být transformována na tabulku, která bude pro daný vztah obsahovat primární, resp. cizí klíč. Primární klíč je „dominantní“ k cizímu. Jako dominantní má tedy být zvolena ta role vztahu, kterou v notaci E/R + Merise nebo Information Engineering interpretujeme proti směru budoucí reference, tj. od primárního k cizímu klíči, resp. od entitní množiny, která bude transformována na tabulku obsahující primární klíč, k entitní množině, která bude transformována na tabulku, ve které bude obsažen cizí klíč.

Pro usnadnění kontroly, zda se aktuální počet pracovníků ve skladu pohybuje v obvyklých mezích, byl do databáze přidán pohled s*tav zaměstnanců na skladech*, který umožní pro každý sklad vypsat počet pracovníků, přičemž u skladů, kde bude počet pracovníků nižší než pět, bude uveden stav „podstav“, a u skladů, kde bude počet pracovníků větší než deset, bude uveden stav „nadstav“. SQL kód pro vytvoření tohoto pohledu včetně příslušného příkazu `SELECT` je uveden ve výpisu 1.

4.7 Příkazy pro vytvoření relačních tabulek a dalších objektů

Výpis 1: Evidence zásob – SQL skript, zdroj: autoři

```
/*==============================================================*/
/* Tabulka: MATERIAL                                            */
/*==============================================================*/
CREATE TABLE MATERIAL  (
   EAN13                VARCHAR2(13)                    NOT NULL,
   DODAVATEL            NUMBER(10,0)                    NOT NULL,
   NAZEV_MATERIALU      VARCHAR2(100)                   NOT NULL,
   POPIS_MATERIALU      CLOB                            NOT NULL,
   MERNA_JEDNOTKA       VARCHAR2(6)                     NOT NULL
      CONSTRAINT CKC_MERNA_JEDNOTKA_MATERIAL
        CHECK (MERNA_JEDNOTKA IN ('g','kg','ks','l','m','m2','m3','t')),
   CONSTRAINT PK_MATERIAL PRIMARY KEY (EAN13),
   CONSTRAINT CKT_MATERIAL CHECK (REGEXP_LIKE (EAN13, '[0-9]{13}'))
);

/*==============================================================*/
/* Index: DODAVATEL_FK                                          */
/*==============================================================*/
CREATE INDEX DODAVATEL_FK ON MATERIAL (
   DODAVATEL ASC
);

```

```
23    /*==============================================================*/
24    /* Tabulka: PARTNER                                             */
25    /*==============================================================*/
26    CREATE TABLE PARTNER  (
27       IDENTIFIKACNI_CISLO  NUMBER(10,0)                    NOT NULL,
28       NAZEV_PARTNERA       VARCHAR2(250)                   NOT NULL,
29       E_MAIL               VARCHAR2(254)                   NOT NULL,
30       D_TELEFON            VARCHAR2(16)                    NOT NULL
31          CONSTRAINT CKC_D_TELEFON_PARTNER
32            CHECK (REGEXP_LIKE (D_TELEFON, '^[+]?[0-9]{1,15}$')),
33       ULICE                VARCHAR2(250)                   NOT NULL,
34       ULICE_UPRESNENI_1    VARCHAR2(250),
35       ULICE_UPRESNENI_2    VARCHAR2(250),
36       LOKALITA             VARCHAR2(75)                    NOT NULL,
37       LOKALITA_UPRESNENI   VARCHAR2(75),
38       POSTOVNI_CISLO       VARCHAR2(20)                    NOT NULL,
39       ADM_UZEMI_U1         VARCHAR2(50),
40       ADM_UZEMI_U2         VARCHAR2(50),
41       STAT                 VARCHAR2(50)                    NOT NULL,
42       CONSTRAINT PK_PARTNER PRIMARY KEY (IDENTIFIKACNI_CISLO)
43    );
44
45    /*==============================================================*/
46    /* Tabulka: PRACOVNIK                                           */
47    /*==============================================================*/
48    CREATE TABLE PRACOVNIK  (
49       OSOBNI_CISLO         INTEGER                         NOT NULL,
50       JMENA                VARCHAR2(100)                   NOT NULL,
51       PRIJMENI             VARCHAR2(100)                   NOT NULL,
52       DATUM_NASTUPU        DATE                            NOT NULL
53          CONSTRAINT CKC_DATUM_NASTUPU_PRACOVNI
54            CHECK (DATUM_NASTUPU >TO_DATE('16.08.2001','DD.MM.YYYY')),
55       PLAT                 NUMBER(8,2)                     NOT NULL
56          CONSTRAINT CKC_PLAT_PRACOVNI CHECK (PLAT >= 0),
57       CONSTRAINT PK_PRACOVNIK PRIMARY KEY (OSOBNI_CISLO)
58    );
59
60    /*==============================================================*/
61    /* Tabulka: PRACUJE                                             */
62    /*==============================================================*/
63    CREATE TABLE PRACUJE  (
64       CISLO_SKLADU         NUMBER(5)                       NOT NULL,
65       OSOBNI_CISLO         INTEGER                         NOT NULL,
66       CONSTRAINT PK_PRACUJE PRIMARY KEY (CISLO_SKLADU, OSOBNI_CISLO)
67    );
68
```

```
69    /*==============================================================*/
70    /* Tabulka: SKLAD                                               */
71    /*==============================================================*/
72    CREATE TABLE SKLAD  (
73       CISLO_SKLADU         NUMBER(5)                     NOT NULL,
74       VEDOUCI              INTEGER                       NOT NULL,
75       NAZEV_SKLADU         VARCHAR2(50)                  NOT NULL,
76       TELEFON              VARCHAR2(16)                  NOT NULL
77          CONSTRAINT CKC_TELEFON_SKLAD
78            CHECK (REGEXP_LIKE (TELEFON, '^[+]?[0-9]{1,15}$')),
79       ULICE                VARCHAR2(250)                 NOT NULL,
80       ULICE_UPRESNENI_1    VARCHAR2(250),
81       ULICE_UPRESNENI_2    VARCHAR2(250),
82       LOKALITA             VARCHAR2(75)                  NOT NULL,
83       LOKALITA_UPRESNENI   VARCHAR2(75),
84       POSTOVNI_CISLO       VARCHAR2(20)                  NOT NULL,
85       ADM_UZEMI_U1         VARCHAR2(50),
86       ADM_UZEMI_U2         VARCHAR2(50),
87       STAT                 VARCHAR2(50)                  NOT NULL,
88       SOURADNICE           VARCHAR2(250)                 NOT NULL,
89       CONSTRAINT PK_SKLAD PRIMARY KEY (CISLO_SKLADU),
90       CONSTRAINT AK_NAZEV_SKL_AI_SKLAD UNIQUE (NAZEV_SKLADU),
91       CONSTRAINT AK_VEDOUCI_AK_SKLAD UNIQUE (VEDOUCI)
92    );
93
94    /*==============================================================*/
95    /* Tabulka: ULOZENI                                             */
96    /*==============================================================*/
97    CREATE TABLE ULOZENI  (
98       CISLO_SKLADU         NUMBER(5)                     NOT NULL,
99       EAN13                VARCHAR2(13)                  NOT NULL,
100      MNOZSTVI             FLOAT                         NOT NULL
101         CONSTRAINT CKC_MNOZSTVI_ULOZENI CHECK (MNOZSTVI >= 0),
102      CONSTRAINT PK_ULOZENI PRIMARY KEY (CISLO_SKLADU, EAN13)
103   );
104
105   /*==============================================================*/
106   /* View: SKLAD_PCT_ZAM                                          */
107   /*==============================================================*/
108   CREATE OR REPLACE VIEW SKLAD_PCT_ZAM AS
109   SELECT CISLO_SKLADU, NAZEV_SKLADU, PCT_ZAM,
110          CASE WHEN PCT_ZAM < 5 THEN 'Podstav'
111          WHEN PCT_ZAM > 10 THEN 'Nadstav'
112          ELSE 'OK' END AS STAV_ZAM
113     FROM
114       (SELECT CISLO_SKLADU, NAZEV_SKLADU, COUNT(OSOBNI_CISLO) AS PCT_ZAM
115          FROM SKLAD LEFT JOIN PRACUJE USING (CISLO_SKLADU)
116          GROUP BY CISLO_SKLADU, NAZEV_SKLADU);
```

```
117
118    ALTER TABLE MATERIAL
119       ADD CONSTRAINT FK_MATERIAL_DODAVA_PARTNER FOREIGN KEY (DODAVATEL)
120          REFERENCES PARTNER (IDENTIFIKACNI_CISLO);
121
122    ALTER TABLE PRACUJE
123       ADD CONSTRAINT FK_PRACUJE_PRACUJE_SKLAD FOREIGN KEY (CISLO_SKLADU)
124          REFERENCES SKLAD (CISLO_SKLADU);
125
126    ALTER TABLE PRACUJE
127       ADD CONSTRAINT FK_PRACUJE_PRACUJE2_PRACOVNI FOREIGN KEY (OSOBNI_CISLO)
128          REFERENCES PRACOVNIK (OSOBNI_CISLO)
129          ON DELETE CASCADE;
130
131    ALTER TABLE SKLAD
132       ADD CONSTRAINT FK_SKLAD_VEDE_PRACOVNI FOREIGN KEY (VEDOUCI)
133          REFERENCES PRACOVNIK (OSOBNI_CISLO);
134
135    ALTER TABLE ULOZENI
136       ADD CONSTRAINT FK_ULOZENI_ULOZENI_SKLAD FOREIGN KEY (CISLO_SKLADU)
137          REFERENCES SKLAD (CISLO_SKLADU);
138
139    ALTER TABLE ULOZENI
140       ADD CONSTRAINT FK_ULOZENI_ULOZENI2_MATERIAL FOREIGN KEY (EAN13)
141          REFERENCES MATERIAL (EAN13)
142          ON DELETE CASCADE;
```

5 Evidence faktur

5.1 Účel úlohy

Tato úloha ilustruje zejména následující problémy/jevy:

- Identifikační závislost entitní množiny (slabé entitní typy).
- Generalizace/specializace a její řešení na konceptuální a fyzické úrovni.
- Jak se vypořádat s tím, že se určitá data mění v čase, přičemž je třeba udržovat jak aktuální data, tak data ve stavu, kdy došlo k nějaké události.

5.2 Popis zvolené výseče světa – zadání

Malé servisní středisko působící v České republice fakturuje zákazníkům své služby. Službou může být operace (např. výměna paměti), nebo náhradní díl (např. určitý typ paměťového modulu). Zatímco u služeb je třeba evidovat popis operace, tak u náhradních dílů je třeba evidovat aktuální skladovou zásobu daného náhradního dílu a volitelně kód produktu od výrobce. Bez ohledu na to, zda se jedná o operaci nebo o náhradní díl, je třeba u služeb evidovat jejich název, jednotkovou cenu, a aktuálně platnou sazbu DPH. Nabízené služby jsou rozlišovány na základě kódu služby, který službám přiděluje servisní středisko. Vedoucí střediska vyžaduje evidovat odděleně operace a náhradní díly a vydal následující pravidla pro tvorbu faktur:

- Každá položka faktury se týká buď náhradního dílu, nebo operace; není přípustné, aby se jedna položka faktury vztahovala k operacím i náhradním dílům.
- Případné slevy se nepromítají přímo do jednotkové ceny, ale u každé položky je třeba v případě slevy uvést jednotkovou cenu v plné výši a samostatně slevu z jednotkové ceny vyjádřenou v procentech.

Ohledně evidence zákazníků stanovil vedoucí servisního střediska následující požadavky: *„Jestliže je možné vystavit zjednodušený daňový doklad, zákazníky nebudeme evidovat, pokud o registraci u nás sami neprojeví zájem. Pokud není možné zjednodušený daňový doklad vystavit, nebo pokud zákazník sám požaduje nezjednodušený daňový doklad, zákazníka zaevidujeme v našem systému, pokud v něm již nebude z dřívějška. O zákaznících potřebujeme evidovat údaje vyžadované zákonem, abychom mohli fakturu vystavit, tj. obchodní firmu nebo jméno včetně dodatku a adresu sídla. U zákazníků, kteří mají přiděleno IČO, budeme evidovat také*

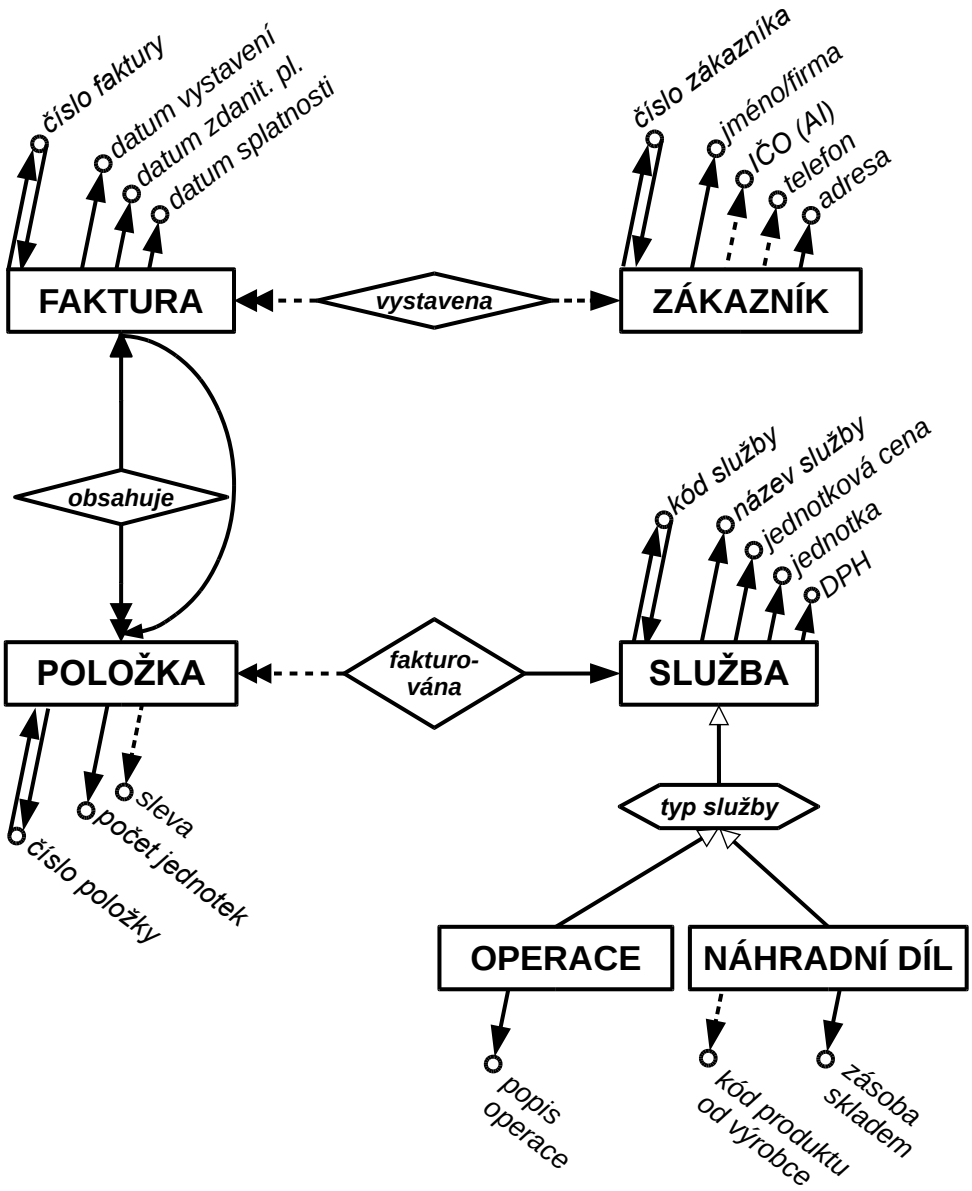
IČO. Abychom mohli zákazníka kontaktovat, tak se při registraci zeptáme, zda nám zákazník poskytne kontaktní telefon, ale to bude dobrovolné. Pokud k nám zákazník přijde opakovaně, tak se před fakturací obsluha zákazníka zeptá, zda jsou námi evidované údaje stále platné, případně je v evidenci aktualizuje.“

5.2.1 Poznámky k zadání

Ve výše popsaném zadání vedoucí servisního střediska poukazuje na skutečnost, že faktura je daňovým dokladem dle zákona č. 235/2004 Sb., o dani z přidané hodnoty. Náležitosti daňového dokladu upravuje § 29 tohoto zákona. Vystavování zjednodušených daňových dokladů je pak upraveno v § 30. Pro řešení úlohy je pak dále relevantní, že faktury také představují účetní doklady podle zákona č. 563/1991 Sb., o účetnictví. Kompletní rozbor dopadů legislativy na řešení této úlohy jde nad rámec těchto skript. Stejně tak zde není cílem provádět návrh kompletní databáze účetního softwaru. Nicméně je dobré poukázat na skutečnost, že platná legislativa návrh databází ovlivňuje, a že vedoucí servisního střediska, aniž by to explicitně řekl, zcela oprávněně očekává, že databáze, resp. informační systém, který databázi bude využívat, budou navrženy tak, aby s jejich pomocí činnosti a procesy, které jsou předmětem zadání, probíhaly v souladu s platnou legislativou. Při řešení této úlohy budeme tedy uvažovat následující:

- Vedoucí servisního střediska sice explicitně neuvedl, jaké údaje má faktura a její položky obsahovat, zde ale vyjdeme z § 29 a § 30 zákona č. 235/2004 Sb.
- Aniž bychom si kladli za cíl navrhnout databázi tak, aby vyhovovala potřebám účetního systému na vedení průkazných účetních záznamů dle zákona č. 563/1991 Sb., ukážeme dále dva možné přístupy, kterými bude možné zajistit, aby na základě dat uložených v databázi bylo kdykoli možné vytisknout fakturu tak, jak vypadala v okamžiku fakturace poskytnuté služby. Tj. pokud si zákazník např. nechá 15. 4. 2019 opravit displej mobilního telefonu, přičemž o něm v té době byla evidována adresa, tak pokud u něj 26. 7. 2019 dojde ke změně adresy, mělo by být možné např. 3. 9. 2019 vytisknout fakturu s původní adresou zákazníka. Pro porovnání s prezentovanými přístupy pro uložení historických dat zde ukážeme i řešení, v rámci kterého historická data ukládána nebudou.

5.3 Konceptuální schéma



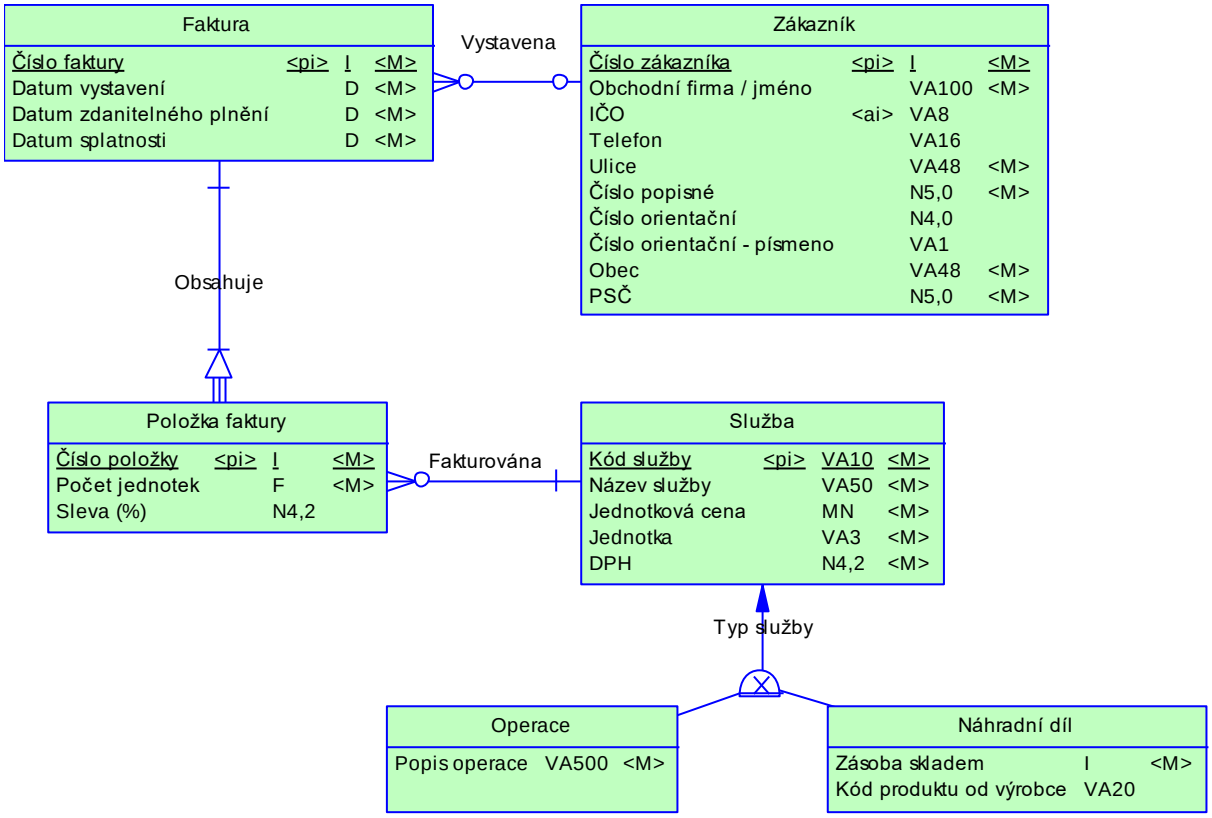
Obrázek 33: Evidence faktur – konceptuální schéma, zdroj: autoři

5.4 Konceptuální datový model

Na obrázcích 34 a 35 je uveden konceptuální datový model pro řešenou úlohu evidence faktur, který vychází z konceptuálního schématu na obrázku 33. Vzhledem ke konceptuální povaze modelu na této úrovni ještě není řešeno, jak se vypořádat se změnami údajů o zákaznících a nabízených službách v čase, a zároveň zajistit, aby na základě dat v databázi bylo možné kdykoli vytisknout fakturu ve stavu, který odpovídá okamžiku, kdy byla služba vyfakturována. V různých systémech řízení báze dat by pro

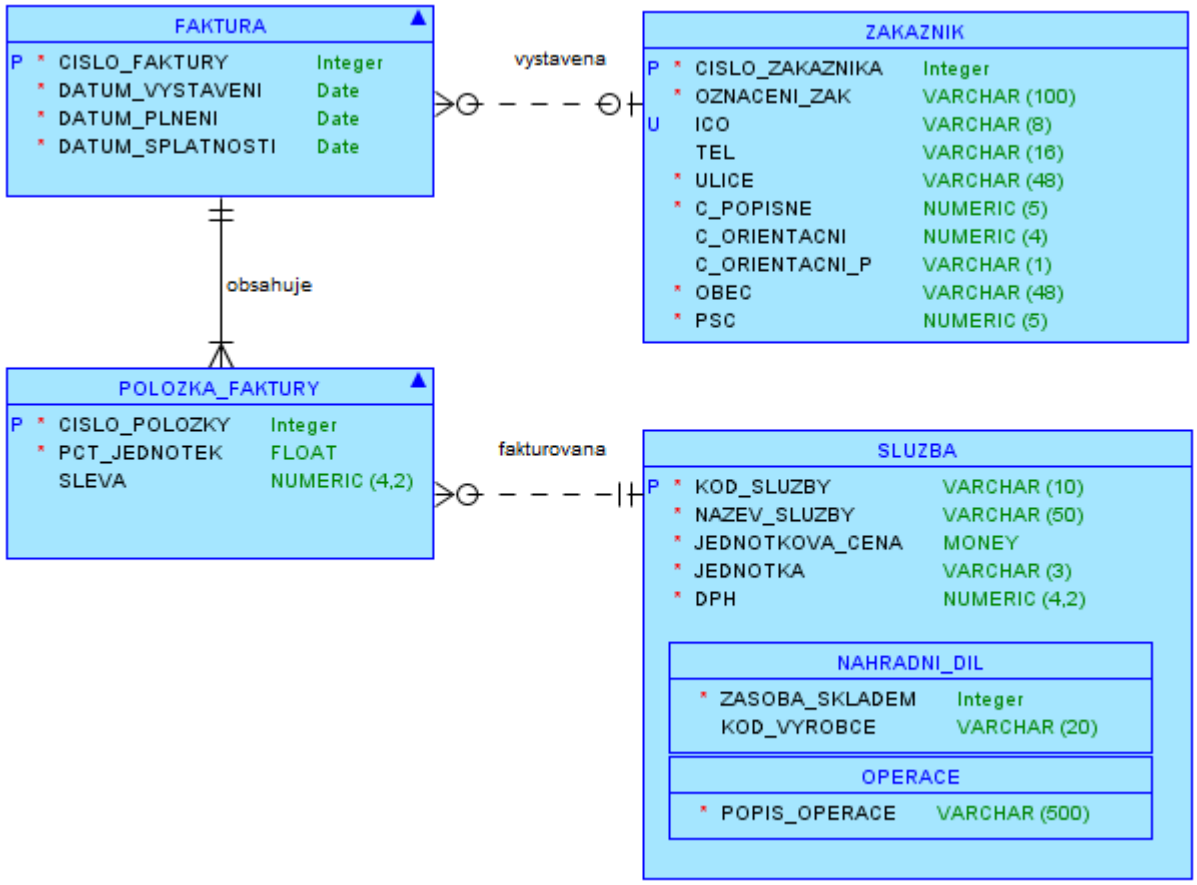
zajištění tohoto požadavku bylo možné volit jiné přístupy, proto budou některé z možných přístupů prezentovány až na úrovni fyzického datového modelu.

5.4.1 KDM v nástroji Power Designer



Obrázek 34: Evidence faktur – konceptuální datový model (Power Designer), zdroj: autoři

5.4.2 KDM v nástroji Data Modeler



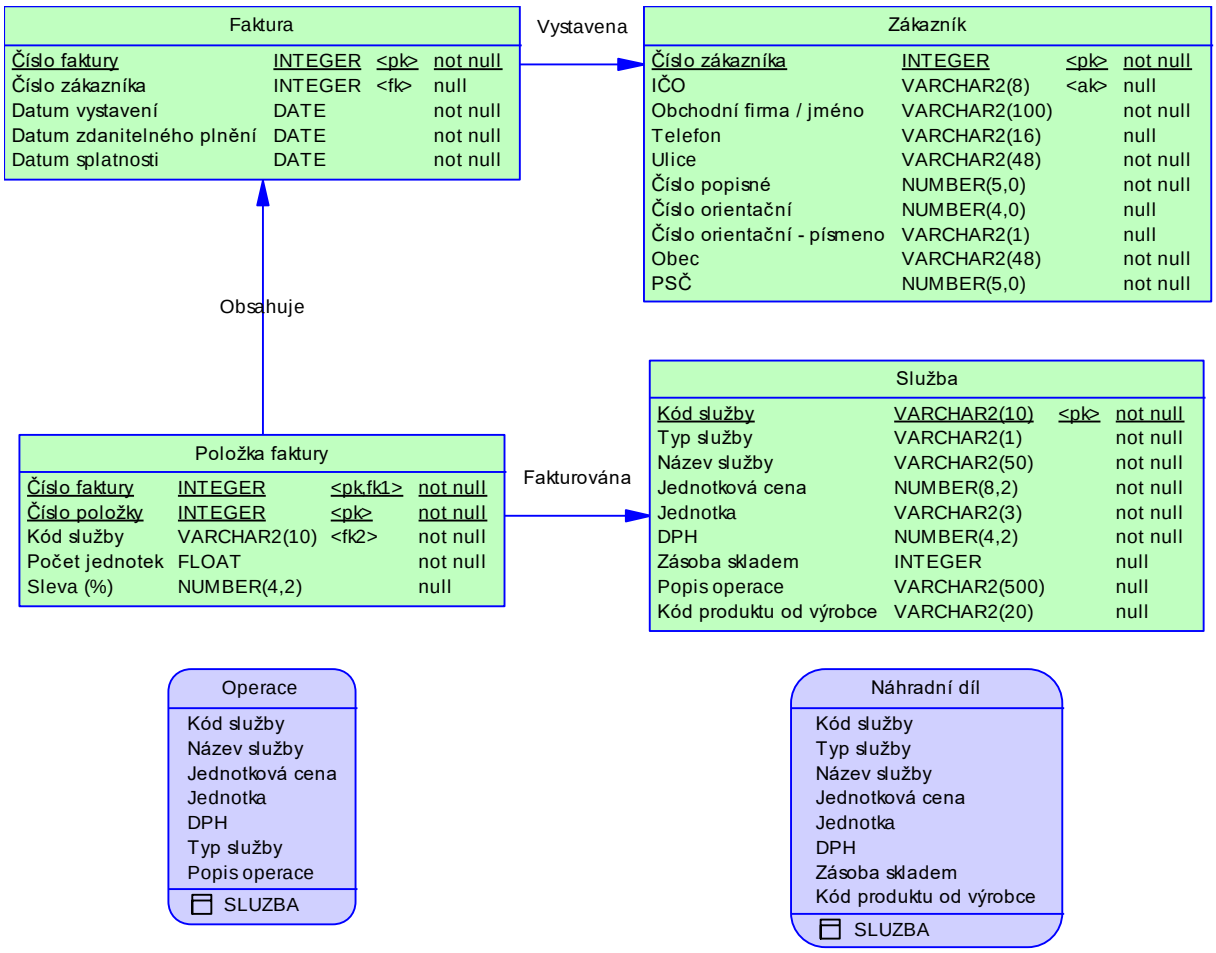
Obrázek 35: Evidence faktur – konceptuální datový model (Data Modeler), zdroj: autoři

5.5 Fyzický datový model

5.5.1 Varianta 1 bez historie změn údajů o zákaznících a službách

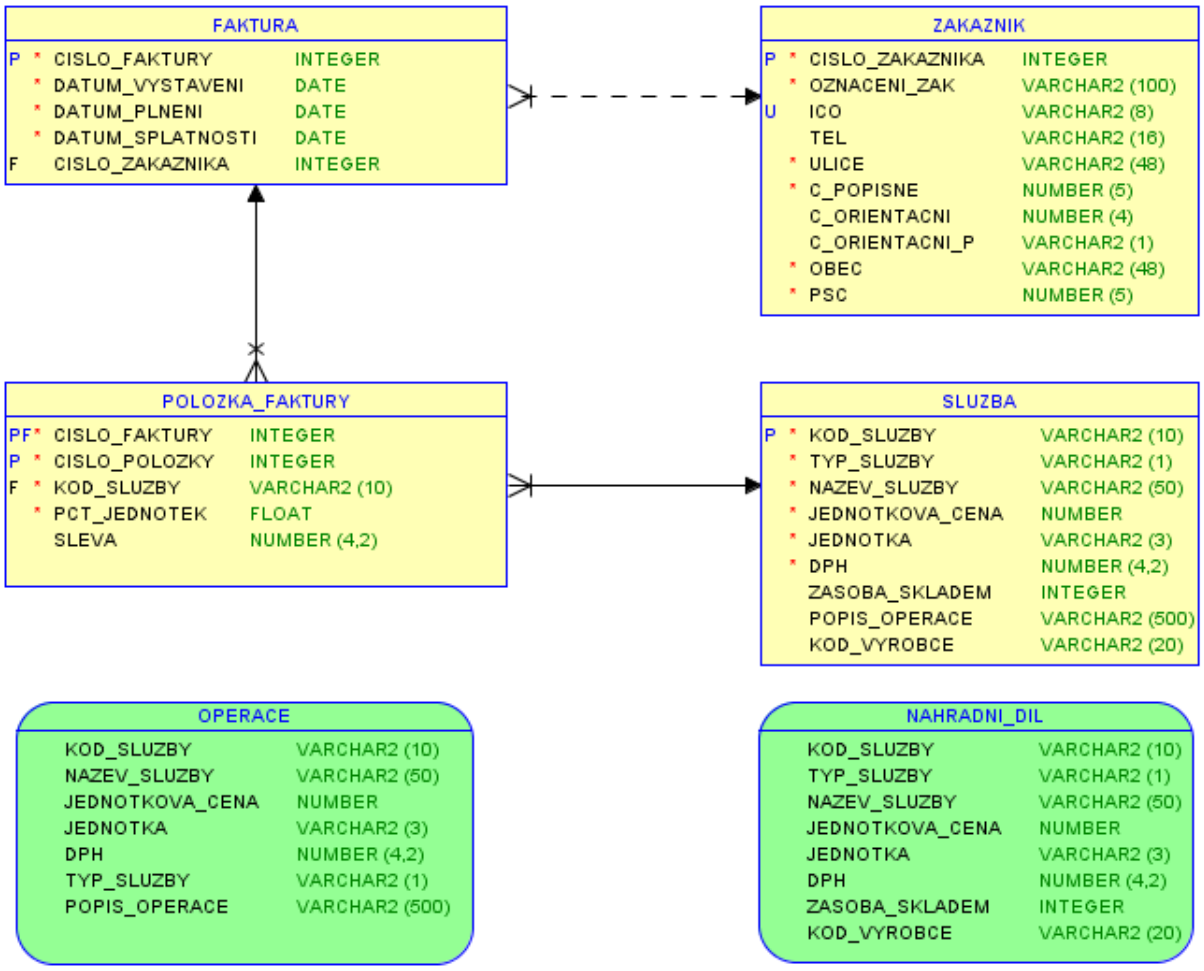
Na obrázcích 36 a 37 je znázorněn fyzický datový model úlohy *Evidence faktur* ve variantě, kdy je v databázi možné měnit údaje o zákaznících a službách, ale při změně těchto údajů dojde k přepsání předešlých údajů. Pokud od okamžiku vystavení faktury dojde např. ke změně adresy zákazníka, nebo k přejmenování služby, na základě dat uložených v databázi nebude možné znovu vytisknout fakturu s těmi údaji, které platily v okamžiku, kdy byla faktura vystavena. Pro zachování faktury ve stavu k okamžiku vystavení by bylo nutné vytvořit snímek příslušných dat a ten někam trvale uložit, např. provést export faktury do souboru (PDF sloužící k tisku, nebo odeslané zákazníkovi). Vytvoření a uložení takovéhoto snímku dat jde nad rámec řešení této úlohy.

FDM v nástroji Power Designer



Obrázek 36: Evidence faktur – fyzický datový model, varianta 1 (Power Designer), zdroj: autoři

FDM v nástroji Data Modeler



Obrázek 37: Evidence faktur – fyzický datový model, varianta 1 (Data Modeler), zdroj: autoři

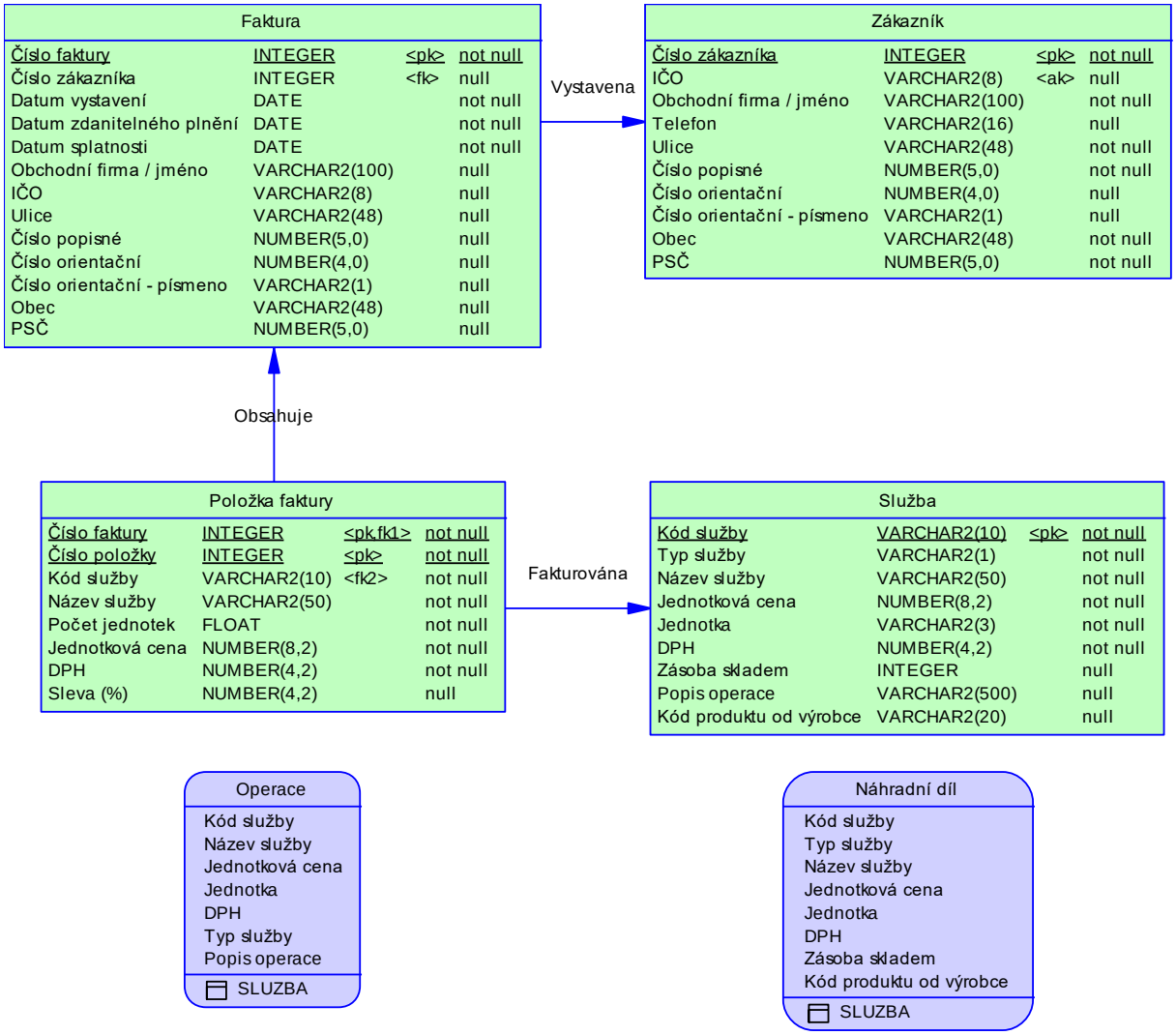
5.5.2 Varianta 2 s ukládáním historických údajů na faktuře

Na obrázcích 38 a 39 je znázorněn fyzický datový model, který umožňuje ukládání historických údajů a to tak, že hodnoty sloupců tabulek *Zákazník* a *Služba*, které se mohou měnit a jsou relevantní pro to, aby bylo možné fakturu vždy vytisknout ve stavu, který by odpovídal stavu v okamžik vystavení faktury, budou zkopírovány při založení faktury do příslušných sloupců v tabulce *Faktura*, resp. *Položka faktury*. Údaje o zákaznících a službách v příslušných tabulkách lze tedy aktualizovat podle skutečnosti, a zároveň bude možné faktury vytisknout na základě dat uložených v tabulkách *Faktura* a *Položka faktury* (za předpokladu, že v těchto tabulkách nedojde ke změně dat).

Toto řešení umožňuje ukládat jak aktuální, tak historická data o zákaznících a službách, nicméně má i své nevýhody, resp. omezení. Ta vyplývají ze skutečnosti, že v tabulkách *Faktura* a *Položka faktury* jsou ukládána data, která ale neodpovídají atributům faktur a jejich položek, tj. jedná se o hodnoty vlastností zákazníků a služeb. Údaje o zákaznících a službách budou ukládány redundantně, jelikož budou uloženy nejen v tabulkách *Zákazník* a *Služba*, ale údaje o zákazníkovi budou uloženy na každé faktuře zákazníka a údaje o službě budou uloženy na každé položce faktury, prostřednictvím které je služba fakturována, a to i když nedojde k jejich změně.

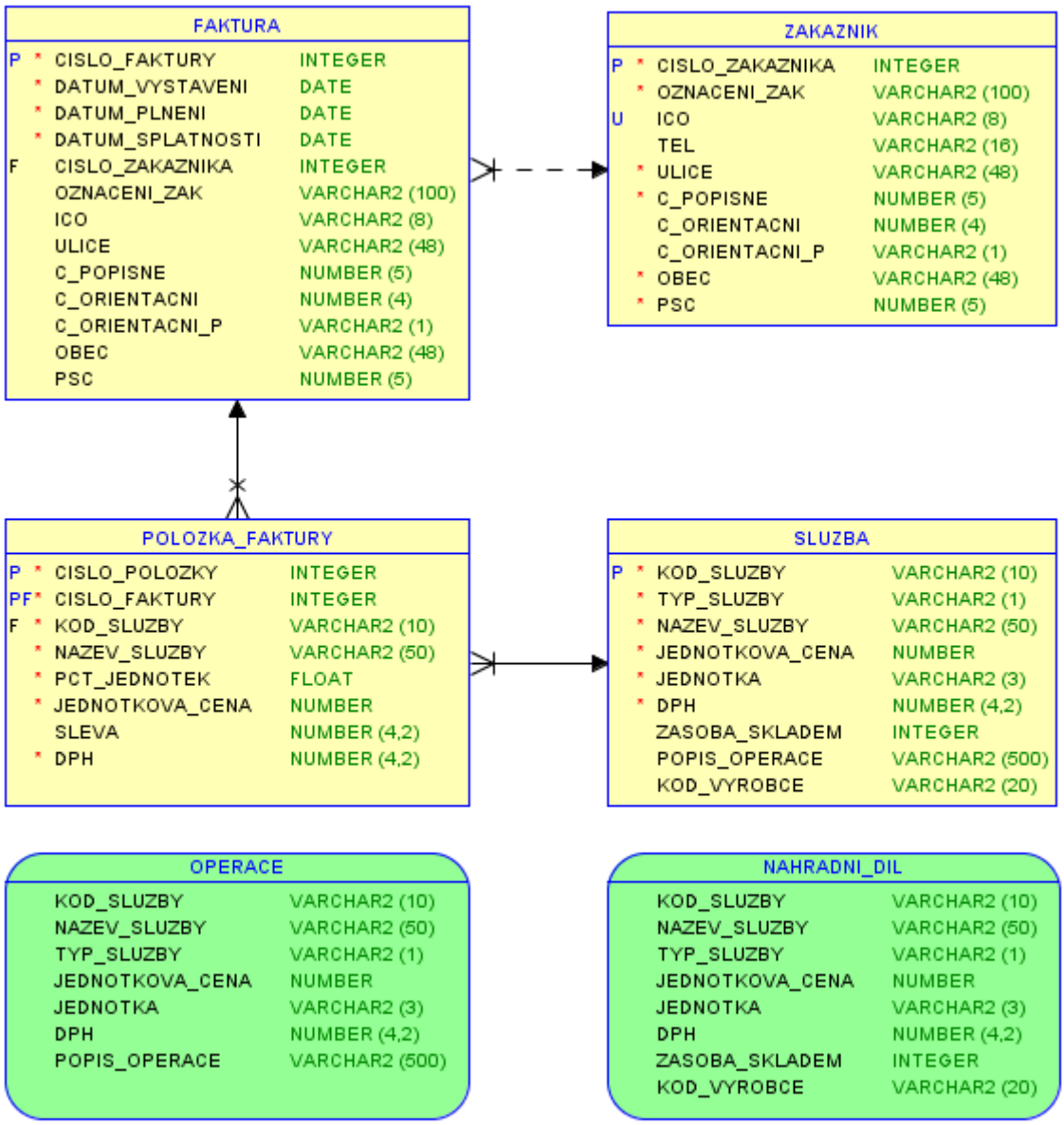
Kdyby vedoucí servisního střediska chtěl zjistit, jak se vyvíjela cena určité služby v čase, bylo by nutné porovnávat ceny dané služby na jednotlivých položkách faktur. Tento přístup nemusí vést k úplnému přehledu o vývoji cen služeb, protože záznam o tom, že služba byla za určitou cenu nabízena, vznikne pouze tehdy, pokud za danou cenu byla služba alespoň jednou vyfakturována. Hypoteticky by mohla nastat situace, kdy servisní středisko bude po určitou dobu nabízet službu za cenu, kterou zákazníci budou vnímat jako příliš vysokou a nikdo za danou cenu službu nebude konzumovat. Nevznikne tak žádná položka faktury, kde by služba měla danou cenu.

FDM v nástroji Power Designer



Obrázek 38: Evidence faktur – fyzický datový model, varianta 2 (Power Designer), zdroj: autoři

FDM v nástroji Data Modeler



Obrázek 39: Evidence faktur – fyzický datový model, varianta 2 (Data Modeler), zdroj: autoři

5.5.3 Varianta 3 s ukládáním historických údajů mimo fakturu

Řešení ukládání historických dat o zákaznících a službách použité ve fyzickém modelu na obrázcích 40 a 41 je inspirováno jednou z technik pro řešení tzv. pomalu se měnících dimenzí v datových skladech popsanou Kimballem a Rossovou (2013, str. 53-56, 148-164). Konkrétně pak variantami řešení pomalu se měnících dimenzí označovaných jako typ 2, resp. typ 7. Řešení je založeno na tom, že každá změna v údajích o zákazníkovi nebo službě je evidována a povede ke vzniku nového řádku v tabulkách

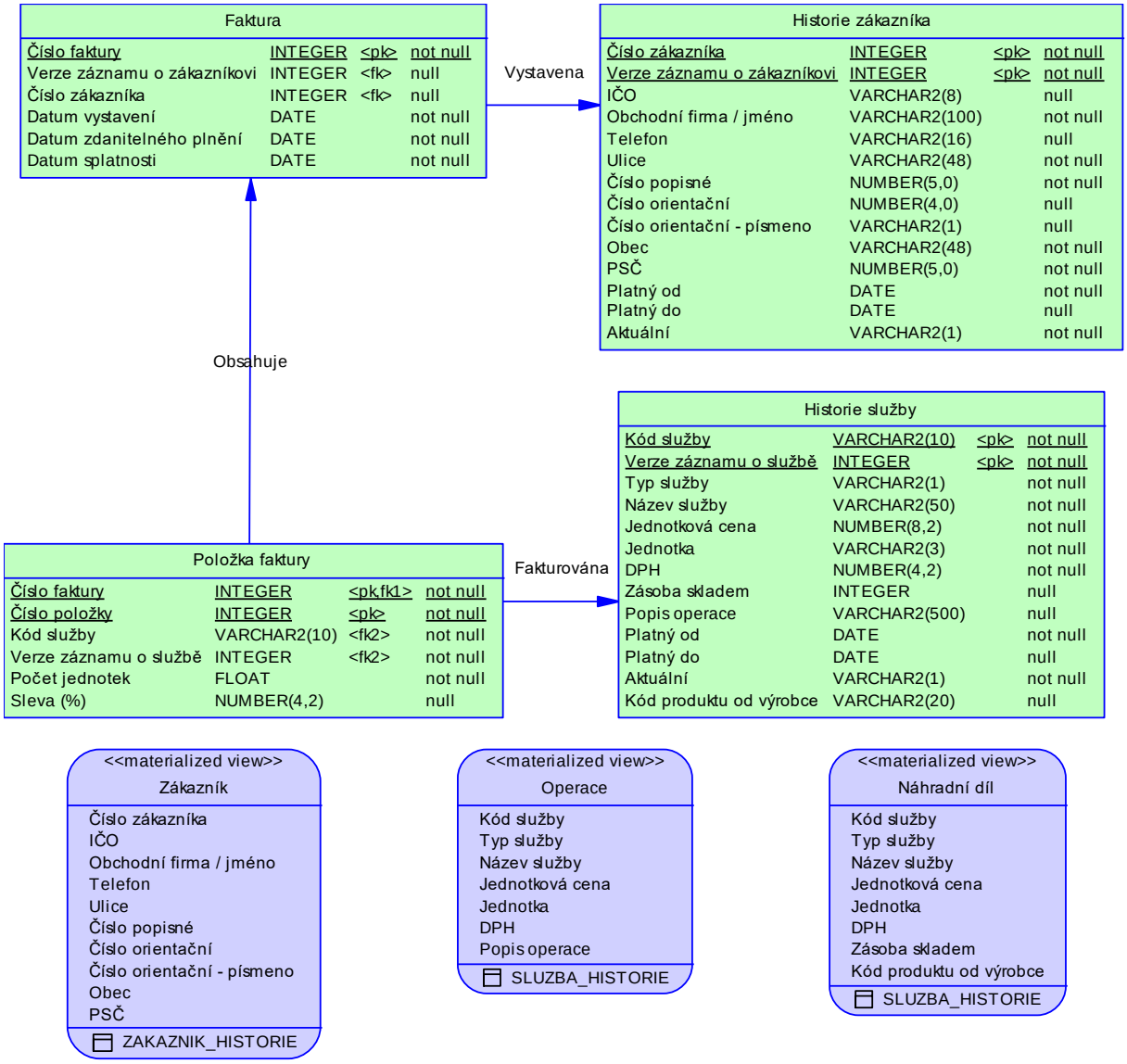
Historie zákazníka a *Historie služby*. V těchto tabulkách jsou kromě sloupců vzniklých transformací atributů příslušných entitních množin z konceptuálního datového modelu obsaženy i sloupce *Platný od*, *Platný do* a *Aktuální*. Tyto sloupce slouží k označení období platnosti záznamu, a k označení, který ze záznamů obsahuje nejaktuálnější údaje. Při změně údajů o zákazníkovi nebo službě tak dojde nejen k přidání nového řádku do příslušné tabulky, ale u řádku, který byl před změnou označen jako aktuální, dojde k ukončení jeho platnosti.

Faktura a položka faktury jsou pomocí reference spojeny s tím řádkem, který v době vzniku faktury nebo její položky představoval aktuální záznam o zákazníkovi nebo službě. Protože změna údajů o zákazníkovi nebo službě povede ke vzniku nových řádků v příslušných tabulkách a nikoli k aktualizaci řádků, na které faktura nebo položka faktury odkazují pomocí reference, bude možné i po změně vytisknout fakturu s údaji platnými v době vystavení faktury.

Databáze umožní získat přehled o zákaznících a službách obsahující aktuální údaje. V SŘBD Oracle lze toto zajistit s využitím materializovaných pohledů. Ty lze vytvořit tak, aby jejich obsah byl automaticky aktualizován na základě potvrzených transakcí v tabulkách *Historie zákazníka* a *Historie služby*. Kromě toho lze deklarovat v materializovaných pohledech primární klíč. Příslušný SQL kód je uveden ve výpisu 4. Materializované pohledy jsou vytvořeny tak, aby zobrazovaly pouze aktuální záznamy o zákaznících a službách. Primárním klíčem v materializovaném pohledu *Zákazník* je *Číslo zákazníka*, v případě materializovaných pohledů *Operace* a *Náhradní díl* je to *Kód služby*. Díky primárním klíčům materializovaných pohledů bude také vynucováno, že pro zákazníka nebo službu bude vždy existovat nejvýše jeden aktuální záznam – při pokusu vložit další řádek s příznakem aktuálního záznamu pro zákazníka nebo operaci, pro které již aktuální záznam existuje, dojde k porušení příslušných omezení primárních klíčů a k selhání transakce.

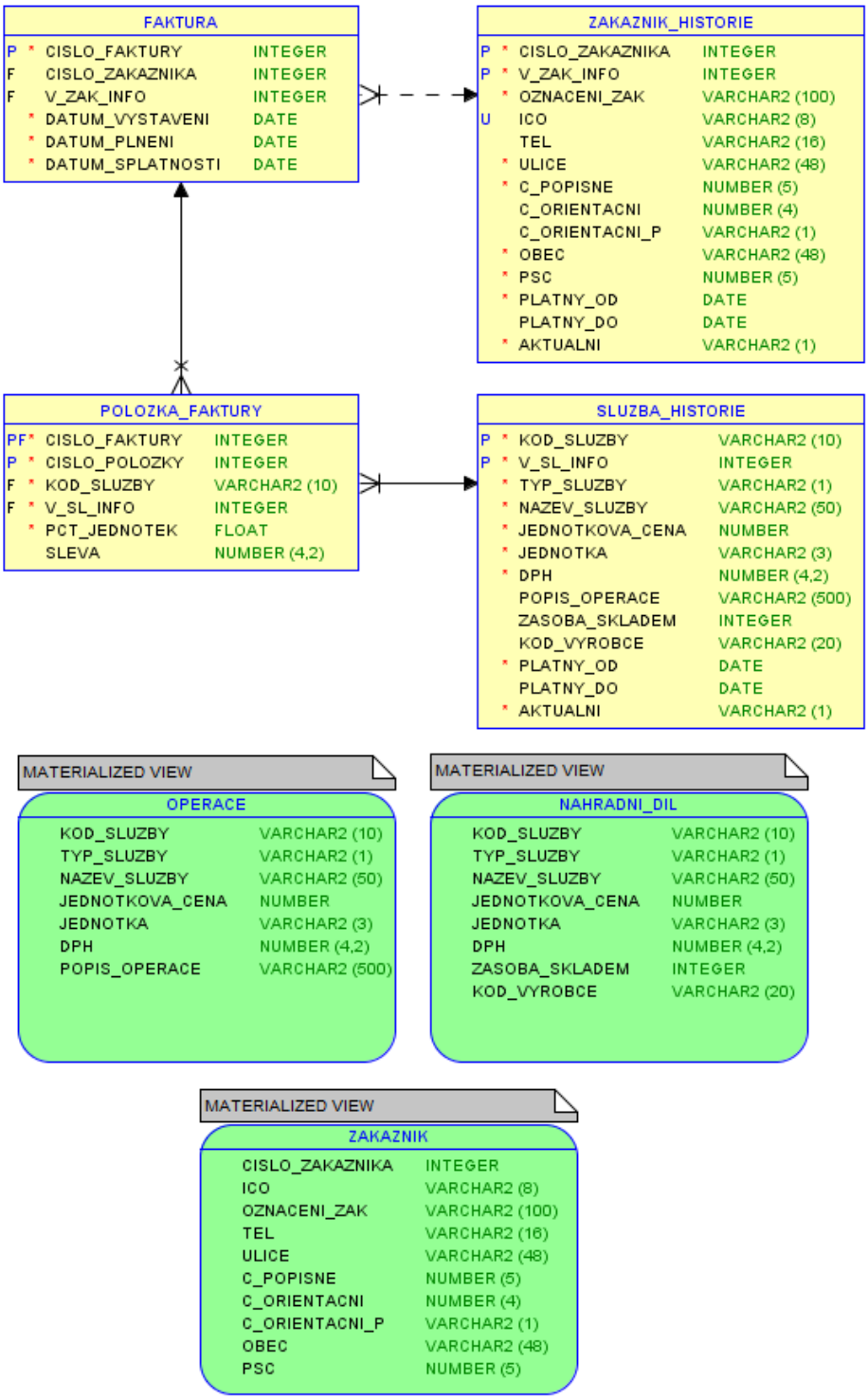
Oproti doporučení Kimballa a Rossové (2013) používat umělé klíče jsou v tabulkách *Historie zákazníka* a *Historie služby* použity klíče tvořené kombinací přirozeného klíče a čísla verze záznamu. Díky tomu bude přirozený klíč figurovat i jako cizí klíč v tabulkách *Faktura* a *Položka faktury*. Faktury a položky faktury tak bude možné snadno spojit operací JOIN s příslušnými řádky materializovaných pohledů, což umožní např. vypsat pro každého zákazníka všechny jeho faktury.

FDM v nástroji Power Designer



Obrázek 40: Evidence faktur – fyzický datový model, varianta 3 (Power Designer), zdroj: autoři

FDM v nástroji Data Modeler



Obrázek 41: Evidence faktur – fyzický datový model, varianta 3 (Data Modeler), zdroj: autoři

5.6 Komentář k návrhu

U jednotlivých variant fyzického modelu již byly vysvětleny zde zvolené způsoby řešení problému dat měnících se v čase, které je zároveň třeba uchovat ve stavu, který platil v určitém okamžiku. V následujících odstavcích je uveden komentář k dalším aspektům řešení úlohy.

Na základě zadání byly identifikovány entitní množiny *zákazník*, *faktura*, *položka*, *služba*, přičemž u *služby* nastává vztah typu generalizace/specializace a služba má dva podtypy: *operaci* a *náhradní díl* (viz dále). Atributy entitních množin v konceptuálním schématu na obrázku 33 vycházejí ze zadání úlohy.

Položky faktury přestavují jednotlivé řádky faktury, na kterých jsou fakturovány služby. Položka faktury je její součástí a nemůže existovat sama o sobě. Vztah *obsahuje* mezi entitními množinami *faktura* a *položka faktury* představuje vztah typu kompozice. Dalším projevem této povahy vztahu mezi *fakturou* a *položkou* je identifikační závislost *položky* na *faktuře*. Protože položka faktury nemůže existovat sama o sobě, je její identita z části odvozena od identity faktury, do které náleží. Protože ale na faktuře může být položek více, je nutné rozlišit jednotlivé položky na téže faktuře. Položky na faktuře lze uspořádat, díky čemuž je možné říci, že určitá položka na faktuře je první, jiná druhá atd. Pořadí položky na faktuře vyjadřuje atribut *číslo položky*, který je také součástí primárního identifikátoru *položky*. Díky tomu je pak ve fyzickém modelu (obrázky 36 a 37) primární klíč tabulky *položka faktury* tvořen kombinací sloupců *číslo faktury* a *číslo položky*.

Entitní množina *položka* se může zdát zbytečná, protože by bylo možné uvažovat vztah přímo mezi *fakturou* a *službou* s kardinalitou M:N. Tento vztah by mohl mít atributy *počet jednotek* a *sleva*, které jsou v konceptuálním schématu na obrázku 33 atributy *položky faktury*. Transformací tohoto vztahu do fyzického modelu by vznikla tabulka, jejíž primární klíč by byl tvořen kombinací sloupců *číslo faktury* a *kód služby*. Dále by tabulka obsahovala sloupce *počet jednotek* a *sleva*. Na faktuře by bylo možné vyfakturovat více různých služeb, každou vyfakturovanou službu v zadaném počtu jednotek, a bylo by i možné udělit slevu na vyfakturovanou službu. Díky primárnímu klíči by ale bylo možné jakoukoli službu na jedné faktuře vyfakturovat pouze jednou. To by se mohlo ukázat jako problematické např. v situaci, kdyby servisní středisko chtělo dát zákazníkům kupón třeba na výměnu poškozeného skla telefonu zdarma,

přičemž by kupón bylo možno uplatnit pouze na výměnu skla jednoho telefonu. Kdyby chtěl zákazník vyměnit sklo na dvou telefonech a na jednu výměnu uplatnit kupón, bylo by třeba vyfakturovat tutéž službu, ale jednou se slevou a podruhé bez slevy, což by primární klíč neumožnil. Bylo by tak třeba rozdělit zakázku a každou z výměn vyfakturovat zvlášť, nebo přepočítat slevu na slevu ve výši 50% z celkové částky za dvě provedené výměny. Rozpoznání, že součástí faktury jsou položky faktury, a že vazba mezi fakturou a službou nastává skrze položku faktury, vede k řešení, které v popisovaném případě umožní na faktuře uvést dvě položky – jednu na výměnu skla za plnou cenu, druhou pro výměnu skla se slevou ve výši 100%.

Ať už je služba operací nebo náhradním dílem, je identifikována pomocí kódu operace, který službám přiřazuje servisní středisko. U operací se jedná o zjevný způsob rozlišení operací, protože servisní středisko je zodpovědné za vymezení operací, které bude nabízet. V případě náhradních dílů vzniká náhradní díl u výrobce, přičemž výrobce může náhradnímu dílu přidělit nějaký kód či jinou formu identifikátoru. Servisní středisko by mohlo od výrobce takovýto kód převzít a použít ho pro rozlišení náhradních dílů. Úskalím takovéhoto přístupu z hlediska identifikace náhradních dílů v databázi servisního střediska je skutečnost, že není zajištěno, že dva výrobci nepřidělí různým náhradním dílům stejný kód, a také, zda bude vždy od výrobce použitelný kód k dispozici. U náhradního dílu je možné kód od výrobce evidovat jako volitelný atribut, ale pro identifikaci v rámci databáze servisního střediska bude sloužit kód služby, který náhradnímu dílu servisní středisko přidělí.[14]

5.6.1 Zvolené řešení poštovní adresy

V této úloze uvažujeme, že servisní středisko je malé a působí v České republice. Proto zde zvolený rozpad poštovní adresy na jednotlivé atributy, které ji tvoří, použité datové typy a jejich velikost vycházejí ze specifikace výměnného formátu Registru územní identifikace, adres a nemovitostí (Český úřad zeměměřičský a katastrální, 2016). Ten sice nespecifikuje adresu jako takovou, nicméně některé prvky tohoto registru, případně jejich atributy, svým významem odpovídají položkám, které tvoří poštovní adresu. Jedná se o název obce, název ulice, číslo popisné, číslo orientační a poštovní směrovací číslo. Ty tak byly použity pro reprezentaci položek poštovní adresy včetně

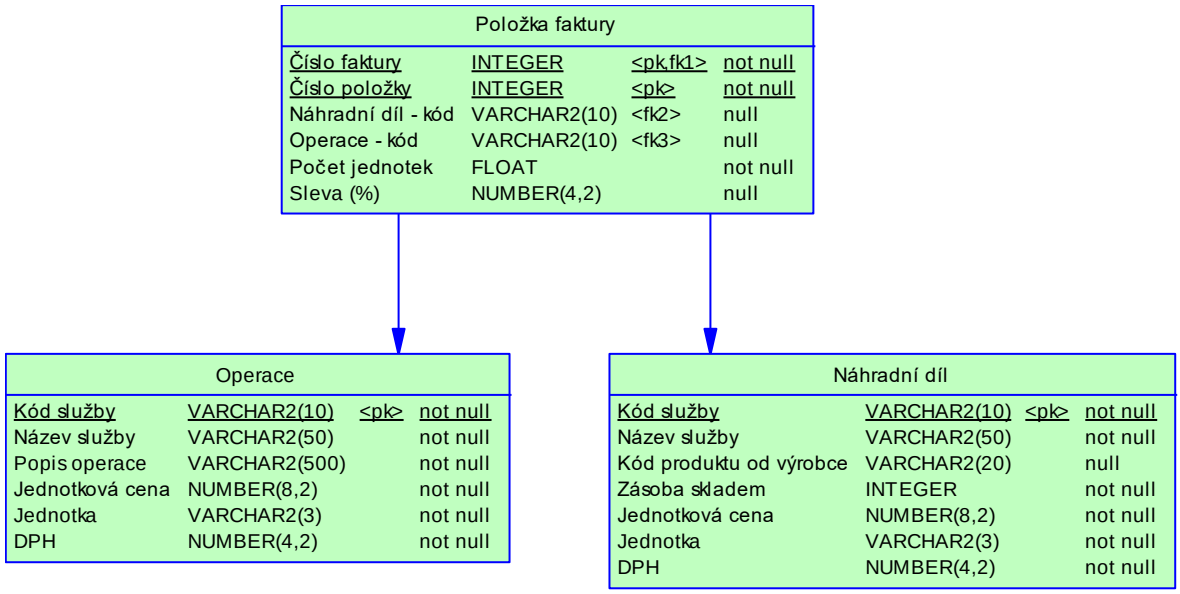
[14] V katalozích některých e-shopů se lze setkat s tím, že v katalogu je zobrazen jak identifikátor položky katalogu zvolený daným provozovatelem e-shopu, tak i kód produktu od výrobce. Zde aplikovaný přístup k identifikaci náhradních dílů je obdobný.

rozdělení čísla orientačního na část tvořenou číslicemi, a část tvořenou písmenem. Blíže o možných způsobech reprezentace poštovní adresy pojednává kapitola 12.2.

5.6.2 Zvolené řešení generalizace/specializace

Skutečnost, že servisní středisko rozlišuje dva typy služeb (operace a náhradní díly) je v konceptuálním schématu a v konceptuálním datovém modelu vyjádřena pomocí vztahu typu generalizace/specializace. Tento vztah je chápán jako úplný a výlučný, tj. nemůže existovat služba, která by nebyla ani operací, ani náhradním dílem (úplnost), přičemž současně platí, že operace nemůže být náhradním dílem, a zároveň náhradní díl nemůže být operací (výlučnost).

Ve fyzickém datovém modelu je vztah *typ služby* řešen vytvořením jedné tabulky pro nadtyp i jeho podtypy. Toto řešení umožní jednoduše zajistit, že se každá položka faktury týká buď operace, nebo náhradního dílu, a zároveň, že na každé položce faktury musí být nějaká služba vyfakturována.

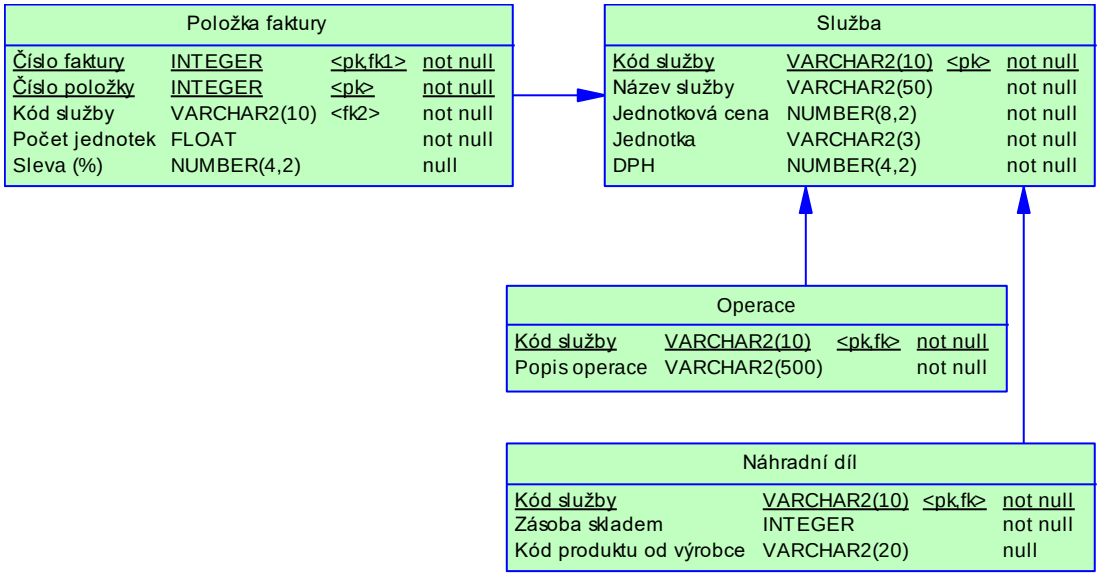


Obrázek 42: Evidence faktur – řešení služeb pomocí tabulek pro podtypy (Power Designer), zdroj: autoři

Kdyby byly vytvořeny samostatné tabulky pro operace a náhradní díly, bylo by nutné, aby v tabulce *položka faktury* byly dva cizí klíče realizující reference na fakturovanou službu – jeden odkazující na primární klíč v tabulce operací, druhý odkazující na primární klíč v tabulce služeb (viz obrázek 42). V případě tohoto řešení by bylo třeba zajistit, aby v každém záznamu v tabulce *položka faktury* byla uvedena hodnota cizího

klíče právě v jednom z těchto sloupců, protože pokud by byla uvedena hodnota v obou těchto sloupcích, znamenalo by to, že na jedné položce faktury je současně fakturována operace i náhradní díl. V souladu s konceptuálním datovým modelem by nebyla přípustná ani situace, kdyby v nějakém záznamu oba tyto sloupce obsahovaly prázdnou hodnotu, protože by na pak na takovéto položce faktury nebyla vyfakturována služba žádná.

Bylo by také možné uvažovat vytvoření samostatných tabulek pro nadtyp i oba podtypy, tj. vytvořit samostatné tabulky pro služby, operace i náhradní díly (viz obrázek 43). Vzhledem k úplnosti a výlučnosti vztahu *typ služby* by ale bylo nutno zajistit, aby ke každému záznamu v tabulce služeb v závislosti na typu dané služby existoval právě jeden záznam v tabulce operací, nebo v tabulce služeb, nikoli ale v obou těchto tabulkách zároveň. Záznam v tabulce služeb bez korespondujícího záznamu v tabulce operací nebo náhradních dílů by představoval službu s neurčeným typem. Pokud by k jednomu záznamu o službě existoval jak záznam v tabulce operací, tak záznam v tabulce služeb, znamenalo by to v rozporu se zadáním, že služba je operací i náhradním dílem současně.



Obrázek 43: Evidence faktur – řešení služeb pomocí tabulek pro nadtyp i podtypy (Power Designer), zdroj: autoři

S výše popisovanými problémy by bylo možné se vypořádat, protože ale není třeba řešit vztahy, ve kterých by vystupovaly operace nebo náhradní díly jakožto samostatné

entitní množiny, vytvoření jedné tabulky pro nadtyp i podtypy se jeví jako nejvhodnější řešení, protože umožňuje přímo řešit požadavky vyplývající za zadání – v tabulce *položka faktury* bude jeden cizí klíč odkazující na primární klíč v tabulce *služba*; pomocí omezení typu `NOT NULL` bude možné vynutit, aby na každé položce faktury byla vyfakturována nějaká služba; a protože záznamy o všech službách obojího typu budou obsaženy pouze v jedné tabulce, je snadné zajistit, aby služba byla buď operací, nebo náhradním dílem, a zároveň vyloučit vznik záznamů o službách bez určeného typu.

Protože při zvoleném řešení generalizace/specializace budou záznamy o obou typech služeb zapisovány do jedné tabulky, byl přidán sloupec *typ služby* pro rozlišení, jaký typ služby každý ze záznamů popisuje. Pro sloupec *typ služby* bylo stanoveno omezení, kterým je zajištěno, že sloupec bude obsahovat jen a pouze hodnoty „o“ (operace), nebo „n“ (náhradní díl).

Důsledkem zvoleného řešení generalizace/specializace je také skutečnost, že v tabulce *služba* musí být jak sloupec *popis operace*, tak sloupce *zásoba sladem* a *kód produktu od výrobce*. V každém záznamu ale smí být vyplněny jen relevantní sloupce v závislosti na tom, jaký typ služby záznam popisuje, resp. na hodnotě sloupce *typ služby*. Toto je řešeno pomocí omezení typu `CHECK`.

Protože bude třeba poskytnout přehled nabízených operací a náhradních dílů samostatně pro jednotlivé typy, a také proto, že je třeba umožnit založení záznamu o nové operaci nebo o novém náhradním dílu, jsou součástí fyzického datového modelu pohledy *operace* a *náhradní díly*. Každý z pohledů obsahuje pouze sloupce relevantní pro daný typ služby. V případě variant řešení na obrázcích 36-39 budou oba pohledy aktualizovatelné[15], přičemž budou vytvořeny s využitím klauzule `WITH CHECK OPTION`, aby skrze daný pohled bylo možné vložit pouze záznam o službě příslušného typu, resp. aby při aktualizaci záznamu skrze pohled nebylo možné změnit jeho typ služby. Tj. aby např. skrze pohled *operace* bylo možné vložit pouze záznam o nově nabízené operaci a nikoli o náhradním dílu.

[15] Viz (Oracle, 2013a, str. 17-19 – 17-20).

5.7 Příkazy pro vytvoření relačních tabulek a dalších objektů

5.7.1 Varianta 1 bez historie změn údajů o zákaznících a službách

Výpis 2: Evidence faktur – SQL skript, varianta 1, zdroj: autoři

```
/*============================================================*/
/* Tabulka: FAKTURA                                           */
/*============================================================*/
CREATE TABLE FAKTURA  (
   CISLO_FAKTURY        INTEGER                        NOT NULL,
   CISLO_ZAKAZNIKA      INTEGER,
   DATUM_VYSTAVENI      DATE                           NOT NULL,
   DATUM_PLNENI         DATE                           NOT NULL,
   DATUM_SPLATNOSTI     DATE                           NOT NULL,
   CONSTRAINT PK_FAKTURA PRIMARY KEY (CISLO_FAKTURY),
   CONSTRAINT CKT_FAKTURA
     CHECK ((DATUM_SPLATNOSTI >= DATUM_VYSTAVENI)
            AND (DATUM_SPLATNOSTI >= DATUM_PLNENI))
);

/*============================================================*/
/* Index: VYSTAVENA_FK                                        */
/*============================================================*/
CREATE INDEX VYSTAVENA_FK ON FAKTURA (
   CISLO_ZAKAZNIKA ASC
);

/*============================================================*/
/* Tabulka: POLOZKA_FAKTURY                                   */
/*============================================================*/
CREATE TABLE POLOZKA_FAKTURY  (
   CISLO_FAKTURY        INTEGER                        NOT NULL,
   CISLO_POLOZKY        INTEGER                        NOT NULL,
   KOD_SLUZBY           VARCHAR2(10)                   NOT NULL,
   PCT_JEDNOTEK         FLOAT                          NOT NULL,
   SLEVA                NUMBER(4,2)
      CONSTRAINT CKC_POLOZKA_SLEVA
        CHECK (SLEVA IS NULL OR (SLEVA BETWEEN 0 AND 100)),
   CONSTRAINT PK_POLOZKA_FAKTURY
     PRIMARY KEY (CISLO_FAKTURY, CISLO_POLOZKY)
);

```

```
38  /*==============================================================*/
39  /* Tabulka: SLUZBA                                              */
40  /*==============================================================*/
41  CREATE TABLE SLUZBA  (
42     KOD_SLUZBY           VARCHAR2(10)                  NOT NULL,
43     TYP_SLUZBY           VARCHAR2(1)                   NOT NULL
44        CONSTRAINT CKC_TYP_SLUZBY_SLUZBA
45          CHECK (TYP_SLUZBY IN ('o','n')),
46     NAZEV_SLUZBY         VARCHAR2(50)                  NOT NULL,
47     JEDNOTKOVA_CENA      NUMBER(8,2)                   NOT NULL,
48     JEDNOTKA             VARCHAR2(3)                   NOT NULL,
49     DPH                  NUMBER(4,2)                   NOT NULL,
50     ZASOBA_SKLADEM       INTEGER,
51     POPIS_OPERACE        VARCHAR2(500),
52     KOD_VYROBCE          VARCHAR2(20),
53     CONSTRAINT PK_SLUZBA PRIMARY KEY (KOD_SLUZBY),
54     CONSTRAINT CKT_SLUZBA
55       CHECK ((TYP_SLUZBY = 'o'
56               AND POPIS_OPERACE IS NOT NULL
57               AND ZASOBA_SKLADEM IS NULL
58               AND KOD_VYROBCE IS NULL)
59              OR (TYP_SLUZBY = 'n'
60                  AND POPIS_OPERACE IS NULL
61                  AND ZASOBA_SKLADEM IS NOT NULL))
62  );
63
64  /*==============================================================*/
65  /* Tabulka: ZAKAZNIK                                            */
66  /*==============================================================*/
67  CREATE TABLE ZAKAZNIK  (
68     CISLO_ZAKAZNIKA      INTEGER                       NOT NULL,
69     ICO                  VARCHAR2(8),
70     OZNACENI_ZAK         VARCHAR2(100)                 NOT NULL,
71     TEL                  VARCHAR2(16),
72     ULICE                VARCHAR2(48)                  NOT NULL,
73     C_POPISNE            NUMBER(5,0)                   NOT NULL,
74     C_ORIENTACNI         NUMBER(4,0),
75     C_ORIENTACNI_P       VARCHAR2(1),
76     OBEC                 VARCHAR2(48)                  NOT NULL,
77     PSC                  NUMBER(5,0)                   NOT NULL,
78     CONSTRAINT PK_ZAKAZNIK PRIMARY KEY (CISLO_ZAKAZNIKA),
79     CONSTRAINT AK_ZAKAZNIK_IC UNIQUE (ICO),
80     CONSTRAINT CHK_ZAKAZNIK_TELEFON
81       CHECK (REGEXP_LIKE (TEL, '^[+]?[0-9]{1,15}$'))
82  );
83
```

```
84  /*==============================================================*/
85  /* View: NAHRADNI_DIL                                           */
86  /*==============================================================*/
87  CREATE OR REPLACE VIEW NAHRADNI_DIL AS
88  SELECT
89     KOD_SLUZBY, TYP_SLUZBY, NAZEV_SLUZBY, JEDNOTKOVA_CENA,
90     JEDNOTKA, DPH, ZASOBA_SKLADEM, KOD_VYROBCE
91  FROM
92     SLUZBA
93  WHERE TYP_SLUZBY = 'n'
94  WITH CHECK OPTION;
95
96  /*==============================================================*/
97  /* View: OPERACE                                                */
98  /*==============================================================*/
99  CREATE OR REPLACE VIEW OPERACE AS
100 SELECT
101    KOD_SLUZBY, TYP_SLUZBY, NAZEV_SLUZBY, JEDNOTKOVA_CENA,
102    JEDNOTKA, DPH, POPIS_OPERACE
103 FROM
104    SLUZBA
105 WHERE TYP_SLUZBY = 'o'
106 WITH CHECK OPTION;
107
108 ALTER TABLE FAKTURA
109    ADD CONSTRAINT FK_FAKTURA_VYSTAVENA_ZAKAZNIK FOREIGN KEY (CISLO_ZAKAZNIKA)
110       REFERENCES ZAKAZNIK (CISLO_ZAKAZNIKA);
111
112 ALTER TABLE POLOZKA_FAKTURY
113    ADD CONSTRAINT FK_POLOZKA_FAKTUROVA_SLUZBA FOREIGN KEY (KOD_SLUZBY)
114       REFERENCES SLUZBA (KOD_SLUZBY);
115
116 ALTER TABLE POLOZKA_FAKTURY
117    ADD CONSTRAINT FK_POLOZKA_OBSAHUJE_FAKTURA FOREIGN KEY (CISLO_FAKTURY)
118       REFERENCES FAKTURA (CISLO_FAKTURY)
119       ON DELETE CASCADE;
```

5.7.2 Varianta 2 s ukládáním historických údajů na faktuře

Výpis 3: Evidence faktur – SQL skript, varianta 2, zdroj: autoři

```
/*==============================================================*/
/* Tabulka: FAKTURA                                             */
/*==============================================================*/
CREATE TABLE FAKTURA  (
   CISLO_FAKTURY        INTEGER                        NOT NULL,
   CISLO_ZAKAZNIKA      INTEGER,
   DATUM_VYSTAVENI      DATE                           NOT NULL,
   DATUM_PLNENI         DATE                           NOT NULL,
   DATUM_SPLATNOSTI     DATE                           NOT NULL,
   OZNACENI_ZAK         VARCHAR2(100),
   ICO                  VARCHAR2(8),
   ULICE                VARCHAR2(48),
   C_POPISNE            NUMBER(5,0),
   C_ORIENTACNI         NUMBER(4,0),
   C_ORIENTACNI_P       VARCHAR2(1),
   OBEC                 VARCHAR2(48),
   PSC                  NUMBER(5,0),
   CONSTRAINT PK_FAKTURA PRIMARY KEY (CISLO_FAKTURY),
   CONSTRAINT CKT_FAKTURA
     CHECK ((DATUM_SPLATNOSTI >= DATUM_VYSTAVENI)
             AND (DATUM_SPLATNOSTI >= DATUM_PLNENI))
);

/*==============================================================*/
/* Index: VYSTAVENA_FK                                          */
/*==============================================================*/
CREATE INDEX VYSTAVENA_FK ON FAKTURA (
   CISLO_ZAKAZNIKA ASC
);

```

```
31  /*==============================================================*/
32  /* Tabulka: POLOZKA_FAKTURY                                     */
33  /*==============================================================*/
34  CREATE TABLE POLOZKA_FAKTURY  (
35     CISLO_FAKTURY        INTEGER                        NOT NULL,
36     CISLO_POLOZKY        INTEGER                        NOT NULL,
37     KOD_SLUZBY           VARCHAR2(10)                   NOT NULL,
38     NAZEV_SLUZBY         VARCHAR2(50)                   NOT NULL,
39     PCT_JEDNOTEK         FLOAT                          NOT NULL,
40     JEDNOTKOVA_CENA      NUMBER(8,2)                    NOT NULL,
41     DPH                  NUMBER(4,2)                    NOT NULL,
42     SLEVA                NUMBER(4,2)
43        CONSTRAINT CKC_POLOZKA_SLEVA
44          CHECK (SLEVA IS NULL OR (SLEVA BETWEEN 0 AND 100)),
45     CONSTRAINT PK_POLOZKA_FAKTURY
46       PRIMARY KEY (CISLO_FAKTURY, CISLO_POLOZKY)
47  );
48
49  /*==============================================================*/
50  /* Tabulka: SLUZBA                                              */
51  /*==============================================================*/
52  CREATE TABLE SLUZBA  (
53     KOD_SLUZBY           VARCHAR2(10)                   NOT NULL,
54     TYP_SLUZBY           VARCHAR2(1)                    NOT NULL
55        CONSTRAINT CKC_TYP_SLUZBY_SLUZBA
56          CHECK (TYP_SLUZBY IN ('o','n')),
57     NAZEV_SLUZBY         VARCHAR2(50)                   NOT NULL,
58     JEDNOTKOVA_CENA      NUMBER(8,2)                    NOT NULL,
59     JEDNOTKA             VARCHAR2(3)                    NOT NULL,
60     DPH                  NUMBER(4,2)                    NOT NULL,
61     ZASOBA_SKLADEM       INTEGER,
62     POPIS_OPERACE        VARCHAR2(500),
63     KOD_VYROBCE          VARCHAR2(20),
64     CONSTRAINT PK_SLUZBA PRIMARY KEY (KOD_SLUZBY),
65     CONSTRAINT CKT_SLUZBA
66       CHECK ((TYP_SLUZBY = 'o'
67               AND POPIS_OPERACE IS NOT NULL
68               AND ZASOBA_SKLADEM IS NULL
69               AND KOD_VYROBCE IS NULL)
70              OR (TYP_SLUZBY = 'n'
71                  AND POPIS_OPERACE IS NULL
72                  AND ZASOBA_SKLADEM IS NOT NULL))
73  );
74
```

```
75    /*==============================================================*/
76    /* Tabulka: ZAKAZNIK                                            */
77    /*==============================================================*/
78    CREATE TABLE ZAKAZNIK  (
79       CISLO_ZAKAZNIKA      INTEGER                       NOT NULL,
80       ICO                  VARCHAR2(8),
81       OZNACENI_ZAK         VARCHAR2(100)                 NOT NULL,
82       TEL                  VARCHAR2(16),
83       ULICE                VARCHAR2(48)                  NOT NULL,
84       C_POPISNE            NUMBER(5,0)                   NOT NULL,
85       C_ORIENTACNI         NUMBER(4,0),
86       C_ORIENTACNI_P       VARCHAR2(1),
87       OBEC                 VARCHAR2(48)                  NOT NULL,
88       PSC                  NUMBER(5,0)                   NOT NULL,
89       CONSTRAINT PK_ZAKAZNIK PRIMARY KEY (CISLO_ZAKAZNIKA),
90       CONSTRAINT AK_ZAKAZNIK_IC UNIQUE (ICO),
91       CONSTRAINT CHK_ZAKAZNIK_TELEFON
92         CHECK (REGEXP_LIKE (TEL, '^[+]?[0-9]{1,15}$'))
93    );
94
95    /*==============================================================*/
96    /* View: NAHRADNI_DIL                                           */
97    /*==============================================================*/
98    CREATE OR REPLACE VIEW NAHRADNI_DIL AS
99    SELECT
100      KOD_SLUZBY, TYP_SLUZBY, NAZEV_SLUZBY, JEDNOTKOVA_CENA,
101      JEDNOTKA, DPH, ZASOBA_SKLADEM, KOD_VYROBCE
102   FROM
103      SLUZBA
104   WHERE TYP_SLUZBY = 'n'
105   WITH CHECK OPTION;
106
107   /*==============================================================*/
108   /* View: OPERACE                                                */
109   /*==============================================================*/
110   CREATE OR REPLACE VIEW OPERACE AS
111   SELECT
112      KOD_SLUZBY, TYP_SLUZBY, NAZEV_SLUZBY, JEDNOTKOVA_CENA,
113      JEDNOTKA, DPH, POPIS_OPERACE
114   FROM
115      SLUZBA
116   WHERE TYP_SLUZBY = 'o'
117   WITH CHECK OPTION;
118
119   ALTER TABLE FAKTURA
120      ADD CONSTRAINT FK_FAKTURA_VYSTAVENA_ZAKAZNIK FOREIGN KEY (CISLO_ZAKAZNIKA)
121         REFERENCES ZAKAZNIK (CISLO_ZAKAZNIKA);
122
```

```
123   ALTER TABLE POLOZKA_FAKTURY
124      ADD CONSTRAINT FK_POLOZKA_FAKTUROVA_SLUZBA FOREIGN KEY (KOD_SLUZBY)
125         REFERENCES SLUZBA (KOD_SLUZBY);
126
127   ALTER TABLE POLOZKA_FAKTURY
128      ADD CONSTRAINT FK_POLOZKA_OBSAHUJE_FAKTURA FOREIGN KEY (CISLO_FAKTURY)
129         REFERENCES FAKTURA (CISLO_FAKTURY)
130         ON DELETE CASCADE;
```

5.7.3 Varianta 3 s ukládáním historických údajů mimo fakturu

Výpis 4: Evidence faktur – SQL skript, varianta 3, zdroj: autoři

```
1   /*==============================================================*/
2   /* Tabulka: FAKTURA                                             */
3   /*==============================================================*/
4   CREATE TABLE FAKTURA  (
5      CISLO_FAKTURY        INTEGER                       NOT NULL,
6      V_ZAK_INFO           INTEGER,
7      CISLO_ZAKAZNIKA      INTEGER,
8      DATUM_VYSTAVENI      DATE                          NOT NULL,
9      DATUM_PLNENI         DATE                          NOT NULL,
10     DATUM_SPLATNOSTI     DATE                          NOT NULL,
11     CONSTRAINT PK_FAKTURA PRIMARY KEY (CISLO_FAKTURY),
12     CONSTRAINT CKT_FAKTURA
13       CHECK ((DATUM_SPLATNOSTI >= DATUM_VYSTAVENI)
14               AND (DATUM_SPLATNOSTI >= DATUM_PLNENI))
15  );
16
17  /*==============================================================*/
18  /* Index: VYSTAVENA_FK                                          */
19  /*==============================================================*/
20  CREATE INDEX VYSTAVENA_FK ON FAKTURA (
21     V_ZAK_INFO ASC,
22     CISLO_ZAKAZNIKA ASC
23  );
24
```

```
/*==============================================================*/
/* Tabulka: POLOZKA_FAKTURY                                     */
/*==============================================================*/
CREATE TABLE POLOZKA_FAKTURY  (
   CISLO_FAKTURY        INTEGER                        NOT NULL,
   CISLO_POLOZKY        INTEGER                        NOT NULL,
   KOD_SLUZBY           VARCHAR2(10)                   NOT NULL,
   V_SL_INFO            INTEGER                        NOT NULL,
   PCT_JEDNOTEK         FLOAT                          NOT NULL,
   SLEVA                NUMBER(4,2)
      CONSTRAINT CKC_POLOZKA_SLEVA
        CHECK (SLEVA IS NULL OR (SLEVA BETWEEN 0 AND 100)),
   CONSTRAINT PK_POLOZKA_FAKTURY
     PRIMARY KEY (CISLO_FAKTURY, CISLO_POLOZKY)
);

/*==============================================================*/
/* Tabulka: SLUZBA_HISTORIE                                     */
/*==============================================================*/
CREATE TABLE SLUZBA_HISTORIE  (
   KOD_SLUZBY           VARCHAR2(10)                   NOT NULL,
   V_SL_INFO            INTEGER                        NOT NULL,
   TYP_SLUZBY           VARCHAR2(1)                    NOT NULL
      CONSTRAINT CKC_TYP_SLUZBY_SL_HIST CHECK (TYP_SLUZBY IN ('o','n')),
   NAZEV_SLUZBY         VARCHAR2(50)                   NOT NULL,
   JEDNOTKOVA_CENA      NUMBER(8,2)                    NOT NULL,
   JEDNOTKA             VARCHAR2(3)                    NOT NULL,
   DPH                  NUMBER(4,2)                    NOT NULL,
   ZASOBA_SKLADEM       INTEGER,
   POPIS_OPERACE        VARCHAR2(500),
   PLATNY_OD            DATE         DEFAULT SYSDATE   NOT NULL,
   PLATNY_DO            DATE,
   AKTUALNI             VARCHAR2(1)                    NOT NULL
      CONSTRAINT CKC_AKTUALNI_SL_HIST CHECK (AKTUALNI IN ('A','N')),
   KOD_VYROBCE          VARCHAR2(20),
   CONSTRAINT PK_SLUZBA_HISTORIE PRIMARY KEY (KOD_SLUZBY, V_SL_INFO),
   CONSTRAINT CKT_SL_HIST
     CHECK ((PLATNY_DO IS NULL) OR (PLATNY_OD < PLATNY_DO)),
   CONSTRAINT CK_SLUZBA_TYPY
     CHECK ((TYP_SLUZBY = 'o'
             AND POPIS_OPERACE IS NOT NULL
             AND ZASOBA_SKLADEM IS NULL
             AND KOD_VYROBCE IS NULL)
            OR (TYP_SLUZBY = 'n'
                AND POPIS_OPERACE IS NULL
                AND ZASOBA_SKLADEM IS NOT NULL))
);

```

```
73    /*==============================================================*/
74    /* Tabulka: ZAKAZNIK_HISTORIE                                   */
75    /*==============================================================*/
76    CREATE TABLE ZAKAZNIK_HISTORIE  (
77       CISLO_ZAKAZNIKA      INTEGER                       NOT NULL,
78       V_ZAK_INFO           INTEGER                       NOT NULL,
79       ICO                  VARCHAR2(8),
80       OZNACENI_ZAK         VARCHAR2(100)                 NOT NULL,
81       TEL                  VARCHAR2(16),
82       ULICE                VARCHAR2(48)                  NOT NULL,
83       C_POPISNE            NUMBER(5,0)                   NOT NULL,
84       C_ORIENTACNI         NUMBER(4,0),
85       C_ORIENTACNI_P       VARCHAR2(1),
86       OBEC                 VARCHAR2(48)                  NOT NULL,
87       PSC                  NUMBER(5,0)                   NOT NULL,
88       PLATNY_OD            DATE       DEFAULT SYSDATE    NOT NULL,
89       PLATNY_DO            DATE,
90       AKTUALNI             VARCHAR2(1)                   NOT NULL
91          CONSTRAINT CKC_AKTUALNI_ZAKAZNIK
92            CHECK (AKTUALNI IN ('A','N')),
93       CONSTRAINT PK_ZAKAZNIK_HISTORIE
94         PRIMARY KEY (V_ZAK_INFO, CISLO_ZAKAZNIKA),
95       CONSTRAINT CKT_ZAKAZNIK_HISTORIE
96         CHECK ((PLATNY_DO IS NULL) OR (PLATNY_OD < PLATNY_DO)),
97       CONSTRAINT CHK_ZAK_HIST_TELEFON
98         CHECK (REGEXP_LIKE (TEL, '^[+]?[0-9]{1,15}$'))
99    );
100
101   /*==============================================================*/
102   /* Index: ZAKAZNIK_AK                                           */
103   /*==============================================================*/
104   CREATE UNIQUE INDEX ZAKAZNIK_AK ON ZAKAZNIK_HISTORIE (
105      ICO ASC
106   );
107
108   ALTER TABLE FAKTURA
109      ADD CONSTRAINT FK_FAKTURA_VYSTAVENA_ZAKAZNIK FOREIGN KEY (V_ZAK_INFO,
      CISLO_ZAKAZNIKA)
110         REFERENCES ZAKAZNIK_HISTORIE (V_ZAK_INFO, CISLO_ZAKAZNIKA);
111
112   ALTER TABLE POLOZKA_FAKTURY
113      ADD CONSTRAINT FK_POLOZKA_FAKTUROVA_SLUZBA_H FOREIGN KEY (KOD_SLUZBY,
      V_SL_INFO)
114         REFERENCES SLUZBA_HISTORIE (KOD_SLUZBY, V_SL_INFO);
115
```

```
116  ALTER TABLE POLOZKA_FAKTURY
117     ADD CONSTRAINT FK_POLOZKA_OBSAHUJE_FAKTURA FOREIGN KEY (CISLO_FAKTURY)
118        REFERENCES FAKTURA (CISLO_FAKTURY)
119        ON DELETE CASCADE;
120
121  /*==============================================================*/
122  /* Materializovaný pohled a související log pro zákazníka       */
123  /*==============================================================*/
124  CREATE MATERIALIZED VIEW LOG ON ZAKAZNIK_HISTORIE
125    WITH ROWID, SEQUENCE (AKTUALNI)
126    INCLUDING NEW VALUES;
127
128  CREATE MATERIALIZED VIEW ZAKAZNIK
129    REFRESH FAST ON COMMIT WITH ROWID AS
130    SELECT ZAKAZNIK_HISTORIE.CISLO_ZAKAZNIKA,
131   ZAKAZNIK_HISTORIE.ICO,
132   ZAKAZNIK_HISTORIE.OZNACENI_ZAK,
133   ZAKAZNIK_HISTORIE.TEL,
134   ZAKAZNIK_HISTORIE.ULICE,
135   ZAKAZNIK_HISTORIE.C_POPISNE,
136   ZAKAZNIK_HISTORIE.C_ORIENTACNI,
137   ZAKAZNIK_HISTORIE.C_ORIENTACNI_P,
138   ZAKAZNIK_HISTORIE.OBEC,
139   ZAKAZNIK_HISTORIE.PSC
140    FROM ZAKAZNIK_HISTORIE
141    WHERE ZAKAZNIK_HISTORIE.AKTUALNI = 'A';
142
143  ALTER MATERIALIZED VIEW ZAKAZNIK
144    ADD CONSTRAINT PK_ZAKAZNIK PRIMARY KEY (CISLO_ZAKAZNIKA);
145
146  /*==============================================================*/
147  /* Materializovaný pohled a související log pro operace         */
148  /*==============================================================*/
149  CREATE MATERIALIZED VIEW LOG ON SLUZBA_HISTORIE
150    WITH ROWID, SEQUENCE (AKTUALNI)
151    INCLUDING NEW VALUES;
152
153  CREATE MATERIALIZED VIEW OPERACE
154    REFRESH FAST ON COMMIT WITH ROWID AS
155    SELECT SLUZBA_HISTORIE.KOD_SLUZBY,
156   SLUZBA_HISTORIE.TYP_SLUZBY,
157   SLUZBA_HISTORIE.NAZEV_SLUZBY,
158   SLUZBA_HISTORIE.JEDNOTKOVA_CENA,
159   SLUZBA_HISTORIE.JEDNOTKA,
160   SLUZBA_HISTORIE.DPH,
161   SLUZBA_HISTORIE.POPIS_OPERACE
162    FROM SLUZBA_HISTORIE
163    WHERE TYP_SLUZBY = 'o' AND AKTUALNI = 'A';
```

```
164
165   ALTER MATERIALIZED VIEW OPERACE
166     ADD CONSTRAINT PK_OPERACE PRIMARY KEY (KOD_SLUZBY);
167
168   /*=============================================================*/
169   /* Materializovaný pohled pro náhradní díly                    */
170   /*=============================================================*/
171   CREATE MATERIALIZED VIEW NAHRADNI_DIL
172     REFRESH FAST ON COMMIT WITH ROWID AS
173     SELECT SLUZBA_HISTORIE.KOD_SLUZBY,
174    SLUZBA_HISTORIE.TYP_SLUZBY,
175    SLUZBA_HISTORIE.NAZEV_SLUZBY,
176    SLUZBA_HISTORIE.JEDNOTKOVA_CENA,
177    SLUZBA_HISTORIE.JEDNOTKA,
178    SLUZBA_HISTORIE.DPH,
179    SLUZBA_HISTORIE.ZASOBA_SKLADEM,
180        SLUZBA_HISTORIE.KOD_VYROBCE
181     FROM SLUZBA_HISTORIE
182     WHERE TYP_SLUZBY = 'n' AND AKTUALNI = 'A';
183
184   ALTER MATERIALIZED VIEW NAHRADNI_DIL
185     ADD CONSTRAINT PK_NAHRADNI_DIL PRIMARY KEY (KOD_SLUZBY);
```

6 Evidence daňových poplatníků

6.1 Účel úlohy

Tato úloha ilustruje zejména následující problémy/jevy:

- N-ární vztahy a jejich řešení na konceptuální a fyzické úrovni.
- Vztah mezi jednou a toutéž entitní množinou.

6.2 Popis zvolené výseče světa – zadání

Pro potřeby Generálního finančního ředitelství (GFŘ) je nutno evidovat poplatníky a jim vznikající povinnosti platby jednotlivých druhů daní u jednotlivých finančních úřadů (FÚ). Obecně pojem daň v českém právním řádu vymezuje zákon č. 280/2009 Sb., daňový řád, přičemž podle § 2, odst. 3 tohoto zákona představují daně kromě takto přímo nazvaných peněžitých plnění např. také cla nebo poplatky. Cílem této úlohy není návrh databáze komplexního systému pro správu daní. Při řešení úlohy budeme primárně uvažovat daň z příjmů fyzických a právnických osob a daň z nemovitosti. Oproti zákonné úpravě je tedy v této úloze správa daní zjednodušená a neodpovídá ve všech ohledech platné právní úpravě. V úloze se také zabýváme pouze ukládáním záznamů o stanovených povinnostech platby daní; nezabýváme se ukládáním záznamů o úhradách stanovených povinností.

Daňovými poplatníky mohou být jak osoby fyzické, tak osoby právnické. U fyzických osob je požadováno evidovat rodné číslo, pokud je fyzické osobě přiděleno, celé jméno (jméno, příjmení a rodné příjmení), adresu trvalého bydliště, telefon a e-mail (nepovinně). U právnických osob je třeba evidovat identifikační číslo (IČO), pokud je právnické osobě přiděleno, název, adresu sídla, telefon.

V případě daně z příjmů fyzických osob mohou fyzické osoby získat daňové zvýhodnění na vyživované děti. U fyzických osoba je tedy třeba dále evidovat jimi vyživované děti, přičemž v rámci jednoho daňového období (kalendářního roku) může na určité dítě čerpat daňové zvýhodnění pouze jedna fyzická osoba.

Jaké existují finanční úřady, upravuje zákon č. 456/2011 Sb., o Finanční správě České republiky. Pro účely této úlohy chceme o finančních úřadech evidovat jejich název, název obce, ve kterém finanční úřad sídlí, a číslo bankovního účtu – viz dokument *Čísla matrik bankovních účtů finančních úřadů včetně Specializovaného finančního úřadu* (Finanční správa, 2019a).

O jednotlivých druzích daní je třeba evidovat název druhu daně, číslo právního předpisu, kterým je příslušný druh daně upraven, a předčíslí účtu pro platbu daně (viz Finanční správa, 2019b).

Databázi je třeba navrhnout tak, aby bylo možné u poplatníka evidovat všechny jemu vzniklé povinnosti platby daní. U každé vzniklé povinnosti platby daně je nutné evidovat výši platby a datum stanovení povinnosti platby daně. Je třeba si uvědomit, že v případě daně z příjmů fyzických a právnických osob je místně příslušným finanční úřad podle jejich místa pobytu, resp. sídla, a v případě, že předmětem daně je věc nemovitá, tak místně příslušný je ten finanční úřad, v jehož obvodu územní působnosti se nemovitá věc nachází (§ 13 zákona č. 280/2009 Sb.).

V tabulce 5 je uveden ilustrativní příklad vzniklých povinností v roce 2018 v případě pana Jaroslava Nováka, který žije v Praze. Pan Novák vlastní byt v Praze a chatu v Karlovarském kraji. Jak vyplývá z uvedeného příkladu, jednomu poplatníkovi může za jedno daňové období vzniknout i vícero daňových povinností, a to i se stejným druhem daně.

Tabulka 5: Ilustrativní příklad daňových povinností pana Nováka za rok 2018, zdroj: autoři

Datum	Poplatník	Druh daně	Místně příslušný FU	Daň
28. 3. 2018	Jaroslav Novák	Daň z příjmů fyzických osob	Finanční úřad pro hlavní město Prahu	200 000 Kč
12. 5. 2018	Jaroslav Novák	Daň z nemovitých věcí	Finanční úřad pro hlavní město Prahu	500 Kč
10. 5. 2018	Jaroslav Novák	Daň z nemovitých věcí	Finanční úřad pro Karlovarský kraj	215 Kč

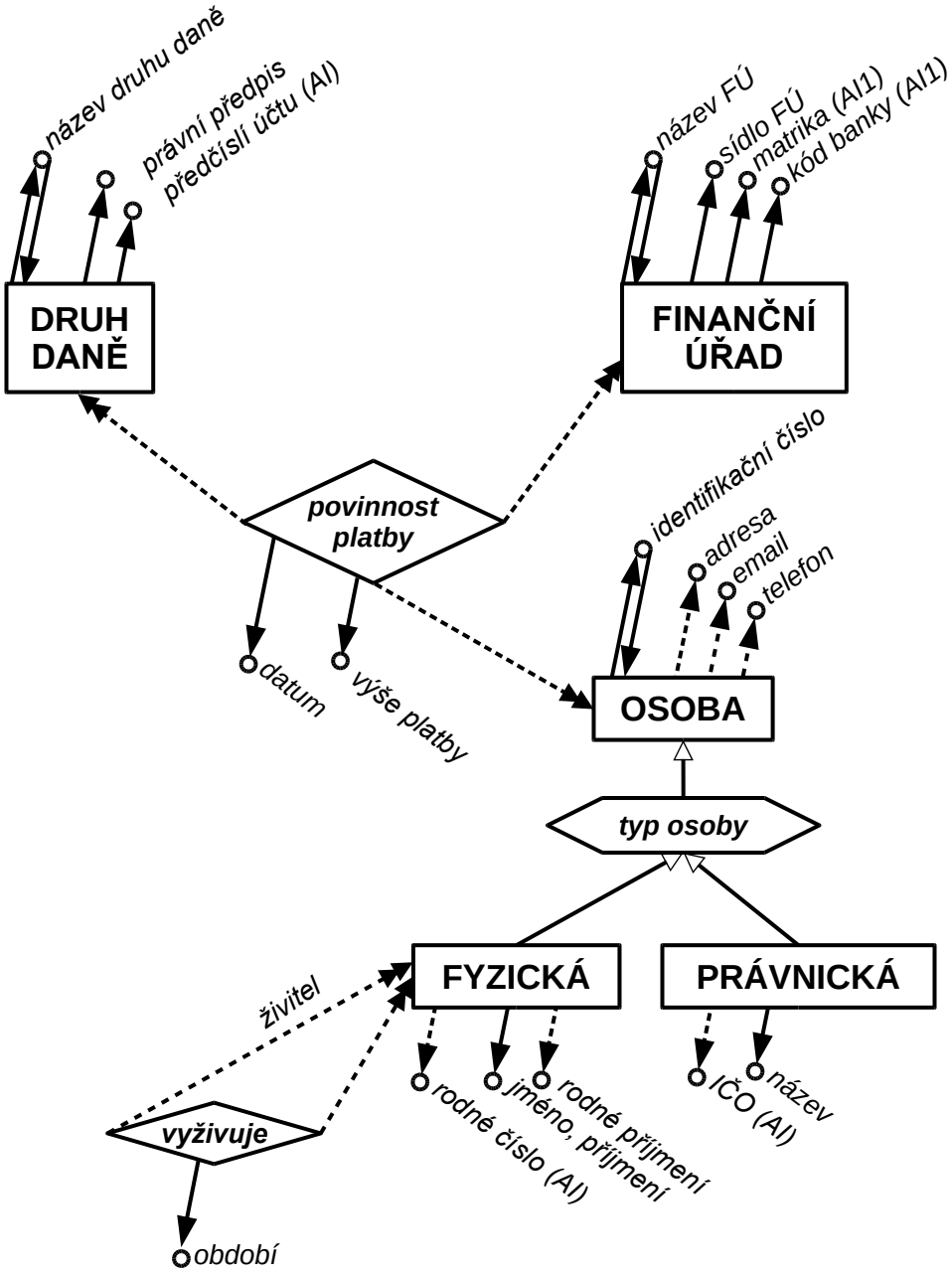
6.3 Konceptuální schéma

Na obrázcích 44 a 45 jsou uvedeny dvě varianty konceptuálního schématu. Varianta na obrázku 44 zachycuje *povinnost platby* jako ternární vztah mezi osobou, druhem daně a finančním úřadem. Ve variantě na obrázku 45 pak byl tento ternární vztah rozložen na entitní množinu *povinnost platby* a tři vztahy 1:N.

Tento rozklad může být proveden také až při tvorbě konceptuálního datového modelu, tj. konceptuální schéma by bylo modelováno s ternárním vztahem, konceptuální

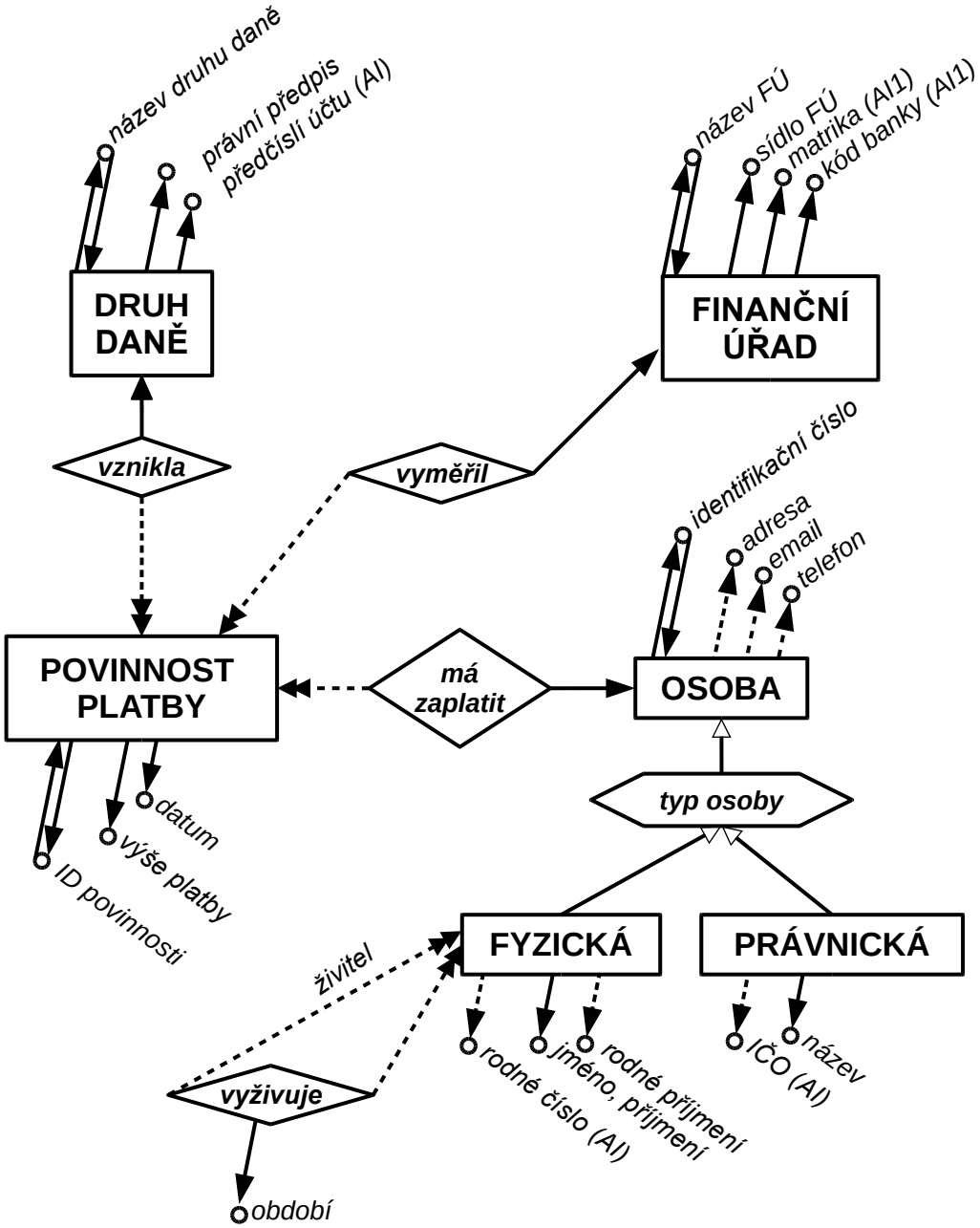
datový model pak již pouze s binárními vztahy. Je třeba poznamenat, že rozklad ternárního vztahu nemusí být na konceptuální úrovni prováděn vůbec a jeho realizace je ponechána až na návrh fyzického modelu. Záleží na tom, zda upřesnění provádí analytik, nebo až návrhář databáze či vývojář. Doporučení autorů textu je uvádět v konceptuálním schématu n-ární vztah, v konceptuálním datovém modelu pak již provést transformaci do entitní množiny, zde *povinnost platby*, aby bylo zadání pro další úroveň návrhu co nejpreciznější, a aby určení identifikátoru entitní množiny *povinnost platby* jako východiska pro určení primárního klíče tabulky, která bude realizovat *povinnost platby*, bylo prováděno (analytikem) s věcnou znalostí.

6.3.1 Modelování s využitím ternárního vztahu



Obrázek 44: Evidence daňových poplatníků – konceptuální schéma (ternární vztah), zdroj: autoři

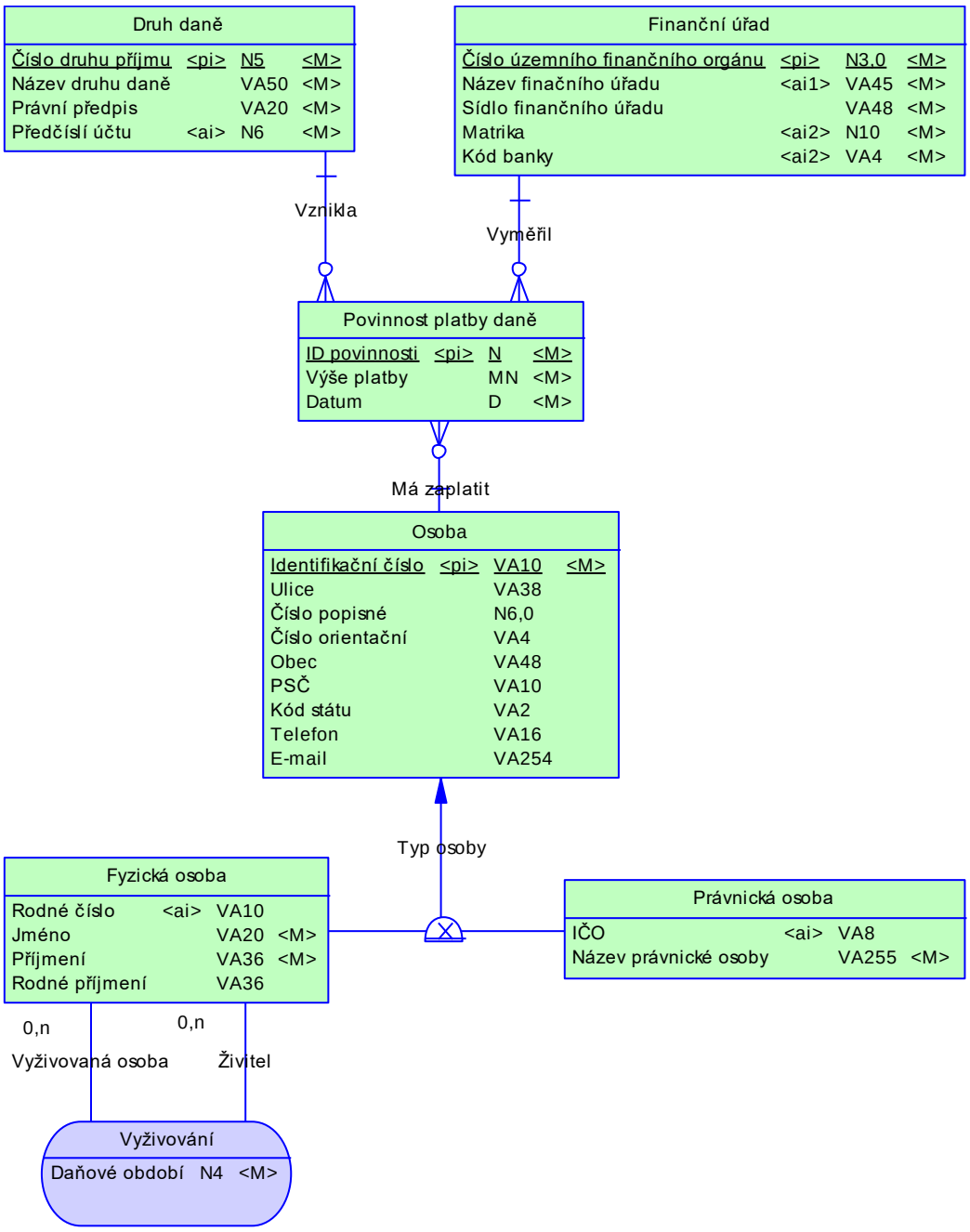
6.3.2 Rozklad ternárního vztahu



Obrázek 45: Evidence daňových poplatníků – konceptuální schéma (rozklad ternárního vztahu), zdroj: autoři

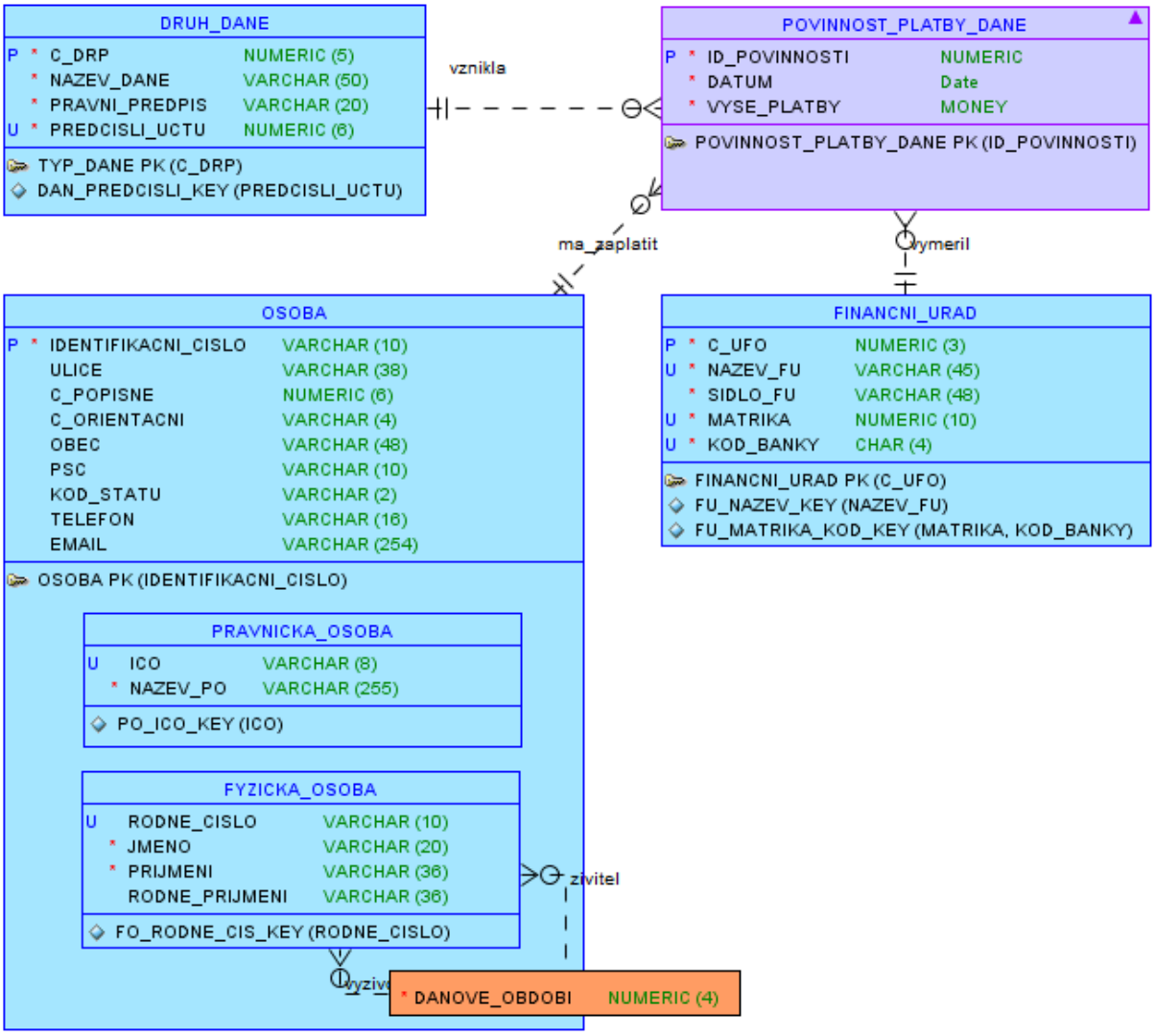
6.4 Konceptuální datový model

6.4.1 KDM v nástroji Power Designer



Obrázek 46: Evidence daňových poplatníků – konceptuální datový model (Power Designer), zdroj: autoři

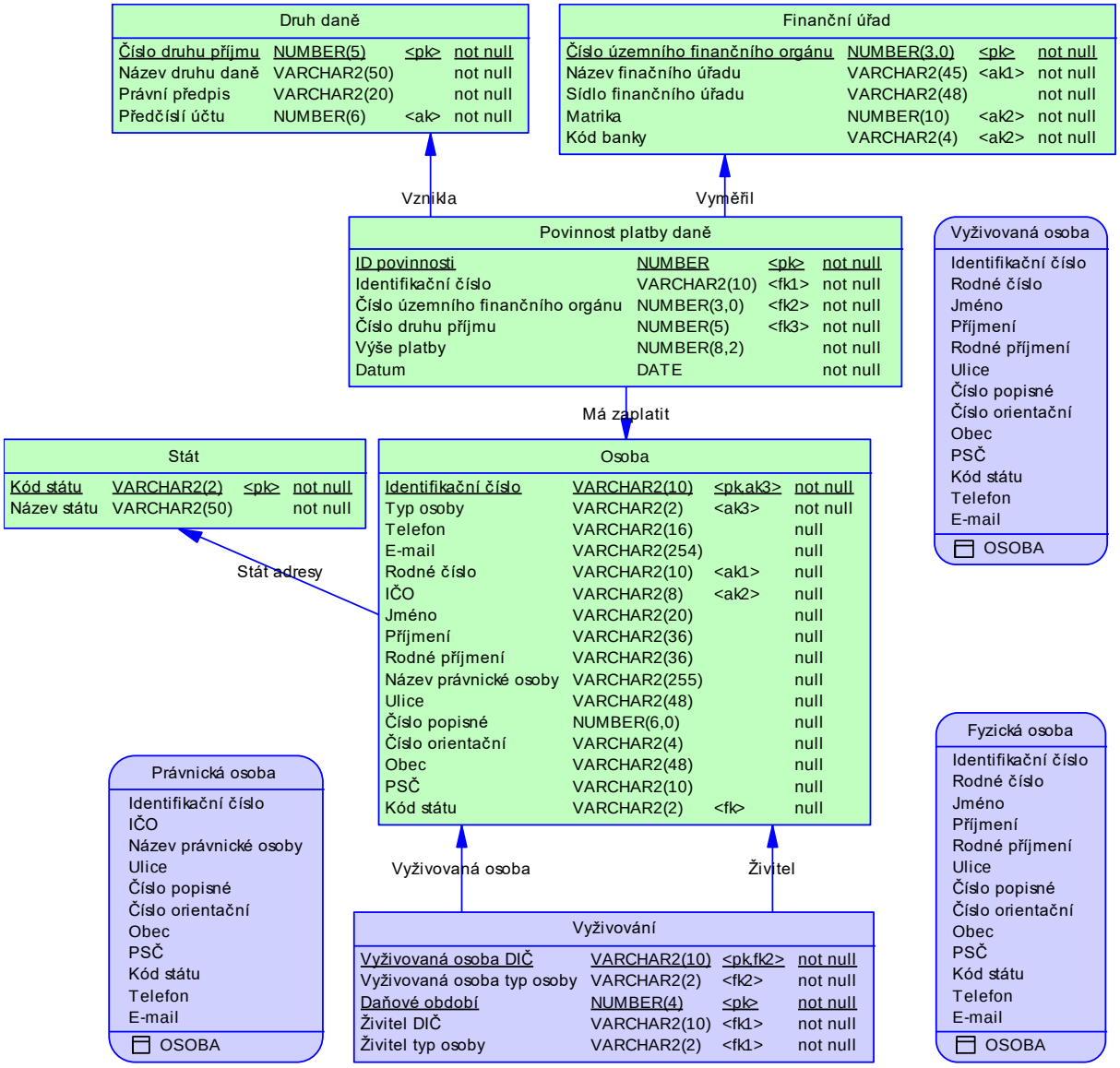
6.4.2 KDM v nástroji Data Modeler



Obrázek 47: Evidence daňových poplatníků – konceptuální datový model (Data Modeler), zdroj: autoři

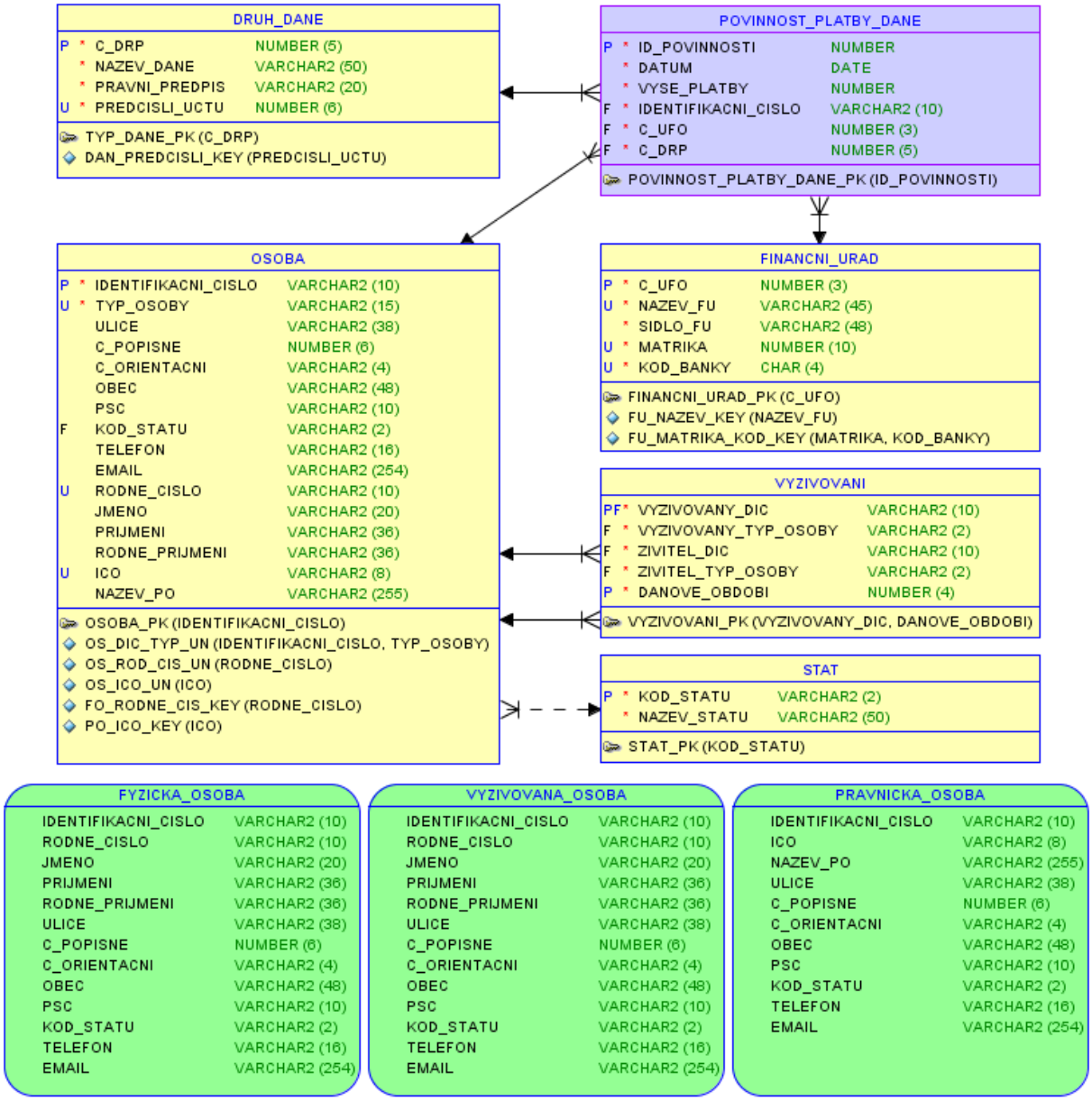
6.5 Fyzický datový model

6.5.1 FDM v nástroji Power Designer



Obrázek 48: Evidence daňových poplatníků – fyzický datový model (Power Designer), zdroj: autoři

6.5.2 FDM v nástroji Data Modeler



Obrázek 49: Evidence daňových poplatníků – fyzický datový model (Data Modeler), zdroj: autoři

6.6 Komentář k návrhu

Na základě zadání byly identifikovány entitní množiny *druh daně*, *finanční úřad*, *osoba*, přičemž členění na osoby fyzické a osoby právnické je modelováno jako vztah typu generalizace/specializace, ve kterém entitní množina *osoba* představuje nadtyp a entitní množiny *fyzická osoba* a *právnická osoba* představují podtypy.

6.6.1 Povinnost platby daně

V konceptuálním schématu na obrázku 44 je povinnost platby chápána jako ternární vztah vyjadřující, že osobě (poplatníkovi) vzniká povinnost zaplatit daň určitého druhu u určitého finančního úřadu. Tj. osoba, druh daně a finanční úřad jsou současně účastníky jednoho vztahu. Klíčovým požadavkem zadání je, že mají být evidovány povinnosti platby daně jednotlivých poplatníků. Kombinace identifikátorů poplatníka, místně příslušného finančního úřadu a druhu daně ale není dostačující pro rozlišení jednotlivých povinností, protože ta by neumožňovala, aby jeden poplatník měl u jednoho finančního úřadu více povinností k platbě stejného druhu daně. To ale ve skutečnosti může nastat, protože např. daň z nemovitosti bude poplatník tomu stejnému úřadu platit každý rok po celou dobu, kdy bude nemovitost vlastnit. Je tedy třeba se zabývat identitou povinnosti platby daně, což podporuje rozhodnutí rozložit ternární vztah uvedený v konceptuálním schématu na obrázku 45 na entitní množinu *povinnost platby* a tři vztahy 1:N.

Protože kombinace identifikátorů poplatníka, místně příslušného finančního úřadu a druhu daně není pro rozlišení povinností dostatečná, byl hledán další atribut, který by zajistil jednoznačnou identifikaci povinnosti platby daně. Dokument popisující práci v Daňové informační schránce (Generální finanční ředitelství, 2014) v sekci *Předpisy vyplývající z písemností* uvádí příklad předpisů daní, které mají stejné datum i druh příjmu. Vzhledem k tomu, že schránka je zřizována daňovým subjektům, bude stejný i poplatník daně. Ani zahrnutím data stanovení povinnosti by nevznikl vhodný primární identifikátor povinnosti platby daně, protože pokud by primární identifikátor byl tvořen kombinací identifikátorů poplatníka, místně příslušného finančního úřadu a data stanovení povinnosti, nebylo by možné, aby jeden finanční úřad stanovil jednomu poplatníkovi ve stejný den více povinností od stejného druhu daně. Výši platby nemá smysl uvažovat jako součást primárního identifikátoru. Z těchto důvodů bylo rozhodnuto již na konceptuální úrovni návrhu zavést umělý identifikátor povinnosti platby daně *ID povinnosti*. Protože má budovaný systém sloužit pro evidenci povinností platby daní jak napříč finančními úřady, tak i pro všechny poplatníky a druhy daně, je možné uvažovat, že tímto umělým identifikátorem bude číslo přiřazené každé povinnosti platby daně.

Databázi je třeba navrhnout tak, aby v ní bylo možno uložit značné množství záznamů o povinnostech platby daní. Např. v roce 2018 bylo v České republice evidováno

7 430 708 daňových subjektů, z čehož 5 101 045 jich bylo aktivních. Pokud bychom uvažovali, že na každý aktivní daňový subjekt bude připadat alespoň jeden záznam ročně, budou počty záznamů o stanovených povinnostech přibývat minimálně po milionech za rok. V tomto případě by bylo opravdu důležité upřesnit, v jakých řádech budou záznamy ročně přibývat s ohledem na volbu datového typu primárního identifikátoru, protože `INTEGER` o délce 4 bytů nemusí být dostatečný. Proto je v řešení této úlohy navrženo použít pro primární identifikátor *povinnosti platby* daně datový typ `NUMBER` bez omezení počtu cifer. Aby bylo možno čísla přiřazovaná povinnostem platby daně generovat, byl vytvořen objekt `POVINNOST_SEQ`, což je objekt typu `SEQUENCE`.

6.6.2 Identifikace finančního úřadu a druhu daně

Finanční úřady jsou vyjmenovány v § 8 zák. č. 456/2011 Sb., přičemž ze znění tohoto paragrafu lze dovodit, že finanční úřady jsou identifikovány svým názvem. Tomu odpovídá i identifikace finančního úřadu v konceptuálním schématu. Obdobně je přistupováno k identifikaci druhu daně. Kdyby ale tyto atributy byly použity jako primární klíče odpovídajících tabulek, tak vzhledem k povolené délce textových řetězců by indexy nad odpovídajícími cizími klíči v tabulce *povinnost platby daně* byly značně neefektivní, což by vzhledem k očekávanému množství záznamů v ní představovalo závažný problém. Dlouhé textové řetězce by také následně zvyšovaly výpočetní nároky na provedení operace JOIN, a v neposlední řadě by taková volba významně zvětšovala velikost tabulky *povinnost platby daně*, a tím i velikost databáze.

Pro účely elektronické komunikace s Finanční správou České republiky Generální finanční ředitelství provozuje Daňový portál[16]. Zde jsou k dispozici číselníky, mezi kterými lze nalézt číselníky *Územní finanční orgány* a *Druhy příjmů – FÚ* (Generální finanční ředitelství, 2019a). Při řešení této úlohy bylo těchto číselníků využito při modelování a návrhu entitních množin *finanční úřad* a *druh daně* v konceptuálním datovém modelu, resp. jim odpovídajících tabulek. Kromě délek *názvu druhu daně* a *názvu finančního úřadu* bylo pro identifikaci druhu daně rozhodnuto použít *číslo druhu příjmu (c_drp)* a *číslo územního finančního orgánu (c_ufo)*.

6.6.3 Identifikace osoby

Dle zadání mohou existovat fyzické osoby, které nemají přiděleno rodné číslo a právnické osoby, které nemají přiděleno IČO. Přesto je potřeba každou osobu

[16] http://adisspr.mfcr.cz/adistc/adis/idpr_pub/dpr/uvod.faces

jednoznačně identifikovat. Proto bylo jako primární identifikátor osoby určeno identifikační číslo. Identifikace subjektu v daňových vztazích je v této úloze zjednodušena, nicméně pojem „identifikační číslo“ je použit např. ve formuláři žádosti o zřízení daňové informační schránky (Generální finanční ředitelství, 2019b).

6.6.4 Zvolené řešení generalizace/specializace

Protože je vztah *typ osoby* výlučný a úplný, bylo pro něj zvoleno řešení jedné tabulky pro nadtyp i podtypy obdobně jako v případě úlohy *Evidence faktur*. Do tabulky *osoba* byl přidán sloupec *typ osoby*, který nabývá hodnoty „FO“ pro fyzické osoby a hodnoty „PO“ pro právnické osoby. Omezením typu `CHECK` je pak zajištěno, že v tabulce *osoba* budou u fyzických, potažmo právnických osob vyplněny pouze relevantní sloupce. Pro fyzické a právnické osoby byly také vytvořeny aktualizovatelné pohledy *fyzická osoba* a *právnická osoba*.

6.6.5 Vyživované děti

Daňové zvýhodnění na vyživované děti mohou uplatňovat pouze osoby fyzické. Protože dítě je také fyzickou osobou, vztah *vyživuje* je vztahem, na jehož obou koncích je entitní množina *fyzická osoba*. Na základě zadání úlohy má pak vztah *vyživuje* atribut *daňové období*, v rámci kterého fyzická osoba daňové zvýhodnění uplatňuje.

Vztah vyživuje je vztahem M:N, jelikož jedna osoba mohla uplatnit daňové zvýhodnění na více dětí, a zároveň na jedno dítě mohlo být daňové zvýhodnění uplatněno ve více daňových obdobích, případně v různých daňových obdobích vícero fyzickými osobami. Primární klíč tabulky *vyživování* ale není tvořen kombinací primárního klíče fyzické osoby-živitele a vyživované fyzické osoby (dítěte), protože takovýto primární klíč by dovolil existenci pouze jednoho řádku tabulky pro určitou kombinaci živitel-dítě, což by znamenalo, že by živitel mohl na konkrétní dítě uplatnit slevu na dani jen v jednom daňovém období. Primární klíč tabulky *vyživování* je tak tvořen kombinací primárního klíče vyživované fyzické osoby (dítěte) a daňového období, protože na jedno dítě smí být dle zadání v jednom období uplatněna sleva na dani jen jednou.

Požadavek, aby vyživování mohlo nastat pouze mezi osobami fyzickými, je pak zajištěn díky tomu, že v tabulce *vyživování* jsou součástí definovaných cizích klíčů i sloupce *vyživovaná osoba typ osoby* a *živitel typ osoby*. V tabulce *vyživování* jsou hodnoty sloupců *vyživovaná osoba typ osoby* a *živitel typ osoby* omezeny na hodnotu „FO“ pomocí omezení typu `CHECK` (viz SQL kód ve výpisu 5). Protože právnické osoby budou

mít hodnotu sloupce *typ osoby* „PO“, toto omezení vyloučí, aby cizí klíče v tabulce *vyživování* odkazovaly na záznamy o právnických osobách.

Protože primárním klíčem v tabulce *osoba* je *identifikační číslo*, bylo v tabulce *osoba* deklarováno omezení typu `UNIQUE` pro kombinaci sloupců *identifikační číslo* a *typ osoby*. Díky tomuto omezení bylo možno v tabulce *vyživování* definovat cizí klíče výše uvedeným způsobem.

Pro získání přehledu vyživovaných fyzických osob pak byl zaveden pohled *vyživovaná osoba*.

6.6.6 Zavedení číselníku států

V rámci řešení této úlohy je využito atributu *kód státu* entitní množiny *osoba* pro ilustraci zavedení číselníku. Číselníky by bylo možné využit i při řešení jiných úloh, nicméně s ohledem na to, aby řešení jednotlivých úloh v těchto skriptech ukazovaly určité vybrané jevy, je zavedení číselníku ukázáno jen na následujícím příkladu.

Zemí na světě je konečný počet a kódy zemí jsou stanoveny standardem ISO 3166 (International Organization for Standardization, n.d.). Pokud by databáze byla navržena tak, že by byl v rámci adresy osoby vyplňován celý název státu, dříve nebo později by se stalo, že pro nějaký stát by v datech bylo uvedeno více názvů. Např. by někdo uvedl Spojené státy americké, jiný USA, a bylo by možné uvažovat i jazykové mutace názvů států. Pokud by někdo chtěl vypsat poplatníky z konkrétního státu, musel by nejprve zjistit, jaké všechny názvy daného státu se v datech vyskytují, aby bylo vůbec možné formulovat podmínku příslušného dotazu. Tomuto problému lze zamezit použitím číselníku států.

Číselník obecně představuje výčet hodnot (OASIS, 2007). Nejjednodušším číselníkem států by tak byl výčet názvů států. Protože politická mapa světa se v průběhu času mění, není příliš praktické implementovat číselník států jako omezení typu `CHECK` aplikované na sloupec *stát*. Vhodnější je zavést samostatnou tabulku států, protože číselník států pak lze upravovat pomocí příkazů `INSERT`, `UPDATE` a `DELETE`. V případě omezení typu `CHECK` by bylo třeba změnu číselníku realizovat provedením příkazů pro definici databázových objektů, resp. jejich změnu. Protože kód státu o délce dva nebo tři znaky, jaké zavádí standard ISO 3166, lze indexovat efektivněji než delší název státu, je vhodné, aby

tabulka číselníku měla minimálně dva sloupce: *kód státu* a *název státu*, přičemž *kód státu* bude primární klíč, který bude v tabulce *osoba* figurovat jako cizí klíč.

K číselníku států je třeba dodat, že v rámci číselníků dostupných na Daňovém portálu je k dispozici i číselník *Země* (Generální finanční ředitelství, 2019a). Tento číselník je komplexnější nežli číselník zavedený při řešení této úlohy. Využití takto komplexního číselníku jde ale nad rámec těchto skript.

6.6.7 Zvolené řešení poštovní adresy

Pro adresu daňového subjektu v rámci elektronické komunikace s Finanční správou České republiky je rozpad na jednotlivé elementy adresy včetně datových typů a maximálních povolených délek hodnot stanoven XML schématy pro datové zprávy. Reprezentace poštovní adresy v řešení této úlohy je založeno na XML schématech pro daň z příjmů fyzických osob (Generální finanční ředitelství, 2018a) a pro daň z příjmů právnických osob (Generální finanční ředitelství, 2018b).

6.6.8 Zvolené řešení čísla účtu

Číslo účtu je reprezentováno v souladu s vyhláškou České národní banky 169/2011 (Česká národní banka, 2011), která mimo jiné upravuje číslo účtu v národním formátu a ve formátu IBAN. V této úloze je aplikován národní formát čísla účtu.

Číslo účtu v národním formátu je dle vyhlášky ČNB 169/2011 tvořeno identifikátorem účtu klienta a kódem platebního styku. Identifikátor účtu klienta je tvořen první a druhou částí. První část (předčíslí) je nepovinná a tvoří ji nejvýše 6 číselných znaků. Úvodní nuly jsou bez významu. Dle (Finanční správa, 2019b) je předčíslí účtu pro platbu daně určeno druhem daně.

Druhá část identifikátoru čísla účtu klienta obsahuje nejméně 2 a nejvýše 10 číselných znaků, přičemž alespoň dva z těchto znaků nesmějí být nulové; úvodní nuly jsou bez významu (Česká národní banka, 2011). Druhá část identifikátoru účtu je v dokumentu (Finanční správa, 2019a) označována jako matrika a je určena finančním úřadem. Tento dokument určuje pro finanční úřady i kód banky, čili kód platebního styku dle vyhlášky ČNB 169/2011. Protože vyhláška v tomto případě neuvádí, že by úvodní nuly byly bez významu, je pro kód platebního styku (*kód banky*) použit datový typ `VARCHAR2`. V případě číselného datové typu by úvodní nuly nebyly uloženy.

6.7 Příkazy pro vytvoření relačních tabulek a dalších objektů

Výpis 5: Evidence daňových poplatníků – SQL skript, zdroj: autoři

```
CREATE SEQUENCE POVINNOST_SEQ;

/*==============================================================*/
/* Tabulka: DRUH_DANE                                           */
/*==============================================================*/
CREATE TABLE DRUH_DANE  (
   C_DRP                NUMBER(5)                     NOT NULL,
   NAZEV_DANE           VARCHAR2(50)                  NOT NULL,
   PRAVNI_PREDPIS       VARCHAR2(20)                  NOT NULL,
   PREDCISLI_UCTU       NUMBER(6)                     NOT NULL
      CONSTRAINT CKC_PREDCISLI_UCTU_DRUH_DAN CHECK (PREDCISLI_UCTU >= 0),
   CONSTRAINT AK_PREDCISLI_AI_DRUH_DAN UNIQUE (PREDCISLI_UCTU)
);

ALTER TABLE DRUH_DANE
   ADD CONSTRAINT PK_DRUH_DANE PRIMARY KEY (C_DRP);

/*==============================================================*/
/* Tabulka: FINANCNI_URAD                                       */
/*==============================================================*/
CREATE TABLE FINANCNI_URAD  (
   C_UFO                NUMBER(3,0)                   NOT NULL,
   NAZEV_FU             VARCHAR2(45)                  NOT NULL,
   SIDLO_FU             VARCHAR2(48)                  NOT NULL,
   MATRIKA              NUMBER(10)                    NOT NULL
      CONSTRAINT CKC_FU_MATRIKA
        CHECK (MATRIKA > 0 AND REGEXP_LIKE(MATRIKA,
                              '(?:[1-9][0-9]{0,9}|[0-9]{0,9}[1-9]){2}')),
   KOD_BANKY            VARCHAR2(4)                   NOT NULL
      CONSTRAINT CKC_FU_KOD_BANKY CHECK (REGEXP_LIKE(KOD_BANKY, '[0-9]{4}')),
   CONSTRAINT AK_NAZEV_AI_FINANCNI UNIQUE (NAZEV_FU),
   CONSTRAINT AK_UCET_AI_FINANCNI UNIQUE (MATRIKA, KOD_BANKY)
);

ALTER TABLE FINANCNI_URAD
   ADD CONSTRAINT PK_FINANCNI_URAD PRIMARY KEY (C_UFO);

```

```
38  /*==============================================================*/
39  /* Tabulka: OSOBA                                               */
40  /*==============================================================*/
41  CREATE TABLE OSOBA  (
42     IDENTIFIKACNI_CISLO  VARCHAR2(10)                    NOT NULL,
43     TYP_OSOBY            VARCHAR2(2)                     NOT NULL
44        CONSTRAINT CKC_TYP_OSOBY_OSOBA CHECK (TYP_OSOBY IN ('FO','PO')),
45     TELEFON              VARCHAR2(16)
46        CONSTRAINT CKC_TELEFON_OSOBA
47          CHECK (TELEFON IS NULL
48                 OR (REGEXP_LIKE (TELEFON, '^[+]?[0-9]{1,15}$'))),
49     EMAIL                VARCHAR2(254),
50     RODNE_CISLO          VARCHAR2(10)
51        CONSTRAINT CKC_RODNE_CISLO_OSOBA
52          CHECK (RODNE_CISLO IS NULL
53                 OR (REGEXP_LIKE(RODNE_CISLO, '[0-9]{9,10}'))),
54     ICO                  VARCHAR2(8),
55     JMENO                VARCHAR2(20),
56     PRIJMENI             VARCHAR2(36),
57     RODNE_PRIJMENI       VARCHAR2(36),
58     NAZEV_PO             VARCHAR2(255),
59     ULICE                VARCHAR2(48),
60     C_POPISNE            NUMBER(6,0),
61     C_ORIENTACNI         VARCHAR2(4),
62     OBEC                 VARCHAR2(48),
63     PSC                  VARCHAR2(10),
64     KOD_STATU            VARCHAR2(2),
65     CONSTRAINT AK_RODNE_C_AI_OSOBA UNIQUE (RODNE_CISLO),
66     CONSTRAINT AK_ICO_AI_OSOBA UNIQUE (ICO),
67     CONSTRAINT AK_DIC_TYP_AI_OSOBA
68       UNIQUE (IDENTIFIKACNI_CISLO, TYP_OSOBY),
69     CONSTRAINT CKT_OSOBA
70       CHECK ((TYP_OSOBY='PO'
71               AND NAZEV_PO IS NOT NULL
72               AND ICO IS NOT NULL
73               AND JMENO IS NULL
74               AND PRIJMENI IS NULL
75               AND RODNE_PRIJMENI IS NULL
76               AND RODNE_CISLO IS NULL)
77              OR
78               (TYP_OSOBY='FO'
79               AND NAZEV_PO IS NULL
80               AND ICO IS NULL
81               AND JMENO IS NOT NULL
82               AND PRIJMENI IS NOT NULL
83               AND RODNE_CISLO IS NOT NULL))
84  );
85
```

```
ALTER TABLE OSOBA
   ADD CONSTRAINT PK_OSOBA
     PRIMARY KEY (IDENTIFIKACNI_CISLO);

/*==============================================================*/
/* Tabulka: POVINNOST_PLATBY_DANE                               */
/*==============================================================*/
CREATE TABLE POVINNOST_PLATBY_DANE  (
   ID_POVINNOSTI        NUMBER                          NOT NULL,
   IDENTIFIKACNI_CISLO  VARCHAR2(10)                    NOT NULL,
   C_UFO                NUMBER(3,0)                     NOT NULL,
   C_DRP                NUMBER(5)                       NOT NULL,
   VYSE_PLATBY          NUMBER(8,2)                     NOT NULL,
   DATUM                DATE                            NOT NULL
);

ALTER TABLE POVINNOST_PLATBY_DANE
   ADD CONSTRAINT PK_POVINNOST_PLATBY_DANE
     PRIMARY KEY (ID_POVINNOSTI);

/*==============================================================*/
/* Tabulka: STAT                                                */
/*==============================================================*/
CREATE TABLE STAT  (
   KOD_STATU            VARCHAR2(2)                     NOT NULL,
   NAZEV_STATU          VARCHAR2(50)                    NOT NULL
);

ALTER TABLE STAT
   ADD CONSTRAINT PK_STAT
     PRIMARY KEY (KOD_STATU);

/*==============================================================*/
/* Tabulka: VYZIVOVANI                                          */
/*==============================================================*/
CREATE TABLE VYZIVOVANI  (
   VYZIVOVANY_DIC       VARCHAR2(10)                    NOT NULL,
   VYZIVOVANY_TYP_OSOBY VARCHAR2(2)                     NOT NULL
      CONSTRAINT CKC_VYZIVOVANY_TYP_OS_VYZIVOVA
        CHECK (VYZIVOVANY_TYP_OSOBY IN ('FO')),
   DANOVE_OBDOBI        NUMBER(4)                       NOT NULL
      CONSTRAINT CKC_DANOVE_OBDOBI_VYZIVOVA
        CHECK (DANOVE_OBDOBI >= 1993),
   ZIVITEL_DIC          VARCHAR2(10)                    NOT NULL,
   ZIVITEL_TYP_OSOBY    VARCHAR2(2)                     NOT NULL
      CONSTRAINT CKC_ZIVITEL_TYP_OSOBY_VYZIVOVA
        CHECK (ZIVITEL_TYP_OSOBY IN ('FO'))
);
```

```
134
135    ALTER TABLE VYZIVOVANI
136       ADD CONSTRAINT PK_VYZIVOVANI
137         PRIMARY KEY (VYZIVOVANY_DIC, DANOVE_OBDOBI);
138
139    /*==============================================================*/
140    /* Index: ZIVITEL_FK                                            */
141    /*==============================================================*/
142    CREATE INDEX ZIVITEL_FK ON VYZIVOVANI (
143       ZIVITEL_DIC ASC,
144       ZIVITEL_TYP_OSOBY ASC
145    );
146
147    /*==============================================================*/
148    /* View: FYZICKA_OSOBA                                          */
149    /*==============================================================*/
150    CREATE OR REPLACE VIEW FYZICKA_OSOBA AS
151    SELECT
152       IDENTIFIKACNI_CISLO, RODNE_CISLO, JMENO, PRIJMENI, RODNE_PRIJMENI,
153       ULICE, C_POPISNE, C_ORIENTACNI, OBEC, PSC, KOD_STATU, TELEFON,
154       EMAIL
155    FROM
156       OSOBA
157    WHERE TYP_OSOBY = 'FO'
158    WITH CHECK OPTION;
159
160    /*==============================================================*/
161    /* View: PRAVNICKA_OSOBA                                        */
162    /*==============================================================*/
163    CREATE OR REPLACE VIEW PRAVNICKA_OSOBA AS
164    SELECT
165       IDENTIFIKACNI_CISLO, ICO, NAZEV_PO, ULICE, C_POPISNE,
166       C_ORIENTACNI, OBEC, PSC, KOD_STATU, TELEFON, EMAIL
167    FROM
168       OSOBA
169    WHERE TYP_OSOBY = 'PO'
170    WITH CHECK OPTION;
```

```
171  /*==============================================================*/
172  /* View: VYZIVOVANA_OSOBA                                       */
173  /*==============================================================*/
174  CREATE OR REPLACE VIEW VYZIVOVANA_OSOBA AS
175  SELECT
176     IDENTIFIKACNI_CISLO, RODNE_CISLO, JMENO, PRIJMENI, RODNE_PRIJMENI,
177     ULICE, C_POPISNE, C_ORIENTACNI, OBEC, PSC, KOD_STATU, TELEFON,
178     EMAIL
179  FROM
180     OSOBA
181  WHERE
182     TYP_OSOBY = 'FO'
183     AND
184     IDENTIFIKACNI_CISLO IN (SELECT VYZIVOVANY_DIC FROM VYZIVOVANI);
185
186  ALTER TABLE OSOBA
187     ADD CONSTRAINT FK_OSOBA_STAT
188       FOREIGN KEY (KOD_STATU)
189        REFERENCES STAT (KOD_STATU);
190
191  ALTER TABLE POVINNOST_PLATBY_DANE
192     ADD CONSTRAINT FK_POVINNOS_MA_ZAPLAT_OSOBA
193       FOREIGN KEY (IDENTIFIKACNI_CISLO)
194        REFERENCES OSOBA (IDENTIFIKACNI_CISLO);
195
196  ALTER TABLE POVINNOST_PLATBY_DANE
197     ADD CONSTRAINT FK_POVINNOS_VYMERIL_FINANCNI
198       FOREIGN KEY (C_UFO)
199        REFERENCES FINANCNI_URAD (C_UFO);
200
201  ALTER TABLE POVINNOST_PLATBY_DANE
202     ADD CONSTRAINT FK_POVINNOS_VZNIKLA_DRUH_DAN
203       FOREIGN KEY (C_DRP)
204        REFERENCES DRUH_DANE (C_DRP);
205
206  ALTER TABLE VYZIVOVANI
207     ADD CONSTRAINT FK_VYZIVOVANI_VYZIV_OS
208       FOREIGN KEY (VYZIVOVANY_DIC, VYZIVOVANY_TYP_OSOBY)
209        REFERENCES OSOBA (IDENTIFIKACNI_CISLO, TYP_OSOBY);
210
211  ALTER TABLE VYZIVOVANI
212     ADD CONSTRAINT FK_VYZIVOVANI_ZIVITEL
213       FOREIGN KEY (ZIVITEL_DIC, ZIVITEL_TYP_OSOBY)
214        REFERENCES OSOBA (IDENTIFIKACNI_CISLO, TYP_OSOBY);
```

7 Plánování výuky – varianta A

7.1 Účel úlohy

Tato úloha ilustruje zejména následující problémy/jevy:

- Nikoli výlučné podmnožiny entitní množiny.
- Komplexní požadavek na unikátnost.

7.2 Popis zvolené výseče světa – zadání

Vysoká škola plánuje výuku a potřebuje evidovat údaje o učitelích a předmětech, které mohou vyučovat nebo které garantují, o studentech, a o výuce jednotlivých předmětů. Ta je plánována vždy na jeden semestr akademického roku (akademické období) a nejsou uvažovány mimo-semestrální kurzy. Výuka předmětů probíhá v rámci rozvrhových akcí (přednáška, nebo cvičení). Rozvrhová akce se koná periodicky po dobu semestru, a to buď každý týden, nebo pouze v sudé či liché týdny semestru.

O učitelích je třeba evidovat: osobní číslo, jméno, titul, login a rozsah úvazku vyjádřený počtem hodin týdně. Učitel musí být schopen vyučovat nebo musí garantovat alespoň jeden předmět. O studentech je třeba evidovat: osobní číslo, jméno, titul, pokud byl již studentovi udělen, login a počet uzavřených semestrů. Student musí být nejprve přijat do studia, než si bude moci zapsat nějaký předmět. Studenti mohou být i učiteli a učitelé mohou i studovat – často je to případ studentů doktorského studia.

O předmětech je požadováno evidovat kód předmětu, název předmětu a popis zaměření předmětu. Předmět musí mít vždy určeného právě jednoho garanta z řad učitelů, nicméně v databázi může být evidován i předmět, ke kterému neexistuje ani jeden učitel, který by ho mohl vyučovat.

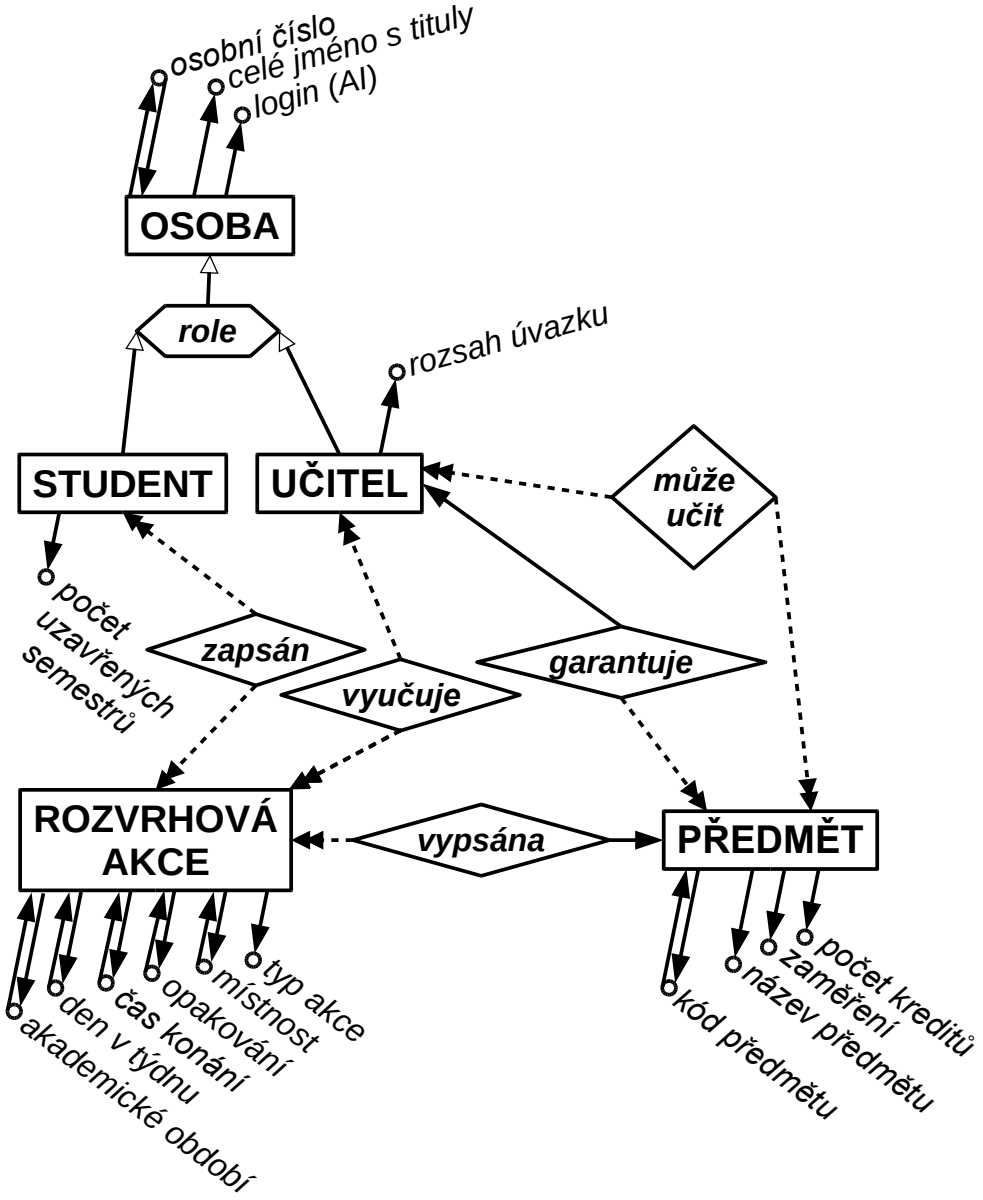
U rozvrhových akcí je nezbytné sledovat následující údaje:

- kdy se výuka koná – akademický rok, semestr, den v týdnu, čas konání výuky;
- opakování – každý týden / lichý týden / sudý týden;
- na které učebně se výuka koná – označení učebny;
- který učitel je vyučujícím rozvrhové akce – jedna rozvrhová akce může být vyučována i více učiteli;
- kteří studenti jsou zapsáni na rozvrhovou akci, nicméně rozvrhovou akci předmětu musí být možné nejprve vypsat, aby se na ni mohli studenti zapsat.

Databázi je třeba navrhnout tak, aby bylo možno pro zvolený semestr vypsat rozvrh konkrétní osoby, ať už se jedná o studenta nebo učitele, nebo rozvrh konkrétního předmětu.

Zadání je zjednodušené v tom, že nepředpokládáme, že by učitel mohl mít více pracovních poměrů, a tedy více úvazků. Tato situace je podrobně řešena ve variantě C v kapitole 9.

7.3 Konceptuální schéma



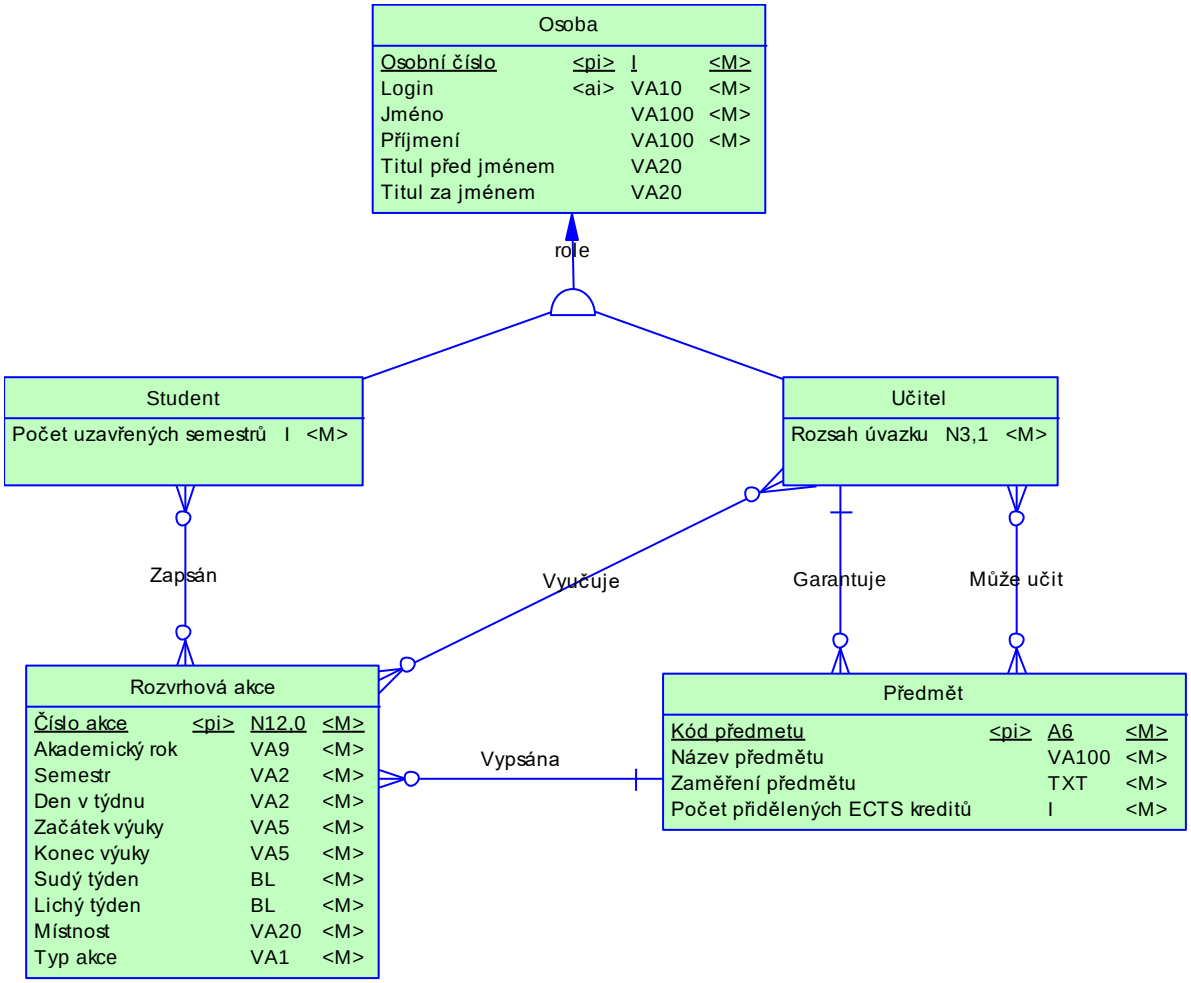
Obrázek 50: Plánování výuky A – konceptuální schéma, zdroj: autoři

V případě konceptuálního schématu na obrázku 50 neumožňuje použitá notace vyjádřit primární identifikátor rozvrhové akce zcela přesně. Aby dvě rozvrhové akce nekolidovaly, tak pokud mají shodné akademické období, den v týdnu, čas konání a místnost, tak jedna musí probíhat jen v liché týdny semestru, a druhá jen v sudé týdny semestru. Pokud rozvrhová akce probíhá každý týden semestru, tj. jak v liché týdny, tak v sudé týdny, nemůže být již naplánována druhá akce ve shodném akademickém období, konající se shodný den, ve shodný čas a ve stejné místnosti. Šipka od atributu *opakování* k entitní množině *rozvrhová akce* vyjadřuje toliko to, že opakování v identifikaci rozvrhové akce hraje roli.

Pokud by byl atribut *opakování* chápán jako atribut nabývající hodnoty *lichý týden*, nebo *sudý týden*, nebo *každý týden*, tak ne všechny možné kombinace hodnot atributů tvořících primární identifikátor rozvrhové akce na obrázku 50 by představovaly přípustné rozvrhové akce. Pro určitou kombinaci akademického období, dne v týdnu, času konání a místnosti by totiž mohly existovat tři akce: jedna s hodnotou atributu opakování rovnou lichému týdnu, druhá s hodnotou tohoto atributu rovnou sudému týdnu, a třetí s hodnotou tohoto atributu rovnou každému týdnu. Tyto akce by ale byly kolizní. Buď mohou být vypsány pouze první dvě, nebo jen třetí z nich, ale nikoli všechny tři současně, což znamená. Na existenci záznamů o rozvrhových akcích v databázi platí stejné podmínky. Požadavek, aby rozvrhové akce nekolidovaly, je tak vyřešen až na fyzické úrovni návrhu za pomoci podmíněných unikátních indexů.

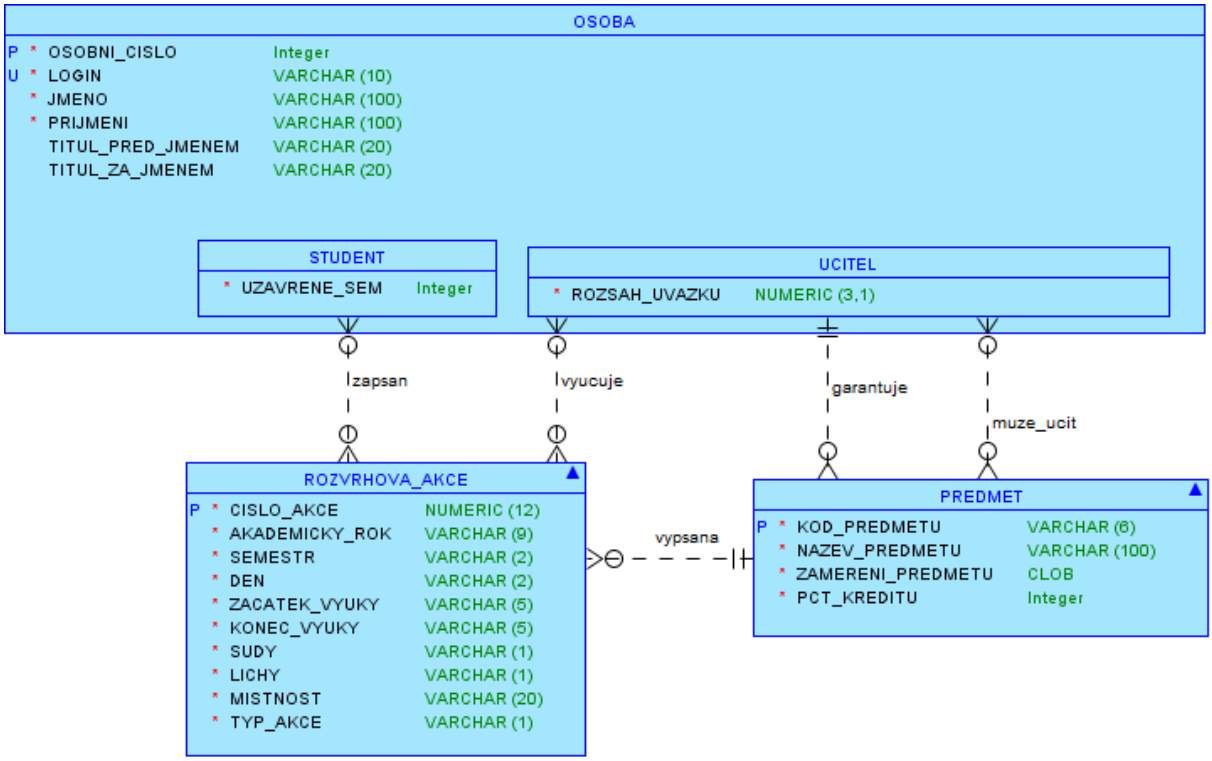
7.4 Konceptuální datový model

7.4.1 KDM v nástroji Power Designer



Obrázek 51: Plánování výuky A – konceptuální datový model (Power Designer), zdroj: autoři

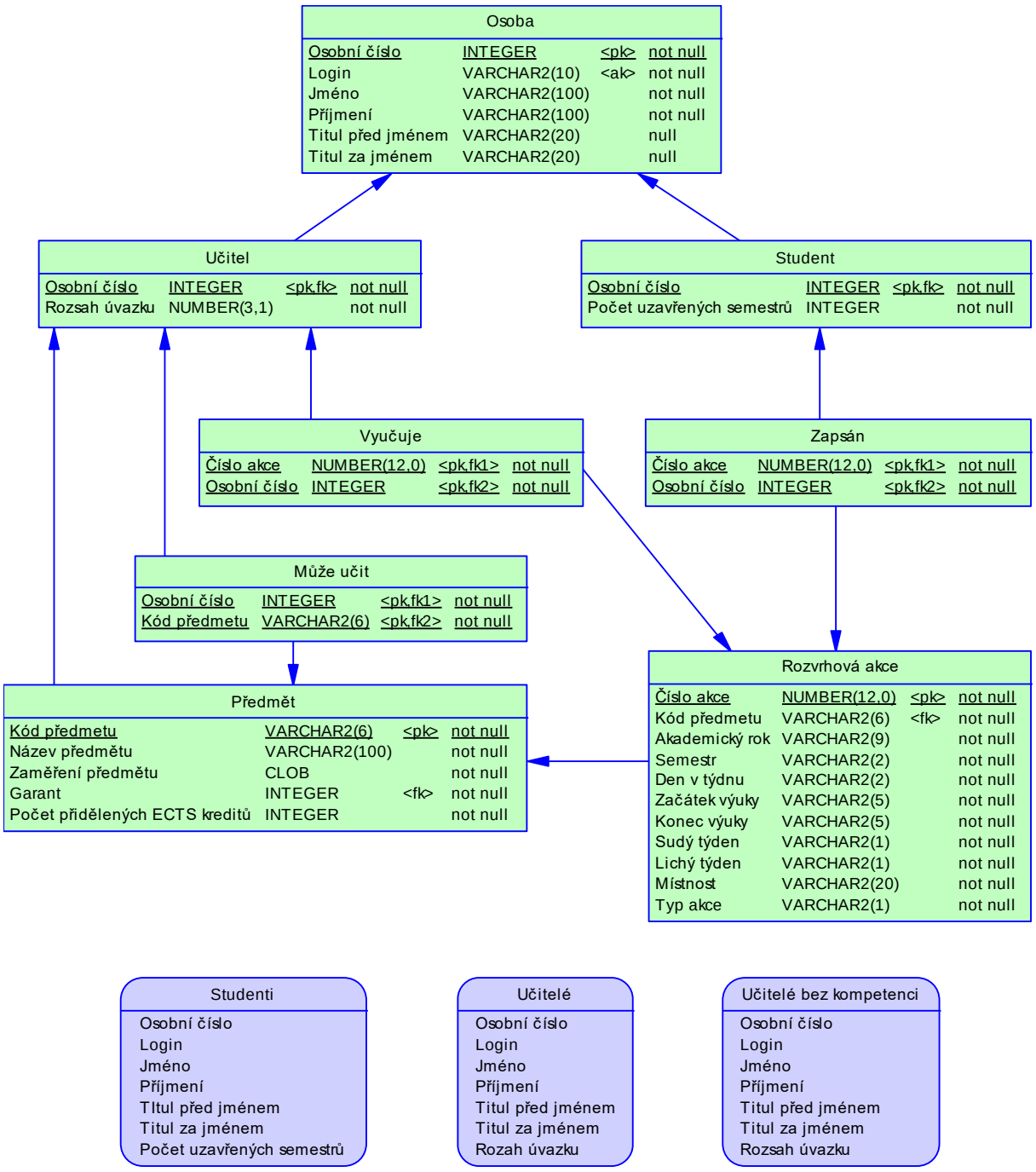
7.4.2 KDM v nástroji Data Modeler



Obrázek 52: Plánování výuky A – konceptuální datový model (Data Modeler), zdroj: autoři

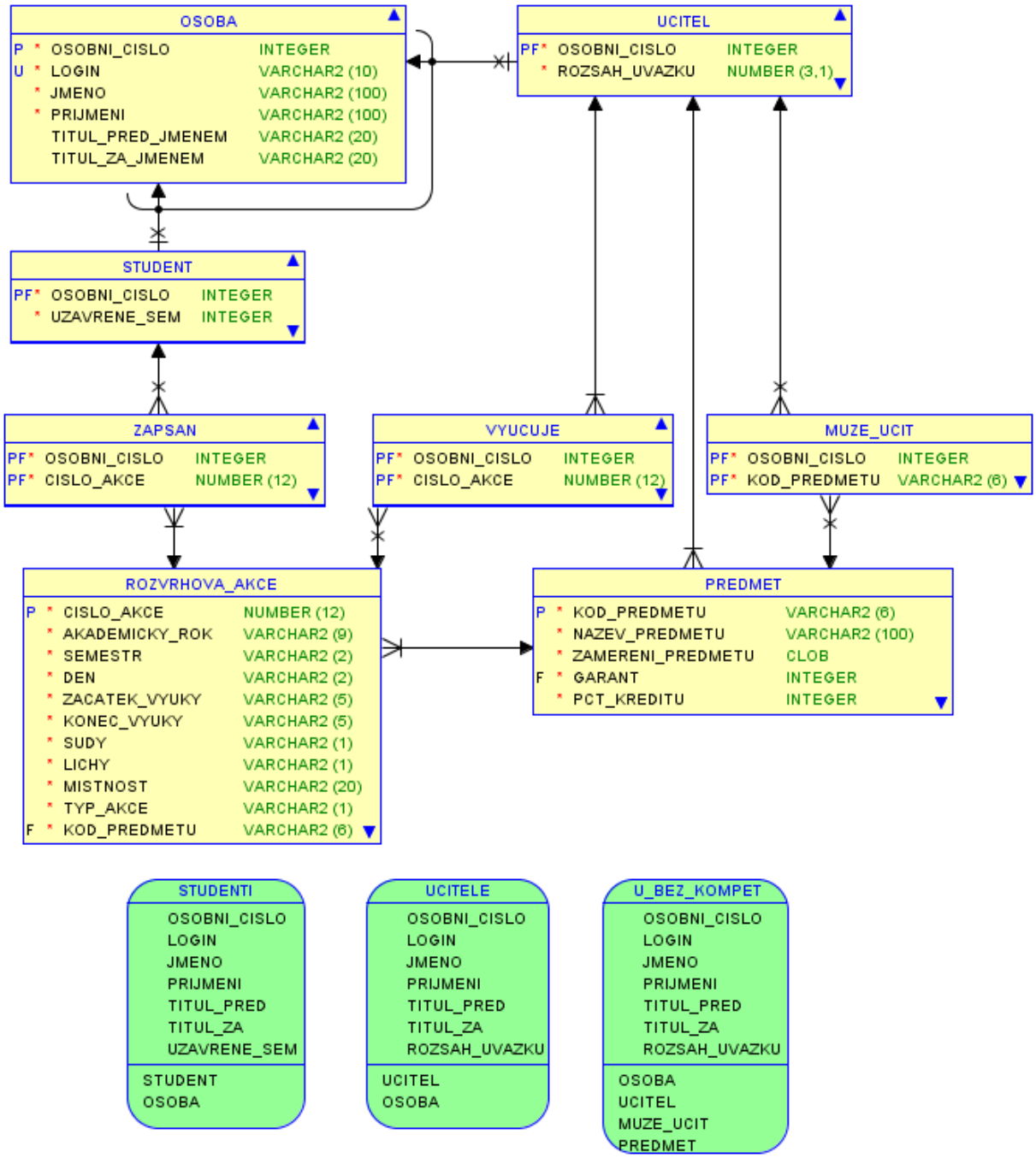
7.5 Fyzický datový model

7.5.1 FDM v nástroji Power Designer



Obrázek 53: Plánování výuky A – fyzický datový model (Power Designer), zdroj: autoři

7.5.2 FDM v nástroji Data Modeler



Obrázek 54: Plánování výuky A – fyzický datový model (Data Modeler), zdroj: autoři

7.6 Komentář k návrhu

Na základě zadání byly identifikovány entitní množiny *osoba*, *student*, *učitel*, *rozvrhová akce* a *předmět*. Entitní množiny *student* a *učitel* představují nevýlučné podtypy *osoby*, protože studenti mohou být učiteli a obráceně. Z tohoto důvodu jsou

pak vytvořeny samostatné tabulky *osoba*, *student* a *učitel*. Protože data o učitelích a studentech jsou rozdělena do více tabulek, jsou vytvořeny pohledy poskytující ucelený pohled na data o učitelích a studentech.

Požadavek na to, že každý učitel je garantem nějakého předmětu nebo může nějaký předmět vyučovat, lze kontrolovat pomocí pohledu, který by vypisoval záznamy daný požadavek nesplňující; tedy vypisoval by učitele, kteří nejsou garanty ani nemají záznam o tom, že mohou něco vyučovat. Tímto pohledem je pohled *učitelé bez kompetencí*.

Pokud by primární klíč tabulky *rozvrhová akce* byl vytvořen dle přirozeného primárního identifikátoru z konceptuálního schématu na obrázku 50, jako složený cizí klíč by se promítl do tabulek *zapsán* a *vyučuje*. Díky většímu množství sloupců tvořících primární, resp. cizí klíč, by bylo přinejmenším komplikované pracovat s tabulkami v rámci SQL dotazů. Proto byl pro *rozvrhovou akci* zaveden umělý identifikátor *číslo akce*.

Požadavek nepřekrývání rozvrhu v místnostech je realizován podmíněnými unikátními indexy pro sudé a liché týdny (viz řádky 191-209 výpisu 6). Konce výuky v tomto řešení nejsou zohledněny, v obecném případě by ale bylo nutno řešit i nepřekrývání rozvrhových akcí z tohoto hlediska.

Index `SUDY_AK` indexuje pouze akce, které mají ve sloupci *sudý* hodnotu „A“. Obdobně index `LICHY_AK` indexuje pouze akce, které mají ve sloupci *lichý* hodnotu „A“. Pokud výuka probíhá každý týden, tj. lichý i sudý, bude mít příslušná rozvrhová akce v obou sloupcích *lichý* a *sudý* hodnotu „A“. Pokus vložit do databáze záznam o rozvrhové akci, která bude probíhat ve stejné akademické období, den v týdnu, čas a na stejné místnosti skončí neúspěchem kvůli porušení alespoň jedné z podmínek unikátnosti výše uvedených indexů.

Naopak, pokud bude rozvrhová akce probíhat např. pouze v sudé týdny, bude možné založit záznam o rozvrhové akci, která bude probíhat ve stejném akademické období, stejný den v týdnu, čas a na stejné místnosti, ale v liché týdny. Akce probíhající v sudé týdny bude mít hodnotu sloupce *sudý* rovnu „A“, a sloupce *lichý* rovnu „N“, v důsledku čehož bude indexována pouze indexem `SUDY_AK`. Pro stejnou kombinaci akademického

období, dne v týdnu, času a místnosti tak bude možné založit akci, která bude indexována pouze indexem LICHY_AK, tj. akci probíhající pouze v liché týdny.

7.7 Příkazy pro vytvoření relačních tabulek a dalších objektů

Výpis 6: Plánování výuky A – SQL skript, zdroj: autoři

```
/*==============================================================*/
/* Tabulka: MUZE_UCIT                                           */
/*==============================================================*/
CREATE TABLE MUZE_UCIT  (
OSOBNI_CISLO        INTEGER                       NOT NULL,
KOD_PREDMETU        VARCHAR2(6)                   NOT NULL,
CONSTRAINT PK_MUZE_UCIT PRIMARY KEY (OSOBNI_CISLO, KOD_PREDMETU)
);

/*==============================================================*/
/* Tabulka: OSOBA                                               */
/*==============================================================*/
CREATE TABLE OSOBA  (
OSOBNI_CISLO         INTEGER                       NOT NULL,
LOGIN                VARCHAR2(10)                  NOT NULL,
JMENO                VARCHAR2(100)                 NOT NULL,
PRIJMENI             VARCHAR2(100)                 NOT NULL,
TITUL_PRED           VARCHAR2(20),
TITUL_ZA             VARCHAR2(20),
CONSTRAINT PK_OSOBA PRIMARY KEY (OSOBNI_CISLO),
CONSTRAINT AK_LOGIN_AI_OSOBA UNIQUE (LOGIN)
);

/*==============================================================*/
/* Tabulka: PREDMET                                             */
/*==============================================================*/
CREATE TABLE PREDMET  (
KOD_PREDMETU         VARCHAR2(6)                   NOT NULL,
GARANT               INTEGER                       NOT NULL,
NAZEV_PREDMETU       VARCHAR2(100)                 NOT NULL,
ZAMERENI_PREDMETU    CLOB                          NOT NULL,
PCT_KREDITU          INTEGER                       NOT NULL
CONSTRAINT CKC_PCT_KREDITU_PREDMET CHECK (PCT_KREDITU >= 0),
CONSTRAINT PK_PREDMET PRIMARY KEY (KOD_PREDMETU)
);

```

```
37  /*==============================================================*/
38  /* Tabulka: ROZVRHOVA_AKCE                                      */
39  /*==============================================================*/
40  CREATE TABLE ROZVRHOVA_AKCE  (
41  CISLO_AKCE           NUMBER(12,0)                    NOT NULL,
42  KOD_PREDMETU         VARCHAR2(6)                     NOT NULL,
43  AKADEMICKY_ROK       VARCHAR2(9)                     NOT NULL,
44  SEMESTR              VARCHAR2(2)                     NOT NULL
45  CONSTRAINT CKC_SEMESTR_ROZVRHOV CHECK (SEMESTR IN ('LS','ZS')),
46  DEN                  VARCHAR2(2)                     NOT NULL
47  CONSTRAINT CKC_DEN_ROZVRHOV CHECK (DEN IN
    ('Po','Ut','St','Ct','Pa','So','Ne')),
48  ZACATEK_VYUKY        VARCHAR2(5)                     NOT NULL
49  CONSTRAINT CKC_ZACATEK_VYUKY_ROZVRHOV CHECK (REGEXP_LIKE(ZACATEK_VYUKY,'^(0[0-
    9]|1[0-9]|2[0-3]):[0-5][0-9]$')),
50  KONEC_VYUKY          VARCHAR2(5)                     NOT NULL
51  CONSTRAINT CKC_KONEC_VYUKY_ROZVRHOV CHECK (REGEXP_LIKE(KONEC_VYUKY,'^(0[0-
    9]|1[0-9]|2[0-3]):[0-5][0-9]$')),
52  SUDY                 VARCHAR2(1)                     DEFAULT 'A' NOT NULL
53  CONSTRAINT CKC_SUDY_ROZVRHOV CHECK (SUDY IN ('A','N')),
54  LICHY                VARCHAR2(1)                     DEFAULT 'A' NOT NULL
55  CONSTRAINT CKC_LICHY_ROZVRHOV CHECK (LICHY IN ('A','N')),
56  MISTNOST             VARCHAR2(20)                    NOT NULL,
57  TYP_AKCE             VARCHAR2(1)                     NOT NULL
58  CONSTRAINT CKC_TYP_AKCE_ROZVRHOV CHECK (TYP_AKCE IN ('P','C')),
59  CONSTRAINT PK_ROZVRHOVA_AKCE PRIMARY KEY (CISLO_AKCE),
60  CONSTRAINT CKT_ROZVRHOVA_AKCE CHECK (SUDY = 'A' OR LICHY = 'A')
61  );
62
63  /*==============================================================*/
64  /* Tabulka: STUDENT                                             */
65  /*==============================================================*/
66  CREATE TABLE STUDENT  (
67  OSOBNI_CISLO         INTEGER                         NOT NULL,
68  UZAVRENE_SEM         INTEGER                         NOT NULL
69  CONSTRAINT CKC_UZAVRENE_SEM_STUDENT CHECK (UZAVRENE_SEM >= 0),
70  CONSTRAINT PK_STUDENT PRIMARY KEY (OSOBNI_CISLO)
71  );
72
73  /*==============================================================*/
74  /* Tabulka: UCITEL                                              */
75  /*==============================================================*/
76  CREATE TABLE UCITEL  (
77  OSOBNI_CISLO         INTEGER                         NOT NULL,
78  ROZSAH_UVAZKU        NUMBER(3,1)                     NOT NULL
79  CONSTRAINT CKC_ROZSAH_UVAZKU_UCITEL CHECK (ROZSAH_UVAZKU >= 0),
80  CONSTRAINT PK_UCITEL PRIMARY KEY (OSOBNI_CISLO)
81  );
```

```
82
83  /*==============================================================*/
84  /* Tabulka: VYUCUJE                                             */
85  /*==============================================================*/
86  CREATE TABLE VYUCUJE  (
87  CISLO_AKCE          NUMBER(12,0)                  NOT NULL,
88  OSOBNI_CISLO         INTEGER                       NOT NULL,
89  CONSTRAINT PK_VYUCUJE PRIMARY KEY (CISLO_AKCE, OSOBNI_CISLO)
90  );
91
92  /*==============================================================*/
93  /* Tabulka: ZAPSAN                                              */
94  /*==============================================================*/
95  CREATE TABLE ZAPSAN  (
96  CISLO_AKCE           NUMBER(12,0)                  NOT NULL,
97  OSOBNI_CISLO         INTEGER                       NOT NULL,
98  CONSTRAINT PK_ZAPSAN PRIMARY KEY (CISLO_AKCE, OSOBNI_CISLO)
99  );
100
101 /*==============================================================*/
102 /* View: V_BEZ_KOMPETENCI                                      */
103 /*==============================================================*/
104 CREATE OR REPLACE VIEW V_BEZ_KOMPETENCI AS
105 SELECT
106 OSOBNI_CISLO, LOGIN, JMENO, PRIJMENI, TITUL_PRED, TITUL_ZA,
107 ROZSAH_UVAZKU
108 FROM
109 OSOBA JOIN UCITEL USING(OSOBNI_CISLO)
110 WHERE
111 OSOBNI_CISLO NOT IN (SELECT OSOBNI_CISLO FROM MUZE_UCIT)
112 AND OSOBNI_CISLO NOT IN (SELECT GARANT FROM PREDMET);
113
114 /*==============================================================*/
115 /* View: V_STUDENT                                              */
116 /*==============================================================*/
117 CREATE OR REPLACE VIEW V_STUDENT AS
118 SELECT
119 OSOBNI_CISLO, LOGIN, JMENO, PRIJMENI, TITUL_PRED, TITUL_ZA,
120 UZAVRENE_SEM
121 FROM
122 OSOBA JOIN STUDENT USING (OSOBNI_CISLO)
123 WITH CHECK OPTION;
124
```

```
125  /*==============================================================*/
126  /* View: V_UCITEL                                               */
127  /*==============================================================*/
128  CREATE OR REPLACE VIEW V_UCITEL AS
129  SELECT
130  OSOBNI_CISLO, LOGIN, JMENO, PRIJMENI, TITUL_PRED, TITUL_ZA,
131  ROZSAH_UVAZKU
132  FROM
133  OSOBA JOIN UCITEL USING(OSOBNI_CISLO)
134  WITH CHECK OPTION;
135
136  ALTER TABLE MUZE_UCIT
137  ADD CONSTRAINT FK_MUZE_UCIT_UCITEL FOREIGN KEY (OSOBNI_CISLO)
138  REFERENCES UCITEL (OSOBNI_CISLO) ON DELETE CASCADE;
139
140  ALTER TABLE MUZE_UCIT
141  ADD CONSTRAINT FK_MUZE_UCIT_PREDMET FOREIGN KEY (KOD_PREDMETU)
142  REFERENCES PREDMET (KOD_PREDMETU) ON DELETE CASCADE;
143
144  ALTER TABLE PREDMET
145  ADD CONSTRAINT FK_PREDMET_GARANTUJE_UCITEL FOREIGN KEY (GARANT)
146  REFERENCES UCITEL (OSOBNI_CISLO);
147
148  ALTER TABLE ROZVRHOVA_AKCE
149  ADD CONSTRAINT FK_ROZVRHOV_VYUKA_PREDMET FOREIGN KEY (KOD_PREDMETU)
150  REFERENCES PREDMET (KOD_PREDMETU);
151
152  ALTER TABLE STUDENT
153  ADD CONSTRAINT FK_STUDENT_ROLE_OSOBA FOREIGN KEY (OSOBNI_CISLO)
154  REFERENCES OSOBA (OSOBNI_CISLO) ON DELETE CASCADE;
155
156  ALTER TABLE UCITEL
157  ADD CONSTRAINT FK_UCITEL_ROLE_OSOBA FOREIGN KEY (OSOBNI_CISLO)
158  REFERENCES OSOBA (OSOBNI_CISLO) ON DELETE CASCADE;
159
160  ALTER TABLE VYUCUJE
161  ADD CONSTRAINT FK_VYUCUJE_VYUCUJE_UCITEL FOREIGN KEY (OSOBNI_CISLO)
162  REFERENCES UCITEL (OSOBNI_CISLO);
163
164  ALTER TABLE VYUCUJE
165  ADD CONSTRAINT FK_VYUCUJE_VYUCUJE_ROZVRHOV FOREIGN KEY (CISLO_AKCE)
166  REFERENCES ROZVRHOVA_AKCE (CISLO_AKCE) ON DELETE CASCADE;
167
168  ALTER TABLE ZAPSAN
169  ADD CONSTRAINT FK_ZAPSAN_ZAPSAN_ROZVRHOV FOREIGN KEY (CISLO_AKCE)
170  REFERENCES ROZVRHOVA_AKCE (CISLO_AKCE);
171
```

```
172    ALTER TABLE ZAPSAN
173    ADD CONSTRAINT FK_ZAPSAN_ZAPSAN_STUDENT FOREIGN KEY (OSOBNI_CISLO)
174    REFERENCES STUDENT (OSOBNI_CISLO) ON DELETE CASCADE;
175
176    /*==============================================================*/
177    /* Indexy pro nepřekrývání výuky                                */
178    /*==============================================================*/
179
180    --indexuje pouze akce, kde SUDY je A
181    CREATE UNIQUE INDEX SUDY_AK ON ROZVRHOVA_AKCE
182    (CASE WHEN SUDY='A' THEN AKADEMICKY_ROK ELSE NULL END,
183     CASE WHEN SUDY='A' THEN SEMESTR ELSE NULL END,
184     CASE WHEN SUDY='A' THEN DEN ELSE NULL END,
185     CASE WHEN SUDY='A' THEN ZACATEK_VYUKY ELSE NULL END,
186     CASE WHEN SUDY='A' THEN MISTNOST ELSE NULL END);
187
188    --indexuje pouze akce, kde LICHY je A
189    CREATE UNIQUE INDEX LICHY_AK ON ROZVRHOVA_AKCE
190    (CASE WHEN LICHY='A' THEN AKADEMICKY_ROK ELSE NULL END,
191     CASE WHEN LICHY='A' THEN SEMESTR ELSE NULL END,
192     CASE WHEN LICHY='A' THEN DEN ELSE NULL END,
193     CASE WHEN LICHY='A' THEN ZACATEK_VYUKY ELSE NULL END,
194     CASE WHEN LICHY='A' THEN MISTNOST ELSE NULL END);
```

8 Plánování výuky – varianta B

8.1 Účel úlohy

Tato úloha ilustruje zejména následující problémy/jevy:

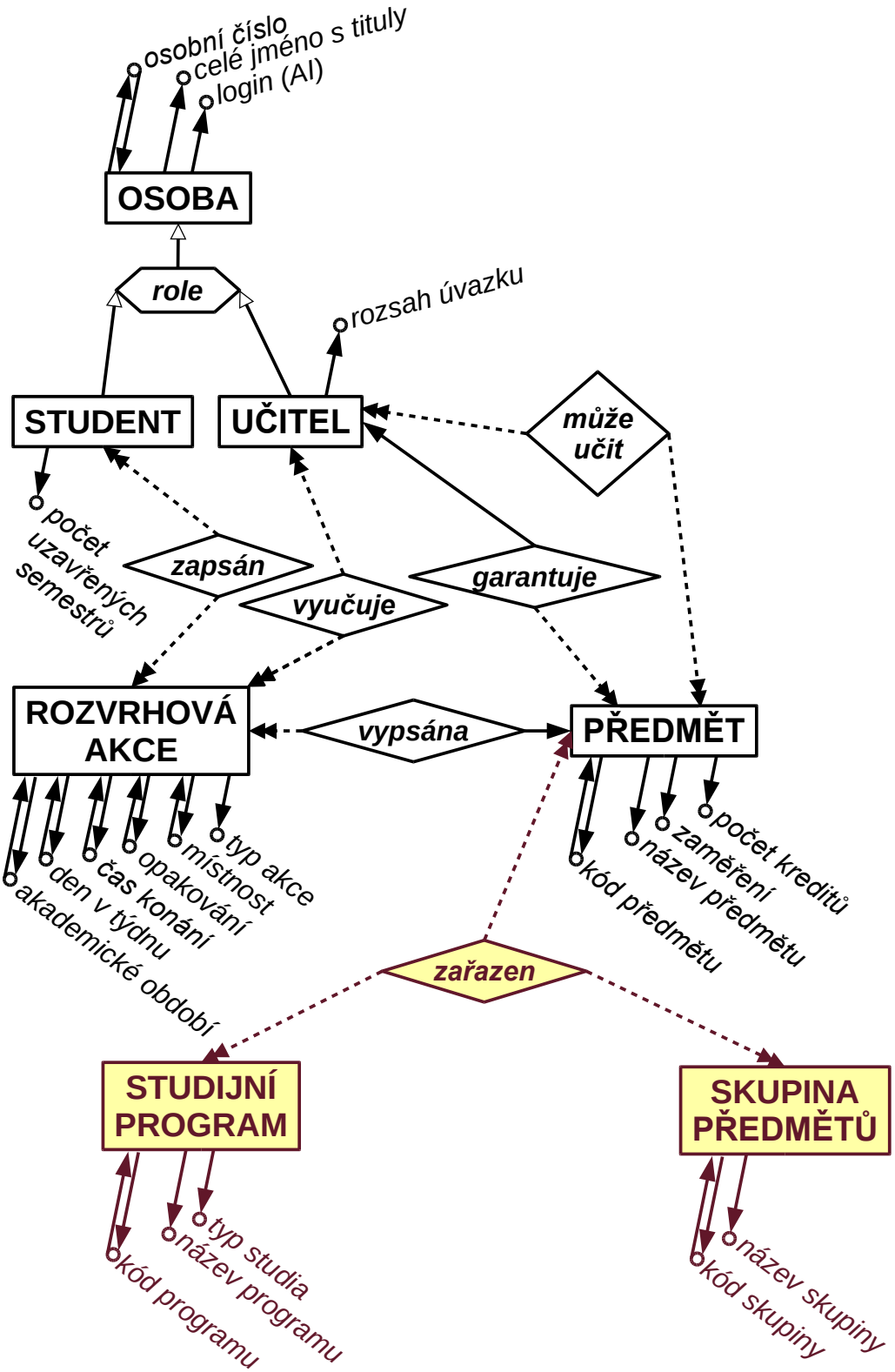
- Změna návrhu databáze na základě rozšířených požadavků.
- N-ární vztah.

8.2 Popis zvolené výseče světa – zadání

Tato úloha je rozšířením úlohy *Plánování výuky* – varianta A. Nad rámec varianty A je požadováno evidovat studijní programy. Eviduje se kód programu, název programu a typ studia (bakalářské, navazující magisterské). Předměty jsou zařazovány do studijních programů v rámci skupin předmětů (oborově povinné předměty, oborově volitelné předměty, celoškolsky volně volitelné předměty apod.). Je tedy třeba evidovat, v rámci jaké skupiny je předmět zařazen do studijního programu.

V databázi mohou být evidovány i předměty, které doposud nebyly zařazeny do žádného studijního programu, např. protože jsou teprve v přípravě. V každém semestru každého akademického roku musí být do bakalářského studijního programu zařazeny předměty alespoň v rozsahu 180 kreditů, do navazujícího magisterského studijního programu musí být zařazeny předměty alespoň v rozsahu 120 kreditů.

8.3 Konceptuální schéma

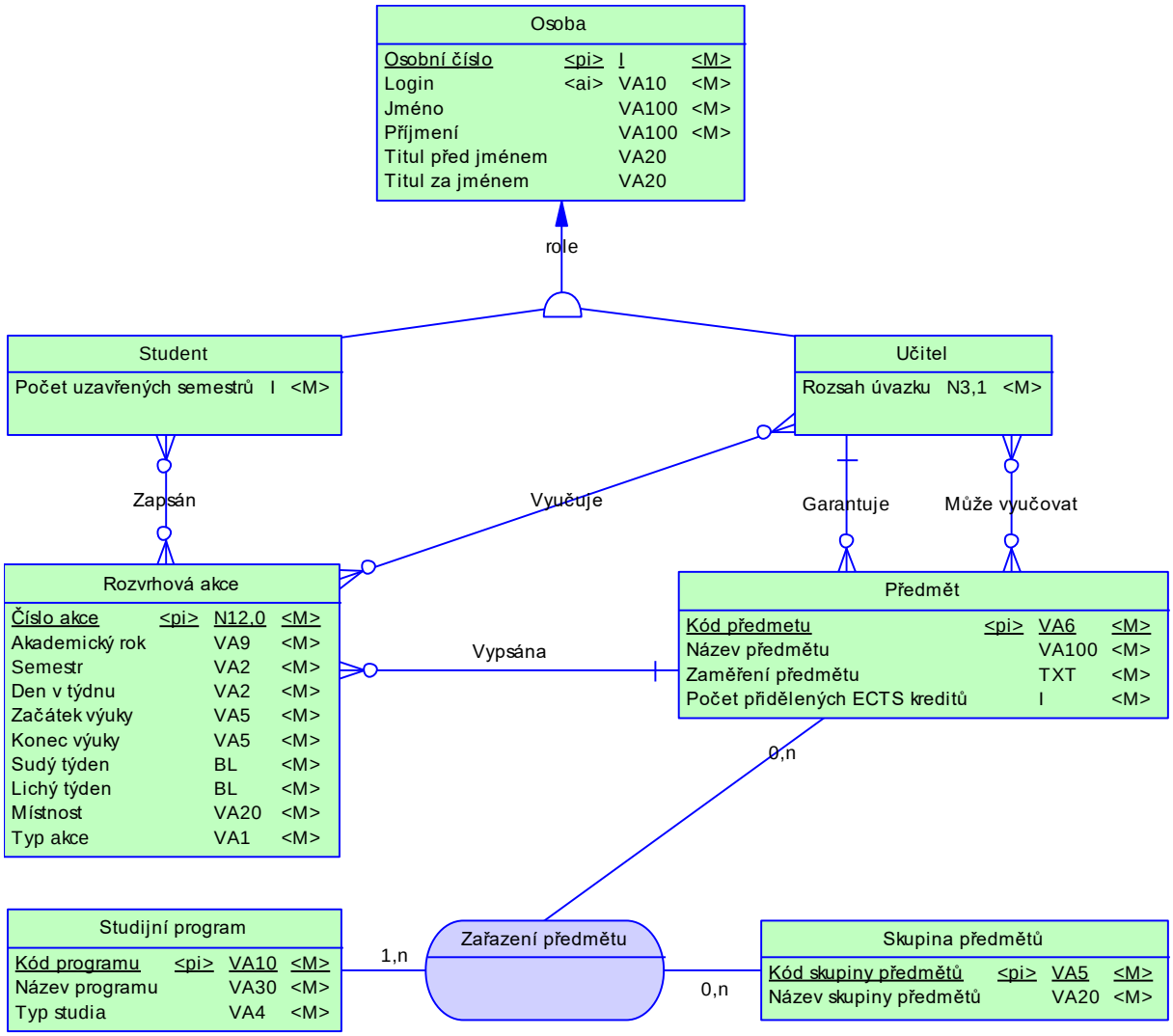


Obrázek 55: Plánování výuky B – konceptuální schéma, zdroj: autoři

Ternární vztah *zařazen* je na obrázku 55 zakreslen s nepovinnými vazbami na propojené entitní množiny, neboť otázka vyjádření povinnosti je u n-árních vztahů složitá, a zde ji zanedbáváme. Podobně je tomu i s kardinalitou n-árního vztahu, kde je obecně více možností. Zde slovně vyjádřeme, že předmět může být zařazen do více studijních programů a v nich i do více skupin.

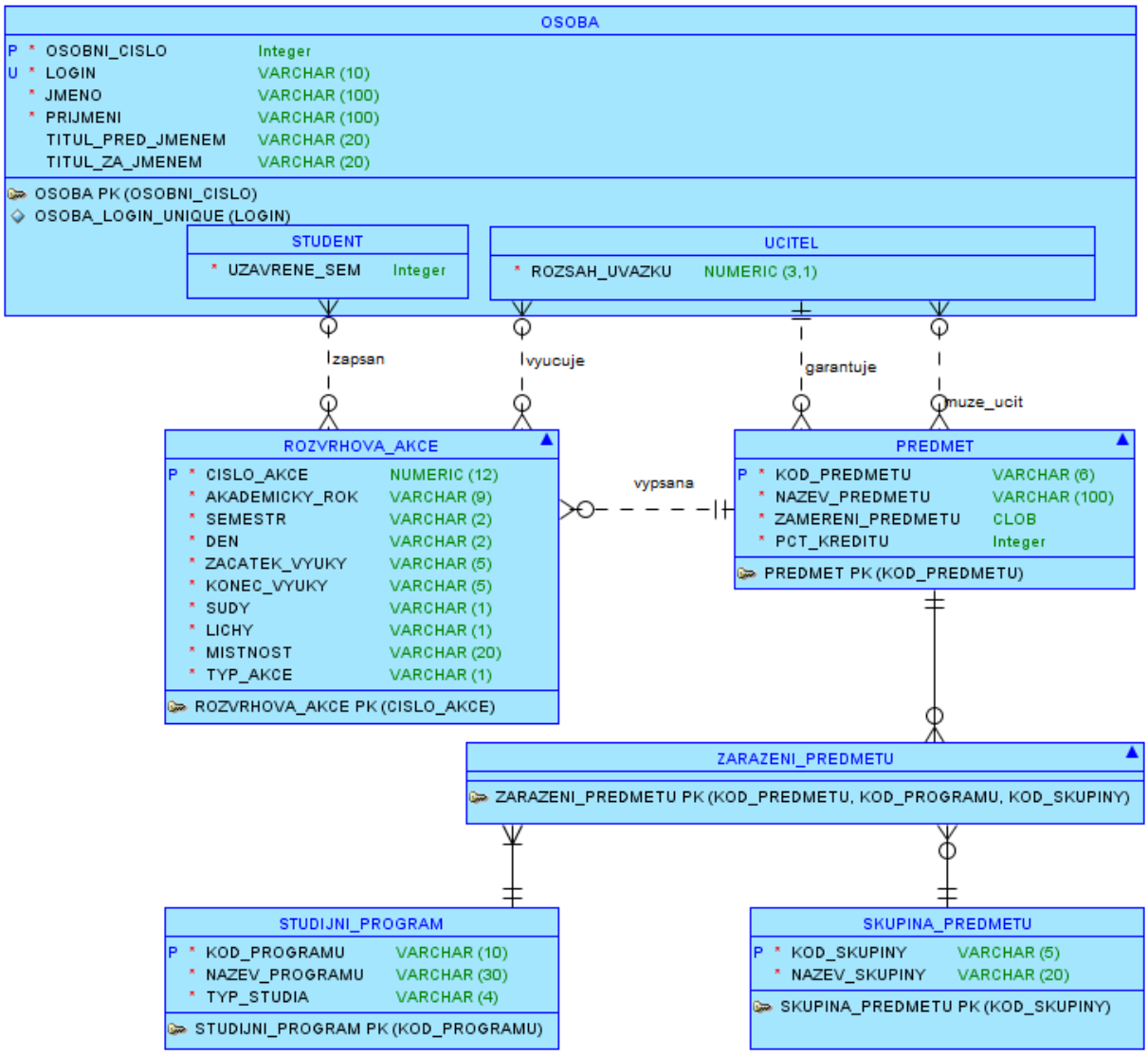
8.4 Konceptuální datový model

8.4.1 KDM v nástroji Power Designer



Obrázek 56: Plánování výuky B – konceptuální datový model (Power Designer), zdroj: autoři

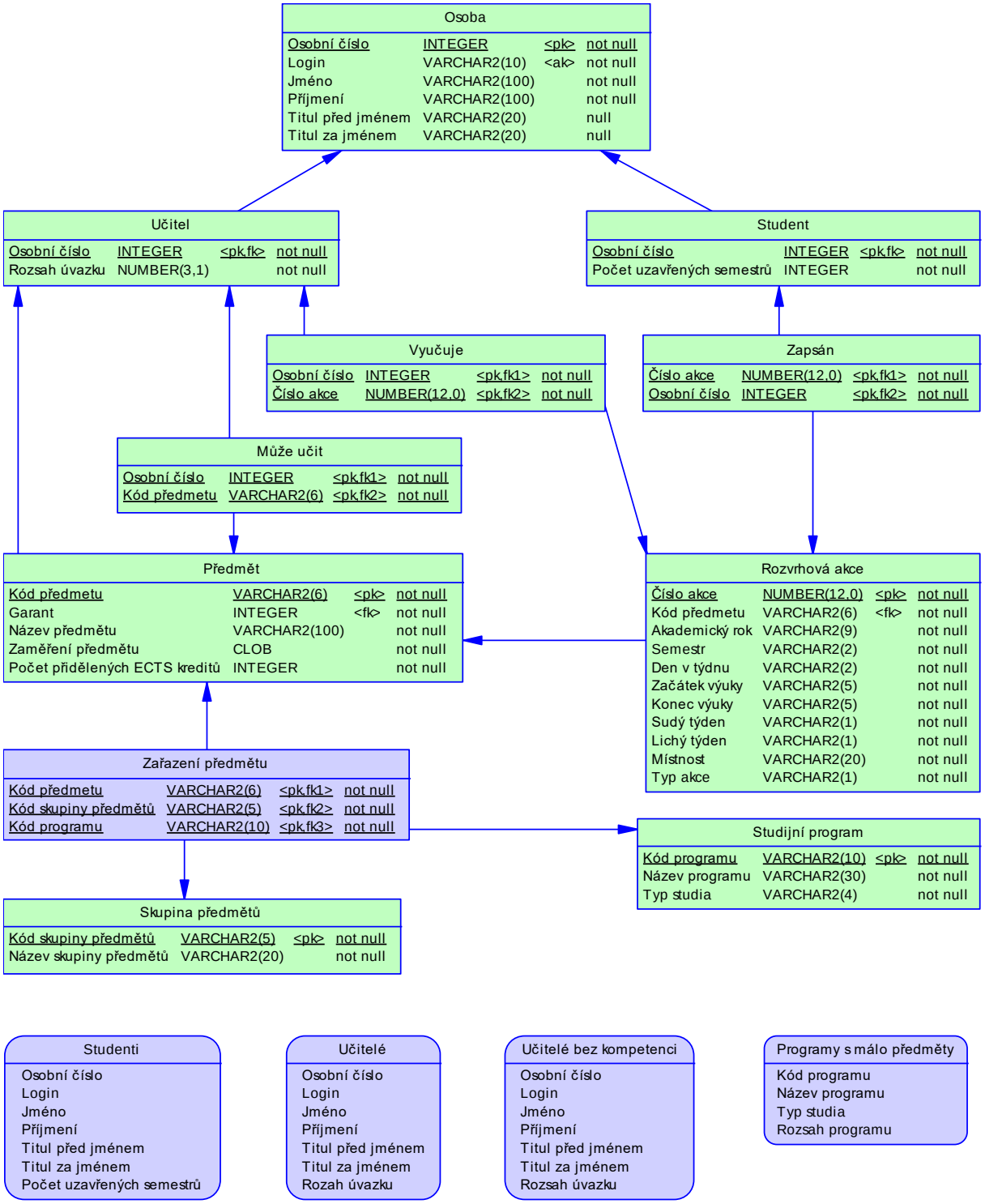
8.4.2 KDM v nástroji Data Modeler



Obrázek 57: Plánování výuky B – konceptuální datový model (Data Modeler), zdroj: autoři

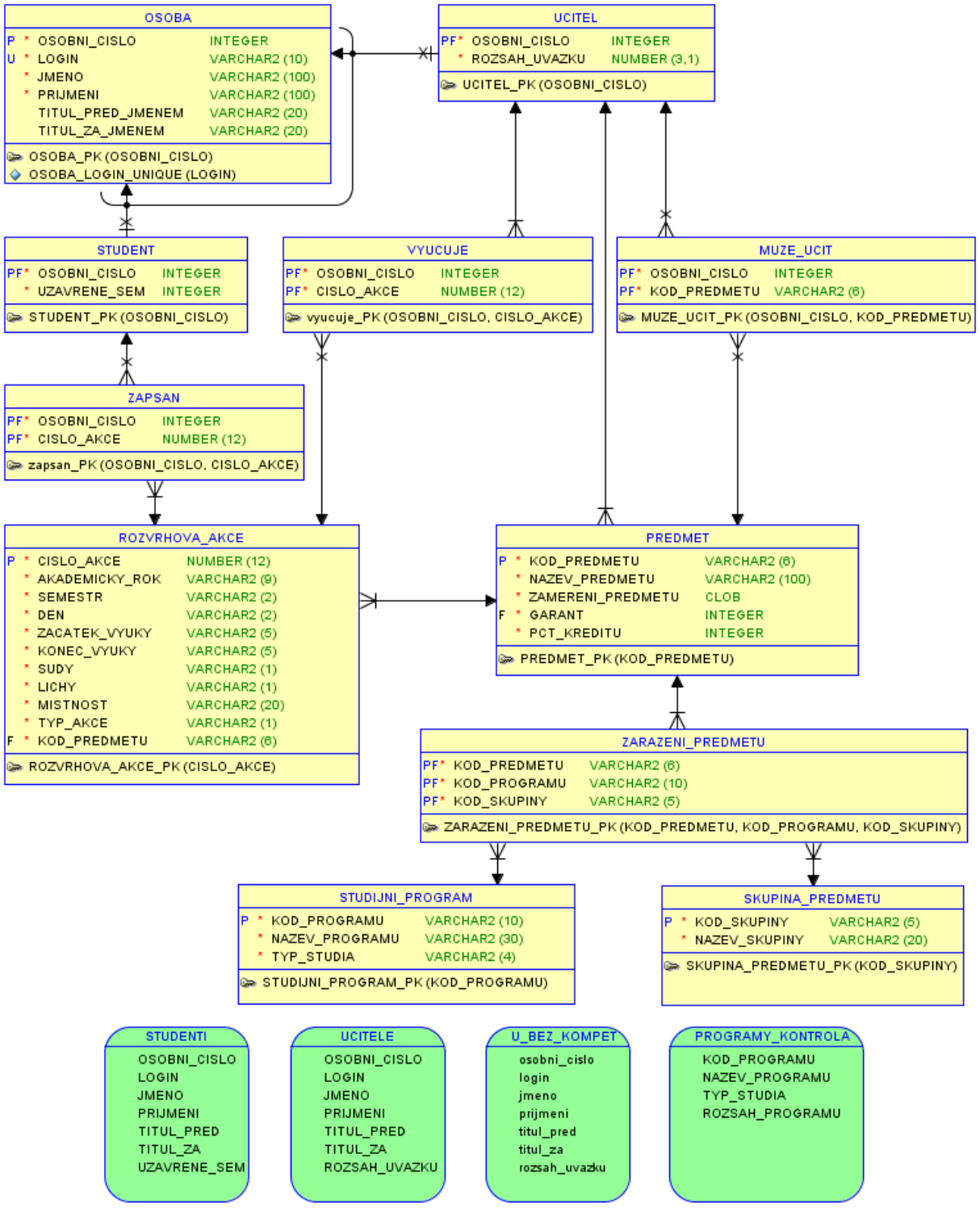
8.5 Fyzický datový model

8.5.1 FDM v nástroji Power Designer



Obrázek 58: Plánování výuky B – fyzický datový model (Power Designer), zdroj: autoři

8.5.2 FDM v nástroji Data Modeler



Obrázek 59: Plánování výuky B – fyzický datový model (Data Modeler), zdroj: autoři

8.6 Komentář k návrhu

Konceptuální schéma na obrázku 55 je rozšířením konceptuálního schématu na obrázku 50 o entitní množiny *studijní program* a *skupina předmětů*, a o ternární vztah *zařazen* vyjadřující zařazení předmětu do studijního programu v rámci určité skupiny předmětů. Nové požadavky v zadání této úlohy lze tedy řešit pouze přidáním nových databázových objektů.

Primární klíč tabulky *zařazení předmětů* je třeba řešit obezřetně tak, aby skutečně odpovídal pravidlům v dané výseči světa. Volba v našem řešení odpovídá tomu, že předmět může být do daného studijního programu zařazen do více skupin.

Požadavek, aby bakalářské programy v celkovém souhrnu nabízely předměty za alespoň 180 kreditů a magisterské programy za 120 kreditů v každém semestru každého akademického roku, kdy tyto programy běží, je ověřováno pomocí pohledu *Programy s málo předměty*, jenž hledá programy dané podmínce nevyhovující.

8.7 Příkazy pro vytvoření relačních tabulek a dalších objektů

Výpis 7: Plánování výuky B – SQL skript, zdroj: autoři

```
/*==============================================================*/
/* Tabulka: MUZE_UCIT                                           */
/*==============================================================*/
CREATE TABLE MUZE_UCIT  (
OSOBNI_CISLO        INTEGER                       NOT NULL,
KOD_PREDMETU        VARCHAR2(6)                   NOT NULL,
CONSTRAINT PK_MUZE_UCIT PRIMARY KEY (OSOBNI_CISLO, KOD_PREDMETU)
);

/*==============================================================*/
/* Tabulka: OSOBA                                               */
/*==============================================================*/
CREATE TABLE OSOBA  (
OSOBNI_CISLO         INTEGER                       NOT NULL,
LOGIN                VARCHAR2(10)                  NOT NULL,
JMENO                VARCHAR2(100)                 NOT NULL,
PRIJMENI             VARCHAR2(100)                 NOT NULL,
TITUL_PRED           VARCHAR2(20),
TITUL_ZA             VARCHAR2(20),
CONSTRAINT PK_OSOBA PRIMARY KEY (OSOBNI_CISLO),
CONSTRAINT AK_LOGIN_AI_OSOBA UNIQUE (LOGIN)
);

```

```
24  /*==============================================================*/
25  /* Tabulka: PREDMET                                             */
26  /*==============================================================*/
27  CREATE TABLE PREDMET  (
28  KOD_PREDMETU         VARCHAR2(6)                    NOT NULL,
29  GARANT               INTEGER                        NOT NULL,
30  NAZEV_PREDMETU       VARCHAR2(100)                  NOT NULL,
31  ZAMERENI_PREDMETU    CLOB                           NOT NULL,
32  PCT_KREDITU          INTEGER                        NOT NULL
33  CONSTRAINT CKC_PCT_KREDITU_PREDMET CHECK (PCT_KREDITU >= 0),
34  CONSTRAINT PK_PREDMET PRIMARY KEY (KOD_PREDMETU)
35  );
36
37  /*==============================================================*/
38  /* Tabulka: ROZVRHOVA_AKCE                                      */
39  /*==============================================================*/
40  CREATE TABLE ROZVRHOVA_AKCE  (
41  CISLO_AKCE           NUMBER(12,0)                   NOT NULL,
42  KOD_PREDMETU         VARCHAR2(6)                    NOT NULL,
43  AKADEMICKY_ROK       VARCHAR2(9)                    NOT NULL,
44  SEMESTR              VARCHAR2(2)                    NOT NULL
45  CONSTRAINT CKC_SEMESTR_ROZVRHOV CHECK (SEMESTR IN ('LS','ZS')),
46  DEN                  VARCHAR2(2)                    NOT NULL
47  CONSTRAINT CKC_DEN_ROZVRHOV CHECK (DEN IN
48  ('Po','Ut','St','Ct','Pa','So','Ne')),
49  ZACATEK_VYUKY        VARCHAR2(5)                    NOT NULL
50  CONSTRAINT CKC_ZACATEK_VYUKY_ROZVRHOV
51  CHECK (REGEXP_LIKE(ZACATEK_VYUKY,'^(0[0-9]|1[0-9]|2[0-3]):[0-5][0-9]$')),
52  KONEC_VYUKY          VARCHAR2(5)                    NOT NULL
53  CONSTRAINT CKC_KONEC_VYUKY_ROZVRHOV
54  CHECK (REGEXP_LIKE(KONEC_VYUKY,'^(0[0-9]|1[0-9]|2[0-3]):[0-5][0-9]$')),
55  SUDY                 VARCHAR2(1)                    DEFAULT 'A' NOT NULL
56  CONSTRAINT CKC_SUDY_ROZVRHOV CHECK (SUDY IN ('A','N')),
57  LICHY                VARCHAR2(1)                    DEFAULT 'A' NOT NULL
58  CONSTRAINT CKC_LICHY_ROZVRHOV CHECK (LICHY IN ('A','N')),
59  MISTNOST             VARCHAR2(20)                   NOT NULL,
60  TYP_AKCE             VARCHAR2(1)                    NOT NULL
61  CONSTRAINT CKC_TYP_AKCE_ROZVRHOV CHECK (TYP_AKCE IN ('P','C')),
62  CONSTRAINT PK_ROZVRHOVA_AKCE PRIMARY KEY (CISLO_AKCE),
63  CONSTRAINT CKT_ROZVRHOVA_AKCE CHECK (SUDY = 'A' OR LICHY = 'A')
64  );
65
```

```
66  /*==============================================================*/
67  /* Tabulka: SKUPINA_PREDMETU                                    */
68  /*==============================================================*/
69  CREATE TABLE SKUPINA_PREDMETU  (
70  KOD_SKUPINY          VARCHAR2(5)                    NOT NULL,
71  NAZEV_SKUPINY        VARCHAR2(20)                   NOT NULL,
72  CONSTRAINT PK_SKUPINA_PREDMETU PRIMARY KEY (KOD_SKUPINY)
73  );
74
75  /*==============================================================*/
76  /* Tabulka: STUDENT                                             */
77  /*==============================================================*/
78  CREATE TABLE STUDENT  (
79  OSOBNI_CISLO         INTEGER                        NOT NULL,
80  UZAVRENE_SEM         INTEGER                        NOT NULL
81  CONSTRAINT CKC_UZAVRENE_SEM_STUDENT CHECK (UZAVRENE_SEM >= 0),
82  CONSTRAINT PK_STUDENT PRIMARY KEY (OSOBNI_CISLO)
83  );
84
85  /*==============================================================*/
86  /* Tabulka: STUDIJNI_PROGRAM                                    */
87  /*==============================================================*/
88  CREATE TABLE STUDIJNI_PROGRAM  (
89  KOD_PROGRAMU         VARCHAR2(10)                   NOT NULL,
90  NAZEV_PROGRAMU       VARCHAR2(30)                   NOT NULL,
91  TYP_STUDIA           VARCHAR2(4)                    NOT NULL
92  CONSTRAINT CKC_TYP_STUDIA_STUDIJNI CHECK (TYP_STUDIA IN ('Bc','Mgr')),
93  CONSTRAINT PK_STUDIJNI_PROGRAM PRIMARY KEY (KOD_PROGRAMU)
94  );
95
96  /*==============================================================*/
97  /* Tabulka: UCITEL                                              */
98  /*==============================================================*/
99  CREATE TABLE UCITEL  (
100 OSOBNI_CISLO         INTEGER                        NOT NULL,
101 ROZSAH_UVAZKU        NUMBER(3,1)                    NOT NULL
102 CONSTRAINT CKC_ROZSAH_UVAZKU_UCITEL CHECK (ROZSAH_UVAZKU >= 0),
103 CONSTRAINT PK_UCITEL PRIMARY KEY (OSOBNI_CISLO)
104 );
105
106 /*==============================================================*/
107 /* Tabulka: VYUCUJE                                             */
108 /*==============================================================*/
109 CREATE TABLE VYUCUJE  (
110 OSOBNI_CISLO         INTEGER                        NOT NULL,
111 CISLO_AKCE           NUMBER(12,0)                   NOT NULL,
112 CONSTRAINT PK_VYUCUJE PRIMARY KEY (OSOBNI_CISLO, CISLO_AKCE)
113 );
```

```
114
115 /*==============================================================*/
116 /* Tabulka: ZAPSAN                                              */
117 /*==============================================================*/
118 CREATE TABLE ZAPSAN  (
119 CISLO_AKCE           NUMBER(12,0)                  NOT NULL,
120 OSOBNI_CISLO         INTEGER                       NOT NULL,
121 CONSTRAINT PK_ZAPSAN PRIMARY KEY (CISLO_AKCE, OSOBNI_CISLO)
122 );
123
124 /*==============================================================*/
125 /* Tabulka: ZARAZENI_PREDMETU                                   */
126 /*==============================================================*/
127 CREATE TABLE ZARAZENI_PREDMETU  (
128 KOD_PREDMETU         VARCHAR2(6)                   NOT NULL,
129 KOD_SKUPINY          VARCHAR2(5)                   NOT NULL,
130 KOD_PROGRAMU         VARCHAR2(10)                  NOT NULL,
131 CONSTRAINT PK_ZARAZENI_PREDMETU PRIMARY KEY (KOD_PREDMETU, KOD_SKUPINY,
132 KOD_PROGRAMU)
133 );
134
135 /*==============================================================*/
136 /* View: V_BEZ_KOMPETENCI                                       */
137 /*==============================================================*/
138 CREATE OR REPLACE VIEW V_BEZ_KOMPETENCI AS
139 SELECT
140 OSOBNI_CISLO, LOGIN, JMENO, PRIJMENI, TITUL_PRED, TITUL_ZA,
141 ROZSAH_UVAZKU
142 FROM
143 OSOBA JOIN UCITEL USING(OSOBNI_CISLO)
144 WHERE
145 OSOBNI_CISLO NOT IN (SELECT OSOBNI_CISLO FROM MUZE_UCIT)
146 AND OSOBNI_CISLO NOT IN (SELECT GARANT FROM PREDMET);
147
148 /*==============================================================*/
149 /* View: V_STUDENT                                              */
150 /*==============================================================*/
151 CREATE OR REPLACE VIEW V_STUDENT AS
152 SELECT
153 OSOBNI_CISLO, LOGIN, JMENO, PRIJMENI, TITUL_PRED, TITUL_ZA,
154 UZAVRENE_SEM
155 FROM
156 OSOBA JOIN STUDENT USING (OSOBNI_CISLO)
157 WITH CHECK OPTION;
158
```

```
159  /*==============================================================*/
160  /* View: V_UCITEL                                               */
161  /*==============================================================*/
162  CREATE OR REPLACE VIEW V_UCITEL AS
163  SELECT
164  OSOBNI_CISLO, LOGIN, JMENO, PRIJMENI, TITUL_PRED, TITUL_ZA,
165  ROZSAH_UVAZKU
166  FROM
167  OSOBA JOIN UCITEL USING(OSOBNI_CISLO)
168  WITH CHECK OPTION;
169
170  /*==============================================================*/
171  /* View: V_PROGRAMY_KONTROLA                                    */
172  /*==============================================================*/
173  CREATE OR REPLACE VIEW V_PROGRAMY_KONTROLA AS
174  SELECT
175  KOD_PROGRAMU, NAZEV_PROGRAMU, TYP_STUDIA,
176  SUM (PCT_KREDITU) AS ROZSAH_PROGRAMU
177  FROM
178  PREDMET JOIN ZARAZENI_PREDMETU USING (KOD_PREDMETU)
179  RIGHT JOIN STUDIJNI_PROGRAM USING (KOD_PROGRAMU)
180  GROUP BY
181  KOD_PROGRAMU, NAZEV_PROGRAMU, TYP_STUDIA
182  HAVING
183  (TYP_STUDIA = 'Bc' AND SUM (PCT_KREDITU) < 180)
184  OR
185  (TYP_STUDIA = 'Mgr' AND SUM (PCT_KREDITU) < 120);
186
187  ALTER TABLE MUZE_UCIT
188  ADD CONSTRAINT FK_MUZE_UCIT_UCITEL FOREIGN KEY (OSOBNI_CISLO)
189  REFERENCES UCITEL (OSOBNI_CISLO) ON DELETE CASCADE;
190
191  ALTER TABLE MUZE_UCIT
192  ADD CONSTRAINT FK_MUZE_UCIT_PREDMET FOREIGN KEY (KOD_PREDMETU)
193  REFERENCES PREDMET (KOD_PREDMETU) ON DELETE CASCADE;
194
195  ALTER TABLE PREDMET
196  ADD CONSTRAINT FK_PREDMET_GARANTUJE_UCITEL FOREIGN KEY (GARANT)
197  REFERENCES UCITEL (OSOBNI_CISLO);
198
199  ALTER TABLE ROZVRHOVA_AKCE
200  ADD CONSTRAINT FK_ROZVRHOV_VYUKA_PREDMET FOREIGN KEY (KOD_PREDMETU)
201  REFERENCES PREDMET (KOD_PREDMETU);
202
203  ALTER TABLE STUDENT
204  ADD CONSTRAINT FK_STUDENT_ROLE_OSOBA FOREIGN KEY (OSOBNI_CISLO)
205  REFERENCES OSOBA (OSOBNI_CISLO) ON DELETE CASCADE;
206
```

```
207    ALTER TABLE UCITEL
208    ADD CONSTRAINT FK_UCITEL_ROLE_OSOBA FOREIGN KEY (OSOBNI_CISLO)
209    REFERENCES OSOBA (OSOBNI_CISLO) ON DELETE CASCADE;
210
211    ALTER TABLE VYUCUJE
212    ADD CONSTRAINT FK_VYUCUJE_VYUCUJE_UCITEL FOREIGN KEY (OSOBNI_CISLO)
213    REFERENCES UCITEL (OSOBNI_CISLO);
214
215    ALTER TABLE VYUCUJE
216    ADD CONSTRAINT FK_VYUCUJE_VYUCUJE_ROZVRHOV FOREIGN KEY (CISLO_AKCE)
217    REFERENCES ROZVRHOVA_AKCE (CISLO_AKCE) ON DELETE CASCADE;
218
219    ALTER TABLE ZAPSAN
220    ADD CONSTRAINT FK_ZAPSAN_ZAPSAN_ROZVRHOV FOREIGN KEY (CISLO_AKCE)
221    REFERENCES ROZVRHOVA_AKCE (CISLO_AKCE);
222
223    ALTER TABLE ZAPSAN
224    ADD CONSTRAINT FK_ZAPSAN_ZAPSAN_STUDENT FOREIGN KEY (OSOBNI_CISLO)
225    REFERENCES STUDENT (OSOBNI_CISLO) ON DELETE CASCADE;
226
227    ALTER TABLE ZARAZENI_PREDMETU
228    ADD CONSTRAINT FK_ZARAZENI_ZARAZENI__PREDMET FOREIGN KEY (KOD_PREDMETU)
229    REFERENCES PREDMET (KOD_PREDMETU);
230
231    ALTER TABLE ZARAZENI_PREDMETU
232    ADD CONSTRAINT FK_ZARAZENI_ZARAZENI__SKUPINA_ FOREIGN KEY (KOD_SKUPINY)
233    REFERENCES SKUPINA_PREDMETU (KOD_SKUPINY);
234
235    ALTER TABLE ZARAZENI_PREDMETU
236    ADD CONSTRAINT FK_ZARAZENI_ZARAZENI__STUDIJNI FOREIGN KEY (KOD_PROGRAMU)
237    REFERENCES STUDIJNI_PROGRAM (KOD_PROGRAMU);
238
239    /*============================================================*/
240    /* Indexy pro nepřekrývání výuky                              */
241    /*============================================================*/
242
243    --indexuje pouze akce, kde SUDY je A
244    CREATE UNIQUE INDEX SUDY_AK ON ROZVRHOVA_AKCE
245    (CASE WHEN SUDY='A' THEN AKADEMICKY_ROK ELSE NULL END,
246     CASE WHEN SUDY='A' THEN SEMESTR ELSE NULL END,
247     CASE WHEN SUDY='A' THEN DEN ELSE NULL END,
248     CASE WHEN SUDY='A' THEN ZACATEK_VYUKY ELSE NULL END,
249     CASE WHEN SUDY='A' THEN MISTNOST ELSE NULL END);
250
```

```
251    --indexuje pouze akce, kde LICHY je A
252    CREATE UNIQUE INDEX LICHY_AK ON ROZVRHOVA_AKCE
253    (CASE WHEN LICHY='A' THEN AKADEMICKY_ROK ELSE NULL END,
254     CASE WHEN LICHY='A' THEN SEMESTR ELSE NULL END,
255     CASE WHEN LICHY='A' THEN DEN ELSE NULL END,
256     CASE WHEN LICHY='A' THEN ZACATEK_VYUKY ELSE NULL END,
257     CASE WHEN LICHY='A' THEN MISTNOST ELSE NULL END);
```

9 Plánování výuky – varianta C

9.1 Účel úlohy

Tato úloha ilustruje zejména následující problémy/jevy:

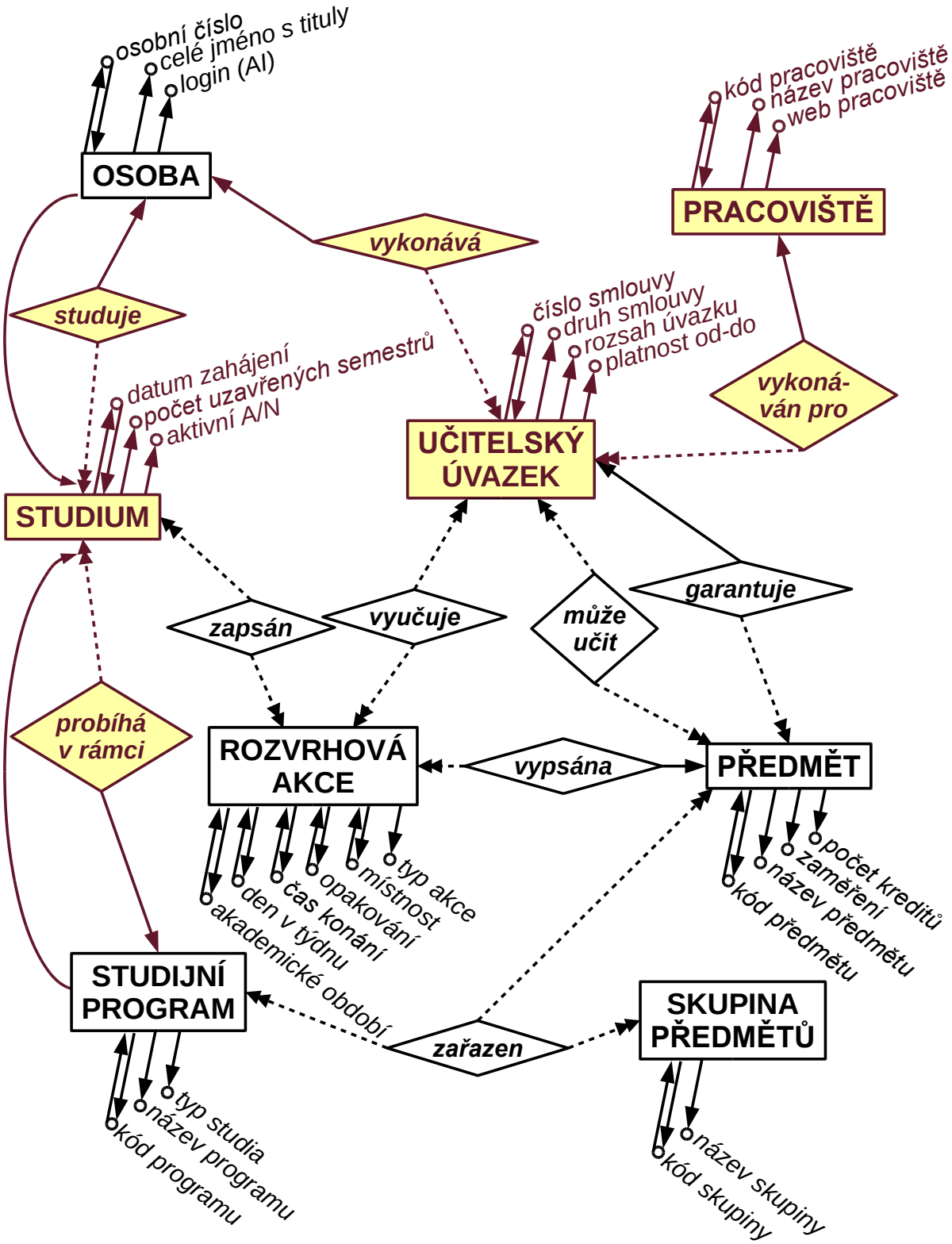
- Role osob jako doplnění k problematice generalizace/specializace a modelování podtypů.

9.2 Popis zvolené výseče světa – zadání

Tato úloha je úpravou úlohy *Plánování výuky* – varianta B. Osoba se stává studentem, pokud je přijata ke studiu v určitém studijním programu, přičemž jedna osoba může být přijata i do více studií – např. může studovat nejprve bakalářský program a po jeho dokončení navazující magisterský program, jedna osoba ale může studovat i více různých programů zároveň, nebo i tentýž program opakovaně, pokud některé z předešlých studií programu neskončilo úspěšným absolvováním programu. Je tedy třeba evidovat jednotlivá studia osob-studentů a platí, že v rámci jednoho studia je studován právě jeden program. Počet uzavřených semestrů je třeba sledovat pro každé studium osoby zvlášť. U studia je dále třeba evidovat datum zahájení studia. Dochází ke změně požadavku na evidenci výuky, kdy se na rozvrhové akce neregistrují studenti jako osoby, ale studenti v rámci svých studií.

Osoba se stává učitelem na základě pracovněprávního vztahu na jednom nebo více pracovištích vysoké školy. Pracovněprávní vztah vzniká na základě pracovní smlouvy, nebo dohody o pracovní činnosti, přičemž bez ohledu na právní formu bude mít smlouva vždy přiřazeno číslo smlouvy. Dále je třeba evidovat období platnosti smlouvy, přičemž pracovní smlouva může být uzavřena na dobu neurčitou, a rozsah úvazku vyjádřený počtem hodin týdně. Smlouva je vždy uzavřena s jedním pracovištěm, o kterém je evidován kód pracoviště, název pracoviště a adresa webu pracoviště. Dále dochází ke změně požadavku na evidenci garantování předmětů, schopnosti učitelů vyučovat předměty a výuky rozvrhových akcí učiteli. Nově je požadováno, aby tyto vztahy nebyly vázány na učitele jako osoby, ale na jejich pracovněprávní vztahy.

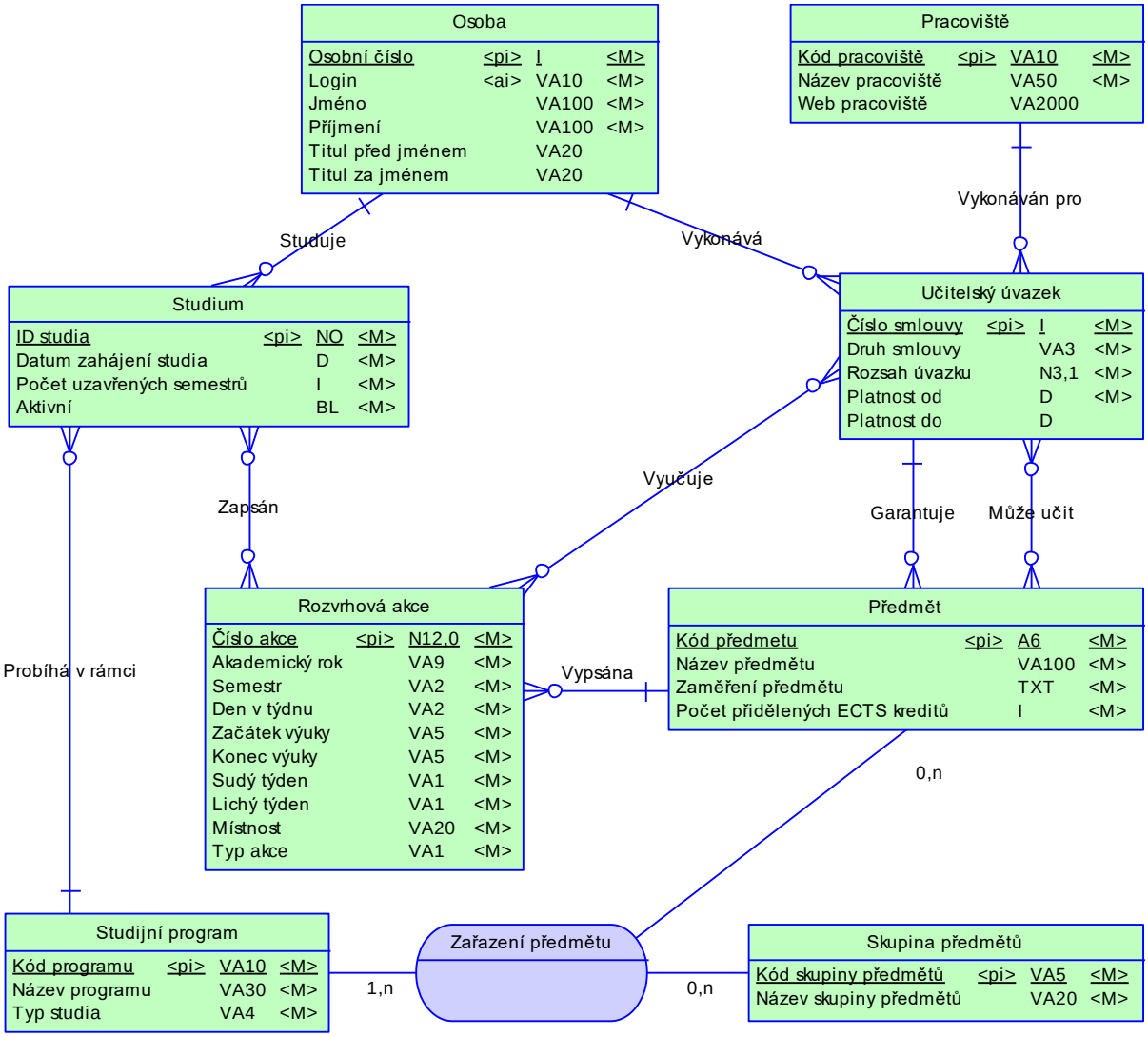
9.3 Konceptuální schéma



Obrázek 60: Plánování výuky C – konceptuální schéma, zdroj: autoři

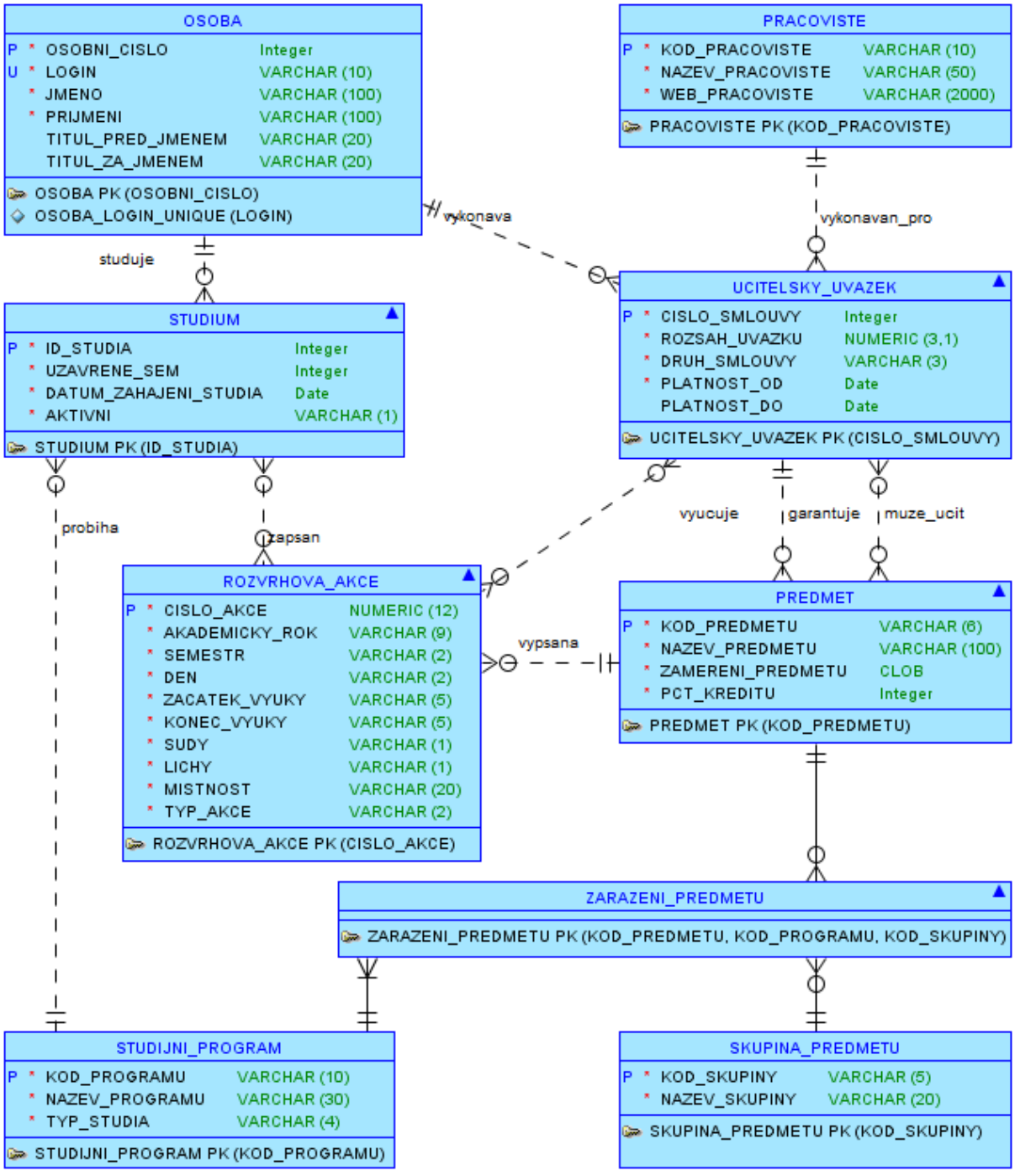
9.4 Konceptuální datový model

9.4.1 KDM v nástroji Power Designer



Obrázek 61: Plánování výuky C – konceptuální datový model (Power Designer), zdroj: autoři

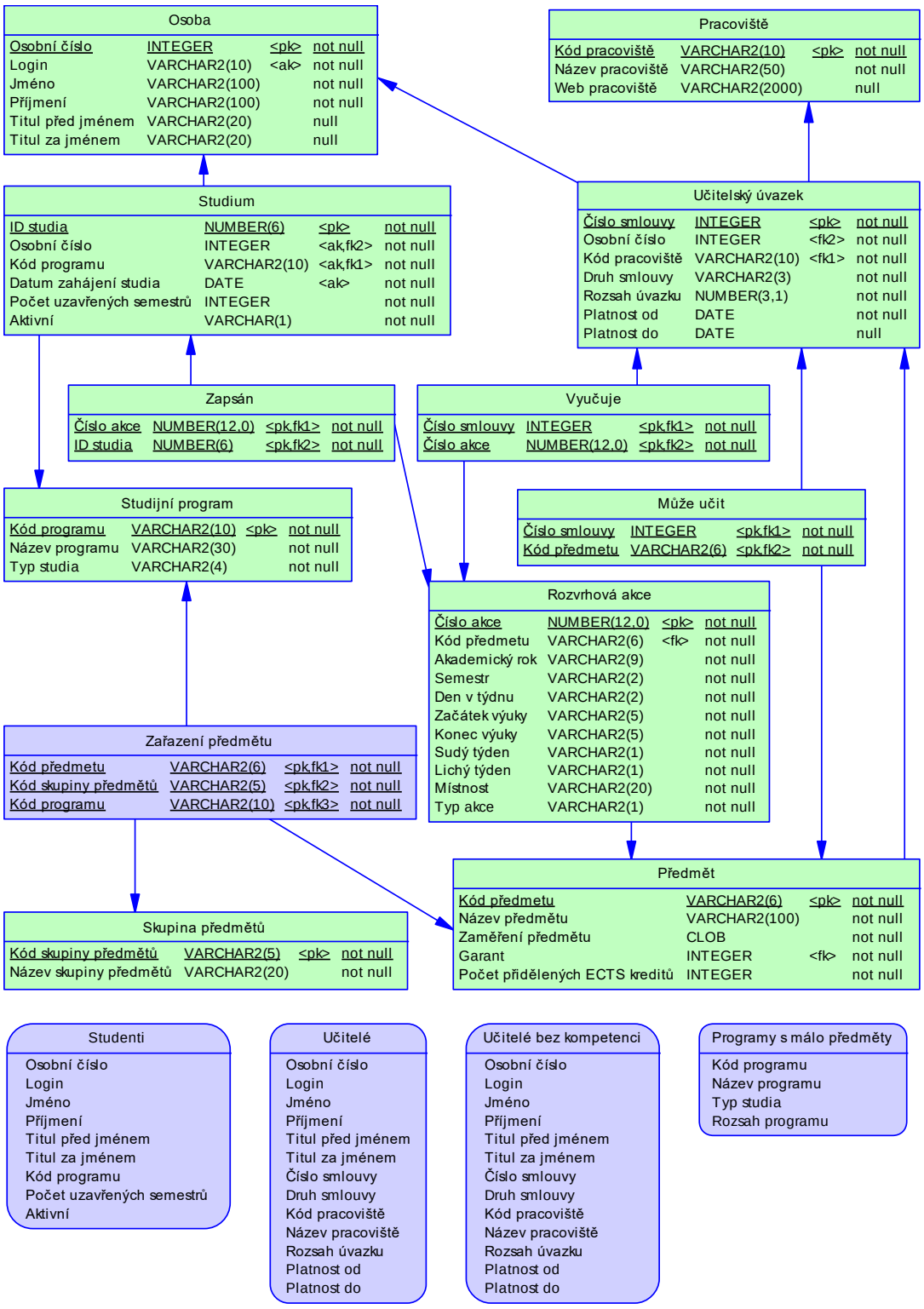
9.4.2 KDM v nástroji Data Modeler



Obrázek 62: Plánování výuky C – konceptuální datový model (Data Modeler), zdroj: autoři

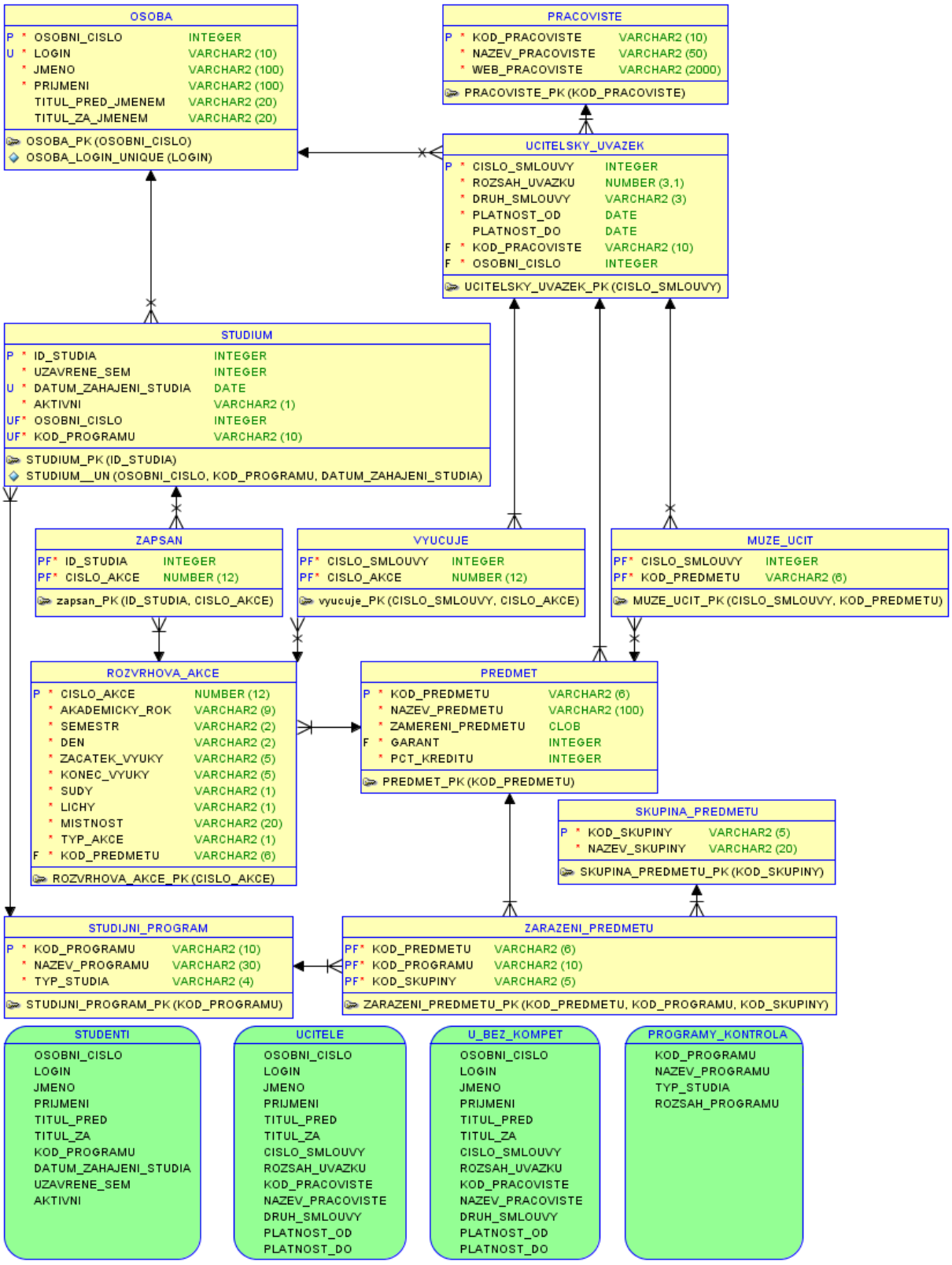
9.5 Fyzický datový model

9.5.1 FDM v nástroji Power Designer



Obrázek 63: Plánování výuky C – fyzický datový model (Power Designer), zdroj: autoři

9.5.2 FDM v nástroji Data Modeler



Obrázek 64: Plánování výuky C – fyzický datový model (Data Modeler), zdroj: autoři

9.6 Komentář k návrhu

Požadavky v zadání této úlohy už nelze realizovat pouze přidáním nových databázových objektů, protože v této úloze může mít osoba více různých studií, ovšem v daném studijním programu může mít nanejvýš jedno studium v daném čase (může jedno studium v daném programu ukončit a pak začít v témže programu další – jiné – studium). Vedle více studií může mít osoba i více učitelských úvazků. Na rozdíl od řešení variant uvedených v kapitolách 7 a 8 řešení uvedené v této kapitole neobsahuje vztah typu generalizace-specializace s označením *„role“*, který v přechozích variantách této úlohy vyjadřoval, že osoba je učitelem anebo studentem. Místo toho vystupují v řešení této úlohy entitní množiny *studium* a *učitelský úvazek*, přičemž související vztahy s označením „*studuje*“ a „*vykonává*“ představují vztahy 1:N. Tímto je vyjádřena výše uvedená skutečnost, že jedna osoba může mít jak vícero učitelských úvazků, tak více studií.

K přirozenému identifikátoru studia v konceptuálním schématu na obrázku 60 lze uvést, že vzhledem k tomu, že studium se začíná pro celý semestr, dvě různá studia stejné osoby ve stejném programu musí začínat v jiných semestrech, tedy určitě začínají v jiný den. Neukončené, běžící studium je označeno jako aktivní. Protože *učitelský úvazek* má primární identifikátor tvořený pouze jedním atributem, kterým je *číslo smlouvy*, byl pro *studium* zaveden umělý primární identifikátor *ID studia*. Důvodem je zejména konzistence návrhu primárních klíčů tabulek *studium* a *učitelský úvazek*, kde *ID studia* představuje analogii *čísla smlouvy*. Přirozený identifikátor studia je transformován do složeného alternativního identifikátoru.

9.7 Příkazy pro vytvoření relačních tabulek a dalších objektů

Výpis 8: Plánování výuky C – SQL skript, zdroj: autoři

```
1   CREATE SEQUENCE S_STUDIUM;
2
3   /*==============================================================*/
4   /* Tabulka: MUZE_UCIT                                           */
5   /*==============================================================*/
6   CREATE TABLE MUZE_UCIT  (
7   CISLO_SMLOUVY        INTEGER                       NOT NULL,
8   KOD_PREDMETU         VARCHAR2(6)                   NOT NULL,
9   CONSTRAINT PK_MUZE_UCIT PRIMARY KEY (CISLO_SMLOUVY, KOD_PREDMETU)
10  );
```

```
11
12  /*==============================================================*/
13  /* Tabulka: PRACOVISTE                                          */
14  /*==============================================================*/
15  CREATE TABLE PRACOVISTE  (
16  KOD_PRACOVISTE       VARCHAR2(10)                  NOT NULL,
17  NAZEV_PRACOVISTE     VARCHAR2(50)                  NOT NULL,
18  WEB_PRACOVISTE       VARCHAR2(2000),
19  CONSTRAINT PK_PRACOVISTE PRIMARY KEY (KOD_PRACOVISTE)
20  );
21
22  /*==============================================================*/
23  /* Tabulka: OSOBA                                               */
24  /*==============================================================*/
25  CREATE TABLE OSOBA  (
26  OSOBNI_CISLO         INTEGER                       NOT NULL,
27  LOGIN                VARCHAR2(10)                  NOT NULL,
28  JMENO                VARCHAR2(100)                 NOT NULL,
29  PRIJMENI             VARCHAR2(100)                 NOT NULL,
30  TITUL_PRED           VARCHAR2(20),
31  TITUL_ZA             VARCHAR2(20),
32  CONSTRAINT PK_OSOBA PRIMARY KEY (OSOBNI_CISLO),
33  CONSTRAINT AK_LOGIN_AI_OSOBA UNIQUE (LOGIN)
34  );
35
36  /*==============================================================*/
37  /* Tabulka: PREDMET                                             */
38  /*==============================================================*/
39  CREATE TABLE PREDMET  (
40  KOD_PREDMETU         VARCHAR2(6)                   NOT NULL,
41  GARANT               INTEGER                       NOT NULL,
42  NAZEV_PREDMETU       VARCHAR2(100)                 NOT NULL,
43  ZAMERENI_PREDMETU    CLOB                          NOT NULL,
44  PCT_KREDITU          INTEGER                       NOT NULL
45  CONSTRAINT CKC_PCT_KREDITU_PREDMET CHECK (PCT_KREDITU >= 0),
46  CONSTRAINT PK_PREDMET PRIMARY KEY (KOD_PREDMETU)
47  );
48
49  /*==============================================================*/
50  /* Tabulka: SKUPINA_PREDMETU                                    */
51  /*==============================================================*/
52  CREATE TABLE SKUPINA_PREDMETU  (
53  KOD_SKUPINY          VARCHAR2(5)                   NOT NULL,
54  NAZEV_SKUPINY        VARCHAR2(20)                  NOT NULL,
55  CONSTRAINT PK_SKUPINA_PREDMETU PRIMARY KEY (KOD_SKUPINY)
56  );
57
```

```
58  /*==============================================================*/
59  /* Tabulka: STUDIJNI_PROGRAM                                    */
60  /*==============================================================*/
61  CREATE TABLE STUDIJNI_PROGRAM  (
62  KOD_PROGRAMU         VARCHAR2(10)                    NOT NULL,
63  NAZEV_PROGRAMU       VARCHAR2(30)                    NOT NULL,
64  TYP_STUDIA           VARCHAR2(4)                     NOT NULL
65  CONSTRAINT CKC_TYP_STUDIA_STUDIJNI CHECK (TYP_STUDIA IN ('Bc','Mgr')),
66  CONSTRAINT PK_STUDIJNI_PROGRAM PRIMARY KEY (KOD_PROGRAMU)
67  );
68
69  /*==============================================================*/
70  /* Tabulka: ROZVRHOVA_AKCE                                      */
71  /*==============================================================*/
72  CREATE TABLE ROZVRHOVA_AKCE  (
73  CISLO_AKCE           NUMBER(12,0)                    NOT NULL,
74  KOD_PREDMETU         VARCHAR2(6)                     NOT NULL,
75  AKADEMICKY_ROK       VARCHAR2(9)                     NOT NULL,
76  SEMESTR              VARCHAR2(2)                     NOT NULL
77  CONSTRAINT CKC_SEMESTR_ROZVRHOV CHECK (SEMESTR IN ('LS','ZS')),
78  DEN                  VARCHAR2(2)                     NOT NULL
79  CONSTRAINT CKC_DEN_ROZVRHOV CHECK (DEN IN
80  ('Po','Ut','St','Ct','Pa','So','Ne')),
81  ZACATEK_VYUKY        VARCHAR2(5)                     NOT NULL
82  CONSTRAINT CKC_ZACATEK_VYUKY_ROZVRHOV
83  CHECK (REGEXP_LIKE(ZACATEK_VYUKY,'^(0[0-9]|1[0-9]|2[0-3]):[0-5][0-9]$')),
84  KONEC_VYUKY          VARCHAR2(5)                     NOT NULL
85  CONSTRAINT CKC_KONEC_VYUKY_ROZVRHOV
86  CHECK (REGEXP_LIKE(KONEC_VYUKY,'^(0[0-9]|1[0-9]|2[0-3]):[0-5][0-9]$')),
87  SUDY                 VARCHAR2(1)                     DEFAULT 'A' NOT NULL
88  CONSTRAINT CKC_SUDY_ROZVRHOV CHECK (SUDY IN ('A','N')),
89  LICHY                VARCHAR2(1)                     DEFAULT 'A' NOT NULL
90  CONSTRAINT CKC_LICHY_ROZVRHOV CHECK (LICHY IN ('A','N')),
91  MISTNOST             VARCHAR2(20)                    NOT NULL,
92  TYP_AKCE             VARCHAR2(1)                     NOT NULL
93  CONSTRAINT CKC_TYP_AKCE_ROZVRHOV CHECK (TYP_AKCE IN ('P','C')),
94  CONSTRAINT PK_ROZVRHOVA_AKCE PRIMARY KEY (CISLO_AKCE),
95  CONSTRAINT CKT_ROZVRHOVA_AKCE CHECK (SUDY = 'A' OR LICHY = 'A')
96  );
97
```

```
98  /*==============================================================*/
99  /* Tabulka: STUDIUM                                             */
100 /*==============================================================*/
101 CREATE TABLE STUDIUM  (
102 ID_STUDIA            NUMBER(6)                    NOT NULL,
103 OSOBNI_CISLO         INTEGER                      NOT NULL,
104 KOD_PROGRAMU         VARCHAR2(10)                 NOT NULL,
105 DATUM_ZAHAJENI_STUDIA DATE                        NOT NULL,
106 UZAVRENE_SEM         INTEGER                      NOT NULL
107 CONSTRAINT CKC_UZAVRENE_SEM_STUDIUM CHECK (UZAVRENE_SEM >= 0),
108 AKTIVNI              VARCHAR2(1)                  DEFAULT 'A' NOT NULL
109 CONSTRAINT CKC_AKTIVNI_STUDIUM CHECK (AKTIVNI IN ('A','N')),
110 CONSTRAINT PK_STUDIUM PRIMARY KEY (ID_STUDIA),
111 CONSTRAINT AK_STUDIUM_AK_STUDIUM UNIQUE (OSOBNI_CISLO, KOD_PROGRAMU,
112 DATUM_ZAHAJENI_STUDIA)
113 );
114
115 /*==============================================================*/
116 /* Tabulka: UCITELSKY_UVAZEK                                    */
117 /*==============================================================*/
118 CREATE TABLE UCITELSKY_UVAZEK  (
119 CISLO_SMLOUVY        INTEGER                      NOT NULL,
120 OSOBNI_CISLO         INTEGER                      NOT NULL,
121 KOD_PRACOVISTE       VARCHAR2(10)                 NOT NULL,
122 DRUH_SMLOUVY         VARCHAR2(3)                  NOT NULL
123 CONSTRAINT CKC_DRUH_SMLOUVY_UCITELSK CHECK (DRUH_SMLOUVY IN ('PRS','DPC')),
124 ROZSAH_UVAZKU        NUMBER(3,1)                  NOT NULL
125 CONSTRAINT CKC_ROZSAH_UVAZKU_UCITELSK CHECK (ROZSAH_UVAZKU >= 0),
126 PLATNOST_OD          DATE                         NOT NULL,
127 PLATNOST_DO          DATE,
128 CONSTRAINT PK_UCITELSKY_UVAZEK PRIMARY KEY (CISLO_SMLOUVY),
129 CONSTRAINT CKT_UCITELSKY_UVAZEK
130 CHECK (PLATNOST_DO IS NULL OR PLATNOST_OD < PLATNOST_DO)
131 );
132
133 /*==============================================================*/
134 /* Tabulka: VYUCUJE                                             */
135 /*==============================================================*/
136 CREATE TABLE VYUCUJE  (
137 CISLO_SMLOUVY        INTEGER                      NOT NULL,
138 CISLO_AKCE           NUMBER(12,0)                 NOT NULL,
139 CONSTRAINT PK_VYUCUJE PRIMARY KEY (CISLO_SMLOUVY, CISLO_AKCE)
140 );
141
```

```
142  /*==============================================================*/
143  /* Tabulka: ZAPSAN                                              */
144  /*==============================================================*/
145  CREATE TABLE ZAPSAN  (
146  CISLO_AKCE           NUMBER(12,0)                  NOT NULL,
147  ID_STUDIA            NUMBER(6)                     NOT NULL,
148  CONSTRAINT PK_ZAPSAN PRIMARY KEY (CISLO_AKCE, ID_STUDIA)
149  );
150
151  /*==============================================================*/
152  /* Tabulka: ZARAZENI_PREDMETU                                   */
153  /*==============================================================*/
154  CREATE TABLE ZARAZENI_PREDMETU  (
155  KOD_PREDMETU         VARCHAR2(6)                   NOT NULL,
156  KOD_SKUPINY          VARCHAR2(5)                   NOT NULL,
157  KOD_PROGRAMU         VARCHAR2(10)                  NOT NULL,
158  CONSTRAINT PK_ZARAZENI_PREDMETU PRIMARY KEY (KOD_PREDMETU, KOD_SKUPINY,
159  KOD_PROGRAMU)
160  );
161
162  /*==============================================================*/
163  /* View: V_PROGRAMY_KONTROLA                                    */
164  /*==============================================================*/
165  CREATE OR REPLACE VIEW V_PROGRAMY_KONTROLA AS
166  SELECT
167  KOD_PROGRAMU, NAZEV_PROGRAMU, TYP_STUDIA,
168  SUM (PCT_KREDITU) AS ROZSAH_PROGRAMU
169  FROM
170  PREDMET JOIN ZARAZENI_PREDMETU USING (KOD_PREDMETU)
171  RIGHT JOIN STUDIJNI_PROGRAM USING (KOD_PROGRAMU)
172  GROUP BY
173  KOD_PROGRAMU, NAZEV_PROGRAMU,
174  TYP_STUDIA
175  HAVING
176  (TYP_STUDIA = 'Bc' AND SUM (PCT_KREDITU) < 180)
177  OR
178  (TYP_STUDIA = 'Mgr' AND SUM (PCT_KREDITU) < 120);
179
180  /*==============================================================*/
181  /* View: V_STUDENT                                              */
182  /*==============================================================*/
183  CREATE OR REPLACE VIEW V_STUDENT AS
184  SELECT
185  OSOBNI_CISLO, LOGIN, JMENO, PRIJMENI, TITUL_PRED, TITUL_ZA,
186  KOD_PROGRAMU, DATUM_ZAHAJENI_STUDIA, UZAVRENE_SEM, AKTIVNI
187  FROM
188  OSOBA JOIN STUDIUM USING (OSOBNI_CISLO)
189  WITH CHECK OPTION;
```

```
190
191    /*==============================================================*/
192    /* View: V_UCITEL                                               */
193    /*==============================================================*/
194    CREATE OR REPLACE VIEW V_UCITEL AS
195    SELECT
196    OSOBNI_CISLO, LOGIN, JMENO, PRIJMENI, TITUL_PRED, TITUL_ZA,
197    CISLO_SMLOUVY, DRUH_SMLOUVY, KOD_PRACOVISTE, NAZEV_PRACOVISTE,
198    ROZSAH_UVAZKU, PLATNOST_OD, PLATNOST_DO
199    FROM
200    OSOBA JOIN UCITELSKY_UVAZEK USING(OSOBNI_CISLO)
201          JOIN PRACOVISTE USING (KOD_PRACOVISTE)
202    WITH CHECK OPTION;
203
204    /*==============================================================*/
205    /* View: V_BEZ_KOMPETENCI                                       */
206    /*==============================================================*/
207    CREATE OR REPLACE VIEW V_BEZ_KOMPETENCI AS
208    SELECT
209    OSOBNI_CISLO, LOGIN, JMENO, PRIJMENI, TITUL_PRED, TITUL_ZA,
210    CISLO_SMLOUVY, DRUH_SMLOUVY, KOD_PRACOVISTE, NAZEV_PRACOVISTE,
211    ROZSAH_UVAZKU, PLATNOST_OD, PLATNOST_DO
212    FROM
213    V_UCITEL
214    WHERE
215    CISLO_SMLOUVY NOT IN (SELECT CISLO_SMLOUVY FROM MUZE_UCIT)
216    AND CISLO_SMLOUVY NOT IN (SELECT GARANT FROM PREDMET);
217
218    ALTER TABLE MUZE_UCIT
219    ADD CONSTRAINT FK_MUZE_UCIT_UVAZEK FOREIGN KEY (CISLO_SMLOUVY)
220    REFERENCES UCITELSKY_UVAZEK (CISLO_SMLOUVY) ON DELETE CASCADE;
221
222    ALTER TABLE MUZE_UCIT
223    ADD CONSTRAINT FK_MUZE_UCIT_PREDMET FOREIGN KEY (KOD_PREDMETU)
224    REFERENCES PREDMET (KOD_PREDMETU) ON DELETE CASCADE;
225
226    ALTER TABLE PREDMET
227    ADD CONSTRAINT FK_PREDMET_GARANTUJE_UCITELSK FOREIGN KEY (GARANT)
228    REFERENCES UCITELSKY_UVAZEK (CISLO_SMLOUVY);
229
230    ALTER TABLE ROZVRHOVA_AKCE
231    ADD CONSTRAINT FK_ROZVRHOV_VYUKA_PREDMET FOREIGN KEY (KOD_PREDMETU)
232    REFERENCES PREDMET (KOD_PREDMETU);
233
234    ALTER TABLE STUDIUM
235    ADD CONSTRAINT FK_STUDIUM_STUDUJICI_OSOBA FOREIGN KEY (OSOBNI_CISLO)
236    REFERENCES OSOBA (OSOBNI_CISLO) ON DELETE CASCADE;
237
```

```
238    ALTER TABLE STUDIUM
239    ADD CONSTRAINT FK_STUDIUM_V_RAMCI_STUDIJNI FOREIGN KEY (KOD_PROGRAMU)
240    REFERENCES STUDIJNI_PROGRAM (KOD_PROGRAMU);
241
242    ALTER TABLE UCITELSKY_UVAZEK
243    ADD CONSTRAINT FK_UCITELSK_ZAMESTNAN_OSOBA FOREIGN KEY (OSOBNI_CISLO)
244    REFERENCES OSOBA (OSOBNI_CISLO) ON DELETE CASCADE;
245
246    ALTER TABLE UCITELSKY_UVAZEK
247    ADD CONSTRAINT FK_UCITELSK_ZAMESTNAV_PRACOVIS FOREIGN KEY (KOD_PRACOVISTE)
248    REFERENCES PRACOVISTE (KOD_PRACOVISTE);
249
250    ALTER TABLE VYUCUJE
251    ADD CONSTRAINT FK_VYUCUJE_VYUCUJE_UCITELSK FOREIGN KEY (CISLO_SMLOUVY)
252    REFERENCES UCITELSKY_UVAZEK (CISLO_SMLOUVY);
253
254    ALTER TABLE VYUCUJE
255    ADD CONSTRAINT FK_VYUCUJE_VYUCUJE_ROZVRHOV FOREIGN KEY (CISLO_AKCE)
256    REFERENCES ROZVRHOVA_AKCE (CISLO_AKCE) ON DELETE CASCADE;
257
258    ALTER TABLE ZAPSAN
259    ADD CONSTRAINT FK_ZAPSAN_ZAPSAN_ROZVRHOV FOREIGN KEY (CISLO_AKCE)
260    REFERENCES ROZVRHOVA_AKCE (CISLO_AKCE);
261
262    ALTER TABLE ZAPSAN
263    ADD CONSTRAINT FK_ZAPSAN_ZAPSAN2_STUDIUM FOREIGN KEY (ID_STUDIA)
264    REFERENCES STUDIUM (ID_STUDIA) ON DELETE CASCADE;
265
266    ALTER TABLE ZARAZENI_PREDMETU
267    ADD CONSTRAINT FK_ZARAZENI_ZARAZENI__PREDMET FOREIGN KEY (KOD_PREDMETU)
268    REFERENCES PREDMET (KOD_PREDMETU);
269
270    ALTER TABLE ZARAZENI_PREDMETU
271    ADD CONSTRAINT FK_ZARAZENI_ZARAZENI__SKUPINA_ FOREIGN KEY (KOD_SKUPINY)
272    REFERENCES SKUPINA_PREDMETU (KOD_SKUPINY);
273
274    ALTER TABLE ZARAZENI_PREDMETU
275    ADD CONSTRAINT FK_ZARAZENI_ZARAZENI__STUDIJNI FOREIGN KEY (KOD_PROGRAMU)
276    REFERENCES STUDIJNI_PROGRAM (KOD_PROGRAMU);
277
```

```
278    /*=============================================================*/
279    /* Indexy pro nepřekrývání výuky                               */
280    /*=============================================================*/
281    --indexuje pouze akce, kde SUDY je A
282    CREATE UNIQUE INDEX SUDY_AK ON ROZVRHOVA_AKCE
283    (CASE WHEN SUDY='A' THEN AKADEMICKY_ROK ELSE NULL END,
284     CASE WHEN SUDY='A' THEN SEMESTR ELSE NULL END,
285     CASE WHEN SUDY='A' THEN DEN ELSE NULL END,
286     CASE WHEN SUDY='A' THEN ZACATEK_VYUKY ELSE NULL END,
287     CASE WHEN SUDY='A' THEN MISTNOST ELSE NULL END);
288
289    --indexuje pouze akce, kde LICHY je A
290    CREATE UNIQUE INDEX LICHY_AK ON ROZVRHOVA_AKCE
291    (CASE WHEN LICHY='A' THEN AKADEMICKY_ROK ELSE NULL END,
292     CASE WHEN LICHY='A' THEN SEMESTR ELSE NULL END,
293     CASE WHEN LICHY='A' THEN DEN ELSE NULL END,
294     CASE WHEN LICHY='A' THEN ZACATEK_VYUKY ELSE NULL END,
295     CASE WHEN LICHY='A' THEN MISTNOST ELSE NULL END);
```

10 Plemenné knihy

10.1 Účel úlohy

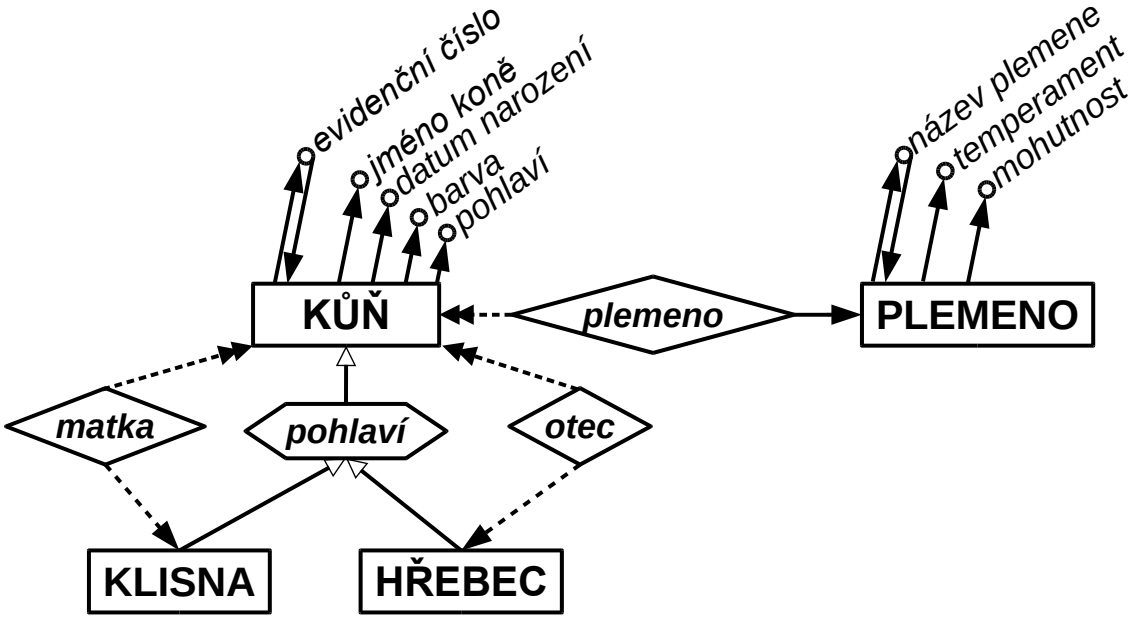
Tato úloha ilustruje zejména následující problémy/jevy:

- Hierarchické vztahy mezi entitními množinami v rámci jedné generalizace/specializace.

10.2 Popis zvolené výseče světa – zadání

Sdružení chovatelů koní chce evidovat původ koní, tj. který hřebec a která klisna jsou předky určitého koně. O každém koni je třeba evidovat jeho evidenční číslo, jméno, datum narození, barvu, pohlaví a plemeno (záznamy o původu koní určitého plemene tvoří tzv. plemennou knihu). O jednotlivých plemenech je třeba evidovat český název plemene, temperament (chladnokrevník, teplokrevník, plnokrevník, polokrevník) a mohutnost plemene (malé, střední, velké). Databázi je třeba navrhnout tak, aby otcem koně mohl být pouze hřebec, a matkou koně, aby mohla být pouze klisna.

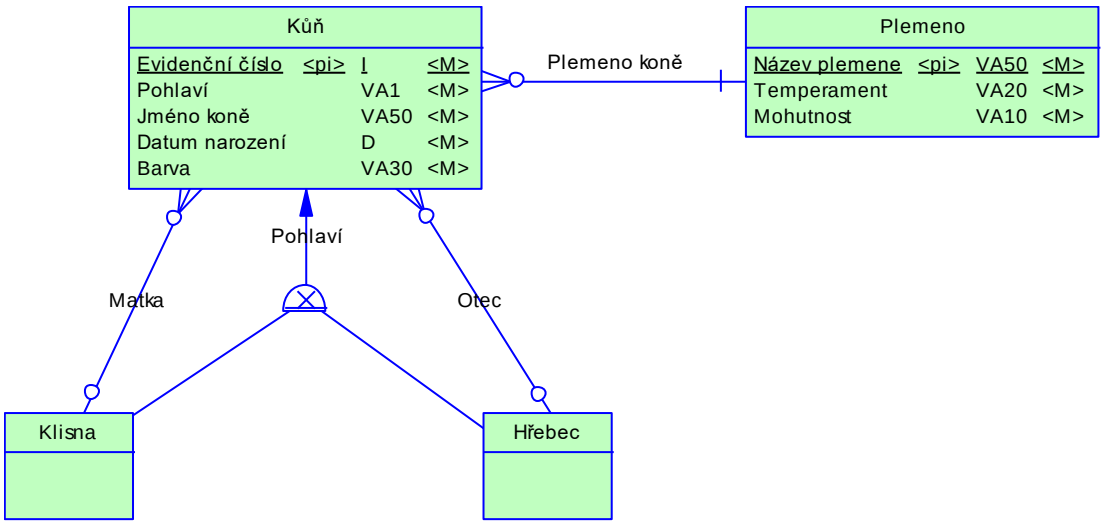
10.3 Konceptuální schéma



Obrázek 65: Plemenné knihy – konceptuální schéma, zdroj: autoři

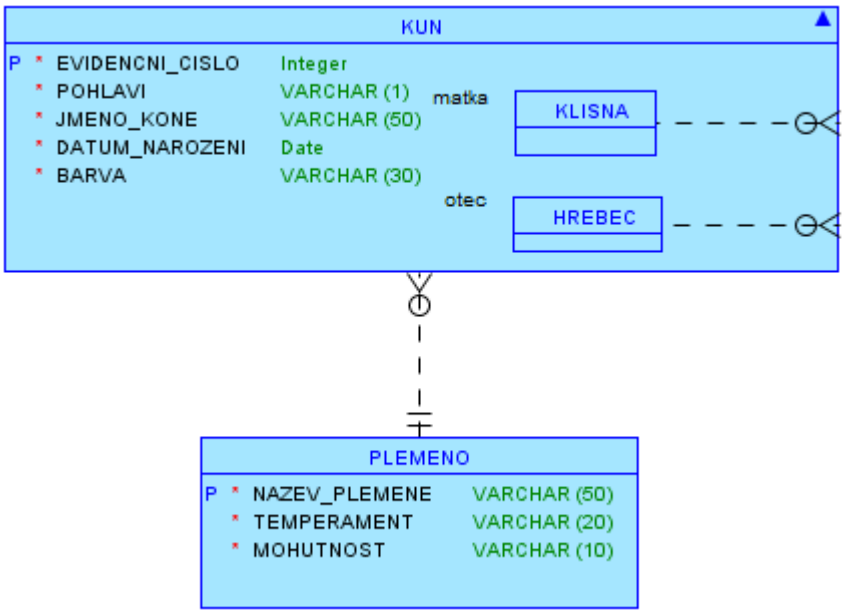
10.4 Konceptuální datový model

10.4.1 KDM v nástroji Power Designer



Obrázek 66: Plemenné knihy – konceptuální datový model (Power Designer), zdroj: autoři

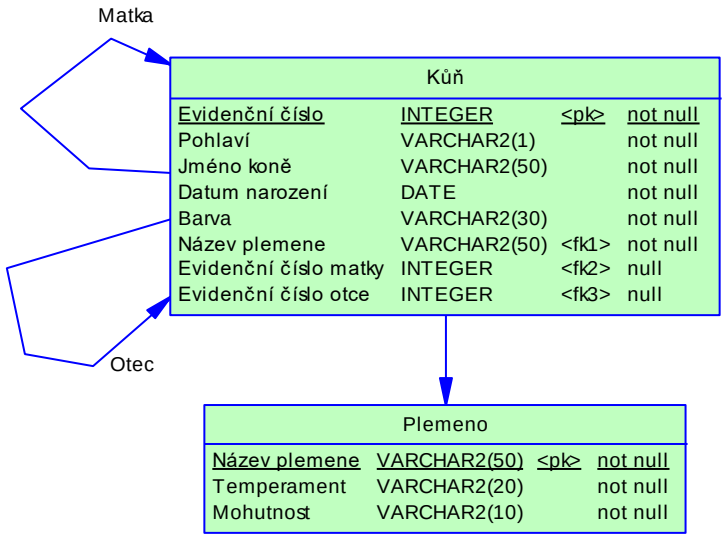
10.4.2 KDM v nástroji Data Modeler



Obrázek 67: Plemenné knihy – konceptuální datový model (Data Modeler), zdroj: autoři

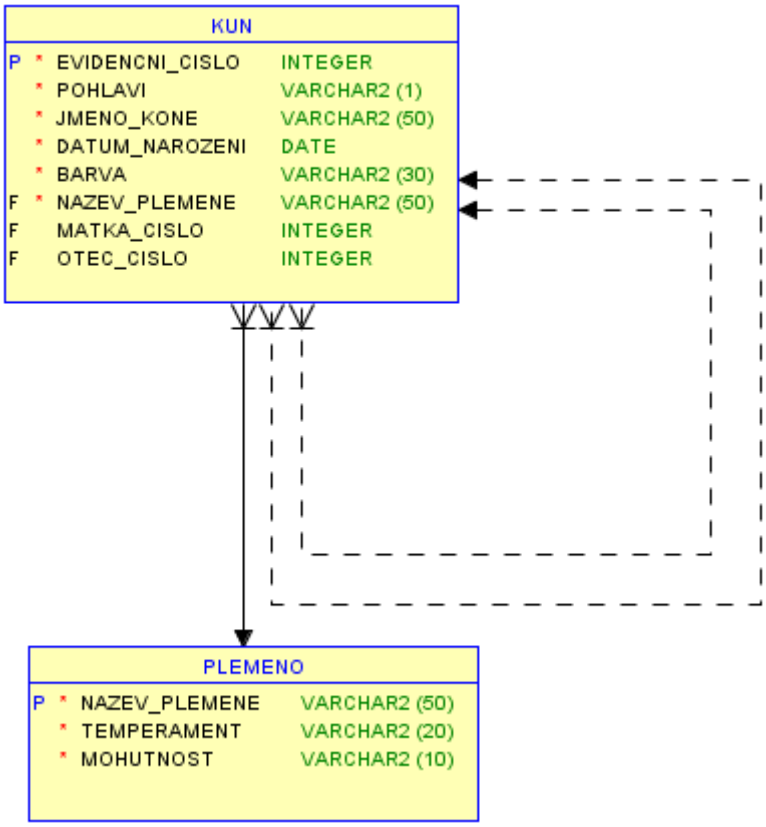
10.5 Fyzický datový model

10.5.1 FDM v nástroji Power Designer



Obrázek 68: Plemenné knihy – fyzický datový model (Power Designer), zdroj: autoři

10.5.2 FDM v nástroji Data Modeler



Obrázek 69: Plemenné knihy – fyzický datový model (Data Modeler), zdroj: autoři

10.6 Komentář k návrhu

Na základě zadání byly identifikovány entitní množiny *plemeno*, *kůň*, *hřebec* a *klisna* (viz obrázek 65). Vztah *pohlaví* je vztahem typu generalizace/specializace, ve kterém *kůň* představuje nadtyp a *hřebec* a *klisna* jeho výlučné a kompletní podtypy. Vyjádření tohoto členění je při řešení úlohy důležité pro správně zachycení vazeb *matka* a *otec*, protože pouze klisna může být matkou koně, a pouze hřebec může být otcem koně.

V rámci fyzického návrhu databáze byla generalizace/specializace řešena vytvořením jedné tabulky pro nadtyp a jeho podtypy, přičemž sloupec *pohlaví* slouží pro rozlišení pohlaví koní. Omezení `CKT_KUN` zajišťuje, aby kůň nemohl být sám sobě matkou ani otcem, aby jeden jedinec nemohl být zároveň matkou a otce jiného jedince. Pravidla, že pouze klisna může být matkou koně, a pouze hřebec otcem koně, a že se potomek nemůže narodit dříve, nežli rodič, jsou zajištěna pomocí podmínky typu `CHECK` deklarované na materializovaném pohledu `MVCK_RODIC_POHLAVI_VEK` (řádky 69-90 výpisu 9). Tento pohled je definován tak, aby bylo provedeno spojení řádků z tabulky *kůň* představujících potomka a rodiče za podmínek, které odporují výše uvedeným pravidlům. Podmínka `CK_RODIC_POHLAVI_VEK` zajišťuje, že materializovaný pohled nesmí obsahovat žádné řádky. Pokud by do databáze byly vloženy záznamy odporující pravidlům, výsledkem by byl nenulový počet řádků materializovaného pohledu. Při operaci `COMMIT` dojde k aktualizaci pohledu a validaci podmínky `CK_RODIC_POHLAVI_VEK`, která při vložení záznamů v rozporu s pravidly skončí neúspěchem a selháním transakce. Čili data v rozporu s pravidly nebude možné vložit.

10.7 Příkazy pro vytvoření relačních tabulek a dalších objektů

Výpis 9: Plemenné knihy – SQL skript, zdroj: autoři

```
/*==============================================================*/
/* Tabulka: PLEMENO                                             */
/*==============================================================*/
CREATE TABLE PLEMENO  (
   NAZEV_PLEMENE        VARCHAR2(50)                    NOT NULL,
   TEMPERAMENT          VARCHAR2(20)                    NOT NULL
      CONSTRAINT CKC_TEMPERAMENT_PLEMENO
        CHECK (TEMPERAMENT IN ('Chladnokrevnik', 'Teplokrevnik',
                               'Plnokrevnik', 'Polokrevnik')),
   MOHUTNOST            VARCHAR2(10)                    NOT NULL
      CONSTRAINT CKC_MOHUTNOST_PLEMENO
        CHECK (MOHUTNOST IN ('male','stredni','velke')),
   CONSTRAINT PK_PLEMENO PRIMARY KEY (NAZEV_PLEMENE)
);

/*==============================================================*/
/* Tabulka: KUN                                                 */
/*==============================================================*/
CREATE TABLE KUN  (
   EVIDENCNI_CISLO      INTEGER                         NOT NULL,
   POHLAVI              VARCHAR2(1)                     NOT NULL
      CONSTRAINT CKC_POHLAVI_KUN CHECK (POHLAVI IN ('H','K')),
   JMENO_KONE           VARCHAR2(50)                    NOT NULL,
   DATUM_NAROZENI       DATE                            NOT NULL,
   BARVA                VARCHAR2(30)                    NOT NULL,
   NAZEV_PLEMENE        VARCHAR2(50)                    NOT NULL,
   MATKA_CISLO          INTEGER,
   OTEC_CISLO           INTEGER,
   CONSTRAINT PK_KUN PRIMARY KEY (EVIDENCNI_CISLO),
   CONSTRAINT CKT_KUN
     CHECK ((MATKA_CISLO <> OTEC_CISLO)
            AND (EVIDENCNI_CISLO <> MATKA_CISLO)
            AND (EVIDENCNI_CISLO <> OTEC_CISLO))
);

/*==============================================================*/
/* Index: PLEMENO_KONE_FK                                       */
/*==============================================================*/
CREATE INDEX PLEMENO_KONE_FK ON KUN (
   NAZEV_PLEMENE ASC
);

```

```
43  /*==============================================================*/
44  /* Index: MATKA_FK                                              */
45  /*==============================================================*/
46  CREATE INDEX MATKA_FK ON KUN (
47     MATKA_CISLO ASC
48  );
49
50  /*==============================================================*/
51  /* Index: OTEC_FK                                               */
52  /*==============================================================*/
53  CREATE INDEX OTEC_FK ON KUN (
54     OTEC_CISLO ASC
55  );
56
57  ALTER TABLE KUN
58     ADD CONSTRAINT FK_KUN_MATKA_KUN FOREIGN KEY (MATKA_CISLO)
59        REFERENCES KUN (EVIDENCNI_CISLO);
60
61  ALTER TABLE KUN
62     ADD CONSTRAINT FK_KUN_OTEC_KUN FOREIGN KEY (OTEC_CISLO)
63        REFERENCES KUN (EVIDENCNI_CISLO);
64
65  ALTER TABLE KUN
66     ADD CONSTRAINT FK_KUN_PLEMENO_K_PLEMENO FOREIGN KEY (NAZEV_PLEMENE)
67        REFERENCES PLEMENO (NAZEV_PLEMENE);
68
69  /*==============================================================*/
70  /* Materializovaný pohled, příslušný log a omezení, které       */
71  /* zajišťují, že matkou je pouze klisna, otcem pouze hřebec,    */
72  /* a že se potomek nenarodil dříve, nežli rodič.                */
73  /*==============================================================*/
74  CREATE MATERIALIZED VIEW LOG ON KUN
75    WITH ROWID, PRIMARY KEY;
76
77  CREATE MATERIALIZED VIEW MVCK_RODIC_POHLAVI_VEK
78    REFRESH FAST ON COMMIT AS
79    SELECT POTOMEK.ROWID AS POTOMEK_ROWID, RODIC.ROWID AS RODIC_ROWID
80      FROM KUN POTOMEK, KUN RODIC
81      WHERE
82      (POTOMEK.MATKA_CISLO || 'H' = RODIC.EVIDENCNI_CISLO || RODIC.POHLAVI)
83      OR (POTOMEK.OTEC_CISLO || 'K' = RODIC.EVIDENCNI_CISLO || RODIC.POHLAVI)
84      OR ((POTOMEK.MATKA_CISLO = RODIC.EVIDENCNI_CISLO
85           OR POTOMEK.OTEC_CISLO = RODIC.EVIDENCNI_CISLO)
86           AND POTOMEK.DATUM_NAROZENI <= RODIC.DATUM_NAROZENI);
87
88  ALTER MATERIALIZED VIEW MVCK_RODIC_POHLAVI_VEK
89    ADD CONSTRAINT CK_RODIC_POHLAVI_VEK
90      CHECK (POTOMEK_ROWID IS NULL AND RODIC_ROWID IS NULL);
```

11 Kusovník

11.1 Účel úlohy

Tato úloha ilustruje zejména následující problémy/jevy:

- Řešení hierarchie v relační databázi.

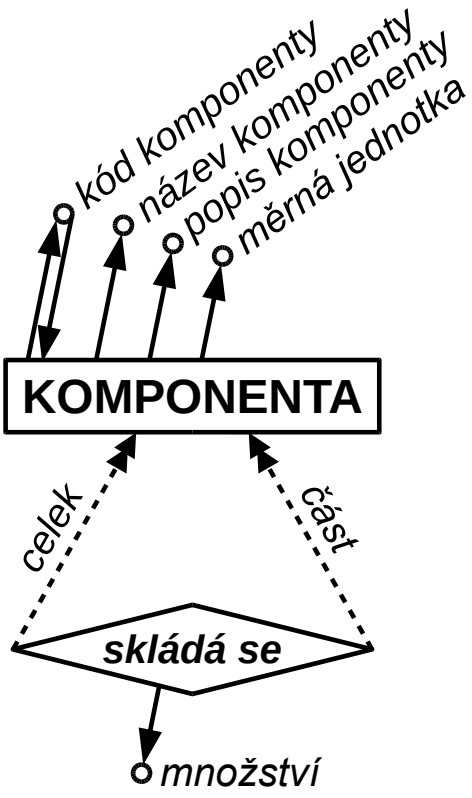
11.2 Popis zvolené výseče světa – zadání

Pro účely výrobního podniku je třeba evidovat tzv. kusovníky (anglicky *bills of materials*). Kusovník je strukturovaný seznam součástek (dílů, komponent), které dohromady tvoří sestavu nebo konečný produkt, a jejich potřebného množství pro výrobu sestavy nebo konečného produktu (volně dle SAP, n.d.a). Sestava je funkční celek určitých součástek. Kusovníky mají zpravidla hierarchickou strukturu, kde na vrcholu hierarchie je konečný produkt (výrobek) nebo sestava, přičemž jedna sestava může být součástkou jiné sestavy nebo konečného produktu. Produkt v jednom kusovníku může v jiném kusovníku plnit úlohu komponenty.

Příkladem může být třeba kuchyňská linka. Ta se zpravidla skládá ze skříněk různých typů a rozměrů a kuchyňské pracovní desky. Pro určitou zakázku nebo typovou linku jako artikl je tedy třeba zaznamenat, z jakých typů skříněk se daný artikl skládá a v jakém množství. Jednotlivé skříňky se pak typicky skládají z podstavy, zad, bočnic, dvířek apod. K jejich montáži jsou potřeba šrouby, vruty, dřevěné kolíčky apod. Opět v různém množství. Některá skříňka je úzká a dvířka má jednokřídlá, jiné jsou širší a mají dvířka dvoukřídlá. Aby výrobní podnik věděl, kolik kusů od šroubků až po skříňky musí pro vyřízení zakázky pořídit nebo vyrobit, potřebuje kusovník pro každý nabízený artikl.

V portfoliu výrobního podniku nemusí být finálním produktem pouze celá kuchyňská linka. Finálním produktem mohou být i jednotlivé skříňky, pokud jsou nabízeny samostatně.

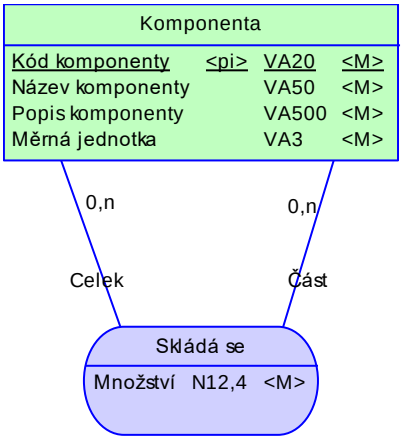
11.3 Konceptuální schéma



Obrázek 70: Evidence zásob – konceptuální schéma, zdroj: autoři

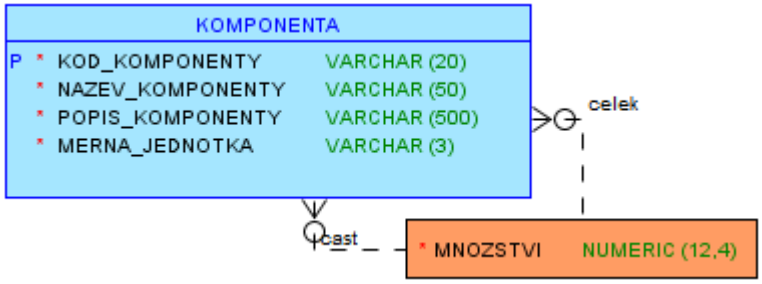
11.4 Konceptuální datový model

11.4.1 KDM v nástroji Power Designer



Obrázek 71: Kusovník – konceptuální datový model (Power Designer), zdroj: autoři

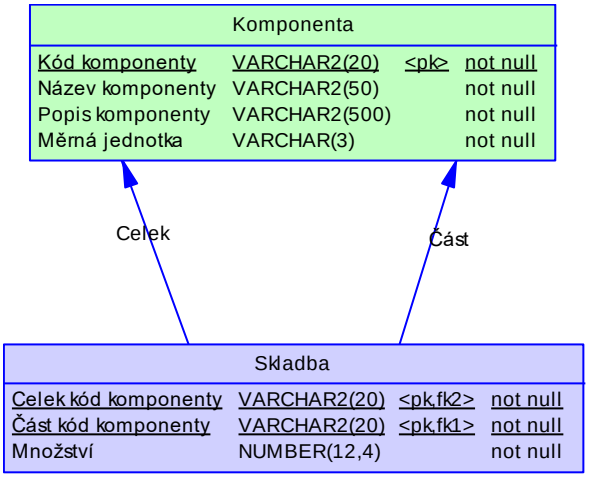
11.4.2 KDM v nástroji Data Modeler



Obrázek 72: konceptuální datový model (Data Modeler), zdroj: autoři

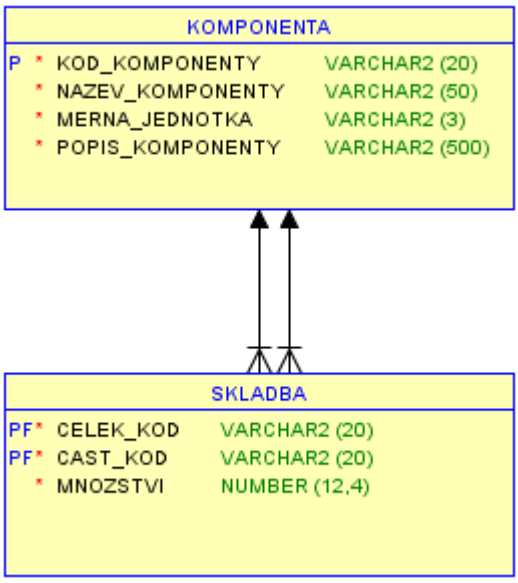
11.5 Fyzický datový model

11.5.1 FDM v nástroji Power Designer



Obrázek 73: Kusovník – fyzický datový model (Power Designer), zdroj: autoři

11.5.2 FDM v nástroji Data Modeler



Obrázek 74: Kusovník – fyzický datový model (Data Modeler), zdroj: autoři

11.6 Komentář k návrhu

Kusovník je vhodné si představit jako strom komponent, kde se komponenty na určité úrovni hierarchie skládají z komponent na úrovni o jednu nižší. Na nejvyšší úrovni hierarchie (vrchol stromu) je pak konečný výrobek nebo sestava. Na nejnižší úrovni hierarchie jsou pak komponenty představující vstupy do výroby. Datový model je pak třeba navrhnout tak, aby každá z komponent mohla být celkem i částí zároveň, protože např. skříňka je celek skládající se z bočnic, zad, dvířek apod., ale sama je zároveň částí kuchyňské linky. Proto je v rámci konceptuálního schématu na obrázku 70 pouze jedna entitní množina *komponenta* a jeden vztah M:N označený *skládá se*, který nastává mezi komponentami a vyjadřuje, že komponenta, která ve vztahu představuje *celek*, se skládá z určitého množství jiných komponent představujících *části celku*.

Komponenty, jejichž kód se nevyskytuje v tabulce *skladba* ve sloupci *část kód*, představují pouze finální produkty. Analogicky komponenty, jejichž kód se nevyskytuje v tabulce *skladba* ve sloupci *celek kód*, představují komponenty, které vystupují pouze jako vstupy výroby.

11.7 Příkazy pro vytvoření relačních tabulek a dalších objektů

Výpis 10: Kusovník – SQL skript, zdroj: autoři

```
/*==============================================================*/
/* Tabulka: KOMPONENTA                                          */
/*==============================================================*/
CREATE TABLE KOMPONENTA  (
   KOD_KOMPONENTY       VARCHAR2(20)                    NOT NULL,
   NAZEV_KOMPONENTY     VARCHAR2(50)                    NOT NULL,
   POPIS_KOMPONENTY     VARCHAR2(500)                   NOT NULL,
   MERNA_JEDNOTKA       VARCHAR(3)                      NOT NULL,
   CONSTRAINT PK_KOMPONENTA PRIMARY KEY (KOD_KOMPONENTY)
);

/*==============================================================*/
/* Tabulka: SKLADBA                                             */
/*==============================================================*/
CREATE TABLE SKLADBA  (
   CELEK_KOD            VARCHAR2(20)                    NOT NULL,
   CAST_KOD             VARCHAR2(20)                    NOT NULL,
   MNOZSTVI             NUMBER(12,4)                    NOT NULL
      CONSTRAINT CKC_MNOZSTVI_SKLADBA CHECK (MNOZSTVI >= 0),
   CONSTRAINT PK_SKLADBA PRIMARY KEY (CAST_KOD, CELEK_KOD),
   CONSTRAINT CKT_SKLADBA CHECK (CAST_KOD <> CELEK_KOD)
);

ALTER TABLE SKLADBA
   ADD CONSTRAINT FK_SKLADBA_CAST_KOMPONEN FOREIGN KEY (CELEK_KOD)
      REFERENCES KOMPONENTA (KOD_KOMPONENTY);

ALTER TABLE SKLADBA
   ADD CONSTRAINT FK_SKLADBA_CELEK_KOMPONEN FOREIGN KEY (CAST_KOD)
      REFERENCES KOMPONENTA (KOD_KOMPONENTY);
```

12 Časté atributy a jejich reprezentace v relační databázi

Tato kapitola se věnuje vybraným atributům a jejich reprezentaci v relační databázi. Databáze velmi často obsahují data o osobách či organizacích, z čehož vyplývá, že při návrhu databází je třeba se často vypořádat s tím, jak zaznamenat např. jméno osoby či organizace, poštovní adresu, e-mailovou adresu nebo telefon. Kapitola shrnuje způsoby řešení těchto atributů, které byly použity v předcházejících kapitolách u řešení jednotlivých úloh, a tato řešení blíže vysvětluje nebo doplňuje o širší kontext.

12.1 Názvy organizací a jména osob

Osoby a organizace jsou velmi často účastníky vztahů nebo událostí, o kterých potřebujeme ukládat data. Díky tomu je třeba často ukládat jména osob nebo názvy organizací. Stejně tomu bylo i ve většině úloh řešených v těchto skriptech, přičemž v jednotlivých řešeních byla zvolena různá řešení reprezentace a ukládání jmen osob nebo názvů organizací. Jak je uvedeno dále v této kapitole, vhodný návrh ukládání jmen osob a názvů organizací v databázi není jednoduchým problémem.

Datovou reprezentací jmen osob, názvů organizací a adres se zabývá standard organizace OASIS[17] s názvem *Customer Information Quality Specification Version 3.0*, zkráceně OASIS CIQ V3.0 (OASIS, 2008). Adresám se věnuje jiná kapitola, proto se text zde věnuje pouze jménům osob a názvům organizací. Výše uvedený standard je v části *extensible Name Language (xNL)* navržen tak, aby umožňoval různé způsoby reprezentace jména/názvu. Ve standardu jsou uvedeny tři příklady lišící se mírou, do jaké zvolený způsob reprezentace umožňuje vyjádřit význam jména/názvu, případně jednotlivých slov, které ho tvoří (OASIS, 2008):

1. **Reprezentace neumožňující vyjádření sémantiky částí jména/názvu.** Jedná se o variantu, kdy je jméno osoby či název organizace reprezentováno jako jedno pole, přičemž význam tohoto pole a jemu odpovídající název nerozlišují mezi tím, zda se jedná o jméno osoby nebo o název organizace.

[17] https://www.oasis-open.org/

2. **Reprezentace umožňující vyjádření minima sémantiky částí jména/názvu.** Jedná se o variantu, kdy jsou používána pole s jiným významem (a názvem) pro ukládání jména osoby a názvu organizace. Význam jednotlivých slov, která tvoří jméno či název ale není rozlišován. Tato varianta tak sice umožňuje rozlišit mezi osobou a organizací, ale již není možné rozlišit např. křestní jméno od příjmení ve jméně osoby.
3. **Reprezentace plně vyjadřující sémantiku částí jména/názvu.** Jedná se o variantu, kdy je nejen rozlišováno mezi jménem osoby a názvem organizace, ale jsou používána specifická pole pro jednotlivá slova tvořící jméno či název, která vyjadřují jejich význam. Tato varianta umožňuje rozlišit např. křestní jméno od příjmení osoby.

Přístup odpovídající variantě č. 1 popsané výše byl aplikován při řešení úlohy *Evidence faktur*, kdy tabulka *Zákazník* pro uložení jména osoby či názvu organizace obsahuje jeden sloupec s názvem *Obchodní firma / jméno*. Přístup odpovídající variantě č. 3 pak byl uplatněn při řešení úlohy *Evidence daňových poplatníků*.

V případě jmen osob je vhodné si uvědomit, že pokud je třeba rozlišovat význam jednotlivých slov tvořících jméno, v různých zemích a kulturách jsou jména tvořena různě a různí se i význam jednotlivých slov tvořících jméno (Ishida, 2016). V České republice, ale zpravidla i v dalších nám kulturně blízkých zemích, je používáno jména a příjmení; obecně tomu tak ale dle Ishidy (2016) není. Standard OASIS CIQ V3.0 přepokládá ukládání dat ve formátu XML, a s výše uvedeným problém se vypořádává tak, že elementu jména lze přiřadit typ vyjadřující význam elementu (např. zda se jedná o křestní jméno, příjmení, nebo třeba přezdívku), přičemž standard předpokládá rozšiřování číselníku typů elementů dle potřeby, aby bylo možné např. zavést typ vyjadřující, že element jména vyjadřuje křestní jméno otce osoby (OASIS, 2008).

V relační databázi je význam atributu zpravidla promítnut do názvu sloupce příslušné tabulky databáze. Pokud by bylo třeba zachovat význam jednotlivých slov tvořících jméno osoby, bylo by třeba různých sloupců s různými významy. Sloupce označené jako „křestní jméno“ a „příjmení“ obecně zachování významu slov tvořících jméno neumožní. Při návrhu relační databáze je tak z hlediska struktury sloupců pro ukládání jména osoby třeba zvážit kontext a požadavky. Na základě toho je pak třeba rozhodnout, zda bude celé jméno uloženo do jednoho sloupce, nebo zda je třeba jméno

rozdělit do více elementů a ty pak ukládat do více samostatných sloupců. V případě rozdělení jména do více sloupců lze pro jejich pojmenování vyjít z Ishidova (2016) doporučení pro mezinárodní formuláře pro zadávání jmen, kdy je doporučeno rozlišovat příjmení / rodové jméno (family name) a ostatní/křestní jména (other/given names) s tím, že název sloupce nemusí vždy přesně vyjadřovat význam skutečně zadaných částí jména určité osoby.

Pokud by bylo třeba ukládat do relační databáze jména osob s přesně vyjádřeným významem jednotlivých slov tvořících jména, bylo by vhodné reprezentovat jména pomocí XML dle části xNL standardu OASIS CIQ V3.0, a XML dokumenty ukládat do sloupce tabulky relační databáze s vhodným datovým typem (`VARCHAR` dostatečné délky, `CLOB`, případně XML, pokud takovýto datový typ příslušný SŘBD podporuje).

Ohledně názvů organizací lze uvést, že existuje mezinárodní organizace Global Legal Entity Identifier Foundation[18], která vydává standard pro mezinárodní identifikátor právnické osoby, přičemž specifikace datového souboru o právnické osobě (GLEIF, 2017) obsahuje pole obchodní firma (legal name) a (volitelně) ostatní názvy organizace (other entity names). Číselník typů elementů tvořících název organizace standardu OASIS CIQ V3.0 rozlišuje element obsahující název organizace včetně označení právní formy, a samostatné elementy pro název organizace bez označení právní formy, resp. pro samotné označení právní formy. V relační databázi tak můžeme sloupce pro ukládání názvu organizace / právnické osoby strukturovat např. v souladu s jedním z těchto dvou standardů.

Kromě struktury sloupců relační databáze je třeba také zvolit vhodný datový typ, příp. včetně maximální délky či velikosti hodnost. V případě jmen osob a názvů organizací bude s ohledem na povahu dat vhodné volit datové typy pro textové řetězce. Obecně je třeba vycházet z toho, že jména a názvy nemusí mít délku omezenu. Např. zákon č. 301/2000 Sb., o matrikách, jménu a příjmení a o změně některých souvisejících zákonů, upravuje mimo jiné počet jmen, které může osoba mít, případně podmínky, kdy smí používat více příjmení, ale explicitně délku jména ani příjmení neomezuje. Např. podle Guinnessovy Knihy světových rekordů se člověk s nejdelším jménem uvedeným v rodném listu jmenuje Rhoshandiatellyneshiaunneveshenk Koyaanisquatsiuth Williams (Guinness World Records Limited, © 2019). Textový

[18] https://www.gleif.org/en/

řetězec tvořený křestními jmény a příjmením tohoto člověka včetně mezer má délku 59 znaků. Podle zprávy Českého rozhlasu Radiožurnál (2014) ve Švédsku údajně žije muž, jehož jméno je tvořeno 63 slovy.

Přestože se lze setkat i s poměrně dlouhými jmény či názvy, v praxi bývají maximální délky textových řetězců jmen a názvů omezovány. Např. v rámci úlohy *Evidence daňových poplatníků* jsou délky jmen fyzických osob a firem právnických osob stanoveny dle délek v datových schématech vydaných Generálním finančním ředitelstvím (2018a, 2018b). Maximální délku textového řetězce pro uložení jména či názvu je tak třeba zvolit tak, aby bylo možné uložit i dlouhá jména. Ishida (2016) pro ukládání mezinárodních jmen dokonce doporučuje neomezovat délku polí pro uložení jména v databázi. Tomuto doporučení by v relační databázi odpovídalo použití datového typu `CLOB`.

Pokud by bylo uvažováno o použití datového typu `CLOB` pro ukládání jména, bylo by vhodné rozmyslet, jaké dotazy zahrnující jméno budou nad databází vykonávány. Podle jména jsou často osoby vyhledávány. Bylo by tedy třeba posoudit, jaké jsou možnosti cílového SŘBD ohledně indexování sloupců s datovým typem `CLOB`, a jaký by byl dopad použití tohoto datového typu na výkonnost příslušných dotazů. Při ukládání celého jména osoby do sloupce s datovým typem `CLOB`, se tak jeví vhodné zvážit další z Ishidových (2016) doporučení, a sice vedle celého jména ukládat samostatně i to, jak si dotyčný přeje, aby byl oslovován. Za předpokladu, že osoba s dlouhým jménem v komunikaci s dalšími lidmi, organizacemi a institucemi používá z praktického hlediska kratší preferované oslovení nežli dlouhé celé jméno, bude i v databázi vyhledávána spíše podle preferovaného oslovení. Ukládání preferovaného oslovení by pak bylo vhodné navrhnout s ohledem na potřebné dotazy pracující se jmény, což bude patrně zahrnovat volbu jiného datového typu pro textové řetězce nežli `CLOB`.

12.2 Poštovní adresa

V databázích se lze setkat s různými způsoby uložení poštovní adresy a také existují různé standardy, které podobu poštovní adresy upravují. Tato část nemá za cíl podrobně porovnat jednotlivé způsoby a standardy, ale spíše poukázat na to, že přístupů je více a při návrhu databáze je tedy třeba určit, jaké typy dotazů zahrnujících poštovní adresy, nebo jednotlivé údaje v nich obsažené, má databáze podporovat, a podle toho zvolit vhodný způsob uložení poštovní adresy.

Doporučení a pravidla pro mezinárodní poštu vydává organizace Universal Postal Union (UPU)[19], která je od roku 1948 specializovanou agenturou spadající pod Organizaci spojených národů (Universal Postal Union, n.d.a, Universal Postal Union, n.d.c). Formát poštovních adres není celosvětově jednotný. Např. v Japonsku adresa neobsahuje název ulice (Universal Postal Union, 2019). Organizace UPU vypracovala standard S42, který specifikuje obecně použitelné elementy adresy a národní profily, které umožňují převod adresy vyjádřené pomocí obecně použitelných elementů na adresu odpovídající národnímu formátu (Universal Postal Union, n.d.b).

Již v kapitole 12.1 byl zmíněn standard OASIS CIQ V3.0. Ten v části *extensible Address Language (xAL)* upravuje poštovní adresu (OASIS, 2008). Obdobně jako v případě jmen osob a názvů organizací je standard koncipován tak, aby s jeho pomocí bylo možné reprezentovat adresu různými způsoby, které se liší mírou, do jaké je přesně zachycen význam jednotlivých elementů, které adresu tvoří. Standard uvádí následující příklady způsobů reprezentace adresy (OASIS, 2008):

1. **Reprezentace neumožňující vyjádření sémantiky elementů adresy.** Tento způsob reprezentace adresy vychází ze skutečnosti, že adresa je zapisována do jednoho nebo více řádků. Data adresy jsou tak ukládána po jednotlivých řádcích bez rozlišení významu údajů, které jsou na jednotlivých řádcích zapsány.
2. **Reprezentace umožňující částečné vyjádření sémantiky elementů adresy.** Tento způsob reprezentace adresy spočívá v tom, že jsou používány některé elementy adresy s vyjádřeným významem, ale význam není vyjádřen u všech elementů adresy. Např. mohou být zavedeny elementy pro územněsprávní celek (např. stát v rámci federace, kraj apod.), lokalitu (např. obec) a místo, přičemž textový řetězec místa může obsahovat např. název ulice společně s číslem domu. Přestože číslo domu je ve své podstatě samostatným elementem adresy, při využití tohoto způsobu reprezentace adresy není oddělen od názvu ulice v označení místa.

[19] http://www.upu.int

3. **Reprezentace plně vyjadřující sémantiku elementů adresy.** Tento způsob reprezentace adresy spočívá v tom, že každý element adresy je reprezentován samostatně atributem vyjadřujícím význam příslušného elementu adresy. Při využití tohoto způsobu reprezentace adresy je např. název ulice oddělen od čísla domu. Standard xAL obsahuje číselník obecně použitelných typů elementů adresy.

Při návrhu uložení poštovní adresy v relační databázi je vhodné posoudit, jakou práci s adresami má systém podporovat. Způsob reprezentace adresy č. 1 uvedený výše lze v relační databázi implementovat tak, že v příslušné tabulce budou obsaženy sloupce pro řádky adresy. Sloupců musí být fixní počet. Tabulka by tak měla obsahovat minimálně řádky pro uložení místa (např. název ulice a číslo domu), lokality (např. obec), poštovního čísla, územněsprávního celku a státu. Pokud sloupce tabulky nebudou pojmenovány „řádek adresy 1-5“, ale budou pojmenovány podle významu elementů adresy uvedených v předcházející větě, návrh uložení adresy bude odpovídat způsobu č. 2 popsanému výše.

V případě způsobu č. 1 by z pořadí sloupců neměl být dovozován význam, protože stát se může vyskytnout např. ve třetím, čtvrtém nebo v pátém sloupci podle toho, na jakém řádku adresy byl uveden na vstupu. Za podmínky, že pořadí řádků adresy na vstupu bude odpovídat pořadí sloupců v tabulce, bude možné při využití způsobu č. 1 na základě dat v databázi zrekonstruovat adresu na vstupu tak, aby mohla být vytištěna např. na obálku zásilky. Pokud by takto byly ukládány např. adresy zákazníků, bylo by obtížné, aby systém podporoval dotazy typu „počet zákazníků v členění dle států a měst“, protože by nebylo zřejmé, ve kterých sloupcích jsou uvedena města a státy, případně by v jednom sloupci mohly být uvedeny hodnoty více významů.

Způsob reprezentace adresy č. 2 popsaný výše lze v relační databázi realizovat s fixním počtem sloupců. Tento způsob reprezentace adresy byl využit např. úloze *Evidence zásob*. Tento způsob neumožňuje úplně přesnou reprezentaci libovolné ve světě existující adresy. Struktura sloupců by tak měla být zvolena tak, aby umožňovala pokrýt většinu případů, které má systém podporovat.

Způsob reprezentace adresy č. 3 není v relační databázi vhodné realizovat strukturou sloupců, protože sloupců s různým významem by bylo velké množství a práce s databází by byla komplikovaná. Jako vhodnější se jeví při využití způsobu

reprezentace adresy č. 3 adresu reprezentovat v podobě XML dokumentu dle standardu OASIS CIQ V3.0 a tento dokument pak ukládat do sloupce s vhodným datovým typem (`VARCHAR` dostatečné délky, `CLOB`, případně XML, pokud takovýto datový typ příslušný SŘBD podporuje).

V České republice je vhodné zmínit Registr územní identifikace adres a nemovitostí (RÚIAN)[20] a vyhlášku o tomto registru č. 359/2011 Sb., která mimo jiné stanovuje způsob zápisu adresy. RÚIAN obsahuje referenční údaje o územních prvcích a obsah RÚIAN je vymezen v § 31 zák. č. 111/2009 Sb., o základních registrech. Data z tohoto registru lze získat prostřednictvím aplikace Veřejného dálkového přístupu (VDP)[21], přičemž soubory dat jsou k dispozici v tzv. Výměnném formátu RÚIAN (VFR)[22]. V dokumentaci VFR je pak popsán datový model prvků v RÚIAN včetně atributů, datových typů a délek. Pro návrh databáze, která má sloužit pro ukládání dat o územních prvcích České republiky, tak je relevantní specifikace VFR. Zápis tuzemské adresy by se pak měl odpovídat vyhlášce č. 359/2011 Sb.

V souvislosti s RÚIAN a adresními místy v České republice je vhodné zmínit ještě skutečnost, že jednotlivá adresní místa mají svůj identifikátor ve formě dereferencovatelného IRI. Např. adresní místo Vysoké školy ekonomické v Praze (náměstí Winstona Churchilla 1938/4, Žižkov, 13000 Praha 3) má následující identifikátor: https://linked.cuzk.cz/resource/ruian/adresni-misto/21788561. Pokud je identifikátor použit jako URL, jsou získána data o adresním místě včetně adresy dle vyhlášky č. 359/2011 Sb. Podle pracovní verze otevřené formální normy pro zveřejňování adres je za nejčistší formu určení adresy v České republice považováno použití IRI adresního místa (Škop a Klímek, 2019). Tj. nikoli zveřejnění adresy ve formě dat představujících jednotlivé řádky či elementy adresy.

12.3 E-mailová adresa

V protokolu SMTP je dle specifikace RFC 5321 velikost cest identifikujících odesílatele (reverse-path) a příjemce (forward-path) omezena na 256 oktetů, přičemž cesta musí vždy začínat znakem „<“ a končit znakem „>“ (Klensin, 2008). Při využití ASCII kódování umožní 256 oktetů zakódovat 256 znaků. E-mailová adresa se skládá z místní

[20] https://www.cuzk.cz/ruian/RUIAN.aspx
[21] https://vdp.cuzk.cz/
[22] https://www.cuzk.cz/Uvod/Produkty-a-sluzby/RUIAN/2-Poskytovani-udaju-RUIAN-ISUI-VDP/Vymenny-format-RUIAN/Vymenny-format-RUIAN-(VFR).aspx

části, znaku „@“ a domény. Ačkoli dle specifikace RFC 3696 (Klensin, 2004) by při zohlednění maximální délky místní a lokální části mohla mít e-mailová adresa až 320 znaků, skutečně použitelná délka je ale limitována maximální velikostí cesty dle RFC 5321. Od té je třeba odečíst dva znaky pro povinné ostré závorky na začátku a konci cesty. Maximální velikost e-mailové adresy tak je 254 znaků (Sayers, 2009).

12.4 Souřadnice bodu v prostoru

Geodata (prostorová data) jsou „*data popisující geografické umístění a další vlastnosti přirozených nebo člověkem vytvořených krajinných útvarů, jevů, nebo hranic na Zemi*“ (Open Geospatial Consortium, © 2019b). Pro reprezentaci prostorových dat existují zvláštní jazyky a formáty, jako např. Geography Markup Language (GML, blíže viz Open Geospatial Consortium, © 2019a). Výklad o ukládání a zpracování geodat jde nad rámec těchto skript, nicméně v této kapitole je stručně vysvětlen způsob uložení souřadnic skladu z úlohy *Evidence zásob*.

Prostorové umístění skladu bylo vyjádřeno bodem, jehož geografické umístění lze vyjádřit pomocí jeho zeměpisné šířky a délky. Do tabulky skladů by se tak nabízelo přidat dva sloupce; jeden pro uložení zeměpisné šířky, druhý pro uložení zeměpisné délky. Nicméně, tento přístup je považován za špatnou praxi, jelikož se vytrácí význam, že data mají reprezentovat geometrický objekt (Ministerstvo vnitra České republiky, 2018). Místo toho je doporučováno uložit data s využitím některého z jazyků či formátů pro geodata, což umožní zaznamenat, že údaje o zeměpisné délce a šířce patří k sobě, a dohromady tvoří bod v prostoru. Využití jazyka GML dále umožní zaznamenat např. i údaje souřadnicovém systému. Reprezentace dat s využitím specializovaných jazyků či formátů pro geodata zajistí snazší uplatnitelnost dat v systémech či aplikacích, které s prostorovými daty pracují, např. je promítají do mapy.

Některé systémy řízení báze dat disponují funkcemi pro podporu práce s geodaty, např. Oracle Spatial and Graph (Oracle, © 2019b), nebo MariaDB (MariaDB, © 2019). Pokud je tedy systém řízení báze dat s takovýmito vlastnostmi využíván, je možné zvážit využití vlastností pro ukládání geodat, kterými můžou být např. zvláštní datové typy pro reprezentaci prostorových útvarů. Návrh databáze uvedený v kapitole 4 s využitím takovýchto funkcí nepočítá. Pro uložení souřadnic byl zvolen jeden sloupec s textovým datovým typem, přičemž se předpokládá, že by jako hodnota byla ukládána reprezentace bodu např. v jazyce GML.

12.5 Telefonní číslo

Podobu telefonních čísel na mezinárodní úrovni upravuje Mezinárodní telekomunikační unie (International Telecommunication Union, ITU)[23]. Při určení délky sloupce pro ukládání telefonního čísla vyjedeme z Mezinárodního číslovacího plánu veřejných telefonních sítí (International Telecommunication Union, 2010) a z Vytáčecích procedur (International Telecommunication Union, 2011). Na národní úrovni pak úprava telefonních čísel spadá do kompetencí Českého telekomunikačního úřadu (ČTÚ)[24], který vydává Číslovací plán veřejných telefonních sítí (Český telekomunikační úřad, 2000).

Mezinárodní telefonní číslo má dle číslovacího plánu vydaného jak ITU, tak ČTU délku maximálně 15 číslic a skládá se z čísla země (jedna až tři číslice), národního směrového čísla a účastnického čísla, přičemž národní směrové číslo a účastnické číslo dohromady tvoří národní (významové) číslo. Mezinárodní telefonní číslo ale nezahrnuje tzv. přestupný znak, který při volání do zahraničí označuje, že za ním následující číslo představuje číslo země. V České republice je tímto přestupným znakem „`00`“ („nula“ „nula“). Přehled přestupných znaků používaných jinými zeměmi uvádí Vytáčecí procedury (International Telecommunication Union, 2011). V souladu s Mezinárodním číslovacím plánem veřejných telefonních sítí lze při zápisu telefonního čísla místo přestupného znaku uvést znak „+“ (International Telecommunication Union, 2010). Podle doporučení pro psaní telefonních čísel vydaných ITU (International Telecommunication Union, 2001) by dokonce v případě mezinárodních čísel měl být před národním směrovým číslem uveden právě znak „+“.

Při ukládání telefonních čísel je tedy třeba obecně předpokládat, že telefonní číslo, které bude třeba uložit, bude začínat nulou, nebo znakem „+“. Pro ukládání telefonního čísla, by tak měl být volen znakový datový typ spíše než celočíselný datový typ, protože by buď nebylo možné uložit znak „+“, nebo by došlo ke ztrátě informace díky tomu, že nuly, kterými by telefonní číslo začínalo, by nebyly uloženy.

Vzhledem k tomu, že volba přestupného znaku, nezávisí na čísle volaného, ale na tom, odkud (z jaké země) volá volající, pro onačení, že telefonní číslo obsahuje národní směrové číslo, se jeví vhodnější ukládat znak „+“, nežli specifické číslo představující

[23] https://www.itu.int/en/Pages/default.aspx
[24] https://www.ctu.cz

přestupný znak. Je pak na volajícím, nebo na systému prostřednictvím kterého volající hovor uskutečňuje, aby přeložil znak „+“ na správné číslo přestupného znaku.

Pokud jde o délku sloupce, jeví se vhodné databázi navrhnout tak, aby umožnila uložit vedle přestupného znaku ve formě „+“ alespoň 15 číslic, což odpovídá délce mezinárodního čísla. Dle Vytáčecích procedur (International Telecommunication Union, 2011) mohou mít národní (významová) čísla v Korejské lidově demokratické republice až 17 číslic. S národním směrovým číslem (850) by tak telefonní číslo v KLDR mohlo mít až 20 číslic (bez přestupného znaku).

13 Závěr

Tato skripta přinášejí sadu řešených příkladů, které čtenáři umožní pochopit návrh databáze na jeho jednotlivých úrovních. Skripta nejsou uceleným návodem datového modelování, ale na řešených příkladech čtenáře seznamují s vybranými aspekty a činnostmi databázového návrhu:

- konceptuální úroveň návrhu a fyzická (implementační) úroveň návrhu,
- vztahy mezi entitními množinami, kardinalita a parcialita vztahů, atributy vztahů,
- určení dominantního směru vztahu v případě vztahu 1:1,
- dekompozice složených atributů při přechodu z konceptuálního schématu na konceptuální datový model,
- identifikační závislost entitní množiny (slabé entitní typy),
- generalizace/specializace a její řešení na konceptuální a fyzické úrovni,
- nevýlučné podmnožiny entitní množiny,
- jak se vypořádat s tím, že se určitá data mění v čase, přičemž je třeba udržovat jak aktuální data, tak data ve stavu, kdy došlo k nějaké události,
- vícenásobné (n-ární) vztahy a jejich řešení na konceptuální a fyzické úrovni,
- vztah mezi jednou a toutéž entitní množinou,
- změna návrhu databáze na základě rozšířených požadavků,
- způsob implementace vybraných atributů.

Rozsah problematiky datového modelování tím zdaleka není pokryt, ovšem pečlivé prostudování a promyšlení úloh pomůže jako start ke kvalifikaci datového architekta i jako dostatečné poučení pro kohokoli, kdo s využíváním databází přichází do styku.

Úlohy jsou záměrně pro účely výuky omezeny a zjednodušeny, reálné řešení skutečných systémů může sestávat ze stovek tabulek, a řešení na databázové úrovni může pro účely efektivního zpracování používat nikoli učebnicové konstrukce. To nesnižuje potřebnost návrhu datového modelu, který je předpokladem pro vybudování kvalitní, stabilní a pružné databáze. Dobře sestavený datový model je také jedním z předpokladů snazšího ladění výkonnosti databázového systému. Tyto dovednosti ovšem přesahují rámec těchto skript i rozsah učiva úvodního předmětu do problematiky databází.

14 Seznam použité literatury

BACHMAN, Charles W., 1969. Data structure diagrams. *ACM SIGMIS Database*. 1969, vol. 1, no. 2, s. 4-10. ISSN 0095-0033.

CHEN, Peter, 1976. The entity-relationship model-toward a unified view of data. *ACM Transactions on Database Systems (TODS)*. 1976, vol. 1, no. 1, s. 9-36. ISSN 0362-5915.

COOK, William, R., Walter HILL a Peter S. CANNING, 1990. *Inheritance is not subtyping*. Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. New York (USA): ACM, 1990, s. 125-135. ISBN 0-89791-343-4. Dostupné z: http://doi.acm.org/10.1145/96709.96721.

Český telekomunikační úřad, 2000. Číslovací plán veřejných telefonních sítí. In: *Český telekomunikační úřad* [online]. 25. 9. 2000 [cit. 2019-06-06]. Dostupné z: https://www.ctu.cz/sites/default/files/obsah/predpisy-opatreni/cislovaci-plan-verejnych-telefonnich-siti/soubory/cislovaci-plan-verejnych-telefonnich-siti-1114435245.pdf.

Česká národní banka, 2011. Vyhláška č. 169/2011, o stanovení pravidel tvorby čísla účtu v platebním styku. In: *Česká národní banka* [online]. 7. června 2011 [cit. 2019-06-11]. Dostupné z: https://www.cnb.cz/export/sites/cnb/cs/platebni-styk/.galleries/pravni_predpisy/download/vyhl_169_2011.pdf.

Český rozhlas Radiožurnál, 2014. Tohle jsou nejšílenější jména na světě! V Česku s neobvyklými jmény uspějete stěží. In: *Radiožurnál* [online]. 9. říjen 2014 [cit. 2019-07-16]. Dostupné z: https://radiozurnal.rozhlas.cz/tohle-jsou-nejsilenejsi-jmena-na-svete-v-cesku-s-neobvyklymi-jmeny-uspejete-6268281.

Český úřad zeměměřičský a katastrální, 2016. Struktura a popis výměnného formátu RÚIAN (VFR). In: *ČÚZK* [online]. Aktualizováno 4. 2. 2016 [cit. 2019-03-15]. Dostupné z: https://www.cuzk.cz/Uvod/Produkty-a-sluzby/RUIAN/2-Poskytovani-udaju-RUIAN-ISUI-VDP/Vymenny-format-RUIAN/Vymenny-format-RUIAN-(VFR)/Struktura-a-popis-VFR-1_8_0.aspx.

Finanční správa, 2019a. Čísla matrik bankovních účtů finančních úřadů včetně Specializovaného finančního úřadu. In: *Finanční správa* [online]. 1. 1. 2019 [cit. 2019-05-24]. Dostupné z: https://www.financnisprava.cz/assets/cs/prilohy/d-placeni-dani/bankovni_ucty_Priloha_1.pdf.

Finanční správa, 2019b. Předčíslí bankovních účtů finančních úřadů (kromě Specializovaného finančního úřadu) pro placení daní. In: *Finanční správa* [online]. 1. 1. 2019 [cit. 2019-05-24]. Dostupné z: https://www.financnisprava.cz/assets/cs/prilohy/d-placeni-dani/bankovni_ucty_Priloha_2.pdf.

Finanční správa, 2018. Přehled o počtu DS evidovaných v agendách správců daní a o počtu DS aktivních v roce 2018. In: *Finanční správa* [online]. [cit. 2019-06-19]. Dostupné z: https://www.financnisprava.cz/assets/cs/prilohy/d-udaje-z-registru-danovych-subjektu/PoctyDS_fu2018.xlsx.

Generální finanční ředitelství, 2019a. Rozhraní číselníků. In: *Daňový portál* [online]. Verze: 40.23.1 ze dne 11.06.2019 [cit 2019-07-18]. Dostupné z: https://adisepo.mfcr.cz/adistc/adis/idpr_pub/epo2_info/rozhrani_ciselniku.faces.

Generální finanční ředitelství, 2019b. Žádost o zřízení daňové informační schránky. In: *Daňový portál* [online]. Verze: 40.23.1 ze dne 11.06.2019 [cit 2019-08-01]. Dostupné z: https://adisepo.mfcr.cz/adistc/adis/idpr_epo/epo2/form/form_uvod.faces?pisemnost=DPRZA1&ext=t.

Generální finanční ředitelství, 2018a. Daň z příjmů fyzických osob pro poplatníky mající pouze příjmy ze závislé činnosti ze zdrojů na území České republiky (včetně daňových nerezidentů České republiky) - od roku 2016 včetně. In: *Daňový portál* [online]. 5.12.2018 [cit. 2019-06-11]. Dostupné z: https://adisepo.mfcr.cz/adistc/adis/idpr_pub/epo2_info/popis_struktury_detail.faces?zkratka=DPFZC1.

Generální finanční ředitelství, 2018b. Daň z příjmů právnických osob - za zdaňovací období roků 2015, 2016, 2017, 2018 a pro části zdaňovacího období roku 2019. In: *Daňový portál* [online]. 5.12.2018 [cit. 2019-06-11]. Dostupné z: https://adisepo.mfcr.cz/adistc/adis/idpr_pub/epo2_info/popis_struktury_detail.faces?zkratka=DPPDP8.

Generální finanční ředitelství, 2014. Popis práce v Daňové informační schránce. In: *Finanční správa* [online]. 7. července 2014 [cit. 2019-06-19]. Dostupné z: http://adisdpr.mfcr.cz/adistc/adis/idpr_pub/dpr_info/popis_prace_v_dis.pdf.

Global Legal Entity Identifier Foundation. LEI Common Data File format V2.1. In: *GLEIF* [online]. 2017-03-21 [cit. 2019-07-16]. Dostupné z: https://www.gleif.org/content/2-about-lei/6-common-data-file-format/1-lei-cdf-format/2017-03-21_lei-cdf-v2-1.pdf.

Guinness World Records Limited, © 2019. Longest personal name. In: *Guinness World Records* [online]. [cit. 2019-07-16]. Dostupné z: https://www.guinnessworldrecords.com/world-records/67285-longest-personal-name.

ISHIDA, Richard, 2016. Personal names around the world. In: *W3C Internationalization* [online]. 2016-01-24 [cit. 2019-07-16]. Dostupné z: https://www.w3.org/International/questions/qa-personal-names.

ISO/IEC 11404 Information technology — General-Purpose Datatypes (GPD). Geneva: nternational Organization for Standardization, 2007. 106 s. Reference number ISO/IEC 11404:2007(E).

International Organization for Standardization, n.d. Country Codes - ISO 3166. In: *International Organization for Standardization* [online]. [cit. 2019-07-18]. Dostupné z: https://www.iso.org/iso-3166-country-codes.html.

International Telecommunication Union, 2011. Dialling Procedures (International Prefix, National (Trunk) Prefix and National (Significant) Number) (in Accordance with ITU-T Recommendation E.164 (11/2010)). In: *International Telecommunication Union* [online]. 15 December 2011 [cit. 2019-06-06]. Dostupné z: https://www.itu.int/dms_pub/itu-t/opb/sp/T-SP-E.164C-2011-PDF-E.pdf.

International Telecommunication Union, 2010. The international public telecommunication numbering plan. In: *International Telecommunication Union* [online]. 11/2010 [cit. 2019-06-06]. Dostupné z: https://www.itu.int/rec/T-REC-E.164/recommendation.asp?lang=en&parent=T-REC-E.164-201011-I.

International Telecommunication Union, 2001. Notation for national and international telephone numbers, e-mail addresses and Web addresses. In: *International Telecommunication Union* [online]. 02/2001 [cit. 2019-06-06]. Dostupné z: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-E.123-200102-I!!PDF-E&type=items.

KANISOVÁ, Hana a MÜLLER, Miroslav, 2006. *UML srozumitelně*. 2. vydání. Computer Press. ISBN 978-80-251-1083-6.

KIMBALL, Ralph a ROSS, Margy, 2013. *The data warehouse toolkit: the definitive guide to dimensional modeling* [online]. 3rd ed. Indianapolis, Ind.: Wiley, ©2013 [cit. 2019-06-13]. Dostupné z: https://ebookcentral.proquest.com/lib/vsep/detail.action?docID=1313513.

KLENSIN, John C., 2008. Simple Mail Transfer Protocol. In: *RFD 5321 – Simple Mail Transfer Protocol* [online]. October 2008 [cit. 2019-06-06]. Dostupné z: https://tools.ietf.org/html/rfc5321.

KLENSIN, John C., 2004. Application Techniques for Checking and Transformation of Names. In: *RFC 3696 – Application Techniques for Checking and Transformation of Names* [online]. February 2004 [cit. 2019-06-06]. Dostupné z: https://tools.ietf.org/html/rfc3696.

MariaDB, © 2019. Geographic & Geometric Features. In: *MariaDB Knowledge Base* [online]. [cit. 2019-06-17]. Dostupné z: https://mariadb.com/kb/en/library/geographic-geometric-features/.

Ministerstvo vnitra České republiky, 2018. Souřadnice X a Y jako samostatné atributy. In: *Otevřená data* [online]. Poslední úprava: 2018/08/25 00:00 [cit. 2019-06-17]. Dostupné z: https://opendata.gov.cz/standardy:geo:sou%C5%99adnice-x-y.

OASIS, 2008. Customer Information Quality Specifications Version 3.0. Name (xNL), Address (xAL), Name and Address (xNAL) and Party (xPIL). In: *OASIS* [online]. 20 September 2008 [cit. 2019-06-06]. Dostupné z: http://www.oasis-open.org/committees/download.php/29877/OASIS%20CIQ%20V3.0%20CS02.zip.

OASIS, 2007. Code List Representation (Genericode) Version 1.0. In: *OASIS* [online]. 28 December 2007 [cit. 2019-07-18]. Dostupné z: http://docs.oasis-open.org/codelist/cs-genericode-1.0/doc/oasis-code-list-representation-genericode.html.

Object Management Group, © 1997-2019. What is UML. In: *Unified Modeling Language* [online]. [cit. 2019-09-27]. Dostupné z: https://www.uml.org/what-is-uml.htm.

OntoUML Community, © 2017. OntoUML. In: *OntoUML Community Portal* [online].

Open Geospatial Consortium, © 2019a. Geography Markup Language. In: *OGC* [online]. [cit. 2019-06-17]. Dostupné z: http://www.opengeospatial.org/standards/gml.

Open Geospatial Consortium, © 2019b. Glossary of Terms – G. In: *OGC* [online]. [cit. 2019-06-17]. Dostupné z: http://www.opengeospatial.org/ogc/glossary/g.

Oracle, © 2019a. Database Concepts. Glossary. In: *Oracle Help Center* [online]. [cit. 2019-07-28]. Dostupné z: https://docs.oracle.com/cd/B28359_01/server.111/b28318/glossary.htm#g1004221.

Oracle, © 2019b. Oracle Spatial and Graph in Oracle Database 19c. In: *Oracle* [online]. [cit. 2019-06-17]. Dostupné z: https://www.oracle.com/a/tech/docs/spatial-and-graph-in-oracle-database-19c-preso.pdf.

Oracle, 2015. Char Vs Varchar. In: *Ask TOM* [online]. Last updated: October 31, 2015 [cit. 2019-07-11]. Dostupné z: https://asktom.oracle.com/pls/asktom/f?p=100:11:0::::P11_QUESTION_ID:2668391900346844476.

Oracle, 2013a. Oracle® Database SQL Language Reference 11g Release 1 (11.1). In: *Oracle Database Online Documentation 11g Release 1 (11.1)* [online]. July 2013 [cit. 2019-07-08]. Dostupné z: https://docs.oracle.com/cd/B28359_01/server.111/b28286.pdf.

SAP, n.d.a. BOMs (PP-BD-BOM). In: *SAP Help Portal* [online]. [cit. 2019-06-04]. Dostupné z: https://help.sap.com/viewer/43fa3cddb00348beb3ed12212a94448f/6.18.11/en-US/f56eb953495bb44ce10000000a174cb4.html.

SAP, n.d.b. PowerDesigner Standard Data Types. In: *SAP Documentation* [online]. [cit. 2019-07-11]. Dostupné z: https://help.sap.com/doc/saphelp_pd1655_data/16.5.5/en-US/c7/c632f46e1b101483c2c3a08082b7d7/frameset.htm.

SAYERS, Dominic, 2009. RFC Errata 1690. In: *RFC Errata Report* [online]. 2009-02-22 [cit. 2019-07-11]. Dostupné z: https://www.rfc-editor.org/errata/eid1690.

ŠKOP, Michal, KLÍMEK, Jakub, 2019. Adresy. In: *Otevřená data České republiky* [online]. 4. července 2019 [cit. 2019-07-17]. Dostupné z: https://data.gov.cz/otevřené-formální-normy/adresy/draft/.

TEOREY, Toby J. a kol., 2011. *Database modeling and design: logical design*. Fifth edition. Burlington: Morgan Kaufmann, [2011], ©2011. xiv, 304, [28] stran. The Morgan Kaufmann series in data management systems. ISBN 978-0-12-382020-4.

Universal Postal Union, n.d.a. About UN Specialized Agency. In: *Universal Postal Union* [online]. [cit. 2019-07-17]. Dostupné z: http://www.upu.int/en/the-upu/un-specialized-agency/about-un-specialized-agency.html.

Universal Postal Union, n.d.b. About UN Specialized Agency. In: *Universal Postal Union* [online]. [cit. 2019-07-17]. Dostupné z:

Universal Postal Union, n.d.c. S42 – easy access and implementation of international address format. In: *Universal Postal Union* [online]. [cit. 2019-07-17]. Dostupné z: http://www.upu.int/uploads/tx_sbdownloader/sheetAddressingS42InternationalAddressingStandardsFactSheetEn.pdf.

Universal Postal Union, 2019. Japan. In: *Universal Postal Union* [online]. 06/2019 [cit. 2019-07-17]. Dostupné z: http://www.upu.int/fileadmin/documentsFiles/activities/addressingUnit/jpnEn.pdf.

15 Přehled použitých zkratek

Tabulka 6: Přehled použitých zkratek

Zkratka	Význam
ČTÚ	Český telekomunikační úřad
FDM	Fyzický datový model
GML	Geography Markup Language
ITU	International Telecommunication Union
KDM	Konceptuální datový model
KS	Konceptuální schéma
OASIS CIQ V3.0	Customer Information Quality Specification Version 3.0
RÚIAN	Registr územní identifikace, adres a nemovitostí
SŘBD	Systém řízení báze dat
UPU	Universal Postal Union
VDP	Veřejný dálkový přístup
VFR	Výměnný formát RÚIAN
xAL	extensible Address Language
xNL	extensible Name Language

Stránky našeho nakladatelství
https://oeconomica.vse.cz/

Název	**Datové modelování a návrh relační databáze: Sbírka řešených úloh**
Autoři	Ing. Dušan Chlapek, Ph.D. Ing. Jan Kučera, Ph.D. RNDr. Helena Palovská, Ph.D.
Vydavatel	Vysoká škola ekonomická v Praze Nakladatelství Oeconomica
Doporučeno	pro bakalářské a magisterské studium na VŠE v Praze
Vydání	první
Návrh obálky	Daniel Hamerník, DiS.
Počet stran	168
DTP	Vysoká škola ekonomická v Praze Nakladatelství Oeconomica
Sazba	autoři

Tato publikace neprošla redakční úpravou.

ISBN 978-80-245-2331-6

Zdarma ke stažení