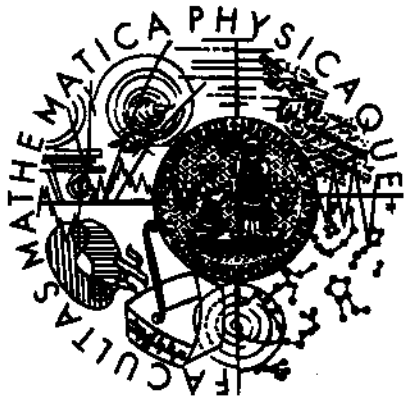
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Nora Ibrahimová

Editor ER diagramů

Katedra softwarového inženýrství

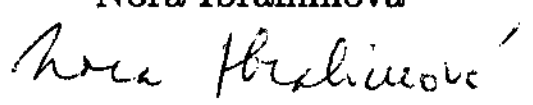
Vedoucí bakalářské práce: RNDr. Jakub Yaghob, Ph.D.
Studijní program: Informatika, Programování

2006

Prohlašuji, že jsem svou bakalářskou práci napsala samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 5.12.2006

Nora Ibrahimová



Obsah

Název práce: Editor ER diagramů
Autor: Nora Ibrahimová
Katedra (ústav): Katedra softwarového inženýrství
Vedoucí bakalářské práce: RNDr. Jakub Yaghob, Ph.D.
e-mail vedoucího: Jakub.Yaghob@mff.cuni.cz

Abstrakt: Tato práce popisuje implementaci editoru ER diagramů, programu pro výuku konceptuálního modelování. Program nazvaný *ER editor* umožňuje nakreslení ER diagramu a jeho převedení na SQL skript pro vytvoření schématu databáze, která diagramu odpovídá. Nástroj je schopen také reverzního inženýringu, tj. vygenerování ER diagramu podle SQL skriptu. Před oběma převody program verifikuje korektnost diagramu, resp. skriptu.

V práci jsou popsány algoritmy použité při převodu ER diagramu na SQL skript a při zpětném inženýringu, včetně způsobu rozvržení diagramu na ploše. Mimo jiné zde nechybí srovnání s podobnými nástroji.

Klíčová slova: ER diagram, návrh databáze, konceptuální modelování

Title: ER diagram editor
Author: Nora Ibrahimová
Department: Department of Software Engineering
Supervisor: RNDr. Jakub Yaghob, Ph.D.
Supervisor's e-mail address: Jakub.Yaghob@mff.cuni.cz

Abstract: The present work describes the implementation of an ER diagram editor, which is meant to help with conceptual modeling learning. The program called *ER editor* enables users to draw ER diagrams and generate accordant SQL scripts for database creation. This tool can reverse engineer an SQL script as well. Before generating both ER diagram and SQL script, the source script/diagram is verified.

The work also describes the theoretical principles of SQL script generation and reverse engineering, including the layout of diagram on canvas. It also features a comparison of the implemented program with similar programs, and other information.

Keywords: ER diagram, database design, conceptual modeling

Kapitola 1

Úvod

Práce s databází není nic neobvyklého pro většinu lidí, ať už s použitím počítače (např. firemní databáze kontaktů) nebo bez něj (tištěný telefonní seznam). Návrh databáze už ovšem tak automatická věc není. Aby byl úspěšný, je vhodné řídit se různými metodologiemi a využívat nástrojů, které jsou k usnadnění a zefektivnění této práce určeny.

Proces návrhu se skládá z několika fází, jak je popsáno dále. Nejprve je nutné shromáždit požadavky a ujasnit si, co vlastně databáze bude obsahovat, jak a k čemu bude využívána. Poté přijde na řadu vytváření vnitřní struktury databáze, aniž by ještě bylo nutné zabývat se samotnými tabulkami a záznamy. V této fázi se často kreslí tzv. ER diagram (z angl. Entity-Relationship). Díky němu je možné přehledně a jednoduše modelovat schéma databáze, bez ohledu na logické nebo dokonce fyzické uspořádání budoucí databáze.

Cílem práce byla implementace editoru ER diagramů. Vznikl program zvaný *ER editor*, který umožňuje nakreslení ER diagramu i určení vlastností jeho jednotlivých prvků. Dále je schopen verifikace schématu a jeho převodu na SQL skript s příkazy pro vytvoření odpovídající databáze, přičemž skript může být vytvořen dle normy ANSI SQL 92 nebo pro databázi Oracle. Program podporuje i tzv. zpětný inženýring, tj. načtení SQL skriptu (skriptu pro vytvoření databáze), jeho verifikaci a vygenerování a vykreslení odpovídajícího ER diagramu.

Program ER editor je určen k výukovým účelům. Zaměřuje se především na modelování tzv. logického schématu databáze, které je popsáno právě ER diagramem.

Následující tři kapitoly této práce se zabývají teoretickým základem

práce, nejprve návrhem databáze a ER diagramy. Pak jsou popsány algoritmy použité pro převod ER diagramu na SQL skript a zpětný inženýring. V další kapitole je popsána práce uživatele s ER editorem. Následující kapitola se zmiňuje o způsobu vyřešení některých důležitých problémů, které byly s implementací programu spojené. V předposlední kapitole je ER editor srovnán s podobnými programy. Na závěr jsou pak nastíněny možnosti jeho dalšího rozvoje.

Kapitola 2

Návrh databázového modelu

2.1 Vývoj databázového modelu

V době před dnešními relačními databázovými modely se používaly především dva modely [1] – *hierarchický databázový model* a *síťový databázový model.*

Hierarchický databázový model nabízel relativně rychlý přístup k datům a byla v něm zabudována referenční integrita. Na druhou stranu se zde často vyskytovala redundance dat a bylo v něm také velmi obtížné modelovat komplexnější vztahy. Tento model se hodil pro ukládání dat na kazetopáskových jednotkách, jaké se používaly v 70. letech na sálových počítačích.

Nedostatky hierarchického modelu se snažil odstranit *síťový databázový model* a jistě šlo o krok kupředu. Ještě zvýšil rychlost přístupu k datům a částečně odstranil zmíněné problémy předchozího modelu. Novou překážkou se ovšem stal fakt, že k získání dat z databáze bylo nutné znát strukturu databáze, což bylo nejen nepraktické, ale také to znamenalo, že bylo velmi obtížné strukturu změnit.

V současnosti nejrozšířenější databázový model je tzv. *relační databázový model.* Vychází ze dvou oborů matematiky: z teorie množin a predikátové logiky prvního řádu. Jeho název je odvozen od pojmu *relace* z teorie množin [1]. Relační databáze ukládá data v dobře známých tabulkách s poli, pomocí záznamů. Fyzické uspořádání databáze nehraje roli a každý záznam v tabulce je identifikován polem, které obsahuje unikátní hodnotu. Těmito vlastnostmi se model odlišuje od předchozích popsaných modelů. Databáze také lépe zajišťuje integritu a přesnost dat. Zná-li uživatel vztahy mezi tabulkami, může z databáze získávat data nejrůznějšími způsoby (a snadněji než

dříve) pomocí dotazovacího jazyka. Nakonec fakt, že model vychází z matematické teorie, ho činí předvídatelným, spolehlivým a solidním [1].

V celkem nedávné době byl vyvinut tzv. *objektově orientovaný model*, a to pro poněkud speciálnější potřeby (např. pro ukládání multimediálních dat). Tento model nemá základy v nějaké matematické teorii, ale přebírá vlastnosti objektově orientovaných jazyků. Jak popisuje [1]:

> „Hlavní myšlenkou (objektově orientovaného modelu) je, že vývojář databáze se stará o všechny aspekty databáze, včetně množin operací, které pracují s daty v databázi ... Neexistuje jasné rozdělení mezi databázovým programem a aplikačním softwarem."

Rozšířením relačního modelu o některé prvky objektově orientovaného modelu vznikl *objektově relační model*. Ten umožňuje v relační databázi používat i složitější datové typy, zavést některé bezpečnostní prvky nebo pracovat s vyšší vrstvou abstrakce.

Zde popisovaný program ER editor lze využívat při tvorbě schématu těch v současnosti nejrozšířenějších druhů databází, tj. relačních a objektově relačních databází.

2.2 Proč se zabývat návrhem databáze

Existuje mnoho nástrojů (a patří mezi ně i popisovaný ER editor), které automaticky generují SQL skripty pro vytvoření databáze, její úpravy apod. Tyto funkce ušetří čas, ale lze je využít, až když je navržena logická struktura databáze. Zanedbání v této části návrhu (nebo některé z dalších) povede ke ztrátě integrity, konzistence a spolehlivosti dat v databázi. Nepřesné údaje, které jsou toho důsledkem, pak mohou ovlivnit nejzákladnější fungování organizace, která databázi využívá [1]. Modelování pomocí diagramu, jako je ER diagram, tedy s pomocí obrázků, je názorné a přehledné a dovoluje návrháři soustředit se na skutečné problémy návrhu.

2.3 Fáze návrhu relační databáze

Při návrhu databáze je vhodné se řídit určitým zavedeným a ověřeným způsobem. Nejčastěji se celý proces dělí na tři základní fáze [8]:

1. *Konceptuální modelování* spočívá ve vytvoření modelu dat v databázi nezávisle na jejich uložení. Jde tedy o užitečnou formu abstrakce. Jedním z používaných druhů konceptuálního modelování je právě ER modelování, o kterém bude pojednáno dále.

2. Ve fázi *logického návrhu* se vytvořené konceptuální schéma, jako např. ER diagram, převede na struktury relační databáze, tj. tabulky obsahující pole, různá integritní omezení, vzájemné vztahy mezi tabulkami.

3. V konečné fázi *fyzického návrhu* se vybírá způsob uložení záznamů, použité přístupové metody, přidávají se indexy ke zrychlení práce s databází a další optimalizace.

2.4 ER model

Model entit a vztahů zavedl poprvé Peter Pin Shan Chen v roce 1967. Jeho ER diagramy se záhy staly obecně uznávanými [6]. Jednou z výhod modelu je jeho snadná pochopitelnost s minimem učení, díky čemuž může být použit i v komunikaci mezi návrhářem databáze a zákazníkem.

> „ER model je založen na chápání světa jako množiny základních objektů: *entit* (Entity) a *vztahů* (Relationship) mezi nimi. Popisuje data „v klidu", neukazuje, jaké operace budou s daty probíhat. Někdy se označuje také jako ERA - třetím základním prvkem modelu jsou *atributy* (Attributes)." [8]

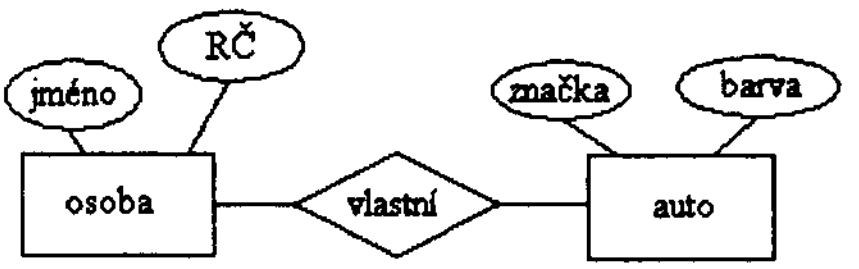
Entita je základní objekt ER modelu. Reprezentuje cokoliv, o čem je třeba v databázi uchovávat informace (např. člověk, zákazník, objednávka, kniha).

Vztah určuje, jak jsou jednotlivé entity v interakci, jak spolu souvisejí. Např. zákazník má (u firmy) nějakou objednávku; pak slovo *má* určuje vztah mezi entitami *zákazník* a *objednávka.*

Atribut určuje vlastnosti nebo skutečnosti o entitě, které mají být spolu s ní zaznamenány. Například entita člověk má atributy rodné číslo, příjmení a jméno. Atributy můžeme rozdělit na *identifikační* a *popisné*. Identifikační atributy (nebo klíče) jednoznačně identifikují entitu, zatímco popisné nesou informaci o entitě.

Jména typů entit a vztahů jsou v korektním ER diagramu jednoznačná globální jména, jména atributů jsou jednoznačná lokální jména.

Autor ER modelu Chen současně navrhl jeho zobrazení v diagramech, které nazval diagramy entit a atributů (Entity Relationship Diagrams, ER diagrams). V ER diagramech se entity zobrazují jako obdélníky, vztahy jako kosočtverce a atributy jako ovály, to vše propojeno čarami (viz obr.).



Obrázek 2.1: ukázka Chenovy notace

V programu ER editor je používaná notace jen mírně odlišná od původní notace a to podle toho, jak byla vyučována v kurzu Databázové systémy na Matematicko-fyzikální fakultě UK. Atributy jsou zde zobrazovány kruhem (vyplněným kruhem, jde-li o klíčový atribut) a název stojí vedle značky, nikoliv uvnitř.

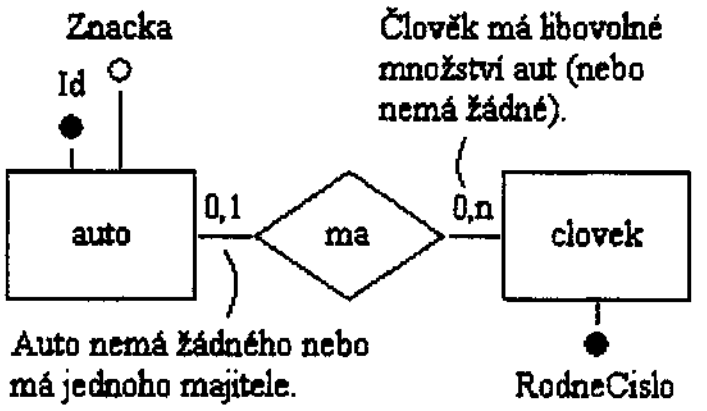
Vztah mezi entitami je charakterizován dvěma vlastnostmi: *kardinalitou* a *aritou*. Arita vztahu může být binární nebo ternární, podle toho, zda se na vztahu podílejí dvě nebo tři entity. Kardinalita vztahu souvisí s tím, s kolika entitami (instancemi) může být entita v daném vztahu. Obvykle jsou rozlišovány tyto kardinality:

(1,1) entita A může být ve vztahu *jedna ku jedné* pouze s jednou entitou B. Například každý občan má jeden cestovní pas a každý cestovní pas náleží jednomu občanovi.

(1,n) entita A může být ve vztahu *jedna ku n* s více entitami (B,C...), ale každá z entit B,C... může být ve vztahu jen s jednou entitou - s entitou A. Například každé oddělení může mít více zaměstnanců, ale zaměstnanec patří vždy do jednoho oddělení.

(m,n) entita A může být ve vztahu *n ku n* s více entitami (B,C...) a každá z entit B,C... může být ve vztahu s vícero entitami (A,D,E...). Příkladem může být zaměstnanec, který pracuje na více projektech a projekt, na kterém může pracovat více zaměstnanců [10].

V ER editoru se navíc vyskytují kardinality (0,1) a (0,n). Díky tomuto rozlišení je jasné, zda je pro entitu účast na vztahu povinná – případ (1,*), nebo nepovinná - případ (0,*). Příklad nepovinné účasti může být vztah (1,1) říkající, že každý dospělý člověk musí mít právě jeden občanský průkaz. Příklad nepovinné účasti: každý člověk může mít maximálně jeden cestovní pas (vztah (0,1)).



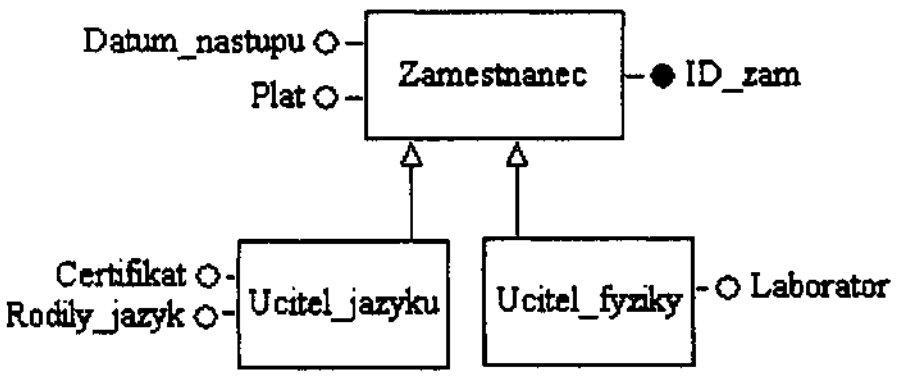
Obrázek 2.2: popis kardinality

Slabá entita je entita identifikačně závislá na jiné entitě (entitách). To znamená, že jí určují identifikátory jiné entity (entit). Pro příklad může posloužit slabá entita *DetailyObjednávky*, která závisí na entitách *Objednávka*, od které získá identifikační atribut *ČísloObjednávky*, a *Produkt*, od nějž získá identifikátor *ČísloProduktu*. Sama nese atributy *Množství* a *ProdejníCena*, které se vztahují k produktu v objednávce. Identifikační závislost implikuje existenční závislost, tedy slabá entita (přesněji její instance, která se ovšem také nazývá entitou) nemůže existovat, aniž by existovaly ty entity, na kterých závisí. Identifikační typy vztahů nesmí v ER diagramu tvořit orientovaný cyklus, jinak by nějaká entita byla identifikována pomocí sebe sama.

2.5 Generalizace v ER modelu

Ve výše uvedených základech ER modelu je entita popisována pomocí atributů, které popisují její vlastnosti. Někdy se ovšem objeví několik entit, které mají některé vlastnosti společné a některé rozdílné. Toto je jiný druh charakterizace entity, který využívá tzv. generalizace, známé také pod jmény ISA hierarchie, generalizace-specializace, dědičnost.

Příkladem mohou být učitelé ve škole. Všichni jsou zaměstnanci školy, mají tedy společné charakteristiky jako je doba nástupu, druh spolupráce, plat. Liší se však od sebe oborem, který učí, a ke kterému je třeba zaznamenávat další atributy. U učitele jazyků to může být druh získaného certifikátu, zda-li je rodilý mluvčí apod. Tyto údaje nemají u ostatních učitelů smysl. V tomto případě se vytvoří generalizační hierarchie (nebo ISA hierarchie), kde tzv. zdrojem hierarchie bude zaměstnanec školy a tzv. subtypy budou učitel jazyků a učitel fyziky (obr.). Subtypy „zdědí" atributy včetně identifikátorů od zdroje hierarchie. V diagramu se generalizace značí šipkou.



Obrázek 2.3: příklad generalizace

Generalizace se využívá zejména pokud se větší množství entit zdá být stejného typu (mají něco společného), nebo pokud se opakují některé atributy. Často se na ni narazí postupem času, kdy se model vyvíjí. Díky ní lze provádět změny pouze u těch entit, kterých se to skutečně týká, čímž zvyšuje konzistenci modelu, a v neposlední řadě přispívá k přehlednosti modelu [10].

Pro vytvoření korektní ISA hierarchie je třeba se řídit pravidly [5]:

- Žádná entita nemá v konceptuálním schématu více než jeden zdroj ISA hierarchie.
- ISA vztahy netvoří v ER diagramu orientovaný cyklus.
- Je-li entita zdrojem ISA hierarchie, pak má identifikační klíč. Ostatní typy v ISA hierarchii nemají identifikační klíč.
- Typ entity v ISA hierarchii, který není zdrojem, není identifikačně závislý na žádném entitním typu (je totiž již identifikován svým zdrojem ISA hierarchie)

Kapitola 3

Převod ER diagramu na SQL skript

Program ER editor nabízí kromě možnosti vytváření vlastních ER diagramů dvě další důležité funkce. První z nich je vygenerování SQL skriptu pro vytvoření schématu databáze, která nakreslenému ER diagramu odpovídá. Teoretickým řešením tohoto převodu se zabývá tato kapitola.

Na následujících řádkách bude popsán algoritmus pro vygenerování SQL skriptu pro vytvoření schématu databáze podle ER diagramu. Předpokládá se, že vstupní ER diagram je formálně korektní; v samotném programu se nejprve provádí verifikace diagramu. Verifikuje se unikátnost názvů entit a vztahů a názvů atributů patřících k jedné entitě. Dále se kontroluje, zda každá entita podílející se na vztahu má primární klíč, příp. že je u slabé entity určeno, na kom je závislá.

Pak se odstraní veškeré generalizační hierarchie. Dále se podle ER diagramu vygenerují tabulky se sloupci. V samotném programu je vygenerované schéma nakonec popsáno v jazyce SQL.

3.1 Odstranění generalizace

Nejprve je potřeba ošetřit generalizaci (také zvanou dědičnost, ISA hierarchie), pokud se v diagramu nachází. Tato hierarchie, týkající se několika entit, se převede na diagram z entit a vztahů. Pak již přijde ke slovu algoritmus, který zpracovává entity, vztahy a atributy.

Možnosti pro „rozbití“ generalizační hierarchie na základní objekty ER diagramu (entity, vztahy a atributy) jsou v zásadě tři. Budou popsány po-

mocí velmi jednoduchého příkladu, kde zdrojem hierarchie je entita *Člověk*, od níž „dědí“ vlastnosti entity *Zaměstnanec* a *Zákazník*. Veškeré entity, které jsou součástí generalizační hierarchie a nejsou jejím zdrojem budou dále nazývány subtypy podle anglického vzoru.

1. V prvním ze způsobů převodu je zachován zdroj hierarchie, v příkladě tedy entita *Člověk*. Kromě vlastních atributů ponese i atributy všech subtypů, tj. entit *Zaměstnanec* a *Zákazník*. Navíc bude mít ještě atribut říkající, o který druh *Člověka* se jedná (zda o *Zaměstnance* nebo o *Zákazníka*). Dále se přidají integritní omezení říkající, že jde-li např. o druh *Zaměstnanec*, pak se vyplňují atributy, které původně patřily této entitě a nikoliv atributy, které patřily entitě *Zákazník*, a obráceně. Entita se bude podílet i na všech vztazích, na kterých se podílely subtypy.

   Tento způsob převodu by byl vhodný pouze v případě, že se subtypy od sebe téměř neliší. V ostatních případech je to způsob nepraktický, který navíc rozhodně nepřispívá k přehlednosti struktury databáze a odporuje jednomu ze základních principů a cílů, který platí nejen pro návrh databází [3]:

   > „Manage complexity.“

2. Druhý způsob spočívá v odstranění zdroje hierarchie. Všem subtypům hierarchie jsou pak předány atributy zdroje i vztahy, na kterých se zdroj podílel. V uvedeném příkladě by tak zůstaly entity *Zaměstnanec* a *Zákazník* a každá z nich by nesla atributy původně patřící k entitě *Člověk*, jako např. identifikátor *RodnéČíslo*. Pro oba subtypy by se také duplikovaly vztahy entity *Člověk*, jako např. vztah *bydlí* s entitou *Adresa*. Jak je vidět, je tento způsob vhodný v případě, že zdroj hierarchie obsahuje velmi málo atributů a podílí se na velmi málo nebo žádném vztahu (např. pokud nese pouze identifikační atribut). V ostatních případech vede ke zbytečné duplikaci a opět nepřispívá k přehlednosti a v důsledku ani ke snadné udržovatelnosti dat.

3. Ve třetím způsobu jak odstranit generalizační hierarchii se zachovávají všechny zúčastněné entity, jak zdroj tak i jeho subtypy. Mezi zdrojem hierarchie a jeho subtypy se pak vytvoří vztahy, které původní hierarchii charakterizují. Ze subtypů se stanou slabé entity, jsou tedy identifikačně i existenčně závislé na zdroji hierarchie. Kardinalita

vztahu ze strany zdroje je (0,1), tedy každá instance zdroje může mít vztah s max. jedním subtypem. Kardinalita ze strany subtypu musí být (1,1), tato slabá entita je identifikována právě jednou zdrojovou entitou. Dále už se entity samozřejmě stávají běžnými entitami, popisy zdroj a subtyp již nemají smysl. V příkladě se tedy z entit *Zaměstnanec* a *Zákazník* stanou slabé entity, závisející na entitě *Člověk*, a jsou tudíž identifikovány jejím identifikátorem (atributem *RodnéČíslo*). Všem entitám zůstanou jejich atributy i vztahy, na kterých se podílejí. Tento způsob převodu se zdá nejvíce univerzálním, zachovává výhody generalizační hierarchie a je v souladu s požadavkem přehlednosti a snadnější udržovatelnosti struktury databáze. Je proto používán v programu ER editor.

3.2 Algoritmus převodu ER diagramu na tabulky

Podle ER diagramu bez generalizací se už vygenerují tabulky se sloupci, které diagramu odpovídají. Používaný algoritmus je následující (pořadí následujících dvou bodů může být libovolné, akce jsou na sobě nezávislé):

- Zpracuj každý ze vztahů v diagramu.
  Spolu se vztahem budou zpracovány i entity, které se na vztahu podílejí, a jejich atributy.
- Zpracuj entity, které se na žádném vztahu nepodílejí.

Způsob zpracování vztahu záleží na kardinalitách čar, které vedou od vztahu k jednotlivým entitám. Pro jednoduchost bude popisováno zpracování binárního vztahu. U ternárního vztahu je zpracování obdobné, pouze se zúčastňuje více entit. Případy se tedy liší podle toho, jakou kardinalitu má každá ze dvou čar vedoucí ke vztahu.

- *(1,1) – (1,1)*
  Pro vztah, na kterém se obě entity podílejí s kardinalitou (1,1), se vytvoří jedna společná tabulka, se sloupci podle atributů obou entit, a kde klíčem bude identifikátor jedné (libovolně vybrané) entity. Identifikátor druhé entity zůstane kandidátním klíčem.

- *(1,1) - (0,1)/(0,n)/(1,n)/(m,n)*
  Nechť se entita A podílí na vztahu kardinalitou (1,1) a entita B některou z jiných kardinalit. Vytvoří se tabulka podle entity A, která bude navíc obsahovat cizí klíč odkazující na entitu B. Pro entitu B se vytvoří samostatná tabulka.

- *(0,1) - (0,1)/(0,n)/(1,n)/(m,n)*
  Nechť se entita A podílí na vztahu kardinalitou (0,1) a entita B některou z vyjmenovaných kardinalit. Pro obě entity (A i B) se vytvoří jejich vlastní tabulka (tabulka A a B). Pro jejich vztah se vytvoří tabulka s cizími klíči odkazujícími na tabulky A a B, ve které bude primárním klíčem ten cizí klíč, který odkazuje na tabulku A.

- *dvojice kardinalit (0,n)/(1,n)/(m,n)*
  Nechť má entita A i entita B některou z vyjmenovaných kardinalit. Opět se pro obě entity vytvoří jejich vlastní tabulka (tabulka A a B). Pro vyjádření vztahu se vytvoří „vztahová" tabulka obsahující cizí klíče odkazující na tabulku A i B, které jsou zároveň primárními klíči.

Poznámky k popsanému zpracování:

- Vytvořit samostatnou tabulku podle nějaké entity znamená, že podle každého atributu patřícího k entitě vznikne sloupec stejného jména, datového typu a integritních omezení, a stane se klíčovým sloupcem, pokud vzniká z identifikačního atributu a pokud není řečeno jinak (může se stát pouhým kandidátním klíčem). Tabulka se podle entity pojmenuje.

- Podle každé entity je vytvořena nejvýše jedna samostatná tabulka. Tedy bez ohledu na to, na kolika různých vztazích se entita podílí, se její tabulka neduplikuje.

- Tvoří-li se tabulka podle slabé entity, získává navíc ty klíčové sloupce, které mají tabulky vzniklé z entit, na kterých je slabá entita závislá.

- Tabulky vytvořené podle vztahu mají název tohoto vztahu a obsahují jen sloupce, které byly v popisu algoritmu vyjmenované.

- Sloupce, které byly označeny jako kandidátní klíč, budou mít integritní omezení UNIQUE a NOT NULL, ale nebudou skutečně klíčem.

Další bod algoritmu, tj. zpracování entit, které se na vztazích nepodílejí, znamená podle každé z takovýchto entit vytvořit samostatnou tabulku (způsobem, který je popsán výše).

Kapitola 4

Zpětný inženýring

Vedle převodu ER diagramu na SQL skript pro vytvoření schématu databáze nabízí ER diagram opačný převod – tzv. zpětný inženýring, tj. vykreslení ER diagramu, který odpovídá danému SQL skriptu (skriptu s příkazy pro vytvoření schématu databáze, tedy CREATE TABLE ...). Tato kapitola se zabývá teoretickým základem zpětného převodu.

Při zpětném inženýringu se nejprve verifikuje SQL skript. U SQL skriptu se hlídá, zda jsou názvy tabulek a sloupců v rámci tabulky unikátní a zda cizí klíče odkazují na existující tabulky. Pak se načtou tabulky z příslušného SQL skriptu, podle nich se pak vytvoří objekty ER diagramu.

Analyzuje se postupně každá tabulka a rozlišují se případy:

- *Tabulka nemá primární klíč.* Pokud má tabulka nějaký cizí klíč, považuje se to za chybu – tabulka se podílí na vztahu, ale chybí jí primární klíč. Pokud nemá ani cizí klíč, vytvoří se podle tabulky entita.

- *Žádný z primárních klíčů tabulky není cizím klíčem.* Tabulka představuje entitu. Může mít jiné cizí klíče (různé od primárních klíčů), což znamená, že je z ní vytvořená entita ve vztahu s dalšími entitami. Vytvoří se příslušný vztah s entitami vytvořenými z tabulek, na které se cizí klíč odkazuje.

- *Všechny primární klíče jsou i cizí klíče.* Jde o tabulku určující vztah mezi entitami. Pokud má tabulka nějaké neklíčové atributy, jde o atributy vztahu, pro které se ke vztahu připojí další slabá entita, která atributy vztahu ponese.

- *Některé primární klíče jsou zároveň cizími klíči, ale ne všechny.* Jde

o zvláštní „hybridní“ tabulku, která určuje vztah s entitami, na které cizí–primární klíče odkazují a zároveň entitu připojenou k témuž vztahu, jejíž cizí klíče, které nejsou primárními, mohou určovat další vztahy s jinými entitami.

Vytvořit entitu z tabulky A znamená vznik entity stejného jména. Za každý sloupec, který není cizím klíčem, se vytvoří atribut stejného jména, dat. typu a integritních omezení a připojí se k entitě. Byl-li sloupec primárním klíčem, půjde o identifikační atribut. Za každý cizí klíč se vytvoří vztah s entitou vzniklou z tabulky, na kterou cizí klíč odkazuje. Kardinalita ze strany tabulky A je (1,1), ze strany entit, na které je odkazováno, pak obecná (0,n).

Vzniká-li z tabulky A vztah, záleží kardinality čar z něj vycházejících na shodě primárních a cizích klíčů. Pokud jsou všechny cizí klíče jsou zároveň primárními klíči, jde o vztah, kde kardinality všech čar z něj vycházejících jsou obecné (0,n). Pokud se však objeví cizí klíč odkazující na tabulku B, který není primárním klíčem, pak entita B je připojena ke vztahu kardinalitou (0,n), ale entita A je připojena kardinalitou (0,1). Jasněji to vyplývá z popisu převodu ER diagramu na tabulky (tedy z opačného převodu), který je popsán v předchozí kapitole.

Při zpětném inženýringu není přímo rozpoznána případná generalizační hierarchie.

4.1 Rozvržení ER diagramu na ploše

Po vytvoření objektů ER diagramu (entit, vztahů a atributů) je třeba všechny rozmístit na danou plochu. Následuje popis tohoto rozvržení v ER editoru.

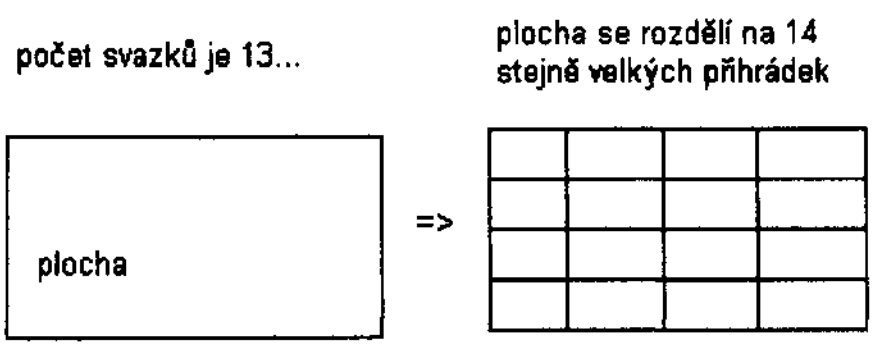
Způsobů, jakými je možno rozmístit entity, vztahy a diagramy po ploše určené k jejich vykreslení je velké množství. Navržená a použitá metoda není příliš složitá, přitom odpovídá obvyklému způsobu kreslení diagramů (čáry bývají rovné, příp. lomené, výjimečně diagonální). Metoda je navržena pro vykreslení ER diagramu nijak závratné velikosti, jelikož program ER editor je koncipován jako výukový program.

Metodu lze podrobněji popsat v několika krocích:

- Pro každý vztah v diagramu se vytvoří tzv. *svazek* (z angl. *bundle*), do kterého patří vztah a entity, které jsou ke vztahu připojeny a které zatím do žádného svazku nepatří. Pokud některá entita už patří do

jiného svazku, a přitom se podílí na tomto vztahu, pouze se to zapamatuje jako propojení s jiným svazkem. Pro každou entitu, která se na žádném vztahu nepodílí, se nakonec také vytvoří samostatný svazek.

- Obdélníková plocha určená k rozmístění objektů se rozdělí na tolik stejně velkých obdélníkových přihrádek, kolik je svazků (příp. se počet přihrádek zaokrouhlí na nejbližší vyšší násobek čtyř nebo pěti, aby se na ně plocha dala rozdělit, některé pak zůstanou volné). Tvar přihrádek odpovídá tvaru kreslící plochy, je-li větší na šířku, budou i přihrádky větší na šířku, stejně tak na délku. Počet přihrádek je omezen.



Obrázek 4.1: rozdělení plochy na přihrádky

- Každému svazku je přidělena jedna přihrádka. Algoritmus přidělování je následující: Pro všechny svazky se spočítá počet jejich propojení s jinými svazky. Ten s nejvíce propojeními získá první přihrádku (přihrádky jsou uspořádané zleva doprava a shora dolů), o další se pak podělí ty svazky, se kterými je právě umístěný svazek propojen. K nim se pak přidají další s nimi propojené atd. Pokud už žádné další svazky propojené s již umístěnými nejsou, umístí se opět ten svazek, který má ze zbývajících neumístěných nejvyšší počet propojení. Algoritmus pokračuje, dokud nejsou umístěny všechny svazky. V každém kroku tohoto algoritmu se umístí jeden svazek, je tudíž konečný. Tento způsob rozdělení přihrádek se snaží, aby propojené svazky byly pokud možno u sebe a odpovídá tomu, že velikost celého diagramu je omezená.

- Každý svazek se vykreslí do své přihrádky. Doprostřed přihrádky je umístěn vztah, kolem něj rozmístěny entity, které do svazku patří, se svými atributy.

- Nakonec se pomocí rovných a lomených čar propojí jednotlivé svazky. Každá entita, která má být spojena čarou se vztahem patřícím do jiné přihrádky, je s ním propojena rovnou nebo zalomenou čarou, podle toho, zda se nacházejí v bezprostřední blízkosti nebo dále od sebe. Tím je celý diagram rozmístěn.

Z popsaného algoritmu plyne, že vykreslení diagramu záleží na aktuálním tvaru kreslící plochy.

Kapitola 5

Práce s ER editorem

5.1 Požadavky na OS a instalace programu

Program ER editor je určen pro operační systém Windows. Vyžaduje verzi Windows 98 nebo vyšší.

Instalace programu je velmi snadná. Uživatel si jen rozbalí soubor *ER editor 1.0.zip*. Získá tak adresář *ER editor 1.0*, v němž ve složce *ER editor* nalezne spustitelný program *ER editor.exe*. Dokumentace se nachází ve stejně pojmenované složce.

Pokud by se náhodou uživateli nezobrazila nápověda po kliknutí na příslušnou položku v menu programu, nechť se ujistí, že se soubor *napoveda.txt* nachází ve stejném adresáři jako samotný spušťený program.

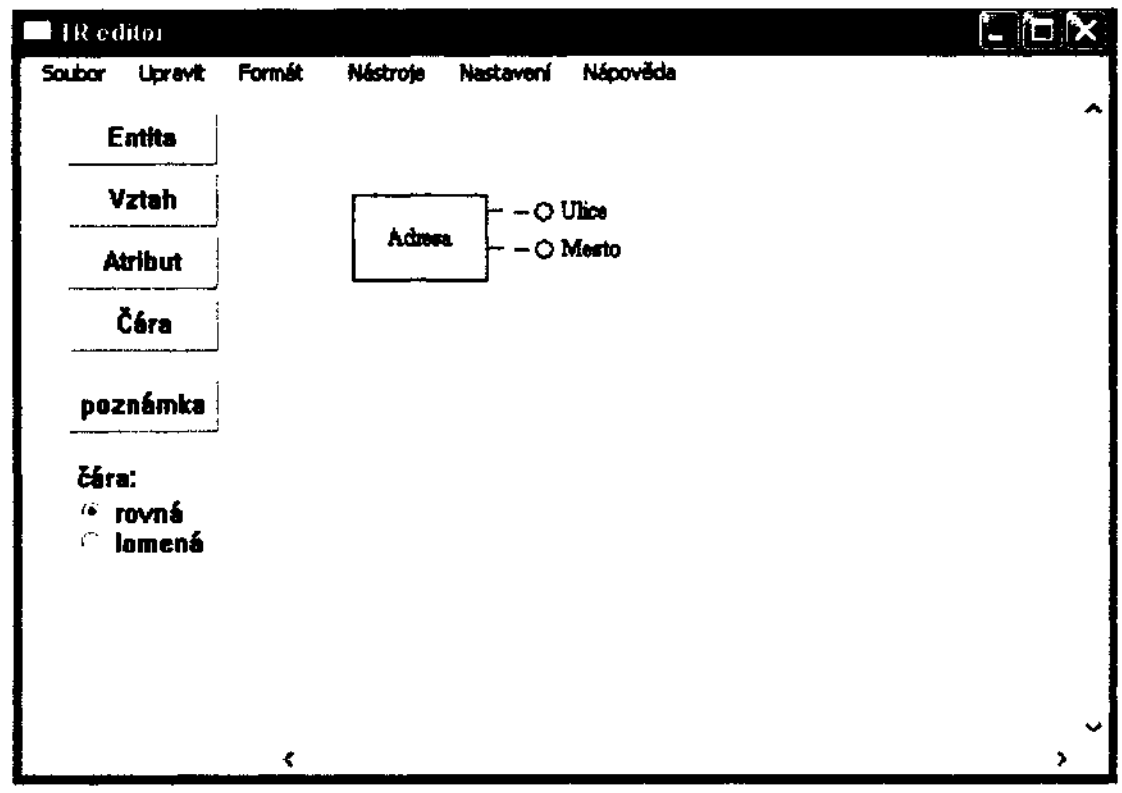
5.2 Kreslení ER diagramů

Okno programu je rozděleno na dvě části. Vlevo se nachází panel s tlačítky a vpravo plocha určená ke kreslení diagramu.

5.2.1 Entity, vztahy a atributy

Pro nakreslení entity, vztahu nebo čáry je třeba zmáčknout příslušné tlačítko a pak namalovat objekt[1] táhnutím a puštěním myši (táhne se směrem dolů doprava), přičemž se obrys budoucího objektu ukazuje přerušovanou čarou. Takto lze nakreslit objekt požadované velikosti. Nebo je možno jednoduše

[1] slovo *objekt* bude dále používáno jako souhrnné označení pro entity, vztahy a atributy



Obrázek 5.1: okno ER editoru (zde černobíle)

dvojitým kliknutím vytvořit objekt standardní velikosti, která je stejná pro entitu a vztah a menší pro atribut.

K entitě lze později připojit libovolné množství čar, ke vztahu tolik čar, kolik dovoluje jeho arita (pro nastavení arity viz sekce *Změna vlastností objektů a čar*) a k atributu lze připojit maximálně jednu čáru.

5.2.2 Čáry

Vztahy, entity a atributy se spojují čarami. K nakreslení čáry se zmáčkne tlačítko *Čára*, na panelu lze také vybrat, zda se bude kreslit čára rovná nebo zalomená.

Rovná čára se kreslí tažením a puštěním myši, přičemž je vidět vznikající čáru. Pokud se konce nakreslené čáry nacházejí nad objektem, k němuž lze čáru připojit (dovoluje to jeho arita), pak se objekt zvýrazní modře a po vytvoření se příslušný konec čáry k němu připojí. Takto lze tedy jednoduše spojit dva objekty rovnou čarou, stisknutím (levého tlačítka) myši nad prvním objektem, tažením nad druhý objekt a puštěním myši.

Pro nakreslení lomené čáry se klikne levým tlačítkem myši vždy v místě, kde má být zalomena (čára se přitom postupně zobrazuje), nakonec stisk-

nutím pravého tlačítka se kreslení ukončí a lomená čára je hotová.

Na nakreslené čáře se zobrazují až tři malá modrá okénka – jedno uprostřed čáry a po jednom na každém volném (nepřipojeném) konci čáry. Ta slouží k manipulaci s čarou (podrobnosti dále).

5.2.3 Pohyb

Nakresleným objektem lze pohybovat stisknutím (levého tlačítka) myši, tažením na cílové místo a puštěním myši. Při pohybu je nová pozice objektu naznačena přerušovanou čarou. S objektem se přemístí i čáry, které jsou k němu připojené.

U čáry lze pohybovat jejími volnými (nepřipojenými) konci. Na volném konci čáry je malé modré okénko, které se chytne a přesouvá pomocí myši, podobně jako entity a ostatní objekty. Pokud je čára připojená na obou koncích, už s ní hýbat nelze.

5.2.4 Označení a mazání

Není-li stisknuté žádné tlačítko na panelu vlevo, pak lze označit více objektů a čar najednou a to klasickým způsobem známým např. z operačního systému Windows. Stisknutím (levého tlačítka) myši, táhnutím a puštěním se označí všechny objekty a čáry, které se celé nachází v takto vyznačené oblasti. Pokud pak budeme jedním z označených objektů hýbat, pohnou se s ním i ostatní označené objekty a čáry.

Jeden objekt se označí (jednoduchým) kliknutím na něj, čára pak kliknutím na některé z jejích okének. Označené objekty jsou červeně zvýrazněné.

Cokoliv je označené lze smazat stisknutím klávesy Delete nebo vybráním položky v menu *Úpravy - Vymazat.*

5.2.5 Změna vlastností objektů a čar

Po dvojitém poklepání (levým tlačítkem) myši na objekt nebo na některé z okének čáry se zobrazí dialogové okno, v němž lze nastavovat některé vlastnosti, jako rozměry objektu, tloušťka čáry, a vlastnosti specifické pro jednotlivé objekty (název, arita vztahu, zda je entita slabá...) a pro čáry (kardinalita čáry apod.). Změny se potvrdí tlačítkem *OK* nebo zruší tlačítkem *Storno.*

Jméno entit a vztahů, zadané v dialogovém okně, se zobrazuje uvnitř objektu, jméno atributu pak vedle něj. Jelikož se ze jmen objektů stávají jména tabulek při generování SQL skriptu, akceptuje se pouze název, který je *regulárním identifikátorem* podle normy ANSI SQL 92, tzn. musí začínat písmenem a smí obsahovat pouze písmena, čísla a podtržítko. Mezery nejsou povoleny [2]. Délka názvu je omezena na 30 znaků, a to s ohledem na přehlednost diagramu i na fakt, že toto omezení někdy vyžaduje databáze Oracle.

5.2.6 Použití poznámek

Na levém panelu se také nachází tlačítko *Poznámka*, po jehož stisknutí lze vytvořit okno pro poznámky podobně, jako se vytváří objekty, tj. stisknutím, tažením a puštěním levého tlačítka myši. Okno je určené k psaní poznámek; jeho velikost lze měnit tažením za některý z jeho rohů. Poznámka sama zmizí, jestliže je prázdná a uživatel se jí už nevěnuje (např. vybírá z menu, stiskne jiné tlačítko apod.).

5.2.7 Vytvoření generalizační hierarchie

Hierarchie generalizace (ISA hierarchie) se vytvoří jednoduše propojením dvou entit čarou. Čára se automaticky změní v šipku mířící nahoru, což znamená, že entita, která byla výše, se stanem zdrojem hierarchie, a entita položená níže bude subtypem.

5.2.8 Self relace

Při tzv. *self relaci* je entita připojena ke vztahu dvakrát, např. v případě entity *Zaměstnanec* a vztahu *je_nadřízený* se self relací vyjádří, že zaměstnanec je nadřízený jinému zaměstnanci (resp. zaměstnancům). To se udělá jednoduše tak, že od vztahu povedou dvě čáry, obě ke stejné entitě.

5.3 Položky menu

5.3.1 Uložení a otevření diagramu

V menu pod se pod položkou *Soubor* nabízejí běžné možnosti uložení souboru, vytvoření nového souboru, otevření a zavření souboru a ukončení pro-

gramu. Většinu položek v menu lze vyvolat také stisknutím příslušné klávesové zkratky. U těchto položek jde o klávesové zkratky známé ze systému Windows, jako Ctrl+S pro uložení souboru apod. Soubory s ER diagramem se ukládají ve formátu **.erd*, který lze opět otevřít pouze tímto programem.

5.3.2 Vrácení/opakování akce, mazání

Pod položkou *Upravit* se nacházejí možnosti opakovat nebo vrátit zpět právě provedenou akci. Počet možných vrácení zpět je omezený.

Nachází se zde i již zmíněná volba *Vymazat*, která smaže všechny označené objekty a čáry (jsou červeně zvýrazněné).

5.3.3 Formát

Pod touto volbou se skrývá možnost *Náhled*, která v novém okně zobrazí náhled na celý právě rozpracovaný ER diagram. Dále tu lze nastavit písmo používané v diagramu např. pro názvy objektů a vybrat nebo zrušit možnost skrývání okének čar. Pokud je kreslený diagram hotový, nejsou již okénka čar potřeba a diagram vypadá přehledněji a lépe, když se skryjí.

5.3.4 Nástroje

Pod volbou *Nástroje* se nacházejí tři důležité funkce. První z nich, *Kontrola spojení*, ověří, zda jsou v aktuálním diagramu všechny čáry připojeny z obou stran, všechny atributy připojeny k nějaké entitě a všechny vztahy propojeny s ohledem na jejich aritu. Pokud tomu tak není, ohlásí se, které objekty nebo čáry nejsou připojené.

Volba *Převést ER diagram na SQL skript* nejprve zkontroluje propojení diagramu, jako při vybrání volby *Kontrola spojení*, a pak jeho korektnost. Verifikuje se unikátnost názvů entit a vztahů a názvů atributů patřících k jedné entitě. Dále se kontroluje, zda každá entita podílející se na vztahu má primární klíč, příp. že je u slabé entity určeno, na kom je závislá.

Poté v novém okně zobrazí SQL skript pro vytvoření schématu databáze, odpovídající nakreslenému diagramu. Skript lze uložit použitím menu, které se v nachází v novém okně se skriptem. SQL skript se vypíše podle pravidel aktuálně nastavené normy (ANSI SQL nebo Oracle). Normu lze nastavit v menu pod volbou *Nastavení*.

Volba *Reverzní inženýring* umožňuje podle SQL skriptu pro vytvoření schématu databáze vygenerovat příslušný ER diagram, který se zobrazí v okně programu.

U SQL skriptu se hlídá, zda jsou názvy tabulek a sloupců v rámci tabulky unikátní a zda cizí klíče odkazují na existující tabulky. Pokud skript obsahuje syntaktickou chybu, reverzní inženýring neproběhne.

5.3.5 Nastavení

Zde lze vybírat mezi dvěma normami, ANSI SQL 92 a Oraclovskou. Rozdíl se projeví v SQL skriptu generovaném při převodu ER diagramu na SQL skript a ve výběru datových typů atributu.

5.3.6 Nápověda

Po vybrání této volby se zobrazí okno se stručným přehledem, jak kreslit jednotlivé prvky ER diagramu.

Kapitola 6

Vybraná řešení problémů při návrhu programu

6.1 Nejdůležitější třídy

Program ER editor byl napsán v objektově orientovaném jazyce C++. Následuje popis těch nejdůležitějších tříd, které nesou informace o ER diagramu.

Pro entitu, vztah i atribut je potřeba samostatná třída vzhledem k tomu, že se v mnoha vlastnostech a způsoby chování od sebe liší. Mají ovšem i něco společného - jde o objekty, které se zobrazují na ploše, mají své rozměry, souřadnice, svůj název apod. Byla proto navržena abstraktní třída *Object*, která je předkem tříd *Entity* (entita), *Relationship* (vztah) a *Attribute* (atribut). Třída *Object* je čistě abstraktní třídou (obsahuje totiž čistě abstraktní metody), nelze tudíž vytvořit její instanci. Třídy *Entity*, *Relationship* a *Attribute* implementují její metody a rozšiřují ji o další atributy a metody.

U čar je situace obdobná. Od abstraktní třídy *Line* (čára) dědí třídy *StraightLine* (rovná čára) a *FractionalLine* (lomená čára), protože mají některé společné a jiné rozdílné vlastnosti.

Další podstatnou třídou je třída *Scene* (scéna), která v kontejnerech shromažďuje veškeré entity, vztahy a atributy pomocí ukazatelů na třídu *Object* a rovné a lomené čáry pomocí ukazatelů na třídu *Line*. Stará se i o převody mezi ER diagramem a SQL skriptem a další funkce, nabízené v menu programu.

Hierarchie tříd splňuje takzvaný substituční princip Liskovové (LSP, Liskov Substitution Principle [3]):

> V jedné práci o objektově orientovaném programování Barbara Liskovová uvádí, že by se mělo dědit ze třídy–předka, pouze pokud třída–potomek skutečně a doslova *je* speciální případ třídy–předka. Andy Hunt a Dave Thomas shrnuli pravidlo LSP takto: „Potomky lze používat skrze rozhraní jejich předka, aniž by byl zřejmý rozdíl."

6.2 Použití oken

Program hojně využívá funkcí Win32 API, mezi které patří i funkce pro vytváření a práci s okny. Jedním ze základních problémů při návrhu programu byl problém rozvržení oken. Ukázalo se, že je vhodné rozdělit hlavní okno aplikace na dvě dceřinná okna: jedno pro panel s tlačítky umístěný v levé části hlavního okna, druhé pro kreslící plochu napravo.

Tím se ale možnosti dceřinných oken zdaleka nevyčerpaly. Pro každou entitu, vztah nebo atribut je vytvořeno vlastní dceřinné okno ve tvaru, ve kterém se má objekt zobrazovat (např. vztah - kosočtverec). To umožňuje snadno rozlišit, zda uživatel pracuje některým konkrétním objektem, nebo s celým diagramem. Odděluje to části programu, které logicky oddělené být mají, což vede k přehlednějšímu a snadněji udržovatelnému kódu. Ze stejných důvodů jsou i okénka čar na volných koncích a uprostřed čáry skutečná samostatná okna.

6.3 Převod mezi datovými typy ANSI SQL a Oracle

ER editor nabízí možnost pracovat s normou ANSI SQL 92 nebo podle standardů Oracle (verzí 7, 8 a 9i). Ty se mírně liší v generovaném SQL skriptu: podle normy ANSI je třeba dávat názvy integritních omezení do apostrofů, u Oracle tomu tak není. Další problém nastane, rozhodne-li se uživatel přepnout mezi normami v době, kdy má již vytvořené nějaké atributy, a je tudíž potřeba převést jejich datový typ z jedné normy do druhé.

Způsob převodu mezi normami, zobrazený v následující tabulce, byl navržen s využitím zdrojů [7] a [9].

ANSI SQL	na	Oracle
INTEGER		NUMBER(38)
SMALLINT		NUMBER(5)
FLOAT(p)		FLOAT(p)
DOUBLE PRECISION		FLOAT(49)
REAL		FLOAT(23)
NUMERIC(p,s)		NUMBER(p,s)
DECIMAL(p,s)		NUMBER(p,s)
CHARACTER(l)		CHAR(l)
CHAR VARYING(l)		VARCHAR2(l)
NCHAR(l)		NCHAR(l)
NCHAR VARYING(l)		NVARCHAR2(l)
DATE		DATE
TIME(p)		DATE
TIMESTAMP(p)		TIMESTAMP(p)
BIT(l)		RAW(l)
BIT VARYING(l)		RAW(l)

Oracle	na	ANSI SQL
NUMBER (p,s)		NUMERIC(p,s)
FLOAT(p)		FLOAT(p)
ROWID		NUMERIC(38)
VARCHAR2(l)		CHAR VARYING(l)
NVARCHAR2(l)		NCHAR VARYING(l)
VARCHAR(l)		CHAR VARYING(l)
CHAR(l)		CHARACTER(l)
NCHAR(l)		NCHAR(l)
DATE		DATE
TIMESTAMP(p)		TIMESTAMP(p)
RAW(l)		BIT VARYING(l)
LONG		BIT
LONG RAW		BIT

Kapitola 7

Srovnání s několika podobnými programy

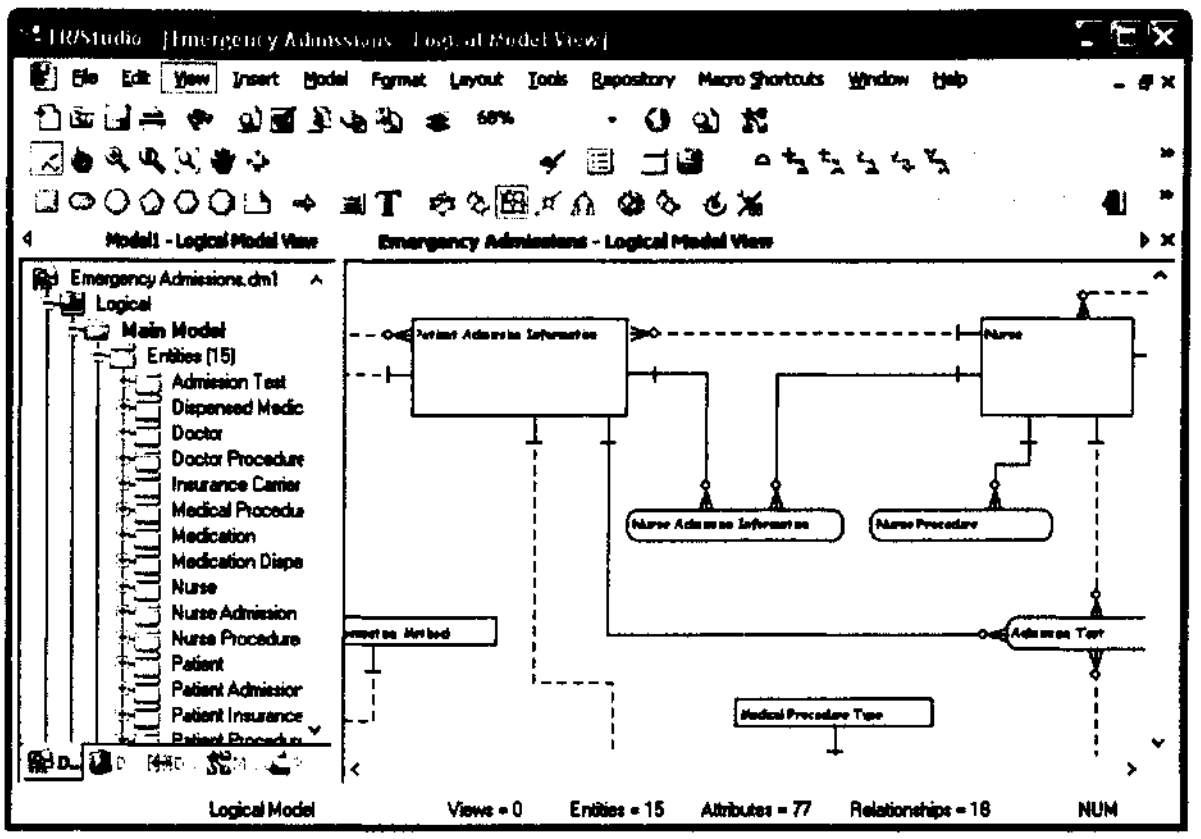
Na trhu lze najít nemálo editorů ER diagramu, podporujících převod na SQL skript i zpětný inženýring. Existují programy komerční jako *ERWin*, *ER/Studio* nebo *Toad Data Modeler*, nebo obecnější, podporující více druhů modelování, jako *Rational Rose* nebo či *Oracle Designer*. Bývají robustnější, obvykle pracují s UML notací a dalšími notacemi odlišnými od původní Chenovy, nabízejí mnoho různých funkcí. K nekomerčním programům patří GNU Ferret a nástroje Sunicat a ER Modeller, které oba vznikly na ČVUT formou několika diplomových a bakalářských prací.

7.1 ER/Studio Enterprise 7.1

ER/Studio firmy *Embarcadero Technologies* si lze vyzkoušet stažením 14-ti denní verze z webové adresy [11]. Studio lze používat jako samostatného klienta, nebo jej připojit k centrálnímu archivu (repository). Podporuje zabezpečení modelu pomocí práv a rolí.

ER/Studio nabízí vytváření ER diagramu v několika notacích, žádná z nich ale není podobná původní Chenově notaci. Novou entitu lze stejně jako v ER editoru vytvořit a umístit kliknutím myší a její vlastnosti upravit po dvojitém kliknutí na její obrázek. V okně s vlastnostmi entity se přidávají i její atributy. Entity se pak propojují čarami, které se liší kardinalitou a tím, zda jde o identifikační vztah či ne. Je možné přidat objekt generalizační hierarchie.

ER/Studio umožňuje generování fyzického modelu podle diagramu i zpětný



Obrázek 7.1: ER/Studio

inženýring z SQL skriptu i přímo z databáze. Nabízí různé způsoby rozvržení (layout) diagramu (např. hierarchický, cyklický, ortogonální, stromový...), které jsou velmi užitečné při zpětném inženýringu. K lepší orientaci dále přispívají submodely, barevné odlišení částí diagramu nebo okénko s náhledem na celý diagram.

Novinkou verze 7.1 z roku 2006 je možnost navrácení a opakování akce, což je důležitá vlastnost, která dříve chyběla. Další novinkou je možnost plánování kapacity, což pomůže architektům naplánovat diskové požadavky systému. Po zadání nejrůznějších parametrů pro odhad velikosti fyzického modelu (jako max. velikost, použití indexů, tempo růstu apod.) lze vygenerovat zprávu popisující růst projektované databáze v čase [4].

Studio působí jako výkonný nástroj pro profesionální účely, který se snadno používá.

7.2 Toad Data Modeler 2.25

Toad Data Modeler firmy *Quest* existuje v plné verzi a v omezené freeware verzi [14]. Na začátku si uživatel vybere jednu z více než dvaceti databází, které program přizpůsobí výběr datových typů, integritních omezení, syntaxi SQL skriptu apod. Používaná notace i kreslení modelu je zde velmi podobné práci v ER/Studiu. Chybí zde ovšem podpora generalizační hierarchie.

Program nabízí verifikaci modelu, kam lze zařadit kontrolu unikátnosti jmen entit a integritních omezení, zjištění duplicity entit či omezení a další. Entita bez atributů způsobila při kontrole chybu, entita bez primárního klíče pak varování. Nachází-li se v diagramu self relace, způsobí při verifikaci chybu, jelikož je kvůli ní k entitě přidán cizí klíč se shodným názvem jako primární klíč.

Před generování SQL skriptu (ale i JSkriptu či VBSkriptu) se nastaví, co všechno má být definováno (tabulky, klíče, omezení, role, pohledy, skript pro smazání tabulek atd.) a kterých entit se generování týká.

Mezi další funkce patří zpětný inženýring (opět z množství různých databází), generování HTML a RTF reportů, a to s příslušným modulem i v češtině, nebo data flow diagramy (diagramy datových toků).

Program na první pohled potěší svou přehledností, není nutné se orientovat v milionech tlačítek. Je určen pro komerční využití, kterému vychází vstříc. ER editor oproti Toad Diagram Modeleru ukazuje čistě logické schéma, bez cizích klíčů a naopak s podporou generalizační hierarchie. Mnohé nabídky Modeleru, jako výběr entit před generováním skriptu či samostatnou verifikaci modelu, by bylo dobré implementovat i v ER editoru.

7.3 Ferret 0.6

Ferret (z angl. Free Entity Relationship and Reverse Engineering Tool) je nástroj dříve známý jako *GerWin*. Od tohoto jména se upustilo kvůli jeho podobnosti s názvem ERWin, která působila problémy ohledně ochranné známky. Jde o freeware pod licencí *GNU General Public License* [12].

V hlavním okně programu najdeme zvlášť záložky pro informace o projektu, tvorbu ER diagramu, zobrazení tabulek a pro převod do SQL skriptu. Při převodu na SQL skript si lze vybrat výslednou databázi, např. SQL 92 a pár dalších, nikoliv však v Evropě tak rozšířený Oracle. Reverzní inženýring podporován není.

Kreslené entity mají vzhled podobný tomu v UML notaci, tedy jako obdélník s názvem entity a seznamem atributů, zde i s uvedením datového typu. U atributů lze nastavit pouze jejich název, datový typ a zda jde o klíčový atribut. Čáry jsou vždy rovné a nenabízejí kardinalitu M:N. Lze vytvořit i self relaci. Ovšem slabou entitu, resp. identifikační vztah, vytvořit nelze.

Je třeba použít zvláštní tlačítko nejen pro vytvoření entity nebo vztahu, ale i pro takové funkce, jako je editace objektu (jinak běžná po dvojitém kliknutí na objekt), pohyb objektem nebo jeho smazání.

V generovaném SQL skriptu překvapí, že je vždy nejprve definována tabulka se sloupci příkazem CREATE TABLE a primární a cizí klíče jsou definovány později pomocí příkazů ALTER TABLE. Mezery v názvu atributu byly nahrazeny pomlčkou, která ovšem v regulárním identifikátoru podle normy ANSI SQL 92 být nesmí (přitom byla nastavena databáze „sql92“).

Vytváření ER diagramu není zcela intuitivní a chvíli trvá, než na něj uživatel přijde. Dokumentace k programu zatím neexistuje. Není tak možné zjistit, zda třeba vytvoření generalizační hierarchie Ferret neumožňuje, nebo zda uživatel prostě jen neví, jak na to.

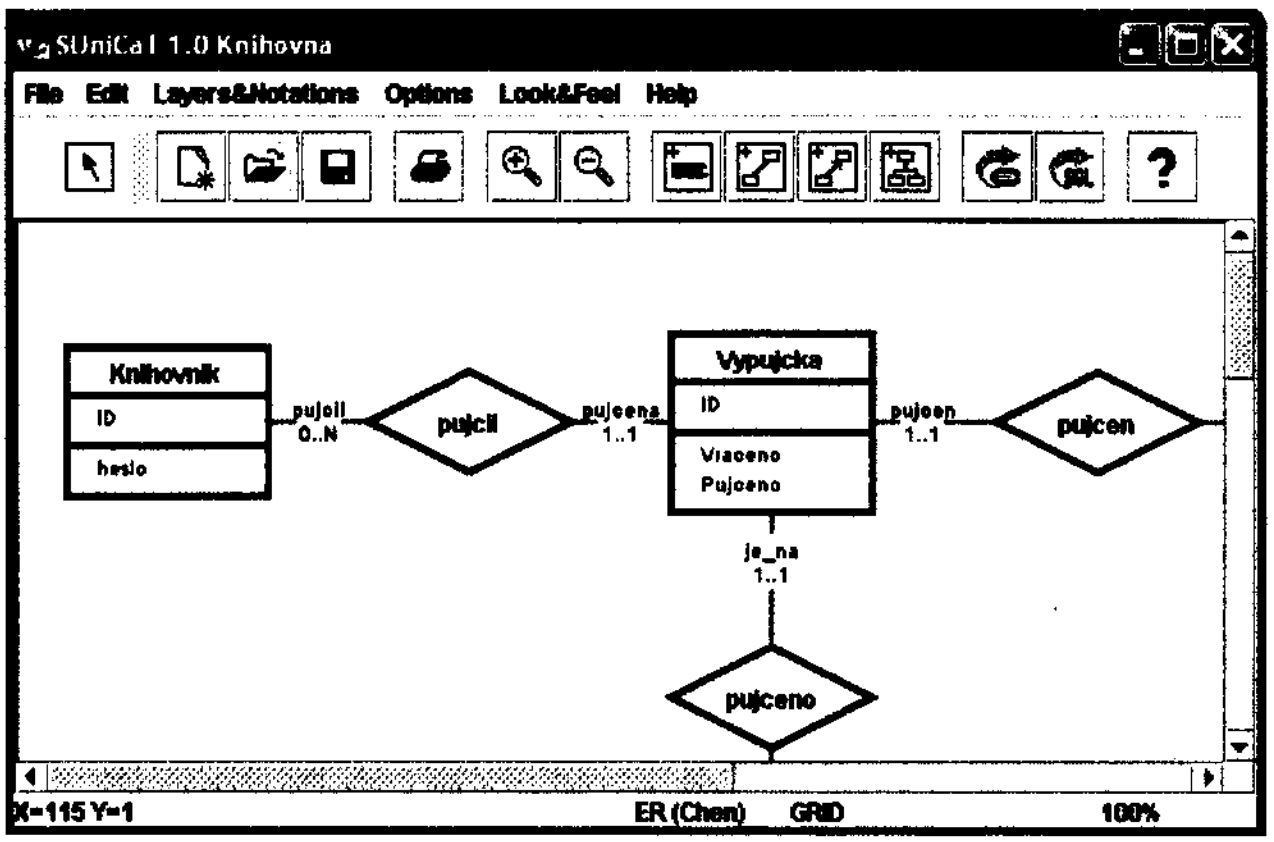
Celkově nástroj působí jednoduše a tedy i přehledně, alespoň když si na něj člověk zvykne. Nabízí ovšem velmi málo funkcí, ať se to týká vlastností vztahů a atributů nebo dalšího (viz výše). To je dáno tím, že je produkt teprve ve fázi vývoje, a první kompletnější verze bude až ta s označením 1.0.

7.4 Sunicat 1.0

Sunicat je projekt ČVUT [13], napsaný v jazyce Java. Nástroj umožňuje vytvářet ER diagramy ve třech notacích včetně Chenovy a Oraclovské notace, mezi nimiž lze libovolně přepínat.

Entity se i v Chenově notaci kreslí opět jako obdélníky se seznamem atributů, který lze skrýt. Vztahy se už ale zobrazují jako kosočtverce, které jsou s entitami propojené rovnými čarami. U čar chybí kardinalita M:N. V nabídce je i vztah identifikační (tedy určující slabou entitu) a generalizační hierarchie. Vlastnosti entit lze měnit po kliknutí pravým tlačítkem myši na entitu. Chybí možnost smazat samotný vztah - vztahy se odstraní až s odstraněním entit k němu připojených. Diagram lze uložit ve formátu XML nebo jako obrázek v jednom ze tří nabízených formátů.

Nástroj nabízí verifikaci schématu, generování SQL skriptu a HTML reportů o schématu. SQL skript lze generovat jen v jedné verzi, pro databázi Oracle. Je možné se připojit přímo k databázi nebo skript uložit do souboru.



Obrázek 7.2: Sunicat

Další užitečnou volbou je slévání tabulek. Zpětný inženýring není podporován.

Program předčí ostatní nekomerční editory ER diagramů svým vzhledem, který je navíc nastavitelný. Podporuje hlavní konstrukce ER modelu, tedy logického modelu databáze. Jde o vhodný nástroj pro výuku ER modelování.

7.5 Závěrečné srovnání

Program ER editor, popisovaný v této práci, je srovnatelný s ostatními nekomerčními editory ER diagramů. Ty se, stejně jako ER editor, zaměřují spíše na logické schéma databáze, což je pro výuku ER modelu přínosem. Oproti všem testovaným nekomerčním programům pak ER editor nabízí navíc zpětný inženýring.

Komerční programy jsou pochopitelně robustnější. Kvůli zaměření se na zákazníka, tedy na praktické využití, kladou větší důraz na fyzické schéma a nabízejí množství v praxi využitelných funkcí navíc.

Kapitola 8

Závěr

Cílem bylo vytvořit editor ER diagramů, ve kterém si uživatel může snadno vytvořit svůj ER diagram v notaci podobné původní Chenově, uložit ho, vygenerovat podle něj SQL skript pro vytvoření databáze, nebo využít zpětného inženýringu. Program nabízí uživateli možnost využívat i generalizační hierarchie a slabé entity. Nedovoluje přiřazovat atributy vztahům, což lze ovšem nahradit slabou entitou. Podobně lze vícenásobný atribut nahradit entitou, což může být i lepší řešení.

ER editor umožňuje generovat a pracovat se skripty zapsané dle normy ANSI SQL nebo s těmi, které byly vytvořené pro databázi Oracle. Zohledňuje tak rozšíření této databáze (nejen) v České republice.

Program je určen ke spíše výukovým účelům, pro diagramy nepříliš velkého rozsahu, ke kterým je ovšem plně dostačující. Je také srovnatelný s ostatními nekomerčními editory ER diagramů. Cíl práce tak lze považovat za splněný.

Naskytuje se mnoho možností, jak v budoucnu ER editor rozšířit:

- ukládání ER diagramů v XML formátu a jako obrázek (např. JPEG)
- možnost volby vzhledu prvků ER diagramu (barvy, standardní velikosti apod.)
- podpora více integritních omezení schématu
- kopírování a vkládání částí diagramu
- zobrazení seznamu entit a vztahů v aktuálním schématu
- podpora dalších notací, především UML notace

- tisk ER diagramu a SQL skriptu
- zoomování diagramu

Příloha A

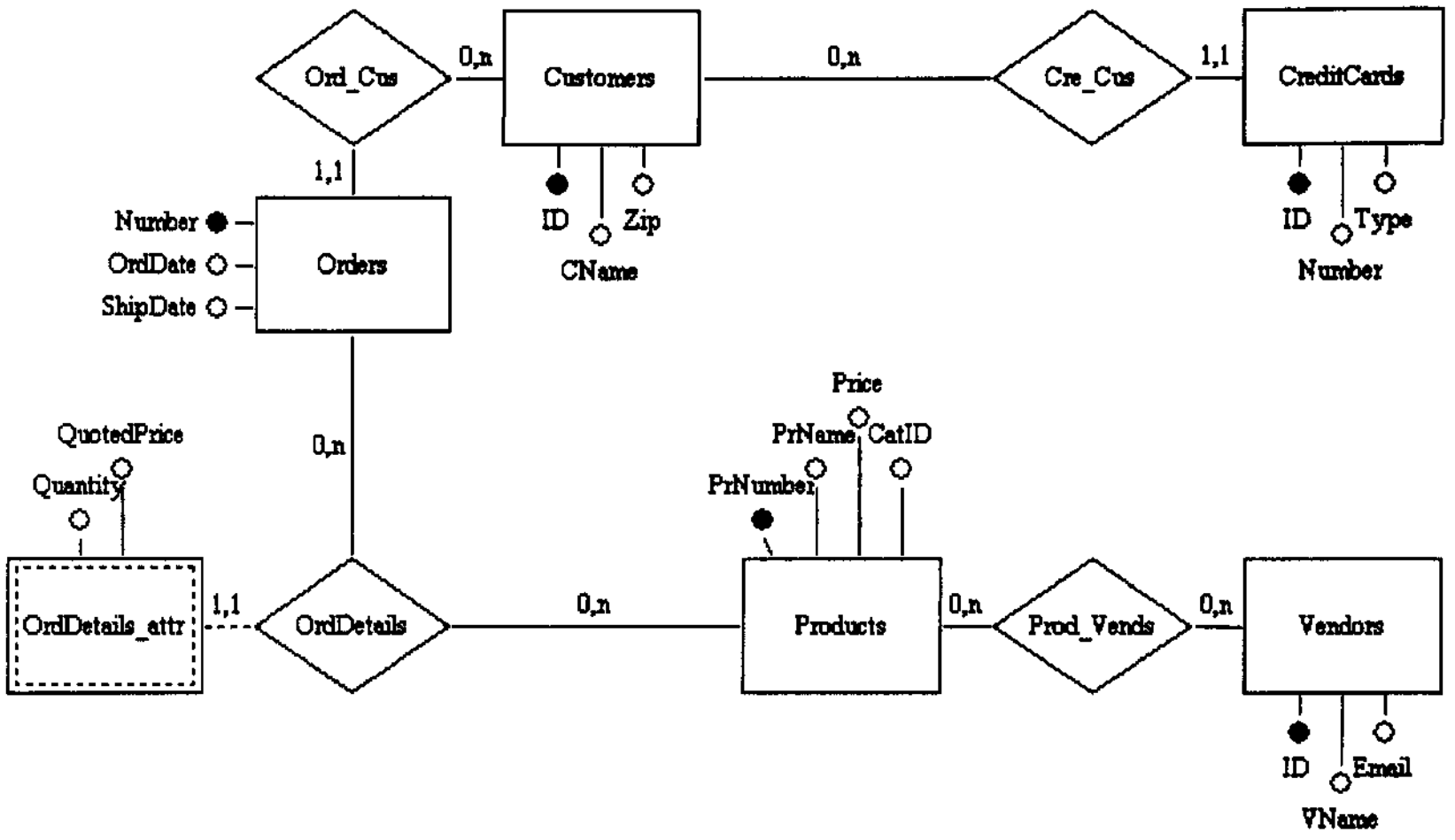
Příklad zpětného inženýringu

Následuje ukázka SQL skriptu a po něm ER diagram, který byl vytvořen zpětným inženýringem skriptu v ER editoru.

```
-- Vytvoreni schematu databaze "Objednavky" - "Sales Order"

-- Zakaznici:
-- jejich ID a adresa, kam je jim zasilano zbozi
CREATE TABLE Customers (
  ID    NUMBER(8)    PRIMARY KEY,
  Cname VARCHAR2(30) NOT NULL,
  Zip   NUMBER(5)    NOT NULL
);

-- Objednavky:
-- ID, data objednani a odeslani, kteremu zakazniku patri
CREATE TABLE Orders (
  Number   NUMBER(10) PRIMARY KEY,
  OrdDate  DATE,                    -- datum objednavky
  ShipDate DATE,                    -- datum odeslani
  CustID   NUMBER(8),               -- ktery zakaznik si ji objednal
     CONSTRAINT Ord_Cust_FK FOREIGN KEY(CustID) REFERENCES Customers(CustID)
);

-- Dodavatele:
-- ID, jmeno, kontakty
CREATE TABLE Vendors (
```

```
  ID    NUMBER(8)      PRIMARY KEY,
  Vname VARCHAR2(30),                -- jmeno dodavatele
  Email VARCHAR2(30)                 -- e-mail. adresa
);

-- Produkty:
-- ID, nazev, cena, mnozstvi a kategorie
CREATE TABLE Products (
  PrNumber  NUMBER(10)    PRIMARY KEY,
  PrName    VARCHAR2(50)  NOT NULL,
  Price     NUMBER(10,2)  NOT NULL,  -- maloobchodni cena
  CatID     NUMBER(3)                -- ID kategorie, do ktere patri
);

-- Platebni karty:
-- ID, cislo, platnost, typ
CREATE TABLE CreditCards (
  ID      NUMBER(9)   PRIMARY KEY,
  Number  NUMBER(16)  NOT NULL,    -- cislo karty
  Type    NUMBER(2)   NOT NULL,    -- druh karty
  CustID  NUMBER(8),               -- zakaznik, kteremu karta patri
     CONSTRAINT Cards_FK FOREIGN KEY(CustID)
        REFERENCES Customers(CustID) ON DELETE CASCADE
);

-- Produkty od dodavatelu
CREATE TABLE Prod_Vends (
  PrNumber  NUMBER(10),       -- cislo vyrobku
  VendorID  NUMBER(8),        -- ID dodavatele
      CONSTRAINT Prod_Vends_PK PRIMARY KEY(PrNumber,VendorID),
      CONSTRAINT PV_ProdNum_FK FOREIGN KEY(PrNumber)
         REFERENCES Products(PrNumber) ON DELETE CASCADE,
      CONSTRAINT PV_VendID_FK FOREIGN KEY(VendorID)
         REFERENCES Vendors(VendID) ON DELETE CASCADE
);

-- Detaily objednavky
CREATE TABLE OrdDetails (
```

```
  Quantity NUMBER(3),
  QuotedPrice  NUMBER(10,2)   NOT NULL,      -- stanovena cenaA
  ProdNum      NUMBER(10)     PRIMARY KEY,
  OrdNum       NUMBER(10)     PRIMARY KEY,
      CONSTRAINT OC_PN_FK FOREIGN KEY(ProdNum) REFERENCES Products(PrNumber),
      CONSTRAINT OC_ON_FK FOREIGN KEY(OrdNum) REFERENCES Orders(Number)
);
```



Příloha B

Příklad generování SQL skriptu

Následuje ukázka SQL skriptu, který byl generován ER editorem podle ER diagramu zobrazeném v předchozí příloze. Jde o diagram popisující strukturu obchodu (např. internetového).

Diagram sám byl výsledkem reverzního inženýringu skriptu, který je také v předchozí příloze vypsán. Při jejich porovnání lze objevit ztrátu informací typu „ON DELETE SET NULL“, které se v diagramu neukládají. Dále se liší některé názvy cizích klíčů a také jeden název sloupce, který byl ER editorem změněn, protože se v tabulce vyskytoval dvakrát (jako název primárního klíče a název odlišného cizího klíče). Také se přejmenovala tabulka *OrdDetails* na *OrdDetails_attr*, čímž je explicitně řečeno, že jde o tabulku nesoucí atributy vztahu.

```
-----------------------------------------------
-- gener_shop.sql

-- SQL skript generovany programem ER editor
-----------------------------------------------


CREATE TABLE Customers (
    ID NUMBER(8),
    CName VARCHAR2(30) NOT NULL,
    Zip NUMBER(5) NOT NULL,
      CONSTRAINT Customers_PK PRIMARY KEY(ID)
);
```

```
CREATE TABLE Vendors (
    ID NUMBER(8),
    VName VARCHAR2(30),
    Email VARCHAR2(30),
      CONSTRAINT Vendors_PK PRIMARY KEY(ID)
);

CREATE TABLE Products (
    PrNumber NUMBER(10),
    PrName VARCHAR2(50) NOT NULL,
    Price NUMBER(10,2) NOT NULL,
    CatID NUMBER(3),
      CONSTRAINT Products_PK PRIMARY KEY(PrNumber)
);

CREATE TABLE Orders (
    Number NUMBER(10),
    OrdDate DATE,
    ShipDate DATE,
    ID NUMBER(8),
      CONSTRAINT ID_FK FOREIGN KEY(ID) REFERENCES Customers(ID),
      CONSTRAINT Orders_PK PRIMARY KEY(Number)
);
```

```
CREATE TABLE CreditCards (
    Column_1 NUMBER(9),
    Number NUMBER(16) NOT NULL,
    Type NUMBER(2) NOT NULL,
    ID NUMBER(8),
      CONSTRAINT ID_FK FOREIGN KEY(ID) REFERENCES Customers(ID),
      CONSTRAINT CreditCards_PK PRIMARY KEY(Column_1)
);

CREATE TABLE Prod_Vends (
    PrNumber NUMBER(10),
    ID NUMBER(8),
      CONSTRAINT PrNumber_FK FOREIGN KEY(PrNumber)
            REFERENCES Products(PrNumber),
```

```
      CONSTRAINT ID_FK FOREIGN KEY(ID) REFERENCES Vendors(ID),
      CONSTRAINT Prod_Vends_PK PRIMARY KEY(PrNumber, ID)
);

CREATE TABLE OrdDetails_attr (
    Number NUMBER(10),
    PrNumber NUMBER(10),
    Quantity NUMBER(3),
    QuotedPrice NUMBER(10,2) NOT NULL,
      CONSTRAINT Number_FK FOREIGN KEY(Number) REFERENCES Orders(Number),
      CONSTRAINT PrNumber_FK FOREIGN KEY(PrNumber)
            REFERENCES Products(PrNumber),
      CONSTRAINT OrdDetails_attr_PK PRIMARY KEY(Number, PrNumber)
);
```

Literatura

[1] Hernandez M. J.: *Návrh databází*, Grada Publishing, Praha, 2006.

[2] Hernandez M. J., Viescas J. L.: *Myslíme v jazyku SQL*, Grada Publishing, Praha, 2004.

[3] McConnell S.: *Code Complete, Second Edition*, Microsoft Press, Redmond, Washington, 2004.

[4] McCown S.: *Modelování dat na mistrovské úrovni*, COMPUTERWORLD č.28, Praha, 2006

[5] Radovan: *Konceptuální datový model*, http://www.owebu.cz/databaze/vypis.php?clanek=298

[6] Riordan R. M.: *Vytváříme relační databázové aplikace*, Computer Press, Praha, 2000.

[7] Sheppard S.: *Oracle Datatypes*, http://www.ss64.com/orasyntax/datatypes.html

[8] Zendulka J.: *Konceptuální modelování a návrh databáze*, http://www.fit.vutbr.cz/study/courses/DSI/public/pdf/nove/2_2.pdf

[9] *Data Type Mapping*, http://www.lc.leidenuniv.nl/awcourse/oracle/server.920/a96544/apb.htm

[10] *Information Technology Services at The University of Texas at Austin*, http://www.utexas.edu/its/windows/database/datamodeling/index.html

[11] *ER/Studio 7.1*, http://www.embarcadero.com/downloads/downloaderstudiostd.jsp

[12] *GNU Ferret 0.6*, http://www.gnu.org/software/ferret/project/what.html

[13] *Sunicat 1.0*,
http://martin.feld.cvut.cz/ molhanec/vyuka/36dbs/files/Sunicat.zip

[14] *Quest Toad Diagram Modeler*, http://www.casestudio.com/csy/download.aspx