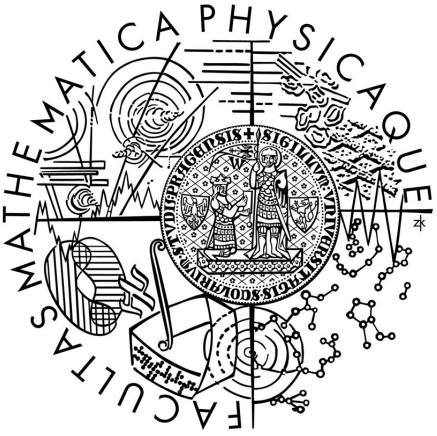
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Peter Gorbár

Editor ER diagramů s podporou transformace do relačního modelu a SQL

Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. Irena Mlýnková

Studijní program: Informatika, Správa počítačových systémů

2007

Ďakujem RNDr. Irene Mlýnkovej, vedúcej mojej bakalárskej práce, za jej pomoc, čas, cenné rady a zhovievavosť pri tvorbe práce.

Prehlasujem, že som svoju prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím so zapožičiavaním práce.

V Prahe dňa 29. 5. 2007 Peter Gorbár

Obsah

Název práce: Editor ER diagramů s podporou transformace do relačního modelu a SQL
Autor: Peter Gorbár
Katedra: Katedra softwarového inženýrství
Vedoucí bakalářské práce: RNDr. Irena Mlýnková
e-mail vedoucího: irena.mlynkova@mff.cuni.cz
Abstrakt: V predloženej práci je popísaný návrh a implementácia CASE nástroja pre konceptuálny a relačný návrh databáz. Program poskytuje dvojúrovňový editor pre tvorbu ER a RMD modelov, umožňuje prevod ER modelu do relačných schém a rovnako aj transformáciu týchto schém do SQL skriptov. V oboch úrovniach program umožňuje prevádzať kontroly vytvorených schém. Súčasťou práce je tiež analýza existujúcich programov s podobným zameraním.
Klíčová slová: ER editor, RMD editor, návrh relačných databáz, SQL

Title: ER diagram editor with the support of transformation into relational model and SQL Author: Peter Gorbár
Department: Department of Software Engineering
Supervisor: RNDr.Irena Mlýnková
Supervisor's e-mail address: irena.mlynkova@mff.cuni.cz
Abstract: In this work is describred the suggestion and implementation of CASE tool for conceptual and relational database design. The Program provides a double-level editor for creation ER and RMD models, admits transformation of ER model into relational model and simultaneously transformation of these schemes into SQL scripts. The Program admits checking of created schemes in both levels. Another part of the work is analysis of existing programs that focus on similar tasks.
Keywords: ER editor, RMD editor, relational database design, SQL

1. Úvod

So slovom databáza sa v dnešnej dobe stretol už takmer každý. Ako pojem je toto slovo pomerne ľahko zavádzajúce. Obyčajne ním rozumieme skupinu informácii usporiadaných podľa určitých pravidiel, skladište dat, v ktorom sú data uložené a spracovávané nezávislé na aplikačných programoch. Pre samotný prístup k datam uloženým v databáze sa používa špeciálný software. Anglický sa nazýva ***Data**base **M**anagment **S**ystem* (DBMS) alebo česky ***S**ystém **ř**ízení **b**áze **d**at* (SŘBD). Uživateľ pritom nemusí vôbec poznať fyzickú štruktúru uložených dat. *Databázový systém* je pojem, ktorý zastrešuje jak samotné údaje uložené v databáze, tak software pre prístup k týmto údajom.

Pre vytváranie moderných databázových aplikácii sú v praxi vyžadované niektoré z bežných technologických postupov. Medzi tieto postupy patrí tiež časť označovaná ako datové modelovanie. Na začiatku návrhu databázového modelu si musíme uvedomiť, akú časť reality budeme chcieť zobrazovať, teda čo bude obsahom datovej základne. Hovoríme o *konceptuálnom modelovaní* alebo *konceptuálnej úrovni*. K najznámejším modelom na tejto úrovni patrí *Entity Relationship model* alebo skrátene *ER model.* V ďalšom kroku sa vytvorí *relačný model dat* (skrátene *RMD model*), ktorý definuje spôsob, akým je možné reprezentovať štruktúru dat, či operácie, ktoré je možné nad datami prevádzať. Relačný model dat dôsledne oddeľuje data, ktoré sú chápané ako relácie od ich implementácie. V poslednom kroku, na *implementačnej úrovni*, sa vyberie databázová platforma, v ktorej bude navrhovaná datová základňa vytvorená. Prihľada sa tiež na špecifiká použitého vývojového prostredia. U rozsiahlejších projektov nie je prakticky možné udržiavať jednotlivé modely takpovediac ručne (kresliť ich na papier, poprípade vo Worde). Každá etapa vývoja totiž vyžaduje nástroj špecifických vlastnosti. Pretože jednotlivé etapy na seba tesne naväzujú, musia byť aj jednotlivé nástroje previazané.

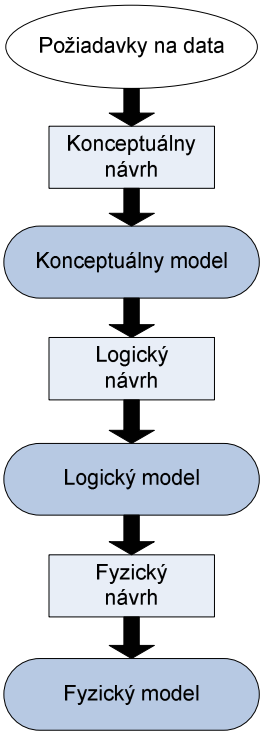
V rámci práce bol implementovaný projekt s názvom *ER[G]edit,* ktorý poskytuje komplexný nástroj pre datové modelovanie. Umožňuje vytváranie ER i RMD schémat, transformáciu ER modelu do RMD modelu a následne umožňuje generovať SQL skript pre vytvorenie potrebných tabuliek v databázovom modele. Súčasťou programu sú tiež funkcie na prevádzanie kontrol korektnosti vytváraných schém.

Následujúci text práce je rozdelený do niekoľkých hlavných kapitol. Druhá kapitola podáva detailnejší pohľad na konceptuálny a relačný model dat. Rozoberá metódy a algoritmy použité na transformáciu ER modelu do RMD a tiež sa v krátkosti zmieňuje o jazyku SQL. Tretia kapitola podáva náhľad na existujúce programy s podobným zameraním a na základe tejto analýzy sú následne vo štvrtej kapitole bližšie špecifikované požiadavky na program. Piata kapitola obsahuje podrobný popis inštalácie a ovládania programu ER[G]edit. Predstavuje grafické rozhranie aplikácie a prácu s jednotlivými modelmi. Šiesta kapitola poskytuje hlbší pohľad na jednotlivé funkčné celky aplikácie, na techniky a metódy vytvárania grafického rozhrania a implementované algoritmy prevodu a kontroly korektnosti schémat. Siedma kapitola je zhrnutím práce, kde sú subjektívne popísane jej klady, zmienené niektoré nedostatky a špecifikované ďalšie možné rozšírenia aplikácie.

2. Úvod do problematiky

2.1. Fázy návrhu relačnej databázy

Návrh relačnej databázy je zložitý proces, ktorý v sebe zahŕňa množstvo rozhodnutí na veľmi rozličných úrovniach. Pri riešení náročných úloh je lepšie rozložiť ich do niekoľkých menších problémov a tie riešiť nezávisle s použitím špecifických metód a techník. Takto môžeme rozdeliť proces návrhu relačnej databázy do niekoľkých fáz (Obr. 2.1) a to *konceptuálneho, logického a fyzického návrhu* [4].



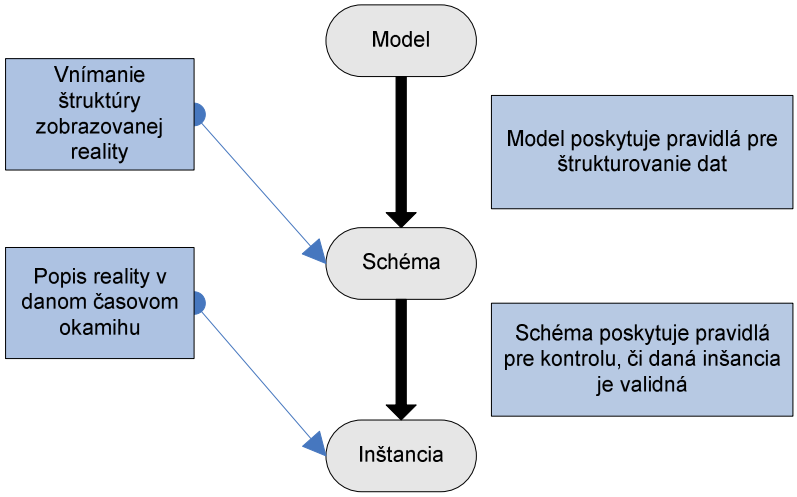
Obr. 2.1 Proces návrhu relačnej databázy

Na začiatku je potrebné špecifikovať datovú základňu a presne vymedziť požiadavky na ukladané data. Zo získaných dat sa vytvorí *konceptuálny model* – podrobnejšie je popísaný v kapitole 2.2. V druhej fáze návrhu (logický návrh) sa vhodnou transformáciou objektov konceptuálneho modelu vytvorí *logický model*. K najznámejším logickým modelom patrí relačný (podrobnejšie je popísaný v kapitole 2.3), sieťový a hierarchický model. V poslednej fáze návrhu (fyzický návrh) sa vyberá konkrétna databázová platforma, definujú sa domény jednotlivých atribútov, prípadne integritné obmedzenia, indexy, apod.

2.2. Konceptuálny model dat

Konceptuálné modely sú pokusom umožniť vytvorenie popisu dat v databáze, tj. konceptuálnej schémy, nezávisle na ich fyzickom uložení. Pri tvorbe tohto modelu je snaha čo najpresnejšie vystihnúť pohľad človeka na danú časť reálneho sveta. Konceptuálnymi modelmi sa väčšionou myslia modely používajúce pojmy blízke konceptuálnemu pohľadu, tj. entita, objekt, vzťah, atribút, vlastnosť apod. [1]. Uvedené

pojmy sú zadefinované v kapitole 2.2.1. Výsledkom tvorby konceptuálneho modelu je konceptuálna schéma alebo inak konceptuálny diagram.
Obecne, pod pojmom *schéma* budeme rozumieť reprezentáciu špecifickej časti reality vytvorenú na základe použitia odpovedajúceho datového modelu. Presnejšie, schéma je statický, časovo invariantný súbor lingvistických a grafických zobrazení, ktoré popisujú štruktúru ukladaných dat. *Inštancia* schémy je naopak dynamický, časovo premnlivý súbor dat, ktorý vyhovuje štruktúre dat definovanej touto schémou [4].



Obr. 2.2 Vzťah medzi modelom, schémou a inštanciou

Cieľom metodológie konceptuálneho návrhu je: [2]

- Upresniť pravidlá pre rozlíšenie medzi entitami, atribútmi a vzťahmi
- Upresniť kardinality vzťahov
- Definovať pravidlá pre vzťahy, ktoré môžu mať atribúty
- Definovať pravidlá pre voľbu primárnych kľúčov z množiny kandidátov na kľúč, vrátane situácie, kedy typ entity vyžaduje ďalšiu identifikáciu
- Špecifikovať detailnú procedúru pre transformáciu konceptuálného schématu do relačného modelu

2.2.1. ER konceptuálny model

ER konceptuálny model (skrátene len ER model) je množina pojmov, ktoré nám pomáhajú na konceptuálnej úrovni abstrakcie popísať uživateľskú aplikáciu za účelom následnej špecifikácie štruktúry databázy [1]. ER model patrí k najrozšírenejšie používaným datovým modelom pre konceptuálny návrh databáz a v súčasnosti je medzi nimi de facto štandardom. Prvykrát bol predstavený Petrom Chenom v roku 1976 a už zanedlho, v roku 1988, ANSI zvolila ER model za štandard pre IRDS systémy (*Information Resource Dictionary Systems*). Pôvodne, ER model obsahoval koncepty iba pre entitu, vzťah a atribúty. Neskôr boli pridané aj ďalšie, ako zložený atribút či generalizčné (ISA) hierarchie.

Základne pojmy spojené s modelom ER:

- **Entitný typ –** množina objektov rovnakého typu zabrazujúca nezávislý objekt reálneho sveta. V ER diagrame sa graficky zobrazuje pomocou obdlžníka.

- **Vzťahový typ –** reprezentuje vzájomne spojenie dvoch a viacerých entitných typov. V ER diagrame sa graficky zobrazuje pomocou kosoštvorca.
- **Entita –** inštancia entitného typu. Každá entita musí byť jednoznačne identifikovateľná.
- **Vzťah –** inštancia vzťahového typu.
- **Atribút –** funkcia, ktorá entitám a vzťahom priraďuje hodnotu určujúcu niektorú ich podstatnú vlastnosť. Pre atribút sa okrem názvu tiež definuje doména, tj. množina hodnôt, ktoré môže atribút nabývať, ďalej informácia či daný atribút je súčasťou identifikačného kĺúča, prípadne obmedzenie, či atribút môže nabývať *null* hodnôt.
- **Identifikačný kľúč –** atribút (prípadne skupina atribútov), ktorého hodnota slúži k identifikácii konkrétnej entity.

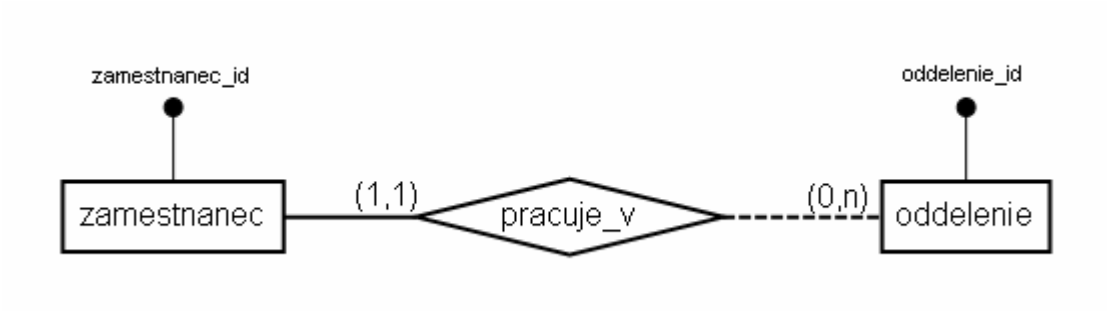
V ďalšom texte, pokiaľ je zrejmý význam z kontextu, je použitý termín entita aj pre entitný typ a termín vzťah pre vzťahový typ.

Súčasťou identifikačného kľúča niektorých entít nemusia byť len ich vlastné atribúty. V prípade, že to tak nie je a k identifikácii entity sú použité tiež identifikačné atribúty inej entity (**identifikačný vlastník**), hovoríme o **slabom entitnom type.** Vzťah, ktorý spája slabý entitný typ s jeho identifikačným vlastníkom nazývame **identifikačný vzťah.** U slabého entitného typu jeho vlastné atribúty tvoria tzv. čiastočný kľúč. Entitný typ, ktorý je identifikovaný výhradne svojimi vlastnými atribútmi, nazývame **silný entitný typ.**

Dôležitým integritným obmedzením (IO), ktoré približuje konceptuálnu schému realite je **kardinalita vzťahu.** Môže nadobúdať tieto hodnoty:

- *Vzťah 1:1* – znamená, že každej jednej entite odpovedá nanajvýš jedna druhá entita. Obecne zahŕňa tiež prípady *1:0* a *0:1*.
- *Vzťah 1:N* – znamená, že prvej entite môže odpovedať viac než jedna druhá entita a druhej entite nanajvýš jedna prvá entita. Tento vzťahový typ obecne zahŕňa aj prípady vzťahov *1:0, 0:1* a *1:1*. Kardinalitu vzťahu inokedy vyjadrujeme tvrdením, že entita jedného typu jednozačne určuje (neurčuje) entitu druhého typu, prípadne že entita jedného typu je (nie je) **determinantom** entity druhého typu. V prípade kardinality 1:N môžeme použitím tejto terminológie tvrdiť, že druhá entita je determinantom prvej entity.
- *Vzťah M:N* – znamená, že prvej entite môže odpovedať viac než jedna druhá entita a druhej entite môže odpovedať viac než jedna prvá entita. Tento vzťahový typ obecne zahŕňa aj všetky predchádzajúce spomínane kardinality vzťahu.

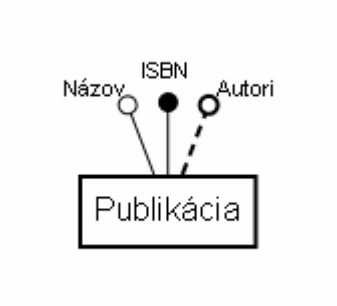
O entitách, ktoré sú zapojené do vzťahu hovoríme, že sú členmi vzťahu. V tejto súvislosti rozlišujeme povinné a nepovinné členstvo vo vzťahu a definujeme pojem **existenčnej závislosti.** Entita, ktorá má povinné členstvo vo vzťahu je existenčne závisla na druhej enite (Obr. 2.3).



Obr. 2.3 Ukážka existenčne závislého entitného typu

Počet výskytov entity vo vzťahu sa obyčajne udáva dvojicou hodnôt (min, max), ktorá určuje minimálny a maximálny počet výskytov druhej entity. Toto IO sa niekedy označuje ako *min-max IO* [1]. Povinné členstvo vo vzťahu je takto vyjadrené hodnotou (1,*) a nepovinné hodnotou (0, *).

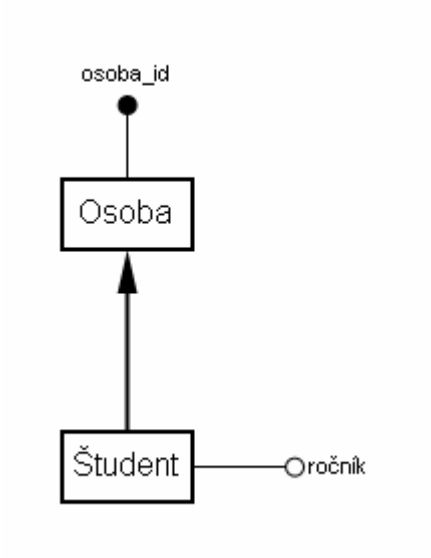
Konceptuálny model nemusí byť obmedzený iba na použitie atomických atribútov. Príkladom takéhoto neatomického atribútu je **viachodnotový** atribút, ktorý môže obsahovať viac hodnôt rovnakého typu. Ako príklad uvažujme typ entity *Publikácia,* ktorej jednoznačným identifikátorom nech je ISBN číslo. Povedzme, že si u každej entity chceme udržiavať okrem názvu tiež mená všetkých autorov. V ER konceptuálnom modeli práve pre tento účel je vhodné použiť viachodnotový atribút *Autori* (Obr. 2.4).



Obr. 2.4 Príklad použitia viachodnotového atribútu

Ďalším príkladom neatomického atribútu, používaného u ER modelov, je **skupinový** atribút. Typickým kandidátom je atribút *Adresa,* ktorý môže byť ďalej rozčlenený na názov štatu, mesta, ulice, psč, apod. Štruktúra skupinových atribútov nemusí byť jednoúrovňová, ale obecne môžu vytvárať hierarchickú štruktúru, ako ju poznáme z datového typu *record.*

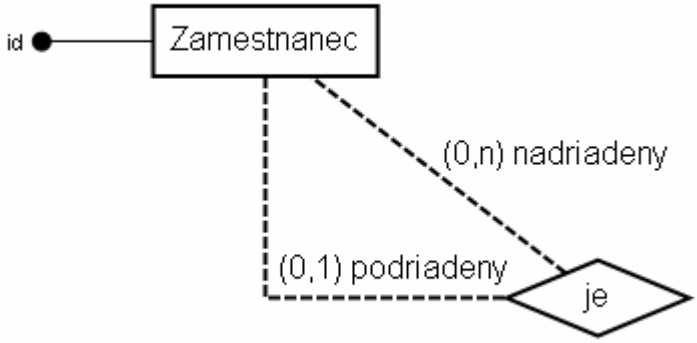
ER model tiež umožňuje vytváranie generalizačných hierarchií označovaných ako **ISA hierarchie.** Pojem ISA hierarchie je odvodený z anglického “is a”, kde napríklad entita *Študent* je (anglicky IS A) *Osoba* (Obr. 2.5).



Obr. 2.5 Príklad ISA vzťahu

Pre daný entitný typ môžeme takto zavádzať jeho podtypy (podentity). Podentity sú identifikované výhradne svojim predkom a dedia jak atribúty, tak vzťahy nadentity. U ISA hierarchie je dovolená len jednonásobná dedičnosť.

ER konceptuálny model dovoľuje vytvárať tiež **rekurzívne** vzťahy (Obr. 2.6). Pri rekurzívnych vzťahoch je nutné kvôli odlišeniu zapojených entít dôsledne používať role, v ktorých dané entity vo vzťahu vystupujú. V ďalšom texte je použitý pojem self-relácia ekvivalentne k binárnemu rekurzívnemu vzťahu.



Obr. 2.6 Príklad rekurzívneho binárneho vzťahu

2.2.2. Korektnosť schémy v ER modele

Korektnosť konceptuálnej schémy znamená, že je zmysluplne a jednoznačne definovaná jej sémantika. Problémy so sémantikou sa začnú objavovať pri uvažovaní schém obsahujúcich ISA hierarchie a identifikačné vzťahy [1]. V ďalšom texte pojem **zdroj ISA hierarchie** označuje entitu v ISA hierarchii, z ktorej už nevychádza žiadna ISA hrana. Povieme, že entita má zdroj ISA hierarchie, ak z tejto entity vedie danou ISA hierarchiou cesta k jej zdroju. Cestou sa rozumie aj cesta nulovej dĺžky, čo znamená, že entitný typ, ktorý nie je zapojený v žiadnej ISA hierarchii (nemá žiadne podentity), je rovnako zdrojom ISA hierarchie.

Požiadavky na korektnú ER konceptuálnu schému špecifikuje definícia **dobre definovanej schémy** [1]**:**

- Žiadna entita nemá v schéme viac než jeden zdroj ISA hierarchie.
- ISA vzťahy netvoria v ER diagrame orientovaný cyklus.
- Identifikačné typy vzťahov netvoria v ER diagrame orientovaný cyklus.
- Entita v ISA hierarchii, ktorá nie je zdrojom, nie je identifikačne závislá na žiadnom type entity.
- Mená typov entít a vzťahov sú jednoznačné globálne mená, zatiaľčo mená atribútov (vrátane zdedených) sú jednoznačné mená vrámci daného typu objektu.
- Ak entita vystupuje vo vzťahu viac než jedenkrát, je charakterizovaná rôznymi rolami.
- Ak je entita zdroj ISA hierarchie (zahŕňa tiež prípady, kedy entita nie je zapojená v žiadnej ISA hierarchii), má identifikačný kľúč. Ostatné entity nemajú identifikačný kľúč.
- Ak je identifikovaný slabý entitný typ pre dva rôzne identifikačné vzťahy ten istý, tak buď pre túto entitu existujú dva rôzne identifikačné zdroje alebo ten istý identifikačný zdroj vystupuje v dvoch rolách a tie je nutné odlíšiť ich označením.

Vizuálna stránka vytváraného ER diagramu nie je obmedzená žiadnými presnými pravidlami. Avšak kvôli zvýšeniu prehľadnosti a čitateľnosti diagramu sa doporučuje pri kreslení ER diagramu dodržiavať nasledujúce pravidlá [2], [5]:

- názvy entít a vzťahov voliť zrozumiteľne, aby čo najviac vyjadrovali svoj skutočný význam
- pre názvy vzťahov používať slovesá, predložky
- pre názvy entít používať podstatné mena
- preferovať horizontálne typy vzťahov
- vyvarovať sa kríženiu čiar a minimalizovať používanie šikmých čiar
- u vzťahov s kardinalitou 1:N preferovať zakreslenie entity s častejším výskytom naľavo, s menej častým výskytom vpravo
- paralelné čiary kresliť ďalej od seba
- jednotlivé ER diagramy identifikovať menom, dátumom a autorom

2.3. Relačný model dat

Relačný model dat (RMD) bol predstavený v roku 1970 a navrhnutý Dr. E.F. Coddom, vtedajším pracovníkom firmy IBM. Pojem **relácie** v relačnom modele vychádza z matematického pojmu relácie. Ide o množinu prvkov, ktoré majú tvar $(a_1, a_2,...,a_n)$, kde *n* je rád relácie. V relačnej terminológii sa prvkom tiež hovorí *n-tice*. Jednotlivé komponenty a_i sú z domény atribútu A_i, kde A_i predstavuje reťazec znakov – **meno atribútu.** O a_i sa obvykle hovorí ako o hodnote atribútu A_i v danej n-tici. **Doména** je špecifikovaná množina hodnôt, ktorých môže atribút nabývať. Teda **atribút** A_i je určený dvojicou A_i:dom(A_i). Pomocou zobrazenia *dom* priraďujeme k atribútu jeho doménu. *Dom* je teda zobrazenie definované na množine mien atribútov [1].

Reláciu pomenovanú R je teda možne popísať ako $R(A_1:D_1, A_2:D_2,..., A_n:D_n)$, kde $D_i = dom(A_i)$, pre i = 1,...,n. Tento zápis tvorí **schému relácie.** Matematicky ide o podmnožinu kartézského súčinu $D_1 \times D_2 \times ... \times D_n$. Relácia teda môže byť aj prázdna. Ak chceme hovoriť o množine atribútov danej relácie, je vhodné tuto množinu taktiež označiť menom, napríklad $A = \{A_1:D_1, A_2:D_2,..., A_n:D_n\}$. Potom môžeme schému relácie zapisovať ako R(A) [1].

Aktuálna doména nejakého atribútu A, značí sa *adom(A),* obsahuje hodnoty, ktoré sa pre atribút A v danej chvíli vyskytujú v databázi [1].

Veľmi dôležitým obmedzením relačného modelu dat je fakt, že hodnoty atribútov sú atomické, teda nedeliteľné. Toto obmedzenie sa nazýva **1. normálna forma relácie** (1NF). Ide o celkom základny predpoklad celého RMD [1].

Vedľa tejto formálnej terminológie, ktorú je možné takto popísanú nájsť v [1], existuje tiež tzv. *tabuľková terminológia*, v ktorej schéma relácie odpovedá **záhlaviu tabuľky,** n-tica relácie odpovedá **riadkom tabuľky** a atribúty odpovedajú **stĺpcom tabuľky.** Je však potrebné poznamenať dva dôležité rozdiely. Zatiaľčo v relácii nezáleži na poradí riadkov, v tabuľke je vždy určené nejaké poradie riadkov a stĺpcov. A druhým rozdielom je, že tabuľka môže obsahovať duplicitné riadky, kým n-tice relácie duplicity nevytvárajú. Ide totiž o množiny. V ďalšom texte pri používaní tabuľkovej terminológie používame pojmy stĺpec tabuľky a atribút tabuľky v rovnakom význame.

2.3.1. Integritné obmedzenia

Schéma relácie popisuje datovú štrukturú a teda z hľadiska použitia relácii ako databáz v uživateľskej aplikácii je potrebné zaistiť, aby sa do relácii dostali len korektné data. Za týmto účelom sa špecifikujú integritné obmedzenia relačného modelu. Tie môžeme chápať ako logické podmienky, ktoré majú byť na datach v databáze splnené. Inštancie relácie, ktoré vyhovujú integritným obmedzeniam sa označujú ako **prípustné.**

K najdôležitejším integritným obmedzeniam patrí špecifikácia kľúča schémy relácie R a referenčná integrita (viď nižšie). **Kľúč** K schémy R(A) je minimálna množina atribútov z A, ktorých hodnoty budú jednoznačne určovať n-tice inštancie relácie R. Obecne, kľúčov môže byť viac, preto sa vyberá jeden, tzv. **primárny kľúč.** Ak sa ku kľúču pripojí ľubovoľný atribút, zostáva zachovaná identifikačná vlastnosť, neplatí však, že množina atribútov je minimálna. Výsledkom je potom **nadkľúč** schémy R(A). Kľúče, ktoré sa skladajú len z jedného atribútu, sa nazývajú jednoduché, ostatné kľúče sú **zložené.** Atribút, ktorý je súčasťou nejakého kľúča, sa nazýva **kľúčový**, ostatné sú **nekľúčové.** Kľúč, ktorý nie je primárny sa niekedy nazýva **sekundárny** alebo **alternatívny** [1]**.**

Ďalším, už spomínaným, dôležitým integritným obmedzením, ktoré je dnes podporované v definičnom jazyku mnohých SŘBD, je **referenčná integrita.** Toto IO popisuje vzťah medzi datami obsiahnutými v dvoch reláciach. Atribút, ktorého sa referenčná integrita týka, sa často nazýva **cudzí kľúč.** Ide o atribút (prípadne skupinu atribútov), ktorého hodnota v každej n-tici relácie je buď prázdna alebo musí byť obsiahnutá ako hodnota primárneho kľúča v inej relácii [1].

2.4. Jazyk SQL

Počiatky jazyka SQL (**S**tructured **Q**uery **L**anguage) siahajú do roku 1974, kedy sa ešte nazýval Sequel (**S**tructured **E**nglish **Que**ry **L**anguage) a bol zameraný hlavne na svoju dotazovaciu časť. Jazyk SQL bol prvykrát štandardizovaný v roku 1986 a v súčastnosti je všeobecne známe minimálne jeho trojaké použitie [1]:

- SQL ako dotazovací (manipulačný) jazyk pre relačné databázy
- SQL ako zložka hostiteľského jazyka pre programovanie databázových aplikácii
- SQL ako jazyk komunikácie medzi rôznymi zdrojmi dat

Z vyššie uvedených bodov je zrejmé, že SQL je viac než len dotazovací jazyk. Je možné v ňom definovať data a prevádzať aktualizácie tak, ako je obvyklé u každého SŘBD. Je možné tiež definovať prístupové práva k tabuľkám. Základne rysy databázového modelovania sveta v SQL sú [1]:

- Data sú uložené v databáze vo forme tabuliek, ktoré sú buď skutočné (odpovedajú schéme databázy) alebo virtuálne (tzv. pohľady).
- SQL vracia data programu (alebo interaktivne uživateľovi), ktorý sa nemusí starať o fyzickú štruktúru či umiestnenie dat.
- Poloha tabuliek v databáze ani poradie stĺpcov v tabuľkách nie sú dôležité, sú totižto identifikované menom.
- Rovnako nie je dôležité ani poradie riadkov v tabuľkách, sú identifikované hodnotami v stĺpcoch.
- Data sú uživateľovi vždy prezentované ako tabuľky, bez ohľadu na ich vnútornu štruktúru použitú v databáze.

Jazyk SQL tiež obsahuje príkazy pre zavedenie integritných obmedzení tabuliek i stĺpcov tabuliek, vrátane referenčnej integrity.

2.5. Transformácia ER schémy do relačného modelu

Transformácia ER schémy do relačného modelu je snahou o implementáciu ER konceptu v relačnej databáze. Aplikáciou tejto transformácie vznikne zoznam tabuliek, ktoré obsahujú (v ideálnom prípade) všetky integritné obmedzenia špecifikované v ER schéme.

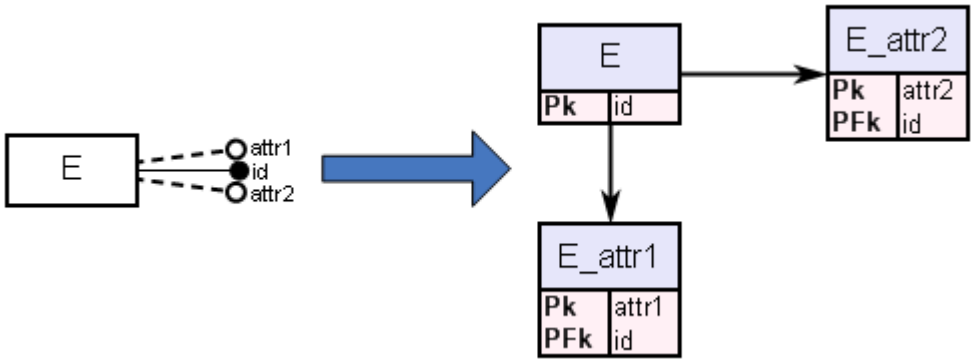
V aplikácii ER[G]edit bol pre túto tranformáciu použitý algoritmus, ktorý uvádzajú skripta [1]. V ďalšom texte sa popisujú mechanizmy tranformácie jednotlivých konštruktov ER modelu bez ohľadu na kontext ich použitia v ER diagrame. Aby relačná schéma čo najpresnejšie odpovedala konceptuálnej ER schéme, z ktorej vychádza, je potrebné pridať ďalšie integritné obmedzenie, a to referenčnú integritu (kapitola 2.3.1).

2.5.1. Reprezentácia silného entitného typu

Reprezentácia silného entitného typu nie je v podstate problém, entita sa prevedie na tabuľku s rovnakým názvom a atribútmi, okrem viachodnotových. Tie sú špeciálnym prípadom a ich reprezentácia je popísaná v kapitole 2.5.2. Primárny kľúč bude tvorený atribútmi odpovedajúcimi atribútom identifikačného kľúča entity.

2.5.2. Reprezentácia viachodnotových atribútov

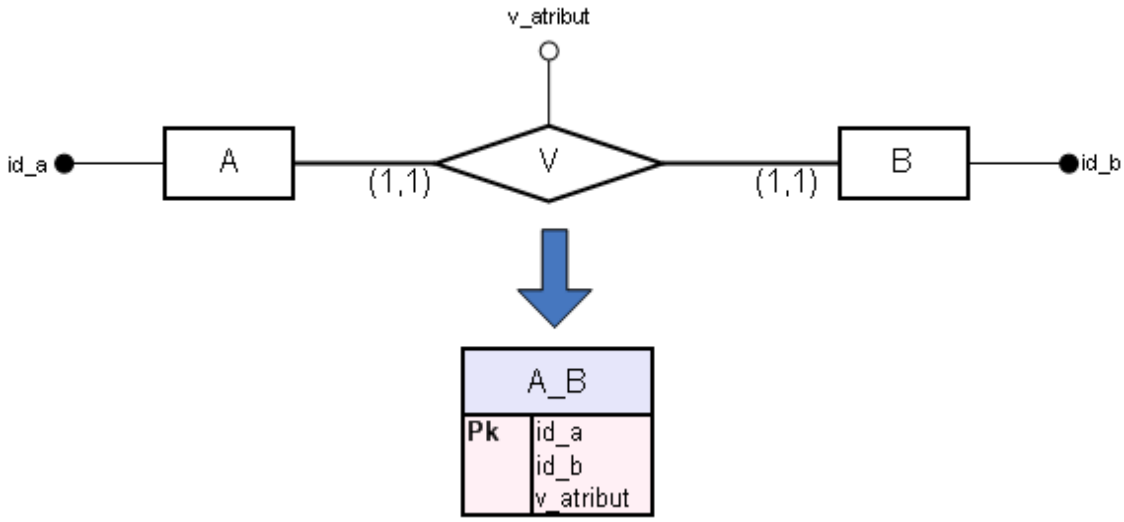
Pre každý viachodnotový atribút sa vytvorí nová tabuľka, ktorá bude obsahovať daný viachodnotový atribút a zároveň všetky kľúčové atribúty entity. V prípade, že kardinalita medzi entitou a daným viachodnotovým atribútom bola 1:N, tento atribút bude primárnym kľúčom novej tabuľky. V prípade, že kardinalita bola M:N, primárnym kľúčom budú všetky atribúty novej tabuľky. Na obrázku 2.7 je ukážka prevodu entity *E* s identifikátorom *id* a dvoma viachodnotovými atribútmi *attr1, attr2* do tabuliek.



Obr. 2.7 Ukážka prevodu viachodnotových atribútov

2.5.3. Reprezentácia binárneho vzťahu s kardinalitou (1,1):(1,1)

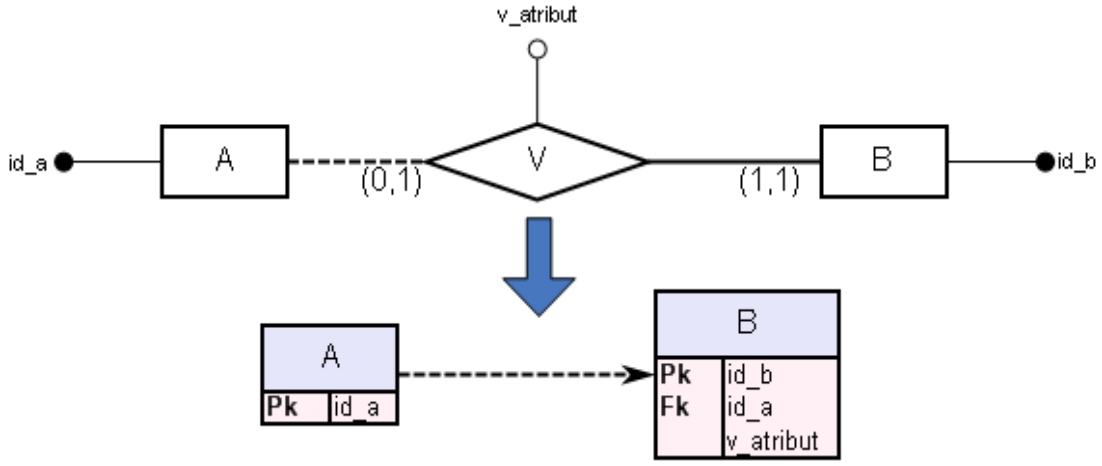
V tomto prípade vystačíme s jednou tabuľkou pre oba zúčastnené entity. Tabuľka bude obsahovať atribúty vzťahu a oboch entít. Primárnym kľúčom tejto tabuľky môžu byť atribúty tvoriace identifikačný kľúč jednej alebo druhej entity (Obr. 2.8).



Obr. 2.8 Ukážka prevodu binárneho vzťahu s kardinalitou (1,1):(1,1)

2.5.4. Reprezentácia binárneho vzťahu s kardinalitou (0,1):(1,1)

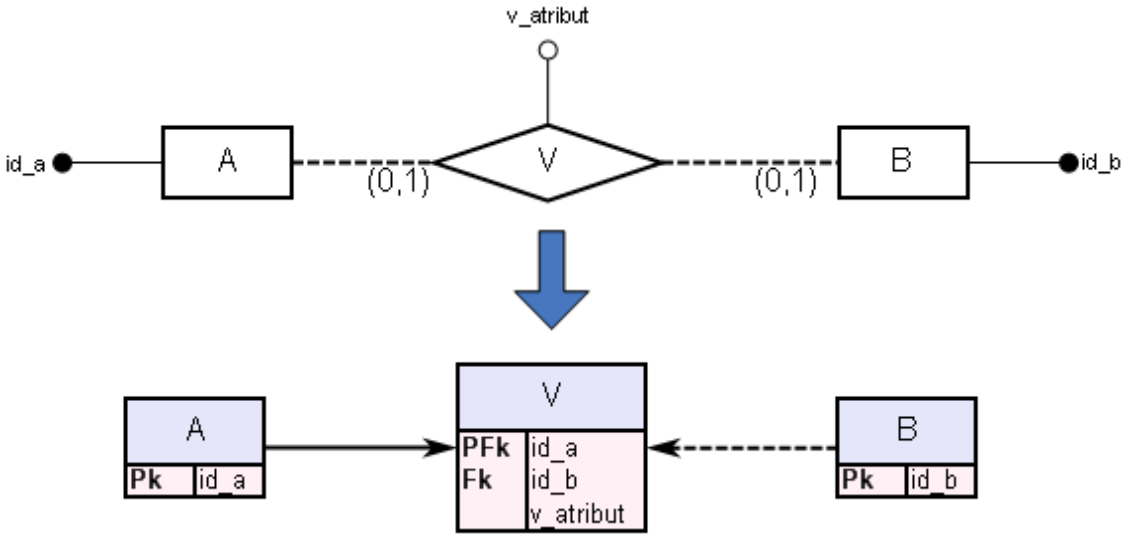
Pre prevod budeme potrebovať dve tabuľky, pre každý zúčastnený entitný typ jednu. Prvá tabuľka bude obsahovať všetky atribúty prvej entity a jej primárnym kľúčom budú atribúty tvoriace identifikačný kľúč tejto entity. Analogicky to platí aj pre druhú tabuľku. Keďže druhá entita je existenčne závislá na prvej entite, bude druhá tabuľka obsahovať tiež atribúty odpovedajúce identifikačnému kľúču prvej entity a tieto atribúty budú tvoriť cudzí kľúč definujúci referenčnú integritu s prvou tabuľkou. K druhej tabuľke sa pridajú tiež atribúty vzťahu (Obr. 2.9).



Obr. 2.9 Ukážka prevodu binárneho vzťahu s kardinalitou (0,1):(1,1)

2.5.5. Reprezentácia binárneho vzťahu s kardinalitou (0,1):(0,1)

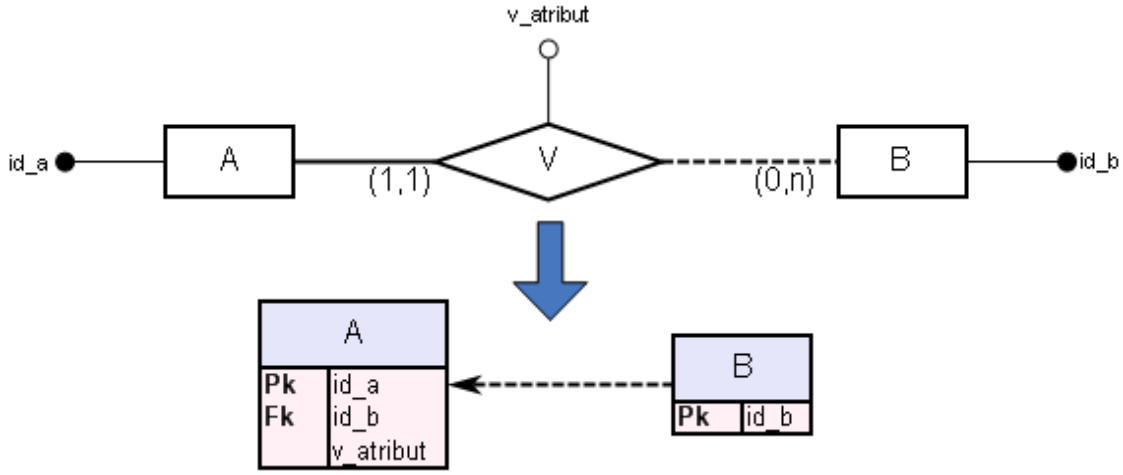
Keďže obe entity majú vo vzťahu nepovinnú účasť, pre prevod vytvoríme celkom tri tabuľky. Dve pre zúčastnené entity a jednu pre vzťah. Primárnymi kľúčmi tabuliek vytvorených z entít budú atribúty týchto entít, ktoré odpovedajú identifikačným kľúčom. Tabuľka odpovedajúca vzťahovému typu, bude obsahovať atribúty vzťahu a kľúčové atribúty oboch entít. Tieto kľúčové atribúty budú tvoriť cudzie kľúče definujúce referenčnú integritu s tabuľkami vytvorenými z entitných typov. Primárnym kľúčom vzťahovej tabuľky môžu byť buď kľúčové atribúty jednej alebo druhej entity (Obr. 2.10).



Obr. 2.10 Ukážka prevodu binárneho vzťahu s kardinalitou (0,1):(0,1)

2.5.6. Reprezentácia binárneho vzťahu s kardinalitou (1,1):(0,N)

Pre prevod budeme potrebovať dve tabuľky, pre každý entitný typ jednu. Tabuľky budú obsahovať atribúty odpovedajúcich entít a primárnymi kľúčmi tabuliek budú kľúčové atribúty týchto entít. Tabuľka vytvorená z entity s kardinalitou (1:1) bude navyše obsahovať atribúty vzťahu a kľúčové atribúty druhej entity, ktoré vytvoria cudzí kľúč definujúci referenčnú integritu medzi tabuľkami (Obr. 2.11).



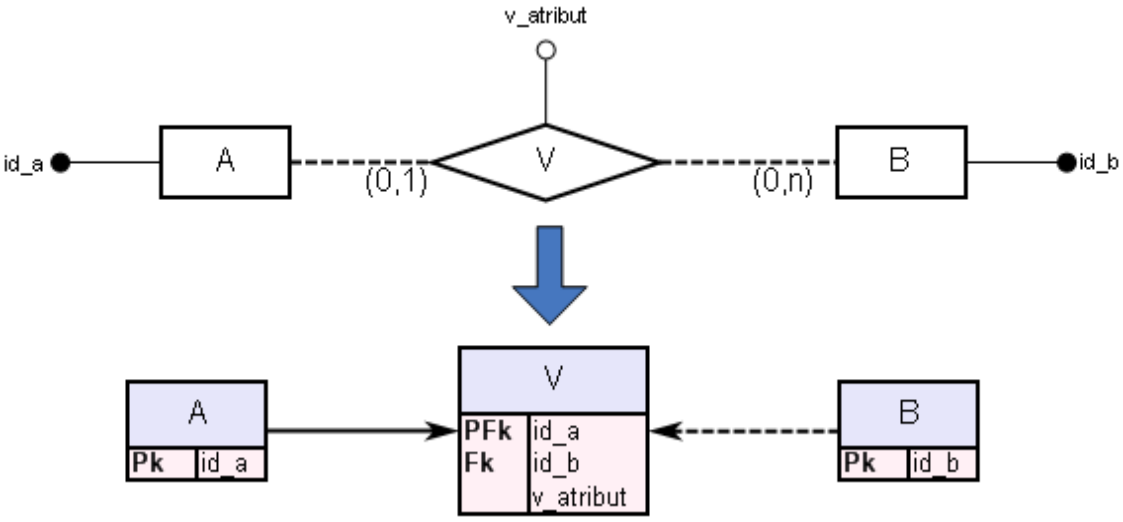
Obr. 2.11 Ukážka prevodu binárneho vzťahu s kardinalitou (1,1):(0,N)

2.5.7. Reprezentácia binárneho vzťahu s kardinalitou (1,1):(1,N)

Tento binárny sa prevádza rovnakým spôsobom ako vzťah s kardinalitou (1,1):(0,N). Viď kapitola 2.5.6.

2.5.8. Reprezentácia binárneho vzťahu s kardinalitou (0,1):(0,N)

Vytvoríme tri tabuľky, dve pre zúčastnené entity a jednu pre vzťah. Primárnymi kľúčmi tabuliek vytvorených z entít budú atribúty týchto entít, ktoré odpovedajú identifikačným kľúčom. Tabuľka odpovedajúca vzťahovému typu, bude obsahovať atribúty vzťahu a kľúčové atribúty oboch entít, ktoré budú tvoriť cudzie kľúče definujúce referenčnú integritu s tabuľkami vytvorenými z entitných typov. Primárnym kľúčom vzťahovej tabuľky budú kľúčové atribúty entity s kardinalitou (0:1) (Obr. 2.12).



Obr. 2.12 Ukážka prevodu binárneho vzťahu s kardinalitou (0,1):(0,N)

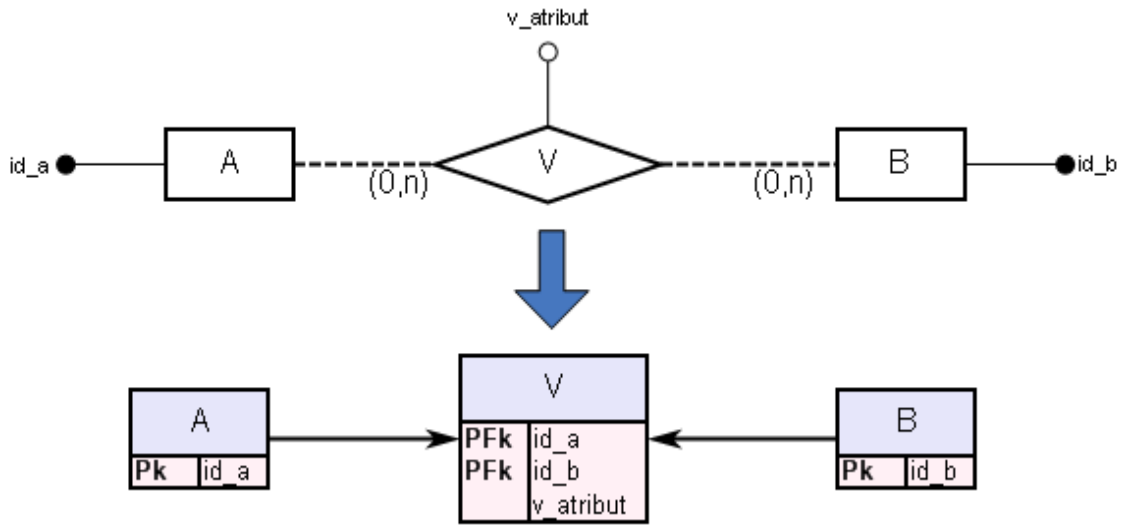
2.5.9. Reprezentácia binárneho vzťahu s kardinalitou (0,1):(1,N)

Tento vzťah sa reprezentuje rovanko ako vzťah s kardinalitou (0,1):(0,N). Viď kapitola 2.5.8.

2.5.10. Reprezentácia binárneho vzťahu s kardinalitou (0,N):(0,N)

Vytvoria sa tri tabuľky, dve pre zúčastnené entitné typy a jedna pre vzťahový typ. Primárnymi kľúčmi tabuliek vytvorených z entít budú atribúty týchto entít, ktoré

odpovedajú identifikačným kľúčom. Tabuľka odpovedajúca vzťahovému typu, bude obsahovať atribúty vzťahu a kľúčové atribúty oboch entít, ktoré budú tvoriť cudzie kľúče definujúce referenčnú integritu s tabuľkami vytvorenými z entitných typov. Primárnym kľúčom vzťahovej tabuľky budú všetky kľúčové atribúty z oboch entít (Obr. 2.13).



Obr. 2.13 Ukážka prevodu binárneho vzťahu s kardinalitou (0,N):(0,N)

2.5.11. Reprezentácia binárneho vzťahu s kardinalitou (0,N):(1,N)

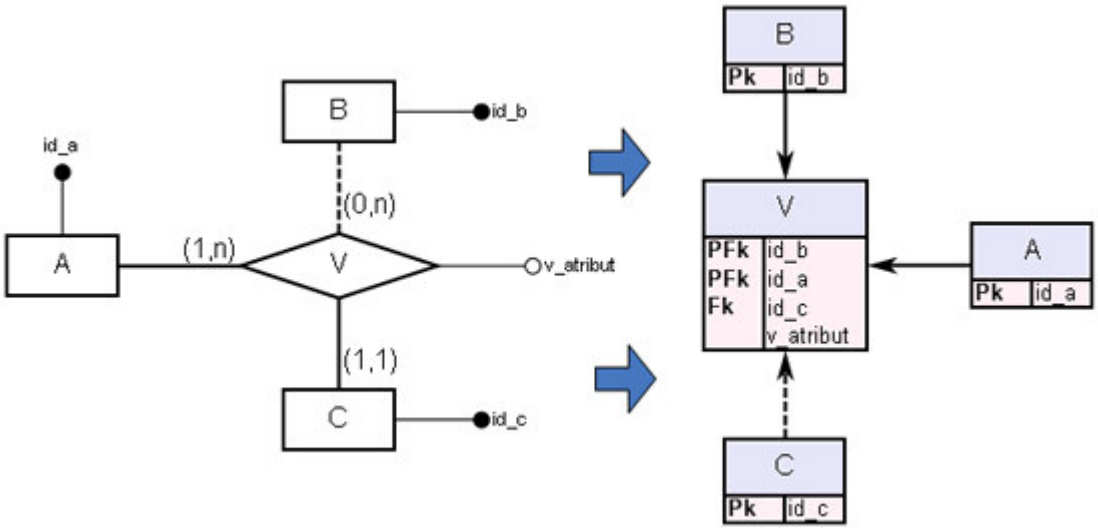
Tento vzťah sa reprezentuje rovnako ako vzťah s kardinalitou (0,N):(0,N). Viď kapitola 2.5.10.

2.5.12. Reprezentácia binárneho vzťahu s kardinalitou (1,N):(1,N)

Tento vzťah sa reprezentuje rovnako ako vzťah s kardinalitou (0,N):(0,N). Viď kapitola 2.5.10.

2.5.13. Reprezentácia n-árneho vzťahu

Pri prevode n-árneho vzťahu, bez ohľadu na typ členstva zúčastnených entít, sa vytvorí *n* tabuliek pre zúčastnené entitné typy a jedna tabuľka pre vzťahový typ. Tabuľky vytvorené z entít budú obsahovať atribúty odpovedajúcich entít a primárne kľúče jednotlivých tabuliek budú tvorené atribútmi, ktoré v danej entite tvorili identifikačný kľúč. Tabuľka odpovedajúca vzťahovému typu, bude obsahovať všetky jeho atribúty a kľúčové atribúty zúčastnených entít, ktoré budú tvoriť cudzie kľúče definujúce referenčnú integritu medzi tabuľkami. Primárnym kľúčom tejto tabuľky budú identifikátory entít, ktoré v kardinalite vzťahu mali priradenú hodnotu (0,n) alebo (1,n) (Obr. 2.14).



Obr. 2.14 Ukážka prevodu ternárneho vzťahu

2.5.14. Reprezentácia slabého entitného typu

Slabý entitný typ sa vyskytuje jedine v identifikačnej závislosti na vzťahu. Z identifikačnej závislosti vyplýva existenčná závislosť a teda reprezentácia slabého entitného typu je vyriešená v rámci reprezentácie vzťahu s kardinalitou (1,1):(0,N) (kapitola 2.5.6) alebo v rámci reprezentácie vzťahu s kardinalitou (1,1):(1,N) (kapitola 2.5.7).

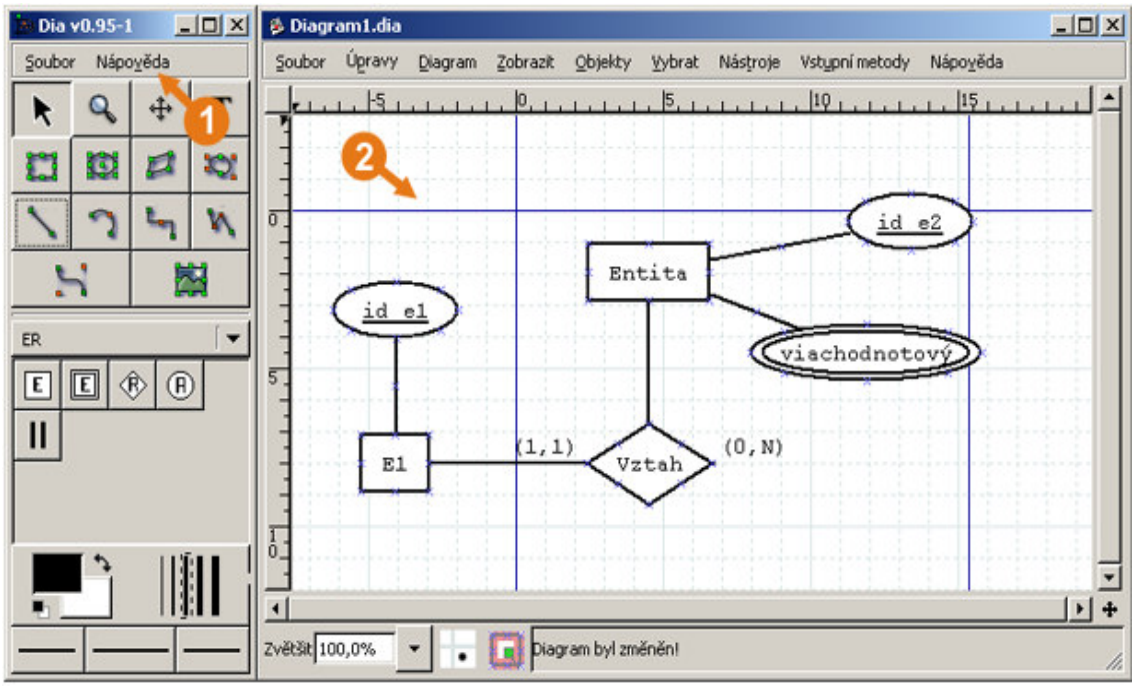
3. Analýza existujúcich programov

Dnes existuje celá rada kresliacich nástrojov, ktoré umožňujú vytvárať rôzne schémy, plány, obvody či diagramy. Ďalší text sa snaží podať krátky popis aspoň niektorých z nich, ktoré podporujú tvorbu ER či relačných schém.

3.1. Dia 0.95-1

Dia [17] je vektorovo orientovaný kresliaci nástroj vhodný pre modelovanie ER diagramov, UML diagramov, elektornických obvodov, či rôznych diagramov súvisiacich s popisom siete, chemických diagramov, chronogramov a mnohých ďalších. Dia obsahuje množstvo kolekcií tvarov používaných pri návrhu diagramov. Formulár pre návrh ER schémat v sebe zahŕňa tvary pre silný aj slabý entitný typ, vzťah, atribút a prepojujúcu čiaru, pričom sa používa Chenova notácia. U každého objektu je možné nastavovať jeho špecifické vlastnosti ako názov, farbu, typ čiar, font a ďalšie. U atribútu tiež napríklad, či bude kľúčový alebo viacnásobný. U vzťahu dovoľuje nastaviť navyše ľavú a pravú kardinalitu, čo je v prípade n-árnych vzťahov veľmi nepríjemne obmedzenie. Okrem týchto tvarov je pri tvorbe diagramu možné použiť akékoľvek iné prvky z ostatných ponúkaných modelov, ktoré nemusia nijako súvisieť s konceptuálnym návrhom a tak dovoľuje vytváť až nezmyselné schémy. Program Dia neponúka žiadnu kontrolu korektnosti diagramu a preto nie je veľmi vhodný na výukové účely. Rovnako neposkytuje žiaden nástroj na prevod takto vytvorenej schémy do relačného modelu, či SQL.

Program Dia pri spustení otvorí hneď tri okná, a to okno príkazového riadku a dve okná pre samotný editor (na obrázku 3.1 tieto dve okna označujú šípky):

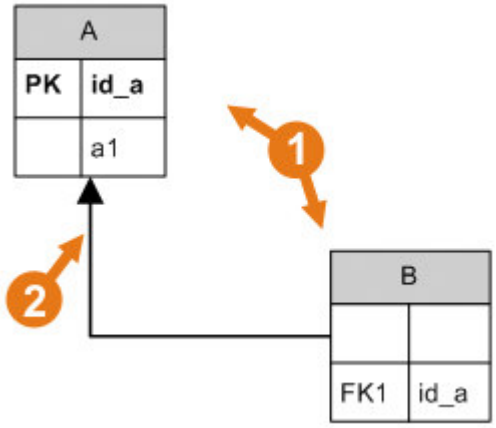


Obr. 3.1 Ukážka kreslenia ER schémy v programe Dia

Dia umožňuje export diagramov do najrôznejších typov obrázkových formátov a zvláda tiež viacstránkovú tlač diagramov. Samozrejmosťou je undo-redo manažment, kopírovanie objektov a zoomovanie diagramu. Príjemnou vlastnosťou tohto modelovacieho nástroja je prakticky neobmedzená pracovná plocha vo všetkých smeroch, čím uživateľovi odpadajú problémy s vhodným umiestnením diagramu na ploche. Program Dia možno považovať za čiste kresliaci a modelovací nástroj zvládajúci okrem kreslenia ER schém množstvo iných typov diagramov. Na výukové účely je takmer nepoužiteľný.

3.2. Microsoft Office Visio 2003

Microsoft Visio [18] je program určený na tvorbu najrôznejších typov obchodných či technických diagramov, v ktorých sú dokumentované a usporiadané zložité plány, procesy a systémy. Obsahuje tiež prostriedky pre prácu s databázami. Tentokrát, na rozdiel od programu Dia (viď kapitola 3.1), nejde čiste o kresliaci nastroj. Visio však pri návrhu databázového modelu chápe entity a vzťahy z pohľadu relačného modelu. Entitou sa myslí už samotná tabuľka a pri jej zakladaní sa definuje meno, názvy stĺpcov, primárny kľúč, indexy, trigery, prípadne ďalšie rozšírenia či poznámky. Vzťahy v chápaní Visia slúžia na vzájomné prepájanie týchto tabuliek a graficky sú zobrazené ako čiary, na ktorých koncové body sa tieto tabuľky zapoja. Na obrázku 3.2, šipka s číslom jedna určuje zapojené entity, šipka s číslom dva zase vzťah medzi nimi. Pri zapojení, ako je uvedené na obrázku, tabuľka *A* deleguje svoje kľúčové atribúty tabuľke *B,* v ktorej definujú cudzí kľúč do tabuľky *A*.



Obr. 3.2 Ukážka prepojenia dvoch tabuliek v aplikácii Visio

Pre datové modelovanie ponúka Visio dvojakú notáciu. Visio tiež dovoľuje špecifikovať integritné obmedzenia pre jednotlivé stĺpce. Aplikácia podporuje export takto navrhnutých tabuliek priamo do nejakej konkretnej databázy v závislosti na existujúcich inštalovaných ODBC driveroch, či generovanie SQL skriptov. Rovnako poskytuje funkcie pre kontroly korektnosti vytvoreného modelu, kde sa kontroluje jedinečnosť názvov tabuliek v rámci modelu, jedinečnosť názvov stĺpcov či existencia primárneho kľúča.

Pri návrhu rozsiahlejších databázových modelov uživatelia ocenia prácu s listami ako ich poznáme napríklad z aplikácie Microsoft Excel či možnosť definovať si vlastné rozmery diagramu. Visio tiež podporuje vytváranie uživateľsky definovaných typov

a tie dovoľuje použiť pri špecifikácii datového typu stĺpcov tabuliek. Okrem toho disponuje so širokou škálou preddefinovaných typov.

Microsoft Visio nie je rozhodne chudobný, čo sa týka podpory práce s rôznymi formátmi a v tomto smere za Dia (viď kapitola 3.1) nijako nezaostáva. A to platí rovnako pre import, ako aj pre export súborov. Z formátov, ktoré sú u programu Visio oproti Dia navyše, spomeniem formáty: AutoCad drawing, HTML, Macintosh PICT Format, Postcript File, Tag Image File Format, XML Drawing, XML Template na strane exportu a na strane importu formáty: Corel Clipart Format, CorelDraw Drawing File Format, Enhanced Metafile, JPEG File Format, Postcript File a ďalšie.

Ďalšou veľmi užitočnou funkčnosťou programu Visio je tzv. *reverse engineer*, ktorý z existujucej databáze dokáže extrahovať relačný model. Reverse engineer pracuje s ODBC zdrojom dat a teda podporuje databázy v závislosti na inštalovaných ODBC driveroch.

Pre výukové účely program Visio nie je vhodný hneď z niekoľkých dôvodov. Uživateľa, ktorý sa zoznamuje s problematikou databázových systémov, bude program odrádzať svojou náročnosťou, diagramy sa vytvárajú v notácii, ktorá je bližšia skôr relačnému modelu ako ER a SQL skripty generované z vytvorených diagramov častokrát obsahuju konštrukcie príkazov, s ktorými sa uživateľ určite nezoznámi na základnom kurze databázových systémov či jazyka SQL.

3.3. SmartDraw 2007

SmartDraw [19] je ďalší z rady kresliacich programov podobný programom Dia (viď kapitola 3.1) či Visio (viď kapitola 3.2), ktorý je možné použiť pre tvorbu širokej škály modelov a návrhov. Je s nim možné modelovať čokoľvek od máp, inžinierskych, medicínskych výkresov, cez softwarové či elektrické diagramy, fomuláre, až po náčrty vesmírnych telies alebo chemických rovníc. Jeho široký záber použitia je až prekvapujúci. Z tejto pestrosti ale vyplýva, že s jednotlivými modelmi pracuje len na úrovni grafického designu. Nejde nijak viac do hĺbky. V prípade modelovania ER diagramov je použiteľný zhruba na rovankej úrovni ako Dia, ma však príjemnejšie grafické rozhranie. Treba však poznamenať, že sa jedná o komerčný, cenovo náročný software.

Uživateľ, ktorý sa rozhodol vytvoriť databázový model pomocou ER schémat, by si mal najprv rozmyslieť, kam na pracovnú plochu svoj diagram umiestni. Oproti Dia je tu priestorové obmedzenie o rozšírovanie diagramu smerom nahor a doľava. SmartDraw po natahaní entít a vzťahov a ich vzájomnom prepojení pomocou spojovacích čiar, nie je schopný meniť pozíciu kotevných bodov čiar na entite či vzťahu autmaticky. To znamená, že ak sa uživateľ v istej chvíli rozhodne opticky premiestniť niektoré entity a vzťahy (napríklad aj kvôli názornejšej vizuálnej predstave), môže sa stať, že spojovacie čiary budú prechádzať skrz entity či vzťahy, čo v diagrame vyzerá veľmi rušivo. Pri rozsiahlych diagramoch bude reorganizácia spojovacíh bodov entít, vzťahov a čiar vyžadovať zbytočnú réžiu.

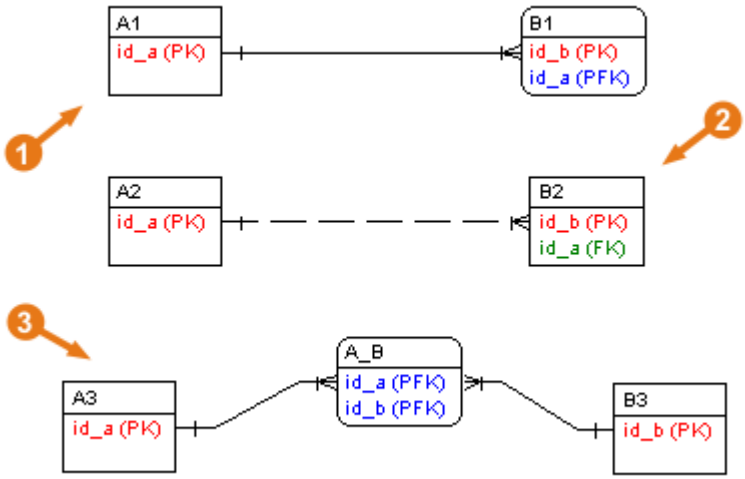
Aplikácia dokáže exportovať vytvorený diagram do rôznych obrazkových formátov a pokiaľ má uživateľ nainštalované programy Microsoft Word, Excel, PowerPoint alebo Adobe Acrobat, umožňuje tiež exportovať diagram ako obrázok priamo do týchto aplikácii.

Na výukové účely je táto aplikácia, rovnako ako Dia a Visio, nevhodná.

3.4. CASE Studio 2.19

CASE Studio [20] patrí, ako samotný názov hovorí ku CASE nástrojom pre datové modelovanie a databázový dizajn. Je stavaný výlučne na tvorbu RMD schémat. Pri spustení aplikácie a vytvární nového projektu (modelu) má uživateľ na výber z bohatého množstva databáz ako Access, Advantage, Clipper, DB2 UDB, Firebird, Informix, Ingres, Interbase, MaxDB, MS SQL, mySQL, Oracle, PostgreSQL a ďalších. Okrem návrhu relačných schém umožňuje CASE Studio vytvárať taktiež data-flow diagramy, pomocou ktorých si uživateľ môže veľmi pohodlne organizovať jednotivé procesy.

Pri samotnom vytváraní návrhu si uživateľ vytorí na pracovnej ploche tabuľky a definuje im atribúty, primárne a sekundárne kľúče, integritné obmedzenia, indexy a prípadne môže pripojiť aj ďalší popis a poznámky. Notácia, ktorú používa CASE Studio, pracuje len s tabuľkami, do vnútra ktorých sa zobrazujú ich atribúty a spájajúcimi čiarami, pre ktoré sa používa názov relácia. V kontexte tejto podkapitoly pojem relácia teda označuje prepájajúcu čiaru medzi tabuľkami. V aplikácii je možné použiť štyri typy týchto relácii, a to identifikačnú, neidentifikačnú, informatívnu a reláciu M:N. Zapájanie relácie sa musí aktivovať z nástrojovej lišty a následným klikom na tabuľku a pretiahnutím a uvoľnením nad druhou tabuľkou sa relácia vytvorí. U identifikačnej a neidentifikačnej relácie to znamená pripojenie kľúčových atribútov zo zdrojovej tabuľky do cieľovej a vytvorenie referenčnej integrity so zdrojovou tabuľkou. U identifikačnej relácie sa nové atribúty zo zdrojovej tabuľky stávajú súčasťou primárneho kľúča. Pri použití relácie M:N sa medzi spájanými tabuľkami vytorí tretia, ktorá automaticky obashuje primárne kľúče oboch spájaných tabuliek, ktoré sú zároveň cudzími kľúčmi týchto tabuliek. Informatívna relácia ma skutočne len informatívny význam, že takáto relácia existuje, no nespôsobí žiadnu akciu. Kontextové menu tabuľky ponúka možnosť vytvorenia ďalšieho typu relácie – self relácie.



Obr. 3.3 Ukážka rôznych relácii v CASE Studiu

Na obrázku 3.3 jednotlivé šipky ukazujú na:

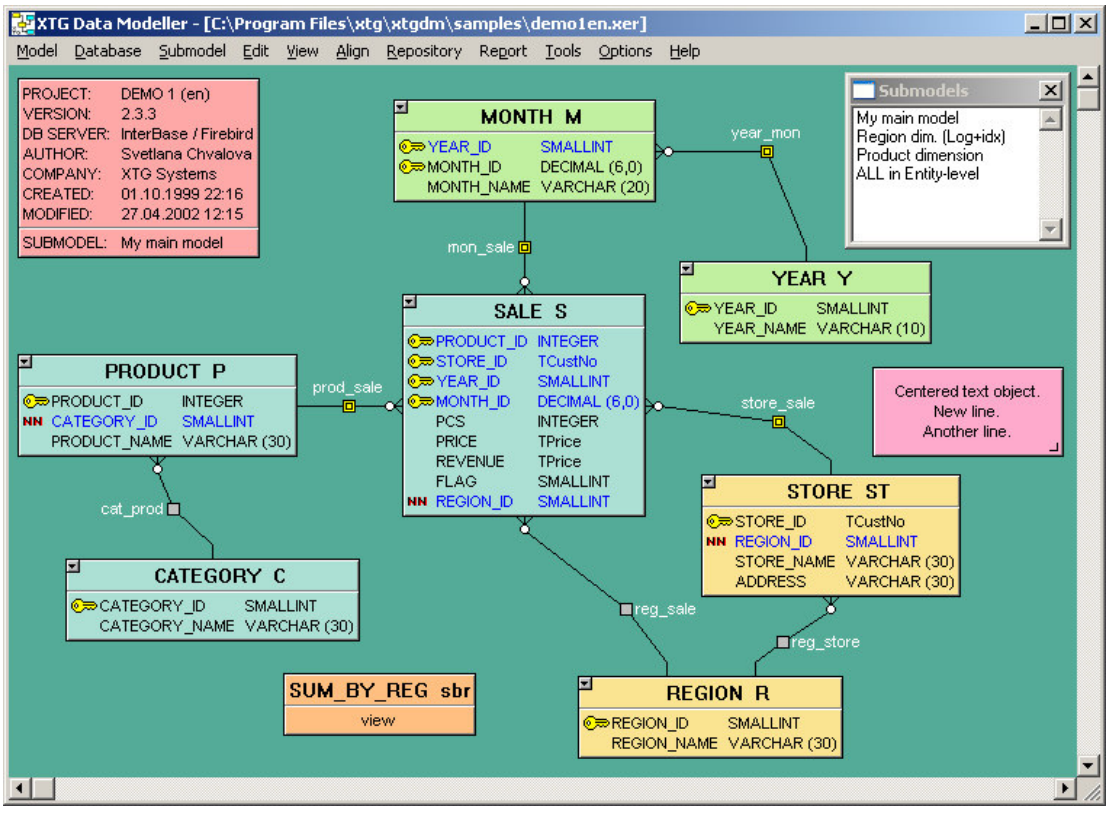
1. príklad použitia identifikačnej relácie
2. príklad použitia neidentifikačnej relácie
3. príklad použitia M:N relácie (tabuľka A_B sa vytvorí automaticky po prepojení tabuliek A, B)

Tento nástroj tiež umožňuje generovať SQL skripty podľa typu cieľovej databázy a podporuje aj reverse engineering. Pri generovaní skriptu je možné nastaviť, aké prvky sa majú generovať – tabuľky, indexy, primárne a cudzie kľúče, drop skript tabuliek a indexov a radu ďalších. Pred samotným generovaním, pri zaškrtnutej voľbe o požiadavok kontroly skriptu, je model tejto kontrole podrobený. Okrem iného sa kontroluje napríklad, či nie sú dve rôzne tabuľky pomenované rovnakým názvom, duplicity vzťahov či duplicity názvov stĺpcov tabuliek. V prípade, že v tabuľkovom návrhu sú chyby, generátor o tom informuje a uživateľ ma tak možnosť previesť model do korektného stavu. Takto vygenerovaný skript je možné uložiť do .txt, .sql alebo .ddl formátu.

CASE Studio ponúka možnosti pre export vytvorených diagramov do viacerých obrázkových formátov a tiež generovanie HTML a RTF reportov o vytvorenom diagrame.

Spomedzi všetkých spomínaných aplikácií, je práve CASE Studio svojím zameraním najviac podobný programu ER[G]edit, presnejšie editoru RMD modelov programu ER[G]edit. Práca bola v mnohých ohľadoch inšpirovaná možnosťami CASE Studia.

3.5. XTG Data Modeller 2.3.4



Obr. 3.4 Ukážka RMD shémy vytvorenej programom XTG Data Modeller

XTG Data Modeller [22] je ďalší z rady CASE nástrojov pre datové modelovanie. Jedná sa o editor RMD diagramov, používa tabuľkovú notáciu (Obr. 3.4 Ukážka RMD shémy vytvorenej programom XTG Data Modeller), kde jednotlivé tabuľky dokáže prepájať relačnými čiarami podobne ako program CASE Studio (viď kapitola 3.4). Ponúka tri typy relácii (v zmysle prepojujúcich čiar):

- identifikačnú reláciu
- neidentifikačnú reláciu
- self-reláciu

Voľbou kontextového menu *Edit* alebo dvojklikom na tabuľku sa vyvolá dialóg s vlastnosťami tabuľky. Dovoľuje editovať jej názov, alias, krátky popis a atribúty. Dialóg s vlastnosťami relačnej čiary umožňuje zmeniť kardinalitu vzťahu a definovať akcie, ktoré sa majú vykonať pri narušení referenčnej integrity.

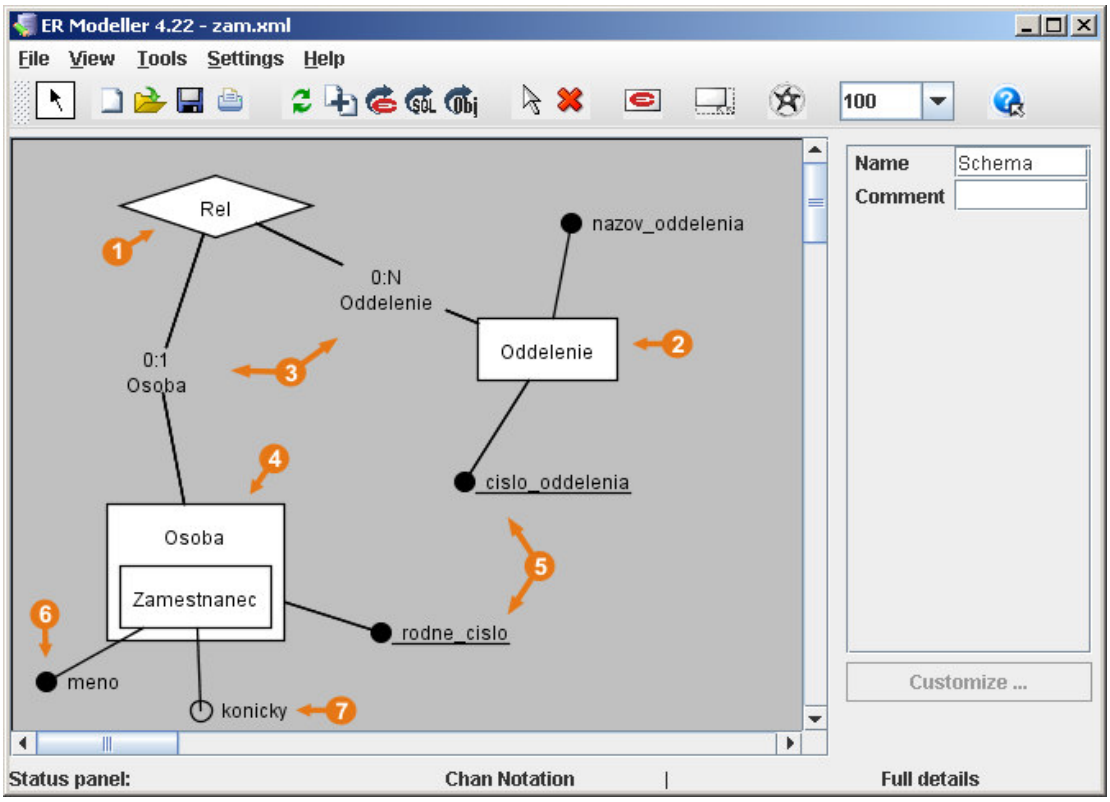
Program má priamu podporu pre prácu s najznámejšími databázami, ako napríklad: Firebird, MySQL, PostgreSQL, Oracle, Microsoft SQL Server, Microsoft Access, Informix a ďalšie. Disponuje tiež reverse engineeringom z ODBC datového zdroja alebo SQL. Diagramy dokáže exportovať do rôznych obrázkových formátov, generovať z nich veľmi prehľadné a užitočné HTML reporty ako aj SQL skripty a podporuje tiež viacstránkovú tlač.

3.6. ER Modeller 4.22

ER Modeller [23] je program, ktorý vznikol na akademickej pôde ako diplomová práca a postupne bol rozširovaný v rámci ďalších diplomových prác študentov ČVUT. Podporuje trojakú notáciu: Chenovú, binárnu a UML notáciu. Notáciu môže užívateľ kedykoľvek počas vytvárania návrhu zmeniť. Program nedovoľuje editáciu viacerých súborov v jednom okne, pracuje len s jedným formulárom pre kreslenie diagramov. Entity a vzťahy sa pridávajú jedine voľbou z kontextového menu. Rovnako aj prepojenia medzi entitami a vzťahmi, atribúty či ISA hierarchie. Každý atribút sa musí pridať zvlášť a následne je potrebné ho umiestniť na požadovanú pozíciu. Veľkosť objektov sa dá meniť ručne a nie je obmedzená veľkosťou ohraničujúceho textu.

Na obrázku 3.5 je kvôli demonštrácii jednotlivých prvkov uvedená ukážka ER modelu vytvoreného programom ER Modeller pri použití Chenovej notácie ako je chápana v kontexte tohto programu. Jednotlivé číselné šipky ukazujú na:

1. Vzťahový typ
2. Entitný typ
3. Kardinality vzťahu
4. ISA hierarchiu
5. Kľúčové atribúty
6. Povinný atribút
7. Nepovinný atribút



Obr. 3.5 Ukážka ER modelu vytvoreného v programe ER Modeller pri použití Chenovej notácie

Vytvorený ER diagram vie program uložiť do XML súboru, podporuje viacstránkovú tlač a export do obrázkových formátov. Program prevádza kontrolu vytvoreného modelu a u korektných diagramov generuje SQL skripty. Takto vytvorený skript umožňuje poslať pomocou JDBC driveru do databáze. Zaujímavou vlastnosťou je tiež spájanie vytvorených schém.

ER Modeller bol vytvorený s cieľom použitia na výukové účely a tento cieľ sa autorom podarilo splniť. Nedostatky mu je možné vytknúť pri zameraní sa na samotný GUI editor. Nepodporuje editáciu viacerých diagramov naraz, nedovoľuje kopírovať jednotlivé objekty, chýba undo-redo managment, vytváranie jednotlivých objektov diagramu je ťažkopádne (niekedy až mätúce) a úprava rozsiahlejších diagramov si vyžaduje veľku časovú réžiu.

3.7. ERTOS 1.0

Program ERTOS [21] vznikol ako bakalárska práca študentky Matematicko-fyzikálnej fakulty UK v roku 2006. Bol vytvorený s cieľom použitia pre výukové účely a keďže vychádza z rovnakej fakulty, je aj svojím zameraním najviac podobný programu ER[G]edit. Hneď pri spustení program zaujme svojím profesionálnym vzhľadom. Postup vytvárania diagramu je veľmi priamočiary a jednoduchý aj pre uživateľov, ktorí sa s problematikou tvorby konceptuálnych schém práve zoznamujú. Entity a vzťahy sa do diagramu pridávajú aktiváciou príslušného tlačítka z nástrojovej lišty a následným kliknutím na pracovnú plochu diagramu. Pridané entity sa so vzťahmi prepájajú pri aktivácii tlačítiek *Propojovat přímkou* alebo *Propojovat lomeně* z panela nástrojov a následným kliknutím nad jedným z objektov a uvoľnením nad druhým. Jednotlivé operácie sa dajú tiež aktivovať voľbou z hlavného menu aplikácie alebo klávesovou

skratkou. Self-relácie a ISA hierarchie sa vytvárajú po voľbe z kontextového menu entity.

Dvojklik nad entitou (prípadne voľba *Vlastnosti* z jej kontextového menu) vyvolá dialóg pre editáciu entity, v ktorom sa nastavujú jej atribúty, názov a identifikačná závislosť na vzťahoch, ku ktorými sa viaže kardinalitou (1,1). Rovnako sa vyvoláva dialógové okno pre vlastnosti vzťahu, v ktorom sa rovnako nastavujú atribúty vzťahu, názov a kardinality a role pre zapojené entity. Uživateľovi odpadá akákoľvek starosť s ručným zapájaním atribútov, či aktualizáciou ich polohy po zmene polohy entity a vzťahu. Problém vykresľovania atribútov je u aplikácie ERTOS riešený veľmi inteligentným (rozhodne nie jednoduchým) spôsobom. Veľkosť entít a vzťahov nie je možné v diagrame meniť, vypočítava sa automaticky podľa veľkosti popisného textu, čím odpadá uživateľovi ďalšia starosť s vizuálnou stránkou diagramu. Veľmi príjemnou vlastnosťou je použitie „vysúvacích“ okien, ktoré sa zobrazia po kliku alebo najetí myšou. Tieto okna je možné v diagrame ľubovoľne premiestňovať a upevňovať na okraje pracovnej plochy, alebo ich ponechať neupevnené kdekoľvek vo vnútri diagramu, či úplne zatvoriť, čím si uživateľ volí vzhľad pracovného prostredia podľa vlastných potrieb. Ide o okná, ktoré zobrazujú formuláre pre slovník objektov modelu, poznámky o diagrame, výsledkoch kontroly a histórii zmien.

Program ERTOS podporuje štandardné funkcie editoru, ako sú napríklad kopírovanie časti diagramu, zoomovanie, undo-redo manažment, ukladanie a načítanie diagramu do XML súboru (dokonca aj možnosť uložiť diagram s históriou zmien) a export do obrázkových formátov. Chýba však možnosť tlače diagramu, čo ale nie je žiaden vážny nedostatok, keďže ponúka možnosti exportu diagramu do obrázkového formátu. Pracovná plocha nie je v medziach dostupnej pamäte nijak obmedzená.

Vytvorený diagram je možné podrobiť kontrole na vyžiadanie alebo automaticky pred generovaním SQL skriptu. Kontroluje sa:

- jedinečnosť názvov entít a vzťahov vrámci diagramu
- jedinečnosť názvov atribútov v rámci entity alebo vzťahu
- existencia cyklu identifikačných závislosti
- existencia cyklu v ISA hierarchii
- násobná dedičnosť v ISA hierarchii
- špecifikovanie rolí u entít, ktoré do vzťahu vstupujú viackrát
- existencia identifikačného kľúča u entít, ktoré sú zdrojom ISA hierarchie
- povinnosť nexistencie identifikačného kľúča u entít, ktoré nie sú zdrojom ISA hierarchie

Pred generovaním skriptu uživateľ špecifikuje, či chce vo výslednom skripte zobrazovať komentáre, pomenovávať integritné obmedzenia a či sa bude generovať tiež drop skript tabuliek. Jednotolivé príkazy pre mazanie tabuliek sa generujú v opačnom poradí ako príkazy pre vytvorenie tabuliek a poradie príkazov pri ich prevedení vo väčšine prípadov nenarúša referenčnú integritu. Po nastavení cesty k súboru pre uloženie skriptu sa vytvorí aj súbor so skriptom. Syntax výsledného skriptu nie je parametrizovateľná podľa typu cieľovej databáze a výsledný skript preto nemusí byť vykonateľný na niektorej konkrétnej databázovej platforme. Zo zamerania práce však plynie, že vytvorené SQL skripty majú skôr ilustračný charakter, než žeby slúžili pre vytváranie konkrétnych databáz.

Program ERTOS je spomedzi všetkých doposiaľ popisovaných programov najvhodnejší na výukové účely a jedným z možných rozšírení programu ER[G]edit by mohol byť import a export diagramov pre tento program.

4. Požiadavky kladené na editor

4.1. Základné požiadavky

Základné požiadavky na nový CASE (z angl. **C**omputer **A**ided **S**oftware **E**ngineering) nástroj s názvom ER[G]edit, ktorý vznikol vrámci tejto práce, vychádzajú z existujúcich riešení. CASE nástroje definujeme ako programy, ktoré podporujú vývoj softwarových aplikácii. V súčasnej dobe ich existuje veľké množstvo a líšia sa v mnohých smeroch. Vychádzame najmä z tých, ktoré podporujú datové modelovanie a návrh databáz aspoň na niektorých úrovniach (kapitola 3). Hlavnou nevýhodou všetkých aplikácii zmienených v tretej kapitole je, že nepodporujú viacúrovňovú tvorbu návrhu databáz. V tomto smere je program ER[G]edit celkom jedinečný v tom, že ponúka dvojúrovňový editor pre tvorbu databázových modelov a generovanie SQL skriptov. Na prvej úrovni to je editor konceptuálnych ER schémat, podobný programom ER Modeller (kapitola 3.6) alebo ERTOS (kapitola 3.7), na druhej úrovni sa jedná o editor RMD schémat, podobný programom CASE Studio (kapitola 3.4), či XTG Data Modeller (kapitola 3.5). Program je zameraný na použitie pre výukové účely, preto si kladie za cieľ poskytnúť tiež nástroje na prevod z konceptuálneho ER diagramu do RMD diagramu, generovanie SQL skriptov z vytvorených RMD diagramov a kontrolu korektnosti vytvorených diagramov. ER[G]edit tak poskytuje komplexný nástroj pre tvorbu databáz od najnižšej konceptuálnej urovne až po výsledný SQL skript. Jednotlivé úrovne editora sú navzájom nezávisle a uživateľ môže začať s databázovým návrhom už tvorbou RMD diagramu.

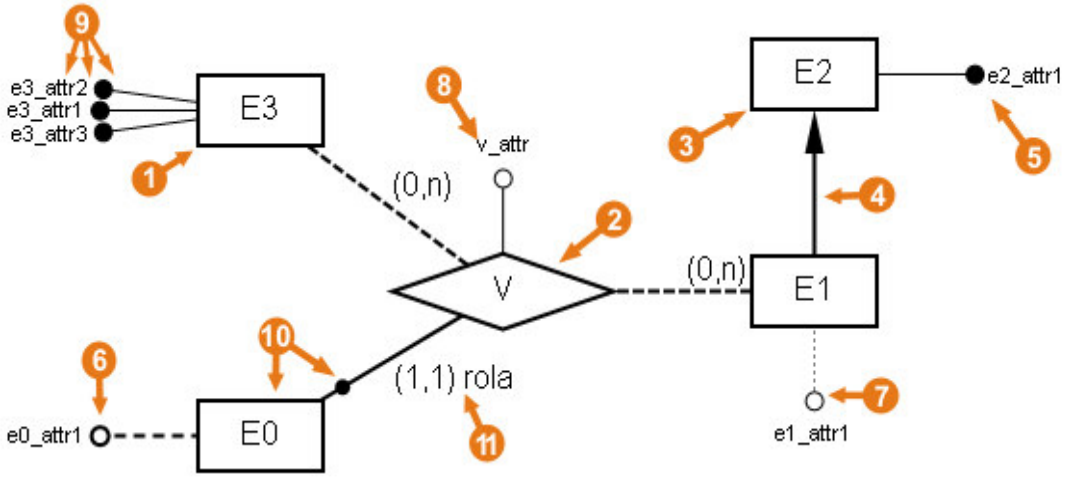
4.2. ER[G]edit ako CASE nástroj

CASE nástroje nie sú len aplikáciami na kreslenie diagramov, ale musia podporovať metodiky podľa svojho zamerania. CASE samo o sebe nie je metodológiou, ale používa už zavedené metodológie. Môže generovať časti kódu no rozhodne nie je náhradou programovacieho jazyka. Z toho, aké sú obecné funkcie a vlastnosti CASE nástrojov, vyplýva tiež to, z akých komponent sa skladajú. Medzi dôležité funkcie a vlastnosti CASE patria [15]:

- Konzistenté grafické ovládacie prostredie. Grafické rozhranie sa skladá z preddefinovaných obrazových primitív (štvorce, priamky, krivky, šipky) s možnosťou ich uloženia. Jednou z jeho vlastností je aj to, že systém automaticky vymaže odkazy na mazaný objekt, aby bola zachovaná konzistencia schémy. Grafické rozhranie by malo dovoliť premiestnenie jedného či viacerých objektov a tiež umožniť priradenie a editáciu názvov objektov.
- Centrálna databáza pre uchovávanie informácii o všetkých objektoch, tzv. *slovník*. Práve prítomnosť slovníka zaraďuje danú aplikáciu do rodiny CASE nástrojov.
- Textový editor pre popis jednotlivých objektov.
- Export a import dat.
- Kontrola korektnosti modelu.

4.3. Grafické prvky ER editora

Veľmi dôležitým požiadavkom na aplikáciu ER[G]edit je možnosť zakreslenia prvkov ER diagramu preberaných v rámci prednášky Databázové systémy na Matematicko-fyzikálnej fakulte Univerzity Karlovej.



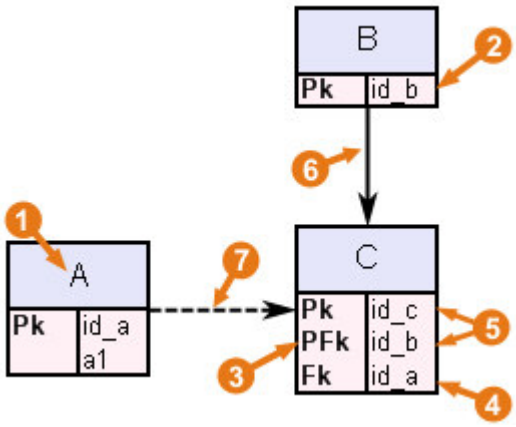
Obr. 4.1 Ilustračná ukážka jednotlivých prvkov ER diagramu v programe ER[G]edit

Na obrázku 4.1 je pre ilustračné dôvody zobrazený ER diagram obsahujúci všetky možné prvky diagramu a jednotlivými číselnými šipkami sú ukázané:

1. Entita
2. Vzťah
3. Zdroj ISA hierarchie
4. Generalizačná šipka (ISA vzťah)
5. Identifikačný atribút (jednoduchý identifikátor)
6. Viachodnotový atribút
7. Nepovinný atribút
8. Atribút vzťahu (povinný)
9. Zložený identifikátor
10. Identifikačne závisla entita – čierny kruh na čiare špecifikuje vzťah, na ktorom je entita identifikačne závislá
11. Kardinalita a rola vzťahu

4.4. Grafické prvky RMD editora

Modelovanie RMD schémat pre vyjadrenie jednotlivých konštruktov používa notáciu podobnú notácii v programoch Microsoft Visio (kapitola 3.2) alebo CASE Studio (kap 3.4), u ktorej sa jednotlivé atribúty (budeme ich tiež nazývať stĺpce tabuľky) zobrazujú do tabuľky pod jej názov a k nim sa pripisujú indikátory, či sa jedna o primárny prípadne cudzí kľúč.



Obr. 4.2 Ilustračná ukážka jednotlivých prvkov RMD diagramu v programe ER[G]edit

Na obrázku 4.2 je pre ilustráciu zobrazený RMD diagram obsahujúci všetky možné prvky diagramu a jednotlivými číselnými šípkami sú ukazané:

1. Tabuľka a jej záhlavie s názvom
2. Atribút (stĺpec tabuľky), ktorý je primárnym kľúčom tabuľky *B* (indikátor **Pk**)
3. Atribút, ktorý je súčastne primárnym i cudzím kľúčom do tabuľky *B* (indikátor **PFk**)
4. Atribút, ktorý je cudzím kľúčom do tabuľky *A* (indikátor **Fk**)
5. Atribúty, ktoré tvoria primárny kľúč tabuľky *B* (indikátor **Pk** alebo **PFk**)
6. *Identifikačná relačná šipka* definujúca referenčnú integritu medzi tabuľkami, kedy sa pridávané atribúty stávajú tiež súčasťou primárneho kľúča
7. *Neidentifikačná relačná šipka* definujúca referenčnú integritu medzi tabuľkami

4.5. Podpora rôznych verzií SQL

Program ER[G]edit ponúka možnosť generovania SQL skriptu pre niekoľko typov najznámajších databáz, aby bolo užívateľom dovolené vyskúšať si ich funkčnosť na konkrétnej databázovej platforme. Parametrizácia pre konkrétnu platformu nie je zohľadnená od začiatku návrhu, ako tomu bolo napríklad u programu CASE Studio (kapitola 3.4). Pre výukove účely boli datové typy použité pre jednotlivé atribúty entít a vzťahov u ER modelu a tabuliek u RMD modelu obmedzené na základné, ktoré používa väčšina databázových platforiem. A to: *integer, float, date, char, varchar.* U výsledneho skriptu sa teda zohľadňuje názov pre daný datový typ, integritné obmedzenia a syntax príkazov pre mazanie tabuliek, aby (pokiaľ to je možné) vykonanie príkazu nekončilo chybou pri pokuse odstrániť neexistujúcu tabuľku. V programe ER[G]edit je teda možné pripôsobiť syntax výsledného SQL skriptu pre tieto databázové platformy:

- SQL-92
- MySQL
- MSSQL
- Oracle
- PostgreSQL 8.1 a nižši
- PostgreSQL 8.2 a vyššie (od verzie 8.2 PostgreSQL ponúka možnosť príkazu `IF EXISTS` pri odstánení tabuľky)

Pred generovaním skriptu môže uživateľ nastaviť, či sa budú generovať aj príkazy pre mazanie tabuliek a či sa majú pomenovávať integritné obmedzenia pre primárne a cudzie kľúče. Skript sa dá uložiť do súboru.

5. Uživateľská dokumentácia

5.1. Požiadavky na OS a prerekvizity

Program ER[G]edit je aplikácia bežiaca pod operačným systémom Windows 98 a vyšším vytvorená v Microsoft Visual Studiu 2005. Program vyžaduje k svojmu behu nainštalovaný .NET Framework 2.0. Ten je možné si stiahnúť zdarma z internetu a rovnako je aj súčasťou inštalačného balíčka programu ER[G]edit pre offline inštaláciu.

5.2. Inštalácia a spustenie programu

Aplikácia ER[G]edit sa inštaluje spustením inštalačného balíčka s názvom ErgeditSetup.msi. Inštalačný program požaduje ako prerekvizitu nainštalovaný .NET Framework 2.0. V prípade, že na cieľovom počítači nie je nainštalovaný, bude uživateľ vyzvaný k jeho stiahnutiu z internetu. Po potvrdení tejto výzvy sa uživateľovi v jeho defaultnom prehliadači automaticky otvorí stránka, odkiaľ si môže .NET Framework stiahnúť. Inštalačný balíček ER[G]edit-u je tiež dodavaný vo verzii pre offline inštaláciu spolu s inštalačným balíčkom .NET Frameworku. Po nainštalovaní aplikácie sa vytvorí odkaz do menu programov a na pracovnú plochu.

Nainštalovaný program sa spúšťa súborom Ergedit.exe, ktorý sa nachádza v inštalačnom adresári aplikácie spolu s adresármi pre ukladanie vytvorených modelov a generovaných skriptov. Pre jednoduchšie zoznámenie sa s aplikáciou, nachádza sa v inštalačnom adresári tiež adresár *Priklady,* ktorý obsahuje sadu ukážkových diagramov. Rovnako sa v tomto adresári nachádza aj uživateľská príručka, s názvom ErgeditDoc.pdf.

Uložené diagramy (prípony .erm, .rmd) nie sú asociované s aplikáciou. Preto pre ich otvorenie je potrebné aplikáciu najprv spustiť a následne si diagram otvoriť voľbou z hlavného menu alebo toolbaru.

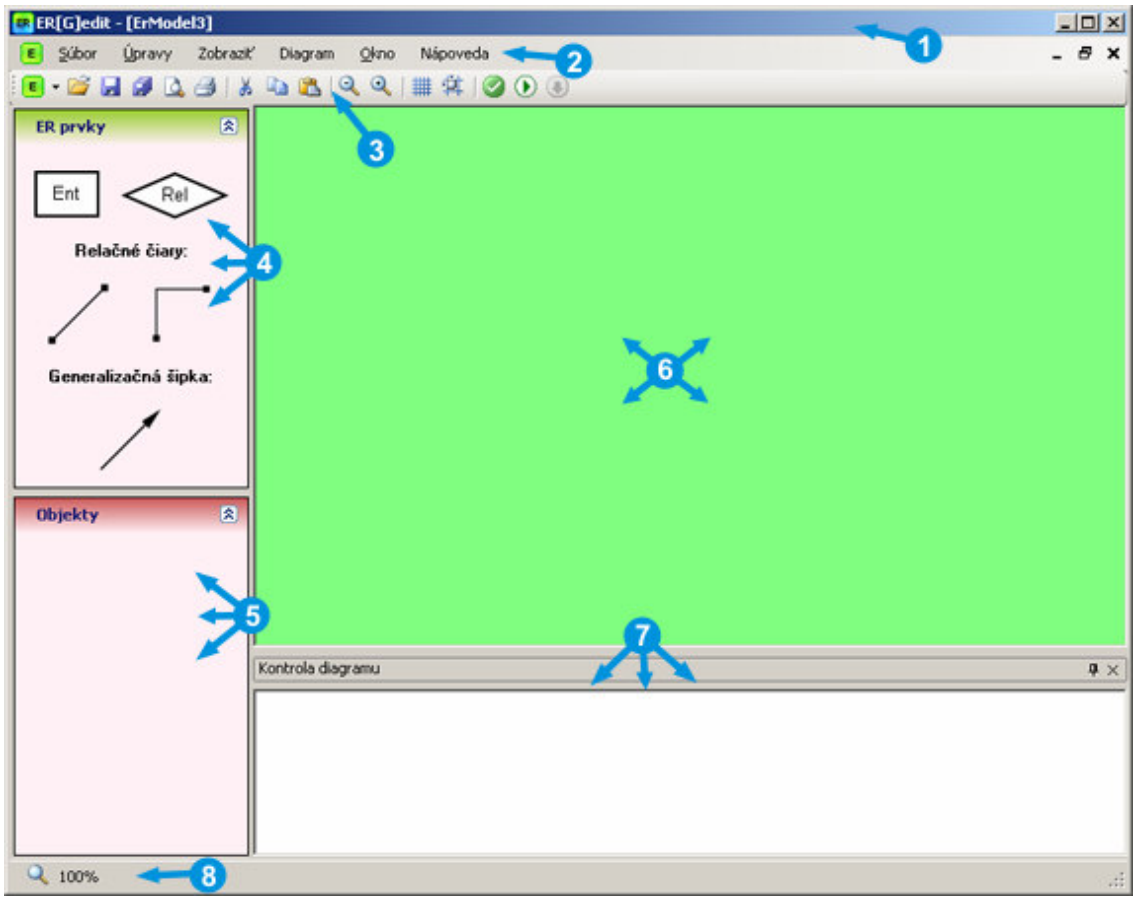
5.3. Editor ER diagramov

Pri spustení aplikácie sa hlavné okno (pri štandartných nastaveniach) otvára maximalizovane a nastaví prázdny ER diagram. Program je MDI aplikáciou (angl. Multiple Document Interface [14]) a teda umožňuje otvorenie viacerých okien vrámci jedného programu. V závislostí na tom, ktorý z modelov je aktívny, sa zobrazuje typ bočnej nástrojovej lišty. Na obrázku 5.1 sú jednotlivými číselnými šipkami zobrazené:

1. Hlavné okno aplikácie.
2. Panel menu.
3. Panel nástrojov.
4. Panel ER prvkov – obsahuje prvky pre entitu, vzťah, priamu čiaru, lomenú čiaru a generalizčnú šipku (ISA vzťah).
5. Panel objektov ER diagramu, ktorý slúži ako slovník (kapitola 4.3). Objekty v ňom sú usporiadané lexikograficky podľa názvu. Kliknutím na niektorý z nich, sa označí odpovedajúci objekt na pracovnej ploche. Dvojklik vyvolá dialóg s vlastnosťami objektu (viď nižšie).
6. Pracovná plocha diagramu, na ktorú je možné prenášat prvky z panela ER prvkov.

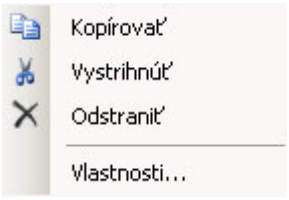
7. Vysúvacie okno pre zobrazenie výsledkov kontroly diagramu.
8. Stavový riadok – informuje o aktuálnom zväčšení/zmenšení pracovnej plochy.

V prípade, že nie je otvorený žiaden z diagramov, aplikácia obsahuje len panel menu, panel nástrojov a vysúvacie okno pre výsledky kontroly. Prepínať sa medzi jednotlivými diagramami je možné pomocou voľby menu *Okno*, kde sa nachádza zoznam všetkých otvorených diagramov. Viac o voľbách menu popisuje kapitola 5.7.



Obr. 5.1 Okno aplikácie s aktívnym ER diagramom

Pridávanie entít, vzťahov, spojujúcich čiar a generalizačných šipiek (ISA vzťahov) v ER diagrame je možné vykonať spôsobom drag-and-drop prenesením z bočného panela prvkov. Objekt sa vykreslí na tom mieste diagramu, na ktorom užívateľ uvoľní tlačítko myši. Objekt uvoľňený mimo viditeľnú oblasť diagramu vyvolá aktiváciu scrollbarových líšt. Ak je objekt uvoľnený mimo pracovnú plochu diagramu, nie je vykreslený vôbec. Pojmom objekt, v kontexte tejto kapitoly, budeme uvažovať entitu, vzťah, priamu čiaru, lomenú čiaru a generalizačnú šipku z panela ER prvkov. Každý z týchto objektov sa v diagrame môže vyskytovať v aktívnom alebo pasívnom stave, čím sa chápe označenosť alebo neoznačenosť objektu. V danej chvíli však môže byť aktívny nanajvýš jeden objekt. Klik myšou do videteľnej časti objektu ho nastaví ako aktívny. Klik myšou mimo ktorýkoľvek objekt diagramu zruší aktívny objekt a program je v režime, kedy čaká na obsluhu ďalšej udalosti. Pre každý z objektov je možne vyvolať kontextové menu, v prípade, že je objekt aktívny. Kontextové menu ponúka možnosti pre kopírovanie, vystrihnutie a odstránenie objektu. Pre všetky objekty s výnimkou generalizačnej šipky existuje tiež voľba *Vlastnosti* (Obr. 5.2).



Obr. 5.2 Ukážka kontextového menu

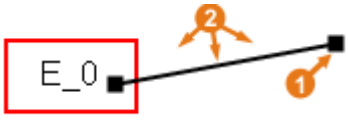
Voľba kontextového menu *Vlastnosti* umožňuje uživateľovi v dialógovom okne nastaviť špecifické vlastnosti daného objektu. Dialóg s vlastnosťami je tiež možné vyvolať dvojklikom na objekt. Nastavenie vlastnosti entity je podrobnejšie popísané v kapitole 5.3.2, vlasnosti vzťahu v kapitole 5.3.3 a nakoniec vlastnosti čiary a lomenej čiary v kapitole 5.3.4.

Ktorýkoľvek z objektov je možné z diagramu odstrániť označením objektu a následným stlačením klávesy Delete alebo voľbou z kontextového menu – *Odstrániť*.

5.3.1. Presun, zmena veľkosti a prepájanie objektov

Aktívny objekt diagramu je možné vrámci diagramu ďalej presúvať metódou *drag-and-drop*. Ktorý objekt je v danej chvíli aktívny, sa pozná podľa zmeny farby pozadia objektu (v prípade, že ide o entitu a vzťah) alebo vykreslením plného čierneho štvorca na koncoch čiary (v prípade, že ide o niektorú z čiar). Zmena veľkosti objektu (v prípade entity a vzťahu) sa deje automaticky so zmenou jeho názvu, prepočítaním veľkosti zadaného reťazca v obrazových bodoch a následným zmenšením či zväčšením objektu.

V prípade aktívneho čiarového objektu je pri manipulácii metódou drag-and-drop dôležitá tiež oblasť uchytenia objektu. Uchytenie a následný posun plného štvorca na konci čiary (na obrázku 5.3 označuje šípka s číslom jedna) spôsobí zväčšovanie a zmenšovanie čiary pri zakotvení protiľahlého koncového bodu. Uchopenie v oblasti medzi týmito koncovými štvorcami (na obrázku 5.3 označuje šípka s číslom dva) a následný posun spôsobí presun objektu na pracovnej ploche diagramu.



Obr. 5.3 Ukážka prepojenia entity a čiary

Čiary je možné so vzťahmi a entitami vzájomne prepájať. Zapájajú sa čiary do entít a vzťahov, nikdy nie naopak. Čiara sa uchopí za niektorý z koncových štvorcov, presunie sa nad viditeľnú oblasť entity alebo vzťahu, s ktorým sa prepojenie požaduje. V prípade, že ide o dovolené spojenie, obvod entity (vzťahu) sa vykreslí červenou farbou a po uvoľňení tlačítka myši sa čiara spoji s entitou (vzťahom) (Obr. 5.3). Aby sa zabránilo vytváraniu nezmyselných návrhov a tiež kvôli uľahčeniu neskoršej kontroly diagramu, prepájanie objektov je čiastočne obmedzené a kontrolované už pri vytvárani ER modelu. Editor neumožňuje prepojiť čiarou vzájomne dve entity či dva vzťahy, vždy len entitu so vzťahom. Taktiež nie je dovolené generalizačnou šipkou vzájomne prepojiť entitu a vzťah, ale len dve entity. V prípade pokusu o nedovolené zapojenie,

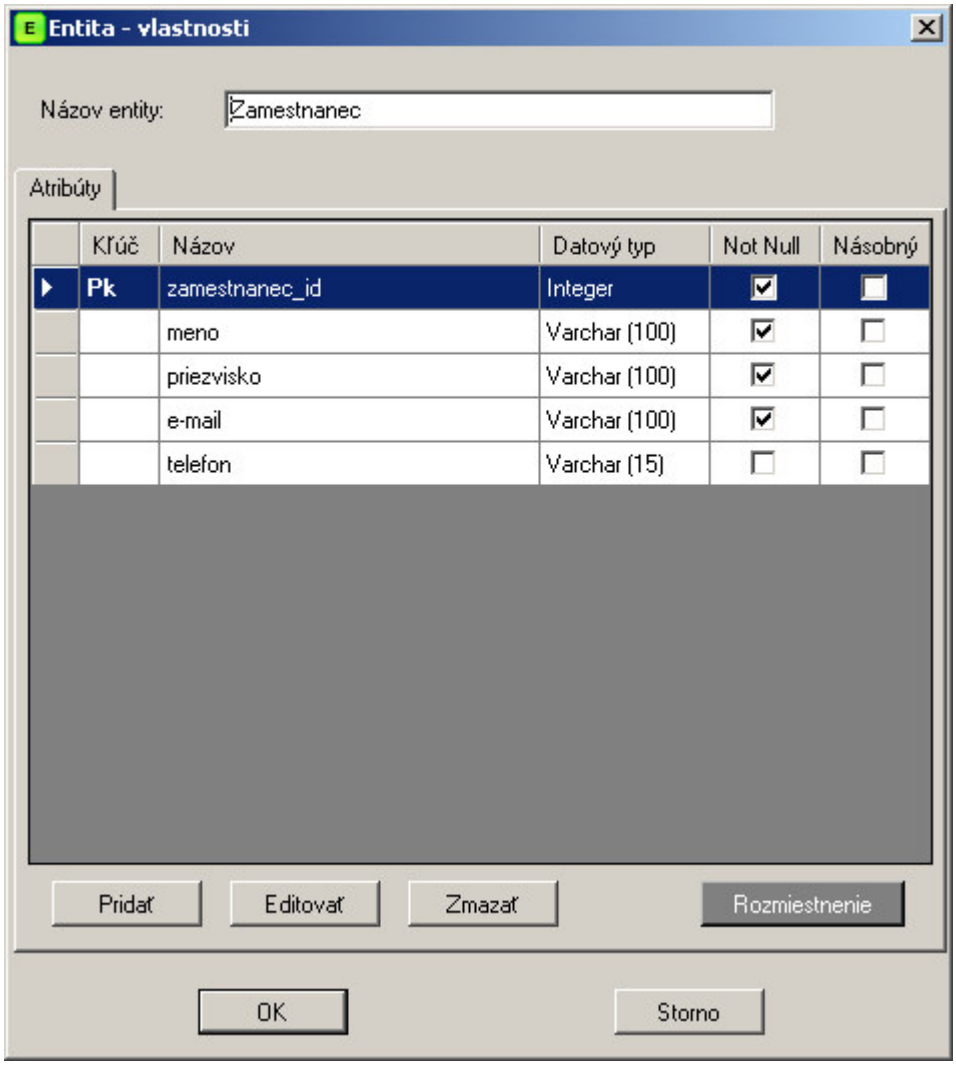
zostane entita (vzťah) bez reakcie a uvoľnenie koncového bodu čiary nad entitou (vzťahom) nespôsobí zapojenie. Čiara bude prekrývať entitu, no oba objekty zostanú navzájom nezávislé.

V prípade, kedy čiara alebo lomena čiara spojuje entitu so vzťahom, je možné nastaviť jej kardinalitu, rolu a identifikačnú závislosť vyvolaním dialógu s vlastnosťami čiary (kapitola 5.3.4).

Odpájanie čiar od entít a vzťahov sa prevádza podobným spôsobom ako zapájanie. Najprv sa nastaví aktívna čiara a uchopením jej koncového bodu a odtiahnutím z vnútornej oblasti entity (vzťahu) a následným uvoľnením kurzora sa čiara odpojí. Pri odpájaní sa odpájaná entita (vzťah) vykreslí modrou farbou.

5.3.2. Editácia a vlastnosti entity

Voľbou *Vlastnosti* z kontextového menu entity alebo dvojklikom na entitu sa vyvolá dialógové okno s vlastnosťami danej entity (Obr. 5.4).

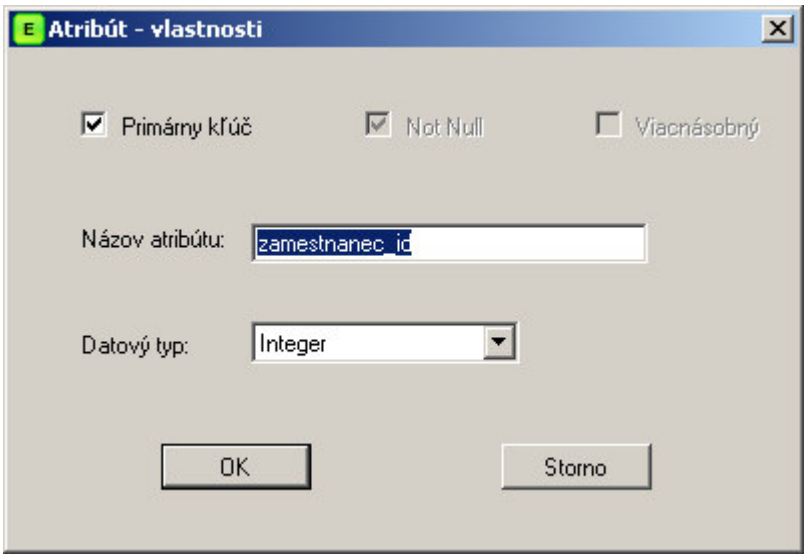


Obr. 5.4 Okno pre nastavenie vlastnosti entity

Dialóg je modálny a teda ďalšie modifikácie diagramu nie sú dovolené dovetedy, kým sa okno nezavrie. Pre entitu je možné nastaviť názov entity a definovať jej atribúty

pomocou tlačítka *Pridať*. Označenie riadku s daným atribútom a klávesa Enter, dvojklik alebo tlačítko *Editovať* umožňuje editovať daný atribút. Atribúty sa odstraňujú označením príslušného riadku a stlačením klávesy Delete alebo pomocou tlačítka *Zmazať*.

Pri pridávaní alebo editácii atribútu sa modálne otvorí dialógove okno, ktoré umožňuje nastaviť vlastnosti atribútu entity (Obr. 5.5).



Obr. 5.5 Okno pre nastavenie vlastnosti atribútu entity

Uživateľ má možnosť nastaviť nasledujúce vlastnosti:

- *Názov atribútu*
- *Datový typ* – uživateľ ma možnosť z roletového (drop-down) zoznamu zvoliť jeden z ponúkaných datových typov – *integer, float, date, char, varchar*. Pri voľbe datového typu *varchar* sa automatický zobrazí textové pole pre zadanie veľkosti varchar-u. V prípade, že uživateľ neuvedie dĺžku pre tento datový typ, bude defaultne nastavená dĺžka 255 znakov. V prípade, že uživateľ do poľa vloží nečíselnú hodnotu, prípadne čislo mimo interval 0 – 255, bude nastavená defaultna dĺžka 255 znakov.
- *Primárny kľúč* – pri zaškrtnutí tejto voľby bude atribút súčasťou primárneho kľúča. Taktiež sa automaticky nastaví vlastnosť *Not Null* (viď ďalší bod), ktorú nie je možné zmeniť po dobu, kým je nastavená voľba *Primárny kľúč.*
- *Not Null* – pri zaškrtutí tejto voľby bude vyžadovaná povinnosť tohto atribútu.
- *Viacnásobný* – zaškrtnutá voľba indikuje, ze atribút môže nabývať viac hodnôt (inak tiež viachodnotový atribút).

Tlačítkom *OK* sa prevedie uloženie vlastností atribútu, okno sa zatvorí a aktívnym oknom bude opäť dialóg vlastností entity. Pred uložením sa však kontroluje, či je vyplnený názov atribútu. Stlačením tlačítka *Storno*, sa neprevedie pridanie či editácia atribútu a aktivným oknom bude opäť dialóg pre editáciu vlastností entity.

Atribúty entity sa v diagrame vykresľujú automaticky, uživateľ však nastavuje hrany entity, ku ktorým sa atribúty budú vykresľovať. A to tlačítkom *Rozmiestnenie,* ktoré

vyvolá modálny dialóg, kde je možné nastaviť dovolené hrany pre zobrazovanie atribútov (Obr. 5.6). V prípade, že nebude zaškrtnutá žiadna z hrán, atribúty entity sa nebudú vykresľovať. Inak sa vykresľujú u zvolených hrán, v smere hodinových ručičiek počnúc hornou hranou. Zmeny sa prevedú stlačením tlačitka *OK,* tlačítko *Storno* ignoruje prevedené zmeny. V oboch prípadoch sa aktívnym oknom opäť stane okno s vlastnosťami entity.

Obr. 5.6 Okno pre nastavenie dovolených hrán pre zobrazovanie atribútov

Uloženie editovaných vlastností entity sa prevedie stlačením tlačítka *OK*. Pred uložením sa vykoná kontrola na neprázdnosť názvu entity. Zrušenie prevádzaných zmien sa prevedie stlačením tlačítka *Storno.* V takomto prípade sa neprevedú žiadne zmeny a entita ma rovnaké vlastnosti ako pred otvorením dialógu s vlastnosťami.

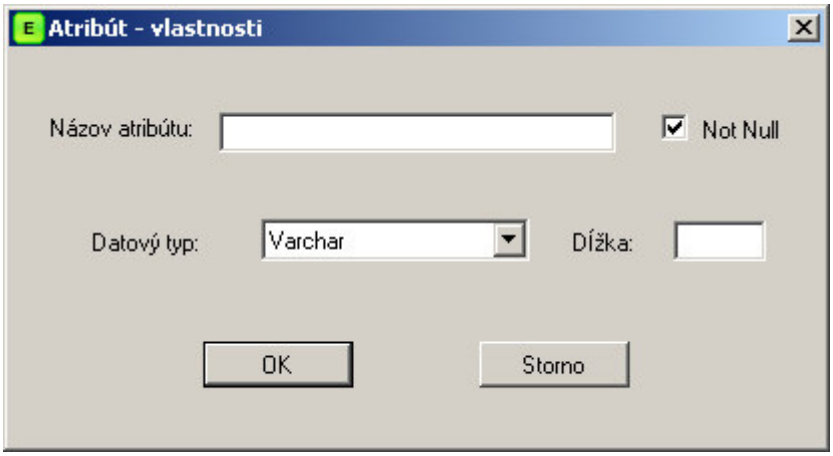
5.3.3. Editácia a vlastnosti vzťahu

Voľbou *Vlastnosti* z kontextového menu vzťahu alebo dvojklikom na vzťah sa vyvolá modálne dialogové okno s vlastnosťami daného vzťahu podobné oknu pre vlastnosti entity (viď kapitola 5.3.2). Pre vzťah je možné nastaviť názov vzťahu a definovať jeho atribúty pomocou tlačítka *Pridať*. Označenie riadku s daným atribútom a stlačenie klávesy Enter, dvojklik na riadku alebo tlačítko *Editovať* umožňuje editovať daný atribút. Atribúty sa odstraňujú označením príslušného riadku a stlačením klávesy Delete alebo pomocou tlačítka *Zmazať*.

Pri pridávaní alebo editácii atribútu sa modálne otvorí dialógove okno, ktoré umožňuje nastaviť vlastnosti atribútu vzťahu (Obr. 5.7).

Uživateľ má možnosť nastaviť nasledujúce vlastnosti:

- *Názov atribútu*
- *Datový typ* – uživateľ má možnosť z roletového (drop-down) zoznamu zvoliť jeden z ponúkaných datových typov – *integer, float, date, char, varchar*. Pri voľbe datového typu *varchar* sa automatický zobrazí textové pole pre zadanie veľkosti varchar-u. V prípade, že uživateľ neuvedie dĺžku pre tento datový typ, bude defaultne nastavená dĺžka 255 znakov. V prípade, že uživateľ do poľa vloží nečíselnú hodnotu, prípadne čislo mimo interval 0 – 255, bude nastavená defaultna dĺžka 255 znakov.
- *Not Null* – pri zaškrtutí tejto voľby bude vyžadovaná povinnosť tohto atribútu.



Obr. 5.7 Okno pre vlastnosti atribútu vzťahu

Tlačítkom *OK* sa prevedie uloženie vlastností atribútu, okno sa zatvorí a aktívnym oknom bude opäť dialóg vlastností vzťahu. Pred uložením sa však kontroluje, či je vyplnený názov atribútu. Stlačením tlačítka *Storno*, sa neprevedie pridanie či editácia atribútu a aktivným oknom bude opäť dialóg pre editáciu vlastností vzťahu.

Atribúty vzťahu sa vykresľujú automaticky, uživateľ však môže obmedziť ich vykresľovanie k jednotlivým stranám stlačením tlačítka *Rozmiestnenie* z okna pre editáciu vlastností vzťahu, podobne ako to bolo u entity (viď kapitola 5.3.2).

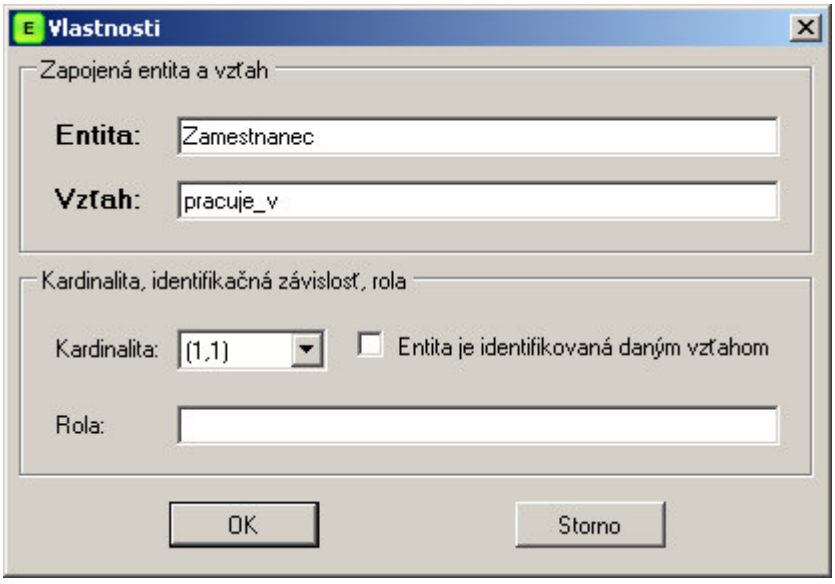
Potvrdenie zmien v editácii vlastnostií vzťahu sa opäť prevedie tlačítkom *OK*. Tlačítko *Storno* neuloží prevedené zmeny a vlastnosti vzťahu sú rovnaké ako pred vyvolaním dialógu.

5.3.4. Editácia a vlastnosti čiar

Voľbou *Vlastnosti* z kontextového menu čiary alebo dvojklikom na čiaru sa vyvolá modálne dialógové okno s vlastnosťami danej čiary (Obr. 5.8).

Pre čiaru sa nastavuje rola a typ kardinality z týchto možností: *(0,n), (0,1), (1,n), (1,1).* V prípade, že uživateľ zvolí kardinalitu (1,1) a čiara je prepojená s entitou aj vzťahom, automaticky sa objaví zaškrtávacie tlačítko, ktorým sa nastavuje identifikačná závislosť entity na danom vzťahu. Kvôli lepšiemu prehľadu dialóg tiež zobrazuje názvy entity a vzťahu, ktoré spája.

Zmeny v editácii vlastnosti čiary sa uložia stlačením tlačítka *OK*. Tlačítko *Storno* ignoruje prevedené zmeny a čiare zostajú zachované vlastnosti, ktoré mala nastavené pred vyvolaním dialógu.



Obr. 5.8 Okno pre nastavenie vlastnosti čiary

Podľa typu nastavenej kardinality sa čiara vykresľuje buď plne alebo prerušovane. Plná čiara sa zobrazuje, ak kardinalita vymedzuje povinnú účasť entity vo vzťahu, teda keď kardinalita je (1,1) prípadne (1,n). Čiara sa vykresľuje prerušovane, keď kardinalita defiuje nepovinnú účasť entity vo vzťahu, teda keď kardinalita je (0,1) alebo (0,n). Typ nastavenej kardinality sa zobrazuje približne v strede čiary, vždy tak, aby pri zmene polohy čiara nepretínala text. Ak je nastavená aj rola, zobrazuje sa text role spolu s kardinalitou.

5.3.5. Kontrola korektnosti diagramu

Voľbou z hlavného menu *Diagram – Skontrolovať diagram*, prípadne klávesou skratkou Ctrl+K, či tlačítkom *Skontrolovať diagram* z panela nástrojov sa vykoná kontrola korektnosti diagramu a zobrazí sa informácia o úspešnosti či neúspešnosti kontroly. V prípade neúspešnej kontroly sa zobrazí dialógove okno s varovaním o chybách. Toto okno slúži ako varovné hlásenie a až po jeho zatvorení (potvrdením tlačítkom *OK*) sa zobrazí vysúvacie okno s výsledkom kontroly, v ktorom je uvedený zoznam chýb v modele.

V rámci kontroly korektnosti ER diagramu sa overuje:

- Jedinečnosť názvov entít a vzťahov v rámci diagramu
- Jedinečnosť názvov atribútov v rámci entity a vzťahu
- Výskyt nezapojených čiar v diagrame
- Existencia cyklu identifikačných závislostí
- Existencia cyklu identifikačných závislostí po prevode ISA vzťahov na identifikačné
- Existencia identifikačného kľúča pre zdroje ISA hierarchie
- Povinnosť neexistencie identifikačného kľúča u entít, ktoré nie sú zdrojom ISA hierarchie
- Existencia rolí pre entity vstupujúce do vzťahu viackrát

5.3.6. Uloženie diagramu

Voľbou z hlavného menu *Súbor – Uložiť ako* sa vyvolá štandardné dialógove okno pre uloženie diagramu. Následne je možné vybrať cestu, kam sa má súbor uložiť. Každý ER diagram sa uloží s príponou .erm. Neodporúča sa meniť príponu súboru po uložení, keďže editor dovoľuje pre ER diagram otvorenie súboru len s príponou .erm.

Voľbou z hlavného menu *Súbor – Uložiť*, prípadne klávesovou skratkou Ctrl+S či pomocou tlačítka *Uložiť* z panela nástrojov sa vykoná uloženie posledných zmien v diagrame. V prípade, že daný diagram od svojho vytvorenia ešte nebol uložený, má tento požiadavok rovnaký účinok ako voľba *Uložiť ako* z hlavného menu *Súbor* (viď vyššie).

5.3.7. Otvorenie diagramu

Voľbou z hlavného menu *Súbor – Otvoriť*, prípadne klávesovou skratkou Ctrl+O či stlačením tlačítka *Otvoriť* z panela nástrojov sa zobrazí štandardný dialóg pre otvorenie súboru. V prípade, že bol aktívny ER diagram prázdny, neuložený a neprebehla na ňom doposiaľ žiadna zmena, bude otváraný súbor zobrazený v tomto diagrame. Ak je aktívny diagram už uložený, prípadne nový a prebehla na ňom nejaká zmena (napríklad pridanie objektu), bude otváraný diagram otvorený v novom MDI okne. Ak otváraný súbor už existuje v zozname všetkých otvorených diagramov, nebude sa otvárať, ale vyhľadá sa v zozname a nastaví sa ako aktívny.

Chovanie pri otváraní a ukladaní diagramu do súboru je veľmi podobné ako napríklad u programu Microsoft Word. Tiež sa jedná o MDI aplikáciu, ktorá dovoľuje pracovať v jednom hlavnom okne s viacerými dokumentami.

5.4. Prevod z ER modelu do RMD

Voľbou z hlavného menu *Diagram – Previesť ER do RMD*, prípadne klávesou skratkou Ctrl+T, či tlačítkom *Previesť ER do RMD* z panela nástrojov je vyvolaná žiadosť o prevod ER diagramu do RMD. Pred samotným prevodom je automatický vyvolaná kontrola korektnosti ER modelu. V prípade, že kontrola prebehla neúspešne, je zobrazená varovná hláška a uvedený zoznam chýb, ako to popisuje kapitola 5.3.5. Táto kontrola síce nebola požadovaná uživateľom, ale pre bezproblémový prevod diagramu je potrebná. V prípade, že kontrola dopadla úspešne, nie je zobrazená informácia o úspešnosti kontroly, ale automaticky sa vytvorí nový RMD diagram, do ktorého je prevedený zdrojový ER diagram. ER diagram pritom zostáva naďalej otvorený. Ako aktívny diagram je nastavený vytvorený RMD diagram.

5.5. Editor RMD diagramov

Pri spustení aplikácie sa defaultne nastaví ako nový diagram ER diagram (viď kapitola 5.3). To vychádza so zamerania programu pre návrh relačnej databázy od najnižšej úrovne, konceptuálneho modelu. Nové okno pre vytváranie RMD modelov je možné aktivovať voľbou z menu *Súbor – Nový – RMD model* alebo klávesovou skratkou Ctrl+R. Nový diagram je možné zaviesť taktiež stlačením tlačítka *Nový RMD model* z nástrojovej lišty. Ide však o tzv. “rozdelené tlačítko” (angl. split-button), ktoré zapuzduruje tlačítko pre nový ER model i RMD model. Kliknutím na drobnú šípku, ktorú tlačitko obsahuje sa spustí roleta s možnosťou výberu typu nového modelu.

Po zavedení nového RMD diagramu sa aktualizuje panel prvkov, s ktorými je možné pri návrhu diagramu pracovať a aktualizuje sa tiež slovník objektov. Hlavné okno aplikácie, panel menu, panel nástrojov, okno pre výsledky kontroly diagramu a stavový riadok zostavajú bez zmeny, ako boli popísané v úvode kapitoly 5.3.

Panel *RMD prvky* obsahuje:

- Tabuľku
- Lomenú a priamu čiaru pre neidentifikačnú reláciu (viď kapitola 5.5.2)
- Lomenú a priamu čiaru pre identifikačnú reláciu (viď kapitola 5.5.3)

Čiary pre neidentifikačnú a identifikačnu reláciu budeme tiež súhrne nazývať relačné čiary alebo relačné šípky, pokiaľ nebude potrebné rozlišovať ich konkrétny typ.

Pridávanie tabuliek a relačných čiar do diagramu je možné vykonať spôsobom drag-and-drop prenesením z bočného panela prvkov. Objekt sa vykreslí na tom mieste diagramu, na ktorom uživateľ uvoľní tlačítko myši. Objekty uvoľňené mimo oblasť pracovnej plochy diagramu sa nevykreslia vôbec. Pojmom objekt, v kontexte tejto kapitoly, budeme uvažovať tabuľku a ktorúkoľvek z relačných čiar z panela RMD prvkov. Každý z týchto objektov sa v diagrame môže vyskytovať v aktívnom alebo pasívnom stave, čím sa chápe označenosť alebo neoznačenosť objektu. Klik myšou do viditeľnej časti objektu ho nastaví ako aktívny. Klik myšou mimo ktorýkoľvek objekt diagramu zruší aktívny objekt. Pre jednotlivé objekty je možné vyvolať kontextové menu, ktoré obsahuje voľby pre kopírovanie, vystrihnutie a zmazanie objektu. Tabuľka má kontextové menu rozšírené o ponuku *Vlastnosti.* Voľba tejto položky z kontextového menu tabuľky umožňuje uživateľovi v dialógovom okne nastaviť jej špecifické vlastnosti ako názov a atribúty. Dialóg s vlastnosťami tabuĺky je tiež možné vyvolať dvojklikom na tabuľku. Nastavenie vlastností tabuľky podrobnejšie popisuje kapitola 5.5.4.

5.5.1. Presun a prepájanie objektov

Každý objekt RMD diagramu je možné vrámci diagramu ďalej presúvať metódou drag-and-drop. Dôležitá je však oblasť uchopenia objektu. V prípade aktívnej tabuľky sa viditeľná časť objektu zvýrazni zmenou farby. Veľkosť tabuľky nie je možné ručne meniť, ale nastavuje sa automaticky v závislosti na dĺžke názvu tabuľky, počtu pridávaných atribútov, referencií a ich dĺžke.

V prípade aktívnej relačnej šipky sa na oboch koncoch objektu vykreslí štvorec. Takto vykreslený objekt indikuje, že je aktívny. Objekt je možné presúvať vrámci diagramu jeho uchopením vo vykreslenej oblasti medzi koncovými štvorcami a následným presunom. Uchopenie objektu za jeden z koncových štvorcov spôsobí ukotvenie štvorca na protiľahlom konci a následnym pohybom kurzora je možné meniť dĺžku šipky, či šipku zapojiť.

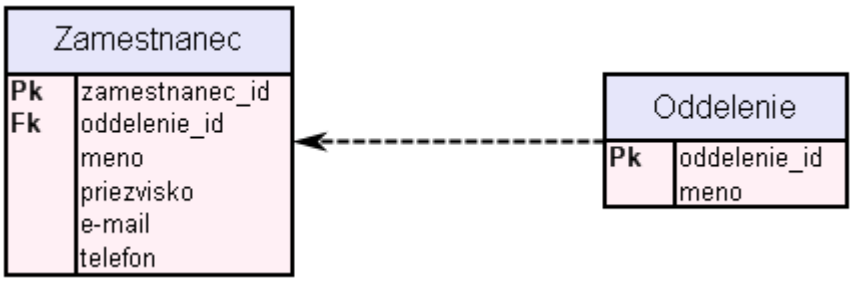
Relačné šipky je možné s tabuľkami vzájomne prepájať. Šipka sa uchopí za koncový štvorec, presunie sa nad viditeľnú oblasť tabuľky, s ktorou spojenie požadujeme. V prípade, že ide o dovolené spojenie, obvod tabuľky sa vykreslí červenou farbou a po uvoľňení tlačítka myši sa šipka spojí s tabuľkou. Aby sa zabránilo vytváraniu nezmyselných návrhov a tiež kvôli uľahčeniu neskoršej kontroly diagramu, prepájanie tabuliek je čiastočne obmedzené a kontrolované. Editor umožňuje prepojiť dve tabuľky

nanajvýš jednou relačnou šipkou. Jedným z možných rozšírení programu ER[G]edit by mohlo byť zrušenie tohto obmedzenia.

Odpájanie šipiek od tabuliek sa prevádza podobným spôsobom ako zapájanie. Najprv sa nastaví aktívna čiara a uchopením jej koncového bodu a odtiahnutím z vnútornej oblasti tabuľky a následným uvoľnením kurzora sa šipka odpojí. Pri odpájaní sa odpájaná tabuľka vykreslí modrou farbou.

5.5.2. Neidentifikačná relácia

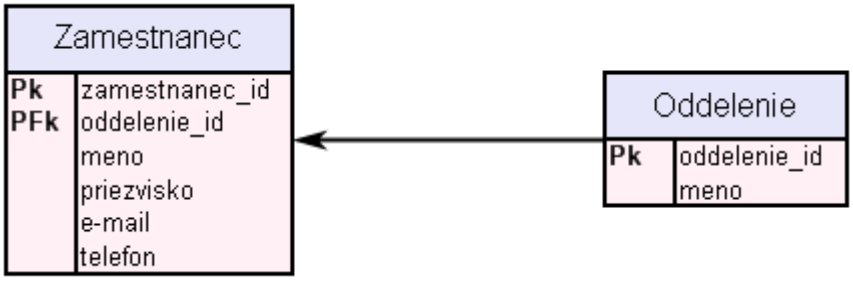
Pri prepojení dvoch tabuliek neidentifikačnou reláciou (neidentifikačnou relačnou šipkou) sa primárný kľúč (môže ho tvoriť aj viac atribútov) rodičovskej tabuľky automaticky stáva cudzím kľúčom dcérskej tabuľky. Majme napríklad tabuľku zamestnancov (*Zamestnanec*). Každý zamestnanec podniku má pridelené identifikačné číslo, ktoré ho jednoznačne identifikuje. V našom modele bude tento identifikátor zamestnanca slúžiť ako primárny kľúč. Majme tiež tabuľku oddelení (*Oddelenie*). Každé oddelenie má jednoznačný identifikátor, ktorý bude slúžiť ako primárny kľúč. Zrejme, každý zamestnanec bude pracovať práve v jednom oddelení. V návrhu RMD modelu relačnej databázy to znamená, že tabuľka zamestnancov obsahuje cudzí kľúč do tabuľky oddelení (Obr. 5.9).



Obr. 5.9 Ukážka použitia neidentifikačnej relácie

5.5.3. Identifikačná relácia

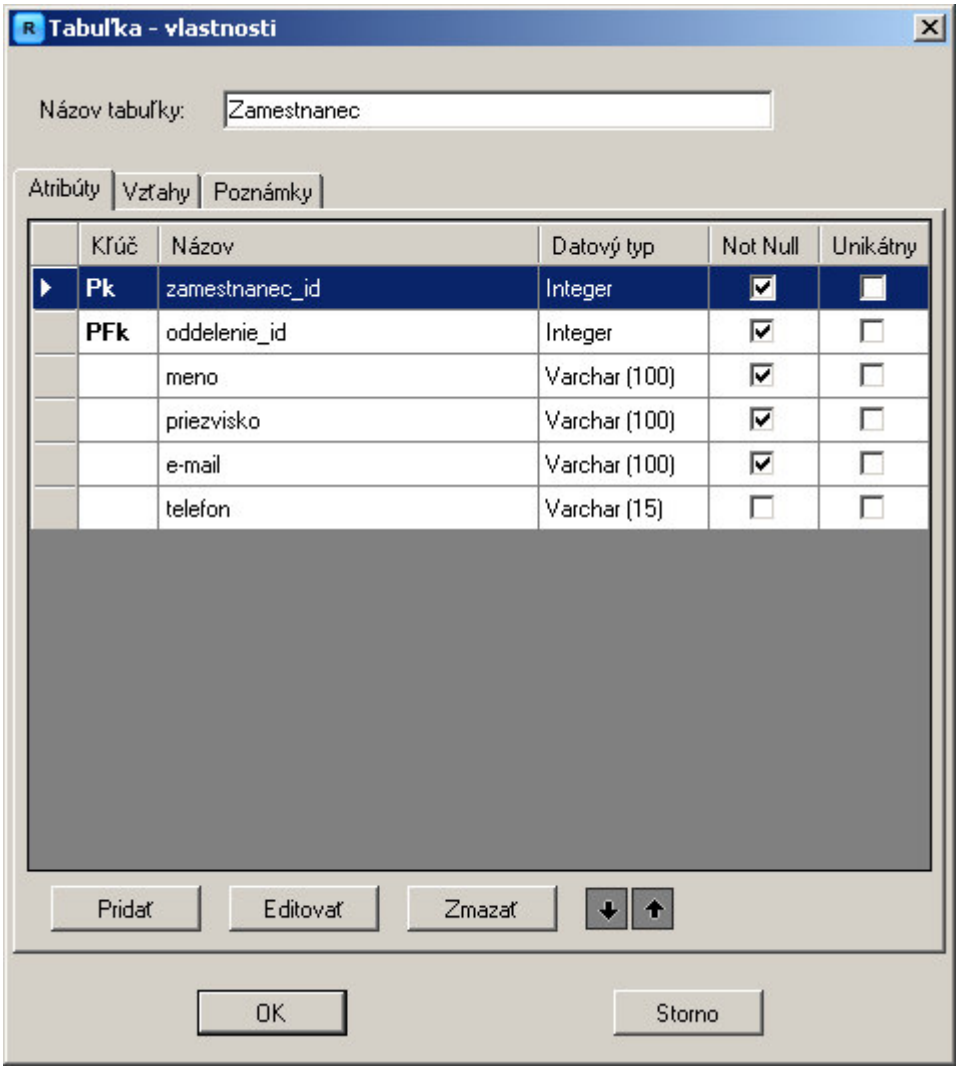
Pri prepojení dvoch tabuliek identifikačnou reláciou (šipkou) sa primárný kľúč (môže ho tvoriť aj viac atribútov) rodičovskej tabuľky automaticky stáva cudzím kľúčom dcérskej tabuľky a zároveň aj súčasťou jej primárneho kľúča. Uvažujme príklad z kapitoly 5.5.2 s jedným rozdielom. A to, že každý zamestnanec ma pridelené identifikačné číslo, ktoré ho však jednoznačne identifikuje len vrámci oddelenia. To znamená, že identifikačné číslo zamestnanca nie je vhodným kandidátom na primárny kľúč. Spojenie identifikačného čísla zamestnanca s identifikačným číslom oddelenia vytvára vhodného kandidáta na primárny kľúč. V prevedení programom ER[G]edit to znamená neidentifikačnú relačnú šipku zameniť šipkou identifikačnou (Obr. 5.10).



Obr. 5.10 Ukážka použitia identifikačnej relácie

5.5.4. Editácia a vlastnosti tabuľky

Voľbou *Vlastnosti* z kontextového menu tabuľky alebo dvojklikom na tabuľku sa vyvolá modálne dialógové okno s vlastnosťami danej tabuľky.



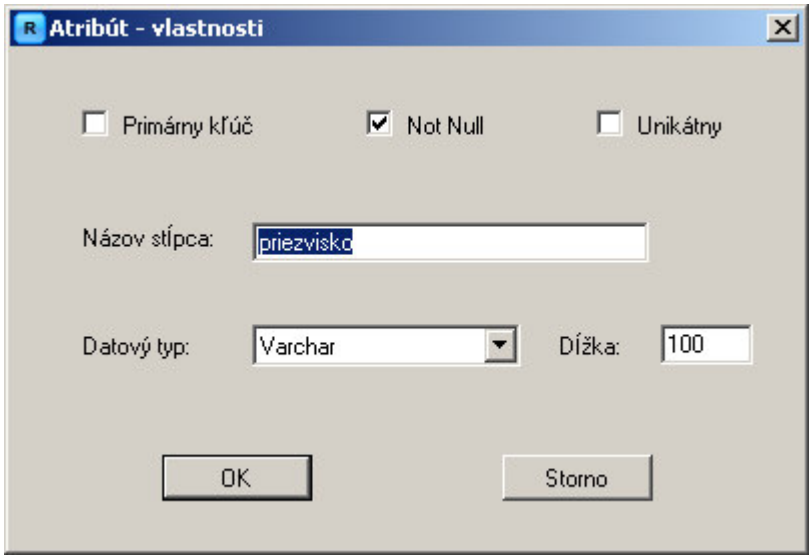
Obr. 5.11 Okno pre nastavenie vlastnosti tabuľky

Pre tabuľku je možné nastaviť názov tabuľky. Na záložke *Atribúty* (viď Obr. 5.11) je možné pridávať atribúty (tlačítko *Pridať*), editovať atribúty (označiť riadok tabuľky a tlačítko *Editovať* alebo dvojklik na riadok) a mazať atribúty tabuľky (označiť riadok a

tlačítko *Zmazať*). Formulár však nedovolí zmazať referenčný stĺpec. Teda taký, ktorý bol do tabuľky automaticky pridaný po prepojení dvoch tabuliek. Odstrániť referenčný stĺpec je možné buď zrušením referencie (teda odpojením, či zmazaním relačnej šipky medzi prepojovanými tabuľkami) alebo odstránením tohto stĺpca z rodičovskej tabuľky, prípadne zrušením indikácie, že sa jedna o kľúčový atribút v rodičovskej tabuľke.

Pojmom rodičovská tabuľka danej tabuľky budeme rozumieť tabuľku, ktorá po zapojení relačnej čiary deleguje svoje kľúčove atribúty do tejto tabuľky. Tú označujeme pojmom dcérska tabuľka. Logika zapajánia relačných šipiek odpovedá tejto zavedenej terminológii. A teda, šipka smeruje od rodičovskej tabuľky k dcérskej. To je síce opačný smer k vytvorenej referenčnej integrite, kedy dcérska tabuľka obsahuje cudzí kľúč do tabuľky rodičovskej, no pri vytvárani ER[G]editu bol uvážený ako názornejší, preto sme sa priklonili k nemu.

Pri pridávaní alebo editácii atribútu sa modálne otvorí dialógove okno, ktoré umožňuje nastaviť vlastnosti atribútu tabuľky.



Obr. 5.12 Okno pre nastavenie vlastnosti atribútu

Uživateľ má možnosť nastaviť nasledujúce vlastnosti:

- *Názov stĺpca*
- *Datový typ* – uživateľ ma možnosť z roletového (drop-down) zoznamu zvoliť jeden z ponúkaných datových typov – *integer, float, date, char, varchar*. Pri voľbe datového typu *varchar* sa automatický zobrazí textové pole pre zadanie veľkosti varchar-u. V prípade, že uživateľ neuvedie dĺžku pre tento datový typ, bude defaultne nastavená dĺžka 255 znakov. V prípade, že uživateľ do poľa vloží nečíselnú hodnotu, prípadne čislo mimo interval 0 – 255, bude nastavená defaultna dĺžka 255 znakov.
- *Primárny kľúč* – pri zaškrtnutí tejto voľby bude atribút súčasťou primárneho kľúča. Taktiež sa automaticky nastaví vlastnosť *Not Null* (viď ďalší bod), ktorú nie je možné zmeniť po dobu, kým je nastavená voľba *Primárny kľúč.*

- *Not Null* – pri zaškrtutí tejto voľby bude vyžadovaná povinnosť tohto atribútu. V reči databáz to znamená nastavenie obmedzenia, ktoré nedovolí vložiť do databázovej tabuľky záznam s absenciou daného atribútu.
- *Unique* – zaškrtnutá voľba indikuje jednoznačnosť daného atribútu. V reči databáz to znamená nastavenie obmedzenia, ktoré nedovolí vložiť do databázovej tabuľky záznam, ktorého hodnota daného atribútu už v tabuľke existuje.

Tlačítkom *OK* sa prevedie uloženie vlastností atribútu, okno sa zatvorí a aktívnym oknom bude opäť dialóg vlastností tabuľky. Pred uložením sa však kontroluje, či je vyplnený názov stĺpca. Stlačením tlačítka *Storno*, sa neprevedie pridanie či editácia atribútu a aktivným oknom bude opäť dialóg pre editáciu vlastností tabuľky. Ku každej tabuľke je možne na záložke *Poznámky* nastaviť textový popis, či akúkoľvek poznámku, ktorá môže byť dosť nápomocná pri rozsiahlejších databázových návrhoch. Záložka *Vzťahy* obsahuje prehľad zapojených rodičovských a dcérskych tabuliek.

Formulár editácie vlastnosti tabuľky obsahuje tiež dve tlačítka s obrázkami šipiek smerujúcimi u jedného z nich smerom nahor, u druhého nadol. Označením niektorého atribútu a použitím tlačítok dochádza k zmene poradia atribútov v tabuľke. Takto nastavené poradie atribútov sa po potvrdení zmien premietne jednak pri vykresľovaní tabuľky v diagrame a rovnako aj pri generovaní SQL skriptu.

Uloženie vlastnosti sa prevedie stlačením tlačítka *OK*. Pred uložením sa vykoná kontrola na neprázdnosť názvu tabuľky. Zrušenie prevádzaných zmien sa prevedie stlačením tlačítka *Storno.*

5.5.5. Kontrola korektnosti diagramu

Podobne ako v prípade editoru ER diagramov (viď kapitola 5.3.5), aj editor RMD schémat poskytuje kontrolu korektnosti vytvoreného schématu. Tú je možné vyvolať voľbou z hlavného menu *Diagram – Skontrolovať diagram*, prípadne klávesou skratkou Ctrl+K, či tlačítkom *Skontrolovať diagram* z panela nástrojov. V prípade, že diagram neobsahuje žiadne chyby, je zobrazená informácia o úspešnej kontrole diagramu. V prípade neúspešnej kontroly sa zobrazí dialógove okno s varovaním o neúspešnosti kontroly a výčtom nájdených chýb v diagrame.

V rámci kontroly korektnosti RMD diagramu sa kontroluje:

- Existencia cyklu v zapojení tabuliek
- Jednoznačnosť názvov tabuliek
- Jednoznačnosť názvov atribútov v rámci tabuľky
- Existencia primárneho kľuča tabuľky
- Existencia nezapojených relačných čiar v diagrame

5.5.6. Uloženie diagramu

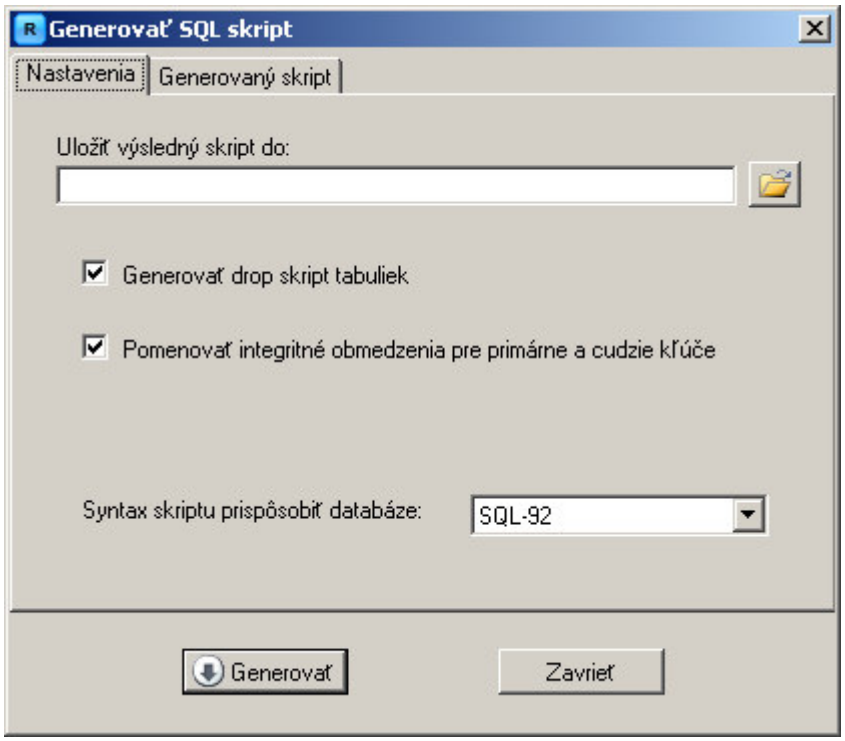
Uloženie diagramu prebieha na rovnakom princípe ako tomu bolo v prípade ukladanaie ER schémy (viď kapitola 5.3.6). Diagramy sú uložené do súborov s príponou .rmd.

5.5.7. Otvorenie diagramu

Otváranie súborov s RMD diagramami je založené na rovnakom princípe ako v prípade ER modelov (viď kapitola 5.3.7).

5.6. Generovanie SQL skriptov z RMD modelu

Z vytvoreného RMD modelu je možné voľbou z hlavného menu *Diagram – Generovať SQL skript*, prípadne tlačítkom z nástrojovej lišty, prípadne klávesovou skratkou Ctrl+G vyvolať dialógové okno pre generovanie SQL skriptu pre databázu.



Obr. 5.13 Okno pre nastavenia vlastností generovaného SQL skriptu

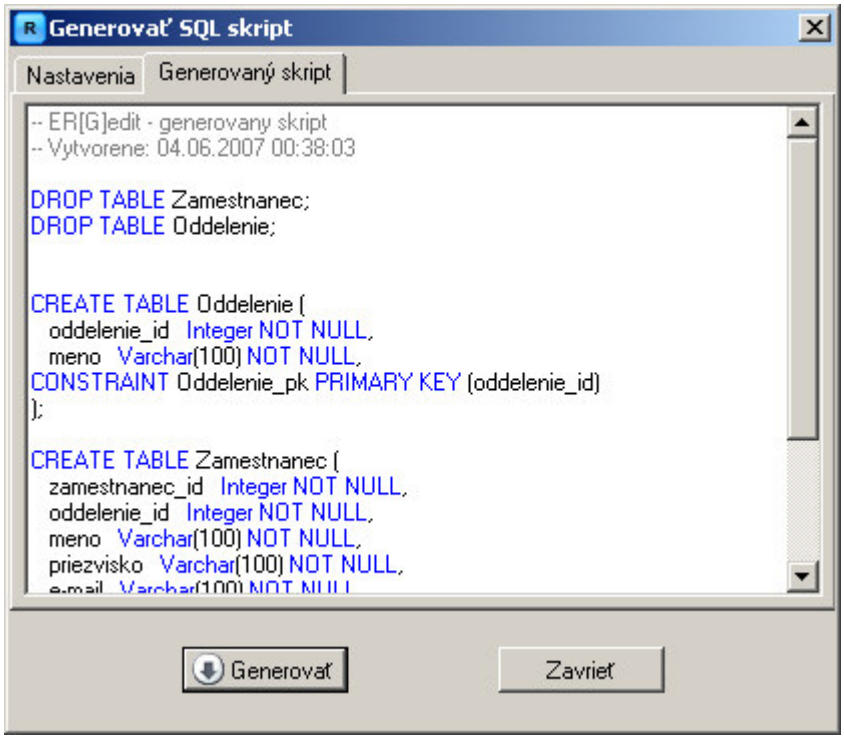
Pred samotným otvorením okna sa spustí kontrola korektnosti diagramu (viď kapitola 5.5.5) a v prípade, že kontrola prebehla neúspešne, je zobrazená varovná hláška a uvedený zoznam chýb, ako to popisuje kapitola 5.5.5. Táto kontrola síce nebola požadovaná uživateľom, ale pre bezproblémové vygenerovanie skriptu je potrebná. V prípade, že kontrola dopadla úspešne, nie je zobrazená informácia o úspešnosti kontroly, ale automaticky sa otvorí modálne dialógové okno pre generovanie skriptu (Obr. 5.13).

Na záložke *Nastavenia* je možné nastaviť cestu k súboru, do ktorého sa vygenerovaný skript uloží. Taktiež sa tu nastavuje, či sa má generovať skript pre premazanie vytváraných tabuliek a či sa budú pomenovávať integritné obmedzenia pre primárne a cudie kľúče. Pre generovaný skript sa voľbou z roletového zoznamu zvolí syntax výsledného skriptu. Tento zoznam ponúka na výber z možností:

- SQL-92
- MySQL
- MSSQL

- Oracle
- PostgreSQL 8.1 a nižšie
- PostgreSQL 8.2 a vyššie

Stlačením tlačítka generovať sa vygeneruje skript so zohľadnením zvolenej syntaxe a aktivuje sa záložka *Generovaný skript,* ktorá ho obsahuje (Obr. 5.14).



Obr. 5.14 Ukážka okna s vygenerovaným SQL skriptom

Ak bola zaškrtnutá voľba pre uloženie skriptu do súboru, skript sa uloží a uživateľ o tom nie je informovaný. Pri chybe s uložením skriptu do súboru je uživateľ informovaný varovnou hláškou o chybe. Tlačítkom *Zavrieť* sa zatvorí dialógové okno.

5.7. Ukážka pred tlačou a tlač diagramu

Program ER[G]edit ponúka možnosť tlače vytvorených diagramov. Voľbou menu *Súbor – Nastavenie stránky* sa vyvolá štandardný dialóg pre nastavenie formátu stránky, jej okrajov a horizontálnej či vertikálnej orientácie. Voľba menu *Súbor – Ukážka pred tlačou,* prípadne tlačítko nástrojovej lišty vyvolá štandardný dialóg náhľadu pred tlačou. Samotnú tlač diagramu je možné vyvolať z dialógu pre náhľad, prípadne voľbou z menu *Súbor – Tlačiť,* prípadne tlačítkom z panela nástrojov alebo klávesovou skratkou Ctrl+P.

5.8. Export diagramu

Vytvorené modely je možné exportovať do niekoľkých obrázkových formátov. Export sa vyvolá voľbou z menu *Súbor – Exportovať*, kedy sa otvorí štandardný dialóg pre uloženie súboru. Výber typu súboru je filtrovaný na tieto obrázkové formáty:

- PNG
- BMP

- JPEG
- GIF
- TIFF

5.9. Menu aplikácie a panel nástrojov

Hlavné okno aplikácie obsahuje v hornej časti menu a panel nastrojov (toolbar). Jednotlivé voľby pokrývajú hlavnú funkčnosť editoru.

5.9.1. Menu aplikácie

V ďalšom texte je uvedený popis jednotlivých položiek a subpoložiek hlavného menu.

Ponuka menu *Súbor:*

- *Nový – ER model* – vytvorí nový dokument pre tvorbu ER diagramu.
- *Nový – RMD model* – vytvorí nový dokument pre tvoru RMD diagramu.
- *Otvoriť* – otvorí dialógové okno pre otvorenie súboru.
- *Uložiť* – uloží zmeny v aktívnom dokumente.
- *Uložiť ako* – otvorí dialógové okno pre výber cesty k súboru, do ktorého sa aktívny diagram požaduje uložiť.
- *Uložiť všetky* – pre všetky otvorené diagramy prevedie operáciu uloženia súboru.
- *Exportovať* – vyvolá dialóg pre export diagramu do obrázkového formátu.
- *Nastavenie stránky* – otvorí dialógové okno pre nastavenie stránky.
- *Ukážka pred tlačou* – otvorí okno s náhľadom diagramu pred tlačou.
- *Tlačiť* – vyvolá dialóg pre tlač diagramu.
- *Koniec* – ukončí prácu s programom. Pre všetky neuložené či zmené diagramy bude zobrazená výzva, či si uživateľ praje uložiť zmeny.

Ponuka menu *Úpravy:*

- *Kopírovať* – uloží označený objekt aktívneho diagramu do schránky Windows.
- *Vystrihnúť* – odstráni označený objekt z aktívneho diagramu a uloží do schránky Windows.
- *Vložiť* – vyberie objekt zo schránky (ak taký objekt existuje) a vloží ho do aktívneho diagramu. Pred vložením sa kontroluje správnosť daného objektu pre vloženie do diagramu. Aplikácia umožňuje kopírovanie (vystrihnutie) a vkladanie objektov medzi všetkými otvorenými oknami diagramu, no vždy len v rámci rovnakého typu modelu. To znamená, že nie je dovolené nakopírovať objekt ER diagramu (napríklad entitu) a vložiť ju do RMD diagramu. Pokus o vloženie teda nevykoná žiadnu akciu. Objekt však zostáva naďalej uložený v schránke Windows.

Ponuka menu *Zobraziť:*

- *Zväčšiť* – zväčšuje aktívny diagram.
- *Zmenšiť* – zmenšuje aktívny diagram.
- *Panel nástrojov* – pri zaškrtnutej voľbe hlavné okno zobrazuje panel nástrojov. Defaultne je táto voľba zaškrtnutá.
- *Kontrola diagramu* – pri zaškrtnutej voľbe obsahuje hlavné okno aplikácie tiež formulár pre zobrazenie výsledkov kontroly. V prípade, že je táto voľba

neoznačená, po vyvolaní žiadosti o kontrolu a pri nájdení chýb v diagrame, nie sú tieto chyby zobrazené. Defaultne je táto voľba zaškrtnutá.
- *Mriežka* - pri zaškrtnutej voľbe sú všetky otvorené diagramy zobrazované s mriežkou na pozadí. Defaultne je táto voľba nezaškrtnutá.

Ponuka menu *Diagram:*

- *Skontrolovať diagram* – prevedie kontrolu korektnosti diagramu aktívneho dokumentu a o výsledku kontorly informuje dialógovým oknom nesúcim danú informáciu a obsahujúcim tlačítko pre uzavretie dialógu.
- *Previesť ER do RMD* – prevádza transformáciu ER diagramu do nového RMD diagramu, ktorý za týmto účelom vytvorí. Táto voľba nie je dostupná, ak je aktívny dokument s RMD diagramom.
- *Generovať SQL skript* – otvorí dialógove okno pre prevod RMD diagramu do SQL skriptu. Táto voľba však nie je dostupná, ak je aktívny dokument s ER diagramom.
- *Zarovnať k mriežke* – pre všetky nečiarové objekty diagramu, zarovná ich stred k mriežke. Poloha čiarových objektov sa aktualizuje so zmenou polohy nečiarových objektov, s ktorými sú prepojené.

Ponuka menu *Okno:*

- *Pod sebou* – horizontálne rozloží všetky otvorené dokumenty v pracovnej ploche programu.
- *Vedľa seba* – vertikálne rozloží všetky otvorené dokumenty v pracovnej ploche programu.
- *Kaskáda* – usporiada otvorené dokumenty do kaskády.
- *Minimalizovať všetky* – minimalizuje všetky otvorené dokumenty do spodnej časti pracovnej plochy hlavného okna aplikácie.
- *Obnoviť všetky* – voľba inverzná k voľbe *Minimalizovať všetky.* V prípade, že sú otvorené dokumenty minimalizované, rozložia sa na pracovnej ploche do pozícii, v ktorých sa nachádzali pred minimalizovaním.

V prípade, že je otvorený aspoň jeden dokument, pokračuje ďalej ponuka menu *Okno* separátorom a výčtom otvorených okien (ten pozostáva z názvov otvorených súborov). Aktívny dokument je označený ikonkou vedľa názvu a klikom na niektorú z položiek ponúkaného výčtu sa nastaví nový aktívny dokument.

Ponuka menu *Pomocník:*

- *O programe* – zobrazí okno s informáciou o názve, verzii a autorovi aplikácie

5.9.2. Panel nástrojov (toolbar)

Panel nástrojov sa náchazda v hornej časti aplikácie, pod hlavným menu a slúži k rýchlemu prístupu k hlavným položkám menu. Pozostavá z rovnakých ikon, aké obsahujú odpovedajúce položky menu a po najetí kurzorom myši nad niektorú z položiek lišty sa zobrazí popisok (*tooltip*) s rovnakým názvom aký nesie odpovedajúca položka menu.

Obr. 5.15 Panel nástrojov

Na obrázku 5.15 jednotlivé číselné šipky zobrazujú na paneli nástrojov tieto tlačítka:

1. Nový ER model (kliknutím na roletový zoznam sa zobrazí tiež možnosť nového RMD modelu)
2. Otvoriť
3. Uložiť
4. Uložiť všetky
5. Ukážka pred tlačou
6. Tlačiť
7. Vystrihnúť, kopírovať, vložiť
8. Zmenšiť, zväčšiť
9. Zobraziť mriežku, zarovnať k mriežke
10. Skontrolovať model
11. Previesť ER do RMD
12. Generovať SQL skript

6. Programátorská dokumentácia

6.1. Vývojové prostredie, platforma, programovací jazyk

Program ER[G]edit bol vytvorený pre operačný system Windows 98 a vyšší vo vývojovom prostredí Visual Studio 2005 na .NET Framework 2.0 platforme, ktorá má bohatú podporu pre formulárove aplikácie a prácu s grafickým prostredím. Vďaka týmto výhodam nebolo potrebné použitie žiadných externých knižnic pre prácu s grafickým rozhraniím. Práca s formulármi Windows však bola podporená použitím externej knižnice *MagicLibrary.dll* [16], ktorá je vo verzii 1.7.4 voľne stiahnuteľna z internetu. Kvôli zvolenej platforme vyžaduje aplikácia pre svoj beh nainštalovaný .NET Framework 2.0. Ako programovací jazyk bol zvolený C#, ktorý je vytvorený špeciálne pre túto platformu.

6.2. Grafické rozhranie, MDI aplikácia

Hlavné okno aplikácie tvorí formulár, ktorý je nastavený ako MDI (angl. multiple document interface) kontajner a teda dovoľuje otvorenie viacerých dokumentov naraz. Dokumenty sú však súčasťou hlavného okna a preto práca s nimi je dovolená len v rámci hlavného okna. Editor funguje v dvoch módoch, a to ako editor ER modelov a RMD modelov, čo znamená tiež dva typy dokumentov, ktoré MDI kontajner zastrešuje. Uživateľsky je od oboch typov dokumentov požadovaná takmer rovnaká funkčnosť, preto je definovaný spoločný interface `IModelForm` pre oba modely, ktorý deklaruje metódy pre hlavné funkcie z volieb menu. Oba formuláre pre modely tieto metódy implementujú. Rovnako implementujú aj rozhranie vytvorené `ICopyable` deklarajúce metódy pre kopírovanie, vystrihnutie a vloženie objektov.

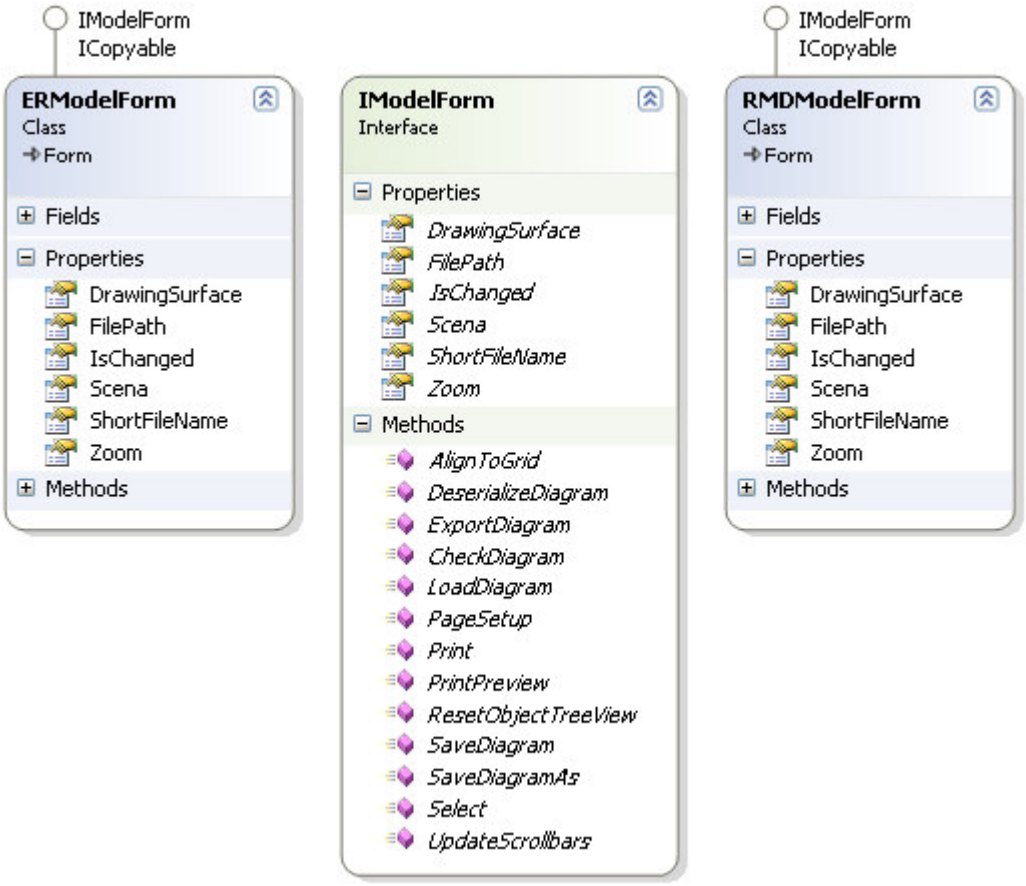
6.3. Formuláre ER a RMD modelu

MDI kontajner umožňuje prácu s dvoma druhmi dokumentov, jeden pre ER diagram a jeden pre RMD diagram. Oba dokumenty su formuláre odvodené zo základnej triedy `System.Windows.Forms` implementujúce tieto metódy interface-u `IModelForm` (Obr. 6.1):

- `AlignToGrid` – zarovnáva objekty scény k mriežke
- `DeserializeDiagram` – prevádza deserializáciu objektov z XML
- `ExportDiagram` – prevádza export diagramu do obrázkových formátov
- `CheckDiagram` – prevádza kontrolu korektnosti diagramu
- `LoadDiagram` – načítava diagram zo súboru
- `PageSetup` – zobrazuje dialóg pre nastavenie stránky a spracováva zadané hodnoty
- `Print` – prevádza tlač diagramu
- `PrintPreview` – zobrazuje ukážku pred tlačou a možnosť tlače z náhľadu
- `ResetObjectTreeView` – zabezpečuje prenastavenie objektov v slovníku hlavného okna pri zmene aktívneho modelu (načítanie aktuálnych objektov a ich lexikografické zotriedenie podľa názvu)
- `Save` – ukladá diagram (bez zobrazenia dialógového okna pre uloženie súboru)
- `SaveDiagramAs` – vyvolá dialógové okno pre uloženie súboru a po jeho potvrdení prevádza uloženie súboru (serializáciu objektov scény do XML)

- `Select` – pri označení objektu z panela objektov hlavného okna (slovníka) označí odpovedajúci objekt na pracovnej ploche
- `UpdateScrollbars` – pri presiahnutí objektu za hranice viditeľnej oblasti pracovnej plochy obsluhuje udalosti spojené s pohybom scrollbarov

Interface `IModelForm` zároveň deklaruje „properties“ (vlastnosti) pre nastavenie „zoom-u“ (veľkosti zväčšenia) pracovnej plochy a veľmi dôležitú vlastnosť pre prístup k `DrawingSurface` objektu, na ktorý sa vykresľujú objekty scény. Jedná sa o špeciálny objekt, ktorý implementuje metódy pre dvojitý buffer pri vykresľovaní, aby sa zabránilo zbytočnému preblikávaniu pri presune objektov po scéne.



Obr. 6.1 Ukážka class diagramu ER a RMD modelu

Formuláre implementujú metódy pre zachytávanie a spracovanie udalosti. Pri kliknutí a podržaní ľavého tlačítka myši nad niektorým z objektov z panela prvkov daného diagramu, sa vyvolá operácia drag-and-drop a po prenesení na pracovnú plochu diagramu a následnom uvoľnení tlačítka myši je vyvolaná udalosť ktorú spracuje špeciálna metóda (*DragEventHandler)*, ktorá je volaná vždy, keď sa objaví udalosť tohto typu. Táto metóda zistí, či ide o dovolený typ objektu a pokiaľ tomu tak je, tak zavolá konštruktor pre daný objekt a nastaví mu potrebné argumenty, ako napríklad polohu kliku, či názov objektu (počet a typ argumentov závisí od typu konštruovaného objektu). Kontrola, či ide o dovolený typ objektu sa vykonáva na základe porovnania typu objektu a typu pracovnej plochy. Oba modely (ER i RMD) presne špecifikujú typy objektov, ktoré môžu byť umiestnené na ich pracovnú plochu .

Klik **ľavým tlačítkom** myši je ďalšia udalosť spracovávaná v rámci diagramu. Pri kliku myšou sa prechádza scéna objektov a v prípade, že súradnice kliku sa nachádzajú vo vnútornej oblasti niektorého z objektov, je objekt nastavený ako aktívny. V prípade kliku mimo vnútornu oblasť ktréhokoľvek z objektov je zrušený aktívny objekt. Spracovanie udalosti pre pohyb myšou umožňuje presúvanie objektov, zmenu tvaru čiar, zapájanie a odpájanie čiar a relačných šipiek. Taktiež sú spracovávané udalosti kliku **pravým tlačítkom** myši a v prípade, že sa klik uskutočnil nad niektorým z objektom poskytujúcim kontextové menu, je toto kontextové menu vyvolané a jeho voľby ďalej spracovávané ako ďalšie udalosti.

Formulár pre ER diagram navyše implementuje metódu pre prevod ER modelu do RMD modelu (kapitola 6.8) a formulár pre RMD diagram implementuje metódu pre generovanie SQL skriptu (kapitola 6.9).

K ďalším formulárom patria modálne dialógy pre nastavenie vlastnosti jednotlivých objektov scény, dialóg pre nastavenie vlastností transformácie do SQL skriptov či formulár pre zobrazenie výsledkov kontroly diagramov.

6.4. Scéna ako slovník objektov ER a RMD modelu

U ER i RMD modelu je vytvorená trieda scéna, ktorá slúži ako slovník pre všetky vykresľované objekty. V oboch prípadoch sa jedná o triedu, ktorá implementuje metódy pre vykreľovanie objektov, vykresľovanie aktívnych objektov a získavanie objektov zo slovníka podľa zadaného *GUID* (z angl. **G**lobally **U**nique **Id**entifier). GUID prvku je špeciálny reťazec generovaný triedou .NET Frameworku `System.Guid`, ktorý bol vytvorený za účelom použitia pri potrebe jednoznačnej globálnej identifikácie objektov. Príklad takéhoto GUID-u: `{F9168C5E-CEB2-4faa-B6BF-329BF39FA1E4}`.

Inštancia triedy scéna (existujú dve rôzne pre ER model i RMD model) je vytvorená pri volaní konštruktoru nového formulára (dokumentu) ER či RMD modelu. Práve scéna je tým objektom ktorý sa serializuje pri ukladaní a deserializuje pri načítaní diagramu. Serializáciu a deserializáciu objektov scény prevádza trieda .NET Frameworku `XmlSerializer` z priestoru názvov `System.Xml.Serialization`. Pri jej použití sa však musel odstrániť problém s nekonečnou rekurziou pri serializácii objektov, ktoré sa vzájomne obsahovali ako svoje členské atribúty.

Zrejmejšie to bude na príklade. Objekt entita obsahuje pole čiar, ktoré naň sú napojené. Objekt čiara zase obsahuje objekty entít, ktoré spája. Pri serializácií sa `XmlSerializer` snaží serializovať entitu, ktorá ako jeden zo svojich atribútov obsahuje čiaru, ktorá ako jeden zo svojich atribútov obsahuje opäť túto entiu. Je zrejmé, že sa tým vytvorí nekonečná rekurzia, s ktoru si serializácia nedokáže poradiť. Tento problém bol odstranený práve použitím jednoznačného GUID pre všetky serializované objekty a namiesto serializácie atribútov ako objektov, sa serializuje len tento GUID. Pri deserializácii sa k týmto identifikátorom doplnia odpovedajúce objekty.

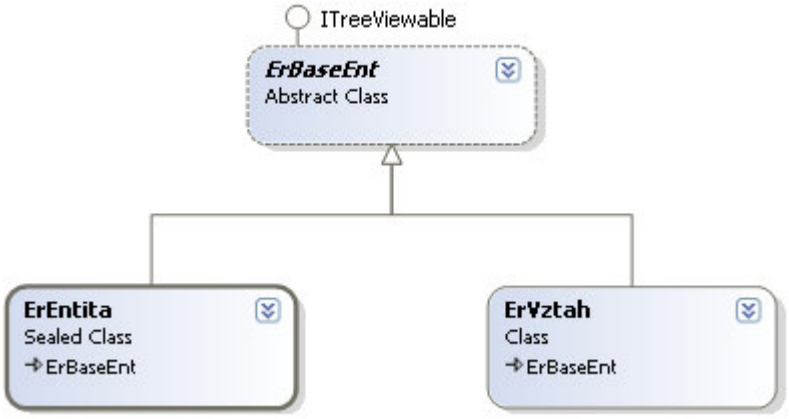
V nasledujúcom príklade časti XML (viď nižšie) je príklad serializácie čiary, ktorá spája entitu a vzťah so svojími špecifickými GUID. Tieto objekty sa serializujú samostatne (kvôli prehľadnosti nie je uvedený celý XML súbor so serializovanými objektami), preto sa u čiary budú serializovať len GUID identifikátory týchto objektov (v ukážke XML to sú XML uzly StartEntGuid a EndEntGuid). Pri načítaní diagramu by v tomto prípade prebehla deserializácia a následne by sa pre čiaru z príkladu vyhľadali

podľa StartEntGuid a EndEntGuid objekty s odpovedajúcimi identifikátormi v poli objektov a pridali by sa k členským atribútom `StartEnt` a `EndEnt` danej čiary.

```
<ErBaseLine xsi:type="ErLine" Guid="f9744511-d00e-451b-837b-ba92881541cf">
   <StartPoint>
      <X>-207.709137</X>
      <Y>-234.644241</Y>
   </StartPoint>
   <EndPoint>
      <X>-275.5</X>
      <Y>-198.38382</Y>
   </EndPoint>
   <StartEntGuid>9ce1015f-00d2-4cdb-8b06-ce0119ea5b15</StartEntGuid>
   <EndEntGuid>9916a30f-7f35-4722-a4d8-e32ae1b51475</EndEntGuid>
   <Kardinalita>OneOne</Kardinalita>
   <IsIdentifying>false</IsIdentifying>
   <Rola>rol1</Rola>
</ErBaseLine>
```

6.5. Grafické objekty ER modelu

Pre editáciu ER diagramov bolo nutné vytvoriť objekty pre entitu, vzťah, spojovacie čiary a generalizačnú šípku (ISA vzťah). Atribúty entít a vzťahov sú vykresľované automaticky po ich pridaní cez vlastností daného objektu.



Obr. 6.2 Ukážka class diagramu entity, vzťahu a ich spoločného abstraktného predka

Spoločným predkom entity a vzťahu je abstraktná trieda `ErBaseEnt`, ktorá vymedzuje atribúty a metódy spoločné pre entitu a vzťah (Obr. 6.2). To umožňuje udržiavať v scéne zoznam (pole) objektov tejto nadtriedy. Špecifikuje atribúty pre:

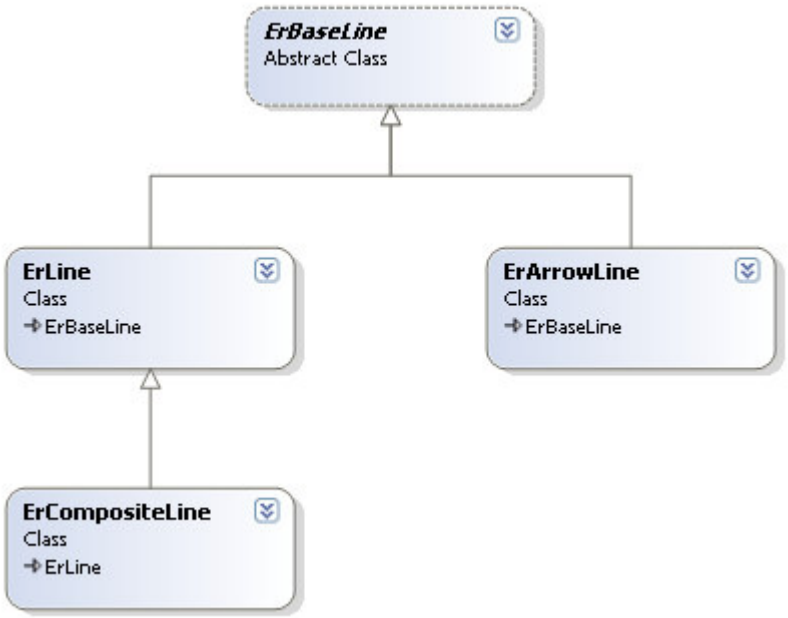
- GUID objektu
- Názov
- Polohu
- Veľkosť
- Zoznam napojených čiar
- Zoznam atribútov

Ďalej vymedzuje virtuálne metódy pre:

- Vykreslenie objektu

- Zisťovanie, či sa daný bod nachádza vo vnútornej oblasti objektu (pre zisťovanie náležitosti kliku myšou do vnútornej oblasti objektu)
- Zobrazenie dialógu s vlastnosťami objektu
- Aktualizáciu polohy zapojených čiar pri zmene polohy entity, či vzťahu

Spoločným predkom čiary, lomenej čiary a generalizačnej šipky je abstraktná trieda `ErBaseLine`, ktorá vymedzuje ich spoločné atribúty a virtuálne metódy. To umožňuje udržiavať v scéne zoznam objektov tejto nadtriedy.



Obr. 6.3 Ukážka class diagramu pre čiary, generalizčnú šipku a ich spoločného predka

Špecifikuje atribúty pre:

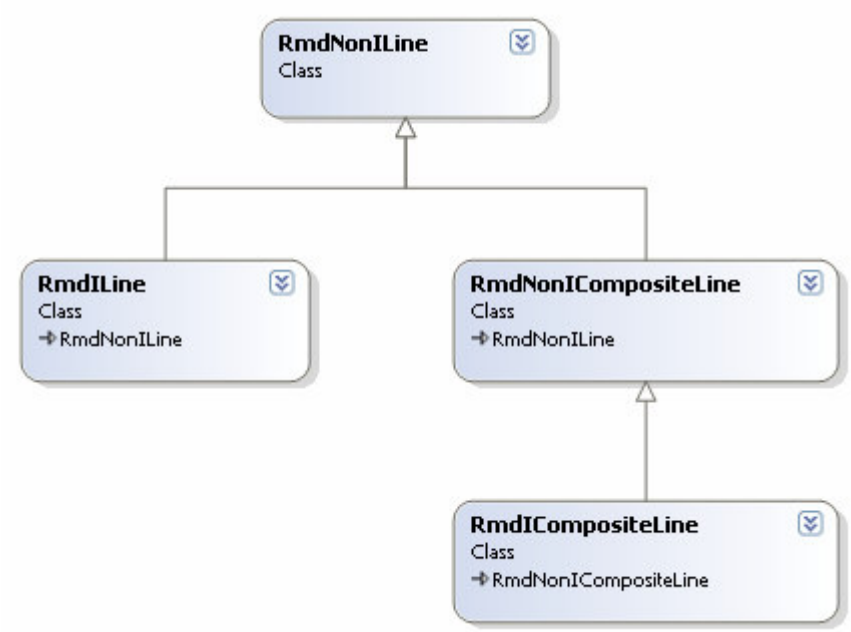
- GUID objektu
- Polohu počiatočného a koncového bodu
- Zapojenú entitu alebo vzťah na počiatočnom a koncovom bode

Ďalej vymedzuje virtuálne metódy pre:

- Vykreslenie objektu
- Zisťovanie, či sa daný bod nachádza vo vnútornej oblasti objektu (pre zisťovanie náležitosti kliku myšou do vnútornej oblasti objektu)
- Zobrazenie dialógu s vlastnosťami objektu
- Zapojenie a odpojenie entity (vzťahu) k čiare

6.6. Grafické objekty RMD modelu

Pre prácu s RMD diagramom boli vytvorené objekty pre tabuľku a relačné šipky a taktiež objekty pre atribúty tabuľky a referencie. Pre relačné šipky bol vytvorený objekt pre neidentifikačnú reláciu s priamym vykreslením (`RmdNonILine`), ktorý sa stal predkom objektov pre neidentifikačnú reláciu s lomeným vykreslením a taktiež pre identifikačnú reláciu s priamym aj lomeným vykreslením (Obr. 6.4).



Obr. 6.4 Ukážka class diagramu pre relačné šipky a ich spoločného predka

To teda umožnilo udržiavanie slovníka (scény) RMD modelu ako dvoch zoznamov. Jedného pre tabuľky, druhého pre relačné šipky.

Objekt tabuľka obsahuje atribúty pre:

- GUID tabuľky
- Polohu
- Velkosť
- Názov
- Zoznam stĺpcov (atribtútov) tabuľky
- Zoznam referenčných stĺpcov (atribútov) tabuľky
- Zoznam rodičov (tj. tabuliek, do ktorých sa odkazuje cudzím kľúčom)
- Zoznam potomkov (tj. tabuliek, ktoré sa odkazujú cudzím kľučom do tejto tabuľky)
- Zoznam napojených relačných šipiek

Definuje metódy pre:

- Vykreslenie tabuľky
- Zisťovanie, či sa daný bod nachádza vo vnútornej oblasti tabuľky (pre zisťovanie náležitosti kliku myšou do vnútornej oblasti tabuľky)
- Zobrazenie dialógu s vlastnosťami tabuľky
- Aktualizáciu polohy zapojených relačných šipiek pri zmene polohy tabuľky

Nadtrieda pre relačné šipky vymedzuje atribúty pre:

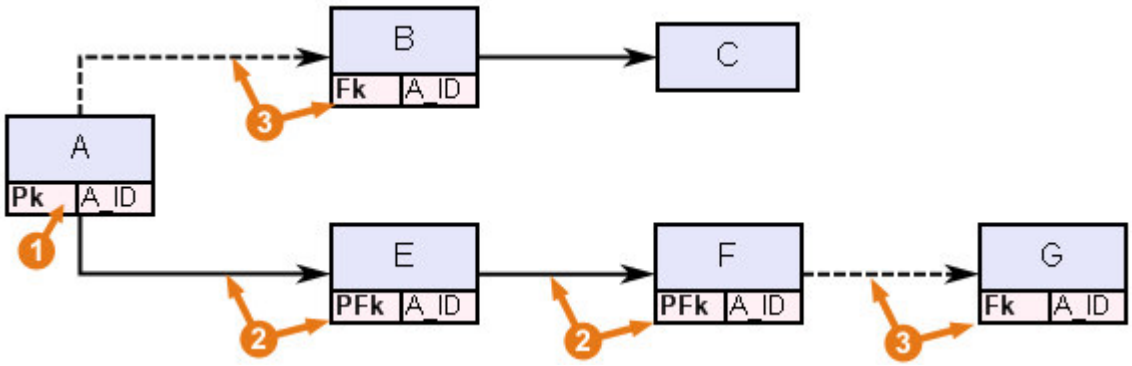
- GUID objektu
- Polohu počiatočného a koncového bodu
- Zapojenú tabuľku na počiatočnom a koncovom bode

Ďalej vymedzuje virtuálne metódy pre:

- Vykreslenie objektu
- Zisťovanie, či sa daný bod nachádza vo vnútornej oblasti objektu (pre zisťovanie náležitosti kliku myšou do vnútornej oblasti objektu)
- Zapojenie a odpojenie tabuľky k šipke

Pri prepájaní tabuliek relačnou šipkou (teda pri vytváraní relácie medzi tabuľkami) sa vytvorí špeciálny objekt pre túto reláciu, ktorý nesie informáciu o type spojenia (identifikačná alebo neidetifikačná relácia) a objekt tabuľky, na ktorú odkazuje. Tento objekt sa pridá do zoznamu rodičov tabuľky, ktorá je v tejto relácii chápana ako dcérska. Zároveň sa vytvorí rovnaký objekt (s protistrannou odkazujúcou tabuľkou) a pridá sa do zoznamu potomkov tabuľky, ktorá je v tejto relácii chápana ako rodičovská. Pri tejto operácii sa z rodičovskej tabuľky vytvoria objekty pre referenčné stĺpce, ktoré sú pridané do dcérskej tabuľky. V prípade, že dcérska tabuľka obsahuje ďalšie tabuľky, do ktorých z nej vedie relácia, sú tieto referenčné stĺpce rekurzívne delegované do jej potomkov. Pri neidentifikačnej relácii iba do prvej úrovne, pri identifikačnej relácii až k listom stromu prípadne k najbližšej neidentifikačnej relaácii.

Na obrázku 6.5 je ilustračná ukážka delegácie atribútu *A_ID* tabuľky *A* (na obrázku označený šípkou s číslom jedna) cez identifikačnú reláciu (na obrázku označená šípkou dva) a neidentifikačnú reláciu (na obrázku označená šípkou tri). V prípade, žeby sme v tabuľke *A* odstránili u atribútu *A_ID* príznak primárneho kľúča, tabuľka *A* by obsahovala tento nekľúčový atribút a všetky ostatné tabuľky by zostali bez akýchkoľvek atribútov. Teda akákoľvek zmena kľúčových atribútov rodičovskej tabuľky sa prenáša do všetkých dcérskych tabuliek a zároveň rekurzívne podľa typu relácie aj do ich potomkov.



Obr. 6.5 Ukážka prenosu referenčných stĺpcov

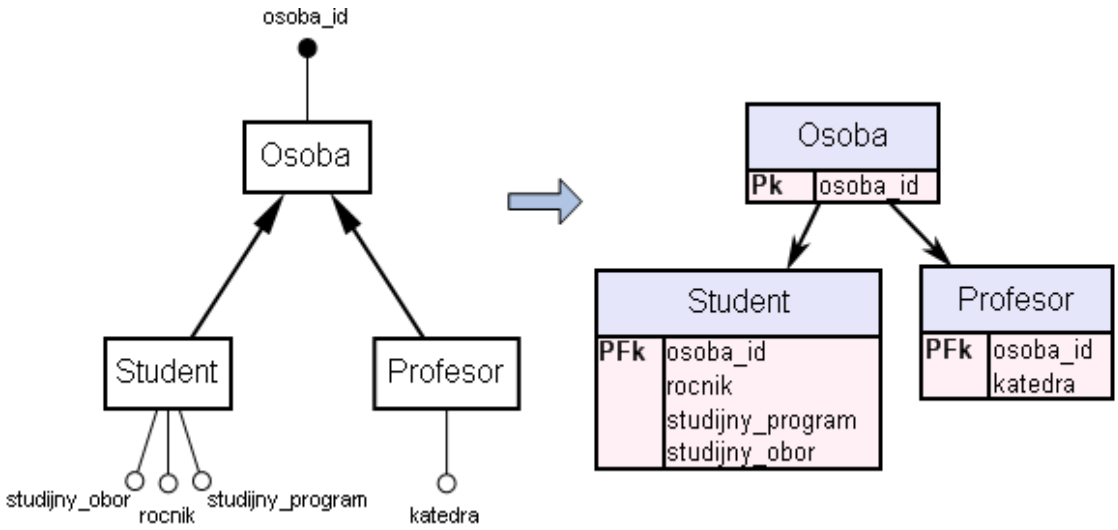
6.7. Validators – triedy pre kontroly korektnosti

Pre kontroly korektnosti diagramov sú implementované triedy, ktoré definujú metódy a strategie prevádzajucé tieto kontroly. Pre oba typy diagramov existuje samostatná trieda.

6.7.1. Trieda ErValidator

Pri vyvolaní požiadavku o kontrolu korektnosti ER diaramu (alebo pred prevodom) do SQL skriptu sa vytvorí inštancia špeciálnej triedy `ErValidator`, ktorá implementuje

metódy pre kontrolu jednoznačnosti názvov entít a vzťahov, jednoznačnosti názvov atribútov, existencie rolí u entít vystupujúcich vo vzťahu viackrát, existencie identifikačného cyklu, existencie identifikačného cyklu po prevode ISA hierarchie, existencie kľúčových atribútov pre zdroje ISA hierarchií a neexistencie kľúčov pre entity, ktoré nie sú zdrojmi ISA hierarchie.



Obr. 6.6 Ukážka prevodu ISA hierarchie do RMD modelu

Najvýznamnejšia je kontrola existencie cyklu po prevode ISA hierarchií na identifikačné vzťahy. V rámci tejto kontroly sú najprv prevedené ISA vzťahy na identifikačné a to tak, že každému potomkovi v ISA hierarchii je nastavený príznak, že je identifkačne závislý na predchodcovi v ISA hierarchii. Následne sa vytvorí graf entít, kde vrcholy sú GUID-y daných entít a každý vrchol obsahuje zoznam vrcholov, do ktorých z neho vedie hrana. Tento zoznam vrcholov je vlastne zoznamom identifikačných vlastníkov entity, teda akoby jej predchodcov. Takto vytvorený graf sa topologicky zotriedi a vytvorí sa zoznam usporiadaných entít pre použitie pri prevode ER modelu do RMD. Topologické triedenie tiež odhalí prípadny cyklus v grafe. Na obrázku 6.6 je ukážka samotného prevodu ISA hierarchie do RMD modelu.

6.7.2. Trieda RmdValidator

Pri vyvolaní požiadavku o kontrolu korektnosti RMD diagramu (alebo pred prevodom do SQL skriptu) sa vytvorí inštancia špeciálnej triedy `RmdValidator`, ktorá zapuzdruje všetky metódy pre vykonanie tejto kontroly. V rámci tejto kontroly sa overuje jednoznačnosť názvov tabuliek, jednoznačnosť názvov atribútov danej tabuľky, existencia primárneho kľúča tabuľky a kontrola existencie cyklu zapojenia tabuliek. Pri kontrole existencie cyklu sa zároveň vytvorí *dobré* usporiadanie tabuliek tak, aby pri generovaní SQL skriptu referencie odkazovali vždy do už vytvorených tabuliek. Pri tejto kontrole sa najprv vytvorí graf tabuliek, kde vrcholmi sú GUID-y tabuliek a každý vrchol obsahuje zoznam rodičovských tabuliek, do ktorých z neho vedie hrana. Zoznam týchto tabuliek je kôli menším nárokom na pamäť zoznamom GUID-ov rodičovských tabuliek. Takto vytvorený graf sa topologicky zotriedi, čím získame dobré usporiadanie tabuliek. Tento zotriedený zoznam je potom použitý pri generovaní SQL skriptu (viď kapitola 6.9). Algoritmus topologického triedenia tiež odhalí prípadny cyklus v grafe.

6.8. Converters – triedy pre prevod ER modelu do RMD

O prevod ER modelu do RMD sa starajú dve hlavné triedy – `HybridTable` a `Converter`. Konštruktor triedy `Converter` je volaný s dvoma parametrami. Jedným je aktuálna scéna ER diagramu a druhým je nová, prázdna scéna RMD diagramu, do ktorej budú pridané transformované objekty po skončení behu algoritmu. Samotný prevod prebieha v niekoľkých krokoch.

Na začiatku sa prejde zoznam topologicky zotriedených entít (ten bol vytvorený v rámci kontroly diagramu pred prevodom – viď kapitola 6.7.1) a pre každú entitu tohto zoznamu sa vytvorí inštancia triedy `HybridTable`. Tá bude ako svoje členské atribúty obsahovať:

- entitu, z ktorej vznikla
- názov (pri konštruovaní odpovedá názvu entity)
- polohu (odpovedajúcu polohe entity)
- príznak o tom, či sa používa
- odkaz na inštanciu `HybridTable`, s ktorou je spojená (tá je pri konštruovaní objektu nastavená na `null` hodnotu)
- tabuľku, ktorú vytvorí v RMD modele (tá je pri konštruovaní objektu tiež nastavená na `null` hodnotu)
- zoznam dcérskych inštancií triedy `HybridTable`, do ktorých sa budú delegovať kľúčové atribúty danej inštancie (pri konštruovaní objektu je nastavený na prázdny zoznam)

V druhom kroku sa prejdú všetky binárne vzťahy s kardinalitou (1,1):(1,1) a dojde ku spojeniu odpovedajúcich inštancií `HybridTable` do jednej inštancie. Druhá inštancia sa nevymazáva, iba sa jej nastaví príznak, že sa nepoužíva a tiež odkaz na prvú inštanciu, s ktorou je spojená. Táto referencia je veľmi dôležitá z dôvodu žiadosti o ďalšie spojenie (napríklad pri spracovávaní iného vzťahu, v ktorom je entita odpovedajúca druhej inštancii zapojená) tejto inštancie s inou. V takom prípade sa nebude spájať táto už nepoužívaná inštancia, ale `HybridTable`, na ktorý sa odkazuje. Pri spájaní inštancií sa názov ponechávanej inštancie reťazí s názvom tej, ktorá sa už ďalej nebude používať a rovnako sa do používanej inštancie nakopírujú atribúty nepoužívanej inštancie.

V tejto chvíli odpovedá zoznam inštancií `HybridTable` (ktoré majú nastavený príznak, že sa budú používať) entitám, ktoré sa prevedú na tabuľky RMD modelu. Preto sa v ďalšom kroku pre každú inštanciu vytvorí tabuľka s odpovedajúcim názvom a zoznamom atribútov. Tabuľky sa pridávajú do pomocného zoznamu. Do výslednej scény sa dostanú až na konci algoritmu. Pre všetky používané inštancie `HybridTable` sa aktualizuje zoznam dcérskych inštancií pre delegovanie kľúčových atribútov. Vychádza sa zo zotriedeného zoznamu entít a im odpovedajúcich používaných inštancií.

V ďalšom kroku sa pre každú tabuľku odpovedajúcu inštancii používaného `HybridTable` vytvoria identifikačné relácie, ktoré sa zapoja do tejto tabuľky na jednej strane a všetkých dcérskych tabuliek odpovedajúcim dcérskym inštanciam `HybridTable` spracovávanej inštancie na strane druhej. Po tomto kroku sú vyriešené všetky identifikačné závislosti a algoritmus ďalej pokračuje spracovávaním jednotlivých vzťahov podľa algoritmu popísaneho v kapitole 2.5. Problém viachodnotových atribútov je riešený až po spracovaní všetkých vzťahov.

V poslednom kroku algoritmu sa už iba prejde zoznam vygenerovaných tabuliek a relačných čiar a pridá sa do cieľovej scény. Tabuľky majú vo výslednom diagrame polohu, akú mali ich vzory v ER schéme. Jedným z možných rozšírení ER[G]edit-u by mohlo byť inteligentné riešenie pre usporiadanie tabuliek tak, aby sa navzájom neprekrývali a neboli krížene spojujúcimi relačnými čiarami.

6.9. Generovanie SQL skriptu

Trieda pre objekt tabuľky implementuje metódy, ktoré generujú *SQL CREATE a DROP* skript tejto tabuľky parametrizovane pre jednotlivé možnosti syntaxe. Členské atribúty tejto triedy obsahujú všetky potrebné referencie na rodičovské tabuľky, preto môže už táto trieda byť producentom svojho *drop* i *create* skriptu, hoci nemusí poznať usporiadanie a definície ostatných tabuliek scény. Pri generovaní výsledného SQL skriptu celej schémy sa už len prejde topologicky zotriedený zoznam tabuliek (ten je vytvorený v rámci kontroly korektnosti diagramu – viď kapitola 6.7.2) v tomto určenom poradí a pre každú tabuľku sa zavolá metóda pre generovanie skriptu tabuľky s príslušnými parametrami. Výsledný skript je zreťazením skriptov pre jednotlivé tabuľky. Ak bolo požadované tiež vytvorenie *drop* skriptu, zotriedený zoznam sa prechádza dvakrát. Prvýkrat v obrátenom poradí pre generovanie *drop* skriptu a druhýkrát v zotriedenom poradí pre generovanie *create* skriptu.

6.10. Helpers – pomocné triedy

Pre operácie, ktoré svojou funkčnosť nevykazujú náležitosť k žiadnej špecifickej triede, ale sú používané vrámci rôznych objektov a formulárov, boli vytvorené pomocné triedy. Definujú napríklad rozšírenia .NET Framework-ových tried pre prácu s grafikou (trieda `Helpers.Gfx`). Táto trieda implementuje statické metódy napríklad pre:

- Zisťovanie kvadrantu daného bodu vzhľadom k nejakému centrálnemu bodu chápanému ako stred dvojrozmernej súradnicovej siete
- Zisťovanie vzdialenosti dvoch bodov
- Vypočítavanie uhla dvoch priamok, či uhla výseku elipsy
- Zisťovanie priesečníku dvoch priamok, alebo priesečníkov priamky a obdlžníka, či priamky a elipsy

Trieda `Helpers.Alerts` poskytuje globálny slovník pre chybové a varovné hlásenia v rámci celej aplikácie a trieda `Helpers.Config` definuje globálne premenné pre obecné nastavenia aplikácie.

7. Záver

Cieľom tejto bakalárskej práce bolo vytvoriť CASE editor pre datové modelovanie podporujúci tvorbu ER diagramov, ich prevod do relačných schém s možnosťou ďalšej editácie a následného prevodu do SQL skriptov. Vytvorený program ER[G]edit tak ponúka dvojúrovňový editor pre tvorbu ER a RMD diagramov s možnosťou prevodu ER diagramu do RMD a technikami pre generovanie SQL skriptov z RMD schém. Dôležitá je tiež možnosť kontroly vytvorených schém a zobrazenie popisného prehľadu o nekorektnostiach v schéme, ktorý je pre výukové účely veľmi užitočný. Program sa snaží uľahčiť uživateľovi tvorbu schém množstvom funkcií spojených s automatickým vykresľovaním grafických objektov, aby sa tak uživateľ mohol viac sústrediť na samotný databázový návrh. Veľkou výhodu programu je možnosť kopírovania objektov, tlače diagramu, exportu do obrázkových formátov a hlavne možnosť práce s viacerými diagramami súčasne v rámci jedného prostredia.

Vychádzajúc z analýzy existujúcich programov s podobným zameraním (kapitola 3), mi je zrejmé, že nie je v silách jedinca, aby mohol sám v rozumnom čase vytvoriť CASE nástroj, ktorý by disponoval všetkými vlastnosťami a funkčnosťami ako komerčné nástroje, či aplikácie ktoré vznikali počas niekoľkých rokov a na ich tvorbe sa podieľal kolektív autorov. Z toho dôvodu, existuje množstvo ďalších rozšírení, o ktoré by mohol byť program ER[G]edit v budúcnosti doplnený. Ako napríklad:

- Inteligentné riešenie rozloženia tabuliek na scéne (layout) po prevode z ER diagramu
- Podpora viacstránkovej tlače diagramov
- Funkcie UNDO a REDO pre vrátenie zmien v diagrame
- Možnosť prepojenia dvoch tabuliek viacerými relačnými čiarami
- Implementácia prevodu z RMD modelu do ER
- Reverse engineering z SQL skriptu, či priamo z databáze po napojení sa k nej
- Podpora pre viacero typov databáz pri generovaní skriptu
- Možnosť definície uživateľských typov
- Možnosť exportu ER diagramu do súboru vo formáte aplikácie ERTOS (kapitola 3.7) a importu súborov z tohto formátu

Použitá literatúra

[1] Pokorný J., Halaška I.: *Databázové systémy: Vybrané kapitoly a cvičení*, Nakladatelství Karolinum UK, 1998.

[2] Pokorný J.: *Konceptuální anlýza a návrh*, prezentace http://kocour.ms.mff.cuni.cz/~pokorny/vyuka/pokorny.konceptual_ad/.

[3] Pokorný J., Halaška I.: *Databázové systémy*, prezentace http://kocour.ms.mff.cuni.cz/~pokorny/vyuka/srbd/ds/.

[4] Batini C., Cero S., Navathe S. B.: *Conceptual database design*, The Benjamin/Cummings Publishing Company, 1992.

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

[6] Pokorný J.: *IS, Binární E-R model, transformace do RMD*, prezentace http://kocour.ms.mff.cuni.cz/~pokorny/vyuka/pokorny.binarni_er/.

[7] Pokorný J., Halaška I.: *E-R modelování*, prezentace http://kocour.ms.mff.cuni.cz/~pokorny/vyuka/srbd/er/.

[8] Pokorný J., Halaška I.: *Relační model dat*, prezentace http://kocour.ms.mff.cuni.cz/~pokorny/vyuka/srbd/rmd/.

[9] Pokorný J., Halaška I.: *Návrh relací*, prezentace http://kocour.ms.mff.cuni.cz/~pokorny/vyuka/srbd/nf/.

[10] Pokorný J., Halaška I.: *Databázové modely*, prezentace http://kocour.ms.mff.cuni.cz/~pokorny/vyuka/srbd/md/.

[11] Pokorný J., Halaška I.: *Transformace*, prezentace http://kocour.ms.mff.cuni.cz/~pokorny/vyuka/srbd/trn/.

[12] Reynolds-Haertle R.: *OOP – objektově orientované programování – Visual Basic .NET, Visual C# .NET krok za krokem*, Mobil Media a.s., 2002.

[13] Robinson S., Allen K., Cornes O., Glynn J. Greenvoss Z., Harvey B., Nagel Ch., Skinner M., Watson K.: *C# Programujeme profesionálně*, Nakladatelství Computer Press, 2003.

[14] MacDonald M.: *Pro .NET 2.0 Windows Forms and Custom Controls in C#,* Appres, 2006.

[15] Procházka J.: Nástroje CASE? Co? Proč? Jak?, článek http://www.dbsvet.cz/view.php?cisloclanku=2004052702

[16] MagicLibrary.dll, http://www.dotnetmagic.com/downloads/MagicInstall174.msi

[17] Dia, http://www.gnome.org/projects/dia/.

[18] Microsoft Visio, http://office.microsoft.com/en-us/FX010857981033.aspx.

[19] SmartDraw, http://www.smartdraw.com/.

[20] CASE Studio, http://www.casestudio.com/csy/default.aspx.

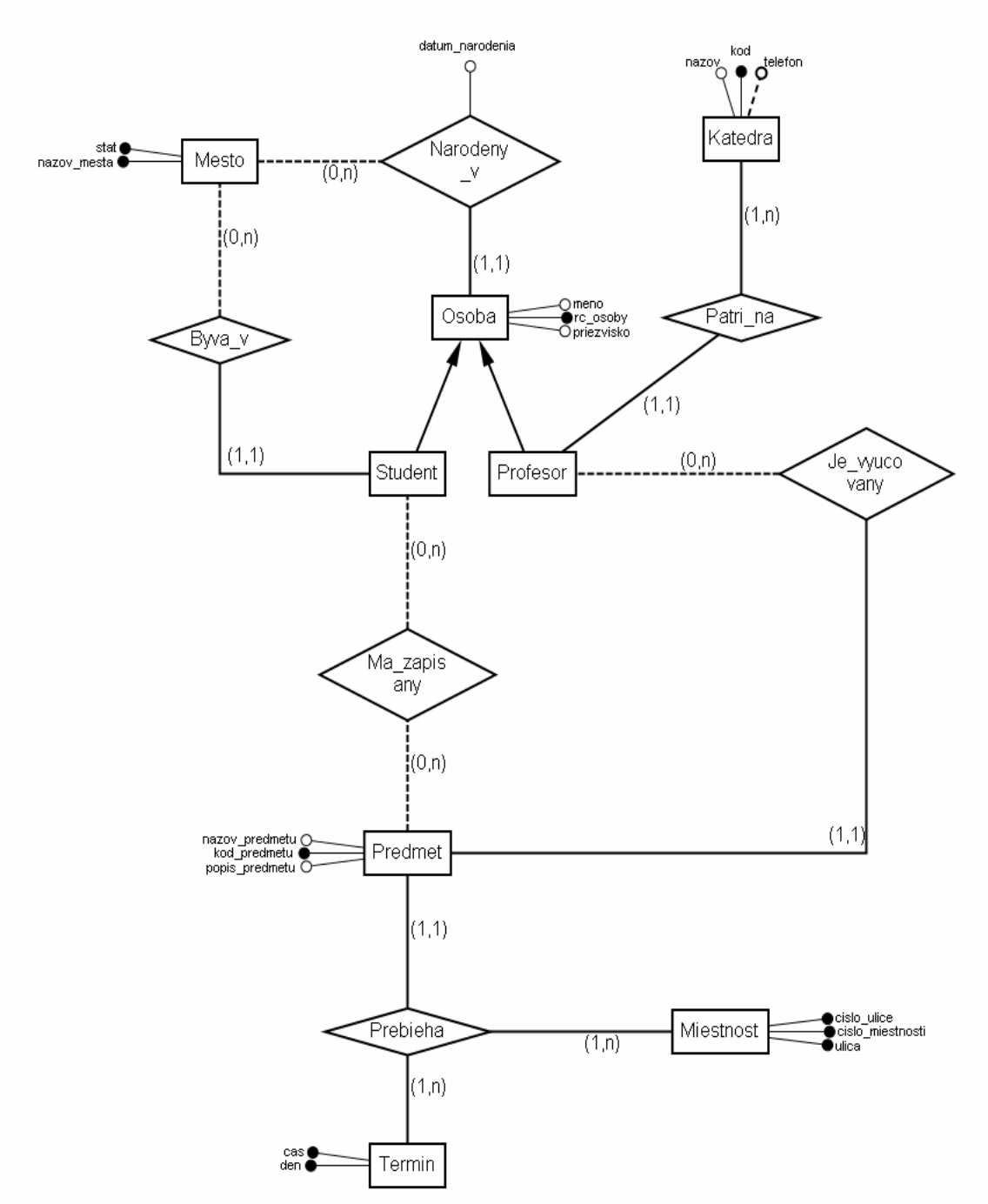
[21] ERTOS, http://urtax.ms.mff.cuni.cz/skopal/projekty/ErtosSetup.exe

[22] XTG Data Modeller, http://www.xtgsystems.com/xtgdm.php3

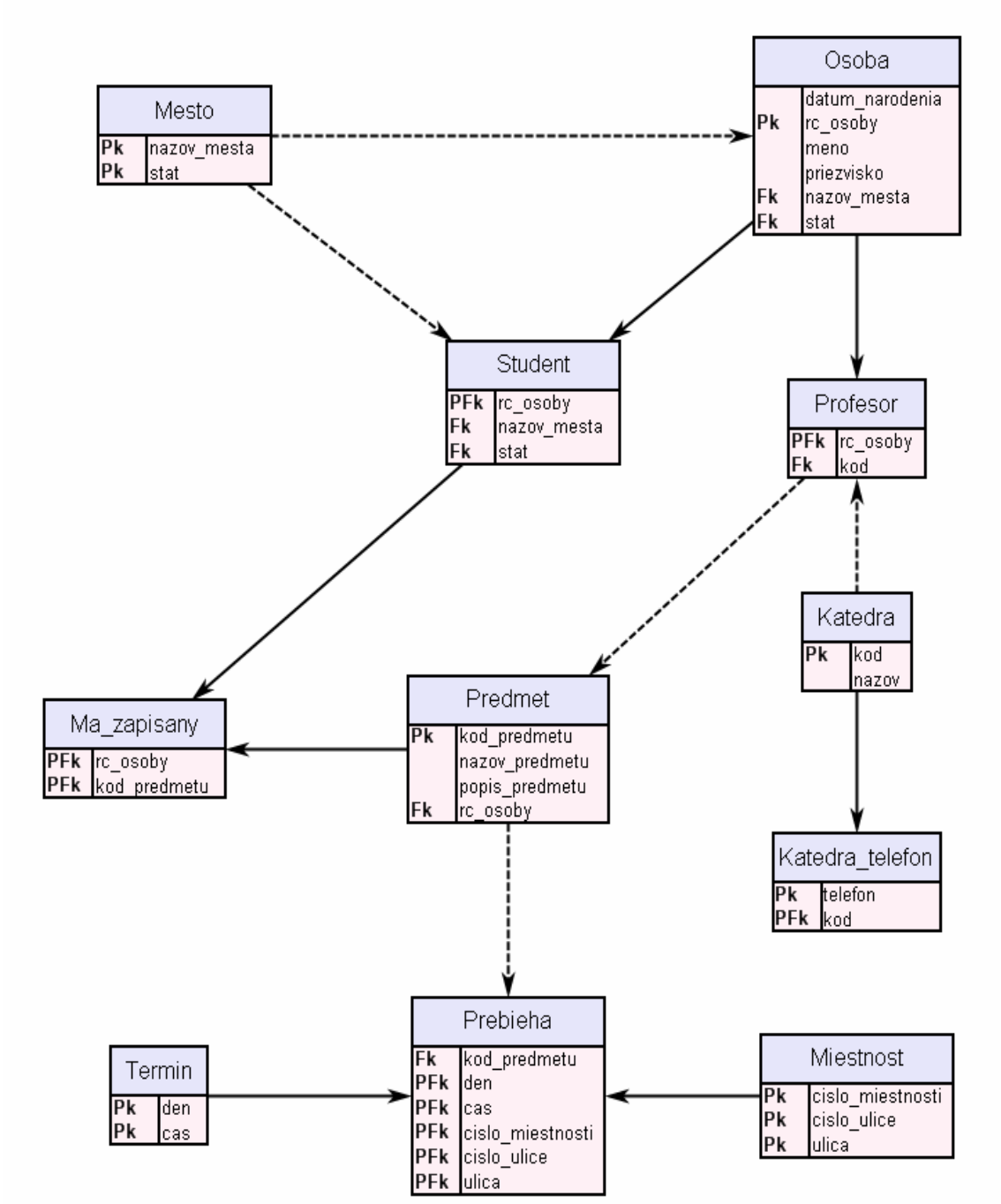
[23] ER Modeller, http://service.felk.cvut.cz/courses/X36DBS/

Prílohy:

PRÍLOHA A – ukážka ER diagramu vytvoreného v aplikácii ER[G]edit



PRÍLOHA B – ukážka RMD diagramu vytvoreného z ER diagramu z prílohy A



PRÍLOHA C – ukážka SQL skriptu vygenerovaného z RMD diagramu v prílohe B

```
-- ER[G]edit - generovany skript
-- Vytvorene: 05.06.2007 05:15:31

DROP TABLE Katedra_telefon;
DROP TABLE Prebieha;
DROP TABLE Ma_zapisany;
DROP TABLE Student;
DROP TABLE Predmet;
DROP TABLE Profesor;
DROP TABLE Osoba;
DROP TABLE Mesto;
DROP TABLE Miestnost;
DROP TABLE Termin;
DROP TABLE Katedra;


CREATE TABLE Katedra (
   kod   Integer NOT NULL,
   nazov   Varchar(100) NOT NULL,
CONSTRAINT Katedra_pk PRIMARY KEY (kod)
);

CREATE TABLE Termin (
   den   Varchar(10) NOT NULL,
   cas   Varchar(10) NOT NULL,
CONSTRAINT Termin_pk PRIMARY KEY (den, cas)
);

CREATE TABLE Miestnost (
   cislo_miestnosti   Integer NOT NULL,
   cislo_ulice   Integer NOT NULL,
   ulica   Varchar(50) NOT NULL,
CONSTRAINT Miestnost_pk PRIMARY KEY (cislo_miestnosti, cislo_ulice,
ulica)
);

CREATE TABLE Mesto (
   nazov_mesta   Integer NOT NULL,
   stat   Integer NOT NULL,
CONSTRAINT Mesto_pk PRIMARY KEY (nazov_mesta, stat)
);

CREATE TABLE Osoba (
   datum_narodenia   Date NOT NULL,
   rc_osoby   Integer NOT NULL,
   meno   Integer NOT NULL,
   priezvisko   Integer NOT NULL,
   nazov_mesta   Integer NOT NULL,
   stat   Integer NOT NULL,
CONSTRAINT Osoba_pk PRIMARY KEY (rc_osoby),
CONSTRAINT Osoba_fk_Mesto FOREIGN KEY (nazov_mesta, stat) REFERENCES
Mesto(nazov_mesta, stat)
);

CREATE TABLE Profesor (
   rc_osoby   Integer NOT NULL,
   kod   Integer NOT NULL,
CONSTRAINT Profesor_pk PRIMARY KEY (rc_osoby),
```

```
CONSTRAINT Profesor_fk_Osoba FOREIGN KEY (rc_osoby) REFERENCES
Osoba(rc_osoby),
CONSTRAINT Profesor_fk_Katedra FOREIGN KEY (kod) REFERENCES
Katedra(kod)
);

CREATE TABLE Predmet (
   kod_predmetu   Integer NOT NULL,
   nazov_predmetu   Varchar(100) NOT NULL,
   popis_predmetu   Varchar(255) NOT NULL,
   rc_osoby   Integer NOT NULL,
CONSTRAINT Predmet_pk PRIMARY KEY (kod_predmetu),
CONSTRAINT Predmet_fk_Profesor FOREIGN KEY (rc_osoby) REFERENCES
Profesor(rc_osoby)
);

CREATE TABLE Student (
   rc_osoby   Integer NOT NULL,
   nazov_mesta   Integer NOT NULL,
   stat   Integer NOT NULL,
CONSTRAINT Student_pk PRIMARY KEY (rc_osoby),
CONSTRAINT Student_fk_Osoba FOREIGN KEY (rc_osoby) REFERENCES
Osoba(rc_osoby),
CONSTRAINT Student_fk_Mesto FOREIGN KEY (nazov_mesta, stat) REFERENCES
Mesto(nazov_mesta, stat)
);

CREATE TABLE Ma_zapisany (
   rc_osoby   Integer NOT NULL,
   kod_predmetu   Integer NOT NULL,
CONSTRAINT Ma_zapisany_pk PRIMARY KEY (rc_osoby, kod_predmetu),
CONSTRAINT Ma_zapisany_fk_Student FOREIGN KEY (rc_osoby) REFERENCES
Student(rc_osoby),
CONSTRAINT Ma_zapisany_fk_Predmet FOREIGN KEY (kod_predmetu)
REFERENCES Predmet(kod_predmetu)
);

CREATE TABLE Prebieha (
   kod_predmetu   Integer NOT NULL,
   den   Varchar(10) NOT NULL,
   cas   Varchar(10) NOT NULL,
   cislo_miestnosti   Integer NOT NULL,
   cislo_ulice   Integer NOT NULL,
   ulica   Varchar(50) NOT NULL,
CONSTRAINT Prebieha_pk PRIMARY KEY (den, cas, cislo_miestnosti,
cislo_ulice, ulica),
CONSTRAINT Prebieha_fk_Predmet FOREIGN KEY (kod_predmetu) REFERENCES
Predmet(kod_predmetu),
CONSTRAINT Prebieha_fk_Termin FOREIGN KEY (den, cas) REFERENCES
Termin(den, cas),
CONSTRAINT Prebieha_fk_Miestnost FOREIGN KEY (cislo_miestnosti,
cislo_ulice,     ulica)     REFERENCES     Miestnost(cislo_miestnosti,
cislo_ulice, ulica)
);

CREATE TABLE Katedra_telefon (
   telefon   Varchar(20) NOT NULL,
   kod   Integer NOT NULL,
CONSTRAINT Katedra_telefon_pk PRIMARY KEY (telefon, kod),
CONSTRAINT Katedra_telefon_fk_Katedra FOREIGN KEY (kod) REFERENCES
Katedra(kod));
```