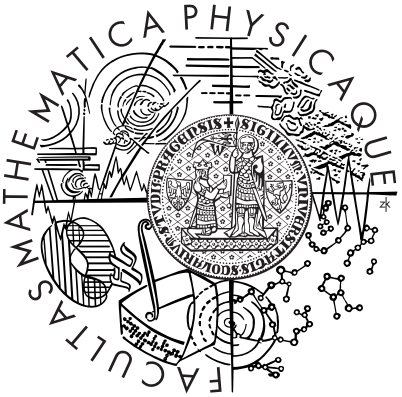
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Peter Marko

Převod relačních schemat do existujících ontologií

Katedra softwarového inženýrství

Vedúci diplomovej práce: Mgr. Jaroslav Tykal
Študijný program: Informatika, Softwarové systémy

2007

Ďakujem vedúcemu diplomovej práce Mgr. Jaroslavovi Tykalovi za ochotu a pomoc pri jej tvorbe, knižnici University of Limerick za poskytnutie zázemia na tvorbu jej prvej časti a v neposlednom rade svojím rodičom, ktorí mi umožnili študovať a celé roky ma podporovali.

Prehlasujem, že som svoju diplomovú prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím s požičiavaním práce a jej zverejňovaním.

V Prahe dňa 8.8.2007 Peter Marko

Obsah

Názov práce: Převod relačních schemat do existujících ontologií
Autor: Peter Marko
Katedra (ústav): Katedra softwarového inženýrství
Vedúci diplomovej práce: Mgr. Jaroslav Tykal
e-mail vedúceho: Jaroslav.Tykal@mff.cuni.cz

Abstrakt: Táto diplomová práca sa venuje problému mapovania relačných schém do už existujúcich ontológií, pričom sa orientuje na ontológie zapísané pomocou RDFS. Úvodná časť podáva prehľad a základné princípy nájdených podobných riešení. Podľa poskytovanej pomoci užívateľovi ich člení na manuálne a semiautomatické prístupy. Obsahom druhej časti je návrh vlastného riešenia mapovania - mDR. Vo svojich základoch vychádza z existujúceho riešenia FDR2, zovšeobecňuje ho však. Je rozdelené do troch fáz: tvorba pomocných informácií o databáze, mapovanie a prevod inštancií. Záverečná tretina práce je venovaná v jazyku JAVA vytvorenému nástroju JmDR kompletne implementujúceho prvé dve fázy.
Kľúčové slová: sémantický web, ontológia, relačná schéma, mapovanie, RDFS

Title: Mapping of Relational Schema onto Existing Ontologies
Author: Peter Marko
Department: Department of Software Engineering
Supervisor: Mgr. Jaroslav Tykal
Supervisor's e-mail address: Jaroslav.Tykal@mff.cuni.cz

Abstract: This master thesis concerns the problematics of mapping of relational schemas to existing ontologies, primarily written in RDFS. Its first part gives a view at already existing solutions and their principles. The solutions are divided into the manual and semiautomatic groups according to the level of a help given to users when mapping. Secondly the thesis presents an own solution - mDR. In its basics it does use ideas of the existing FDR2 system, however it generalises them. The mapping is done in three steps: creation of helping information about a database, mapping and instances transfer. The last third of the work involves JmDR, a (in JAVA) developed tool completely implementing the first two phases.
Keywords: semantic web, ontology, relational schema, mapping, RDFS

Kapitola 1

Úvod

> „Úspech sémantického webu kľúčovo závisí na jednoduchej tvorbe, integrácii a použití metadát.“ S. Handschuh[9]

Sémantický web, v ktorom informáciam v ňom zahrnutých dokážu okrem ľudí rozumieť i počítače je pojem, ktorý v ostatných rokoch rezonuje čoraz viac a viac. Jeho základná myšlienka, že istým dohodnutým spôsobom stroju pristupujúcemu k dátam „vysvetlíme“, čo dáta znamenajú je pekná, no naráža na pre svoju existenciu podstatnú prekážku - ako získať takéto vysvetľujúce dáta o dátach v reálnom prostredí?

Tieto takzvané metadáta môžeme prirodzene uvažovať a nejakým spôsobom spisovať (napríklad manuálne) hneď pri tvorení samotných dát. Čo ale s dátami už existujúcimi? Veľká časť z nich býva uložená v relačných databázach, ktoré sa postupom času z rôznych dôvodov široko rozvinuli a upevnili a bývajú napríklad zdrojom množstva dynamicky generovaných webových prezentácií, elektronických obchodov a podobne. Ak má mať sémantický web šancu na budúce väčšie rozšírenie, nesmie podceniť význam dát existujúcich databáz. Schodnejšou cestou než zmenou dávnovytvorených a dlhoročne používaných prístupov a prostriedkov ukládania dát sa preto zdá byť (aspoň dočasné) prispôsobenie sa prístupov nových. To znamená nutnosť vývoja rôznych pomocných nástrojov na uľahčenie prevodu, konverzie a mapovania dát relačných databáz na dáta ontológií[1] sémantického webu a naopak.

Už v aj tak širokej oblasti je problematika samotného prevodu dát stále dosť rozsiahla. Podľa definovania, čo a pri akých podmienkach sa vlastne prevodom, respektíve mapovaním dát myslí, by sa dala ďalej rozdeľovať. Ktorým smerom (alebo oboma?) máme na mysli dáta mapovať? Je dovolené so vstupnými dátami hýbať? Ako môžu ontológie, respektíve tabuľky databázy (alebo dokonca databáz?) vyzerať? A tak ďalej.

Táto diplomová práca sa zaoberá problémom prevodu dát výlučne smerom z relačnej schémy na už existujúcu, vopred vytvorenú ontológiu. Dô-

[1] Pojem ontológie je zľahka vysvetlený v nasledujúcej kapitole.

raz kladie na spôsob definovania, ako by sa schéma tabuliek databázy dala pretransformovať na „schému“ sémantického webu. Kedže ale tento spôsob následne poskytuje zväčša dostatočnú informáciu aj pre prevod konkrétnych inštancií, tak predstavuje princíp celého prevodu. Technickejšiemu presunu dát sa preto práca venuje iba okrajovo.

Po krátkom vysvetlení potrebných pojmov kvôli následnému presnejšiemu definovaniu cieľa sa vo svojej prvej časti práca zameriava na prehľad nájdených a zaujímavých, už jestvujúcich riešení podobného mapovania. Prístupy sa rovnako ako ich východzie podmienky líšia od jedného k druhému, preto nie všetky dokonale spĺňajú podmienky vymedzené predchádzajúcim odsekom. Druhá časť práce je venovaná návrhu vlastného spôsobu prevodu schém a následne jeho čiastočnej implementácii.

Kapitola 2

Stručný úvod k ontológiam

Pri snahách o zmapovanie existujúcich prístupov prevodu relačných schém na existujúce ontológie a o vytvorenie vlastného systému je nutné sa často odkazovať na isté termíny z oblasti Sémantického webu a to predovšetkým na ontológie a ich vlastnosti. Pre účely ľahšej orientácie v nasledujúcich častiach práce je tak vhodné zľahka predstaviť samotný pojem ontológie a zjednotiť a vysvetliť niektoré pojmy ďalej používané. Keďže mnou navrhnutý prevod počíta s ontológiami zapísateľnými v RDF Schema[1], krátky úvod sa orientuje predovšetkým smerom k tomuto jazyku popisu ontológií.

2.1 Ontológie

Podľa často uvádzanej definície je *ontológia* „explicitnou a formálnou špecifikáciou konceptualizácie istej domény nášho záujmu“[8]. Iná, voľnejšia definícia vraví, že ontológie poskytujú spôsob klasifikácie vecí, o ktorých sa dá rozprávať v danom kontexte. Klasifikáciam sa priradzujú mená a definujú vlastnosti a vzťahy, ktoré môžu nadobúdať[12]. Jednoducho povedané však ide o istý dohodnutý spôsob reprezentácie vedomostí.

Sémantický web je v rámci informatiky len jednou z najmladších oblastí, kde sa v rôznych formách[2] ontológie používajú. Dlhšie využitie nachádzajú v umelej inteligencii, v spracovaní prirodzeného jazyka, v znalostnom manažmente, či v informačných architektúrach.

„Najnižším“ pojmom vrámci ontológii je *inštancia* predstavujúca konkrétnu „vec“, objekt, prvok domény, v ktorej sa pohybujeme, teda napríklad istého človeka, produkt, mesto, poprípade slovo, číslo alebo pesničku.

Trieda ako abstraktnejší pojem predstavuje typ, kategóriu vecí - inštancií, napríklad ľudia, mobilné telefóny, slová, pesničky. Inštancie môžu patriť

[1] Ďalej často len RDFS.

[2] Uplatnenie ontológie nachádzajú napríklad v implementáciach rozumne prepojených elektronických obchodov, inteligentných vyhľadávacích nástrojov alebo efektívnejších znalostných webov.

do žiadnej, alebo niekoľkých tried. Nad triedami je definovateľná ich hierarchia (pomocou definovaní predkov, respektíve potomkov; napríklad ľudia sú potomkami organizmov, skratky potomkami slov a podobne).

Inštancie majú v rámci tried svoje *vlastnosti - atribúty*. Konkrétny telefón môže mať atribút sériové číslo, farbu, označenie, človek svoje meno, vek a podobne.

Medzi triedami je umožnené definovať (binárne) *relácie - vzťahy* (ak meno považujeme za samotnú triedu, nie len atribút, tak trieda človek môže byť napríklad v relácii „má meno“ s touto triedou meno). Relácie môžu byť podobne ako triedy hierarchicky zorganizované. Pre ilustráciu tak napríklad relácia „má meno“ môže byť potomkom relácie „má názov“. Jednou z najdôležitejších relácií je samotná relácia má predka, respektíve potomka, pomocou ktorej sa definuje hierarchia tried, či samotných relácií. Rozdiel medzi reláciou a atribútom je, že relácia definuje vzťahy medzi triedami, kdežto atribút je len predpisom priradzujúcim argumentu - konkrétnej inštancii svoju primitívnu „dátovo-typovú“ hodnotu (ktorá neodpovedá žiadnej inštancii).

Možno sa zdá, že opísaný systém má mnoho spoločného s pojmami objektovo orientovaného programovania. Zčasti áno, no na druhej strane však existujú významové rozdiely. Podobným črtom je možnosť definovania hierarchie tried. Inštancie patria triedam, jazyky ontológií však vo všeobecnosti umožňujú príslušnosť inštancií k žiadnej, či viacerým triedam. Triedy sú narozdiel od typov inštancií považované za množiny inštancií. Relácie pridávajú možnosť popísania širšieho množstva vzťahov medzi triedami. Nie sú však napevno spojené so žiadnou triedou, sú „občanmi prvej kategórie“[15], ich vzťah k triedam je definovaný integritnými obmedzeniami (definičný obor a obor hodnôt). Triedy môžu nadobúdať viacero rovnakých atribútov, či relácií (človek môže mať viacero telefónov). Samotné jazyky ontológií (a rovnako RDFS) všeobecne takisto neposkytujú spôsob vynútenia si správneho zápisu vzťahov z oblasti záujmu a vo všeobecnosti slúžia väčšinou skôr k popisu tried a ich vzťahov než konkrétnych inštancií.

V posledných desaťročiach sa vyvinuli mnohé typy ontológií, ako napríklad *Cyc*, *Ontolingua*, z novších a pre sémantický web priamo určených potom *SHOE*, *RDFS*, *OIL*, *DAML+OIL*, či *OWL*. Popísané názvy základnych pojmov ontológii a ich charakteristika sa líšia od systému k systému. Zvolená terminológia tohto úvodu k ontológiam je orientovaná smerom k jazyku dôležitom z hľadiska tejto práce - RDFS. Dôkladnejší prehľad ontológií by mal spomenúť možnosti silnejších systémov pridávajúcich možnosti n-árnych relácií, funkcií (hodnota je narozdiel od jedného určená n argumentami), atribútov tried, disjunktných rozkladov, či rôznorodých integritných obmedzení a podobne.

```
<?xml version="1.0"?>
  <rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:exterms="http://www.example.org/terms/">

  <rdf:Description rdf:about="http://www.example.org/index.html">
    <exterms:creation-date>August 16, 1999</exterms:creation-date>
    <dc:language>en</dc:language>
    <dc:creator rdf:resource="http://www.example.org/staffid/85740"/>
   </rdf:Description>
</rdf:RDF>
```

Príklad 2.1: RDF zapísané v RDF/XML (Zdroj W3C[18])

2.2 RDF, RDF Schema

Jedným zo spôsobov zápisu ontológií je jazyk na popis znalostí s názvom RDF Schema[18] často nazývaný aj RDFS. Ide o sémantické rozšírenie jazyka RDF, ktorý stručne povedané štandartizuje zápis informácií.

RDF[18] vznikol v roku 1999 ako jazyk pre potreby popisu zdrojov webu (autorov, dátumu publikovania). Jeho všeobecnosť však umožnila možnosť popisovania vecí a vzťahov reálneho sveta, nielen webu. Funguje na princípe „dekompozície znalostí na menšie časti s dohodnutým významom týchto častí“[17]. Dekompozícia tu znamená rozloženie znalostí na *trojice* skladajúce sa z *podmetu* - *subjektu*, *prísudku* - *predikátu* a *predmetu* - *objektu* (Česká republika, je členom, Európska únia; Európska únia, má obyvateľov, 500 miliónov). Subjekty, predikáty a objekty sú menami odkazujúcimi sa na skutočné, alebo abstraktné veci a vzťahy popisovaného sveta nazývané *zdrojmi* - *resources*[3]. Zdroje v rôznych trojiciach môžu vystupovať v rôznych roliach a ich mená sa kvôli možnosti distribuovania znalostí a súvisiacej nutnosti jednoznačnej identifikácie zapisujú v URI. V pozícii objektu môžu okrem zdrojov identifikovaných pomocou URI vystupovať aj takzvané *literály*, čo sú viacmenej len reťazcové konštanty predstavujúce istú vlastnosť subjektu (popísanú predikátom). Základom pre reprezentáciu RDF je jeho grafový model (kde jednoducho povedané orientované hrany od subjektov k objektom predstavujú predikáty), konkrétny zápis grafu nie je až tak dôležitý. Pre ľahkú manipuláciu pomocou už vyvinutých prostriedkov pre XML je možné RDF zapísať v RDF/XML (Príklad 2.1), človekom ľahšie čitateľný je takzvaný zápis trojicami (triples notation, Príklad 2.2[4]).

RDF nepredstavuje nič viac než len štandartizovaný spôsob dekompozície a zápisu znalostí a ako také subjektom, predikátom a objektom až na pár

[3]Odtiaľ je aj pomenovanie Resource Description Framework.

[4]Uvedený príklad nie je v úplne presnom formáte zápisu trojicami.

výnimiek žiadny špeciálny význam nedáva. Popis vzťahov a konceptov na vyšších úrovniach, a teda popis ontológií prináša vďaka možnosti definovania tried a ich vlastností (v zmysle predchádzajúcej kapitoly) až jeho nadstavba s názvom RDF Schema.

RDFS je slovník (vocabulary) v RDF zapísaných špeciálnych zdrojov s dohodnutým významom, ktoré predstavujú „typový systém pre RDF“[18]. Na definovanie tried a hierarchií tried sa používajú zdroje `rdfs:Class`, `rdfs:Resource`, `rdf:type` a `rdfs:subClassOf`. Špeciálny význam pre definíciu vlastností, hierarchie vlastností a ich integritných obmedzení majú zdroje `rdf:Property`, `rdfs:subPropertyOf`, `rdfs:domain` a `rdfs:range`.

Nie je účelom tejto práce podrobne charakterizovať všetky črty RDFS[5], preto základné črty pre potreby tejto práce zachytí iba konkrétny príklad. Keďže RDFS dokument je len RDF dokument s dohodnutým významom niektorých zdrojov, tak sa na nasledujúci príklad možno pozerať aj ako na bežný RDF zápis trojíc subjekt, predikát, objekt bez špeciálneho významu.

```
ex:Book           rdf:type            rdfs:Class .
ex:Person         rdf:type            rdfs:Class .
ex:Male           rdf:type            rdfs:Class .
ex:Female         rdf:type            rdfs:Class .
ex:Male           rdfs:subClassOf     ex:Person .
ex:Female         rdfs:subClassOf     ex:Person .
ex:hasParent      rdf:type            rdf:Property .
ex:hasMother      rdf:type            rdf:Property .
ex:hasFather      rdf:type            rdf:Property .
ex:hasMother      rdfs:subPropertyOf  ex:hasParent .
ex:hasFather      rdfs:subPropertyOf  ex:hasParent .
ex:hasParent      rdfs:domain         ex:Person .
ex:hasMother      rdfs:range          ex:Female .
ex:hasFather      rdfs:range          ex:Male .
ex:favBook        rdf:type            rdf:Property .
ex:favBook        rdfs:domain         ex:Person .
ex:favBook        rdfs:range          ex:Book .
ex:age            rdf:type            rdf:Property .
ex:age            rdfs:domain         ex:Person .
ex:age            rdfs:range          xsd:integer .
xsd:integer       rdf:type            rdfs:Datatype .
```

Príklad 2.2: RDFS zapísané v trojiciach

Príklad 2.2 (inšpirovaný RDF Primer[18]) popisuje triedu `ex:Person`[6], s potomkami `ex:Male`, `ex:Female` a triedu `ex:Book`. Trieda `ex:Person` má de-

[5]Všetky črty sú dokonale definované v samotných odporúčaniach konzorcia W3C, medzi ktoré patrí pekne pochopiteľný úvod RDF Primer[18] a v početných menej technických vysvetleniach typu Tauberer[17].

[6]Prefix `ex`, podobne ako prefixy `rdf`, či `rdfs` je skratkou pre menný priestor (namespace), v ktorom je trieda `Person` zahrnutá a spolu s názvom zdroja tvorí jeho jedinečné URI používané v RDF (čím RDFS stále je). V demonštračných príkladoch predstavuje namespace `ex` fiktívnu adresu `http://www.example.org/`.

finovanú vlastnosť `ex:hasParent`, vlastnosti `ex:hasMother` a `ex:hasFather` sú od nej odvodené. Definičným oborom týchto vlastností je `ex:Person`, oborom hodnôt pre `has:Mother` je len `ex:Female` a pre `has:Father` len `ex:Male`. Triedy `ex:Person` (a tým aj triedy `has:Mother` a `has:Father`) a `ex:Book` sú spojené cez vlastnosť `ex:favBook`. Atribútom `ex:Person` je `ex:age`, ktorej oborom hodnôt je literál (a teda nie trieda) typu `xsd:integer`.

RDFS vo svojej terminológii nerozlyšuje medzi pojmami relácia a atribút, vzťahy medzi triedami a ako aj medzi triedou a literálom nazýva *vlastnosťou - property* (relácia medzi triedami `ex:hasParent`, ako aj atribút `ex:age` medzi triedou a literálovými hodnotami sú obe v Príklade 2.2 typu `rdf:Property`). V ďalšom texte sa však napriek tomu rozlišujú pojmy relácie a atribútu.

Jazyk popisu ontológií RDF Schema je expresívne slabší[7] (ďalšie možnosti pridávajú prehĺbenia DAML-ONT, DAML+OIL, či najnovšie OWL), no je jednoduchší, rozšírenejší a často aj postačujúci. Mnohé existujúce prístupy mapovania uvažujú buď iba RDFS, alebo ak aj uvažujú širšie jazyky, tak len teoreticky a s tým, že implementované majú len RDFS.

Kapitola 3

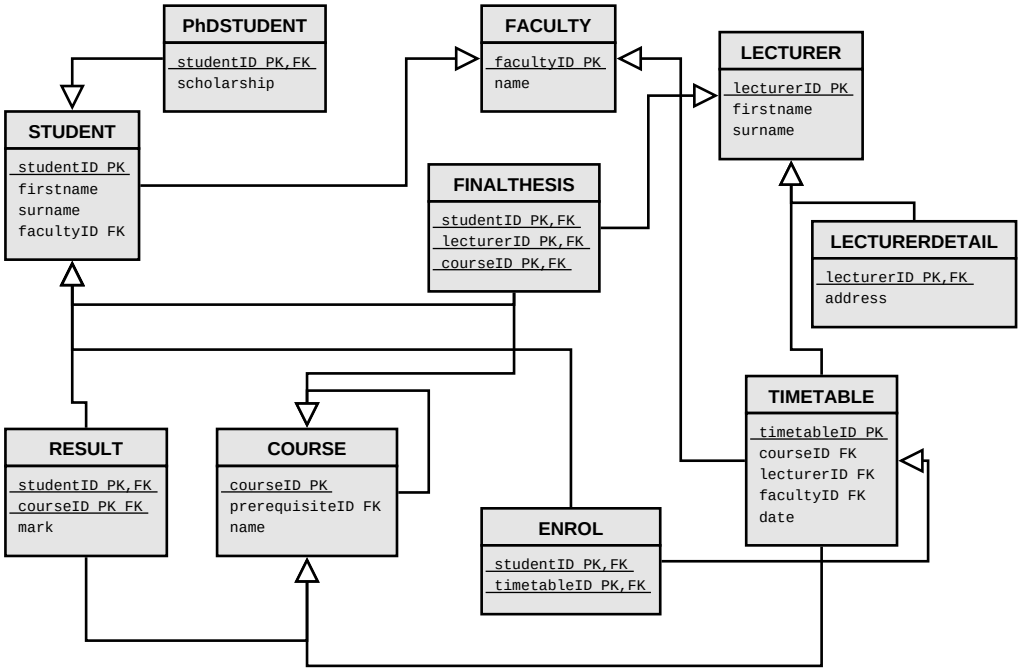
Priblíženie problému

V úvodnej kapitole už bolo spomenuté, že práca sa zaoberá jestvujúcimi a vlastným prístupom k presunu relačnej schémy do už existujúcej ontológie a taktiež, že sa sústreďuje predovšetkým na mapovanie schém. To je ale veta stále príliš všeobecná. Účelom nasledujúcich riadkov je preto tento presun a jeho východzie podmienky presnejšie vymedziť a špecifikovať.

Čo sa týka databáz, dôležité je zodpovedať, aké informácie a do akej miery myslíme pod pojmom databáza, respektíve ako približne môže taká databáza vyzerať. V situáciach nutnosti prevodu dát databáz do čohokoľvek sú väčšinou k dispozícii samotné dáta a metadáta len na nižšej úrovni, teda bez akýchkoľvek dodatočných „vyšších“ informácii napríklad vo forme E/R diagramu. Túto nevýhodu čiastočne zjemňuje zvyčajné poctivejšie indikovanie existujúcich vzťahov medzi tabuľkami definíciou cudzích kľúčov. Na druhej strane ale v databázach štandartne nastáva veľký počet rôznych situácií, ktoré je vhodné aj pri prevode na ontológie uvažovať. Typicky napríklad vzťahy medzi entitami tabuliek s kardinalitami *1:n* (poprípade *1:1*, zachytené v jednej z tabuliek), alebo *m:n* (s využitím špeciálnej väzobnej tabuľky), reflexívne relácie, rozdelenie informácií o entite do viacerých tabuliek, dedičnosť entít, viac než binárne relácie a podobne. V prezentovaných existujúcich prístupoch a aj vo vlastnom navrhovanom riešení je skúmaná (resp. hľadaná) schopnosť prevodu práve v takýchto najbežnejších situáciach. Podmienkou je prístup k reláciam relačnej schémy len na úrovni SQL, kedy spomenuté situácie medzi entitami za tabuľkami môžeme len odhadovať podľa tvaru tabuliek[1]. Indikácia cudzích kľúčov je vítaná. Podobne sú výhodou taktiež relácie vo vyšších normálnych formách. Keďže bežné stĺpce tabuliek podmienku atomickosti v nich obsiahnutých údajov spĺňajú, požadované je aspoň neporušovanie prvej normálnej formy.

Pre ilustratívne účely pri posudzovaní vlastností riešení je namieste definovať „typickú“ relačnú schému, ktorá má byť namapovaná na ontoló-

[1] V niektorých prípadoch (dedičnosť medzi entitami) ide o hádanie v najpravejšom zmysle slova.

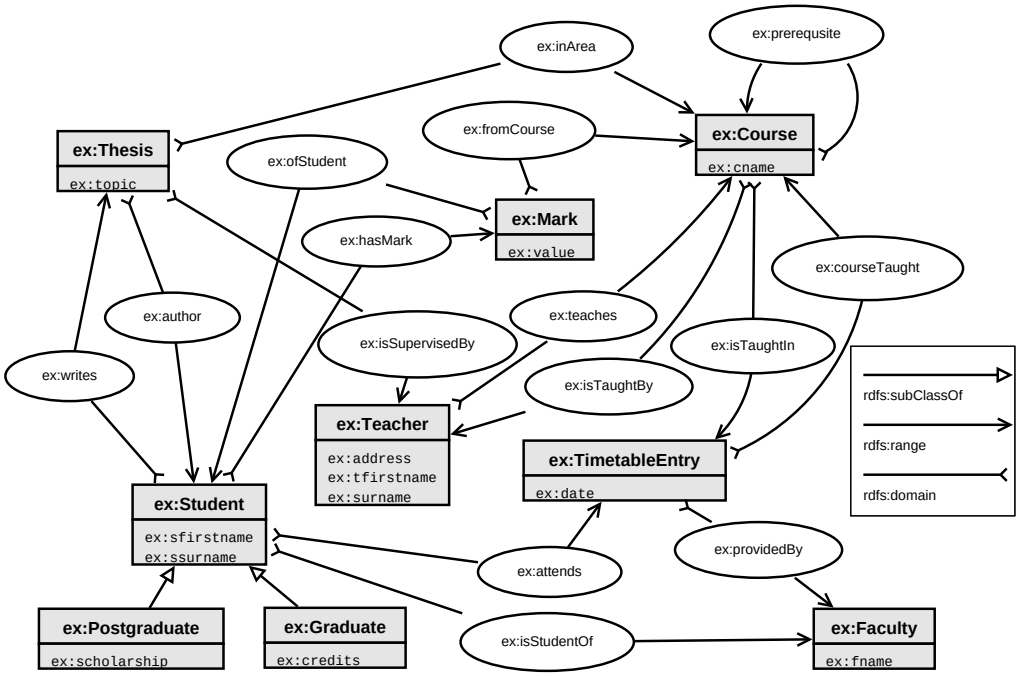


Obrázok 3.1: Príklad prevádzanej relačnej schémy o univerzite

giu. Schéma na Obrázku 3.1 obsahuje mnohé zo štandartných situácií. Zastúpené sú prepojenia typu *1:n* (`STUDENT` a `FACULTY`), *m:n* (`STUDENT` a `TIMETABLE` v `ENROL`), viac než binárny vzťah (`FINALTHESIS`), dedičnosť entít (`PhDSTUDENT` a `STUDENT`), rozšírenie informácií o entite ďalšiou entitou (`LECTURER` a `LECTURERDETAIL`), či reflexívny vzťah entity (`prerequisiteID` v `COURSE`). Niektoré prepojenia sú z dôvodov vyvstania rôznych situácií pri mapovaní na ontológie dokonca v rôznych formách (tabuľka `RESULT` predstavuje iný typ prepojenia *m:n* s dodatočnou informáciou `mark`; `TIMETABLE` je tiež viac než binárne prepojenie so samostatným primárnym kľúčom).

Na druhej strane, schéma cieľovej, už existujúcej ontológie je zapísaná v regulérnom RDFS. Neobsahuje teda nič zložitejšie než definíciu tried, podtried, ich relácií a podrelácii. Ontológiu považujeme za danú, vrámci riešenia nemennú a bez existujúcej populácie[2]. Pochopiteľne je nutná „istá podobnosť“ so vstupnou relačnou schémou. Čo to ale znamená? Rozvláčne povedané, už zo samotnej nutnosti mapovania existujúcej databázy do prázdnej ontológie plynie, že ontológia bola vytvorená nedávno s vedomosťou tejto nutnosti, a tak sa schémy podobajú „dostatočne“. Úplne odlišnú schému zmysel mapovať tak či tak nemá. Z ilustratívnych dôvodov (najmä pri hľadaní vlastného prístupu) sú podobne ako pre databázu vytvorené ukážky vo forme dvoch rôznych RDFS ontológií.

[2]Pridanie populácie by skomplikovalo v podstate len prevod konkrétnych inštancií, kde



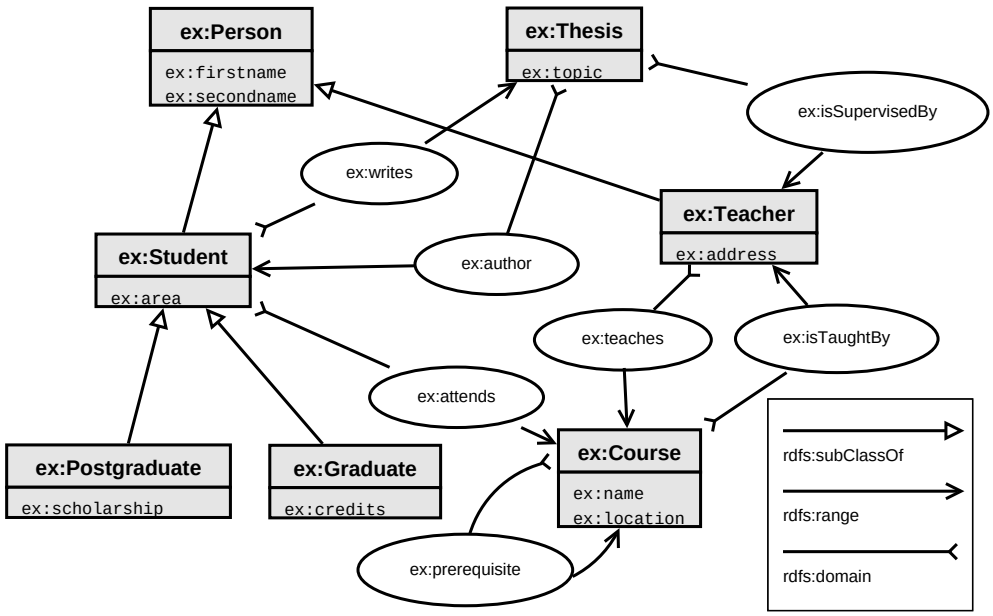
Obrázok 3.2: Príklad cieľovej ontológie väčšej podobnosti

Obrázok 3.2 predstavuje ontológiu viacmenej odpovedajúcu pôvodnej schéme. Jedinými markatnejšími rozdielmi sú mierne (od názvov tabuliek) rozdielne názvy tried (`TEACHER` od `LECTURER`), neexistencia všetkých relácií (nie ku všetkým reláciam je definovaná aj inverzia - `ex:isSupervised`) a z hľadiska nemožnosti namapovania zbytočná trieda a atribút (`ex:Graduate` a atribút `topic` triedy `ex:Thesis`). Menší problém nastáva pri otázke, čo namapovať na relácie `ex:teaches` a `ex:isTaughtBy` (použiteľné by mohli byť stĺpce `courseID` a `lecturerID` v `TIMETABLE`). Inak by riešenie mapovanie na takto podobnú schému malo zvládať.

Zaujímavejšou je ale ontológia na Obrázku 3.3, kde pribúdajú otáznejšie situácie. Atribúty tried neodpovedajú stĺpcom tabuliek tak dokonale. Navyše triedy `ex:Student` a `ex:Teacher` sú odvodené od `ex:Person`, pri mapovaní atribútov `ex:firstname` a `ex:surname` z tabuliek `STUDENT` a `LECTURER` je tak na ňu potrebné nezabudnúť. Zo schémy vypadla trieda `ex:Mark`, `ex:Faculty` a `ex:TimetableEntry`. Neprítomnosť tretej menovanej komplikuje sitáciu s mapovaním `ex:Course` vo vzťahu k `ex:Teacher` a najmä k `ex:Student`.

Uvedené príklady približujú typ problémov, ktoré treba pri mapovaní v mantineloch tejto práce riešiť. Pri inak definovaných príkladoch by pravdepodobne vyvstali aj iné problémy. Kombinácia relačnej schémy na Obrázku 3.1 predovšetkým s RDFS ontológiou na Obrázku 3.3 sa ale snaží pokryť veľké množstvo problémov. Preto okrem toho, že v nasledujúcich častiach

by vznikla nutnosť riešenia správneho namapovania URI inštancií a občasných kolízií.



Obrázok 3.3: Príklad cieľovej ontológie menšej podobnosti

práce sú na nej ilustrované existujúce prístupy mapovania, je aj základom pre vlastný návrh prevodu.

Kapitola 4

Existujúce prístupy

Problematika prevodu dát relačných databáz na dáta ontológií ako súčasť Sémantického webu je síce v porovnaní s inými oblasťami informatiky relatívne mladá, no zároveň je stará natoľko, aby umožnila vznik prvých rôzných riešení. Ich rozličnosť je daná rôznorodosťou definovania základného problému mapovania a podmienok, či obmedzení, vrámci ktorých sa pohybujú. To znamená napríklad, že kým zadanie prevodu istých prístupov predpokladá dáta v databázach tvorené „slušne“ (to znamená tabuľky vo vyšších normálných formách, tvorené na základe E/R diagramov, s údajmi o cudzích kľučoch a podobne), zadania iných sa snažia viac priblížiť často reálnym prostrediam s nie najväčšou „slušnosťou“ tabuliek databázy[1].

Inými zdrojmi rozdielov býva, či dáta tabuliek sú mapovaciemu, či konverznému riešeniu prístupné priamo, alebo napríklad len pomocou (v lepšom prípade) rôznych webových služieb, respektíve (v horšom prípade) len z existujúcích dynamicky generovaných webových stránok. Výstupom niektorých býva len akýsi mapovací dokument definujúci spôsob prevodu, iné riešenia poskytujú celý mechanizmus prevodu od databázy k ontológii (teda aj prevod inštancií). To ale nutne neznamená, že takýto spôsob je dokonalejší. Definovanie len mapovania býva napríklad výhodnejšie pri integrácii ontológie s viacerými a častomeniteľnými rôznorodými zdrojmi. Samotné slovo mapovanie tak môže byť chápané rôznymi spôsobmi – len ako istý popis vzájomnej konverzie dát, ako prevod dát, alebo ako oboje. Z kontextu by mal byť jasný jeho konkrétny význam.

Nie všetky riešenia dokonca uvažujú prevod dát relačnej databázy do už existujúcich ontológií. Keďže ale zmyslom tejto časti práce je okrem podania prehľadu nepochybne aj inšpirácia pre mnou hľadané riešenie, prístupy mapovania dát relačných schémat na ontológie neuvažujúce existenciu ontológií sú na nasledujúcich stranách zahrnuté tiež.

Cieľom prehľadu nie je podať absolútne vymenovanie všetkých konkrét-

[1] V tejto oblasti niekedy jestvujúce riešenia vybehujú z podmienok určených predchádzajúcou kapitolou.

nych existujúcich riešení, ale predstaviť rôzne typy prístupov pomocou popisu ich typických predstaviteľov. Veľmi dôležitým rozdielom medzi prístupmi, ktorý vyplýva z počiatočných podmienok zadania je stupeň automatiky mapovania relačných schém na ontológie. Kým niektoré zverujú do rúk a mysle manuálneho obsluhovateľa prevodu všetko, iné sa snažia do istej miery prevod uľahčiť, zautomatizovať. Aj keď sa od seba konkrétne riešenia líšia v mnohých vymenovaných vlastnostiach a často nie úplne zreteľne, umelo sú zaradené práve do abstraktných kategórii *manuálne* a *semiautomatické* riešenia podľa stupňa ich samostatnosti od užívateľa. Nakoľko miera automatiky nutne určuje mnohé z vlastností, tak práve takéto rozvrstvenie sa podľa môjho názoru zdá byť pre prezentačné účely najvhodnejšie.

4.1 Manuálne riešenia

Do prvej kategórie predstavených prístupov patria tie, u ktorých je celé mapovanie v rukách užívateľa. Ten osobne a ručne nadefinuje prevod, poprípade aj prevod sám pomocou nástrojov prevedie. Nepomáha mu v tom nijaká ukrytá logika.

4.1.1 D2R MAP

D2R MAP[5] je riešenie pozostávajúce z D2R, čo je deklaratívny XML jazyk umožňujúci popis mapovaní medzi relačnou databázovou schémou a ešte neexistujúcimi ontológiami typu RDFS[2], a D2R procesoru, ktorý na základe konkrétneho zápisu v tomto jazyku do istej miery prevedie relačnú databázu vrátane inštancií do novotvorenej ontológie.

Jazyk okrem technických elementov popisujúcich spojenie s databázou, definície menných priestorov pre vytvárané zdroje, či iných technických správ D2R procesoru obsahuje pre mapovanie dôležité tri typy elementov. Element `d2r:ClassMap` slúži na definície vytvorení tried a ich atribútov vznikajúcej ontológie. Obsahuje pritom SQL dotaz, ktorý definuje, ktoré tabuľky databázy pri tvorbe triedy a jej inštancií prezrieť, ktoré stĺpce vybrať a pomocou svojho `groupBy` atribútu[3] taktiež podľa hodnôt ktorých stĺpcov má výsledky dotazu zoskupovať (pomocou `GROUP BY` časti SQL dotazu) a vytvárať tak inštancie tried. Vnáraný element `d2R:DatatypePropertyBridge` definuje, ako z vybratých stĺpcov výsledku toho istého SQL dotazu vytvoriť atribúty (nie relácie) inštancií tvorenej triedy. Na definovanie relácií, a teda spojení medzi triedami (a konieckoncov tak i ich inštanciami) slúži element `d2r:ObjectPropertyBridge`, ktorý pomocou svojich atribútov `refferedClass` a `referredGroup` spojí vytváranú triedu (v ktorej elemente je vnorený tento spájací element) s inou vytváranou triedou s identifikátorom `id` a jej `groupBy` atribútmi.

Príklad 4.1 je zápisom tvorby tried ontológie `ex:Student` a `ex:Mark` z tabuliek databázy `STUDENT(studentID, firstname, surname, facultyID)` a `RESULT(studentID, courseID, mark)` (z Obrázka 3.1). Atribútmi tried sú `ex:firstname`, `ex:surname` a `ex:value`. Keďže primárným kľúčom tabuľky `RESULT` je dvojica stĺpcov `studentID` a `courseID` a údaje o druhom menovanom stĺpci v tabuľke `STUDENT` nie sú, tak pre potreby tvorby relácie `ex:hasMark` medzi dvoma triedami je nutný `SELECT` cez obe tabuľky, čo D2R umožňuje. Okrem mnohých elementov pre tvorbu ostatných prvkov v ukážke kódu očividne chýba definovanie relačného spojenia `ex:Student` s `ex:Faculty`, ktoré by ale bolo analogické. Za povšimnutie stojí systém dosa-

[2]Špecifikácia jazyka spomína OWL, v súčasnosti funguje len RDFS.

[3]V prípade rozpravy o XML a ontológiach zároveň je potrebné rozlišovať o atribút „čoho“ sa jedná. V tomto prípade o atribút XML elementu.

```
<?xml version="1.0"?>
<d2r:Map xmlns:d2r="http://...D2RMap/0.1#">
  <d2r:DBConnection d2r:odbcDSN="uniDB" />
  <d2r:Namespace d2r:prefix="ex" d2r:namespace="...example.org/uni#"/>
  <d2r:ClassMap d2r:type="ex:Student"
    d2r:sql="SELECT STUDENT.studentID, firstname, surname, courseID
             FROM STUDENT,RESULT WHERE STUDENT.studentID=RESULT.studentID"
    d2r:groupBy="STUDENT.studentID"
    d2r:uriPattern="ex:Student@@STUDENT.studentID@@">
      <d2r:DatatypePropertyBridge d2r:property="ex:sfirstname"
       d2r:column="STUDENT.firstname" />
      <d2r:DatatypePropertyBridge d2r:property="ex:ssurname"
       d2r:column="STUDENT.surname" />
      <d2r:ObjectPropertyBridge d2r:property="ex:hasMark"
       d2r:referredClass="ex:Mark"
       d2r:referredGroupBy="STUDENT.studentID,courseID"/>
  </d2r:ClassMap>
  <d2r:ClassMap d2r:type="ex:Mark"
    d2r:sql="SELECT studentID, courseID, mark FROM RESULT"
    d2r:groupBy="studentID, courseID"
    d2r:uriPattern="ex:Mark@@studentID@@-@@courseID@@">
      <d2r:DatatypePropertyBridge d2r:property="ex:value"
           d2r:column="mark" />
  </d2r:ClassMap>
</d2r:Map>
```

Príklad 4.1: D2R zápis konkrétneho prevodu tabuliek relačnej databázy na ontológiu

dzovania hodnôt stĺpcov do tvorených inštancií dovoľujúci s nimi slabo manipulovať, čo sa hodí napríklad pri tvorbe URI inštancií (`d2r:uriPattern`).

Výhodou D2R je používanie SQL umožňujúce definovať triedy v rôzne navrhnutých databázach. Vznik ontológií z Obrázkov 3.2 a 3.3 by z ukážkovej databázy bolo možné v D2R úspešne definovať. Dokonca využitie zložitejších SQL dotazov by vynahradilo aj neexistenciu `ex:TimetableEntry` v druhej menovanej, či definovať triedy vzniknuvšie cez viaceré tabuľky, čo iné systémy nedokážu. Užitočnou vlastnosťou je počítanie s `GROUP BY`, ktoré dovoľuje v prípade potreby vytvárať inštancie tried určované viacerými stĺpcami.

Sila D2R Map si berie svoju daň v podobe ťažkopádnosti a nutnej manuálnej tvorbe D2R súboru človekom, kto databáze a žiadúcej ontológii rozumie. Systém nepočíta s RDFS vzťahmi `rdfs:subClass`, `rdfs:subProperty` a možnosti úprav a transformácií hodnôt stĺpcov databáz sú oproti inak veľkým možnostiam priveľmi obmedzené. Z pohľadu tejto práce je taktiež dôležité, že v pôvodnom návrhu sa neuvažujú existujúce ontológie. Stručne povedané, D2R predstavuje doslova len zápis techniky prevodu dát tabuliek databázy na schému a inštancie novotvorenej ontológie.

4.1.2 R_2O

Súčasťou komplexného riešenia ESPERONTO[4] snažiaceho sa „prekonať medzeru medzi WWW a Sémantickým webom upgradovaním obsahu WWW“ je projekt Fund Finder[3] vznikajúci v Madride. Jeho úlohou je migrácia dát databáz do ontológii, vstupom databáza a taktiež ontológia, ktorú chceme naplniť. To znamená, že toto riešenie už od začiatku počíta s existujúcou ontológiou.

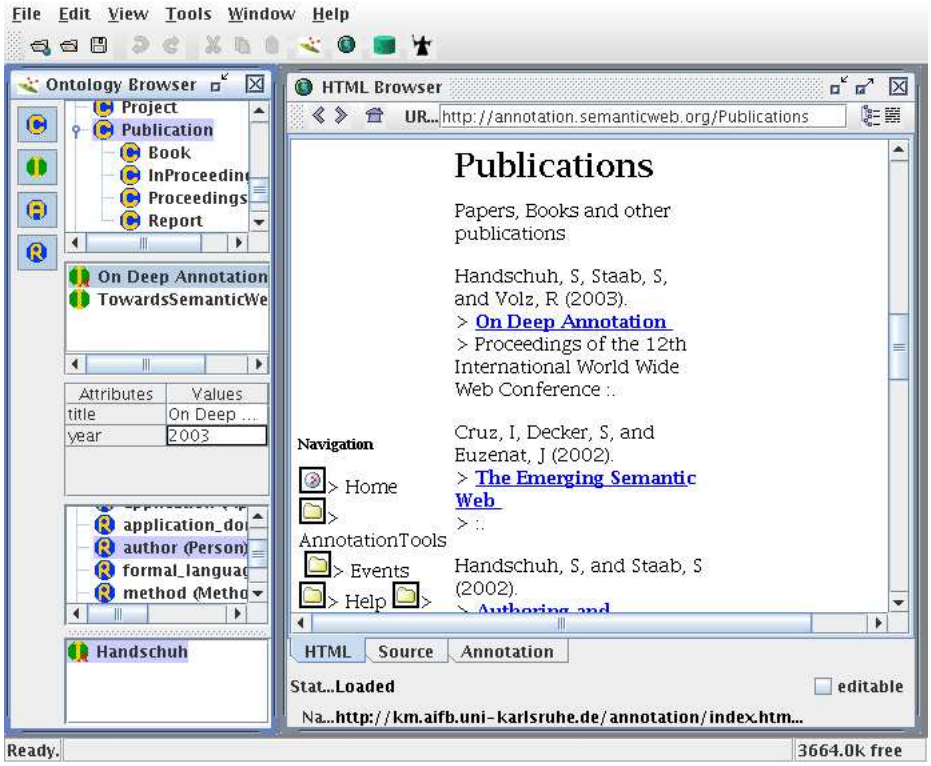
Princíp fungovania je podobný projektu D2R. V deklaratívnom jazyku zapísané mapovanie databázy na ontológiu sa predloží procesoru pre prevod, v tomto prípade ODEMapster. Pôvodne sa ako tento jazyk navrhoval eD2R ako rozšírenie D2R o ďalšie elementy umožňujúce (aj podmienené) transformácie hodnôt stĺpcov s využitím regulárnych výrazov, vyhľadávaní kľúčových slov a podobne. Jeho miesto ale následne zaujal podobný novovytvorený jazyk R_2O[4] pridávajúci ešte ďalšie možnosti. Tento jazyk už vznikal za predpokladu, že databáza a ani ontológia neboli primárne vytvorené pre účel mapovania, a jeho cieľom je popísať mapovanie z databázy na existujúcu ontológiu vo formej v akej sú. Tiež ide o XML súbor využívajúci SQL a tiež obsahuje elementy pre tvorbu tried, relácií, atribútov. Aby bol schopný riešiť viac reálnych problémov súvisiacich s uvažovaním existencie ontológie a „škaredších“ databáz, pridáva možnosti zložitých transformácií hodnôt (podreťazce, spájanie výrazov, matematické operácie) a podmienok (rovnosť, obsahovanie kľúčového slova, väčšie než, menšie než a podobne) za ktorých sa rôzne pravidlá (mapovania, transformácie) vrámci otcovských elementov pri prevode procesorom uplatňujú.

Takto expresívne silný jazyk umožňuje zápis mapovania v širšom množstve reálnych prostredí. R_2O[5] zvláda pomocou svojej sily napríklad aj popis prevodu ľahko štruktúrovaných tabuliek, ktoré nie sú v prvej normálnej forme. Vysoká vyjadrovacia sila však so sebou okrem nesporných výhod nesie i veľkú nevýhodu, a to oproti D2R ešte väčšiu zložitosť implementácie nástrojov tento jazyk využívajúcich. Jazyk je svojou veľkosťou neprehľadný, a tak narozdiel od možnosti manuálneho popisu prevodu v prípade D2R sa rovno počíta s vývojom pomocných grafických nástrojov. Zatiaľ ale také nástroje ešte neexistujú.

Autori tvrdia, že R_2O narozdiel od D2R uvažuje existujúce ontológie. To je síce pravda, no treba si uvedomiť, že mapovanie na ontológie v oboch prípadoch znamená len zápis prevodu v tom, či onom jazyku. Uvažovanie existujúcich ontológií v R_2O tak predstavuje len vyššiu expresivitu schopnú zachytiť viac reálnych mapovacích situácií. Napriek tomu, že pre to pôvodne tvorený nebol, D2R v niektorých prípadoch podobne využitý pravdepodobne byť môže.

[4] http://www.esperonto.net/

[5] Relational to Ontology



Obrázok 4.1: Prostredie Ontomat-Annotizer

4.1.3 Hlboká anotácia pomocou OntoMat-Annotizer

Za možno nie najšťastnejším prekladom anglického On Deep Annotation[9] sa skrýva komplexné riešenie prevodu dát databáz do existujúcich ontológií. Jeho fundamentálne podmienky sú od podmienok iných riešení, vrátane riešenia hľadaného tohto prácou rozdielne. Dáta určené k prevodu totiž síce takisto pochádzajú z databáz, no nie sú dostupné priamo. Pristupuje sa k nim len cez existujúce webové služby, ktoré ich prezentujú vo forme tabuliek webových stránok. Úlohou mapovaciej časti riešenia je tak mapovanie prevodu dát získavaných z webových stránok na ontológie, čo je anotácia dát. Keďže sa ale vie o databáze za nimi skrytej[6], jedná sa o takzvanú hlbokú anotáciu.

Mapovanie prebieha v grafickom prostredí OntoMat-Annotizer (Obrázok 4.1). To rozlišuje medzi takzvanou literálovou a generickou anotáciou. V prvom prípade hodnota vrámci výsledku webovej služby „predstavuje sama seba“ a má tvoriť hodnotu inštancie alebo atribútu, v druhom prípade predstavuje len meno, handler pre ktorý sa má vytvoriť premenná (trieda, relácia, či atribút). Program pre pojmy zobrazovaného výsledku we-

[6]Ku anotácii z webovej služby je napríklad nutná prítomnosť súboru podrobne popisujúceho závislosť tabuliek vracaných službou na tabuľkách databázy.

bovej služby(vpravo na obrázku), napríklad hodnôt, či nadpisov tabuliek ponúka možnosti rôznych mapovaní. Ak užívateľ namapuje pojem stránky na novú triedu ontológie, vytvorí sa nová trieda (cieľová ontológia je na obrázku vľavo). Namapovaním na novú inštanciu sa naopak zas vytvorí jej nová inštancia. Následne je umožnené pridávať hodnoty jej atribútov preťahovaním pojmov služby do tabuľky s atribútmi triedy. Podobne užívateľsky prívetivo sa vytvárajú relácie.

Prevod medzi dátami databázy a existujúcou ontológiou je iného typu než v prípade D2R alebo R_2O. Nejde ani tak o definíciu mapovania, ako o samotný manuálny prevod dát webovej služby na inštancie ontológie využitím sofistikovaného grafického programu. Webové služby a ontológie musia počítať s mantinelmi tohto prístupu, nakoľko ten „nevyzerá, že by riešil viac komplexné situácie“[3]. Mapovanie v OntoMat-Annotizer je len jednou z častí celkového riešenia, tie sú však z pohľadu tejto práce vedľajšie[7].

Napriek tomu, že prezentované riešenie je svojím zadaním a orientáciou na inštancie a webové služby dosť rozdielne, poskytuje jedno z mála už vyvinutých grafických mapovacích prostredí podávavajúcich predstavu, ako by podobné nástroje mohli vyzerať.

[7]Pre zaujímavosť systém po prevode umožňuje vďaka všetkým informáciam, ktoré má k dispozícii zadávanie dotazov na ontológiu meniť na dotazy webovej službe cez ktorú sa nepriamo získajú potrebné dáta databázy a následne vrátia späť vo forme ontológie.

4.2 Semiautomatické riešenia

Nie všetky riešenia nechávajú celý prevod na užívateľa. Niektoré prístupy sa v splnení svojich cieľov snažia využiť dodatočné informácie plynúce zo schém databáz, či ontológií, respektíve nejakým iným spôsobom mapovanie užívateľovi uľahčiť. Prevod je stále do veľkej miery závislý na zásahoch užívateľa. Daňou za takúto „poloautomatickosť“ sú väčšia komplikovanosť a často širšie požiadavky na tvar vstupných dát.

4.2.1 KAON-REVERSE - reverzné inžinierstvo

V tématike prevodu relačných schém na ontológie veľmi často referencovaným prístupom sa stal prístup prezentovaný Stojanovičovými a Volzom[13]. Pri poloautomatickom tvorení ontológií využíva myšlienky reverzného inžinierstva, a teda hľadaní závislostí tvorenej ontológie z formy tabuliek databázy. To znamená využitie dostupných informácií o atribútoch, typoch atribútov, primárnych a cudzích kľúčoch, či tvaru tabuliek.

Autori približujú celkové riešenie prevodu dát, od databázy až po prevod inštancií. Za vstup sa považuje obyčajný logický databázový model bez akýchkoľvek dotatočných informácií o reláciach na vyššej úrovni (t.z. SQL, žiadny E/R diagram). Výstupným formátom je takzvaný F-Logic[8], z ktorého je možné výsledok skonvertovať do RDF. Prevod dát sa skladá z dvoch fáz. V prvej, mapovacej dochádza k poloautomatickému mapovaniu schémy relácií (tabuliek) na štruktúry ontológií. Druhú fázu tvorí následný prevod samotných konkrétnych relácií na inštancie ontológie. Výsledkom je súbor v RDF popisujúci vzťahy vrámci ontológie a RDF súbor s inštanciami odkazujúci sa na vzťahy pomocou systému menných priestorov.

Základným kameňom prvej fázy, a teda semiautomatického mapovania sú špeciálne pravidlá reverzného inžinierstva, podľa ktorých sa z dostupných informácií o tabuľkách vytvoria triedy, relácie a atribúty. Delia sa na *pravidlá pre tvorbu tried*, *pravidlá na dedičnosť* a *pravidlá pre tvorbu relácií a atribútov*. V skupinách sa postupne všetky uplatnia na všetky tabuľky databázy a to v nasledujúcom poradí[9]:

1. Tvorba tried

   (a) Tabuľka vyjadruje závislosti typu *m:n*, nevytvára sa žiadna trieda (napríklad `ENROL` z Obrázka 3.1).

[8]Model, ktorý vznikol „kvôli kombinácií primitív objektovo orientovaných databáz s logickými jazykmi vyvinutými pre deduktívne databázy“[13].

[9]Najprv sa na všetky tabuľky uplatnia pravidla pre tvorbu tried, následne na všetky tabuľky pravidlo pre dedičnosť a napokon sa prejdú relačné a atribútové pravidlá.

(b) Tabuľka predstavuje len dodatočné informácie pre entitu zachytenú v inej tabuľke (`LECTURERDETAIL` len rozširuje `LECTURER`), nevytvára sa žiadna trieda.

(c) Ak sa neuplatnili predchádzajúce dve pravidlá, vytvorí sa trieda odpovedajúca relácii (`FACULTY`).

2. Hľadanie dedičností

   (a) Ak medzi tabuľkou predstavujúcou triedu a inou tabuľkou predstavujúcou inú triedu existuje inklúzna závislosť, vytvorí sa vzťah dedičnosti (`PhDSTUDENT` a `STUDENT`)[10].

3. Tvorba relácií a atribútov

   (a) Tabuľka neodpovedajúca triede predstavuje spojenie *m:n* dvoch tabuliek už odpovedajúcich triedam, vytvoria sa relácie (oba dva smery) medzi týmito dvoma triedami (`ENROL` spôsobí spojenie `STUDENT` a `TIMETABLE`).

   (b) Ošetrenie vzťahov *1:n*, cudzí kľuč spôsobí vytvorenie relácií medzi dvoma triedami zachytenými v tabuľkách (`STUDENT` a `FACULTY`).

   (c) Tabuľka predstavuje rozšírenie inej tabuľky predstavujúcej triedu, vytvorené atribúty odpovedajúce stĺpcom sú pridané k „základnej“ triede (`LECTURERDETAIL` a `LECTURER`).

   (d) Vzťahy *1:1* medzi tabuľkami predstavujúcimi triedy, vytvorenie relácie (situácia by vznikla, ak by schéma obsahovala navyše napríklad `COURSEMATERIAL(courseID,timetableID,material)` a `COURSELOCATION(courseID,timetableID,location)`)[11].

   (e) Tabuľky reprezentujúce viac než binárne vzťahy, tvorba pomocných tried a viacerých binárnych relácií (`FINALTHESIS`).

   (f) Neuplatnenie posledných štyroch pravidiel znamená vytvorenie odpovedajúceho atribútu všetkým stĺpcom (okrem kľučov) tabuľky.

Pri hľadaní závislostí sa využívajú informácie schémy o cudzích kľučoch, ktoré sú tak nutné. Nakoľko z relačného modelu nevyplýva „orientácia relácií“, tak všetky tvorené relácie sú reflexívne (tvoria sa aj inverzie). Autori z dôvodu neexistencie typov v RDFS zvolili mapovanie typov stĺpcov databázy na samostatné triedy vrámci ontológie (to znamená, že stĺpcu `address` typu `string` odpovedá atribút ontológie smerujúci ku triede `ex:String`).

[10] Aby sa uplatnilo toto pravidlo, nesmie sa na tabuľku uplatniť pravidlo 1b. Preto je potrebný zásah užívateľa, ktorý rozhodne, ktoré pravidlo sa má použiť.

[11] Na zistenie tohto vzťahu musí medzi tabuľkami existovať vzájomná inklúzia kľúčov.

V druhej fáze sa podľa definovaného mapovania vytvoria inštancie, priradia sa im URI a ich vytvorené atribúty (ich nekľúčové stĺpce). Cudzie kľúče následne nájdu uplatnenie pri tvorení relácií medzi vytvorenými inštanciami (využíva sa pomocná funkcia priradzujúca hodnote cudzieho kľuča URI odpovedajúcej inštancie).

Jasnými výhodami takéhoto a podobných prístupov je zrýchlenie a zjednodušenie tvorby ontológií z dát relačnej databázy. Pracnosť manuálnych riešení totiž môže byť pre skutočné rozšírenie semantického webu závažnou prekážkou. Na druhej strane sa však ticho uvažuje databázová schéma vo vyšších normálnych formách, so starostlivo definovanými údajmi o cudzích kľúčoch, čo môže v mnohých reálnych prípadoch spôsobiť problémy. Daňou je i menšie množstvo zachytiteľných situácií (oproti manuálnemu riešeniu využívajúcemu silu SQL typu D2R napríklad nevytvorí triedu odvodenú z kombinácie viacerých tabuliek). Nakoľko ide o poloautomatické nachádzanie mapovania, nad užívanými pravidlami je stále nutný dozor dohliadajúceho užívateľa.

V pôvodnom návrhu ([13]) sa neuvažujú už existujúce ontológie. Autori ale v ďalšej práci ([14]) prezentujú program na automatizáciu mapovacieho procesu KAON-REVERSE. Okrem užívania spomenutých pravidiel reverzného inžinierstva na tabuľky pôvodnej databázy podporuje tento program do istej miery aj „zarovnávanie“ pojmov navrhovanej ontológie s pojmami ontológie už existujúcej. Na to využíva takzvanú lexikálnu vrstvu, ktorá dokáže napríklad „namapovať reláciu Projekt na triedu Project, a to vďaka existencii definície synonymity týchto dvoch reťazcov“[14]. Výsledkom aplikácie podobných pravidiel sú odporúčania pre definovanie odpovedajúcich si pojmov, ktoré užívateľ pri definícii mapovania môže využiť.

4.2.2 FDR2

Využitie reverzného inžinierstva pre uľahčenie prevodu dát databáz do ontológii nie je jedinou možnosťou. Dokazuje to prístup prezentovaný Korotkiym a Topom s názvom FDR2[11], ktorý bol vyvinutý na prenos dát tabuliek špeciálne už do existujúcich ontológii zapísanými v RDFS. Riešenie je oproti reverznému inžinierstvu uplatnenému na tabuľky možno manuálnejšie, nejde však len o ručný zápis pravidiel prenosu v špeciálne vyvinutom jazyku typu D2R alebo R_2O.

Základná myšlienka je jednoduchá. Zo vstupu, ktorou je tabuľka databázy sa vytvorí takzvaná „relačná schéma“, čo je v terminológii prezentovaného riešenia RDFS súbor popisujúci všetky kombinácie (jednostĺpcových) tried a ich atribútov, ktoré môže daná tabuľka prezentovať. To znamená, že každý z názvov stĺpcov tabuľky sa považuje za „kvázitriedu“, ktorú tabuľka môže obsahovať a ku každej takejto triede sú priradené takzvané „virtuálne atribúty“, ktorými sú ostatné stĺpce tabuľky. Konkrétne povedané, ak vstup-

nou tabuľkou je STUDENT(studentID, surname, facultyID), tak vytvorenými kvázitriedami sú triedy fd:studentID, fd:surname, fd:facultyID s odpovedajúcimi si dvojicami virtuálnych atribútov fd:STUDENTID-SURNAME, fd:STUDENTID-FACULTYID,fd:SURNAME-STUDENTID,fd:SURNAME-FACULTYID a fd:FACULTYID-SURNAME s fd:FACULTYID-STUDENTID. Výstupom prvej fázy mapovania sú práve všetky takéto triedy a atribúty zapísané v štandartnom RDFS súbore.

V druhej fáze sú následne vstupmi dva RDFS súbory - vytvorený súbor odpovedajúci tabuľke a súbor s existujúcou ontológiou. Užívateľ osobne určí, ktoré triedy a ktoré atribúty spolu súvisia. V existujúcom webovom rozhraní[12] mu k tomu dokonca pomáha vyhľadávanie jednoduchých lexikálnych podobností medzi zdrojmi v súboroch. Zaujímavé je, že definovanie týchto závislosti sa zapisuje tiež využitím RDFS a to určením podtried a podvlastností ponúkaných týmto jazykom. Pre ilustráciu tak vo výstupnom mapovaciom súbore môže byť trojica fd:studentID rdfs:subClassOf ex:Student, či tiež trojica fd:STUDENTID-FACULTYID rdfs:subPropertyOf ex:isStudentOf defacto mapujúce stĺpce tabuliek na triedy, relácie a atribúty existujúcej ontológie. Podobné trojice môžu postupne vzniknúť ku každej z tabuliek databázy a sú následne dostatočnou informáciou pre prenos jestvujúcích dát prípadným procesorom.

Výhodami popísaného riešenia je jeho jednoduchosť a prehľadnosť. Nutnosť SQL dotazov cez viacero tabuliek v D2R a R_2O pre tvorbu niektorých atribútov či relácií tried sa šikovne obchádza používaním jednoduchých RDF trojíc a odvodzovaním kvázitried z rôznych tabuliek od tých istých tried ontológie. Výhodné je vyhnutie sa špecializovanému novému mapovaciemu jazyku.

Na druhej strane je však daňou príliš vysoká redundancia tvorených pomocných dát v podobe kvázitried a virtuálnych atribútov (pre tabuľku s 10 stĺpcami ide o 10 kvázitried a 90 virtuálnych atribútov), či pracné postupné mapovanie množiny viacerých tabuliek. Systém taktiež síce umožňuje zachytiť elementárne previazanosti tabuliek[13], v iných prípadoch kvôli svojej jednoduchosti ale zlyhá. Ak by napríklad existovala tabuľka RESULT(studentID, courseID, mark), tak pomocou ponúkaného mechanizmu sa nedá povedať, že by vlastne mala odpovedať triede ex:Mark s reláciami ex:ofStudent, ex:fromCourse a atribútom ex:value. Hodila by sa totiž možnosť tvorby kvázitried na základe viacerých stĺpcov.

[12] Aplikácia je prístupná na http://www.cs.vu.nl/~maksym/ .

[13] Ak napríklad tabuľka predstavuje spojenie *m:n* entít obsiahnutých v iných tabuľkách, tak jej príslušné kvázitriedy je možné označiť ako odvodené od tých istých tried existujúcej ontológie ako kvázitriedy odvodené od samotných entít, na ktoré sa stĺpce m:n tabuľky odkazujú.

4.2.3 DartGrid a MAPONTO (LAV pohľady)

Nie pre všetky riešenia predstavuje pojem mapovania aj prevod konkrétnych dát. Naopak, existujú situácie, kedy je vhodnejšie pod mapovaním rozumieť len tvorbu popisu spôsobu, akým odvodzovať inštancie ontológie z dát tabuliek databázy, či databáz. Príkladom môže byť potreba namapovať na existujúcu ontológiu skupinu podobných (ale rôznych) objemovo veľkých a často sa meniacich relačných schém. Inštancie sa v prípade potreby dajú vďaka existencii mapovania získať[14] a jediné, na čo je potrebné reagovať aktualizáciami sú zmeny tvarov schém.

Takýto princíp okrem iných (tu nespomínaný OntoGrate[6]) zvolila aj skupina čínskych autorov pri mapovaní medicínskych dát o liekoch do spoločnej ontológie v projekte DartGrid[10]. Pre svoje potreby integrácie, dotazovania a vyhľadávania rozmanitých dát o liečivách využívajú myšlienku tzv. „Sémantických pohľadov".

Mapovacie pohľady všeobecne môžu byť dvoch typov - globálne GAV (global-as-view) a lokálne LAV (local-as-view). Kým prvý druh definuje každú triedu, reláciu a atribút ontológie ako pohľad nad tabuľkami databázy, druhý to činí úplne naopak. Autori argumentujú, že používanie LAV pohľadov, kedy sú ako pohľady nad štruktúrami ontológie definované tabuľky databázy, je síce pre bežné dotazy nad ontológiou krkolomnejšie, je však rozšíriteľnejšie (pridanie databázy nevyžadaje prebudovanie už tak objemných pravidiel tvorby prvkov ontológie). Sémantické pohľady sú preto v ich riešení typu LAV. Ide o pravidlá tvaru $T(X) : -\Phi(X, Y)$, kde T je relácia relačnej schémy (tabuľka z databázy), X vybrané premenné odpovedajúce jej atribútom (stĺpcom) a Φ konjunktívna formula nad predikátmi ontológie s existenčne kvantifikovanými premennými Y.

V prostredí databázy z Obrázka 3.1 a ontológie z Obrázka 3.2 by napr. pohľad pre tabuľku `STUDENT(studentID,firstname,surname,facultyID)` vyzeral v tomto zápise približne takto[15]:

$$STUDENT(sID, fn, sn, fID) \text{ :- } ex{:}Student(y_1), ex{:}sfirstname(y_1, fn),$$

$$ex{:}ssurname(y_1, sn), ex{:}Faculty(y_2), ex{:}isStudentOf(y_1, y_2)$$

V projekte DartGrid sa podobné pohľady vytvárajú manuálne užívateľom v grafickom nástroji DartMapping[10]. Ako taký by teda tento projekt mal byť zaradený skôr v kapitole manuálnych mapovaní. Dôvodom zaradenia medzi semiautomatické riešenia sú myšlienky poloautomatického hľadania pohľadov ďalších autorov, ktoré na prácu typu DartGrid nadväzujú, menovite práce Ana, Borgidu a Mylopoulosa ([1] a [2]) o nástroji MAPONTO.

[14] Jednou z elegantných vlastností takýchto prístupov je možnosť transformovácie dotazov nad ontológiou na SQL dotazy príslušných databáz, ktorých výsledky sa následne transformujú späť a vrátia v požadovanej forme odpovede na dotaz na ontológiu.

[15] Zápis nie je pre ilustratívnosť úplne exaktný.

Autori predstavujú nástroj na „pomoc užívateľom v hľadaní LAV pohľadov medzi relačnými databázami a ontológiami“[1]. Predpokladom sú užívateľom dopredu zadané jednoduché závislostí medzi stĺpcami tabuliek a atribútmi ontológie (napr. že stĺpec `surname` tabuľky `STUDENT` odpovedá atribútu `ex:ssurname` triedy `ex:Student`). Základom je algoritmus (prezentovaný v [2]) následne hľadajúci v grafe tabuľke odpovedajúcich prvkov ontológie „rozumné“ minimálne kostry, ktoré tak predstavujú hľadané pohľady. Prebieha v dvoch fázach – „najprv spojí triedy odpovedajúce kľúčom do tzv. kostrových stromov a následne pripojí ostatné uzly odpovedajúce ostatným stĺpcom do kostry s preferovaním funkčných relácií[16]“[1]. Vo svojích pravidlách navyše využíva napriek ich neprítomnosti princípy budovania relačných schém z E/R diagramov.

Za podmienky vytvorenia databázy podľa týchto princípov je medzi vrátenými „rozumnými kostrami“ aj kostra so správnou sémantikou. Testovaním na reálnych dátach autori ukazujú, že ak aj automaticky vytvorené pohľady neodpovedajú užívateľovým predstavám, deje sa tak iba zčasti a prípadné následné manuálne zmeny sú výrazne uľahčené.

Výhodami LAV pohľadov je ich veľká pružnosť umožňujúca nasadenie v komplexnejších situáciach napríklad s viacerými databázami. Na druhej strane sú ale zložitejšie. Na ich tvorbu môže byť využitá heurestika, tá je ale zložitejšia než napríklad bežné reverzné inžinierstvo nad kľúčmi tabuliek a tiež nie je stopercentná.

4.3 Zhrnutie a dôsledky prehľadu

Už aj v relatívne mladej tématike mapovania relačných schém na ontológie evidentne existujú rôzne prístupy. Jeden od druhého sa líšia v mnohých aspektoch od účelu cez spôsob mapovania až po úroveň automatiky. Z toho plynú ich výhody a tiež nedostatky.

Vysoká expresivita jazykov D2R a (predovšetkým) R_20 je potlačená zložitosťou ich prípadného automatického generovania. Okrem podobných manuálnych vznikli tiež rôzne riešenia snažiace sa do istej miery mapovanie užívateľovi uľahčiť. Či už odhadovaním sémantiky databáz pomocou reverzného inžinierstva, istého grafového algoritmu vytvárajúceho lokálne pohľady na tabuľky, alebo inak. Šikovný je prístup FDR2, má však viaceré (možno ale prekonateľné) nedostatky. Spomedzi ostatných sa kúsok vymyká hlboká anotácia v Ontomat-Annotizer, ktorá tak iba dokazuje rôznorodosť podmienok a cieľov mapovania. V prehľade by mohli byť mnohé ďalšie podobné (napríklad METAmorphoses[16]), či rozdielne prístupy (komplexný OntoGrate[6]).

Vo všeobecnosti sa ukazuje, že čím menšie počiatočné podmienky sú

[16]V MAPONTO sa počíta s vyššími ontológiami umožňujúcimi vyjadrenie kardinalít reláciam (a tak v špeciálnych prípadoch existenciu funkcií medzi triedami).

na databázu, ontológiu a ich podobnosť kladené, tým komplikovanejšia, až nemožná je snaha o čo najväčšie zautomatizovanie prevodu, a teda tým nutnejšia je aj nejaká forma manuálnej definície, či interaktívnej obsluhy. Na druhej strane však aj napriek tomu, že medzi tabuľkami a existujúcimi ontológiami „automatické nástroje ani pri najväčšej snahe nedokážu všeobecne vygenerovať stopercentne presné a vykonateľné mapovania len na základe sémantiky dát“[6], je ale pomoc užívateľom rôznymi heurestikami užitočná.

V ďalšej kapitole navrhujem vlastné riešenie. Vrámci možností rozsahu diplomovej práce sa (do istej miery) spomenutými riešeniami inšpiruje a poúča.

Kapitola 5

Navrhnuté riešenie mDR

5.1 Požiadavky na riešenie

Okrem prehľadu existujúcich riešení je cieľom tejto práce tiež navrhnutie vlastného riešenia nimi poučeného. V súlade s kapitolou o vymedzení problému práce sú na neho kladené nasledujúce vstupné podmienky a žiadané vlastnosti:

- Vstupmi sú databázová schéma na úrovni SQL a existujúca schéma ontológie v RDFS.
- Databáza neporúša prvú normálnu formu a môže informovať o cudzích kľučoch.
- Schémy databázy a ontológie nie sú diametrálne odlišné[1].
- Riešenie by malo byť schopné čeliť štandartným situáciam, napríklad Obrázkov 3.1, 3.2 a 3.3.
- Žiadaným výstupom je dokument popisujúci spôsob prevodu databázy na ontológiu na základe ktorého môže dôjsť k samotnému prenosu inštancií, poprípade ktorý by mohol byť využitý na prípadné iné aplikácie.
- Návrh musí umožniť jednoduchší vývoj pomocného obslužného grafického nástroja, ktorý je predmetom záverečnej časti práce.

Požiadavky vychádzajú z možných najbežnejších situácií menších systémov, na ktoré sa táto práca z rozsahových dôvodov zameriava. Nesmerujú tak k nasadeniu v širokých komplexných systémoch, ale v jednoduchších prostrediach s potrebou pridania sémantiky tabuľkám databázy.

[1] Ide o veľmi vágnu definíciu, no nie je účelom a ani nutné exaktne definovať podobnosť schém.

5.2 Dôvody zvoleného riešenia

Kapitola o existujúcich riešeniach odkryla možné cesty k splneniu požiadaviek. Mnohé z vlastností jestvujúcich prístupov boli určené mierou automatiky. Základnou otázkou preto aj pre navrhované riešenie je, do akej miery v riešení umožniť užívateľovi pomoc v podobe semiautomatizmu. Nájsť spôsob pomáhajúci stopercentne je nemožné. Používanie čiste manuálnych riešení je naopak zdĺhavé. Nástroj by však do istej miery mohol užívateľovi pomáhať celkom rozumne i pri nízkych nárokoch na riešenie. Ďalej prezentované riešenie tak bude slabo poloautomatické.

Kvôli jednoduchosti sa v tom zdá byť rozumné využiť najmä najprístupnejšie informácie. Takými sú informácie o tabuľkách a primárnych či cudzích kľúčoch, čo znamená aplikovanie jednoduchšieho reverzného inžinierstva. Užitočným sa ukáže byť i zúžovanie ponúkaných možností mapovania. Prístupná by bola tiež možnosť jemného lexikálneho porovnávania názvov a možných popisov tabuliek, stĺpcov databázy s prvkami ontológie.

Čo sa týka spôsobu mapovania schém, prvou možnosťou by bolo vytvoriť systém (a následne grafický program) na generáciu zápisov v jazykoch typu D2R a R_2O. Takýto systém by mal nesporné výhody prameniace v sile SQL, no bol by ťažkopádny, a to najmä pri snahe o využitie reverzného inžinierstva. Kvôli menšej komplexite na druhej strane nie je nutné generovanie mapovania na spôsob LAV pohľadov. Poloautomatika by mohla byť príliš zložitá (viď [2]). Schodnou cestou sa preto zdá byť inšpirovanie niečím jednoduchým, no stále relatívne silným.

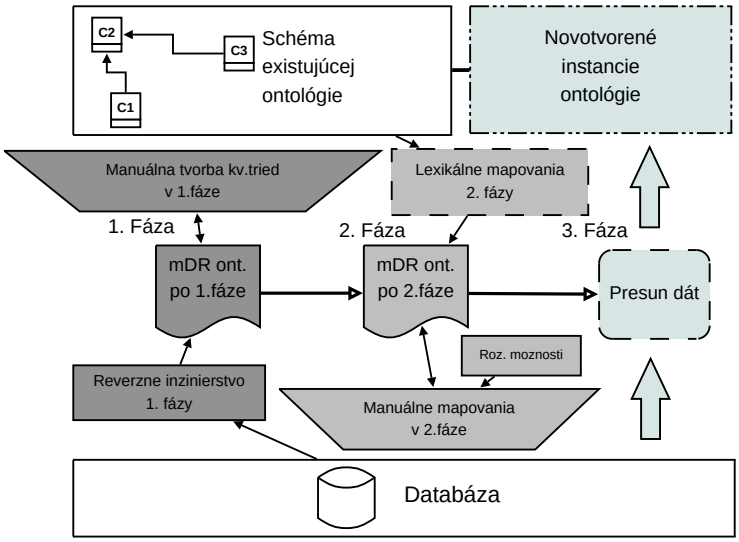
Takým riešením bolo FDR2. Jeho jednoduchosť a šikovnosť plynula z nepoužívania ničoho iného než RDF. Mapovanie niektorých z ním vygenerovaných kvázitried a virtuálnych relácii i atribútov pomocou `rdfs:subClassOf` a `rdfs:subPropertyOf` elegantne obchádza nutnosť zložitejších konštrukcií pri malých vyjadrovacích stratách. Niektoré bežné situácie však FDR2 zachytiť schopný nie je a navyše sa vyznačuje vysokou redundanciou.

Navrhnuté riešenie s názvom mDR sa zo spomenutých dôvodov inšpiruje najmä eleganciou využitia RDF v FDR2, je však všeobecnejšie a obchádza jeho nedostatky. Nezanedbateľnou príčinou orientácie na RDF tiež je, že pomocný nástroj môže využívať pre neho už vyvinuté funkčné nástroje. Pri mapovaní mDR do istej miery uplatní myšlienky reverzného inžinierstva, rozumného zúžovania ponúkaných možností a poprípade aj hľadania lexikálnych podobností medzi schémami.

5.3 Princíp mDR

Základným stavebným kameňom prístupu je špeciálne navrhnutý *mDR slovník* (*mDR schéma*). Je zapísaný v RDFS a obsahuje triedy a relácie, ktoré

dokážu popísať vzájomné vzťahy stĺpcov tabuliek *vstupnej databázy* a ich súvislosť so *vstupnou ontológiou*[2]. Konkrétnym významom prvkov mDR slovníka sa venujú nasledujúce podkapitoly, v tomto bode je dôležité, že úlohou riešenia je z veľkej časti práve vytvorenie správnej *mDR ontológie*[3] s prvkami mDR slovníka a jej následné využitie.



Obrázok 5.1: mDR

Ako sa tak deje zachycuje názorne Obrázok 5.1. Tvorba mDR dokumentu je rozdelená do dvoch fáz. V *prvej* dôjde k načítaniu informácií o tabuľkách a ich stĺpcoch z databázy. Následne sa ne ňu aplikujú pravidlá zohľadňujúce predovšetkým primárne a cudzie kľúče a vzniká predbežná verzia mDR ontológie. Užívateľ má možnosť ju manuálne ovplyvniť. Aplikovaním poloautomatiky a zásahu užívateľa vznikne definitívna ontológia mDR prvej fázy. Dokument obsahuje istú reprezentáciu stĺpcov databázy.

Úlohou *druhej fázy* je vytvorený mDR prvej fázy prispôsobiť existujúcej vstupnej ontológii. Užívateľ porovnáva mDR ontológiu so vstupnou ontológiou a určením podtried a podrelácií medzi dokumentami definuje mapovanie a defacto taktiež rozhodne o nepotrebnosti niektorých prvkov prvej verzie. Do istej miery mu nástroj môže pomáhať ponúkaním len rozumných možností[4]. mDR je navyše navrhnuté tak, že dovoľuje využitie dodatočných

[2]Kvôli odlíšeniu od mDR ontológie sa existujúca ontológia, na ktorú je cieľom databázu mapovať ďalej nazýva *vstupná ontológia*.

[3]V navrhnutom riešení sa pojmom mDR nazýva viacero vecí. *Slovník mDR* (resp. *Schéma mDR*) je ďalej predstavená vyvinutá skupina všeobecných tried a vlastností so špeciálnymi významami pre mapovanie. Tie využíva *Ontológia mDR*. Ňou sa rozumie konkrétna ontológia RDFS s prvkami mDR slovníka popisujúca mapovanie konkrétnej databázy. Výraz *mDR* taktiež dáva názov celému riešeniu.

[4]O tom bližšie kapitola o druhej fáze a o vlastnom nástroji.

informácií (lexikálnych podobností) pre poloautomatickú pomoc aj v tejto fáze. Vzniká druhá a definitívna verzia mDR ontológie definujúca mapovanie medzi vstupmi. Dokument je dostatočnou informáciou pre prípadnú *tretiu fázu*. V nej môže dôjsť k presunu konkrétnych inštancií, poprípade v nej môžu byť pravidlá o mapovaní využité inak.

Nasledujúce časti tejto kapitoly sa venujú formátu mDR slovníka (teda aj mDR ontológie) a podrobnému opisu priebehu všetkých fáz. Záver je venovaný kladným vlastnostiam a tiež obmedzeniam navrhnutého riešenia.

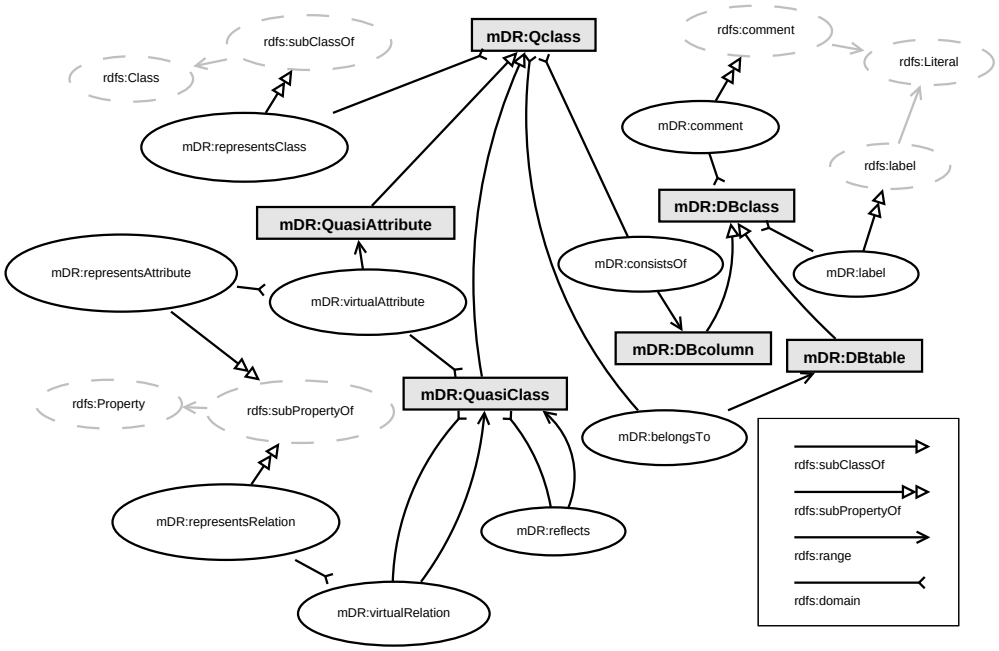
5.4 mDR slovník

Každý dvojfázovo tvorený súbor ontológie mDR popisuje konkrétne mapovanie určením svojich zdrojov pomocou `rdf:type` ako inštancií špeciálnych tried a vlastností definovaných v mDR slovníku. Úplne znenie definície slovníka sa nachádza v Dodatku A. Zjednodušený grafový tvar zachycuje Obrázok 5.2. Mapovací význam prvkov je vysvetlený v nasledujúcich riadkoch.

Nech je daná vstupná databáza. Hlavnou myšlienkou mDR je sústava navzájom pospájaných takzvaných kvázitried, kváziatribútov, virtuálnych atribútov a relácií. *Kvázitriedou* sa rozumie skupina stĺpcov (pokojne aj jednoprvková) niektorej tabuľky databázy, ktorá má predpoklady na to, aby mohla odpovedať niektorej triede vstupnej ontológie. V tabuľke `STUDENT` Obrázka 3.1 je vhodným kandidátom samostatný stĺpec `studentID`, v `RESULT` zasa dvojica `studentID` a `courseID` a tiež samostatný `studentID` a samostatný `courseID`. Pre každú takúto kvázitriedu sa v príslušnej mDR ontológii vytvorí samostatný zdroj zaradený do triedy `mDR:QuasiClass`. Kvázitrieda odpovedajúca primárnemu kľúču `studentID` zo `STUDENT` je rozdielna od kvázitriedy rovnomenného stĺpca tabuľky `RESULT`. Keďže ale spolu „evidentne súvisia“, tak budú v druhej fáze namapované pravdepodobne na rovnakú triedu vstupnej ontológie. Informácia o takejto previazanosti (typicky vznikajúcej pri cudzích a odpovedajúcich primárnych kľúčoch) sa využije neskôr pri prenose inštancií (vyjadruje vlastne ich totožnosť) a vyjadrí sa vzájomnou reláciou `mDR:reflects`.

Ak má vzťah medzi dvoma kvázitriedami jednej tabuľky (typicky medzi stĺpcami primárneho a cudzieho kľúča spoločnej tabuľky) predpoklady na namapovanie na určitú reláciu medzi potenciálne odpovedajúcimi triedami vstupnej ontológie, spoja sa *virtuálnou reláciou*[5]. Príkladom je možná výhodnosť existencie virtuálnej relácie medzi kvázitriediami odpovedajúcimi samostatnému `studentID` a dvojici `studentID` s `courseID` tabuľky `RESULT` (ktorá by mohla byť namapovaná na reláciu `ex:hasMark` medzi triedami

[5]Obmedzenie na jednu tabuľku nie je prekážkou, keďže vzťahy rôznych tabuliek sú tak či tak vyjadrené kľučmi v spoločnej tabuľke.



Obrázok 5.2: mDR ontológia

`ex:Student` a `ex:Mark` ontológie z Obrázka 3.2). Virtuálna relácia sa vyjadrí reláciou mDR slovníka `mDR:virtualRelation`.

Než triede samotnej, môže mať skupina stĺpcov istej tabuľky lepšie predpoklady na odpovedanie určitému atribútu triedy. V takom prípade je zdroj reprezentujúci skupinu stĺpcov inštanciou triedy `mDR:QuasiAttribute`, daná skupina stĺpcov tabuľky je takzvaným *kváziatribútom*. Typicky ide o skupinu jedného stĺpca, príkladom by mohol byť stĺpec `mark` tabuľky `RESULT`. Ako taký však kváziatribút nie je nijako spojený so žiadnou kvázitriedou. Aby sa vedelo, ku akej kvázitriede skupina stĺpcov patrí, musí byť príslušná kvázitrieda spojená s daným kváziatribútom takzvaným *virtuálnym atribútom*, čo je (v rámci mDR) názov špeciálnej relácie spájajúcej inštancie kvázitried s inštanciami kváziatribútov. Virtuálnym atribútom sa nazýva z dôvodu, že pri možnom neskoršom mapovaní sa na atribút triedy (ktorej stĺpce kváziatribútu odpovedajú) mapuje práve táto relácia[6]. Existencia virtuálneho atribútu sa zapíše vytvorením zdroja typu `mDR:virtualAttribute`.

Ako vyzerá konkrétna mDR ontológia využívajúca mDR slovník približuje Príklad 5.1. Podstatnou charakteristikou mDR mapovania je, že pre

[6]RDFS nerozlišuje medzi atribútmi a reláciami, všetko sú tzv. properties, teda vlastnosti, z ktorých niektoré sú medzi triedami a niektoré medzi triedami a literálmi. Kvôli názornosti sa práca napriek tomu drží rozlišovania medzi pojmami relácia a atribút. Problém nastáva pri miešaní významov prvkov v mDR ontológii a v pôvodnej ontológii. Ak je niečo kvázi, alebo virtuálne, znamená to, čomu by to v pôvodnej ontológii malo odpovedať a nie čím to v mDR ontológii naozaj je. Technicky je v mDR napríklad `mDR:QuasiAttribute` triedou a `mDR:virtualAttribute` reláciou, ich účelom je ale odpovedanie atribútu.

```
qua:RESULT_studIDcourID           rdf:type      mDR:QuasiClass
qua:RESULT_studID                 rdf:type      mDR:QuasiClass
qua:RESULT_studID                 mDR:reflects  qua:STUDENT_studID
qua:RESULT_studID_hasstudIDcourID rdf:type      mDR:virtualRelation
qua:RESULT_studID_hasstudIDcourID rdfs:domain   qua:RESULT_studID
qua:RESULT_studID_hasstudIDcourID rdfs:range    qua:RESULT_studIDcourID
qua:RESULT_mark                   rdf:type      mDR:QuasiAttribute
qua:RESULT_studIDcourID_hasmark   rdf:type      mDR:virtualAttribute
qua:RESULT_studIDcourID_hasmark   rdfs:domain   qua:RESULT_studIDcourID
qua:RESULT_studIDcourID_hasmark   rdfs:range    qua:RESULT_mark
```

Príklad 5.1: Zápis kvázitried, kváziatribútov a virtuálnych atribútov

virtuálne atribúty a relácie sa tvoria samostatné zdroje, inštancie príslušných mDR vlastností, čo bežne typické nie je[7]. Úlohou prvej fázy je definovať všetky možné zaujímavé kvázitriedy a kváziatribúty (bude ich ale podstatne menej než v FDR2). K zrovnávaniu pojmov so vstupnou ontológiou dochádza až vo fáze druhej. Aby bolo možné špecifikovať, ktoré vzájomné vzťahy sa na relácie a atribúty vstupnej ontológie namapujú, je nutné ich vedieť rozlíšiť. To je umožnené inštancianizáciou virtuálnych atribútov a relácií v prvej fáze. Bez nej by k definovaniu kváziprvkov a mapovaniu muselo dochádzať inak a naraz.

Pre prípadný prenos inštancií je nutné vedieť, ktoré stĺpce databázy kváziatribútový alebo kvázitriedový zdroj zahŕňa. Kvôli tomu sa pre všetky tabuľky (a analogicky stĺpce) databázy v rodiacej sa mDR ontológii vytvoria ďalšie samostatné zdroje, inštancie triedy `mDR:DBtable` (`mDR:DBcolumn` pre stĺpce). S príslušnými kvázitriedami a kváziatribútmi sa spoja reláciami `mDR:belongsTo` a `mDR:consistsOf` slovníka mDR. Triedy majú atribúty `mDR:label` a `mDR:comment`[8], do ktorých sa k nim ukladá ich textový názov a ich dodatočný popis z databázy, ak taký existuje. Tieto informácie môžu byť využité pri lexikálnom porovnávaní prvkov mDR ontológie s prvkami vstupnej ontológie. Názorným príkladom je Príklad 5.2.

```
qua:RESULT_studIDcourID   mDR:belongsTo   db:RESULT
qua:RESULT_studIDcourID   mDR:consistsOf  db:RESULT.studID
qua:RESULT_studIDcourID   mDR:consistsOf  db:RESULT.courID
db:RESULT                 mDR:label       "RESULT"
db:RESULT                 mDR:comment     "Student marks table"
db:RESULT.studentID       mDR:label       "studentID"
db:RESULT.studentID       mDR:comment     "Foreign key to STUDENT table"
```

[7]V RDFS sú takéto netradičné veci povolené. Veľa vecí, vrátane `rdf:Property` je z definície vlastne triedou, inštanciou `rdfs:Class`. Zdroje, ktorým sa nastaví `rdf:type` ako `rdf:Property` sú tak inštanciami triedy obsahujúcej zdroje, ktorými sa myslia vlastnosti medzi triedami. Inštancia inštancie, trieda, ktorá je svojou inštanciou alebo podvlastnosť podvlastnosti sú všetko možné konštrukcie. Bližšie definícia RDFS slovníka[19].

[8]Atribúty sú potomkami všeobecnejších `rdfs:label` a `rdfs:comment`. Odlíšené sú kvôli elegancii a naznačeniu, že v mDR majú svoj vlastný význam.

```
db:RESULT.courseID        mDR:label       "courseID"
```

Príklad 5.2: Informácie o pôvode kvázitried a kváziatribútov

Do tohto bodu boli vysvetlené prvky slovníka využívané v prvej fáze tvorby konkrétnych mDR ontológií. Jej úlohou je popísať zaujímavé skupiny stĺpcov tabuliek, ich vzájomné vzťahy, miesto kde ich získať a tiež viaceré neskôr nápomocné doplnkové informácie. K určovaniu odpovedania si kvázitried a tried, virtuálnych relácii a relácii, či virtuálnych atribútov s kváziatribútmi a atribútmi, a teda k mapovaniu ako takému dochádza vo fáze druhej. V nej sa pre tento účel do dokumentu medzi prvky pôvodnej mDR a vstupnej ontológie pridajú doplnkové relácie `mDR:representsClass`, `mDR:representsAttribute` a `mDR:representsRelation`. Významom týchto relácii je poddanosť prvkov, čo značí aj ich odvodenie od `rdfs:subClassOf` a `rdfs:subPropertyOf`. Príkladom takéhoto mapovania je Príklad 5.3.

```
qua:RESULT_studIDcourID           mDR:representsClass     ex:Mark
qua:RESULT_studID                 mDR:representsClass     ex:Student
qua:RESULT_studID_hasstudIDcourID mDR:representsRelation  ex:hasMark
qua:RESULT_studIDcourID_hasmark   mDR:representsAttribute ex:value
```

Príklad 5.3: Mapovanie na triedy, atribúty a relácie

Riešenie mDR vychádza z princípov FDR2. Systém mapovania je veľmi podobný. V mDR ale dochádza ku tvorbe len rozumných, nie všetkých možných kombinácií kvázitried a virtuálnych atribútov. FDR2 kvôli svojej jednoduchosti žiaden špeciálny slovník nedefinoval. Navrhované mDR však pridaním možností kvázitried viacerých stĺpcov (pokryjú napríklad situácie viac než binárnych relácií), vzájomného si odpovedania kvázitried a informácií o pôvodoch stĺpcov princíp mapovania zovšeobecňuje. Z tohto dôvodu už vyžaduje špeciálny presne definovaný slovník - slovník mDR (Obrázok 5.2), ktorého triedam a vlastnostiam patria všetky zdroje každej tvorenej mDR ontológie. Ich vzájomne vzťahy a tvar typického mDR dokumentu by mali byť v tento okamih jasnejšie (doslovnú definíciu obsahuje Dodatok A). Ako sa mapovací dokument vytvorí, približujú nasledujúce dve podkapitoly.

5.5 Prvá fáza tvorby mDR ontológie

Jediným vstupom prvej fázy je vstupná databáza. Úlohou je vytvoriť prvú verziu mDR dokumentu využívanú pri mapovaní v spojení so vstupnou ontológiou vo fáze druhej. Prvá verzia obsahuje kvázitriedy, kváziatribúty, virtuálne relácie a virtuálne atribúty, ich vzájomné prepojenia a doplnkové informácie o stĺpcoch a tabuľkách databázy. Jediné, čo neobsahuje, sú mapovacie relácie `mDR:representsClass`, `mDR:representsAttribute` a `mDR:representsRelation` tvorené neskôr. Tvorba prebieha v tejto fáze poloautomaticky. Na databázovú schému sa aplikujú pravidlá automatickej

tvorby prvkov mDR ontológie, ktorých výsledok môže obsluhujúci užívateľ následne ručne pozmeniť.

Prvým úkonom je pripojenie sa na databázu a získanie z informácií o tabuľkách (poprípade aj o pohľadoch), kľúčoch a popisoch, teda čisto technická záležitosť. Zaujímavý je až spôsob aplikovania automatických pravidiel reverzného inžinierstva:

1. Úvodom sa prejdú všetky tabuľky, pre každú sa automaticky pre skupinu stĺpcov primárneho kľúča (aj viacerostĺpcového) vytvorí kvázitrieda. Výnimkou sú tabuľky typu *m:n*, ktoré obsahujú len dva rôzne cudzie kľúče (odkazujúce na rozdielne tabuľky), v takom prípade sa pre primárny kľúč (ktorým sú všetky stĺpce) kvázitrieda automaticky nevytvára. V ilustratívnej databáze Obrázka 3.1 sa napríklad vytvorí kvázitrieda odpovedajúca dvojici `studentID` a `courseID` v `RESULT`, všetkým stĺpcom vo `FINALTHESIS`, samotnému stĺpcu `timetableID` v `TIMETABLE`, ale naopak žiadna kvázitrieda pre tabuľku `ENROL`.

2. Tabuľky sa prejdú opäť. Pre každú skupinu stĺpcov rovnakého cudzieho kľúča sa vytvorí kvázitrieda odkazujúca sa na kvázitriedu odpovedajúceho primárneho kľúča (ak tá existuje) pomocou `mDR:reflects`. V tabuľke `STUDENT` je takou skupinou samostatný stĺpec `facultyID`, vo `FINALTHESIS` a `ENROL` všetky stĺpce samostatne, v `RESULT` stĺpce `studentID` a `courseID` samostatne.

3. Dosiaľ nedotknuté ostatné stĺpce tabuliek sú kváziatribútmi a pomocou virtuálneho atribúta sa spoja s kvázitriedou primárneho kľúča danej tabuľky. K stĺpcu `mark` tabuľky `RESULT` sa vytvorí kváziatribút spojený s kvázitriedou odpovedajúcou dvojici stĺpcov `studentID` a `courseID`, ktorá je primárnym kľučom tabuľky.

4. Na záver sa medzi vhodnými kvázitriedami dotvoria virtuálne relácie. Pre každú tabuľku sa z nej odvodená kvázitrieda primárneho kľúča spojí virtuálnou reláciou s ostatnými kvázitriedami tejto tabuľky. Každá relácia sa vytvorí aj v inverznej podobe. V tabuľke `FINALTHESIS` sa pre názornosť spojí kvázitrieda odpovedajúca všetkým stĺpcom s kvázitriedami odpovedajúcimi jednotlivým stĺpcom a opačne.

Aplikovaním týchto pravidiel sa vytvorí „rozumná“ - schopná a nie príliš veľká množina inštancií prvkov mDR slovníka. V prostrediach bežných a štandartných databázových schém bude často aj požadovanou verziou mDR ontológie prvej fázy. Aplikovanie pravidiel na databázu Obrázka 3.1 je toho príkladom. Žiadaný výstup prvej fázy je pre ňu uvedený v Dodatku B. Pre mapovanie na ontológiu Obrázka 3.2 obsahuje všetky potrebné kvázitriedy, virtuálne relácie a atribúty. Jediný menší problém nastane pri otázke, čo

(ak vôbec) namapovať na `ex:teaches` (a `ex:isTaughtBy`). Priamy vzťah medzi učiteľmi a kurzami v databáze neexistuje, čiastočne by sa ale mohol využiť vzťah medzi `lecturerID` a `courseID` z tabuľky `TIMETABLE`, alebo `FINALTHESIS`, alebo dokonca oboch naraz. Medzi odpovedajúcimi kvázitriedami ale virtuálna relácia automaticky vytvorená nebola. Riešením by mohla byť automatická tvorba virtuálnych relácií medzi všetkými kvázitriedami tabuľky, vznikalo by však veľa neskôr nepodstatných vzťahov. Iné riešenie ponúka možnosť manuálnej úpravy výsledku aplikovania automatických pravidiel. Užívateľ si pre každú tabuľku môže ručne pozmeniť, ktoré ďalšie skupiny stĺpcov by mohli byť kváziatriedami či kváziatribútmi[9], a kde medzi nimi ešte dotvoriť virtuálne relácie a atribúty. Tak by potom mohol určiť, že stĺpcom `courseID` a `lecturerID` tabuliek `TIMETABLE` a `FINALTHESIS` odpovedajú aj samostatné kvázitriedy a medzi nimi zadefinovať virtuálne relácie oboma smermi. Tie by sa v druhej fáze namapovali na `ex:teaches` a `ex:isTaughtBy`.

Manuálnym zásahom sa dá v prípade potreby podobným spôsobom ovplyvniť každý výstup prvej fázy mDR riešenia[10] . Keďže prezentovaná automatika nie je na škodu a zvyčajne vytvorí vhodné prvky, sú ručné zmeny rozumné až po aplikovaní reverzného inžinierstva. Ak z nejakého dôvodu databázová schéma neobsahuje informácie o kľúčoch, požadovanú mDR ontológiu prvej fázy môže stále vytvoriť užívateľ manuálne. Inak a obyčajne je ale tvorba mDR ontológie prvej fázy určite rýchlejšia, pohodlnejšia a poloautomatická.

5.6 Druhá fáza tvorby mDR ontológie

Vstupom druhej fázy je už dokument obsahujúci kvázitriedy, virtuálne relácie a atribúty s predpokladmi na namapovanie na prvky vstupnej ontológie. „Zliatie“ prebieha pomocou mDR vlastností `mDR:representsClass`, `mDR:representsAttribute` a `mDR:representsRelation`, o ktoré sa vstupný mDR dokument obohatí. Všetky sú potomkami bežných RDFS vlastností `rdfs:subClassOf` a `rdfs:subPropertyOf` a doslova tak vyjadrujú fakt, že ich vstupné kvázi a virtuálne prvky sú potomkami prvkov vstupnej ontológie.

Mapovanie je rozdelené do menších krokov:

[9]Poprípade naraz oboma, čo by mohlo byť využiteľné pri primárnych kľúčoch typu rodného čísla, kedy sa identifikátor okrem rozlíšenia aj reálne používa a tak môže byť atribútom nejakej triedy vstupnej ontológie.

[10]Pre úplnosť by sa podobne postupovalo aj pre tvorbu mDR dokumentu vhodného pre ontológiu Obrázka 3.3. Problémy s touto ontológiou sa viac ukážu až pri mapovaní v druhej fáze, kedy bude potrebné rozhodnúť, čo namapovať na `ex:attends`, keďže priamy vzťah medzi študentom a kurzom vyjadrujúci, čo študent navštevuje, neexistuje. O tom ale až ďalšia podkapitola.

1. Niektoré z kvázitried, ktoré neodpovedajú žiadnej inej kvázitriede (nemajú nastavené `mDR:reflects`) sa mapujú na vhodné, odpovedajúce triedy vstupnej ontológie pomocou `mDR:representsClass`.

2. Mapovanie kvázitried reflektujúcich iné kvázitriedy. Často požadovaným správaním je namapovanie na rovnaké cieľe ako reflektované kvázitriedy.

3. Tretím krokom je mapovanie kváziatribútov a virtuálnych atribútov. Vhodnými odpovedajúcimi atribútmi cieľovej ontológie môžu byť pravé atribúty s literálovým oborom hodnôt, alebo v skutočnosti relácie smerujúce k iným triedam[11]. V oboch prípadoch sa na atribút (alebo reláciu) pomocou `mDR:representsAttribute` namapuje inštancia virtuálneho atribútu. Kváziatribút by sa mohol namapovať na obor hodnôt atribútu-relácie vstupnej ontológie. Keďže ale vo výslednom mDR dokumente ostanú aj príslušné (s virtuálnymi atribútmi spojené) kváziatribúty, tak by ale takéto mapovanie neprinieslo žiadnu novú informáciu a z tohto dôvodu sa teda nekoná.

4. Na záver sa pomocou `mDR:representsRelation` na relácie existujúcej ontológie mapujú virtuálne relácie.

Pri mapovaní by bolo výhodne uplatniť istým spôsobom hľadanie lexikálnych podobností medzi tabuľkami, stĺpcami a prvkami vstupnej ontológie. Pri jednotlivých krokoch by sa užívateľovi mohli do popredia dostávať možnosti viac pravdepodobné (pre mapovanie stĺpca `studentID` tabuľky `STUDENT` najprv trieda `ex:student`, pre `mark` s popisom "Value of result" tabuľky `RESULT` podobne atribút `ex:value` triedy `ex:Mark` a podobne). Využité by mohli byť už vyvinuté metriky vzdialeností textových reťazcov. Aj keby bolo jasné, akú metriku použiť, nie je jasné, ktoré hodnoty pri porovnávaní porovnávať a prečo. Otázka odporúčania mapovaní na základe lexikálnych (alebo iných) podobností prvkov je aj ako samostatná príliš rozsiahla a je mimo rozsah tejto práce. Riešenie mDR vytvára špeciálnymi prvkami mDR slovníka pre prípadné uplatnenie takéhoto mapovania vstupné podmienky, v tejto podobe sa mu ale ďalej nevenuje. Mapovanie je tak v druhej fáze dnes len manuálne.

Nástroj však môže užívateľovi do istej miery pomáhať aj napriek neexistencii hľadania lexikálnych podobností. Pri ponúkaní možností na mapovanie totiž môže využívať informácie z už definovaných mapovaní (z tohto dôvodu prebieha mapovanie v uvedenom poradí). Pre kvázitriedy reflektujúce iné kvázitriedy zobrazením len tried, na ktoré je namapovaná „otcovská“ kvázitrieda. Pri virtuálnych reláciach a atribútoch je rozumné zobraziť len relácie

[11] Príkladom môže byť, ak vstupná ontológia obsahuje triedu `ex:String`, ktorá je oborom hodnôt atribútov (technicky ale vlastne už relácií) niektorých tried.

a atribúty, u ktorých doména a obor hodnôt je mapovaná z kvázitried domény a oboru hodnôt konkrétnej virtuálnej relácie, či atribútu. V takom prípade však nástroj vrámci vstupnej ontológie musí brať do úvahy dedičnosť zdrojov (pri Obrázku 3.3 napríklad pri atribúte `ex:firstname` triedy `ex:Student`). Vlastnosťou RDFS je, že aj z jednoducho definovanej ontológie môžu vďaka definovaným inferenciám plynúť ďalšie dodatočné informácie. Je však otázne, do akej miery ich v dokumente hľadať. Hlbokým skúmaním sa dá zistiť mnoho svet komplikujúcich informácií[12]. Preto spôsob, akým nástroj k dedičnostiam a pomáhaniu pristúpi je na ňom[13]. Z dôvodu možnosti definície neštandartných mapovaní je ale vhodné ponúknúť i nepomáhanie rozumnými možnosťami.

V ontológiach viacmenej odpovedajúcim tabuľkám databázy by druhá fáza mala byť priamočiara. V ontológii Obrázka 3.2 nastáva pri mapovaní výsledku prvej fázy jediný problém, a to spomenutý problém s `ex:teaches` a `ex:isTaughtBy`. Keďže ale tento vzťah databáza priamo nevyjadruje, zdá sa rozumné naň namapovať napríklad vzťah medzi kvázitriedami odpovedajúcimi `courseID` a `lecturerID` v `TIMETABLE` a `FINALTHESIS`, ak tie užívateľ v prvej fáze manuálne dotvoril. Problém neexistencie niečoho, čo by sa namapovalo na `ex:Graduate` a `ex:topic` sa vyrieši prostým nenamapovaním.

Väčšie problémy nastanú pri menej podobnej ontológii z Obrázka 3.3. Prvou prekážkou by pre prípadnú lexikálnu analýzu bola rozdielnosť názvov odpovedajúcich si prvkov. Nástraha dedičnosti v podobe `ex:Person` musí byť zvládnutá nástrojom (ale iba ak užívateľovi pomáha obmedzovaním relácií a atribútov, na ktoré virtuálne relácie a atribúty možno mapovať). Na `ex:Graduate` a sa opäť nemapuje nič. Údaje o fakultách ontológia nevedie, takže sa tiež nevyužijú. Orieškom je, čo s `ex:attends`. Ak sa táto relácia nechá nenamapovaná, ontológia stráca dosť podstatnú a cennú informáciu. Možnosťou by mohlo byť využitie vzťahu medzi `studentID` a `courseID` v `RESULT` a tiež vo `FINALTHESIS`. Z charakteru týchto tabuliek, kde pre študentov nie je veľa záznamov (`FINALTHESIS`), alebo sú tam neskoro (`RESULT`) plynie, že by to pomohlo len málo. Inou možnosťou by bolo namapovať na `ex:Course` kvázitriedu odpovedajúcu `timetableID` z `TIMETABLE` a na `ex:attends` využiť tabuľku `ENROL`. Stratili by sa však informácie o mene a prerekvizitách kurzu, kurze finálnych prác a výsledkoch. Otázka je, ktorá strata bolí menej. Treťou, najlepšou cestou by bolo vytvorenie špeciálneho pohľadu databázy (ako dotazu nad `ENROL` a `TIMETABLE`) spájajúceho študenta a kurz priamo, kedy by sa problém vyriešil úplne. Správne vytvorené a používané pohľady by všeobecne vyriešili mnohé problémy. V takom prípade je ale nutný zásah do databázy a uvažovanie pohľadov v prvej fáze.

[12]Napríklad z domény relácie sa dá zistiť dodatočný typ triedy, keďže trieda v danej relácii je automaticky aj potomkom triedy z domény. Situácia sa komplikuje pri viacnásobných doménach, dedičnostiach tried a vlastností, atď.

[13]Záverečná časť práce vysvetľuje prístup navrhnutého nástroja.

Podobné problémy a ich obchádzky by vyvstali pri iných nepodobnostiach databázy so vstupnou ontológiou. Bežné situácie mapovania vzťahov tabuliek s kardinalitami *1:n*, *m:n* naopak problémy robiť nebudú, šikovnosťou jednoduchosti princípu totiž miznú mnohé problémy. Dokonca napríklad aj v situáciach reflexívnych tabuliek (`COURSE`), dedičnosti tabuliek (`PhDSTUDENT`), rozširovania hlavných tabuliek (`LECTURERDETAIL`), reflexívnych relácií (`prerequisiteID`) a podobne. Mapovanie je na nižšej úrovni a ak sa pri tvorbe inštancií zabezpečí tvorba jednotných URI, tak trojice RDF o reflektívnych kvázitriedach vzniknuté z rôznych tabuliek budú vypovedať nezávisle, ale konzistentne[14].

Výsledkom mapovacej fázy za predpokladu dobre vykonanej prvej fázy je finálna verzia mDR dokumentu s definíciou jednoznačného mapovania pripravená na ďalšie využitie. Z dokumentu je navyše možné v tomto momente odstrániť ďalej nepotrebné informácie o nevyužitých kvázi a virtuálnych prvkoch, poprípade tiež nepoužívané elementy stĺpcov a tabuliek. Možnú finálnu mDR ontológiu pre ontológiu Obrázka 3.3 obsahuje Dodatok C.

5.7 Využitie vytvoreného mDR dokumentu

Finálna mDR ontológia (príkladom je Dodatok C) predstavuje definíciu mapovania vstupnej databázovej schémy na vstupnú ontológiu. Využíva inštancie špeciálnych prvkov mDR slovníka, ktoré „odvodzuje“ od prvkov ontólogie a informuje o miestach ich pôvodu. Všetky tieto informácie tvoria dostatočné poznatky pre prípadné ďalšie využitie.

Pre malé systémy najvyužiteľnejším je tretia fáza v podobe presunu konkrétnych inštancií. Za predpokladu, že všetky kváziprvky sú iba jednostĺpcové je tvorba inštancií priamočiara:

1. Pre namapované kvázitriedy sa z hodnôt ich zdrojových[15] stĺpcov vytvoria inštancie ich odpovedajúcich tried. Ak na triedu boli namapované viaceré kvázitriedy, tak z viacerých stĺpcov, štandartne si totiž tak či tak odpovedajú, niekedy však môžu prispieť dodatočnými inštanciami (napr. pre Obrázky 3.1, 3.2 a 3.3 inštancie `ex:Teacher` z hodnôt stĺpcov `lecturerID` v `LECTURER` a `LECTURERDETAIL`).

2. Pre inštancie virtuálnych atribútov namapované na literálový atribút sa pre (v prvom kroku vytvorené) inštancie triedy odpovedajúce hodnotám stĺpca zdrojovej kvázitriedy vytvoria z hodnôt stĺpca cieľového kváziatribútu nové hodnoty atribútu (`ex:address` v `ex:Teacher` z `address` v `LECTURERDETAIL`). Pre falošný atribút, ktorý v cieľovej ontológii smeruje na triedu (napr. `ex:String`) sa prvotne vytvoria nové

[14] Tento fakt je jasnejší z nasledujúcich dvoch kapitol.

[15] Zdrojom a cieľom sú v tomto kontexte myslené doména a obor hodnôt.

inštancie tejto triedy z hodnôt cieľového kváziatribútového stĺpca. Následne sa tie spoja s odpovedajúcimi inštanciami triedy stĺpca zdrojovej kvázitriedy.

3. Dvojice inštancií tried odpovedajúce dvojiciam hodnôt stĺpcov zdrojových a cieľových kvázitried namapovaných virtuálnych relácií sa spoja mapovanou reláciou vstupnej ontológie (`ex:teaches` medzi `ex:Teacher` a `ex:Course` z hodnôt dvojíc `lecturerID` a `courseID` v `TIMETABLE`).

Jednoduchý princíp FDR2 zostáva zachovaný aj napriek vlastnému slovníku. V prípade jednostĺpcových kvázitried a kváziatribútov nie je problémom ani tvorba URI pre inštancie tried (nejakým spôsobom odpovedajú hodnotám stĺpcov). V prípade viacstĺpcových je potrebné URI tvorené z hodnôt viacerých stĺpcov tvoriť jednotne. Možností je mnoho, napr. hodnoty stĺpcov spojené podľa abecedného poradia názvov stĺpcov, poradia v mDR dokumente (potom ale musí byť rovnaké pre všetky reflektívne kvázitriedy) alebo inak, záleží na konkrétnej implementácii. Spôsob tvorby URI je na nástroji.

Keďže finálna mDR ontológia je bez inštancií a definuje jednoznačné vzájomné si odpovedanie niektorých prvkov databázy a ontológie, ponúka sa jej využitie aj na iné účely. Príkladom by mohlo byť využitie takýchto mapovacích dokumentov medzi jednou vstupnou ontológiou a (aj viacerými naraz) databázami na dotazovanie nad ontológiou bez existencie konkrétnych inštancií. To by znamenalo preklad dotazu na SQL dotazy databáz. Či je tento mapovací spôsob na takéto použitie vôbec vhodný, alebo iné použita sú otázky pre iné práce ...

5.8 Pozitíva, obmedzenia a možné vylepšenia

Prezentované riešenie mDR má viacero kladných vlastností plynúcich najmä z jeho všeobecnosti a jednoduchosti:

+ Základný a jednoduchý princíp FDR2 ostáva zachovaný. Mapovanie využíva možnosti RDF a nie nový mapovací jazyk. Nástroje môžu využiť existujúce knižnice.

+ Redundancia tvorených kvázi a virtuálnych prvkov je tvorením len rozumných kombinácií znížená. Za cenu komplikovanejšieho slovníka mDR môžu byť kváziprvky viacstĺpcové. To umožňuje oproti FDR2 všeobecnejšie možnosti mapovania[16].

[16]Zachytiteľné sú napríklad viac než binárne spojenia rôznych entít tabuliek typu `FINALTHESIS` databázy Obrázka 3.1;trieda `ex:Mark` môže odpovedať dvojici stĺpcov `studentID` a `courseID` tabuľky `RESULT` a byť spojená s `ex:Student` aj `ex:Course` a podobne.

+ Riešenie zvláda mapovanie bežných situácií (prepojenia tabuliek *1:n*, *m:n*), poradí si s dedičnosťou entít tabuliek, s rozširujúcimi tabuľkami, reflexívnymi vzťahmi[17]. Kľúčom k tomu je konzistentnosť tvorených trojíc. Trojice RDF (vychádzajúce z rôznych tabuliek) sú vďaka zhode domén stĺpcov namapovaných reflektujúcich sa kvázitried celistvé.

+ Nezabúda sa na možnú obojstrannosť relácií, virtuálne relácie medzi kvázitriedami sa tvoria pre istotu v oboch smeroch.

+ Na triedy a relácie môžu byť namapované stĺpce a ich vzťahy z viacerých zdrojov naraz. Ak z tabuliek plynú viaceré prepojenia dvoch kvázitried (vrátane kvázitried ich reflektujúcich), tak vďaka inštancianizácii virtuálnych relácií sú (konzistentne) namapovateľné všetky tieto prepojenia.

+ Užívateľovi pomáha reverzné inžinierstvo prvej fázy a ponúkanie rozumných možností v druhej fáze, čo robí riešenie poloautomatickým.

+ Prípadné nežiadúce automatické návrhy prvej fázy dokáže skorigovať možnosť manuálnej úpravy ich výsledku. Porušenie druhej a tretej normálnej formy tak okrem nutnosti manuálneho dodefinovania kvázi a virtuálnych prvkov v prvej fáze nepredstavuje žiaden výraznejší problém[18]. Je na nástroji, akú hlbokú manuálnu úpravu povolí. Čím väčšia, tým viac pokryteľných možností, ale aj väčšia zložitosť.

+ Riešenie vytvára predpoklady na uplatnenie lexikálneho zarovnávania pojmov mDR druhej fázy a vstupnej ontológie. Podvlastnosti a podtriedy vo vstupnej ontológií sú riešiteľné na úrovni nástroja.

+ Finálna mDR ontológia s mapovaním je využiteľná aj inak než len na prenos inštancií.

[17]Pri rozširovaní hlavnej tabuľky sa napríklad kvázitriedy hlavnej a rozširujúcej tabuľky namapujú na rovnakú triedu (stĺpce `lecturerID` v `LECTURERDETAIL` a `LECTURER`). Tvorené trojice pre relácie, či atribúty sa budú odkazovať na spoločné inštancie triedy. Pri dedičnosti sa naopak kvázitriedy syna namapujú na inú triedu (`studentID` v `PhDSTUDENT` a `STUDENT` na `ex:Postgraduate` a `ex:Student`) (užívateľ ale musí ručne pozmeniť výsledok druhej časti druhej fázy, kedy by sa kvázitrieda pre `studentID` z `PhDSTUDENT` automaticky namapovala na `ex:Student`). Ak ale URI otcovských a synovských inštancií budú v rovnakom formáte (z informácie o `mDR:reflects` by to mohol zaistiť nástroj prenosu inštancií), tak ak otcovská tabuľka obsahuje dodatočné informácie o potomkoch, tak budú napriek nezávislosti trojíc automaticky zachytené aj v inštanciách syna ontológie (`ex:sfirstname` a `ex:ssurname` pre postgraduálnych študentov).

[18]Pre príkladovú tabuľku `employee_siklls(id,skill,address)` (kde `address` je funkčne závislý na časti kľúča `skill`) porušujúcu druhú normálnu formu automatika navrhne iba kvázitriedu odpovedajúcu `id` a `skill` a jej virtuálny atribút s `address`, pričom požadované by možno boli dve kvázitriedy a ich vzájomné virtuálne relácie a atribúty s `address`. Podobne je možná manuálna úprava automatiky uplatnenej na tabuľku porušujúcu tretiu a aj vyššie normálne formy.

Na druhej strane má však riešenie vo svojej terajšej podobe i svoje nedostatky, z ktorých najmarkantnejšími sú:

- Napriek mnohým možnostiam je popis mapovania slabší než pri niektorých iných jazykoch (R_20). Nedokáže napríklad vytvoriť kvázitriedy cez viaceré tabuľky a definovať tak triedy odvoditeľnejšie zložitejším spôsobom.
- Pri neatomických stĺpcoch (porušenie prvej normálnej formy) neposkytuje spôsob transformácie na ich zatomizovanie.
- V neštandartných situáciach sa automatická tvorba kvázi a virtuálnych prvkov môže ľahko líšiť od požadovanej. Je na pozornosti užívateľa, aby takéto sitácie vedel odhaliť. V ešte viac neštandartných situáciach nemusí poskytovať dostatok možností ani manuálna tvorba mDR.
- Pravidlá reverzného inžinierstva prvej fázy nedefinujú, ako sa správať v situácií, keď sa cudzí kľúč odkazuje len na časť primárneho kľúča cieľovej tabuľky. Kvázitrieda môže reflektovať maximálne jednu inú kvázitriedu.
- Pohľady obchádzajúce rôzne mapovacie inkonzistencie môžu znamentať zásah do databázy. Ich nepoužitie zas znamená ťažkú namapovateľnosť niektorých vzťahov.
- Databáza a ontológia sa pre použiteľnosť riešenia musia preto „dosť podobať“.
- Pri neexistencii informácií o cudzích kľúčoch je mapovanie prvej fázy odkázane čisto na užívateľa.
- Dnešná podoba riešenia nešpecifikuje lexikálne porovnávanie názvov prvkov pre účely mapovania v druhej fáze.
- Slovník mDR využíva neštandartné možnosti RDFS, čo môže v implementáciach vyvolať problémy.
- Niektoré problémy sú prenesené na možné nástroje. Príkladom je tvorba URI, ktorú mDR bližšie nepopisuje.
- V dnešnej verzii je mDR orientované len na RDFS, nepočíta s OWL.

Priestorom na ďalšie zlepšenie je určite snaha o odstránenie spomenutých nevýhod riešenia, možno prenos niektorých problémov (tvorba URI) do nových možností mDR slovníka. Pre užitočnosť celého riešenia v bežných aplikáciach (pre ktoré je koniecckoncov stavané) by veľký posun znamenalo budúce dodefinovanie vhodného systému lexikálneho mapovania v druhej fáze.

Kapitola 6

JmDR - pilotná implementácia mDR

Treťou a záverečnou časťou práce je vytvorenie pilotného nástroja, ktorý umožní mapovanie podľa navrhnutého spôsobu. Táto kapitola pojednáva o nástroji *JmDR* implementujúceho prvé dve fázy mDR. Jeho súčasťou je aplikovanie reverzného inžinierstva na tabuľky vstupnej databázy v prvej fáze a ponúkanie rozumných mapovacích možností vo fáze druhej. Lexikálne porovnávanie pojmov neobsahuje, potrebné informácie k takejto analýze pre možný rozvoj vo svojích dátových štruktúrach však uchováva.

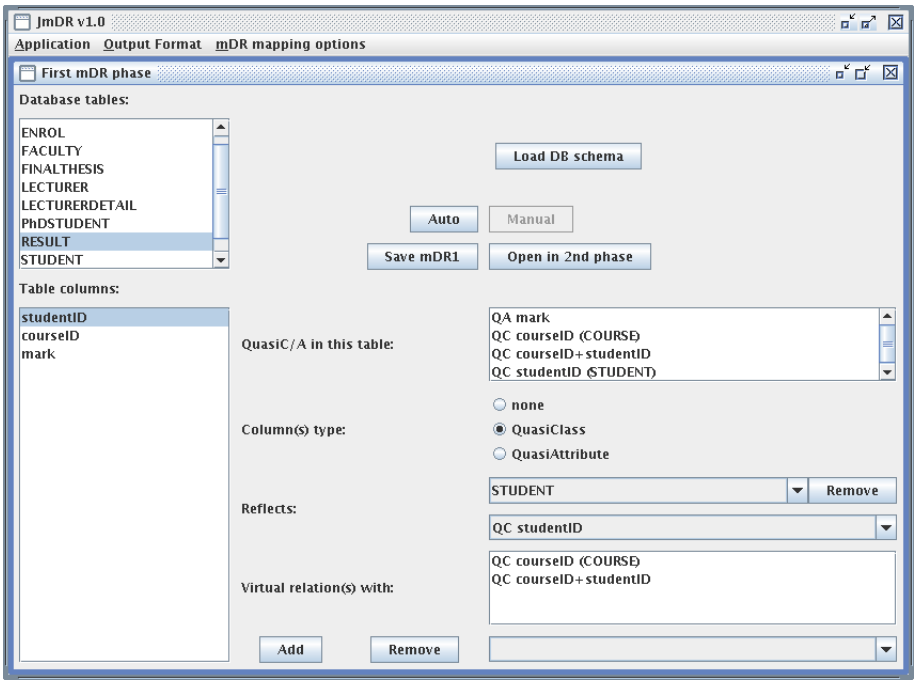
Zdrojový kód aplikácie, skript na jeho preloženie a spustenie, kompletnú Javadoc dokumentáciu, podmienky a návody na spustenie a užívanie sa nachádzajú na priloženom CD. Súčasťou nosiča sú tiež skript pre tvorbu MySQL databázy podľa Obrázka 3.1 a RDFS ontológie Obrázkov 3.2 a 3.3. Podrobne o obsahu CD pojednáva Dodatok D. Obsahom tejto kapitoly je popis JmDR, jeho programátorských princípov, vlastností a obmedzení.

6.1 Popis JmDR

JmDR implementuje prvé dve fázy mDR. Pre každú z nich je vytvorené samostatné okno vrámci aplikácie. Práca v nich je priblížená na konkrétnych príkladoch univerzitnej databázy a ontológie spomínaných v celej práci.

Obrázok 6.1 zachycuje moment vytvárania požadovaných kvázi a virtuálnych prvkov odpovedajúcich vstupnej databáze[1] z Obrázka 3.1 v prvom okne. Na tabuľky bola po načítaní aplikovaná automatická tvorba prvkov reverzným inžinierstvom využívajúca informácie o kľúčoch v podobe predstavenej v Kapitole 5.5. Pre tabuľku `RESULT` vytvorila kvázitriedy odpovedajúce `courseID`, `courseID` s `studentID` a `studentID` a kváziatribút pre `mark`. Keďže databáza bude mapovaná na ontológiu Obrázka 3.3, tak je k

[1] Vstupnou databázou je MySQL databáza.



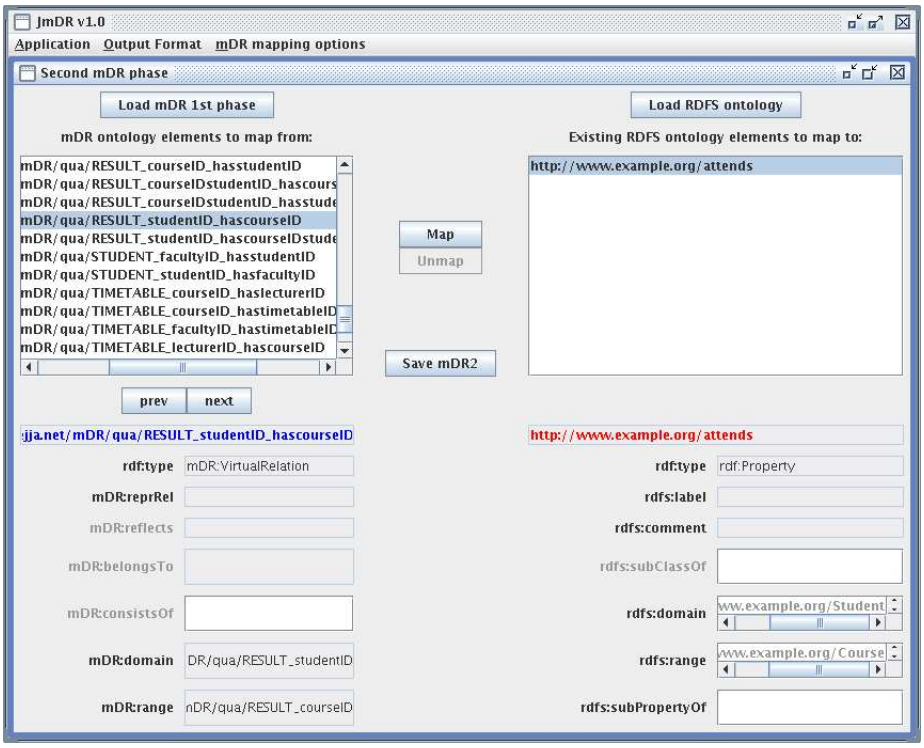
Obrázok 6.1: Okno prvej fázy v JmDR

automaticky vytvoreným virtuálnym atribútom a reláciam (pre `studentID` relácia s dvojicou `courseID` a `studentID`) manuálnym zásahom pridaná virtuálna relácia medzi `studentID` a `courseID`, ktorá bude neskôr namapovaná na `ex:attends`. Kvázitriedam možno nastaviť reflektované kvázitriedy (`studentID` na obrázku automaticky reflektuje rovnomenný stĺpec z tabuľky `STUDENT`). Každá množina stĺpcov môže mať nastavený len jeden typ (kváziatribút, alebo kvázitrieda), ľubovoľný počet virtuálnych relácií (alebo atribútov[2]) a kvázitrieda jednu reflektovanú kvázitriedu. Výstup prvej fázy (v menu sa dá zvoliť jeho formát) sa dá uložiť do súboru alebo priamo otvoriť v okne druhej fázy.

Mapovanie druhej fázy prebieha v okne znázornenom na Obrázku 6.2. Prvky vstupnej mDR ontológie sú zobrazené naľavo[3], vhodné mapovacie cieľe zo vstupnej existujúcej ontológie (z Obrázka 3.3) napravo. Vhodnými sa myslia triedy pre kvázitriedy a vlastnosti (relácie a atribúty) pre virtuálne relácie a atribúty. Ak je v menu nastavené zobrazovanie len rozumných možností, sú tieto vhodné prvky ešte viac zúžené. Pre reflektujúce kvázitriedy to v JmDR znamená zobrazenie len tried mapovaných reflektovanými kvázitriedami. Pre virtuálne relácie a atribúty je situácia zložitejšia. K virtu-

[2]Z implementačných dôvodov sa virtuálny atribút nastavuje od kváziatribútu ku kvázitriede, pri virtuálnych reláciach je smer normálny.

[3]Zobrazené sú v nasledujúcom poradí: nereflektujúce kvázitriedy, reflektujúce kvázitriedy, kváziatribúty, virtuálne atribúty a nakoniec virtuálne relácie.



Obrázok 6.2: Okno druhej fázy v JmDR

álnemu atribútu sa zobrazujú len tie vlastnosti vstupnej ontológie, u ktorých množina ich domén je podmnožinou množiny tried, do ktorých patrí trieda namapovaná z kvázitriedovej domény daného virtuálneho atribútu. Za množinu domén nejakej vlastnosti vstupnej ontológie sa pritom považuje doména tejto vlastnosti obohatená o domény otcovských vlastností (nastavených pomocou `rdfs:subPropertyOf`). Dedičnosť (získaná z `rdfs:subClassOf`) sa uvažuje aj pri stanovení množiny tried, do ktorých patrí istá trieda vstupnej ontológie. Pri virtuálnych reláciach sa analogicky kontrolujú aj obory hodnôt.

Takéto správanie umožňuje v situácii dedičnosti ontológie z Obrázka 3.3 ponúknuť pre virtuálny atribút medzi `studentID` a `firstname` zo `STUDENT` na namapovanie aj vlastnosť `ex:firstname` triedy `ex:Person`[4]. V menej štandartných prípadoch môže byť lepšie ponúkanie len rozumných možností vypnúť a umožniť ľubovoľné mapovanie. To dovolí namapovať `studentID` v `PhDSTUDENT` reflektujúci `studentID` v `STUDENT` aj na `ex:Postgraduate`.

Mapovanie virtuálnej relácie medzi `studentID` a `courseID` v `RESULT` vytvorenej manuálne v prvej fáze (Obrázok 6.1) zachycuje Obrázok 6.2. Vďaka faktom, že kvázitriedy pre `studentID` a `courseID` v `RESULT` už sú

[4]Ak `studentID` bol namapovaný na `ex:Student`, pričom `ex:Student` je potomkom `ex:Person`.

namapované na `ex:Student` a `ex:Course`, že doménou a oborom hodnôt `ex:attends` sú práve `ex:Student` a `ex:Course` a že doménou a oborom hodnôt danej virtuálnej relácie sú `studentID` a `courseID` z `RESULT` sa medzi rozumnými cieľovými vlastnosťami zobrazuje `ex:attends`, na ktorý sa virtuálna relácia namapuje.

Implementovaný program JmDR zvláda mapovania v okrem tu spomenutých aj vo všetkých ostatných situáciách ilustračnej univerzitnej databázy a ontológiach. V iných reálnych situáciach by mapovanie vyzeralo podobne. Okrem užitočnej poloautomatickej pomoci umožní pružnosť využitia manuálneho editovania v prvej fáze a vypnutia rozumných možností v druhej fáze zvládnutie i viac špeciálnych situácií. Finálnym výstupom je mapovacia ontológia mDR v nastavenom formáte. Oproti svojej prvej verzii obsahuje údaje o mapovaní a je zbavená všetkých nepotrebných elementov.

6.2 Princípy a problémy implementácie

Aplikácia JmDR je vytvorená v jazyku Java verzie 1.5. Ako databázu využíva databázu MySQL, spojenie na ňu zabezpečuje MySQL Connector-J 5.0[5]. Na prácu s ontológiami a RDFS využíva mohutnú knižnicu Jena[6].

Základom okna prvej fázy, ktoré pripraví kvázi a virtuálne prvky pre fázu druhú je špeciálna dátová štruktúra `Dschema`[7]. Obsahuje štruktúry pre tabuľky so štruktúrami pre ich stĺpce a zoznamami kvázi elementov. Štruktúry sú odvodené od `DefaultListModel`, čo umožňuje jednoduché napĺňanie ich obsahu do `JList` komponentov na okne. Po načítaní databázy sa inštancia `Dschema` správne vyplní, podľa jej informácií o cudzích kľúčoch (ktoré si vytvorí tiež) aplikovaním pravidiel reverzného inžinierstva sa naplnia zoznamy správne vzájomne prepojených kvázi elementov. Programátorsky nie je prvá fáza nijako výnimočne zaujímavá. Problémom bolo len vytvorenie vhodnej štruktúry a ustráženie správneho chovania sa formulárov. Kvôli možnej lexikálnej analýze v druhej fáze sa ku tabuľkám a stĺpcom starostlivo uchovávajú aj ich komentáre. V použitej verzii MySQL Connectoru sa však z databázy nevracaju komentáre tabuliek. Aplikácia pracuje akoby áno, a tak pri oprave tejto chyby budú vznikať i elementy komentárov tabuliek. Pre tvorbu ontológie mDR prvej verzie sa na dátové štruktúry aplikujú funkcie knižnice Jena.

Obmedzením pilotnej verzie JmDR je, že neuvažuje neštandartné situácie s cudzími a primárnymi kľúčmi. Cudzí kľúč sa napríklad musí odkazovať na celý primárny kľúč inej tabuľky (nie len jeho vlastnú podmnožinu). Pre

[5] `http://dev.mysql.com/downloads/connector/j/5.0.html`

[6] `http://jena.sourceforge.net/`

[7] Detailnejšie je význam všetkých tu spomenutých tried pochopiteľný z Javadoc dokumentácie aplikácie na CD.

domény a obory hodnôt virtuálnych relácií a atribútov a pre reflektované kvázitriedy je dovolená len jedna hodnota. Zovšeobecnenie aj v týchto oblastiach by možno pokrylo nejaké ďalšie neštandartné možnosti.

Okno druhej fázy používa štruktúry typu `MyResource` (uchovávajúce informácie o typoch, doménach, reflexiách, mapovaniach a pod.) napĺňajúce inštancie `MyResourceList`, ktoré tiež rozširujú `DefaultListModel`. Aplikácia pri načítaní ontológií pomocou Jena naplní ich obsah. Súčasťou sú samostatne prístupné informácie o názvoch a komentároch zdrojových tabuliek a stĺpcov kvôli zjednodušeniu doprogramovania prípadnej lexikálnej analýzy. Programátorsky zaujímavým je len zabezpečenie zobrazenia správnych rozumných možností. Deje sa tak presne podľa popisu v predchádzajúcej podkapitole. Možné komplikácie pri mapovaní umožňuje obísť vypnutie ponúkania rozumných možností, kedy sa nekontroluje, čo sa na čo mapuje. Po nadefinovaní mapovania sa do finálnej mDR ontológie nepridajú nepotrebné elementy. Tými sú v mapovaní nepoužité kvázitriedy, kváziatribúty, nenamapované virtuálne relácie a atribúty a neodkazované tabuľky a stĺpce databázy.

Malým problémom je, že Jena v použitej verzii (možno konfigurácii) nevyhadzuje správne výnimky. Taktiež nefunguje načítanie mDR slovníka z Internetu a zároveň alternatívne z lokálneho adresára, preto sa vyžaduje prítomnosť slovníka v adresári s aplikáciou. Aplikácia pri hľadaní domén a typov zdrojov vstupnej ontológie nevyužíva inferenciu okrem vyjadrenej priamo pomocou `rdfs:subClassOf` a `rdfs:subPropertyOf`.

6.3 Zhrnutie

Súčasťou práce je aplikácia JmDR. Z prezentovaného riešenia mDR implementuje jeho prvé dve fázy, vrátane využívania informácií o kľúčoch tabuliek a ponúkania rozumných „mapovacích“ možností. Neponúka lexikálnu analýzu, no pracuje viacmenej presne podľa Kapitoly 5. Aplikácia bola testovaná s ilustračnou databázou a ontológiami o univerzite, kde dokázala pokryť všetky neštandartné situácie a vygenerovala požadované mapovacie ontológie. Pri iných vstupoch by aplikácia mala bez problémov zvládnuť podobné i ďalšie situácie. Z rozsahových dôvodov však neboli testované.

Napriek mnohým dobrým vlastnostiam obsahuje aj svoje nedostatky, niekedy zapríčinené používanými knižnicami. Ovládanie vyžaduje užívateľovu dobrú znalosť princípov mDR a vstupnej databázy a ontológie. Pri náhodnom klikaní a nerozumení vstupom je navyše možné vytvoriť logické nezmysly (najmä pri manuálnych úpravach spojených s vypnutím rozumných možností). Stále však ide len o pilotný návrh, ktorého mnohé nedostatky by sa dali prípadným rozšírením aplikácie odstrániť.

Cestou do budúcna je preto ich potlačenie a najmä výzva v podobe

doplnenia lexikálnej analýzy. Vhodné by bolo tiež prepracovanie užívateľských formulárov, doplnenie prehľadného zobrazenia vstupnej ontológie, zovšeobecnenie na prácu s ľubovoľnou databázou.

Terajšia verzia JmDR je ako súčasť diplomovej práce umiestnená na priloženom CD (Dodatok D).

Kapitola 7

Záver

Myšlienky a cieľe Sémantického webu sú zaujímavé a pre oblasti jeho využitia by mohli znamenať výrazný krok dopredu (ak už tak nečinia). V súčasnosti sa však napriek rokom výskumu s týmto pojmom stále nestretávame bežne. Roztopiť ľady by mohla pomôcť jednoduchosť nasadenia v stávajúcich systémoch, čo okrem iného znamená aj nutnosť nekomplikovanej tvorby jeho dát.

Prezentovaná diplomová práca sa zaoberá malou podmnožinou tvorby dát - mapovaním jestvujúcich dát v databázach na už existujúce ontológie. Cieľom prvej časti práce bolo zistiť, čo už bolo v tejto tématike vymyslené. Ako ukázala, tak prekvapujúco už aj tu relatívne dosť. Od manuálnejších pomocných nástrojov po sofistikovanejšie a nápomocné. Niektoré z nich boli v samostatnej kapitole bližšie priblížené.

Cieľom väčšinovej časti práce bol návrh a čiastočná implementácia vlastného spôsobu mapovania. Vo svojej podstate sa prezentované mDR inšpiruje už existujúcim riešením FDR2[11]. Zachováva si jeho jednoduchosť a pružnosť, široko ho však zovšeobecňuje a zároveň inteligizuje. Mapovanie delí do dvoch fáz. V prvej dochádza nad relačnou schémou pre pravdepodobne namapovateľné skupiny stĺpcov k tvorbe tzv. kvázi a virtuálnych prvkov. Pri tom sa využívajú informácie o kľúčoch tabuliek. Náplňou druhej fázy je zarovnanie niektorých z vytvorených prvkov s prvkami vstupnej ontológie. Uľahčením užívateľovi je ponúkanie len „rozumných“ možností. Výstupom je súbor definujúci mapovanie využiteľný napríklad v prípadnom prenose samotných inštancií. Prvé dve fázy implementuje pilotná aplikácia JmDR.

Už aj takto zoštíhlená problematika je stále široká. Navyše jej hranice sú definovateľné často iba veľmi rozvláčne. Z týchto a rozsahového dôvodu práca využila na ilustrovanie požadovaných vlastností a možných problémov špeciálne vytvorenú databázu a ontológie o univerzite. Snahou pri ich vytvorení bolo pokrytie čo najviac bežných vzniknuteľných mapovacích situácií.

Niektoré námety pre ďalší vývoj už boli spomenuté. Výzvou by mohlo byť najmä domyslenie efektívneho spôsobu využitia lexikálneho porovná-

vania názvov prvkov databázy a ontológie a zovšeobecnenie na širší jazyk OWL (prezentované riešenie sa orientuje na RDFS). K možnej reálnej nasaditeľnosti by určite prispelo vyvinutie dokonalejšieho mapovacieho nástroja a využitie mapovania treťou fázou. Čo sa týka textu práce, priestor by bol v nahradení používania univerzitného príkladu exaktnejšími definíciami, to by však na druhej strane mohlo prispieť k väčšej neprehľadnosti. Určite užitočným by bolo otestovanie v ďalších situáciách, čo práca z rozsahových dôvodov nečinila.

Práca sa pohybuje v problematike s viditeľným potenciálnym využitím. Prínos, či už v priamej podobe alebo v podobe inšpirovania niekoho ďalšieho tak určite má. Svoje zadanie (zmapovanie problematiky, návrh a implementácia vlastného riešenia) napriek nedostatkom naplnila.

Dodatok A

Definícia mDR slovníka

Kompletná definícia mDR slovníka, zapísaná v RDF/XML:

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
 <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
 <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
 <!ENTITY mDR "http://muro.hejja.net/mDR/mDR.rdf">]>

<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xml:base="http://muro.hejja.net/mDR/mDR.rdf">

<rdfs:Class rdf:ID="DBclass">
        <rdfs:isDefinedBy rdf:resource="&mDR;"/>
</rdfs:Class>

<rdfs:Class rdf:ID="DBtable">
        <rdfs:isDefinedBy rdf:resource="&mDR;"/>
        <rdfs:subClassOf rdf:resource="#DBclass"/>
</rdfs:Class>

<rdfs:Class rdf:ID="DBcolumn">
        <rdfs:isDefinedBy rdf:resource="&mDR;"/>
        <rdfs:subClassOf rdf:resource="#DBclass"/>
</rdfs:Class>

<rdf:Property rdf:ID="comment">
        <rdfs:isDefinedBy rdf:resource="&mDR;"/>
        <rdfs:domain rdf:resource="#DBclass"/>
        <rdfs:subPropertyOf rdf:resource="&rdfs;comment"/>
</rdf:Property>

<rdf:Property rdf:ID="label">
        <rdfs:isDefinedBy rdf:resource="&mDR;"/>
        <rdfs:domain rdf:resource="#DBclass"/>
```

```
        <rdfs:subPropertyOf rdf:resource="&rdfs;label"/>
</rdf:Property>

<rdfs:Class rdf:ID="Qclass">
        <rdfs:isDefinedBy rdf:resource="&mDR;"/>
</rdfs:Class>

<rdfs:Class rdf:ID="QuasiClass">
        <rdfs:isDefinedBy rdf:resource="&mDR;"/>
        <rdfs:subClassOf rdf:resource="#Qclass"/>
</rdfs:Class>

<rdfs:Class rdf:ID="QuasiAttribute">
        <rdfs:isDefinedBy rdf:resource="&mDR;"/>
        <rdfs:subClassOf rdf:resource="#Qclass"/>
</rdfs:Class>

<rdf:Property rdf:ID="consistsOf">
        <rdfs:isDefinedBy rdf:resource="&mDR;"/>
        <rdfs:domain rdf:resource="#Qclass"/>
        <rdfs:range rdf:resource="#DBcolumn"/>
</rdf:Property>

<rdf:Property rdf:ID="belongsTo">
        <rdfs:isDefinedBy rdf:resource="&mDR;"/>
        <rdfs:domain rdf:resource="#Qclass"/>
        <rdfs:range rdf:resource="#DBtable"/>
</rdf:Property>

<rdf:Property rdf:ID="virtualAttribute">
        <rdfs:isDefinedBy rdf:resource="&mDR;"/>
        <rdfs:domain rdf:resource="#QuasiClass"/>
        <rdfs:range rdf:resource="#QuasiAttribute"/>
</rdf:Property>

<rdf:Property rdf:ID="virtualRelation">
        <rdfs:isDefinedBy rdf:resource="&mDR;"/>
        <rdfs:domain rdf:resource="#QuasiClass"/>
        <rdfs:range rdf:resource="#QuasiClass"/>
</rdf:Property>

<rdf:Property rdf:ID="reflects">
        <rdfs:isDefinedBy rdf:resource="&mDR;"/>
        <rdfs:domain rdf:resource="#QuasiClass"/>
        <rdfs:range rdf:resource="#QuasiClass"/>
</rdf:Property>

<rdf:Property rdf:ID="representsClass">
        <rdfs:isDefinedBy rdf:resource="&mDR;"/>
        <rdfs:subPropertyOf rdf:resource="&rdfs;subClassOf"/>
        <rdfs:domain rdf:resource="#Qclass"/>
</rdf:Property>
```

```
<rdf:Property rdf:ID="representsAttribute">
        <rdfs:isDefinedBy rdf:resource="&mDR;"/>
        <rdfs:subPropertyOf rdf:resource="&rdfs;subPropertyOf"/>
        <rdfs:domain rdf:resource="#virtualAttribute"/>
</rdf:Property>

<rdf:Property rdf:ID="representsRelation">
        <rdfs:isDefinedBy rdf:resource="&mDR;"/>
        <rdfs:subPropertyOf rdf:resource="&rdfs;subPropertyOf"/>
        <rdfs:domain rdf:resource="#virtualRelation"/>
</rdf:Property>

</rdf:RDF>
```

Príklad A.1: Definícia mDR slovníka

Dodatok B

Príklad mDR ontológie prvej fázy

Obsahom tejto časti dodatkov je požadovaný automaticky vytvorený dokument s mDR ontológiou prvej fázy pre databázu z Obrázka 3.1.

Zápis je podobný zápisu trojicami, kvôli prehľadnosti sú zdroje zoradené podľa tabuliek (v nástrojoch typu JmDR môže byť výstup komplikovanejší a menej čitateľný). Používané menné priestory sú:

- db - http://muro.hejja.net/mDR/db/ pre zdroje tabuliek a stĺpcov
- mDR - http://muro.hejja.net/mDR/mDR.rdf# pre prvky mDR slovníka
- qua - http://muro.hejja.net/mDR/qua/ pre zdroje reprezentujúce inštancie kvázi a virtuálnych prvkov

```
qua:STUDENT_studID              rdf:type         mDR:QuasiClass
qua:STUDENT_studID              mDR:belongsTo    db:STUDENT
qua:STUDENT_studID              mDR:consistsOf   db:STUDENT.studentID
qua:STUDENT_fname               rdf:type         mDR:QuasiAttribute
qua:STUDENT_fname               mDR:belongsTo    db:STUDENT
qua:STUDENT_fname               mDR:consistsOf   db:STUDENT.firstname
qua:STUDENT_studID_hasfname     rdf:type         mDR:virtualAttribute
qua:STUDENT_studID_hasfname     rdfs:domain      qua:STUDENT_studID
qua:STUDENT_studID_hasfname     rdfs:range       qua:STUDENT_fname
qua:STUDENT_sname               rdf:type         mDR:QuasiAttribute
qua:STUDENT_sname               mDR:belongsTo    db:STUDENT
qua:STUDENT_sname               mDR:consistsOf   db:STUDENT.surname
qua:STUDENT_studID_hassname     rdf:type         mDR:virtualAttribute
qua:STUDENT_studID_hassname     rdfs:domain      qua:STUDENT_studID
qua:STUDENT_studID_hassname     rdfs:range       qua:STUDENT_sname
qua:STUDENT_facID               rdf:type         mDR:QuasiClass
qua:STUDENT_facID               mDR:reflects     qua:FACULTY_facID
qua:STUDENT_facID               mDR:belongsTo    db:STUDENT
qua:STUDENT_facID               mDR:consistsOf   db:STUDENT.facultyID
qua:STUDENT_studID_hasfacID     rdf:type         mDR:virtualRelation
qua:STUDENT_studID_hasfacID     rdfs:domain      qua:STUDENT_studID
qua:STUDENT_studID_hasfacID     rdfs:range       qua:STUDENT_facID
qua:STUDENT_facID_hasstudID     rdf:type         mDR:virtualRelation
qua:STUDENT_facID_hasstudID     rdfs:domain      qua:STUDENT_facID
qua:STUDENT_facID_hasstudID     rdfs:range       qua:STUDENT_studID
```

```
qua:PhDSTUDENT_studID           rdf:type        mDR:QuasiClass
qua:PhDSTUDENT_studID           mDR:reflects    qua:STUDENT_studID
qua:PhDSTUDENT_studID           mDR:belongsTo   db:PhDSTUDENT
qua:PhDSTUDENT_studID           mDR:consistsOf  db:PhDSTUDENT.studentID
qua:PhDSTUDENT_sship            rdf:type        mDR:QuasiAttribute
qua:PhDSTUDENT_sship            mDR:belongsTo   db:PhDSTUDENT
qua:PhDSTUDENT_sship            mDR:consistsOf  db:PhDSTUDENT.scholarship
qua:PhDSTUDENT_studID_hassship  rdf:type        mDR:virtualAttribute
qua:PhDSTUDENT_studID_hassship  rdfs:domain     qua:PhDSTUDENT_studID
qua:PhDSTUDENT_studID_hassship  rdfs:range      qua:PhDSTUDENT_sname

qua:FACULTY_facID               rdf:type        mDR:QuasiClass
qua:FACULTY_facID               mDR:belongsTo   db:FACULTY
qua:FACULTY_facID               mDR:consistsOf  db:FACULTY.facultyID
qua:FACULTY_name                rdf:type        mDR:QuasiAttribute
qua:FACULTY_name                mDR:belongsTo   db:FACULTY
qua:FACULTY_name                mDR:consistsOf  db:FACULTY.name
qua:FACULTY_facID_hasname       rdf:type        mDR:virtualAttribute
qua:FACULTY_facID_hasname       rdfs:domain     qua:FACULTY_facID
qua:FACULTY_facID_hasname       rdfs:range      qua:FACULTY_name

qua:LECTURER_lectID             rdf:type        mDR:QuasiClass
qua:LECTURER_lectID             mDR:belongsTo   db:LECTURER
qua:LECTURER_lectID             mDR:consistsOf  db:LECTURER.lecturerID
qua:LECTURER_fname              rdf:type        mDR:QuasiAttribute
qua:LECTURER_fname              mDR:belongsTo   db:LECTURER
qua:LECTURER_fname              mDR:consistsOf  db:LECTURER.firstname
qua:LECTURER_lectID_hasfname    rdf:type        mDR:virtualAttribute
qua:LECTURER_lectID_hasfname    rdfs:domain     qua:LECTURER_lectID
qua:LECTURER_lectID_hasfname    rdfs:range      qua:LECTURER_fname
qua:LECTURER_sname              rdf:type        mDR:QuasiAttribute
qua:LECTURER_sname              mDR:belongsTo   db:LECTURER
qua:LECTURER_sname              mDR:consistsOf  db:LECTURER.surname
qua:LECTURER_lectID_hassname    rdf:type        mDR:virtualAttribute
qua:LECTURER_lectID_hassname    rdfs:domain     qua:LECTURER_lectID
qua:LECTURER_lectID_hassname    rdfs:range      qua:LECTURER_sname

qua:LECTURERDETAIL_lectID       rdf:type        mDR:QuasiClass
qua:LECTURERDETAIL_lectID       mDR:reflects    qua:LECTURER_lectID
qua:LECTURERDETAIL_lectID       mDR:belongsTo   db:LECTURERDETAIL
qua:LECTURERDETAIL_lectID       mDR:consistsOf  db:LECTURERDETAIL.lecturerID
qua:LECTURERDETAIL_addr         rdf:type        mDR:QuasiAttribute
qua:LECTURERDETAIL_addr         mDR:belongsTo   db:LECTURERDETAIL
qua:LECTURERDETAIL_addr         mDR:consistsOf  db:LECTURERDETAIL.address
qua:LECTURERDETAIL_lectID_hasaddr  rdf:type     mDR:virtualAttribute
qua:LECTURERDETAIL_lectID_hasaddr  rdfs:domain  qua:LECTURERDETAIL_lectID
qua:LECTURERDETAIL_lectID_hasaddr  rdfs:range   qua:LECTURERDETAIL_addr

qua:RESULT_studIDcourID         rdf:type        mDR:QuasiClass
qua:RESULT_studIDcourID         mDR:belongsTo   db:RESULT
qua:RESULT_studIDcourID         mDR:consistsOf  db:RESULT.studentID
qua:RESULT_studIDcourID         mDR:consistsOf  db:RESULT.courseID
qua:RESULT_studID               rdf:type        mDR:QuasiClass
qua:RESULT_studID               mDR:reflects    qua:STUDENT_studID
qua:RESULT_studID               mDR:belongsTo   db:RESULT
qua:RESULT_studID               mDR:consistsOf  db:RESULT.studentID
qua:RESULT_studIDcourID_hasstudID   rdf:type    mDR:virtualRelation
qua:RESULT_studIDcourID_hasstudID   rdfs:domain qua:RESULT_studIDcourID
qua:RESULT_studIDcourID_hasstudID   rdfs:range  qua:RESULT_studID
qua:RESULT_studID_hasstudIDcourID   rdf:type    mDR:virtualRelation
qua:RESULT_studID_hasstudIDcourID   rdfs:domain qua:RESULT_studID
qua:RESULT_studID_hasstudIDcourID   rdfs:range  qua:RESULT_studIDcourID
qua:RESULT_courID               rdf:type        mDR:QuasiClass
qua:RESULT_courID               mDR:reflects    qua:COURSE_courID
```

```
qua:RESULT_courID                mDR:belongsTo   db:RESULT
qua:RESULT_courID                mDR:consistsOf  db:RESULT.courseID
qua:RESULT_studIDcourID_hascourID   rdf:type    mDR:virtualRelation
qua:RESULT_studIDcourID_hascourID   rdfs:domain qua:RESULT_studIDcourID
qua:RESULT_studIDcourID_hascourID   rdfs:range  qua:RESULT_courID
qua:RESULT_courID_hasstudIDcourID   rdf:type    mDR:virtualRelation
qua:RESULT_courID_hasstudIDcourID   rdfs:domain qua:RESULT_courID
qua:RESULT_courID_hasstudIDcourID   rdfs:range  qua:RESULT_studIDcourID
qua:RESULT_mark                  rdf:type        mDR:QuasiAttribute
qua:RESULT_mark                  mDR:belongsTo   db:RESULT
qua:RESULT_mark                  mDR:consistsOf  db:RESULT.mark
qua:RESULT_studIDcourID_hasmark rdf:type         mDR:virtualAttribute
qua:RESULT_studIDcourID_hasmark rdfs:domain      qua:RESULT_studIDcourID
qua:RESULT_studIDcourID_hasmark rdfs:range       qua:RESULT_mark

qua:COURSE_courID                rdf:type        mDR:QuasiClass
qua:COURSE_courID                mDR:belongsTo   db:COURSE
qua:COURSE_courID                mDR:consistsOf  db:COURSE.courseID
qua:COURSE_prereqID              rdf:type        mDR:QuasiClass
qua:COURSE_prereqID              mDR:reflects    qua:COURSE_courID
qua:COURSE_prereqID              mDR:belongsTo   db:COURSE
qua:COURSE_prereqID              mDR:consistsOf  db:COURSE.prerequisiteID
qua:COURSE_courID_hasprereqID    rdf:type        mDR:virtualRelation
qua:COURSE_courID_hasprereqID    rdf:domain      qua:COURSE_courID
qua:COURSE_courID_hasprereqID    rdf:range       qua:COURSE_prereqID
qua:COURSE_prereqID_hascourID    rdf:type        mDR:virtualRelation
qua:COURSE_prereqID_hascourID    rdf:domain      qua:COURSE_prereqID
qua:COURSE_prereqID_hascourID    rdf:range       qua:COURSE_courID
qua:COURSE_name                  rdf:type        mDR:QuasiAttribute
qua:COURSE_name                  mDR:belongsTo   db:COURSE
qua:COURSE_name                  mDR:consistsOf  db:COURSE.name
qua:COURSE_courID_hasname        rdf:type        mDR:virtualAttribute
qua:COURSE_courID_hasname        rdfs:domain     qua:COURSE_courID
qua:COURSE_courID_hasname        rdfs:range      qua:COURSE_name

qua:FINALTHESIS_studIDlectIDcourID  rdf:type        mDR:QuasiClass
qua:FINALTHESIS_studIDlectIDcourID  mDR:belongsTo   db:FINALTHESIS
qua:FINALTHESIS_studIDlectIDcourID  mDR:consistsOf  db:FINALTHESIS.studID
qua:FINALTHESIS_studIDlectIDcourID  mDR:consistsOf  db:FINALTHESIS.lectID
qua:FINALTHESIS_studIDlectIDcourID  mDR:consistsOf  db:FINALTHESIS.courID
qua:FINALTHESIS_studID           rdf:type           mDR:QuasiClass
qua:FINALTHESIS_studID           mDR:reflects       db:STUDENT.studID
qua:FINALTHESIS_studID           mDR:belongsTo      db:FINALTHESIS
qua:FINALTHESIS_studID           mDR:consistsOf     db:FINALTHESIS.studID
qua:FINALTHESIS_studIDlectIDcourID_hasstudID  rdf:type    mDR:virtualRelation
qua:FINALTHESIS_studIDlectIDcourID_hasstudID  rdfs:domain qua:FINALTHESIS_studIDlectIDcourID
qua:FINALTHESIS_studIDlectIDcourID_hasstudID  rdfs:range  qua:FINALTHESIS_studID
qua:FINALTHESIS_studID_hasstudIDlectIDcourID  rdf:type    mDR:virtualRelation
qua:FINALTHESIS_studID_hasstudIDlectIDcourID  rdfs:domain qua:FINALTHESIS_studID
qua:FINALTHESIS_studID_hasstudIDlectIDcourID  rdfs:range  qya:FINALTHESIS_studIDlectIDcourID
qua:FINALTHESIS_lectID           rdf:type        mDR:QuasiClass
qua:FINALTHESIS_lectID           mDR:reflects    db:LECTURER.lectID
qua:FINALTHESIS_lectID           mDR:belongsTo   db:FINALTHESIS
qua:FINALTHESIS_lectID           mDR:consistsOf  db:FINALTHESIS.lectID
qua:FINALTHESIS_studIDlectIDcourID_haslectID  rdf:type    mDR:virtualRelation
qua:FINALTHESIS_studIDlectIDcourID_haslectID  rdfs:domain qua:FINALTHESIS_studIDlectIDcourID
qua:FINALTHESIS_studIDlectIDcourID_haslectID  rdfs:range  qua:FINALTHESIS_lectID
qua:FINALTHESIS_lectID_hasstudIDlectIDcourID  rdf:type    mDR:virtualRelation
qua:FINALTHESIS_lectID_hasstudIDlectIDcourID  rdfs:domain qua:FINALTHESIS_lectID
qua:FINALTHESIS_lectID_hasstudIDlectIDcourID  rdfs:range  qua:FINALTHESIS_studIDlectIDcourID
qua:FINALTHESIS_courID           rdf:type        mDR:QuasiClass
qua:FINALTHESIS_courID           mDR:reflects    db:COURSE.courID
qua:FINALTHESIS_courID           mDR:belongsTo   db:FINALTHESIS
qua:FINALTHESIS_courID           mDR:consistsOf  db:FINALTHESIS.courID
qua:FINALTHESIS_studIDlectIDcourID_hascourID  rdf:type    mDR:virtualRelation
```

```
qua:FINALTHESIS_studIDlectIDcourID_hascourID  rdfs:domain qua:FINALTHESIS_studIDlectIDcourID
qua:FINALTHESIS_studIDlectIDcourID_hascourID  rdfs:range  qua:FINALTHESIS_courID
qua:FINALTHESIS_courID_hasstudIDlectIDcourID  rdf:type    mDR:virtualRelation
qua:FINALTHESIS_courID_hasstudIDlectIDcourID  rdfs:domain qya:FINALTHESIS_courID
qua:FINALTHESIS_courID_hasstudIDlectIDcourID  rdfs:range  qua:FINALTHESIS_studIDlectIDcourID

qua:TIMETABLE_timeID            rdf:type        mDR:QuasiClass
qua:TIMETABLE_timeID            mDR:belongsTo   db:TIMETABLE
qua:TIMETABLE_timeID            mDR:consistsOf  db:TIMETABLE.timetableID
qua:TIMETABLE_courID            rdf:type        mDR:QuasiClass
qua:TIMETABLE_courID            mDR:reflects    qua:COURSE_courID
qua:TIMETABLE_courID            mDR:belongsTo   db:TIMETABLE
qua:TIMETABLE_courID            mDR:consistsOf  db:TIMETABLE.courseID
qua:TIMETABLE_timeID_hascourID  rdf:type        mDR:virtualRelation
qua:TIMETABLE_timeID_hascourID  rdfs:domain     qua:TIMETABLE_timeID
qua:TIMETABLE_timeID_hascourID  rdfs:range      qua:TIMETABLE_courID
qua:TIMETABLE_courID_hastimeID  rdf:type        mDR:virtualRelation
qua:TIMETABLE_courID_hastimeID  rdfs:domain     qua:TIMETABLE_courID
qua:TIMETABLE_courID_hastimeID  rdfs:range      qua:TIMETABLE_timeID
qua:TIMETABLE_lectID            rdf:type        mDR:QuasiClass
qua:TIMETABLE_lectID            mDR:reflects    qua:LECTURER_lectID
qua:TIMETABLE_lectID            mDR:belongsTo   db:TIMETABLE
qua:TIMETABLE_lectID            mDR:consistsOf  db:TIMETABLE.lecturerID
qua:TIMETABLE_timeID_haslectID  rdf:type        mDR:virtualRelation
qua:TIMETABLE_timeID_haslectID  rdfs:domain     qua:TIMETABLE_timeID
qua:TIMETABLE_timeID_haslectID  rdfs:range      qua:TIMETABLE_lectID
qua:TIMETABLE_lectID_hastimeID  rdf:type        mDR:virtualRelation
qua:TIMETABLE_lectID_hastimeID  rdfs:domain     qua:TIMETABLE_lectID
qua:TIMETABLE_lectID_hastimeID  rdfs:range      qua:TIMETABLE_timeID
qua:TIMETABLE_facID             rdf:type        mDR:QuasiClass
qua:TIMETABLE_facID             mDR:reflects    qua:FACULTY_facID
qua:TIMETABLE_facID             mDR:belongsTo   db:TIMETABLE
qua:TIMETABLE_facID             mDR:consistsOf  db:TIMETABLE.facultyID
qua:TIMETABLE_timeID_hasfacID   rdf:type        mDR:virtualRelation
qua:TIMETABLE_timeID_hasfacID   rdfs:domain     qua:TIMETABLE_timeID
qua:TIMETABLE_timeID_hasfacID   rdfs:range      qua:TIMETABLE_facID
qua:TIMETABLE_facID_hastimeID   rdf:type        mDR:virtualRelation
qua:TIMETABLE_facID_hastimeID   rdfs:domain     qua:TIMETABLE_facID
qua:TIMETABLE_facID_hastimeID   rdfs:range      qua:TIMETABLE_timeID
qua:TIMETABLE_date              rdf:type        mDR:QuasiAttribute
qua:TIMETABLE_date              mDR:belongsTo   db:TIMETABLE
qua:TIMETABLE_date              mDR:consistsOf  db:TIMETABLE.date
qua:TIMETABLE_timeID_hasdate    rdf:type        mDR:virtualAttribute
qua:TIMETABLE_timeID_hasdate    rdfs:domain     qua:TIMETABLE_timeID
qua:TIMETABLE_timeID_hasdate    rdfs:range      qua:TIMETABLE_date

qua:ENROL_studID                rdf:type        mDR:QuasiClass
qua:ENROL_studID                mDR:reflects    qua:STUDENT_studID
qua:ENROL_studID                mDR:belongsTo   db:ENROL
qua:ENROL_studID                mDR:consistsOf  db:ENROL.studentID
qua:ENROL_timeID                rdf:type        mDR:QuasiClass
qua:ENROL_timeID                mDR:reflects    qua:TIMETABLE_timetabletID
qua:ENROL_timeID                mDR:belongsTo   db:ENROL
qua:ENROL_timeID                mDR:consistsOf  db:ENROL.timetableID
qua:ENROL_studID_hastimeID      rdf:type        mDR:virtualRelation
qua:ENROL_studID_hastimeID      rdfs:domain     qua:ENROL_studID
qua:ENROL_studID_hastimeID      rdfs:range      qua:ENROL_timeID
qua:ENROL_timeID_hasstudID      rdf:type        mDR:virtualRelation
qua:ENROL_timeID_hasstudID      rdfs:domain     qua:ENROL_timeID
qua:ENROL_timeID_hasstudID      rdfs:range      qua:ENROL_studID

db:STUDENT                  mDR:label    "STUDENT"
db:STUDENT                  mDR:comment  "University students"
db:STUDENT.studentID        mDR:label    "studentID"
db:STUDENT.studentID        mDR:comment  "Student unique identificator"
```

```
db:STUDENT.firstname          mDR:label    "firstname"
db:STUDENT.firstname          mDR:comment  "Student's first name"
db:STUDENT.surname            mDR:label    "surname"
db:STUDENT.surname            mDR:comment  "Student's second name"
db:STUDENT.facultyID          mDR:label    "facultyID"
db:STUDENT.facultyID          mDR:comment  "Student's faculty"
db:PhDSTUDENT                 mDR:label    "PhDSTUDENT"
db:PhDSTUDENT                 mDR:comment  "Additional info about PhD"
db:PhDSTUDENT.studentID       mDR:label    "studentID"
db:PhDSTUDENT.scholarship     mDR:label    "scholarship"
db:FACULTY                    mDR:label    "FACULTY"
db:FACULTY                    mDR:comment  "Faculties"
db:FACULTY.facultyID          mDR:label    "facultyID"
db:FACULTY.name               mDR:label    "name"
db:LECTURER                   mDR:label    "LECTURER"
db:LECTURER                   mDR:comment  "Teachers"
db:LECTURER.lecturerID        mDR:label    "lecturerID"
db:LECTURER.firstname         mDR:label    "firstname"
db:LECTURER.surname           mDR:label    "surname"
db:LECTURERDETAIL             mDR:label    "LECTURERDETAIL"
db:LECTURERDETAIL             mDR:comment  "Teachers' additional info"
db:LECTURERDETAIL.lecturerID mDR:label    "lecturerID"
db:LECTURERDETAIL.address     mDR:label    "address"
db:RESULT                     mDR:label    "RESULT"
db:RESULT                     mDR:comment  "Student marks table"
db:RESULT.studentID           mDR:label    "studentID"
db:RESULT.studentID           mDR:comment  "Foreign key to STUDENT table"
db:RESULT.courseID            mDR:label    "courseID"
db:RESULT.mark                mDR:label    "mark"
db:RESULT.mark                mDR:comment  "Students' course results"
db:COURSE                     mDR:label    "COURSE"
db:COURSE.courseID            mDR:label    "courseID"
db:COURSE.prerequisiteID      mDR:label    "prerequisiteID"
db:COURSE.name                mDR:label    "name"
db:FINALTHESIS                mDR:label    "FINALTHESIS"
db:FINALTHESIS.studentID      mDR:label    "studentID"
db:FINALTHESIS.lecturerID     mDR:label    "lecturerID"
db:FINALTHESIS.courseID       mDR:label    "courseID"
db:TIMETABLE                  mDR:label    "TIMETABLE"
db:TIMETABLE.timetableID      mDR:label    "timetableID"
db:TIMETABLE.courseID         mDR:label    "courseID"
db:TIMETABLE.lecturerID       mDR:label    "lecturerID"
db:TIMETABLE.facultyID        mDR:label    "facultyID"
db:TIMETABLE.date             mDR:label    "date"
db:ENROL                      mDR:label    "ENROL"
db:ENROL.studentID            mDR:label    "studentID"
db:ENROL.timetableID          mDR:label    "timetableID"
```

Príklad B.1: Príklad mDR ontológie prvej fázy

Dodatok C

Príklad mDR ontológie druhej fázy

Výstupom druhej fázy prezentovaného riešenia mDR je ontológia jednoznačne popisujúca mapovanie s dostatočnou informáciou pre prípadný prenos inštancií. Príkladom je takáto ontológia vyjadrujúca mapovanie databázy Obrázka 3.1 na ontológiu Obrázka 3.3. Zápis je opäť podobný zápisu trojicami. Používané menné priestory ostávajú skoro totožné, pribúda priestor pre prvky vstupnej ontológie:

- `db` - `http://muro.hejja.net/mDR/db/` pre zdroje tabuliek a stĺpcov
- `mDR` - `http://muro.hejja.net/mDR/mDR.rdf#` pre prvky mDR slovníka
- `qua` - `http://muro.hejja.net/mDR/qua/` pre zdroje reprezentujúce inštancie kvázi a virtuálnych prvkov
- `ex` - `http://www.example.org/` menný priestor vstupnej ontológie

Za pozornosť stojí spôsob namapovania `ex:attends`, keď sa v tomto prípade využíva kvázirelácia `qua:RESULT_studID_hascourID` manuálne pripravená v prvej fáze[1]. V druhej fáze sa z výstupu fázy prvej odstraňujú pre prenos inštancií všetky nepotrebné informácie.

```
qua:STUDENT_studID              rdf:type          mDR:QuasiClass
qua:STUDENT_studID              mDR:representsClass  ex:Student
qua:STUDENT_studID              mDR:belongsTo     db:STUDENT
qua:STUDENT_studID              mDR:consistsOf    db:STUDENT.studentID
qua:STUDENT_fname               rdf:type          mDR:QuasiAttribute
qua:STUDENT_fname               mDR:belongsTo     db:STUDENT
qua:STUDENT_fname               mDR:consistsOf    db:STUDENT.firstname
qua:STUDENT_studID_hasfname     rdf:type          mDR:virtualAttribute
qua:STUDENT_studID_hasfname     mDR:representsAttribute  ex:firstname
qua:STUDENT_studID_hasfname     rdfs:domain       qua:STUDENT_studID
qua:STUDENT_studID_hasfname     rdfs:range        qua:STUDENT_fname
```

[1] V Dodatku B je dokument prvej fázy vytvorený čisto automaticky, a tak sa v ňom táto kvázirelácia nenachádza.

```
qua:STUDENT_sname               rdf:type         mDR:QuasiAttribute
qua:STUDENT_sname               mDR:belongsTo    db:STUDENT
qua:STUDENT_sname               mDR:consistsOf   db:STUDENT.surname
qua:STUDENT_studID_hassname     rdf:type         mDR:virtualAttribute
qua:STUDENT_studID_hassname     mDR:representsAttribute  ex:secondname
qua:STUDENT_studID_hassname     rdfs:domain      qua:STUDENT_studID
qua:STUDENT_studID_hassname     rdfs:range       qua:STUDENT_sname

qua:PhDSTUDENT_studID           rdf:type         mDR:QuasiClass
qua:PhDSTUDENT_studID           mDR:representsClass  ex:Postgraduate
qua:PhDSTUDENT_studID           mDR:reflects     qua:STUDENT_studID
qua:PhDSTUDENT_studID           mDR:belongsTo    db:PhDSTUDENT
qua:PhDSTUDENT_studID           mDR:consistsOf   db:PhDSTUDENT.studentID
qua:PhDSTUDENT_sship            rdf:type         mDR:QuasiAttribute
qua:PhDSTUDENT_sship            mDR:belongsTo    db:PhDSTUDENT
qua:PhDSTUDENT_sship            mDR:consistsOf   db:PhDSTUDENT.scholarship
qua:PhDSTUDENT_studID_hassship  rdf:type         mDR:virtualAttribute
qua:PhDSTUDENT_studID_hassship  mDR:representsAttribute  ex:scholarship
qua:PhDSTUDENT_studID_hassship  rdfs:domain      qua:PhDSTUDENT_studID
qua:PhDSTUDENT_studID_hassship  rdfs:range       qua:PhDSTUDENT_sname

qua:LECTURER_lectID             rdf:type         mDR:QuasiClass
qua:LECTURER_lectID             mDR:representsClass  ex:Teacher
qua:LECTURER_lectID             mDR:belongsTo    db:LECTURER
qua:LECTURER_lectID             mDR:consistsOf   db:LECTURER.lecturerID
qua:LECTURER_fname              rdf:type         mDR:QuasiAttribute
qua:LECTURER_fname              mDR:belongsTo    db:LECTURER
qua:LECTURER_fname              mDR:consistsOf   db:LECTURER.firstname
qua:LECTURER_lectID_hasfname    rdf:type         mDR:virtualAttribute
qua:LECTURER_lectID_hasfname    mDR:representsAttribute  ex:firstname
qua:LECTURER_lectID_hasfname    rdfs:domain      qua:LECTURER_lectID
qua:LECTURER_lectID_hasfname    rdfs:range       qua:LECTURER_fname
qua:LECTURER_sname              rdf:type         mDR:QuasiAttribute
qua:LECTURER_sname              mDR:belongsTo    db:LECTURER
qua:LECTURER_sname              mDR:consistsOf   db:LECTURER.surname
qua:LECTURER_lectID_hassname    rdf:type         mDR:virtualAttribute
qua:LECTURER_lectID_hassname    mDR:representsAttribute  ex:secondname
qua:LECTURER_lectID_hassname    rdfs:domain      qua:LECTURER_lectID
qua:LECTURER_lectID_hassname    rdfs:range       qua:LECTURER_sname

qua:LECTURERDETAIL_lectID       rdf:type         mDR:QuasiClass
qua:LECTURERDETAIL_lectID       mDR:representsClass  ex:Teacher
qua:LECTURERDETAIL_lectID       mDR:reflects     qua:LECTURER_lectID
qua:LECTURERDETAIL_lectID       mDR:belongsTo    db:LECTURERDETAIL
qua:LECTURERDETAIL_lectID       mDR:consistsOf   db:LECTURERDETAIL.lecturerID
qua:LECTURERDETAIL_addr         rdf:type         mDR:QuasiAttribute
qua:LECTURERDETAIL_addr         mDR:belongsTo    db:LECTURERDETAIL
qua:LECTURERDETAIL_addr         mDR:consistsOf   db:LECTURERDETAIL.address
qua:LECTURERDETAIL_lectID_hasaddr  rdf:type      mDR:virtualAttribute
qua:LECTURERDETAIL_lectID_hasaddr  mDR:representsAttribute  ex:address
qua:LECTURERDETAIL_lectID_hasaddr  rdfs:domain   qua:LECTURERDETAIL_lectID
qua:LECTURERDETAIL_lectID_hasaddr  rdfs:range    qua:LECTURERDETAIL_addr

qua:RESULT_studID               rdf:type         mDR:QuasiClass
qua:RESULT_studID               mDR:representsClass  ex:Student
qua:RESULT_studID               mDR:reflects     qua:STUDENT_studID
qua:RESULT_studID               mDR:belongsTo    db:RESULT
qua:RESULT_studID               mDR:consistsOf   db:RESULT.studentID
qua:RESULT_courID               rdf:type         mDR:QuasiClass
qua:RESULT_courID               mDR:representsClass  ex:Course
qua:RESULT_courID               mDR:reflects     qua:COURSE_courID
qua:RESULT_courID               mDR:belongsTo    db:RESULT
qua:RESULT_courID               mDR:consistsOf   db:RESULT.courseID
qua:RESULT_studID_hascourID   rdf:type    mDR:virtualRelation
qua:RESULT_studID_hascourID   mDR:representsRelation  ex:attends
```

```
qua:RESULT_studID_hascourID   rdfs:domain qua:RESULT_studID
qua:RESULT_studID_hascourID   rdfs:range  qua:RESULT_courID

qua:COURSE_courID               rdf:type        mDR:QuasiClass
qua:COURSE_courID               mDR:representsClass  ex:Course
qua:COURSE_courID               mDR:belongsTo   db:COURSE
qua:COURSE_courID               mDR:consistsOf  db:COURSE.courseID
qua:COURSE_prereqID             rdf:type        mDR:QuasiClass
qua:COURSE_prereqID             mDR:representsClass  ex:Course
qua:COURSE_prereqID             mDR:reflects    qua:COURSE_courID
qua:COURSE_prereqID             mDR:belongsTo   db:COURSE
qua:COURSE_prereqID             mDR:consistsOf  db:COURSE.prerequisiteID
qua:COURSE_courID_hasprereqID   rdf:type        mDR:virtualRelation
qua:COURSE_courID_hasprereqID   mDR:representsRelation  ex:prerequisite
qua:COURSE_courID_hasprereqID   rdf:domain      qua:COURSE_courID
qua:COURSE_courID_hasprereqID   rdf:range       qua:COURSE_prereqID
qua:COURSE_name                 rdf:type        mDR:QuasiAttribute
qua:COURSE_name                 mDR:belongsTo   db:COURSE
qua:COURSE_name                 mDR:consistsOf  db:COURSE.name
qua:COURSE_courID_hasname       rdf:type        mDR:virtualAttribute
qua:COURSE_courID_hasname       mDR:representsRelation  ex:name
qua:COURSE_courID_hasname       rdfs:domain     qua:COURSE_courID
qua:COURSE_courID_hasname       rdfs:range      qua:COURSE_name

qua:FINALTHESIS_studIDlectIDcourID  rdf:type        mDR:QuasiClass
qua:FINALTHESIS_studIDlectIDcourID  mDR:representsClass  ex:Thesis
qua:FINALTHESIS_studIDlectIDcourID  mDR:belongsTo   db:FINALTHESIS
qua:FINALTHESIS_studIDlectIDcourID  mDR:consistsOf  db:FINALTHESIS.studID
qua:FINALTHESIS_studIDlectIDcourID  mDR:consistsOf  db:FINALTHESIS.lectID
qua:FINALTHESIS_studIDlectIDcourID  mDR:consistsOf  db:FINALTHESIS.courID
qua:FINALTHESIS_studID          rdf:type            mDR:QuasiClass
qua:FINALTHESIS_studID          mDR:reflects        db:STUDENT.studID
qua:FINALTHESIS_studID          mDR:belongsTo       db:FINALTHESIS
qua:FINALTHESIS_studID          mDR:consistsOf      db:FINALTHESIS.studID
qua:FINALTHESIS_studIDlectIDcourID_hasstudID  rdf:type    mDR:virtualRelation
qua:FINALTHESIS_studIDlectIDcourID_hasstudID  mDR:representsRelation  ex:author
qua:FINALTHESIS_studIDlectIDcourID_hasstudID  rdfs:domain qua:FINALTHESIS_studIDlectIDcourID
qua:FINALTHESIS_studIDlectIDcourID_hasstudID  rdfs:range  qua:FINALTHESIS_studID
qua:FINALTHESIS_studID_hasstudIDlectIDcourID  rdf:type    mDR:virtualRelation
qua:FINALTHESIS_studID_hasstudIdlectIDcourID  mDR:representsRelation  ex:writes
qua:FINALTHESIS_studID_hasstudIDlectIDcourID  rdfs:domain qua:FINALTHESIS_studID
qua:FINALTHESIS_studID_hasstudIDlectIDcourID  rdfs:range  qya:FINALTHESIS_studIDlectIDcourID
qua:FINALTHESIS_lectID          rdf:type        mDR:QuasiClass
qua:FINALTHESIS_lectID          mDR:reflects    db:LECTURER.lectID
qua:FINALTHESIS_lectID          mDR:belongsTo   db:FINALTHESIS
qua:FINALTHESIS_lectID          mDR:consistsOf  db:FINALTHESIS.lectID
qua:FINALTHESIS_studIDlectIDcourID_haslectID  rdf:type    mDR:virtualRelation
qua:FINALTHESIS_studIDlectIDcourID_haslectID  mDR:representsRelation  ex:isSupervised
qua:FINALTHESIS_studIDlectIDcourID_haslectID  rdfs:domain qua:FINALTHESIS_studIDlectIDcourID
qua:FINALTHESIS_studIDlectIDcourID_haslectID  rdfs:range  qua:FINALTHESIS_lectID
qua:FINALTHESIS_courID          rdf:type        mDR:QuasiClass
qua:FINALTHESIS_courID          mDR:representsClass  ex:Course
qua:FINALTHESIS_courID          mDR:reflects    db:COURSE.courID
qua:FINALTHESIS_courID          mDR:belongsTo   db:FINALTHESIS
qua:FINALTHESIS_courID          mDR:consistsOf  db:FINALTHESIS.courID

qua:TIMETABLE_courID            rdf:type        mDR:QuasiClass
qua:TIMETABLE_courID            mDR:representsClass  ex:Course
qua:TIMETABLE_courID            mDR:reflects    qua:COURSE_courID
qua:TIMETABLE_courID            mDR:belongsTo   db:TIMETABLE
qua:TIMETABLE_courID            mDR:consistsOf  db:TIMETABLE.courseID
qua:TIMETABLE_lectID            rdf:type        mDR:QuasiClass
qua:TIMETABLE_lectID            rdf:representsClass  ex:Teacher
qua:TIMETABLE_lectID            mDR:reflects    qua:LECTURER_lectID
qua:TIMETABLE_lectID            mDR:belongsTo   db:TIMETABLE
```

```
qua:TIMETABLE_lectID            mDR:consistsOf  db:TIMETABLE.lecturerID

qua:ENROL_studID                rdf:type        mDR:QuasiClass
qua:ENROL_studID                mDR:representsClass   ex:Student
qua:ENROL_studID                mDR:reflects    qua:STUDENT_studID
qua:ENROL_studID                mDR:belongsTo   db:ENROL
qua:ENROL_studID                mDR:consistsOf  db:ENROL.studentID

db:STUDENT                    mDR:label    "STUDENT"
db:STUDENT.studentID          mDR:label    "studentID"
db:STUDENT.firstname          mDR:label    "firstname"
db:STUDENT.surname            mDR:label    "surname"
db:STUDENT.facultyID          mDR:label    "facultyID"
db:PhDSTUDENT                 mDR:label    "PhDSTUDENT"
db:PhDSTUDENT.studentID       mDR:label    "studentID"
db:PhDSTUDENT.scholarship     mDR:label    "scholarship"
db:FACULTY                    mDR:label    "FACULTY"
db:FACULTY.facultyID          mDR:label    "facultyID"
db:FACULTY.name               mDR:label    "name"
db:LECTURER                   mDR:label    "LECTURER"
db:LECTURER.lecturerID        mDR:label    "lecturerID"
db:LECTURER.firstname         mDR:label    "firstname"
db:LECTURER.surname           mDR:label    "surname"
db:LECTURERDETAIL             mDR:label    "LECTURERDETAIL"
db:LECTURERDETAIL.lecturerID mDR:label    "lecturerID"
db:LECTURERDETAIL.address     mDR:label    "address"
db:RESULT                     mDR:label    "RESULT"
db:RESULT.studentID           mDR:label    "studentID"
db:RESULT.courseID            mDR:label    "courseID"
db:RESULT.mark                mDR:label    "mark"
db:COURSE                     mDR:label    "COURSE"
db:COURSE.courseID            mDR:label    "courseID"
db:COURSE.prerequisiteID      mDR:label    "prerequisiteID"
db:COURSE.name                mDR:label    "name"
db:FINALTHESIS                mDR:label    "FINALTHESIS"
db:FINALTHESIS.studentID      mDR:label    "studentID"
db:FINALTHESIS.lecturerID     mDR:label    "lecturerID"
db:FINALTHESIS.courseID       mDR:label    "courseID"
db:TIMETABLE                  mDR:label    "TIMETABLE"
db:TIMETABLE.timetableID      mDR:label    "timetableID"
db:TIMETABLE.courseID         mDR:label    "courseID"
db:TIMETABLE.lecturerID       mDR:label    "lecturerID"
db:TIMETABLE.facultyID        mDR:label    "facultyID"
db:TIMETABLE.date             mDR:label    "date"
db:ENROL                      mDR:label    "ENROL"
db:ENROL.studentID            mDR:label    "studentID"
db:ENROL.timetableID          mDR:label    "timetableID"
```

Príklad C.1: Príklad mDR ontológie druhej fázy

Dodatok D

Obsah priloženého CD

Súčasťou práce je CD obsahujúce aplikáciu JmDR, použité knižnice, MySQL server, ukážkové vstupy a výstupy a text tejto práce. Nosič je štrukturovaný nasledovne (spomenutý je len tu podstatný obsah):

- `JmDR/` - aplikácia JmDR
- `JmDR/src/` - zdrojové súbory aplikácie
- `JmDR/target/` - preložená aplikácia
- `JmDR/lib/` - používané knižnice (Jena a MySQL Connector)
- `JmDR/javadoc/` - kompletná Javadoc dokumentácia k zdrojom
- `JmDR/guide.html` - užívateľská príručka
- `JmDR/build.xml` - kompilačný a spúšťací ANT skript
- `JmDR/mDR.rdf` - mDR slovník
- `MySQL_install/` - adresár s inštalačnými súbormi pre MySQL server
- `thesis/` - text diplomovej práce vo formátoch PDF a DVI
- `university_example/universityDB_mysql.sql` - vytvárací MySQL skript databázy z Obrázka 3.1
- `university_example/universityONT1.rdf` - RDFS ontológia z Obrázka 3.2
- `university_example/universityONT2.rdf` - RDFS ontológia z Obrázka 3.3
- `university_example/jmdr_output/mDR1_university.rdf` - príklad výstupu prvej fázy JmDR (zdrojovou databázou databáza z Obrázka 3.1)

– `university_example/jmdr_output/mDR2_university.rdf` - príklad finálneho výstupu druhej fázy JmDR (mapovanie na ontológiu Obrázka 3.3)

Literatúra

[1] An Y., Borgida A., Mylopoulos J.: *Building Semantic Mappings from Databases to Ontologies*. In Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06), Boston, MA, 2006. http://www.cs.toronto.edu/semanticweb/maponto/papers/aaai06-nectar.pdf

[2] An Y., Borgida A., Mylopoulos J.: *Inferring Complex Semantic Mappings between Relational Tables and Ontologies from Simple Correspondences*. In Proceedings of On The Move to Meaningful Internet Systems (OTM'05): 1152-1169. Springer Verlag, 2005. http://www.cs.utoronto.ca/~yuana/research/publications/odbase05.paper13.pdf

[3] Barrasa J., Corcho O., Gómez-Pérez A.: *Fund Finder: A case study of database-to-ontology mapping*. Semantic Integration Workshop, International Semantic Web Conference (ISWC) 2003, Sanibel Island, Florida, September 2003. http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-82/SI_paper_02.pdf

[4] Barrasa J., Corcho O., Gómez-Pérez A.: *R_2O, an Extensible and Semantically Based Database-to-ontology Mapping Language*. FU Berlin, 2004. http://www.cs.man.ac.uk/ocorcho/documents/SWDB2004_BarrasaEtAl.pdf

[5] Bizer C.: *D2R Map Language Specification*. FU Berlin, 2003. http://www.wiwiss.fu-berlin.de/suhl/bizer/d2rmap/D2Rmap.htm

[6] Dou D., LePendu P., Kim S. ,Qi P.: *Integrating Databases into the Semantic Web through an Ontology-based Framework*. 22nd International Conference on Data Engineering Workshops (ICDEW'06): 56, Atlanta, 2006. http://www.cs.uoregon.edu/~dou/research/papers/swdb06.pdf

[7] Gómez-Pérez A.: *Ontology Languages for the Semantic Web*. IEEE Intelligent Systems: 54-60, Január 2002. http://ranger.uta.edu/~alp/ix/readings/ontologyLanguages.pdf

[8] Gruber T.: *A translation approach to portable ontologies*. Knowledge Acquisition 5(2):199-220, Apríl 1993. http://tomgruber.org/writing/ontolingua-kaj-1993.pdf

[9] Handschuh S., Staab S., Volz R..: *On Deep Annotation*. 12th International World Wide Web Conference, Budapešť, Máj 2003. http://www-ksl.stanford.edu/KSL_Abstracts/KSL-92-71.html

[10] Chen H., Wang Y., Wang H., Mao Y., Tang J., Zhou C., Yin A., Wu Z.: *Towards a Semantic Web of Relational Databases: a Practical Semantic Toolkit and an In-Use Case from Traditional Chinese Medicine*. International Semantic Web Conference (ISWC) 2006: 750-763, Springer Verlag, November 2006. http://iswc2006.semanticweb.org/items/Chen2006kx.pdf

[11] Korotkiy M., Top J. L.: *From Relational Data to RDFS Models*. International Conference on Web Engineering (3140 LNCS): 430-434, Springer 2004. http://springerlink.metapress.com/content/adctlqvgd4nypajp/fulltext.pdf

[12] Passin B. T.: *Explorer's Guide to the Semantic Web*. Manning Publications, Greenwich, 2004.

[13] Stojanovic L., Stojanovic N., Volz R.: *Migrating data-intensive Web sites into the Semantic Web*. Symposium on Applied Computing, Madrid, Marec 2002. http://www.fzi.de/KCMS/kcms_file.php?action=link&id=46

[14] Stojanovic L., Stojanovic N., Volz R.: *A Reverse Engineering Approach for Migrating Data-intensive Web Sites to the Semantic Web*. Intelligent Information Processing, Montreal, 2002. http://www.aifb.uni-karlsruhe.de/WBS/nst/docs/papers/IIPv31finalv1.pdf

[15] Svátek V.: *Ontologie a WWW*. DATAKON, Brno, 2002. http://nb.vse.cz/~svatek/onto-www.pdf

[16] Švihla M., Jelínek I.: *Sémantický web a automatické generování jeho obsahu*. Tvorba softwaru 2004: 294-300. Ostrava 2004.

[17] Tauberer J..: *What is RDF and what is it good for?*. 2006. http://www.rdfabout.com/intro/

[18] W3C: *RDF Primer*. W3C, 2004. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

[19] W3C: *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C, 2004. http://www.w3.org/TR/rdf-schema/