Jedan pristup za reprezentaciju šeme relacione baze podataka standardnom UML notacijom

Dražen Brđanin, Slavko Marić
Elektrotehnički fakultet
Univerzitet u Banjoj Luci
Banja Luka, Bosna i Hercegovina
bdrazen@etfbl.net, ms@etfbl.net

Zvjezdan Spasić Pavković
Institut RT-RK
Banja Luka, Bosna i Hercegovina
zvjezdanspasic@gmail.com

Sažetak—**U ovom radu predložen je jedan pristup za reprezentaciju šeme relacione baze podataka primjenom standardne UML notacije. U odnosu na postojeće pristupe, koji se zasnivaju na primjeni specijalizovane notacije, tj. UML profila, u ovom radu se za reprezentaciju šeme relacione baze podataka koristi standardni dijagram klasa. Predloženi pristup zasniva se na korišćenju operacija za modelovanje ključeva, odnosno na činjenici da se standardom uređeni redoslijed parametara operacije, u klasi koja reprezentuje relacionu šemu (tabelu), može iskoristiti za modelovanje redoslijeda segmenata u ključevima. Predloženi pristup ilustrovan je na jednom modelu u procesu direktnog inžinjeringa relacione baze podataka.**

Ključne riječi-šema relacione baze podataka; UML; dijagram klasa; profil; Eclipse-Topcased, direktni inžinjering (key words)

I. UVOD

Relacioni model baza podataka [1] prošao je dug razvojni put od 1970-ih godina i postao dominantan model organizacije baza podataka. Opis sveukupne strukture podataka i ograničenja u relacionoj bazi podataka specifikuje se šemom relacione baze podataka. U procesu projektovanja relacione baze podataka, šema relacione baze podataka predstavlja prelazni model između konceptualnog modela podataka (platformski nezavisan model) i fizičke baze podataka, i kao takva predstavlja model koji u određenoj mjeri reflektuje specifičnosti implementacije (specifični tipovi podataka i dr.). U postojećoj literaturi ne postoji jedinstven niti standardizovan pristup za reprezentaciju šeme relacione baze podataka. Postojeći alati za projektovanje baza podataka (i komercijalni i *open-source*) koriste različite notacije za reprezentaciju šeme, pri čemu se dominantno koriste tradicionalne notacije kao što je IE [2]. S obzirom na to da je portabilnost takvih modela veoma ograničena, sve širu primjenu ima UML.

UML (*Unified Modeling Language*) [3] je standardni grafički jezik namijenjen za projektovanje, specifikaciju, vizuelizaciju i dokumentovanje softverskih sistema u različitim fazama njihovog životnog ciklusa. Široku primjenu UML zahvaljuje nezavisnosti notacije od procesa modelovanja, ali i otvorenosti, jer je standardnu notaciju moguće proširivati i specijalizovati za primjene u različitim domenima. Skup proširenja standardne UML notacije za primjenu u nekom specifičnom domenu naziva se UML profil. S obzirom na to da šema relacione baze podataka predstavlja platformski specifičan model, većina postojećih radova, predloženih pristupa i realizovanih alata za UML reprezentaciju šeme relacione baze podataka zasniva se na primjeni specijalizovane UML notacije, odnosno na primjeni UML profila. U ovom radu predložen je jedan pristup za vizuelizaciju šeme relacione baze podataka, koji se zasniva na primjeni standardne UML notacije, čime se eliminiše potreba za definisanjem i primjenom domenski specifičnog profila, što za direktnu posljedicu ima jednostavniji i efikasniji proces projektovanja relacione baze podataka.

Rad je organizovan na sljedeći način. Nakon uvodnog dijela, u drugom dijelu dat je kratak pregled postojećih pristupa za UML reprezentaciju šeme relacione baze podataka. Predloženi pristup prikazan je u trećem dijelu. U četvrtom dijelu ilustrovana je primjena predloženog pristupa u procesu direktnog inžinjeringa relacione baze podataka. Na kraju su dati zaključci i pravci za nastavak istraživanja.

II. POSTOJEĆI PRISTUPI ZA UML REPREZENTACIJU ŠEME RELACIONE BAZE PODATAKA

UML reprezentacija šeme relacione baze podataka je predmet istraživanja još od početka razvoja UML-a. Iako je OMG (*Object Management Group*) objavila poziv [4] za prijedlog UML profila namijenjenog za modelovanje baza podataka prije više od deset godina (2005), još uvijek ne postoji standardizovan pristup za UML-zasnovanu reprezentaciju šeme relacione baze podataka.

Prvu značajnu industrijsku implementaciju (*Rational Software Corp.*) UML profila za projektovanje baza podataka prikazali su Naiburg i Maksimchuk u [5]. Većina svih narednih prijedloga za UML-zasnovanu reprezentaciju šeme relacione baze podataka slijedila je tu inicijalnu specifikaciju, koja podrazumijeva: a) da se tabela reprezentuje klasom sa odgovarajućim stereotipom, b) da se kolona u nekoj tabeli reprezentuje odgovarajućim atributom u odnosnoj klasi, c) da se kolone koje pripadaju ključevima, reprezentuju atributima sa specifičnim stereotipovima («PK», «FK» itd.), pri čemu se ne prikazuje redoslijed kolona u složenim ključevima, d) da se veze između tabela reprezentuju (kompozitnim) asocijacijama sa specifičnim stereotipovima («identifying»/«non-identifying»), e) da se ograničenja u šemi relacione baze podataka reprezentuju operacijama sa specifičnim stereotipovima («PK», «FK» itd.), f) da se indeksi reprezentuju operacijama sa specifičnim stereotipom («INDEX»).

Li i Zhao u [6] predlažu korišćenje specifičnih označenih vrijednosti za prikazivanje kolona koje pripadaju ključevima i njihovog redoslijeda u okviru složenih ključeva. Za reprezentaciju veza između tabela, Li i Zhao predlažu korišćenje zavisnosti sa specifičnim stereotipom, pri čemu se za reprezentaciju ograničenja referencijalnog integriteta koriste specifične označene vrijednosti. Platformske specifičnosti šeme relacione baze podataka nisu razmatrane.

Ambler u [7] predlaže specifične stereotipove za modelovanje kolona koje pripadaju ključevima, ali se redoslijed kolona u okviru složenih ključeva reprezentuje označenim vrijednostima. Za razliku od [5], Ambler predlaže da se svaki indeks, definisan nad nekom tabelom, reprezentuje klasom sa stereotipom «INDEX», koja je sa korespondentnom klasom koja reprezentuje odnosnu tabelu, povezana odgovarajućom zavisnošću.

Lo i Hung u [8] predlažu UML profil za modelovanje operacija za pristup bazi podataka. Iako predloženi profil omogućava reprezentaciju šeme relacione baze podataka, fokus nije na efikasnoj vizuelizaciji šeme, nego na modelovanju upita.

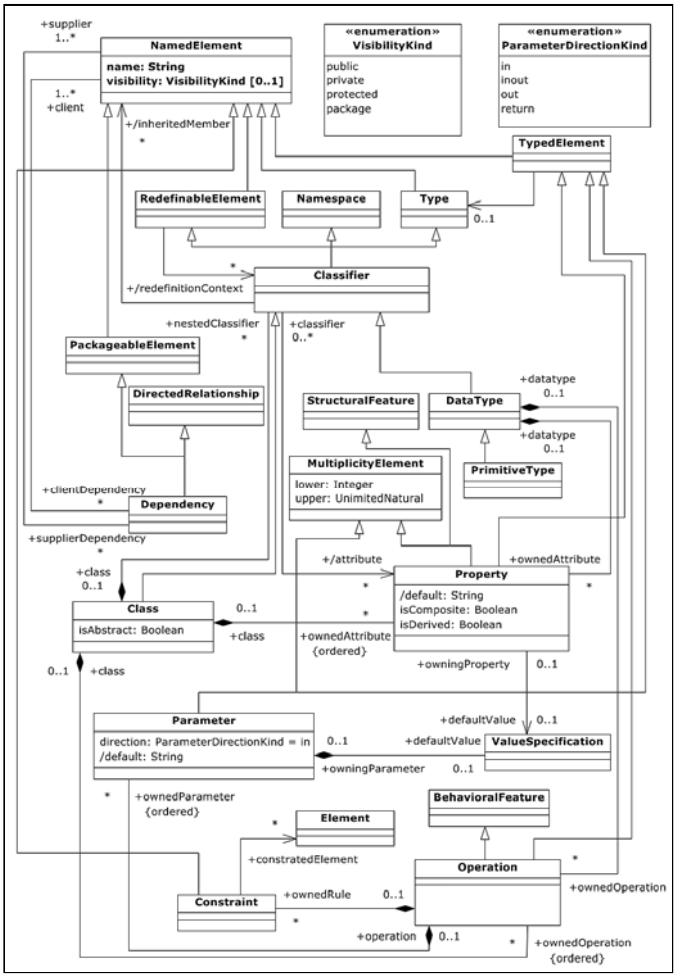
Marcos i dr. u [9] predlažu metodološki pristup za projektovanje objektno-relacionih baza podataka primjenom UML-a, pokrivajući cjelokupan proces projektovanja od konceptualnog do fizičkog nivoa. U pogledu UML-zasnovane reprezentacije šeme relacione baze podataka, oni slijede [5], osim za reprezentaciju indeksa (takođe predlažu korišćenje klasa sa odgovarajućim stereotipom, kao u [7]).

Tomić i dr. u [10] predlažu UML profil u kojem se ograničenja specifikuju operacijama sa odgovarajućim stereotipovima («PK», «FK» itd.), slično kao u [5], ali predlažu da se standardom uređeni redoslijed parametara operacije koristi za specifikaciju i vizuelizaciju redoslijeda segmenata u složenim ključevima. Predloženi pristup smanjuje broj dodatnih atributa u stereotipovima i omogućava efikasniju vizuelizaciju šeme relacione baze podataka. Ideja da se redoslijed parametara operacije koristi za specifikaciju i vizuelizaciju redoslijeda segmenata u složenim ključevima, predstavlja osnov za ovaj rad, jer omogućava da se osnovni elementi šeme relacione baze podataka predstave standardnom notacijom bez potrebe za definisanjem profila.

Pored prethodno navedenih radova, fokusiranih na specifikaciju UML profila za reprezentaciju šeme relacione baze podataka, u literaturi postoji veći broj radova u kojima se koristi ograničen skup proširenja standardnog dijagrama klasa za reprezentaciju samo nekih koncepata (tipično tabele i primarni ključevi). Međutim, takav model je bliži konceptualnom nego implementacionom modelu baze podataka.

III. PREDLOŽENI PRISTUP

Kao što je u uvodnom dijelu naglašeno, u ovom radu predložen je jedan pristup za reprezentaciju šeme relacione baze podataka standardnom UML notacijom. Na sl. 1 prikazan je odgovarajući dio UML metamodela [11] standardnog dijagrama klasa, koji je neophodan za reprezentaciju šeme relacione baze podataka.



Slika 1. Dio UML metamodela [11] dijagrama klasa relevantan za reprezentaciju šeme relacione baze podataka

Prvi koncept bitan za reprezentaciju šeme relacione baze podataka jeste sama šema. U postojećim radovima za reprezentaciju šeme koristi se paket sa odgovarajućim stereotipom (npr. «`databaseSchema`»). Takav stereotip uglavnom nema atribute, ili ima atribut kojim se reprezentuje ciljni DBMS (eng. *Database Management System*), što za vizuelizaciju šeme relacione baze podataka nije od presudnog značaja. Zbog toga se za reprezentaciju šeme može koristiti standardna notacija (`Package`).

Šema relacione baze podataka sadrži relacione šeme (tabele). U postojećoj literaturi za reprezentaciju tabela koristi se klasa sa stereotipom «`table`». U ovom radu se za reprezentaciju tabela koristi standardna notacija (`Class`).

Svaka tabela (relaciona šema) sadrži kolone (atribute). I za reprezentaciju kolona može da se koristi standardna notacija (`Property`), jer atributi metaklase `Property` omogućavaju modelovanje svih važnih svojstava kolona u tabelama. Tipično svojstvo, (ne)dozvoljena NULL vrijednost, jednostavno se reprezentuje multiplikativnošću. Podrazumijevana multiplikativnost [1..1] (koja se na dijagramu ne prikazuje) znači da atribut mora da ima vrijednost, odnosno da nisu dozvoljene NULL vrijednosti u datoj koloni. Donja multiplikativnost 0 ([0..1]) znači da su dozvoljene NULL vrijednosti u datoj

koloni. Drugo bitno svojstvo, tip podataka, ne može da se reprezentuje ugrađenim tipovima, jer standardni tipovi nisu adekvatni za reprezentaciju platformski specifičnih tipova. Međutim, projektant lako može da instancira potrebne platformski specifične tipove (`PrimitiveType`).

Tabela (može da) posjeduje primarni ključ. Primarni ključ, kao jedinstveni identifikator *torki* u relaciji, može biti prost (sastoji se od jednog atributa/kolone) ili složen (sastoji se od više atributa/kolona). Veoma važno svojstvo složenog primarnog ključa jeste redoslijed njegovih segmenata, tj. redoslijed atributa koji sačinjavaju primarni ključ. Ovaj redoslijed mora da se specifikuje prilikom kreiranja fizičke šeme relacione baze, odnosno generisanja korespondentnog DDL (eng. *Data Definition Language*) skripta, a kasnije direktno utiče na formiranje odgovarajuće indeksne strukture.

Kao što je vidljivo iz pregleda literature, primarni ključevi se reprezentuju na različite načine (stereotipovi «PK» na kolonama i operacijama, označene vrijednosti, ograničenja). Poseban izazov u reprezentaciji primarnog ključa upravo predstavlja redoslijed atributa u složenom primarnom ključu. Analizom UML metamodela (sl. 1) može se doći do zaključka da standardna notacija omogućava reprezentaciju primarnog ključa bez potrebe za uvođenjem dodatnih koncepata. Naime, činjenica da parametri operacije u klasi predstavljaju sekvencu, odnosno da je redoslijed parametara operacije uređen (ograničenje `ordered` na `OwnedProperty` kraju kompozitne asocijacije `Operation`←`Property`), omogućava reprezentaciju primarnog ključa pomoću odgovarajuće operacije, pri čemu parametri operacije treba da redom reprezentuju segmente odnosnog primarnog ključa. Za ilustraciju, ako relaciona šema `Ispit` ima složeni primarni ključ, kojeg redom sačinjavaju atributi `idPredmeta` i `datum`, tada primarni ključ možemo da reprezentujemo operacijom

```
PK(idPredmeta:int,datum:date).
```

Predloženi način za modelovanje primarnog ključa omogućava jednostavnu i potpunu vizuelizaciju primarnog ključa, jer se pomoću jednog standardnog koncepta mogu vizuelizovati svi segmenti primarnog ključa (nema potrebe za specijalizovanom notacijom kojom se dodatno označavaju atributi koji pripadaju primarnom ključu), kao i njihov redoslijed unutar primarnog ključa (što je ranije bio poseban izazov).

Na sličan način mogu da se reprezentuju i alternativni ključevi, pri čemu treba voditi računa o imenovanju operacija. Npr. činjenicu da neka tabela ima dva alternativna ključa mogli bismo reprezenotvati sljedećim operacijama:

```
AK_kljuc1(ak11,...,ak1N),
AK_kljuc2(ak21,...,ak2N),
```

gdje su `ak11,...,ak1N` i `ak21,...,ak2N` specifikacije parametara, koje redom reprezentuju redoslijed atributa u alternativnim ključevima `kljuc1` i `kljuc2`.

Još jedan koncept od izuzetnog značaja za reprezentaciju šeme relacione baze podataka jeste strani ključ. Strani ključ, slično primarnom ključu, može biti složen (u situacijama kada referencira složeni (primarni) ključ u referenciranoj tabeli). Za razliku od primarnog ključa, koji je jedinstven za tabelu, tabela može da ima više stranih ključeva (prema istoj ili različitim tabelama). Pored toga, uz strani ključ veže se i odgovarajuće ograničenje referencijalnog integriteta (RI), odnosno odgovarajuće akcije kojima se obezbjeđuje RI u bazi podataka. Sve pobrojano predstavlja još veći izazov u UML reprezentaciji šeme relacione baze podataka.

Predloženi pristup za modelovanje redoslijeda segmenata u ključu, zasnovan na standardom uređenom redoslijedu parametara operacije, omogućava da i strane ključeve reprezentujemo standardnom UML notacijom. Za ilustraciju, pretpostavimo da tabela `Polaganje` ima dva strana ključa. Prvi strani ključ (kolona id`Studenta`) referencira primarni ključ (kolona `indeks`) u tabeli `Student`. Drugi strani ključ (kojeg sačinjavaju kolone `idPredmeta` i `datum`) referencira primarni ključ u tabeli `Ispit` (ranije navedena za ilustraciju složenog primarnog ključa). Navedene strane ključeve možemo reprezentovati sljedećim operacijama:

```
FK_Student(idStudenta:char[10]),
FK_Ispit(idPredmeta:int,datum:date).
```

Iz navedenog primjera je očigledno da predloženi pristup omogućava jednostavno modelovanje redoslijeda segmenata u složenim stranim ključevima, a da se postojanje višestrukih stranih ključeva u jednoj tabeli može razriješiti korišćenjem operacija sa različitim imenima (slično modelovanju alternativnih ključeva). Uvođenje odgovarajućeg prefiksa u nazivu operacije (FK, AK, PK) olakšava direktni inžinjering iskaza za generisanje odgovarajućih ograničenja u ciljnoj bazi podataka — na osnovu operacije sa prefiksom FK treba da se generiše iskaz za dodavanje ograničenja po osnovu stranog ključa, itd.

Da bismo mogli generisati odgovarajući iskaz na osnovu operacije koja reprezentuje strani ključ, potrebno je još adekvatno specifikovati referenciranu tabelu. Kao što je vidljivo iz prethodnog primjera, naziv referencirane tabele može da se specifikuje i u samom nazivu operacije. Očigledno, operacija `FK_Student` reprezentuje strani ključ prema tabeli `Student`, a `FK_Ispit` prema tabeli `Ispit`.

U slučaju da tabela ima više stranih ključeva koji referenciraju istu tabelu, nazivi odnosnih operacija i korespondentnih ograničenja u ciljnoj bazi podataka treba da se razlikuju. Za ilustraciju, činjenicu da tabela `Student` ima dva strana ključa (prvi po osnovu prebivališta, a drugi po osnovu boravišta) koji referenciraju primarni ključ (`posta`) u tabeli `Mjesto`, možemo reprezentovati sljedećim operacijama:

```
FK_Mjesto_prebivaliste(idPrebivalista:int),
FK_Mjesto_boraviste(idBoravista:int).
```

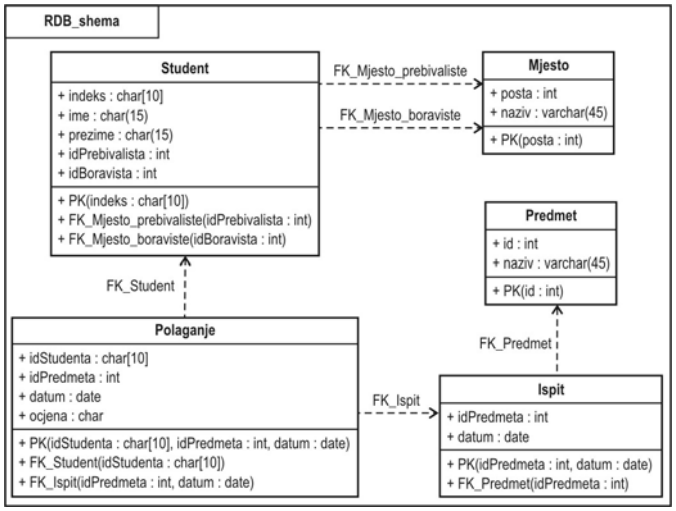
Dodatno još treba omogućiti specifikaciju RI akcija za svaki strani ključ. Standardna UML notacija omogućava specifikaciju ograničenja (`Constraint`) za operacije. Za svaku operaciju može da se definiše skup ograničenja (`ownedRule`). U konkretnom slučaju, za svaku operaciju, koja reprezentuje strani ključ, treba definisati odgovarajuće ograničenje kojim se specifikuju potrebne RI akcije. Iako se za specifikaciju ograničenja tipično koristi OCL (*Object Constraint Language*) [12], ograničenja je moguće specifikovati i na drugi način. Najjednostavniji način jeste direktna DDL specifikacija, npr.

```
ON DELETE RESTRICT ON UPDATE CASCADE.
```

Predloženi način za reprezentaciju stranog ključa je dovoljan za reprezentaciju svih bitnih aspekata vezanih za strani ključ. Dodatno je još poželjno vizuelizovati veze između

tabela na odgovarajući način, kako bi vizuelizovana šema relacione baze podataka (dijagram klasa) bila intuitivnija. U postojećim radovima predloženi su različiti pristupi (koji ponekad ne poštuju standardom definisanu semantiku) za vizuelizaciju veza između tabela. Najjednostavniji, i semantički ispravan, način za vizuelizaciju veza između tabela, jeste korišćenje zavisnosti (`Dependency`) usmjerene od referencirajuće prema referenciranoj tabeli. Poželjno je da naziv zavisnosti odgovara nazivu operacije koja reprezentuje korespondentni strani ključ, odnosno ograničenja referencijalnog integriteta.

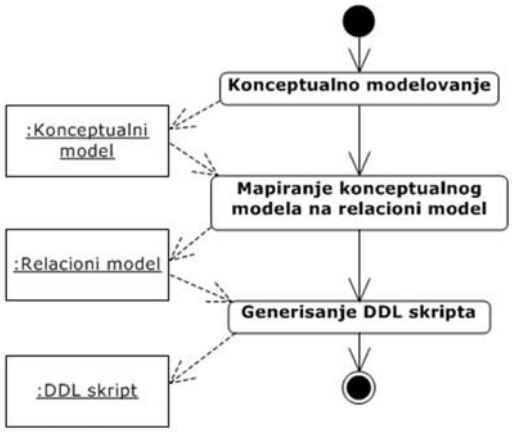
Na sl. 2 prikazana je jednostavna šema relacione baze podataka u skladu sa predloženim pristupom. Prikazana šema sadrži primjere koji su korišćeni za ilustraciju pristupa.



Slika 2. Ilustrativni primjer šeme relacione baze podataka predstavljene standardnom UML notacijom u skladu sa predloženim pristupom

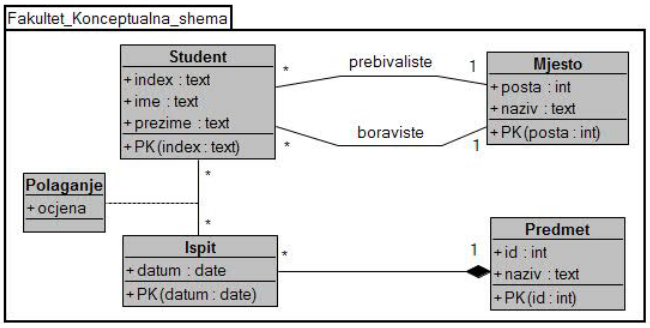
IV. DIREKTNI INŽINJERING RELACIONE BAZE PODATAKA ZASNOVAN NA PREDLOŽENOM PRISTUPU

Proces direktnog inžinjeringa relacione baze podataka (sl. 3) prolazi kroz tri faze: (i) konceptualno modelovanje, (ii) mapiranje konceptualnog modela na relacioni model, (iii) generisanje fizičke šeme, odnosno odgovarajućeg DDL skripta.



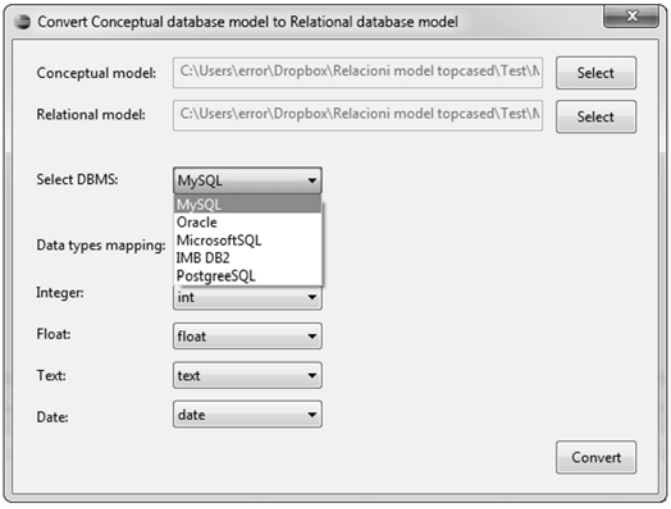
Slika 3. Proces direktnog inžinjeringa relacione baze podataka

Početna faza projektovanja baze podataka jeste konceptualno modelovanje, koje za cilj ima specifikaciju sveukupne strukture baze podataka na visokom nivou apstrakcije. Rezultat konceptualnog modelovanja jeste konceptualni model/šema. Konceptualna šema predstavlja semantički model podataka kojim se reprezentuju struktura podataka i njihove međusobne veze, njihova semantika i različita ograničenja. Na sl. 4 prikazan je UML dijagram klasa koji reprezentuje jednostavan konceptualni model, koji u radu koristimo kao polazni model za ilustraciju procesa direktnog inžinjeringa relacione baze podataka na osnovu predloženog pristupa. Prikazana konceptualna šema veoma je jednostavna i intuitivna pa izostavljamo njen detaljan opis.



Slika 4. Primjer polaznog UML konceptualnog modela

Nakon razvoja konceptualnog modela slijedi mapiranje konceptualne šeme na korespondentnu šemu relacione baze podataka. Relativno jednostavan skup pravila za mapiranje konceptualne šeme moguće je automatizovati. U našem slučaju, alat je realizovan kao Eclipse-Topcased [13] plugin, koji procesira ulaznu, XMI-serijalizovanu [14], UML konceptualnu šemu, primjenjuje transformaciona pravila i generiše XMI-serijalizovanu UML šemu relacione baze podataka. Vizuelizacija automatski generisane šeme vrši se pomoću ugrađene Topcased funkcionalnosti. Na sl. 5 prikazan je GUI (eng. *Graphical User Interface*) implementiranog alata, koji omogućava izbor polazne šeme, definisanje putanje i naziva ciljne šeme, izbor ciljnog DBMS, kao i izbor podrazumijevanih platformski specifičnih tipova podataka.



Slika 5. GUI realizovanog alata

Mapiranje elemenata konceptualne šeme u odgovarajuće elemente ciljne šeme relacione baze podataka odvija se prema sljedećim pravilima.

Prvo se mapiraju jaki entitetski tipovi, odnosno klase koje sadrže operaciju koja reprezentuje primarni ključ, tako što se u relacionom modelu kreira relaciona šema sa istim atributima i istim primarnim ključem kao polazni jaki entitetski tip.

Nakon toga se mapiraju veze generalizacije (specijalizacije). U našem slučaju se koristi "vertikalno" mapiranje — za svaku natklasu/potklasu se kreira korespondentna relaciona šema, a relaciona šema koja reprezentuje potklasu dobija atribute primarnog ključa natklase, pri čemu oni ujedno čine i primarni ključ i strani ključ prema šemi koja korespondira natklasi.

Slabi entitetski tip, tj. klasa koja reprezentuje stranu "dio" u kompoziciji, mapira se u relacionu šemu kojoj se dodaju atributi primarnog ključa relacione šeme koja odgovara jakom entitetskom tipu, tj. klasi na strani "cjelina". Primarni ključ šeme, koja korespondira slabom entitetskom tipu, zajedno čine atributi koji potiču od primarnog ključa jakog entitetskog tipa te atributi koji čine diskriminator u slabom entitetskom tipu. Dodati atributi istovremeno čine i strani ključ prema referenciranoj šemi (jaki entitetski tip).

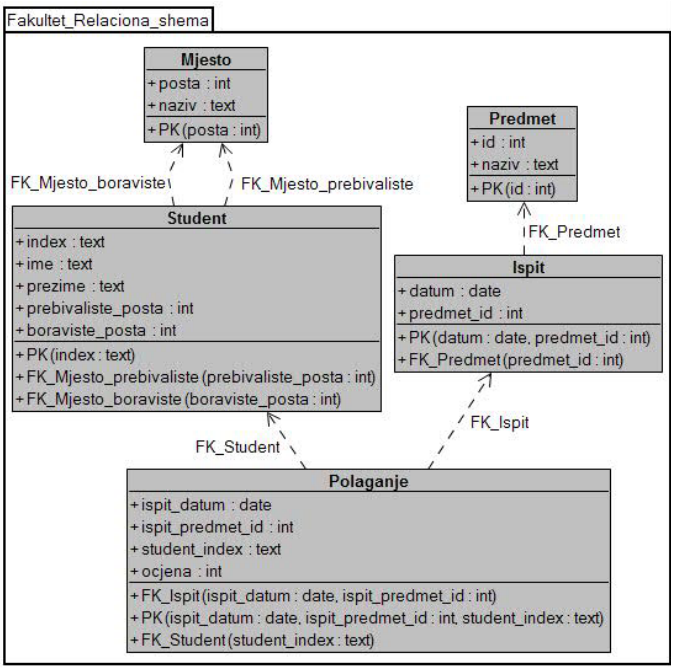
Nakon slabih entitetskih tipova mapiraju se vezne klase (vezni tipovi koji imaju sopstvene atribute), tako što se za svaku veznu klasu kreira korespondentna relaciona šema koja uključuje odgovarajuće atribute vezne klase i atribute primarnih ključeva klasa koje učestvuju u vezi. Dodati atributi čine primarni ključ u relacionoj šemi i istovremeno čine strane ključeve prema šemama koje korespondiraju klasama koje učestvuju u vezi.

Konačno, vrši se mapiranje veznih tipova. Ukoliko je veza tipa *:*, mapiranje je identično mapiranju veznih klasa. Tipovi veze *:1 ili 1:1 mogu se mapirati na više načina, zavisno od toga da li je učešće entiteta u tipu veze totalno ili parcijalno. Realizovani alat koristi metod kreiranja stranog ključa u šemi koja korespondira klasi na strani *, koji referencira primarni ključ u relacionoj šemi koja korespondira klasi na strani jedan.

Ukoliko postoje višeznačni atributi, oni se mapiraju u relacioni model tako što se kreira nova relaciona šema čiji su atributi odnosni višeznačni atribut i atributi koji čine primarni ključ u klasi u kojoj se nalazio višeznačni atribut. Svi navedeni atributi čine primarni ključ novokreirane relacione šeme, a strani ključ čine atributi primarnog ključa polazne klase u kojoj se nalazio višeznačni atribut.

Na sl. 6 prikazana je šema relacione baze podataka dobijena primjenom realizovanog alata na polazni konceptualni model.

Generisanu šemu relacione baze podataka, projektant može dodatno da mijenja i prilagođava, nakon čega treba generisati odgovarajući DDL skript. Postojeći CASE alati za projektovanje baze podataka imaju ugrađenu funkcionalnost za automatsko generisanje DDL kôda. Topcased omogućava jednostavno automatsko generisanje DDL kôda primjenom specijalizovanih M2T (eng. *Model To Text*) transformacionih jezika. Za ilustraciju, u ovom radu koristi se Acceleo [15]. Na sl. 7 prikazan je Acceleo transformacioni program za generisanje DDL skripta na osnovu UML šeme relacione baze podataka.



Slika 6. Programski generisana šema relacione baze podataka na osnovu polaznog konceptualnog modela sa sl. 4

```
[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/uml2/3.0.0/UML')]
[template public generateElement(aPackage : Package)]
[comment @main/]
[file (aPackage.name.concat('.ddl'), false, 'UTF-8')]
CREATE SCHEMA  [aPackage.name/];
  [for (aClass:Class | aPackage.ownedElement) after('\n')]
CREATE TABLE [aPackage.name.concat('.').concat(aClass.name)/]
(
  [for (aProperty:Property | aClass.ownedAttribute)
separator('\n') after('\n')]
  [aProperty.name/] [aProperty.type.name/][if
(aProperty.lower>0)] NOT NULL[/if],[/for]
  [for (aOperation:Operation | aClass.ownedOperation->
           select(o | o.name.startsWith('FK') or
o.name.equalsIgnoreCase('PK')) )
   separator(',\n') after('\n')]
     [if (aOperation.name.equalsIgnoreCase('PK'))]
  PRIMARY KEY ([for (op:Parameter | aOperation.ownedParameter)
     separator(', ')][op.name/][/for])[/if]
     [if (aOperation.name.startsWith('FK'))]
  CONSTRAINT [aOperation.name/]
    FOREIGN KEY ([for (op:Parameter |
aOperation.ownedParameter)
     separator(', ')][op.name/][/for])
    REFERENCES [for (aDependency:Dependency |
aPackage.ownedElement)]
      [if (aDependency.name.equalsIgnoreCase(aOperation.name))]
[aPackage.name.concat('.')/][aDependency.supplier.name/][/if][/
for]
[for (r:Constraint | aOperation.ownedRule)]
[r.name/][/for][/if]
  [/for]
);
[/for]
[/file]
[/template]
```

Slika 7. Acceleo transformacioni program za generisanje DDL skripta na osnovu šeme relacione baze podataka

Izvršavanjem transformacionog programa, nad programski generisanom šemom relacione baze podataka, dobijamo DDL skript prikazan na sl. 8.

```
CREATE SCHEMA  Fakultet_Relaciona_shema;
CREATE TABLE Fakultet_Relaciona_shema.Polaganje
(
  ispit_datum date NOT NULL,
  ispit_predmet_id int NOT NULL,
  student_index text NOT NULL,
  ocjena int NOT NULL,
  CONSTRAINT FK_Ispit
    FOREIGN KEY (ispit_datum, ispit_predmet_id)
    REFERENCES Fakultet_Relaciona_shema.Ispit
    ON DELETE RESTRICT ON UPDATE CASCADE,
  PRIMARY KEY (ispit_datum, ispit_predmet_id, student_index),
  CONSTRAINT FK_Student
    FOREIGN KEY (student_index)
    REFERENCES Fakultet_Relaciona_shema.Student
    ON DELETE RESTRICT ON UPDATE CASCADE
);
CREATE TABLE Fakultet_Relaciona_shema.Student
(
  index text NOT NULL,
  ime text NOT NULL,
  prezime text NOT NULL,
  prebivaliste_posta int NOT NULL,
  boraviste_posta int NOT NULL,
  PRIMARY KEY (index),
  CONSTRAINT FK_Mjesto_prebivaliste
    FOREIGN KEY (prebivaliste_posta)
    REFERENCES Fakultet_Relaciona_shema.Mjesto
    ON DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT FK_Mjesto_boraviste
    FOREIGN KEY (boraviste_posta)
    REFERENCES Fakultet_Relaciona_shema.Mjesto
    ON DELETE RESTRICT ON UPDATE CASCADE
);
CREATE TABLE Fakultet_Relaciona_shema.Predmet
(
  id int NOT NULL,
  naziv text NOT NULL,
  PRIMARY KEY (id)
);
CREATE TABLE Fakultet_Relaciona_shema.Ispit
(
  datum date NOT NULL,
  predmet_id int NOT NULL,
  PRIMARY KEY (datum, predmet_id),
  CONSTRAINT FK_Predmet
    FOREIGN KEY (predmet_id)
    REFERENCES Fakultet_Relaciona_shema.Predmet
    ON UPDATE CASCADE ON DELETE RESTRICT
);
CREATE TABLE Fakultet_Relaciona_shema.Mjesto
(
  posta int NOT NULL,
  naziv text NOT NULL,
  PRIMARY KEY (posta)
);
```

Slika 8. DDL skript generisan na osnovu šeme sa sl. 6

V. ZAKLJUČAK

U ovom radu predložen je jedan pristup za reprezentaciju šeme relacione baze podataka standardnom UML notacijom. Predloženi pristup temelji se na korišćenju standardom uređenog redoslijeda parametara u operaciji za reprezentaciju redoslijeda atributa unutar složenog ključa. Redoslijed atributa u postojećim pristupima predstavljao je poseban izazov, koji je uglavnom rješavan specijalizacijom standardne UML notacije.

Predloženi pristup ima nekoliko direktnih prednosti u odnosu na postojeće pristupe: (i) za reprezentaciju šeme relacione baze podataka koristi se standardna notacija bez potrebe za definisanjem profila, (ii) modelovanje šeme relacione baze podataka standardnom notacijom je jednostavnije i brže nego modelovanje specijalizovanom notacijom, jer ne zahtijeva primjenu specifičnog profila (iii) vizuelizacija šeme je bolja, jer se koristi standardna notacija bez specifičnih stereotipova, (iv) direktni inžinjering relacione baze podataka je jednostavniji, jer je korespondentni DDL skript za kreiranje ciljne fizičke šeme lakše generisati na osnovu standardnog UML modela.

Preliminarni rezultati primjene predloženog pristupa pokazuju da je osnovne elemente šeme relacione baze podataka moguće reprezentovati standardnom UML notacijom na jednostavan i intuitivan način, pa će nastavak istraživanja biti fokusiran na: (i) implementaciju pristupa u ADBdesign [16] alatu za automatizovano projektovanje baze podataka vođeno poslovnim modelom, (ii) analizu mogućnosti primjene standardne UML notacije za reprezentaciju i drugih značajnih koncepata, kao što su indeksi, pogledi, upiti, trigeri itd.

LITERATURA

[1] E. Codd, "A relational model of data for large shared data banks", Communications of the ACM, 13(5), pp. 377-387, 1970.

[2] J. Martin, Information Engineering, Prentice Hall, 1990.

[3] Object Management Group (OMG), Unified Modeling Language (OMG UML), v2.5, OMG, 2015.

[4] Object Management Group (OMG), Request for Proposal Information Management Metamodel (IMM), OMG, 2005.

[5] E. J. Naiburg and R. A. Maksimchuk, UML for Database Design, Addison-Wesley, Boston, USA, 2001.

[6] L. Li and X. Zhao, "UML specification of relational database", Journal of object technology, 2(5), pp. 87-100, 2003.

[7] S. W. Ambler, Agile Database Techniques, John Wiley and Sons, Indianapolis, USA, 2003.

[8] C. M. Lo and H. Y. Hung, "Towards a UML profile to relational database modeling", Applied Mathematics & Information Sciences, 8(2), pp. 733-743, 2014.

[9] E. Marcos, B. Vela, and J. M. Cavero, "A methodological approach for object-relational database design using UML", Software and Systems Modeling, 2, pp. 59-72, 2003.

[10] I. Tomic, D. Brdjanin, and S. Maric, "A novel UML profile for representation of a relational database schema", Proc. of EUROCON 2015, pp. 1-6, 2015.

[11] Object Management Group (OMG), Unified Modeling Language (OMG UML), Infrastructure, v2.3, OMG, 2010.

[12] ISO, Information technology Object Management Group Object Constraint Language (OCL). ISO/IEC 19507:2012, 2012.

[13] Topcased. http://www.topcased.org.

[14] OMG: OMG MOF 2 XMI Mapping Specification, v2.4.1. OMG, 2013.

[15] Acceleo. http://www.eclipse.org/acceleo/

[16] D. Brdjanin, and S. Maric, "An approach to automated conceptual database design based on the UML activity diagram", Computer Science and Information Systems, 9(1), pp. 249-283, 2012.

ABSTRACT

In this paper we propose an approach for representation of relational database schema by standard UML notation. Unlike the existing approaches using specialized notation (profiles), in this paper we use the standard class diagram for representation of a relational database schema. In the proposed approach, keys are modeled by class operations, since the standardized order of operation parameters can be used to represent the order of key segments. The proposed approach is illustrated by a simple model in the forward engineering database design process.

AN APPROACH TO STANDARD UML REPRESENTATION OF RELATIONAL DATABASE SCHEMA

Drazen Brdjanin, Slavko Maric, Zvjezdan Spasic Pavkovic