Зборник радова Факултета техничких наука, Нови Сад

UDK: 004.9
DOI: https://doi.org/10.24867/25BE09Vucinic

НАМЕНСКИ ЈЕЗИК ЗА МОДЕЛОВАЊЕ МИГРАЦИЈЕ РЕЛАЦИОНЕ БАЗЕ ПОДАТАКА У БАЗУ ПОДАТАКА ЗАСНОВАНУ НА ДОКУМЕНТИМА

A DOMAIN SPECIFIC LANGUAGE FOR MIGRATION OF RELATIONAL DATABASE TO DOCUMENT-ORIENTED DATABASE

Милица Вучинић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИЧКО И РАЧУНАРСКО ИНЖЕЊЕРСТВО

Кратак садржај – *У овом раду представљен је текстуални наменски језик за моделовање миграције релационе базе података у базу података засновану на документиа. Модел миграције састоји се од модела релационе базе, модела базе засноване на документима и моделу мапирања. На основу језика, могуће је развити више генератора при чему су у овом раду описани генератори за креирање валидатора колекције и индекса (миграција ограничења) као и генератори за сваку врсту мапирања миграције (мигрирање података). Резултујући код је генерисан у програмском језику python. За развој мета-модела коришћено је радно окружење Eclipse Modeling Framework, радни оквир Xtext за развој текстуалне синтаксе и језик Xtend за развој генератора.*

Кључне речи: *релационе базе података, нерелационе базе података, миграција, наменски језици*

Abstract – *In this thesis, we present a textual domain-specific language for migration of relational database to document-oriented database. Migration model consists of relational database model, document-oriented database model and mapping model. Multiple generators can be created based on this language and in this thesis presented generators are: collection validator and index generator (migration of constraints) and generator for each mapping type (data migration). Resulting code is generated in python programming language. Eclipse Modeling Framework was used for the creation of the meta-model. Xtext was used to develop the concrete language syntax, while Xtend was used for the implementation of documentation generators*

Keywords: *Relational databases, Nosql databases, Migration, Domain-Specific Languages*

1. УВОД

Успоном интернета и експоненцијалним растом количине података, појављују се нови изазови са којима релационе базе не успевају да се изборе у потпуности.

Нерелационе базе података, такође познате и као *NoSql* базе података [1], ступају на сцену са новим приступима решавања проблема. Пројектоване тако да подрже велике количине неструктуираних и полуструктуираних података, интеграцију са дистрибуираним системима као и скалабилне архитектуре много ефикасије од релационих система.

Приликом поређења *NoSql* база података са релационим, постаје евидентно да *NoSql* базе података нуде шири спектар модела података при чему сваки од њих задовољава специфичне захтеве и случајеве коришћења. Једна од тих врста модела јесте модел података заснован на документима.

Циљ овог рада јесте представити један од могућих начина миграције података и ограничења из постојеће релационе базе података у базу података засновану на документима. Решење које је у овом раду представљено јесте наменски језик за моделовање те миграције *Migration Domain Specific Language (migDsl)* као и пратећи генератори кода. Да би се моделовали детаљи овог процеса, описани наменски језик укључује концепте релационе базе података, концепте базе података засноване на документима и подржане врсте мапирања од којих се миграција састоји.

2. ТЕОРИЈСКЕ ОСНОВЕ МОДЕЛИМА ВОЂЕНОГ РАЗВОЈА СОФТВЕРА И НАМЕНСКИХ ЈЕЗИКА

Моделима вођен развој софтвера (*Model-Driven Software Engineering*, скраћено *MDSE*) се може дефинисати као методологија која примењује предности моделовања у активностима развоја софтвера. Као и свака методологија, састоји се од следећих аспеката: концепти, нотације, процеси, правила и алати [2].

Језици за моделовање представљају централну компоненту *MDSE* методологије. Могу се поделити на наменске језик (енг. *domain-specific language*) и језике опште намене (енг. *general-purpose language*). Језици опште намене представљају нотације за моделовање употребљиве у било ком сектору или домену. Типични примери језика опште намене су *UML (Unified Modeling Language)* и Петријеве мреже. Наменски језици су језици развијени за конкретан домен, контекст или компанију са циљем да омогуће моделовање прилагођено искључиво том домену. Сваки језик има три кључна елемента: апстрактну синтаксу (структура језика и начини комбинације

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији је ментор био др Милан Челиковић, доцент.

основних елемената), конкретну синтаксу (репрезентација језика) и семантику (значење елемената). Трансформације, поред самих модела, представљају главну компоненту *MDSE* методологије. Постоје две врсте трансформација: трансформације модела у модел и трансформације модела у текст.

3. ПРЕГЛЕД СТАЊА У ОБЛАСТИ

Традиционално, *etl* (*extract, transform, and load*) процес се обављао коришћењем алата, као што су *Informatica, Talend, SAP Data Services* и остали. Ови алати обично поседују *drag and drop* интерфејс који олакшава креирање *data pipeline*-ова. Овај начин је познат и као *tool-based* приступ.

Међутим, код *tool-based* приступа се јављају нека ограничења, па осим њега постоји и *code-based* приступ који подразумева писање програма и скрипти од стране програмера. Једна од главних предности *code-based* приступа је флексибилност. Са језицима као што су *python* и *R* могуће је извршити много комплексније трансформације које одговарају специфичном проблему. Осим тога, *code-based* приступ омогућава успешну сарадњу на нивоу тима, захваљујући контроли верзија. Још једна предност је могућност рада на *cloud* платформама што може донети уштеду и могућност скалирања.

У раду [3] изложен је приступ *code-based etl*-а чије команде се излажу коришћењем наменског језика, уместо програмског или упитног језика. Граматика је изражена у Бакус-Науровој форми, синтакса се темељи на релационој алгебри и резултат који настаје из *dsl* скрипте је код у програмском језику *Java*.

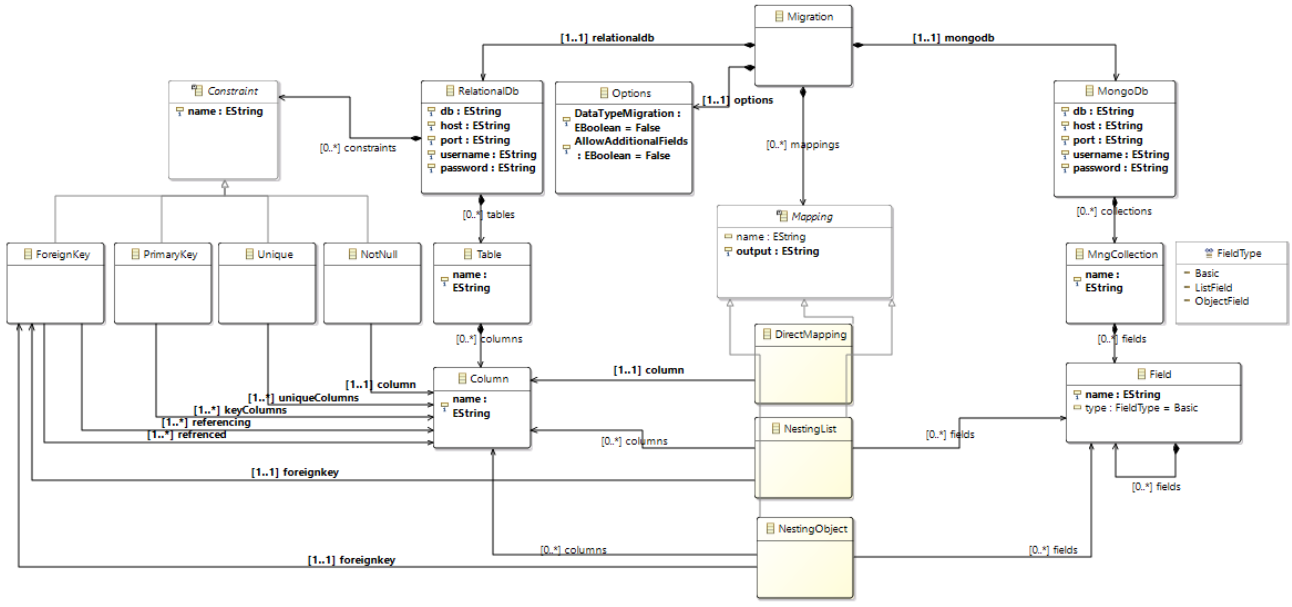
Циљ рада [4] јесте да се предложе правила трансформација за три различите категорије нерелацииох база: колонске, граф и базе оријентисане ка документима. Правило трансформације је дефинисано за сваку врсту везе која постоји у релационој бази података (један на један, један на више, више на више, наслеђивање-специјализација, наслеђивање-унија и агрегација). Што се тиче база оријентисаних ка документима, издвојена су два начина моделовања веза, а то су референцирање и угњеждавање, који уједно представљају и два могућа резултата трансформација.

Циљ рада [5] јесте да се установи која *NoSql* категорија подржава комплетну миграцију из релационе базе, знајући да су усвојене различите структуре које не имплементирају одређена правила релационог модела. Аутори су, након детаљне анализе релационог модела, установили 19 битних критеријума. Најбољи резултат, односно најмање губитка информација, има миграција у базу података засновану на документима.

Etl процеси и миграција података представљају кључне аспекте у управљању подацима и омогућавају ефикасну интеграцију, трансформацију и анализу података унутар организације. Познавање и разумевање различитих приступа и алата је од суштинског значаја за успех у модерном пословном окружењу а прави избор свакако зависи од природе проблема који се решава.

4. НАМЕНСКИ ЈЕЗИК ЗА МИГРАЦИЈУ РЕЛАЦИОНЕ БАЗЕ ПОДАТАКА У БАЗУ ПОДАТАКА ЗАСНОВАНУ НА ДОКУМЕНТИМА

Први корак у развоју наменског језика *migDsl* је креирање апстрактне синтаксе која је приказана помоћу мета-модела (*Слика 4.1.*). За креирање мета-модела коришћен је радни оквир *Eclipse Modeling Framework* (*EMF)* који као језик за мета-моделовање користи језик *Ecore* [6]. Ограничења су описана помоћу наменског језика *Object Constraint Language* (*OCL*), односно имплементације овог језика под називом *Eclipse OCL*. На основу апстрактне синтаксе креирана је конкретна синтакса, користећи радни оквир *Xtext*.



Слика 4.1. – *Апстрактна синтакса наменског језика migDsl*

4.1. Апстрактна синтакса

Коренски концепт апстрактне синтаксе је миграција (*Migration)*. Миграција мора да има једну релациону базу података (*RelationalDb*), једну *Mongo* базу података (*MongoDb*) и опције миграције (*Options*). Осим тога, коренски елемент може да има више мапирања кроз које се остварује процес миграције.

Релациона база може да има више табела (*Table*) од којих свака може имати више колона (*Column*). Са друге стране, *Mongo* база података има низ својих колекција (*MngCollection*) које се састоје од поља (*Field*). За разлику од колоне, поље може да има своја угњеждена поља, објекте и листе.Поља колекција и колоне табела представљају улазе и излазе мапирања по правима врсте мапирања.

Ради потребе мигрирања и самих ограничења у новонасталу базу података, осим табела, релациона база има и листу ограничења (*Constraint*). Уведен је концепт за сваку врсту ограничења, осим за *check* ограничење.

У језику *migDsl* тренутно су подржана три типа мапирања:

- *DirectMapping* - моделује директно мапирање постојеће колоне у релационој бази података на ново поље документа. Поседује само асоцијацију према колони.
- *NestingObject* - Резултат ове врсте мапирања је нови објекат чији је назив назначен у оквиру обележја *output* апстракне класе *Mapping.* Поља новог објекта су настала од референцираних колона (асоцијација *columns*) и постављена је и референца на њих после креирања (асоцијација *fields*). За ово мапирање мора постојати страни кључ при чему су изворне колоне из табеле у којој се налази референцирана колона тог страног кључа.
- *NestingList* - Разлика у односу на прошлу врсту мапирања је то што не настаје један објекат, већ листа објеката. Овога пута, референциране колоне мапирања морају припадати табели у којој се налази референцирајућа колона страног кључа (спојној табели).

4.2. Конкретна синтакса са примером

За наменски језик *migDsl* почетна верзија граматике креирана је на основу мета-модела описаног у претходном поглављу. У том процесу, овај радни оквир креће од коренског елемента, који је у овом случају Миграција.

Да би се кренуло са писањем конкретне синтаксе језика *migDsl*, потребно је пре свега попунити фајл конфигурације. Фајл треба да садржи креденцијале за конекције над изворном и одредишном базом као и опције миграције. У претходном поглалју, приказан је велики број референци од којих се метамодел састоји, односно за креирање мапирања, потребно је да концепти релационе базе већ постоје. Да корисник не би морао ручно да уноси део мета-модела који описује релациону шему, она је учитана у метамодел на основу садржаја фајла конфигурације.

Да би се добавила читава шема релационе базе података и учитала у метамодел, прави се низ упита ка речнику података. Конкретно, ка табелама *all_tables, all_tab_columns, user_constraints* и *user_cons_columns.*

Пример који је одабран за демонстрацију је релациона шема позоришта и креирање нове колекције *actors* у *Mongo* бази података (*Листинг 4.1.*).

У табели *ACTOR*, колона *NAME_SC* се мапира у поље *name*, а колона *STATUS_AC* у поље *status*. Веза глумца и позоришта је моделована кроз страни кључ *ACTOR_THEATER_FK*, стварајући објекат *theatre*. Табела *ASSIGNMENT* представља спојну таблу за везу глумца и улога. За добијање листе прошлих улога, користи се четврто мапирање. Пето мапирање директно мапира колону *SALARY* у поље *salary*.

```
{
            output: "actors.name"
            input: "ACTOR.NAME_AC"
}
{
            output: "actors.status"
            input: "ACTOR.STATUS_AC"
}
{
            NESTING_OBJECT
            output: "actors.theatre"
            inputs: ACTOR_THEATRE_FK
("THEATRE.NAME_TH","THEATRE.WEBSITE_TH","THEATRE.ADDRESS_TH")
AS ( "actors.theatre.name" , "actors.theatre.website",
"actors.theatre.address" )
}
{
            NESTING_LIST
            output: "actors.roles"
            inputs: ASSIGNMENT_ROLE_FK
("ROLE.NAME_RO","ROLE.TYPE_RO","ROLE.ID_RO")
AS ( "actors.roles.role_name" , "actors.roles.role_type" ,
"actors.roles.role_id" )
 }
{
             output: "actors.salary"
             input: "ACTOR.SALARY_AC"
 }
```

Листинг 4.1 - *Пример конкретне синтаксе језика migDsl*

5. ГЕНЕРАТОРИ КОДА

Генерисање кода подразумева генерисање два фајла у програмском језику *python*. Садржај првог фајла представља код за дефинисање ограничења, односно валидатора за сваку новокреирану колекцију. Осим тога, за сваку колекцију генеришу се и индекси. У обзир се узимају опције миграције које су постављене у фајлу конфигурације. Садржај другог фајла су *sql* упити ка изворној бази података, учитавање података, њихова трансофрмација у одговарајући облик за мапирање и пренос у одредишну базу података. Сви параметри конекције су преузети из мета-модела, а специфицирани су у конфигурационом фајлу.

Ако је опција *DataTypeMigration* постављена на *True*, за свако поље се умеће нови шаблон у коме се за поље налази изворна колона, прави упит ка релационој бази података који на основу назива колоне и табеле враћа тип податка. Затим се тај тип податка из *Oracle* базе мапира на одговарајући тип у *Mongo* бази података. Опција валидатора *additionalProperties* такође зависи од вредности у опцијама миграције. Ова опција дозвољава или забрањује појаву поља у колекцији која нису дефинисани унутар валидатора.

Приликом генерисања *python* фајла за миграцију података, пролази се кроз листу свих мапирања миграције, и у зависности од врсте мапирања позива се одговарајући шаблон. Пре овога, на самом почетку фајла, уметнути су и статички шаблони за потребне импорте и остваривање конекција. Пример шаблона за директно мапирање приказан је на листингу 5.1.

```
def String getDirectMappingContent(Mapping mapping, RelationalDb relDb){

		val map = mapping as DirectMapping
		val input_table_name =
Utils.getTableNameFromReferencedColumn(relDb, map.column)
		val collection_name =
Utils.getCollectionNameFromOutput(map.output)
		val output_field = Utils.getFieldNameFromOutput(map.output)
		'''
		# %% Direct mapping of «map.column.name»«IF
map.name!==null»:«map.name»«ENDIF»

		cursor = connection.cursor()
		cursor.execute("SELECT «map.column.name» FROM
«input_table_name»")
		rows = cursor.fetchall()
		cursor.close()
		simple_list = []
		simple_list = [item[0] for item in rows]

		collection = database['«collection_name»']

		for index,i in enumerate(simple_list):
			new_document = {"«output_field»": i}


			if collection.count_documents({}) == index:

				collection.insert_one(new_document)
			else:
				cursor = collection.find()
				count = 0
				for document in cursor:
				count += 1
				if count == index+1:

		collection.update_one({'_id': document['_id']}, {'$set':
new_document})
			'''
		}
```

Листинг 5.1 – *Шаблон за директно мапирање*

6. ЗАКЉУЧАК

Наменски језик *migDsl* омогућава креирање формалног модела једне миграције из релационе базе података у базу података засновану на документима. Миграција се састоји од низа мапирања који референцирају концепте релационе базе и базе засноване на документима, при чему су сви део апстрактне синтаксе језика. У раду је описано и више генератора који имају за циљ генерисање програмског кода. Резултујуће скрипте садрже код за извршавање сваког мапирања, валидатора и индекса за колекције. Доступне су и додатне опције миграције.

Овај наменски језик могуће је проширити концептима који би подржали миграцију у дистрибуирани систем. Осим генератора за миграцију података и валидацију шеме, могли би се додати и генератори скрипти за остале административне задатаке. На пример, скрипта за извршавање *backup*-а или конфигурацију појединачних кластера. Једна од предности овог наменског језика је његова проширивост. Поред три подржана типа мапирања, на једноставан начин је могуће додати још типова. Примери би били: мапирање у коме поље колоне настаје тако што се обави нека операција над колоном табеле или комбиновање више колона за резултујуће поље документа.

7. ЛИТЕРАТУРА

[1] Vatika Sharma, Meenu Dave. SQL and NoSQL Databases. Доступно на: https://www.researchgate.net/profile/Meenu-Dave/publication/303856633_SQL_and_NoSQL_Databases/links/5758557f08ae9a9c954a7573/SQL-and-NoSQL-Databases.pdf

[2] Marco Branbilla, Jordi Cabot, Manuel Wimmer, Model-Driven Software Engineering in Practice Morgan & Claypool Publishers 2012.

[3] Sunisa Junsawang, Yachai Limpiyakorn. A Domain Specific Langugae for Scripting ETL Process. 2017. Доступно на: http://www.wcse.org/WCSE_2017/041.pdf.

[4] Obaid Alotaibi, Eric Pardede. Transformation of Schema from Relational Database (RDB) to NoSQL Databases. 27.11.2019. Доступно на: https://www.mdpi.com/2306-5729/4/4/148.

[5] Abdelhak Erraji, Abderrahim Maizate, Mohammed Ouzzif. Multi-criteria Analysis Between Nosql Databases Categories Toward A Complete Migration From Relational Database. 15.1.2022. Доступно на: http://www.jatit.org/volumes/Vol100No1/6Vol100No1.pdf

[6] Eclipse Modeling Framework (EMF) документација. Доступно на: https://www.eclipse.org/modeling/emf

Кратка биографија:

Милица Вучинић рођена је 1999. год. у Невесињу, Босна и Херцеговина. Факултет техничких наука уписала је 2018. године. Дипломски рад из области Електротехника и рачунарство – Рачунарске науке и информатика одбранила је 2022. године. Мастер студије на Факултету техничких наука уписала је 2022. године.