Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.42
DOI: https://doi.org/10.24867/08BE22Gajic

МОДЕЛИ ПОДАТАКА ЕЛЕКТРОЕНЕРГЕТСКОГ СИСТЕМА ДЕФИНИСАНИ У ГРАФ ОРИЈЕНТИСАНИМ БАЗАМА ПОДАТАКА

POWER DATA MODELS DEFINED IN GRAPH ORIENTED DATABASES

Марко Гајић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *Овим радом обухваћено је истраживање рада граф оријентисаних база података са моделима података електроенергетских система. Улазни подаци су представљени у једној од RDF форми серијализације, док је као граф оријентисана база коришћена MarkLogic база података. Подаци се у бази података смештају у формату тројки (субјекат-предикат-објекат). Кроз развијену клијентску апликацију тестиране су CRUD (Create, Read, Update, Delete) операције ослањајући се на SPARQL језик. Такође, испитана је могућност закључивања коју MarkLogic база података поседује. Закључивање представља могућност MarkLogic базе података да на основу већ постојећих информација представљених као тројке генерише нове тројке. На крају је извршено тестирање перформанси базе података у зависности од величине улазних података и могућности закључивања над њима, као и тестирање конективности дефинисаних елемената електроенергетске мреже у случајевима са и без могућности закључивања.*

Кључне речи – *Модели података електроенергетског система, граф оријентисане базе података, закључивање, MarkLogic, SPARQL.*

Abstract – *Graph oriented databases and power data models are covered by this paper. Input data are presented as one of the forms of RDF serialization and graph-oriented database is MarkLogic database. Data in database are stored in triples format (subject-predicate-object). CRUD (Create, Read, Update, Delete) operations were tested with developed client application relying on SPARQL query language. Also, MarkLogic Inferencing option was tested. Inferencing presents capability of MarkLogic database to create new triples based on the existing information. An analysis of database performances based on input data size and inference capability is performed. Network connectivity test based of inference capability is performed, too.*

Key words – *Power data models, graph oriented databases, inferencing, MarkLogic, SPARQL.*

1. УВОД

У данашњим информационим системима постоји велики број различитих типова података. У зависности од типа података, разликује се више врста база података.

Поред традиционалних релационих база података, кључ-вредност организованих и документ оријентисаних база података, постоје и граф оријентисане базе података. Предност граф оријентисаних база података представља могућност прилагођавања променама података које се дешавају у реалном времену. Такође, пружају боље перформансе приликом претраге повезаних података, као и објектно оријентисан приступ приликом конструисања упита.

Данашњи информациони системи у електроенергетици презентују податке уз помоћ стандардизованог модела података који је заједнички за све системе. Такав модел података назива се CIM (*Common Information Model*) и његова појава представља решење проблема где је потребан један модел података који ће бити независан од апликације која га користи. Да би се са подаци преносили између апликација, CIM приступ предлаже њихово представљање у стандардизованој форми за размену података познатијој као RDF (Resource Description Framework). RDF податке представља у облику тројки субјекат-предикат-објекат.

Такви подаци се затим серијализују у један од формата RDF серијализације ради даљег рада са њима. Када подаци који су сада представљени у стандардизованом формату стигну у систем, могуће их је складиштити у одређену базу података. Осим самог складиштења, граф оријентисане базе података имају и друге могућности рада са подацима, као што је њихово конструисање, добављање и преглед доступних података. Језик који се користи при тим операцијама назива се SPARQL (Sparql Protocol and RDF Query Language) и представља стандардизован упитни језик над граф оријентисаним базама података.

2. ОПИС КОРИШЋЕНИХ ТЕХНОЛОГИЈА

Ово поглавље садржи кратак осврт на технологије, методе и принципе који су коришћени за имплементацију пројектног решења. Описане су само технологије и принципи који су највише утицали на имплементацију решења.

2.1 CIM

CIM (*Common Information Model*) представља апстрактни информациони модел који се користи за моделовање података електричне мреже и различите опреме која се користи у мрежи, као и моделовању података од значаја за управљање електроенергетском мрежом. Коришћењем овог модела података, компаније смањују трошко-

---

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био доц. др Милан Гаврић.

ве интеграција, што им омогућава да више ресурса усмере ка повећању функционалности и оптимизацији електричних система [1]. Креиран је да реши проблем који су предузећа имала са EMS (*Energy Management System*), зато што је сваки EMS користио своје технологије за креирање апликација. Као решење за тај проблем, произвођачи софтвера су прихватили CIM као основни информациони модел. CIM се ослања на XML и UML (*Unified Modeling Language*), док су две најкоришћеније серије стандарда IEC 61970 (модел мреже) и IEC 69168 (дистрибуција).

2.2 RDF

RDF (*Resource Description Framework*) представља стандард за размену података који је развијен од стране W3C [2]. У RDF се искази формирају у облику тројке (субјекат-предикат-објекат). Субјекат представља ресурс дефинисан преко URI-а, објекат представља особину субјекта и може бити дефинисан као ресурс или литерал, док предикат представља везу између субјекта и објекта. Да би се RDF подаци размењивали између различитих рачунарских система, потребно их је текстуално презентовати, односно серијализовати. Неке од стандардних форми серијализације RDF података су: Turtle, N-Triples, N-Quads, RDF/XML (CIMXML).

2.3 SPARQL

SPARQL (*Sparql Protocol and RDF Query Language*) представља стандардизован упитни језик за рад са RDF подацима, и као такав је погодан за рад са граф оријентисаним базама података. Користи се за добављање и манипулисање информацијама смештеним у бази података у RDF формату. Резултат упита може бити скуп тројки или граф [3]. Корисник са својом базом података комуницира уз помоћ SPARQL упита. Ако се погледа структура упита, она се може поделити на два дела, главу и тело упита. Глава разликује четири различите врсте које престављају израз за конструисање одговора упита. То су Select, Describe, Ask и Construct. Резултат упита се добија провером слагања тела упита са графом над којим се врши упит, односно повезивањем променљивих из тела упита са резултатима из графа. Тело упита може бити једноставна тројка субјекат-предикат-објекат ако су кориснику потребни најосновнији подаци, али се такође те једноставне тројке могу комбиновати са опционим деловима у случају да кориснику требају неке комплексније информације.

Поред упита, постоје и SPARQL Update опције. Синтаксно се незнатно разликују од упита и деле се у две групе: операције над графом (*create, drop, copy, move, add*) и операције над подацима (*insert data, delete data, clear, delete where, insert where, delete..insert where*).

2.4 Граф оријентисане базе података

Са повећањем броја података који захтевају чување као и операције претраге и управљања над њима, јавила се потреба за новим решењима у области складиштења података, односно потреба за базама података које поседују особине скалабилности како би могле да одговоре на све већи број информација које је потребно чувати. Такође, неопходно је да поседују подршку за један од упитних језика који ће омогућити брзу претрагу базе података и који ће клијенту приказати тражене податке у прегледној форми. Форма субјекат-предикат-објекат одговара нечему што је у математици познато као граф, по дефиницији математичка репрезентација објеката (чворови) и веза између њих (гране). Граф оријентисане базе података се организационо ослањају на горепоменуту математичку дефиницију графа, где чвор представља објекат, док грана представља везу између два објекта које спаја. Сваки чвор је дефинисан јединственим идентификатором (UID), скупом грана који долазе и одлазе од њега као и чворовима који су са њим повезани [4].

2.4.1 MarkLogic база података

MarkLogic база података је NoSQL мултимодел база, што значи да подаци нису организовани на један начин, односно према једном моделу, већ подржава различите моделе организације података, међу којима је и граф оријентисани приступ. Не ослања се на шему, што значи да се подаци могу учитати у базу онакви какви су у оригиналном формату, без потребе за претходним дефинисањем шеме података. База пружа флексибилну интеграцију и чување података, као и њихов каснији преглед у виду документа, графа или релационе организације, што је предност мултимодел приступа.

2.5 Закључивање

Закључивање (*Inference*) се врши на основу доказа и образложења. Самим тим, закључивање представља технику извођења нових знања на основу постојећих. Са стране *MarkLogic* семантике, закључивање представља аутоматско откривање нових чињеница на основу података и правила понашања тих података. Када се та теорија пренесе на RDF тројке, добије се техника генерисања нових тројки на основу већ постојећих тројки [5]. Постоје два начина на који се нове тројке могу генерисати преко технике закључивања. Прва се назива *forward-chaining* закључивање и представља технику генерисања нових тројки тако да се оне додају у базу података. Друга техника се назива *backward-chaining* закључивање и она креира нове тројке за време извршавања самог упита. *MarkLogic* се ослања на аутоматско *backward-chaining* закључивање [5]. Аутоматско закључивање се примењује помоћу скупа правила и онтологија које корисник дефинише.

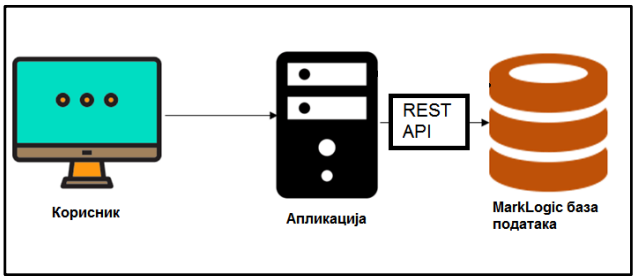
3. ДИЗАЈН РЕШЕЊА

Овим делом рада описана је архитектура решења, као и функционалности апликације.

3.1 Архитектура решења

Први корак представља подешавање радног окружења, односно инсталацију и покретање *MarkLogic* сервера. Затим је потребно направити базу података у оквиру *MarkLogic* сервера. Након тога је потребно направити корисничку апликацију која ће представљати везу између корисника и базе података. Корисник треба да буде у могућности да преко апликације комуницира са базом, врши операције читања и писања и креирања и извршавања SPARQL

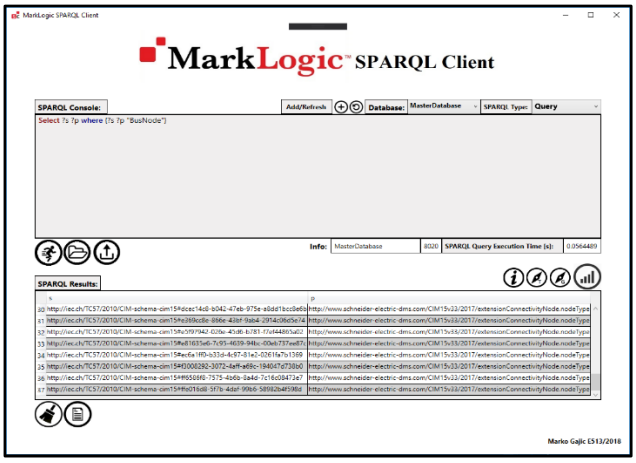
упита. Да би веза између базе и апликације била могућа, искоришћен је REST API интерфејс. Он омогућава комуникацију са *MarkLogic* базом података независно од програмског језика. Слика 1 приказује изглед архитектуре решења.



Слика 1. *Архитектура решења*

3.2 Функционалности апликације

Клијентска апликација је дизајнирана тако да омогући клијенту што лакшу интеракцију са базом података која се налази на *MarkLogic* серверу. Даље су набројане неке од најважнијих функционалности које поседује клијентска апликација. Корисник је у могућности да дефинише SPARQL *Query* и SPARQL *Update* исказе, ручно или учитавањем из фајла. Такође, корисник може у саму базу података учитати податке у једном од облика серијализације RDF формата. Могуће је направити нову базу података као и REST API инстанцу, као и изабрати жељену базу са којом корисник жели да ради. Такође, постоји посебан део који кориснику приказује заузеће процесорске јединице и RAM меморије од стране *MarkLogic* сервера за време рада апликације. Кориснику се резултати интеракције са базом података приказују у формату табеле. Корисника интересује и време извршавања SPARQL исказа, тако да се и та информација приказује у оквиру клијентске апликације. Слика 2 приказује изглед апликације у раду са *MarkLogic* сервером.



Слика 2. *Почетна страна апликације*

4. МЕРЕЊЕ ПЕРФОРМАНСИ РЕШЕЊА

Виртуелној машини на којој се налази *MarkLogic* сервер обезбеђен је *Intel(R) Core(TM) i3-3220 CPU @ 3.30 GHz* процесор и 13.7 *GB* RAM меморије. Као улазни подаци за тестирање су коришћена пет CIMXML фајлова који представљају пет фидера из једне северноамеричке дистрибуције.

Тестирањем се приказују перформансе апликације у зависности да ли је код базе података укључена могућност закључивања и како та могућност утиче на пораст броја тројки, време извршавања самог упита заузеће процесора.

Такође, тестирана је и конективност учитаних елемената. Као улазни подаци код тестирања конективности коришћен је модел мреже у CIMXML формату обезбеђен од ENTSO-E (*European Network of Transmission System Operators for Electricity*).

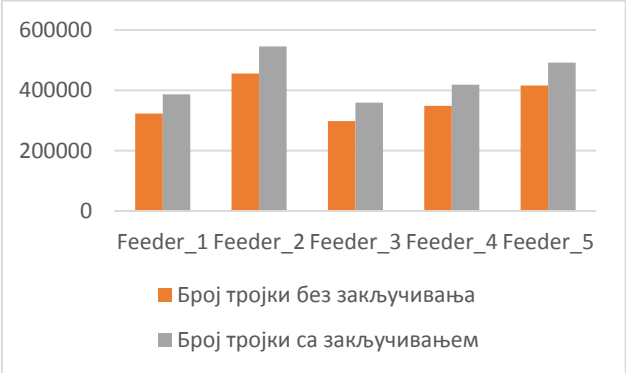
4.1 Перформансе са и без закључивања

Особине које су тестиране су: број тројки, време извршавања SPARQL упита који добавља све тројке и заузеће процесора током добављања тројки. Циљ је био упоредити резултате који су добијени без укљученог закључивања са резултатима који су добијени са укљученим закључивањем.

Закључивање се остварује уз помоћ онтолошког фајла насталог на основу *CIM16v33legacy-rdfs* и *rdfs.rules* скупа правила који се учитава у базу података. Табела 1 приказује број тројки док Слика 3 приказује број тројки добијених са и без укљученог закључивања.

Табела 1. *Број тројки са и без закључивања*

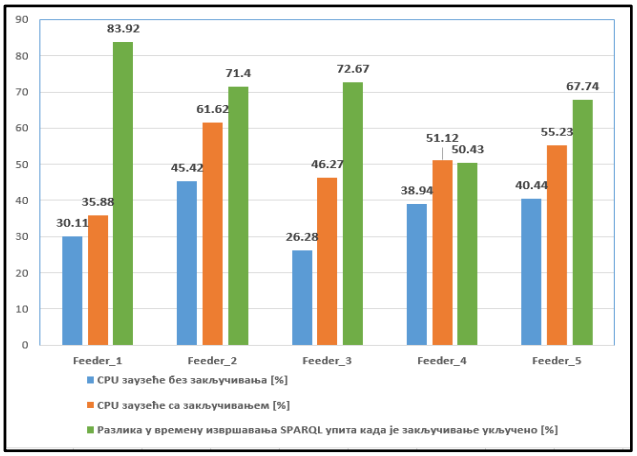
Фајл	Број тројки без закључивања	Број тројки са закључивањем
Feeder_1	323473	387093
Feeder_2	455931	545704
Feeder_3	298256	359292
Feeder_4	348550	418563
Feeder_5	416041	491921



Слика 3. *Поређење и тренд броја тројки са и без могућности закључивања*

Приметан је пораст у броју тројки, што је и главна особина закључивања. Слика 4 приказује заузеће процесора и процентуалну разлику у времену извршавања SPARQL упита са и без укљученог закључивања. Време које је потребно да би се извршио SPARQL упит када је закључивање укључено процентуално је дуже од времена извршавања SPARQL упита када је закључивање искључено.

Такође, вредност заузећа процесора је већа када је закључивање укључено. Такви резултати су очекивани јер упит враћа више тројки када је закључивање укључено.



Слика 4. *CPU заузеће и SPARQL време извршавања*

4.2 Тестирање конективности

Конективност је прво тестирана у случају када је сва опрема повезана. Затим су укањањем одређених *Breaker* елемената и њихових терминала добијена четири одвојена острва повезане опреме смештена у 4 различита именована графа. Након тога је над сваким од њих тестирана конективност. Циљ је био добити времена извршавања SPARQL упита конективности за случај када је опрема подељена на више острва и упоредити та времена са временом добијеним у случају када је сва опрема повезана и упоредити резултате за случај када је закључивање укључено и када закључивање није укључено. Табела 2 приказује број проводне опреме и број тројки док табела 3 приказује времена извршавања SPARQL упита.

Табела 2. *Број проводне опреме и тројки*

Повезаност	Број проводне опреме	Број тројки без закључивања	Број тројки са закључивањем
Све повезано	1547	40013	64898
Острво 1	959	26851	45848
Острво 2	125	1951	3056
Острво 3	58	907	1457
Острво 4	392	6119	9482

Табела 3. *SPARQL време извршавања*

Повезаност	Разлика у времену извршавања када је закључивање укључено [%]
Све повезано	204
Острво 1	466
Острво 2	433
Острво 3	1700
Острво 4	1100

Очекивано, број тројки који се добије у случају укључене могућности закључивања је већи у односу на број тројки који се добије када закључивање није укључено. Такође, времена извршавања SPARQL упита за конективност су процентуално већа у случају укљученог закључивања. Дељењем повезане опреме на више острва и претрагом појединачних острва добијају се знатно мања времена извршавања. Ово је практично у случају великих мрежа када корисник зна у ком делу мреже се неки елемент налази.

5. ЗАКЉУЧАК

Радом је покривена анализа *MarkLogic* базе података као граф оријентисане базе за рад са моделима електроенергетских система. Предност *MarkLogic* базе података представља могућност извршавања CRUD операција над подацима, као и над графовима који садрже те податке преко SPARQL исказа. Предност представља и могућност закључивања, односно генерисања нових знања на основу постојећих. *MarkLogic* кориснику нуди опцију да сам дефинише правила на основу којих ће база података да извршава процес закључивања. Тестирањем су добијени резултати перформанси које база података остварује у случајевима када је закључивање укључено и када закључивање није укључено.

Као закључак се намеће то да ако корисник жели могућност закључивања, јавиће се пораст времена извршавања, као и веће заузеће процесора.

Клијентском апликацијом кориснику је омогућено писање и учитавање SPARQL упита из фајла, учитавање података у одабрану базу, креирање базе као и приказ заузећа рачунарских ресурса у току рада апликације од стране *MarkLogic* сервера. Предлог за даље усавршавање представља графичку визуелизацију података над којим корисник ради у циљу повећања свести корисника о моделу електроенергетског система над којим се извршавају неопходне операције.

6. ЛИТЕРАТУРА

[1] L.King, The Common Information Model for Distribution:An Introduction to the CIM for Integrating Distribution Applications and Systems, EPRI, 2008.
[2] Ontotext, „What is RDF?,“ https://ontotext.com/knowledgehub/fundamentals/what-is-rdf/ . [Последњи приступ 28 Октобар 2019].
[3] P. Leslie F. Sikos, Mastering Structured Data on The Semantic Web.
[4] WhatIs.com, „Graph database“ https://whatis.techtarget.com/definition/graph-database/ . [Последњи приступ 28 Oktobar 2019].
[5] MarkLogic Corporation, MarkLogic Server:Semantic Graph Developer’s Guide, 2019.

7. ПОДАЦИ О КАНДИДАТУ

Марко Гајић рођен је 31. марта 1995. год. у Шапцу. Завршио је основну школу “Јеврем Обреновић” у Шапцу 2010. год. Завршио је средњу школу “*Техничка школа Шабац*” у Шапцу, 2014. године. Исте године је уписао основне академске студије на Факултету техничких наука, смер примењено софтверско инжењерство. У септембру 2018. год. дипломирао је и уписао мастер студије. Испунио је све обавезе и положио све испите у оквиру мастер студија у року са просечном оценом 8.67.