Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Систем за авторизација кај SPARQL протоколот со користење на дистрибуирани кориснички атрибути

Ристе Стојанов

Ментор:
Проф. д-р Димитар Трајанов

Скопје, Април 2018

Комисија

Вон. Проф. д-р Игор Мишковски, претседател
Факултет за информатички науки и компјутерско инженерство
Универзитет Св. "Кирил и Методиј", Скопје, Македонија

Проф. д-р Димитар Трајанов, ментор
Факултет за информатички науки и компјутерско инженерство
Универзитет Св. "Кирил и Методиј", Скопје, Македонија

Акад. Проф. д-р Љупчо Коцарев, член
Факултет за информатички науки и компјутерско инженерство
Универзитет Св. "Кирил и Методиј", Скопје, Македонија

Вон. Проф. д-р Весна Димитрова, член
Факултет за информатички науки и компјутерско инженерство
Универзитет Св. "Кирил и Методиј", Скопје, Македонија

Проф. д-р Слободан Бојанич, надворешен член
Universidad Politécnica de Madrid, Spain

Систем за авторизација кај SPARQL протоколот со користење на дистрибуирани кориснички атрибути

Докторска дисертација

Ристе Стојанов

Апстракт

Проширувањето на паметните уреди, зголемената популарност на социјалните мрежи и ширењето на корпоративните услуги наметнуваат огромни количини на хетерогени податоци кои треба да се генерираат и складираат во посебни силоси на дневна основа. Делови од овие податоци се приватни и високо чувствителни, бидејќи го одразуваат однесување на сопственикот, неговите обврски, навики и преференции. Од друга страна, новите услуги на заедницата предизвикуваат сопствениците да ги изложуваат овие податоци за возврат на погодностите што ги нудат. Затоа, неопходно е не само да се заштити интеракцијата со чувствителни податоци, туку и селективно да се отвори пристапот без загрозување на личниот интегритет на сопственикот.

Еден од главните поддржувачи на услугите на заедницата се поврзаните поадтоци во подем, чија главна цел е отворање на хетерогените знаења од одделни податочни силоси. Нејзината растечка популарност ги поттикнува сопствениците на податоци да ги објавуваат своите лични податоци во поврзан формат на податоци. Со фузијата на сензорските, социјалните и корпоративните податоци се отвораат нови безбедносни предизвици со кои се придвижуваат стандардните безбедносни системи кон пофлексибилни платформи за контекстна авторизација.

Во овој труд, предлагаме платформа за авторизација на поврзани податоци (LDA), над јазикот за полиси кој е доволно флексибилен за да ги опфати сите новонастанати барања, вклучувајќи ја и свесноста за контекстот. Предложениот јазик за полисите го проширува SPARQL прашалниот јазик за пребарување на семантички податоци и ја користи неговата експресивност за да го заштити секој дел од податоците. Новина во LDA платформата е нејзината уникатна способност за валидација на полисите во време на дизајнирање преку самостојно тестирање, откривање на конфликти и целосна екстракција на незаштитените податоци.

Клучни зборови: Авторизација, Семантички Веб, Поврзани податоци, Сигурност, Јазик за полиси, Модификација на прашања, Привремено податочно множество, Управување со полиси

Ментор: Проф. д-р Димитар Трајанов

Authorization system for the SPARQL protocol using distributed user attributes

PhD Thesis

Riste Stojanov

Abstract

The expansion of the smart devices, the growing popularity of the social networks and the wide spread of the corporate services impose huge amounts of heterogeneous data to be generated and stored in separate silos on a daily basis. Parts of this data are private and highly sensitive as they reflect owner's behavior, obligations, habits, and preferences. On the other hand, the emerging crowd services challenge the owners to expose this data in return to the convenience they offer. Therefore, it is imperative not only to protect the interaction with sensitive data, but also to selectively open it in an unharmful manner for the owner's personal integrity.

One of the main enablers of the crowd services is the emerging Linked Data, which is all about opening heterogeneous knowledge from separate data silos. Its growing popularity encourages the data owners to publish their personal data in linked data format. The fusion of sensor, social and corporate data opens new security challenges which extend the standard security considerations towards more flexible and context aware authorization platforms.

In this paper, we propose a Linked Data Authorization (LDA) platform atop a policy language flexible enough to cover all newly emerged requirements, including context awareness. The proposed policy language extends the widely accepted W3C's SPARQL query language and leverages its expressiveness to protect every part of the data. The novelty of our LDA platform is its unique capability of design time policy validation through stand alone testing, conflict detection and overall protection coverage extraction.

Keywords: Authorization, Semantic Web, Linked Data, Security, Policy Language, Query Rewriting, Temporal dataset, Policy management

Supervisor: Prof. Dimitar Trajanov, PhD

Благодарност

Би сакал да ја изразам својата срдечна благодарност до менторот, проф д-р Димитар Трајанов за огромната поддршка со идеи и знаење при изработката на оваа докторска дисертација.

Благодарност до вонр. проф. д-р Игор Мишковски, доц. д-р Милош Јовановиќ и доц. д-р Владимир Здравески за инспиративните разговори на различни теми и заедничката работа и дружење низ сите години на учење и студирање, а особено на доц. д-р Сашо Граматиков, кој навистина ми помогна да ја пренесам идејата на тезата на поразбирлив начин.

Благодарност и до колегите од Факултетот за информатички науки и компјутерско инженерство за прекрасните моменти за време на заедничката работа и научните дискусии.

Би сакал да ја изразам мојата особена благодарност на моите родители Николинка и Гичо Стојанови, за огромната поддршка низ целиот живот, за нивната пожртвуваност, несебична помош и секогаш добронамерни насоки. Навистина знаеја да ме инспирираат и извлечат максумум од мене.

Особено ми значеше логистичката подршка од брат ми Иван Стојанов и останатите членови на мојата потесна фамилја: Весна Симовска, Александар, Златка и Ѓорѓи Трајкови.

И на крајот, огромна благодарност до сопругата Слободанка Стојанова, која беше покрај мене и ме бодреше и инспирираше во целиот тек на изработката на мојата докторска дисертација, а особено во моментите кога ми беше најтешко. Благодарност за сите жртви кои ги направи за да ме подржи, а особено затоа што ме убеди да не се откажам кога ми беше најтешко.

Докторската теза им ја посветвам на моите синови, Петар и Стефан, кои ми се најголема инспирација во животот, без кои, ништо од ова немаше да има смисла.

Трудови објавени во списанија со имапкт фактор:

[1] **Stojanov, R.**, Gramatikov, S., Mishkovski, I. and Trajanov, D.. Linked data authorization platform. IEEE Access, Issue 99. 2017. DOI: 10.1109/ACCESS.2017.2778029 **Imparct factor: 3.224**

Трудови објавени во меѓународни списанија и конференции:

[2] Trajanov D., Zdraveski V., **Stojanov R**. and Kocarev L.. Dark Data in Internet of Things (IoT): Challenges and Opportunities, Proceedings of the 7th Small Systems Simulation Symposium, Niš, Serbia, 12th-14th February 2

[3] **Stojanov R.** and Jovanovik M.. Authorization proxy for sparql endpoints. In ICT Innovations 2017, pages 205–218, Cham, 2017. Springer International Publishing. CCIS, volume 778

[4] Gjorgjevikj A., **Stojanov R.**, and Trajanov D.. Enhancing text-based relatedness measures with semantic web data. In ICT Innovations 2016, pages 182–192, Cham, 2018. Springer International Publishing. AISC, volume 665

[5] **Stojanov R.**, Georgiev M., Zdraveski V., Jovanovik M., and Trajanov D.. Live objects-collaborative window in the corporate documents. In New Trends in Database and Information Systems II , pages 71–81. Cham, 2015. Springer International Publishing. AISC, volume 312

[6] Trajkova S., **Stojanov R.**, and Trajanov D.. Semantic Web Access Control Aspects. In The 12th International Conference on Informatics and Information Technologies , pages 243-245, 2015.

[7] Andreevski A., **Stojanov R.**, Jovanovik M., and Trajanov D.. Semantic web integration with sparql autocomplete. In The 12th International Conference on Informatics and Information Technologies , pages 1–4, 2015.

[8] Nikolovski V., **Stojanov R.**, Mishkovski I., Chorbev I., and Madjarov G.. Educational data mining: Case study for predicting student dropout in higher education.

[9] Najdenov B., Petkovski G., Jovanovik M., **Stojanov R.**, and Trajanov D.. Automated linked data generation from the transport administration domain. In Telecommunications Forum Telfor (TELFOR), 2015, pages 827–830. IEEE, 2015.

[10] Gjorgjevik A., **Stojanov R.**, and Trajanov D.. Semccm: Course and competence management in learning management systems using semantic web

technologies. In Proceedings of the 10th International Conference on Semantic Systems , SEM '14, ISBN: 978-1-4503-2927-9, pages 140–147, New York, NY, USA, 2014. ACM. DOI: 10.1145/2660517.2660535

[11] Popovski V., Kostadinov B., **Stojanov R.**, Mishkovski I., and Trajanov D.. Web-based disaster and crisis management system. 2013.

[12] Mitrevski M., Jovanovik M., **Stojanov R.**, and Trajanov D.. Open university data. In The 9th International Conference on Informatics and Information Technologies . Macedonia, 2012.

[13] Trajanov D., **Stojanov R.**, Jovanovik M., Zdraveski V., Ristoski P., Georgiev M., and Filiposka S.. Semantic sky: a platform for cloud service integration based on semantic web technologies. In Proceedings of the 8th International Conference on Semantic Systems, ISBN: 978-1-4503-1112-0, pages 109–116. ACM, 2012. DOI: 10.1145/2362499.2362515

Содржина

Листа на слики

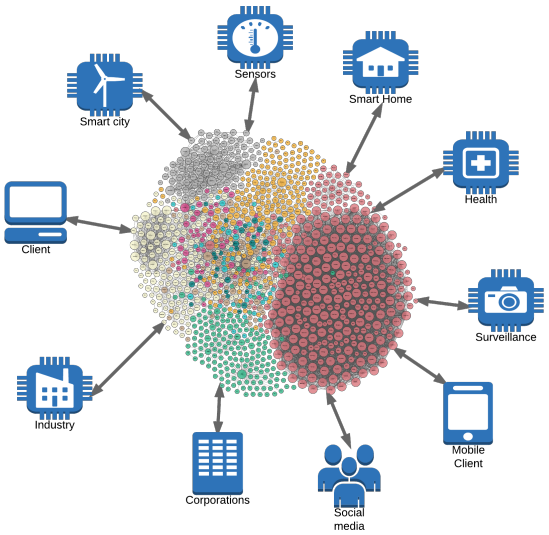
Листа на табели

Глава 1

Вовед

Во паметниот свет околу нас, милијарди уреди, корисници и апликации континуирано генерираат огромни количини на хетерогени податоци. Овие уреди се движат од стандардни компјутери до мобилни телефони и сензори. Благодарение на интернетот, сите тие се поврзани заедно, што им овозможува да ги разменат и поврзат податоците со ресурсите од социјалните мрежи, па дури и да ги интегрираат со корпоративните бази на податоци. Ваквото поврзување на податоците им овозможува на апликациите и на луѓето корисници подобро да ја разберат нивната околина. Комбинацијата на повеќе извори на податоци дополнително може да им помогне контекстно базираните апликации за да донесуваат интелегентни одлуки, како одговор на промените во нивната околина. Овие поврзани податоци континуирано се собираат и складираат на различни корпорациски сервери, Cloud инфраструктури, па дури и на приватни сервери.

Хетерогеноста на уредите ја прави нивната интероперабилност предизвикувачки проблем поради различната природа на генерираните податоци (температура, светлина, видео, текст, локација), неконзистентен квалитет и веродостојноста на изворите. Во изминатите децении, W3C го препозна овој проблем и воведе нови стандарди познати како Web 3.0, или Семантички веб [10]. Главната цел на овие W3C стандарди е да ја придвижат World Wide Web мрежата од страници кон мрежата на податоци, трансформирајќи ги страниците во поврзани ресурси со податоци што може да се обработат од софтверски агенти. Пристапите на иницијативата за поврзани податоци (Linked Data) [44, 14] обезбедуваат нови начини за интегрирање и консолидирање на податоците од различни и дистрибуирани извори, како и решавање на проблемот за поврзување на изолираните податочни силоси, како што се традиционалните системи со релациони бази на податоци. Овој пристап овозможува објавување и контекстуално поврзување на податоци на Интернетот со податоци (Web of Data) [40, 39]. Илустрација на некои од ентитетите кои објавуваат податоци во Linked Open Data (LOD) облакот



Слика 1-1: Облак со поврзани отворени податоци (Linked Open Data Cloud) и објавувачи на податоци

[24] е прикажана на сликата 1-1. LOD облакот [1] визуелизира дел од податочните множества [2] објавени користејќи ги принципите на поврзаните податоци [14].

Податоците што ги генерираме, директно или преку уредите што ги поседуваме, во комбинација со општо прифатени бази на знаење (како што се DBpedia [6]) и корпоративните бази на податоци може да ги опишат нашите навики, животната средина, а во случај на уредите што ги носиме на себе, дури и нашето здравје. Бидејќи овие податоци се од чувствителна природа, треба да останат приватни и заштитени. Доколку овие податоци не се заштитат соодветно, тие може да предизвикаат сериозно нарушување на приватноста. Анализата на злоупотреба на чувствителните податоци за локација е прикажана во [5], каде што авторите дискутираат дека рекламната индустрија најмногу ги искористува личните информации кои се изложени поради неправилно ракување со нив. Сепак, кога корисничките податоци се постапуваат правилно, може да се користат за да се обезбедат услуги наменети за групи на индивидуи (Crowd source апликации) што можат значително да ги подобрат нашите животи[70, 62]. Таков пример е услугата Google Traffic [3], која ги користи локациите на корисниците за да ги детектира пренатрупаните рути и да предложи алтернативни побрзи патишта. Оваа услуга заштедува драгоцено време на нејзините корисниците.

Потребата за персонализирана, корисничка дефинирана заштита на огромни количини на хетерогени податоци, досега не била разгледана во таков размер.

[1] Дијаграм на Linking Open Data облакот од 2017, изработен од Andrejs Abele, John P. McCrae, Paul Buitelaar, Anja Jentzsch и Richard Cyganiak. http://lod-cloud.net/.

[2] http://stats.lod2.eu/

[3] http://www.dailymail.co.uk/sciencetech/article-4706666/Google-introduces-traffic-time-maps.html, пристапена на 20 јули 2017 година

Безбедносните протоколи, процедури и алатки се секогаш чекор назад во справувањето со новите безбедносни предизвици. Кога станува збор за чувствителни податоци, без разлика дали тие се лични, социјални или корпоративни, мора да се применат строги правила за да се осигури дека тие се соодветно пристапувани и ракувани. Можноста на сопствениците на податоците да контролираат кој и под кои услови ќе има пристап до нивните податоци може да ги поттикне да ги споделат податоците кои придонесуваат за јавното добро, а истовремено и да ја заштитат нивната приватност[70]. Алатките што овозможуваат тестирање на безбедносните полиси и преглед на заштитените податоци се важен чекор кон стекнување доверба во платформите за авторизација.

Системите за чување и управување со податоците најчесто ги прават достапни податоците преку сервисни акции, кои може да се повикаат од барателите, кои може да се луѓе, софтверски агенти или кој било други субјекти кои може да ја испратат својата намера до системот. Барателот се идентификува според својствата што ги презентира на системот, како што е неговиот токен за автентикација или други параметри кои ја опишуваат неговата околина, обично обезбедени од страна на софтверскиот агент. Овие параметри, заедно со состојбата на системот, го дефинираат контекстот во кој се извршува акцијата. При извршувањето на акцијата, нејзините операции, параметри и контекст ја опишуваат целта на барателот, односно неговата *Намера*.

Безбедносните правила во системите обично се искажуваат преку барања изразени во природен јазик. Иако барањата исразени преку слободен текст имаат голема експресивност, нивната разновидност ги прави речиси невозможни за имплементирање. Затоа, барањата треба да бидат формализирани и преведени на машински читливи безбедносни полиси кои се полесни за имплементација. Формализмот на полисите ги дефинира границите на процесот за трансформацијата на барање искажано преку слободен текст во безбедносните полиси. Безбедносните полиси дефинираат кои делови од податоците можат да бидат заштитени во дадено контекстуално опкружување. Понатаму, јазикот за полисите обезбедува синтакса која го поддржува овој формализам и го поедноставува процесот на нивно управување. Во традиционалните корпорации, процесот на преведување на барањата во полиси најчесто се изведува од соодветно обучени службеници за безбедност и администратори. Сепак, обемот на податоците што треба дате се прават достапни брзина на нивно создавање и нивната хетерогеност го прави централизирано управување со полиси неизводливо. Затоа, формализмот на полисите треба да поддржува изработка на алатки кои ќе им овозможат на обичните луѓе да ги управуваат сопствените хетерогени и дистрибуирани податоци.

Валидацијата на соодветноста на полисите во однос на барањата за безбедност за време на нивното дизајнирање е од клучно значење за правилна заштита на податоците. Соодветноста може да биде потврдена ако јазикот за полисите овозможува екстракција на податоците заштитени со одредена полиса. Дополнително, секој добро дизајниран јазик за полиси треба да овозможи акцијата

откривање и решавање на можните конфликти помеѓу полисите.

Бидејќи иницијативите за семантички веб на W3C имаат дефинирано стандарди кои овозможуваат поврзување на ресурсите складирани во различни бази на податоци [53, 10] и иницијацијата Linked Open Data има дефинирано механизми за совпаѓање на истите ресурси со различни репрезентации [14], еден од проблемите што остануваат отворени е дизајнот на јазик за полиси што може да ги заштити податоците складирани или изложени како Linked Data [25, 13]. Ваковиот јазик за полиси треба да обезбеди разни нивоа на грануларност за заштита на дистрибуираните множества на податоци и истовремено да обезбеди флексибилност за да моделирање на барањата искажани преку природен јазик. Покрај флексибилноста, јазикот за полисите треба да биде лесен за разбирање и учење и треба да овозможи креирање на алатки кои ќе го поедностават управувањето со полиси и нивната валидација.

Главната мотивација на оваа теза е дизајнирање на платформа која ќе ги штити сите интеракции со произволни делови од поврзаните податоци и ќе овозможи валидацијата на точноста на полисите за време на нивното дизајнирање. Предложената платформа за авторизација на поврзани податоци (Linked Data Authorization - LDA) ја исполнува оваа цел и е способна да ги заштити операциите за пристап, модификација и управување користејќи го својот флексибилен јазик за полиси, кој ги поврзува заштитените податоци со контекстот на авторизацијата. Јазикот за полисите се заснова на на добро воспоставената стандардизирана синтакса на SPARQL [4], што овозможува реискористување на постојните знаења и алатки за поедноставување на процесот на управување со полисите. Дополнително, имплементиравме и модул треба за управување со полиси кој овозможува валидација на точноста за времето на дизајнот на полисите, приказ податоците заштитени со полисите, откривање на конфликти, како и екстракција на заштитените и незаштитените делови од податочните множества.

[4]SPARQLa е јазик за прашања на Семантичкиот веб: https://www.w3.правилна/TR/rdf-sparql-query.

Глава 2

Поврзана работа

2.1 Структура на семантички поврзани податоци

Иницијативата поврзана со податоци (Linked Data) [14] го решава проблемот на поврзување и споделување на податоци помеѓу различни услуги [3] и ги проширува семантичките веб стандарди [10], кои пак овозможуваат податочна репрезентација независна од технологијата на чување на податоците преку стандардите Resource Description Framework - RDF[1], RDF Schema - RDFS[2] и Ontology Web Language - OWL[3]. Иницијативата за поврзани податоци има дефинирано повеќе методологии кои овозможуваат усогласување на различни концепти со исто значење од различни податочни извори. Брзиот раст на облакот со отворени поврзани податоци (Linked Open Data Cloud) [44, 50] ја докажува применливоста на овие стандарди и методологии за меѓусебно поврзување и повторна употреба на податоците меѓу различни извори. Секојдневно се појавуваат различни пристапи за објавување на корпоративни поврзани податоци [92, 3, 89, 84], но главниот проблем за нивното широко прифаќање надвор од корпоративните граници е недостатокот на механизми за авторизација [52].

Поврзаните податоци може да се претстават како повеќе меѓусебно поврзани ресурси од различни извори. Ресурсите може уникатно да се претастават преку интернационален идентификатор на ресурси (International Resource Identifier - IRI)[4]. Ресурсите се опишуваат со својства, кои исто така се претставуваат со IRI. Својствата се користат за опис на ресурсите, но може да се користат и за нивно поврзување, при што се формира насочена граф структура, во која ресурсите се јазли, а својствата се рабови. Секој пар јазли, поврзан со својството кое ги

[1] https://www.w3.org/TR/rdf-concepts/
[2] https://www.w3.org/TR/rdf-schema/
[3] https://www.w3.org/TR/owl-features/
[4] https://tools.ietf.org/html/rfc3987

поврзува формира таканаречена *RDF тројка* (Дефиниција 1).

Дефиниција 1 ***RDF тројка*** *е торка* $\langle S, P, O \rangle \in (I \cup B \cup L) \times I \times (I \cup B \cup L)$, *каде* S *е субјектот кој се опишува со предикатните својства* P *преку соодветно доделената вредност* O. *Множествата* I, B *и* L, *ги претставуваат IRI-ата, анонимните јазли и литералите, соодветно.*

Изворниот код 2.1 покажува пример на RDF тројки во N3 форматот[5]. Примерот ги користи префиксите *rdf:* и *sm:* за шемата која ја дефинира синтаксата на RDF и онтологијата за мерење на сензорот од Слика 2-1, додека *ex:* е префиксот за тековното податочно множество. Овие префикси ќе бидат користени во остатокот на дисертацијата и ќе бидат намерно изоставени заради концизност. Ресурсите од исто податочно множество обично делат ист префикс. Поврзувањето на податоци од различни множества може да се постигне со едноставно поврзување на соодветените ресурси преку нивните IRI. Линија 6 прикажува RDF тројка која опишува дека *ex:john* е *sm:User*. Во оваа тројка, *ex:john* е субјектот S, *rdf:type* е предикатното својство P, а *sm:User* е објектот. N3 синтаксата се користи во примерот, при што ги одделува тројките со помош на точка (.), додека пак може да додели повеќе парови со својство-вредност на истиот субјект користејќи точка-запирка (;) како сепаратор. Линиите 7-8 ги опишуваат вредностите на својствата *sm:works_at* и *sm:phone* на субјектот *ex:john*. Сите елементи од тројките (линии 6-8) се претставени со IRI, со исклучок на телефонскиот број, кој што е литерал. Литералите (Literals) L ги покриваат примитивните типови на податоци, како што се низи, boolean, цели броеви (integer, long), децимални броеви (float, double) итн. Иако во примерот не се прикажани анонимни јазли (blank nodes) B, тие обично се користат како ресурси за опис на транзитивна врска на други ресурси, без експлицитно наведување на IRI-то на поврзувачкиот елемент.

```
1 @PREFIX ex: <http://example.com/>
2 @PREFIX rdf: <http://www.w3.org/1999/02/
3                 22-rdf-syntax-ns#>
4 @PREFIX sm: <https://ontology.example.com#>
5
6 ex:john rdf:type sm:User .
7 ex:john sm:works_at ex:hospital ;
8   sm:phone '070 111 111'.
```

Изворен код 2.1: Пример RDF тројки во N3 формат

RDF тројките дополнително може логички да се организираат во RDF графови (Дефиниција 2), кои може да бидат именувани со користење на IRI идентификаторите. Со ова се формира поопшта репрезентација на знаењето со торка од

[5] https://www.w3.org/TeamSubmission/n3/

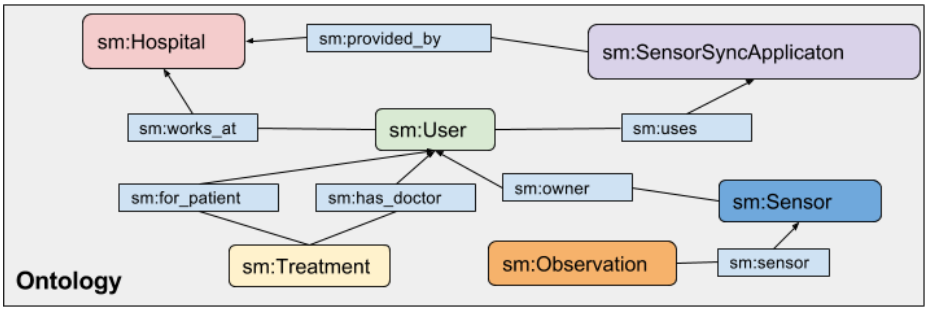
четири елементи, наречена RDF четворка (Дефиниција 3). Податочните множества (Дефиниција 4) нудат физичко зачувување на поврзаните податоците, при што групираат повеќе именувани графови и еден предефиниран граф. Податочните множества вообичаено се управувани од иста индивудуа или организација.

Дефиниција 2 ***RDF граф*** *е множество од RDF тројки. RDF графовите може да се претстават преку IRI.*

Дефиниција 3 ***RDF четворка*** *е торка* $\langle S, P, O, G\rangle \in (I \cup B \cup L) \times I \times (I \cup B \cup L) \times (I \cup B)$*, каде* S, P *и* O *се елементи од RDF тројка кои припаѓаат во RDF графот* G*. Множествата* I*,* B *и* L *ги претставуваат IRI идентификаторите, анонимните јазли и литералите, соодветно.*

Дефиниција 4 ***RDF податочно множество*** *DS се состои од нула или повеќе именувани RDF графови и точно еден предефиниран граф. RDF податочното множество може да се претстави преку парот* $\langle name, RDF\ Graph\rangle$*, каде* $name \in (I \cup B)$*.*

Изворниот код 2.2 прикажува пример податочно множество кое е моделирао со онтологијата од Слика 2-1. Заобленитe правоаголници од Слика 2-1 ги означуваат класите од ресурси, додека пак именуваните врсики ги означуваат својствата. Податоците кои се моделирани со оваа онтологија се препрезентираат преку соодветни инстанци од прикажаните класи. Припадноста на инстанците во одредени класи се претставува преку *rdf:type* својството, или често означено само со буквата *a*. Онтологиите покрај тоа што ја опишуваат структурата на податоците, тие може да се искористат за да се заклучат дополнителни факти или за да ги ограничат можните поврзувања помеѓу семантичките ресурси[6].



Слика 2-1: Онтологија која поврзува сензорски податоци со систем за управување со пациенти

[6]Класите на семантичките податоци се опишани преку *rdfs:Class* ресурсот, каде *rdfs:* е префиксот на RDF Schema: http://www.w3.org/2000/01/rdf-schema#

```
ex:hospital a sm:Company;
 sm:network_address '192.168.100.0/24';
 sm:location _:l1.

ex:ssa a sm:SensorSyncApplicaton;
  sm:provided_by ex:hospital.

ex:john a sm:User; sm:works_at ex:hospital;
 sm:phone '070 111 111'.
ex:ben a sm:User; sm:works_at ex:hospital;
 sm:phone '075 555 555'.
ex:bob a sm:User; sm:uses ex:ssa;
 sm:emergency_phone '075 123 456'.
ex:alice a sm:User; sm:uses ex:ssa;
 sm:emergency_phone '075 987 654'.

ex:t1 a sm:Treatment; ex:for_patient ex:bob;
 ex:has_doctor ex:john; ex:from '2017-07-20';
 ex:to '2017-09-20'.
ex:t2 a sm:Treatment; ex:for_patient ex:alice;
 ex:has_doctor ex:ben; ex:from '2017-04-13';
ex:to '2017-04-23'.
ex:t1 a sm:Treatment; ex:for_patient ex:john;
 ex:has_doctor ex:ben; ex:from '2017-07-01';
 ex:to '2017-09-20'.

ex:s1 a sm:HealthSensor; sm:stype 'Pulse'; sm:unit 'bpm';
 sm:regular_from 60; sm:regular_to 140; sm:owner ex:bob.
ex:s2 a sm:Sensor; sm:stype 'Temperature'; sm:unit 'C';
 sm:onwer ex:alice; sm:location _:l2.

_:l1  sm:latitude 42.004 ;sm:longitude 21.409 .
_:l2 sm:latitude 42.010 ;sm:longitude 21.410.

ex:ssa {
  ex:o1 a sm:Observation; sm:sensor ex:s1;
   sm:val 66; sm:time 1500386600319.
  ex:o2 a sm:Observation; sm:sensor ex:s1;
   sm:val 57; sm:time 1500386690319.
  ex:o3 a sm:Observation; sm:sensor ex:s2;
   sm:val 28; sm:time 1500386690319.
}
```

Изворен код 2.2: Пример податочно множество

Податочното множество од изворниот код 2.2 е составено од еден предефиниран граф и именуваниот RDF граф *ex:ssa*. Предефинираниот граф ги содржи

тројките од линиите 1-33, опишувајќи ја болницата *ex:hospital*, апликацијата која ги синхронизира опсервациите на сензорите *ex:ssa*, корисниците *ex:john, ex:ben, ex:bob* и *ex:alice* и сензорите *ex:s1* и *ex:s2*. Именуваниот RDF граф *ex:ssa* (линии 35-42) содржи опис за опсервациите што се синхронизирани во податочното множество со соодветната апликација *ex:ssa*. Во примерот, секоја тројка од именуваниот RDF граф *ex:ssa* може да биде претставена како четворка. Една таква четворка е *ex:o1 sm:value 66* ***ex: ssa*** од линијата 36. Овде, првите три елементи ја опишуваат RDF тројката *ex:o1 sm:вредност 66*, додека пак ***ex: ssa*** го опишува именуваниот граф на кој му припаѓа оваа тројка. Елементите со префикс *_:* се анонимни јазли кои не се претставени со соодветен IRI.

2.1.1 SPARQL прашален јазик

Најчесто извршувани операции врз поврзаните податоци се пристап, модификација и управување. Еден од најкомплетните стандарди што ги опфаќаат сите овие операции е SPARQL 1.1 [7] [38]. SPARQL е дефиниран од W3C работната група и неговите прашања користат комбинација на шаблони за тројќи (Дефиниција 5) за да ги опишат условите што треба да ги задоволат бараните променливи. Прашањата се составени од *Форма на прашањето (Query Form - QF)* и *алгебарски израз (Algebra Expression - AE)*. Алгебарскиот израз *AE* е претставен преку елементот *WHERE* (линии 2-10 од изворниот код 2.3) во кој се комбинираат еден или повеќе шаблони за тројки (линии 4-5,7- 9) за дефинирање на потребните услови кои треба да ги исполнат податоци кои ќе бидат вратени. SPARQL прашањата користат променливи што почнуваат со *?*, кои може да ги заменат ресурсите, литералите или анонимните јазли кои се од интерес и треба да се пронајдат во податочното множество врз кое се извршува прашањето. Во SPARQL прашањето од изворниот код 2.3 се користат променливите *?o,?p,?v,?s* и *?t*. Евалуацијата на *AE* бара пресликување од променливите кон ресурсите, литералите и анонимните јазли. Ова пресликување се нарекува *мапирање на променливите (variable binding)*. Ако прашањето од изворниот код 2.3 се изврши над податочното множество од изворниот код 2.2, променливите ќе се мапираат со резултатите прикажани во табелата 2.1. Кога мапираните променливи ќе се заменат во шаблоните за тројки искористени во прашањето, се добива подмножество на податочното множество над кое е извршено прањањето. SPARQL имплементациите се дизајнирани за да ги пронајдат и вратат сите мапирања на променливите како резултат на процесирањето на *AE*.

Дефиниција 5 ***Шаблон за тројки (Triple pattern)*** *се претствува преку торката* $\langle S, P, O\rangle \in (I \cup B \cup L \cup V) \times (I \cup V) \times (I \cup B \cup L \cup V)$*, каде* V *го претставува множеството на променливи, кои заменуваат произволно IRI, анонимен јазел или литерал.*

```
1 SELECT ?o ?p ?v
2 WHERE {
3   GRAPH ex:ssa {
4     ?o a sm:Observation.
5     ?o ?p ?v
6   }
7   ?o sm:sensor ?s.
8   ?s a sm:HealthSensor.
9   ?s sm:stype ?t
10  FILTER (?t = 'Pulse')
11 }
```

Изворен код 2.3: Селекција на сите обсервации синхронизирани од *ex:ssa*

Табела 2.1: Мапирање на променливи од алгебарски израз

?o	?p	?v	?s	?t
ex:o1	rdf:type	sm:Observation	ex:s1	"Pulse"
ex:o1	sm:sensor	ex:s1	ex:s1	"Pulse"
ex:o1	sm:val	66	ex:s1	"Pulse"
ex:o1	sm:time	1500386600319	ex:s1	"Pulse"
ex:o2	rdf:type	sm:Observation	ex:s1	"Pulse"
ex:o2	sm:sensor	ex:s1	ex:s1	"Pulse"
ex:o2	sm:val	57	ex:s1	"Pulse"
ex:o2	sm:time	1500386690319	ex:s1	"Pulse"

Сепак, тројките не можат да го дефинираат графот во кој тие логички припаѓаат. Затоа, за пофлексибилно пребарување, се користат шаблони за четворки (Дефиниција 6). Шаблоните за четворки ги прошитуваат шаблоните за тројки со дополнителен IRI или променлива со што се дефинира графот на кој тројката му припаѓа. Елементот *GRAPH* (линии 3-6) при процесирањето на прашањето може да се трансформира во два шаблони за четворки: *?o a sm:Observation ex:ssa* и *?o ?p ?v ex:ssa*. Комбинацијата на шаблоните за тројки и *FILTER* елементи формира основен шаблон за графови (Дефиниција 7). Во изворниот код 2.3, линиите 4-5 и 7-10 прикажуваат два основни шаблони за графови кои овозможуваат избор на ресурси преку не-тривијална комбинација на нивните врски и својства.

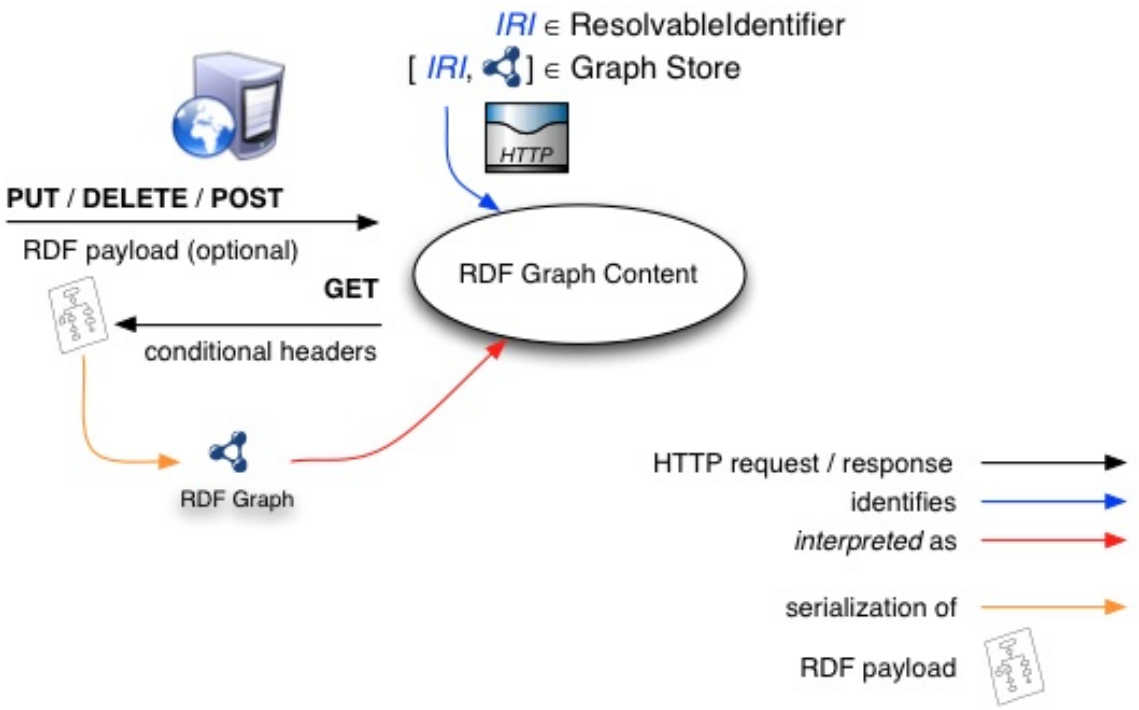
Дефиниција 6 ***Шаблон за четворка (Quad pattern)*** *се претставува преку торката* $\langle S, P, O, G\rangle \in (I \cup B \cup L \cup V) \times (I \cup V) \times (I \cup B \cup L \cup V) \times (I \cup B \cup V)$.

Дефиниција 7 ***Основен шаблон за граф (Basic Graph Pattern)*** *претставува множество на шаблони за тројки со опционални изрази за филтрирање.*

[7] https://www.w3.org/TR/sparql11-query

Формата на прашањето (линија 1 од изворниот код) 2.3) опишува како променливите врски ќе се користат за компонирање на решението. *SELECT ?var...* формата на прашања враќа мапирање на променливи како решение, *ASK* формата враќа дали *AE* може да пронајде мапирање на променливите и *CONSTRUCT {triple pattern }* формата креира множество на тројки кои се конструираат со замена на мапирањето на променливите од *AE* во шаблоните на *CONSTRUCT* изразот. Во нашиот пример, само сивите колони од Табела 2.1 ќе бидат вратени како краен резултат на извршувањето на барањето и тие ја претставуваат материјализацијата на променливите за ова *SELECT* прашање. SPARQL UPDATE[8] е проширување на основната W3C спецификација и овозможува дополнителни форми на прашања за модификација на поврзаните податоци (*INSERT, DELETE*) и управување со графови (*CREATE, DROP, MOVE, COPY*). Иако операцијата за ажурирање (*UPDATE*) не е експлицитно овозможена, може да се постигне со последователно бришење и вметнување.

2.1.2 SPARQL протокол



Слика 2-2: SPARQL протокол

Потребата од оддалечено извршување на SPARQL прашања го иницира специфицирањето на SPARQL протоколот[9] со кој се овозможува извршување на

[8] https://www.w3.org/TR/sparql11-update/
[9] `https://www.w3.org/TR/sparql11-protocol/`

прашања преку HTTP протоколот [32], како и специфицирање на SPARQL протоколот за зачувување на графови преку HTTP (SPARQL graph store HTTP)[10].

SPARQL протоколот овозможува проследување на SPQRQL прашањата преку *query* HTTP параметарот, каде сервисот знае да го извлече овој параметар, да го изврши прашањето и во зависност од прифатените типови на содржина[11] да го серијализира резултатот. Дополнително, клиентот може да специфицира над кои графови да се изврши прашањето преку испраќање на *default-graph-uri* параметарот за IRI за предефинираниот граф и *named-graph-uri* параметарот за IRI за останатите именувани графови.

SPARQL протоколот овозможува и извршување на UPDATE прашања кои го модифицираат податочното множество, каде што се користи *update* HTTP параметарот за да се специфицира прашањето кое треба да се изврши. Притоа, за овој тип на прашања е задолжително користење на HTTP POST методот за комуникација.

SPARQL протоколот за зачувување на графови преку HTTP овозможува интеракција со испраќање на RDF податоците кои треба да се зпишат или избришат. Овој протокол обезбедува интеракција со именуваните графови специфицирани преку *graph* параметарот, или пак so предефинираниот граф ако е испратен *default* параметарот. Дополнително, разлините HTTP методи се интерпретираат во соодветни форми на SPARQL UPDATE прашања:

- *HTTP GET* методот се користи за добивање на тројките кои се зачувани во специфицираните графови и се интерпретира како:

```
CONSTRUCT { ?s ?p ?o } WHERE { GRAPH <graph_uri> { ?s ?p ?o } }
```

- *HTTP PUT* методот се користи за замена на податоците во специфицираниот граф. Овој метод прво ги отстранува сите тројки од специфицираниот граф, по што ги вметнува податоците кои се наоѓаат во телото на барањето. Овој метод се интерпретира како:

```
DROP SILENT GRAPH <graph_uri>;
INSERT DATA { GRAPH <graph_uri> { .. RDF payload .. } }}
```

- *HTTP POST* се користи за вметнување на нови тројки во специфицираниот граф и се интерпретира како:

```
INSERT DATA { GRAPH <graph_uri> { .. RDF payload .. } }
```

- *HTTP DELETE* се користи за бришење на специфицираниот граф и се интерпретира како:

```
DROP GRAPH <graph_uri>
```

[10] https://www.w3.org/TR/sparql11-http-rdf-update/

[11] Прифатливите типови на содржина се наведуваат преку HTTP *Accept* header-от.

2.2 Принципи за дизајн на авторизација на поврзани податоци

Дизајнирањето на комплетна платформа за авторизација бара идентификување на безбедносни аспекти за проценка на квалитетот. Експанзијата на поврзаните податоци мотивира многу истражувачи да работат на безбедносни прашања, некои со ограничување на множеството на достапни акции [45, 88, 49], а други со филтрирање на податоците достапни за операциите [1, 77, 51, 34, 35, 67, 19, 68]. Во овој дел ги разгледуваме аспектите на авторизација кои се потребни за детална заштита, не само на податоците објавени користејќи ги принципите за дизајнирање на поврзаните податоци, туку и за податоците произлезени од различни мобилни уреди и сензори, складирани дури и во традиционални системи за бази на податоци. Врз основа на детално прегледување на литературата ги дефинираме следните четири принципи на дизајн: (1) *флексибилноста* ја дефинира потребата за покривање на широк опсег на барања во системот за авторизација; (2) *одржливоста* се фокусира на оптимизација на процесот за управување со сигурносните полиси; (3) *валидација на точноста* ја нагласува важноста на проверката на точноста на полисите и (4) *разбирливоста* ја моделира важноста на користење стандардизирана синтакса како основа на јазикот за полисите. Понатаму, ги користиме овие принципи за да ја анализираме поврзаната работа во §2.3 и да ги поставите темелите на нашиот јазик за полиси и платформа за авторизација.

2.2.1 Флексибилност

Флексибилноста ја дефинира способноста да се трансформираат произволни барања за авторизација од слободен текст во безбедносни полиси. Флексибилноста произлегува од моќта на јазикот за полиси при моделирањето на различни безбедносни барања. Според [51], полисата може да биде генерално формализирана со користење на торката $\langle R, D, AR\rangle$, која ги опишува правата за пристап AR доделен на барателите R (или субекти) за податоците D (или ресурси). Во врска со оваа формализација на полисите, постојат различни аспекти на флексибилност кои треба да ги задоволи секоја модерна платформа за авторизација на податоци:

Опфатеност на операции

Овој аспект ги дефинира операциите поддржани од платформата за авторизација, која се нарекува Модел на дозвола во [22]. Во оваа теза го употребуваме терминот *атомични операции* (или само операции, за концизност) за неразделни операции кои можат да се извршат, додека поимот акција (*action*) се користи за

да се означи посложена обработка (обично од бизнис домен) која бара извршување на една или повеќе операции. Низ литературата, најчесто се опфатени операциите *читање (Read)*, *вметнување (Insert)* (честопати се нарекува *креирање (Create)* [22] или *запишување (Write)* [45, 75]) и *бришење (Delete)*. Инспирирани од [51], каде што авторите ги користат типовите на SPARQL прашања како атомични операции за заштита, во оваа теза дополнително е вклучена и атомичната опрерација за *управување (Manage)* со именувани графови [12], кои вклучуваат CREATE, DROP, COPY и MOVE типовите на прашања.

Грануларност

Поврзаните податоци формираат структура на насочен граф, каде што јазлите се претставени со ресурси и литерали, додека рабовите се претставени со својства на ресурсите. Ресурсите се користат за опишување на реални или апстрактни концепти, а литералите ги претставуваат примитивните вредности кои може да се доделат како вредност на својствата на одреден ресурс. Секој пар од поврзани јазли, вклучувајќи го и својството што ги поврзува (раб), се нарекува *тројка (triple)*, која логички може да припаѓа на *именуван граф (named graph)* и физички да се зачува во *податочно множество (dataset)* [10, 44].

Аспектот за грануларност ја дефинира способноста на јазик за полиси да ја моделира структурата на податоци и семантиката на различни нивоа. Нивото на грануларност се одредува преку способноста да се заштитат ресурсите, тројките, именуваните графови и множества на податоци. Грануларноста е еден од најчесто дискутираните аспекти во другите платформи за авторизација [52]. Во нашата работа ги користиме следните карактеристики кои овозможуваат различни нивоа на грануларност за заштита:

- *Шаблони за тројки (Triple Patterns - TP)* се во суштина тројки кои можат да имаат променлива на која било позиција, што се користи за селекција на ресурси кои имаат одредено својство или врска со друг ресурс. TP селекцијата резултира со множество од тројки.
- *Basic Graph Patterns (BGP)* се множество од шаблони за тројки кои можат да моделираат посложени семантички и структурни релации меѓу ресурсите.
- *Припадност во именуван граф (Gm)* овозможува дефинирање на тројки кои припаѓаат на даден именуван граф.
- *Припадност во податочно множество (DSm)* овозможува дефинирање на тројки кои припаѓаат на дадено податочно множество.

[12] *Именуваните графови* обезбедуваат логичка организација на поврзаните податоци, додека податочните множества на податоци обезбедуваат физичко складирање.

Контекстна заштита

Зголемената популарност и присуството на сензорски уреди го прават контекстот неизбежен дел од сите модерни системи. Со цел да обезбеди соодветна заштита на приватните и чувствителните податоци што ги произведуваат сензорите, флексибилна рамка за овластување мора да го земе предвид контекстот [1, 22, 88]. Од друга страна пак, од контекстот на системот може да се заклучи дека се работи за итна ситуација која бара објавување на чувствителни податоци за соодветните органи да можат соодветно да реагираат. На пример, ако некоја личност носи сензор за срцев ритам, најверојатно ќе сака да ги задржи своите мерења приватни, но ако настане итна ситуација, пожелно е податоците да се отворат за медицинските лица да можат правилно и навремено да реагираат. Оттука, платформите за авторизација мора да обезбедат истовремена заштита и изложување на податоци, врз основа на контекстот на системот.

Асоцијација на контекст-податоци-барател

Барањата за авторизација можат да ги дефинираат заштитените податоци D во однос на барателот [61, 29, 68] и неговата околина [1, 88]. Затоа, јазикот за полиси мора да ги поврзе својствата на барателот, заштитените податоци и контекстот.

Агрегирано споделување на податоци

Сведоци сме дека *Crowd source* апликациите обезбедуваат многу корисни услуги за корисниците, но за возврат зависат од споделување на анонимни и агрегирани чувствителни податоци. Следствено, платформите за авторизација треба да овозможат изложување на делови од заштитените податоци врз основа на согласноста на корисникот.

Решавање конфликти

Бидејќи повеќето од платформите за авторизација поддржуваат повеќе полиси за дозвола и забрана на пристап, конфликт може да се појави кога една полиса дозволува и друга забранува интеракции со ист дел од заштитените податоци за истиот барател. Механизмите за разрешување на конфликти [51, 61, 34, 88] се клучни за конзистентна заштита на податоците, давајќи им можност на сопствениците на податоци да изберат кој дел од спротивставените податоци треба да биде дозволен.

2.2.2 Одржливост

Одржливоста се одредува со времето потребно за да се трансформираат безбедносните барања во полиси. Механизмите кои го поддржуваат управувањето со полиси можат да го поедностават овој процес и да обезбедат попрецизна заштита [15, 52].

Детекција на конфликти

Конфликтите не се секогаш очигледни и тие можат да останат незабележани подолг временски период [15, 56]. Затоа, треба да се обезбеди механизам за откривање на конфликти со цел да се детектираат конфликтите во самата фаза на креирање полиси, спречувајќи несоодветна заштита на податоците.

Севкупна заштита на податоци

Податоците заштитени со секоја полиса, заедно со севкупните заштитени и незаштитени податоци се многу важни за точна заштита на податоците [59]. Ако одредена платформа за авторизација нуди преглед на покриеноста на податоците, сопственикот на податоци е свесен за деловите кои се незаштитени и може да реагира со наведување на дополнителни полиси по потреба. Немањето вакви информации може да доведе до двосмисленост и изложеност на податоци кои треба да бидат заштитени.

Валидација на точност

Тестирањето на полисите е предизвикувачка задача, бидејќи барателите и контекстот воведуваат високодимензионален простор, кој е тешко да се покрие со тест сценарија, па дури и да се разбере. Со цел да се обезбеди правилна заштита за време на дизајнот, неопходно е да се потврди дека секоја полиса одговара на барањето што го претставува [59]. Затоа, платформите за авторизација мора да овозможат механизми кои ќе обезбедат преглед на податоците заштитени со полисите, така што сопственикот на податоците може да провери дали овие податоци одговараат на барањето.

Разбирливост

Разбирливоста може да се набљудува како време што е потребно за учење и прилагодување на формат на полисите и јазикот за нивно дефинирање во плат-

формите за авторизација [52, 74]. Оттука, употребата на стандардизиран јазик за полиси, кој е поддржан од поголема заедница, може значително да го намали времето на учење и да го направи јазикот поразбирлив.

2.3 Преглед на системи за заштита на семантички податоци

За да ја разгледаме поврзаната работа во однос на принципите за авторизација во §2.2, ја креиравме Табела 2.2, во која се резимирани и споредени најрелевантните пристапи за авторизација на поврзани податоци.

Колоната *Операција* ја претставува *Опфатеноста на операции* и покажува дека заштитата на *read* операцијата е најпопуларна. Сепак, јазиците за полиси кои ги поддржуваат операциите за вметнување и бришење [45, 75, 29, 22, 51] не нудат можност за селекција на податоци преку комбинација на еден или повеќе шаблони за графови (Basic Graph Patterns - BGP). Исклучок е јазикот за полиси во [29], каде што авторите само ги споменуваат овие операции, без објаснување како се спроведуваат. Операцијата за управување е поддржана само во [51]. Во [88], полисите ги штитат акциите како целина, со користење на онтолошки дефиниран контекст за активација.

Групата на колони *Грануларност* го опфаќа соодветниот аспект на флексибилност во однос на барателот (групата колони *Барател*) и податоците што се заштитени (групата колони *Заштитени податоци*). Способноста да се користат шаблони за тројки за *Барателот* и *Заштитени податоци* е прикажана во колоните *TP*. Колоните *BGP* покажуваат дали чуваните податоците или барателот може да бидат избрани преку основни шаблони за графови. Во колоните *Gm* и *DSm* се прикажува способноста за претставување на припадност во граф или податочно множество, соодветно. Во [45, 35, 67, 51], барателот е специфициран со својот Интернационален идентификатор на ресурси (International Resource Identifier - IRI)[13], со својата класа во [45, 57, 51], и со својата улога во [57, 19]. SPARQL прашалниот јазик се користи во [75, 22] и овозможува активирање на полисите во однос на припадноста на барателот во одреден именуван граф. Кога станува збор за дефинирање на заштитените податоци, јазиците за полиси базирани на SPARQL [29] нудат најголема флексибилност, бидејќи ја искористува SPARQL експресивноста, по што следуваат јазиците кои користат SWRL [61] или еквивалентна синтакса [1, 68], чии што недостаток е неможноста да ја изразат припадноста на податоците во именувани графови.

Колоната *Контекст* опишува дали пристапот го опфаќа аспектот за *Контекстна заштита*. Иако контекстот е дел од полисите во [1], авторите не даваат

[13] http://www.ietf.org/rfc/rfc3987.txt

Табела 2.2: Покриеност на принципите за флексибилност при авторизација

	Операции	Грануларност						Контекст	Асоцијација		
		Барател		Заштитени податоци							
		TP	BGP	TP	BGP	Gm	DSm		CR	CD	RD
Hollenbach et al. [45]	R, I, D	p					✓				
Sacco et al. [76]	R, I, D	✓	✓	✓		✓					
Toninelli et al. [88]	A	p	p	p	p			✓	✓	✓	✓
Muhleisen et al. [61]	R	✓	✓	✓							✓
Flouris et al. [34]	R			✓	✓						
Kirrane [57]	R	p		✓		✓					
Dietzold & Auer [29]	R, I[?], D[?]	✓	✓	✓	✓	✓					✓
Costabello et al. [22]	R, I, D	✓	✓	p		✓		✓	✓		
Franzoni et al. [35]	R	p		✓	p						
Abel et al. [1]	R	✓	✓	✓	✓			✓	✓	✓	✓
Chen&Stuckenschmidt [19]	R	p		p							
Oulmakhzoune et al. [67]	R	p		✓	p						
Kirrane [51]	R, I, D, M	p		✓		✓					
Padia et al. [68]	R	✓	✓	✓							✓
TP - шаблони на тројки; BGP - basic graph pattern; Gm - припандост во граф; DSm - припадност во податочно множество; ✓- да; p - делумно; i - IRI на ресурс; c - Класа; r - улога; R - Читање; I - Вметнување; D - Бришење; M - Управување; A - Акција; C - Контекстуална заштита; []$^{?}$ - Не е објаснето; CR - контекст-барател асоцијација; CD - контекст-податоци асоцијација; RD - барател-податоци асоцијација;											

детално објаснување за својствата што се користат за негова изградба. *Schi3ld рамката* [22] користи статички контекст претставен со *prismma* онтологијата [14], додека [88] користи дефиниција на динамички контекст за опис на околината на извршување на акциите.

Колоните групирани како *Асоцијација* покажуваат дали е можно да се комбинира контекстот со барателот (колона *CR*) или со податоците (колона *CD*) и дали пристапот поддржува комбинација на својства на барателот со податоците (колона *RD*). Оваа група на колони ја сумира покриеноста на аспектот за *Асоцијација на податоци од контекст-податоци-барател*. Од постоечките системи, само [88] и [1] можат да ги поврзат заедно својства на сите овие податоци. Рамката Proteus [88] го активира контекстот врз основа на правилата дефинирани во описна логика (Descriptive Logic - DL) и логичко програмирање (Logic Programming - LP), со што се овозможува поврзување на барателот со податоците и контекстот. Работата во [1] го користи јазикот за полиси Protune [27] за да дефинира полиси кои овозможуваат дозвола или забрана на тројки врз основа на совпаѓање на основните шаблони за графови (BGP). Во [61, 29], полисите се дефинирани во јазикот за семантички веб правила (Semantic Web Rule Language - SWRL) [46], што овозможува комбинирање на својствата на барателот со податоците. Формат на полисите со слична експресивност како и SWRL се користи во [68]. Работата во [22] ги поврзува статичките контекстни информации со својствата на барателот преку дефинирање на SPARQL прашање [71, 43], кое се користи за активација на полисите. Јазиците за полиси кои не ги поврзуваат својствата на барателите со податоците кои се заштитуваат обично имаат фаза на активирање на полисите [22, 75, 45, 51, 35], што зависи од барателот и од акцијата што тој/таа има намера да ја изврши. Оваа фаза на активирање ги избира применливите полиси кои подоцна се применуваат за да се заштитат податоците. Фазата на примена на полисите може да ги филтрира само дозволените податоци [61, 34, 29, 22, 1, 35, 67, 51, 68], или може да ја дозволи или забрани бараната акција како целина [45, 88].

Остатокот од принципите за авторизација се маргинално опфатени само во разгледуваната литература, и затоа, тие не се прикажани во Табела 2.2. Повеќето од авторите не ги покриваат аспектите за *разрешување на конфликти*, додека во [1, 34, 22] се користи предефинирана стратегија за разрешување на конфликти, каде што полисите кои забрануваат пристап имаат поголем приоритет, или обратно. Сепак, предефинираната стратегија не обезбедува флексибилност во решавањето на конфликтите, бидејќи администраторот на полисите не може да наведете која има повисок приоритет, а со самото тоа ја губи контролата на процесот. Алгоритмите за решавање на конфликти [51] делумно го решаваат овој проблем, но тие се сложени и бараат обемно тестирање и валидација. Приоритетите на полисите [88, 54] обезбедуваат најфлексибилен начин за разрешување на конфликти и се најблиску до начинот на кој луѓето ги разрешуваат конфликтите.

[14] http://ns.inria.fr/prissma/v2/prissma_v2.html

Детекцијата на конфликтите е потребена за откривање на аномалии на барањата и овозможување нивно решавање пред да биде направена некоја штета врз податоците со неовластен пристап. Правила за откривање на аномалии низ полисите, кои се способни да детектираат противречности или конфликти, се дефинирани во [19]. Меѓутоа, во овие пристапи јазикот на полиси нема доволно флексибилност и кореспонденцијата на полисите со соодветните барања не се разгледува. Слично, во [74] безбедноста и конзистентноста на полисите се разгледува преку нивните интеракции. Во врска со аспектот за *Севкупната заштита на податоци*, авторите во [34, 67] ги земаат предвид конфликтите и податоците за покриеноста на полисите, но тие само дискутираат за предефинираните механизми за разрешување на конфликти помеѓу полисите и не ги прикажуваат овие информации за време на примената на полисите.

Покрај флексибилноста, *Валидацијата на точноста* на полисите ретко се дискутира во разгледаните платформи. Во [51] е предложена методологија за проверка на правилноста на рамките за авторизација со препишување на прашањата, но оваа методологија не е погодна за валидација на дизајнот на полисите во однос на барањата. Јазикот за полиси треба да овозможи дизајнирање на алатки кои ќе обезбедат тестирање на полисите за да може да се провери дали тие ги исполнуваат барањата за кои се дизајнирани.

Главниот предизвик кој останува отворен е обезбедување на валидација на точноста на полисите за време на нивното дизајнирање, истовремено обезбедувајќи флексибилност за да може да се моделираат барањата изразени преку природен јазик [90]. Во оваа насока, системот презентиран во [1] користи најкомплетниот јазик за полиси - Protune [27]. Сепак, овој систем ги штити само *read* операциите и не е во можност да ги заштити податоците според нивната припадност во именуван граф или податочно множество. Дополнително, авторите не дискутираат за детекција и решавање на потенцијанлите конфликти. Само авторите во [51] ја потврдуваат коректноста на имплементацијата на нивниот систем за авторизација, но тие не обезбедуваат проверка соодветноста на полисите во однос на барањата за време на нивниот дизајн. Во итни ситуации, свеста за контекстот е од суштинска важност за сопственикот на податоците. Во такви сценарија, полисите треба да може да се активираат при "вонредни состојби"и да го променат стандардното однесување на системот така што ќе ги "отворат"чувствителните приватни информации за сервисите или личностите што можат да му помогнат на сопственикот на податоците во таквите моменти. Освен тоа, изложување на агрегирани информации со анонимизација на осетливите лични информации може да овозможи иновација на нови и понапредни услуги за заедницата (Crowd Source Services) наменети што ќе го подобрат квалитетот на животот воопшто. Според сите наши сознанија, не постои јазик за полиси кој обезбедува ваква заштита. Нашата работа има за цел да ги пополни празнините на постојните пристапи со предлагање на флексибилна и комплетна платформа за авторизација на податоци, која ги опфаќа сите претходно дискутирани принципи за дизајн во §2.2.

Глава 3

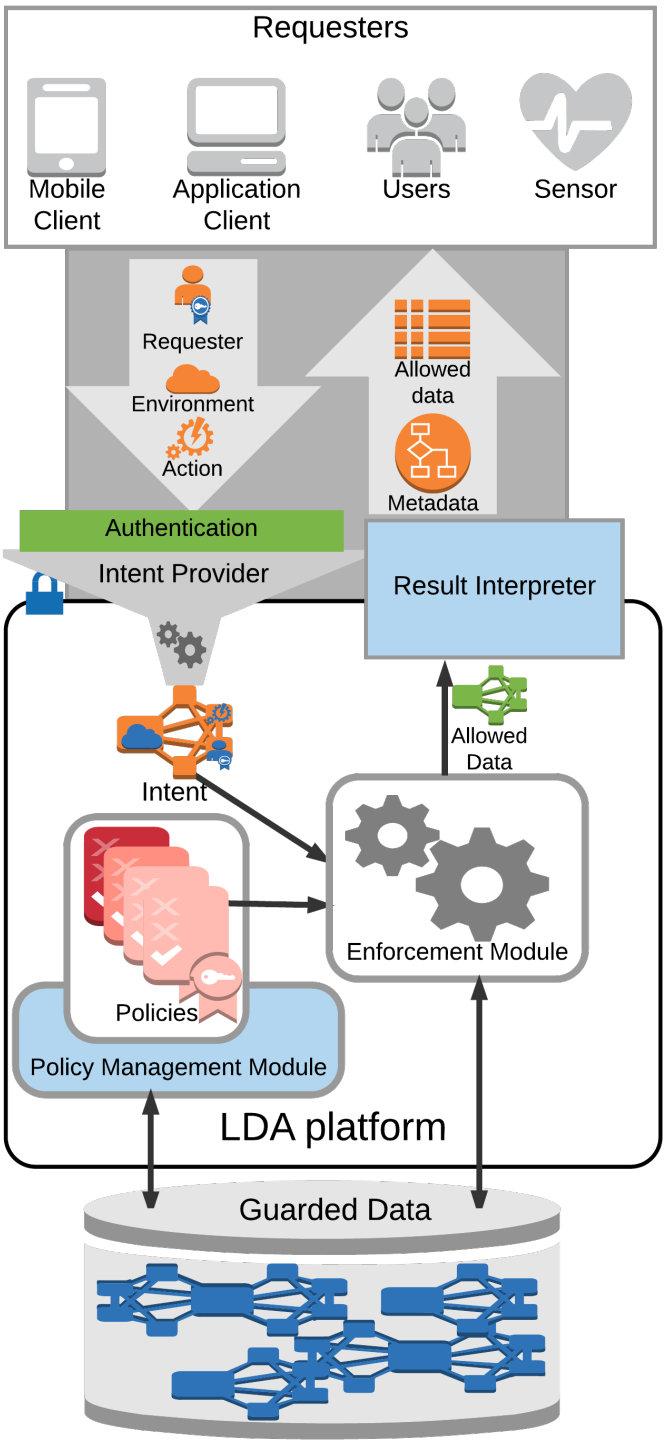
Платформа за авторизација кај SPARQL протоколот

3.1 Архитектура на платформата

Во ова поглавје ќе биде прикажана архитектурата на платформата за авторизација на поврзани податоци (Linked Data Authorization - LDA)[85].

Целта на LDA платформата е да ги примени безбедносните барања на клиентите претставени во платформата како полиси. Архитектурата на LDA платформата е прикажана на сликата 3-1. LDA платформата ги штити чуваните податоци $\mathbb{D}$ (Дефиниција 9) со примена на барањата за заштита претставени со безбедносните полиси (Дефиниција 15). Намерата (Дефиниција 10) го опишува дејството што барателот се обидува да го изврши, кое во одреден момент треба да има интеракција со чуваните податоци $\mathbb{D}$. Сликата 3-5 (a) ги покажува *Дозволените податоци* (Дефиниција 11) синтетизирани од *модулот за извршување* за повеќе активирани полиси, што е подмножество на заштитените податоци, додека Сликата 3-5 (b) дополнително покажува дека *модулот за извршување* го филтрира и враќа само дозволениот дел од бараните резултати (означени со IR).

Една од централните компоненти на платформата е компонентата *IntentProvider* која ја дизајниравме за да ја извлечеме Намерата (Intent) $\mathbb{I}$ од достапните податоци во платформата, кои обично го опишуваат барателот, околината и дејството. Оваа компонента овозможува вградување на динамички контекстуални податоци во Намерата $\mathbb{I}$ и дизајнирање заштита околу неа со полиците. Идејата за структурата на *Намерата (Intent)* се базира на нашата анализа на различни дистрибуирани мултипроцесорски платформи коишто чуваат и генерираат голем дел од денешните податоци и е конкретно инспирирана од



Слика 3-1: Архитектура на LDA платформата

компонентата *Intent* во оперативниот систем Android [1]. Концептот на *Intent* во Андроид обезбедува флексибилен начин на изразување на намерата на корисникот и неговата околина, така што ја прифативме оваа структура на податоци со цел да ја претставиме намерата на барањето во LDA платформата. Во LDA платформата, Намерата $\mathbb{I}$ се имплементира како множество тројки на RDF што го опишуваат контекстот во кој се извршува планираната акција, вклучувајќи ја операцијата, параметрите на акцијата, барателот со своите својства и настанот кој ја повикува акцијата. Дополнително, изразот Намера е избран поради тоа што претставува имплицитна намера на корисникот, но не сме сигурни дали ќе биде дозволена, одбиена или делумно извршена.

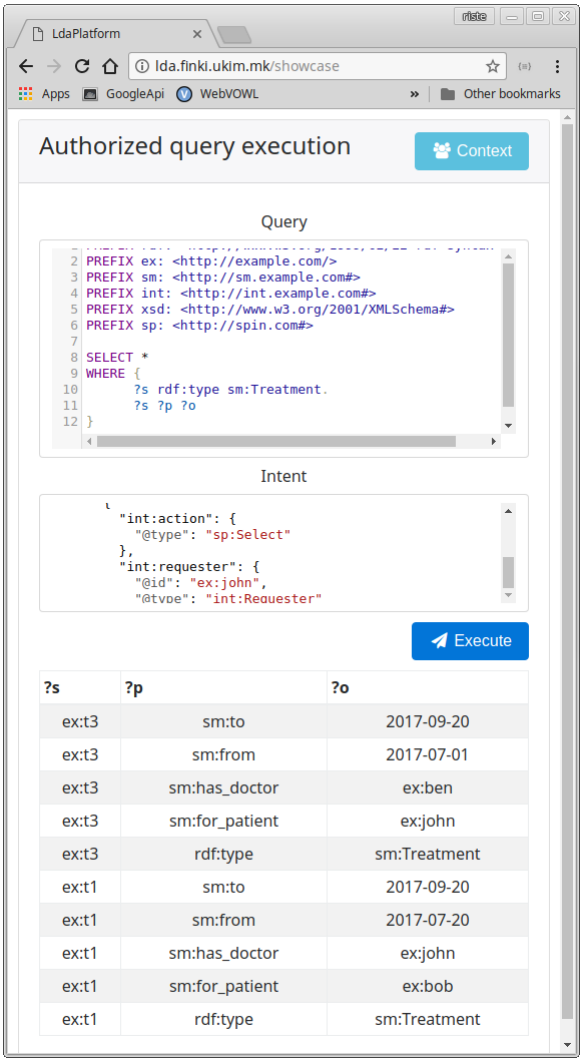
Модулот за извршување ги спроведува алгоритмите со кои се обезбедува овластена интеракција со дозволените податоци. Неговата задача е да осигура дека барателот нема да има можност за интеракција со податоци кои не се дозволени за него/неа. Поради модуларната архитектура на платформата, имплементирани се неколку варијанти на овој модул, кои овозможуваат различни нивоа на заштита на податоците и имаат различни импликации на перформансите.

Однесувањето *Модулот за извршување* и соодветната заштита зависат од точноста на полисите. *Модулот за управување со полиси* ја поддржува валидацијата на точноста на правилата за време на дизајнирање и обезбедува откривање на конфликти, самостојни тестирања на полисите, преглед на заштитените податоци според полиса и намера и екстракција на заштитените и незаштитените податоци воопшто. Овој модул е важна алатка за сопствениците на податоци и го поедноставува процесот на одржување на полисите.

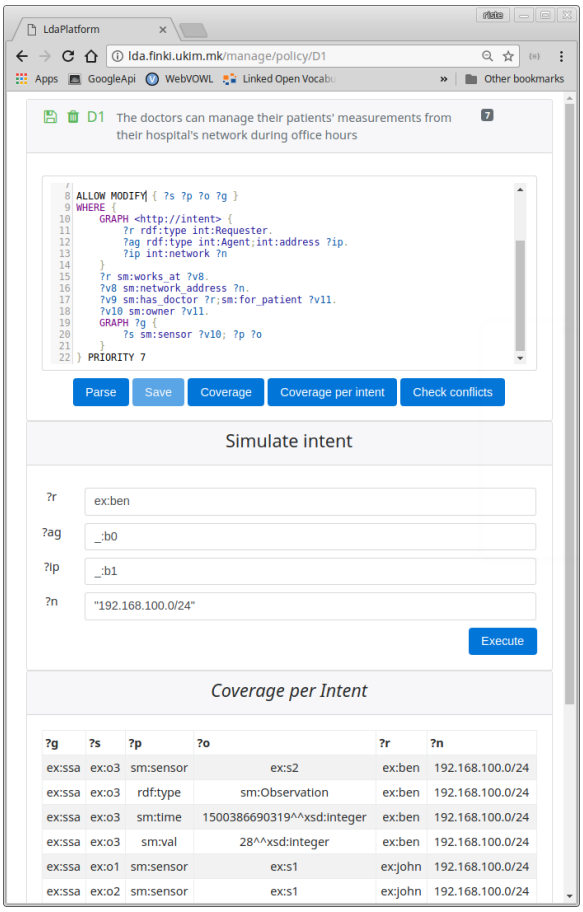
3.1.1 Генеричка имплементација

Со цел да се потврди имплементацијата на LDA платформата, развивме целосно функционална прототип имплементација. За да овозможиме интерактивно тестирање на платформата, развивме заштитена конзола за извршување на барањето (Слика 3-2)[2]. Оваа интерактивна конзола обезбедува испраќање на произволно барање кон податоци заштитени LDA платформата во контекст што може да се конфигурира. Копчето *Context* на Слика 3-2 отвора модален прозорец, каде што корисникот ги специфицира контекстните параметри во RDF формат[3]. Податоците од конзолата се испраќаат до компонентата *IntentProvider* која ги анализира овие податоци, создава Намера и ја испраќа кон модулот за извршување, кој ја обезбедува интеракција само со дозволените податоци. Во случај на операцијата за читање со користење на прашање, *ResultInterpreter* компонентата ги враќа само дозволените резултати до корисникот, заедно со објаснување на резулта-

[1] https://developer.android.com/reference/android/content/Intent.html
[2] http://lda.finki.ukim.mk
[3] https: // www. w3.org/TR/rdf11-concepts/



Слика 3-2: Извршување на авторизирани барања во LDA платформата



Слика 3-3: Модул за управување со полисите

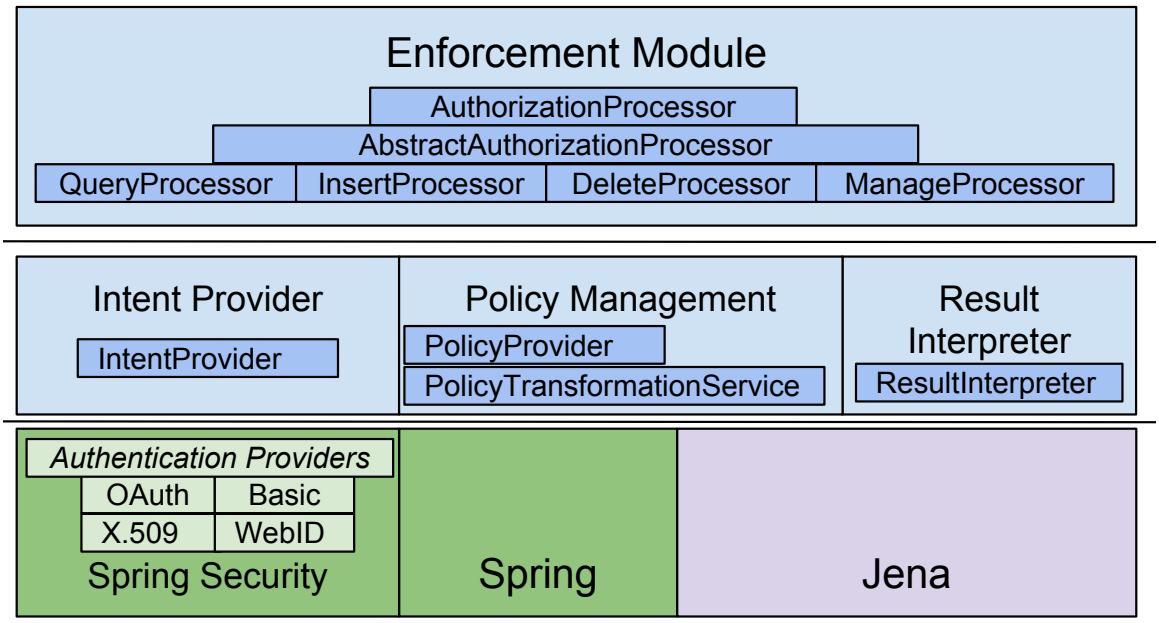
тите. Дополнително, со цел да се поддржи и поедностави администрацијата на полисите, развивме кориснички интерфејс за функционалностите на *модулот за управување со полиси* (Слика 3-3).

Главниот предизвик на LDA платформата е дизајнот на проширлива и модуларна архитектура, независна од механизмот за автентикација, способен за флексибилна авторизација и управување со безбедносните полиси. Со цел да се задоволат барањата за модуларност, LDA платформата ја користи Spring рамката[4], како што е прикажано на Сликата 3-4. Сите семантички операции се спроведуваат со користење на библиотеката Apache Jena [58], која главно се користи во процесите за примена на авторизацијата и управувањето со полисите.

Вториот слој на Слика 3-4 ги прикажува модулите кои го поддржуваат *модулот за извршување*, кој ги извршува алгоритмите за авторизација. Модулот *IntentProvider* има клучна улога во авторизацијата заснована на контекст. Сегашната имплементација на модулот ја користи една од најсовремените библиотеки, Spring Security [79], која обезбедува повеќе механизми за автентикација. Тековна-

[4]http://spring.io

та имплементација на *IntentProvider* го користи протоколот WebID за автентикација [83, 87]. Сепак, поради конфигурабилноста што ја обезбедува библиотеката Spring Security, компонентата *IntentProvider* може лесно да се модифицира за да се користи друг *AuthenticationProvider* и да се добие Намерата со користење на друг протокол за автентикација, како што е протоколот OAuth[5], X509 сертификати[6], Base Authentication footnote https://tools.ietf.org/html/rfc2617 и многу други. Модулот за *управување со полисите* овозможува едноставно одржување и проверка на точноста. *PolicyProvider* компонентата ги враќа полисите врз основа на операцијата што се извршува и податочното множество за кое се наменети. *PolicyTransformationService* се користи за извлекување на деловите од полисите со користење на Jena API и ги спроведува алгоритмите за трансформација опишани во §4. Модулот *ResultInterpreter* ги запишува резултатите во бараниот формат и додава објаснување за резултатите кои се враќаат.



Слика 3-4: Технологии користени за имплементација на LDA платформата

Сликата 3-4 исто така покажува дека *модулот за извршување* го изложува интерфејсот *AuthorizationProcessor*, кој го прифаќа поднесеното барање кое треба да биде заштитено, *Намерата*, податочното множество над кое треба да се изврши прашањето и имплементацијата на интерфејсот *ResultInterpreter* кој се користи за серијализирање и објаснување на резултатите. Притоа во рамките на LDA платформата постојат повеќе имплементации на *AuthorizationProcessor* кои функционираат на различен начин и се однесуваат за различни атомични операции. Деталите за различните имплементациина овој модул се опишани во §3.4 и §3.5.

[5]https://tools.ietf.org/html/rfc6749
[6]https://tools.ietf.org/html/rfc5280

3.1.2 Формални дефиниции

Со цел да се поддржи проверка на одржливоста и точноста, се обидуваме формално да ја дефинираме LDA платформата и јазик за полиси што овозможува дизајнирање на алгоритмите за трансформација кои се користат во *модулот за управување со полисите* и кои овозможуваат независни тестирања на полисите, откривање конфликти и екстракција на заштитените и незаштитените податоци воопшто.

Дефиниција 8 ***Платформата за авторизацијана поврзани податоци (Linked Data Authorization - LDA)*** *може да се претстави како торка* $\langle \mathbb{D}, \mathbb{I}, \mathbb{P} \rangle$*, каде* $\mathbb{D}$ *се чуваните податоци,* $\mathbb{I}$ *е Намерата на барателот и* $\mathbb{P}$ *е множеството на полиси кои се дефинирани во платформата.*

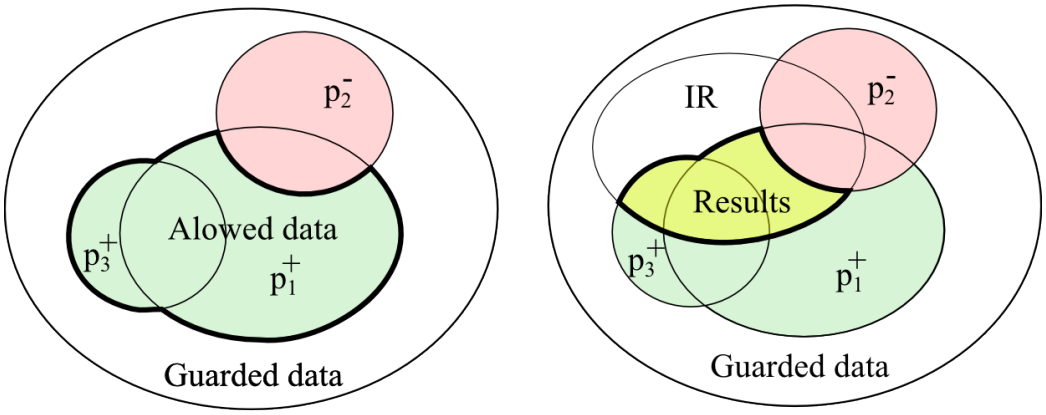
Дефиниција 9 ***Чуваните податоци*** ($\mathbb{D}$) *може да се претстават со множество од торки* $\langle S, P, O, G, DS \rangle$ *кое ги опишува RDF тројките* $\langle S, P, O \rangle \in (I \cup B) \times (I \cup B) \times (I \cup B \cup L)$ *кои се логички организирани во именувани графови* G*, а физички се зачувани во податочно множество* DS*.* I, B *и* L *ги претставуваат множествата од сите интернационални идентификатори за ресурси (IRI), анонимните јазли и литералите, соодветно.*

Дефиниција 10 ***Намерата*** ($\mathbb{I}$) *е семантичка претстава на намерата на барателот, составена од повеќе RDF тројки.*

Дефиниција 11 ***Дозволените податоци*** *се најголемото подмножество на чуваните податоци* $\mathbb{D}$ *кои се дозволени за Намерата* $\mathbb{I}$ *од полисите* $\mathbb{P}$*.*

3.1.3 Авторизација на атомичните операциии

Секој систем треба иницијално да ги идентификува атомичните операции кои се користат во неговото фуниционирање. Кога станува збор за систем за управување со поварзани податоци, атомични операции се читање, додавање, бришење и управување со именувани графови. Акциите кои треба да обезбедат бизнис доменска функционалност се зансоваат на атомичните операции во нивната имплементација. Поради оваа причина, LDA платформата овозможува авторизациска обвивка околу атомичните опрации, со што се добива дека бизнис доменските акции не би добиле пристап до податоци за кои не се авторизирани.



Слика 3-5: Заштитени податоци по намера

Авторизација на операции за чиање: READ

Операциите за читање се користат за да екстрахираат податоци кои се од интерес и кои задоволуваат одреден критериум. Потребниот критериум кој треба да го задоволат податоците најчесто се изразува преку прашање $Q(D)$ кое се извршува над податочните множества D. Сепак, во процесот на авторизација на операциите за читање, тие зависат и од Намерата $\mathbb{I}$ и од конфигурираните полиси $\mathbb{P}$ за да може да определат кои податоци се дозволени.

Дефиницијата 12 го формализира авторизираниот резултат на операциите за читање и покажува дека дозволените податоци за прашањата треба да се добијат со нивно извршување над дозволените податоци. Изразот за *условна авторизација* од Дефиницијата 12 се однесува на сценариото каде не е дозволено враќање само на делот со дозволени податоци, туку резултатите се враќаат само доколку се' што е побарано е дозволено. Овој начин на заштита е соодветен доколку е потребна конзистентност на авторизираните резултати.

Дефиниција 12 ***(Заштита на читање)***

а) авторизација со филтрирање: $R(\mathbb{I}, Q) := Q(\alpha_{read}(\mathbb{I}, \mathbb{D}, \mathbb{P}))$

б) условна авторизација: $Q(D) = R(\mathbb{I}, \mathbb{Q})$

Со овој пристап се овозможува имплицитна заштита на податоците дури и во случаите кога барателот е свесен за структурата на податоците и е во можност да изврши повеќе прашања со цел да ја открие потребната вредност. Пример за вакви прашања е кога одреден барател би ги побарал имињата на личностите со телефони кои содржат одредени цифри. Доколку пристапот до телефонските

броеви не е дозволен, системот треба да се осигура дека нема истите да се искористат во филтрирањето на податоците, а не само дека тие како такви нема да бидат вратени како резултат. Точноста на дефиницијата 12 како и можноста за имплицитна заштита на податоците се потврдува и во [51], каде авторите ги користат дозволените податоци од оваа дефиниција за да докажат дека нивниот алгоритам со препишување на прашањата е *сигурен*.

Заштита на операциите за бришење: DELETE

Операцијата за бришење обично бара да се избришат една или повеќе четворки од постојните податоци, дефинирани директно или со прашање што ги избира четворките за отстранување. Како што е покажано со Дефиниција 13, само дозволените податоци треба да се избришат во сценариото за авторизација со филтрирање. Бидејќи податоците се веќе присутни во податочното множество, функцијата за екстракција на дозволените податоци α_{delete} ќе ги врати постоечките четворки што смеат да се отстранат. Податоците авторизирани за отстранување се всушност пресекот на дозволените и оние кои се побарани. Кога треба да се зачува конзистентноста на податоците, што е во повеќето случаи, сите барани податоци мора да бидат присутни во дозволеното множество.

Дефиниција 13 ***Delete protection***

а) авторизација со филтрирање: $D(\mathbb{I}, q_d) := q_d \cap \alpha_{delete}(\mathbb{I}, \mathbb{D}, \mathbb{P})$

б) условна авторизација: $q_d = D(\mathbb{I}, q_d)$

Заштита на операциите за додавање: INSERT

Операциите за додавање на податоци (*INSERT*) обично ја модифицираат состојбата на базата на податоци и примаат множество од четворки што треба да се додадат (d_i) како дополнување на Намената. Бидејќи повеќето од платформите побаруваат конзистентност на податоците, условната заштита е највообичаен начин за додавање на податоци, каде што треба да бидат одобрени сите четворки што се побарани за креирање. Таков пример се релациони бази на податоци кои ги ограничуваат податоците со нивниот модел. Сепак, постојат и платформи со полабавни ограничувања кои можат да овозможат делумен внес на податоци. Во овој случај, делумните податоци дозволени за додавање се $I(\mathbb{I}, q_i)$ од Дефиниција 14.

Дефиниција 14 ***Insert protection***

а) авторизација со филтрирање: $I(\mathbb{I}, q_i) := q_i \cap \alpha_{insert}(\mathbb{I}, \mathbb{D} \cup q_i, \mathbb{P})$

б) условна авторизација: $q_i = I(\mathbb{I}, q_i)$

Бидејќи во нашиот јазик полисите ја опишуваат конечната состојба на базата на податоци по извршувањето на интеракцијата, функција за екстракција на дозволени податоци α_{insert} ги избира дозволените последици за операцијата за додавање. Овие податоци се користат за да се одреди која од бараните четворки е дозволена за додавање со $I(\mathbb{I}, q_c)$. Кога е потребна конзистентност на податоците, изразот за условна авторизација од Дефиниција 14 покажува дека операцијата ќе биде дозволена само кога сите барани четворки се дозволени за додавање.

Заштита на операциите за управување со графови: MANAGE

Атомичните операциите за управување со графови примаат аргументи кои означуваат која е операцијата и врз кој или кои именувани графови треба да се изврши. Притоа овие информации се инкорпорирани во рамките на Намерата. Кај овие операции не е возможна авторизација со филтрирање бидејќи нивниот ефект не е деллив според аргументите кои се проследени. Токму затоа кај овие операции е можна само условна авторизација каде се дозволува или забранува нивното извршување во целост. Поради ова, доколку постои конфликт кај полисите кои се однесуваат на операциите за управување со графови, резултатот го диктира полисата со највисок приоритет т.е. дозволата од полисата која е со највисок приоритет се смета за конечна.

Бидејќи и аргументите за бизнис доменските акции се опишани и преку намерата, истата постапката за заштита на операциите за упраување со графови се користи и за нивна заштита.

3.2 Јазик за полиси

Полисата (Дефиниција 15) ги специфицира заштитените податоци по извршувањето на акцијата. Дополнително, акциите може да имаат потреба од извршување на повеќе операции, кои треба да бидат авторизирани. Полисите за операцијата за читање (*READ*) ги дефинираат податоците кои можат да се добијат од чуваните податоци $\mathbb{D}$, додека полисите за модификација (*INSERT* и *DELETE*) ја дефинираат дозволената состојба на чуваните податоци $\mathbb{D}$ по извршувањето на операцијата. *MANAGE* полисите опишуваат дали операцијата треба да биде дозволена или не. На овој начин, полисите полесно да се претстават, разбираат и дефинираат. Во зависност од операцијата, активираните полиси дефинираат дали дејството ќе биде одбиено, дозволено или ќе има делумен пристап до дозволените податоци.

Дефиниција 15 ***Полисата*** *е торка* $\langle \epsilon, o, q, \varphi_i, \varphi_d, \rho \rangle$*, каде:*

- $\epsilon \in \{allow(+), deny(-)\}$ *е дозволата за интеракција со заштитените податоци* $\mathbb{D}^{(\mathbb{I},p)}$ *на полисата.*
- $o \in \{READ, INSERT, DELETE, MANAGE\}$ *е операцијата која треба да се заштити*
- q *е шаблон за четворки кој служи за проектирање на екстрахираните мапирања на променливи во заштитени податочни четворки* $\mathbb{D}^{(\mathbb{I},p)}$*.*
- φ_i *и* φ_d *се предикатни функции кои се користат за екстракција на мапирања на променливите од Намерата* $\mathbb{I}$ *и од чуваните податоци* $\mathbb{D}$*, соодветно.*
- ρ *е приоритетот на полисата кој се користи при разрешување на конфликти.*

Дефиниција 16 ***Предикатна функција*** $\varphi(t_1, ..., t_n)$ *е n-арна функција која враќа true ако елементите* $t_i \in T = I \cup B \cup L \cup \{nil\}$ *ги задоволуваат нивните предикати, каде* $n = |vars(\varphi)|$*.* $vars(\varphi)$ *е функција која екстрахира множество на променливи кои се дефинирани во предикатната функција* φ*, додека* $vars(\varphi, i)$ *ја враќа променливата која е i-тиот аргумент на функцијата* φ*.*

Заштитените податоци на една полиса зависат од Намерата $\mathbb{I}$ и тие се подмножество на чуваните податоци $\mathbb{D}$. Дефиницијата 17 опишува како се добиваат овие податоци за дадена полиса p и Намера $\mathbb{I}$.

Дефиниција 17 ***Заштитени податоци*** $\mathbb{D}^{(\mathbb{I},p)}$ *за Намера* $\mathbb{I}$ *и полиса* $p = \langle \epsilon, o, q, \varphi_i, \varphi_d, \rho \rangle$ *се множеството од четворки кои се добиваат како резултат на изразот:*

$$\mathbb{D}^{(\mathbb{I},p)} = \pi(q, \sigma(\varphi_i, \mathbb{I}, \varphi_d, \mathbb{D}))$$

Дефиниција 18 ***Функцијата за евалуација на променливи*** $\sigma(\varphi_1, D_1, \varphi_2, D_2)$ *го екстрахира множество од елементи од* $D_1 \cup D_2$ *кои ги задоволуваат предикатните функции* φ_1 *и* φ_2 *така што:*

$$\begin{aligned}
\sigma(\varphi_1, D_1, \varphi_2, D_2) = \{&(t_1, ..., t_{m+n}) \mid \\
&m = |vars(\varphi_1)| \ \wedge \ n = |vars(\varphi_2)| \\
&\wedge \ \varphi_1(t_1, ..., t_m) = true \ \wedge \ \varphi_2(t_{m+1}, ..., t_{m+n}) = true \\
&\wedge \ \forall i \in [1..m], \ \forall j \in [m+1..m+n] \Rightarrow \\
&\quad t_i \in D_1 \ \wedge \ t_j \in D_2 \\
&\quad \wedge \ vars(\varphi_1, i) = vars(\varphi_2, j) \Rightarrow t_i = t_j \\
\}&.
\end{aligned}$$

Дефиниција 19 ***Функцијата за проектирање на променливи*** $\pi(v_d, v_s, t)$ *го ограничува множеството на елементи на торката* t*, при што променливите чии имиња припаѓаат во* v_s *се редуцираат и преименуваат во променливи кои припаѓаат во множеството* v_d*, така што за* $m = |v_d|$*,* $n = |v_s|$ *имаме:*

$$\forall(t'_1, ..., t'_m) \in \pi(v_d, v_S, (t_1, ..., t_n)) \Rightarrow$$
$$\forall j \in [1..m] \Rightarrow t'_j = \begin{cases} t_i, \ \exists v_d[j] = v_s[i] \\ nil, \ otherwise \end{cases}$$

Според Дефиниција 17 и Дефиниција 18, заштитените податоци се добиваат со екстракција на торките $(t_1, ..., t_m)$ од Намерата $\mathbb{I}$ кои ги задоволуваат предикатите во φ_i, торките $(t_{m+1}, ..., t_{m+n})$ од чуваните податоци $\mathbb{D}$ кои ги задоволуваат предикатите во φ_d, по што ги поврзуваат овие резултати со користење на изразот $vars(\varphi_1, i) = vars(\varphi_2, j) \Rightarrow t_i = t_j$. Потоа, проекцијата π од Дефиниција 19 ги редуцира овие резултати во торки со четири елементи кои одговараат на променливите што ги опишуваат финалните податоци (четворки) кои треба да се заштитат q.

Комбинација на полисите

Во секој не-тривијален систем барањата се репрезентираат со повеќе полиси. Во ваквите случаи, постои можност повеќе полиси да се активираат за ист Настан, па оттука, постои потреба за комбинирање на податоците кои тие ги заштитуваат.

Дефиниција 20 ***Комбинацијата на заштитени податоци*** $\odot$ *е бинарен, не-комутативен оператор кој ги комбинира заштитените податоци на две полиси со иста операција* $p_1 = \langle \epsilon_1, o, q_1, \varphi_{i1}, \varphi_{d1}, \rho_1 \rangle$ *и* $p_2 = \langle \epsilon_2, o, q_2, \varphi_{i2}, \varphi_{d2}, \rho_2 \rangle$ *на следниот начин:*

$$\langle \epsilon_1, \mathbb{D}^{(\mathbb{I}, p_1)} \rangle \odot \langle \epsilon_2, \mathbb{D}^{(\mathbb{I}, p_2)} \rangle = \begin{cases} \langle \epsilon_1, \mathbb{D}^{(\mathbb{I}, p_1)} \cup \mathbb{D}^{(\mathbb{I}, p_2)} \rangle, \epsilon_1 = \epsilon_2 \\ \langle \epsilon_1, \mathbb{D}^{(\mathbb{I}, p_1)} \setminus \mathbb{D}^{(\mathbb{I}, p_2)} \rangle, \epsilon_1 \neq \epsilon_2 \end{cases}$$

Операторот $\odot$ одредува како се комбинираат податоците од две полиси. Кога две полиси имаат иста дозвола, нивните заштитени податоци се комбинираат заедно со операцијата унија $\cup$, задржувајќи ја својата дозволата. Во спротивен случај, заштитените податоци од втората полиса се отстрануваат од заштитените податоци од првата полиса со користење на операторот за разлика на множества $\setminus$, задржувајќи ја дозволата од првата полиса.

Операторот $\odot$ не е комутативен, бидејќи резултатот зависи од редоследот на операндите кога полисите имаат различни дозволи. На пример, ако $p1$ е полиса

која *дозволува* интеракција, а $p2$ е полиса која *забранува*, тогаш $p1 \odot p2 \neq p2 \odot p1$ бидејќи $\langle \epsilon_+, \mathbb{D}^{(\mathbb{I},p_1)} \setminus \mathbb{D}^{(\mathbb{I},p_2)} \rangle \neq \langle \epsilon_-, \mathbb{D}^{(\mathbb{I},p_2)} \setminus \mathbb{D}^{(\mathbb{I},p_1)} \rangle$. Ова својство на операторот $\odot$ е од големо значење за нашиот јазик, бидејќи овозможува флексибилно разрешување на конфликти. Ние секогаш ги користиме полисите за повисок приоритет како втор аргумент бидејќи тие ги прошируваат резултатите со користење на унија операторот $\cup$ или ги редуцираат резултатите со користење на операторот за разлика на множества $\setminus$.

Операторот $\odot$ може да врати различен резултат доколку се променат приоритетите на полисите. На пример, да ги разгледаме полисите p_1 и p_2 кои дозволуваат пристап ϵ_+ и p_3 која забранува пристап ϵ_-. Нека $\mathbb{D}^{(\mathbb{I},p_1)} = \{a, b\}$, $\mathbb{D}^{(\mathbb{I},p_2)} = \{b, c, d\}$ и $\mathbb{D}^{(\mathbb{I},p_3)} = \{b, d\}$, каде a, b, c, d се произволни четворки. Бидејќи овие полиси може да имаат различни приоритети, администраторот треба да избере едно од 6-те можни нивни подредувања, со што ќе го определи резултатот. Слика 3-5 (а) ја визуелизира комбинацијата на овие полиси кога $\rho_1 < \rho_2 < \rho_3$ (ρ_i го означува приоритетот на полисата p_i). Дополнително, во продолжение е прикажан пример на три дополнителни подредувања, заедно со различните резултати од комбинацијата на полисите со овие подредувања:

$$\begin{aligned}
\rho_1 < \rho_2 < \rho_3 &\Rightarrow \langle \epsilon_+, (\mathbb{D}^{(\mathbb{I},p_1)} \cup \mathbb{D}^{(\mathbb{I},p_2)}) \setminus \mathbb{D}^{(\mathbb{I},p_3)} \rangle = \langle \epsilon_+ \{a, c\} \rangle \\
\rho_2 < \rho_3 < \rho_1 &\Rightarrow \langle \epsilon_+, (\mathbb{D}^{(\mathbb{I},p_2)} \setminus \mathbb{D}^{(\mathbb{I},p_3)}) \cup \mathbb{D}^{(\mathbb{I},p_1)} \rangle = \langle \epsilon_+ \{a, b, c\} \rangle \\
\rho_3 < \rho_2 < \rho_1 &\Rightarrow \langle \epsilon_-, (\mathbb{D}^{(\mathbb{I},p_3)} \setminus \mathbb{D}^{(\mathbb{I},p_2)}) \setminus \mathbb{D}^{(\mathbb{I},p_1)} \rangle = \langle \epsilon_- \{\emptyset\} \rangle
\end{aligned}$$

Дефиниција 21 ***Функцијата за екстракција на дозволени податоци*** $\alpha(\mathbb{I}, \mathbb{D}, \mathbb{P})$ *е функција која враќа подмножество од чуваните податоци* $\mathbb{D}$ *кое е дозволено за Намерата* $\mathbb{I}$*, со користење на полисите конфигурирани во LDA платформата* $\mathbb{P}$*, така што:*

$$\begin{aligned}
\alpha &= \alpha^{(k)} \\
\alpha^{(i)} &= \alpha^{(i-1)} \odot \langle \epsilon_i, \mathbb{D}^{(\mathbb{I},p_i)} \rangle \\
\alpha^{(0)} &= \begin{cases} \langle \epsilon_+, \emptyset \rangle, \epsilon_1 = + \\ \langle \epsilon_+, \mathbb{D} \rangle, \epsilon_1 = - \end{cases}
\end{aligned}$$

каде $\mathbb{P} = \{p_i = \langle \epsilon_i, o_i, q_i, \varphi_{ii}, \varphi_{di}, \rho_i \rangle \mid i \in [1, k] \wedge \forall j \in [2, k] \Rightarrow \rho_{j-1} \leq \rho_j\}$

Дефиницијата 21 прикажува дека функцијата за екстракција на дозволените податоци α го екстрахира делот од чуваните податоци $\mathbb{D}$ кој е достапен за дадена Намера $\mathbb{I}$ и е дозволен од целото множество на полиси $\mathbb{P}$ кои се конфигурирани во LDA платформата. Од друга страна, заштитените податоци за полисата

$\mathbb{D}^{(\mathbb{I},p)}$ ги содржат само четворките кои се дозволени или забранети само од една полиса за дадена Намера $\mathbb{I}$. Функцијата α всушност ги комбинира заштитените податоци од сите полиси со користење на $\odot$ операторот, земајќи го во предвид приоритетот на полисите ρ. Изразот $\alpha = \alpha^{(k)}$ означува дека дозволените податоци за Намерата ќе бидат резултатот кој се добива по процесирањето на полисата со највисок приоритет p_k. Изразот $\alpha^{(i)} = \alpha^{(i-1)} \odot \langle \epsilon_i, \mathbb{D}^{(\mathbb{I},p_i)} \rangle$ го дефинира подредениот процесирање на полисите, бидејќи $\forall j \in [2,k] \Rightarrow \rho_{j-1} \leq \rho_j$ го покажува нивното подредување. Бидејќи операторот $\odot$ секогаш ја зема дозволата од својот прв аргумент, $\alpha^{(0)}$ се користи за да дозвола на интеракција како резултат на $\alpha^{(1)}$, додека $\alpha^{(i)} = \alpha^{(i-1)} \odot \langle \epsilon_i, \mathbb{D}^{(\mathbb{I},p_i)} \rangle$ се користи за да ја пренесе оваа дозвола до финалниот резултат. На пример, ако полисата со најнизок приоритет дозволува интеракција p_1, тогаш $\alpha^{(1)} = \langle \epsilon_+, \emptyset \cup \mathbb{D}^{(\mathbb{I},p_1)} \rangle = \langle \epsilon_+, \mathbb{D}^{(\mathbb{I},p_1)} \rangle$, т.е., ништо не е дозволено на почетокот и дозволените податоци се прошируваат со секоја наредна полиса. Во спротивниот случај, кога p_1 забранува интеракција, тогаш $\alpha^{(1)} = \langle \epsilon_-, \mathbb{D} \setminus \mathbb{D}^{(\mathbb{I},p_1)} \rangle$, т.е., се е дозволено на почетокот и секоја наредна полиса ги редуцира дозволените податоци.

Синтакса на јазикот за полиси

Улогата на јазикот за полиси е да обезбеди синтакса која ќе го поддржи формализмот на полисите. Јазикот за полисите треба да биде дизајниран за да одговара на формализамот на полисите, истовремено упростувајќи ја трансформацијата на барањата во полиси. Нашиот јазик за полиси ја проширува синтаксата SPARQL за дефинирање на полисите. Важно е да се нагласи дека нашиот јазик за полиси не е дизајниран за одредена архитектура за спроведување на заштитата и може да се искористи од различни алгоритми. Проширувањето на SPARQL се користи за да се направат полисите повеќе читливи и разбирливи, без мешање повеќе синтакси, што е случај во [75, 22, 29]. Големото количество на ресурси SPARQL го олеснува разбирањето и учењето на јадрото на нашиот јазик за полиси. Дополнителните делови воведени во нашиот јазик, надвор од SPARQL синтаксата се *permission, operation, priority* и *dataset* елементите од изворниот код 3.1, кои се едноставни и не воведуваат многу сложеност.

Во изворниот код 3.1 се прикажува дефиницијата на нашиот јазик во логика од прв ред преку Prologue синтаксата, каде се надоврзуваме на оригиналната SPARQL спецификација преку референцирање на соодветните нејзини елементи преку нивните идентификатори. Изразите во форма <#something> од овој изворен код покажуваат кон соодветните дефиниции кои може да се пристапат преку линкот <https://www.w3.org/TR/sparql11-query/#something>. Елементот *permission* служи за опис на тоа дали полисата дозволува или забранува одредена операција (елементот *operation*) и го претставува елементот ϵ од формалната Дефиниција 15. Елементот *operation* се состои од еден од операциските клучни зборови (означени со *opKeyword*), кои ја означуваат операцијата o од Дефини-

```
policy:=permission operation selection priority datasets?;
permission:=('ALLOW' | 'DENY');
operation:=dataOp | manageOp;
selection:=where solutionModifier?
priority:='PRIORITY' decimal;
datasets:='DATASETS' iri+;
dataOp:=opKeyword '{' quad '}';
manageOp:='MANAGE';
where:=<#rWhereClause>;
solutionModifier:=<#rSolutionModifier>;
opKeyword:='READ'| 'INSERT'| 'DELETE'| 'MODIFY';
quad:=varOrIri varOrIri varOrIriOrLiteral varOrIri;
varOrIri:=var| iri;
varOrIriOrLiteral:=( var | iri | literal);
var:=<#rVar>;
iri:=<#riri>;
literal:=<#rRDFLiteral>;
decimal:=<#rDECIMAL>;
```

Изворен код 3.1: Дефиниција на јазикот заполисите со логика од прв ред преку prologue синтакса

ција 15, опционално проследени со шаблон за четворки *quad* кој ја претставува проекцијата на променливите q. Иако нашиот формализам експлицитно не поддржува *MODIFY* операции, оваа опција е додадена за да овозможиме полесно управување со полисите. Сепак, *MODIFY* полисите формално се претставуваат како посебни *INSERT* и *DELETE* полиси.

Нашиот јазик за полиси ја користи предноста на експресивноста на SPARQL јазикот за селекција на податоци, кој се користи за да се специфицираат податоците заштитени со полисата $\mathbb{D}^{(\mathbb{I},p)}$ од Дефиниција 15. Делот *selection* од јазикот за полисите е валиден SPARQL елемент кој ги екстрахира мапирања на променливите и одговара на функцијата $\sigma(\varphi_i, \mathbb{I}, \varphi_d, \mathbb{D})$. Предикатната функција φ_i се специфицира како SPARQL *GRAPH* елемент[7], додека φ_d е специфициран како GroupGraphPattern[8] од SPARQL јазикот. И двата елементи се користат за извлекување на мапирање на променливи: првиот од имплицитно управуваниот именуван граф кој ја претставува Намерата $\mathbb{I}$, додека вториот, од чуваното податочно множество кон кое е извршена барањето $\mathbb{D}$. Елементот *quad* ги дефинира четирите променливи од интерес од решенијата добиени со функцијата σ кои се користат за формирање на четворките од дозволените податоци. LDA платформата ги користи овие решенија за изградба на множество од четворки, кои ќе ги претставуваат заштитените податочни четворки $\mathbb{D}^{(\mathbb{I},p)}$. На овој начин, администраторите можат да наведат произволен ресурс, тројка, четворка или граф кој

[7] https://www.w3.org/TR/rdf-sparql-query/#GGraphGraphPattern
[8] https://www.w3.org/TR/rdf-sparql-query/#GGroupGraphPattern

треба да биде заштитен и да го тестират изборот над нивните чувани податоци $\mathbb{D}$.

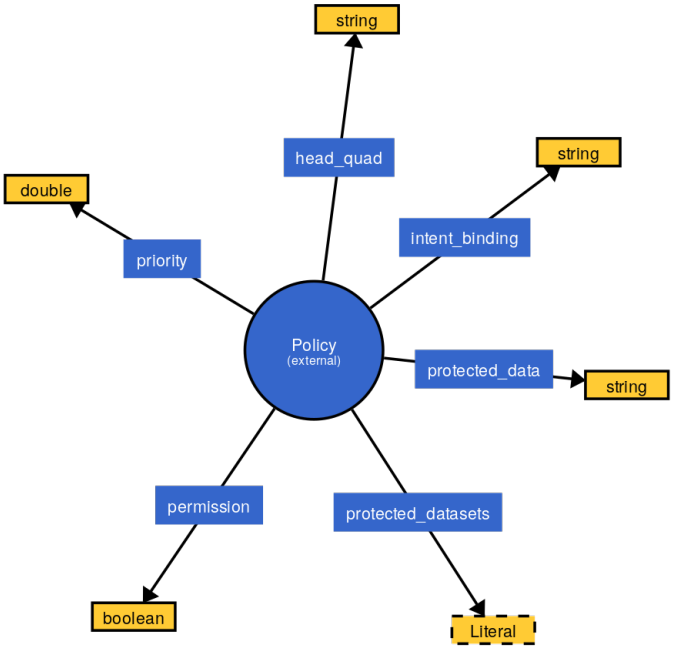
Елементот *priority* се користи за разрешување на конфликти. Користиме разрешување на конфликти со користење на приоритети затоа што овој начин е најблиску до начинот на кој луѓето ги разрешуваат конфликтите во барањата. Луѓето при разрешувањето на конфликтите на барањата, или додаваат друга мета-полиса која има повисок приоритет во споредба со другите, или само да дефинираат дека еден од условите е поважен од останатите. Двата случаи може да се моделираат со приоритетите на полисите, каде што полисите со повисок приоритет ги заменуваат резултатите на оние што се со понизок приоритет.

Една од клучните карактеристики на семантичките податоци се нивната интероперабилност и повторна употреба. Со цел да го следиме овој тренд, нашиот јазик за полиси е способен за дефинирање на полиси кои можат да ги заштитат множествата на податоци преку елементот *datasets*. Оваа функција ја поттикнува дистрибуцијата на податоци и овозможува дефинирање на полиси кои ќе ги заштитат овие податоци.

Онтологија на јазикот за полиси

Во рамките на платформата е дефинирана и онтологија за опис на полисите кој се базира на RDF и SPARQL стандардите од W3C. Секоја полиса е инстанца на класата *p:Policy*. Притоа во онтологијата се претпоставува дека истите имиња на променливи искористени во различните својства на полисата се однесуваат на една иста работа, т.е. променливите имаат опсег дефиниран од полисата. Ова е затоа што модулот за извршување на полисите ги комбинира вредностите на својствата во процесот на трансформација на барањето со вредностите на својствата *p:intent_ binding*, *p:protected_ data* и *p:head_ quad*. Следниве својства треба да се конфигурираат за полисите:

- *p:permission* дефинира дали полисата дозволува или забранува пристап до заштитените податоци. Го претставува елементот ϵ од Дефиниција 15.
- *p:operation* дефинира за кои операции o важи полисата.
- *p:head_ quad* е шаблонот за четворки q кој дефинира кои четворки ќе бидат заштитени.
- *p:intent_ binding* својството ја содржи дефиницијата на предикатната функција φ_i, која означува за кои Намери е применлива полисата. Вредноста на ова својство треба да е валиден SPARQL WHERE елемент, односно содржината на *GRAPH <http://intent>* елементот од јазикот за полисите.
- *p:protected_ data* својството ја содржи дефиницијата на предикатната функција φ_d која означува кои податоци се заштитуваат. Вредноста на ова својство треба да е валиден SPARQL WHERE елемент, односно содржината



Слика 3-6: Онтологија за LDA полисите

на *WHERE* елементот без *GRAPH <http://intent>* елементот од јазикот за полисите.

- *p:protected_ datasets* ги дефинира податочните множества за кои полисата е применлива.
- *p:priority* го претставува приоритетот на полисата ρ кој се користи за разрешување на конфликтите. Типот на податоци на вредностите на ова својство е *xsd:double*.

Модулот за извршување на авторизацијата на платформата ја користи објектна репрезентација на полисите која одговара на опишаната онтологија. Дополнително, оваа онтологија (дополнително визуелизирана на Слика 3-6) се користи за зачувување на полисите во системот од страна на *OntologyPolicyProvider* имплементацијата, каде се ограничува кој ќе има привилегии да ги управува а кој не, преку предефинирани полиси со највисок приоритет. *OntologyPolicyProvider* имплементацијата притоа забранува внес на полиси со повисок приоритет од предефинираните. Предефинираните полиси во овој случај се вчитуваат од конфигурабилна локација и максималниот приоритет се заклучува од проследената конфигурација.

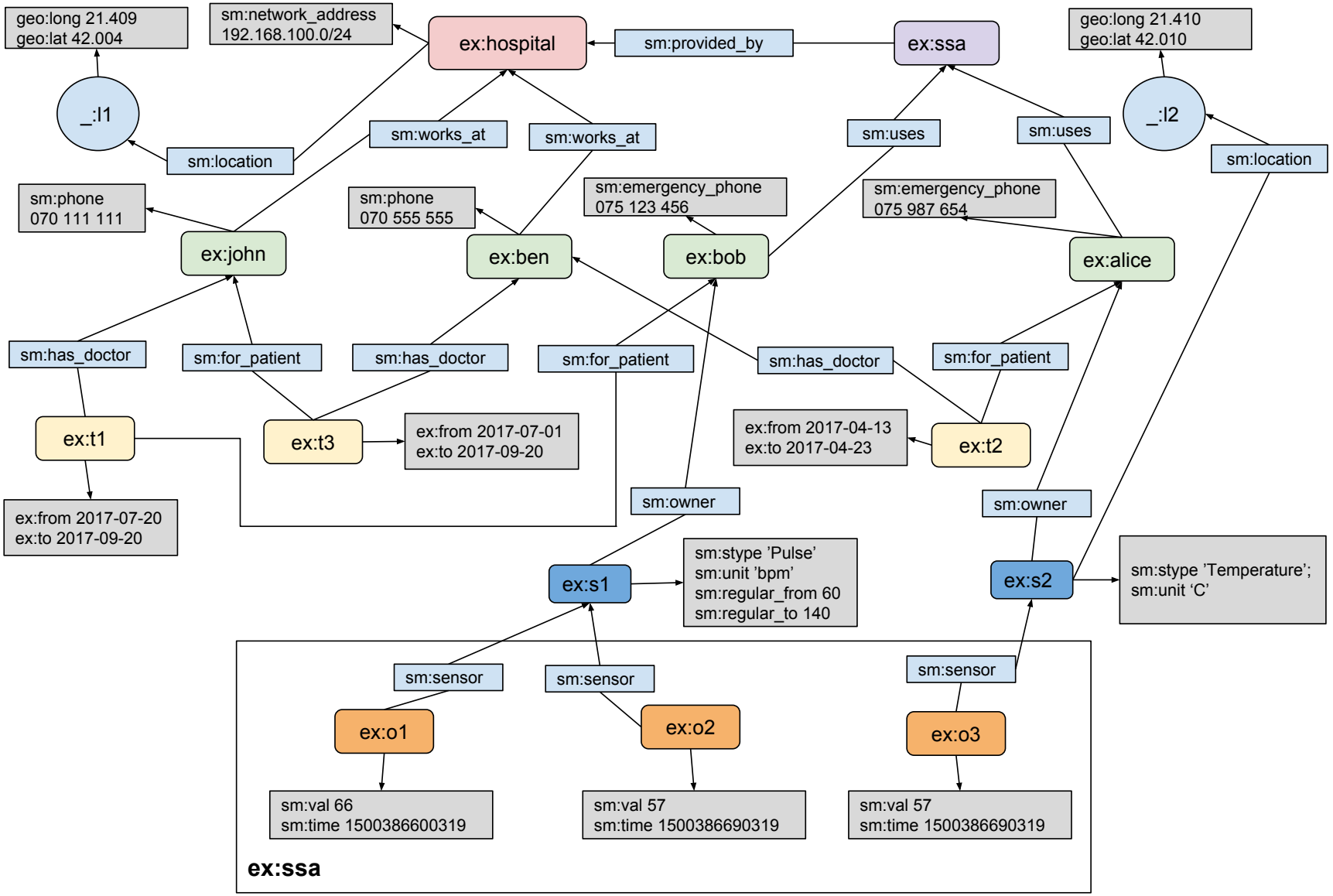
3.2.1 Стандардизирање на полисите

Јазикот за полисите дозволува користење на произволни имиња на променливите, кои во различни полиси може да се однесуваат на исти концепти. За LDA платформата да се овозможи комбинација на повеќето полиси, детекција на конфликти и проценка на заштитените податоци се врши стандардизација на полисите. Во овој процес, се поистоветуваат имињата на променливите кои се однесуваат на исти ресурси.

- Стандардизацијата започнува со заменување на имињата на сите променливи со генерирани и глобално уникантни имиња. На овој начин се гарантира дека променливи со исто име од различни полиси, кои се однесуваат на различни ресурси нема да бидат поврзани заради именувањето.
- Наредниот чекор е стандардизирање на шаблоните за четворки од сите полиси *quad*, каде се обезбедува дека елементите од четворката која се заштитува секогаш ќе се референцираат со уникатни променливи. Во овој процес се користи шаблонот за четворки $q_0 = (?s, ?p, ?o, ?g)$ за да ги унифицира проекциите q на сите полиси. Во овој процес, секое појавување на оригиналното име на променливите од проекцијата се заменува со новото име насекаде низ таа полиса, односно во телата на предикатните функции φ_i и φ_d. Новите имиња на променливите во проекцијата одговараат на имињата искористени во q_0.
- Потоа се стандардизирани променливите во шаблоните за тројки кои се користат во предикатната функција за специфицирање на намерите φ_i. Променливите кои се појавуваат на иста позиција во шаблоните за четворки кои се „еквивалентни“ се стандардизирани со користење на променливата од полисата со понизок приоритет. Еквивалентни шаблони за четворки кај кои единствена можна разлика е во имињата на променливите на соодветните позиции. Овој чекор се извршува за секој пар полиси, процесирајќи ги полисите во растечки редослед.

3.3 Приказ на функционирањето на LDA платформата преку пример

Со цел да ги објасниме сите детали во врска со имплементацијата на LDA платформата и нејзините можности, ние дизајниравме сценарио кое ги покрива сите важни аспекти на функционирањето на LDA платформата. Сликата 3-7 покажува пример заштитени податоци $\mathbb{D}$ што го претставуваат популарното мешање на податоци од паметни уреди со социјални и корпоративни податоци. Сликата поконкретно прикажува мерења од сензори кои се поврзани со личните податоци на корисниците и со податоците на системот за управување со пациентите.

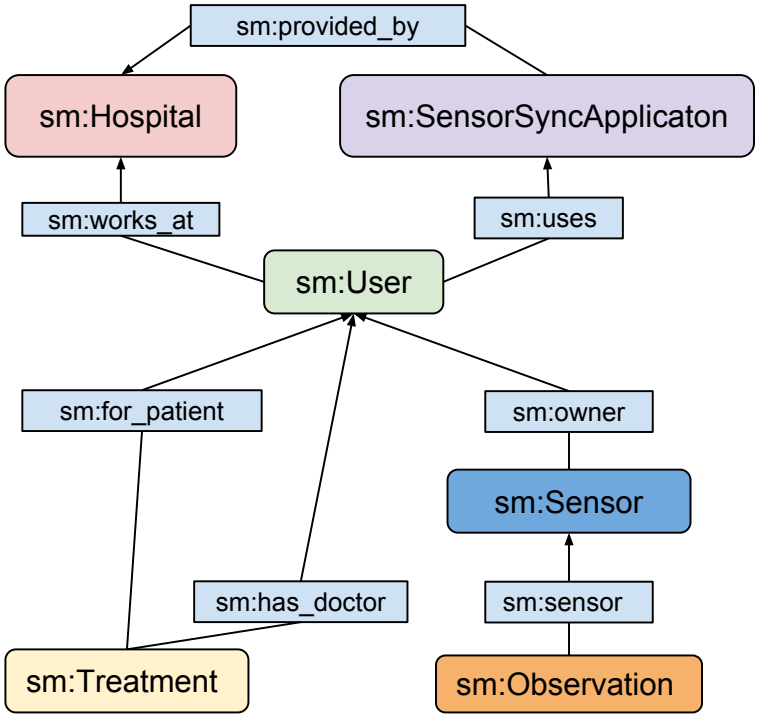


Слика 3-7: Пример податочно множество

Табела 3.1: Барања за заштита на податоци

Барање	[45]	[75]	[88]	[61]	[34]	[29]	[21]	[1]	[51]	[68]	LDA платформата
A1: Својствата на болниците и апликациите се јавни и достапни за секој.	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
A2: Мобилните и итните телефонски броеви на корисниците се приватни.		✓		✓	✓	✓		✓	✓	✓	✓
A3: Просечните дневни мерења од сензорите кои не се чувствителни (не се за здравје) се јавни.											✓
U1: Корисниците можат да пристапат до сопствените својства и директните својства на ресурсите поврзани со нив.			✓	✓		✓		✓		✓	✓
U2: Корисниците можат да ги управуваат податоците за своето име, телефон и итен телефон или е-пошта.			✓	✓		✓		✓		✓	✓
P1: Пациентите можат да пристапат до сè за лекарите од *ex:hospital*.		✓	✓	✓		✓		✓	✓	✓	✓
D1: Лекарите можат да ги менуваат мерењата на нивните пациенти од мрежата на болницата во текот на работното време.											✓
D2: Лекарите не можат да ги менуваат мерењата надвор од временски период на третманот.											✓
TS1: Техничкиот персонал може да управува со апликации само за нивната болница.									✓		✓
SU1: Корисникот ex:ben може да генерира извештаи.	✓		✓	✓							✓
EM1: Лекарот може да пристапи до телефонскиот број на своите пациенти за време на абнормални мерења.			✓	✓				✓			✓

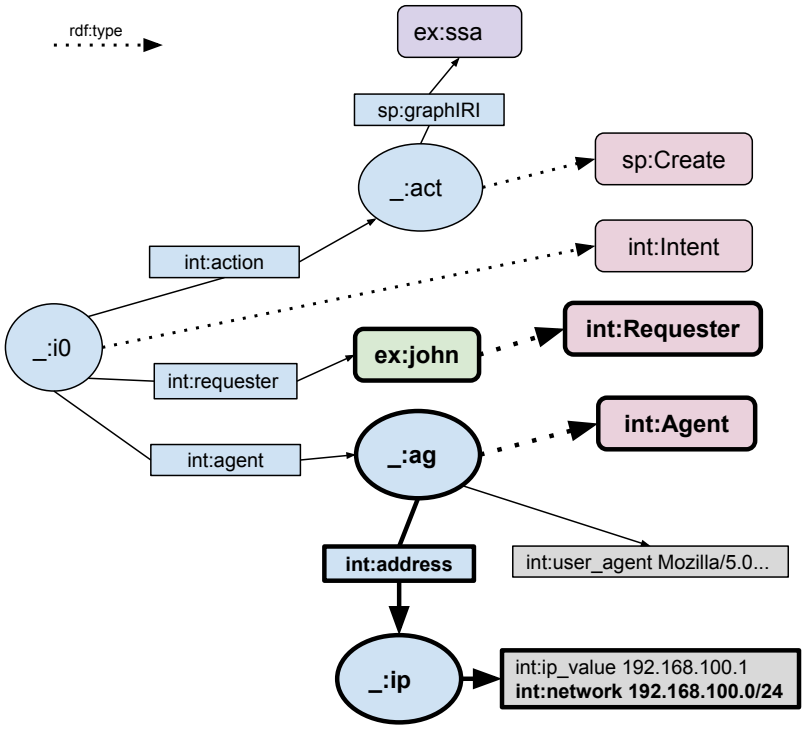
Сликата 3-8 ја покажува онтологијата која ги моделира класите и својствата на податоците во Слика 3-7. Нашиот пример опфаќа една болница, *ex:hospital*, со *sm:network_address "194.168.100.0/24"*, лоцирана на локацијата *_:l1*, опишана со geo-names онтологијата[9]. Иако, онтологијата на Слика 3-8 покажува дека болниците можат да обезбедат повеќе апликации за синхронизација на сензорите, *ex:hospital* обезбедува само апликација *ex:ssa*. Апликацијата *ex:ssa* се поврзува со сензорите *ex:s1* и *ex:s2* во сопственост на корисниците *ex:bob* и *ex:alice*, соодветно и ги синхронизира нивните податоци. Овие корисници се пациенти со третмани *ex:t1* и *ex:t2*, спроведени од лекарите *ex:john* и *ex:ben*, соодветно. Третманот *ex:t3* покажува дека *ex:john* е пациент на *ex:ben*. Фактот дека *ex:john* е лекар и пациент во зависност од контекстот, овозможува моделирање на различни нивоа на поделба на службата, како што се дискутира во [33]. Локациите *_:l1* и *_:l2* и мрежната IP адреса на болницата овозможуваат дефинирање на гео-просторни контекстуални барања, додека времетраењето на третманите обезбедува временски контекстуални барања.



Слика 3-8: Онтологија за опис на податоците од примерот

Сликата 3-9 покажува пример Намера *_:i*, каде барателот *ex:john* се обидува да го создаде именуваниот граф *ex:ssa* користејќи го софтверскиот агент *_:ag* преку планираната акција *_:act*. Во овој пример Spin онтологијата[10] ја моделира синтаксата SPARQL и се користи за претставување на акцијата. Со користење на оваа онтологија се опишува дека планираното дејство е SPARQL CREATE

[9] http://www.w3.org/2003/01/geo/wgs84_pos#
[10] http://spinrdf.org/sp#



Слика 3-9: Пример за содржина на намера

(*sp:Create*) прашање, а својството *sp:graphIRI* кажува кој именуван граф треба да се креира. Префиксот *int:* ги моделира класите и својствата што се користат за да ја покажат очекуваната структура на податоци за *Намерата*. Својството *int:requester* и класата *int:Requester* се користат да дефинираат кој е барателот. Ресурсот *_:ag* го дефинира софтверскиот агент кој се користи за интеракција со LDA платформата.

Претстава на барањата во јазикот за полиси

Во табелата 3.1 се прикажани барањата за заштита на податоците кои треба да ги исполни системот за авторизација. Овие барања се избрани за да ги опфатат сите предизвици во однос на флексибилноста на јазикот за полисите. Јазикот за полиси кој се користи во LDA платформата ги покрива сите овие барања и овозможува дефинирање за полиси за секое од нив. Во продолжение се прикажани полисите за секое од барањата, како и податоците кои се заштитени од соодветната полиса.

A1: Својствата на болниците и апликациите се јавни и достапни за секој.

```
ALLOW READ { ?s ?p ?o ?g }
WHERE {
   ?s ?p ?o .
   {
     {   ?s a sm:Hospital }
     UNION { ?s a sm:SensorSyncApplicaton }
   }
} PRIORITY 1
```

?s	?p	?o	?g
ex:hospital	rdf:type	sm:Hospital	
ex:hospital	sm:network_address	192.168.100.0/24	
ex:hospital	sm:location	_:b0	
ex:ssa	rdf:type	sm:SensorSyncApplicaton	
ex:ssa	sm:provided_by	ex:hospital	

A2: Мобилните и итните телефонски броеви на корисниците се приватни.

```
DENY READ { ?s ?p ?o ?g }
WHERE {
   ?s a sm:User; ?p ?o.
   FILTER (?p=sm:phone || ?p=sm:emergency_phone)
} PRIORITY 3
```

?s	?p	?o	?g
ex:john	sm:phone	070 111 111	
ex:ben	sm:phone	075 555 555	
ex:bob	sm:emergency_phone	075 123 456	
ex:alice	sm:emergency_phone	075 987 654	

A3: Просечните дневни мерења од сензорите кои не се чувствителни (не се за здравје) се јавни.

```
ALLOW READ {?s ?p ?o ?g}
WHERE {
   BIND(sm:avg_value as ?p)
   ?s sm:owner ?v3.
   { SELECT (AVG(?v1) as ?o) ?s
      WHERE {
         GRAPH ?g {
            ?v2 sm:sensor ?s; sm:val ?v1; sm:time ?v4
         }
      } group by ?s ceil(?v4/86400000)
   }
} PRIORITY 9
```

?s	?p	?o	?g
ex:s2	sm:avg_value	28	

U1: Корисниците можат да пристапат до сопствените својства и директните својства на ресурсите поврзани со нив.

```
ALLOW READ { ?s ?p ?o ?g }
WHERE {
   GRAPH <http://intent> { ?r rdf:type int:Requester }
   ?s ?p ?o.
   {  { ?r  rdf:type sm:User; ?v6 ?s }
      UNION {
         ?r  rdf:type sm:User.
         ?s  ?v6 ?r  }
   }
} PRIORITY 4
```

?s	?p	?o	?g	?r
ex:alice	sm:uses	ex:ssa		ex:alice
ex:alice	rdf:type	sm:User		ex:alice
ex:alice	sm:emergency_phone	075 987 654		ex:alice
ex:ssa	sm:provided_by	ex:hospital		ex:alice
ex:ssa	rdf:type	sm:SensorSyncApplicaton		ex:alice
ex:t2	sm:for_patient	ex:alice		ex:alice
ex:t2	sm:has_doctor	ex:ben		ex:alice
ex:t2	rdf:type	sm:Treatment		ex:alice
ex:t2	sm:from	2017-04-13		ex:alice
ex:t2	sm:to	2017-04-23		ex:alice
ex:ben	sm:works_at	ex:hospital		ex:ben
ex:ben	rdf:type	sm:User		ex:ben
ex:ben	sm:phone	075 555 555		ex:ben
ex:hospital	sm:location	_:b0		ex:ben
ex:hospital	rdf:type	sm:Hospital		ex:ben
ex:hospital	sm:network_address	192.168.100.0/24		ex:ben
ex:t2	sm:for_patient	ex:alice		ex:ben
ex:t2	sm:has_doctor	ex:ben		ex:ben
ex:t2	rdf:type	sm:Treatment		ex:ben
ex:t2	sm:from	2017-04-13		ex:ben
ex:t2	sm:to	2017-04-23		ex:ben
ex:t3	sm:has_doctor	ex:ben		ex:ben
ex:t3	sm:for_patient	ex:john		ex:ben
ex:t3	rdf:type	sm:Treatment		ex:ben
ex:t3	sm:from	2017-07-01		ex:ben
ex:t3	sm:to	2017-09-20		ex:ben
ex:bob	sm:uses	ex:ssa		ex:bob
ex:bob	rdf:type	sm:User		ex:bob
ex:bob	sm:emergency_phone	075 123 456		ex:bob
ex:s1	sm:owner	ex:bob		ex:bob
ex:s1	rdf:type	sm:HealthSensor		ex:bob
ex:s1	sm:regular_to	140		ex:bob
ex:s1	sm:regular_from	60		ex:bob
ex:s1	sm:stype	Pulse		ex:bob
ex:s1	sm:unit	bpm		ex:bob
ex:ssa	sm:provided_by	ex:hospital		ex:bob
ex:ssa	rdf:type	sm:SensorSyncApplicaton		ex:bob
ex:t1	sm:for_patient	ex:bob		ex:bob
ex:t1	sm:has_doctor	ex:john		ex:bob
ex:t1	rdf:type	sm:Treatment		ex:bob
ex:t1	sm:from	2017-07-20		ex:bob
ex:t1	sm:to	2017-09-20		ex:bob
ex:hospital	sm:location	_:b0		ex:john
ex:hospital	rdf:type	sm:Hospital		ex:john
ex:hospital	sm:network_address	192.168.100.0/24		ex:john
ex:john	sm:works_at	ex:hospital		ex:john
ex:john	rdf:type	sm:User		ex:john
ex:john	sm:phone	070 111 111		ex:john
ex:s2	sm:location	_:b1		ex:john
ex:s2	sm:owner	ex:john		ex:john
ex:s2	rdf:type	sm:Sensor		ex:john
ex:s2	sm:unit	C		ex:john
ex:s2	sm:stype	Temperature		ex:john
ex:t1	sm:for_patient	ex:bob		ex:john
ex:t1	sm:has_doctor	ex:john		ex:john
ex:t1	rdf:type	sm:Treatment		ex:john
ex:t1	sm:from	2017-07-20		ex:john
ex:t1	sm:to	2017-09-20		ex:john
ex:t3	sm:has_doctor	ex:ben		ex:john
ex:t3	sm:for_patient	ex:john		ex:john
ex:t3	rdf:type	sm:Treatment		ex:john
ex:t3	sm:from	2017-07-01		ex:john
ex:t3	sm:to	2017-09-20		ex:john

U2: Корисниците можат да ги управуваат податоците за своето име, телефон и итен телефон или е-пошта.

```
ALLOW MODIFY {?s ?p ?o ?g}
WHERE {
    GRAPH <http://intent> { ?r a int:Requester }
    BIND (?r as ?s)
    ?s ?p ?o
    FILTER (?p=sm:name || ?p=sm:phone ||
        ?p=sm:email || ?p=sm:emergency_phone)
}
PRIORITY 5
```

?s	?p	?o	?g	?r
ex:alice	sm:emergency_phone	075 987 654		ex:alice
ex:ben	sm:phone	075 555 555		ex:ben
ex:bob	sm:emergency_phone	075 123 456		ex:bob
ex:john	sm:phone	070 111 111		ex:john

P1: Пациентите можат да пристапат до сè за лекарите од *ex:hospital*.

```
ALLOW READ {?s ?p ?o ?g}
WHERE {
    GRAPH <http://intent> { ?r a int:Requester }
    ?v7 sm:for_patient ?r.
    ?v8 sm:has_doctor ?s .
    ?s ?p ?o
}
PRIORITY 2
```

?s	?p	?o	?g	?r
ex:ben	sm:works_at	ex:hospital		ex:alice
ex:ben	sm:works_at	ex:hospital		ex:alice
ex:ben	rdf:type	sm:User		ex:alice
ex:ben	rdf:type	sm:User		ex:alice
ex:ben	sm:phone	075 555 555		ex:alice
ex:ben	sm:phone	075 555 555		ex:alice
ex:john	sm:works_at	ex:hospital		ex:alice
ex:john	rdf:type	sm:User		ex:alice
ex:john	sm:phone	070 111 111		ex:alice
ex:ben	sm:works_at	ex:hospital		ex:bob
ex:ben	sm:works_at	ex:hospital		ex:bob
ex:ben	rdf:type	sm:User		ex:bob
ex:ben	rdf:type	sm:User		ex:bob
ex:ben	sm:phone	075 555 555		ex:bob
ex:ben	sm:phone	075 555 555		ex:bob
ex:john	sm:works_at	ex:hospital		ex:bob
ex:john	rdf:type	sm:User		ex:bob
ex:john	sm:phone	070 111 111		ex:bob
ex:ben	sm:works_at	ex:hospital		ex:john
ex:ben	sm:works_at	ex:hospital		ex:john
ex:ben	rdf:type	sm:User		ex:john
ex:ben	rdf:type	sm:User		ex:john
ex:ben	sm:phone	075 555 555		ex:john
ex:ben	sm:phone	075 555 555		ex:john
ex:john	sm:works_at	ex:hospital		ex:john
ex:john	rdf:type	sm:User		ex:john
ex:john	sm:phone	070 111 111		ex:john

D1: Лекарите можат да ги менуваат мерењата на нивните пациенти од мрежата на болницата во текот на работното време.

```
ALLOW MODIFY { ?s ?p ?o ?g }
WHERE {
   GRAPH <http://intent> {
      ?r a int:Requester.
      ?ag a int:Agent; int:address ?ip.
      ?ip int:network ?n
   }
   ?r sm:works_at ?v8.
   ?v8 sm:network_address ?n.
   ?v9 sm:has_doctor ?r; sm:for_patient ?v11.
   ?v10 sm:owner ?v11.
   GRAPH ?g { ?s sm:sensor ?v10; ?p ?o }
} PRIORITY 7
```

?s	?p	?o	?g	?r	?n
ex:o3	sm:sensor	ex:s2	ex:ssa	ex:ben	192.168.100.0/24
ex:o3	rdf:type	sm:Observation	ex:ssa	ex:ben	192.168.100.0/24
ex:o3	sm:time	1500386690319	ex:ssa	ex:ben	192.168.100.0/24
ex:o3	sm:val	28	ex:ssa	ex:ben	192.168.100.0/24
ex:o1	sm:sensor	ex:s1	ex:ssa	ex:john	192.168.100.0/24
ex:o1	rdf:type	sm:Observation	ex:ssa	ex:john	192.168.100.0/24
ex:o1	sm:time	1500386600319	ex:ssa	ex:john	192.168.100.0/24
ex:o1	sm:val	66	ex:ssa	ex:john	192.168.100.0/24
ex:o2	sm:sensor	ex:s1	ex:ssa	ex:john	192.168.100.0/24
ex:o2	rdf:type	sm:Observation	ex:ssa	ex:john	192.168.100.0/24
ex:o2	sm:time	1500386690319	ex:ssa	ex:john	192.168.100.0/24
ex:o2	sm:val	57	ex:ssa	ex:john	192.168.100.0/24

D2: Лекарите не можат да ги менуваат мерењата надвор од временски период на третманот.

```
DENY MODIFY { ?s ?p ?o ?g }
WHERE {
   GRAPH <http://intent> { ?r a int:Requester }
   ?v19 sm:has_doctor ?r; sm:for_patient ?v20;
    sm:from ?v22; sm:to ?v23   .
   ?v21 sm:owner ?v20.
   GRAPH ?g {
      ?s sm:sensor ?v21.
      ?s ?p ?o
   }
   BIND (xsd:date(now()) as ?v24)
   FILTER (xsd:date(?v22) > ?v24 || xsd:date(?v23)<?v24)
} PRIORITY 8
```

Coverage per minimal intent[a]:

?s	?p	?o	?g	?r	?n
ex:o1	sm:sensor	ex:s1	ex:ssa	ex:john	192.168.100.0/24
ex:o1	rdf:type	sm:Observation	ex:ssa	ex:john	192.168.100.0/24
ex:o1	sm:time	1500386600319	ex:ssa	ex:john	192.168.100.0/24
ex:o1	sm:val	66	ex:ssa	ex:john	192.168.100.0/24
ex:o2	sm:sensor	ex:s1	ex:ssa	ex:john	192.168.100.0/24
ex:o2	rdf:type	sm:Observation	ex:ssa	ex:john	192.168.100.0/24
ex:o2	sm:time	1500386690319	ex:ssa	ex:john	192.168.100.0/24
ex:o2	sm:val	57	ex:ssa	ex:john	192.168.100.0/24

[a]The query is executed on 2017-08-04

TS1: Техничкиот персонал може да управува со апликации само за нивната болница.

```
ALLOW MANAGE
WHERE {
    GRAPH <http://intent> {
        ?r a int:Requester.
        {
            {
                ?a rdf:type sp:Create
            } UNION {
                ?a rdf:type sp:Drop
            }
        }
        ?a sp:graphIRI ?sp
    }
    ?sp a sm:SensorSyncApplication; sm:provided_by ?h.
    ?r sm:works_at ?h; sm:dtype "TechnicalStaff"
}
PRIORITY 10
```

SU1: Корисникот ex:ben може да генерира извештаи.

```
ALLOW MANAGE
WHERE {
    GRAPH <http://intent> {
        ?i int:requester ex:john; int:action ?a.
        ?a a ex:GenerateReport.
    }
} PRIORITY 12
```

EM1: Лекарот може да пристапи до телефонскиот број на своите пациенти за време на абнормални мерења.

```
ALLOW READ {?s ?p ?o ?g}
WHERE {
    GRAPH <http://intent> { ?r a int:Requester }
    BIND(sm:emergency_phone as ?p)
    ?v12 sm:for_patient ?s; sm:has_doctor ?r .
    ?v13 sm:owner ?s; sm:regular_from ?v14;
     sm:regular_to ?v15 .
    GRAPH ?v16 {
        ?v17 sm:sensor ?v13; sm:val ?v18
    }
    ?s ?p ?o
    FILTER (?v18 > ?v15 || ?v18 < ?v14)
}
PRIORITY 12
```

?s	?p	?o	?g	?r
ex:bob	sm:emergency_phone	075 123 456		ex:john

3.3.1 Заштитени податоци од полисата $\mathbb{D}^{(\mathbb{I},p)}$

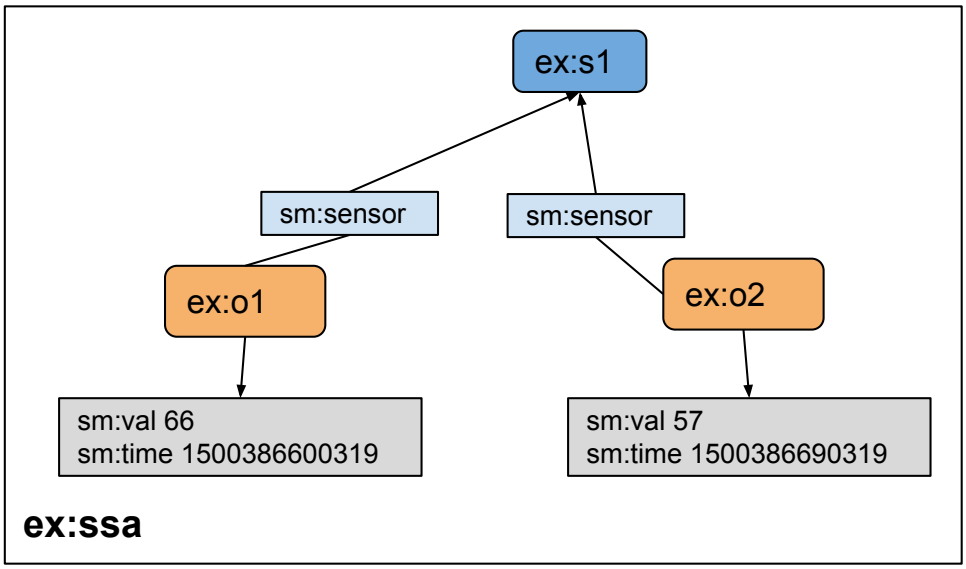
```
1 ALLOW READ {
2 ?s ?p ?o ?g
3 } WHERE {
4  GRAPH <http://intent> {
5   ?v1 a int:Requester.
6   ?v2 a int:Agent.
7   ?v2 int:address ?v3.
8   ?v3 int:network ?v4
9  }
10 GRAPH ?g {
11  ?s a sm:Observation.
12  ?s sm:sensor ?v5.
13  ?s ?p ?o
14 }
15 ?v1 sm:works_at ?v6.
16 ?v5 sm:owner ?v7.
17 ?v8 sm:for_patient ?v7.
18 ?v8 sm:has_doctor ?v1.
19 ?v6 sm:network_address ?v4
20 }
21 PRIORITY 1
```

Изворен код 3.2: E1: Пример полиса

Изворниот код 3.2 покажува пример полиса која им овозможува на лекарите да ги пристапат набљудувањата за нивните пациенти само од мрежата на болницата во која се вработени. Во овој пример, телото на елементот *GRAPH <http://intent>* (линии 5-8) одговара на предикатната функција φ_i која ја дефинира применливоста на полисата за намерата. Во нашиот случај, користиме посебен именуван граф за одржување на податоците на намерата. Полисата го користи елементот *GRAPH <http://intent>* (линија 4) за да потврди дека намерата ги исполнува условите за активација. Полисата се активира и се разгледува во процесот на извршување кога функцијата за применливост за намерата φ_i ги извлекува потребните ресурси од намерата. Функцијата за избор на податоци φ_d е претставена со содржината на линиите 10-19. Ние го користиме елементот SPARQL *WHERE* (линии 4-19), бидејќи неговата обработка одговара на функцијата за евалуација на променливите $\sigma(\varphi_i, \mathbb{I}, \varphi_d, \mathbb{D})$ и враќа мапирања за променливите со соодветните евалуирани вредности. Овие резултати се мапирања на променливи кои се совпаѓаат со двете предикатни функции на полисата, но тие не дефинираат што треба да биде заштитено со полисата. Функцијата за проекција π се користи за да се креираат шаблоните за четворки со користење на променливите од q во однос на мапирањата на променливите вратени со σ. *SPARQL SELECT* формата на прашања ги проектира саканите променливи и е соодветна имплементација на функцијата за проекција π. Шаблонот за четворки

?r ?p ?o ?app (линија 2) ги конструира заштитените податоци $\mathbb{D}^{(\mathbb{I},p)}$.

Да претпоставиме дека податоците на Сликата 3-7 се заштитена само со полисата од изворниот код 3.2 и *ex:john* поднесува намера од Слика 3-9 за извршување на акцијата *sp:Create* што бара операција за читање да биде извршена пред создавањето на именуваниот граф. Ова сценарио е валидно бидејќи секоја акција може да бара повеќе операции да бидат извршени за нејзино завршување. Предикатната функцијата за применливоста на намерата φ_i се евалуира според податоците во имениот граф за намерата <http://intent> и се создава посебно мапирање за секоја комбинација на ресурси што се совпаѓа со сите шаблоните за тројки од φ_i(задебелениот дел од Слика 3-9). Во нашиот случај, $vars(\varphi_i)$ =*(?doc, ?ag, ?ip, ?n)* и има само едно мапирање на променливите што одговара на сите шаблони за тројки од φ_i. Ова мапирање за променливите од $vars(\varphi_i)$ е *(ex:john, _:ag, _:ip, "192.168.100.0/24")*. Слично на тоа, предикатната функција φ_d се користи за да се добијат мапирањата на променливите прикажани во табелата 3.2 (без колоните *?ag* и *? ip*) од чуваните податоци $\mathbb{D}$. Конечните променливи врски се добиваат со спојување на двата резултати кои имаат иста вредност за променливите кои се заеднички и во φ_i и φ_d, т.е., *?doc* и *?n*. Корисникот *ex:john* е единствениот лекар кој има пациент со набљудувања (за пациентот *ex:bob*) и работи во болницата *ex:hospital* со мрежна IP адреса *"192.168.100.0/24"*, што е причината зошто сите мапирања на променливите од φ_d се во конечниот резултат во Табела 3.2. Овие поврзувања ги содржат ресурсите, литералите и анонимните јазли кои се совпаѓаат со сите шаблони за тројки и четворки од *WHERE* елементот на полисата. Потоа проекцијата q се користи за конструирање на заштитените податоци прикажани на сликата 3-10. Првата линија на полисата во изворниот код 3.2 опишува дека операциите за читање ќе бидат дозволени за податоците од сликата 3-10.



Слика 3-10: Заштитени податоци за полисата

Табела 3.2: Мапирање на променливите од селекцијата на заштитените податоци

?doc	?ag	?ip	?n	?s	?pat	?t	?h	?r	?p	?o	?app
ex:john	_:ag	_:ip	192.168.100.0/24	ex:s1	ex:bob	ex:t1	ex:hospital	ex:o1	rdf:type	sm:Observation	ex:ssa
ex:john	_:ag	_:ip	192.168.100.0/24	ex:s1	ex:bob	ex:t1	ex:hospital	ex:o1	sm:sensor	ex:s1	ex:ssa
ex:john	_:ag	_:ip	192.168.100.0/24	ex:s1	ex:bob	ex:t1	ex:hospital	ex:o1	sm:time	1500386600319	ex:ssa
ex:john	_:ag	_:ip	192.168.100.0/24	ex:s1	ex:bob	ex:t1	ex:hospital	ex:o1	sm:val	66	ex:ssa
ex:john	_:ag	_:ip	192.168.100.0/24	ex:s1	ex:bob	ex:t1	ex:hospital	ex:o2	rdf:type	sm:Observation	ex:ssa
ex:john	_:ag	_:ip	192.168.100.0/24	ex:s1	ex:bob	ex:t1	ex:hospital	ex:o2	sm:sensor	ex:s1	ex:ssa
ex:john	_:ag	_:ip	192.168.100.0/24	ex:s1	ex:bob	ex:t1	ex:hospital	ex:o2	sm:time	1500386690319	ex:ssa
ex:john	_:ag	_:ip	192.168.100.0/24	ex:s1	ex:bob	ex:t1	ex:hospital	ex:o2	sm:val	57	ex:ssa

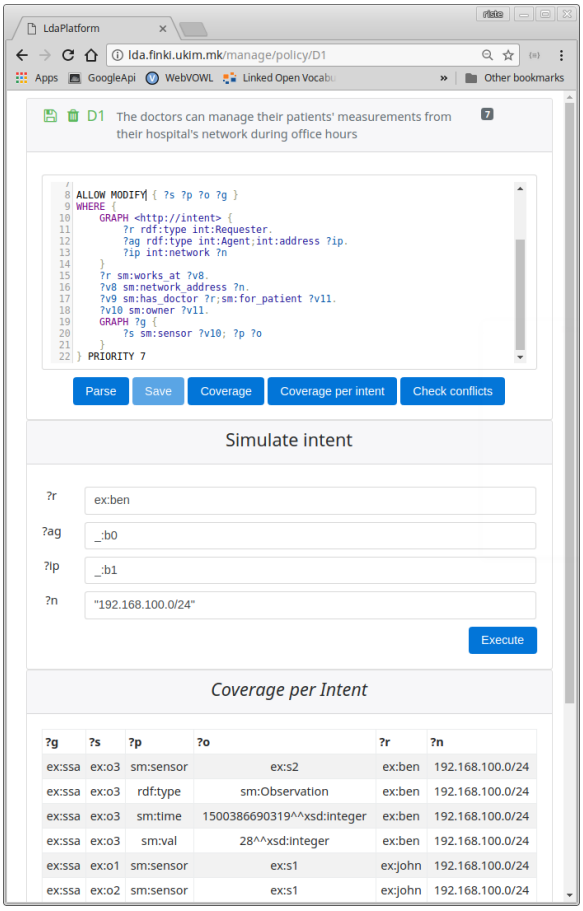
Откако полисите се внесени и зачувани, *модулот за управување со полиси* (Слика 3-11) врши нивна стандардизација, како што е опишано во §3.2.1. Копчето "Parse"ја анализира полисата, како што е објаснето во §4.1.1. Дополнително, за време на оваа акција, *модулот за управување со полиси* ги извлекува променливите од предикатната функција за применливост на намерата $vars(\varphi_i)$ и ја гради формата "Simulate intent"која може да се користи за симулирање на заштитените податоци на полисата за селектираните мапирањата за минималните намери. Во оваа форма се внесуваат само вредностите за мапирањата за минималните намери бидејќи единствено овие променливи од намерата влијаат на заштитените податоци. *Модулот за управување со полисите* табеларно ги прикажува резултатите за симулираната намера.

3.4 Имплементација на модулот за извршување со привремени податочни множества

Ние дефинираме посебен алгоритам за спроведување за секоја операција, каде што операцијата се извршува врз претходно креирани *дозволени податоци* (Дефиниција 11) за намерата [85]. Полисите што се користат за изградба на дозволените податоци се селектирани врз основа на бараната операција и Намерата $\mathbb{I}$. Во LDA платформата, *дозволените податоци* се зачувани во привремено податочно множество. За разлика од пристапите во [29, 34, 61], кои создаваат *"заштитени графови"*, ние создаваме привремено податочно множество за да ја зачуваме структурната организација на податоците без губење на какви било информации. Откако *модулот за извршување* ќе заврши со обработка и филтрирање на податоците, дозволените податоци (доколку ги има) се пренесуваат во модулот за *серијализација на податоците*, кој ги трансформира резултатите во бараната форма врз основа на намерата $\mathbb{I}$. Компонентата *ResultInterpreter* дополнително ги вметнува активираните полиси како објаснување на резултатот. Кога операцијата е одбиена како целина, *модулот за извршување* фрла исклучок, кој се серијализира со компонентата за *ResultInterpreter*.

Како што е прикажано во сликата 3-4, во оваа имплементација на *модулот за извршување* за секоја операција е обезбедена посебна имплементација, која обезбедува авторизација за добиената намера. Во продолжение е детален опис за тоа како се авторизира секоја операција:

- *QueryProcessor* извршува авторизација на операциите за читање. Оваа компонента прво го креира привременото податочно множество, по што го извршува оригиналното прашање над привременото податочно множество и ги проследува резултатите до *ResultInterpreter* компонентата.
- *InsertProcessor* прво ги вметнува податоците кои треба да додадат во оригиналното податочно множество, по што го креира привременото податочно



Слика 3-11: Модул за управување со полисите - анализа на полиса

множество кое ги содржи само податоците кои би смеело да бидат внесени. Потоа, сите претходно вметнати четворки кои не се во привременото податочно множество се отстрануваат. Целиот овој процес се одвива во трансакција управувана од имплементацијата на Dataset интерфејсот од Apache Jena библиотеката.

- *DeleteProcessor* го креира привременото податочно множество и ги отстранува само четворките кои се побарани и се присутни во ова множество. Операцијата се извршува во трансакција.
- *ManageProcessor* ги процесира полисите во опаѓачки редослед во однос на нивниот приоритет. Во овој процес, прво се проверува применливоста на полисата за намерата, со извршување на φ_i предикатната функција над намерата, т.е., со извршување на SPARQL ASK прашање кое во својот *WHERE* елемент го содржи само *GRAPH <http://intent>* блокот. Кај првата пронајдена применлива полиса се извршува φ_d предикатната функција во форма на SPARQL ASK прашање. Доколку ова прашање врати позитивен одговор, се зема дозволата конфигурирана во полисата, а во спротивен случај се применува инверзната дозвола за полисата.

3.4.1 Екстракција на дозволени податоци

Во LDA платформата, *дозволените податоци* се зачувани во привремено податочно множество бидејќи дозволените податоци комбинираат повеќе четворки кои се добиени со комбинација на заштитените податоци од полисите $\mathbb{D}^{(\mathbb{I},p)}$. На овој начин се постигнува извршување на добиените прашања во нивната основна форма, без дополнителна модификација. Во случаите кога се користи *"привремен граф"*[29, 34, 61], барањата што содржат GRAPH елемент треба да се трансформираат за да се добијат дозволените резултати, обично со отстранување на GRAPH елементите во барањето. Оваа модификација на барањето ги губи информациите и често може да произведе неточни резултати.

Алгоритмот 1 ја опишува имплементацијата на функцијата за екстракција на дозволените податоци α од Дефиниција 21 и создавањето на привремените податочни множества DS_{tmp}. Конфигурираните полиси во системот $\mathbb{P}$ се стандардизирани и сортирани според нивниот приоритет во растечки редослед. Бидејќи SPARQL јазикот за пребарување поддржува комбинирање на елементите со користење на *UNION* и *MINUS* операторите, дозволените податоци α се добиваат единствено со извршувањето на прашањето кое е креирано во овој алгоритам.

Според алгоритмот 1, намерата прво се регистрира (линија 1) со вметнување на податоците $\mathbb{I}$ во ново-создадениот именуван граф *<http://intent/{id}>*, каде што *{id}* се генерира секвенцијално за да се овозможи истовремена обработка на повеќе намери. *registerIntent*($\mathbb{I}, \mathbb{D}, \mathbb{P}$) дополнително го заменува <http://intent> текстот со ново-создаденото име на графот во секоја полиса од $\mathbb{P}$ и враќа множе-

Algorithm 1: Креирање на привремено податочно множество

```
   Data: 𝕀, 𝔻, ℙ
   Result: DS_tmp
 1 ℙ′:=registerIntent(𝕀, 𝔻, ℙ);
 2 if p_1.permission = ALLOW then
 3 |  W:= {φ_∅};
 4 else
 5 |  W:={φ_q0};
 6 foreach p in ℙ′ do
 7 |  if p.permission = ALLOW then
 8 |  |  W:=W UNION {p.WHERE} ;
 9 |  else
10 |  |  W:=W MINUS {p.WHERE} ;
11 end
12 varMappings:=execute(W, 𝔻);
13 DS_tmp:=asQuads(varMappings, q_0);
14 return DS_tmp.
```

ство од изменетите полиси $\mathbb{P}'$. Потоа, SPARQL *WHERE* изразот W се иницијализира во зависност од дозволата на полисата со најнизок приоритет $p_1.permission$. Иницијализацијата на α_0 од Дефиниција 11 (линиите 2-5) го користи шаблонот за четворки φ_0 ={ *?s ?p ?o ?g* } , кој се совпаѓа со секоја четворка од чуваните податоци $\mathbb{D}$, и $\varphi_\emptyset$, што е празна предикатна функција која нема променливи и не избира никакви податоци. Потоа, сите полиси од $\mathbb{P}'$ се обработуваат (линиите 6-11). Кога полисата ја дозволува операцијата, содржината на нејзиниот *WHERE* елемент [11] се додава со користење на операцијата *UNION* (линија 8). Во спротивен случај, се користи операцијата *MINUS* (линија 10). Откако сите полиси од $\mathbb{P}'$ се обработени, резултираниот *WHERE* елемент се извршува над чуваните податоци $\mathbb{D}$ за да се извлечат мапирањата на променливите (линија 12). Извлечените мапирања за q_0 се трансформираат во четворки за да се формира резултантното привремено податочно множество DS_{tmp} (линија 13).

Откако ќе се создаде привременото податочно множество, сите операции се извршуваат над него. Бидејќи ова податочно множество ги содржи само дозволените податоци, тоа обезбедува имплицитна безбедност, што значи дека забранува пребарувањата да имаат интеракција со податоците што не им се дозволени.

[11] $p.WHERE$

3.5 Имплементација со препишување на иницијалното прашање

Ова поглавје ја опишува имплементацијата на модулот за извршување на полисите со користење на промена на прашањето за авторизиран пристап [86]. Компонентата *PermissionTransformer* е одговорна за процесот на промена на иницијалното барање. Оваа компонента се обидува да ги заклучи ограничувањата за овластување за секој од шаблоните за четворки во прашањата. Во овој процес, прашањето прво се трансформира со користење на *AlgebraQuad*[12], со што се заменува секој шаблон за тројки со шаблон за четворки, додека блоковите со тројки се трансформираат во блокови со четворки. Потоа, *PermissionTransformer* компонентата ги обработува блоковите со четворки, кои се состојат од множество со шаблони за четворки. Изворниот код 3.3 го прикажува методот *PermissionTransformer.transform* со користење на функционална програмска нотација. Методот $transform$ се повикува за секој од блоковите за четворки (*Quad Blocks*) од прашањето. Притоа за секој блок се иницијализира Q_{filter} во празна алгебра операција која одговара на предикатната функција φ, и се менува преку процесот на посета и го претставува ограничувањето кое ќе биде додадено во блокот откако ќе се посетат сите негови четворки.

Линијата 4 од Изворниот Код 3.3 го трансформира потокот (stream) со полиси во поток со парови $\langle p, m \rangle$, каде што m е мапирање на променливи кое го содржи резултат од извршувањето на прашањето *p:intent_ binding* над податоците од намерата. Парот се враќа бидејќи двата елементи се неопходни во следните чекори за обработка. Во линија 5 паровите кои содржат празни мапи ќе бидат отстранети, а само активираните полиси за намерата остануваат за понатамошна обработка. Потоа, во линијата 6, паровите полиси - мапирање се трансформираат во друг пар кој го заменува мапирањето на променливите со препишана верзија на полисата. На крајот, сите применливи полиси се собираат во листата *applicablePolicies*, која подоцна се користи за препишување на прашањето.

Процедурата *apply* се користи за проширување на својството Q_{filter} од *PermissionTransformer* за секој шаблон со четворки со условите од активираните полиси. Во текот на овој процес, секој шаблон за четворки од блокот е усогласен со *p:head_ quad* од полисата користејќи ја функцијата *mapping*. Оваа функција за мапирање прво проверува дали два шаблони за четворки се совпаѓаат еден со друг[13], и кога ова е случај, се проверува дали нивните соодветни променливи не се од типови кои не се не-компатибилни. Кога ова е случај, променливите од првиот шаблон за четворки се мапираат со соодветните елементи од вториот шаблон за четворки, додека променливите од вториот шаблон за четворки се

[12]Jena ARQ алгебра трансформатор http://bit.ly/2rgvvLw

[13]Два шаблони со четворки се совпаѓаат ако сите нивни елементи се совпаѓаат, што е случај кога барем еден од елементите е променлива или кога се исти.

```
1  transform(QuadBlock) {
2   qbConstr:=induceVariableClasses(QuadBlock,model)
3   applicablePolicies := policies.stream()
4    .map(p → ⟨p, m := executeSelect(p.intent_binding, intent)⟩)
5    .filter(⟨p, m⟩ → m ≠ ∅)
6    .map(⟨p⟩ → p_rw := rewrite(p, m))
7    .collectToList()
8
9   QuadBlock.stream()
10   .foreach(qp → apply(qp, applicablePolicies, qbConstr))
11
12  return filteredBlock(QuadBlock)
13 }
14
15 apply(qp, applicablePolicies, qbConstr) {
16  applicablePolicies.stream()
17   .map(p → ⟨p, m := mapping(p.head_quad, qp, qbConstr, p.varConstr)⟩)
18   .filter(⟨p, m⟩ → m ≠ ∅)
19   .foreach(⟨p, m⟩ → {
20     Q_filter := appendFilterExpression(Q_filter, m)
21     dir := (dir ≠ null ? dir : p.permission)
22     Q_filter := (p.permission = dir ?
23       Q_filter ∪ p.permitted_data :
24       Q_filter \ p.permitted_data)
25   })
26
27   appendFilter(Q_filter)
28   dir:= null
29   Q_filter:=null
30 }
```

Изворен код 3.3: Имплементација на PermissionTransformer

мапираат само на конкретни вредности од првиот т.е., со вредности кои не се променливи.

По филтрирањето во линијата 18, остануваат само полисите кои се релевантни за разгледуваниот шаблон за четворки. Полисите потоа се комбинираат за да се добие елементот кој ќе се додаде при препишувањето на прашањето. Мапирањето m потоа се додава во елементот Q_{filter} во форма:

FILTER (?v1=hq[0] && ?v2=hq[1] && ?v3=hq[2] && ?vg=hq[3])

Во овој израз *hq[i]* го означува i-тиот елемент од својството *p:head_ quad* на полисата. Својството *dir* што се користи во линијата 21 е внатрешно својство на *PermissionTransformer* и се иницира со дозволата на применливата полиса со најнизок приоритет. Сите препишани полиси се комбинираат заедно во операцијата Q_{filter} со користење на SPARQL UNION или MINUS операции, во зависност од дозволата на полисата и вредноста на својството *dir*.

Откако ќе се обработат сите применливи полиси, елементот Q_{filter} претставува комбинација на дозволените ограничувања на податоците за тековниот шаблон за четворки. Притоа, *FILTER* изразот ги поврзува променливите од шаблонот со соодветните променливи во стандардизираните полиси. Оваа операција, всушност, ги комбинира податоците од различните полиси во зависност од нивната дозвола и приоритет. Полисата со најнизок приоритет генерално го отсликува предефинираното однесување, со што се овозможува конфигурација на предефинирана забрана или дозвола за пристап до целото множество на податоци. Така, ако полисата со најнизок приоритет забранува пристап, тогаш Q_{filter} ги филтрира податоците што се отстрануваат од резултатите, односно податоците кои ќе бидат дел од резултатите во спротивниот случај.

Откако ќе заврши процесирањето на сите применливи полиси за шаблонот за четворки, како резултат се добива Q_{filter} израз за соодветниот шаблон. Во линијата 27, Q_{filter} елементот се додава на прашањето како *FILTER EXISTS* елемент кога *dir* е *ALLOW*, а како *FILTER NOT EXISTS* елемент во спротивниот случај. Во првиот случај, само резултатите од оригиналното барање што се вкрстуваат со резултатите од Q_{filter} ќе бидат вратени. Изразот *FILTER (NOT) EXISTS* е избран бидејќи ги користи мапирањата на променливите од оригиналното барање во процесот на филтрирање и не ги редуцира резултатите доколку некои од шаблоните за четворки од полисите не се задоволени во податочното множество.

Екстрахираните *FILTER (NOT) EXISTS* изрази за секоја од четворките во блокот се додаваат на самиот блок со функцијата *filteredBlock(QuadBlock)* (линија 12) со што се добива авторизирана верзија за соодветниот блок.

```
:f a u:Faculty; u:network_address '10.10.0.0/16';.
:cs a u:StudyProgram; u:faculty :f.
:john a u:User; u:works_at :f; u:phone '070111222'.
:ben a u:User; u:works_at :f; u:phone '075333444'.
:sw_17 a u:Course; u:has_professor :ben; u:year 2017.
:sec_17 a u:Course; u:has_professor :john; u:year 2017.
:cs {
  :bob a u:User; u:enrolled_at :cs; u:phone '077123456'.
  :alice a u:User; u:enrolled_at :cs; u:phone '071654321'.
  :g1 a u:Grade; u:for_student :alice; u:for_course :sw_17; u:grade_value
     'B'.
  :g2 a u:Grade; u:for_student :bob; u:for_course :sec_17.
}
```

Изворен код 3.4: Пример податоци

```
SELECT * WHERE {
   ?s ?p ?o.
   ?g u:for_student ?s.
   OPTIONAL {
      ?g u:grade_value ?v
   }
}
```

Изворен код 3.5: Пример прашање

3.5.1 Дискусија на имплементацијата

Изворниот код 3.4 прикажува пример податоци од универзитетската онтологија[14], каде што за секоја студиска програма има посебен именуван граф кој логично ги организира студентите со нивните оценки кои се запишани во дадената студиска програма [60].

Изворните кодови 3.6 и 3.7 покажуваат два пара конфликтни полиси. Полисата *:publicUser* дозволува пристап до сите податоци на корисникот, додека пак *:protectedPhone* го забранува телефонскиот број за сите корисници за сите. Полисата *otherGrades* го забранува прегледот на оценките за сите корисници, додека *:profGrades* им овозможува на професорите пристап до оценките за нивните студенти од нивната факултетска мрежа. Приоритетите што се користат во овие полиси го дефинираат редоследот на нивното процесирање во алгоритмот за препишување на прашањето, каде полисите со повисок приоритет ќе ги заменат резултатите на оние со понизок приоритет. Полисата *:publicUser* не содржи својство *p:intent_binding* и таа не зависи од намерата. Полисата *:profGrades* ја користи флексибилноста на намерата, и ако *IntentProvider* компонентата ја додаде IP-адресата на агентот на барателот и неговата мрежа, оваа полиса ќе се

[14] https://github.com/ristes/univ-datasets/ont/univ.owl

```
:otherGrades a p:Policy;
  p:intent_binding
   'SELECT ?r WHERE {
   ?r a int:Requester }';
  p:protected_data ' WHERE {
    ?g u:for_student ?s .
    ?g ?p ?o.
    FILTER (?s != ?r) }';
  p:head_quad '?g ?p ?o ?x';
  p:permission 'DENY';
  p:priority 10.

:protectedPhone a p:Policy;
  p:intent_binding
   'SELECT ?r WHERE {
   ?r a int:Requester }';
  p:protected_data ' WHERE {
   ?s u:phone ?o .
   FILTER (?s != ?r) }';
  p:head_quad
   '?s u:phone ?o ?x';
  p:permission 'DENY';
  p:priority 30.
```

Изворен код 3.6: Полиси кои забрануваат пристап до податоците

```
:publicUser a p:Policy;
  p:protected_data ' WHERE {
   ?u a u:User.
   ?u ?p ?o }';
  p:head_quad '?u ?p ?o ?x';
  p:permission 'ALLOW';
  p:priority 20.

:profGrades a p:Policy;
 p:intent_binding
  'SELECT ?r, ?net WHERE {
   ?r a int:Requester .
   ?ag a int:Agent .
   ?ag int:ip_address ?ip.
   ?ip int:network ?net }';
  p:protected_data ' WHERE {
   ?c u:has_professor ?r.
   ?r u:works_at ?f.
   ?f u:network_address ?net.
   ?g u:for_course ?c.
   ?g ?p ?o
  }';
  p:head_quad '?g ?p ?o ?x';
  p:permission 'ALLOW';
  p:priority 50.
```

Изворен код 3.7: Полиси кои дозволуваат пристап до податоците

активира кога професорот пристапува кон системот од факултетската мрежа.

Изворниот код 3.9 ја покажува променетата верзија на иницијалното прашање од изворниот код 3.5, користејќи ги полисите дефинирани во изворните кодови 3.6 and 3.7, каде што барателот е *:john*. За ова барање се активирани сите конфигурирани полиси и нивните активирани верзии се прикажани во изворниот код 3.8. Овде, променливите се заменуваат во фазата на стандардизацијата на полисите така што четворките за проекцијата се секогаш во форма *?v1 ?v2 ?v3 ?vg*, а другите променливи последователно се заменуваат како обработени. Овие активирани полиси се поврзани со намерата преку замена на променливите *?r* и *?net* со вредностите *:john* и *10.10.0.0/16*, соодветно.

Елементите *FILTER (NOT) EXISTS* се избрани бидејќи не вклучуваат дополнителни поврзувања во прашањето. Овие елементи враќаат мапирање на променливите кои се вклучуваат/исклучуваат од резултатите на пребарувањето, во зависност од тоа дали елементот е со или без негација. На овој начин, ако полисата бара присуство на *u:network_address* својството за некој факултет,

```
1  Q_otherGrades='{ ?v1 ?v2 ?v3.
2    ?v1 u:for_student ?v4.
3    FILTER (?v4 != :john)
4  }'
5  Q_publicUser='{
6   ?v1 a u:User; ?v2 ?v3
7  }'
8  Q_protectedPhone='{ ?v1 ?v2 ?v3.
9    FILTER (?v1!=:john
10     && ?v2=u:phone)
11 }'
12 Q_profGrades='{
13   ?v5 u:has_professor :john.
14   :john u:works_at ?v6.
15   ?v6 u:network_address
16     "10.10.0.0/16".
17   ?v1 u:for_course ?v5.
18   ?v1 ?v2 ?v3
19 }'
```

Изворен код 3.8: Активирани полиси

отсуството на ова својство во податоците нема да влијае на резултатот на оригиналниот прашање. Секој од *FILTER (NOT) EXISTS* елементите за прашањето од изворниот код 3.9 е составен од два главни дела: FILTER израз и блок со четворки кој е составен од повеќе UNION / MINUS елементи. Изразот *FILTER* се користи за поврзување на променливите на полисата со променливите од шаблонот за четворки, додека пак блокот со четворки од *FILTER (NOT) EXISTS* изразот ги комбинира активираните полиси за тековниот шаблон за четворки. На овој начин, елементот *FILTER (NOT) EXISTS* ќе ги ограничи мапирањата на променливите присутни во оригиналното барање со условите зададени во активираните полиси.

Кога прашањето од изворниот код 3.5 се процесира, *модулот за извршување* заклучува дека променливата *?g* е од тип *u:Grade*, *?s* е *u:User* и *?v* е *double* со користење на својствата *u:for_student* и *u:grade_value* од дефиницијата на онтологијата. Тројките *?g u:for_student ?s* и *?g u:grade_value ?v* се поклопуваат со четворката *?v1 ?v2 ?v3 ?vg* при што се заклучува дека *?v1* → *u:Grade*. Овој услов го исполнуваат само полисите *:otherGrades* и *:profGrades*. Затоа, овие полиси се активираат и комбинираат во двата *FILTER NOT EXISTS* изрази од авторизираното прашање прикажано во изворниот код 3.9, каде дополнително се додаваат *FILTER* елементите за да ги поврзат променливите од активираните полиси со тековната четворка. Слично, тројката *?s ?p ?o* ги активира полисите *:protectedPhone* и *:publicUser*, додавајќи го *FILTER* (*?v1=?s && ?v3=?o && ?v2=?p*) изразот.

```
SELECT * WHERE {
 ?s ?p ?o.
 ?g u:for_student ?s.
 OPTIONAL {
   ?g u:grade_value ?v
   FILTER NOT EXISTS {
     FILTER (?v1=?g && ?v3=?v && ?v2=u:grade_value)
     {
       Q_otherGrades
       MINUS Q_profGrades
     }
   }
 }
 FILTER EXISTS {
  FILTER (?v1=?s && ?v2=?p && ?v3=?o)
  {
    Q_publicUser
    MINUS Q_protectedPhone
  }
 }
 FILTER NOT EXISTS {
  FILTER (?v1=?g && ?v3=?s && ?v2=u:for_student)
  {
   Q_otherGrades
   MINUS Q_profGrades
  }
 }
}
```

Изворен код 3.9: Пример заштитено прашање

```
:otherGrades {
   :g2 rdf:type u:Grade; u:for_student :bob; u:for_course :sec_17.
   :g1 rdf:type u:Grade; u:for_student :alice; u:for_course :sw_17;
      u:grade_value "B".
}
:publicUser {
  :john rdf:type u:User; u:works_at :f; u:phone "070111222".
  :ben rdf:type u:User; u:works_at :f; u:phone "075333444".
  :bob rdf:type u:User; u:phone "077123456"; u:enrolled_at :cs.
  :alice rdf:type u:User; u:phone "071654321"; :enrolled_at :cs.
}
:protectedPhone {
  :ben u:phone "075333444". :bob u:phone "077123456".
  :alice u:phone "071654321"
}
:profGrades {
   :g2 rdf:type u:Grade; u:for_student :bob; u:for_course :sec_17.
}
```

Изворен код 3.10: Заштитени податоци од активираните полиси

Изворниот код 3.10 ги покажува податоците заштитени со секоја активирана полиса[15] за прашањето од изворниот код 3.5.

Изворниот код 3.11 ги покажува резултатите што се враќаат при извршувањето на прашањето од изворниот код 3.5. По процесот на препишување на прашањето со примена на авторизација, резултатите кои се добиваат се прикажани во изворниот код 3.12, каде што се бришат забранети податоци за оценатa *:g1*, заедно со телефонскиот број на *:ben*. На овој начин, резултатите се оние кои се достапни за барателот *:john*, кога пристапува до податоците од факултетската мрежа.

Ограничувањата на променливите со функцијата *mapping* можат да го намалат бројот на приложените елементи во изразот *FILTER (NOT) EXISTS*, кој го оптимизира времето за обработка на прашањето. Меѓутоа, ако не постои онтологија која ги опишува домените и опсезите на својствата, алгоритамот за препишување на прашањето сеуште ќе функционира, но времето на извршување може да биде значително подолго. Примерот за пребарување од изворниот код 3.5 ги активира сите дефинирани полиси, но ако барањето е *SELECT * WHERE {?s a u:User. ?s ?p ?o }*, само полисите *:publicUser* и *:protectedPhone* би биле активирани, и повторно корисникот ќе пристапи само до дозволени податоци, но овој пат за пократко време, бидејќи пребарувањето *FILTER EXISTS* ќе има само две четворки. *FILTER EXISTS* изразот ќе биде додаден бидејќи *:publicUser* е полиса со најнизок приоритет од применливите и таа има *ALLOW* дозвола.

[15] Изворниот код 3.10 го изоставуваат граф елементот од четворките заради едноставност

```
| g    | s       | p              | o           | v    |
=================================================================
| :g2  | :bob    | rdf:type       | u:User      |      |
| :g2  | :bob    | u:phone        | "077123456" |      |
| :g2  | :bob    | u:enrolled_at  | :cs         |      |
| :g1  | :alice  | rdf:type       | u:User      | "B"  |
| :g1  | :alice  | u:phone        | "071654321" | "B"  |
| :g1  | :alice  | u:enrolled_at  | :cs         | "B"  |
```

Изворен код 3.11: Резултати од извршување на иницијалното прашање

```
| g    | s       | p              | o           | v    |
=================================================================
| :g2  | :bob    | rdf:type       | u:User      |      |
| :g2  | :bob    | u:enrolled_at  | :cs         |      |
```

Изворен код 3.12: Резултати од авторизираното променето прашање

Во [1] е презентиран алгоритам за препишување на барањето кој е доволно флексибилен за да ги заштити тројките и овозможува полиси зависни од контекстот. Ова е најфлексибилниот пристап од прегледаните. Меѓутоа, полисите не се во можност да ги заштитат податоците складирани во именувани графови и во различни податочни множества, а авторите не ги земаат предвид конфликтите и нивното разрешување. *Модулот за извршување* со препишување на прашањето е чекор во оваа насока, обезбедувајќи полиси зависни од контекстот кои можат да ги заштитат податоците од различни именувани графови и податочни множества. Приоритетите на полисите се инкорпорирани во алгоритамот за препишување на прашањето и обезбедуваат флексибилен механизам за разрешување конфликти. *IntentProvider* компонентата овозможува динамичка дефиниција на контекстот и параметрите што се конфигурираат да ја направат авторизацијата погодна за различни имплементации на SPARQL протоколот.

3.6 Анализа на перформансите

Според [52], архитектурата за спроведување на LDA платформата користи стратегија за *парцијално филтрирање на податоците*, каде што се создава привремено податочно множество за секое барање. Оваа стратегија ја има предноста да обезбеди имплицитна заштита на податоците, бидејќи забранетите податоци воопшто не се разгледуваат за време на обработката на намерите. Сепак, евалуациите презентирани во [51, 34] покажуваат дека перформансите се цената што треба да се плати за возврат кон оваа погодност.

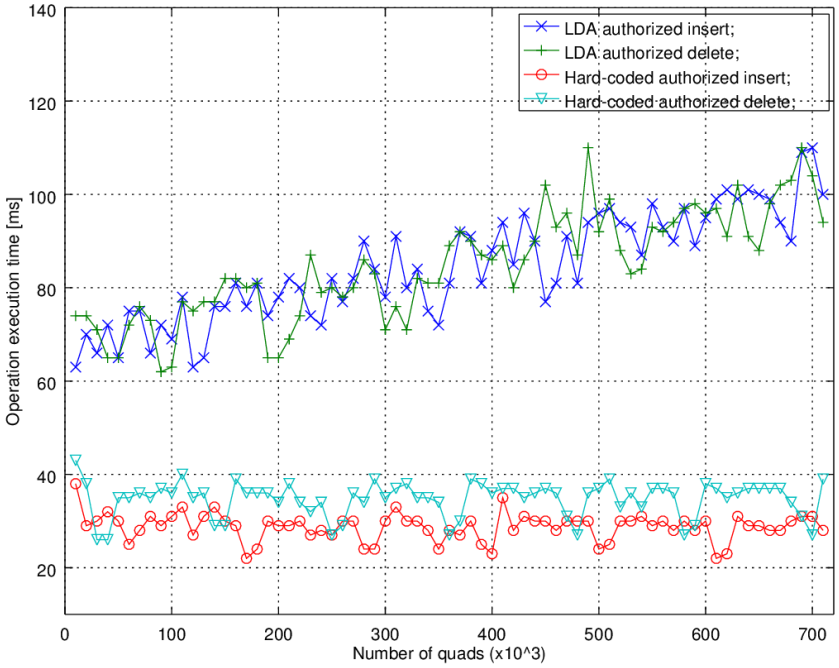
Со цел да се евалуира ефикасноста на LDA платформата, извршивме експеримент каде што системот е конфигуриран со полисите од §3.3, податоците се

моделираат со онтологијата прикажана на Слика 3-8 и иницијалните заштитени податоци се прикажани на Слика 3-7. Целта на експериментот е да ја испита зависноста на времето за обработка во однос на обемот на дозволени податоци. За да се постигне оваа цел, ние извршуваме 70 евалуациски циклуси, каде што секој следен циклус се зголемува обемот на дозволените податоци за 10000 четвроки, претставени со 2500 ресурси од типот *sm:Observation*. По секој циклус ги оценуваме различните операции користејќи 10 загревачки циклуси и 20 евалуациски циклуси. Во експериментот ги оценуваме следниве операции:

- *Standard LDA authorized query: Стандардно авторизирано пребарување со креирање на привремено податочно множество.* Се врши селекција на сите дозволени четворки во тековниот циклус. LDA платформата се користи за да се авторизира операцијата со користење на сите конфигурирани полиси за креирање на привременото податочно множество.
- *Query-centric LDA authorized query: Оптимизирано авторизирано пребарување со креирање на привремено податочно множество со полисите применливи за прашањето.* Се врши избор на сите дозволени четворки во тековниот циклус. LDA платформата се користи за да се овласти операцијата користејќи ги само полисите што можат да влијаат на резултатите од пребарувањето.
- *LDA rewriting query authorization: авторизирано пребарување со препишување на прашањето според конфигурираните полиси.* Се врши избор на сите дозволени четворки во тековниот циклус. LDA платформата се користи за да се овласти операцијата со модификација на прашањето така што ќе ги пристапи само податоците дозволени од полисите кои се применливи за него.
- *Hard-coded authorization query: авторизација со закодирани услови.* Се селектираат сите дозволени четворки во тековниот циклус со користење на закодирани шаблони за авторизација на пребарувањето, така да би се добиле истите резултати од претходните операции.
- *LDA authorized insert and delete: авторизирано креирање и бришење на четворки со LDA платформата.* Овластен барател извршува креирање и бришење на ресурси. LDA платформата се користи за да се авторизира намерата.
- *Hard-coded authorization insert and delete: авторизирани креирање и бришење на четворки со закодирани авторизациски правила.* Овластен барател извршува креирање и бришење на ресурси. Се користат закодирани правила за применливите барања за да се авторизира намерата со користење на *SPARQL ASK* прашања. Креирањето или бришењето се извршуваат само кога сите резултати од пребарувањата се позитивни.

Најчестиот пристап за авторизација кој се користи за заштита на податоци со фина грануларност е употребата на закодирани правила (hard-codded) во сервисниот слој, бидејќи на овој начин, развивачот може да ги оптимизира перфор-

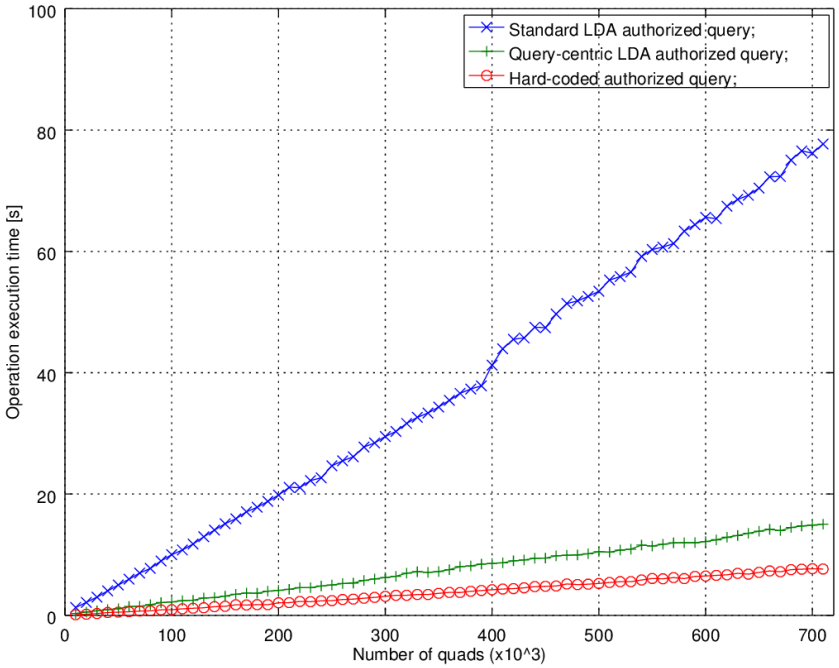
мансите. Затоа, пристапот за закодирана авторизација го сметаме за оптимален пристап за заштита во однос на перформансите.



Слика 3-12: Перформанси на модификација на четворки

Сликата 3-12 ја покажува зависноста на времето за извршување на операциите *insert / delete* според обемот на дозволените податоци. Операциите со *закодирани услови* имаат скоро константно време за извршување, затоа што извршуваат посебно пребарување *SPARQL ASK* за секоја полиса, пред да дозволат извршување на операцијата. Затоа, перформансите главно зависат од бројот на извршени пребарувања, односно бројот на полиси. Од друга страна, операциите авторизирани со *LDA платформата* имаат линеарно зголемување во времето за извршување, бидејќи LDA платформата создава привремено податочно множество за секоја намера. Трендот на забавување во овој случај се должи на линеарната зависност на времето за креирање на податочното множество од неговата големина.

Слика 3-13 покажува дека сите операции за авторизација на прашањата имаат линеарна временска зависност во однос на големината на дозволените податоци. Сепак, *стандардната авторизација на пребарување со креирање на привремени податочни множества* е околу 10 пати побавна поради комплексноста на прашањето за креирање на привременото податочно множество. Затоа, го оптимизиравме овој процес користејќи ги само релевантните полиси за бараното прашање во процесот на извршување. Овие полиси се извлекуваат со користење на автоматско заклучување на типот на секоја од промеливите од четворките



Слика 3-13: Перформанси за извршување на прашања

кои се присутни во прашањето и во полисата. Избраните полиси се само оние кои имаат совпаѓање на нивната *проекција* со некои од шаблоните од прашањето, земајќи ги во предвид на заклучените типови на променливите. Оваа оптимизација резултира со петкратно подобрување на перформансите, што е близу до оптималните перформанси, како што е прикажано со резултатите од *query-centric LDA authorized query* графикот.

Иако *закодираната* авторизација има супериорни перформанси, овој пристап, покрај тоа што не е флексибилиен, тој е тежок за тестирање и одржување. Секоја промена на барањата бара модификација на кодот и дополнително тестирање. Од друга страна, LDA платформата обезбедува флексибилна репрезентација на барањата, едноставно одржување и алатки за валидација на полисите, како и нивно тестирање. Единствената цена што треба да се плати за оваа погодност е делумно жртвување на перформансите. Сепак, оваа цена може да биде прифатлива кога е потребна комплетна флексибилност и одржливост при заштитата на податоците.

Глава 4

Централизирано управување со привилегиите

4.1 Алгоритми за трансформации

4.1.1 Трансформација на полисите во SPARQL

Со овој алгоритам за трансформација, секоја полиса станува регуларно SPARQL прашање што може да се изврши за одредена намера $\mathbb{I}$. Прашањата што се добиваат со оваа трансформација се користат во *модулот за извршување на авторизација* за време на заштитата на податоците. Покрај тоа, овие прашања овозможуваат дизајнирање на автоматски тестови за полисите и нивоно самостојно извршување во дизајн фазата.

```
1 SELECT q
2 WHERE {
3    φ_i .
4    φ_d
5 }
```

Изворен код 4.1: T_{select}: Трансформација на полисите во SPARQL SELECT прашања

Трансформацијата во Изворниот код 4.1 се однесува на полисите со *READ, INSERT, DELETE* и *MODIFY* операции, додека трансформацијата во Изворниот код 4.2 се однесува на *MANAGE* полисите. Со првата трансформација, T $_{select}$, полисата станува регуларно SPARQL SELECT прашање, кое прво ги извлекува мапирањата на променливите од графот на намерата кои ги исполнуваат условите на предикатната функција φ_i, а потоа, ги поврзува со мапирањата на промен-

ливите кои се резултат за φ_d. Променливата од интерес се враќаат со SELECT q конструкцијата од прашањето. Во овој процес ја користиме на алгебрата на SPARQL [23], која е слична на релационата алгебра [26], за имплементација на функцијата за екстракција на мапирањата на променливите σ и функцијата за проекција π.

```
ASK
WHERE {
   φ_i .
   φ_d
}
```

Изворен код 4.2: T_{ask}:Трансформација на полисите за управување со именувани графови во SPARQL ASK прашање

Слично на T_{select}, трансформацијата T_{ask} покажува како полисите со операцијата *MANAGE* можат да се трансформираат во SPARQL ASK прашања, кои одговараат дали постојат соодветни податоци за барањата за заштита.

4.1.2 Екстракција на мапирање за минимални намери

Изградбата на намерата што ќе ја активира полисата е едноставна задача бидејќи ако секоја променлива од предикатната функција φ_i се замени со произволен ресурс, резултатите тројки ќе формираат намера која ќе ја активира полисата. Сепак, оваа намера не гарантира дека полисата ќе заштитува дел од податоците $\mathbb{D}^{(\mathbb{I},p)}$, па затоа мора внимателно да ги одбереме ресурсите. Наоѓање на намера која ќе резултира со не-празни заштитени податоци $\mathbb{D}^{(\mathbb{I},p)}$ може да бара извршување на многу дополнителни пребарувања за да се најдат соодветните ресурси за намерата. Затоа, целта за екстракција на мапирање за минимални намери е да се најдат сите намери кои резултираат со не-празни заштитени податоци $\mathbb{D}^{(\mathbb{I},p)} \neq \emptyset$. Овој алгоритам за трансформација е основа за откривање на конфликти и екстракција на заштитени податоци по мапирање за минимални намери.

Дефиниција 22 ***Мапирањето за минимални намери*** *ги содржи мапирањата на променливите кои се среќаваат во двете предикатни функции од полисата (φ_i and φ_d) и кои ја задоволуваат само предикатната функција за заштитените податоци φ_d:*

$$\pi(vars(\varphi_i) \cap vars(\varphi_d), vars(\varphi_d), \sigma(\varphi_\emptyset, \emptyset, \varphi_d, \mathbb{D}))$$

Маприрањето за минимални намери ги извлекува само променливите од φ_i кои влијаат на екстракција на заштитени податоци, односно променливите што

се појавуваат и во φ_i и во φ_d. Променливите кои се појавуваат само во предикатната функција φ_i не се земаат во предвид бидејќи тие само придонесуваат за активирање на полисата, но не ги ограничуваат заштитените податоци $\mathbb{D}^{(\mathbb{I},p)}$. Затоа во формализацијата ја користиме празната предикатна функција $\varphi_\emptyset$, за σ да ги извлече само решенијата кои ја задоволуваат φ_d. Мапирањето за минимални намери ги ограничува променливите од интерес на $vars(\varphi_i) \cap vars(\varphi_d)$. Изворниот код 4.3 ја претставува структурата на SPARQL прашањето со кое се извлекуваат сите мапирања за минимални намери.

```
1 SELECT DISTINCT vars(φ_i) ∩ vars(φ_d)
2 WHERE {
3   φ_d
4 }
```

Изворен код 4.3: Екстракција на мапирање за минимална намера

Оваа трансформација им помага на сопствениците на податоците поедноставно да ги изберат соодветните намери со кои ќе ја тестираат точноста на полисите. Сите можни намери може да се добијат со замена на заедничките променливи од двете предикатни функции од полисата $vars(\varphi_i) \cap vars(\varphi_d)$ секаде каде што се појавуваат во шаблоните за тројки во φ_i со соодветните решенија за мапирањето за минимални намери, додека променливите кои се во $vars(\varphi_i) \setminus vars(\varphi_d)$ се заменуваат со анонимни јазли.

Дополнително, екстракцијата на мапирањето за минимални намери може да се користи за откривање на аномалии во дефинициите на полисите. Еден пример за аномалија што може да се открие е кога има променливи кои треба да се мапираат во минималната намера $vars(\varphi_i) \cap vars(\varphi_d) \neq \emptyset$, но екстракција на мапирањето за минимални намери не враќа резултатите. Во овој случај, полисата никогаш нема да биде активирана, што значи дека најверојатно е направен пропуст при нејзиното дефинирање и треба да се преиспита.

Изворниот код 4.4 го покажува прашањето со кое се избираат заштитените податоци за секое мапирање за минимална намера, кое ги содржи релевантните променливи од резултатот и од предикатната функција за намерата $vars(\varphi_i) \cap vars(\varphi_d)$. Предикатната функција φ_d ги избира податоците за сите можни намери, а елементот $ORDER\_BY$ ги собира заштитените податоци што се добиваат со иста намера.

Пример за мапирање за минимални намери

Пример намера за *E1* полисата од Изворниот код 3.2 е намерата $\mathbb{I}$={*ex:sam a int:Requester. _:ag a int:Agent; int:address _:ip. _:ip int:network "192.168.100.0/24"*}. Меѓутоа, кога ќе се примени полисата *E1* над податочното

```
SELECT q ∪ vars(φ_i) ∩ vars(φ_d)
WHERE {
  φ_d
}
ORDER BY vars(φ_i) ∩ vars(φ_d)
```

Изворен код 4.4: Заштитени податоци од полиса по мапирање за минимална намера

множество $\mathbb{D}$ од сликата 3-7, оваа Намера нема да резултира со заштитени податоци $\mathbb{D}^{(\mathbb{I},E1)}$. Со цел да се најде намера која ќе резултира со не-празни заштитени податоци, неопходно е да се најдат мапирањата за минимални намери кои ќе ја активираат полисата.

Предикатната функција за применливост на намерата φ_i од полисата *E1* ги содржи променливите $vars(\varphi_i)$ =*(?doc, ?ag, ?ip, ?n)*, додека предикатната функција за заштитените податоци φ_d ги содржи променливите $vars(\varphi_d)$ =*(?app, ?r, ?s, ?p, ?o, ?doc, ?h, ?pat, ?t,?n)*. Пресекот на овие множества се променливите $vars(\varphi_i) \cap vars(\varphi_d)$ =*(?doc, ?n)*, кои се користат во прашањето од изворниот код 4.5, со кое се извлекуваат мапирањата за минималните намери од Дефиницијата 22.

```
SELECT DISTINCT ?doc ?n
WHERE {
  GRAPH ?app {
    ?r a sm:Observation; sm:sensor ?s; ?p ?v
  }
  ?doc sm:works_at ?h.
  ?h sm:network_address ?n.
  ?t sm:has_doctor ?doc; sm:for_patient ?pat.
  ?s sm:owner ?pat.
}
```

Изворен код 4.5: Пример за добивање на мапирањата за минималните намери

Ако барањето во изворниот код 4.5 се изврши врз нашите пример податоци, единственото мапирање за минимални намери што се извлекува е *(ex:john, "192.168.100.0/24")*. Намерите кои ќе резултираат со не-празни заштитени податоци $\mathbb{D}^{(\mathbb{I},E1)}$ можат да се синтетизираат од ова мапирање за минимални намери. Во овој процес, променливите од предикатната функција φ_i можат да се заменат со добиените мапирања за минимални намери, а останатите променливи може да се заменат со празни јазли, со што се добива намерата која резултира со не-празно множество на заштитени податоци за полисата:

$\mathbb{I}$=*{ex:john a int:Requester. _:ag a int:Agent; int:address _:ip. _:ip int:network "192.168.100.0/24"}*

Оваа синтетизирана намера е единствената која може да ја активира полисата *E1* и притоа да се заштити не-празно множество на податоци $\mathbb{D}^{(\mathbb{I},E1)} \neq \emptyset$.

4.1.3 Заштитени податоци за полиса

Кога сопственикот на податоци е заинтересиран за целокупните податоци заштитени со полисата, без разлика каква е намерата, може да го искористи прашањето од изворниот код 4.6. Целокупните податоци заштитени од полисата се независни од намерата и се под-множество од чуваните податоци $\mathbb{D}$ што ја задоволува предикатната функција φ_d. Формално, овие податоци може да се претстават како $\pi(q, vars(\varphi_d), \sigma(\varphi_\emptyset, \emptyset, \varphi_d, \mathbb{D}))$, каде $\varphi_\emptyset$ е празна предикатна функција која нема променливи ($vars(\varphi_\emptyset) = \emptyset$) и секогаш се еволуира на *true*.

```
1 SELECT q
2 WHERE {
3   φ_d
4 }
```

Изворен код 4.6: $T_{policy\_coverage}$:Екстракција на податоците заштитени од полиса

Оваа трансформација е многу важна за процесот на креирање полиси, бидејќи обезбедува преглед на податоците опфатени со полисата и му овозможува на нејзиниот креатор да потврди дали се вклучени сите податоци од интерес.

Покриеност на полисата

Со алгоритамот за трансформација $T_{policy\_coverage}$ од изворниот код 4.6, за полисата *E1* се креира прашањето во изворниот код 4.7. Ова прашање ги извлекува податоците прикажани на сликата 3-10 од податочното множество од Сликата 3-7. Резултатите од акцијата "Coverage per Intent"("Покриеност по намери"), добиени со трансформација на полисата во согласност со изворниот код 4.4 и извршување на добиеното прашање, може да се види во табелата на Слика 3-3. Ова барање ги извлекува заштитените податоци на полисата $\mathbb{D}^{(\mathbb{I},p)}$ по мапирањата за минималните намери. На овој начин, сопственикот на податоците може да ги добие, прегледа и потврди заштитените податоци по намера со притискање на едно копче.

4.1.4 Откривање на конфликти

Кога полисите дозволуваат и/или забрануваат интеракција со податоците, можат да се појават конфликти. Откривањето на конфликти во фазата на дизајнирање

```
SELECT DISTINCT ?r ?p ?v ?app
WHERE {
  GRAPH ?app {
    ?r a sm:Observation; sm:sensor ?s; ?p ?v
  }
  ?doc sm:works_at ?h.
  ?h sm:network_address ?n.
  ?t sm:has_doctor ?doc; sm:for_patient ?pat.
  ?s sm:owner ?pat.
}
```

Изворен код 4.7: Прашање за покриеноста на E1

не е тривијална задача, бидејќи конфликтните податоците заштитени со полиците $\mathbb{D}^{(\mathbb{I}^{(c)},p)}$ зависат од конфликтната намера $\mathbb{I}^{(c)}$, чија содржина не е однапред позната. Затоа, процесот на откривање на конфликти треба прво да ги пронајде конфликтните намери $\mathbb{I}^{(c)}$ за кои постојат две полиси со спротивна дозвола што ги штитат истите податоци. Излезот од процесот на откривање конфликти мора да биде сеопфатен, со цел да се обезбеди поедноставно решавање на конфликтите. Дефиницијата 23 покажува дека две полиси се во конфликт кога имаат спротивни дозволи и ги заштитуваат истите податоци.

Дефиниција 23 *Две полиси* $p_1 = \langle \epsilon_1, o_1, q_1, \varphi_{i1}, \varphi_{d1}, \rho_1 \rangle$ *и* $p_2 = \langle \epsilon_2, o_2, q_2, \varphi_{i2}, \varphi_{d2}, \rho_2 \rangle$ *се во конфликт ако* $\epsilon_1 \neq \epsilon_2$ *и постои конфликтен интент* $\mathbb{I}^{(c)}$ *за кој* $\mathbb{D}^{(\mathbb{I}^{(c)},p_1)} \cap \mathbb{D}^{(\mathbb{I}^{(c)},p_2)} \neq \emptyset$.

Дефиницијата 24 покажува дека процесот на откривање на конфликти треба да биде во можност да ги извлече мапирањата за минималните намери, за кои се појавува конфликтот, заедно со конфликтните податоци. На овој начин, сопствениците на податоци можат да реагираат во фазата на дизајнирање за да спречат несакани последици преку процесот на разрешување на конфликти.

Дефиниција 24 *За дадено чувано множество* $\mathbb{D}$*, процесот за откривање на конфликти прифаќа две стандардизирани полиси* $p'_1 = \langle \epsilon_1, o_1, q_0, \varphi'_{i1}, \varphi'_{d1}, \rho_1 \rangle$ *и* $p'_2 = \langle \epsilon_2, o_2, q_0, \varphi'_{i2}, \varphi'_{d2}, \rho_2 \rangle$ *и ги пребарува мапирањата за минималните намери кои за конфликтна намера* $\mathbb{I}^{(c)}$ *и делот од податоци кои се заштитени од двете полиси* $\mathbb{D}^{(c)} := \pi(q_0, \sigma(\varphi'_{i1}, \mathbb{I}^{(c)}, \varphi'_{d1}, \mathbb{D})) \cap \pi(q_0, \sigma(\varphi'_{i2}, \mathbb{I}^{(c)}, \varphi'_{d2}, \mathbb{D}))$ *кои го задоволуваат условот* $\mathbb{D}^{(c)} \neq \emptyset$.

Кога две полиси се во конфликт, Дефиницијата 23 покажува дека треба да постои конфликтна намера $\mathbb{I}^{(c)}$ која ќе ги активира и двете полиси, т.е. $\mathbb{I}^{(c)}$ ги задоволува и φ'_{i1} и φ'_{i2}, што води кон $\sigma(\varphi'_{i1}, \mathbb{I}^{(c)}, \varphi'_{i2}, \mathbb{I}^{(c)}) \neq \emptyset$. Променливите извлечени од претходниот израз се $vars(\varphi'_{i2}) \cup vars(\varphi'_{i2})$. Бидејќи мапирањата за минималните намери за една полиса се претставени со променливите

$vars(\varphi_i) \cap vars(\varphi_d)$, конфликтните мапирања за минимални намери се претставени со $vars(\varphi'_{i1}) \cap vars(\varphi'_{d1}) \cup vars(\varphi'_{i2}) \cap vars(\varphi'_{d2})$. Покрај тоа, со оглед на тоа дека полисите p'_1 и p'_2 се стандардизирани, нивните заштитени податоци се претставени со променливите од шаблонот за четворки q_0. Изворниот код 4.8 го покажува прашањето што ги избира податоците од функцијата:

$\pi(q_0 \cup vars(\varphi'_{i1}) \cap vars(\varphi'_{d1}) \cup vars(\varphi'_{i2}) \cap vars(\varphi'_{d2}), vars(\varphi'_{d1}) \cup vars(\varphi'_{d2}), \sigma(\varphi'_{d1}, \mathbb{D}, \varphi'_{d2}, \mathbb{D}))$

Групирањето ги собира заедно конфликтите податоци $\mathbb{D}^{(c)}$ за исто конфликтно мапирање за минимална намера $\mathbb{I}^{(c)}$.

```
1 SELECT
2 q_0 ∪ vars(φ'_i1) ∩ vars(φ'_d1) ∪ vars(φ'_i2) ∩ vars(φ'_d2)
3 WHERE {
4     φ'_d1 .
5     φ'_d2
6 }
7 GROUP BY
8 vars(φ'_i1) ∩ vars(φ'_d1) ∪ vars(φ'_i2) ∩ vars(φ'_d2)
```

Изворен код 4.8: Трансформација за откривање на конфликти

Пример за откривање на конфликти

Да претпоставиме дека безбедносните барања од Табела 3.1 треба да се имплементираат како полиси кои ги штитат податоците од Слика 3-7. *Модулот за управување со полисите* обезбедува споредба на тековната полиса со преостанатите полиси дефинирани во платформата, прикажувајќи ги конфликтните намери заедно со конфликтните податоци. Процесот за откривање на конфликти за полисите во Табела 3.1 наоѓа 3 конфликтни парови на полиси: A2 - P1, A2 - U2 и D1 - D2. Излезот за конфликтот помеѓу A2 - P1 е прикажан во Табела 4.1. Оваа информација му овозможува на сопственикот на податоците правилно да ги разреши откриените конфликти со користење на приоритетите на полисите и со евентуално додавање на нови полиси.

4.1.5 Целокупни заштитени и незаштитени податоци

Во §4.1.3, покажавме трансформација која ги добива податоците опфатени со полисата. Оваа трансформација може да се генерализира за да се добијат целокупните заштитени и незаштитени податоци во платформата. Изворниот код 4.9 го покажува прашањето кое ги враќа целокупните заштитени податоци во платформата, каде сите стандардизирани и подредени полиси $p'_i \in \mathbb{P}$ $(i \in [1..k],\ k = |\mathbb{P}|)$

Табела 4.1: Откривање на конфликти за полисите A2 – P1

?s	?p	?o	?g	?r
ex:ben	sm:phone	075 555 555		ex:alice
ex:john	sm:phone	070 111 111		ex:alice
ex:ben	sm:phone	075 555 555		ex:bob
ex:john	sm:phone	070 111 111		ex:bob
ex:ben	sm:phone	075 555 555		ex:john
ex:john	sm:phone	070 111 111		ex:john

се комбинираат заедно со SPARQL *UNION* елементот.

```
SELECT q0
WHERE {
   { φ'd1 } UNION
   ...
   UNION { φ'dk }
}
```

Изворен код 4.9: Целокупни заштитени податоци

Барањето кое ги добива целокупните незаштитени податоци е прикажано во изворниот код 4.10, каде што φ_{q_0} е предикатна функција која ги користи променливите од q_0 за да конструира шаблон за четворки кој ќе ги селектира сите чувани податоци $\mathbb{D}$. Ова барање ги отстранува заштитените податоци избрани во изворниот код 4.9 од чуваните податоци $\mathbb{D}$ користејќи го MINUS елементот од SPARQL синтаксата. Резултатот од ова прашање претставува важна алатка за валидација на комплетноста на авторизацијата - кога има незаштитени податоци, сопствениците на податоци ќе бидат предупредени во самата фаза на дизајнирање на полисата.

```
SELECT q0
WHERE {
   { φq0 }
   MINUS
   {
      { φ'd1 } UNION
      ...
      UNION { φ'dk }
   }
}
```

Изворен код 4.10: Целокупни незаштитени податоци

Табела 4.2: Незаштитени податоци за операцијата READ

?s	?p	?o	?g
_:b0	geo:lat	"42.004"8sd:double	
_:b0	geo:long	"21.409"8sd:double	
_:b1	geo:lat	"42.010"8sd:double	
_:b1	geo:long	"21.410"8sd:double	
ex:o1	rdf:type	sm:Observation	
ex:o1	sm:sensor	ex:s1	
ex:o1	sm:val	66	
ex:o1	sm:time	1500386600319	
ex:o2	rdf:type	sm:Observation	
ex:o2	sm:sensor	ex:s1	
ex:o2	sm:val	57	
ex:o2	sm:time	1500386690319	
ex:o3	rdf:type	sm:Observation	
ex:o3	sm:sensor	ex:s2	
ex:o3	sm:val	28	
ex:o3	sm:time	1500386690319	

Екстракција на (не)заштитени податоци

Честа двосмисленост што се појавува во платформите за авторизација е дали треба да се дозволат или забранат незаштитените податоци. Табелата 4.2 покажува дека полисите од Табела 3.1 не се доволни за целосна заштита на пример податоците кои ги разгледуваме. Оваа информација може да спречи изложување на приватни податоци (како набљудувањата *ex:o1, ex:o2* и *ex:o3*) или да ги сокривање на податоците што се наменети за јавно добро (како што е локацијата на болницата *_:b0*). Трансформацијата за екстракција на незаштитените податоци му сигнализира на сопственикот на податоците дека треба да се креираат дополнителни полиси за целосна заштита на податоците. *Модулот за управување со полисите* ги открива овие информации со притискање на копче (Слика 3-11), покажувајќи ги незаштитените податоци за секоја операција. Дополнително, во разгледаниот пример има уште повеќе незаштитени четворки за операциите *INSERT* и *DELETE*, односно 43 од 59 четворки.

4.2 Евалуација на LDA платформата

LDA платформата е дизајнирана да ги поддржува сите принципи за дизајн на авторизација дефинирани во поглавјето 2.2. Табелата 4.3 ги резимира компонентите и можностите кои ги поддржуваат принципите на *Флексибилност, одржс-*

Табела 4.3: Подршка на принципите за авторизација во LDA платформата

Компонента/својство на LDA платформата	Флекс.	Одрж.	Валид.	Разбирл.
Јазик за полиси	✓	✓		✓
Intent Provider	✓			
Модул за извршување			✓	
Модул за управување со полисите		✓	✓	
Трансформација во SPARQL		✓	✓	
Заштитени податоци за полиса		✓	✓	
Екстракција на мапирање за минимални намери		✓	✓	
Откривање на конфликти		✓	✓	
Целокупни заштитени и незаштитени податоци		✓	✓	

ливост, коректност и *разбирливост*. Јазикот за полисите е дизајниран да ги опфати сите аспекти за *флексибилност* дискутирани во §2.2. Со цел да се зголеми *разбирливоста*, нашиот јазик е базиран на SPARQL јазикот, стандардизиран од W3C организацијата, а со тоа полисите лесно може да се трансформираат во SPARQL прашања кои можат да се извршат над чуваните податоци $\mathbb{D}$, што ја поедноставува нивната *одржливост*. Компонентата *IntentProvider* обезбедува динамичка контекстна застапеност, што е важен дел од принципот за *флексибилност*. *Модулот за извршување* може да користи привремени податочни множества со што осигуруваат точна заштита [51] и обезбедува имплицитна сигурност. Алгоритмите за трансформација на полисите означени како *Policy to SPARQL*, *Policy coverage*, *Minimal Intent Binding*, *Conflict detection* и *Protected/Unprotected data* се користат во *модулот за управување со полисите* и може да се извршат во фазата на дизајнирање, со што тие го поедноставуваат процесот на одржување и може да се користат за да се потврди дека чуваните податоци се соодветно заштитени.

Главната цел на нашата работа е да изградиме флексибилна, одржлива и разбирлива LDA платформа која ги опфаќа сите принципи за дизајн опишани во §2.2. *Флексибилноста* на LDA платформата во споредба со поврзаната работа се демонстрира со множество разновидни барања за авторизација од Табела 3.1. Барањата во оваа табела се избрани за да ги покријат сите принципи за *флексибилен* дизајн во однос на сценариото од §3.3. Оваа табела јасно покажува дека LDA платформата е единствената која целосно ги опфаќа сите овие барања, т.е. сите аспекти на *флексибилност*. Полисите за секој поединечен услов, напишани во нашиот јазик за полиси, заедно со податоците од Слика 3-7 што ги штитат, се прикажани во §3.3.

Потребата за *контекстна заштита* е присутна во барањата *D1, D2* и *EM1*, бидејќи *D1* зависи од IP адресата на софтверскиот агент на барателот и од времето на пристап, *D2* зависи само од времето на пристап и *EM1* е полиса која

се активира во итни случаи и зависи од контекстот на податоците. Иако платформите [88, 61, 1] моделираат контекстна заштита во нивните полиси, тие ги поддржуваат само операциите за читање. Нашата LDA платформа е уникатна во поддршката на контекстна заштита за секоја операција, бидејќи обезбедува целосна *покриеност на операциите* за поврзани податоци. Компонентата *IntentProvider* овозможува динамичко претставување на контекстот преку намерата, додека *Модулот за извршување* ја штити секоја операција во тековниот контекст.

Друга флексибилна особина што ја прави нашата LDA платформа препознатлива е способноста да се дефинира *агрегирано споделување на податоци* преку полисите, што е посочено со барањето *A3*. Ниту една од разгледаните платформи за авторизација не ја поддржува оваа функција.

LDA платформата исто така е супериорна од гледна точка на одржување, бидејќи таа е единствената која обезбедува приближно една полиса за секое специфицирано барање, откривање на конфликти и екстракција на целокупните заштитени и незаштитени податоци преку *модулот за управување со полисите*.

Извлекувањето на мапирањата за минималните намери опишано во §4.1.2 значително го намалува бројот на потребните тест случаи со отстранување на оние што не заштитуваат никакви податоци. Заштитените податоци по мапирање за минимална намена обезбедува преглед на сите можни исходи за полисата, што е значајна информација за проверка на нејзината точност. Заштитените податоци по мапирање за минимална намена за секоја од полисите во Табела 3.1 е прикажана во §3.3.

Уште еден придонес на нашата работа е *модулот за управување со полисите*. Овој модул е моќна алатка која значително го намалува напорот за одржување и времето на моделирање на полисите. Обезбедува преглед на резултатите со разновидните алгоритми за трансформација на полисите, како што се мапирањата за минимални намери, откривање на конфликти и екстракција на целокупните заштитени и незаштитени податоци уште во времето за дизајн. Резултатите од *модулот за управување со полисите* овозможуваат откривање на аномалии во дизајнот на полисите, додека флексибилноста на предложениот јазик за полиси овозможува пишување полиси кои ќе ги надминат овие безбедносни пропусти.

Употребата на SPARQL јазикот за пребарување како основа на нашиот јазик за полиси го поедноставува процесот на одржување со намалување на бројот на потребните полиси за секое барање за заштита. Големата заедница и квантитетот на достапни материјали за SPARQL јазикот ја намалуваат бариерата на усвојувањето на нашиот јазик. Веќе постојат алатки кои им овозможуваат на не-технички личности да напишат SPARQL прашања [31, 12, 11], кои можат да бидат усвоени за управување со нашите полиси со малку напор и прилагодувања.

Бидејќи главна цел на LDA платформата е да обезбедиме флексибилна авто-

ризација на податочните множества, ги искористивме најсовремените протоколи за автентикација обезбедени од Spring рамката за развој. Модулот *Intent Provider* ги обединува процесите за авторизација и автентикација. Ги извлекува својствата обезбедени преку протоколот за автентикација и ги користи за составување на *намерата* што се користи за авторизација во *модулот за извршување*. Тековната имплементација на модулот *Intent Provider* користи *Authetication Provider* од Spring Security библиотеката за протоколот WebID, кој е најшироко користен протокол за автентикација во заедница која се бави со поврзани податоци. Во зависност од сценариото на употреба, може да се користи и различен протокол со цел да се извлечат својствата на барателите и да се обезбедат нивните намери.

LDA платформата е робусна во смисла на *Query Injection* бидејќи не прифаќа параметризирани прашања, кои најчесто се користат за овој тип на напади. Во суштина, *Query Injection* е чин на замена на очекуваните параметри со друго злонамерно прашање, при што се извршуваат повеќе барања наместо планираното. LDA платформата осигурува дека постои единствено барање за намерата и да осигури дека прашањето ќе има интеракција само со дозволените податоци.

Бидејќи заштитата на податоците зависи од полисите, од клучно значење е да се осигура дека нема напад што ќе ги измени конфигурираните полиси. Тековната имплементација на LDA платформата е конзервативна во однос на администрацијата на полисите, и иако овозможува тестирање и управување со полисите, само лицето кое има контрола на инстанцата од LDA платформата може да ги објави полисите. Друга опција е да се ослободи овој процес со зачувување на полисите во RDF форма преку опишаната онтологија и да се одобри нивното одржување преку полисите на LDA платформата. Во ова сценарио, условите за управување со полисите мора да имаат најголем приоритет, така што никој нема да може да ги пребрише со нови полиси, односно, невозможно е да се креира полиса со повисок приоритет. Овој предизвик се решава со ограничување на максималниот приоритет на ново-додадените или изменетите полиси до претходно дефинирана вредност.

Еден од највисоките безбедносни ризици во секоја платформа доаѓа од внатрешните човечки грешки. Со цел да се надминат грешките и недостатоците во конфигурацијата, *модулот за управување со полисите* обезбедува алатки за опширно тестирање на валидноста во однос на барањата, како што е опишано во §3.3. Овие алатки им овозможуваат на сопствениците на податоци да обезбедат соодветна заштита и да спречат несоодветна изложеност на податоци. Имајќи предвид дека полисите се користат за изградба на привремено податочно множество без никакво надворешно влијание, можеме да осигуриме дека резултатите од тестирањето се релевантни во реална продукциска средина, без оглед на тоа какво прашање ќе се изврши над нив. Фактот што привременото податочно множество ги содржи само дозволените податоци, LDA платформата спречува добивање на податоци со користење на повеќе прашања, и затоа нуди имплицитна заштита на податоците, како што е објаснето во §3.1.3.

Глава 5

Примена на LDA платформата во доменот на интернет на нештата

5.1 Сигурност кај интернетот на нештата

Поголемиот дел од работата во областа на авторизацијата кај интернетот на нештата (Internet of Things - IoT) се потпира на концептот за обезбедување на комуникациски канал со сигурност на транспортниот слој (TLS) или Datagram TLS (DTLS) [73], преку различни начини на дефинирање и дистрибуција на клучеви [2, 47, 66, 36]. Овие пристапи ја спроведуваат авторизацијата со користење на споделени клучеви за уредите и корисниците, со цел да обезбедат нивно идентификување и да комуницираат безбедно. Тие не обезбедуваат опција за филтрирање на делови од податоците што се забранети во процесот на размена. Ваквата заштита може да резултира со несакано споделување на информации. Овој пристап е делумно проширен со употреба на OAuth протоколот [42], каде што овластувањето е определено врз основа на опсегот и достапните ресурси. Неколку имплементации на овој пристап се достапни во IoT доменот, некои за MQTT протоколот [66, 36], и други за CoAP протоколот [20]. Работата во [37] опишува како може да се заштитат веб-интерфејсите за IoT инфраструктура користејќи го OAuth протоколот за авторизација.

Во [65] авторите дискутираат дека заштитата на IoT системите треба да обезбеди поддршка за динамички контекст, доверливо управување, контрола на проток на информации и акции за контрола на актуаторите, како и анонимизација на податоците. Со цел да ги обезбедат овие функционалности, тие ја имплементираат SecKit алатката, која овозможува управување со повеќе аспекти, како што се податоците, однесувањето и контекстот, помеѓу другите. Овие модели имаат структура на графови која се управува преку стебло компонента, што го прави овој процес двосмислен. Дефиницијата на моделот во нивниот систем ба-

ра многу техничко знаење и вештини за моделирање. Ова наметнува потреба за добро обучени специјалисти за безбедност, кои ќе да се грижат заштитата на IoT системите со оваа алатка. Заштитата е дефинирана преку правила во форма настан-состојба-акција, што обезбедува флексибилна дефиниција на полисите. Сепак, користењето на концептите од другите модели го отежнува одржувањето. Овој модел на полиси се користи во [64] за MQTT протоколот од апликацискиот слој. Авторите дефинираат компонента за спроведување на полисите која е вградена во MQTT брокерот за пораки, која ги спроведува дефинираните правила за заштита.

Авторизацијата на пристапот до IoT системите исто така зависи од контекстот во кој работи уредот [65, 81, 78]. Во [78] е воведен концепт на систем за лоцирање на личности базиран на идентитет, каде што локацијата се дели само во случај на итност. Полисите предложени во оваа работа дефинираат „ниво на вонредна состојба“, што е условот според кој полисата се активира и се дозволува локацијата. Сепак, авторите даваат само еден пример за полиса, во човечки читлива форма и не даваат никаков понатамошен формализам. Во [81] авторите ја опишуваат потребата за „итни полиси“ преку употреба во доменот на здравствената заштита, но не е претставен начинот на управување со контекстните информации и форматот на полисата.

Податоците од IoT уредите можат да бидат логички претставени како тек, така што постои потреба за овластување на преносот на податоци. Работата во [69] обезбедува теоретска основа за заштита на тек од податоци. Авторите се фокусираат на автентичноста и комплетноста на податоците во резултатите. Во [18] и [17] авторите дефинираат безбеден преглед, читање, агрегирање и внесување во заштитен тек од податоци со филтрирање на авторизираните податоци. Безбедните операции користат изрази со логички оператори и оператори за множества, во комбинација со изрази за филтрирање на податоци, со цел да ги дефинираат податоците што треба да бидат достапни во текот за примачот. Полисите се чуваат и обработуваат од страна на Системот за управување со податоци. Овие системи бараат високо теоретски знаења од администраторите на системот со цел да се дефинираат полисите. Не постои опција за децентрализирано управување со полисите, оставајќи ги сопствениците на уредот без опција да дефинираат како нивните податоци ќе бидат заштитени. Овој проблем е решен во [63], каде што сопственикот на податоците ги внесува полисите во генерираниот тек со податоци, а процесорите на текот или брокерите можат да одлучат на кого да ги дистрибуираат информациите. Во оваа работа полисите се во форма на торки и филтри, каде што сопственикот дефинира кои улоги можат да добијат кои типови на податоци.

Заштитата на комуникацијата помеѓу IoT уреди кои се управувани од апликации во облак (Cloud) е анализирана во [7, 82] со користење на проширен модел за контрола на информациите базиран на работата во [28]. Авторите го истакнуваат значењето на контролата на протокот во доменот на IoT и дефинираат форма-

лен модел за нивните полиси во форма на асоцирање на безбедносни ознаки на податоците и процесите (услуги или уреди), а потоа ја применуваат заштитата врз основа на овие ознаки. Во [41] е претставен безбедносен модел заснован на контекст, каде полисите ја дефинираат способноста на секоја корисничка улога, а правото на пристап се добива врз основа на достапната способност за корисникот. Контекстот обезбедува информации кои се користат за одредување способностите.

Иако постојат трудови кои моделираат различни аспекти од овластувањето на IoT, како што се заштитата на тек од податоци, свесност за контекстот, контрола на проток на информации и обезбедување на идентитет (со сертификати или OAuth), не постои комплетно решение кое обезбедува полиси кои заедно ги опфаќаат сите овие функции. Ниту една од анализираните решенија не обезбедува надминување на хетерогеноста во IoT доменот во процесот на заштита на податоците, како што се различни единици мерки или различна точност на мерењата добиени од различни сензори. Овие предизвици укажуваат и за потребата за комплетен модел за полиси кој треба да ги опфати и сите карактеристики од традиционалните системи (базирани на API), бидејќи уредите на IoT се координирани и консумирани од вакви апликации, а полисите треба да обезбедат дистрибуирана и целосна заштита на целата инфраструктура.

5.2 Придобивки од поврзаните податоци и LOD платформата за безбедноста во IoT доменот

Достапните системи за заштита опишани во §5.1 се придржуваат кон карактеристиките на IoT уредите, како што е природата на нивните податоци, нивната неразделна врска со контекстот во кој тие функционираат и потребата за нивна комуникација, со цел да се обезбеди автономна функционирање на системот. Сепак, работата во оваа област не го адресира на прашањето за хетерогеност воведено од различните платформи, протоколи и формати на податоци кои се користени кај различните уреди. Исто така, пристапите за контрола на пристап кои се анализирани овде не земаат во предвид дека повеќето корисници на IoT системите ќе бидат регуларни луѓе без техничко знаење за основните технологии и не се свесни за безбедносните ризици наметнати од околината на паметните IoT уреди околу нив.

5.2.1 Безбедност на семантички податоци

Употребата на семантичка мрежа може да овозможи подобра перцепција и дејствување од страна на IoT уредите доколку се обезбеди флексибилна контрола

на пристап до нив и нивните податоци. Луѓето веќе стануваат свесни за вредноста на нивните податоци, како и за ризиците по нивната приватност доколку нивните податоци се користат неовластено. Од овие причини, постои потреба за унифицирани полиси кои се лесни за управување и разбирање, но доволно флексибилни за заштита на најситниот дел од нивните податоци. Семантичката мрежа обезбедува стандарди кои ги надминуваат проблемите на хетерогеност во доменот на IoT и овозможуваат полесна интеграција на доменски-ориентираните апстракции, наместо поврзување на сурови податоци.

Заштитата на текови од податоци од IoT уреди треба да дефинира кој дел од текот на податоци е видлив за кој субјект. Друг аспект е заштитата и авторизацијата за претплата (subscription) за обработка на текот. Овие предизвици може да се решат со дозволување на претплата за сите текови, но филтрирање на сите податоци од текот доколку не е дозволена претплата до него. Ова значително ги поедноставува полисите, но ја зголемува обработката и комплексноста на основниот систем. Главните отворени предизвици тука се:

- Заштита на текови од семантички анотирани податоци
- Заштита на податоци од повеќе комбинирани текови истовремено
- Контекстна заштита која ги комбинира податоците од текот со описот на барателот

И покрај тоа што постои значајна работа за контрола на пристапот во семантичките мрежи, аспектите на управување со откривањето на уредот и контрола на проток на информации не се опфатени. Исто така, иако постои работа која ги вклучува тековите на податоци во семантичките мрежи [55, 8], нема значителен напредок во областа на контролирање на пристапот за овие стандарди.

Јазикот за полиси во LDA платформата е дизајниран за да може да моделира барања кои го поврзуваат контекстот со барателот и податоците. Ваквите барања се многу чести во IoT доменот, што го прави нашиот јазик погоден. Бидејќи во основата на [55, 8] јазиците е *SPARQL* прашалниот јазик, форматот за полиси опишан во §3.2 е соодветен и за заштита и на овој тип на прашања. Притоа, доколку описот на секој од IoT уредите, заедно со нивниот контекст се чува во посебно податочно множество, може да се искористи целосниот потенцијал за заштита на податоците со LDA платформата.

Тековите на податоци исто така може да се заштитат. Во овој случај конзумерите треба да се претплатат на податоците кои им се од интерес преку некоја од варијантите на SPARQL за тек од податоци. Како резултат би добивале филтрирани податоци кои се дозволени за барателот, добиени со додавање заштита на прашањето кое е претплатено. Алгоритмот за промена на прашањето од §3.5 е соодветен за во овој случај, затоа што основата на варијантите на SPARQL за тек од податоци е SPARQL прашалниот јазик.

Сепак, полисите од LDA платформата не се во можност да моделираат барања кои се однесуваат на времетраењето на „прозорот“ на филтрирање на stSparql[55] и C-SPARQL[8] јазиците, а за дозвола и забрана на цели текови од податоци би требало да се пренамени *DATASETS* елементот од јазикот или да се воведе нов.

5.2.2 Откривање на уреди

Во [9] е опишана архитектура на систем кој користи семантича апстракција на податоците од IoT уредите, каде податоците за откривањето на уредот исто така се претставени како поврзани податоци, овозможувајќи иста полиса да може да го заштитува откривањето на уредот и неговите податоци во корелација. На овој начин, процесот на откривање на уредите може да се постигне со усвојување на техники за контрола на пристапот на SPARQL крајните точки. Доколку податоците за секој од уредите се чуваат во семантички формат, можноста за федерација на SPARQL прашања овозможува во LDA платформата да се дефинираат полиси кои динамички ќе ги филтрираат податоците според карактеристиките и контекстот на уредот. Контролата за тоа кој ќе може да дознае за постоењето на уредот, како и добивање на неговите информации исто така може да се контролира преку LDA платформата, затоа што овие податоци се во семантички формат.

5.2.3 Автентикација

Со цел да се идентификува барателот кој сака да го добие потокот или да го открие уредот, протоколот WebID [83, 87] може да се прилагоди за IoT доменот. Овој протокол го пренесува описот на барателот како именуван граф од поврзани податоци во заглавјето на HTTP-пораката, а протоколот обезбедува валидациски механизми користејќи јавни и приватни клучеви преку X509 сертификатите. Бидејќи овој протокол е базиран на X509 сертификати за доверливо одржување, истите сертификати може повторно да се користат за заштита на комуникација на транспортниот слој, било преку TLS или DTLS протоколите.

Модулот за автентикација ги обезбедува информациите за барателот кој се обидува да добие дел од податоците, или да ги открие атрибутите и услугите на уредот. Ако се користи WebID протоколот, барањето со себе носи граф од поврзани податоци кој го опишува барателот, што ќе им овозможи на уредите да донесат одлука за податоците што би ги споделиле.

Дополнително, способноста на LDA платформата за интеграција со овој протокол за автентикација овозможува дополнителен бенефит, бидејќи полисите може директно да го искористат семантичкиот опис на барателот, без потреба од дополнителна конверзија на податоците, што би заштедило доста време и енер-

гија кај уредите.

5.2.4 Примена на полисите

Главниот предизвик за примена на LDA платформата во IoT околините е да обезбеди истовремена поддршка на текови на податоци и за стандардни пребарувања за откривање на уреди и услуги, без наметнување на значителни забавувања на перформансите.

Во оваа дисертација веќе дискутиравме дека LDA платформата е соодветна за заштита на повеќе податочни множества истовремено, но не и за тоа како може да се заштитат текови од податоци. За заштита на тековите на податоци е потребна делумна модификација на модулот за извршување, така што *DATASETS* елементот ви се оденсувал на тоа за кој текови од податоци ќе се примени заштитата. Дополнително, би требало да се оптимизираат перформансите на алгоритмот за промена на прашањето со заштита со цел да се намалат доцнењата на опслужувањата на барателите на податоците.

5.2.5 Чување и управување со полиси

Барањата за интероперабилност и приспособливост на IoT наметнуваат дека полисите за контрола на пристап треба да се зачуваат на дистрибуиран начин. Архитектурата дефинирана во [9] е погодна бидејќи полисите може да се зачуваат и да се превземаат од секој портал со помош на SPARQL веб интерфејсите.

Моделот на предложениот јазик за полиси е сеопфатен и ќе овозможи нивно лесно одржување, контекстна заштита и флексибилна заштита на пристапот до дел од податоците. Бидејќи SPARQL е формален јазик и повеќето од редовните корисници не се запознаени со него, предлагаме едноставен интерфејс за градење на барањето да им овозможи на корисниците да го користат без да ја жртвуваат флексибилноста[89, 84]. Уште подобро решение може да биде употребата на водени природни јазични интерфејси како што е [11, 4]. Модулот за управување со полисите треба да се подобри за да обезбеди интуитивен интерфејс за дефинирање на полисата од не-технички личности. Овој модул исто така треба а се прошири и да овозможи откривање на полисите од страна на IoT уредите, се со цел да одлучат каде и кои податоци да ги споделат.

Глава 6

Бизнис модели за користење на заштитени семантички податоци

Сведоци сме дека јавни и приватни ентитети продуцираат и собираат огромни количини на податоци како дел од нивните дневни операции. Истовремено, потребни се зголемеи инвестиции во IT инфраструктурата и потребните квалификации за одржување и користење на овие податоци во комплексни хардверски и софтверски системи. Поврзаните податоци овозможуваат креирање на подобри и помасивни сервиси кои би ги користеле овие податоци, користејќи ги постоечките инфраструктури до нивниот целосен потенијал. Поврзаните податоци имаат голем потенцијал за компаниите затоа што нудат можност за поврзување на хетерогени податоци од различни извори и овозможуваат изедначување на истите концепти со различни репрезентации [80].

Сепак, овие податоци се уште не се доволно искористени во бизнис секторот и главно се користат во истражувачки и владини проекти. За владиниот сектор, поврзаните податоци нудат отвореност, транспарентност и ефикасност во надгледувањето. Исто така, постои значителен економски потенцијал во поврзаните податоци за отворена влада, каде главниот придонес е подобрување на постоечките и создавање на услуги со додадена вредност. Овој потенцијал може да се реализира на корисни бизнис проекти доколку се достигне одреден праг на количество и квалитет на податоци и релевантно знаење.

За адаптрација на поврзаните податоци во компаниите е потребен соодветен бизнис модел со кој ќе се генерира вредност за нив и ќе се овозможи тргување со овие податоци. Дополнително, овие бизнис модели би требало да овозможат поефикасно управување со знаењето и поголема вклученост и колаборација помеѓу произведувачите и конзумерите на податоците. Меѓутоа, во денешно време, компаниите кои поседуваат најмногу податоци се и најбогати. Знаењето добиено од овие податоци им дава компетитивна предност и подобро прилагодување на

нивните клиенти. Токму затоа, главната пречка за овие значајни податоци да бидат поврзани со останатиот свет е неможноста да се тргува со нив, односно да се контролира кој податок ќе биде достапен и за која цена.

Програмабилните веб интерфејси (Web APIs) се најраспространетиот начин на поврзување на податоците од повеќе различни сервиси. Меѓутоа, овие интерфејси не се стандардизирани и не нудат флексибилно отворање на податоците. Доколку е потребен пристап до некое специфично подмножество од податоци, кое не е овозможено преку овој интерфјес, заинтересираните конзумери не може да ги добијат овие податоци, дури и да сакаат да платат за нив. Поврзаните поадтоци нудат унифициран и флексибилен начин за пристап преку SPARQL протоколот, меѓутоа, потребно е да може да се контролира пристапот и да се дефинираат цените кои клиентите би ги платиле за овие податоци.

6.1 Бизнис модели за поврзани податоци

Во [91] е направена детална анализа за бизнис моделите кои се адекватни за поврзаните податоци. Во продолжение е детален опис на секој од идентификуваните бизнис модели:

1. **Јавни сервиси:** Сервиси кои вклучуваат поврзани податоци кои главно се произведуваат од јавниот сектор. Пример за вакви податоци се државни статистики, адреси, демографски податоци и податоци кои ја зголемуваат транспарентноста во работењето на владините институции. Основната инфраструктура на овие податоци треба да се иницира и одржува од владини тела.
2. **Сервиси од заедницата:** Интернет мрежата е отворена и овозможува развој на сервиси преку кои групи од корисници може да креираат, уредуваат и објавуваат отворени и поврзани податоци на доброволна база. Кај овие сервиси приходите главно произлегуваат од донации, како што е примерот со Википедија (Wikipedia) и Отворената Фондација за Знаење (Open Knowledge Foundation). Одржливоста на овој модел се базира на лојалност на корисниците и на нивните високи инвестиции во време и емоции [72].
3. **Претплати:** Во онлајн и офлајн маркетите, корисниците треба да платат периодични средства за да се претплатат на различни нивоа на пристап до податоците:
   (а) **Целосен пристап:** Плаќање за пристап до детални и подобро поврзани податоци, како на пример линкови до други податочни множества [30], вклучувајќи средства за лиценцирање за да им се овозможи на развивачите да ги користат податоците во други околини [16].
   (б) **Временски пристап:** Претплати за пристап на најновите верзии од податоците, како на пример, одложени цени на акции [30].

(в) **Пристап по потреба (on-demand):** Плаќање за пристап по потреба за индивидуални прашања или конкретни податочни множества, односно овозможување за извршување на микро-трансакции [16].

(г) **Блоков пристап:** Плаќања за пристап до податочните множества врз основа на време, на пример, дневена дозвола, или врз основа на бројот, фреквенцијата и конкурентноста на пристапот [30].

(д) **Архивски пристап:** Бидејќи објавувањето и одржувањето на големи архивски бази на податоци на интернет може да биде скапо во смисла на конверзија на податоци, дигитализација, меѓусебно поврзување и обезбедување на услуги, со овој модел се дозволува пристап само на оние кои се претплатиле на архивските податоци [30].

(е) **Удобен пристап:** Плаќања за пристап до податоците преку специфичен механизам [30].

(ж) **Freemium:** Корисниците можат да уживаат во слободен, но ограничен пристап до предефиниран примерок од податоците, но се наплаќа надоместок за продолжен пристап до останатите податоци [16].

4. **Прилагодени сервиси:** Некои компании наплатуваат надоместоци за да обезбедат професионални услуги, интелигентни алатки и кориснички решенија за креирање, уредување, објавување и повторна употреба на поврзаните податоци за одредени деловни потреби, како на пример, статистичка анализа на сообраќајот со поврзани пдоатоци, интеграција во корпорациски системи и прилагодување на онтологиите.
5. **Спонзорство:** Рекламни компании можат да плаќаат одредени сретства за да бидат вклучени во множества од поврзани податоци, а на мал број рекламни компании може и да им се наплаќа за видливост на брендот за спонзорирање на податоците [30].
6. **Рекламирање:** Контекстуалното рекламирање на веб-страници и спонзорираното пребарување може да биде силнен извор на приходи од поврзаните податоци, како што е случај и во тековната Интернет економија. Меѓутоа, рекламирањето на ниво на податоци не е корисно ниту пак изводливо во пракса, затоа што овие реклами може лесно да се идентификуваат и да се игнорираат од апликациите како што се RSS рекламирање [30].
7. **Пазар:** Производителите и администраторите на поврзаните податоци можат да обезбедат податоци за партнерот во замена за напредни и корисни сервиси [16].
8. **Придружни програми:** Како и во случајот на програмабилните веб интерфејси (API), креаторите на поврзани податоци продуцираат податочни текови на партнерите кои ги испорачуваат до крајните корисници преку апликации, во замена на дел од профитот кој го остваруваат[16]. Покрај тоа, програмите на постоечките успешни партнери, како што се Амазон и Фликр, можат да ги прифатат поврзаните податоци со цел да ги надградат своите услуги. Партнерите во придружните програми можат да ги поврзат партнерските производи со податоците, во замена за провизија од продажбите [16].

9. **Повеќе-страни платформи:** Повеќе-страни платформи (или двострани мрежни ефекти) се евидентирани во случаи кога зголемувањето на употребата од едно множество на потрошувачи ја зголемува вредноста на комплементарното добро на друго, различно множество на потрошувачи и обратно. На пример, Google нуди бесплатно богато множество на услуги со цел да се промовира продажбата на рекламната платформа. Во моментов, услугите базирани на поврзани податоци се вклучени во пакетот бесплатни услуги на Google, како што се на пример Google Refine [48] и Google public data explorer[1].
10. **Генерирање на сообраќај/Оптимизација за пребарувачи (SEO):** Објавувањето на поврзани податоци може да им помогне на веб-сајтовите да добијат повисоки позиции во пребарувачите и другите директориуми за да привлечат повеќе корисници [16].

6.2 Подржани бизнис модели од LDA платформата

Контролата на пристапот на податоците е во основата на бизнис моделите за *претплата*, *придружни програми* и *пазар*, затоа што приливот на сретствата кај овие модели зависи од тоа кои податоци ќе бидат пристапени. Флексибилноста на контролата на пристапот кај овие модели е клучна за дефинирање на повеќе пакети за претплата или производи, или пак прилагодување на податоците достапни за партнерите. Дополнително, *сервисите од заедницата* може да имаат огромен бенефит доколку се споделат агрегирани и анонимизирани податоци, без притоа да се загрози приватноста и безбедноста на сопственикот на податоците.

Со својата флесибилна заштита на податоците, LDA платформата ги подржува бизнис моделите *претплата*, *придружни програми* и *пазар*, кои генерираат директни приходи, како и бизнис моделот за *сервисите од заедницата*, со што се овозможуваат нови и поиновативни сервиси и начини за искористување на постоечките податоци.

Во продолжение ќе опишеме како секој од специфичните бизнис модели може да се подржи со LDA платформата.

[1] https://www.google.com/publicdata/directory

6.2.1 Претплати

Во онлајн и офлајн маркетите, корисниците треба да платат периодични средства за да се претплатат на различни нивоа на пристап до податоците. Сопствениците на податоците во овие сценарија треба да имаат можност за детална контрола за тоа кои податоци ќе бидат објавени и за која цена. Флексибилноста на заштитата игра клучна улога во овој процес, но не е единствениот одлучувачки фактор за профитабилност. Имено, многу е важно сопствениците да се осигураат дека нема да настане неовластен пристап до дел од податоците, затоа што на тој начин тие би изгубиле дел од профитот. Поради ова, валидацијата за време на дизајнот на пакетите за претплата игра клучна улога.

Нашата LDA платформа е значителна подршка при користење на овој бизнис модел, затоа што флексибилниот јазик за полисите овозможува дизајнирање на пакети кои ќе содржат произволно множество на податоци. Пакетите за пристап во нашата платформа се репрезентирани со една или комбинација од повеќе полиси. Меѓутоа, уште поважно е тоа што овозможува валидација за тоа кои податци може да се пристапат од различни корисници, преглед на корисниците кои би добиле дел од податоците во пакетот, како и можност за рано детектирање на податоците кои воопшто не се заштитени или во конфликт помеѓу различни претплатнички пакети.

За да се овозможи овој бизнис модел, потребно е најпрво да се адаптира *Intent Provider* компонентата така што ќе има пристап до информациите за претплатите. Една можна изведба е со зачувување на податоците за претплата во посебен именуван граф, во кој привилегии за запис би имала само апликацијата која им овозможува претплата на корисниците. На овој начин се овозможува конзистентност и сигурност во точноста на овие податоци, кои се клучни за генерирањето на приходите. Без оглед на протоколот за автентикација кој се користи, *Intent Provider* компонентата ќе го идентификува барателот на податоците и ќе ја збогати неговата намера со информации за неговите претплата од именуваниот граф за претплати. На овој начин, намерата ги собира заедно дистрибуирано зачуваните атрибути на барателот и му овозможува на *модулот за извршување* да одлучи кои податоци се дозволени за таа намера.

Дополнително, LDA платоформата овозможува проверка за податоците достапни за секој од претплатените корисници, како и персонализирани пакети за претплата, со што би можело да се дефинираат специјални пакети кои би важеле само за одреден барател. Валидацијата спречува грешки во споделувањето на податоците, што е критично поради можноста за брза репликација на податоците.

Во продолжение е прикажано како е возможно да се креираат различни типови на пакети за претплата на различни делови од поврзаните податоци зачувани во различни податочни множества, како и пристап во различни контексти и сценарија.

1. **Целосен пристап:** Овој пакет овозможува плаќање за пристап до детални и подобро поврзани податоци, како на пример линкови до други податочни множества, при што имаме груба грануларност на заштита на ниво на податочно множество или именуван граф, во зависност од начинот на чување на податоците. LDA платформата овозможува креирање на вакви пакети со користење на полиси кои ги заштитуваат податоците на ниво на податочно множество или именуван граф. Дополнително, информациите за барателот може да се поврзат со податочното множество кое ги чува неговите информации за претплата и да се провери дали ваквиот пристап е овозможен или не, односно дали ќе бидат достапни сите детали за корисникот или не.
2. **Временски пристап:** Пакет кој овозможува претплати за пристап на најновите верзии од податоците. Овој пакет се овозможува со контекстни полиси кои ги селектираат податоците креирани или ажурирани во даден временски период, сметано од денот на пристап.
3. **Пристап по потреба (on-demand):** Пакет кој овозможува плаќање за пристап по потреба за индивидуални прашања или конкретни податочни множества, односно овозможување за извршување на микро-трансакции. Овој пакет отвора сосема нови перспективи за заработка преку можноста на спогодување помеѓу различни софтверски агенти.
   Овој тип на пакети не е моментално овозможен во рамки на LDA платформата, но модуларната архитектура нуди можност за подршка во иднина. Во продолжение е опишано што треба да се промени за да се овозможи овој тип на пакети. При користење на овој тип на пристап, потребно е модификација на *Intent Provider* компонентата и *интерпретерот на резултатите*.
   Секое вакво барање треба прво во себе да содржи информација дека се бараат податоци за кои дополнително би се платило. Оваа информација треба *Intent Provider* компонентата да ја додаде во намерата .
   При секое добиено барање, *интерпретерот на резултатите* треба од намерата да заклучи дека треба да побара плаќање, па наместо да ги врати резултатите, треба да ја пресмета цената на побараните податоци и истата да ја врати преку специјален тип на одговор кон барателот, заедно со подржаните системи за плаќање. На овој начин LDA платформата би генерирала еден вид на фактура за пристапот до бараните податоци. Притоа, не се враќа дозволениот дел од побараните податоци, туку тие се користат за да се пресмета цената. Една стратегија за пресметка на цената е количината на податоци кои се враќаат како резултат, но можни се и доста други стратегии, како што се цена на податок заштитен со полиса[2].
   Доколку цената за податоците е прифатлива, барателот треба да изврши плаќање, по што треба да го проследи бројот на трансакцијата во наредно барање за пристап. Во овој чекор, *Intent Provider* компонентата треба да се интегрира со системот за плаќање и да ја валидира добиената трансакција. Во случај на валидна трансакција, треба да се вратат побараните податоци,

[2]Цената за податоци по полиса би барала модификација и на модулот за извршување, со што би требало да се врати која полиса го дозволува пристапот до податокот

а во спротивен случај да се прикаже соодветен статус и порака.

4. **Блоков пристап:** Пакет кој овозможува плаќања за пристап до податочните множества врз основа на време, на пример, дневена дозвола, или врз основа на бројот, фреквенцијата и конкурентноста на пристапот.
   Доколку е потребно ограничување на периодот на пристап, можно е да се дефинира времетраење на пакетот. Притоа полисата ќе провери дали датумот на претплата е постар од претплатеност рок за пристап. На овој начин се користи времето за пристап како клучна контекстна информација во дизајнот на полисата.
   Доколку е потребно да се користат бројот на пристапи или фреквенцијата, потребно е овие информации да се зачуваат од посебна компонента во некое податочно множество, по што податоците од тоа множество ќе се искористат за проверка дали корисникот ја исполнил квотата за која се претплатил или не. На овој начин се врши интеграција на историјата на користењето на системот со пакетите за претплата и заштитените податоци, што е овозможено од флексибилноста на јазикот за полиси кој се користи во LDA платформата.
   Пакетите базирани на конкурентен пристап исто така може да се моделираат, со инкорпорирање на сумарните статистики за користење на системот во полисите. На овој начин може да се дизајнираат полиси кои ќе дозволат пристап само кога системот има помлку од даден број на пристпи во даден временски интервал.
5. **Архивски пристап:** Со овој пакет се дозволува пристап само на оние кои се претплатиле на архивските податоци. Бидејќи архивските податоци може да се зачуваат на различни начини, флексибилноста на јазикот за полиси во LDA платформата овозможува детална спецификација за тоа кои се и од каде треба да се пристапат архивските податоци.
6. **Удобен пристап:** Пакет со кој се бара плаќања за пристап до податоците преку специфичен механизам. *Intent Provider* компонентата овозможува екстракција на прифатените механизми и формати на податоци, што дава можност да се провери дали корисникот е претплатен за бараниот начин. Дополнително, доколку корисникот сака да се претплати на одредени податоци во реално време, потребно е да се адаптира *интерпретерот на резултатите* така што ќе ги извести сите претплатени корисници заинтересирани за дадениот податок преку медиумот кој го избрале во ситуациите кога се модифицираат податоците.
7. **Freemium:** Пакет во кој корисниците можат да уживаат во слободен, но ограничен пристап до предефиниран примерок од податоците, но се наплаќа надоместок за продолжен пристап до останатите податоци. Овој пакет е подржан со тоа што сопствениците на податоци може да изберат кои податоци ќе бидат слободни и да ги објават преку соодветни полиси. На овој начин сопствениците може прво да ги тестираат јавните податоци, за потоа контролирано да ги објават.

6.2.2 Придружни програми

Како и во случајот на програмабилните веб интерфејси (API), креаторите на поврзани податоци продуцираат податочни текови на партнерите кои ги испорачуваат до крајните корисници преку апликации, во замена на дел од профитот кој го остваруваат. Овој начин на работа е овозможен на тој начин што специфичните партнери би им биле дадени соодветни привилегии за пристап. Полисите би се однесувале за партнерите и би дозволиле пристап кој е претходно договорен со нив.

6.2.3 Пазар

Производителите и администраторите на поврзаните податоци можат да обезбедат податоци за партнерот во замена за напредни и корисни сервиси. Администраторите имаат целосна контрола преку полисите да го променат пристапот доколку сервисот престане да биде достапен.

6.2.4 Сервиси од заедницата

Заштитата не е единствената услуга која ја нуди LDA платформата. Дополнително, со полисите кои дозволуваат пристап до агрегирани и анонимизирани податоци е возможно да се дозволи пристап за сервисите од заедницата, без да се загрози легитимитетот, интегритетот и приватноста на сопствениците на податоците. На овој начин се овозможува дефинирање на полиси со кои ќе се врши споделување на податоци за поголемо добро, без никој притоа да не биде повреден.

Глава 7

Заклучок

Во овој труд презентираме комплетна платформа за авторизација на поврзани податоци, развиена над флексибилен јазик за полиси, дизајниран специјално за заштита на поврзаните податоци. LDA платформата нуди целосна покриеност на принципите за дизајн на авторизација за *флексибилност*, додека употребата на стандардизиран SPARQL јазик за пребарување како основа на јазикот за полисите ја зголемува *разбирливоста* и го поедноставува процесот *одржување* со намалување на бројот на потребните полиси за барањата за заштита на податоците. *Модулот за управување со полисите* ги поддржува принципите *одржливост* и *валидација на точноста* во времето на дизајнот, додека *модулот за извршување* ја обезбедува имплицитната заштита на податоците.

Платформата LDA користи контекстни полиси со широка грануларност за заштита, покривајќи ресурси, тројки, именувани графови и податочни множества. Контекстот на авторизацијата е моделиран преку *Намерата*, која динамички се обезбедува преку модулот *Intent Provider* и ги содржи сите потребни докази за барателот и неговата околината, заедно со планираната акција и нејзините параметри. Користењето на *контекстот* овозможува дизајнирање на полиси кои се активираат во итни случаи, додека флексибилниот јазик за полисите е единствен во поддршката на полиси кои можат да ги изложат агрегирани и анонимизирани податоци кои може да служат за поголемо добро во сервисите за заедницата.

Модулот за управување со полисите поддржува проверка за време на дизајнот на полисите и значително го намалува напорот за одржување, времето и трошоците. Во овој процес, не само сопствениците можат да ги прегледаат сите податоци заштитени со полисата, но исто така можат и да добијат предлози за можните тест сценарија што резултираат со не-празни заштитени податоци. Покрај тоа, LDA платформата овозможува откривање и решавање на конфликти со користење на приоритети. Оваа теза, исто така, претставува како незаштитените податоци во платформата можат да бидат прегледани за време на дизајнирање

на полисите, кое ги води сопствениците на податоци кон целосна заштита на нивните податоци.

LDA платформата е имплементирана со користење на широко прифатените *Spring Framework* и *Apache Jena* библиотеки и има прифатливи перформанси во споредба со оптималното закодирано овластување. Модуларната архитектура на платформата овозможи развој на повеќе алгоритми за примена на авторизацијата на податоците, како и повеќе начини за извлекување и компонирање на дистрибуираните кориснички атрибути во процесот на креирањето на намерата. Сепак, за да постигнеме подобри перформанси, ја фокусираме нашата идна работа на воведување механизми за кеширање за привремените податочни множества. Со евалуацијата на пристапот со препишување на прашањата покажуваме дека наспроти верувањето дека овој пристап води кон подобри перформанси за авторизација на податоците, тоа не е случај кога е потребно комбинирање на поголем број на полиси заради поголема флексибилност на заштитата.

Анализата на поддржаните бизнис модели покажува дека LDA платформата охрабрува поврзување на податоците за соодветен повратен приход, што особено се истакнува со подршката на бизнис моделите за претплата. Дополнително, можноста за споделување на агрегирани и анонимизирани податоци оди во прилог на бизнис моделот за сервиси од заедницата, со што се охрабрува споделување на податоците кои се од јавно добро.

Со оглед на сите овие карактеристики, предложената LDA платформа е чекор напред кон пошироко прифаќање на поврзаните податоци како заедничка платформа за репрезентација и меѓусебно поврзување на хетерогени податоци. Дополнително, *модулот за управување со полисите* може да послужи како корисна алатка за валидација на заштитата во време на дизајнирање на полисите, дури и за платформи различни од нашата.

Литература

[1] Fabian Abel, Juri Luca De Coi, Nicola Henze, Arne Wolf Koesling, Daniel Krause, and Daniel Olmedilla. Enabling advanced and context-dependent access control in rdf stores. In *The Semantic Web*, pages 1–14. Springer, 2007.

[2] Almudena Alcaide, Esther Palomar, José Montero-Castillo, and Arturo Ribagorda. Anonymous authentication for privacy-preserving iot target-driven applications. *Computers & Security*, 37:111–123, 2013.

[3] Dean Allemang. Semantic web and the linked data enterprise. In *Linking enterprise data*, pages 3–23. Springer, 2010.

[4] Aleksandar Andreevski, Riste Stojanov, Milos Jovanovik, and Dimitar Trajanov. Semantic web integration with sparql autocomplete. In *The 12th International Conference on Informatics and Information Technologies*, pages 1–4, 2015.

[5] Gennady Andrienko, Aris Gkoulalas-Divanis, Marco Gruteser, Christine Kopp, Thomas Liebig, and Klaus Rechert. Report from dagstuhl: the liberation of mobile location data and its implications for privacy research. *ACM SIGMOBILE Mobile Computing and Communications Review*, 17(2):7–18, 2013.

[6] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. *The semantic web*, pages 722–735, 2007.

[7] Jean Bacon, David Eyers, Thomas FJ-M Pasquier, Jatinder Singh, Ioannis Papagiannis, and Peter Pietzuch. Information flow control for secure cloud computing. *IEEE Transactions on Network and Service Management*, 11(1):76–89, 2014.

[8] Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus. C-sparql: Sparql for continuous querying. pages 1061–1062, 2009.

[9] Payam Barnaghi, Wei Wang, Lijun Dong, and Chonggang Wang. A linked-data model for semantic sensor streams. pages 468–475, 2013.

[10] Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.

[11] Abraham Bernstein and Esther Kaufmann. Gino-a guided input natural language ontology editor. In *International Semantic Web Conference*, volume 2006, pages 144–157. Springer, 2006.

[12] Abraham Bernstein, Esther Kaufmann, and Christian Kaiser. Querying the semantic web with ginseng: A guided input natural language search engine. In *15th Workshop on Information Technologies and Systems, Las Vegas, NV*, pages 112–126, 2005.

[13] Christian Bizer and Richard Cyganiak. D2r server-publishing relational databases on the semantic web. In *Poster at the 5th international semantic web conference*, volume 175, 2006.

[14] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data-the story so far. *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, pages 205–227, 2009.

[15] Piero Bonatti and Daniel Olmedilla. Driving and monitoring provisional trust negotiation with metapolicies. In *Policies for Distributed Systems and Networks, 2005. Sixth IEEE International Workshop on*, pages 14–23. IEEE, 2005.

[16] Scott Brinker. Business models for linked data and web 3.0. *chiefmartec. com*, 10, 2010.

[17] Barbara Carminati, Elena Ferrari, Jianneng Cao, and Kian Lee Tan. A framework to enforce access control over data streams. *ACM Transactions on Information and System Security (TISSEC)*, 13(3):28, 2010.

[18] Barbara Carminati, Elena Ferrari, and Kian Lee Tan. Enforcing access control over data streams. In *Proceedings of the 12th ACM symposium on Access control models and technologies*, pages 21–30. ACM, 2007.

[19] Willy Chen and Heiner Stuckenschmidt. A model-driven approach to enable access control for ontologies. In *Wirtschaftsinformatik (1)*, pages 663–672, 2009.

[20] Simone Cirani, Marco Picone, Pietro Gonizzi, Luca Veltri, and Gianluigi Ferrari. Iot-oas: An oauth-based authorization service architecture for secure services in iot scenarios. *IEEE sensors journal*, 15(2):1224–1234, 2015.

[21] Luca Costabello, Serena Villata, Nicolas Delaforge, and Fabien Gandon. Linked data access goes mobile: Context-aware authorization for graph stores. In *LDOW-5th WWW Workshop on Linked Data on the Web-2012*, 2012.

[22] Luca Costabello, Serena Villata, Oscar Rodriguez Rocha, and Fabien Gandon. Access control for http operations on linked data. In *Extended Semantic Web Conference*, pages 185–199. Springer, 2013.

[23] Richard Cyganiak. A relational algebra for sparql. *Digital Media Systems Laboratory HP Laboratories Bristol. HPL-2005-170*, 35, 2005.

[24] Richard Cyganiak and Anja Jentzsch. Linking open data cloud diagram. *LOD Community (http://lod-cloud. net/)*, 12, 2011.

[25] Souripriya Das, Seema Sundara, and Richard Cyganiak. R2rml: Rdb to rdf mapping language. 2012.

[26] Christopher John Date. *An introduction to database systems*. Pearson Education India, 2006.

[27] Juri Luca De Coi, Daniel Olmedilla, Piero A Bonatti, and Luigi Sauro. Protune: A framework for semantic web policies. In *International Semantic Web Conference (Posters & Demos)*, volume 401, page 128, 2008.

[28] Dorothy E Denning. A lattice model of secure information flow. *Communications of the ACM*, 19(5):236–243, 1976.

[29] Sebastian Dietzold and Sören Auer. Access control on rdf triple stores from a semantic wiki perspective. In *ESWC Workshop on Scripting for the Semantic Web*. Citeseer, 2006.

[30] L Dodds. Thoughts on linked data business models, 2010.

[31] Sébastien Ferré. Sparklis: an expressive query builder for sparql endpoints with guidance in natural language. *Semantic Web*, 8(3):405–418, 2017.

[32] Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext transfer protocol–http/1.1. Technical report, 1999.

[33] Tim Finin, Anupam Joshi, Lalana Kagal, Jianwei Niu, Ravi Sandhu, William Winsborough, and Bhavani Thuraisingham. R owl bac: representing role based access control in owl. In *Proceedings of the 13th ACM symposium on Access control models and technologies*, pages 73–82. ACM, 2008.

[34] Giorgos Flouris, Irini Fundulaki, Maria Michou, and Grigoris Antoniou. Controlling access to rdf graphs. In *Future Internet Symposium*, pages 107–117. Springer, 2010.

[35] Stefano Franzoni, Pietro Mazzoleni, Stefano Valtolina, and Elisa Bertino. Towards a fine-grained access control model and mechanisms for semantic databases. In *IEEE International Conference on Web Services (ICWS 2007)*, pages 993–1000. IEEE, 2007.

[36] Paul Fremantle and Benjamin Aziz. Oauthing: privacy-enhancing federation for the internet of things. 2016.

[37] Paul Fremantle, Jacek Kopeckỳ, and Benjamin Aziz. Web api management meets the internet of things. In *European Semantic Web Conference*, pages 367–375. Springer, 2015.

[38] Paul Gearon, Alexandre Passant, and Axel Polleres. Sparql 1.1 update. *Working draft WD-sparql11-update-20110512, W3C (May 2011)*, 2013.

[39] Ana Gjorgjevik, Riste Stojanov, and Dimitar Trajanov. Semccm: Course and competence management in learning management systems using semantic web technologies. In *Proceedings of the 10th International Conference on Semantic Systems*, SEM '14, pages 140–147, New York, NY, USA, 2014. ACM.

[40] Ana Gjorgjevikj, Riste Stojanov, and Dimitar Trajanov. Enhancing text-based relatedness measures with semantic web data. In Georgi Stojanov and Andrea Kulakov, editors, *ICT Innovations 2016: Cognitive Functions and Next Generation ICT Systems*, pages 182–192, Cham, 2018. Springer International Publishing.

[41] Sergio Gusmeroli, Salvatore Piccione, and Domenico Rotondi. A capability-based security approach to manage access control in the internet of things. *Mathematical and Computer Modelling*, 58(5):1189–1205, 2013.

[42] Dick Hardt. The oauth 2.0 authorization framework. 2012.

[43] Steve Harris, Andy Seaborne, and Eric Prud'hommeaux. Sparql 1.1 query language. *W3C Recommendation*, 21, 2013.

[44] Tom Heath and Christian Bizer. Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: theory and technology*, 1(1):1–136, 2011.

[45] James Hollenbach, Joe Presbrey, and Tim Berners-Lee. Using rdf metadata to enable access control on the social semantic web. In *Proceedings of the Workshop on Collaborative Construction, Management and Linking of Structured Knowledge (CK2009)*, volume 514, 2009.

[46] Ian Horrocks, Peter F Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean, et al. Swrl: A semantic web rule language combining owl and ruleml. *W3C Member submission*, 21:79, 2004.

[47] Chunye Hu, Jie Zhang, and Qiaoyan Wen. An identity-based personal location system with protected privacy in iot. In *Broadband Network and Multimedia Technology (IC-BNMT), 2011 4th IEEE International Conference on*, pages 192–195. IEEE, 2011.

[48] David Huynh and Stefano Mazzocchi. Google refine, 2011.

[49] Lalana Kagal, Tim Finin, and Anupam Joshi. A policy language for a pervasive computing environment. In *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*, pages 63–74. IEEE, 2003.

[50] Shahan Khatchadourian and Mariano Consens. Explod: summary-based exploration of interlinking and rdf usage in the linked open data cloud. *The Semantic Web: Research and Applications*, pages 272–287, 2010.

[51] Sabrina Kirrane. *Linked data with access control*. PhD thesis, 2015.

[52] Sabrina Kirrane, Alessandra Mileo, and Stefan Decker. Access control and the resource description framework: A survey. *Semantic Web*, 8(2):311–352, 2017.

[53] Graham Klyne and Jeremy J Carroll. Resource description framework (rdf): Concepts and abstract syntax. 2006.

[54] Vladimir Kolovski, James Hendler, and Bijan Parsia. Analyzing web access control policies. In *Proceedings of the 16th international conference on World Wide Web*, pages 677–686. ACM, 2007.

[55] Manolis Koubarakis and Kostis Kyzirakos. Modeling and querying metadata in the semantic sensor web: The model strdf and the query language stsparql. pages 425–439, 2010.

[56] Jian Li and William K Cheung. Query rewriting for access control on semantic web. In *Workshop on Secure Data Management*, pages 151–168. Springer, 2008.

[57] Nuno Lopes, Sabrina Kirrane, Antoine Zimmermann, Axel Polleres, and Alessandra Mileo. *A Logic Programming approach for Acess Control over RDF*. PhD thesis, 2012.

[58] Brian McBride. Jena: A semantic web toolkit. *IEEE Internet computing*, 6(6):55, 2002.

[59] H Michael and Lipner Steve. The security development lifecycle sdl: A process for developing demonstrably more secure software. *Publishing House of Electronics Industry*, 2008.

[60] Martin Mitrevski, Milos Jovanovik, Riste Stojanov, and Dimitar Trajanov. Open university data. In *The 9th International Conference on Informatics and Information Technologies*. Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University in Skopje, Macedonia, 2012.

[61] Hannes Muhleisen, Martin Kost, and Johann-Christoph Freytag. Swrl-based access policies for linked data. *Procs of SPOT*, 80, 2010.

[62] Bojan Najdenov, Goran Petkovski, Milos Jovanovik, Riste Stojanov, and Dimitar Trajanov. Automated linked data generation from the transport administration domain. In *Telecommunications Forum Telfor (TELFOR), 2015 23rd*, pages 827–830. IEEE, 2015.

[63] Rimma V Nehme, Elke A Rundensteiner, and Elisa Bertino. A security punctuation framework for enforcing access control on streaming data. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 406–415. IEEE, 2008.

[64] Ricardo Neisse, Gary Steri, and Gianmarco Baldini. Enforcement of security policy rules for the internet of things. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2014 IEEE 10th International Conference on*, pages 165–172. IEEE, 2014.

[65] Ricardo Neisse, Gary Steri, Igor Nai Fovino, and Gianmarco Baldini. Seckit: a model-based security toolkit for the internet of things. *Computers & Security*, 54:60–76, 2015.

[66] Aimaschana Niruntasukrat, Chavee Issariyapat, Panita Pongpaibool, Koonlachat Meesublak, Pramrudee Aiumsupucgul, and Anun Panya. Authorization mechanism for mqtt-based internet of things. In *Communications Workshops (ICC), 2016 IEEE International Conference on*, pages 290–295. IEEE, 2016.

[67] Said Oulmakhzoune, Nora Cuppens-Boulahia, Frédéric Cuppens, and Stephane Morucci. fquery: Sparql query rewriting to enforce data confidentiality. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 146–161. Springer, 2010.

[68] Ankur Padia, Tim Finin, and Anupam Joshi. Attribute-based fine grained access control for triple stores. In *3rd Society, Privacy and the Semantic Web-Policy and Technology workshop, 14th International Semantic Web Conference*, 2015.

[69] Stavros Papadopoulos, Yin Yang, and Dimitris Papadias. Cads: Continuous authentication on data streams. In *Proceedings of the 33rd international conference on Very large data bases*, pages 135–146. VLDB Endowment, 2007.

[70] Vasko Popovski, Bojan Kostadinov, Riste Stojanov, Igor Mishkovski, and Dimitar Trajanov. Web-based disaster and crisis management system. 2013.

[71] Eric Prud'Hommeaux, Andy Seaborne, et al. Sparql query language for rdf. *W3C recommendation*, 15, 2008.

[72] Michael Rappa. Business models on the web. *Available at Managing the Digital Enterprise website: http://digitalenterprise. org*, 2003.

[73] Eric Rescorla and Nagendra Modadugu. Datagram transport layer security version 1.2. 2012.

[74] Tatyana Ryutov, Tatiana Kichkaylo, and Robert Neches. Access control policies for semantic networks. In *Policies for Distributed Systems and Networks, 2009. POLICY 2009. IEEE International Symposium on*, pages 150–157. IEEE, 2009.

[75] Owen Sacco and John G Breslin. Ppo & ppm 2.0: Extending the privacy preference framework to provide finer-grained access control for the web of data. In *Proceedings of the 8th International Conference on Semantic Systems*, pages 80–87. ACM, 2012.

[76] Owen Sacco and Alexandre Passant. A privacy preference manager for the social semantic web. In *SPIM*, pages 42–53, 2011.

[77] Ravi S Sandhu and Pierangela Samarati. Access control: principle and practice. *IEEE communications magazine*, 32(9):40–48, 1994.

[78] Reijo M Savola and Habtamu Abie. Metrics-driven security objective decomposition for an e-health application with adaptive security management. In *Proceedings of the International Workshop on Adaptive Security*, page 6. ACM, 2013.

[79] Carlo Scarioni. *Pro Spring Security*. Apress, 2013.

[80] François-Paul Servant. Linking enterprise data. In *LDOW*, 2008.

[81] Jatinder Singh and Jean M Bacon. On middleware for emerging health services. *Journal of Internet Services and Applications*, 5(1):1–19, 2014.

[82] Jatinder Singh, Thomas FJ-M Pasquier, Jean Bacon, and David Eyers. Integrating messaging middleware and information flow control. In *Cloud Engineering (IC2E), 2015 IEEE International Conference on*, pages 54–59. IEEE, 2015.

[83] Manu Sporny, Toby Inkster, Henry Story, Bruno Harbulot, and Reto Bachmann-Gmür. Webid 1.0: Web identification and discovery. *Editor's draft, W3C*, 2011.

[84] Riste Stojanov, Marjan Georgiev, Vladimir Zdraveski, Milos Jovanovik, and Dimitar Trajanov. Live objects-collaborative window in the corporate documents. In *New Trends in Database and Information Systems II*, pages 71–81. Springer, 2015.

[85] Riste Stojanov, Sasho Gramatikov, Igor Mishkovski, and Dimitar Trajanov. Linked data authorization platform. *IEEE Access*, 2017.

[86] Riste Stojanov and Milos Jovanovik. Authorization proxy for sparql endpoints. In Dimitar Trajanov and Verica Bakeva, editors, *ICT Innovations 2017: Data-Driven Innovation. 9th International Conference, ICT Innovations 2017, Skopje, Macedonia, September 18-23, 2017, Proceedings*, pages 205–218, Cham, 2017. Springer International Publishing.

[87] Henry Story, Bruno Harbulot, Ian Jacobi, and Mike Jones. Foaf+ ssl: Restful authentication for the social web. In *Proceedings of the First Workshop on Trust and Privacy on the Social and Semantic Web (SPOT2009)*, 2009.

[88] Alessandra Toninelli, Rebecca Montanari, Lalana Kagal, and Ora Lassila. Proteus: A semantic context-aware adaptive policy model. In *Eighth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'07)*, pages 129–140. IEEE, 2007.

[89] Dimitar Trajanov, Riste Stojanov, Milos Jovanovik, Vladimir Zdraveski, Petar Ristoski, Marjan Georgiev, and Sonja Filiposka. Semantic sky: a platform for cloud service integration based on semantic web technologies. In *Proceedings of the 8th International Conference on Semantic Systems*, pages 109–116. ACM, 2012.

[90] Slobodanka Trajkova, Riste Stojanov, and Dimitar Trajanov. Semantic web access control aspects. In *The 12th International Conference on Informatics and Information Technologies*, pages 243–245, 2015.

[91] Michalis Vafopoulos. A framework for linked data business models. In *Informatics (PCI), 2011 15th Panhellenic Conference on*, pages 95–99. IEEE, 2011.

[92] David Wood. *Linking enterprise data*. Springer Science & Business Media, 2010.