КУРСОВ ПРОЕКТ

Онтология на криминална случка

Част 2

**Факултет по Математика и Информатика**  
**Студент:** Борислав Стоянов Марков  
**Факултетен номер:** **0MI3400048**  
**Учебен план:** **Изкуствен Интелект (редовно, магистър)**  
Курс: **Курс 1**; Група: **Група 1**  
**Активен период**: 2021/2022 летен, магистри  
**Дисциплина**: Бази от знания

Дата: 11.06.2021г.

# 1. Съдържание

[1. Съдържание 2](#_Toc92064595)

[2. Увод 2](#_Toc92064596)

[3. Сапунена криминална случка 3](#_Toc92064597)

[4. Онтология на затворен свят 4](#_Toc92064598)

[4.1 Елементи на началната онтология 4](#_Toc92064599)

[4.2. Вторична онтология на базата на правила 8](#_Toc92064600)

[4.3 Изводи със SPARQL[4] 10](#_Toc92064601)

[5. Реализация на проекта 11](#_Toc92064602)

[6. Идеи за бъдещо развитие и подобрения 12](#_Toc92064603)

[7. Източници и използвана литература 13](#_Toc92064604)

[Приложения 13](#_Toc92064605)

[1. Сорс код (Source code) 13](#_Toc92064606)

# 2. Увод

Идеята за проекта е взета от сапунената опера от книгата „Knowledge Representation and Reasoning“ (Brachman и Levesque)[1] и е допълнение на онтологията направена от първи семестър. Проектът моделира събитие от измислен свят. В града се е случила криминална случка. Има изтичане на информация от една софтуерна фирма и градският детектив разпитва заподозрените. Предстои да опишем това с онтология. Ще проследим кой кого познава и така ще помогнем на детектива да стигне до заподозрените. Описателният език, който ще използваме е OWL DL[3], използваната база за графи е GraphDB. Ще се използва машината за изводи (Reasoning) предлагана от OWL и GraphDB. В този проект няма да използваме наши правила, както е в предишният проект, а изводите ще се правят на база на SPARQL заявки. Всички данни са въведени ръчно (поради неголемият обем) и са в TURTLE формат. Така ще бъдат импортирани в базата GraphDB, а в последствие ще направим няколко извода със SPARQL от уеб интерфейса предлаган от базата.

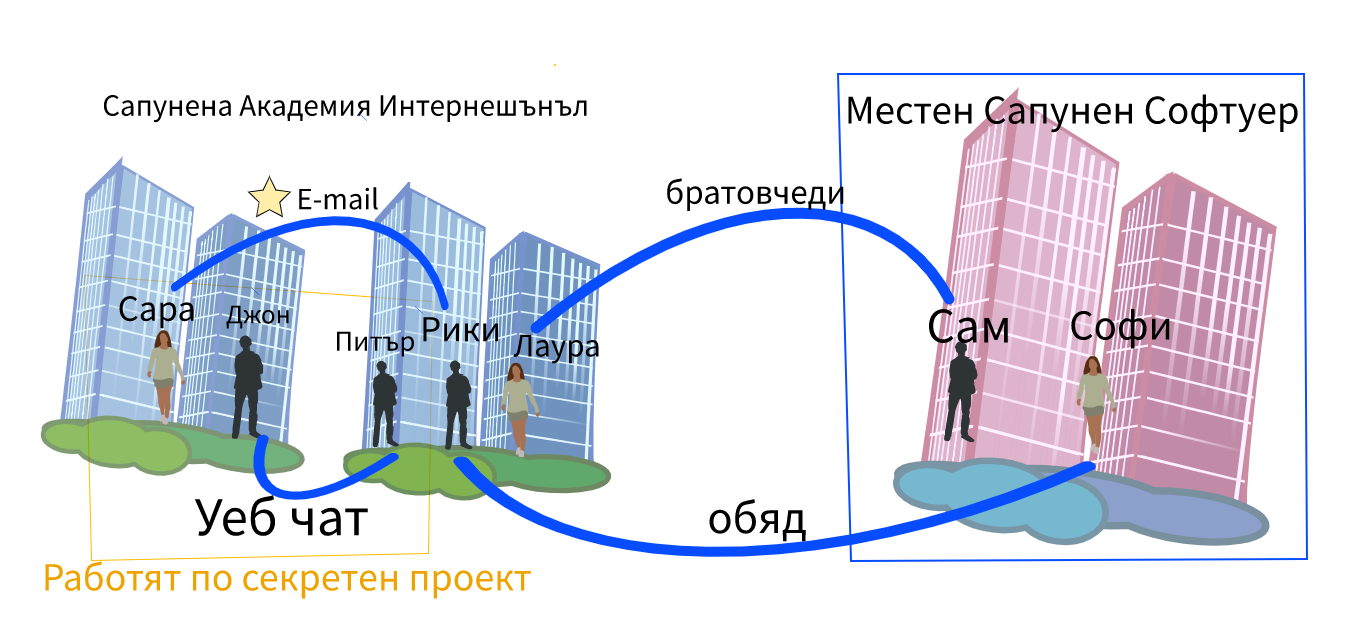
Целите на проекта са да се покаже как може да се моделират знанията от реалния свят и да се правят изводи, когато това не е очевидно. Областта може да е криминалистика, но не се ограничаваме само до нея.

# 3. Сапунена криминална случка

Джон(John) работи във софтуерната фирма „Сапунена Академия Интернешънъл“ (Soap Academy International). Проектът на Джон е конфиденциален и се казва „Робо-Сапунена Стратегия“ (Robo Soap Strategy). Джон има колега, с когото добре се познават, който се казва Питър (Peter) и работят по един проект. Пишат си постоянно във фирмения чат. Джон и Питър имат колега Рики (Riki), който се познава с Питър, но не се познават с Джон лично. Джон има също и колежка Сара(Sara), която работи по същият проект като него. Рики не работи по проекта „Робо-Сапунена Стратегия“. Лаура тайно харесва Джон, но е в друг офис, не работят заедно и се виждат само по фирмени събирания или на онлайн срещи. Джон не харесва Лаура и не откликва на чувствата й. От друга страна Рики харесва Лаура, която не му обръща внимание. Рики е готов на всичко да я заинтереува. От друга фирма „Местен Сапунен Софтуер“ (Domestic Soap Software), конкурент на Сапунена Академия Интернешънъл, са получили информация за проекта Робо-Сапунена Стратегия и информацията е обявена публично. Местният детектив разпитва участниците за случките през последната седмица и установява, че Джон не работи в една сграда с Питър. Питър работи в една сграда с Лаура и Рики. Лаура харесва Джон. Питър харесва Лаура, но Лаура него – не . Рики работи зад гърба на Питър и може да види екрана му. Питър не заключва монитора си, когато ходи до кухнята в офиса. Вечерта преди да изтече инфромацията очевидци са видели Лаура да говори с братовчед си Сам (Sam). Сам работи за конкурентната фирма „Местен Сапунен Софтуер“. Софи(Sophie) и Сам са колеги. Лаура и Рики постоянно си говорят в офиса в кухнята.

Засечен е email изпратен от Сара до Рики. Има свидетели, които са видели Рики и Софи да обядват заедно. Рики харесва Софи, но тя не откликва на чувствата му.

На Рисунка 3.1 е онагледена случката. Изобразени са само базовите факти, а допълнителните ще бъдат изложени по-надолу поетапно.



Рисунка 3.1

# 4. Онтология на затворен свят

За да моделираме случката можем да използваме OWL DL. Ще представим таблично класовете и атрибутите към тях, описващи само това, което ние е необходимо за да намерим теча на информация. В началото ние имаме базовите класове, и към тях добавяме още два класа LeakInfoSourcePeople и LeakInfoDestinationPeople, които са рестрикции по дадени правила. Именно чрез тях ще можем лесно да формулираме някои заявки.

## 4.1 Елементи на онтологията

В таблица 4.1.1 са дадени класовете на началната онтология, според описаната история в т.3.

|  |  |
| --- | --- |
| OWL клас | Родителски клас и описание |
| Person | Thing |
| Man | Person |
| Woman | Person, complementOf (Man) |
| Company | Thing |
| Project | Thing |
| Event | Thing |
| WitnessedEvent | Event |
| Conversation | WitnessedEvent |
| OnlineConversation | Conversation |
| Email | OnlineConversation |
| LeakInfoSourcePeople | owl:equivalentClass  [rdfs:subClassOf sc:Person ;  rdf:type owl:Restriction;  owl:hasValue sc:roboSoapStrategyProject;  owl:onProperty sc:worksFor] |
| LeakInfoDestinationPeople | owl:equivalentClass  [ rdf:type owl:Restriction;  rdfs:subClassOf sc:Person ;  owl:hasValue sc:domesticSoapSoftwareCompany;  owl:onProperty sc:worksFor] |

Таблица 4.1.1

В таблица 4.1.2 са описани свойствата на онтологията. Свойството, което е особено важно за изводите на детектива е „**canShareSensitiveInfo“.**

|  |  |  |  |
| --- | --- | --- | --- |
| Property | Domain | Range | Characteristics |
| worksFor | Person | Company, Project |  |
| hasConversationParticipant | Conversation | Person | *inverseOf* isPartOfConversation |
| isPartOfConversation | Person | Conversation | *inverseOf* hasConversationParticipant |
| canSeeMonitor | Person | Person | - |
| isGoodFriendTo | Person | Person | *symmetric* |
| **canShareSensitiveInfo** | Person | Person | TransitiveProperty |
| loveButNotLoved | Person | Person | *functional* |

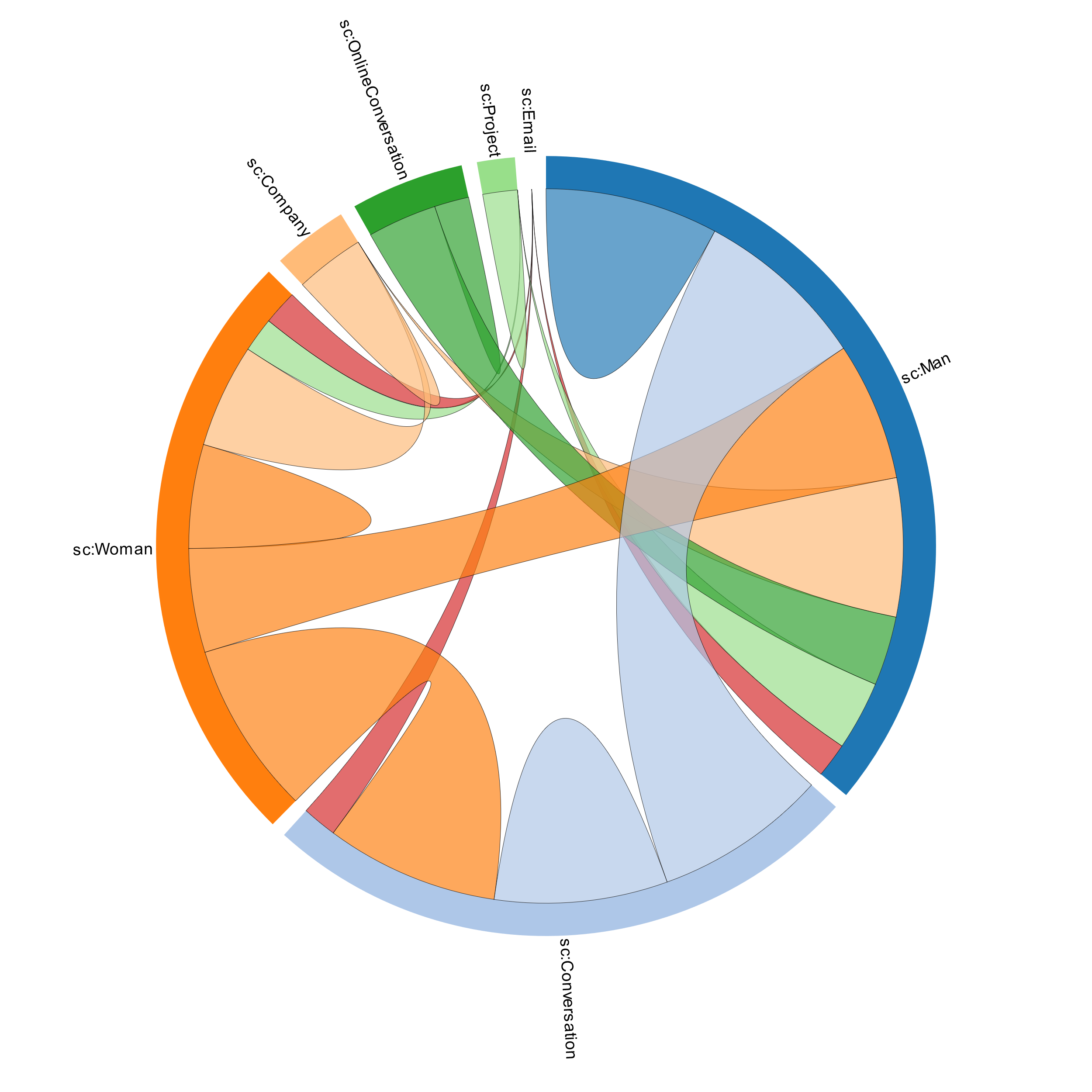
Таблица 4.1.2

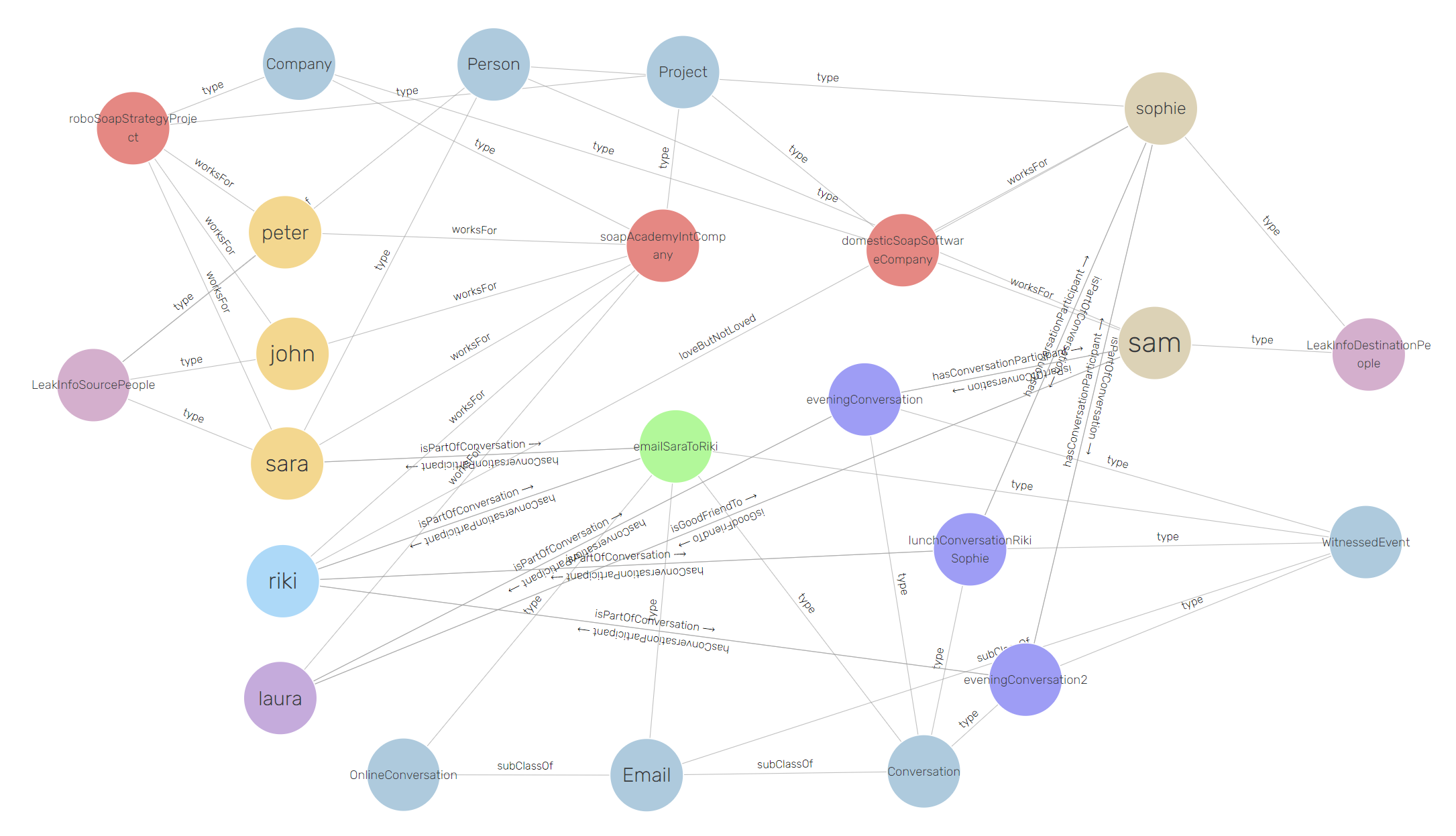
В Таблица 4.1.3 са дадени индивидите използвани в случката и техните свойства. Ограничили сме описанието само до ниво, за което можем да направим съответния извод. Например това, кои хора работят в една сграда, не е необходимо да се описва.

|  |
| --- |
| sc:john rdf:type sc:Man ;  sc:worksFor sc:roboSoapStrategyProject,sc:soapAcademyIntCompany. |
| sc:sara rdf:type sc:Woman ;  sc:worksFor sc:roboSoapStrategyProject, sc:soapAcademyIntCompany. |
| sc:peter rdf:type sc:Man;  sc:loveButNotLoved sc:laura ;  sc:worksFor sc:roboSoapStrategyProject,sc:soapAcademyIntCompany. |
| sc:riki rdf:type sc:Man ;  sc:canSeeMonitor sc:peter ;  sc:worksFor sc:soapAcademyIntCompany . |
| sc:sam rdf:type sc:Man ;  sc:worksFor sc:domesticSoapSoftwareCompany . |
| sc:sophie rdf:type sc:Woman ;  sc:worksFor sc:domesticSoapSoftwareCompany . |
| sc:soapAcademyIntCompany  rdf:type sc:Company . |
| sc:domesticSoapSoftwareCompany  rdf:type sc:Company . |
| sc:roboSoapStrategyProject  rdf:type sc:Project . |
| sc:eveningConversation  rdf:type sc:Conversation ;  sc:hasConversationParticipant sc:sam , sc:laura . |
| sc:kitchenConversation  rdf:type sc:Conversation ;  sc:hasConversationParticipant sc:riki , sc:laura . |
| sc:peterAndJohnChat rdf:type sc:OnlineConversation ;  sc:hasConversationParticipant sc:john , sc:peter . |
| sc:emailSaraToRiki  rdf:type sc:Email ;  sc:hasConversationParticipant sc:sara , sc:riki . |
| sc:lunchConversationRikiSophie  rdf:type sc:Conversation ;  sc:hasConversationParticipant sc:riki , sc:sophie . |

Таблица 4.1.3

Ще представим и визуализация на онтологията на фигура 4.1.4.





Фигура 4.1.4

За да се направи качествен извод, от къде би могло да изтече информация, са необходими определени заявки на SPARQL.

## 4.2. Изводи със SPARQL[4] към базата

Като извод искаме да намерим всички които биха могли да споделят информация в разговор. За целта знаем, е всички хора, които имат секретната информация са от клас sc:LeakInfoSourcePeople. Нека да видим следната заявка.

|  |
| --- |
| # sparql/sparql1.sparql  PREFIX sc: <urn:soap/crime/ontology#>  select \* where {  ?p1 a sc:LeakInfoSourcePeople .  ?p1 sc:isPartOfConversation ?c.  ?p2 sc:isPartOfConversation ?c.  FILTER (!sameTerm(?p1, ?p2))  }  p1 c p2  1 sc:peter sc:peterAndJohnChat sc:john  2 sc:john sc:peterAndJohnChat sc:peter  3 sc:sara sc:emailSaraToRiki sc:riki |

В случая не ни води надалеч. Нека да изведем и графично какво представлява резултата, използвайки бутона VISUAL от уеб-клиента за GraphDB.

|  |
| --- |
| # sparql/sparql2.sparql  PREFIX sc: <urn:soap/crime/ontology#>  construct {  ?p1 a sc:Person .  ?p2 a sc:Person.  } where {  ?p1 a sc:LeakInfoSourcePeople .  ?p1 sc:isPartOfConversation ?c.  ?p2 sc:isPartOfConversation ?c.  FILTER (!sameTerm(?p1, ?p2))  } |

Нека да разширим заявката и проследим и всички които са участвали в разговори тръгващи от хора с клас sc:LeakInfoSourcePeople.

|  |
| --- |
| # sparql/sparql3.sparql PREFIX sc: <urn:soap/crime/ontology#>  select \* where {  ?p1 a sc:LeakInfoSourcePeople .  ?p1 (sc:isPartOfConversation/sc:hasConversationParticipant)\* ?p2  FILTER (!sameTerm(?p1, ?p2)) }  p1 p2  1 sc:peter sc:john  2 sc:john sc:peter  3 sc:sara sc:riki  4 sc:sara sc:laura  5 sc:sara sc:sophie  6 sc:sara sc:sam |

Сега можем да сложим и ограничение да ни води само до хора от класа „sc:LeakInfoDestinationPeople“ за да намалим резултатите.

|  |
| --- |
| # sparql/sparql4.sparql  PREFIX sc: <urn:soap/crime/ontology#>  select \* where {  ?p1 a sc:LeakInfoSourcePeople .  ?p2 a sc:LeakInfoDestinationPeople.  ?p1 (sc:isPartOfConversation/sc:hasConversationParticipant)\* ?p2  }  p1 p2  1 sc:sara sc:sam  2 sc:sara sc:sophie |

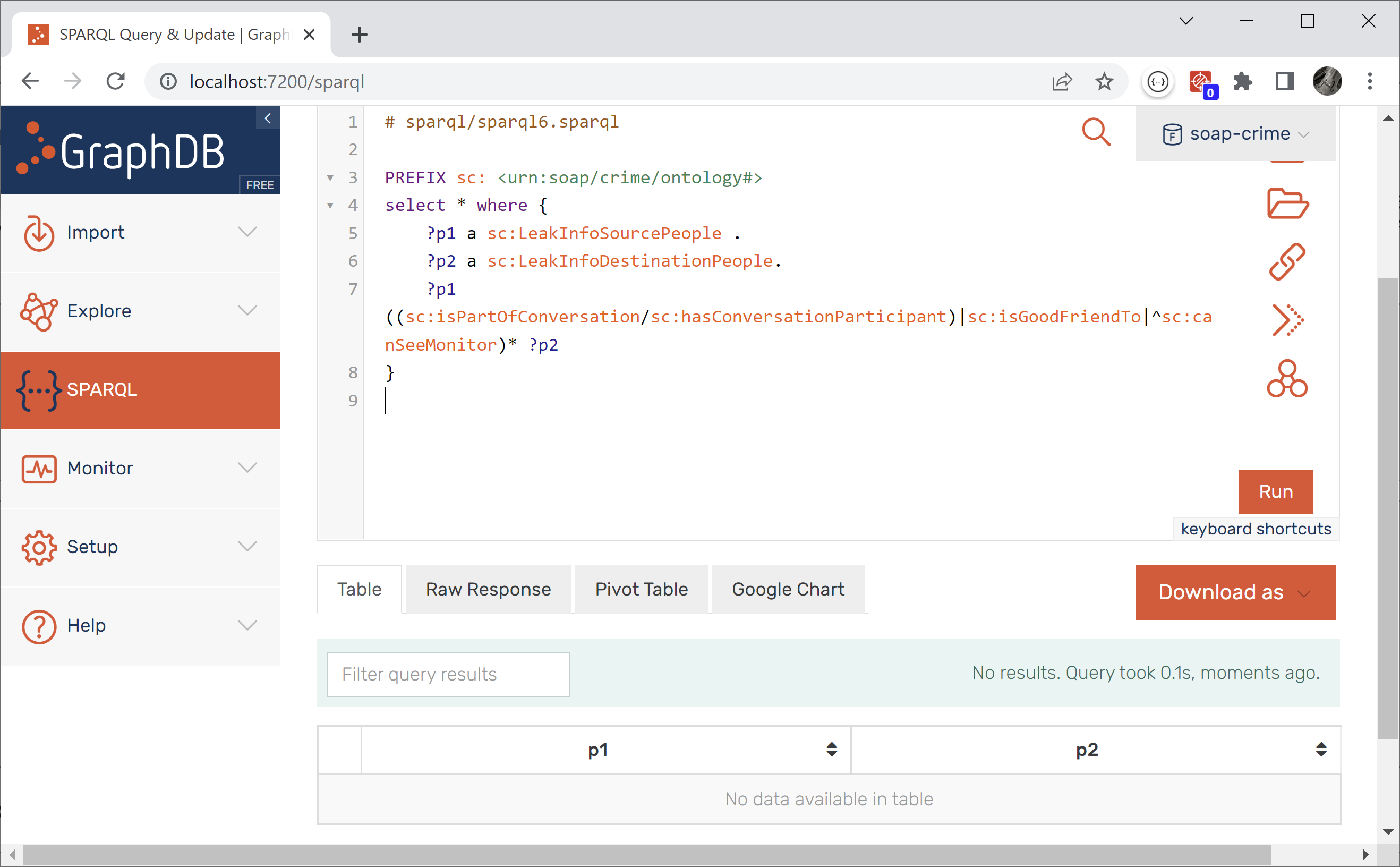
Тук можем да отбележим, че не включваме всички възможности да се пренесе информация. Какво сме пропуснали? Детективът има отбелязани само разговори, които са със свидетели или записани, но може да има и такива, за които никой не знае. Това са разговорите между добрите приятели. Така можем да включим още едно разклонение за проследяване на информацията между ?p1 и ?p2.

|  |
| --- |
| # sparql/sparql5.sparql  PREFIX sc: <urn:soap/crime/ontology#>  select \* where {  ?p1 a sc:LeakInfoSourcePeople .  ?p2 a sc:LeakInfoDestinationPeople.  ?p1 ((sc:isPartOfConversation/sc:hasConversationParticipant)|sc:isGoodFriendTo)\* ?p2  }  p1 p2  1 sc:sara sc:sam  2 sc:sara sc:sophie |

Резултатът не се промени. Има и още един начин да се предаде информация. Колегите могат да видят монитора на тези, които не заключват екраните си. Можем да включим и това като допълнително условие, но със символа „^“ защото търсим обратно насочена дъга за пренос на инфорация, а именно към ?p2.

|  |
| --- |
| # sparql/sparql6.sparql  PREFIX sc: <urn:soap/crime/ontology#>  select \* where {  ?p1 a sc:LeakInfoSourcePeople .  ?p2 a sc:LeakInfoDestinationPeople.  ?p1 ((sc:isPartOfConversation/sc:hasConversationParticipant)|sc:isGoodFriendTo|**^sc:canSeeMonitor**)\* ?p2  }  p1 p2  1 sc:sara sc:sam  2 sc:peter sc:sam  3 sc:john sc:sam  4 sc:sara sc:sophie  5 sc:peter sc:sophie  6 sc:john sc:sophie |

Излязоха твърде много резултати, това са твърде много хипотези, които трябва да бъдат проверени. Нека да изключим мащината за изводи на базата и да видим какво ще се получи като резултат.



Разбира се това е празен резултат. Тоест това ни дава увереност, че машината за изводи ни помага реално да разберем какво се е случило.

Сега искаме да разберем през какви разговори е излязла информацията на от първа ръка, това вероятно ще ни помогне да елиминираме част от хипотезите.

|  |
| --- |
| # sparql/sparql7.sparql  PREFIX sc: <urn:soap/crime/ontology#>  select ?p1 ?firstHop ?p2  where {  ?p1 a sc:LeakInfoSourcePeople .  ?p2 a sc:LeakInfoDestinationPeople.  ?p1 sc:isPartOfConversation ?firstHop .  ?firstHop ((sc:hasConversationParticipant)|sc:isGoodFriendTo|^sc:canSeeMonitor)\* ?p2 .  }  p1 firstHop p2  1 sc:peter sc:peterAndJohnChat sc:sam  2 sc:john sc:peterAndJohnChat sc:sam  3 sc:sara **sc:emailSaraToRiki** sc:sam |

Тук можем да видим, че без да имаме предвид изтичане на информация чрез приятелството и гледането на монитор, имаме един имейл. Детектива се заема да попита администратора на имейл сървъра и този имейл може да бъде проверен дали има чувствителна информация. Няма проблем с имейла и не е имало теч от него, така че можем да го елиминираме като изходящи имейли от първа ръка.

## 4.3 Изводи със SPARQL[4]

Вече имаме двете онтологии с индивидите и насложените атрибути, като втората онтология съдържа всички елементи на първата. Сега можем да изпозваме двата нови класа и да направим заключения.

Първо дали има възможно изтичане ще направим със следната заявка от таблица 4.3.1.

|  |
| --- |
| PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX owl: <http://www.w3.org/2002/07/owl#> PREFIX sc: <urn:soap/crime/ontology#> SELECT ?insider ?outsider WHERE { ?insider a sc:LeakInfoSourcePeople ;  sc:canShareSensitiveInfoIndirectTo ?outsider .  ?outsider a sc:LeakInfoDestinationPeople . } |

Таблица 4.3.1

Тук просто потвърждаваме че има възможен теч между хората от LeakInfoSourcePeople към LeakInfoDestinationPeople. Заявката връща три резултата за възможна индиректна (транзитивна) връзка:

1. **insider=john, outsider=sam;**
2. **insider=sara, outsider=sam;**
3. **insider=peter, outsider=sam**;

С втора заявка можем да зададем въпроса към системата къде от секретният проект именно имаме възможен и вероятен пробив. Селектираме всички хора от секретния проект от една страна и всички други възможни хора, които не работят за този проект. Заявката е дадена в Таблица 4.3.2.

|  |
| --- |
| PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX owl: <http://www.w3.org/2002/07/owl#> PREFIX sc: <urn:soap/crime/ontology#> SELECT ?insider ?outsider  WHERE { ?insider a sc:LeakInfoSourcePeople ;  sc:canShareSensitiveInfoTo ?outsider .  MINUS {  ?outsider sc:worksFor sc:roboSoapStrategyProject  } } |

Таблица 4.3.2

Върнатите резултати са **insider=peter**, **outsider=riki**. Тоест това е вече криминално деяние и предстои на детектива да установи подробностите.

Третата заявка ни дава хората, между които би могло да изтече информация от фирмата и е дадена в Таблица 4.3.3.

|  |
| --- |
| PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX owl: <http://www.w3.org/2002/07/owl#> PREFIX sc: <urn:soap/crime/ontology#> SELECT DISTINCT ?insider ?outsider  WHERE { ?x a sc:LeakInfoSourcePeople ;   sc:canShareSensitiveInfoIndirectTo ?insider .   ?insider sc:canShareSensitiveInfoTo ?outsider.   ?insider sc:worksFor sc:soapAcademyIntCompany.   MINUS {  ?outsider sc:worksFor sc:soapAcademyIntCompany  } } |

Таблица 4.3.3

Тук селектираме вече евентуална тройка от хора. А именно – някой който може да сподели информация от проекта във фирмата, друг, който работи във фирмата и му е споделено нещо и трети който му е споделено нещо, но не работи за въпросната фирма. Извеждаме само двамата на границата работещ или не във фирмата soapAcademyIntCompany. Резултата е **insider=laura**, **outsider=sam**. Това също е криминално деяние и предстои на детектива да установи детайлите.

# 5. Реализация на проекта

Общата структура на проекта е реализирана с Java, езикът за билдване на проекта е MAVEN. Линк към сорс кода е качен в гитхъб (Вж. Приложение 1) и е неразделна част от този документ. Структурата на приложението е дадена на фигура 5.1. Резултатът от едно изпълнение е даден на фигура в 5.2. Използваната развойа среда е IntelliJ.

|  |  |
| --- | --- |
| Фигура 5.1 | Фигура 5.2 |

Главният изпълним файл е App.java и в него се извикват статични методи от останалите класове. Подробни инструкции са дадени в README.md файла. Например първоначално класовете и инстанциите се създават по следния начин:

OntModel ont1 = CriminalStory.*describeCriminalStory*();

Втората стъпка е правенето на изводи и вторична онтология:

OntModel ont2 = CriminalInference.*makeDeductions*(ont1);

Последната стъпка е извикване на SPARQL заявки и печат на конзолата:

CriminalDeduction.*criminalDeduction*(ont2);

# 6. Идеи за бъдещо развитие и подобрения

Изложеният пример на криминална случка е само една демонстрация на технологията в един неин вариант. Би могло проекта да се адаптира към по-общ модел на данни, вкарвани или извличани по автоматизиран начин. Например някакви взаимодействия на хора от Фейсбук, или трафик на автомобили с цел проследимост. Би могло също така описателната част (онтологията) да е статична, а индивидите или самите данни да се събират постоянно от Интернет или от някакви датчици.

# 7. Източници и използвана литература

[1] Knowledge Representation and Reasoning, Ronald Brachman, Hector Levesque, 2004, The Morgan Kaufmann Series in Artificial Intelligence

[2] Apache Jena project, <https://jena.apache.org/documentation/inference/index.html#OWLconfiguration>

[3] Web Ontology Language (OWL), W3C, <https://www.w3.org/OWL/>

[4] SPARQL By Example, W3C, <https://www.w3.org/2009/Talks/0615-qbe/>

# Приложения

## 1. Сорс код (Source code)

<https://github.com/borkox/uni-sofia-ontology-project>