

Графы знаний

Лекция 5 - Интеграция данных в графы

М. Галкин, Д. Муромцев



Сегодня

1. Introduction
2. Представление знаний в графах - RDF & RDFS & OWL
3. Хранение знаний в графах - SPARQL & Graph Databases
4. Однородность знаний - RDF* & Wikidata & SHACL & ShEx
- 5. Интеграция данных в графы знаний - Semantic Data Integration**
6. Введение в теорию графов - Graph Theory Intro
7. Векторные представления графов - Knowledge Graph Embeddings
8. Машинное обучение на графах - Graph Neural Networks & KGs
9. Некоторые применения - Question Answering & Query Embedding

Содержание

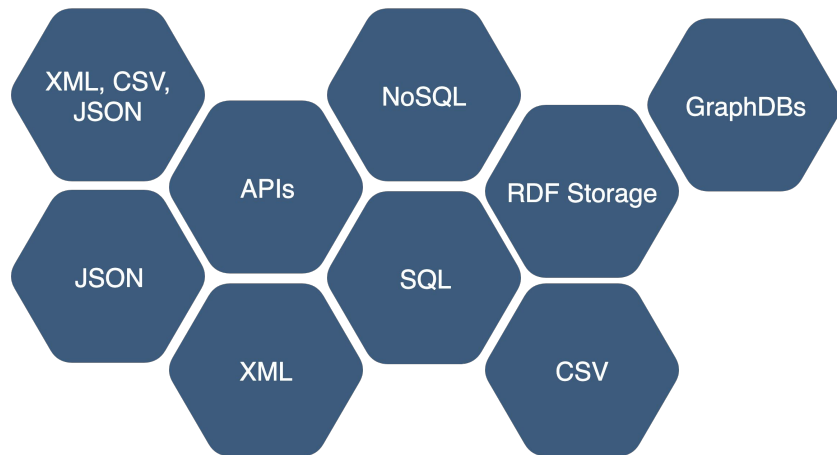
- **Способы интеграции данных в графы**
- Semantic Data Integration
 - Global-as-View
 - Local-as-View
- Физическая интеграция ETL
 - R2ML
 - RML
- Виртуальная интеграция
 - Архитектура Mediator-Wrapper
 - Федеративные запросы
 - SPARQL 2 SQL

Как строить графы знаний

Knowledge Graph

Semantic Data Integration

Structured Sources

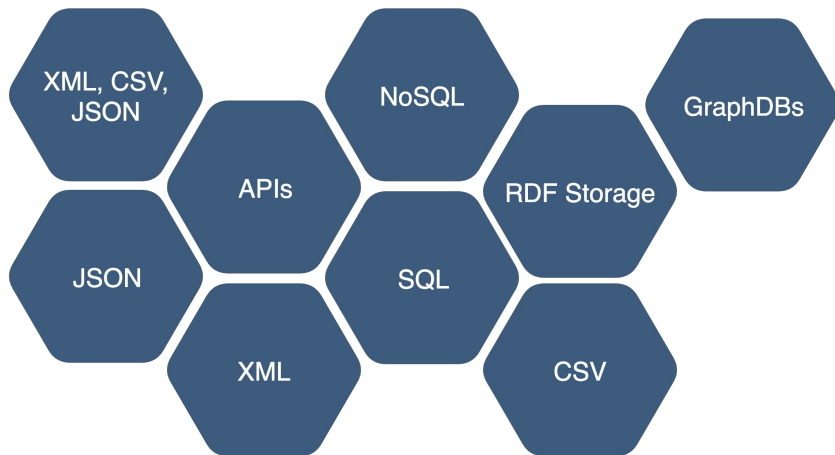


Как строить графы знаний

Knowledge Graph

Semantic Data Integration

Structured Sources



Knowledge Graph

Information Retrieval & NLP

Unstructured Sources

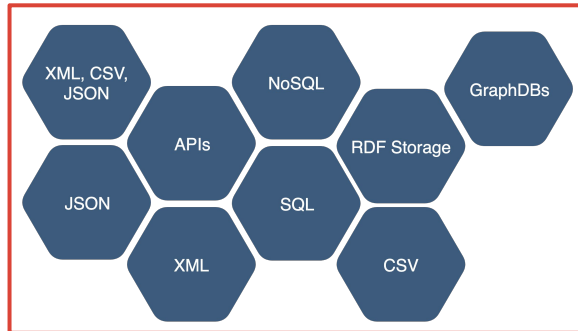


Semantic Data Integration

Knowledge Graph

Semantic Data Integration

Structured Sources



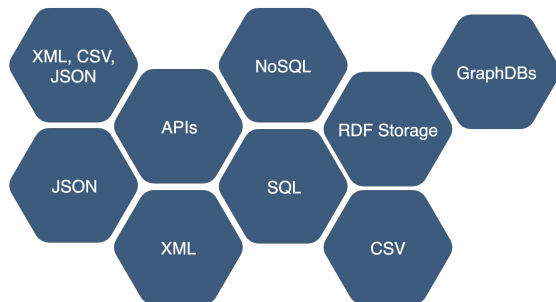
- **Неоднородность форматов**
- Семантическая неоднородность
- Распределенность данных
- Неоднородность именования
- Неоднозначность и эволюция данных

Semantic Data Integration

Knowledge Graph

Semantic Data Integration

Structured Sources



- Неоднородность форматов
- Семантическая неоднородность
- Распределенность данных
- Неоднородность именования
- Неоднозначность и эволюция данных

Люди

Артисты

Животные

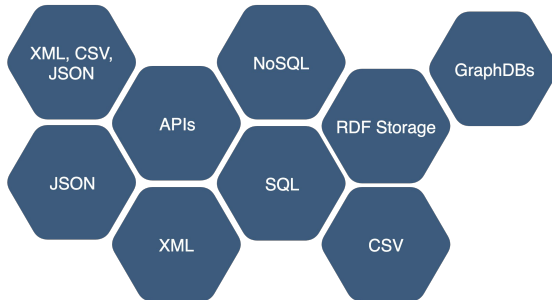
Кошки

Semantic Data Integration

Knowledge Graph

Semantic Data Integration

Structured Sources



- Неоднородность форматов
- Семантическая неоднородность
- **Распределенность данных**
- Неоднородность именования
- Неоднозначность и эволюция данных

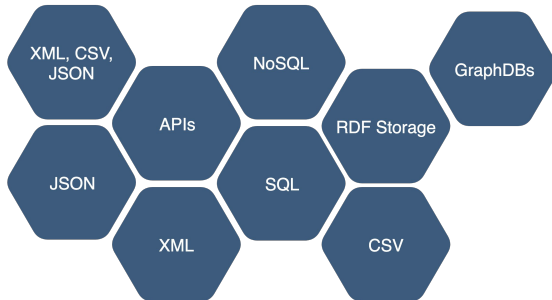


Semantic Data Integration

Knowledge Graph

Semantic Data Integration

Structured Sources



- Неоднородность форматов
- Семантическая неоднородность
- Распределенность данных
- **Неоднородность именования**
- Неоднозначность и эволюция данных

“Кронверский, 49”

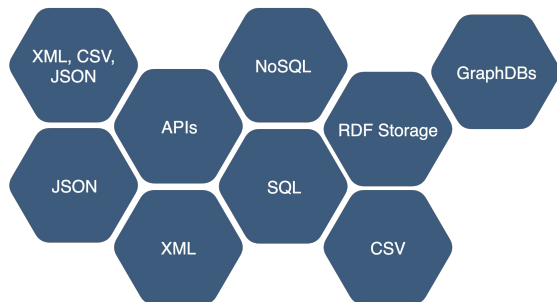
“Кронверский пр-кт, д. 49”

Semantic Data Integration

Knowledge Graph

Semantic Data Integration

Structured Sources



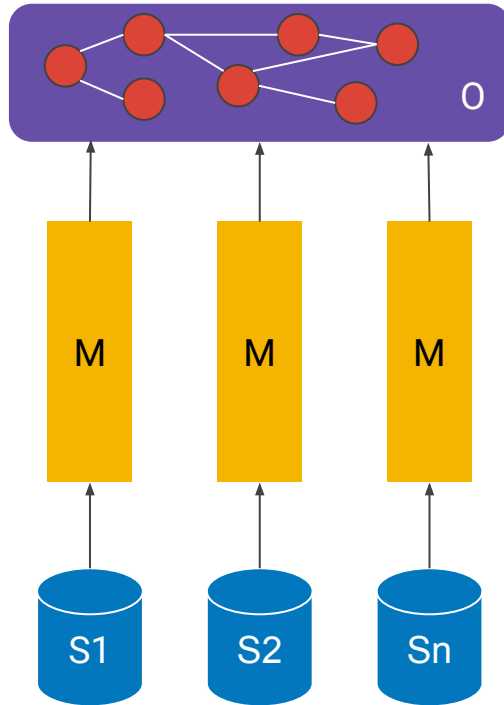
- Неоднородность форматов
- Семантическая неоднородность
- Распределенность данных
- Неоднородность именования
- **Неоднозначность и эволюция данных**

вчера	Астана
сегодня	Нурсултан

Содержание

- Способы интеграции данных в графы
- **Semantic Data Integration**
 - Global-as-View
 - Local-as-View
- Физическая интеграция ETL
 - R2ML
 - RML
- Виртуальная интеграция
 - Архитектура Mediator-Wrapper
 - Федеративные запросы
 - SPARQL 2 SQL

Semantic Data Integration



Integration System

$$IS = \langle O, S, M \rangle$$

O - множество концептов в общей схеме

S - множество источников данных

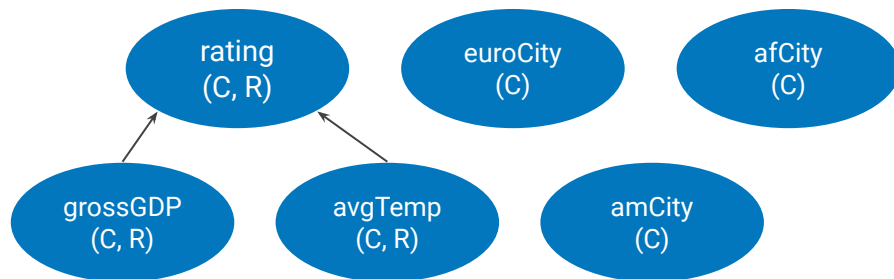
M - множество отображений (маппингов)

- Global-as-View

- Local-as-View

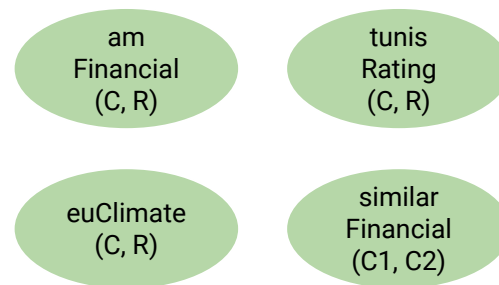
- Global-and-Local-as-View

Semantic Data Integration



```
grossGDP    rdf:type  rdf:Property .
avgTemp     rdf:type  rdf:Property .
rating      rdf:type  rdf:Property .
grossGDP    rdfs:subPropertyOf rating .
avgTemp     rdfs:subPropertyOf rating .
euroCity    rdf:type  rdfs:Class .
amCity      rdf:type  rdfs:Class .
afCity      rdf:type  rdfs:Class .
```

Глобальная онтология



```
amFinancial    rdf:type  rdf:Property .
euClimate      rdf:type  rdf:Property .
tunisRating    rdf:type  rdf:Property .
similarFinancial rdf:type  rdf:Property .
```

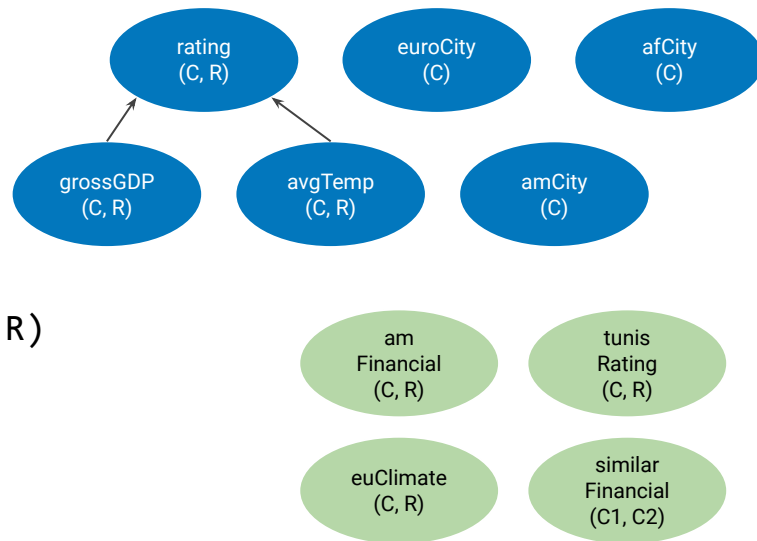
Локальная схема источника

Semantic Data Integration - Global-as-View

Global-as-View: определение сущностей **глобальной** онтологии O в терминах **локальных** источников S

```
amCity(C)           :- amFinancial(C,R)
grossGDP(C,R)       :- amFinancial(C,R)
euroCity(C)         :- euClimate(C,R)
avgTemp(C,R)        :- euClimate(C,R)
grossGDP("Tunis",R) :- tunisRating("financial", R)
avgTemp("Tunis", R) :- tunisRating("climate", R)
afCity("Tunis")

amCity(C1)          :- similarFinancial(C1,C2)
amCity(C2)          :- similarFinancial(C1,C2)
grossGDP(C1,R)      :- similarFinancial(C1,C2), amFinancial(C2, R)
```



Semantic Data Integration - Global-as-View - Queries

Global-as-View: определение сущностей **глобальной** онтологии O в терминах **локальных** источников S

```
query(C) :- grossGDP(C,R), amCity(C)
```

```
query_1(C) :- amFinancial(C,R), similarFinancial(C,C2)
```

```
query_2(C) :- similarFinancial(C,C2), amFinancial(C2,R), similarFinancial(C1,C2)
```

Алгоритмическая сложность переписывания GaV:

- EXPTIME - без ограничений на маппинги
- P - с ограничениями на переписывания

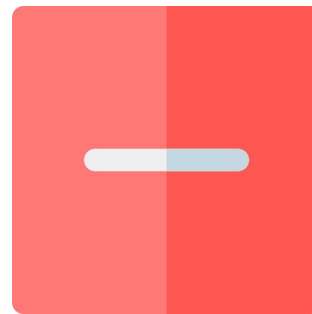
```
amCity(C)           :- amFinancial(C,R)
grossGDP(C,R)       :- amFinancial(C,R)
euroCity(C)         :- euClimate(C,R)
avgTemp(C,R)        :- euClimate(C,R)
grossGDP("Tunis",R) :- tunisRating("financial", R)
avgTemp("Tunis", R) :- tunisRating("climate", R)
amCity(C1)          :- similarFinancial(C1,C2)
amCity(C2)          :- similarFinancial(C1,C2)
grossGDP(C1,R):- similarFinancial(C1,C2),
amFinancial(C2, R)
```

Semantic Data Integration - Global-as-View - Queries

Global-as-View: определение сущностей **глобальной** онтологии O в терминах **локальных** источников S



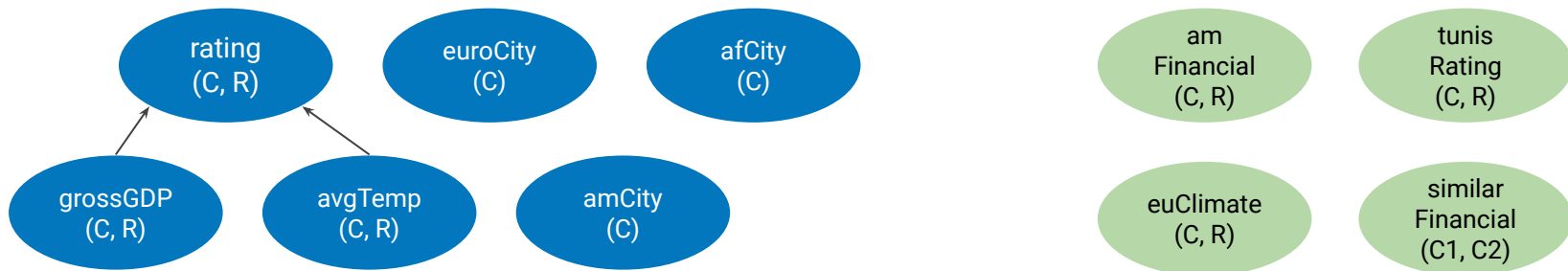
- Полиномиальная сложность переписывания запросов при некоторых ограничениях
- Используют, когда глобальная схема меняется, а схема источников стабильная



- Если схема источников часто меняется, а глобальная схема нет

Semantic Data Integration - Local-as-View

Local-as-View: определение сущностей **локальных** источников S в терминах **глобальной** онтологии O



<code>amFinancial(C,R)</code>	<code>:- amCity(C), grossGDP(C,R)</code>
<code>euClimate(C,R)</code>	<code>:- euCity(C), avgTemp(C,R)</code>
<code>tunisRating("financial",R)</code>	<code>:- afCity("Tunis"), grossGDP("Tunis", R)</code>
<code>tunisRating("climate",R)</code>	<code>:- afCity("Tunis"), avgTemp("Tunis",R)</code>
<code>similarFinancial(C1,C2)</code>	<code>:- amCity(C1),amCity(C2),grossGDP(C1,R),grossGDP(C2,R)</code>

Semantic Data Integration - Local-as-View

Local-as-View: определение сущностей **локальных** источников S в терминах **глобальной** онтологии O

```
query(C) :- grossGDP(C,R), amCity(C)

query_1(C) :- amFinancial(C,R)
query_2(C) :- similarFinancial(C,C2)
```

```
amFinancial(C,R) :- amCity(C), grossGDP(C,R)
euClimate(C,R) :- euCity(C), avgTemp(C,R)
tunisRating("financial",R) :- afCity("Tunis"), grossGDP("Tunis", R)
tunisRating("climate",R) :- afCity("Tunis"), avgTemp("Tunis",R)
similarFinancial(C1,C2) :-
amCity(C1),amCity(C2),grossGDP(C1,R),grossGDP(C2,R)
```

Алгоритмическая сложность переписывания LaV:

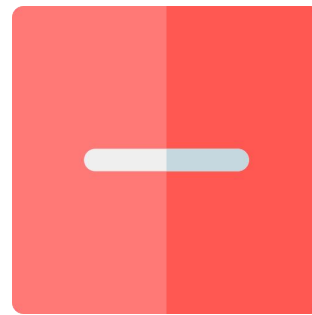
- NP-hard если нужно решать Query Containment

Semantic Data Integration - Local-as-View - Queries

Local-as-View: определение сущностей **локальных** источников S в терминах **глобальной** онтологии O



- Используют, когда глобальная схема постоянна, а схема источников изменяется



- Если глобальная схема часто изменяется

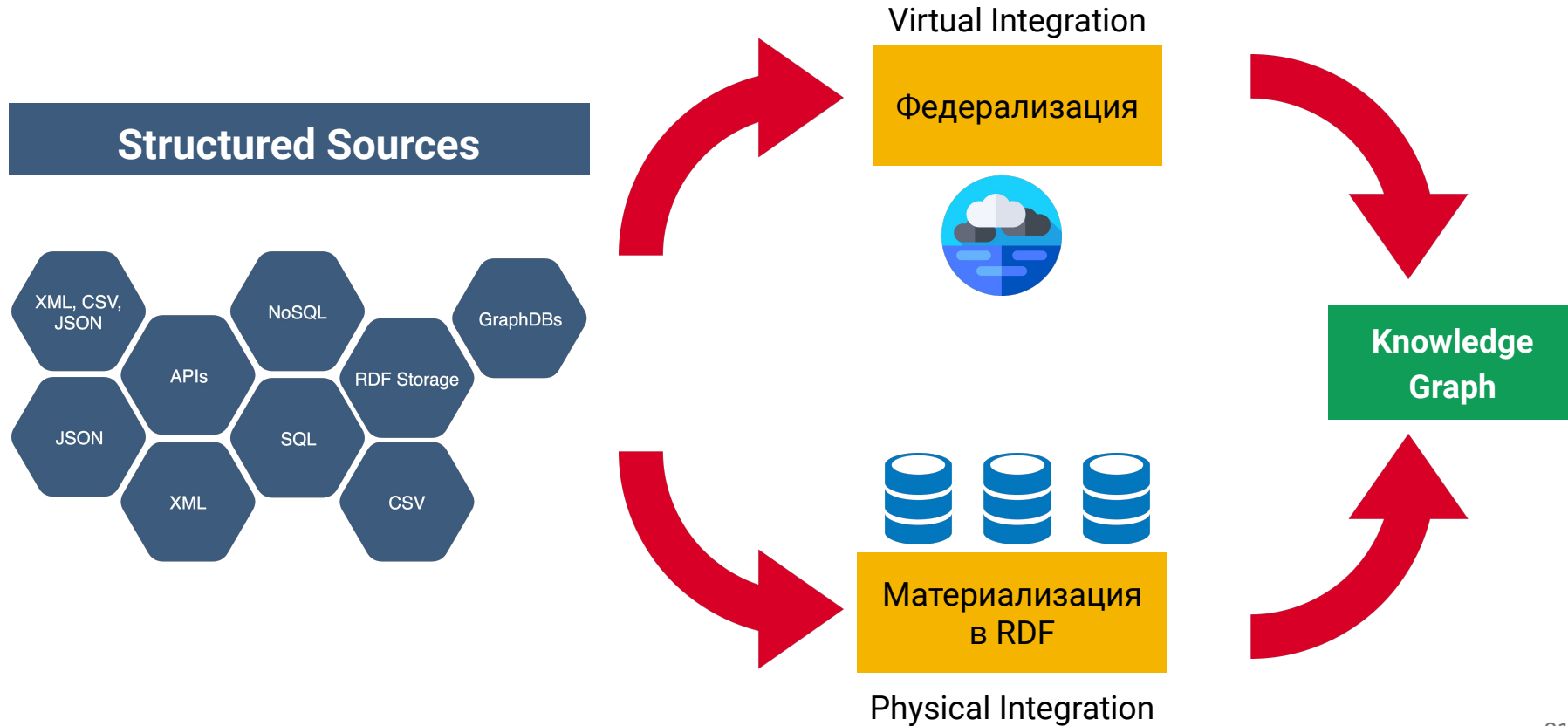
Semantic Data Integration - Global-and-Local-as-View

Global-and-Local-as-View: определение комбинации терминов **глобальной** онтологии O через комбинацию терминов **источников** S

```
amCity(C1), amCity(C2), grossGDP(C1,R), grossGDP(C2,R) :-  
    amFinancial(C1,R), similarFinancial(C1,C2)
```

Используется, когда маппинги источников S *относительно* просты

Semantic Data Integration

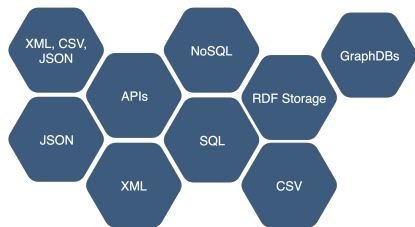


Содержание

- Способы интеграции данных в графы
- Semantic Data Integration
 - Global-as-View
 - Local-as-View
- **Физическая интеграция ETL**
 - R2ML
 - RML
- Виртуальная интеграция
 - Архитектура Mediator-Wrapper
 - Федеративные запросы
 - SPARQL 2 SQL

Physical Integration (Materialization)

Extract



- Выделение данных для трансформации
- Подготовка и очистка датасетов

Transform



- Физическое преобразование в RDF с помощью маппингов
- Рекомендации W3C RDB2RDF

Load

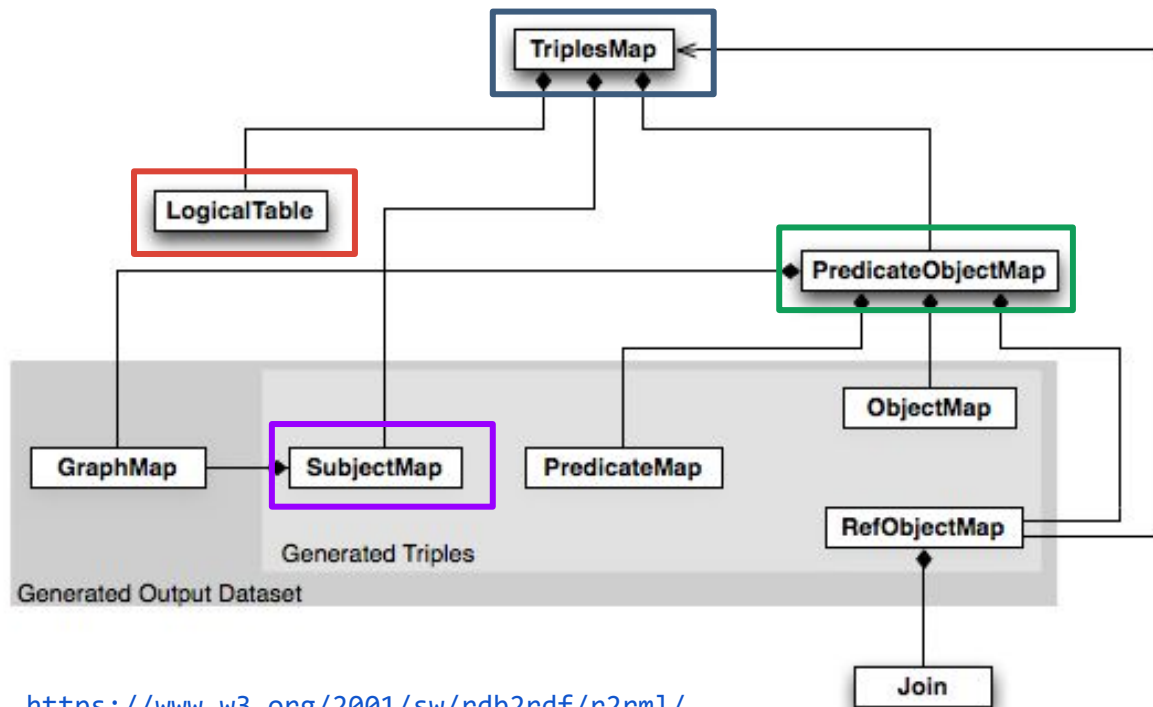


- Загрузка в единое хранилище (Data Warehouse)

Physical Integration (Materialization) - R2RML

R2RML - рекомендация W3C для создания отображений реляционных баз в RDF

rr: <http://www.w3.org/ns/r2rml#>



Logical Table - исходная база или view как результат SQL запроса

Triples Map - набор правил для преобразования строк таблиц в триплеты

Subject Map - способ задания URI генерируемой сущности

Predicate-Object Map - способ генерации предиката и объекта

<https://www.w3.org/2001/sw/rdb2rdf/r2rml/>

Physical Integration (Materialization) - R2RML

EMP

EMPNO	ENAME	JOB	DEPTNO
INTEGER PRIMARY KEY	VARCHAR(100)	VARCHAR(20)	INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10

DEPT

DEPTNO	DNAME	LOC
INTEGER PRIMARY KEY	VARCHAR(30)	VARCHAR(100)
10	APPSERVER	NEW YORK

Physical Integration (Materialization) - R2RML

EMP

EMPNO	ENAME	JOB	DEPTNO
INTEGER PRIMARY KEY	VARCHAR(100)	VARCHAR(20)	INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10

DEPT

DEPTNO	DNAME	LOC
INTEGER PRIMARY KEY	VARCHAR(30)	VARCHAR(100)
10	APPSERVER	NEW YORK

@prefix rr: <http://www.w3.org/ns/r2rml#>.

@prefix ex: <http://example.com/ns#>.

<#TriplesMap1>

rr:logicalTable [rr:tableName "EMP"];

rr:subjectMap [

rr:template "http://data.example.com/employee/{EMPNO}";

rr:class ex:Employee;

];

rr:predicateObjectMap [

rr:predicate ex:name;

rr:objectMap [rr:column "ENAME"];

].

Physical Integration (Materialization) - R2RML

EMP

EMPNO	ENAME	JOB	DEPTNO
INTEGER PRIMARY KEY	VARCHAR(100)	VARCHAR(20)	INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10

@prefix rr: <http://www.w3.org/ns/r2rml#>.

@prefix ex: <http://example.com/ns#>.

<#TriplesMap1>

```
rr:logicalTable [ rr:tableName "EMP" ];
rr:subjectMap [
  rr:template "http://data.example.com/employee/{EMPNO}";
  rr:class ex:Employee;
];
rr:predicateObjectMap [
  rr:predicate ex:name;
  rr:objectMap [ rr:column "ENAME" ];
].
```

DEPT

DEPTNO	DNAME	LOC
INTEGER PRIMARY KEY	VARCHAR(30)	VARCHAR(100)
10	APPSERVER	NEW YORK

```
<http://data.example.com/employee/7369> rdf:type ex:Employee.
<http://data.example.com/employee/7369> ex:name "SMITH".
```

Physical Integration (Materialization) - R2RML

EMP

EMPNO	ENAME	JOB	DEPTNO
INTEGER PRIMARY KEY	VARCHAR(100)	VARCHAR(20)	INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10

DEPT

DEPTNO	DNAME	LOC
INTEGER PRIMARY KEY	VARCHAR(30)	VARCHAR(100)
10	APPSERVER	NEW YORK

```
<#DeptTableView> rr:sqlQuery ""  
SELECT DEPTNO, DNAME, LOC,  
       (SELECT COUNT(*) FROM EMP  
WHERE EMP.DEPTNO=DEPT.DEPTNO) AS  
STAFF FROM DEPT;  
"".
```

Physical Integration (Materialization) - R2RML

EMP

EMPNO	ENAME	JOB	DEPTNO
INTEGER PRIMARY KEY	VARCHAR(100)	VARCHAR(20)	INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10

DEPT

DEPTNO	DNAME	LOC
INTEGER PRIMARY KEY	VARCHAR(30)	VARCHAR(100)
10	APPSERVER	NEW YORK

```
<#TriplesMap2>
  rr:logicalTable <#DeptTableView>;
  rr:subjectMap [
    rr:template "http://data.example.com/department/{DEPTNO}";
    rr:class ex:Department;
  ];
  rr:predicateObjectMap [
    rr:predicate ex:name;
    rr:objectMap [ rr:column "DNAME" ];
  ];
  rr:predicateObjectMap [
    rr:predicate ex:location;
    rr:objectMap [ rr:column "LOC" ];
  ];
  rr:predicateObjectMap [
    rr:predicate ex:staff;
    rr:objectMap [ rr:column "STAFF" ];
  ].
```

```
<#DeptTableView> rr:sqlQuery """
SELECT DEPTNO, DNAME, LOC,
       (SELECT COUNT(*) FROM EMP
        WHERE EMP.DEPTNO=DEPT.DEPTNO) AS
STAFF FROM DEPT;
""".
```

```
<http://data.example.com/department/10> rdf:type ex:Department.
<http://data.example.com/department/10> ex:name "APPSERVER".
<http://data.example.com/department/10> ex:location "NEW YORK".
<http://data.example.com/department/10> ex:staff 1.
```

Physical Integration (Materialization) - R2RML

EMP

EMPNO	ENAME	JOB	DEPTNO
INTEGER PRIMARY KEY	VARCHAR(100)	VARCHAR(20)	INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10

DEPT

DEPTNO	DNAME	LOC
INTEGER PRIMARY KEY	VARCHAR(30)	VARCHAR(100)
10	APPSERVER	NEW YORK

```
<#TriplesMap1>
  rr:predicateObjectMap [
    rr:predicate ex:department;
    rr:objectMap [
      rr:parentTriplesMap <#TriplesMap2>;
      rr:joinCondition [
        rr:child "DEPTNO";
        rr:parent "DEPTNO";
      ];
    ];
  ].
```

```
<http://data.example.com/employee/7369> ex:department <http://data.example.com/department/10>.
```

Physical Integration (Materialization) - R2RML

EMP

EMPNO	ENAME	JOB	DEPTNO
INTEGER PRIMARY KEY	VARCHAR(100)	VARCHAR(20)	INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10

DEPT

DEPTNO	DNAME	LOC
INTEGER PRIMARY KEY	VARCHAR(30)	VARCHAR(100)
10	APPSERVER	NEW YORK

<#TriplesMap1>

```
rr:logicalTable [ rr:sqlQuery ""
```

```
  SELECT EMP.*, (CASE JOB
```

```
    WHEN 'CLERK' THEN 'general-office'
```

```
    WHEN 'NIGHTGUARD' THEN 'security'
```

```
    WHEN 'ENGINEER' THEN 'engineering'
```

```
  END) ROLE FROM EMP "" ];
```

```
rr:subjectMap [
```

```
  rr:template "http://data.example.com/employee/{EMPNO}";
```

```
];
```

```
rr:predicateObjectMap [
```

```
  rr:predicate ex:role;
```

```
  rr:objectMap [ rr:template "http://data.example.com/roles/{ROLE}" ];
```

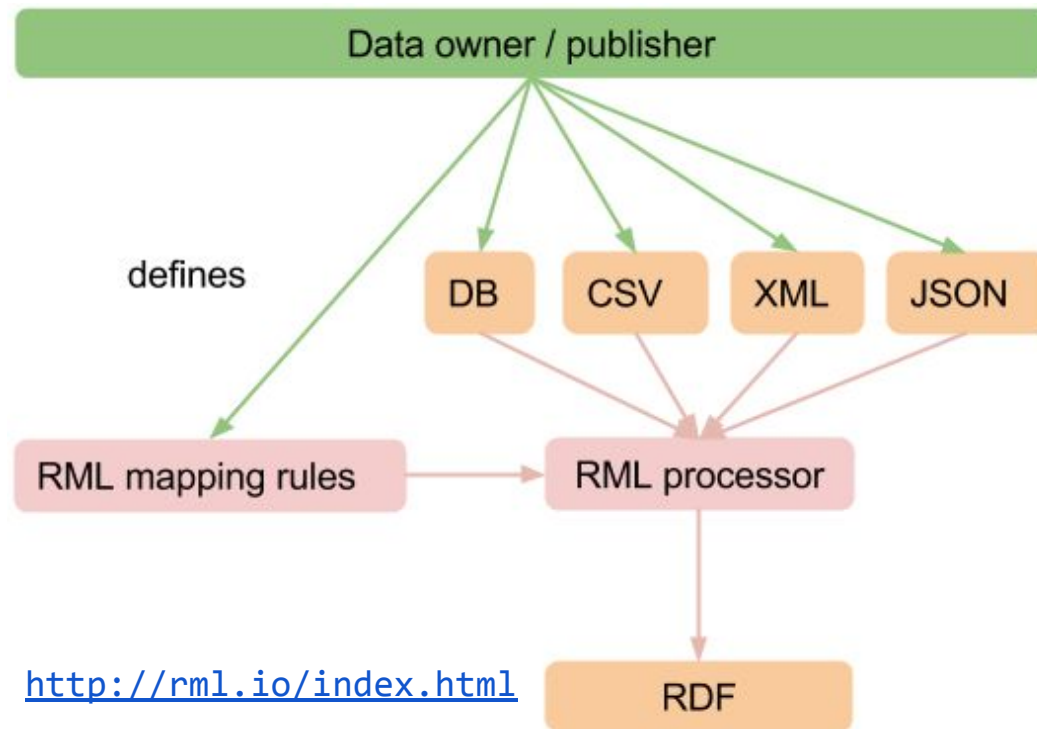
```
].
```

<http://data.example.com/employee/7369> ex:role <http://data.example.com/roles/general-office>.

Physical Integration (Materialization) - RML

RML (RDF Mapping Language) - надмножество R2RML, поддерживающее CSV, JSON, XML

rr: <http://www.w3.org/ns/r2rml#>



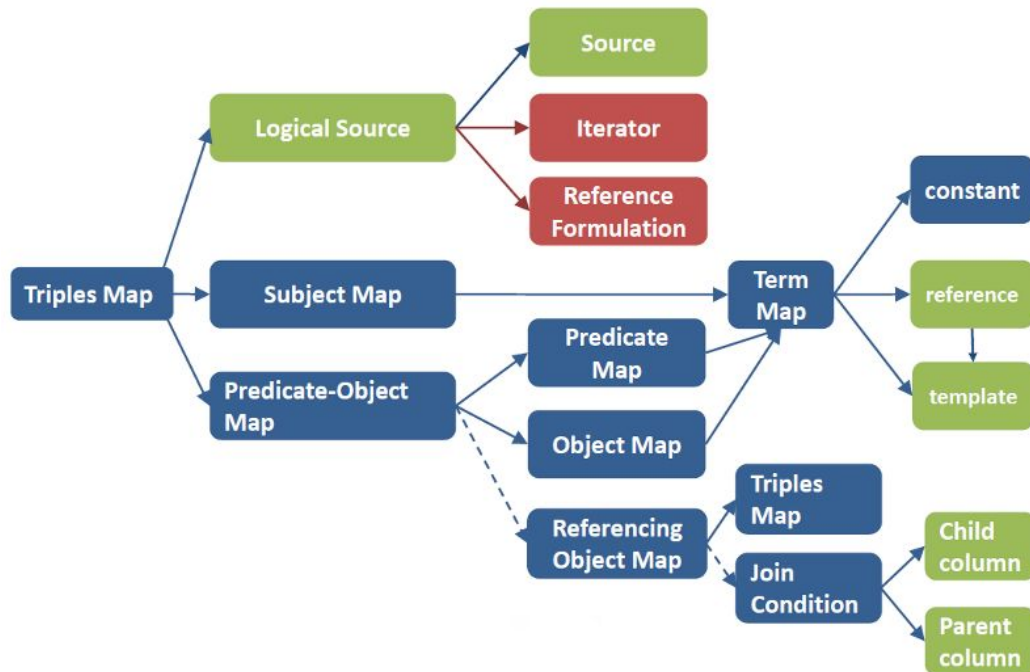
RML содержит:

- Процессор правил
- GUI
- Валидатор

Physical Integration (Materialization) - RML

RML (RDF Mapping Language) - надмножество R2RML, поддерживающее CSV, JSON, XML

rr: <http://www.w3.org/ns/r2rml#>



Logical Source теперь состоит из трех частей:

Source - ссылка на источник

Iterator - итератор по источнику

Reference Formulation - формат источника

Physical Integration (Materialization) - R2RML vs RML

R2RML		RML	
Logical Table (relational database)	rr:logicalTable	Logical Source (CSV, XML, JSON,HTML, ...)	rml:logicalSource
Table Name	rr:tableName	URI (pointing to the source)	rml:source
column	rr:column	reference	rml:reference
(SQL)	rr:SQLQuery	Reference Formulation	rml:referenceFormulation
per row iteration		defined iterator	rml:iterator

Physical Integration (Materialization) - RML

CSV

```
id, stop, latitude, longitude  
6523, 25, 50.901389, 4.484444
```

Physical Integration (Materialization) - RML

CSV

id	stop	latitude	longitude
6523	25	50.901389	4.484444

```
<http://airport.example.com/6523>
  rdf:type transit:Stop ;
  transit:route 25 ;
  :lat 50.901389 ;
  :long 4.484444 .
```

```
<#Mapping1>
  rml:logicalSource [
    rml:source "http://www.example.com/airports.csv" ;
    rml:referenceFormulation ql:CSV
  ];
  rr:subjectMap [
    rr:template "http://airport.example.com/{id}";
    rr:class transit:Stop
  ];
  rr:predicateObjectMap [
    rr:predicate transit:route;
    rr:objectMap [
      rml:reference "stop";
      rr:datatype xsd:int
    ]
  ];
  rr:predicateObjectMap [
    rr:predicate :lat;
    rr:objectMap [ rml:reference "latitude" ]
  ]; ].
```

Physical Integration (Materialization) - RML

JSON

```
{
  "venue":
  {
    "latitude": "51.0500000",
    "longitude": "3.7166700"
  },
  "location":
  {
    "continent": " EU",
    "country": "BE",
    "city": "Brussels"
  }
}
```

Physical Integration (Materialization) - RML

```
<#VenueMapping>
  rml:logicalSource [
    rml:source "http://www.example.com/files/Venue.json";
    rml:referenceFormulation ql:JSONPath;
    rml:iterator "$"
  ];
  rr:subjectMap [
    rr:template
      "http://loc.example.com/city/{$.location.city}";
    rr:class schema:City
  ];
  rr:predicateObjectMap [
    rr:predicate wgs84_pos:lat;
    rr:objectMap [
      rml:reference "$.venue.latitude"
    ]
  ];
].
```

JSON

```
{
  "venue":
  {
    "latitude": "51.0500000",
    "longitude": "3.7166700"
  },
  "location":
  {
    "continent": " EU",
    "country": "BE",
    "city": "Brussels"
  }
}
```

Physical Integration (Materialization) - RML

```
<#VenueMapping>
  rml:logicalSource [
    rml:source "http://www.example.com/files/Venue.json";
    rml:referenceFormulation ql:JSONPath;
    rml:iterator "$"
  ];
  rr:subjectMap [
    rr:template
    "http://loc.example.com/city/{$.location.city}";
    rr:class schema:City
  ];
  rr:predicateObjectMap [
    rr:predicate wgs84_pos:lat;
    rr:objectMap [
      rml:reference "$.venue.latitude"
    ]
  ];
]; ]
```

JSON

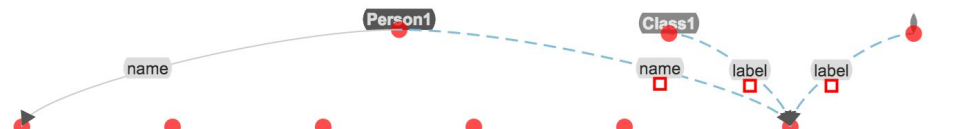
```
{
  "venue":
  {
    "latitude": "51.0500000",
    "longitude": "3.7166700"
  },
  "location":
  {
    "continent": " EU",
    "country": "BE",
    "city": "Brussels"
  }
}
```

```
<http://loc.example.com/city/Brussels> rdf:type schema:City ;
wgs84_pos:lat "50.901389" ;
wgs84_pos:long "4.484444" ;
gn:countryCode "BE".
```

Physical Integration (Materialization) - Tools - Karma

Karma - платформа для интеграции данных из CSV, XML, JSON, RDB, HTML. Есть GUI !

<https://github.com/usc-isi-i2/Web-Karma>

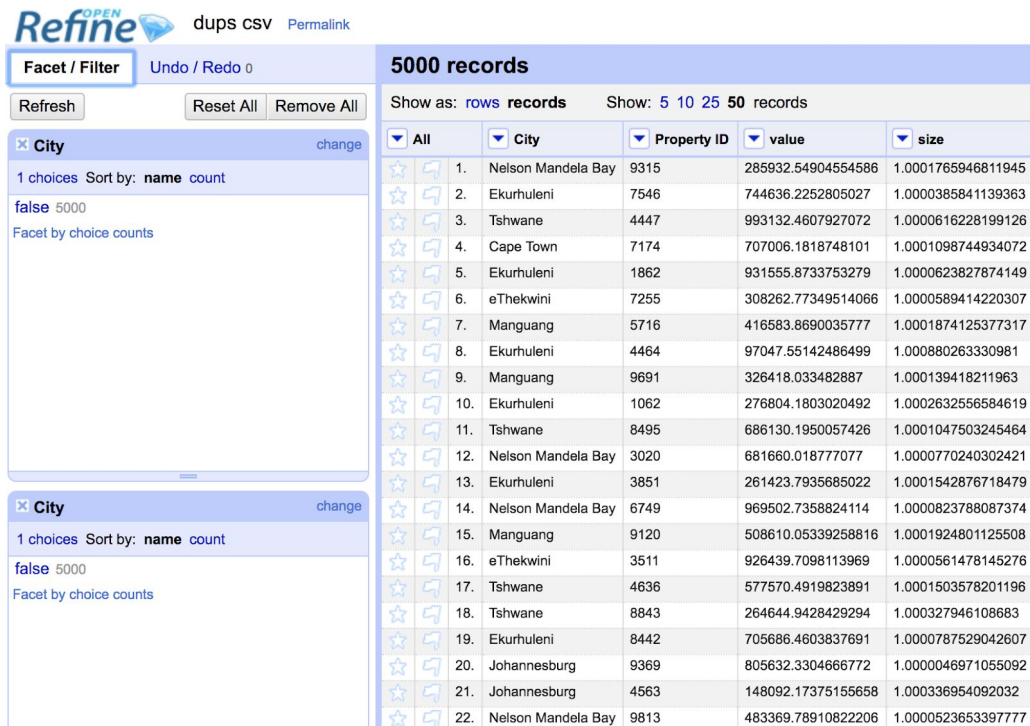



	artist ▾	birthDeath ▾	creditLine ▾	dimensions ▾	title ▾	sitters ▾ sitter ▾	sourceURL
	Frishmuth, Harriet Whitney	1880-1980	Gift of the Friends of American Art	H: 61 in.	Joy of the Waters	Henry Lancom Abott	http://www.ima.waters-frishmuth-harriet-whitney

- Визуальная разработка маппингов
- Собственные функции преобразования данных на Python
- Может собирать данные с HTML
- RML-совместимый формат маппингов

Physical Integration (Materialization) - OpenRefine

OpenRefine - инструмент для очищения, трансформации и обогащения табличных данных.



Refine  dups csv Permalink

Facet / Filter Undo / Redo 0

Refresh Reset All Remove All

City change

1 choices Sort by: name count

false 5000

Facet by choice counts

City change

1 choices Sort by: name count

false 5000

Facet by choice counts

5000 records

Show as: rows records Show: 5 10 25 50 records

	All	City	Property ID	value	size
1.	Nelson Mandela Bay	9315	285932.54904554586	1.0001765946811945	
2.	Ekurhuleni	7546	744636.2252805027	1.0000385841139363	
3.	Tshwane	4447	993132.4607927072	1.0000616228199126	
4.	Cape Town	7174	707006.1818748101	1.0001098744934072	
5.	Ekurhuleni	1862	931555.8733753279	1.0000623827874149	
6.	eThekwini	7255	308262.77349514066	1.0000589414220307	
7.	Manguang	5716	416583.8690035777	1.0001874125377317	
8.	Ekurhuleni	4464	97047.55142486499	1.000880263330981	
9.	Manguang	9691	326418.033482887	1.000139418211963	
10.	Ekurhuleni	1062	276804.1803020492	1.0002632556584619	
11.	Tshwane	8495	686130.1950057426	1.0001047503245464	
12.	Nelson Mandela Bay	3020	681660.018777077	1.0000770240302421	
13.	Ekurhuleni	3851	261423.7935685022	1.0001542876718479	
14.	Nelson Mandela Bay	6749	969502.7358824114	1.0000823788087374	
15.	Manguang	9120	508610.05339258816	1.0001924801125508	
16.	eThekwini	3511	926439.7098113969	1.0000561478145276	
17.	Tshwane	4636	577570.4919823891	1.0001503578201196	
18.	Tshwane	8843	264644.9428429294	1.000327946108683	
19.	Ekurhuleni	8442	705686.4603837691	1.0000787529042607	
20.	Johannesburg	9369	805632.3304666772	1.0000046971055092	
21.	Johannesburg	4563	148092.17375155658	1.000336954092032	
22.	Nelson Mandela Bay	9813	483369.78910822206	1.0000523653397777	

- LOD Refine - расширение для экспорта в RDF

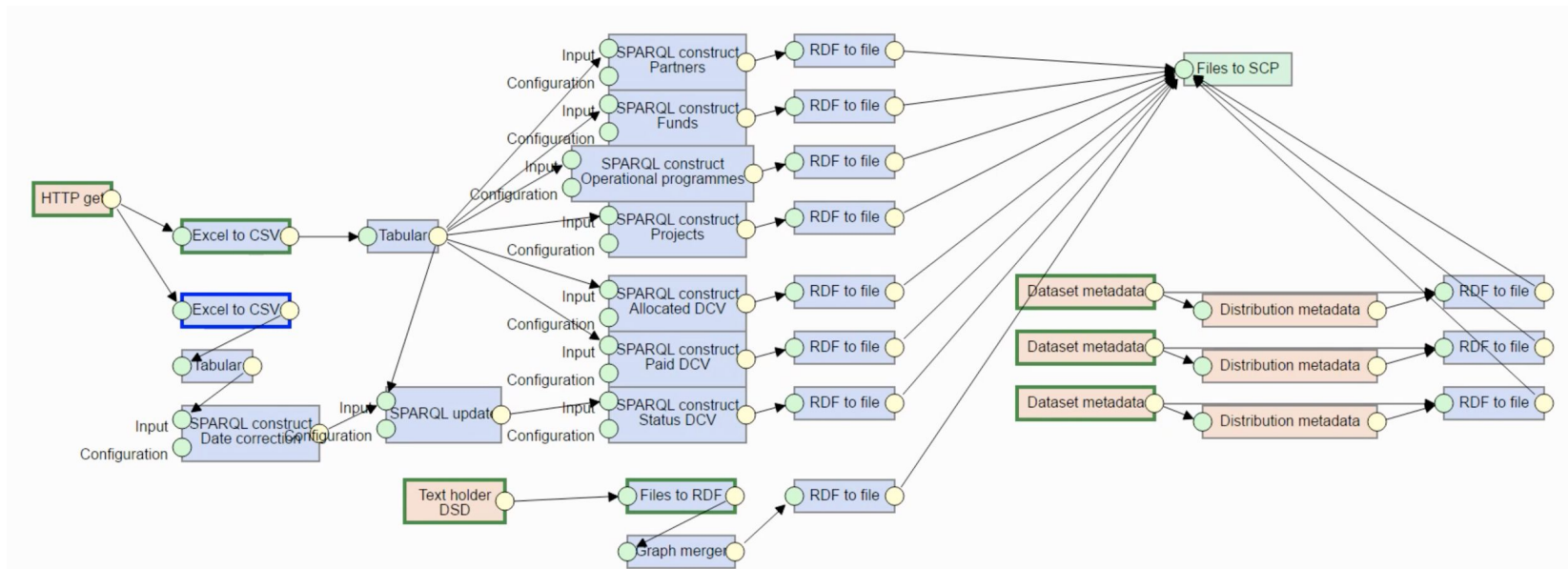
<http://openrefine.org/>
<https://github.com/sparkica/LODRefine>

Physical Integration (Materialization) - LinkedPipes

LinkedPipes - инструмент для разработки и выполнения ETL сценариев (пайплайнов).

- GUI для разработки пайплайнов
- Готовые компоненты для стандартных кейсов
- Кастомизированные компоненты

<https://linkedpipes.com/>



Physical Integration (Materialization) - SPARQL Generate

SPARQL-Generate включает маппинги прямо в SPARQL запрос

- Можно экспортировать готовый RDF сразу HDT
- Можно обрабатывать текст (regex)
- Расширение SPARQL 1.1

<https://ci.mines-stetienne.fr/sparql-generate/>

Queries

Links to the documentation of [iterator functions](#) and [binding functions](#).

Default query

```
6
7 GENERATE {
8
9   GENERATE {
10     <store/{?store}> ex:totalAmount ?{ sum("{?amount}"^^xsd:decimal ) } .
11   } GROUP BY ?store .
12
13   GENERATE {
14     <statistics> ex:perDay LIST( ?bnode ) .
15     ?bnode ex:averageAmount ?{ avg("{?amount}"^^xsd:decimal ) } ;
16     ex:date ?date
17   } WHERE {
18   } GROUP BY ?date
19   ORDER BY ?date
20   LIMIT 10
21   EXPRESSIONS ( fun:bnode(str(?date)) AS ?bnode ) .
22
```

 Run Query

☐ run automatically ☐ return stream ☐ debug Template

Result

```
7
8 <http://example.com/store/78>
9   ex:totalAmount 4682.83 .
10
11 <http://example.com/store/31>
12   ex:totalAmount 2532.70 .
13
14 <http://example.com/store/72>
15   ex:totalAmount 3583.27 .
16
17 <http://example.com/store/44>
18   ex:totalAmount 4524.08 .
19
20 <http://example.com/store/16>
21   ex:totalAmount 4540.59 .
```

Содержание

- Способы интеграции данных в графы
- Semantic Data Integration
 - Global-as-View
 - Local-as-View
- Физическая интеграция ETL
 - R2ML
 - RML
- **Виртуальная интеграция**
 - Архитектура Mediator-Wrapper
 - Федеративные запросы
 - SPARQL 2 SQL

Virtual Integration

Когда материализация не подходит:

- Слишком большой объем данных
- Нет доступа к исходным данным, только через API
- Данные и источники часто меняются -> преобразование не выгодно

Virtual Integration

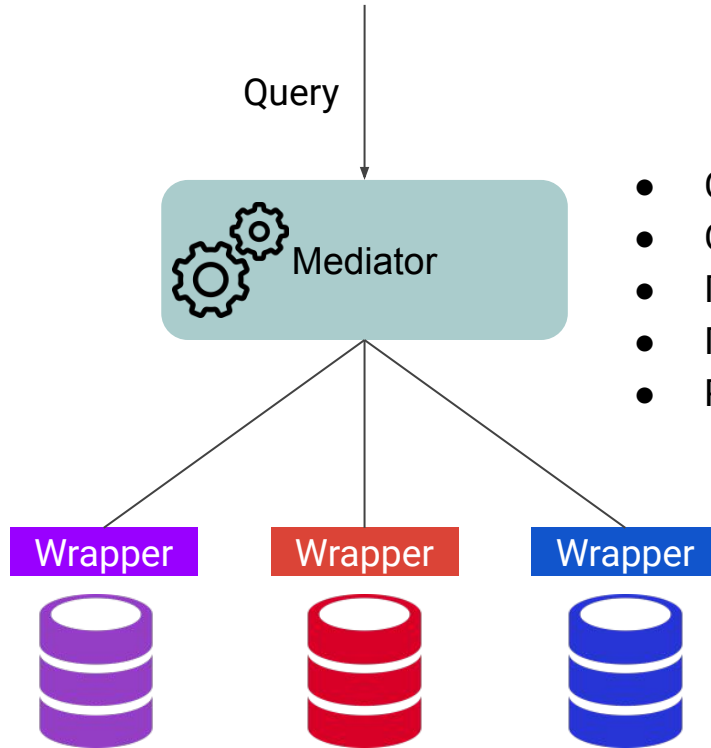
Когда материализация не подходит:

- Слишком большой объем данных
- Нет доступа к исходным данным, только через API
- Данные и источники часто меняются -> преобразование не выгодно

Виртуальная интеграция

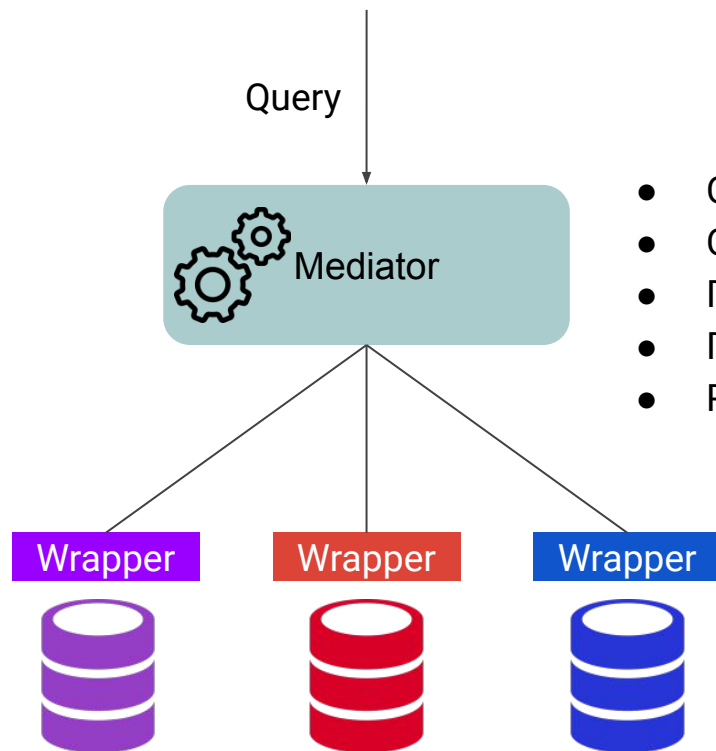
- Создание единой логической схемы источников
- Единый язык запросов
- Источники опрашиваются в нативных форматах
- Запросы переписываются на лету

Virtual Integration - Mediator-Wrapper



- Обработка и разбиение исходного запроса
- Определение нужных источников для выполнения
- Построение эффективного плана выполнения
- Преобразование результатов в единый формат
- Работает с глобальной схемой

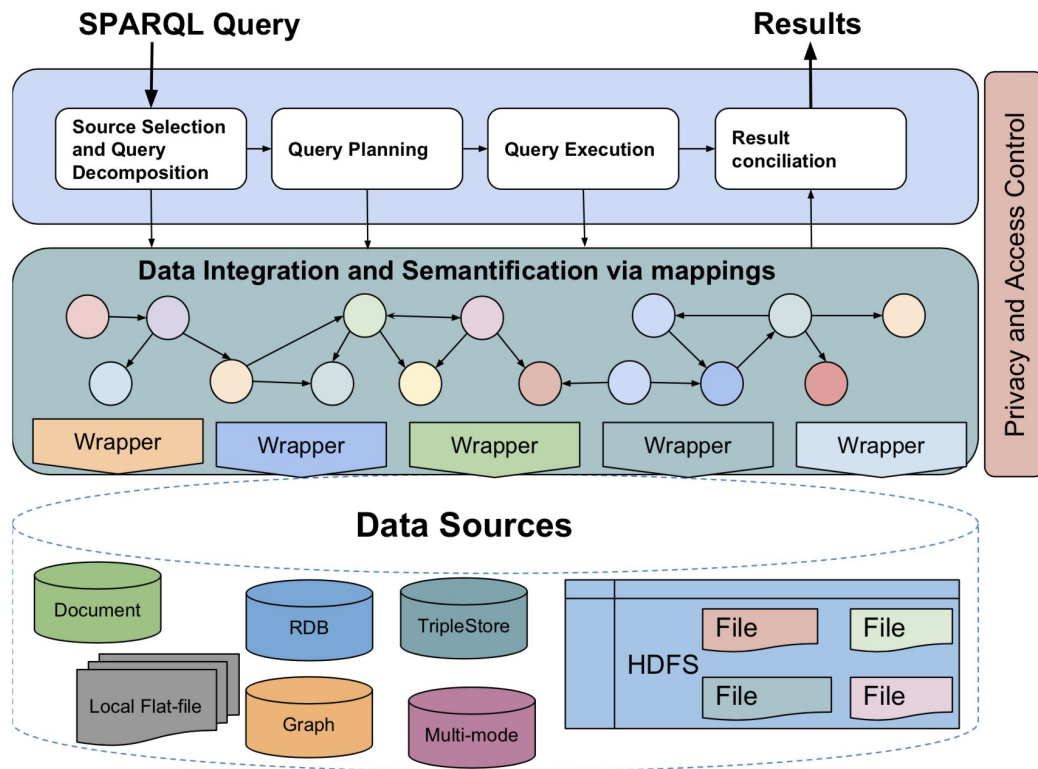
Virtual Integration - Mediator-Wrapper



- Обработка и разбиение исходного запроса
- Определение нужных источников для выполнения
- Построение эффективного плана выполнения
- Преобразование результатов в единый формат
- Работает с глобальной схемой

- Программный компонент над источником
- Отправляет запросы в нативном для источника формате
- Работает с уникальной схемой источника

Virtual Integration - Semantic Data Lake



SPARQL как универсальный язык

Стадии обработки запроса

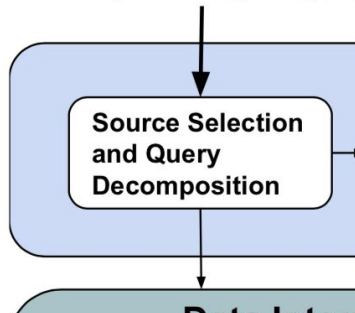
Единое логическое представление источников в Data Lake

Wrapper на каждый формат источника

Data Lake вместо Data Warehouse

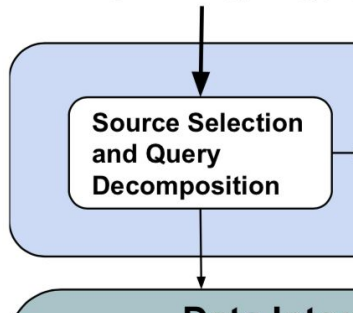
Virtual Integration - Federated Querying

```
SELECT DISTINCT ?s WHERE {  
  ?s foaf:page      ?page .  
  ?s owl:sameAs   ?sameas .  
  ?s geonames:inCountry ?inCountry . }
```



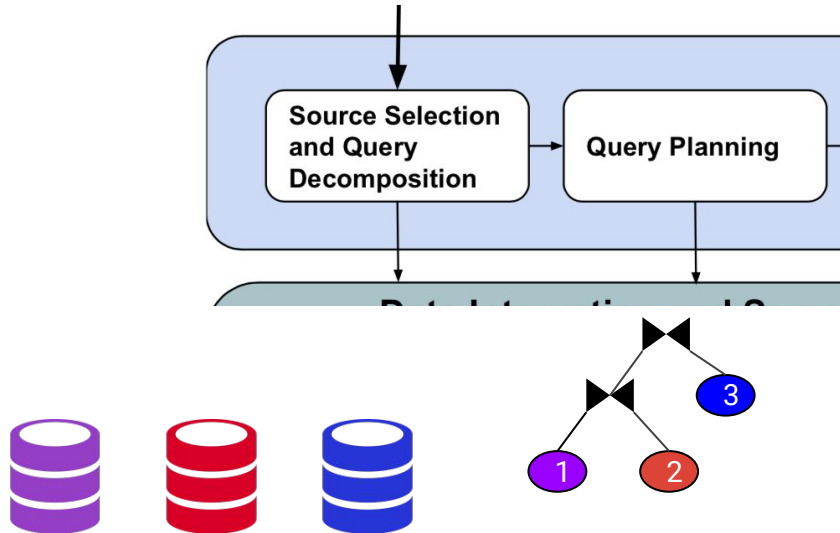
Virtual Integration - Federated Querying

```
SELECT DISTINCT ?s WHERE {  
  1 ?s foaf:page      ?page .  
  2 ?s owl:sameAs    ?sameas .  
  3 ?s geonames:inCountry ?inCountry . }
```



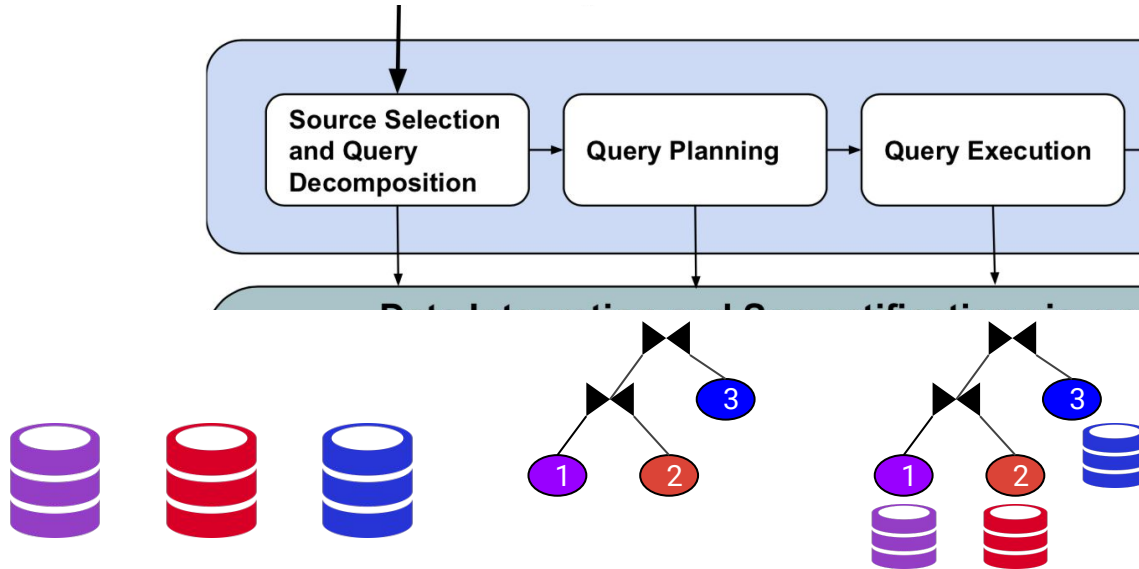
Virtual Integration - Federated Querying

```
SELECT DISTINCT ?s WHERE {  
  1 ?s foaf:page      ?page .  
  2 ?s owl:sameAs    ?sameas .  
  3 ?s geonames:inCountry ?inCountry . }
```



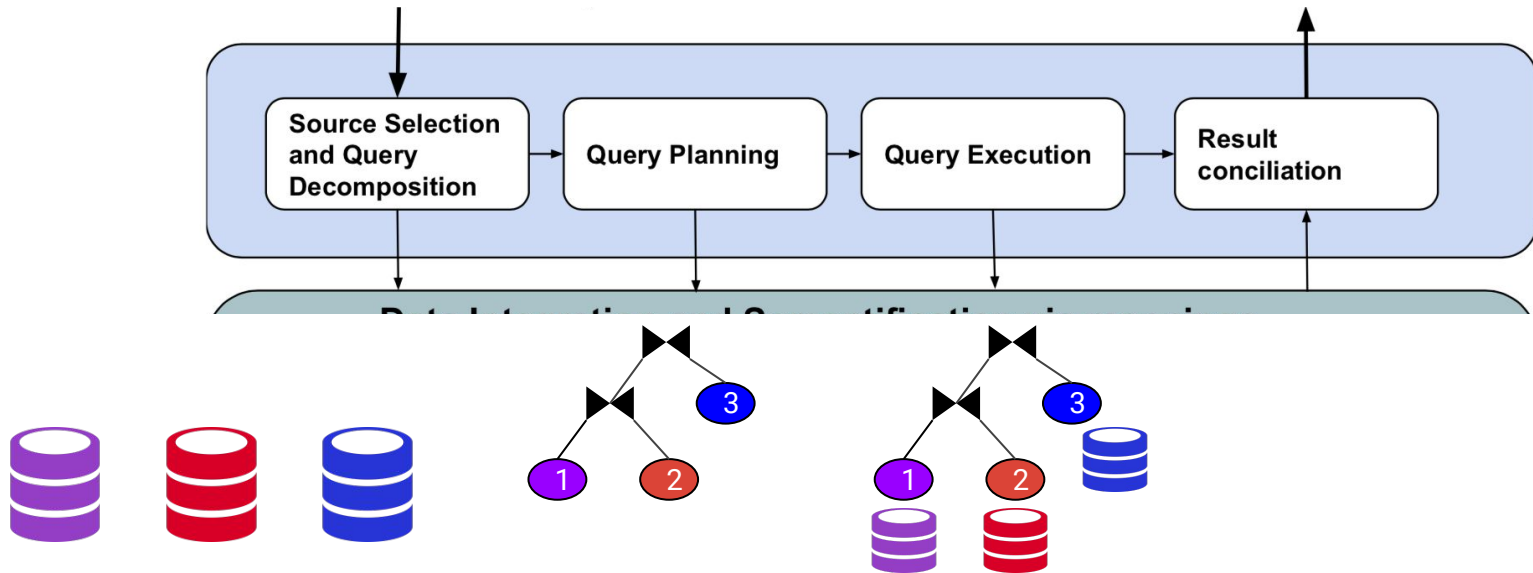
Virtual Integration - Federated Querying

```
SELECT DISTINCT ?s WHERE {  
  1 ?s foaf:page      ?page .  
  2 ?s owl:sameAs    ?sameas .  
  3 ?s geonames:inCountry ?inCountry . }
```



Virtual Integration - Federated Querying

```
SELECT DISTINCT ?s WHERE {  
  1 ?s foaf:page      ?page .  
  2 ?s owl:sameAs    ?sameas .  
  3 ?s geonames:inCountry ?inCountry . }
```



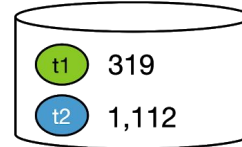
Virtual Integration - Query Decomposition & Source Selection

Задача - определить релевантные источники, содержащие искомые triple pattern и максимизировать полноту возвращаемых значений

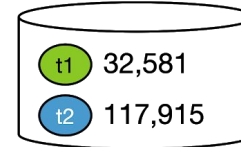
SELECT DISTINCT ?s WHERE {

t1 ?s foaf:page ?page .
t2 ?s owl:sameAs ?sameas .
t3 ?s geonames:inCountry ?inCountry . }

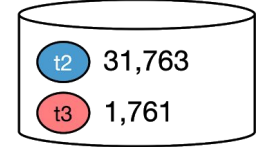
SWDF



Geonames



NYTimes

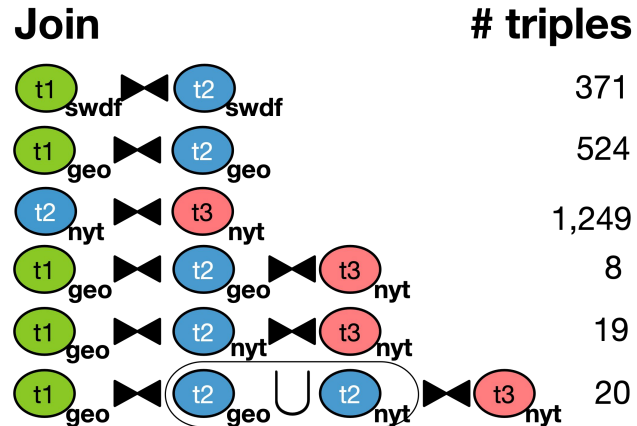
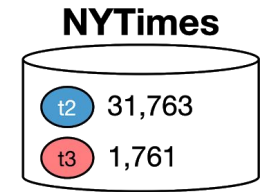
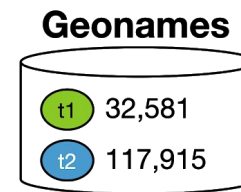
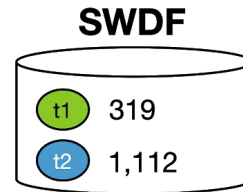


Virtual Integration - Query Decomposition & Source Selection

Задача - определить релевантные источники, содержащие искомые triple pattern и максимизировать полноту возвращаемых значений

SELECT DISTINCT ?s WHERE {

}

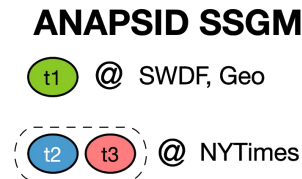
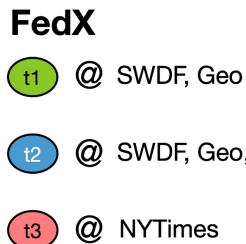
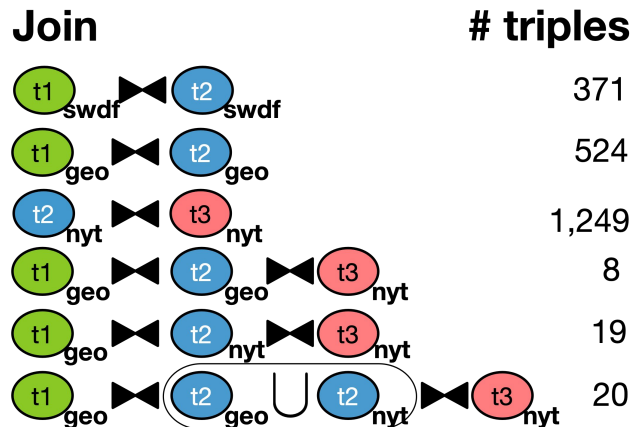
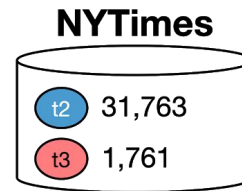
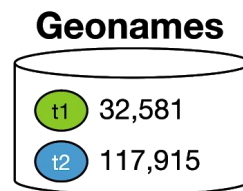
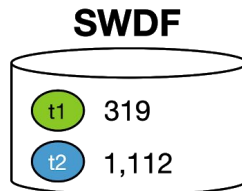


Virtual Integration - Query Decomposition & Source Selection

Задача - определить релевантные источники, содержащие искомые triple pattern и максимизировать полноту возвращаемых значений

SELECT DISTINCT ?s WHERE {

}



sec 239.4
 # triples 20

0.338
 0

88.9
 19

Virtual Integration - Query Planning

Задача - создать эффективный план выполнения запроса по выбранным источникам

```
SELECT DISTINCT * WHERE {  
t1 ?s dct:subject ?o1.    #Count: 115 259 581  
t2 ?s dbo:director ?o2.    #Count: 385 773  
t3 ?s dbo:genre dbr:Concert. #Count: 98  
t4 ?s dbo:artist dbr:Ana_Gabriel. #Count: 34  
t5 ?s dbo:genre dbr:Ranchera. } #Count: 205
```

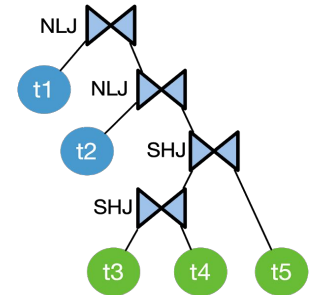
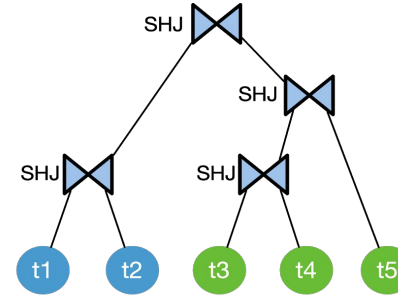
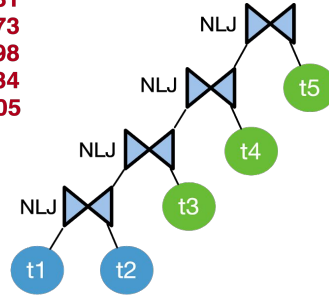
- low-selective pattern : count_max >> count_min
- high-selective pattern: count_min << count_max

Virtual Integration - Query Planning

Задача - создать эффективный план выполнения запроса по выбранным источникам

SELECT DISTINCT * WHERE {

t1 ?s dct:subject ?o1. #Count: **115 259 581**
t2 ?s dbo:director ?o2. #Count: **385 773**
t3 ?s dbo:genre dbr:Concert. #Count: **98**
t4 ?s dbo:artist dbr:Ana_Gabriel. #Count: **34**
t5 ?s dbo:genre dbr:Ranchera. } #Count: **205**

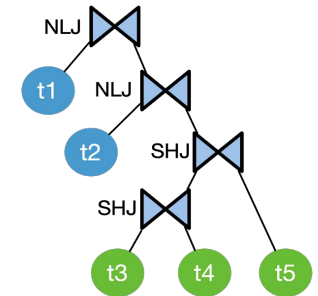
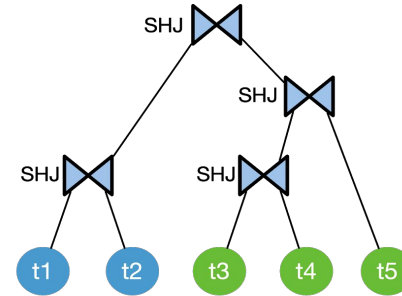
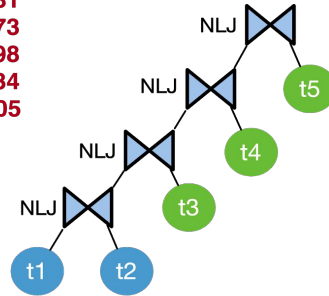


Virtual Integration - Query Planning

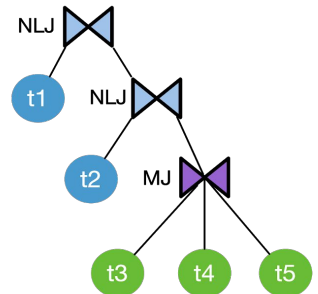
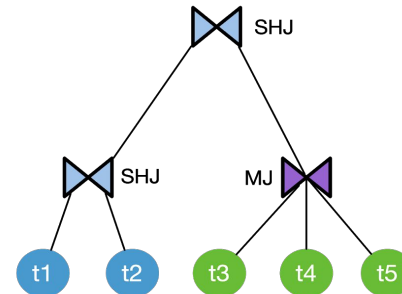
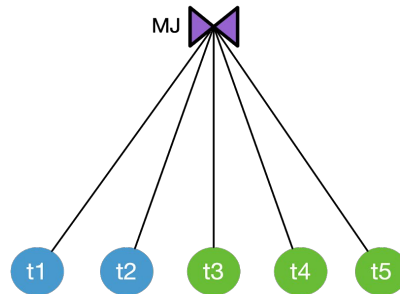
Задача - создать эффективный план выполнения запроса по выбранным источникам

SELECT DISTINCT * WHERE {

t1 ?s dct:subject ?o1. #Count: **115 259 581**
t2 ?s dbo:director ?o2. #Count: **385 773**
t3 ?s dbo:genre dbr:Concert. #Count: **98**
t4 ?s dbo:artist dbr:Ana_Gabriel. #Count: **34**
t5 ?s dbo:genre dbr:Ranchera. } #Count: **205**



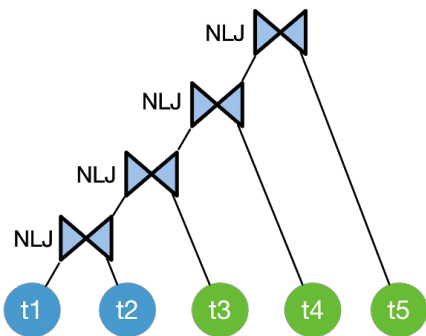
Plan	ET, ms	Results
NLJs, (b)	634	12
SHJs, (c)	timeout	0
NLJs + SHJs, (d)	243	12
MJ-only, (f)	timeout	0
MJ + SHJs, (g)	timeout	0
MJ + NLJs, (h)	195	12



Virtual Integration - Adaptive Query Execution

Задача - адаптировать план выполнения запроса к внешним воздействиям

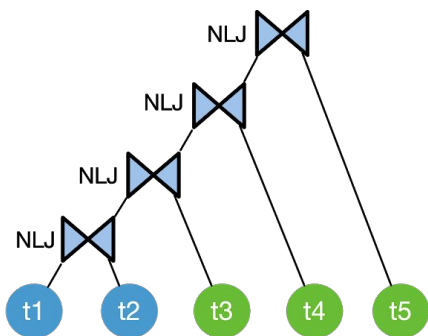
- Планы кумулятивные (результаты выдаются по мере появления, а не все сразу)
- Источники могут стать недоступными
- Сетевые задержки
- Динамическое изменение плана в процессе выполнения запроса



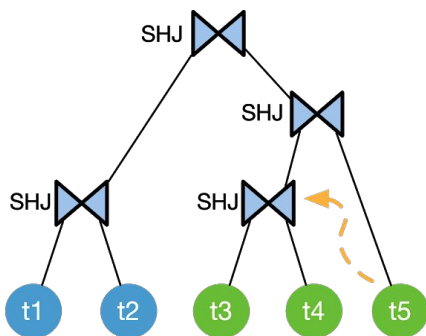
Virtual Integration - Adaptive Query Execution

Задача - адаптировать план выполнения запроса к внешним воздействиям

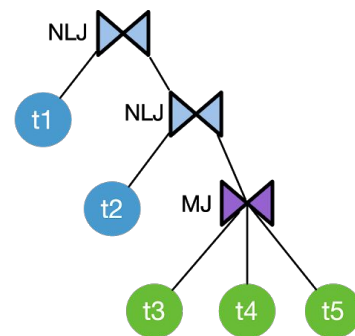
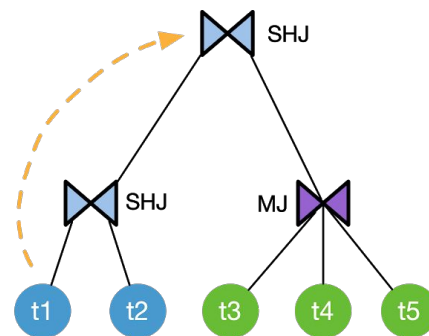
- Планы кумулятивные (результаты выдаются по мере появления, а не все сразу)
- Источники могут стать недоступными
- Сетевые задержки
- Динамическое изменение плана в процессе выполнения запроса



Изменение плана в целом



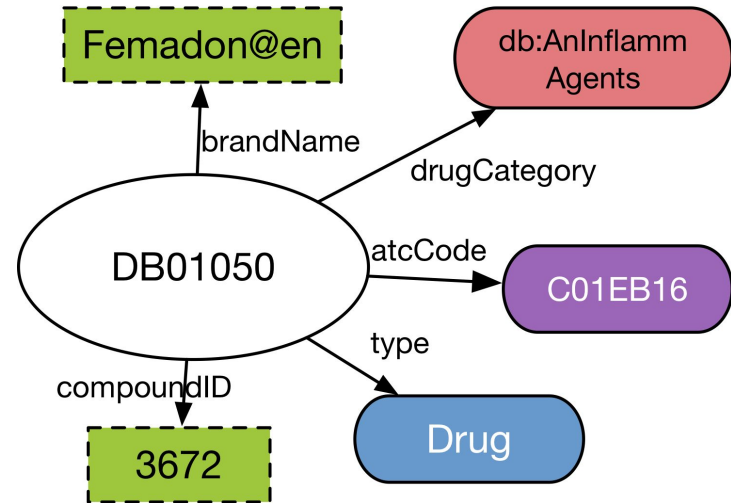
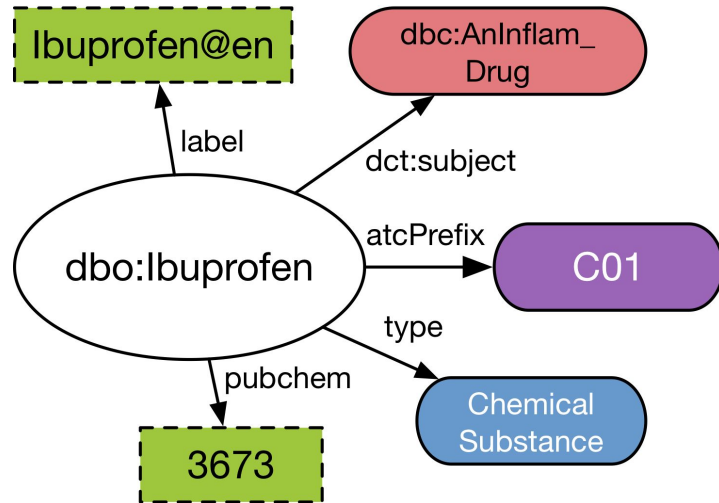
(t3, t4, t5)
Symmetric -> MJoin



Изменения порядка операторов

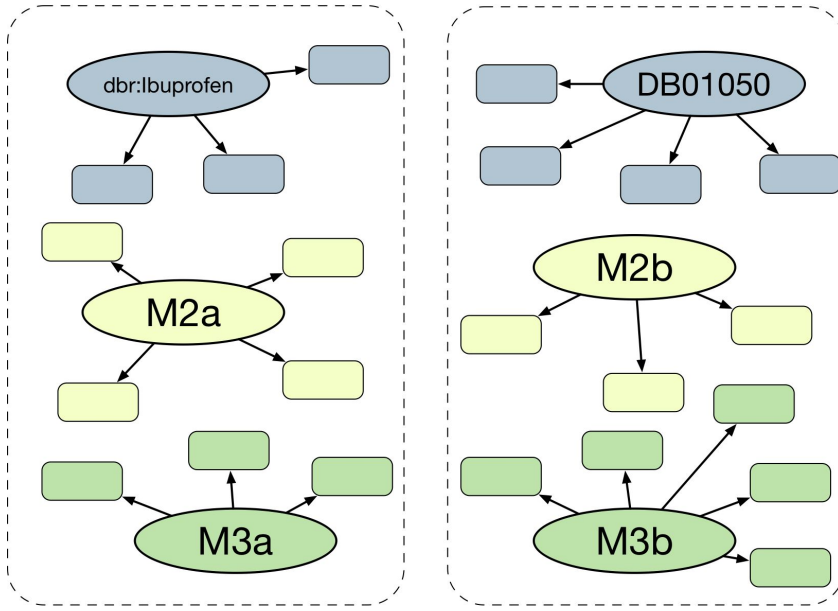
Virtual Integration - Results Reconciliation

Медиатор может выполнять вспомогательное агрегирование результатов для дедупликации



Virtual Integration - Results Reconciliation

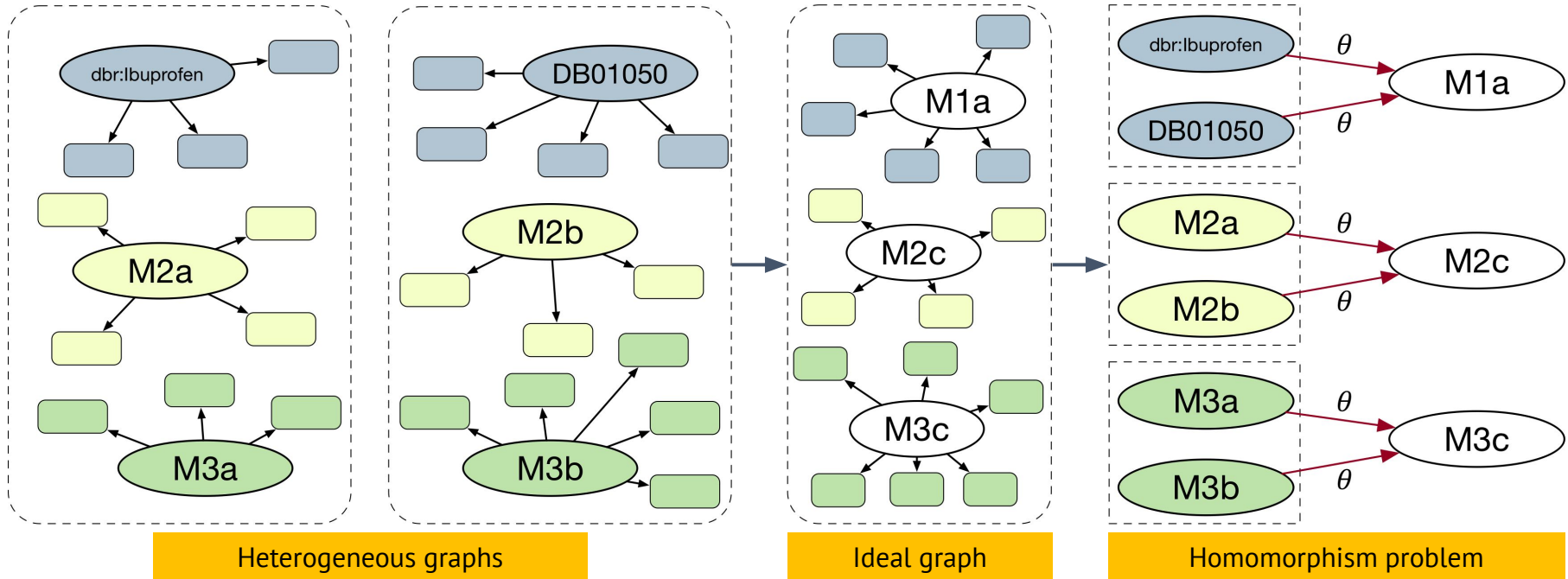
Медиатор может выполнять вспомогательное агрегирование результатов для дедупликации



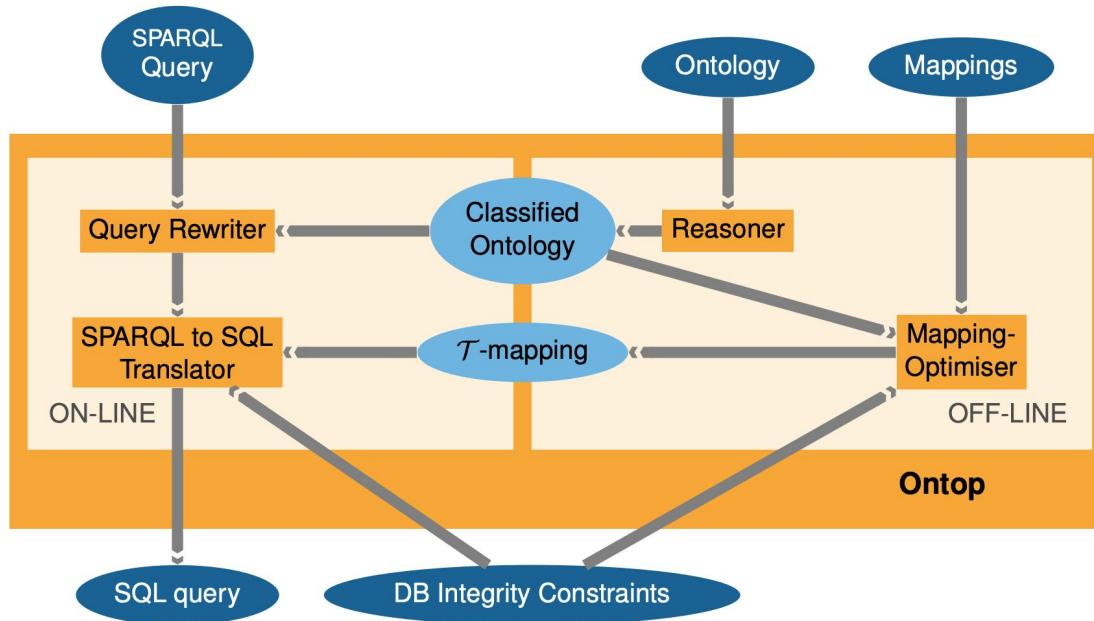
Heterogeneous graphs

Virtual Integration - Results Reconciliation

Медиатор может выполнять вспомогательное агрегирование результатов для дедупликации



Virtual Integration - Wrappers - SPARQL 2 SQL - Ontop



- Трансляция SPARQL в SQL на лету
 - Маппинги Quest / R2RML
1. Подготовительный этап (преобразование онтологии и маппингов в τ -mappings)
 2. Исполнительный этап (SPARQL -> SQL -> JDBC -> RDF)

Virtual Integration - Wrappers - SPARQL 2 SQL - Ontop

```
SELECT  ?tumor WHERE {  
    ?tumor rdf:type :Neoplasm ;  
           :hasStage :stage-IIIa .  
}
```

Virtual Integration - Wrappers - SPARQL 2 SQL - Ontop

```
SELECT ?tumor WHERE {  
  ?tumor rdf:type :Neoplasm ;  
          :hasStage :stage-IIIa .  
}
```

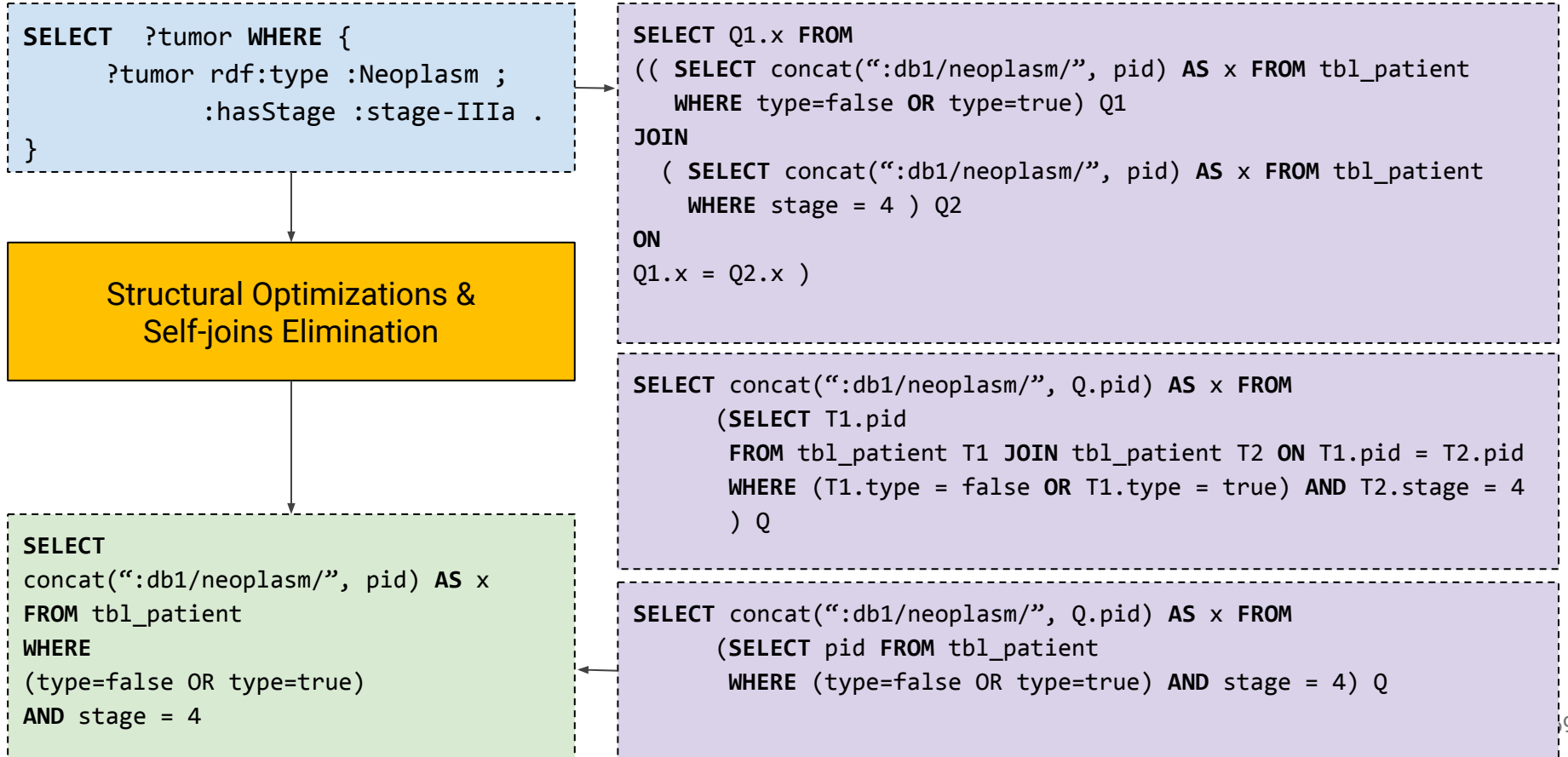
Structural Optimizations &
Self-joins Elimination

```
SELECT Q1.x FROM  
(( SELECT concat(":db1/neoplasm/", pid) AS x FROM tbl_patient  
  WHERE type=false OR type=true) Q1  
JOIN  
  ( SELECT concat(":db1/neoplasm/", pid) AS x FROM tbl_patient  
    WHERE stage = 4 ) Q2  
ON  
Q1.x = Q2.x )
```

```
SELECT concat(":db1/neoplasm/", Q.pid) AS x FROM  
(SELECT T1.pid  
  FROM tbl_patient T1 JOIN tbl_patient T2 ON T1.pid = T2.pid  
  WHERE (T1.type = false OR T1.type = true) AND T2.stage = 4  
  ) Q
```

```
SELECT concat(":db1/neoplasm/", Q.pid) AS x FROM  
(SELECT pid FROM tbl_patient  
  WHERE (type=false OR type=true) AND stage = 4) Q
```

Virtual Integration - Wrappers - SPARQL 2 SQL - Ontop



В следующей серии

1. Introduction
2. Представление знаний в графах - RDF & RDFS & OWL
3. Хранение знаний в графах - SPARQL & Graph Databases
4. Однородность знаний - RDF* & Wikidata & SHACL & ShEx
5. Интеграция данных в графы знаний - Semantic Data Integration
- 6. Введение в теорию графов - Graph Theory Intro**
7. Векторные представления графов - Knowledge Graph Embeddings
8. Машинное обучение на графах - Graph Neural Networks & KGs
9. Некоторые применения - Question Answering & Query Embedding