Правительство Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Санкт-Петербургский государственный университет»

Математическое обеспечение и администрирование информационных систем

Кафедра Системного программирования

Соковикова Светлана Алексеевна

Выявление типов объектов в графовой базе данных на основе кластеризации

Бакалаврская работа

Научный руководитель:
д.ф.-м.н., профессор Новиков Б.А.

Рецензент:
Смирнов К.К.

Санкт-Петербург
2017

SAINT-PETERSBURG STATE UNIVERSITY

Software Engineering Chair

Svetlana Sokovikova

Object type detecting in graph database by clustering

Bachelor's Thesis

Scientific superviser:
professor Boris Novikov

Reviewer:
Kirill Smirnov

Saint-Petersburg
2017

Оглавление

Введение……4

Постановка задачи……6

1. Обзор.....7

1.1 Обзор литературы……7

1.2 Описание используемых методов и технологий……...8

2. Архитектура инструментария……11

3. Модуль выявления типов объектов……..14

4. Конвертация в реляционную базу данных……...18

5. Оценка качества кластеризации……21

6. Апробация инструментария на DBLP……...23

7. Заключение……27

Список литературы……28

Введение

Данные, которые содержатся в Интернет, обычно ориентированы на восприятие людьми. И это затрудняет их машинную обработку. В связи с этим в настоящее время ведется работа над созданием семантической паутины (Semantic web) – глобальной семантической сети, формируемой на основе Интернета и содержащей информацию в виде, пригодном для машинной обработки.

Семантическая паутина – частный случай слабоструктурированных данных (linked data). Данные в этой модели хранятся в виде ориентированного графа с именованными дугами, вершины которого – хранимые объекты, а дуги – связи между объектами. Есть множество способов хранения слабоструктурированных данных: в виде графовой базы данных, XML-файла, файла RDF и другие. Также существуют специальные языки запросов для работы со слабоструктурированными данными: SPARQL для графовых баз данных, XPath для XML.

Недостаток всех способов хранения и языков запросов для слабоструктурированных данных – низкая эффективность в случае обработки данных большого размера. Отсюда необходимость в средствах повышения эффективности, при проектировании которых полезно помнить следующее обстоятельство: если база данных содержит миллионы узлов и связей, то каждый узел не будет резко отличаться от всех остальных. Напротив, узлы в этом случае можно разбить на кластеры таким образом, что узлы в каждом кластере будут иметь похожие по некоторой метрике наборы имен исходящих дуг. Иначе говоря, в больших слабоструктурированных данных неизбежно можно выделить структуру. Когда в данных присутствует структура, уже можно говорить об их представлении в реляционной модели. Исполнение запросов в реляционных базах данных оптимизировано лучше, чем в любых других, поэтому есть основания полагать, что перевод данных из слабоструктурированной модели в реляционную модель повысит

эффективность их обработки. Так, если удалось выделить кластеры, то данные об объектах каждого кластера можно поместить в одну таблицу реляционной базы данных.

В этой работе будет сделан шаг к реализации описанной выше идеи.

Постановка задачи

Целью этой работы является создание инструментария для выделения структуры в слабоструктурированных данных. Для достижения этой цели были выделены следующие задачи:

1. Разработка архитектуры инструментария.

2. Реализация модуля для выявления типов объектов в слабоструктурированных данных.

3. Реализация модуля конвертации слабоструктурированных данных в реляционную БД, в каждой таблице которой будут храниться данные об объектах одного типа.

4. Реализация модуля оценки качества кластеризации.

5. Апробация инструментария на базе данных научных публикаций DBLP.

1. Обзор

1.1 Обзор литературы

Для выявления типов объектов решено использовать кластеризацию. Кластеризация данных используется очень часто, и совершенно невозможно сделать полный обзор связанных с ней работ. Упомянем лишь некоторые из них, наиболее связанные с поставленной задачей.

Все работы по кластеризации основаны на известных алгоритмах, подробное описание которых приведено в [2]. Большинство алгоритмов требуют выбранной заранее меры расстояния на кластеризуемых векторах. Существует огромное разнообразие мер. Все меры, когда-либо применявшиеся на практике, приведены в [1]. Чтобы провести сравнение алгоритмов, что является одной из целей работы, необходима тщательно продуманная оценка качества кластеризации, которая подробно рассмотрена в [3].

В работе [5] рассматриваются усовершенствования методов кластеризации двоичных векторов и применение полученной кластеризации в системе рекомендаций медиаконтента. Статья [6] проводит исследование, касающееся и кластеризации, и графовой базы данных: составляется граф из авторов, чьи статьи опубликованы в DBLP, и с помощью нескольких алгоритмов кластеризации выполняется поиск сообществ (авторы, работающие на близкие темы, должны попасть в одно сообщество).

Графовые базы данных популярны в последнее время, и проводится много связанных с ними исследований. В моей работе выполняется перевод данных из сетевой модели в реляционную, в статье [8] рассматривается смежная тема – сравнение времени исполнения запросов в графовой и реляционной базах данных. Конечно, для того чтобы ускорить исполнение запросов в слабоструктурированных данных, можно не только перевести их в реляционную модель, но и разработать качественный инструментарий для

графовых баз данных, что и сделано в [7]. Также способ хранения слабоструктурированных данных предложен в [4].

Ещё один аспект, для которого может быть полезна эта работа – хранение RDF. RDF-формату посвящено достаточно много исследований. В [9] подробно рассмотрено несколько способов хранения RDF, с использованием реляционных и нереляционных (NoSQL) баз данных. Для хранения и исполнения запросов на больших объемах RDF-данных используются распределенные хранилища. Схема распределенного хранения и эффективного исполнения запросов предложена в [10].

1.2 Описание используемых методов и технологий

В рамках работы создается инструментарий для анализа слабоструктурированных данных, представленных либо в виде графовой базы данных, либо в формате XML.

В качестве графовой БД используется Neo4j. Neo4j база данных – это набор узлов и направленных связей между ними. Каждый узел имеет id, может иметь или не иметь свойства. Также узел может иметь или не иметь метки, но они неважны в этой работе. Каждая связь имеет тип, а также может иметь или не иметь свойства, но свойства связей для этой работы роли не играют. На рис.1 приведен небольшой пример Neo4j базы данных.

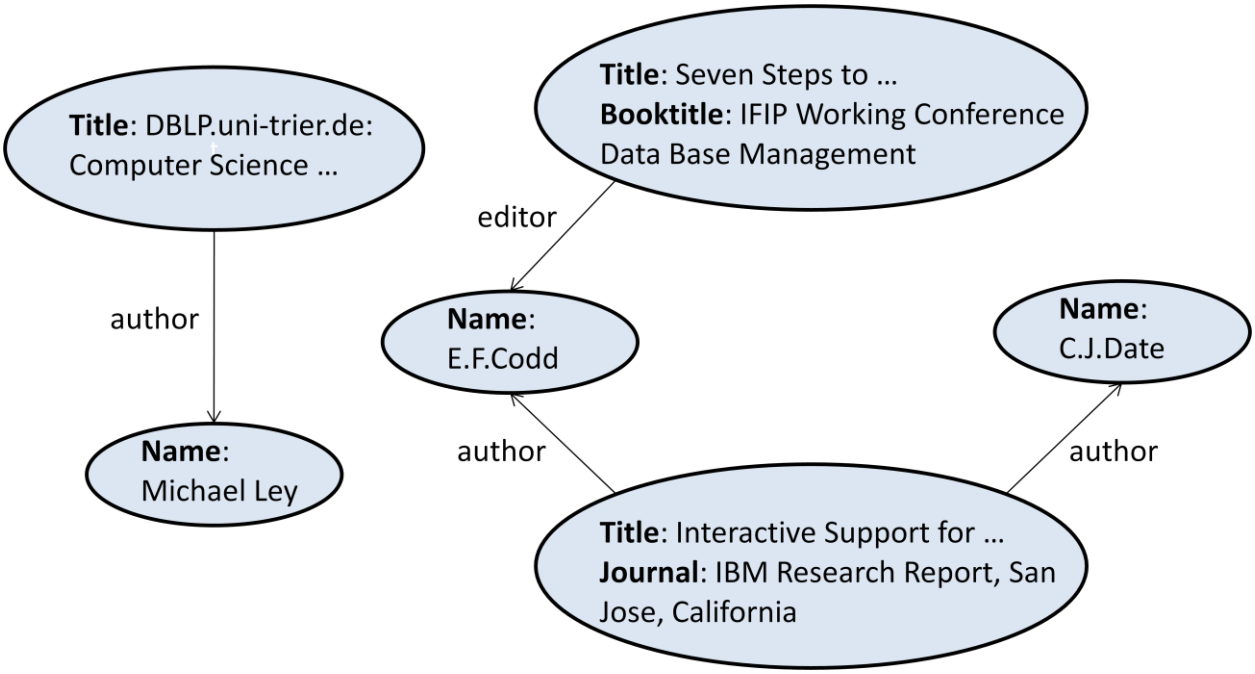

Рис. 1. Пример графовой базы данных

На Рис. 1 свойства узлов выписаны внутри самих узлов, типы связей подписаны рядом со связями.

Интерфейс с Neo4j базой данных осуществляется посредством следующего репозитория (используется утилита Maven, приведена одна из зависимостей файла pom.xml):

```
<groupId>org.neo4j</groupId>
<artifactId>neo4j</artifactId>
<version>3.1.0</version>
```

Для чтения данных формата XML используется XML-парсер StAX. В языке Java существует три XML-парсера: DOM, SAX, StAX. DOM пригоден только для работы с файлами небольшого размера, так как он одновременно записывает в память данные из всего XML-файла. Ввиду того, что материал для апробации – DBLP – представлен XML-файлом размера 1.8 Gb, DOM в этой работе неприемлем. Выбор между SAX и StAX был сделан в пользу StAX, так как он гораздо более удобен с точки зрения синтаксиса.

Непосредственно кластеризация осуществляется с использованием алгоритмов, реализованных в языке R. Обмен данными между Java и R осуществляется по следующей схеме: данные записываются в csv-файл Java-утилитой, полученный файл считывается в языке R, с данными производятся необходимые действия, результаты которых записываются из среды R в другой csv-файл.

Слабоструктурированные данные конвертируются в реляционную БД, в качестве которой используется PostgreSQL.

Для взаимодействия с PostgreSQL в файле pom.xml прописывается следующая зависимость:

```
<groupId>org.postgresql</groupId>
<artifactId>postgresql</artifactId>
<version>9.4.1212.jre7</version>
```

2. Архитектура инструментария

Инструментарий позволяет работать со слабоструктурированными данными, представленными либо в фомате XML, либо в графовой базе данных Neo4j. Слабоструктурированные данные – это набор объектов и связей между ними, где каждый объект может иметь свойства. Общая схема работы выглядит следующим образом: для каждого объекта специальным образом составляется битовая шкала, затем этот список битовых шкал кластеризуется. Диаграммы потоков данных, реализующие эту схему для XML и для Neo4j базы данных, выглядят несколько по-разному.

На рис.2, 3 приведены диаграммы потоков данных для Neo4j базы данных.

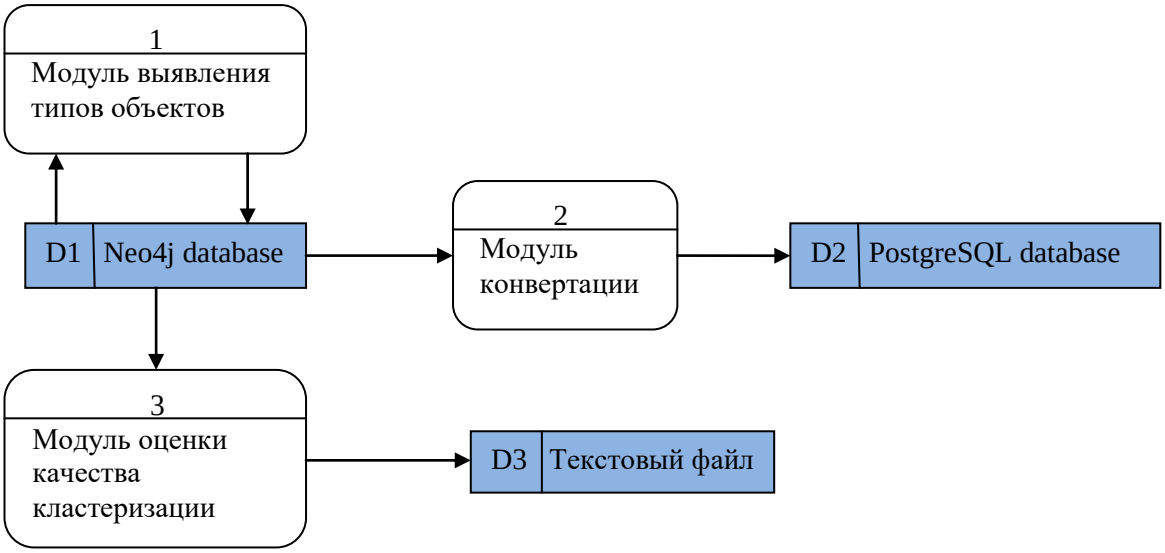

Рис. 2. Диаграмма потоков данных на уровне подсистем для Neo4j

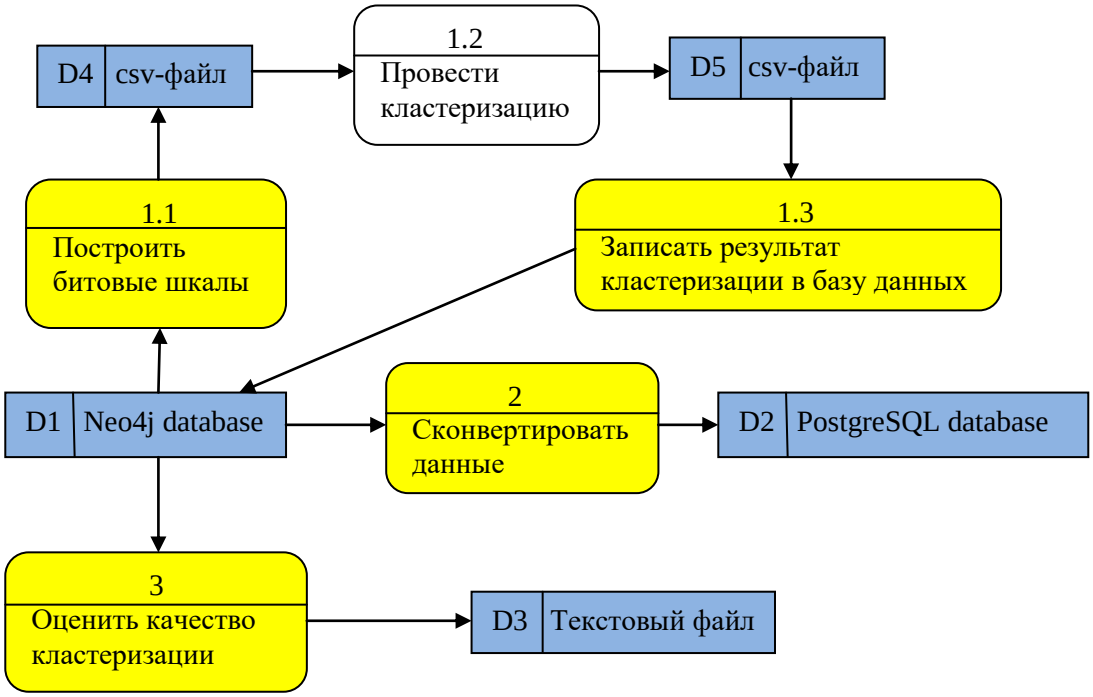

Рис. 3. Диаграмма потоков данных на уровне процессов для Neo4j

На рис. 2 представлена диаграмма на уровне подсистем, на рис. 3 – более детальная диаграмма на уровне процессов. Компоненты для реализации процессов, отмеченных желтым цветом, были созданы в рамках этой работы.

Диаграммы для данных, представленных в формате XML, изображены на рис.4, 5.

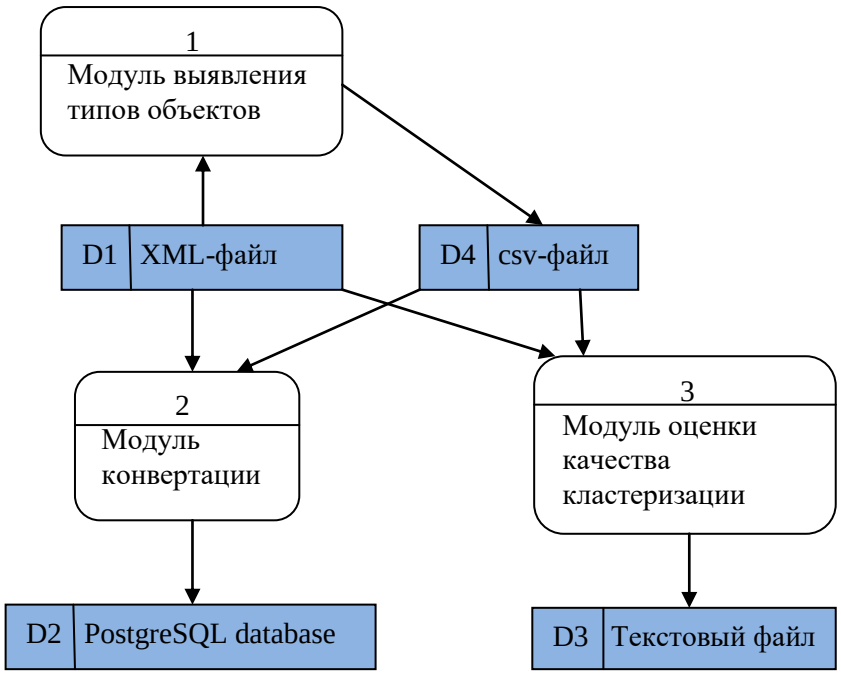

Рис. 4. Диаграмма потоков данных на уровне подсистем для XML

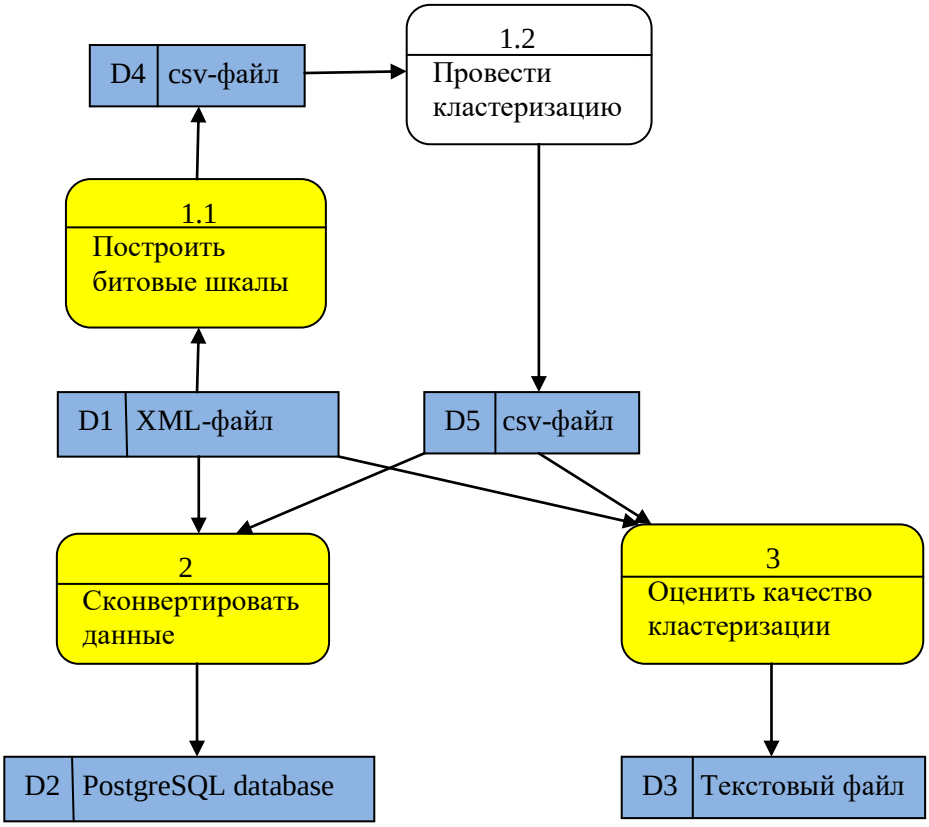

Рис. 5. Диаграмма потоков данных на уровне процессов для XML

3. Модуль выявления типов объектов

Напомним, что слабоструктурированные данные – это набор объектов, которые могут иметь свойства, и направленных именованных связей между ними. Атрибуты объекта могут быть представлены его свойствами и исходящими из него связями. Если атрибут представлен свойством, то название и значение атрибута – это название и значение свойства соответственно. Если же атрибут представлен исходящей связью, то название атрибута – это имя связи, а значение атрибута – это id объекта, на который направлена связь.

Для Neo4j базы данных дело обстоит понятным образом: объект – это узел, связи и свойства – это ровно те же связи и свойства, которые есть в Neo4j. Однако в XML-файле нет ни явных объектов, ни явных связей и свойств, и нужно объяснить, что они в этом случае значат. Будем представлять XML данные в виде дерева. Если внутри тега нет вложенных тегов, то этому тегу соответствует лист, иначе – нелистовая вершина. Для примера приведем XML-скрипт и соответствующее ему дерево, изображенное на рис.6:

```
<person key = ON>
<name> Olga </name>
<surname> Nechaeva </surname>
<child key = AS>
   <name> Anna </name>
   <surname> Smirnova </surname>
   <year> 2000 </year>
</child>
<child key = AnS>
   <name> Anastasia </name>
   <surname> Smirnova </surname>
```

<year> 2004 </year>
</child>
<year> 1974 </year>
<work> School 29 </work>
<work> School 41 </work>
</person>

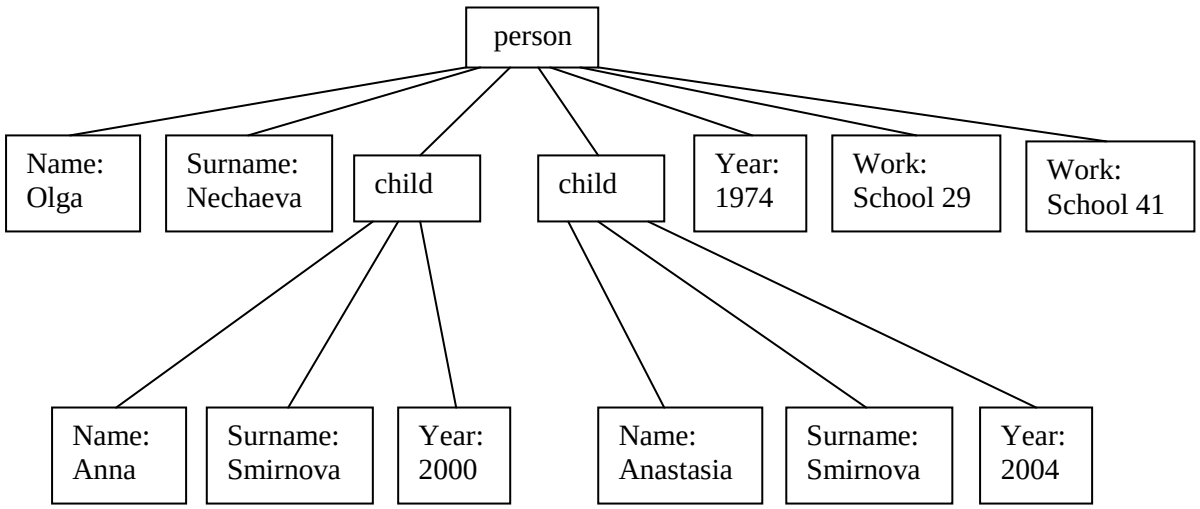

Рис. 6. Дерево, соответствующее приведенному выше скрипту

Теперь все вершины кроме листьев – это объекты, связи между нелистовыми вершинами – это связи между соответствующими объектами, причем в направлении от предка к потомку, имя потомка становится именем связи, вершины-листья – это атрибуты объектов-родителей (здесь у объекта вполне может быть несколько одноименных атрибутов), а атрибуты тега (в данном случае атрибут тега – это key; не путать атрибуты тега с атрибутами объекта!) становятся атрибутами соответствующих объектов. Согласно этому правилу, дереву, приведенному на рис.4, соответствует графовая база данных, приведенная на рис.7.

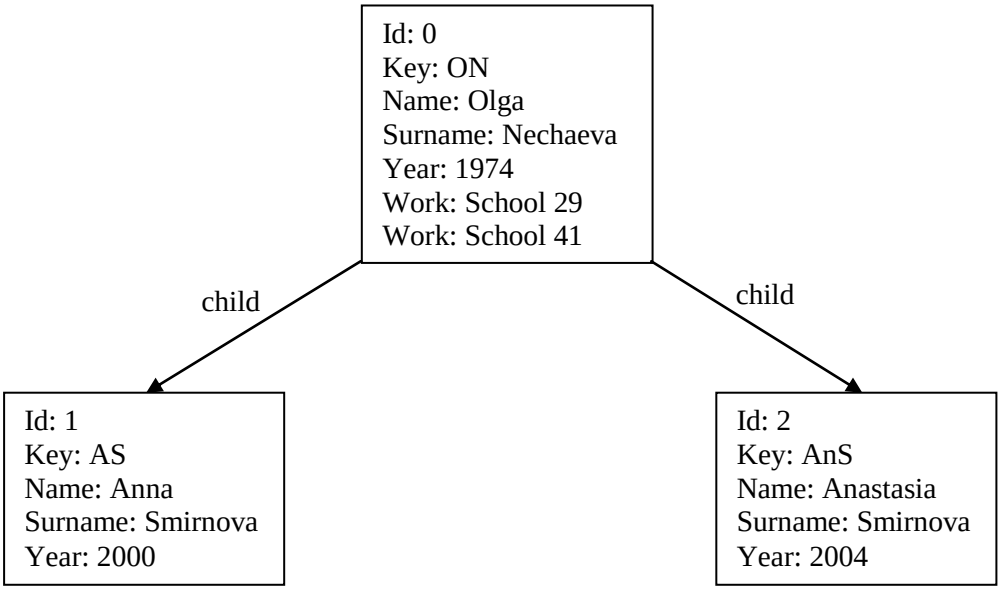

Рис.7. Графовая база данных, соответствующая дереву рис.6

Здесь и далее данные формата XML представляются в виде объектов и связей именно по таким правилам.

Теперь, договорившись о моделях представления данных, можно переходить непосредственно к работе модуля выявления типов объектов. Типы объектов считаются одинаковыми, если объекты имеют «похожие» наборы атрибутов. Значения атрибутов роли не играют.

В этой работе наборы атрибутов представляются битовыми шкалами. Сначала упорядочиваются все атрибуты, которые есть в базе данных (это все названия свойств и все имена связей). Затем для каждого объекта строится битовая шкала, длина которой равна количеству атрибутов в базе данных. Правило постороения шкалы следующее: если номер места равен порядковому номеру атрибута, который у объекта имеется, то на этом месте ставится 1, на всех остальных местах ставится 0. Полученный список битовых шкал записывается в csv-файл. Построение битовых шкал выполняется компонентом, реализованным в рамках этой работы на языке Java.

Следующий шаг – собственно кластеризация. Кластеризация выполняется в среде R. На вход подается набор битовых шкал, полученный

на предыдущем шаге и записанный в виде csv-файла. Результат кластеризации записывается в новый csv-файл в виде столбца. i-е число в этом столбце означает номер кластера, к которому отнесена i-я битовая шкала (точнее объект, соответствующий этой шкале). Из большого разнообразия алгоритмов кластеризации выбран алгоритм k-средних с евклидовой метрикой, обладающий очень важным для этой работы преимуществом: он не использует матрицу попарных расстояний между векторами. Даже для 1% коллекции DBLP не удалось запустить алгоритм иерархической кластеризации по причине того, что для этого нужна матрица попарных расстояний, которую R записывает в память, а она уже для 1% данных DBLP становится слишком велика и не помещается в память. Если слабоструктурированные данные представлены в XML-формате, то работа модуля выявления типов объектов на этом шаге заканчивается.

Если данные представлены в виде Neo4j базы данных, то последний шаг в работе этого модуля – запись информации о типах объектов в базу данных. Номера кластеров считываются из записанного после кластеризации csv-файла и записываются в специальное свойство в каждый объект.

4. Конвертация в реляционную базу данных

Напомним, что после выявления типов (см. предыдущую главу) каждому объекту соответствует тип, представленный числом. Он означает номер кластера, к которому относится объект.

Конвертация в реляционную базу данных спроектирована следующим образом.

Сначала в реляционной базе данных создается таблица navigate, состоящая из двух столбцов: id объекта и его тип для того, чтобы, зная id объекта, иметь возможность быстро определить, в какой таблице хранятся данные о нем.

Далее для каждого типа объектов создается по одной таблице. Каждому атрибуту, который может иметь объект определенного типа, соответствует столбец в таблице, созданной для этого типа.

Если атрибут представлен свойством, то название соответствующего столбца совпадает с названием свойства. Если атрибут представлен связью, то значением атрибута является другой объект. В этом случае название столбца получается добавлением к типу дуги постфикса _id и в этом столбце хранятся именно id. Последний принцип применяется по двум причинам. Во-первых, id всегда уникален и по нему можно однозначно идентифицировать объект. Во-вторых, возможна ситуация, представленная на рис.8:

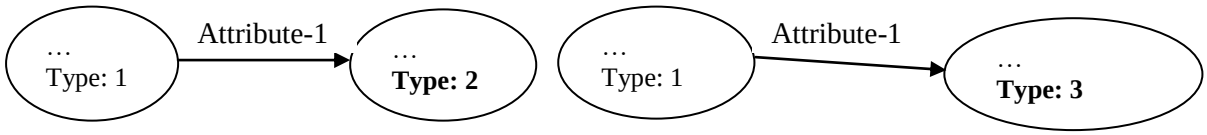

Рис.8. Возможная конфигурация типов объектов и связей между ними

Т.е. в таблице, соответствующей типу 1 (объекты, у которых Type = 1), в столбце Attribute-1 должны оказаться ссылки на объекты из разных таблиц (т.к. разное значение Type). Отсюда невозможность сделать Attribute-1 внешним ключом (foreign key).

Если у одного объекта может быть несколько одноименных атрибутов, то все присутствующие значения атрибута с этим именем будут храниться в соответствующем столбце таблицы в виде массива jsonb.

Каждый объект записывается в таблицу, соответствующую его типу, вместе со всеми своими атрибутами.

Поясним описанную схему конвертации на примере. На рис. 9 представлена графовая база данных для конвертации. Овалами обозначены объекты, внутри объектов перечислены их свойства, связи между объектами представлены направленными ребрами, типы связей подписаны рядом со связями. Тип каждого объекта представлен свойством type.

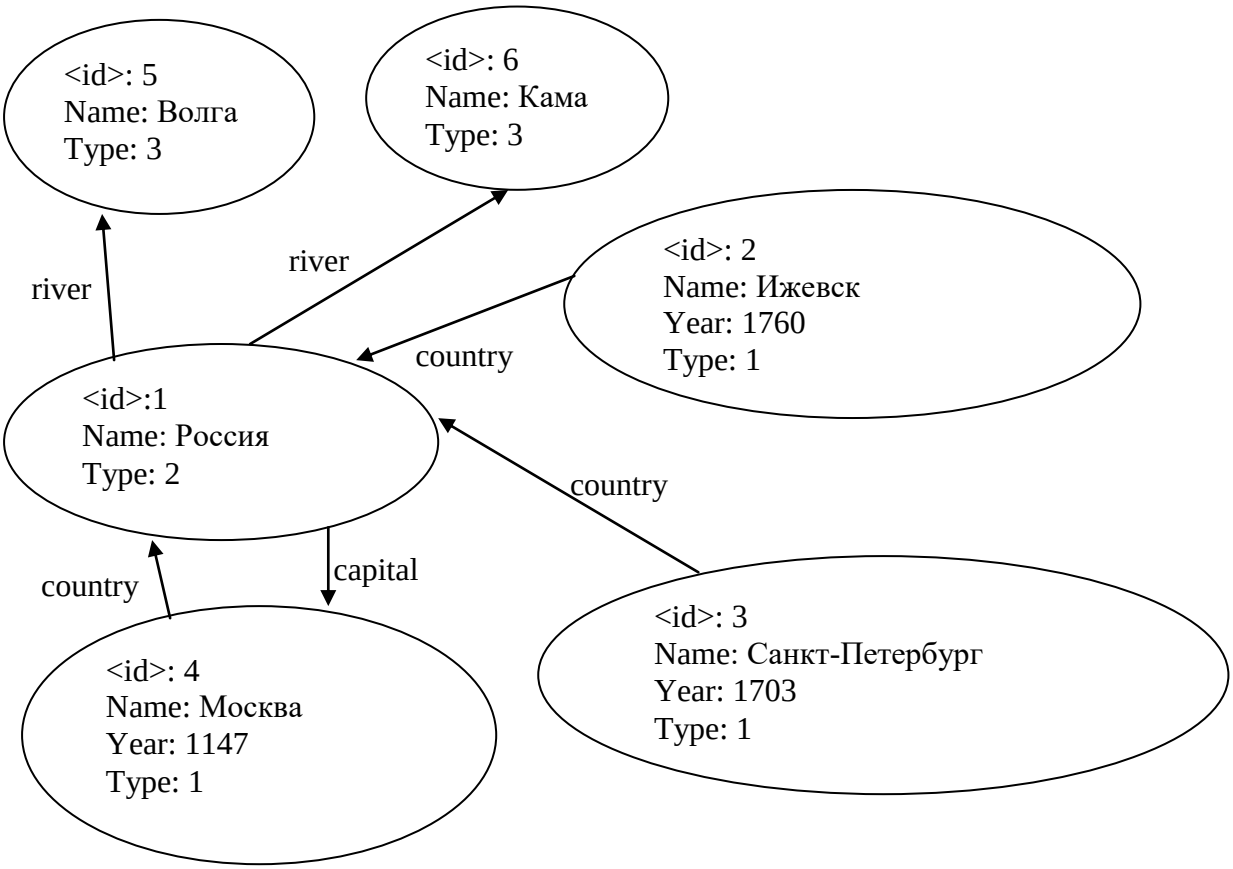

Рис. 9. Графовая база данных для конвертации

В приведенной базе данных type может принимать значения 1, 2 или 3, что соответствует трем типам объектов. Значит, должны быть созданы 3 таблицы: табл.1, табл.2, табл.3.

Табл. 1. Объекты типа 1 (Type = 1)

Id	Name	Year	Country_id
2	Ижевск	1760	1
3	Санкт-Петербург	1703	1
4	Москва	1147	1

Табл. 2. Объекты типа 2 (Type = 2)

Id	Name	Capital_id	River_id
1	Россия	4	[5,6]

Табл. 3. Объекты типа 3 (Type = 3)

Id	Name
5	Волга
6	Кама

Атрибуты country, river и capital представлены связями, поэтому, согласно правилу, им соответствуют столбцы country_id, river_id и capital_id, в которых хранятся id объектов. river_id – множественный атрибут, соответственно, его значение хранится в виде массива jsonb.

5. Модуль оценки качества кластеризации

В этой работе используется только внешняя оценка качества кластеризации, то есть правильное разбиение объектов по типам известно заранее. Модуль оценки качества работает либо с базой данных Neo4j, где у каждого объекта есть свойство, представляющее его полученный при кластеризации тип, и свойство, представляющее его идеальный тип, либо с xml-файлом и полученным при кластеризации csv-файлом (в этом случае идеальный тип объекта – это имя тега). Результат работы модуля записывается в текстовый файл.

Результатом работы модуля является таблица соответствия кластеров идеальным типам и несколько величин, характеризующих качество кластеризации.

Таблица соответствия строится по следующим правилам. Имена всех столбцов кроме первого – это настоящие типы объектов.

Табл. 4. Пример таблицы соответствия кластеров идеальным типам

Номер кластера	Incollection	www	Book	Proceedings	Inproceedings	Phdthesis	Article
1	3729	1	29	1569	97061	0	0
2	33	1802631	45	0	10	0	9
3	0	9	1	1	0	0	143180
4	287	5	1839	1409	30462	11	2
5	34942	0	899	29082	1755791	0	245
6	594	1	9983	3	9	34951	6
7	3	0	1	0	16	0	1389912

Обведенное число означает, что к 5-му кластеру было отнесено 1755791 объектов, имеющих настоящий тип inproceedings. Числа в остальных ячейках интерпретируются аналогично.

Теперь о величинах, характеризующих качество. Сначала введем 4 вспомогательных величины: SS – количество пар объектов, имеющих один правильный тип и отнесенных к одному кластеру, SD – количество пар объектов, имеющих один правильный тип и отнесенных к разным кластерам, DS – количество пар объектов, имеющих разные правильные типы и отнесенных к одному кластеру, DD – количество пар объектов, имеющих разные правильные типы и отнесенных к разным кластерам. Нетрудно видеть, что SS, SD, DS, DD могут быть легко вычислены по таблице соответствия. Качество кластеризации характеризуется следующими величинами:

Rand statistic = $\frac{SS + DD}{SS+SD+DS+DD}$,

Jaccard index = $\frac{SS}{SS+SD+DS}$,

Folkes and Mallows index = $\sqrt{\frac{SS}{SS+SD} * \frac{SS}{SS+DS}}$.

Очевидно, значения всех трех величин в случае идеальной кластеризации равны 1, и качество реальной кластеризации тем выше, чем ближе к 1 эти величины.

6. Апробация инструментария на DBLP

Апробация созданного инструментария проводилась на коллекции публикаций DBLP [http://dblp.org/xml/release/dblp-2016-11-02.xml.gz]. Эта база данных занимает 1.8 Gb и содержит 5338770 публикаций.

Апробация проводилась по следующим шагам. На первом шаге инструментарий получал на вход 1% публикаций, на втором шаге – 5%, на третьем шаге – полную коллекцию. Возможность выделения доли из полной коллекции предусмотрена в созданном инструментарии и реализована в каждом модуле по следующей схеме. Сначала собираются данные о количестве объектов каждого из «правильных» типов в полном объеме коллекции, затем полученные цифры умножаются на долю, которую необходимо извлечь для рассмотрения, и таким образом вычисляется количество объектов каждого из «правильных» типов, которое должно быть извлечено и рассмотрено. Далее при работе с коллекцией ведется подсчет извлеченных объектов каждого из «правильных» типов. Если количество извлеченных объектов определенного типа совпадает с количеством объектов этого типа, которое должно быть извлечено, то объект этого типа игнорируется, иначе – извлекается.

Рассмотрим первый шаг. Инструментарий запускался на 1% всех публикаций. Результат представлен в табл. 5 (интерпретация таблицы подробно описана в предыдущей главе).

Табл. 5. Таблица соответствия, полученная при кластеризации 1% коллекции DBLP

Номер кластера	Incollection	www	Book	Proceedings	Inproceedings	Phdthesis	Article
1	0	38	1	2	0	0	14
2	0	17988	0	0	0	0	6
3	0	0	126	86	0	0	0
4	0	0	0	232	0	0	0
5	395	0	0	0	18833	0	0
6	0	0	0	0	0	349	0
7	0	0	0	0	0	0	15313

Величины, характеризующие качество кластеризации, имеют следующие значения:

Rand statistic = 1.012,

Jaccard index = 0.982,

Folkes and Mallows index = 0.991.

Рассмотрим второй шаг. Было взято 5% всех публикаций. Получившийся результат представлен в табл. 6.

Табл. 6. Таблица соответствия, полученная при кластеризации 5% коллекции DBLP

Номер кластера	Incollection	www	Book	Proceedings	Inproceedings	Phdthesis	Article
1	0	0	15	264	0	0	0
2	0	90125	0	0	0	0	13
3	0	5	343	605	2	0	0
4	0	0	130	734	0	0	0
5	1979	1	0	0	94165	0	0
6	0	1	151	0	0	1748	4
7	0	0	0	0	0	0	76650

При этом получились следующие характеристики качества:

Rand statistic = 1.217,

Jaccard index = 0.785,

Folkes and Mallows index = 0.886.

И, наконец, на третьем шаге была взята вся коллекция. Результат представлен в табл. 7.

Табл. 7. Таблица соответствия, полученная при кластеризации полной коллекции DBLP

Номер кластера	Incollection	www	Book	Proceedings	Inproceedings	Phdthesis	Article
1	3729	1	29	1569	97061	0	0
2	33	1802631	45	0	10	0	9
3	0	9	1	1	0	0	143180
4	287	5	1839	1409	30462	11	2
5	34942	0	899	29082	1755791	0	245
6	594	1	9983	3	9	34951	6
7	3	0	1	0	16	0	1389912

Получившиеся величины, характеризующие качество:

Rand statistic = 0.924,

Jaccard index = 0.792,

Folkes and Mallows index = 0.886.

7. Заключение

В рамках работы были достигнуты следующие результаты.

1. Разработана архитектура инструментария.

2. Реализован на языке Java с использованием среды R модуль выявления типов объектов в графовой базе данных Neo4j и в формате XML.

3. Реализован на языке Java модуль конвертации из базы данных Neo4j и из XML в реляционную базу данных.

4. Реализован на языке Java модуль оценки качества кластеризации.

5. Проведена апробация созданного инструментария на известной базе данных научных публикаций DBLP.

Список литературы

[1] Seung-Seok Choi, Sung-Hyuk Cha, Charles C. Tappert, (2010), “A Survey of Binary Similarity and Distance Measures”, Systemics, Cybernetics and Informatics, Vol.8, No.1

[2] А.А.Барсегян, М.С.Куприянов, «OLAP и Data Mining», БХВ-Петербург, 2004, стр. 149 – 207

[3] Е.В.Сивоголовко, «Оценка обоснованности кластеризации для данных высокой размерности», дипломная работа, СПбГУ, Математико-Механический факультет, кафедра Системного Программирования, Санкт-Петербург, 2008

[4] I. Nekrestyanov, B. Novikov, and E. Pavlova, “An analysis of alternative methods for storing semistructured data in relations.” In Proc. of the ADBIS’2000, volume 1884 of Lecture Notes in Computer Science, pages 354–361, Prague, Chech, September 2000

[5] Lyle H. Ungar and Dean P. Foster, “Clusterig Methods for Collaborative Filtering”, AAAI Technical Report WS-98-08

[6] Laurel Orr and Jennifer Ortiz, “Clustering with the DBLP Bibliography to Measure External Impact of a Computer Science Research Area”, доступен по ссылке http://homes.cs.washington.edu/~jortiz16/images/MLProjectPaper.pdf

[7] Seyed-Mehdi-Reza Beheshti, Boualem Benetellah and Hamid Reza Motahari-Nezhad, “Scalable graph-based OLAP analytics over process execution data”, Distributed and Parallel Databases, September 2016, Volume 34, Issue 3, pp. 379-423

[8] Jeevan Joishi, Ashlish Sureka, “Graph or Relational Databases: A Speed Comparison for Process Mining Algorithm”, CoRRabs/1701.00072 (2017)

[9] Zongmin Ma, Miriam A.M.Capretz and Li Yan, “Storing massive Resource Description Framework (RDF) data: a survey”, Knowledge Eng. Review 31 (4), pp.391-413 (2016)

[10] Jung-Ho Um, Seungwoo Lee, Taehong Kim, Chang-Hoo Jeong, Sa-Kwang Song and Hanmin Jung, “Distributed RDF Store for efficient searching billions of triples based on Hadoop”, The Journal of Supercomputing 72(5), pp. 1825-1840 (2016)