Среда интеграции больших неоднородных коллекций данных[1]

УДК 004.652

В. И. Будзко – д.т.н., профессор, зам. директора по научной работе ИПИ РАН, E-mail: vbudzko@ipiran.ru

Л. А. Калиниченко – д.ф.-м.н., профессор, зав. лабораторией, ИПИ РАН, E-mail: leonidandk@gmail.com

С. А. Ступников – к.т.н., с.н.с. ИПИ РАН, E-mail: ssa@ipi.ac.ru

А. Е. Вовченко - к.т.н., с.н.с. ИПИ РАН, E-mail: alexey.vovchenko@gmail.com

Д. О. Брюхов – к.т.н., с.н.с. ИПИ РАН, E-mail: brd@ipi.ac.ru

Д. Ю. Ковалев – прог. ИПИ РАН, E-mail: dm.kovalev@gmail.com

В работе рассматривается подход к организации среды интеграции неоднородных коллекций данных различного вида (структурированных, слабоструктурированных и неструктурированных). Основная идея подхода состоит в объединении возможностей технологии предметных посредников и свободно распространяемой платформы распределенного хранения и обработки данных Hadoop, а также системы организации реляционных хранилищ данных над Hadoop, в качестве которой могут использоваться платформы IBM Big SQL или Hive. Обсуждаются вопросы разрешения сущностей (Entity Resolution) и слияния данных (Data Fusion) в

[1] Работа выполнена при поддержке РФФИ (гранты 13-07-00579, 14-07-00548), Президиума РАН (Программа фундаментальных исследований Президиума РАН № 16 «Фундаментальные проблемы системного программирования», ИПИ РАН (Тема 38.25 «Спецификация и решение задач анализа данных в концептуальных терминах предметных областей с интенсивным использованием данных» государственного задания ФГБУН ИПИ РАН).

контексте интеграции больших данных в среде Hadoop. Дан краткий обзор методов извлечения информации из текстов. Приведены примеры способов программирования методов извлечения информации из текстов на языке AQL и методов слияния данных на языке высокого уровня HIL. Рассмотрен пример задачи интеграции неоднородных коллекций данных в предлагаемой среде.

Ключевые слова: интеграция данных, большие данные, платформы распределенного хранения и обработки данных, разрешение сущностей, слияние данных, предметные посредники

Keywords: data integration, big data, software frameworks for distributed storage and processing of data, entity resolution, data fusion, subject mediators

1. Введение

Новая парадигма [1] в науке и информационных технологиях, доминирующая в последнее время, основывается на исследовании данных (data exploration). Роль данных особо подчеркивается и становится критической. Объемы данных фактически во всех областях деятельности растут со временем экспоненциально. Поэтому новая парадигма требует создания методов и средств оперирования данными, объемы которых выходят за рамки возможностей технологий баз данных, развивавшихся в последние десятилетия (преимущественно реляционных). Необходимы подходы, позволяющие справляться с разнообразием моделей данных (включая неструктурированные и слабоструктурированные данные), метаданных, семантики данных.

К моделям данных, появившимся в последнее время, относятся разнообразные NoSQL-модели: документные модели (например, MongoDB), модели с колоночным хранением (например, HBase), модели «ключ-значение» (например, Voldemort). Развиваются онтологические и семантические модели, такие, как семейства RDF и OWL; графовые модели (например, Neo4j); модели, основанные на многомерных массивах – ММ-модели (SciDB).

Методы создания унифицированного представления различных видов нетрадиционных моделей в канонической информационной модели (общем языке, унифицирующем языки разнообразных моделей данных) в последнее время активно исследовались в ИПИ РАН. Были рассмотрены подходы к унификации моделей данных различных классов: семантических (OWL [2], RDF [3]), графовых [4], ММ-моделей [5], NoSQL-моделей [6]. В качестве ка-

нонической модели рассматривался язык СИНТЕЗ [7], поддерживающий комбинированную слабоструктурированную (фреймовую) и объектную модель данных.

Для поддержки новых моделей данных создаются системы управления данными, обладающие масштабируемостью, высокой доступностью, возможностью разбиения коллекций данных произвольным образом на разделы для параллельной обработки.

Растущая популярность использования слабоструктурированных баз данных (в различных видах NoSQL) наряду с реляционными базами, в совокупности с технологиями программирования Hadoop и MapReduce, обеспечивающими параллельную обработку огромных массивов слабоструктурированных данных, объясняется множеством фактических и потенциальных применений. Например, развитие Веб-приложений, социальных сетей, сенсорных сетей, финансовых и торговых приложений, интенсивная работа с данными в науке (в науках о Земле, в биоинформатике), и пр. требуют использования данных, которые с трудом поддаются частичной структуризации. Твиты и посты в блогах являются включают слабоструктурированные текстовые отрывки, изображения и видео-ролики. Преобразование такого контента в структурированный формат для семантического анализа и поиска является трудной задачей, имеющей целью отображение структур и семантики данных в форму, «понимаемую» компьютером для извлечения информации из данных.

Данная работа[2] относится к области конструирования средств поддержки систем с интенсивным использованием данных. Методы унификации моделей данных образуют формальную базу работы [6] [3] [4] [5] [2].

Целью работы является анализ подходов к созданию *среды интеграции неоднородных коллекций данных* различного вида (структурированных, слабоструктурированных и неструктурированных). Такая среда должна поддерживать как виртуальную, так и материализованную интеграцию коллекций данных, представленных как в традиционных, так и нетрадиционных моделях данных.

Необходимость поддержки двух различных видов интеграции объясняется тем, что как виртуальный, так и материализованный подходы интеграции имеют свои достоинства и недостатки.

Виртуальная интеграция осуществляется с использованием технологии предметных посредников [8], образующих промежуточный слой между пользователем (приложением) и неоднородными информационными ресурсами. При этом данные из ресурсов не материализуются в посреднике. К достоинствам виртуальной интеграции относится то, что разверты-

[2] Статья основана на трех работах авторов, принятых для представления и публикации в трудах конференции RCDL 2014 [9][10][11].

вание и поддержка системы виртуальной интеграции значительно дешевле развертывания и поддержки системы материализованной интеграции (хранилища данных) исходя как из временных, так и из материальных затрат. Однако, данные в интегрируемых ресурсах не должны быть слишком большими, либо запросы к ресурсам должны обладать достаточной степенью селективности для того, чтобы объем данных, передаваемых из ресурса, не был слишком велик.

При материализованной интеграции предполагается создание хранилища данных (warehouse), в которое загружаются коллекции данных, подлежащие интеграции. В процессе загрузки происходит преобразование данных из схемы коллекции в общую схему хранилища. При необходимости, хранилища могут масштабироваться на большие объемы данных (хотя это требует соответствующих материальных затрат). Хранилища предоставляют удобную и эффективную платформу преобразования и интеграции данных, а также решения сложных аналитических задач над данными.

Материализованную интеграцию предлагается реализовывать с использованием свободно распространяемой платформы распределенного хранения и обработки данных Hadoop [12]; а также системы организации реляционных хранилищ данных над Hadoop (WR-Hadoop для краткости), в качестве которой могут использоваться, например, платформы Big SQL [13] или Hive [14].

Традиционно процесс интеграции данных можно представить состоящим из следующих этапов: сопоставление схем (schema matching), интеграция схем (schema integration), трансформация данных (data transformation), разрешение сущностей (entity resolution) [15] [16][17], слияние данных (data fusion) [18]. В данной статье основное внимание уделяется разрешению сущностей и слиянию данных при интеграции массивных данных в среде Hadoop.

Основной целью статьи является изложение основных принципов построения среды интеграции неоднородных коллекций данных и подходы к ее реализации. Создание программных средств реализации рассматриваемой архитектуры является предметом дальнейшей работы.

Статья организована следующим образом.

В разделе 2 рассматриваются основные черты среды интеграции неоднородных коллекций.

В разделе 3 иллюстрируются преобразования коллекций, представленных в нетрадиционных моделях данных, в их интегрированное представление в модели данных сопряжения Hadoop – среда посредников.

В разделе 4 дан краткий обзор традиционных методов извлечения информации из текстов, разрешения сущностей и методов слияния данных. Рассмотрены вопросы адаптации стандартных методов разрешения сущностей при интеграции данных в среде Hadoop. Проиллюстрированы способы программирования методов извлечения информации из текстов и методов слияния данных.

В разделе 5 рассматривается пример задачи интеграции неоднородных коллекций данных в предлагаемой среде.

2. Основные принципы функционирования среды интеграции неоднородных коллекций данных

Устройство среды интеграции неоднородных коллекций данных (рис. 1) основано на существующей архитектуре предметных посредников [8].

Основные черты архитектуры посредников выглядят следующим образом:

- концептуальная схема данных, предлагаемая пользователю системой интеграции (посредником), формируется независимо от интегрируемых ресурсов;
- ресурсы, релевантные концептуальной схеме, регистрируются в посреднике: их схемы связываются с концептуальной схемой при помощи представлений (взглядов). Определение взглядов осуществляется полуавтоматически и требует привлечения экспертов;
- пользователь задает программы (запросы) над концептуальной схемой. Эти запросы переписываются в посреднике с использованием взглядов в частичные подзапросы в терминах конкретных ресурсов. Переписывание осуществляется в языке правил канонической модели посредников;
- взаимодействие ресурсов и посредника реализуется при помощи адаптеров, располагающихся между посредником и ресурсами. Адаптеры преобразуют запросы из языка правил канонической модели в язык ресурса, а также преобразуют результат исполнения запроса из формата ресурса в формат посредника.

Таким образом, для виртуальной интеграции ресурсов, основанных на традиционных и нетрадиционных моделях, необходимо построение адаптеров соответствующих моделей данных. В области построения адаптеров для среды предметных посредников накоплен большой опыт, разработаны общая архитектура адаптеров и методы конструирования адаптеров [19], реализованы адаптеры реляционной и объектно-реляционной моделей, адаптер веб-сервисов, XML-адаптер и другие. Формальной базой для построения адаптеров служат

отображения моделей данных ресурсов в каноническую модель [20], разработанные в результате унификации моделей ресурсов.

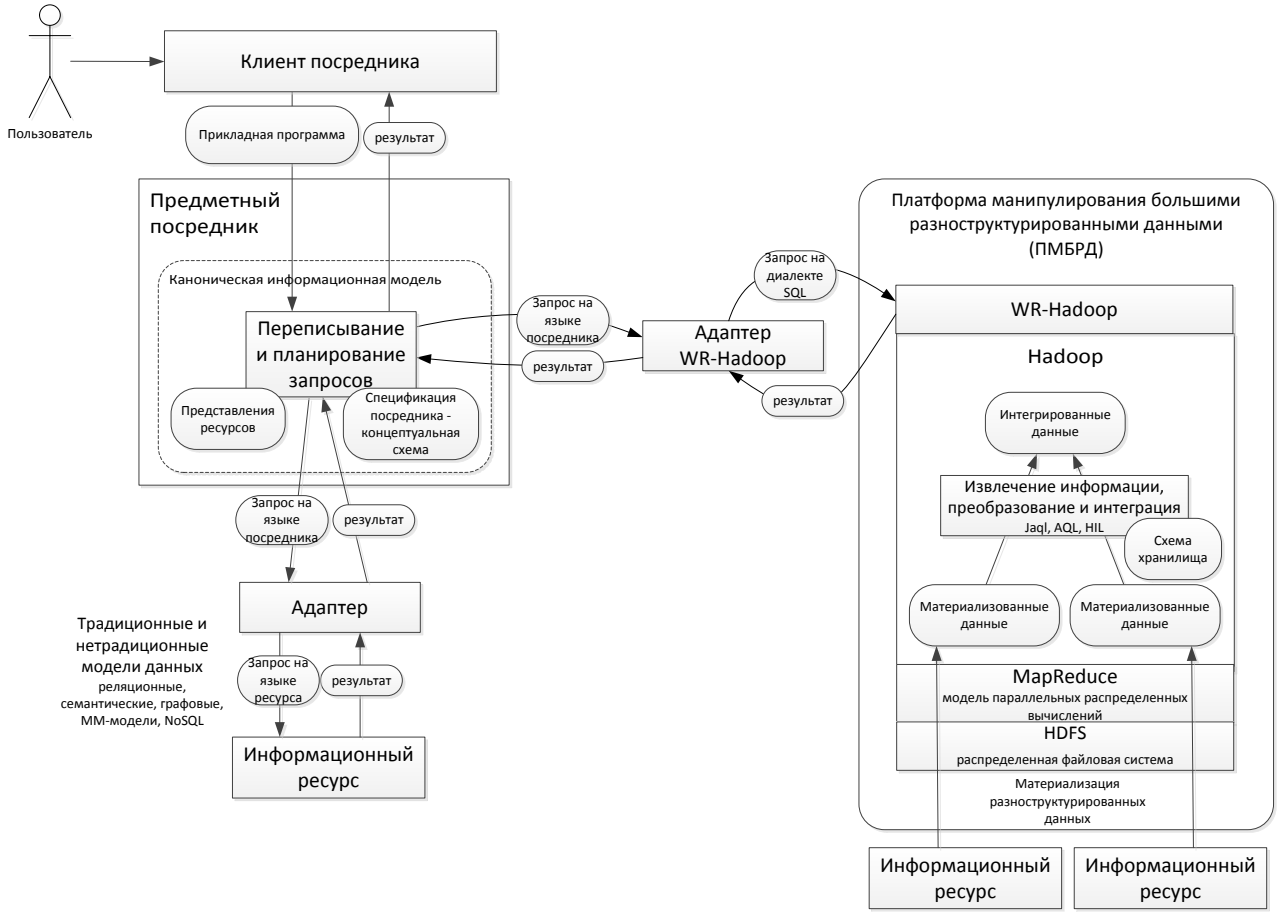

Рис. 1. Архитектура среды интеграции неоднородных коллекций данных

2.1 Платформа для материализованной интеграции ресурсов

Для материализованной интеграции ресурсов необходима масштабируемая платформа манипулирования большими разноструктурированными данными (ПМБРД). В качестве такой платформы в данной работе выбрана связка системы Hadoop и системы организации реляционных хранилищ данных над Hadoop – WR-Hadoop (в качестве которой могут использоваться, например, системы Big SQL [13] или Hive [14]).

Платформа Apache Hadoop [12] была впервые представлена в 2005 г. в составе проекта Apache Software Foundation и включает набор программных средств распределенного хранения и обработки больших объемов данных. Платформа Hadoop предназначена для развертывания на вычислительных кластерах. В состав кластера входят узлы, хранящие фрагменты файлов. Масштабируемость платформы на значительные объемы данных достигается за счет параллельной обработки фрагментов на узлах с использованием программной модели параллельных распределенных вычислений MapReduce [21].

Платформы Hive [14] и Big SQL [13] представляют собой решения для организации реляционных хранилищ данных, разработанные на основе среды Hadoop. В них реализованы известные структуры реляционных баз данных – таблицы, столбцы, разделы. Платформы поддерживают реляционные языки манипулирования данными (HiveQL и Big SQL соответственно) для работы в неструктурированной среде Hadoop: моделью данных Hive является стандарт SQL92 с некоторыми дополнениями, моделью данных Big SQL – практически полный стандарт SQL 2011. Фактически, системы проецируют реляционную структуру на данные, хранящиеся в Hadoop и предоставляют возможность исполнения SQL-подобных запросов на больших наборах данных путем компиляции их в программы MapReduce, исполняемые в среде Hadoop.

Система Hive является свободно распространяемым решением, а Big SQL – проприетарным, распространяемым в составе продукта IBM InfoSphere BigInsights [22].

Следует отметить, что системы Hive и Big SQL не в полной мере реализуют функции хранилищ данных (warehouses). В частности, напрямую не поддерживается процесс извлечения-преобразования-загрузки (ETL). Для реализации недостающих функций в рассматриваемой среде интеграции предлагается использование декларативно-императивных языков высокого уровня над Hadoop (раздел 2.2).

Таким образом, в предлагаемой среде обеспечивается возможность распределенного хранения, преобразования и интеграции больших разноструктурированных данных (при помощи Hadoop), а также унифицированный взгляд на материализованные данные через реляционную модель (при помощи Hive или Big SQL).

В среде интеграции ПМБРД рассматривается как еще один вид ресурсов, подлежащий виртуальной интеграции. Интеграция становится двухслойной: материализованная интеграция осуществляется внутри ПМБРД, виртуальная – на уровне предметных посредников.

Для включения ПМБРД Hadoop/WR-Hadoop в среду предметных посредников необходимо разработать адаптер для WR-Hadoop (Hive или Big SQL) в соответствии с подходом, изложенным в работе [19]. При этом модель данных WR-Hadoop выступает в роли модели сопряжения ПМБРД со средой предметных посредников. Модель данных WR-Hadoop должна быть унифицирована (отображена в каноническую модель посредников). Концептуально унификация модели данных WR-Hadoop опирается на проведенную при конструировании реляционного адаптера [19] унификацию реляционной модели. В конструкцию реляционного адаптера будут внесены два важных усовершенствования:

- поддержка объектных таблиц;
- поддержка сложных (complex) типов данных, таких, как массивы (ARRAY), структуры (STRUCT) и отображения (MAP).

2.2. Языки и инструменты для материализованной интеграции

Материализация в ПМБРД осуществляется путем помещения в Hadoop-кластер файлов, экспортированных из информационных ресурсов. Файлы могут быть экспортированы в различных открытых форматах: JSON, XML, CSV, в виде текстовых файлов, а также в бинарном формате JSON (JSON binary). Материализации, как и виртуальной интеграции, могут подлежать данные, представленные в различных традиционных и нетрадиционных моделях. Решение о том, какой способ интеграции следует применить для конкретного ресурса, следует принимать исходя из целей системы интеграции (например, характерных запросов пользователя), эффективности, стоимости и т.д.

Извлечение информации из текстовых данных предполагается реализовать в рассматриваемой среде с использованием языка Annotation Query Language (AQL) [23]. AQL – это декларативный язык для разработки экстракторов текстовой аналитики в среде IBM InfoSphere BigInsights. Экстрактор – это программа, написанная на языке AQL, которая извлекает структурированную информацию из неструктурированных и слабоструктурированных текстовых документов. Синтаксис AQL похож на синтаксис языка SQL, а модель данных языка подобна реляционной модели. Каждый AQL-экстрактор состоит из коллекции представлений (views), каждое представление определяет отношение. При определении представлений широко используется язык регулярных выражений.

Преобразование данных к реляционному виду для последующей интеграции производится при помощи программ на языке Jaql.

Jaql [24] представляет собой язык запросов и сценариев, разработанный IBM и использующий формат обмена данными JavaScript Object Notation (JSON [25]). Jaql поддерживает произвольную глубину вложенности структур данных, является в высокой степени функционально-ориентированным, чрезвычайно гибким, и хорошо применимым для работы со слабоструктурированными данными. Язык ориентирован на прозрачное применение модели программирования MapReduce: декларативно-императивные запросы Jaql переписываются в последовательность программ MapReduce, исполняемых в среде Hadoop. Язык Jaql поставляется в составе InfoSphere BigInsights [22] – программной платформы обработки больших данных, основанной на Hadoop.

Итак, преобразованные к реляционному виду данные сохраняются в формате JSON. Преобразование коллекций, представленных в нетрадиционных моделях данных, в их инте-

грированное представление в реляционной модели данных иллюстрируется в разделе 3. Сложные потоки обработки данных (очистки, устранения дублирования, слияния) и их интеграции реализуются с использованием комбинации языков Jaql и HIL.

Декларативный язык HIL (High-level Integration Language) [26] разработан IBM для программирования сложных потоков обработки данных (ETL), агрегирующих факты из больших коллекций разноструктурированной информации в целевые коллекции унифицированных сущностей. Программы на HIL компилируется в Jaql, что позволяет использовать HIL для преобразования и интеграции данных в Hadoop.

Краткий обзор методов разрешения сущностей и методов слияния данных, вопросы адаптации таких методов к их применению при интеграции больших данных в Hadoop и способы их программирования рассматриваются в разделе 4.

3. Преобразование коллекций данных нетрадиционных моделей в интегрированное представление

Рассмотрим пример коллекции данных, представленной в модели данных графовой СУБД Neo4j [27]. Небольшой подграф базы данных о фильмах, включающий основные виды вершин, ребер и атрибутов, изображен на рис. 2.

Вершины графа соответствуют фильмам и людям, ребра графа соответствуют участию людей в создании фильма как актеров (CAST) или режиссера (DIRECTS). Люди характеризуются именем (name), фильмы – названием (title) и годом создания (year), роли актеров – именем персонажа (character). Вершины и ребра графа обладают уникальными идентификаторами.

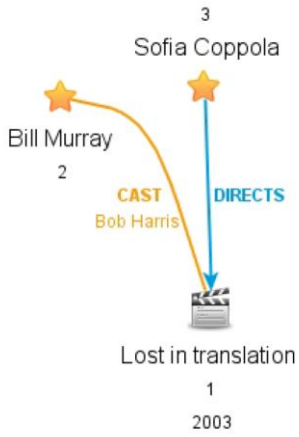

Рисунок 2. Пример графа в модели данных Neo4j

Графовая БД системы Neo4j может быть сериализована в формате JSON. Представление для вышеприведенного графа в JSON выглядит следующим образом:

```
[{ "id": 1, "type": "node", "labels": ["MOVIE"],
"properties": {"title": "Lost in translation", "year": 2003} },
{ "id": 2, "type": "node", "labels": ["PEOPLE"],
"properties": {"name": "Bill Murray"} },
{ "id": 3, "type": "node", "labels": ["PEOPLE"],
"properties": {"name": "Sofia Coppola"} },
{ "id": 4, "type": "relationship",
"start_node": 1, "end_node": 2,
"relationship_type": "CAST",
"properties": {"character": "Bob Harris"} },
{ "id": 5, "type": "relationship",
"start_node": 3, "end_node": 1,
"relationship_type": "DIRECTS" }]
```

Здесь start_node и end_node обозначают исходящую и входящую вершины ребра соответственно; labels обозначает множество меток, присвоенных вершине.

Методологически преобразованию графовой коллекции в интегрированное представление должна предшествовать унификация графовой модели (построение отображения графовой модели в модель данных WR-Hadoop, т.е. реляционную, с сохранением информации и семантики операций). Подобное преобразование проводится согласно унификации графовой модели в канонической модели посредников [4].

Реляционное представление данной графовой БД может выглядеть следующим образом. Схема реляционной БД состоит из двух отношений, одно из которых соответствует вершинам графа (Nodes, таблица 1), другое – ребрам (Relationships, таблица 2).

Таблица 1. Отношение *Nodes*

id	labels	title	year	name
1	[“MOVIE”]	“Lost in translation”	2003	
2	[“PEOPLE”]			“Bill Murray”
3	[“PEOPLE”]			“Sofia Coppola”

Таблица 2. Отношение Relationships

id	start_node	end_node	relationship_type	character
4	1	2	“CAST”	“Bob Harris”
5	3	1	“DIRECTS”	

Преобразование графовой БД в реляционное представление может быть осуществлено при помощи следующих двух функций, определенных на языке Jaql:

```
createNodesRelation = fn(movie_graph_db)(
   movie_graph_db->filter $.type == "node"->
          transform { id: $.id, labels: $.labels,
            title: $.properties.title, year: $.properties.year,
            name: $.properties.name }  );

createRelationshipRelation = fn(movie_graph_db)(
   movie_graph_db->filter $.type == "relationship"->
          transform { id: $.id,
            start_node: $.start_node, end_node: $.end_node,
            relationship_type: $.relationship_type,
            character: $.properties.character }  );
```

Функции принимают на вход представление графовой БД в формате JSON. Первая из функций возвращает отношение Nodes в формате JSON, пригодное для загрузки в соответствующую реляционную таблицу, вторая – отношение Relationships. При определении функций используются выражения фильтрации массивов (filter) и преобразования элементов массивов (transform), связанные оператором организации потоков данных (->).

4. Методы извлечения, разрешения сущностей и слияния данных при интеграции больших данных в Hadoop

4.1. Методы извлечения информации из текстов

Извлечение информации из текстов можно разделить на следующие этапы:

- распознавание сущностей (Entity Recognition);
- аннотация сущностей (Named Entity Annotation);
- устранение неоднозначности сущностей (Entity Disambiguation).

Распознавание сущностей является задачей извлечения имен сущностей из текста. Возможны два варианта распознавания сущностей: с помощью словарей и с помощью регулярных выражений. Словари содержат список имен и фраз определенного класса (например, список стран или компаний), которые будут извлекаться из текста. Однако, зачастую невозможно составить полный словарь списка фильмов, книг, имен людей. Но существуют классы сущностей, которые определяются конкретными правилами, например, адреса электронной почты, даты. В этом случае для извлечения сущностей из текста можно использовать регулярные выражения.

Аннотация сущностей – это задача извлечения имен сущностей из текста и их аннотация классом из заданного набора классов (например, классы Person, Organization, Location). Существует два подхода к аннотированию сущностей [28]: на основе правил и на основе статистических моделей.

Правила аннотируют фрагмент текста некоторым классом, если он удовлетворяет некоторому образцу (pattern) [29]. Образец представляет собой последовательность свойств (features), каждое из которых может быть регулярным выражением, словарем или другим образцом. Обычно правила задаются вручную, но могут и автоматически выявляться из текста.

Аннотация сущностей на основе статистических моделей [30] основана на разделении исходного текста на вектор слов (токенов) X и аннотировании каждого из этих слов классом из заданного вектора классов Y. Аннотирование происходит на основе вычисления вектора классов Y, максимизирующего функцию $\sum_i \sum_j w_j f_j(X, i, y_i)$, где свойство f_j - это функция, отображающая вектор слов X, позицию i в X и класс y_i в значение типа *real*, а w_j – это вес соответствующего свойства.

Многие имена имеют неоднозначное толкование, например, Форд – марка машины или имя человека. Примерами устранения неоднозначности сущностей могут служить следующие методы:

- разрешение неоднозначностей по контексту (context disambiguation) осуществляется на основе сравнения контекста имени в тексте (контекст формируется как множество всех имен, встречающихся в одном предложении с рассматриваемым именем) и контекста имени в базе знаний (контекст определяется как множество всех имен, связанных с рассматриваемым именем, в базе знаний);
- разрешение неоднозначностей по важности (prior disambiguation) осуществляется по наиболее часто встречающемуся значению этого имени в заданной коллекции документов.
- разрешение неоднозначностей по критерию согласованности (coherence disambiguation). Критерий согласованности утверждает, что имена, встречаемые в одном документе, должны быть связаны в базе знаний.

4.2. Пример программирования методов извлечения информации из текстов

В качестве примера задачи извлечения информации из текстов рассмотрим определение тональной окраски отзывов об организациях в текстах сообщений социальных сетей. Задача реализуется в виде программы на языке AQL.

Входными данными программы являются тексты, оформленные в формате CSV в кодировке UTF-8 в следующем виде: <идентификатор сообщения, текст сообщения в нормализованном виде>. Также на входе программа получает словари: словарь позитивных слов, словарь негативных слов и словарь организаций. Словари представляются в текстовом виде, где каждая строка содержит соответствующий термин.

Пример словаря позитивных слов:

```
АВТОРИТЕТНЫЙ
БЕЗОПАСНЫЙ
БЛАГОПОЛУЧИЕ
БЛАГОПОЛУЧНЫЙ
```

Извлечение организаций осуществляется на основе соответствующего словаря. На языке AQL это выглядит следующим образом:

```
create view Organizations as
extract dictionary 'OrganizationDict'
   on D.text as name
from Document D;
```

Исходя из того, что сообщения в социальных сетях имеют небольшую длину, для данного примера используется простая формула определения коэффициента тональной окраски извлекаемых сущностей (организаций). Коэффициент тональной окраски равен разнице в количестве позитивных и негативных слов в рамках одного и того же сообщения.

Сначала с помощью регулярных выражений исходный документ разбивается на конкретные сообщения:

```
create view Lines as
extract  split using B.boundary retain right split point on B.text as text
   from (
          extract D.text as text, regex /(\n)/ on D.text as boundary
          from Document D );


create view Messages as
extract regex /\"(\w+)\",\"(.*)\"/ on L.text
   return group 1 as name and  group 2 as text
from Lines L;
```

Затем выявляются сообщения, где встречаются организации и позитивные или негативные слова:

```
create view MessagesOrganizationPositive as
select m.name, o.name as organization_name, pw.word as word, 1 as cnt
from Messages m, Organizations o, PositiveWords pw
where Contains(m.text, o.name) and Contains(m.text, pw.word);
```

```
create view MessagesOrganizationNegative as
select m.name, o.name as organization_name, nw.word as word, -1 as cnt
from Messages m, Organizations o, NegativeWords nw
where Contains(m.text, o.name) and Contains(m.text, nw.word);
```

Для каждой найденной организации подсчитывается коэффициент тональной окраски этой организации:

```
create view MessagesOrganizationSentiment as
(select mop.name, mop.organization_name, mop.cnt
from MessagesOrganizationPositive mop)
union all
(select mon.name, mon.organization_name, mon.cnt
from MessagesOrganizationNegative mon);

create view OrganizationSentiment as
select GetText(mos.name) as message_id,
   GetText(mos.organization_name) as organization_name,
   Sum(mos.cnt) as sentiment
from MessagesOrganizationSentiment mos
group by GetText(mos.name), GetText(mos.organization_name);
```

Результатом выполнения компонента является таблица (таблица 1), содержащая идентификатор сообщения, извлеченная из этого сообщения организация, и коэффициент тональной окраски этой организации.

Таблица 3. Тональность упоминания организаций

message_id	organization_name	sentiment
33650_553	ОКСФОРДСКИЙ УНИВЕРСИТЕТ	0
36147_2826	СОЮЗ	1
36147_2936	МТС	-1
43079_4740	СОЮЗ	1
61454_2016	СОЮЗ	-15

4.3. Методы разрешения сущностей

В общем случае под термином разрешения сущностей [15][16][31][17] понимается извлечение информации об одной и той же сущности реального мира из разнообразных структурированных коллекций данных, приведение извлеченных данных к унифицированному представлению. При этом применяются методы извлечения, сопоставления (matching), группирования, связывания (linking), устранения дублирования (deduplication) различных представлений информации.

В общем случае процесс разрешения сущностей включает следующие этапы [31]:

- подготовку данных (Data preparation);
- выбор методов сопоставления данных (Match Feature);
- определение методов разрешения пар сущностей (Pairwise ER);
- определение ограничений (ER Constraints);
- реализацию алгоритма.

Важным этапом для успешного разрешения сущностей является подготовка данных, которая включает нормализацию схем и нормализацию данных. Нормализация схем включает, например, сопоставление атрибутов схем («контактный телефон» и «мобильный телефон»), слияние атрибутов («полный адрес» получается из атрибутов «город», «индекс», «улица»). Нормализация данных может включать приведение к строчному или заглавному регистру; удаление разделителей; поиск и исправления опечаток; поиск сокращений и аббревиатур и замена их на полные стандартные формы; использование словарей для нормализации строк, и много другое.

Сопоставление сущностей может осуществляться разнообразными способами оценки сходства (similarity) сущностей [32]. Мера сходства может быть как булевой, так и вещественной. Рассматривается также сходство отношений. Меры, используемые для отношений, обычно основаны на сходстве множеств, и предполагают использование функций сходства, таких, как Common Neighbors, Jaccard's Coefficient, Adar Coefficient [33]. При сравнении пар сущностей они рассматриваются как вектора, для которых нужно вычислить их сходство. Традиционным подходом является подсчет сходства некоторым методом для каждого из атрибутов независимо и дальнейший подсчет взвешенной суммы, например:

```
0.5*1st-author-match-score + 0.2*venue-match-score + 0.3*paper-match-score
```

Для сопоставления пар сущностей применяют также специальные методы машинного обучения, которые позволяют автоматизировать процесс формулирования критериев для сопоставления сущностей: Decision trees [34], Support vector machines [35], Ensembles of classifiers [36], Conditional Random Fields (CRF) [37].

Недостатком этих подходов является: несбалансированность результирующих классифицированных множеств (так, в результате образуется значительно больше несхожих объектов, чем схожих), а также высокая вероятность того, что объект не будет причислен ни к какому классу (схожих, несхожих). Но оба эти недостатка могут решаться путем тонкой настройки алгоритмов.

Примеры правил, используемых для установления сходства сущностей:

- транзитивность: если M1 и M2 схожи, и M2 и M3 схожи, тогда и M1 и M2 схожи;
- эксклюзивность: если M1 и M2 схожи, тогда M3 не может быть схож с M2;

- функциональные зависимости: если M1 и M2 схожи, тогда M3 и M4 должны быть схожи.

Транзитивность часто используется для методов удаления дубликатов (Deduplication), а эксклюзивность используется в методах установления связей (Record Linkage).

Разрешение сущностей является быстро развиваемой областью. Исследуются новые меры сходства [32], ведутся работы по применению перспективных методов машинного обучения [38][39]. Развивается применение функциональных зависимостей при очистке данных (data cleaning) [40]. Также ведутся работы по методам, когда решения по сходству двух сущностей принимается на основе анализа совокупности сущностей, применения вероятностных логик сходства, латентной модели Дирихле [41].

4.4. Методы слияния данных

Под слиянием данных (Data Fusion [18][42][43]) понимается образование интегрированного представления информации об одной же сущности реального мира, полученной из различных источников данных. Задачами процесса слияния данных является: слияние записей о сущностях, разрешение возможных конфликтов, обнаружение и удаления ошибочных данных. Методы слияния данных, кратко рассмотренные в данном разделе, исследованы в Потсдамском университете в диссертации [42].

4.4.1. Типы конфликтов при слиянии данных и стратегии их разрешения

Различают два типа конфликтов: конфликты, вызванные неопределенными значениями и конфликты, вызванные противоречивыми значениями. Неопределенность означает, что в одном источнике данных содержатся неизвестные значения (null), а в другом известные. Проблема заключается в том, что семантика неопределенных значений (null) может сильно отличаться. Различают три варианта: неизвестные значения, несуществующие значения (например, атрибут «имя супруга» всегда будет null для неженатых), скрытые значения (такие данные, которые по каким-то причинам не позволено видеть). Противоречивость значений означает появление двух различных не нулевых (not null) значений.

Различают следующие виды подходов к разрешению конфликтов: игнорирование, избегание, разрешение.

Стратегия *игнорирования* конфликтов предполагает извлечение всей доступной информации. Например, для строк это может быть обычная конкатенация строк, а пользователь уже сам решает, какие данные верны.

Стратегия *избегания* конфликтов предполагает выбор данных на основе самих данных (по некоторому алгоритму) или на основе метаданных. Примером функции на основе данных может служить функция coalesce (выбор первого не нулевого значения), или функция выбора самого длинного значения. Примером функций на основе метаданных может выступать выбор в зависимости от самого источника (например, известно, что один из источников наиболее достоверный). Другим примером является функция выбирающая значение из того источника, в котором большее число значений было выбрано для других атрибутов.

Стратегии *разрешения* конфликтов учитывают все значения, и выбирают из них «достоверное». Примером подобной функции могут выступать всевозможные функции голосования, функции выбора случайного значения, функции среднего значения, функции наиболее часто встречающегося значения, и другие.

4.4.2. Пример функции разрешения конфликтов

Вводится функция *tuple subsumption* [42]. Говорят, что кортеж t1 поглощает другой кортеж t2 (поглощаемый кортеж), если у них:

- совпадают схемы;
- в t2 больше неизвестных (null) значений чем в t1;
- в t2 все известные значения совпадают со значения в t1.

Например, пусть дан кортеж t1 = (5, ‘text’, null, 7) и t2 (5, null, null, 7). Видно, что каждый атрибут в t2 либо совпадает с аналогичным атрибутом в t1, либо он null. В этом примере кортеж t1 поглощает кортеж t2.

4.4.3. Примеры операций слияния данных

Различают два основных подхода к слиянию данных. Эти подходы основаны на операции объединения (union based) или на операции соединения (join based).

Примером union based операции является *Minimum Union* [42]. Операция представляет собой выполнение операции outer union, а затем удаления из результата всех поглощаемых (subsumed [42]) кортежей. Пример операции представлен на рисунке 3.

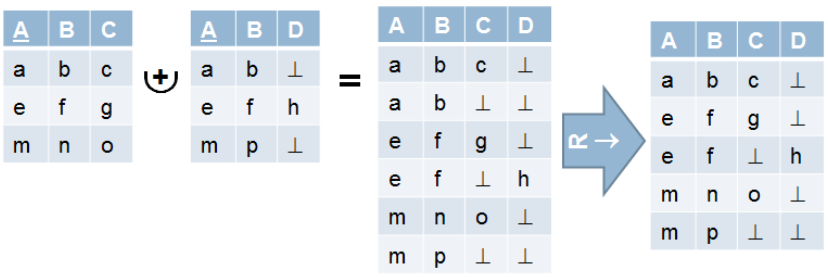

Рис. 3. Пример операции Minimum Union

Примером join based операции является *Full Disjunction* [44]. Операция представляет собой full outer join (стандартная реляционная операция), после чего применяется subsumption к результату. Пример представлен на рисунке 4.

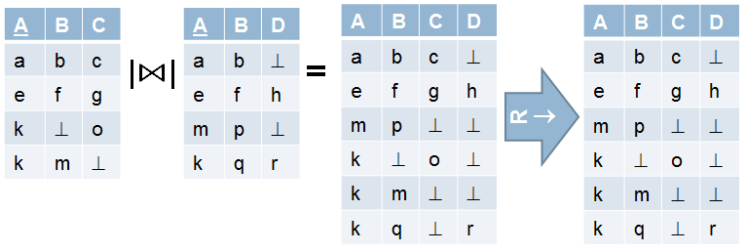

Рис. 4. Пример операции Full Disjunction

4.5. Разрешение сущностей для больших данных

Для манипулирования большими разно-структурированными данными служат Hadoop-инфраструктуры [12], предоставляющие масштабируемое хранилище и обеспечивающие высокую скорость обработки больших данных за счет распределенной их обработки. Таким образом, для применения методов нужна адаптация алгоритмов для их распределенного выполнения на различных узлах Hadoop кластера с использованием парадигмы Map-Reduce [21]. Другим вариантом является реализация алгоритма на одном из языков высокого уровня, таких как HiveQL [14] или Jaql [24].

В случае больших данных и распределенных инфраструктур традиционные подходы требуют доработок. Различают два основных метода разрешения сущностей над большими данными: разбиение данных на блоки (blocking [45]) и распределенный метод разрешения сущностей.

Суть разбиения на блоки заключается в следующем. Пусть у нас представлены 1000 компаний в 1000 городах. И нам нужно сравнить компании. Алгоритм полного попарного сравнения потребует 10^{12} сравнений. При этом, если предположить, что компании из разных городов не могут совпадать, то потребуется 10^9 сравнений. Ключевой проблемой данного подхода является выбор критерия, по которому разбивать данные.

Распределенный метод разрешения сущностей предполагает реализацию традиционных алгоритмов этого семейства в виде Map-Reduce приложения, что требует зачастую полного пересмотра исходного алгоритма. Другой вариант – реализация алгоритма разрешения сущностей на специализированных языках, чему будет посвящен следующий раздел. Третий вариант – использование специализированных инструментов, направленных на распределенное выполнение методов разрешения сущностей над Hadoop [46].

4.6. Пример программирования операции слияния данных на языке HIL

На данном этапе будем считать, что этап разрешения сущностей уже пройден и нам дана некоторая коллекция *Deduplicated,* в которой уже установлены соответствия одним из выше перечисленных способов (раздел 4.1). Например, пусть у нас есть две коллекции A (id, a, b, c) и B (id, a, b, d). Атрибуты a, b, c, d могут содержать NULL значения, атрибуты id совпадают. Ниже дан пример подобных данных для коллекции A в формате JSON:

```
[{"a":null,"b":null,"c":"wmqhxfgmac","id":919132322},
{"a":null,"b":null,"c":"wmqhxfgmac","id":919132322}]
```

Тогда коллекция разрешенных сущностей может быть получена следующим образом:

```
create link Deduplicated as
select
[gen: [id: a.id, a:a.a, b:a.b, c:a.c], dup: [id: b.id, a:b.a, b:b.b, d:b.d]]
from A a, B b
match using  rule1: a.id = b.id;
```

Рассмотрим теперь реализацию операции Minimum Union (раздел 4.4.3) на языке HIL.

Операция Minimum Union - это последовательное применение операций outer union и subsumption [42]. Outer Union фактически реализуется с помощью индекса FusionIndex. Использование индекса оправдано, т.к. существует несколько записей, описывающих одну сущность. Ключом является атрибут id. Ниже представлена реализация операции Outer Union.

```
insert into FusionIndex![id: f.gen.id]
select [a: f.gen.a, b: f.gen.b, c: f.gen.c] from Deduplicated f;

insert into FusionIndex![id: f.dup.id]
select [a: f.dup.a, b: f.dup.b, d: f.dup.d] from Deduplicated f;
```

Далее для реализации subsumption требуется удалить все ненужные кортежи. Это делается на языке Jaql. Для этого нужна функция, которая бы определяла, поглощается ли один кортеж другим:

```
is_subsumed = fn(i,j) ( (i != j) and
( isnull(j.a) or (i.a == j.a) ) and ( isnull(j.b) or (i.b == j.b) ) and
( isnull(j.c) or (i.c == j.c) ) and ( isnull(j.d) or (i.d == j.d) )  );
```

Функция *is_subsumed(i, j)* проверяет, поглощает ли один кортеж другой кортеж при помощи попарного сравнения атрибутов или проверки на null.

```
removeSubsumed = fn (a) ( b = a,
subs = for (i0 in b) [a->filter is_subsumed(i0,$)], s = subs -> expand,
```

```
a -> filter not $ in s);
```

Функция *removeSubsumed* удаляет все поглощенные записи из кортежа. Здесь реализован наивный алгоритм, который попарно для каждого кортежа находит все поглощенные им, и удаляет их.

```
minUnion = fn(id,a) ( {id:id, minunion : removeSubsumed(a)});
```

Функция *minUnion* нужна для построения результирующих кортежей при реализации Minimum Union.

Теперь операцию Minumum Union можно описать следующим образом на языке HIL:

```
insert into MinimumUnion
select minUnion(i.dup.id, FusionIndex![id : i.dup.id])
from Deduplicated i;
```

Для каждого id достаются все соответствующие записи и удаляются те, которые ими поглощаются.

5. Пример задачи интеграции неоднородных коллекций данных

Рассматриваемая задача состоит в определении отношения населения к экономическим и политическим вопросам в конкретном регионе. В задаче используются следующие информационные ресурсы:

- архивы региональных электронных СМИ;
- социальные сети (например, ВКонтакте);
- сервисы микроблогов (например, Twitter).

Ресурсы существенным образом отличаются по структуре, а также по характеру и лексике текстов. Например, сообщение из Twitter, представленное в формате JSON, содержит текст, идентификатор сообщения, язык сообщения, дату создания, информацию о пользователе (приведена лишь часть данных, составляющих сообщение):

```
{ “text”: “В первом квартале 2014 ввод жилья в Бобруйской области вырос на 30
процентов http://t.co/9cTJcnkucI”,
“id”: 441892291058622460,
“lang”: "ru",
“created_at”: "Fri Mar 07 11:05:06 +0000 2014",
“user”: { "name":"Петр Иванов", "screen_name":"32_minute",“id”: 2191013094}
}
```

Сообщение из сети ВКонтакте содержит идентификатор сообщения, идентификатор автора, идентификатор получателя, дату создания, текст, количество «лайков», количество переотправок (приведена лишь часть данных, составляющих сообщение; текст сообщения приведен не полностью):

```
{ "id": 254, "from_id":2785124, "to_id":6835,
"date":1387737719,
"text":"Бобруйская область - регион# в котором добывается больше половины все-
го леса в России. Однако на благосостоянии простых жителей Ухтомского района
это никак не сказывается. …",
"likes":{"count": 10},
"reposts":{"count": 5, "user_reposted": 5}
}
```

В рамках задачи анализу подлежат статьи из СМИ и сообщения из социальных сетей за определенный период времени (текущий год, квартал и т.д.) и опубликованные авторами, проживающими в определенном регионе. Такие статьи и сообщения извлекаются из соответствующих ресурсов, загружаются в Hadoop и пропускаются через средства текстовой аналитики, развернутые на каждом из узлов кластера. Анализ текста позволяет извлечь из текстов статей и сообщений различные упоминаемые объекты: персоны (и их должности), территориальные образования, организации и т.д. (раздел 4.1). Также анализ текста позволяет оценить тональность сообщения – выявить позитивное, нейтральное или негативное отношение автора текста к теме текста (раздел 4.2). Таким образом, тональность текста может быть связана с объектами, упоминаемыми в тексте.

Например, информация, извлеченная из приведенного выше сообщения из сети ВКонтакте, может выглядеть следующим образом:

```
{ "user_id": "6835", "message_id": 254, "source": "ВКонтакте",
"extracted_objects":{
   "persons": [{ "name": "Василий", "surname": "Петров",
                 "position": "губернатор"}],
   "territorial_entities": [
           {"name": "Бобруйская", "type": "область"},
           {"name": "Ухтомский", type: "район"}]  },
"sentiment": "negative",
"negative_keywords": ["барак", "отчаяние", "прогнивший"]
}
```

В тексте сообщения упоминаются губернатор *Василий Петров*, территориальные образования *Бобруйская область* и *Ухтомский район*. Текст носит отрицательную тональность.

Материализованные статьи и сообщения, обогащенные информацией, извлеченной из текстов, преобразуются к виду, удовлетворяющему единой схеме хранилища[3]. В хранилище помещаются также данные, извлеченные из профилей пользователей социальных сетей.

[3] Принципы построения схемы хранилища для задачи выходят за рамки данной статьи.

Определение отношения населения к экономическим и политическим вопросам может быть осуществлено путем различных запросов к схеме хранилища. Могут быть сделаны выводы, например, об отношении населения к конкретным лицам, организациям за определенные промежутки времени и на определенной территории путем подсчета позитивных, нейтральных и негативных сообщений и статей, упоминающих этих лиц или организации.

Отношение населения к экономическим и политическим вопросам может быть также ограничено некоторой темой, например, *«проблемы жилищно-коммунального хозяйства»*. В этом случае анализу подвергаются лишь те сообщения, в которых присутствуют ключевые слова (или их комбинации), относящиеся к теме, например, *{жилье, ЖКХ, отопление, электричество, тариф, барак, ветхий, аварийный}*.

Дополнительные аналитические возможности предоставляет виртуальная интеграция социальных графов (хранимых в графовой базе данных, например, Neo4j [27]) и хранилища сообщений и статей в одном предметном посреднике. Социальные графы (образуемые отношениями «друг» или «подписчик») загружаются в графовую базу данных, предоставляющую операции и алгоритмы анализа графов. Предметный посредник, интегрирующий социальные графы и хранилище статей и сообщений, позволяет формулировать и исполнять программы, определяющие влияние социальных сетей на формирование отношения населения к экономическим и политическим вопросам. Например, может быть отслежено распространение во времени позитивных, нейтральных и негативных сообщений, относящихся к некоторой теме (заданной ключевыми словами) в связных подграфах социальных сетей.

6. Заключение

В статье рассмотрены основные принципы организации среды интеграции больших неоднородных коллекций данных и подходы к ее реализации. Целью среды является сопряжение возможностей предметных посредников по интеграции разномодельных коллекций данных и возможностей по манипулированию разноструктурированными данными, предоставляемыми платформой Hadoop и ее надстройками. Масштабирование обработки коллекций данных при помощи технологии Hadoop существенно расширяет возможности технологии предметных посредников по решению задач над большими данными, представленными неоднородными информационными ресурсами. Реализация среды, ее практическое применение и сравнение с родственными подходами являются задачами дальнейшей работы.

Список литературы

1. The Forth Paradigm: Data-Intensive Scientific Discovery. Eds. Tony Hey, Stewart Tansley, and Kristin Tolle. Redmond: Microsoft Research, 2009. – URL: http://goo.gl/GqkDX1 (дата обращения: 13.08.2014).
2. Kalinichenko L.A., Stupnikov S.A. OWL as Yet Another Data Model to be Integrated. Advances in Databases and Information Systems: Proc. II of the 15th East-European Conference. - Vienna: Austrian Computer Society, 2011. - P. 178-189.
3. Скворцов Н. А. Отображение модели данных RDF в каноническую модель предметных посредников // Труды 15-й Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» RCDL'2013. – Ярославль: Ярославский государственный университет им. П. Г. Демидова, 2013. С. 202-209.
4. Ступников С. А. Отображение графовой модели данных в каноническую объектно-фреймовую информационную модель при создании систем интеграции неоднородных информационных ресурсов // Труды 15-й Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» RCDL'2013. – Ярославль: Ярославский государственный университет им. П. Г. Демидова, 2013. С. 193-202.
5. Ступников С. А. Унификация модели данных, основанной на многомерных массивах, при интеграции неоднородных информационных ресурсов // Труды 14-й Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» RCDL'2012. – Переславль-Залесский: Университет города Переславля, 2012. С. 67-77.
6. Скворцов Н. А. Отображение моделей данных NoSQL в объектные спецификации // Труды 14-й Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» RCDL'2012. – Переславль-Залесский: Университет города Переславля, 2012. С. 78-87.
7. Kalinichenko L.A., Stupnikov S.A., Martynov D.O. SYNTHESIS: a Language for Canonical Information Modeling and Mediator Definition for Problem Solving in Heterogeneous Information Resource Environments. Moscow: IPI RAN, 2007. - 171 p.
8. Kalinichenko L.A., Briukhov D.O., Martynov D.O., Skvortsov N.A., Stupnikov S.A. Mediation Framework for Enterprise Information System Infrastructures. Proc. of the 9th International Conference on Enterprise Information Systems ICEIS 2007. - Funchal, 2007. - Volume Databases and Information Systems Integration. - P. 246-251.

9. Ступников С. А., Вовченко А. Е. Комбинированная виртуально-материализованная среда интеграции больших неоднородных коллекций данных. Труды 16-й Всероссийской научной конференции «Электронные библиотеки: перспек-тивные методы и технологии, электронные коллекции» RCDL'2016. – Дубна: ОИЯИ, 2014.
10. Вовченко А.Е., Калиниченко Л.А., Ковалев Д.Ю. Программирование методов разрешения сущностей и слияния данных при реализации ETL в среде Hadoop. Труды 16-й Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» RCDL'2016. – Дубна: ОИЯИ, 2014.
11. Брюхов Д.О., Скворцов Н.А. Извлечение информации из больших коллекций русскоязычных текстовых документов в среде Hadoop. Труды 16-й Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» RCDL'2016. – Дубна: ОИЯИ, 2014.
12. White T. Hadoop: The Definitive Guide. Third Edition. O'Reilly Media. 2012.
13. Saracco C. M., Jain U. What's the big deal about Big SQL? Introducing relational DBMS users to IBM's SQL technology for Hadoop. IBM DeveloperWorks, 2013. – URL: http://www.ibm.com/developerworks/library/bd-bigsql/bd-bigsql-pdf.pdf (дата обращения: 13.08.2014).
14. Capriolo E., Wampler D., Rutherglen J. Programming Hive Data Warehouse and Query Language for Hadoop. O'Reilly Media, 2012.
15. Christen P. Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Data-Centric Systems and Applications, 2012.
16. Wenfei Fan, Geerts F. Foundations of Data Quality Management. Synthesis Lectures on Data Management. #29, 2012.
17. Naumann F., Herschel M. An Introduction to Duplicate Detection. Synthesis Lectures on Data Management. #3, 2010.
18. Bleiholder J., Naumann F. Data Fusion. ACM Computing Survey. 2009.
19. Вовченко А. Е. Рассредоточенная реализация приложений в среде предметных посредников. Диссертация на соискание ученой степени кандидата технических наук по специальности 05.13.11. На правах рукописи. Москва: ИПИ РАН, 2012. 216 с.
20. Ступников С. А., Скворцов Н. А., Будзко В. И., Захаров В. Н., Калиниченко Л. А. Методы унификации нетрадиционных моделей данных. Системы высокой доступности. Радиотехника, 2014. – Вып. 1. – С. 18-39.
21. Miner D. MapReduce Design Patterns: Building Effective Algorithms and Analytics for Hadoop and Other Systems. O'Reilly Media, 2012.

22. IBM InfoSphere BigInsights Information Center. 2014. – URL: http://pic.dhe.ibm.com/infocenter/bigins/v2r1/index.jsp (дата обращения: 13.08.2014).
23. Annotation Query Language. URL: http://goo.gl/wJ6X1d (дата обращения: 13.08.2014).
24. Beyer K. S., Ercegovac V., Gemulla R., Balmin A., Eltabakh M., Kanne C.-C., Ozcan F., Shekita E. J. Jaql: A Scripting Language for Large Scale Semistructured Data Analysis. VLDB 2011.
25. Introducing JSON. 2014. - http://www.json.org/ (дата обращения: 13.08.2014).
26. Hernández M., Koutrika G., Krishnamurthy R., Popa L., Wisnesky R. HIL: a high-level scripting language for entity integration. Proceedings of the 16th International Conference on Extending Database Technology EDBT 2013. P. 549-560.
27. The Neo4j Manual. 2014. - http://goo.gl/cHiOGF (дата обращения: 13.08.2014).
28. Sarawagi S. Information Extraction. Foundations and Trends in Databases. - 2008.- 1(3): 261-377.
29. Cunningham H., Maynard D., Bontcheva K., Tablan V. Gate: A framework and graphical development environment for robust NLP tools and applications. Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics. - 2002.
30. Getoor L., Taskar B. (Eds.). Introduction to Statistical Relational Learning. MIT Press, 2007.
31. Getoor L., Machanavajjhala A. Entity Resolution for Big Data. 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. Chicago: ACM SIGKDD, 2013.
32. String metric. URL: http://en.wikipedia.org/wiki/String_metric (дата обращения: 13.08.2014).
33. Adamic L. A., Adar E. Friends and neighbors on the Web. Social networks. – 2003. - 25 (3), 211-230.
34. Cochinwala M. et al. Efficient data reconciliation. Information Sciences. 2001.
35. Christen P. Automatic record linkage using seeded nearest neighbour and support vector machine classification. KDD. 2008.
36. Chen Z. et al. Exploiting context analysis for combining multiple entity resolution systems. SIGMOD. 2009.
37. Gupta R., Sarawagi S. Answering Table Augmentaton Queries from Unstructured Lists on the Web. PVLDB. 2009. - 2(1).
38. Herzog T. et al. Data Quality and Record Linkage Techniques. Springer, 2007.
39. Bellare K. et al. Active sampling for entity matching. KDD. 2012.
40. Wenfei Fan. Dependencies revisited for improving data quality. PODS. 2008.
41. Bhattacharya I., Getoor L. A Latent Dirichlet Model for Unsupervised Entity Resolution. SDM. 2007.

42. Bleiholder J. Data Fusion and Conflict Resolution in Integrated Information Systems. Dissertation. Hasso-Plattner-Institut, 2010.
43. Dong X. L., Naumann F. Data Fusion – Resolving data conflicts in Integration. VLDB. 2009.
44. Rajaraman A., Ullman J. D. Integrating information by outerjoins and full disjunctions. PODS. 1996.
45. Sarma A. D. et al, “An Automatic Blocking Mechanism for Large-Scale De-duplication Tasks”, CIKM 2012
46. Kolb L., Thor A., Rahm E. Dedoop: Efficient Deduplication with Hadoop. Proc. 38th Intl. Conference on Very Large Databases (VLDB). VLDB Endowment, 2012. - 5(12).

Реферат

Новая парадигма в науке и информационных технологиях, доминирующая в последнее время, основывается на исследовании данных (data exploration). В рамках этой парадигмы необходимы подходы, позволяющие справляться с разнообразием моделей данных (включая неструктурированные и слабоструктурированные данные), метаданных, семантики данных. Растущая популярность использования слабоструктурированных баз данных (в различных видах NoSQL) в совокупности с технологиями программирования Hadoop и MapReduce, обеспечивающими параллельную обработку огромных массивов слабоструктурированных данных, объясняется множеством фактических и потенциальных применений. Данная работа относится к области конструирования средств поддержки систем с интенсивным использованием данных. Целью работы является анализ подходов к созданию среды интеграции неоднородных коллекций данных различного вида. Такая среда должна поддерживать как виртуальную, так и материализованную интеграцию коллекций данных, представленных как в традиционных, так и в нетрадиционных моделях данных. Виртуальная интеграция осуществляется с использованием технологии предметных посредников, образующих промежуточный слой между пользователем (приложением) и неоднородными информационными ресурсами. Материализованную интеграцию предлагается реализовывать с использованием свободно распространяемой платформы распределенного хранения и обработки данных Hadoop; а также системы организации реляционных хранилищ данных над Hadoop, в качестве которой могут использоваться, например, платформы Big SQL или Hive. В работе рассматриваются основные черты среды интеграции неоднородных коллекций. Иллюстрируются преобразования коллекций, представленных в нетрадиционных моделях данных, в их интегрированное представление в модели данных сопряжения Hadoop – среда посредников. Дан краткий обзор традиционных методов извлечения информации из

текстов, разрешения сущностей и методов слияния данных. Рассмотрены вопросы адаптации стандартных методов разрешения сущностей при интеграции данных в среде Hadoop. Проиллюстрированы способы программирования методов извлечения информации из текстов и методов слияния данных. Рассматривается пример задачи интеграции неоднородных коллекций данных в предлагаемой среде.

Environment for Integration of Large Heterogeneous Data Collections

Abstract

An approach for the development of an environment for integration of heterogeneous data collections (structured, semi-structured and unstructured) is considered. The main idea of the approach is a combination of subject mediation technology, the Hadoop open source platform for distributed data storage and processing and a relational warehouse system over Hadoop. As a warehouse the systems like IBM Big SQL or Hive can be used. Issues of entity resolution and data fusion in the context of big data integration in the Hadoop are considered. Brief overview of meth-ods for information extraction from text is provided. Techniques for programming of the entity resolution and data fusion methods using HIL high-level integration language are illustrated. An example of a problem to be solved using the proposed environment for integration of heterogeneous data collections is provided.

Summary

New science and IT paradigm dominating during the last years is based on data exploration. Techniques for overcoming the diversity of data models (including semi-structured and unstructured data), metadata, and data semantics are required in the frame of this paradigm. Popularity of semi-structured NoSQL databases combined with Hadoop and MapReduce technologies aimed at parallel processing of large semi-structured data collections is steadily growing. Such recognition is explained by a multitude of actual and potential applications. This work concerns the area of development of data intensive systems. The aim of this work is analysis of approaches to the development of environment for integration of heterogeneous data collections. The environment should support both virtual and materialized integration of data collections represented in traditional and non-traditional data models. Virtual integration is provided by the subject mediation technology. The mediators form a layer between users (applications) and heterogeneous information resources. Materialized integration is proposed to be implemented using the Hadoop open source system for

distributed data storage and processing. The Hadoop should be combined with a data warehouse system over Hadoop (Hive or IBM Big SQL systems can be used for that).

Basic features of the environment for integration of large heterogeneous collections are presented in the paper. Transformation of collections represented using non-traditional data models into the integrated data model is illustrated. The integrated representation is based on the warehouse data model. Brief overview of methods for information extraction from text, entity resolution and data fusion is provided. Techniques for programming of the entity resolution and data fusion methods using HIL high-level integration language are illustrated. An example of a problem to be solved using the proposed environment for integration of heterogeneous data collections is provided.