Комбинированная виртуально-материализованная среда интеграции больших неоднородных коллекций данных

© С. А. Ступников ©А. Е. Вовченко

ИПИ РАН,
Москва

ssa@ipi.ac.ru alexey.vovchenko@gmail.com

Аннотация

В работе рассматривается архитектура комбинированной виртуально-материализованной среды интеграции неоднородных коллекций данных различного вида (структурированных, слабоструктурированных и неструктурированных). Необходимость поддержки двух различных видов интеграции объясняется тем, что как виртуальный, так и материализованный подходы к интеграции имеют свои достоинства и недостатки. Виртуальная интеграция осуществляется с использованием технологии предметных посредников. Материализованная интеграция реализуется 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 платформы Big SQL или Hive.

Работа выполнена при поддержке РФФИ (гранты 13-07-00579, 14-07-00548) и Президиума РАН (Программа фундаментальных исследований Президиума РАН № 16 «Фундаментальные проблемы системного программирования»).

1 Введение

Новая парадигма [24] 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 (системы SimpleDB, MongoDb, CouchDB), модели с колоночным хранением (системы HBase, HyperTable, Cassandra), модели «ключ-значение» (системы Voldemort, Riak, Redis, Scalaris). Развиваются онтологические и семантические модели, такие, как семейства RDF и OWL; графовые модели (системы Neo4j, Dex, GraphDB, HyperGraphDB, Trinity, Pregel); 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 [10], RDF [3]), графовых [5], ММ-моделей [4], NoSQL-моделей [2]. В качестве канонической модели рассматривался язык СИНТЕЗ [11], 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

Труды 16-й Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» — RCDL-2014, Дубна, Россия, 13–16 октября 2014 г.

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 относится к области конструирования средств поддержки систем с интенсивным использованием данных. Методы унификации моделей данных образуют формальную базу работы.

Целью работы является разработка и реализация комбинированной виртуально-материализованной архитектуры 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 [12], 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 развертывание и поддержка системы виртуальной интеграции значительно дешевле развертывания и поддержки системы материализованной интеграции (хранилища данных) исходя как из временных, так и из материальных затрат. Обычно системы виртуальной интеграции используются для интеграции ресурсов, данные из которых трудно преобразовывать и (или) владельцами которых являются разные лица. 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.

Материализованная интеграция реализуется в комбинированной архитектуре с использованием свободно распространяемой платформы распределенного хранения и обработки данных Hadoop [27]; а также системы организации реляционных хранилищ данных над Hadoop (WR-Hadoop для краткости), в качестве которой могут использоваться, например, платформы Big SQL [22] или Hive [9].

В данной статье рассматриваются основные принципы построения комбинированной виртуально-материализованной архитектуры и подходы к ее реализации. Создание программных средств реализации рассматриваемой архитектуры является предметом дальнейшей работы. Ближайшей задачей является выбор конкретной системы WR-Hadoop, одним из вариантов которой является поддержка обеих систем Big SQL и Hive.

Статья организована следующим образом. В разделе 2 рассматриваются основные черты комбинированной архитектуры. 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 рассматривается пример задачи интеграции неоднородных коллекций данных в комбинированной среде.

2 Архитектура комбинированной виртуально-материализованной среда интеграции коллекций данных

Архитектура комбинированной среды интеграции неоднородных коллекций данных (рис. 1) основана на существующей архитектуре предметных посредников [12].

Основные черты архитектуры 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;

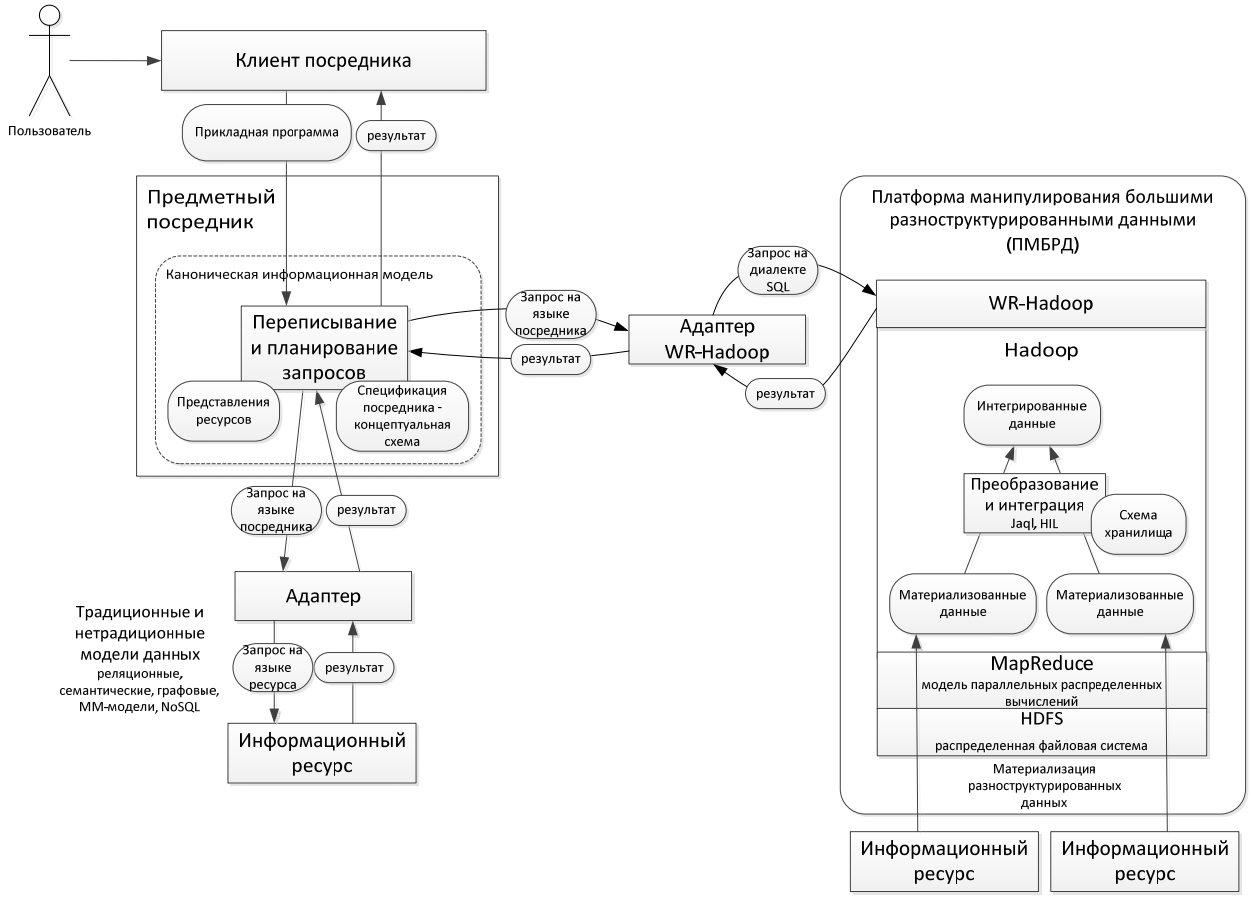

Рисунок 1. Архитектура комбинированной среды интеграции неоднородных коллекций данных

- 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 [1], 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 [6], разработанные в результате унификации моделей ресурсов.

2.1 Платформа для материализованной интеграции ресурсов в комбинированной архитектуре

Для материализованной интеграции ресурсов в комбинированной архитектуре 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 [22] или Hive [9]).

Платформа *Apache Hadoop* [7][27] была впервые представлена в 2005 г. в составе проекта Apache Software Foundation и представляет собой набор программных средств распределенного хранения и обработки больших объемов данных. Платформа Hadoop предназначена для развертывания на вычислительных кластерах и основана на распределенной файловой системе HDFS (Hadoop Distributed File System). В состав кластера входят узлы, хранящие фрагменты файлов. Теоретически могут быть сотни и тысячи таких узлов, основанных на недорогих вычислительных платформах (commodity hardware). Для обеспечения высокой надежности поддерживается избыточность путем

создания копий фрагментов между узлами. С точки зрения пользователя такая структура выглядит как обычная древовидная файловая система.

Масштабируемость платформы на значительные объемы данных достигается за счет параллельной обработки фрагментов на узлах с использованием программной модели параллельных распределенных вычислений MapReduce [21]. Модель MapReduce позволяет разработчикам приложений абстрагироваться от сложностей работы распределенных программ, связанных с распределением данных, планированием нагрузки и обеспечением отказоустойчивости.

Платформы *Hive* [9] и *Big SQL* [22] 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 [16].

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 комбинированной архитектуре предлагается использование декларативно-императивных языков высокого уровня над Hadoop (раздел 2.2).

Таким образом, в комбинированной архитектуре 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).

Потенциально, в качестве ПМБРД может быть выбрана и другая платформа: например, вместо Hadoop могут быть использованы такие системы, как Apache Spark [8, 28], GraphLab [20, 25], Disco [14] и др. Однако, Hadoop в настоящее время является наиболее распространенной и универсальной системой, поддерживающей модель вычислений MapReduce.

В комбинированной архитектуре ПМБРД рассматривается как еще один вид ресурсов, подлежащий виртуальной интеграции. Интеграция становится двухслойной: материализованная интеграция осуществляется внутри ПМБРД, виртуальная – на уровне предметных посредников. Виртуальной интеграции при этом могут подлежать ресурсы

- данные из которых сложно или невозможно материализовать по разным причинам и (или)
- модель данных включает специфические операции и алгоритмы, адаптация которых в реляционной модели сложна и (или) неэффективна. К таким моделям относятся, например, графовые, ММ-модели и т.д.

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 [1]. 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 [1] 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 в комбинированной архитектуре

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.

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

Jaql представляет собой язык запросов и сценариев, разработанный IBM и использующий формат обмена данными JavaScript Object Notation (JSON [19]). 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.

Основными структурами данных, подлежащими манипулированию при помощи Jaql, являются объекты (коллекции пар <имя, значение>) и массивы (упорядоченные списки значений). Поддерживается множество встроенных функций над массивами и объектами. Язык предоставляет широкий набор операций преобразования данных (фильтрация, группировка, сортировка, соединение, объединение и т.д.).

Язык Jaql поставляется в составе InfoSphere BigInsights [16] – программной платформы обработки больших данных, основанной на Hadoop.

Итак, преобразованные к реляционному виду данные сохраняются в формате JSON. Преобразования коллекций, представленных в нетрадиционных моделях данных, в их интегрированное представление в реляционной модели данных иллюстрируются в разделе 5. Сложные потоки обработки данных (очистки, устранения дублирования, слияния) и их интеграции реализуются с использованием комбинации языков Jaql и *HIL*[1].

Декларативный язык *HIL* (High-level Integration Language) разработан IBM для программирования сложных потоков обработки данных (ETL), агрегирующих факты из больших коллекций разноструктурированной информации в целевые коллекции унифицированных сущностей.

Язык позволяет реализовать методы извлечения, сопоставления и группирования, разбора, связывания, устранения дублирования (deduplication) различных разноструктурированных представлений информации об одних и тех же сущностях реального мира (entity resolution). HIL также позволяет реализовать методы и операции слияния (интеграции) данных об одних и тех же сущностях реального мира и их связей, представленных в разных коллекциях, образованных в процессе разрешения сущностей (включая реализации стратегий и устранения конфликтующих данных, операции поглощения и слияния данных). Программы на HIL компилируется в Jaql, что позволяет использовать HIL для преобразования и интеграции данных в Hadoop.

Следует отметить, что методы материализованной интеграции, организации ETL-процессов, очистки данных, предлагаются в составах программных решений ведущих разработчиков баз данных. К таким решениям относится, например, IBM InfoSphere Information Server [17]. Этот программный продукт потенциально может быть использован в комбинированной архитектуре для материализованной интеграции. При этом в качестве хранилища может быть использована платформа IBM InfoSphere Warehouse [18] (являющаяся частью СУБД DB2). Однако, Information Server предлагает лишь ограниченное количество относительно простых методов выделения, отображения и слияния сущностей. Недостаточность таких средств была осознана компанией IBM, в результате чего и появился язык HIL.

Рассматриваемая в данной статье комбинированная среда интеграции нацелена, в частности, на исследование методов интеграции больших разноструктурированных данных. Этим и мотивирован выбор в качестве ПМБРД платформы Hadoop/WR-Hadoop и комбинации языков Jaql и HIL в качестве инструментов материализованной интеграции.

3 Преобразование коллекций данных нетрадиционных моделей в интегрированное представление

Рассмотрим пример коллекции данных, представленной в модели данных графовой СУБД Neo4j [26]. 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.

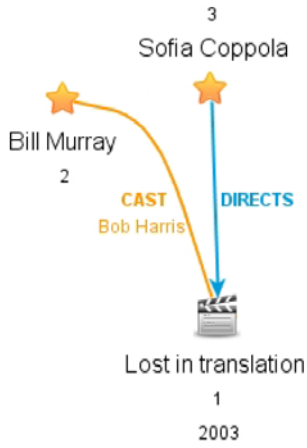

Рисунок 2. Пример графа в модели данных Neo4j

[1] Вопросы выделения сущностей и интеграции разноструктурированной информации при помощи программ на языке HIL выходят за рамки данной статьи.

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.

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

В качестве еще одного примера коллекции данных рассмотрим базу знаний *DBpedia* (http://dbpedia.org/). Коллекция содержит структурированную информацию, извлеченную из свободной энциклопедии Википедия. Данные в коллекции представлены в модели RDF. В качестве примеров данных рассмотрим RDF-спецификации города *Кембридж* (Великобритания) и его представителя в Парламенте *Эндрю Лэнсли*. Спецификации представлены в формате JSON:

```
{ "http://dbpedia.org/resource/Cambridge": {
"http://dbpedia.org/property/officialName": [ {
  "type": "literal", "lang" : "en",
  "value" : "City of Cambridge" } ],
"http://dbpedia.org/ontology/areaTotal": [ {
  "type": "literal", "value": 115650000,
  "datatype": "http://www.w3.org/XMLSchema#double"}],
"http://dbpedia.org/ontology/isPartOf": [
 { "type": "uri",
 "value": "http://dbpedia.org/resource/East_of_England" },
 { "type": "uri",
 "value": "http://dbpedia.org/resource/Cambridgeshire" } ],
"http://dbpedia.org/ontology/leaderName": [ {
"type": "uri",
 "value": "http://dbpedia.org/resource/Andrew_Lansley"} ]
} }

{ "http://dbpedia.org/resource/Andrew_Lansley": {
"http://dbpedia.org/property/name": [ {
  "type" : "literal", "lang" : "en",
  "value" : "Andrew Lansley" } ],
"http://dbpedia.org/ontology/birthDate": [ {
  "type": "literal", "value": "1956-12-10+02:00",
  "datatype": http://www.w3.org/XMLSchema#date } ],
"http://dbpedia.org/ontology/party": [ {
  "type": "uri",
 "value": "http://dbpedia.org/resource/Conservative_
    Party_(UK)" } ] ,
"http://dbpedia.org/ontology/electionMajority": [ {
  "type": "literal", "value": 7838,
 "datatype" : "http://www.w3.org/XMLSchema#integer"}]
} }
```

Свойства Кембриджа как объекта включают, в частности, название (*officialName*), площадь (*areaTotal*), вышестоящие территориальные образования (*isPartOf*), лидера (*leaderName*). Свойства Эндрю Лэнсли, как объекта базы знаний , включают имя (*name*), дату рождения (*birthDate*), партию (*party*), количество голосов на выборах (*electionMajority*).

Реляционное представление объектов такого рода может выглядеть следующим образом. Схема реляционной БД состоит из двух отношений, одно из которых соответствует городам (*Cities*), другое – персонам (*Persons*). Атрибутам отношений соответствуют свойства объектов. В RDF-хранилищах подобные отношения, группирующие свойства однородных объектов, называются таблицами свойств (*property tables* [29]).

Таблица 3. Отношение *Cities*

subject	officialName	area Total	isPartOf	leaderName
"http://dbpedia.org/ resource/Cambridge"	["City of Cambridge"]	[115650000]	["http://dbpedia.org/resource/ East_of_England", "http://dbpedia.org/resource/ Cambridgeshire"]	["http://dbpedia.org/ resource/Andrew_Lansley"]

Таблица 4. Отношение *Persons*

subject	name	birthdate	party	election Majority
"http://dbpedia.org/ resource/Andrew_Lansley"	["Andrew Lansley"]	["1956-12-10+02:00"]	["http://dbpedia.org/resource/ Conservative_Party_(UK) "]	[7838]

Преобразование RDF-объектов персон в кортежи отношения *Persons* может быть осуществлено при помощи функции *createPersonTuple*, определенной на языке Jaql:

```
createPersonTuple = fn(person_rdf) (
    pa = ["http://dbpedia.org/property/name",
          "http://dbpedia.org/ontology/birthDate",
          "http://dbpedia.org/ontology/party",
          "http://dbpedia.org/ontology/electionMajority"],
    properties_record = index(values(person_rdf), 0),
    if( not
       containedIn( "false",
         for($pa in pa)
           if(containedIn($pa,
                names(record(values(person_rdf)))))
             ["true"]
           else ["false"])
    )
    if( arity(person_rdf) == 1 )
    {  subject: index(names(person_rdf), 0),
       name:   for( $name in
         properties_record."http://dbpedia.org/property/
           name")   [$name.value],
     birthday: for( $birthday in
         properties_record."http://dbpedia.org/ontology/
           birthDate")  [$birthday.value],
     party:   for( $party in
        properties_record."http://dbpedia.org/ontology/
           party")   [$party.value],
     electionMajority:    for( $em in
       properties_record."http://dbpedia.org/ontology/
           electionMajority")     [$em.value]
    }
);
```

Функция принимает на вход RDF спецификацию персоны в формате JSON и возвращает кортеж, пригодный для загрузки в соответствующую реляционную таблицу. При определении функции используются условное выражение *if,* выражение цикла *for*, встроенные функции манипулирования массивами и записями *names, values, record, index* [16] и вспомогательная функция *containedIn(elm, arr),* возвращающая значение *true*, если значение *elm* является элементом массива *arr*:

```
containedIn = fn(elm, arr)(
    exists(for( $iter in arr ) if($iter == elm) [true])  );
```

Функция преобразования RDF-объектов городов определяется аналогичным образом.

4 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 всего 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[2], 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. Также анализ текста позволяет оценить тональность сообщения – выявить позитивное, нейтральное или негативное отношение автора текста к теме текста. 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.

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 [26]) 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.

5 Родственные работы

В качестве примера подхода, родственного предлагаемой в данной статье комбинированной виртуально-материализованной архитектуре, необходимо упомянуть средства федерализации, поддерживаемые в Big SQL [30]. Эти средства появились в июле 2014 г., в момент создания финальной версии статьи. Федерализация является прямым аналогом виртуальной интеграции в предметных посредниках.

Архитектура системы федерализации BigSQL (рис. 2) средства выполнения программ (*Runtime Engine*), адаптеры (*Wrappers*) и удаленные базы данных. Важным компонентом является *Optimizer*, осуществляющий планирование исполнения запроса с использованием статистической оценочной модели. Для доступа к конкретным ресурсам используются адаптеры. На текущий момент поддерживается федерализация для реляционных СУБД DB2, Oracle, Teradata и Netezza. Поддерживается реализация собственных адаптеров пользователем на языках C++ и Java.

Архитектура и подход к выполнению распределенных запросов в посредниках [1] и Big SQL обладают рядом сходных черт: слои выполнения программ, адаптеров и ресурсов; планирование запросов; настраиваемые адаптеры.

[2] Обсуждение методов и средств текстовой аналитики выходит за рамки данной статьи.

[3] Принципы построения схемы хранилища для задачи выходят за рамки данной статьи, ввиду ограниченности ее объема.

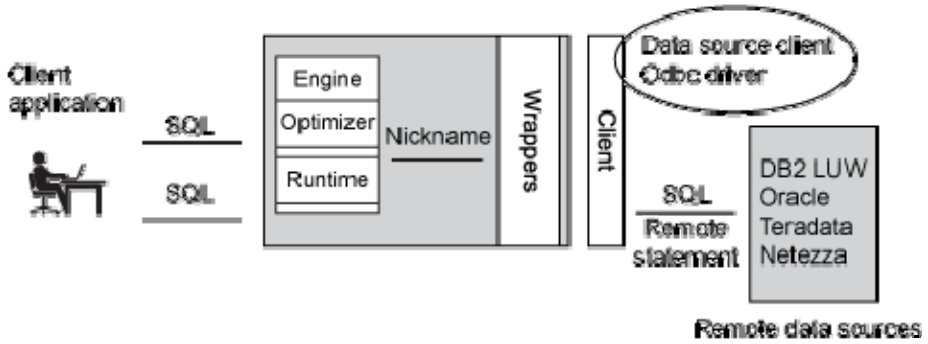

Рисунок 2. Архитектура BigSQL Federalization

Преимуществом подхода предметных посредников является ориентация на разно-модельные ресурсы. Посредники позволяют интегрировать модельно неоднородные ресурсы: реляционные, объектно-реляционные, XML, NoSQL, веб-сервисы и др. Средства BigSQL ориентированы на виртуальную интеграцию реляционных баз данных. Подробный анализ средств федерализации Big SQL и их сравнение с предметными посредниками являются задачами дальнейшей работы.

Заключение

В статье рассмотрены основные принципы организации архитектуры комбинированной виртуально-материализованной среды интеграции больших неоднородных коллекций данных и подходы к ее реализации. Целью архитектуры 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 представляется дополнением, существенно расширяющим возможности технологии предметных посредников по решению задач над неоднородными информационными ресурсами. Реализация архитектуры, ее практическое применение и сравнение с родственными подходами являются задачами дальнейшей работы.

Литература

[1] А.Е. Вовченко Рассредоточенная реализация приложений в среде предметных посредников : дис. … канд. техн. наук : 05.13.11. – М., 2012. – 216 с.

[2] 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: Ун-т города Переславля, 2012. – С. 78–87.

[3] Н.А. Скворцов. 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 гос. ун-т им. П.Г. Демидова, 2013. –С. 202–209.

[4] 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: Ун-т города Переславля, 2012. – С. 67–77.

[5] С.А. Ступников. 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 гос. ун-т им. П.Г. Демидова, 2013. – С. 193–202.

[6] С.А. Ступников, Н.А. Скворцов, В.И. Будзко, В.Н. Захаров, Л.А. Калиниченко. Методы унификации нетрадиционных моделей данных. Системы высокой доступности // Радиотехника. – 2014. – Вып. 1. – С. 18–39.

[7] Apache Hadoop Project. 2014. – http://hadoop.apache.org/

[8] Apache Spark project. 2014. – http://spark.apache.org/

[9] Edward Capriolo, Dean Wampler, Jason Rutherglen. Programming Hive Data Warehouse and Query Language for Hadoop. O'Reilly Media, 2012.

[10] 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.

[11] 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.

[12] 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.

[13] Kevin S. Beyer, Vuk Ercegovac, Rainer Gemulla, Andrey Balmin, Mohamed Eltabakh, Carl-Christian Kanne, Fatma Ozcan, Eugene J. Shekita. Jaql: A Scripting Language for Large Scale Semistructured Data Analysis. VLDB 2011.

[14] Disco framework for distributed computing based on the MapReduce paradigm. 2014. http://discoproject.org/

[15] Mauricio Hernández, Georgia Koutrika, Rajasekar Krishnamurthy, Lucian Popa, Ryan Wisnesky. HIL: a high-level scripting language for entity integration. Proceedings of the 16th International Conference on Extending Database Technology EDBT 2013. – P. 549–560.

[16] IBM InfoSphere BigInsights Information Center. 2014. – http://pic.dhe.ibm.com/infocenter/bigins/v2r1/index.jsp

[17] IBM InfoSphere Information Server Information Center. 2014. – http://pic.dhe.ibm.com/infocenter/iisinfsv/v9r1/index.jsp

[18] IBM DB2 Warehouse Information Center. 2014. http://pic.dhe.ibm.com/infocenter/db2luw/v10r5/index.jsp

[19] Introducing JSON. 2014. – http://www.json.org/

[20] Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M. Hellerstein (2010). "GraphLab: A New Parallel Framework for Machine Learning." Conference on Uncertainty in Artificial Intelligence (UAI).

[21] Donald Miner. MapReduce Design Patterns: Building Effective Algorithms and Analytics for Hadoop and Other Systems. O'Reilly Media, 2012.

[22] Cynthia M. Saracco, Uttam Jain. What's the big deal about Big SQL? Introducing relational DBMS users to IBM's SQL technology for Hadoop. IBM DeveloperWorks, 2013. – http://www.ibm.com/developerworks/library/bd-bigsql/bd-bigsql-pdf.pdf

[23] The Apache Hive data warehouse software. 2014. – http://hive.apache.org/

[24] The Forth Paradigm: Data-Intensive Scientific Discovery. Eds. Tony Hey, Stewart Tansley, and Kristin Tolle. Redmond: Microsoft Research, 2009. – http://goo.gl/GqkDX1

[25] The GraphLab Project. http://graphlab.org/projects/index.html

[26] The Neo4j Manual. 2014. – http://goo.gl/cHiOGF

[27] Tom White. Hadoop: The Definitive Guide. O'Reilly Media; Third Edition edition. 2012.

[28] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica. Spark: Cluster Computing with Working Sets. HotCloud 2010.

[29] Kevin Wilkinson, Craig Sayers, Harumi Kuno, Dave Reynolds. Efficient RDF Storage and Retrieval in Jena2. Proc. of the First International Workshop on Semantic Web and Databases. – 2003.

[30] Mara Elisa de Paiva Fernandes Matias. Set up and use federation in InfoSphere BigInsights Big SQL V3.0. 22 July 2014 (First published 08 July 2014). – http://www.ibm.com/developerworks/library/ba-federation-biginsights/index.html

Combined Virtual and Materialized Environment for Integration of Large Heterogeneous Data Collections

Sergey Stupnikov, Alexey Vovchenko

Architecture of a combined virtual and materialized environment for integration of heterogeneous data collections is provided. Collections are assumed to contain structured, semi-structured or unstructured data. Combination of virtual and materialized integration is motivated by advantages and disadvantages of both approaches. Virtual integration is supported by subject mediation technology. Materialized integration is provided by Hadoop (open source software framework for storage and distributed processing of large datasets) accompanied by a system implementing relational warehouse over Hadoop (as examples, Hive and Big SQL are considered).