МЕТОДЫ ПОСТРОЕНИЯ ОТОБРАЖЕНИЙ КОЛЛЕКЦИЙ, ПРЕДСТАВЛЕННЫХ В НЕТРАДИЦИОННЫХ МОДЕЛЯХ ДАННЫХ, В ИНТЕГРИРОВАННОЕ ПРЕДСТАВЛЕНИЕ[1]

С. А. СТУПНИКОВ[2], А. Е. ВОВЧЕНКО[3]

Аннотация. В статье рассмотрены вопросы преобразования коллекций, представленных в нетрадиционных моделях данных (таких, как графовые, триплетные, модели ключ-значение), в их интегрированное представление в реляционной модели данных. Общим контекстом, в котором находится работа, является создание комбинированной виртуально-материализованной архитектуры среды интеграции неоднородных коллекций разноструктурированных данных. Рассмотренные подходы к преобразованию коллекций являются основой для материализованной интеграции информационных ресурсов в реляционных хранилищах данных над Hadoop.

Ключевые слова: интеграция баз данных, графовые модели данных, RDF, NoSQL, трансформация коллекций данных

1 Введение

Растущее разнообразие моделей данных в информационных технологиях, наблюдаемое в настоящее время, требует создания подходов к интеграции моделей и коллекций данных, представленных в этих моделях. Методы создания унифицированного представления различных видов нетрадиционных моделей в канонической информационной модели (общем языке, унифицирующем

[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).

[2] Институт проблем информатики РАН, ssa@ipi.ac.ru

[3] Институт проблем информатики РАН, alexey.vovchenko@gmail.com

языки разнообразных моделей данных) в последнее время активно исследовались в ИПИ РАН. Были рассмотрены подходы к унификации моделей данных различных классов: семантических (OWL, RDF [1]), графовых [2], многомерных массивов, NoSQL-моделей [3]. Была предложена также архитектура комбинированной виртуально-материализованной архитектуры среды интеграции неоднородных коллекций данных различного вида (структурированных, слабоструктурированных и неструктурированных) [4]. Основные черты архитектуры рассмотрены на рис. 1. Среда поддерживает как виртуальную, так и материализованную интеграцию коллекций данных, представленных как в традиционных (реляционных), так и нетрадиционных моделях данных.

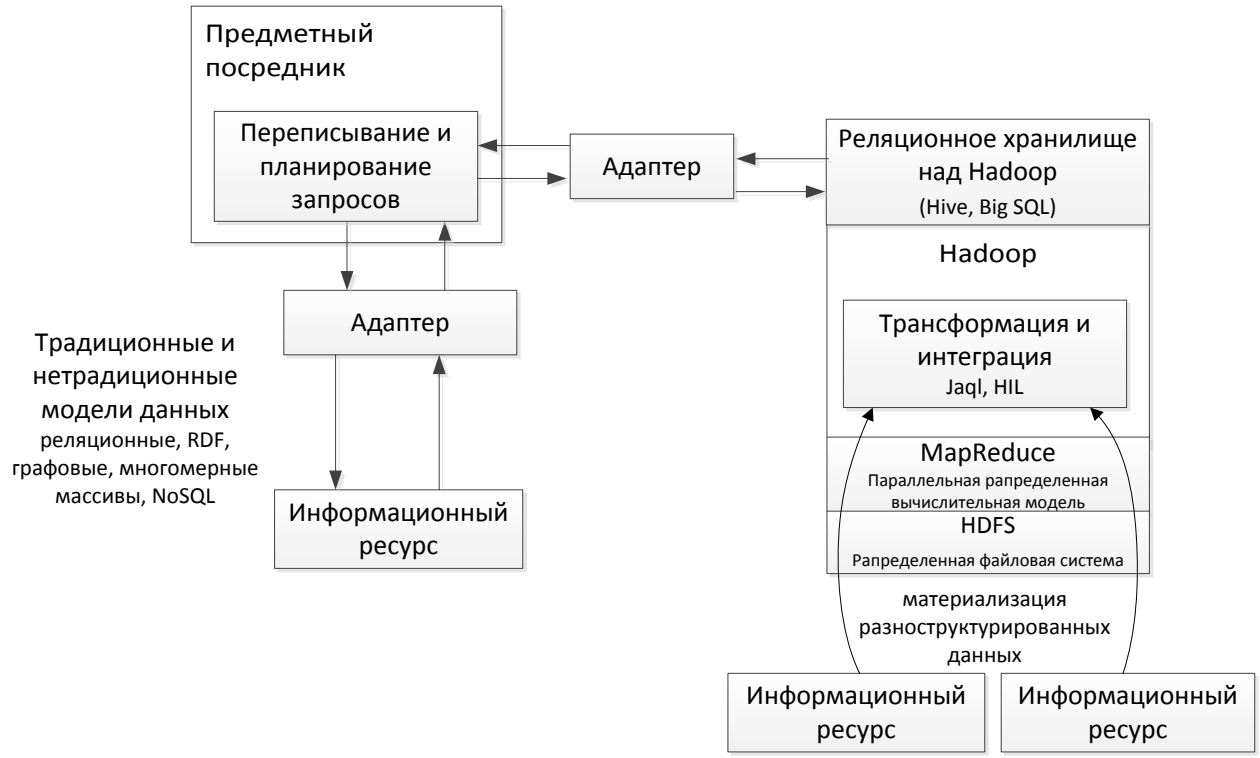

Рис. 1. Архитектура среды виртуально-материализованной интеграции

Виртуальная интеграция осуществляется с использованием технологии предметных посредников [5], образующих промежуточный слой между пользователем (приложением) и неоднородными информационными ресурсами. При этом данные из ресурсов не материализуются в посреднике. 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. Материализованную интеграцию предлагается реализовывать с использованием свободно распространяемой платформы распределенного хранения и обработки данных Hadoop [6]; а также системы организации реляционных хранилищ данных над Hadoop (WR-Hadoop для краткости), в качестве которой могут использоваться, например, платформы Big SQL [7] или Hive [8] (система Hive является свободно распространяемым решением, а Big SQL – проприетарным, распространяемым в составе продукта IBM InfoSphere BigInsights [9]). Основные принципы построения среды интеграции неоднородных коллекций данных и подходы к ее реализации рассмотрены в работе [4]. Данная же работа сосредоточена на более узком вопросе преобразования коллекций, представленных в нетрадиционных моделях данных, в их интегрированное представление в реляционной модели данных системы WR-Hadoop. Этот вопрос является актуальным, поскольку известные системы организации реляционных хранилищ над Hadoop (такие, как Hive и Big SQL) не в полной мере реализуют функции хранилищ данных. В частности, напрямую не поддерживается процесс извлечения-преобразования-загрузки (ETL). Для реализации недостающих функций в среде интеграции [4] предлагается использование декларативно-императивных языков высокого уровня над Hadoop.

Материализация в среде 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 и т.д. Для определенности в данной статье в качестве основного формата рассматривается JavaScript Object Notation (JSON [10]). Преобразование данных к реляционному виду для последующей интеграции производится при помощи программ на языке Jaql [11] (язык запросов и сценариев, разработанный IBM и использующий формат JSON; поставляется в составе IBM InfoSphere BigInsights [9] – программной платформы обработки больших данных, основанной на Hadoop). Конструирование преобразований

коллекций в настоящий момент производится вручную, на формальной базе унификации нетрадиционных моделей [3][1][2]. При унификации осуществляется конструирование сохраняющих семантику отображений схем исходных моделей в каноническую модель, а также отображений операций манипулирования данными. На основании этих отображений и происходит конструирование преобразований коллекций. Автоматизация генерации преобразований является предметом дальнейшей работы.

В данной статье рассматриваются и иллюстрируются на примерах подходы к преобразованию в реляционное представление коллекций данных трех нетрадиционных моделей: графовой модели Neo4J [12] (раздел 2), семантической модели RDF [13] (раздел 3) и NoSQL модели СУБД HBase [14] (раздел 4).

2 Преобразование коллекций данных графовых моделей

Рассмотрим пример коллекции данных, представленной в модели данных графовой СУБД Neo4j [12]. Небольшой подграф базы данных о фильме «Матрица», включающий основные виды вершин, ребер и атрибутов, изображен на рис. 2.

Вершины графа соответствуют людям (PEOPLE) и программам (PROGRAM), ребра графа соответствуют знакомству людей и программ друг с другом (KNOWS) и созданию программы человеком (CODED_BY). Люди характеризуются именем (*name*), фамилией (*lastName*), возрастом (*age*), родом занятий (*occupation*), званием (*rank*); программы – названием (*name*), языком программирования (*language*), версией (*version*); знакомство людей и программ – длительностью (*age*), степенью публичности (*disclosure*). Вершины и ребра графа обладают уникальными идентификаторами.

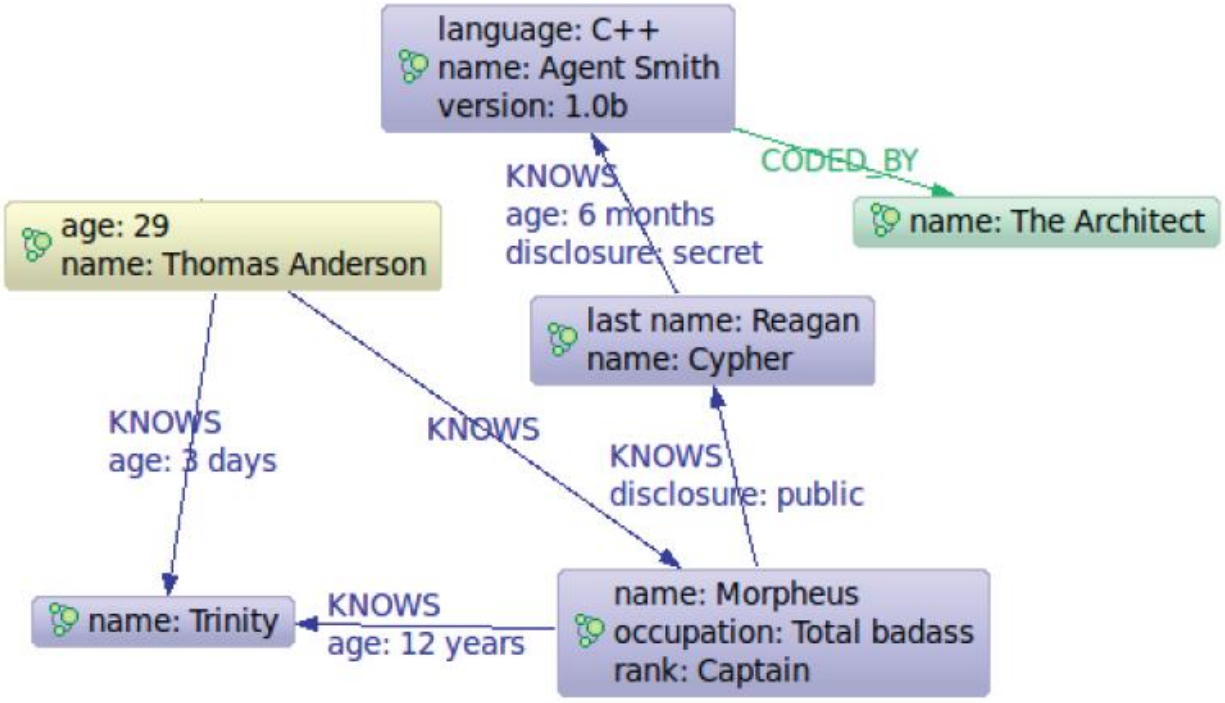

Рисунок 2. Пример графа в модели данных Neo4j

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

```
[{ "id": 1, "type": "node", "labels": ["PERSON"],
"properties": {"name": "Thomas", "lastName": "Anderson",
               "age": 29} },
{ "id": 2, "type": "node", "labels": ["PERSON"],
"properties": {"name": "Trinity"} },
{ "id": 3, "type": "node", "labels": ["PERSON"],
"properties": {"name": "Morpheus", "occupation": "Total badass",
"rank": "Captain"} },
{ "id": 4, "type": "node", "labels": ["PERSON"],
"properties": {"name": "Cypher", "lastName": "Reagan"} },
{ "id": 5, "type": "node", "labels": ["PERSON"],
"properties": {"name": "The Architect"} },
{ "id": 6, "type": "node", "labels": ["PROGRAM"],
"properties": {"name": "Agent Smith", "language": "C++",
               "version": "1.0b"} },
```

```
{ "id": 7, "type": "relationship", "relationship_type": "KNOWS",
"start_node": 1, "end_node": 2,
"properties": {"age": "3 days"} },
{ "id": 8, "type": "relationship", "relationship_type": "KNOWS",
"start_node": 1, "end_node": 3 },
{ "id": 9, "type": "relationship", "relationship_type": "KNOWS",
"start_node": 3, "end_node": 2,
"properties": {"age": "12 years"} },
{ "id": 10, "type": "relationship", "relationship_type": "KNOWS",
"start_node": 3, "end_node": 4,
"properties": {"disclosure": "public"} },
{ "id": 10, "type": "relationship", "relationship_type": "KNOWS",
"start_node": 4, "end_node": 6,
"properties": {"age": "6 months", "disclosure": "secret"} },
{ "id": 10, "type": "relationship",
"relationship_type": "CODED_BY"
"start_node": 6, "end_node": 5
}]
```

Здесь *start_node* и *end_node* обозначают исходящую и входящую вершины ребра соответственно; *labels* обозначает множество меток, присвоенных вершине; *properties* означает атрибуты (свойства) вершин и ребер.

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**	**age**	**name**	**lastName**	**occupation**	**rank**	**language**	**version**
1	["PERSON"]	29	"Thomas"	"Anderson"	null	null	null	null
2	["PERSON"]	null	"Trinity"	null	null	null	null	null
3	["PERSON"]	null	"Morpheus"	null	"Total badass"	"Captain"	null	null

4	["PERSON"]	null	"Cypher"	"Reagan"	null	null	null	null
5	["PERSON"]	null	"The Architect"	null	null	null	null	null
6	["PROGRAM"]	null	"Agent Smith"	null	null	null	"C++"	"1.0b"

Таблица 2 Отношение *Relationships*

id	start_node	end_node	relationship_type	age	disclosure
7	1	2	"KNOWS"	"3 days"	null
8	1	3	"KNOWS"	null	null
9	3	2	"KNOWS"	"12 years"	null
10	3	4	"KNOWS"	null	"public"
11	4	6	"KNOWS"	"6 months"	"secret"
12	6	5	"CODED_BY"	null	null

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

```
createNodesRelation = fn(matrix_graph_db)(
  matrix_graph_db->filter $.type == "node"->
        transform { id: $.id, labels: $.labels,
         age: $.properties.age,
         name: $.properties.name,
         lastName: $.properties.lastName,
         rank: $.properties.rank,
         occupation: $.properties.occupation,
         version: $.properties.version,
         language: $.properties.language }  );

createRelationshipRelation = fn(matrix_graph_db)(
  matrix_graph_db->filter $.type == "relationship"->
        transform { id: $.id,
         start_node: $.start_node, end_node: $.end_node,
         relationship_type: $.relationship_type,
```

```
        age: $.properties.age,
        disclosure: $.properties.disclosure, }  );
```

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 (->).

3 Преобразование коллекций данных модели RDF

В качестве примера коллекции данных модели RDF [13] рассмотрим базу знаний *DBpedia* (http://dbpedia.org/). Коллекция содержит структурированную информацию, извлеченную из свободной энциклопедии Википедия. В качестве примеров данных рассмотрим RDF-спецификации города *Кембридж* (Великобритания) и его представителей в Парламенте *Эндрю Лэнсли* (от партии консерваторов) и Джулиана Хапперта (от партии либералов). 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/Julian_Huppert" } ,
 { "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 парламентариев как объектов 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* [15]).

Таблица 3. Отношение *Cities*

subject	officialName	areaTotal	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/Julian_Huppert", "http://dbpedia.org/resource/Andrew_Lansley"]

Таблица 4. Отношение *Persons*

subject	name	birthdate	party	electionMajority
"http://dbpedia.org/resource/Andrew_Lansley"	["Andrew Lansley"]	["1956-12-10+02:00"]	["http://dbpedia.org/resource/Conservative_Party_(UK) "]	[7838]
"http://dbpedia.org/resource/Julian_Huppert"	["Huppert, Julian", "Dr Julian Huppert"]	["1978-07-21+02:00"]	"http://dbpedia.org/re-source/Liberal_Democrats"	[6792]

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

```
createCityTuple = fn(city_rdf) (
  pa = ["http://dbpedia.org/property/officialName",
        "http://dbpedia.org/ontology/areaTotal",
        "http://dbpedia.org/ontology/isPartOf",
        "http://dbpedia.org/ontology/leaderName"],
  properties_record = index(values(city_rdf), 0),
   if( not containedIn( "false",
     for($pa in pa)
      if(containedIn($pa, names(record(values(city_rdf)))))
        ["true"]
      else ["false"]) )
   if( arity(city_rdf) == 1 )
   { subject: index(names(city_rdf), 0),
    officialName: for( $name in
     properties_record."http://dbpedia.org/property/officialName")
     [$name.value],
```

```
    areaTotal: for( $area in
      properties_record."http://dbpedia.org/ontology/areaTotal")
      [$area.value],
    isPartOf: for( $partOf in
      properties_record."http://dbpedia.org/ontology/isPartOf")
      [$partOf.value],
    leaderName: for( $leader in
      properties_record."http://dbpedia.org/ontology/leaderName")
      [$leader.value]
  }
);
```

Функция принимает на вход RDF спецификацию города 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* [9] 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])  );
```

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]
    }
  );
```

4 Преобразование коллекций данных модели HBase

В качестве примера коллекции данных модели «ключ-значение» рассмотрим фрагмент базы данных о книгах по информационным технологиям в СУБД HBase [14].

База данных HBase в общем виде представляет собой множество пар *ключ → значение*, где *ключ* – это четверка *(ключ строки, семейство стобца, идентификатор столбца, временная метка)*, а значение – обычное значение некоторого встроенного типа (*boolean, int, float* и т.д.). Фрагмент базы данных о книгах в общем виде выглядит следующим образом:

```
(01, language, natural, t1) → “English”
(01, language, programming, t1) → “Java”
```

```
(01, description, edition, t1) → ”second”
(01, description, edition, t2) → “third”
(01, description, author, t1) → “Herbert Schildt”
(07, language, natural, t1) → “Russian”
(07, language, programming, t1) → “C++”
(07, description, edition, t1) → “second”
(07, description, author, t1) → “E.Balagurusamy”
```

Здесь *t1*, *t2* – временные метки. В формате JSON эти данные могут быть представлены следующим образом:

```
[ { "key":"01",
    "language": { "natural": {t1:"English"},
                  "programming": {t1:"Java"} },
    "description": { "edition": [{t1:"second"}, {t2:"third"}],
      "author": {t1:"Herbert Schildt"}}   },
   {"key":"07",
    "language": { "natural": {t1:"Russian"},
                  "programming": {t1:"C++"} },
    "description": { "edition": {t1:"second"},
      "author": {t1:"E.Balagurusamy"}}    }
]
```

Возможны различные способы представления данных модели HBase в реляционном виде. В данном разделе будут рассмотрены два из них.

При *первом способе* используется фиксированная реляционная схема (для произвольной базы данных), состоящая из одного отношения *HBaseData(String key, String column_family, String column_qualifier, String timestamp, String value)*. Первичным ключом отношения *HBaseData* является совокупность атрибутов <*key*, *column_family*, *column_qualifier*, *timestamp*>. Количество кортежей в отношении соответствует количеству пар ключ-значение в базе данных HBase. Реляционное представление рассмотренного выше фрагмента базы данных о книгах при этом способе приведено в таблице 5.

Таблица 5 Отношение *HBaseData*

key	column family	column qualifier	timestamp	value
01	language	natural	t1	“English”
01	language	programming	t1	"Java"
01	description	edition	t1	”second”
01	description	edition	t2	“third”
01	description	author	t1	“Herbert
07	language	natural	t1	“Russian”
07	language	programming	t1	“C++”
07	description	edition	t1	“second”
07	description	author	t1	“E.Balagurusamy”

Преобразование исходных данных в реляционное представление может быть осуществлено при помощи функции *createNativeRelation* на языке Jaql:

```
createNativeRelation(hbasedata) (
 for ($obj in hbasedata) (
  for ($f in names($obj)) (
    if ($f != 'key') (
      for ($q in names($obj.($f))) (
        for ($t in names($obj.($f).($q))) (
         [{ key: $obj.key, column_family: $f,
            column_qualifier: $q, timestamp: $t,
             value: $obj.($f).($q).($t) }]
))))) );
```

Второй способ предполагает использование реляционной схемы, зависящей от базы данных HBase. При этом для каждой группы однородных объектов (имеющих значения для одного набор колонок) создается свое отношение в реляционной схеме. Так, в рассматриваемом примере присутствует одна группа однородных объектов – книги, имеющих значения для набора колонок *(natural, programming, edition , author)*. Идея подхода в том, чтобы собрать все данные об одном объекте (из пар *ключ* → *значение* с одним значением ключа) в одном

кортеже. Реляционное представление фрагмента базы данных о книгах при этом способе приведено в таблице 6.

Таблица 6 Отношение *Books*

key	**language_natural**	**language_programming**	**description_edition**	**description_author**
01	t1:“English”	t1:"Java"	t1:”second”;t2:"third"	t1:"Herbert Schildt"
07	t1:“Russian”	t1:"C++"	t1:”second”	t1:"E.Balagurusamy"

Преобразование исходных данных в реляционное представление может быть осуществлено при помощи функции *createBooksRelation* на языке Jaql:

```
createBooksRelation(hbasedata) (
  hbasedata -> transform {
    "key": $.key,
    "language_natural": $.language.natural,
    "language_programming": $.language.programming,
    "description_edition": $.description.edition,
    "description_author": $.description.author  }
);
```

Заключение

В статье рассмотрены вопросы преобразования коллекций, представленных в нетрадиционных моделях данных (таких, как графовые, триплетные, модели ключ-значение), в их интегрированное представление в реляционной модели данных. Общим контекстом, в котором находится работа, является создание комбинированной виртуально-материализованной архитектуры среды интеграции неоднородных коллекций данных различного вида (структурированных, слабоструктурированных и неструктурированных). Рассмотренные подходы к преобразованию коллекций являются основой для материализованной интеграции информационных ресурсов в реляционных хранилищах данных над Hadoop. Задачами дальнейшей работы являются создание методов автома-

тической генерации функций преобразования коллекций и их применение при решении конкретных задач интеграции коллекций.

Список литературы

1. 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 государственный университет им. П. Г. Демидова, 2013. С. 202-209.
2. 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 государственный университет им. П. Г. Демидова, 2013. С. 193-202.
3. 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.
4. С. А. Ступников, А. Е. Вовченко Комбинированная виртуально-материализованная среда интеграции неоднородных коллекций данных // Труды 16-й Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электроные коллекции» RCDL'2014. – Дубна: ОИЯИ, 2014. – принято в печать.
5. Брюхов Д.О., Вовченко А. Е., Захаров В.Н., Желенкова О.П., Калиниченко Л.А., Мартынов Д.О., Скворцов Н.А., Ступников С.А. Архитектура промежуточного слоя предметных посредников для решения задач над множеством интегрируемых неоднородных распределенных информационных ресурсов в гибридной грид-инфраструктуре виртуальных обсерваторий // Информатика и ее применения. – М., 2008. – Т. 2, Вып. 1. – С. 2-34.
6. Apache Hadoop Project. 2014. - http://hadoop.apache.org/
7. 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
8. Edward Capriolo, Dean Wampler, Jason Rutherglen. Programming Hive Data Warehouse and Query Language for Hadoop. O'Reilly Media, 2012.

9. IBM InfoSphere BigInsights Information Center. 2014. - http://pic.dhe.ibm.com/infocenter/bigins/v2r1/index.jsp
10. Introducing JSON. 2014. - http://www.json.org/.
11. 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.
12. The Neo4j Manual. 2014. - http://goo.gl/cHiOGF
13. R. Cyganiak, D. Wood, M. Lanthaler. (Eds.) RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation 25 February 2014. - http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/
14. Lars George. HBase: The Definitive Guide. O'Reilly Media, 2011. - 556 P.
15. 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.

Ступников Сергей Александрович – к.т.н., с.н.с. лаб. Композиционных методов и средств построения информационных систем Института проблем информатики РАН, ssa@ipi.ac.ru

Sergey Stupnikov – Ph.D. in Theoretical Informatics, senior scientist, Institute of Informatics Problems, Russian Academy of Sciences, ssa@ipi.ac.ru

Вовченко Алексей Евгеньевич - к.т.н., с.н.с. ИПИ РАН, alexey.vovchenko@gmail.com

Vovchenko Alexey, Ph. D., senior scientist, Institute of Informatics Problems, Russian Academy of Sciences, alexey.vovchenko@gmail.com