

# Спецификация и реализация разномодельных правил интеграции данных

© С. А. Ступников  
Институт проблем информатики ФИЦ ИУ РАН,  
Москва  
sstupnikov@ipiran.ru

**Аннотация.** В работе рассматривается подход к спецификации правил интеграции данных с использованием рекомендации W3C - логического диалекта RIF-BLD. Это позволяет использовать в одном правиле сущности из разных коллекций, представленных в разных моделях данных. Логическая семантика RIF-BLD также позволяет однозначным образом интерпретировать спецификации рассматриваемых правил интеграции. Предложен подход к реализации правил RIF-BLD в языке HIL: это позволяет компилировать правила интеграции в программы вычислительной модели MapReduce и исполнять их в распределенных инфраструктурах, основанных на Hadoop.

**Ключевые слова:** интеграция данных, модели данных, логические правила, реализация правил

## Specification and Implementation of Multimodel Data Integration Rules

© Sergey Stupnikov  
Institute of Informatics Problems, Federal Research Center “Computer Science and Control” of  
the Russian Academy of Sciences,  
Moscow  
sstupnikov@ipiran.ru

**Abstract.** The paper considers an approach for specification of data integration rules using RIF-BLD logic dialect that is a recommendation of W3C. This allows us to reference entities defined in different collections represented using different data models in the same rule. Logical semantics of RIF-BLD provides also unambiguous interpretation of data integration rules. The paper proposes an approach for implementation of RIF-BLD rules using HIL language. Thus data integration rules are compiled into MapReduce programs and can be executed over Hadoop-based distributed infrastructures.

**Keywords:** data integration, data models, logic rules, rule implementation

### 1 Введение

Хранилища данных в настоящее время являются одной из основных составляющих инфраструктур поддержки систем бизнес-аналитики. Данные извлекаются из различных коллекций, преобразуются к схеме хранилища, интегрируются и загружаются в хранилище. Над хранилищем выстраивается слой приложений, осуществляющих анализ данных (например, при помощи методов математической статистики, машинного обучения и др.) и выдающих результат пользователю в необходимой форме.

Ввиду роста неоднородности источников данных, их количества и объемов данных, актуальными остаются вопросы разработки методов спецификации и реализации интеграции данных в масштабируемых инфраструктурах распределенного хранения и обработки данных, подобных Apache Hadoop [1].

Традиционные промышленные решения по интеграции данных и разработке хранилищ (например, IBM InfoSphere Information Server [2]) предлагают визуальные средства проектирования потоков работ интеграции данных, оперирующие, преимущественно, понятиями и операциями реляционной модели и порождающие программы трансформации и интеграции данных на языке SQL. Отдельно выделяются визуальные средства сопоставления схем исходных коллекций данных и схемы хранилища (целевой схемы), например,

---

Труды XIX Международной конференции «Аналитика и управление данными в областях с интенсивным использованием данных» (DAMDID/RCDL'2017), Москва, Россия, 10-13 октября 2017

InfoSphere FastTrack [2], позволяющие автоматически порождать части потоков работ интеграции данных. Для обеспечения интеграции данных, представленных в различных моделях данных, такие средства предоставляют отдельные компоненты, позволяющие преобразовывать исходные данные к реляционной модели (например, компонент преобразования XML в IBM InfoSphere DataStage [3]).

Параллельно промышленным решениям, проводятся исследования методов формальной спецификации правил интеграции данных. Классическим примером работы в данном направлении является подход *обмена данными* [9]. Подход позволяет описывать интеграцию данных реляционной модели с использованием логических правил, которым сообщается формальная семантика в логике предикатов первого порядка.

В данной статье рассматривается подход к спецификации правил интеграции данных с использованием рекомендации W3C - логического диалекта RIF-BLD [6]. RIF-BLD является диалектом каркаса логических диалектов RIF-FLD формата обмена правилами RIF [5], нацеленного на унификацию синтаксиса и семантики языков логических правил. RIF-BLD включает достаточно широкий спектр возможностей спецификации, в частности, позиционные термины и термины с именованными аргументами (обобщающие понятие термина в логике первого порядка), фреймовые термины (выражающие утверждения о структуре объектов), классификационные термины, термины равенства, внешние термины (использующиеся для ссылок на сущности, рассматриваемые как «черные ящики» в пределах спецификации). Это позволяет использовать в одном правиле сущностей из разных коллекций, представленных в разных моделях данных. Рассматриваются правила, в голове которых могут присутствовать лишь предикаты, соответствующие сущностям целевой схемы, а в теле – предикаты, соответствующие сущностям исходных схем.

Логическая семантика RIF-BLD позволяет однозначным образом интерпретировать спецификации рассматриваемых правил интеграции и допускает их реализацию с использованием различных языков – от декларативных (например, SQL) до императивных (например, Java). В данной работе рассматривается подход к реализации логических правил RIF-BLD в языке высокого уровня HIL [10][7], разработанного компанией IBM, и поставляемого в составе Hadoop-решения BigInsights 3.0 [11], а также как часть InfoSphere Big Match for Hadoop [12]. Распределенное исполнение программ на HIL в среде Hadoop достигается путем их компиляции в программы на Java, которые, в свою очередь, исполняются с использованием средств поддержки в Hadoop вычислительной модели MapReduce [14].

Подходы к спецификации и реализации правил интеграции иллюстрируются на примерах

интеграции неоднородных коллекций данных по Арктической зоне в хранилище информации, нацеленной на поддержку поисково-спасательных операций. Были выбраны неоднородные коллекции данных, подлежащие интеграции [16] и разработана единая схема хранилища [20]. В данной статье в качестве примеров рассматриваются коллекции, представленные в реляционной модели, XML, документной модели MongoDB, графовой модели Neo4j. Схема хранилища представлена в реляционной модели. В дальнейшем предполагается использование результатов работы в различных проектах, предусматривающих интеграцию данных.

Структура статьи выглядит следующим образом. В разделах 2-4 иллюстрируются подход к спецификации правил интеграции данных, представленных в различных моделях (раздел 2 – XML и реляционная модель, раздел 3 – документная модель, раздел 4 – графовая модель) с использованием диалекта RIF-BLD и подход к реализации правил в языке HIL. В разделе 5 обобщаются применяемые принципы спецификации и реализации правил интеграции.

## 2 Спецификация и реализация правил интеграции данных XML и реляционной модели с разрешением конфликтов

В данном разделе рассматриваются два примера правил интеграции данных, оперирующих сущностями XML и реляционной модели.

В первом примере рассматривается правило преобразования данных из XML к реляционной схеме хранилища (целевой схеме) с разрешением структурных конфликтов, конфликтов имен и значений.

Во втором примере рассматривается правило преобразования данных из соединения двух коллекций, одна из которых представлена в XML, другая – в реляционной модели.

### 2.1 Интеграция данных о маршрутах объектов

В левом столбце таблицы 1 приведен пример данных в формате XML о маршруте объекта, полученных из системы мониторинга, учета и классификации судов КИИС «MoPe» [18]. Маршрут (элемент *ISSKOI\_Track*) состоит из маршрутных точек (*ISSKOI\_TrackPoint*). В каждой точке заданы значения координат (*pos*), времени (*Time*) и высоты (*BarAltitude*). Данные о других маршрутах имеют ту же структуру элементов и отличаются значениями элементов и атрибутов.

В правом столбце таблицы приведены элементы целевой схемы, соответствующие исходным данным. Так, элемент *ISSKOI\_Track* соответствует отношению целевой схемы *Track*, вложенный элемент *Id* соответствует первичному ключу (PK) *Track.trackId*, вложенный элемент *TrackName* – атрибуту *Track.name*. Элемент *ISSKOI\_TrackPoint* соответствует отношению целевой схемы

*TrackPoint*, атрибут элемента *id* соответствует первичному ключу *TrackPoint.pointId*, вложенные элементы *Time* и *BarAltitude* – атрибутам *TrackPoint.time* и *TrackPoint.height* соответственно, вложенный элемент *pos* – атрибутам *TrackPoint.longitude* и *TrackPoint.latitude*.

**Таблица 1** Данные о маршруте объекта и соответствующие элементы целевой схемы

Пример данных в исходной модели (XML)	Элементы целевой схемы (реляционная модель)
<pre>&lt;ISSKOI_Track&gt; &lt;Id&gt;56473&lt;/Id&gt; &lt;TrackName&gt;copter-1&lt;/TrackName&gt; &lt;ISSKOI_TrackPoints&gt; &lt;ISSKOI_TrackPoint   id="uuid-2b7ca14"&gt;   &lt;Position&gt;   &lt;Point id="uuid-859bef91"&gt;   &lt;pos&gt;33.8957 246.37&lt;/pos&gt;   &lt;/gml:Point&gt;   &lt;/Position&gt;   &lt;Time&gt;2016-12-12     T13:33:11&lt;/Time&gt;   &lt;BarAltitude&gt;533.89   &lt;/BarAltitude&gt;   &lt;HSpeed&gt;108.1&lt;/HSpeed&gt;   &lt;VSpeed&gt;2&lt;/VSpeed&gt;   &lt;/TrackPoint&gt; &lt;/TrackPoints&gt; &lt;/Track&gt;</pre>	<pre>Track(   PK trackId,   name)  TrackPoint(   PK pointId,   FK path,   time,   height,   latitude,   longitude)</pre>

В рассмотренном примере, как и в других примерах, приведенных ниже, приведена лишь часть элементов, составляющих исходные данные и целевую схему.

Очевидно, что при спецификации правила интеграции данных о маршрутах необходимо разрешить конфликты имен (например, элемент *BarAltitude* и атрибут *height* имеют одинаковую семантику, но разные имена), структурные конфликты и конфликты значений (элемент *pos*, содержащий широту и долготу, соответствует двум отдельным атрибутам *longitude*, *latitude*).

С использованием диалекта RIF-BLD необходимое правило интеграции описывается следующим образом:

```
forall ?Track ?Id ?TrackName ?TrackPoints
?TrackPoint ?Position ?Time ?Height ?Point ?pos
( AND( Track(trackId->?Id name->?TrackName)
  TrackPoint(pointId->?pid path->?Id
    time->?Time height->?Height
    latitude->External(get_latitude(?pos))
    longitude->External(get_longitude(?pos)) ) )
:-
AND(?Track#ISSKOI_Track
  ?Track[Id->?Id TrackName->?TrackName
    TrackPoints->?TrackPoint]
  ?TPoint#ISSKOI_TrackPoint
  ?TPoint[id->?pid Position->?Position
    Time->?Time BarAltitude->?Height]
  ?Position#Position ?Position[Point->?Point]
  ?Point#Point ?Point[pos->?pos]))
```

*forall* в правиле обозначает квантор всеобщности, знак «:-» обозначает импликацию – логическое следование формулы-головы правила из формулы-тела правила, операция AND обозначает конъюнкцию. Идентификаторы вида ?X обозначают переменные.

В правиле используется три вида предикатов. В голове правила используются *предикаты с именованными аргументами* [6] *Track* и *TrackPoint*, соответствующие отношениям целевой схемы.

В теле правила используются *предикаты членства* [6] (например, предикат *?Track#ISSKOI\_Track*, обращающийся в истину, когда переменная *?Track* принимает значение произвольного элемента *ISSKOI\_Track*) и *фреймовые предикаты* [6], отражающие структуру XML-элементов исходных данных. Так, предикат *?Point[pos->?pos]* обращается в истину на таких парах значений переменных *?Point* и *?pos*, что элемент – значение переменной *?Point* имеет вложенный элемент *pos* и его значение есть *?pos*. Конъюнкция предикатов в теле правила полностью задает структуру вложенных элементов и атрибутов произвольного элемента *ISSKOI\_Track*.

Связь значений атрибутов отношений в голове правила и значений элементов и атрибутов в теле правила задается при помощи переменных, таким образом разрешаются конфликты имен и структурные конфликты.

Конфликты значений могут быть разрешены при помощи функций (например, *get\_latitude*), представляемых в RIF-FLD как *внешние термы (External)*. Их семантика в рамках RIF-FLD напрямую не определяется и уточняется при реализации (например, семантика функции *get\_latitude* состоит в том, чтобы вернуть первое число, входящее в исходную строку).

Рассмотренное правило имеет естественную логическую семантику: для всех наборов значений подкванторных переменных, обращающих в истину все предикаты тела правила на исходной коллекции, отношения хранилища должны содержать кортежи, соответствующие предикатам головы правила с тем же набором значений переменных. Для рассмотренного примера хранилище должно содержать кортежи *Track(trackId: 56473, name: "copter-1")*, *TrackPoint(pointId: "uuid-2b7ca14", path: 56473, time: "2016-12-12T13:33:11", height: 533.89, latitude: 33.8957, longitude: 246.37)*.

Вышеописанное правило может быть реализовано в языке NPL следующей программой:

```
declare ISSKOI_Track: ?;
declare Track: ?;
declare TrackPoint: ?;

declare get_latitude:
  function string to double;
declare get_longitude:
  function string to double;

@jaql{
get_latitude = fn($s) convert(substring(
  $s, 0, strpos($s, ' ')-1), schema double);
```

```

get_longitude = fn($s) convert(substring(
  $s, strpos($s, ' ') + 1, strlen($s)),
  schema double);
}

```

```

insert into Track
select [ trackId: t."Id", name: t."TrackName" ]
from ISSKOI_Track t;

```

```

insert into TrackPoint
select [ path: t."Id", time: p."Time",
  height: tpt."BarAltitude",
  hSpeed: tpt."hSpeed", vSpeed: tpt."vSpeed",
  course: tpt."Course",
  latitude: get_latitude(pt."pos"),
  longitude: get_longitude(pt."pos") ]
from ISSKOI_Track t, t."TrackPoints" tpt,
tpt."Position" ps, ps."Point" pt;

```

В программе объявляются (*declare*) исходная сущность (внешний элемент *ISSKOI\_Track*) и целевые сущности, соответствующие отношениям (*Track*, *TrackPoint*). Знак «?» в объявлении означает, что структура сущностей не задана при определении, а выводится из программы.

Объявляются функции разрешения конфликтов (*get\_latitude*, *get\_longitude*) и их сигнатуры. Реализация функций производится с использованием языка Jaql [4] (директива *@jaql*) и его функций работы со строками *substring*, *strpos*.

Для целевых отношений *Track*, *TrackPoint* определяются операторы *insert*, порождающие кортежи этих отношений. В секции *from* операторов *insert* объявляются исходные сущности, каждому предикату членства в теле правила RIF-BLD (например, *?Track#ISSKOI\_Track*) соответствует объявление с точностью до имени переменной (например, *ISSKOI\_Track t*).

Атрибуты целевых отношений и выражения порождения их значений определяются в секции *select* в соответствии с предикатами в голове правила и фреймовыми предикатами в теле правила. Например, предикату головы *Track(trackId->?Id)* и фреймовому предикату тела *?Track[Id->?Id]* соответствует определение *trackId: t."Id"* в секции *select* оператора *insert into Track*.

Заметим, что язык HIL оперирует данными в формате JSON [13], поэтому для применения рассмотренного правила на языке HIL для преобразования данных о маршрутах в целевую схему необходимо преобразовать исходные XML-документы в JSON при помощи встроенной функции *xmlToJson* языка Jaql [4]. Полученные на выходе HIL-программы JSON-документы затем загружаются в реляционное хранилище над *Nadoor* (например, *Hive* [15]).

## 2.2 Интеграция данных о судах

В левом столбце таблицы 2 приведен пример данных о судах в формате XML, полученный из системы «Поиск-Море» (*ERRTableShips*) [19]; а также пример данных о судах в реляционной модели, полученный из системы ЕСИМО [17] (данные получены в формате CSV и преобразованы в JSON). В обеих исходных коллекциях могут быть

найжены данные об одних и тех же судах (судно идентифицируется по названию и позывному). Кроме того, коллекции содержат различные взаимодополняющие данные о судах.

**Таблица 2** Данные о судах и соответствующие элементы целевой схемы

Пример данных в исходных моделях (XML, реляционная)	Элементы целевой схемы (реляционная модель)
<pre> &lt;ERRTableShips&gt; &lt;Id&gt;64694571&lt;/Id&gt; &lt;Name&gt;мвс Ростов Великий&lt;/Name&gt; &lt;Callsing&gt;UBZG5&lt;/Callsing&gt; &lt;Dates&gt;   &lt;StartDate&gt;2017-01-08   &lt;/StartDate&gt;   &lt;EndDate&gt;2017-01-09&lt;/EndDate&gt; &lt;/Dates&gt; &lt;/ERRTableShips&gt;  [ {   "Platforma_nazvanie":     "мвс Ростов Велкий",   "Strana_naimenovanie":     "Россия",   "Organizaciya_nazvanie":     "ФГУП Балтийское ВАСУ -     Сахалинский филиал",   "Pozyvnoj": "UBZG5",   "nomer_IMO": "9586796" } ] </pre>	<pre> SARUnit(   beginDuty,   endDuty,   vehicle)  Vessel(   PK   vehicleId,   name,   call,   country,   FK owner,   imoNumber)  LegalEntity(   PK   entityId,   name) </pre>

В правом столбце таблицы приведены элементы целевой схемы, соответствующие исходным данным. Данные, соответствующие судну, как транспортному средству, сосредоточены в отношении *Vessel*; как спасательной единице – в отношении *SARUnit*; как юридической сущности - в отношении *LegalEntity*.

С использованием диалекта RIF-BLD необходимое правило интеграции описывается следующим образом:

```

forall ?Id ?name ?call ?beginDuty ?endDuty
  ?country ?ownerName ?imoNumber
( exists ?owner (
  AND( SARUnit(beginDuty->?beginDuty
    endDuty->?endDuty vehicle->?Id )
  Vessel(vehicleId->?Id
    name->External(normalize(?nazv))
    call->?call Country->?country
    owner->?owner imoNumber->?imoNumber ) )
  LegalEntity(entityId->?owner
    name->?ownerName) ) )
:-
AND( ?ERRTableShips#ERRTableShips
  ?ERRTableShips[Id->?Id Name->?name
  Callsing->?call Dates->?Dates]
  ?Dates#Dates
  ?Dates[StartDate->?beginDuty
    EndDate->?endDuty]
  Ship("Platforma_nazvanie"->?nazv
    "Pozyvnoj"->?call
    "Strana_naimenovanie"->?country
    "Organizaciya_nazvanie"->?ownerName
    "nomer_IMO"->?imoNumber)
  External(compareShipName(?name, ?nazv) ) )

```

Аналогично примеру, описанному в предыдущем подразделе, в голове правила конъюнкцией соединяются предикаты, соответствующие отношениям целевой схемы. Кроме того, конъюнкция в голове правила заключена под квантор существования (*Exists*) по переменной *?owner*<sup>1</sup>, значение которой (не определенное в исходных данных) является первичным ключом *entityId* в кортеже отношения *LegalEntity* и внешним ключом в кортеже отношения *Vessel*.

В теле правила конъюнкцией соединяется предикат *Ship*, соответствующий отношению из системы ЕСИМО, предикаты членства и фреймовые предикаты, соответствующие структуре XML-документов из системы «Поиск-Море».

Отдельной особенностью правила является то, что при соединении сущностей из исходных коллекций происходит проверка на соответствие имен судов с некоторой точностью (возможны варианты записей и ошибки) при помощи функции *compareShipName*.

Вышеописанное правило может быть реализовано в языке HIL следующей программой (объявления сущностей и функций опущены):

```
create link ShipLink as
select [
  Callsing_Name:
  [Callsing: es.Callsing, Name: es.Name],
  "Pozyvnoj_Platforma_nazvanie":
  ["Pozyvnoj": s."Pozyvnoj",
  "Platforma_nazvanie": s."Platforma_nazvanie"]]
from ERRTableShips es, Ship s
match using
  rule1: es.Callsing = s."Pozyvnoj" and
  compareShipName(es.Name,
  s."Platforma_nazvanie");

insert into SARUnit
select [ vehicle: s.Id,
  beginDuty: d.StartDate,
  endDuty: d.EndDate
]
from ShipLink sl, ERRTableShips s, s.Dates d
where sl.Callsing_Name.Name = s.Name and
  sl.Callsing_Name.Callsing = s.Callsing;

insert into Vessel
select [ vehicleId: s.Id,
  name: normalize(s."Platforma_nazvanie"),
  call: s."Pozyvnoj",
  country: s."Strana_naimenovanie",
  owner: get_id(s."Organizaciya_nazvanie"),
  imoNumber: s."nomer_IMO"]
from ShipLink sl, Ship s
where sl.Callsing_Name.Callsing = s."Pozyvnoj"
  and sl.Callsing_Name.Name =
  s."Platforma_nazvanie";

insert into LegalEntity
select [
  entityId: get_id(s."Organizaciya_nazvanie"),
  name: s."Organizaciya_nazvanie"]
from ShipLink sl, Ship s
where sl.Callsing_Name.Name =
```

<sup>1</sup> Строго говоря, квантор всеобщности в голове правила выходит за границы диалекта RIF-BLD, однако входит в каркас RIF-FLD.

```
s."Platforma_nazvanie" and
sl.Callsing_Name.Callsing = s."Pozyvnoj";
```

Соединение сущностей исходных коллекций производится с использованием оператора разрешения сущностей *create link*. В секции *from* оператора указываются соединяемые коллекции (*ERRTableShips*, *Ship*), в секции *select* – составные ключи (*Callsing\_Name*, *Pozyvnoj\_Platforma\_nazvanie*), однозначно идентифицирующие исходные сущности, правило сопоставления сущностей *rule1* (совпадение позывных и соответствие имен с точностью до функции *compareShipName*).

Как и в примере, описанном в предыдущем подразделе, для каждого отношения целевой схемы в программе определен оператор *insert*. Особенность данной программы состоит в том, что целевые отношения пополняются на основании только тех сущностей, которые связаны оператором *create link*.

Квантор всеобщности реализуется с использованием функции *get\_id*, порождающей уникальный идентификатор, и, тем самым, означающей подкванторную переменную *?owner*. Значение порожденного идентификатора присваивается первичному ключу *LegalEntity.entityId* и внешнему ключу *Vessel.owner*.

### 3 Спецификация и реализация правил интеграции данных документной модели

В данном разделе рассматривается пример правила интеграции данных, оперирующего сущностями документной модели. Исходная коллекция содержит сообщения о происшествиях в Арктической зоне из социальных сетей и сущности (персоны, суда, географические локации и т.д.), извлеченные средствами анализа текстов из сообщений [8]. Данные хранятся в документной СУБД MongoDB и экспортируются для дальнейшей интеграции в файлах в формате JSON.

В левом столбце таблицы 3 приведен пример данных о сообщении (*Messages*) и данных о сущностях, извлеченных из сообщения (*Entites*). Связь сообщений и сущностей, извлеченных из них, устанавливается на основании значения составного ключа *id*. В правом столбце таблицы приведены элементы целевой схемы, соответствующие исходным данным.

**Таблица 3** Данные о сообщениях и соответствующие элементы целевой схемы

Пример данных в исходной модели (документная)	Элементы целевой схемы (реляционная модель)
<pre>{ "Messages": [{   "id": { "coll_id": "8002",   "res_id": {   "site_id": "9b290c9f3bda",   "doc_id": "3649a5559a62" }},   "annotation": "Chinese   seismic vessel aimed for</pre>	<pre>Document(   documentId,   collection,   source,   content,   time,   language,</pre>

<pre>Russian #Barents Sea oil at logistics port #Kirkenes", "metafields": {   "mf203": "2016-4-30",   "mf205": "eng",   "mf200": "iceblogger"} }]}</pre> <pre>{ "Entities": [ {   "id": { "coll_id": "8002",   "res_id": {   "site_id": "9b290c9f3bda",   "doc_id": "3649a5559a62"}},   "entities": [   { "s_token": "Barents Sea",   "s_tag": "I-ALOC",   "s_end": 53,   "s_begin": 42},   { "s_token": "Kirkenes",   "s_tag": "I-ALOC",   "s_end": 85,   "s_begin": 77}   ] } ] }</pre>	<pre>author) ExtractedEntity(   entityId,   document,   token,   tag,   begin   end)</pre>
---	--

С использованием диалекта RIF-BLD правило интеграции данных о сообщениях описывается следующим образом:

```
forall ?m ?ext ?doc ?coll ?src ?cont
  ?time ?lang ?auth ?mid ?mres ?mf ?eid ?eres
( Exists ?ent( AND(
Document(documentId->?doc collection->?coll
source->?src content->?cont time->?time
language->?lang author->?auth)
ExtractedEntity(entityId->?ent document->?doc
token->?tok tag->?tag begin->?beg
end->?end) ))
:-
AND( ?m#Messages ?ext#Entities
?m[id->?mid annotation->?cont metafields->?mf]
?mid[coll_id->?coll res_id->?mres]
?mres[site_id->?src doc_id->?doc]
?mf[mf203->?time mf205->?lang mf200->?auth]
?ext[id->?eid entities->?ents]
?eid[coll_id->?coll res_id->?eres]
?eres[site_id->?src doc_id->?doc]
?ext#ents
?ext[s_token->?tok s_tag->?tag
s_begin->?beg s_end->?end] ) )
```

Аналогично примерам, приведенным в предыдущем разделе, в голове правила конъюнкцией соединяются предикаты, соответствующие отношениям целевой схемы. В теле правила при помощи предикатов членства и фреймовых предикатов отражена структура документов, соответствующих сообщениям и извлеченным сущностям. Правило может быть реализовано в языке НПЛ следующей программой:

```
insert into Document
select [ documentId: mres."doc_id",
collection: mid."coll_id",
source: mres."site_id",
content: m."annotation",
time: mf."mf203",
language: mf."mf205",
author: mf."mf200"]
from Message m, m."id" mid, mid."res_id" mres,
m."metafields" mf;

insert into ExtractedEntity
select [ entityId: get_id(strcat(
eres."site_id", eres."doc_id")),
document: eres."doc_id",
```

```
token: e."s_token",
tag: e."s_tag",
begin: e."s_begin",
end: e."s_end"]
from Entities ext, ext."id" eid,
eid."res_id" eres, ext.entities e;
```

Для обоих отношений целевой схемы в программе определен оператор *insert*.

#### 4 Спецификация и реализация правил интеграции данных графовой модели

В данном разделе рассматривается пример правила интеграции данных, оперирующего сущностями графовой модели. Исходная коллекция содержит данные о спасательных операциях, данные хранятся в графовой СУБД Neo4j и экспортируются для дальнейшей интеграции в файлах в формате JSON.

В левом столбце таблицы 4 приведен пример данных о спасательных операциях. Данные содержат три вида вершин (*Nodes*): спасательная операция (*SarOperation*), координационный центр (*CoordCenter*), координатор операции (*Coordinator*); и два вида ребер (*Relationships*): *rCoordCenter* (ребра, связывающие операцию и центр), *rCoordinator* (ребра, связывающие операцию и ее координатора).

Таблица 4 Данные о спасательных операциях и соответствующие элементы целевой схемы

Пример данных в исходной модели (графовая)	Элементы целевой схемы (реляционная модель)
<pre>{ "Nodes": [ { "id": "12", "labels": [ "SarOperation" ], "properties": { "OperationID": "5824", "OperationDate": "2016-09-15T00:00:00", "OperationName": "Arctic Sunrise Sinking", "countrycode": "643" } }, { "id": "13", "labels": [ "CoordCenter" ], "properties": { "CoordCenterID": "mrcc667340", "Name": "MRCC Dikson" } }, { "id": "14", "labels": [ "Coordinator" ], "properties": { "Name": "Schurov V. A.", "CoordinatorID": "5773" } } ] }</pre>	<pre>SAROperation( PK operationId, FK coordCenter, FK coordinator, country, creationTime, name)  CoordinationCenter( PK centerId, name)  Person( PK personId, name)</pre>
<pre>{ "Relationships": [ { "id": "4", "type": "rCoordCenter", "startNode": "12", "endNode": "13" }, { "id": "9", "type": "rCoordinator", "startNode": "12", "endNode": "14" } ] }</pre>	

В правом столбце таблицы приведены элементы целевой схемы, соответствующие исходным данным.

С использованием диалекта RIF-BLD интеграция данных о спасательных операциях описывается совокупностью следующих правил:

```
Forall ?pid ?name ?lbl ?prp(
Person(personId->?pid name->?name) :-
AND( ?n#Nodes
?n[labels->?lbl properties->?prp]
"Coordinator"##?lbl
?prp[CoordinatorID->pid Name->?name]))

Forall ?cid ?name ?lbl ?prp(
CoordinationCenter(centerId->?cid
name->?name) :-
AND( ?n#Nodes
?n[labels->?lbl properties->?prp]
"CoordCenter"##?lbl
?prp[CoordCenterID->cid Name->?name]))

Forall (
SAROperation(operationId->?opid
creationTime->?time name->?name
country->?country coordCenter->?cntr
coordinator->?coord)
:-
AND( ?n#Nodes ?n[id->?id]
?n[properties->?prp labels->?lbl]
?r1#Relationships ?r2#Relationships
?n1#Nodes ?n1[id->?id1]
?n2#Nodes ?n2[id->?id2]
"SarOperation"##?lbl
?r1[type->"rCoordCenter"
startNode->?id endNode->?id1]
?r2[type->"rCoordinator"
startNode->?id endNode->?id2]
))
```

Каждому отношению целевой схемы соответствует свой тип вершин, например, отношению *SAROperation* соответствуют вершины с меткой "SarOperation" (атрибут *labels*). Поэтому для каждого отношения удобно определить свое порождающее правило. Предикаты, соответствующие отношениям целевой схемы, составляют головы правил. В телах правил при помощи предикатов членства и фреймовых предикатов отражены структуры данных, соответствующие вершинам и ребрам графа.

Правила могут быть реализованы в языке NIL следующей программой:

```
insert into Person
select [
personId: prp."CoordinatorID",
name: prp."Name"]
from Nodes n, n."properties" prp,
n."labels" lbl
where
array_includes(lbl, "Coordinator");

insert into CoordinationCenter
select [
centerId: prp."CoordCenterID",
name: prp."Name"]
from Nodes n, n."properties" prp
n."labels" lbl
where
array_includes(lbl, "CoordCenter");

insert into SAROperation
select [
operationId: prp."OperationID",
```

```
creationTime: prp."OperationDate",
name: prp."OperationName",
country: prp."countrycode",
coordCenter: prp1."CoordCenterID",
coordinator: prp2."CoordinatorID"]
from Nodes n, n."properties" prp,
n."labels" lbl,
Relationships r1, Relationships r2,
Nodes n1, Nodes n2
where
array_includes(lbl, "SarOperation") and
n."id" = r1."startNode" and
r1."type" = "rCoordCenter" and
n1."id" = r1."endNode" and
n."id" = r2."startNode" and
r2."type" = "rCoordinator" and
n2."id" = r2."endNode";
```

Каждое из правил представляется отдельным оператором *insert*. Предикаты проверки типа вершины (например, "Coordinator"##?lbl) реализуются с использованием функции *array\_includes*:

```
@jaql{ array_includes = fn(a, e)
if (not exists(a->filter $ == e)) false
else true; }
```

## 5 Общие принципы спецификации правил интеграции данных с использованием RIF-BLD и их реализации в языке NIL

Основные принципы спецификации правил интеграции данных с использованием RIF-BLD, примененные в данной работе, можно обобщить следующим образом:

- правила интеграции имеют вид импликации, посылка которой называется *телом*, а заключение – *головой*;
- голова и тело правил представляют собой формулы, связывающие предикаты операциями конъюнкции или дизъюнкции;
- в теле правила допускаются предикаты, соответствующие сущностям только исходных схем, в голове – только целевой схеме;
- для описания свойств кортежей, принадлежащих отношениям реляционной модели, в правилах используются предикаты с именованными аргументами [5];
- для описания свойств структур данных произвольной степени вложенности (при помощи которых представляются данные различных нереляционных моделей – XML, документной, графовой и т.д.) используются фреймовые предикаты и предикаты членства [5];
- связь значений атрибутов сущностей исходных и целевой схем осуществляется при помощи переменных;
- свободные переменные, используемые в теле правила, связываются внешним квантором всеобщности;
- свободные переменные в голове правила, не встречающиеся в теле (фактически, они соответствуют данным, не определенным явно в

исходных коллекциях), связываются квантором существования;

- нетривиальные предикаты-условия (отличные от равенства) и функции разрешения конфликтов определяются как внешние термы [5].

Основные принципы реализации правил интеграции данных на RIF-BLD в языке HIL, примененные в данной работе, можно обобщить следующим образом:

- сущности исходных и целевой схем, функции разрешения конфликтов и нетривиальные предикаты-условия объявляются при помощи директивы *declare*; для функций определяется их сигнатура;
- функции реализуются на языке Jaql в отдельных секциях программы, либо на языке Java во внешних файлах;
- для каждой сущности (отношения) в голове правила создается отдельный оператор *insert*, порождающий кортежи этих отношений:
  - в секции *from* объявляются исходные сущности, каждому предикату с именованными переменными или предикату членства в теле правила соответствует объявление с точностью до имени переменной;
  - в секции *select* указываются атрибуты целевых отношений (в соответствии с их названиями в предикатах головы правила) и выражения порождения их значений; выражения формируются в соответствии с термами в предикатах головы правила либо с термами во фреймовых предикатах тела правила;
  - в секции *where* указываются предикаты, отражающие способы соединения исходных сущностей (формируемые на основании совпадения имен переменных во фреймовых предикатах тела) и их отбора (соответствующие предикатам, налагающим условия на данные отдельных исходных коллекций в теле правила);
- если соединение сущностей исходных коллекций в теле правила осуществляется с использованием нетривиальных предикатов и функций (отличных от равенства атрибутов), предварительное соединение сущностей производится с использованием оператора разрешения сущностей *create link*:
  - в секции *from* указываются соединяемые коллекции;
  - в секции *select* указываются составные ключи, однозначно идентифицирующие исходные сущности;
  - в секции *match* указываются правила сопоставления сущностей;
  - коллекция пар сопоставленных сущностей затем используется в качестве входной в операторах *insert*, порождающих кортежи

целевых отношений.

## 6 Заключение

В работе рассматривается подход к спецификации правил интеграции данных с использованием рекомендации W3C - логического диалекта RIF-BLD. Широкий спектр возможностей спецификации RIF-BLD позволяет использовать в одном правиле сущности из разных коллекций, представленных в разных моделях данных. Рассматриваются правила, в голове которых могут присутствовать лишь предикаты, соответствующие сущностям целевой схемы, а в теле – предикаты, соответствующие сущностям исходных схем. Логическая семантика RIF-BLD позволяет однозначным образом интерпретировать спецификации рассматриваемых правил интеграции и допускает их реализацию с использованием различных языков. В данной работе рассматривается подход к реализации логических правил RIF-BLD в языке высокого уровня HIL, разработанного компанией IBM. Путем компиляции программ на HIL в программы вычислительной модели MapReduce достигается распределенное исполнение правил интеграции в среде Hadoop.

К дальнейшим направлениям работы можно отнести вопросы интерпретации и реализации логических программ на RIF-BLD, в телах правил которых допускаются предикаты целевой схемы. Это требует адаптации, в частности, процедуры погони [9]. Другим направлением является реализация автоматического преобразования спецификаций RIF-BLD в программы, компилируемые и исполняемые на распределенных вычислительных инфраструктурах (например, на языке HIL).

**Поддержка.** Работа выполнена при поддержке Министерства образования и науки Российской Федерации (уникальный идентификатор проекта RFMEFI60717X0176).

## Литература

- [1] Apache Hadoop Project. 2016. <http://hadoop.apache.org/>
- [2] Ballard, C., Alon, T., Dronavalli, N., Jennings, S., Lee, M., Toratani, S. IBM InfoSphere Information Server Deployment Architectures. ibm.com/redbooks (2012)
- [3] Bar-Or, A., Choudhary, S. Transform XML using the DataStage XML stage. IBM developerWorks (2011)
- [4] 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. In: 37th International conference on very large data bases VLDB, pp. 1272-1283. Curran Associates, New York (2011)

- [5] Boley, H., Kifer, M. (eds.): RIF Framework for Logic Dialects. W3C Recommendation, 2nd edn. (February 5, 2013)
- [6] Boley, H., Kifer, M. (eds.): RIF Basic Logic Dialect. W3C Recommendation, 2<sup>nd</sup> edn. (February 5, 2013)
- [7] Burdick, D., Hernández, M. A., Ho, H., Koutrika, G., Krishnamurthy, R., Popa, L., Stanoi, I.R., Vaithyanathan, S., Das, S.: Extracting, Linking and Integrating Data from Public Sources: A Financial Case Study. *IEEE Data Eng. Bull.*, 34(3):60–67 (2011)
- [8] Devyatkin, D., Shelmanov, A. Text Processing Framework for Emergency Event Detection in the Arctic Zone. In: Kalinichenko L., Kuznetsov S., Manolopoulos Y. (eds) *Data Analytics and Management in Data Intensive Domains. DAMDID/RCDL 2016. Communications in Computer and Information Science*, vol 706, pp. 74–88. Springer (2017)
- [9] Fagin, R., Kolaitis, P., Miller, R., Popa, L.: Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1):89–124 (2005)
- [10] Hernandez, M., Koutrika, G., Krishnamurthy, R., Popa, L., Wisnesky, R.: HIL: A high-level scripting language for entity integration. In: 16th Conference (International) on Extending Database Technology Proceedings EDBT 2013, pp. 549–560 (2013)
- [11] IBM InfoSphere BigInsights Version 3.0 Information Center. <https://goo.gl/lZpEQd>
- [12] InfoSphere Big Match for Hadoop. Technical Overview. - <https://goo.gl/OTMqvw>
- [13] Introducing JSON. - <http://www.json.org/>
- [14] Miner, D. *MapReduce Design Patterns: Building Effective Algorithms and Analytics for Hadoop and Other Systems*. O'Reilly Media (2012)
- [15] The Apache Hive data warehouse software. - <http://hive.apache.org/>
- [16] Брюхов, Д. О., Скворцов, Н. А., Ступников, С. А. Методы интеграции разнотипных данных по Арктической зоне для извлечения информации, нацеленной на поддержку поисково-спасательных операций. *Системы высокой доступности*. Т. 13, № 2. М.: Радиотехника (2017) Принято к публикации.
- [17] Единая государственная система информации об обстановке в мировом океане. - <http://portal.esimo.ru/portal>
- [18] Комплексная интегрированная информационная система «МоРе». - <http://www.marsat.ru/ciis-more>
- [19] Программный комплекс «Поиск-Море». - <http://map.geopallada.ru/>
- [20] Скворцов, Н. А., Брюхов, Д. О. Разработка схемы хранилища информации для поддержки поисковых действий в Арктической зоне. *Системы высокой доступности*. Т. 13, № 2. М.: Радиотехника (2017). Принято к публикации.