Методы и средства доступа к потоковым данным из предметных посредников ♣

© А.Е. Вовченко, Л.А. Калиниченко, М.Ю. Костюков
ИПИ РАН
itsnein@gmail.com, leonidk@synth.ipi.ac.ru, klarell@gmail.com

Аннотация

В последнее время получил широкое распространение новый класс задач, требующих обработки данных большого объема, представленных в виде потоков. Все чаще появляются ресурсы потоковых данных (РПД), предоставляющие большие потоки данных в реальном времени. В инфраструктуре предметных посредников важную роль играют адаптеры, реализующие унифицированный интерфейс доступа посредника к разнородным информационным ресурсам.

В статье дано описание особенностей и семантики модели потоковых данных, представлены подход к реализации различных видов запросов в адаптере потоковых данных (АПД), а также архитектура адаптера, после чего описан пример решения задач в разработанной архитектуре.

1 Введение

Настоящий период развития науки характеризуется взрывоподобным процессом накопления информационных ресурсов и сервисов обработки информации, число которых экспоненциально растет. При этом данные в ресурсах могут быть представлены в различных моделях данных, таких, как модель данных XML, реляционная модель данных, объектно-реляционная модель данных, онтологическая модель данных и другие. Для решения научных задач зачастую требуется доступ к неоднородным информационным ресурсам, что требует создания особой информационной инфраструктуры.

В основе такой инфраструктуры лежит идея предметных посредников [1]. Схема посредника определяется приложением и не зависит от схемы ресурсов. Релевантные посреднику ресурсы регистрируются как взгляды над виртуальными классами схемы посредника. При решении задач запросы посредника формулируются в терминах его схемы. Посредник переписывает запрос в терминах локальных ресурсов. Ресурсы возвращают объединенный результат посреднику для выполнения дальнейших операций в ходе решения задачи. Ресурсы регистрируются в посредниках независимо друг от друга.

В то же время, все чаще появляются ресурсы потоковых данных (РПД), предоставляющие большие потоки данных в реальном времени. Вместе с ресурсами появляются и задачи, требующие как обработки потоковых данных, так и интегрированного доступа к подобным ресурсам.

В последнее время получил широкое распространение новый класс задач, в которых данные представляются в виде потоков данных. Примеры подобных задач включают финансовые приложения (Traderbot [2]), сетевой мониторинг (мониторинг click streams в Google, Yahoo и.т. д.), задачи безопасности (iPolicy Networks [3]), управление телекоммуникационными данными, задачи в науке, сетевые приложения и многие другие.

В настоящее время существует несколько технологий, включая готовые к использованию процессоры обработки потоков данных в реальном времени. В то же время традиционные технологии, такие, как СУБД, поддерживающие базы данных в основной памяти (main memory DBMS), и процессоры правил (rule engine), могут быть также "перенацелены" на решение вышеуказанных задач. Тем не менее, ни одно из готовых решений не предоставляет возможности к интеграции потоковых данных, что немаловажно для решения сложных задач, когда данные поступают не из одного источника.

Модель потоковых данных отличается от привычной реляционной модели данных. В модели потоковых данных все данные или некоторая их часть недоступны для извлечения из памяти, а поступают как один или несколько непрерывных потоков данных. Отличия от реляционной модели заключаются в следующем:

- элементы в потоке данных появляются в режиме реального времени;
- система не контролирует порядок, в котором элементы данных поступают, как внутри одного потока, так и между потоками;
- потоки данных потенциально не ограничены в размере;
- как только элемент из потока данных обработан, он отбрасывается и более не учитывается или же архивируется. Храниться в памяти такие данные не могут ввиду ограниченного объема памяти.

Запросы в модели потоковых данных можно разделить на 4 вида: одиночные, одиночные с вре-

Труды 12ой Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» – RCDL'2010, Казань, Россия, 2010

менными ограничениями, непрерывные [4] не материализованные, непрерывные материализованные.

В инфраструктуре предметных посредников важную роль играют адаптеры [5, 6], реализующие унифицированный интерфейс доступа посредника к разнородным информационным ресурсам. Целью настоящей работы является разработка архитектуры адаптера потоковых данных (АПД), обеспечивающего возможность использования РПД в архитектуре предметных посредников интегрировано с другими информационными ресурсами.

В следующем разделе представлено описание особенностей и семантики запросов к потоковым данным. Затем представлен подход к реализации различных видов запросов в АПД, после чего описывается архитектура разработанного адаптера потоковых данных. Наконец, в завершение представлен пример решения задач с использованием адаптера потоковых данных.

2 Особенности и семантика запросов к ресурсам потоковых данных

Определение 1. Поток S представляет собой мультимножество пар <s, t>, где s – это кортеж, соответствующий схеме S, а t – это время элемента.

Определение 2. Временным слепком потока (Time Stream Snapshot) TSS(S, t1) потока S является мультимножество пар <s, t> потока S, для которых t = t1.

Замечание. Мультимножество может быть однозначно отображено во множество путем приписывания каждому элементу еще одного атрибута, его кратности (multiplicity) в исходном мультимножестве.

Определение 3. Классом C называется множество объектов заданного абстрактного типа данных (АТД).

Определение 4. Отображение *потока-в-класс* представляет собой операцию, принимающую на вход поток S и возвращающую класс C. При этом тип объектов класса C включает все атрибуты, описанные в схеме S, а также время t и кратность элемента.

Таким образом, каждой паре <s, t> из потока S ставится в соответствие объект, содержащий данные {s(1), s(2), … , s(n), t, multiplicity}, где s(i) – i-й атрибут кортежа s.

При этом в каждый момент времени t1 должен быть вычисляем TSS(S, t1) по потоку S и времени t1. Кортежи из мультимножества TSS(S, t1) добавляются в качестве объектов в класс.

Определение 5. Временным окном (Time Window) TW(S,t1,t2) называется мультимножество пар <s, t> потока S, для которых $t \geq t1$ и $t \leq t2$.

Определение 6. Временным окном по умолчанию (Default Time Window) DTW(S) называется временное окно TW(S, now-time, now), где now – текущее время, a time – это максимальный срок хранения устаревших данных, задаваемый администратором системы.

Замечание. Окна определяются для любого мультимножества пар вида <s, t>, как ограниченного, так и не ограниченного. Поток относится к неограниченному мультимножеству подобного вида. Важно, что и само окно относится к ограниченному мультимножеству подобного вида, таким образом, допустимо определение окна над окном, например: TW(DTW(S), t1,t2)

Одиночные запросы – это запросы, аналогичные запросам в традиционных СУБД. При этом поток в общем случае непрерывен, и объем результата ограничивается системой управления потоковыми данными. Для ограничения используется механизм окон. При этом используются как временные окна по умолчанию, так и окна с количественным ограничением.

Одиночные запросы с временными ограничениями – это запросы, где явно указано ограничение для потока по времени. Фактически в этом запросе явно указываются границы для временного окна.

Пример (пусть текущая дата: 5.05.2005)

```
Select * from streamTable where time > 10.12.2000
and time < 10.12.2009
```

Непрерывные запросы – это запросы, которые выполняются непрерывно во времени. При этом в каждый момент времени t1 предполагается, что все события, относящиеся ко времени t < t1, уже обработаны, и непрерывный запрос порождает результат, относящийся только к моменту времени t1.

Непрерывные запросы бывают не материализованными и материализованными. Результатом нематериализованных непрерывных запросов является новый поток. Результатом материализованных непрерывных запросов является временное окно TW. Таким образом, гарантируется, что в любой момент времени $t' \geq t1$ в результате (потоке или временном окне) не могут оказаться элементы, соответствующие времени t < t1.

Непрерывные не материализованные запросы – это запросы, которые определяются один раз, и к результату их выполнения можно обратиться в любой момент. Аналогом этих запросов являются взгляды (views) в СУБД. Этот тип запросов наиболее распространен в системах потоковых данных, т. к. определение подобного запроса есть определение нового потока данных.

Определение 8. Пусть дан поток S, содержащий множество пар <s, t>. Тогда результатом непрерывного нематериализованного запроса CQ является поток CQS, представляющий собой мультимножество пар <s1, t>. При этом в любой момент времени временное окно по умолчанию DTW(CQS) представляет собой результат выполнения одиночного запроса CQ над потоком S. Если учесть семантику одиночных запросов, то он, фактически, представляет собой результат выполнения одиночного запроса CQ над временным окном по умолчанию DTW(S).

Непрерывные материализованные запросы отличаются от не материализованных тем, что накапливают данные, соответствующие запросу. Немате-

риализованный запрос – это, по сути, новый поток, и на него действуют те же ограничения системы, что и на обычные потоки. Если используется механизм окон, то результат лимитируется временными рамками, если ограничивается объем результата, то ограничивается объем. Таким образом, данные в нематериализованных запросах могут устареть или же не быть включенными в результат из-за ограничений объема.

Напротив, в материализованных непрерывных запросах данные устареть не могут. По сути, данный вид запроса – это указание системе накапливать поступающий поток данных в некотором буфере (таблице) с возможностью в любой момент считать данные из таблицы. Таким образом, в общем случае данные могли бы накапливаться бесконечно.

Определение 10. Пусть дан поток S, содержащий мультимножество пар <s, t>. Пусть в момент времени t1 задан непрерывный материализованный запрос CQ. Тогда результатом непрерывного материализованного запроса CQ является мультимножество CQM, представляющее собой мультимножество пар <s1, t>. CQM представляет собой временное окно TW(S, t1 – time, now), где now – текущее время, time – это максимальный срок хранения устаревших данных, задаваемый администратором системы для временных окон по умолчанию.

3 Подход к реализации разных видов запросов к потоковым данным в адаптере потоковых данных (АПД)

Одиночные запросы

В посредниках в АПД используется механизм окон. Адаптер позволяет задавать одиночные запросы к данным за некоторый временной промежуток, иными словами, запрос задается к временному окну по умолчанию над всем потоком. Размер временного окна задается администратором при инициализации адаптера. Таким образом, объем данных всегда ограничен, что делает возможным реализацию данного типа запросов.

Одиночные запросы с временными ограничениями

При регистрации АПД схема РПД дополняется временным атрибутом, который хранит время, когда данные были получены. Таким образом, поддерживается возможность выполнения одиночных запросов с временными ограничениями.

Непрерывные не материализованные запросы

В посредниках предполагается подобный вид запросов реализовать как взгляды при регистрации.

Рассмотрим абстрактный пример (пример реальной задачи представлен в разделе 5):

Пусть определены в ресурсе классы C1 (a1,b1,c1), C2 (a2, b2, c2), C3 (a3, b3, c3);

В посреднике определены классы MC1(a1,b1,c1,a2), MC2(b2,c2,a3,b3,c3);

Пусть нам нужно реализовать непрерывный нематериализованный запрос вида

```
Select c1,c2,c3 from C1, C2, C3 where C1.a1 = C2.a2
and C1.a1 = C3.a3.
```

При этом подходе в АПД помимо таблиц, хранящих данные из потоков C1, C2, C3, создается таблица, представляющая собой взгляд (view) над таблицами C1, C2, C3. Создание выглядит так:

```
Create view CQ1 as
Select c1,c2,c3 from C1, C2, C3 where C1.a1 = C2.a2
and C1.a1 = C3.a3
```

И тогда в посреднике по потребности могут регистрироваться любые из классов ресурса C1,C2,C3,CQ1 как взгляды, выраженные через MC1, MC2

Непрерывные материализованные запросы

В посредниках предполагается использовать предопределенные непрерывные материализованные запросы. Посредники виртуальны. И в посредниках нет такого понятия, как материализованные взгляды, именно поэтому этот тип запросов в виде взглядов выразить нельзя. Поэтому данные предполагается накапливать в базе адаптера.

В СУБД нередко для похожих целей используются материализованные взгляды. Данные в них представляют слепок (snapshot) с текущего взгляда исходной таблицы. Если исходная таблица обновляется, то обновляется и материализованный взгляд. Невозможность использования материализованных взглядов заключается в том, что при обновлении данные как добавляются в материализованный взгляд, так и удаляются, в случае, если эти же данные были удалены из исходной таблицы. В случае же материализованных непрерывных запросов данные нужно накапливать, т. е. те данные, что уже удалены из исходной таблицы, должны сохраняться в таблице-результате материализованного непрерывного запроса.

При этом важным свойством материализованных взглядов является то, что для обновления взглядов не требуется заново перестраивать всю таблицу, а изменения касаются лишь данных, которые изменились в исходной таблице. Этим же свойством и обладает результат непрерывных материализованных запросов.

Рассмотрим пример. Пусть определены в ресурсе классы C1 (a1,b1,c1), C2 (a2, b2, c2) C3 (a3, b3, c3). В посреднике определены классы MC1(a1,b1,c1,a2), MC2(b2,c2,a3,b3,c3).

Пусть требуется реализовать непрерывный материализованный запрос вида

```
Select c1,c2,c3 from C1, C2, C3 where C1.a1 = C2.a2
and C1.a1 = C3.a3
```

В ресурсе создается таблица MCQ1 (Materialized Continuous Query). Запрос для создания выглядит следующим образом:

```
Create table MCQ1 as
Select c1,c2,c3 from C1, C2, C3 where C1.a1 = C2.a2
and C1.a1 = C3.a3
```

Кроме того, запрос

```
Select c1,c2,c3 from C1, C2, C3 where C1.a1 = C2.a2
and C1.a1 = C3.a3
```

регистрируется в АПД для того, чтобы АПД, обновляя данные новой порцией, обновлял и таблицу MCQ1. Далее классы C1,C2,C3,MCQ1 могут быть зарегистрированы в посреднике.

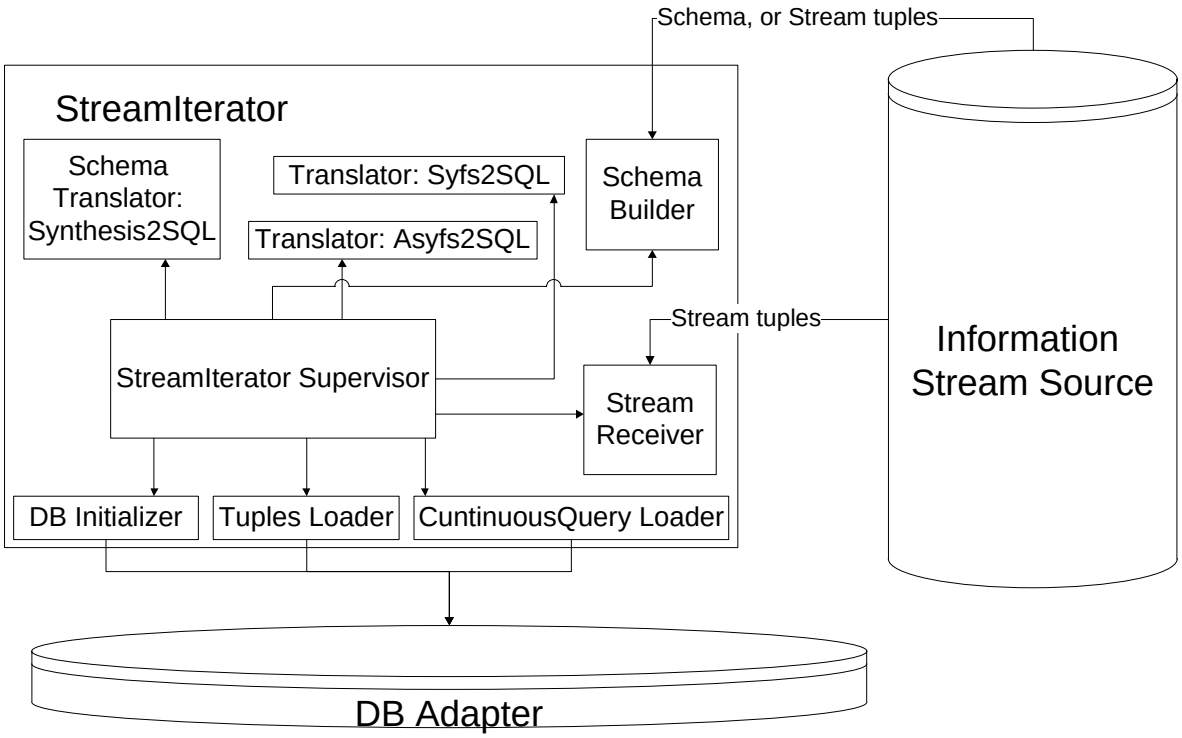

Рис. 1. Архитектура StreamIterator-а

4 Архитектура адаптера потоковых данных

Архитектура адаптера в среде посредников для реляционных СУБД остается неизменной, за тем исключением, что в качестве ресурса выступает не совокупность таблиц, а СУБД промежуточных данных DB Adapter. С самим ресурсом потоковых данных связь осуществляется компонентом StreamIterator, который осуществляет поддержку работоспособности адаптера потоковых данных.

На рис. 1 изображена архитектура компонента АПД StreamIterator. Данный компонент не может быть универсальным для всех РПД. Вместе с тем, лишь малая часть его функциональности зависит от конкретных РПД.

Компоненты SchemaTranslator, Syfs2SQL, Asyfs2SQL, StreamIterator Supervisor, DB Initializer, Tuples Loader, ContinuousQuery Loader являются общими для всех StreamIterator-ов. Только компоненты Schema Builder и Stream Receiver зависят от конкретного интерфейса РПД, и именно их необходимо разрабатывать под каждый отдельный РПД. Важно отметить, что необходимости в трансляторе запросов из языка посредника в язык ресурса РПД нет, так как поток всегда помещается а базу адаптера полностью в исходном виде, без каких либо изменений. Именно за это и отвечает компонент Stream Receiver. Если РПД поддерживают SQL-подобный язык запросов, то этим компонентом выполняется запрос вида Select * from. Запросы же от посредника транслируются реляционным адаптером и выполняются над базой адаптера, в которую заранее помещается исходный поток.

Компонент StreamIterator Supervisor связывает все компоненты воедино и определяет алгоритм поведения компонента StreamIterator.

Транслятор схем необходим для преобразования схемы на языке СИНТЕЗ в схему на языке SQL. Данная операция необходима для инициализации юазы данных адаптера, а также в случае, если администратору нужно получить схему в виде спецификации на SQL. Подобное может понадобиться в случае, если непрерывные запросы определяются на языке SQL, как на языке СУБД. В случае, если запросы задаются на языке запросов Syfs или Asyfs посредника, они в начале преобразуются в SQL трансляторами Asyfs2SQL или Syfs2SQL, после чего отправляются на выполнение в СУБД.

Компонент DB Initializer отвечает за инициализацию БД адаптера. В частности, в БД создаются таблицы в соответствии со схемой, полученной от РПД.

Компонент Tuples Loader выполняет три задачи: загрузку новых объектов, получаемых из потока, в БД адаптера, удаление из БД адаптера устаревших объектов, обновление непрерывных материализованных запросов всякий раз, когда обновляются данные в БД адаптера.

Компонент ContinuousQuery Loader отвечает за выполнение непрерывных запросов. Если это не материализованный запрос, то просто создается

взгляд в БД адаптера; если запрос материализованный, то создается таблица, а также запрос запоминается и в будущем используется компонентом Tuples Loader при обновлении данных.

Компонент Schema Builder конструирует схему на языке СИНТЕЗ, описывающую данные в потоке РПД. Его реализация зависит от конкретного РПД.

Компонент Stream Receiver позволяет получать данные из РПД. Функциональность компонента позволяет получить текущий кортеж из РПД. Этой функциональностью пользуется компонент StreamIterator Supervisor, который, получая данные из РПД, загружает их в базу.

5 Пример решения научной задачи

Задача определения вторичных стандартов для фотометрической калибровки оптических компонентов космических гамма-всплесков сформулирована ИКИ РАН. Задача определения стандартов рассматривается как одна из задач, решаемых в рамках виртуальной обсерватории (ВО). Задача заключается в том, что по координатам гамма-всплеска, необходимо отобрать ряд стандартных звезд (звезд с хорошо изученными параметрами) на основании различных параметров. Задача возникает в связи с тем, что после гамма-всплеска многие обсерватории начинают наблюдать ту область неба, где произошел гамма-всплеск. Почти сразу же начинают поступать большие объемы сырых наблюдаемых данных, которые требуют калибровки. Для калибровки используются стандартные звезды.

Данные о гамма-всплесках поступают непрерывно по электронной почте. Пример сообщения:

```
TITLE: GCN/SWIFT NOTICE
NOTICE_DATE: Wed 28 Jul 10 10:48:41 UT
NOTICE_TYPE: Swift-XRT Nack-Position
TRIGGER_NUM: 430172, Seg_Num: 0
POINT_RA: 44.040d {+02h 56m 10s} (J2000)
POINT_DEC: +0.269d {+00d 16' 08"} (J2000)
IMG_START_DATE: 15405 TJD; 209 DOY; 10/07/28
IMG_START_TIME: 38012.85 SOD {10:33:32.85}
UT, 97.0 [sec] since BAT Trigger Time
COUNTS: 17 Min_needed= 20
STD_DEV: 0.00 Max_StdDev_for_Good=28.44
[arcsec]
PH2_ITER: 1 Max_iter_allowed= 4
ERROR_CODE: 1
COMMENTS: SWIFT-XRT Nack Position.
COMMENTS: No source found in the image.
```

Количество подобных сообщений составляет 3 – 5 сообщений в день. Для обработки сообщений о гамма-всплесках был использован адаптер потоковых данных. В качестве времени хранения данных выбран один месяц, т. к. для более ранних гамма-всплесков наверняка стандарты уже найдены и сырые данные откалиброваны. В качестве схемы использовались все данные из письма:

```
{GRB_ALERT; in: type;
 TITLE: string;
 NOTICE_DATE: string;
 NOTICE_TYPE: string;
 TRIGGER_NUM: string;
 POINT_RA: string;
 POINT_DEC: string;
 IMG_START_DATE: string;
 IMG_START_TIME: string;
 COUNTS: string;
 STD_DEV: string
 PH2_ITER: string
 ERROR_CODE: string;
}
```

Наиболее важными параметрами для задачи является название Title, а также координаты Point_RA, Point_DEC. Для этих трех параметров был создан непрерывный нематериализованный взгляд:

```
Create view GRB_ALERT_CQ as
Select TITLE, POINT_RA, POINT_DEC
from GRB_ALERT
```

В посреднике описывался тип GRB_ALERT_CQ, и соответствующий ему класс grb_ALERT_CQ.

```
{GRB_ALERT_CQ; in: type;
 name: string;
 ra: real;
 de: real;
}
{grb_ALERT_CQ; in: class;
 instance_section: GRB_ALERT_CQ;
}
```

Координаты в типе посредника в отличие от ресурса представлены не строковыми типами, для их преобразования на этапе регистрации использовались функции разрешения конфликтов.

Также в посреднике описаны две функции решения задачи получения стандартов по координатам и радиусу:

```
{chooseStandards; in: function;
 params: {+ra/real, +de/real, +radius/real, -
standards/{set; type_of_element: Standard;}};
 },
{showStandards; in: function;
 params: {+ra/real, +de/real, +radius/real,
+standards/{set; type_of_element: Standard;}, -
image/Image};
 };
```

Функция chooseStandards описывает собственно задачу поиска стандартов. В качестве входных данных функция принимает координаты центра площадки и радиус. Функция возвращает множество звезд, удовлетворяющих всем требованиям для выбора в качестве стандартов.

Функция showStandards нужна для того, чтобы пользователь мог представить потенциальные стандарты на изображении (с помощью Aladin), тем самым визуально представляя результат.

Процесс решения задачи сводится к тому, что для каждого объекта из класса grb_ALERT_CQ (фактически для каждого гамма-всплеска) по его координатам, а также по некоторому радиусу, выбираемому пользователем, выполняются две функции: выбора стандартов (choseStandards) и отображения их на изображении (showStandards). При решении задачи выполняется следующий параметри-

зованный запрос к посреднику (параметр radius – задается пользователем, в задаче использовался 0.1deg):

```
r(x/[title, standards, image])
:- grb_ALERT_CQ (x1/[title: name, ra, de])
& choseStandards(ra, de, 0.1, standards)
& showStandards(ra, de, 0.1, standards, image)
```

Функция choseStandards представляет собой поток работ, включающий в себя обращение к двум посредникам. Схематично поток работ включает в себя следующие шаги.

Шаг 1. На первом шаге среди всех астрономических объектов мы выбираем те, что попадают в указанную площадку (по координатам и радиусу, задаваемым параметрами функции). При этом нас интересуют только координаты, магнитуды, тип объекта, собственное движение и качество данных.

Шаг 2. На втором шаге происходит кросс идентификация объектов из разных каталогов, объединяющая магнитуды объектов. По сути это означает, что если среди данных есть два объекта, для которых будет установлено, что они идентичны (по близости координат), то в результате мы получим только один объект, содержащей магнитуды обоих. Это необходимо, чтобы объединить данные магнитуд из разных ресурсов.

Шаг 3. На третьем шаге мы отсеиваем те объекты, которые не являются изолированными. Изолированность означает отсутствие вблизи других объектов.

Шаг 4. На четвертом шаге мы выбираем только те объекты, для которых было проверено, что они не являются переменными (проверка переменности – это отдельный запрос к посреднику), что они являются звездами и для них в эталонном каталоге (каталогах) не значится, что их тип – Галактика. Ко всему этому выбираются лишь те объекты, для которых собственное движение очень мало. Также выбираются лишь те объекты, для которых данные качественны.

6 Заключение

В работе предложена архитектура АПД, основанная на ранее разработанном реляционном адаптере в среде посредников. Разработана архитектура программного средства StreamIterator, обеспечивающего в АПД взаимодействие с РПД. Составные части StreamIterator разделены на компоненты, общие для всех StreamIterator-ов и уникальные для каждого конкретного РПД. Все общие компоненты были реализованы. Все компоненты, уникальные для конкретного РПД, реализованы для конкретного потокового РПД.

Предлагаемый подход разрабатывался в среде предметных посредников, осуществляющих решение научных задач над неоднородными информационными ресурсами. Для доступа к неоднородным ресурсам используется механизм адаптеров с фиксированным интерфейсом. Именно поэтому основной целью работы была разработка возможности использования потоковых данных из предметных посредников, посредством стандартного интерфейса (адаптера). Данная задача была выполнена полностью. Эффект от предложенного подхода становится понятным при решении комплексных задач (пример описан в разделе 5), требующих доступа к множеству неоднородных распределенных информационных ресурсов, среди которых могут быть и ресурсы потоковых данных.

Литература

[1] 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. – 2008. – Т. 2, Вып. 1. – С. 2-34.

[2] Traderbot home page. – http://www.traderbot.com.

[3] iPolicy Networks home page. – http://www. ipolicynetworks.com.

[4] Terry D., Goldberg D., Nichols D., Oki B. Continuous queries over append-only databases//Proc. of the 1992 ACM SIGMOD Int. Conf. on Management of Data, June 1992. – P. 321-330.

[5] Вовченко А.Е. Автоматизация создания адаптеров для сред неоднородных распределенных информационных источников// Сб. тез. XIV Межд. науч. конф. студентов, аспирантов и молодых ученых «Ломоносов». – М.: МГУ, 2007. – С. 14.

[6] Вовченко А.Е., Крупа А.В. Планирование запросов над множеством неоднородных распределенных информационных ресурсов в архитектуре средств поддержки предметных посредников// Тр. RCDL'2009. – Петрозаводск, 2009. – С. 335-342.

Methods and tools for subject mediators access to the streaming data

A.E. Vovchenko, L.A. Kalinichenko, M.U. Kostukov

Recently a new class of problems that require intensive stream data processing becomes widespread. Also the number of streaming resources, providing large data streams in real time increasingly grows. Wrappers play an important role in subject mediator infrastructure. Wrapper provides a uniform interface to access heterogeneous information resources from mediator.

The paper describes characteristics and semantics of streaming data model. An approach of implementation of the various types of queries in the streaming wrapper is described. Also streaming wrapper architecture is presented. It is shown how streaming wrapper may be used for scientific problem solving.

♣ Работа выполнена при частичной финансовой поддержке РФФИ (проекты 08-07-00157 и 10-07-00342)