Модел. и анализ информ. систем. Т. **21**, № **6** (2014) 169–175

УДК 519.683.5

PolarDB – система создания специализированных NoSQL баз данных и СУБД

Марчук А. Г.

Институт систем информатики им. А.П. Ершова СО РАН, НГУ,
630090 Россия, г. Новосибирск, пр. Лаврентьева, 6

e-mail: mag@iis.nsk.su

получена 3 октября 2014

Ключевые слова: PolarDB, Big Data, NoSQL, RDF, базы данных, СУБД

Представлена новая система, ориентированная на создание специализированных баз данных и систем управления базами данными. На основе анализа ряда NoSQL решений были выработаны принципы создания такой системы. В статье излагаются основные особенности примененного подхода, которые заключаются в том, что: а) предоставляются гибкие средства типовых определений, позволяющие создавать схемы структуризации данных, соответствующие разным парадигмам; б) сформирована система различных форм поддержания структурных значений и их отображений в файловые представления. Были проведены эксперименты с реализацией графов RDF, связанных реляционных таблиц, таблиц имен, объектно-реляционных отображений. Подход позволяет решать некоторые задачи создания технологий работы с большими данными.

Введение

В Институте систем информатики СО РАН накопился полезный опыт по работе с данными формата RDF при решении задач создания электронных архивов и исторической фактографии [1]. RDF – достаточно универсальное средство структуризации, ориентированное на формирование баз данных различной природы. К тому же данное представление соответствует ориентированному графу, поэтому возможно использовать RDF в широком спектре задач дискретной математики. В конце 2012 – начале 2013 годов, были проведены исследования по сопоставлению реализаций RDF средствами универсальный и специализированных баз данных.

В результате исследований [2] оказалось, что опробованные СУБД плохо приспособлены для реализации RDF. Это относится и к универсальным реляционным СУБД (MS SQL Server, MySQL), и к новомодным NoSQL-решениями [3] (MongoDB, Cassandra). Все решения плохо или чрезвычайно плохо справлялись с большими объемами данных (>100 млн. дуг в графе). Более или менее прилично вела себя специализированная (под RDF) СУБД Open Link Virtuoso [4], однако это платное

решение и наличие свободно-распространяемого варианта не спасает разработчиков от неприятных сюрпризов в будущем. Полученный опыт позволил провести эксперимент по формированию хранилища RDF-данных, пользуясь более примитивным базисом, т.е. бинарными файлами. В результате была создана и испытана на реальных данных в 1 млрд триплетов (дуг) своя специализированная СУБД. Это решение показало хорошие перспективы развития подхода, и в середине 2013 года был запущен проект создания системы PolarDB, которая предназначена для конструирования специализированных СУБД и для работы, в том числе, с большими данными.

1. Принципы создания PolarDB

Основными принципами, которые легли в основу PolarDB, являются следующие:

1. Базу данных реализовывать средствами файловой системы.
2. Все данные должны быть типизированы, при этом необходимо обеспечить гибкость структурирования данных.
3. Способ достижения быстродействия – минимизация количества чтений/записи, минимизация перемещений указателя Position.
4. Максимально использовать потоковую парадигму обработки.

Система структуризации базируется на типовой системе языка Поляр [5]. Основные ее элементы:

Структурное значение – древовидное построение (значение), интерпретируемое в соответствии с заданным типом.

Тип – древовидное построение, задающее интерпретацию для структурных значений. Тип может быть выражен в виде структурного значения.

Отображение структурных значений на систему хранения осуществляется системными средствами файловой системы. Оптимизации такого отображения также выполняются системными средствами операционной системы. Эти два решения позволяют воспользоваться наработанными в современных операционных системах и существующими в современных архитектурах механизмами ускорения работы с дисками и дисковыми файлами и при этом избежать интерференции различных кешевых структур.

2. Система типов

Тип может быть примитивным, или атомарным, или конструируемым, или строками (strings).

Примитивные типы:

boolean, character, integer, longinteger, real и, кроме того, тип none – множество значений, не содержащее ни одного элемента. Строки: sstring – соответствует понятию строки в объектно-ориентированном программировании.

Конструируемые типы:

- запись (record) – фиксированный набор типизированных полей. Есть поля 0, 1, ... n-1, каждое из которых представляет значение заданного в определении записи типа. Количество полей – фиксировано, типы полей – фиксированы;

- последовательность (sequence) – упорядоченный набор, состоящий из неопределенного (ноль или более) числа однотипных элементов;

- объединение (union) – значение, состоящее из тега и подзначения. Тег (динамически) определяет вариант типа для подзначения, теги нумеруются, начиная с 0.

Типовая система задается абстрактным классом PType, пример задания системы типов, соответствующих RDF представлениям:

```
PType tp_rec = new PTypeRecord(
 new NamedType("f1", new PType(PTypeEnumeration.integer)),
  new NamedType("f2", new PType(PTypeEnumeration.sstring)),
  new NamedType("f3", new PType(PTypeEnumeration.longinteger)));
PType tp_seq = new PTypeSequence(new PTypeRecord(
 new NamedType("id", new PType(PTypeEnumeration.sstring)),
  new NamedType("name", new PType(PTypeEnumeration.sstring)),
   new NamedType("fd", new PType(PTypeEnumeration.sstring)),
    new NamedType("deleted", new PType(PTypeEnumeration.boolean))));
Ptype  seqtriples = new PTypeSequence(
  new PTypeUnion(
    new NamedType("empty", new PType(PTypeEnumeration.none)),
      new NamedType("op",
        new PTypeRecord(
          new NamedType("subject", new PType(PTypeEnumeration.sstring)),
new NamedType("predicate", new PType(PTypeEnumeration.sstring)),
          new NamedType("obj", new PType(PTypeEnumeration.sstring)))),
          new NamedType("dp",
            new PTypeRecord(
new NamedType("subject", new PType(PTypeEnumeration.sstring)),
new NamedType("predicate", new PType(PTypeEnumeration.sstring)),
            new NamedType("data", new PType(PTypeEnumeration.sstring)),
new NamedType("lang", new PType(PTypeEnumeration.sstring))))));
```

3. Структурные значения

Структурные значения заданных типов хранятся (упаковываются) в “ячейках”. Ячейки играют роль переменных, хранение структурного значения в ячейке аналогично хранению значений в переменных традиционных языков программирования. Разница в том, что по завершении процесса исполнения программы значение не прекращает своего существования. Ячейки реализуются в виде потоков байтов, как правило, в форме бинарных файлов прямого доступа.

На рисунке 1 схематично изображены ячейки свободного и фиксированного форматов. В обоих случаях это поток байтов, предваряемый служебной информацией.

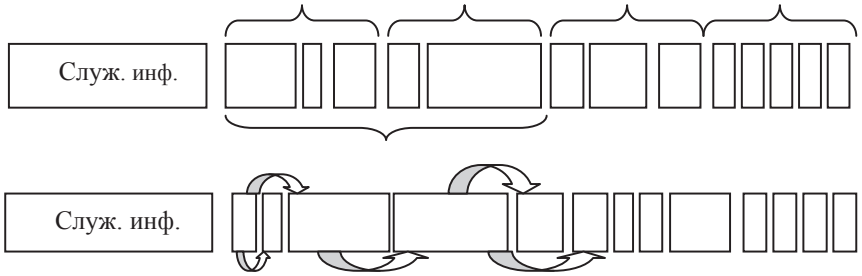

Рис. 1. Структура ячеек свободного и фиксированного форматов

Разница – в расположении структурных значений. Для ячеек свободного формата иерархия структурных значений разворачивается в сериализованный поток подзначений. Для ячеек фиксированного формата используется ссылочная связь между информационными блоками.

Для структурных значений существует текстовая интерпретация. Текстовое представление значений примитивных типов – традиционно. Запись изображается как – {поле1, поле2, ...}, последовательность – [элемент1, элемент2, ...], объединение – имя_тега^Значение.

Например, значение типа seqtriples может выглядеть:

```
[
  op^{"fogid.net/e/id3kjf", "rdf:type", "fogid.net/o/person"},
  dp^{"fogid.net/e/id3kjf", "fogid.net/o/name", "Иванов И.И.", "ru"},
  op^{"fogid.net/e/id3kjf", "fogid.net/o/father", "fogid.net/e/d9fdjf"}
]
```

Тестовое представление (интерпретация) используется для отладки или в качестве промежуточной формы, удобной для редактирования и напрямую годной для машинного применения.

Структурные значения также имеют объектную форму представления в оперативной памяти. При этом примитивные типы представляются значениями системных типов bool, int, long, string, запись – массив объектов (object[]), последовательность – массив объектов, объединение – массив объектов (массив из двух элементов: тега и значения варианта).

Средствами Linq объектную форму легко порождать, например из XML, и преобразовывать объектную форму в другие представления. Объектную форму можно записывать Set() во вход ячейки и читать Get() из входа.

Доступ к элементам структурных значений, хранящихся в ячейках, осуществляется через "входы" (Entries). Для ячеек свободного формата структура, определяющая вход, задается классом PaEntry, для ячеек фиксированного формата – PxEntry. Корневое (все) значение доступно через поле Root, соответственно во фрагменте

```
PxCell xcell $=$ new PxCell(type, path, true);
PxEntry ent2 $=$ xcell.Root;
```

первым оператором определена ячейка и ее связь с файлом, во втором операторе определен вход, сопоставленный с корневым значением, хранящимся в ячейке.

```
Методы класса PxEntry:
PxEntry Field(int nfield);
Int64 Count();
PxEntry Element(long ind);
IEnumerable\textless PxEntry\textgreater Elements() // Создает поток элементов записи
Int Tag();
PxEntry UElement()
object Get()
void Set(object value)
```

задают как выделения входов, соответствующих подзначениям структурных значений, так и возможность прочитать и записать эти значения. Аналогичный набор методов определен для PaEntry.

Одна из проблем, на которую обычно не обращают внимания создатели СУБД, это ввод данных, особенно ввод больших данных. Для реляционных СУБД считается достаточным добавление записи к определенной ранее таблице. В нашем случае структура хранимых значений может быть разной, и были сформированы специальные способы заполнения ячейки данными. Для ячеек свободного формата базовым методом является использование т.н. серийного скобочного потока (событий), который можно проиллюстрировать спецификацией интерфейса такого потока:

```
public interface ISerialFlow
{
        void StartSerialFlow();
        void EndSerialFlow();
        PType Type { get; }
        void V(object val);
        void R();
        void Re();
        void S();
        void Se();
        void U(int tag);
        void Ue();
}
```

Ввод через данную группу методов допустим для ячеек свободного формата. Для ячеек фиксированного формата задается другой вид потока – польский структурный поток. Польский структурный поток представляет собой поток примитивных элементов, объектных представлений структур, развернутых последовательностей, предваряемых количеством элементов, признака развертки записи, развернутых записей, объединение представляется в виде пары: значение тега и подзначение.

Поскольку предполагаемые формы использования включают в себя работу с большими данными, довольно большое количество усилий было направлено на эффективную реализацию разных видов сортировок и бинарных поисков.

4. Заключение

В настоящее время система PolarDB первично реализована и находится в стадии совершенствования через ее использование для построения экспериментальных СУБД

различного назначения. Были проведены эксперименты по следующим направлениям:

1. Реализация работы с RDF базами данных в рамках ведущихся в ИСИ работ по архивным системам и исторической фактографии.
2. Проверка возможности работы с реляционными таблицами.
3. Экспериментальная система RDF-Sparql, нацеленная на достижения рекордных характеристик при работе с RDF.
4. Проверка системы на реализации равновесных бинарных деревьев.
5. Реализация (больших) таблиц имен.
6. Проводится эксперимент с созданием объектно-реляционного представления в стиле ORM.

Список литературы

1. Марчук А.Г., Марчук П.А. Платформа реализации электронных архивов данных и документов // Электронные библиотеки: перспективные методы и технологии, электронные коллекции: Труды XIV Всероссийской научной конференции RCDL'2012. Переславль-Залесский, Россия, 15–18 октября 2012 г. Переславль-Залесский: изд-во «Университет города Переславля», 2012. С. 332–338. [Marchuk A.G., Marchuk P.A. Platforma realizatsii elektronnykh arkhivov dannykh i dokumentov // Elektronnye biblioteki: perspektivnye metody i tekhnologii, elektronnye kollektsii: Trudy XIV Vserossiyskoy nauchnoy konferentsii RCDL'2012. Pereslavl-Zalesskiy, Rossiya, 15–18 oktyabrya 2012 g. Pereslavl-Zalesskiy: izd-vo «Universitet goroda Pereslavly», 2012. S. 332–338 (in Russian)].
2. Марчук А.Г. На пути к большим RDF данным // Электронные библиотеки: перспективные методы и технологии, электронные коллекции: Труды XV Всероссийской научной конференции RCDL'2013. Ярославль, Россия, 14–17 октября 2013 года. Ярославль: ЯрГУ, 2013. С. 51–56 [Marchuk A.G. Na puti k bolshim RDF dannym // Elektronnye biblioteki: perspektivnye metody i tekhnologii, elektronnye kollektsii: Trudy XV Vserossiyskoy nauchnoy konferentsii RCDL'2013. Yaroslavl, Rossiya, 14–17 oktyabrya 2013 goda. Yaroslavl: YarGU, 2013. S. 51–56 (in Russian)].
3. NoSQL: Ultimate Guide to the Non-Relational Universe // http://nosql-database.org/
4. Open Link Software // http://www.openlinksw.com/
5. Марчук А.Г., Лельчук Т.И. Язык программирования Поляр: описание, использование, реализация. Новосибирск, 1986. 96 с. [Marchuk A.G., Lel'chuk T.I. Yazyk programmirovaniya Polyar: opisanie, ispol'zovanie, realizatsiya, Novosibirsk, 1986, 96 s. (in Russian)].

PolarDB – Infrastructure for Specialized NoSQL Datebases and DBMS

Marchuk A. G.

A.P. Ershov Institute of Informatics systems SB RAS, NSU,
Acad. Lavrentjev pr., 6, Novosibirsk, 630090, Russia

Keywords: PolarDB, Big Data, NoSQL, RDF, Databases, DBMS

The report presents a new system focused on the creation of specialized databases and database management systems. Based on the analysis of various NoSQL solutions there have been formulated some principles of such a system. The main features of the approach are: a) the infrastructure provides a flexible system of type definitions which allow to create data structures based on different paradigms; b) different forms of structured data support and mapping to the file system are used in the infrastructure. Experiments were provided with the implementation of RDF graphs, relational tables, name tables, object-relational mappings. The approach allows to solve some problems of technologies creation for work with Big Data.

Сведения об авторе:
Марчук Александр Гурьевич,
Институт систем информатики им. А.П. Ершова СО РАН, НГУ,
доктор физ.-мат. наук, профессор