Отображение моделей данных NoSQL в объектные спецификации

© Н. А. Скворцов
Институт проблем информатики РАН
nskv@ipi.ac.ru

Аннотация

Системы баз данных, принадлежащие к классу NoSQL, используются для обеспечения горизонтального масштабирования данных и работы со сверхбольшими объёмами данных. При решении задач над множественными неоднородными информационными ресурсами необходимо их использовать. В статье рассмотрены подходы к отображению моделей NoSQL разных видов в объектную модель языка СИНТЕЗ, используемого в качестве унифицирующей информационной модели при решении задач в среде неоднородных ресурсов. Показаны общие проблемы отображения на примере нескольких систем баз данных NoSQL и предложен подход, включающий выявление схем баз данных, не заданных явным образом, и отображения операций. Рассмотрен случай, когда схему данных восстановить невозможно из-за неполной информации о структурах, хранимых в базах, или слабой структурированности данных.

Работа выполнена при поддержке РФФИ (гранты 10-07-00342-а, 11-07-00402-а) и Президиума РАН (программа 16П, проект 4.2).

1 Введение

Проекты, работающие в информационном пространстве, сталкиваются с необходимостью горизонтального распределения по узлам большого количества данных и быстрого доступа к ним. Для решения проблемы нередко прибегают к использованию для хранения и обработки данных систем баз данных, принадлежащих к классу NoSQL [1]. К таковым причисляют хранилища «ключ-значение», кортежные системы баз данных с колоночным хранением, документные системы баз данных. Они использовались, в первую очередь, в поисковых сервисах, в крупных социальных сетях, и сегодня применяются в тех проектах, где необходимо работать с большим количеством данных или высоконагруженными хранилищами данных.

При решении научных задач над множественными информационными ресурсами также приходится сталкиваться с базами данных на основе NoSQL. Для интеграции базы данных в среду решения задач необходимо построить отображение схемы базы в концептуальную схему задачи. С целью разрешения модельной неоднородности между спецификациями базы данных и задачи модель данных системы баз данных NoSQL должна быть отображена в некоторую унифицирующую целевую модель, используемую на уровне спецификаций решаемой задачи. Однако путь такого отображения не всегда является очевидным, так как в моделях данных систем баз данных NoSQL наряду со структурированными данными могут присутствовать слабоструктурированные и неструктурированные, схема базы чаще всего не имеет спецификации, а предполагается неявно, может быть нефиксированной, изменяемой динамически, может содержать сложные переплетения экземпляров данных и структурных элементов.

Статья посвящена исследованию отображения моделей данных NoSQL различных видов в объектную модель языка СИНТЕЗ [3], используемую в качестве канонической модели предметных посредников для решения задач над множественными информационными ресурсами. В разделе 2 описаны общие принципы, положенные в основу построения систем баз данных NoSQL и анализ того, как описанные принципы влияют на отображение моделей данных NoSQL в целевую модель задачи. В разделе 3 приведены краткие сведения об объектной модели данных языка СИНТЕЗ, используемой в статье в качестве целевой при отображении моделей. В разделе 4 описаны возможности модели данных системы баз данных «ключ-значение» Oracle NoSQL и показаны принципы отображения данной модели в объектную модель языка СИНТЕЗ, обозначены проблемы такого отображения. Разделы 5 и 6 посвящёны исследованию системы с колоночным хранением Cassandra и документной системы баз дынных MongoDB. В разделе 7 сделаны выводы об общих проблемах отображения систем баз данных NoSQL

Труды 14[й] Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» - RCDL-2012, Переславль-Залесский, Россия, 15-18 октября 2012 г.

разных классов, о подходах к отображению моделей в тех случаях, когда не удаётся выделить фиксированную структуру данных.

2 Общие принципы построения систем баз данных NoSQL

Технологии NoSQL направлены на обеспечение горизонтального масштабирования и доступа к данным сверхбольших объёмов или работу с высоконагруженными по чтению или обновлению данными, они часто обеспечивают более приемлемую производительность простых операций, чем реляционные системы баз данных. Системы баз данных NoSQL, в основном, пользуются следующими принципами работы с данными.

В NoSQL произошёл отказ от табличной организации данных, хранимых по кортежам, что позволяет избежать обращений к целым кортежам, если необходимо прочитать только некоторые их компоненты. Для быстрого поиска необходимых данных они организованы в виде пар «ключ-значение», для доступа к значениям по ключам используется принцип хеш-таблиц. Для надёжности и физического распределения по узлам данные могут быть связаны с определённым узлом или реплицироваться на несколько узлов с помощью механизмов, таких как распределённые хеш-таблицы (DHT) [7].

Так как обращения производятся только по ключам, то упрощается и язык манипулирования данными. Обычно используется набор операций, подобный CRUD (Create/Read/Update/Delete) [4]. В зависимости от организации ключей и значений на основе операций могут составляться SQL-подобные запросы, однако и они должны укладываться в обращение по ключам.

Обращение по ключам вносит существенные ограничения в возможностях запросов, которые, в результате, влияют на организацию данных. В большинстве систем возможные разновидности запросов, используемых при решении задач, должны быть известны заранее. Те данные, по которым необходимо организовать поиск, должны оказаться ключами в некоторых парах «ключ-значение», а искомые данные – значениями. Для этого формируются вспомогательные пары, дублирующие данные в разной организации ключей и значений. Фактически они реализуют над данными в основных парах материализованные взгляды, которые должны формироваться в соответствии с данными в основных парах при каждом их обновлении. Таким образом, происходит сдвиг обработки запросов на этап обновления данных, что усложняет обновление, но позволяет быстро выполнять чтение данных. Используется журнальная структура обновлений, при которой необходимые обновления буферизуются и распространяются на необходимые узлы по мере возможности. Операции чтения используют уже имеющиеся данные, даже если эти данные ожидают обновления. Непротиворечивость обновляемых данных гарантируется только конечная.

Ключи и значения могут иметь различную структуру и организовываться в различные структуры в разных системах. С точки зрения моделей данных системы баз данных NoSQL можно разделить на следующие основные классы:

- базы данных «ключ-значение», основанные напрямую на парах ключей и значений, в которых, однако, и ключи, и значения, возможно, будут иметь определённую структуру или быть слабоструктурированными;
- базы данных с колоночным хранением, в которых с одним ключом связан набор колонок значений, имеющих имена, и таким образом, имитируется табличная организация;
- документные базы данных, в которых разрешены иерархические структуры, основанные на вложенных парах «ключ-значение», то есть, значением любой пары может становиться последовательность пар ключ значение.

Те решения, которые используются для достижения распределённости, возможности интенсивного чтения и записи, особенностей обеспечения целостности данных, являются прозрачными для пользователей, работающих с системами баз данных NoSQL. То, что имеется в распоряжении у пользователя, включает некоторым образом организованные структуры на основе пар «ключ-значение» и обычно простой интерфейс к ним.

Поэтому на отображение моделей данных систем баз данных NoSQL в информационную модель, используемую для решения задач над множественными информационными ресурсами, влияет только организация структуры данных и состав операций системы. Остальные используемые принципы: распределённое хранение, способы обновления данных, – скрыты за интерфейсом системы баз данных и на отображение не влияют.

Как в традиционных системах баз данных, так и в системах баз данных NoSQL при решении задачи отображения моделей строится отображение языков определения и манипулирования данными. В случае моделей NoSQL модель не определяется схемой данных в явном виде, и отображение на уровне языка определения данных сталкивается с отсутствием её описания. Интерфейс приложения, работающего над базой, обеспечивает более высокий уровень семантики данных. Однако приложения создаются для решения определённых задач, одна база данных может использовать в ряде приложений, не всегда на уровне приложений предоставляют достаточные средства доступа к данным для отображения схем и трансляции данных. Поэтому, несмотря на неопределённость схемы, представляется оправданным выявление структуры данных из доступных источников: кода приложения, документации, знаний создателя базы

и приложения, анализа самих данных, – и отображение моделей на уровне операций системы баз данных.

Отображение моделей с участием моделей NoSQL, в основном, востребовано в обратном направлении: из произвольных моделей в NoSQL. При отображении реляционной модели в NoSQL разрабатываются способы представления в NoSQL данных, ранее представленных в реляционной модели и имитация операций реляционной алгебры в моделях NoSQL [6]. Отображение объектной модели в NoSQL имеет целью исследовать представление объектов при работе языков программирования с базами данных NoSQL [5]. При этом подчёркивается отсутствие дорогостоящего объектно-реляционного отображения, с каким приходится сталкиваться при работе в языках программирования с реляционными системами баз данных.

3 Сведения об объектной модели данных языка СИНТЕЗ

Язык СИНТЕЗ [3], используемый в данной статье в качестве целевого языка при отображении моделей данных, является языком спецификации информационных ресурсов и включает в качестве синтаксической основы язык фреймов, построенную над ним объектную модель и средства выражения логических формул.

Фреймы имеют идентификатор и набор слотов, каждый из которых может иметь набор значений слота. Язык фреймов позволяет специфицировать метафреймы, метаслоты и метазначения, которые сами определяются как фреймы. Средствами языка фреймов выражаются произвольные виды информации, в том числе, слабоструктурированные.

Объектная модель рассматривает фреймы как типизированные значения. Фрейм может отражать состояние объекта определённого типа, в этом случае, его структура должна соответствовать спецификации некоторого абстрактного типа данных. Помимо абстрактных типов данных, определяющих структуру и поведение объектов, вводятся также классы как множества однотипных объектов. Фрейм может становиться экземпляром класса, если соответствует типу экземпляров этого класса.

Язык формул в язык СИНТЕЗ используется для предикативных спецификаций, задающих ограничения целостности в типах, для определения правил в логических программах и для задания запросов над спецификациями задач или информационных ресурсов.

Перечисленными средствами и должны быть выражены спецификации, отображённые из моделей данных NoSQL. Подробные сведения о языке СИНТЕЗ можно почерпнуть из [3].

4 Отображение модели данных «ключ-значение»

Рассмотрение отображения моделей данных NoSQL в объектную модель языка СИНТЕЗ начнём с отображения системы баз данных Oracle NoSQL [9]. Эта система относится к классу систем баз данных «ключ-значение».

Хранимыми элементами в данной системе являются ключи и значения. Ключи являются составными и разделяются на старший ключ (major key) и младший ключ (minor key).

Старший ключ представляет собой последовательность из одного или более компонентов, являющихся строками. По старшему ключу, помимо прочего, производится выбор узла в кластере, так что пары, у которых старшие ключи совпадают, гарантированно будут находиться на одном и том же узле.

Младший ключ также является последовательностью строк, однако он может быть и пустым. Вместе со старшим ключом младший ключ определяет уникальное значение ключа, с которым в базе может быть связано одно значение.

Значения в парах Oracle NoSQL могут быть произвольными. Они рассматриваются как строки байтов, и в них могут быть записаны данные с любой семантикой, структурированные или неструктурированные, сериализованные.

Состав языка манипулирования данными соответствует упомянутому выше набору операций CRUD, именованных по-своему.

Операция put(k, v) записывает в базу значение v по ключу k. Она используется одновременно для вставки и обновления данных. В языке есть условные варианты данной операции: putIfAbsent – создание пары, если такой ключ не найден в базе, putIfPresent – обновление значения, если такой ключ присутствует в базе, а также putIfVersion – для работы с версиями значений.

Операция get(k) читает значение, соответствующее ключу k. Операция множественного чтения multiGet(k, r) читает пары «ключ-значение», соответствующие неполному ключу k, задающему только часть первых компонентов полных ключей, в r налагаются условия на значения компонента ключа, следующего за компонентами, заданными в k.

Список существующих ключей, хранящихся в базе, не затрагивая соответствующие им значения, можно прочитать с помощью операции multiGetKeys (k, r).

Операция delete (k) удаляет из базы пару, соответствующую ключу k. Операция множественного удаления multiDelete (k, r) работает с параметрами по аналогии с операцией multiGet.

Приведём пример спецификации на языке Java, работающий с интерфейсом системы Oracle NoSQL.

```
List<String> majorComponents = new ArrayList<String>();
List<String> minorComponents = new ArrayList<String>();
```

```
majorComponents.add("Smith");
majorComponents.add("Bob");
minorComponents.add("phonenumber");
minorComponents.add("home");
Key myKey = Key.createKey
  (majorComponents, minorComponents);
String data = "555 5555";
Value myValue = Value.createValue (data.getBytes());
kvstore.put(myKey, myValue);
ValueVersion vv = kvstore.get(myKey);
Value v = vv.getValue();
String data = new String(v.getValue());
```

В данном примере определяется старший ключ “Smith/Bob”, состоящий из двух компонентов, и младший ключ “phonenumber/home”, также из двух компонентов. Полный же ключ составляет строку: “Smith/Bob/-/phonenumber/home”. В данном случае, старший ключ определяет имя и фамилию человека, к которому относятся данные, а младший ключ – путь в структуре, описывающей вид данных об этом человеке, в данном случае, в значении, связанном с данным ключом записан домашний телефон человека.

На языке JSON [2], который часто используется для представления пар, таким образом определяемые данные можно записать следующим образом:

```
{"Smith/Bob/-/phonenumber/home": "555 5555"}
{"Smith/Bob/-/phonenumber/mobile": "333 3333"}
```

Фактически ключами определяется некоторая иерархическая структура:

```
Smith
└ Bob
  └ phonenumber
    ├ home: "555 5555"
    └ mobile: "333 3333"
```

То, что так распределены старший и младший ключи, гарантирует, что все данные об одном и том же человеке будут всегда храниться на одном узле.

Касательно отображения модели данных системы Oracle NoSQL в целевую модель, определённую языком СИНТЕЗ, можно заключить, что различие между старшими и младшими ключами никак не отражается на отображении. Полный ключ рассматривается как список строк. Однако приведённый пример показывает, что сложность отображения будет заключаться в семантике этих компонентов. В любом месте составного ключа могут быть данные (“Smith”), а могут быть фиксированные имена в структурах данных (“phonenumber”).

Это, в свою очередь, означает, что полностью автоматическое отображение в объектную модель, учитывающее такую структуру данных, организовать будет сложно. Если отображение зависит от семантики строкового значения в компонентах ключей, то правильное отображение с уверенностью может определить только эксперт.

Поэтому в целом, если возможно отображение и трансляция данных на уровне приложения над базой данных, то предпочтительно использовать этот уровень. В этом случае особенности модели NoSQL теряются за интерфейсом приложения. Если приложение не позволяет это сделать, для отображения модели данных NoSQL на уровне языка манипулирования данными в модель языка СИНТЕЗ необходимо сформировать структуры, в которые будут преобразовываться данные, и отобразить операции манипулирования данными. Интерес исследования представляет последний случай.

4.1 Формирование структуры данных

Первым делом, для отображения структур данных, хранящихся в базе данных модели Oracle NoSQL, важно понять, какие из компонентов ключей являются для объектной модели экземплярами (“Smith”), а какие определяют атрибуты структур (“phonenumber”). Для этого необходимо разобраться в особенностях использования компонентов ключей разной структуры.

Наиболее верным решением может быть, если эксперт укажет, какие из значений компонентов являются фиксированными и определяют структуру, а какие – данными. Для фиксированных значений компонентов также необходимо определить, какие из них являются дочерними для других, то есть связанными с определёнными значением родительского компонента. Эксперт может решить эти задачи на основании анализа:

- примеров ключей,
- документации к базе данных,
- кода приложений, работающих с базой данных;
- «логов» операций обращения к базе.

Возможна и частичная автоматизация. Интуитивно, если компонент имеет фиксированный набор значений, который к тому же связан с единственным определённым значением другого компонента, он может оказаться определяющим структуру данных. Такую статистику можно набрать в самой базе, запрашивая в ней ключи. Однако статистический подход может быть дорогостоящим и при этом не гарантировать точность результатов. В любом случае, эксперт должен подтвердить результаты статистического анализа. В данной статье детали возможного статистического подхода не рассматриваются.

После того, как выяснена природа фиксированных значений компонентов ключей, можно переходить к отображению структуры базы данных. Можно выделить следующие правила отображения:

1. Связанные по структуре ключей пары отображаются в классы (Class1, Class2, …) и типы их экземпляров. Если какой-либо компонент ключа фиксирован и имеет

единственное значение в подобных ключах, то он становится именем класса.

2. В соответствие компонентам ключей, имеющим нефиксированные значения, ставятся атрибуты типа экземпляров класса (majorKey1, majorKey2, …, minorKey1, minorKey2, …). Значения компонентов станут при этом строковыми значениями атрибутов. Однако при известной семантике компонентов ключей возможно задавать другие типы атрибутов и преобразование строковых значений компонентов ключей к ним. Атрибуты могут иметь пустое значение, если компонент ключа не определён в некоторых парах.
3. Если в качестве компонента ключа используется фиксированное значение, его следует сделать атрибутом с именем, соответствующим значению компонента. Сопоставление атрибутов со значениями будет задано позже.
4. Если для компонента с фиксированным значением существуют дочерние фиксированные компоненты, то есть, фиксированные значения связаны с единственным определённым значением данного родительского компонента, то создаётся одноимённый со значением родительского компонента абстрактный тип данных с атрибутами, соответствующими фиксированным значениям дочернего компонента. Атрибут, порождённый родительским компонентом, приобретает в качестве типа значений созданный абстрактный тип данных.
5. Значение пары отображается в значение атрибута, соответствующего последнему фиксированному значению, у которого нет дочерних компонентов. Либо, если такового не нет, то значение пары отображается в атрибут value. Типом значения атрибута по умолчанию является фрейм языка СИНТЕЗ. Необходимо знать способ преобразования значений во фреймы. Если выявить такой способ невозможно, значения будут иметь тип Bitstring языка СИНТЕЗ. Значения могут приводиться к любым другим типам в зависимости от их известной семантики и структуры.
6. Набор всех атрибутов типа, образованных компонентами полных ключей, указывается как уникальный. Если с атрибутом, соответствующим фиксированному значению компонента ключа, связано значение, соответствующее значению в паре, то включать его в набор излишне.

Приведём пример отображения в язык СИНТЕЗ следующих пар «ключ-значение»:

{“Smith/Bob/-/phonenumber/home”: “555 5555”}
{“Smith/Bob/-/phonenumber/mobile”: “333 3333”}

Как мы выяснили, первые два компонента ключа являются нефиксированными, а последние два – фиксированными. В результате отображения получим следующую структуру данных:

```
{ class1; in: class;
  instance_section: {
    majorKey1: String;
    majorKey2: String;
    phonenumber: Phonenumber;
    key: { unique; { majorKey1, majorKey2 } };
}}

{ Phonenumber; in: type;
   home: String;
   mobile: String;
}
```

Проследим, как применяются правила при этом отображении. В соответствии с ключами с похожей структурой создан класс class1 и тип его экземпляров в разделе instance_section (применяется правило 1 из вышеперечисленных). Нефиксированным компонентам ключа ставятся в соответствие атрибуты majorKey1 и majorKey2 типа String (правило 2). В соответствии с первым фиксированным значением компонента создаётся атрибут phonenumber (правило 3). Типом значений этого атрибута становится абстрактный тип данных Phonenumber, содержащий атрибуты, соответствующие фиксированным значениям последнего компонента ключа: home и mobile (правило 4). Значениям пар соответствуют значения атрибутов home и mobile, их типами значений становится строка (правило 5). В типе экземпляров класса class1 указывается уникальность значений пары атрибутов majorKey1 и majorKey2 (правило 6).

Здесь необходимо также отметить, что обычным подходом в NoSQL, как уже было сказано ранее, является дублирование данных в дополнительных парах «ключ-значение» с другой структурой ключей и значений, которые фактически являются материализованными взглядами над теми же данными и служат для создания других ключей и возможности задания запросов по ним. Например, если в базе с приведённой выше структурой необходим поиск по номерам телефона, то в ней, помимо описанных пар необходимо будет создать дополнительные пары, в которых номера телефонов являются ключами:

{"555 5555": ["Smith", "Bob"]}

Такие пары отображаются в уже построенные структуры, и для них не нужно создавать дополнительные структуры, а важно разработать отображение операций в язык манипулирования данными целевой модели данных. Ведь присутствие таких пар расширяет разновидности возможных запросов, выражаемых в целевой модели, а именно, то, на какие атрибуты можно налагать условия, и

какие данные при таких условиях можно возвращать.

В худшем случае, семантика данных, хранимых в базе, может быть вообще не известна. В этом случае отображение может строиться только исходя из структур, используемых для организации ключей и значений. При этом структуры, используемые в Oracle NoSQL, отображаются в структуры, состоящие из последовательностей строковых значений младшего и старшего ключей и произвольной битовой строки в качестве значения:

```
{ db; in: class;
   instance_section: {
      minorKey: {sequence; type_of_element: String};
      majorKey: {sequence; type_of_element: String};
      value: Bitstring;
      key: { unique; { minorKey, majorKey } };
}}
```

4.2 Отображение операций

После того, как отображена структура данных, необходимо построить отображение языка манипулирования данными. Такое отображение необходимо построить в две стороны. Отображение операций Oracle NoSQL в запросы на языке формул языка СИНТЕЗ позволит понять, какие виды запросов возможно отобразить в операции Oracle NoSQL. Обратное отображение позволит восстановить операции Oracle NoSQL по запросам на языке формул.

Рассмотрим операции системы Oracle NoSQL. Во-первых, спецификации с операцией get(k). Параметр k должен быть полным уникальным ключом, по которому требуется узнать единственное значение пары.

```
majorComponents.add("Smith");
majorComponents.add("Bob");
minorComponents.add("phonenumber");
minorComponents.add("home");
Key myKey = Key.createKey
  (majorComponents, minorComponents1);
ValueVersion vv = kvstore.get(myKey);
Value v = vv.getValue();
```

Данной программе соответствует спецификация запроса:

```
q([v]) :- class1([majorKey1, majorKey2,
   v : phonenumber.home]) &
majorKey1="Smith" & majorKey2="Bob"
```

Из класса class1, соответствующего ключам данной структуры, выбирается фрагмент значений типа его экземпляров, состоящий из атрибутов majorKey1 и majorKey2, построенных в соответствии с компонентами, содержащими нефиксированные значения, и пути phonenumber.home к значению пары. На атрибуты majorKey1 и majorKey2 накладываются ограничения, соответствующие заданным значениям компонентов ключа. Значение пары переименовывается в переменную v и возвращается в классе q, являющемся ответом на запрос. Так как пара атрибутов majorKey1 и majorKey2 уникальна, в ответе будет возвращено одно значение, если только значение ключа присутствует в базе.

Обратное отображение подобного запроса полностью восстановит операции задания ключа и обращения к базе в исходном виде.

Операция multiGet(k) передаёт в качестве параметра фрагмент ключа и возвращает множество пар, ключи которых соответствуют данному фрагменту.

```
majorComponents.add("Smith");
majorComponents.add("Bob");
Key myKey = Key.createKey(majorComponents);
SortedMap<Key, ValueVersion>  x =
  kvstore.multiGet(myKey);
```

Данной программе соответствует спецификация запроса:

```
q(x) :- class1(x/class1.inst) &
x.majorKey1="Smith" & x.majorKey2="Bob"
```

В отличие от Oracle NoSQL запрос на языке формул возвратит в классе результата всего один объект. Дело в том, что в ключе myKey, определённом в программе, отсутствуют только фиксированные компоненты, а значит, из нефиксированных значений компонентов ключа и значений пар будут сформированы все значения атрибутов в объекте-экземпляре класса class1.

В случае, если определённый ключ будет содержать, помимо нефиксированных, фиксированные компоненты, например, phonenumber, возвращаться будет не полный объект, а только объект, связанный с соответствующим атрибутом, то есть, в данном случае – объект типа Phonenumber.

Обратное отображение запроса в программу вызова операций системы в данном случае также восстановит операции формирования неполного ключа из ограничений значений атрибутов и из типа возвращаемого в запросе значения и вызов операции multiget.

Операция multiGet поддерживает также задание диапазона ключей, пары с которыми необходимо вернуть. В следующей программе задаётся первый компонент ключа, а для второго компонента указывается диапазон значений.

```
majorComponents.add("Smith");
Key myKey = Key.createKey(majorComponents);
KeyRange kr = new KeyRange(
  "Bob", true, "Patricia", true);
SortedMap<Key, ValueVersion>  x =
  kvstore.multiget(myKey, kr);
```

Появление условия на диапазон значений дополнит отображённый запрос условиями для атрибута majorKey2.

```
q(x) :- class1(x/class1.inst) &
x.majorKey1="Smith" &
x.majorKey2>="Bob" & x.majorKey2<="Patricia"
```

Ответ на данный запрос будет содержать все объекты-экземпляры class1, соответствующие условиям, наложенным на атрибуты.

Последняя операция, которую мы рассмотрим, - операция возврата ключей, присутствующих в базе multiGetKeys.

```
majorComponents.add("Smith");
Key myKey = Key.createKey(majorComponents);
SortedSet<Key> k = kvstore.multiGetKeys(myKey);
```

Отображённый запрос может вернуть только данные, соответствующие компонентам ключей, являющимся нефиксированными. Возможностей возвращать метаинформацию о схеме в языке СИНТЕЗ нет.

```
q([majorKey1, majorKey2]) :-
class1([majorKey1, majorKey2]) & majorKey1="Smith"
```

Обратное отображение запроса восстановит программу полностью, однако возвращать он будет множество полных ключей вместе с разными значениями фиксированных компонентов.

Операции обновления и удаления пар отображаются подобным образом с помощью специальных конструкций в языке правил языка СИНТЕЗ, вставляющих и удаляющих экземпляры классов.

Для примера отображения вспомогательных пар, дублирующих основные данные, рассмотрим приведённый ранее пример.

```
{ "555 5555": ["Smith", "Bob"] }
```

Следующий код демонстрирует работу с такими парами:

```
majorComponents.add("555 5555");
Key myKey = Key.createKey(majorComponents);
ValueVersion vv = kvstore.get(myKey);
```

Для правильного отображения операций в построенные ранее структуры необходимо знать, какие элементы основных пар дублируются элементами вспомогательных. Отображение в язык СИНТЕЗ будет выглядеть так:

```
q([majorKey1, majorKey2]) :-
class1([majorKey1, majorKey2,
v : phonenumber.home]) &
v="555 5555"
```

Оно выражается в терминах определённого ранее класса class1 на языке СИНТЕЗ, возвращая фамилию и имя человека по его домашнему телефону.

При обратном отображении запроса на языке СИНТЕЗ использование значения основной пары в операции сравнения (v=”555 5555”) говорит о том, что в отображении будет использоваться вспомогательная пара, где эти значения являются ключами.

5 Отображение модели данных с колоночным хранением

Система баз данных Cassandra [8] относится к классу систем с колоночным хранением. Колонками в ней называются множества пар «ключ-значение». Семейства колонок, имитирующие табличную организацию данных, состоят из множества колонок, у каждой из которых есть название, значение и временная метка, и на которые ссылается ключ. Ключ является строкой, и с одним значением ключа в семействе может быть связано единственное значение каждой колонки. Также значения колонки могут включать множество подколонок с именами и значениями (без ключа). В этом случае она называется суперколонкой. Манипулирование данными производится с помощью операций get, multiGet, getSlice, multiGetSlice для вариантов чтения данных, insert и remove для добавления/обновления и удаления данных.

Особенность использования модели в отличие от традиционных реляционных баз данных состоит в том, что имена и значения в колонках могут использоваться, как угодно. Например, семейство может состоять из неограниченного количества колонок, в которых имена не являются именами колонок, но и в именах, и в значениях хранятся данные. Это означает, что и в данной системе мы сталкиваемся с той же проблемой определения, какие из элементов являются фиксированными именами для построения структур, а в каких хранятся данные. Запросы к семействам выполняются только по ключу, соответственно для выполнения запросов по другим элементам в системе Cassandra, как и в Oracle NoSQL, используются вспомогательные семейства с материализованными взглядами, имеющими ключом необходимый элемент.

Для формирования объектной структуры, соответствующей исходной базе, используем следующие правила:

1. Семейство колонок отображается в класс и связанный с ним тип экземпляров класса, ключ семейства становится атрибутом id типа экземпляров, с ним связано свойство уникальности значения.
2. Если имена колонок фиксированы (в именах не данные, а названия колонок, одни и те же для всех ключей), то имена колонок отображаются в

одноимённые атрибуты типа экземпляров класса. Значения колонок, связанные с одним значением ключа, отображаются в значения атрибутов в объекте с id, соответствующим значению ключа семейства.

3. Если имена колонок нефиксированы (являются данными), то для пар имя-значение колонки создаётся абстрактный тип данных с двумя атрибутами, а семейство колонок отображается в класс с атрибутом id и атрибутом, значением которого является последовательность значений созданного типа.
4. Ключи, имена колонок и значения вспомогательных семейств (материализованных взглядов на основные семейства) отображаются в атрибуты уже существующего класса, если все колонки присутствуют в основных семействах.
5. Подколонки отображаются в отдельный абстрактный тип данных. В зависимости от того, фиксированы или не фиксированы имена подколонок, структура типа задаётся по тому же принципу, что и для колонок семейств.

Представим прежний пример, в виде семейства колонок для системы Cassandra:

```
users:
{ "1": { "firstName": "John",
         "lastName": "Smith",
         "phoneNumbers":
           { "home": "555 5555",
             "mobile": "333 3333"}
       }, …
}
```

Семейство users по значению ключа связывает колонки firstName, lastName и phoneNumbers, причём последняя является суперколонкой, хранящей две колонки: home и mobile.

Такая структура данных будет отображена в спецификацию на языке СИНТЕЗ в виде класса users и типа его экземпляров с атрибутами id для ключа, firstName и lastName для фиксированных колонок и атрибутом phoneNumber с отдельным типом, создаваемым в соответствии со структурой суперколонки.

```
{ users; in: class;
  instance_section: {
    id: String;
    firstName: String;
    lastName: String;
    phoneNumbers: PhoneNumbers;
    key: { unique; id };
}}
{ PhoneNumbers; in: type;
  home: String;
  mobile: String;
}
```

Задавать запросы к такой базе возможно только по ключу, которому соответствует атрибут id. Так, операция get, запрашивающая колонки firstName и lastName по ключу "1", будет отображена в запрос:

```
q([firstName, lastName]) :-
users(x/[id, firstName, lastName]) & x.id="1"
```

Как и в случае систем баз данных «ключ-значение», в колоночных базах для задания запросов с условиями, накладываемыми на элементы, не являющиеся ключом семейства колонок, в базе должны быть заранее созданы вспомогательные семейства, для которых эти элементы являются ключами. Отображение вспомогательных семейств производится в структуры, построенные для основных семейств. Для этого необходимо знать, какие элементы дублируют друг друга в разных колонках.

6 Отображение документной модели данных

Последняя рассматриваемая в статье система MongoDB [10] относится к классу документных систем баз данных. Документом в ней является последовательность пар «ключ-значение» с возможностью создавать последовательности вложенных пар в значениях. Интерфейс системы использует для задания документов и их фрагментов язык JSON. Помимо запросов по ключам здесь возможно строить запросы и по значениям, для этого можно создавать вторичные индексы.

```
users:
{ "firstName": "John",
  "lastName": "Smith",
  "phoneNumbers":
    { "home": "555 5555",
      "mobile": "333 3333"
    }
}
```

Структуры в документах могут быть неограниченными по вложенности, например, описывая XML-документ. В этом случае документы системы баз данных MongoDB должны отображаться во фреймы языка СИНТЕЗ. С другой стороны, они могут быть неограничены по количеству пар «ключ-значение» в последовательности. И если семантика хранимых данных такова, что в именах и значениях находится последовательность данных без вложения, то целесообразно отображать их в тип данных последовательности языка СИНТЕЗ. Однако если возможно выяснить фиксированную структуру документов, как в приведённом примере, то их имеет смысл отображать в значения абстрактных типов данных с формированием классов.

```
{ users; in: class;
  instance_section: {
    firstName: String;
    lastName: String;
```

```
    phoneNumbers: PhoneNumbers;
}}
{ PhoneNumbers; in: type;
  home: String;
  mobile: String;
}
```

Пример операций, запрашивающие номер домашнего телефона home по фамилии lastName будет выглядеть следующим образом:

```
db.things.ensureIndex({'lastName':1});
db.users.find({'lastName': 'Smith'},
  {'phoneNumbers.home':1});
```

И отображение данного запроса, соответственно, будет таким:

```
q([v]) :- users([lastName,
  v : phonenumber.home)
& lastName="Smith"
```

Как видно, при отображении документных баз мы сталкиваемся с теми же проблемами, что и при работе с системами баз данных NoSQL остальных классов. В зависимости от семантики данных, хранимых в парах, отображение в объектную модель может быть различным.

7 Выводы об общих проблемах отображения моделей NoSQL в объектные спецификации

Обобщая результаты анализа отображения систем баз данных NoSQL различных классов, можно сделать вывод, что вне зависимости от класса системы – «ключ-значение», колоночной или документной – приходится сталкиваться с похожими проблемами отображения.

Так как в системах баз данных NoSQL схема бывает не определена и может присутствовать неявно, подразумеваемая создателем базы данных и реализованная в коде приложения, то семантически значимое отображение в объектную модель автоматически строить невозможно. Каждая отдельная база данных потребует выявления схемы с участием эксперта. Есть смысл строить отображение на уровне приложения, работающего над базой данных, минуя отображение моделей системы баз данных. Однако приложения могут быть закрытыми для внешних запросов, не предоставляющими произвольного доступа к данным, быть разбитыми на части, работающими с разными подмножествами данных из одних и тех же баз, с одной базой может работать множество приложения. Поэтому оправдано проведённое в статье исследование для построения отображения моделей на уровне операций системы NoSQL, включающего выявление семантики хранимых данных и отображения структур с её учётом.

Основная причина проблем, возникающих при семантическом отображении моделей данных NoSQL в объектную модель, заключается в их слабой структурированности:

- в большинстве из них не определяется схема данных в явном виде;
- значения в парах «ключ-значение» часто могут быть произвольными и неструктурированными, ключи могут рассматриваться и как элементы структуры, и как данные;
- при отсутствии схемы структура пар может изменяться динамически. Фактически, с помощью операции обновления можно в любой момент ввести новую структуру пар «ключ-значение»;
- базы данных обычно не ограничены по количеству пар в структурах, по длине ключей, по вложенности пар.

Такие системы могут использоваться для решения задач, изначально рассчитанных на использование нефиксированных и неограниченных слабоструктурированных данных (например, иерархических или потоковых). С другой стороны, многие приложения используют системы баз данных NoSQL для хранения достаточно жёстко типизированных структур. И тот, и другой случаи приходится учитывать при отображении в объектную модель языка СИНТЕЗ.

При выявлении семантики данных обращается внимание на использование в ключах фиксированных (имена элементов структур) или нефиксированных (данные) значений. В случае выявляемой статической схемы данных фиксированные значения ключей отображаются в атрибуты типов, нефиксированные – в значения атрибутов.

При отображении создаются следующие структуры:

- типы последовательности для последовательностей пар;
- фреймовые структуры для вложенных пар;
- абстрактные типы данных в случае, если данные подчиняются выявляемой чёткой структуре.

Возможные разновидности запросов, используемых при решении задач над данными в моделях NoSQL должны быть известны заранее. Данные, по которым необходимо организовать поиск, должны оказаться ключами в некоторых парах «ключ-значение», а искомые данные – значениями, для чего формируются вспомогательный пары над основными данными с соответствующей структурой данных. Такие пары при отображении не образуют новые структуры, а отображаются в типы и классы, образованные основными парами. В соответствии с вспомогательными парами в целевой модели образуются вторичные ключи. Запросы со сравнением по атрибутам, являющимся ключами в вспомогательных парах, при обратном отображении отображаются в операции над этими парами.

Заключение

На примере отображения моделей трёх систем баз данных NoSQL различных классов в язык СИНТЕЗ показаны принципы отображения моделей NoSQL в объектную модель. Затронуты проблемы отображения, которые по большинству являются общими для различных систем NoSQL. Данная работа может быть также использована для отображения моделей данных NoSQL в другие целевые модели данных.

Литература

[1] *R. Cattell.* Scalable SQL and NoSQL Data Stores. ACM SIGMOD Records. - Vol. 39. Iss. 4. - NY:ACM New York, 2010. - P. 12-27

[2] *D. Crockford.* The application/json Media Type for JavaScript Object Notation (JSON). – RFC 4627. – 2006. – URL: http://tools.ietf.org/html/rfc4627

[3] *Kalinichenko L.A., Stupnikov S.A., Martynov D.O.* SYNTHESIS: a Language for Canonical Information Modeling and Mediator Definition for Problem Solving in Heterogeneous Information Resource Environments. – Moscow: IPI RAN, 2007. – 171 p.

[4] *J. Martin.* Managing the Data-base Environment. – New Jersey: Prentice-Hall. - 381 p. – ISBN 0-13-550582-8.

[5] *H.J.M. Meijer.* Object model to key-value data model mapping – US Patent App. 12/938,168, 2010. – Google Patents.

[6] *D. Merriman.* SQL to Mongo Mapping Chart. – 2011. – URL: http://www.mongodb.org/display/DOCS/SQL+to+Mongo+Mapping+Chart

[7] *I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan.* Chord: a scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Transactions on Networking (TON), 11(1):17.32, 2003.

[8] The Apache Saccandra Project. – URL: http://cassandra.apache.org/

[9] Getting Started with NoSQL Database 11g Release 2. – Oracle. – 2011. – URL: http://docs.oracle.com/cd/NOSQL/html/GettingStartedGuide/Oracle-NoSQLDB-GSG.pdf

[10] MongoDB. – URL: http://www.mongodb.org/

Mapping of NoSQL data models to object specifications

Nikolaj Skvortsov

Database systems belonging to NoSQL class are used to supply horizontal data scaling and to work with extralarge data volumes. It is necessary to involve them into the problem solving over multiple heterogeneous information resources. The paper considers approaches to mapping of NoSQL models of different types to object model of the SYNTHESIS language used as the unifying information model in problem solving at heterogeneous resources environment. The paper shows common mapping problems using several NoSQL databes systems and proposes an approach for discovery of database schemes not specified explicitly, as well as mapping of operations. It considers a case when it is impossible to recover data scheme because of lack of information about data structures stored in databases including the semi-structured data case.