Two-way Mapping of Graph and Relational Data Models for Multi-model Databases

Arkady Osheev

Lomonosov Moscow State University, Moscow, Russia
arkadii_osheev@mail.ru

Abstract. Modern information systems simultaneously use data structured in various ways. The relational data model is no longer sufficient to represent them in a natural and efficient way. In addition, conceptual schemas of many subject areas are much more convenient to represent in NoSQL models like graph data model. Combining several DBMSs implementing different data models within the same information system leads to significant complication of system management. One of the possible solutions to overcome this problem is application of a multi-model DBMS. A formal basis for the development of such DBMS is a two-way mapping between the applied data models. Existing approaches to mapping between graph and relational models either implement one-way mapping or implement two-way mapping using an intermediate representation. This work is devoted to a two-way mapping between graph and relational data model schemas, as well as between the relational (SQL) and graph (Cypher) query languages. Mapping is illustrated by examples of schemas and queries. Comparison of existing multi-model databases that implement graph and relational models is also provided.

Взаимное отображение графовой и реляционной моделей данных для мультимодельных баз данных

Аркадий Ошеев

Московский государственный университет им. М.В. Ломоносова, Москва, Россия
arkadii_osheev@mail.ru

Аннотация. Современные информационные системы одновременно используют данные различной структуры. Для их представления уже недостаточно реляционной модели данных. Также, концептуальные схемы многих предметных областей значительно удобнее представлять в NoSQL мо-

делях, например, в графовой модели. Использование в рамках одной информационной системы нескольких СУБД, реализующих разные модели данных, приводит к существенному усложнению управления системой. Одним из возможных решений данной проблемы является использование мультимодельной СУБД. Формальной основой для разработки такой СУБД является взаимное отображение между применяемыми моделями данных. Данная работа посвящена взаимному отображению между схемами графовой и реляционной моделей данных, а также между языками запросов SQL и Cypher. В настоящее время существует множество подходов к отображению графовой модели в реляционную модель и наоборот. Все эти подходы в полной мере реализуют отображение в одну сторону или взаимное отображение с использованием промежуточного представления. Рассмотрены и продемонстрированы на конкретных примерах идеи отображения схем и языков запросов. Проведено сравнение существующих мультимодельных баз данных, реализующих графовую и реляционную модели.

Ключевые слова: интеграция моделей данных, графовая модель данных, реляционная модель данных.

1 Введение

Современные информационные системы одновременно используют данные различной структуры. Для их представления уже недостаточно реляционной модели данных. Также, концептуальные схемы многих предметных областей значительно удобнее представлять в NoSQL моделях, например, в графовой модели. Использование в рамках одной информационной системы нескольких СУБД, реализующих разные модели данных, приводит к существенному усложнению управления системой.

Одним из возможных решений данной проблемы является использование мультимодельной СУБД [6]. Формальной основой для разработки такой СУБД является взаимное отображение между применяемыми моделями данных.

Эта работа посвящена теме интеграции реляционной и графовой моделей. Реляционная модель является самой популярной в мире и занимает 60% рынка на 2019 год[1]. Графовая модель естественна для представления некоторых предметных областей, например, социальных сетей. Также, из-за отсутствия схемы, возможно хранить данные разнородных объектов в одном месте.

В настоящее время развиваются мультимодельные СУБД, поддерживающие графовую и реляционную модели такие как Agens Graph, CosmosDB, OrientDB, MarkLogic.

Несмотря на наличие данных СУБД, вопрос интеграции реляционной и графовой моделей не решен полностью. Одной из проблем является разнообразие языков манипулирования графовыми данными: Cypher, SPARQL, Gremlin, PGQL. Также ведутся работы по разработке единого стандарта языка запросов Graph

[1] https://scalegrid.io/blog/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use/

Query Language (GQL). Стандарт GQL [17] был одобрен в сентябре 2019 года, на него повлияли работы [2][5]. Запросы в GQL описываются в декларативной форме подобно SQL.

Для интеграции интересующих моделей MarkLogic и OrientDB отображают данные в промежуточное представление. Agens Graph не предоставляют отображение моделей друг в друга, а просто хранит данные графовой и реляционной моделей в раздельных хранилищах и для одновременного обращения к этим хранилищам необходимо производить гибридные запросы.

На данный момент существует множество подходов к отображению графовой модели в реляционную модель и наоборот. Все эти подходы в полной мере реализуют отображение только в одну сторону или взаимное отображение с использованием промежуточного представления. Например, в работе [14] предложен алгоритм трансформации реляционных данных в графовое представление. В статье [10] описывается взаимное отображение языков запросов SQL, Cypher в язык запросов, основанный на лямбда-исчислении.

Таким образом, по сведениям автора, в настоящее отсутствуют полноценные подходы к взаимному отображению реляционной и графовой моделей без использования промежуточного представления. Данная работа направлена на объединение существующих решений и предоставления подхода по взаимному отображению языков запросов и подхода по отображению схем данных реляционной и графовой моделей.

Целью данной работы является разработка взаимного отображения графовой и реляционных моделей для обеспечения возможности обращения на реляционном языке запросов к графовым данным и наоборот. Такое отображение может быть применено в качестве формальной основы для мультимодельной СУБД.

Для достижения данной цели необходимо решить следующие основные задачи:

1. Произвести обзор существующих мультимодельных баз данных, реализующих графовую и реляционную модели.
2. Определить варианты взаимного отображения схем графовой и реляционной моделей.
3. Определить варианты взаимного отображения языков запросов реляционной и графовой моделей.
4. Реализация отображений схем и запросов для конкретных выбранных реляционной и графовой СУБД.
5. Произвести сравнение эффективности вариантов отображения на тестовых базах данных и наборах запросов.

В качестве графового языка запросов выбран Cypher, который близок к GQL, и был разработан в компании Neo4j. В 2015 реализация языка была открыта в рамках проекта openCypher. В качестве реляционного языка запросов выбран SQL, реализованный в большом количестве проприетарных и свободно распространяемых СУБД.

Работа выполняется в рамках магистерской диссертации на факультете Вычислительной математики и кибернетики МГУ им. М.В. Ломоносова, программа «Большие данные: инфраструктуры и методы решения задач».

В статье приведены результаты первого года исследований: произведен обзор существующих мультимодельных СУБД, предложено по одному варианту основных идей отображений (i) реляционной схемы в графовую схему, (ii) графовой схемы в реляционную схему, (iii) запросов Cypher в SQL, (iv) запросов SQL в Cypher. Все идеи сопровождаются примерами отображений с пояснениями.

Данная статья структурирована следующим образом. В разделе 2 представлены родственные работы и приведено краткое сравнение мультимодельных СУБД. В разделе 3 приведен вариант отображения схем графовых и реляционных баз данных друг в друга. В разделе 4 представлен вариант отображения языков запросов. Варианты отображений проиллюстрированы на примерах. В разделе 5 приведено заключение и направления дальнейших работ.

2 Родственные работы

Работа [6] представляет собой подробное сравнение существующих мультимодельных СУБД по многим параметрам, таким как: объединяемые модели, используемый язык запросов к данным, стратегия оптимизация запросов и другие. Авторы выделили следующие проблемы существующих мультимодельных СУБД: (i) существующие языки запросов в мультимодельных СУБД не полноценны, так как не позволяют предложить оптимальный план исполнения запроса, (ii) трудность в разработке мультимодельной схемы для СУБД, которая будет учитывать все нюансы интегрируемых моделей, (iii) сложность вывода схемы из экземпляра базы данных, так как дополнительно необходимо учитывать связи между моделями, (iv) сложность в управлении развития схемы мультимодельной СУБД, так как необходимо учитывать межмодельные связи в виде ссылок, внешних ключей и пр.,

В данной работе приведено краткое сравнение мультимодельных СУБД Agens Graph[1], CosmosDB[2], OrientDB[3], MarkLogic[4]. Сравнение проводилось на основе документации СУБД по следующим критериям: поддерживаемые языки запросов, модели данных и некоторые аспекты интеграции данных. Информация сгруппирована и представлена в таблице 1. По результатам анализа были сделаны следующие выводы:

- В MarkLogic и OrientDB интегрируемые модели отображаются в промежуточное представление, с которым СУБД умеет работать. Мы же хотим избежать введения промежуточного представления с помощью отображения графовой и реляционной моделей друг в друга. Это позволит нам предоставлять пользователю выбор удобного ему интерфейса для работы с данными.

---

1 https://bitnine.net/documentations/manual/agens_graph_developer_manual_en.html
2 https://docs.microsoft.com/ru-ru/azure/cosmos-db/
3 http://www.orientdb.com/docs/last/index.html
4 https://docs.marklogic.com

- В AgensGraph отсутствует возможность обращаться к графовому хранилищу на реляционном языке запросов и наоборот. Предоставление такой возможности является одним из результатов данной работы.
- CosmosDB является проприетарной облачной базой данных. Это накладывает ограничения на использование ее в проектах, где отсутствует связь с внешним миром.
- OrientDB использует в качестве языка запроса свой диалект SQL. Это накладывает ограничение в виде отображения диалекта и стандарта SQL на миграцию на данную СУБД или с данной СУБД.
- CosmosDB, OrientDB в качестве графового языка запроса используют Gremlin, который внес меньший вклад в стандартизацию графового языка запроса, чем Cypher [16].

В течение последних лет было множество работ по поводу миграции данных с реляционных баз данных в графовые базы данных [7][9][13]. Данная ситуация обусловлена получением возможностей, которые предоставляются графовыми СУБД, такими, как масштабируемость и явное представление желаемых структур данных в определенных предметных областях.

Таблица 1. Сравнение мультимодельных баз данных

СУБД / Показатель	AgensGraph	CosmosDB	MarkLogic	OrientDB
Реляционный язык запросов	SQL к реляционному хранилищу	LINQ	SQL	Свой диалект SQL
Графовый язык запросов	Cypher к графовому хранилищу	Gremlin	SPARQL	Gremlin
Модели	Графовая, реляционная, документоориентированная, ключ-значение	Графовая, реляционная, документоориентированная,ключ-значение, колоночная	Документоритегрированая, графовая, реляционная	Графовая, реляционная, документоориентированная,ключ-значение, объектноориентированная
Интеграция моделей	SQL исполняется над реляционным хранилищем, Cypher над графовым. Интеграцию данных хранилищ можно произвести с помощью гибридных запросов	Позволяет выполнять запросы на реляционном и графовом языках запросов над данными представленными в формате JSON.	При интеграции источников необходимо отобразить интегрируемые данные в промежуточную схему, называемую Entiy	Работает с записями, которые реализуют один из четырех типов: документ, BLOB, ребро, вершина. Записи объединяются в классы, похожие на отношения в реляционной модели.

В работе [14] предложен алгоритм трансформации реляционных данных в графовое представление. Представлены идеи, как при обходе отношений выяснить, что является вершиной, а что ребром. В данной работе слабо затронута тема отображения языка запросов.

Neo4j в работе [15] при переходе от реляционного представления в графовое дает следующие рекомендации:

1. Все записи таблиц представляются как вершины с именем сущности, как название таблицы;
2. В качестве ребер берутся внешние ключи или возможные соединения между таблицами.

Представления графов в виде ER-диаграммы было предложено в статье [3]. В рамках этой работы было создано приложение, которое на основе построения ER-диаграммы также строит и граф. В работе [1] описывается процесс переноса связей в ER-моделе в NoSQL базы данных. Приводится сравнительная характеристика отражения вида связей в различные формы баз данных. Среди них выделяется характеристики графовой базы данных, в которую можно отразить все виды связей.

В работе [9] рассматривается отображение ER-диаграмм только в структуру графа с учетом видов связей.

Тема интеграции реляционных и графовых баз данных затрагивается в работе [11]. В статье описывается опыт по переносу данных о публичном транспорте из реляционной системы хранения в графовую базу данных и использовании их одновременно. Данные действия обусловлены эффективностью графовых алгоритмов. В статье [10] предлагается схема интеграции данных, язык запросов которой построен на лямбда-исчислении, приводятся примеры перевода его в язык SQL и Cypher.

Тема отображения Cypher в другие языки запросов не остается без внимания исследователей. В работе [12] представлен инструмент, который теоретически мог отображать Cypher в любой язык запросов, при правильном формировании файла топологического отображения gTop.

В gTop выделяется два уровня: абстрактный и реализационный. Абстрактный отвечает за формальное описание вершин и ребер в графе, реализационный описывает как граф отображается в данной системе с указанием, какие с какими таблицами описываемая таблица может соединяться. Получается, что пользователь этой библиотеки должен сам описать, как в реляционной системе отображается граф.

Топологический файл помогает при отображении избежать комбинационного взрыва при поиске необходимого отображения в другой язык запросов.

Практически реализован функционал по отображению Cypher в язык SQL в виде библиотеки Cytosm.

Работа [8] посвящена формализации запросов на языке Cypher посредством адаптации операторов реляционной алгебры для работы с графами, также пред-

ставляют новые операторы. Авторы представляют алгоритм по отображению запросов в нее. Данная работа не покрывает всего множества языка. Авторы не утверждают, что их адаптация операторов отображается в обычную реляционную алгебру.

Целью магистерской работы является объединение перечисленных решений для построения взаимного отображения графовой и реляционной моделей данных. Идеи отображения схем данных основаны на работах [3][9][12][13][14]. В качестве источника для идей отображения языков используются работы [5][15]. В статье приведено по одному варианту отображений (i) реляционной схемы в графовую схему, (ii) графовой схемы в реляционную схему, (iii) запросов Cypher в SQL, (iv) запросов SQL в Cypher

3 Отображения схем данных

В данном разделе рассматриваются (i) отображение схем реляционной модели в схемы графовой модели и (ii) отображение схем графовой модели в схемы реляционной модели. Отображения иллюстрируются на примерах конкретных схем.

Реляционная модель была предложена Коддом в 1970 году [4], и она предусматривает представление данных в виде отношений. Отношения состоят из кортежей, которые в свою очередь состоят из пар (*название атрибута*, *значение атрибута*). В данной работе атрибуты могут принимать лишь атомарные значения, такие типов, как целочисленный, строковый и пр. Предполагается, что в каждом отношении R определен первичный ключ - значение (или совокупность значений), которая точно идентифицирует каждый кортеж в R. Если же в отношении R имеется атрибут A, который определен в другом отношении S, как первичный ключ, то A является внешним ключом в R.

В качестве структур *графовой модели данных* в данной работе используются атрибутные графы. Атрибутным графом называется множество вершин и соединяющих их ребер. Вершины и направленные ребра которые могут быть аннотированы совокупностью пар (*название атрибута, значение атрибута*) и меткой (*label*). Каждое ребро соединяет ровно две вершины.

Основные идеи отображения схем реляционной модели в графовую модель состоят в следующем:

- Тип отношения с составным первичным ключом, состоящим из двух внешних ключей, отображаются в множество ребер с одноименной меткой, при этом предполагается, что каждому кортежу соответствует отдельное ребро. Атрибуты кортежа отображаются в соответствующие атрибуты ребра.
- Остальные типы отношений отображаются в множество вершины с одноименной меткой, при этом предполагается, что каждому кортежу соответствует отдельная вершина. Атрибуты кортежа отображаются в соответствующие атрибуты вершины.
- Внешние ключи типов отношений отображаются в ребра между вершинами.

Заметим, что в данных идеях не рассматриваются случаи, когда первичный ключ состоит из более чем двух элементов или составлен специфичным образом. Например, как композиция внешних ключей, которые в свою очередь состоят из нескольких атрибутов. Также считается, что внешний ключ состоит из одного атрибута. Не учитываются функциональные зависимости между типа отношений. Разработка данных аспектов является предметом дальнейшей работы.

Для демонстрации отображения используем фрагмент базы данных North-Wind [15], схема которого представлена на рис.1.

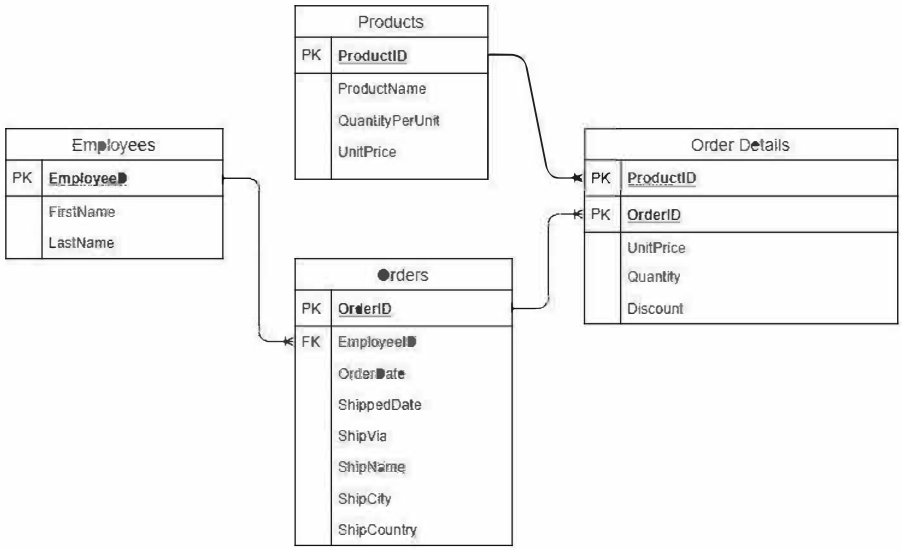

Рис. 1. Фрагмент схемы базы данных NorthWind

Отображение данной реляционной схемы в графовую схему конкретизируется следующим образом:

- Типы отношений *Employees, Orders, Products* отображаются множества вершин с одноименными метками.
- Формируются следующие ребра:
  - между вершинами с метками *Employees* и *Orders* – ребра с меткой *EmployeesID*;
  - между вершинами с метками *Products* и *Orders* – ребра с метками и атрибутами *OrderDetails*.

На рис. 2 представлена результирующая графовая схема.

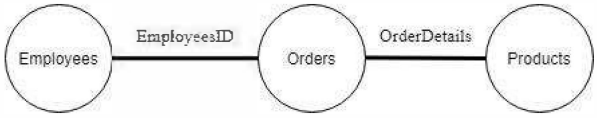

Рис. 2. Отображение реляционной схемы в графовую схему

Основные идеи отображения схем графовой модели в реляционную модель состоят в следующем:

- Каждая метка вершин отображается в одноименный тип отношения. Вершины представляются кортежами соответствующих типов отношений. Атрибуты вершин представляются одноименными атрибутами кортежей. Первичные ключи вершин представляются первичными ключами соответствующих типов отношений.
- Каждая метка ребер отображается в одноименный тип отношения. Ребра представляются кортежами соответствующих типов отношений. Атрибуты ребра представляются одноименными атрибутами кортежей. В качестве составного первичного ключа берутся первичные ключи типов отношений, полученных из вершин.

Для иллюстрации вышеприведенного отображения рассмотрим подграф, взятый из модельного примера, который предоставляет компания Neo4j (см. рис. 3).

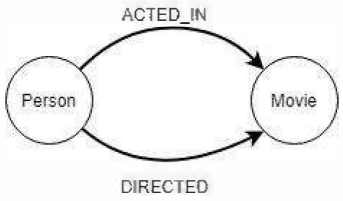

Рис. 3. Схема подграфа графа базы данных фильмов

Разберем подробнее схему подграфа:

- *Person* — метка вершины, соответствующей человеку.
- *Movie* — метка вершины, соответствующей кино.
- *ACTED_IN* — метка ребра с атрибутами, связывающее вершины с метками *Person* и *Movie.*
- *DIRECTED* — метка ребра без атрибутов, связывающее вершины с метками *Person* и *Movie.*

Отображение данной графовой схемы в реляционную схему конкретизируется следующим образом:

- Вершины с метками *Person* отображаются в кортежи одноименного типа отношения. Первичный ключ типа отношения имеет имя *PersonID.*
- Вершины с метками *Movie* отображаются в кортежи одноименного типа отношения. Первичный ключ типа отношения имеет имя *MovieID.*
- Ребра с метками *ACTED_IN* отображаются в одноименный тип отношения. Атрибуты ребра отображаются в одноименные атрибуты отношения. Первичный ключ является составным и состоит из внешних ключей *PersonID, MovieID.*

- Ребра с метками *DIRECTED* отображаются в одноименный тип отношения. Первичный ключ является составным и состоит из внешних ключей *PersonID, MovieID.*

На рис. 4 представлена результирующая реляционная схема.

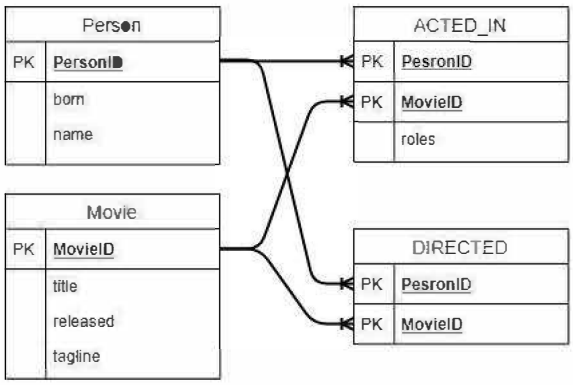

Рис. 4. Реляционное представление подграфа.

4 Отображение языков запросов

В данном разделе рассматриваются идеи отображений между графовым и реляционным языками запросов. Запросы к схемам являются SELECT-FROM-WHERE запросами. В качестве конкретных языков запросов рассматриваются Cypher и SQL.

Приведенные запросы и их отображения протестированы в СУБД Neo4j и PostgreSQL над базами данных, схемы которых представленны в разделе 3. Реализацию планируется выполнить в ходе дальнейшей работы.

Cypher был представлен в 2011 году компанией Neo4j. В 2015 его реализация раскрыта в рамках проекта openCypher.

Пример 1: Запрос на Cypher поиска сотрудников, которые продавали шоколад

```
MATCH (e:Employee)-[:SOLD]-(o:Order)-[:CONTAINS]->(p:Product)
WHERE p.productName="Chocolade"
RETURN e.firstName, e.lastName
```

Пример состоит из оператора MATCH, который в свою очередь состоит из следующих разделов запроса MATCH, WHERE, RETURN.

В разделе запроса MATCH перечислен набор вершин и связывающих их ребер. Согласно [5] вершины и ребра имеют следующие шаблоны.

Вершина представляется выражением $(a{:}\, l\, \{P\})$, где:

- $a \in A \cup \{nil\}$ – имя вершины в запросе. Наличие имени дает возможность использоваться эту вершину дальше в теле запроса.

- $l \subset L$ — возможно пустое конечное множество меток вершины. В данной работе состоит из одного элемента.
- P — возможно пустое конечное множество частичных отображений атрибутов вершины в выражения. Выражения накладывают дополнительные ограничения, которые должны быть в искомых вершинах.

Пример 2: Вершина в Cypher

$$(x{:}\,Person\ \{name{:}\,'Peter'\})$$

В примере 2: x — имя вершины в запросе; $Person$ — метка вершины; $name{:}\,'Peter'$ — отображение атрибута $name$ в имя $'Peter'$

Ребро представляется выражением [$a{:}\,t * I\{P\}$] , где:

- $a \in A \cup \{nil\}$ — имя ребра. Наличие имени дает возможность обращаться к этому ребру дальше в теле запроса.
- $t \in T$ — возможно пустое конечное множество меток ребра. В данной работе состоит из одного элемента. Отражает какие метки используются в запросе.
- P — возможно пустое конечное множество частичных отображений атрибутов ребра в выражения. Выражения накладывают дополнительные ограничения, которые должны быть в искомых ребрах.
- I — интервал, состоящий из минимального и максимального числа ребер, которые необходимо пройти для нахождения необходимой вершины. Показатель либо пустой, либо состоит из $(m, n) \in \mathbb{N} \cup \{nil\}$.

Пример 3: Ребро в Cypher

$$[k{:}\,KNOWS * 1..2\ \{since{:}\,1985\}]$$

В примере 3: k — имя вершины; $KNOWS$ — метка ребра; $1..2$ — интервал, показывающий то, что ребра с данной меткой могут встречаться в пути от 1 до 2 раз; $\{since{:}\,1985\}$ – отображение атрибута вершины $since$ в 1985.

В разделе запроса WHERE указываются фильтрующие условия для запроса. В примере 1 это название продукта – шоколад.

В разделе запроса RETURN указываются возвращаемые значение. В примере 1 это имя и фамилия сотрудника.

Для соединения запросов в одну цепь используется раздел запроса WITH. Он позволяет использовать результаты одного запроса, как входящие данные для следующего запроса.

Пример 4: Использование WITH в запросе

```
MATCH (n{name:'David'})
WITH n
MATCH (n)-[:KNOWS]-(m)
RETURN m.name
```

В примере 4 сначала мы выбираем вершину с именем 'David', потом используем ее для следующего раздела запроса MATCH, где мы ищем всех людей, которых знает 'David'.

4.1 Отображение графового языка запросов в реляционный язык запросов

Основные идеи отображения запроса на Cypher в SQL состоят в следующем:

- Шаблон (вершина)-[ребро]-(вершина)-…-(вершина) в разделе запроса MATCH отображается, как набор соединений между отношениями.
- Шаблон (вершина) или [ребро] в разделе запроса MATCH отображается в одно отношение.
- При отображении вершин, ребер из раздела запроса MATCH необходимо учитывать их шаблоны и отобразить их следующим образом:
  - Метки вершин, ребер отобразить в разделе запроса FROM реляционного запроса.
  - Множество частичных отображений отображаются в предикаты в раздел запроса WHERE реляционного запроса.
- Предикаты из раздела запроса WHERE отображаются в предикаты раздела запроса WHERE реляционного запроса.
- Атрибуты результатов запроса в разделе запроса RETURN отображаются в одноименные атрибуты в реляционный раздел запроса SELECT.
- Разделы запроса, которые предшествуют разделу запроса WITH отображаются в вложенный запрос, который соединяется с отображением следующих после WITH разделов запроса.

Далее приведены примеры запросов к реляционной схеме на рис. 1. Схема для запросов на Cypher берется из рис.2.

Запрос 1.1. Вывести продавцов, которые продавали шоколад.

```
MATCH (e:Employees)-[:EmployeeID]-()-[:OrderDetails]-(p:Prod-
ucts{ProductName:'Chocolade'})
RETURN e.firstName, e.lastName
```

Конкретизируем отображения данного запроса:

- В разделе запроса MATCH указаны марки вершин и ребер, одна из вершин () — анонимная. Из графовой схемы, можно определить, что в анонимной вершине может присутствовать только вершина *Orders*.
- Отображаем это в набор соединений между отношениями, учитывая идеи отображения, что ребрами являются внешние ключи и отношения с составным

первичным ключом. Некоторые вершины и ребра не имеют имени, то присвоим им имя в реляционном отображении. Так ребру *OrderDetails* присваивается имя od.

- Отображение *{ProductName:'Chocolade'}* отображается в предикат *p.ProductName='Chocolade'* реляционного запроса в разделе WHERE.
- Атрибуты в разделе запроса RETURN отображаются в одноименные атрибуты в раздел запроса SELECT.

Результат отображения

```
SELECT e.firstName, e.lastName
FROM Employees as e
JOIN Orders as o ON o.EmployeeID = e.EmployeeID
JOIN Order_Details as od ON od.orderId=o.orderId
JOIN Products as p ON p.productID=od.productID
WHERE p.ProductName='Chocolade',
```

Запрос 1.2. Вывести имена и фамилию сотрудников, дату заказа, которые участвовали в отгрузке товара из Сиэтла и их имена начинаются на A.

```
MATCH (e:Employees)
WHERE e.firstName STARTS WITH 'A'
WITH e
MATCH (e)-[:EmployeeID]-(o:Orders{shipCity:'Seattle'})
RETURN e.lastName, e.firstName, o.orderDate
```

Конкретизируем отображения данного запроса:

- До WITH: раздел запроса MATCH состоит из одной вершины, которая имеет только метку. Вершина отображается в раздел запроса FROM реляционного запроса. Раздел запроса WHERE с предикатом START WITH отображается в предикат LIKE 'A%' реляционного раздела запроса WHERE.
- WITH: так как не указаны, какие именно атрибуты будут использоваться дальше, то происходит отображение всех атрибутов в раздел запроса SELECT вложенного запроса.
- После WITH: раздел запроса MATCH состоит из шаблона (вершина)-[ребро]-(вершина), который мы отобразим в набор соединений между отношениями. Вершина *e* передана с помощью WITH, поэтому соединение будет происходит с вложенным запросом по внешнему ключу *EmployeeID* с отношением *Orders*. Составляющие вершин и ребер отображаются согласно идее, изложенным выше. Атрибуты в разделе запроса RETURN отображаются в одноименные атрибуты в раздел запроса SELECT.

Результат отображения

```
SELECT e.lastName, e.firstName, o.orderDate
FROM Orders as o
JOIN (SELECT e.* FROM Employees as e WHERE e.firstName LIKE 'A%') as e
ON(o.EmployeeID=e.EmployeeID)
WHERE o.shipCity = 'Seattle'
```

4.2 Отображение реляционного языка запросов в графовый язык запросов

Основные идеи отображения запроса на SQL в Cypher состоят в следующем:

- Если в реляционном запросе имеется вложенный запрос, то вложенный запрос необходимо отобразить запрос на языке Cypher, а результаты запроса передать в основной с помощью раздела запроса WITH.
- Раздел запроса FROM, состоящий из перечисления отношений, отображается в названия меток в вершинах и ребрах. При наличии оператора JOIN, соединяемые отношения отображаются в шаблон (вершина)-[ребро]-(вершина)-…-(вершина).
- Предикаты-условия в разделе запроса *WHERE* отображаются согласно следующим идеям:
  - Если предикат-условие — оператор равенства атрибута какому-то значению, то переносим в оператор MATCH в вершину или ребро в отображение атрибутов в выражения
  - Другие предикаты-условия отображаем в предикаты условия в графовый раздел запроса WHERE.
- Атрибуты результатов запроса в разделе запроса SELECT отображаются в одноименные атрибуты в реляционный раздел запроса в RETURN.

Далее приведены примеры запросов к графовой схеме на рис. 3. Схема для запросов на SQL берется из рис. 4.

Запрос 2.1. Выбрать всех актеров, которые снимались в фильме «Матрица».

```
SELECT p.name
FROM Person as p,
JOIN ACTED_IN as a ON (a.PersonID=p.PersonID),
JOIN Movie as m ON (a.MovieID = m.MovieID)
WHERE m.title = 'The Matrix'
```

Конкретизируем отображения данного запроса:

- Атрибут в разделе запроса SELECT отображается в одноименный атрибут раздела запроса RETURN.

- Раздел запроса FROM в связи с наличием оператора JOIN отображается в шаблон (вершина)-[ребро]-(вершина)-…-(вершина). По графовой схеме определим, что *Person, Movie* — вершины, *ACTED_IN* — ребро между ними.
- Предикат в разделе запроса WHERE отображается в шаблон вершины *Movie,* так как является предикатом равенства значению.

Результат отображения

```
MATCH (p:Person)-[:ACTED_IN]->(m:Movie{title: 'The Matrix'})
RETURN p.name
```

Запрос 2.2. Выбрать всех актеров, которые снимались в фильме «Матрица» с использованием вложенного запроса.

```
SELECT p.name
FROM Person as p,
WHERE p.PersonID IN (
SELECT per.PersonID
FROM Person as per
JOIN ACTED_IN as a ON (a.PersonID=per.PersonID),
JOIN Movie as m ON (a.MovieID = m.MovieID)
WHERE m.title = 'The Matrix')
```

Конкретизируем отображения данного запроса:

- Вложенный запрос отображается в запрос на языке Cypher, как запрос, описанный выше. Результатом вложенного запроса является коллекция идентификаторов. Поэтому результат отобразится в аргумент функции *collect.* Это все передастся в основной запрос с помощью WITH.
- Предикат IN в разделе запроса WHERE отобразится в одноименный предикат в графовый раздел запроса WHERE.
- Атрибут в разделе запроса SELECT основного запроса отображается в одноименный атрибут раздела запроса RETURN.

Результат отображения

```
MATCH (per:Person)-[:ACTED_IN]->(m:Movie{title: 'The Matrix'})
WITH collect(per.PersonID) as l
MATCH (p:Person)
WHERE p.PersonID IN l
RETURN p.name
```

5 Заключение и направления дальнейшей работы

В данной работе был рассмотрен вопрос отображения графовых и реляционных моделей данных друг в друга. Приведено краткое сравнение мультимодельных баз данных, которые объединяют эти модели. Рассматриваются основные понятия реляционной и графовой моделей.

Был произведен обзор существующих мультимодельных СУБД, предложено по одному варианту основных идей отображений (i) реляционной схемы в графовую схему, (ii) графовой схемы в реляционную схему, (iii) запросов Cypher в SQL, (iv) запросов SQL в Cypher. Все идеи сопровождаются примерами отображений с пояснениями.

Результаты данной работы могут быть использованы как формальное основание для мультимодельной базы данных или системы интеграции графовой и реляционной моделей.

Основные направления дальнейшей работы выглядят следующим образом:

- учет при отображении важных, но не рассмотренных понятий реляционной и графовой моделей, таких, как функциональные зависимости различного вида, составные внешние ключи и т.д.;
- исследование других вариантов отображения между схемами и языками запросов (например, отображение каждого типа отношения в отдельную вершину графа). Планируется рассмотреть влияние функциональных зависимостей на отображение;
- формализация алгоритмов отображения схем и запросов между Cypher и SQL. Языки запросов планируется формализовать с использованием реляционной алгебры и реляционной алгебры над графами, предложенной в работе[7]. Для формализации правил отображения языков запросов планируется использовать движимый моделями подход (MDA) и язык отображения моделей ATL[1];
- реализация отображений схем и запросов для конкретных выбранных реляционной и графовой СУБД;
- сравнение эффективности вариантов отображения на тестовых базах данных и наборах запросов.

Благодарности. Автор выражает благодарность профессору НИУ ВШЭ в Санкт-Петербурге Борису Асеновичу Новикову за идею работы и ассистенту СПБГУ Георгию Алексеевичу Чернышеву за замечания и предложения по улучшению работы. Работа выполняется в рамках магистерской диссертации на факультете Вычислительной математики и кибернетики МГУ им. М.В. Ломоносова под научным руководством С. А. Ступникова, ведущего научного сотрудника Федерального исследовательского центра «Информатика и управление» Российской Академии Наук.

[1] https://www.eclipse.org/atl/

Литература

1. Alotaibi, Obaid, Pardede E.: Transformation of Schema from Relational Database (RDB) to NoSQL Databases. Data **4**, pp. 148 (2019). doi:10.3390/data4040148.
2. Angles R., Arenas M., Barcelo P., Boncz P., Fletcher G., Gutierrez C., Lindaaker T., Paradies M., Plantikow S., Sequeda J., Voigt H., Oskar van Rest : G-CORE: A Core for Future Graph Query Languages. In Proceedings of the 2018 International Conference on Management of Data (SIGMOD '18). Association for Computing Machinery, New York, NY, USA, pp.1421–1432 (2018). doi:10.1145/3183713.3190654 .
3. Chehal, D, Bhardwaj, H.: A Tool To Convert ER Diagram To Property Graph Database. International Journal of Applied Engineering Research (2015). doi:10(9):23207-21.
4. Codd, E.: A relational model of data for large shared data banks. Commun. ACM 13, 6,pp. 377–387 (June 1970).https://doi.org/10.1145/362384.362685
5. Francis, N., Green, A., Guagliardo, P., Libkin, L., Lindaaker, T., Marsault, V., et al. : Cypher: An Evolving Query Language for Property Graphs. In Proceedings of the 2018 International Conference on Management of Data (SIGMOD '18). Association for Computing Machinery,NewYork,NY,USA,pp. 1433-1445. doi:https://doi.org/10.1145/3183713.3190657.
6. Lu, J.,Holubová, I.: Multi-model Databases: A New Journey to Handle the Variety of Data. ACM Comput. Surv. 52, 3, Article 55, 38 pages (July 2019). https://doi.org/10.1145/3323214
7. Lee, S., Park, B. H., Lim, S., Shankar, M. : Table2Graph: A Scalable Graph Construction from Relational Tables Using Map-Reduce. IEEE First International Conference on Big Data Computing Service and Applications, Redwood City, CA, 2015, pp. 294-301, doi:10.1109/BigDataService.2015.52.
8. Marton, J., Szárnyas, G., Varro, D.: Formalising openCypher Graph Queries in Relational Algebra. Advances in Databases and Information Systems, Springer International Publishing, pp. 182-196 (2017). doi:10.1007/978-3-319-66917-5_13.
9. Nan, Z., Xue, B.: The study on data migration from relational database to graph database. Journal of Physics: Conference Series, vol. 1345, pp. 022061 (2019). doi:10.1088/1742-6596/1345/2/022061.
10. Pokorný, J. : Integration of Relational and Graph Databases Functionally. Foundations of Computing and Decision Sciences, 44(4), pp 427-441 (2019). doi:10.2478/fcds-2019-0021.
11. Serin, F., Mete, S., Gül, M., Celik, E. : Mapping Between Relational Database Management Systems And Graph Database For Public Transportation Network. Conference paper, 21st International Research/Expert Conference: Trends in the Development of Machinery and Associated Technology (2018).
12. Steer, B., Alnaimi, A., Lotz, M., Cuadrado, F., Vaquero, L., Varvenne, J. Cytosm: Declarative Property Graph Queries Without Data Migration. In Proceedings of

the Fifth International Workshop on Graph Data-management Experiences & Systems. Association for Computing Machinery, New York, NY, USA, Article 4, 1–6, (2017). https://doi.org/10.1145/3078447.3078451
13. Ünal, Y. , Oğuztüzün, H.: Migration of data from relational database to graph database. In Proceedings of the 8th International Conference on Information Systems and Technologies. Association for Computing Machinery, New York, NY, USA, Article 6, 1–5 (2018). https://doi.org/10.1145/3200842.3200852
14. Virgilio, R., Maccioni, A., Torlone, R.: Converting relational to graph databases. In First International Workshop on Graph Data Management Experiences and Systems. Association for Computing Machinery, New York, NY, USA, Article 1, 1–6 (2013). https://doi.org/10.1145/2484425.2484426
15. Tutorial: Import Relational Data Into Neo4j, https://neo4j.com/developer/guide-importing-data-and-etl/
16. SQL ... and now GQL, https://www.opencypher.org/articles/2019/09/12/SQL-and-now-GQL/
17. GQL Standard, https://www.gqlstandards.org