DOI: 10.25728/mlsd.2023.1337

ИНФОРМАЦИОННЫЕ СИСТЕМЫ НА ОСНОВЕ РЕЛЯЦИОННЫХ И ГРАФОВЫХ БАЗ ДАННЫХ

Орлов В.Л., Курако Е.А.
Институт проблем управления им. В.А. Трапезникова РАН, Москва, Россия
ovl@ipu.ru, kea@ipu.ru

Аннотация. Рассматривается вопрос создания для информационных систем реляционных и графовых баз данных. Кратко определяется методика графического построения схем данных различного типа с указанием программных средств, применяемых для этой цели. Задаются методы, использующиеся для создания реальных баз данных на основе созданных схем.

Ключевые слова: реляционная база данных, графовая база данных, информационная система, онтология.

Введение

В настоящее время при построении информационных систем наряду с широко распространенными реляционными базами данных (БД) начинают применяться графовые БД. Основное отличие между этими базами данных состоит в том, что в основе реляционных баз лежит совокупность таблиц-сущностей, а в основе графовых баз – набор узлов-сущностей, объединяемых ребрами [1]. То есть в реляционных базах данных организация связей между таблицами осуществляется либо с использованием ключей, либо с помощью дополнительных таблиц. В частности, связь «многие ко многим» обеспечивается построением дополнительной таблицы. Например, множество персон, работающих в различных организациях, объединяются связывающей таблицей «Связь персона-организация» (рис. 1).

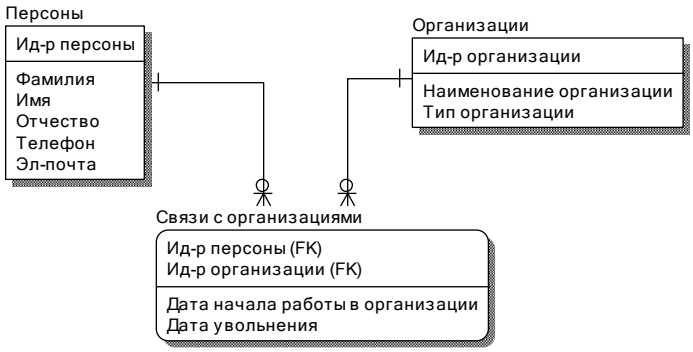

Рис. 1. Связь «персона-организация» в реляционной БД

В графовых базах данных связь организуется непосредственно построением ребер, которые по существу представляют собой связи (рис. 2).

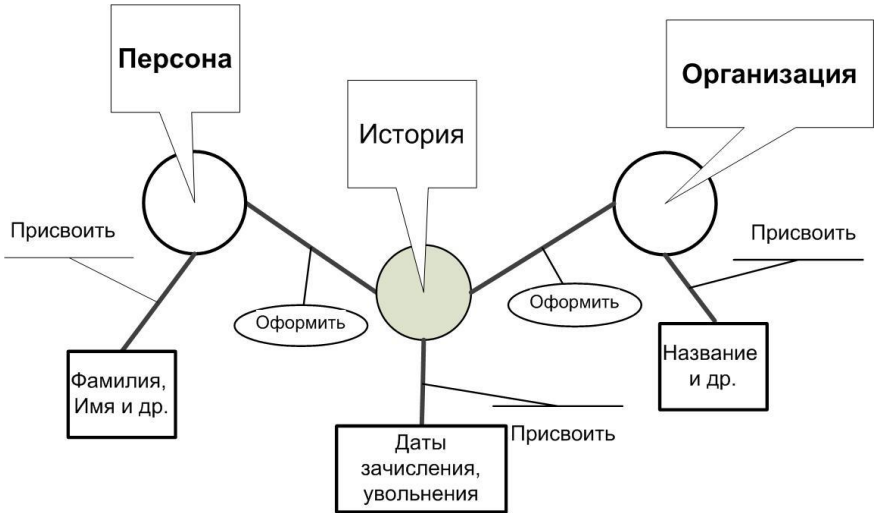

Рис. 2. Связь «персона-организация» для онтологии графовой БД

На приведенном выше фрагменте онтологии (рис.1) представлены шесть вершин и пять дуг. Вершины имеют вид окружностей и прямоугольников. Если окружности – это классы объектов, то прямоугольники – это объекты, ориентированные на определенные типы данных. Они принимают конкретные значения, например, даты зачисления и увольнения, фамилии, названия и др.

Отметим, что при построении онтологической схемы предикаты (ребра) и соединяемые ими объекты (вершины) образуют триплеты [2], совокупность которых лежит в основе конкретной графовой базы данных.

1. Построение схем баз данных

В качестве примера рассмотрим построение схем баз данных для системы учета публикаций в нескольких организациях. При построении схемы БД учитывается публикации (книги, статьи, доклады и др.), а также персоны, подготавливающие эти публикации, и организации, с которыми аффилированы данные персоны. Кроме того, учитывается множество вспомогательных данных, например, подразделения, месторасположение организаций, виды, источники публикаций и другие параметры.

1.1. Схема реляционной БД

Схема реляционной БД для системы учета публикаций представлена в виде двух изображений. На первом (рис.3) в основном представлены таблицы, отображающие взаимосвязь персон, организаций и данных, связанных с ними.

Рис. 3. Схема реляционной БД (часть 1)

Во втором изображении (рис.4) отражены таблицы, отображающие связь публикаций и персон, а также таблицы, содержащие данные по связанным с ними параметрами, таким, как ученые степени, научные должности персон, источники публикаций, издательства, их местонахождение, типы и индексы источников и другие.

Схемы могут быть изображены с помощью различных средств. Например, в данном случае они сформированы с использованием методов, предоставляемых программами типа ERWIN. Удобство использования средств такого вида дает возможность после графического построения базы данных и соответствующей настройки провести генерацию в автоматическом режиме сценария (набора скриптов) с использованием языка SQL для создания реальной БД.

Рис. 4. Схема реляционной БД (часть 2)

1.2. Схема графовой БД

Отметим, что схема графовой БД как таковая не строится. Но вместе с тем для создания реальной БД необходимо иметь основу. Такой основой может быть онтология [2]. Онтология разрабатывается вручную или с использованием специальных программ. Одной из наиболее известных программ является Protégé [3]. Упрощенный граф онтологии учета публикаций, построенный с использованием Protégé, представлен на рис.5.

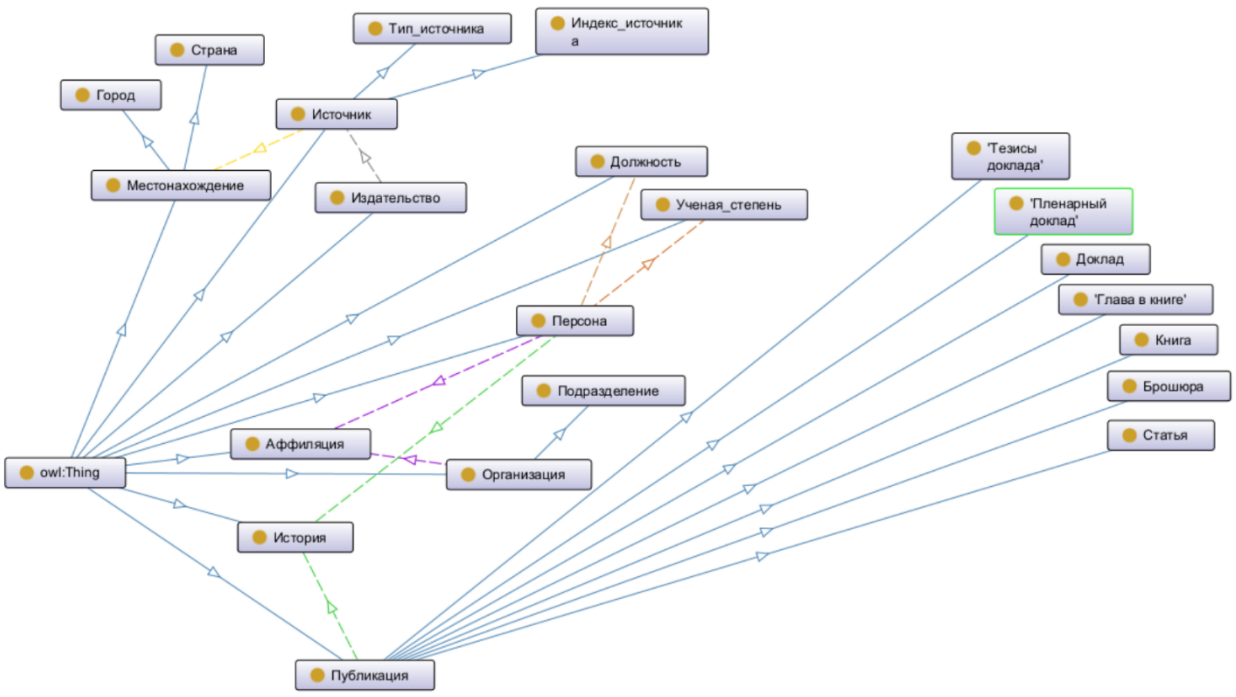

Рис. 5. Пример графического изображения онтологии учета публикаций

2. Загрузка данных

2.1. Загрузка данных в реляционную БД

Загрузка данных в реляционную БД (PostgreSQL, MySQL, Oracle) осуществляется с использованием языка SQL на основе предоставляемых данных. Обычно процедуры первоначальной и текущей загрузок являются частью разрабатываемой информационной системы.

При этом следует понимать, что разработка этих процедур и формирование SQL-запросов реализуется на основании схемы БД, содержащей описание таблиц и связей между ними, с учетом ранее заполненных справочников.

2.2. Загрузка данных в графовую БД

Загрузка данных в графовую БД в настоящее время является более сложным процессом. Для работы с графовыми БД существует ряд систем управления (СУБД), в том числе Neo4j, Virtuso, GraphDB и другие. Среди этого многообразия можно воспользоваться как платными системами с поддержкой, так и бесплатными или полностью открытыми.

Для тестирования процессов создания и загрузки информации на начальном этапе целесообразно использовать СУБД GraphDB, созданной компанией Ontotext. Эта СУБД имеет бесплатный вариант, с несущественными для тестовых задач ограничениями: не более двух запросов, выполняющихся параллельно без очередей, недоступность некоторых некоторые коннекторов, работающих в платной версии, и невозможность создания высокопроизводительного кластера. При этом данную СУБД отличает хорошая документированность и стабильность работы.

При инсталляции средств GraphDB дополнительно устанавливается программное обеспечение, представляющее собой административный сайт, позволяющей работать с данной базой путем использования веб-браузера. Пользователь на локальном сайте может как администрировать, так и выполнять запросы на языке SPARQL. Так же сайт позволяет визуализировать хранимую информацию в виде графа.

Взаимодействие разрабатываемого программного обеспечения с графовой базой предполагается по отдельному порту (по умолчанию 7200) с использованием протокола HTTP.

Если внешней программе необходимо получать данные из БД, ей потребуется организовать с помощью метода POST отправку HTTP запроса по адресу сервера вида «http://<server_name>:7200/repositories/<repository_name>.

Здесь:

<server_name> - это имя сервера, на котором расположена БД GraphDB;

<repository_name> - это имя репозитория, который содержит данные (фактически имя БД).

В теле запроса необходимо передать сам запрос:

query=<SPARQL>, где <SPARQL> - запрос с экранированными символами на языке SPARQL.

Такой способ не удобен, поэтому лучше воспользоваться готовыми библиотеками для взаимодействия с графовой БД. Есть библиотеки, созданные специально под конкретную СУБД. Для GraphDB создано несколько библиотек для языка Python, например, gtraphdb. К сожалению, они находятся на начальной стадии развития, и на момент тестирования не все функции были доступны. В тоже время есть общие библиотеки для работы с графовыми базами, которые устойчиво работают с БД GraphDB. Так как для взаимодействия используется язык SPARQL, рекомендованный консорциумом W3C, то в дальнейшем можно будет поменять СУБД на другую без замены исходного кода внешней программы. Что является несомненным плюсом.

Одной из таких библиотек является SPARQLWrapper, которая входит в семейство пакетов RDFlib для работы с графовыми базами. Этот пакет разработан для языка программирования Python.

В качестве примера приведем вызов запроса, которое ищет первые 100 публикаций, выполнение которого можно организовать следующим образом:

```
from SPARQLWrapper import SPARQLWrapper, JSON
sparql = SPARQLWrapper("http://www.ipu.ru:7200/repositories/publications")
sparql.setQuery("""
    PREFIX : <https://www.ipu.ru/ontologies/publications#>
    select ?publ where {
          ?publ a :Публикация.
          ?publ :опубликовано ?hist .
```

```
            ?hist :датаПубликации ?date
        } order by ?date limit 100
        """)
    sparql.setReturnFormat(JSON)
    results = sparql.query().convert()
```

Для библиотеки важно указать URL-адрес, определяющий запущенную базу, и текст SPARQL-запроса. Библиотека выполнит запрос и вернет массив данных в указанном формате. В данном случае формат возвращаемых данных – JSON.

С использованием этой библиотеки информацию в БД можно как получать, так и изменять. Понятно, что с малым объемом данных это не вызовет сложностей, но если требуется внести огромное количество информации, то потребуется другой способ.

Этот способ определяет механизм массового импорта данных. Он позволяет загрузить информацию, записанную в каком-нибудь RDF-формате, в том числе в формате turtle или ovl. Эти форматы поддерживает любое ПО для работы со знаниями, представленными в графовом виде. Таким образом, используя готовые библиотеки можно загружаемую информацию сохранить в файл, например, *.ttl. И затем импортировать его в БД.

Основная библиотека RDFlib позволяет создать графовое хранилище в памяти компьютера. То есть мы можем взять подготовленный с использованием комплекса Protege файл с описанием графовой БД, загрузить его в базу, затем подготовить и внести данные в соответствии с описанием. Созданную базу можно сохранить как ttl-файл. И уже этот ttl-файл можно загрузить в базу с помощью административного сайта или программного запроса по порту 7200.

Как пример можно рассмотреть загрузку данных об авторе в графовое хранилище (RDF-граф), организованное библиотекой RDFlib.

```
# Создание объекта RDF Graph
g = Graph()
g.parse("file_onto.ttl", format="ttl")  # файл из Protege

# bind the : namespace to a prefix for more readable output
g.bind("", "http://www.ipu.ru/ontologies/publications")
ipu_ns = Namespace("http://www.ipu.ru/ontologies/publications")
…
# Заносим данные персоны
person_id += 1 # Простой счетчик для идентификатора
p_iri = ipu_ns[f"author_{person_id}"]
g.add((p_iri, RDF.type, isand_ns.Персона))
g.set((p_iri, isand_ns.nid, Literal(person_id, datatype=XSD.int)))
g.set((p_iri, isand_ns.фамилия, Literal(p_last)))
if p_first is not None:
    g.set((p_iri, isand_ns.имя, Literal(p_first)))
if p_patr is not None:
    g.set((p_iri, isand_ns.отчество, Literal(p_patr)))
g.set((p_iri, RDFS.label, Literal(p_last+' '+p_first+' '+p_patr)))
if p_email_list is not None:
    for p_email in p_email_list:
        g.add((p_iri, isand_ns.email, Literal(p_email)))
```

Этот пример демонстрирует, как создать RDF-граф и заполнить его данными о каком-либо авторе. Здесь автор, согласно модели из Protege, имеет фамилию (программное обозначение - p_last), имя (p_first) отчество (p_patr) и список email-адресов (p_email_list).

Таким образом получается для полноценной работы с информацией в графовом виде есть средства проектирования, средства организации хранения и программные библиотеки, обеспечивающие удобство создания программ.

3. Заключение

Следует отдавать себе отчет, что базы данных, построенные по реляционным схемам не соответствуют базам знаний, которые организованы на основании онтологического моделирования. Основная причина заключается в том, что на основе реляционных баз данных удобно проводить вычисления, а на основании баз знаний можно делать кроме того те или иные выводы с учетом логики, в них присутствующей. В частном случае, когда логика сведена к минимуму, а такие случаи возможны, базы двух типов фактически могут представлять одно и то же информационное поле и обеспечивать выдачу идентичных результатов.

В настоящей работе показано, что построение двух типов баз возможно уже сейчас и требуется проведение дальнейших исследований, уточняющих условия эффективного решения задач при определенных условиях как тем, так и другим способом. Вместе с тем нужно определить параметры, влияющие на области применения различных средств, и критерии, определяющие эффективность такого применения.

Литература

1. *Палагашвили А., Ступников С.* Взаимообратное отображение реляционных и графовых баз данных. [Электронный ресурс] // XXIV International Conference «Data Analytics and Management in Data Intensive Domains» conference (DAMDID), 2022. URL: https://damdid2022.frccsc.ru/files/article/DAMDID_2022_paper_9907.pdf (дата обращения 15.05.2023)
2. *Горшков С.* Введение в онтологическое моделирование. – ООО «ТриниДата», 2014-2018. – 150с.
3. A free, open-source ontology editor and framework for building intelligent systems. - [Электронный ресурс] // The Board of Trustees of the Leland Stanford Junior University. URL: https://protege.stanford.edu (дата обращения 15.05.2023)