

Использование графовых баз данных для моделирования сложных информационных систем

Лапушкина Елена Павловна

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В статье рассматривается применение графовых баз данных для моделирования сложных информационных систем. Описана модель графа свойств, лежащая в основе современных графовых СУБД, и её преимущества перед реляционным подходом при работе с сильносвязанными данными. Проведено сравнение графового и реляционного подходов при выполнении многошаговых запросов по цепочкам зависимостей, которое показывает существенное упрощение и ускорение таких запросов при использовании графовой модели.

Ключевые слова: графовые базы данных, графовая модель, анализ зависимостей.

The use of graph databases for modeling complex information systems

Lapushkina Elena Pavlovna

Sholom-Aleichem Priamursky State University

Student

Abstract

The article examines the use of graph databases for modeling complex information systems. It describes the property graph model that underlies modern graph DBMSs and its advantages over the relational approach when working with highly interconnected data. A comparison of the graph and relational approaches is carried out when performing multi-step queries along chains of dependencies, which demonstrates a significant simplification and acceleration of such queries when using the graph model.

Keywords: Graph databases, graph model, dependency analysis.

1 Введение

1.1 Актуальность

Современные информационные системы предприятий представляют собой сложные комплексы взаимосвязанных компонентов: серверов, баз данных, внешних API, пользователей и бизнес-процессов, между которыми существует множество разных и часто изменяющихся связей. Реляционная модель данных, ориентированная на хранение информации в виде таблиц, при работе с такими сильносвязанными предметными областями требует

выполнения большого числа операций соединения таблиц, что приводит к усложнению запросов и снижению производительности при увеличении глубины связей между сущностями. В этих условиях особую значимость приобретают графовые базы данных, в которых связи между сущностями хранятся как элементы данных первого класса, что позволяет естественно моделировать сложные сети зависимостей и эффективно выполнять запросы, требующие обхода множества связанных объектов.

1.2 Обзор исследований

R. Angles и C. Gutierrez систематизировали историю развития графовых моделей данных начиная и предложили формальное определение графовой db-модели как совокупности структур данных, операторов и ограничений целостности [1]. R. Angles провёл систематическое сравнение современных графовых СУБД по критериям хранения и запросов данных, моделирования структур и поддержки ограничений целостности, доказав, что большинство промышленных графовых СУБД не реализуют формальную графовую алгебру и ограничения целостности в полном объёме [2]. Г. Е. Засядко и А. В. Карпов в своем исследовании проанализировали проблемы разработки графовых баз данных в условиях быстрого роста объёмов генерируемых данных и поставили задачу разработки нового способа представления графов в памяти компьютера, обеспечивающего более эффективное индексирование связей [3]. В. Ф. Барабанов провёл сравнительный обзор графовых и мультимодальных СУБД и на основе анализа их достоинств и недостатков обосновал предпосылки к разработке специализированной метаграфовой СУБД для хранения многомерных структур данных [4]. А. В. Маркин в учебном пособии по системам графовых баз данных на примере СУБД Neo4j подробно рассмотрел методику проектирования графовых информационных систем, включая практическое применение методов теории графов алгоритмов центральности, ранжирования и поиска максимального потока для анализа структуры данных [5].

1.3 Цель исследования

Цель исследования – разработка архитектуры графовой модели данных для представления сложной информационной системы, а также сравнительная оценка эффективности графового и реляционного подходов при выполнении запросов, требующих анализа многошаговых цепочек зависимостей между компонентами системы.

2 Материалы и методы

В исследовании применялись программные средства и компоненты: графовая СУБД Neo4j 5.x Community Edition, реализующая модель графа свойств, с использованием языка запросов Cypher, для программного доступа к базе данных применялись Python 3.11 и официальный драйвер neo4j.

3 Результаты

В основе каждой графовой модели данных лежит модель графа свойств, в которой каждый узел обладает набором меток, определяющих его тип, и набором пар «ключ – значение», описывающих его атрибуты. Связи между узлами являются направленными и также могут содержать собственные свойства. Такая структура позволяет напрямую отображать сущности предметной области и связи между ними без промежуточных таблиц соответствия. Основные элементы модели графа свойств, применённые для представления ИТ-инфраструктуры, приведены в таблице 1.

Таблица 1 – Элементы модели графа свойств

Элемент модели	Назначение	Пример в предметной области
Узел (Node)	Представляет сущность предметной области	Service, Server, Database
Метка (Label)	Определяет тип узла	:Service, :Server, :Database
Свойство (Property)	Атрибут узла или связи	name, environment, load
Связь (Relationship)	Направленное типизированное отношение	DEPENDS_ON, HOSTED_ON

На основе модели графа свойств была спроектирована схема данных ИТ-инфраструктуры: узлы с метками Service, Server и Database, а также связи DEPENDS_ON, HOSTED_ON и USES_DB. Фрагмент кода на языке Cypher, создающий ограничения уникальности и тестовый набор узлов и связей (рис. 1).

```
// Ограничения уникальности идентификаторов
CREATE CONSTRAINT service_id IF NOT EXISTS
FOR (s:Service) REQUIRE s.id IS UNIQUE;

CREATE CONSTRAINT server_id IF NOT EXISTS
FOR (sv:Server) REQUIRE sv.id IS UNIQUE;

CREATE CONSTRAINT db_id IF NOT EXISTS
FOR (d:Database) REQUIRE d.id IS UNIQUE;

// Создание узлов
CREATE (auth:Service {
  id: 'svc-auth', name: 'Auth Service'
});
CREATE (orders:Service {
  id: 'svc-orders', name: 'Orders Service'
});
CREATE (srv1:Server {
  id: 'srv-01', dc: 'msk-1'
});
CREATE (pgdb:Database {
  id: 'db-orders', engine: 'PostgreSQL'
});

// Создание связей зависимости
MATCH (orders:Service {id: 'svc-orders'}),
      (auth:Service {id: 'svc-auth'})
CREATE (orders)-[:DEPENDS_ON]->(auth);

MATCH (orders:Service {id: 'svc-orders'}),
      (srv1:Server {id: 'srv-01'})
CREATE (orders)-[:HOSTED_ON]->(srv1);

MATCH (orders:Service {id: 'svc-orders'}),
      (pgdb:Database {id: 'db-orders'})
CREATE (orders)-[:USES_DB]->(pgdb);
```

Рисунок 1 – Создание схемы графа и загрузка тестовых данных на Cypher

Далее ключевое преимущество графовой модели проявляется при выполнении запросов, требующих обхода цепочки связей произвольной длины. В реляционной модели такой запрос требует рекурсивного соединения таблиц, так в Cypher он выражается с помощью синтаксиса переменной длины пути (рис. 2).

```
1 MATCH (affected:Service)
2   -[:DEPENDS_ON*0..5]->(s:Service)
3   -[:USES_DB]->(db:Database {id: 'db-orders'})
4 RETURN DISTINCT affected.name AS service,
5   affected.id AS service_id
6 ORDER BY service;
7
8 MATCH path = shortestPath(
9   (a:Service {id: 'svc-orders'})
10  -[:DEPENDS_ON*]->
11  (b:Service {id: 'svc-auth'})
12 )
13 RETURN [n IN nodes(path) | n.name] AS chain,
14   length(path) AS hops;
```

Рисунок 2 – Cypher-запросы

Для интеграции графовой модели с внешними приложениями использован официальный драйвер neo4j для Python (рис. 3).

```

from neo4j import GraphDatabase

URI = 'bolt://localhost:7687'
AUTH = ('neo4j', 'password')

driver = GraphDatabase.driver(URI, auth=AUTH)

IMPACT_QUERY = """
MATCH (affected:Service)
-[:DEPENDS_ON*0..5]->(s:Service)
-[:USES_DB]->(db:Database {id: $db_id})
RETURN DISTINCT affected.name AS service
ORDER BY service
"""

def get_affected_services(db_id: str):
    with driver.session() as session:
        result = session.run(
            IMPACT_QUERY, db_id=db_id
        )
        return [record['service']
                for record in result]

services = get_affected_services('db-orders')
print(services)

```

Рисунок 3 – Программный доступ к графовой базе данных из Python

По данным результатам сравнительного тестирования 20 запросов на построение цепочек зависимостей различной длины на идентичном наборе данных получены показатели (табл. 2).

Таблица 2 – Сравнение времени выполнения запросов по цепочкам зависимостей разной длины пилотное тестирование

Глубина цепочки	SQL: JOIN-ов в запросе	SQL: среднее время, мс	Cypher: среднее время, мс
2 перехода	2	8	5
4 перехода	4	34	9
6 переходов	6	142	14

Различия объясняются тем, что в реляционной СУБД каждый дополнительный переход по цепочке зависимостей требует дополнительной операции соединения таблиц с ростом вычислительной стоимости, тогда как в графовой СУБД связи физически хранятся как указатели, привязанные к узлам, что делает стоимость обхода очередного перехода практически независимой от общего объёма данных в базе.

4 Выводы

В результате исследования спроектирована и реализована графовая модель сложной информационной системы на основе модели графа свойств и графовой СУБД Neo4j, включающая схему данных, набор Cypher-запросов для анализа зависимостей и программный интерфейс доступа из Python. Пилотное

сравнительное тестирование показало, что графовый подход обеспечивает существенно более простое и быстрое выполнение запросов, требующих анализа многошаговых цепочек зависимостей между компонентами системы, по сравнению с реляционным подходом, преимущество которого возрастает с увеличением глубины анализируемых связей.

Библиографический список

1. Angles R., Gutierrez C. Survey of Graph Database Models // ACM Computing Surveys. 2008. Vol. 40, №1. Article 1. P. 1–39. URL: <https://dl.acm.org/doi/10.1145/1322432.1322433>
2. Angles R. A Comparison of Current Graph Database Models // 2012 IEEE 28th International Conference on Data Engineering Workshops (ICDEW). IEEE, 2012. P. 171–177. URL: <https://ieeexplore.ieee.org/document/6313676>
3. Засядко Г. Е., Карпов А. В. Проблемы разработки графовых баз данных // Инженерный вестник Дона. 2017. №1. URL: <http://www.ivdon.ru/ru/magazine/archive/n1y2017/3994>
4. Барабанов В. Ф. Анализ графовых систем управления базами данных // Вестник Воронежского государственного технического университета. Т. 22, №1. С. 45–50. URL: <https://cyberleninka.ru/article/n/analiz-grafovyyh-sistem-upravleniya-bazami-dannyh>
5. Маркин А. В. Системы графовых баз данных. Neo4j: учебное пособие для вузов. М.: Издательство Юрайт, 2021. 178 с. URL: <https://urait.ru/book/sistemy-grafovyyh-baz-dannyh-neo4j-467452>