Федеральное государственное автономное образовательное учреждение высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

**Факультет информатики, математики и компьютерных наук**

**Доклад**

Графовая база данных NoSQL– Neo4j

Название темы

|  |  |
| --- | --- |
|  | Выполнили: |
|  | Студенты группы 23КНТ-5  Дашинимаева В.Б.  Мельникова Н.М. |
|  | Хлопочкина Н.В.  Ф.И.О. |
|  |

Нижний Новгород 2025

**Содержание**

[1. Введение 3](#_Toc192974975)

[2. История создания 3](#_Toc192974976)

[3. Основные особенности 3](#_Toc192974977)

[4. Обоснование принадлежности к NoSQL 4](#_Toc192974978)

[5. Преимущества Neo4j в сравнении с классическим SQL 4](#_Toc192974979)

[6. Недостатки Neo4 в сравнении с классическим SQL 5](#_Toc192974980)

[7. Основные функциональные возможности Neo4j 5](#_Toc192974981)

[8. Возможность интеграции со сторонними приложениями и требования 7](#_Toc192974982)

[9. Вывод 8](#_Toc192974983)

# **Введение**

Neo4j – это графовая база данных NoSQL с открытым исходным кодом, которая использует полнофункциональную модель данных узлов, связанных полноправными отношениями, что лучше подходит для связанных больших данных, чем традиционные подходы с использованием реляционных СУБД.

# **История создания**

История Neo4j начинается с идеи, возникшей в 2000 году у Эмиля Эйфрем, Йохана Свенссона и Питера Нойбауэра. Они работали над системой управления контентом и осознали ограничения реляционных баз данных при работе с сильно связанными данными.

Первый публичный выпуск Neo4j состоялся в 2007 году.

В 2011 году был представлен язык запросов Cypher, связанный с Neo4j. Он был разработан для упрощения работы с графовыми данными и вдохновлён SQL.

В 2019 году вышел релиз Neo4j 4.0, который привнёс значительные улучшения, включая новый механизм хранения, улучшенную масштабируемость и расширенные функции безопасности.

С 2010-х годов и до настоящего времени Neo4j продолжает активно развиваться, с регулярными выпусками, которые вводят новые функции, улучшают производительность и оптимизируют систему.

В июне 2022 года компания привлекла около 325 млн долларов в рамках раунда финансирования Series F, что увеличило оценку Neo4j до более чем 2 млрд долларов.

# **Основные особенности**

* + **Графовая модель данных**

В Neo4j данные хранятся в виде узлов (nodes), отношений (relationships) и свойств (properties), что позволяет легко представлять и анализировать взаимосвязи между объектами.

* + **Производительность**

Благодаря встроенной модели графов, Neo4j обеспечивает высокую производительность при выполнении запросов, связанных со сложными отношениями, что делает их более быстрыми по сравнению с традиционными реляционными базами данных в аналогичных сценариях.

* + **Язык запросов Cypher**

Neo4j использует собственный язык запросов, называемый Cypher, который разработан для удобного и интуитивного представления запросов к графовым данным. Cypher похож на SQL, но оптимизирован для работы с графами.

* + **Гибкость схемы**

В отличие от реляционных баз данных, где схема данных должна быть строго определена, Neo4j позволяет использовать динамическую схему, что предоставляет большую гибкость в моделировании данных.

* + **Масштабируемость**

Neo4j поддерживает горизонтальное масштабирование и может обрабатывать большие объемы данных и высокие нагрузки.

* + **Поддержка транзакций**

Как и традиционные базы данных, Neo4j поддерживает ACID-транзакции (атомарность, согласованность, изолированность, долговечность), что гарантирует надежность операций с данными.

# **Обоснование принадлежности к NoSQL**

Neo4j относится к NoSQL базам данных из-за своей графовой модели данных и гибкости схемы. NoSQL базы данных ориентированы на обработку больших объемов данных, предоставление высокой производительности и поддержку различных типов моделей данных (документы, колонки, ключ-значение, графы). В отличие от реляционных СУБД, которые используют фиксированную схему и таблицы, Neo4j обеспечивает динамическое представление данных, что характерно для NoSQL решений.

# **Преимущества Neo4j в сравнении с классическим SQL**

* **Графовая модель данных**

Neo4j использует графовую модель, которая естественным образом отображает сложные и взаимосвязанные данные, такие как социальные сети, организации, системы рекомендаций и т.д. Это позволяет легко моделировать и манипулировать отношениями между данными.

* **Производительность при обработке сложных запросов**

Для запросов с глубокими взаимосвязями Neo4j может значительно превзойти реляционные базы данных. Поиск по графу выполняется быстрее благодаря оптимизированному хранению и индексации отношений.

* **Гибкость схемы**

В отличие от реляционных баз данных, где структура данных должна быть заранее определена, Neo4j позволяет изменять схему без необходимости изменения всех связанных объектов, что дает большую гибкость в проектировании.

* **Естественное представление аналитики**

Графовые базы данных, такие как Neo4j, предоставляют возможности для выполнения сложной аналитики, такой как нахождение узлов с высокой центральностью или анализ кластеров, что может быть более сложным и ресурсозатратным в реляционных системах.

* **Cypher**

Язык запросов Cypher прост и интуитивно понятен для работы с графовыми данными. Запросы могут быть легче формулированы по сравнению с SQL для графовых операций.

# **Недостатки Neo4 в сравнении с классическим SQL**

* **Отсутствие универсальности**

Реляционные базы данных, такие как PostgreSQL или MySQL, более универсальны и лучше подходят для транзакционных систем и обработки структурированных данных, где связи не так важны.

* **Низкая производительность при простых запросах**

В случаях, когда данные не слишком взаимосвязаны, реляционные базы данных могут быть более производительными, особенно при простых запросах к таблицам.

* **Обучение и адаптация**

Переход с реляционной модели на графовую может потребовать дополнительного обучения для разработчиков и администраторов баз данных. Необходимо адаптироваться к новой модели и языку запросов.

* **Ограниченная поддержка некоторых функций**

Некоторые функции, такие как сложные транзакции или определенные типы агрегации, могут быть не так хорошо поддержаны в графовых базах данных, как в реляционных.

# **Основные функциональные возможности Neo4j**

|  |  |  |
| --- | --- | --- |
| **Функциональные возможности** | **Neo4j** | **SQL** |
| **Структуризация данных** | Данные структурируются в виде узлов (nodes), отношений (relationships) и свойств (properties). Это позволяет легко моделировать сложные взаимосвязи и иерархии, отражая реальный мир. | Использует таблицы, строки и столбцы для представления данных. Связи между таблицами обычно реализуются через внешние ключи. |
| **Язык запросов** | Cypher язык запросов, специально разработанный для работы с графами. Он позволяет легко составлять запросы для выборки, изменения и удаления данных с использованием понятного синтаксиса. | Можно использовать для манипуляции с данными в реляционных таблицах и предоставляет мощные возможности для работы с агрегированными данными, анализом и управлением транзакциями. |
| **Обработка сложных запросов** | Запросы, включающие сложные слияния и взаимосвязи, могут выполняться быстрее благодаря графовой модели. Для выполнения поиска по графу используются алгоритмы графового поиска. | При работе с сложными связями таблицы могут требовать большого числа соединений (JOIN), что может привести к снижению производительности. |
| **Индексация и производительность** | Поддерживает индексацию узлов и рёбер, что существенно ускоряет поиск узлов по атрибутам и навигацию по графу. | Индексы также поддерживаются, но их создание и обслуживание может быть сложнее, особенно при работе с множественными таблицами и сложными запросами. |
| **Гибкость схемы** | Не строгая схема; вы можете добавлять новые свойства и отношения к узлам без необходимости изменения всей структуры базы данных. | Требует более строгого определения схемы, и любые изменения могут требовать миграции данных или изменения структуры таблиц. |
| **Графовые алгоритмы** | Включает в себя встроенные графовые алгоритмы для анализа данных, такие как алгоритмы центральности, кластеризации, маршрутизации и т.д. | Эти алгоритмы не встроены; для их реализации требуется значительное количество пользовательского кода, и они могут быть менее эффективными |
| **Поддержка транзакций** | Поддерживает ACID-транзакции, что гарантирует целостность данных даже при параллельной обработке запросов. | Также поддерживает ACID-принципы, что делает реляционные базы данных надежными для транзакционных систем. |
| **Работа с временными данными** | Позволяет легко представлять временные изменения в графах через свойства или новые узлы. | Может поддерживать временные данные с помощью временных таблиц или специализированных типов данных, но это требует более сложной структуры и запросов. |

Основные функциональные возможности Neo4j основываются на графовой модели данных и языке запросов Cypher, которые делают эту базу данных особенно эффективной для работы с взаимосвязанными данными. В отличие от классических SQL-баз данных, Neo4j предлагает гибкость в схеме, производительность при работе с глубоко взаимосвязанными данными и встроенные графовые алгоритмы. Выбор между Neo4j и SQL обычно зависит от специфики проекта и структуры данных, которые необходимо обрабатывать.

# **Возможность интеграции со сторонними приложениями и требования**

Интеграция Neo4j со сторонними приложениями может быть выполнена различными способами, что обеспечивает гибкость и удобство работы с графовыми данными.

Способы интеграции:

* **Официальные драйверы Neo4j**:

Neo4j предоставляет официальные драйверы для различных языков программирования, таких как Java, Python, JavaScript, Go, .NET и другие. Использования драйвера необходимо установить соответствующий пакет, а также настроить подключение к экземпляру Neo4j (указать URI, имя пользователя и пароль). Например, для подключения через Python можно использовать библиотеку neo4j. Этот драйвер обеспечивает удобный интерфейс для выполнения запросов Cypher и работы с графовыми данными.

* **REST API**:

Neo4j предоставляет RESTful API, которое позволяет выполнять запросы и управлять графом через HTTP. Можно совершать запросы к ресурсам, используя стандартные методы GET, POST, DELETE и PUT. Чтобы использовать REST API, необходимо отправлять HTTP-запросы к серверу Neo4j по указанному URL. Для работы API может потребоваться аутентификация.

* **Bolt Protocol**:

Bolt - это бинарный протокол, оптимизированный для работы с графами. Он используется для более быстрого и эффективного соединения с Neo4j. Для его использования также необходимы соответствующие драйверы, поддерживающие Bolt. Приложения, подключающиеся к базе данных, должны обеспечить совместимость с Bolt. Например, можно использовать драйвер Neo4j для Java, который поддерживает протокол Bolt, что позволяет устанавливать соединение и выполнять запросы.

* **Библиотеки для аналитики и визуализации данных**:

Neo4j поддерживает интеграцию с библиотеками для визуализации и анализа данных, такими как D3.js, Grafana, Python библиотеки (например, Pandas и Matplotlib). Может потребоваться установка дополнительных библиотек и возможность подключения к Neo4j через драйверы или API. Пример: визуализация графов с использованием D3.js с данными, получаемыми из Neo4j через API.

* **Neo4j в экосистемах больших данных**:

Neo4j можно интегрировать с инструментами, такими как Apache Kafka, Apache Spark и другие фреймворки для обработки больших данных. Каждое решение может требовать установки плагинов или написания адаптеров для передачи данных между системами. Пример: использование Kafka для обработки потоков данных и последующего их сохранения в Neo4j.

# **Вывод**

Neo4j представляет собой решение для хранения и анализа графовых данных, которое открывает новые горизонты для аналитики и работы с взаимосвязанными данными. Однако при выборе Neo4j важно учитывать специфику проекта и оценивать, насколько графовая модель будет соответствовать требованиям. Оптимальное использование Neo4j сможет существенно повысить эффективность работы с данными и упростить процесс их анализа, особенно в тех случаях, когда связи между данными играют ключевую роль.