ПРИМЕНЕНИЕ ГРАФОВЫХ МОДЕЛЕЙ ДАННЫХ ДЛЯ E-COMMERCE-ПРИЛОЖЕНИЯ

Трошина Л.М., старший преподаватель,
Тимофеев Е.О., бакалавр,
САФУ имени М.В. Ломоносова, Россия

Аннотация: В статье рассмотрен пример реализации графовой базы данных, как дополнение функционала реляционной СУБД в вопросах задач анализа поведения клиентов интернет-магазина. Приведены особенности построения графовых схем на примере neo4j. Обосновано применение графовых моделей для формирования контекстно-ориентированных схем для повышения эффективности продаж за счёт более точного моделирования связей между товарами, пользователями и их поведением.

Ключевые слова: граф, базы данных, запросы, покупательское поведение, спрос.

Сегодня бизнесу доступны различные инструменты, которые помогают принимать обоснованные решения, повышать эффективность и укреплять отношения с аудиторией. Анализ поведения клиентов, в том числе анализ продуктовых корзин, ABC и XYZ-анализ, retention-маркетинг, персональные предложения, выполняется с помощью различных методик и используются десятилетиями. Одним из современных способов выявления тенденций и предпочтений покупателей является применение графовых подходов и построение графовой базы данных, которая позволит наглядно представить сложные взаимосвязи между действиями пользователей и выявить неочевидные паттерны.

Графовые базы данных — это не просто новый тип систем управления базами данных (СУБД), а принципиально иной способ мышления о данных. Известно, что реляционные базы хранят информацию в строгих

структурированных таблицах. Для обнаружения взаимосвязей или выявления цепочки предпочтений в классических реляционных базах данных каждый новый вопрос, требует сложных дорогостоящих JOIN-соединений множества таблиц, что сильно замедляет запрос. Графовые базы данных строятся в виде сети связанных узлов (сущностей) и рёбер (связей между ними), каждые из которых могут обладать своими свойствами, что позволяет естественным образом моделировать ситуации, где важны не только сами объекты, но и контекст их взаимодействия. Реализуется скорость на глубоких связях.

Графовые системы имеют множество других преимуществ, одно из которых – гибкость и естественность, так как графовая модель интуитивно понятна. Добавление нового типа связи не требует изменения существующей схемы, что критически важно для быстро меняющегося e-commerce-приложения. В отличие от «чёрного ящика» нейросетей, результат графового запроса легко объяснить. Всегда можно увидеть путь, по которому система пришла к тому или иному результату, то есть графы позволяют видеть полный «контекст» (напртимер, покупатель приобрел товар А, потому что он связан с его хобби Б, которое было вычислено по его прошлым просмотрам), а не просматривать сухую статистику продаж.

Одной из самых популярных графовых баз, применяемых для различных задач, является Neo4j, которая предоставляет богатый набор инструментов для аналитики (таких как, библиотека Graph Data Science для запуска алгоритмов кластеризации, поиска сообществ, расчёта центральности). Но следует понимать, что графовые базы данных — это не замена, а значительное дополнение к реляционным СУБД для решения задач, где ключевую роль играют связи. Если в реляционной схеме интернет-магазина ключевыми сущностями определить «Товар», «Категория товара», «Бренд», «Покупатель», «Заказ», то при проектировании модели данных графа можно опираться на эту же самую схему, определяя при этом взаимосвязи между нодами – вершинами графа (пример приведен на рисунке 1).

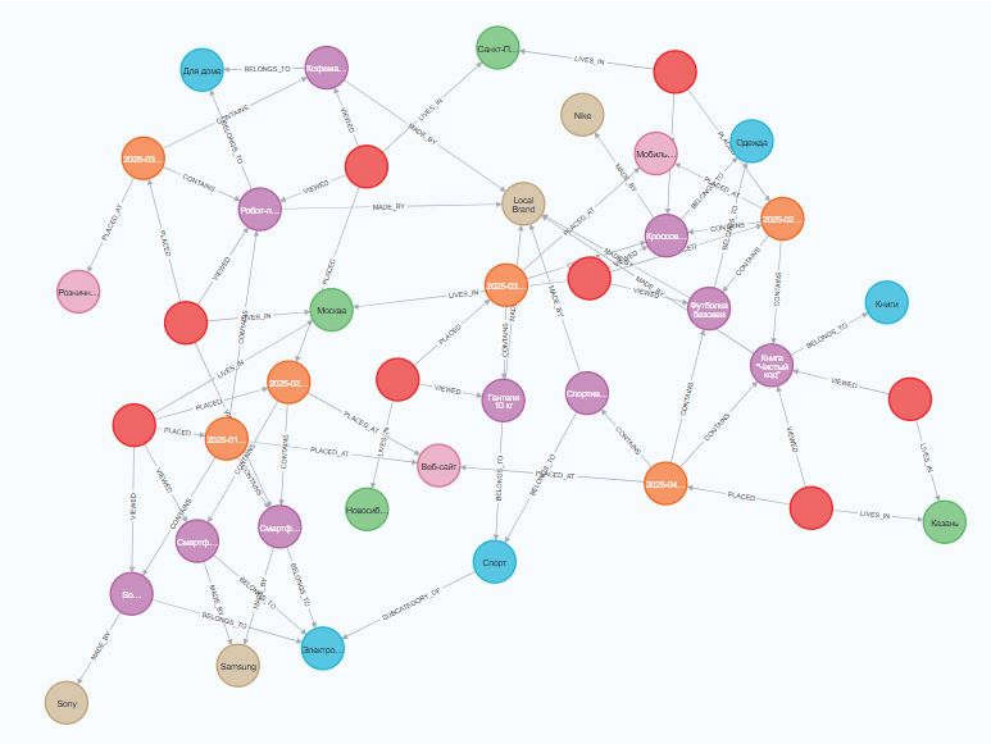


Рисунок 1. Фрагмент графовой модели

Важно прояснить, что ноды в Neo4j и таблицы в реляционной СУБД, например, PostgreSQL нельзя «связать» как одну базу с общими внешними ключами, так как это принципиально разные модели данных, и механизмы вроде FOREIGN KEY между ними не работают. PostgreSQL даёт строгие ACID-транзакции: при выполнении команды INSERT с FOREIGN KEY, база гарантирует, что целевая строка существует. Neo4j тоже даёт ACID, но он работает внутри графа. Нет единого механизма, который бы атомарно создал узел в Neo4j, строку в PostgreSQL и связал их одним внешним ключом.

Существует несколько архитектурный решений, реализующих подобную связь. Одним из них является событийная синхронизация (Event-Driven) - подход к построению распределённых систем, при котором компоненты взаимодействуют через обмен событиями — сигналами об изменении состояния [2]. Вместо прямых синхронных компонент-производитель публикует событие, а компоненты-потребители подписываются на него и асинхронно реагируют. Запрос может читать данные из определённых таблиц в PostgreSQL и сразу создавать узлы в Neo4j. Таким образом, физической связи между объектами в разных СУБД не существует, она определена только в

бизнес-логике [1]. Условная схема событийной синхронизации приведена на рисунке 2.

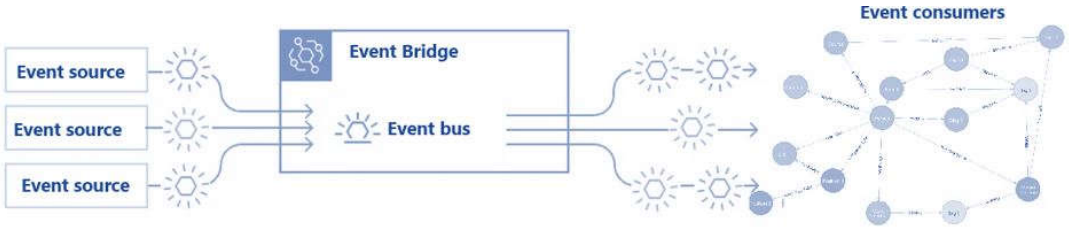


Рисунок 2. Схема событийно-ориентированной архитектуры

Кроме прочего, для описания паттерна графа Neo4j применяется собственный декларативный язык запросов Cypher, который использует синтаксис, похожий на ASCII-графику, что позволяет разработчику буквально «нарисовать» то, что он ищет в базе данных. Узлы здесь обозначаются в круглых скобках: (n), (person:Person); связи - с помощью стрелок и квадратных скобок: -[r]->, -[:KNOWS]->; сами паттерны комбинируются в цепочки, например: (a)-[:KNOWS]->(b). Такой подход означает, что запрос на Cypher часто выглядит как схема на доске или диаграмма связей, что делает его очень наглядным. Таким образом, чтобы, например, определить какие просмотры покупателей не привели к покупке товаров (так называемые, упущенные возможности) следует сформировать подобный запрос:

```
1 MATCH (c:Customer)-[:VIEWED]→(p:Product)
2 WHERE NOT (c)-[:PLACED]→(:Order)-[:CONTAINS]→(p)
3 RETURN c.name AS Покупатель, p.name AS Просмотренный_товар
4 LIMIT 10;
5
```

Table | Text | Code

	Покупатель	Просмотренный_товар
1	"Сергей Смирнов"	"Смартфон Galaxy S24"
2	"Дмитрий Сидоров"	"Кофемашина DeLonghi"
3	"Дмитрий Сидоров"	"Робот-пылесос Xiaomi"
4	"Ольга Новикова"	"Книга "Чистый код""

Рисунок 3. Пример запроса на языке Cypher

Таким образом, графовые модели позволяют строить различные контекстно-ориентированные схемы, которые могут учитывать не только сходство товаров, но и сложные шаблоны взаимодействия пользователей с

ними. В итоге графовая база даёт не только хранилище данных, - это удобный инструмент для глубокого анализа контекста покупательского поведения, что напрямую помогает улучшать маркетинг, повышать лояльность клиентов и увеличивать продажи товаров.

Литература

1. Событийно-ориентированная архитектура: принципы, преимущества и инструменты реализации EDA [Электронный ресурс] // Yandex Cloud. – URL: https://yandex.cloud/ru/blog/event-driven-architecture (дата обращения: 29.10.2026).
2. Building a Knowledge Base Service With Neo4j, Kafka, and the Outbox Pattern [Electronic resource] // dev.to. – URL: https://dev.to/thegoncalomartins/building-a-knowledge-base-service-with-neo4j-kafka-and-the-outbox-pattern-460b (дата обращения: 29.10.2026).
3. PostgreSQL vs Neo4j: Choosing the Right Database for Your Project [Electronic resource] // dev.to. – URL: https://dev.to/pawnsapprentice/postgresql-vs-neo4j-choosing-the-right-database-for-your-project-1o59 (дата обращения: 29.10.2026).