РЕЛЯЦИОННЫЙ И НЕРЕЛЯЦИОННЫЙ ПОДХОДЫ К РЕАЛИЗАЦИИ БАЗ ДАННЫХ

РИМСКИЙ Денис

Технический Университет Молдовы

***Аннотация**: Статья посвящена анализу двух различных подходов к реализации баз данных: реляционный и нереляционный. Показаны преимущества и недостатки каждого из подходов. Дано описание основных трендов в этой области в настоящее время.*

***Ключевые слова:** СУБД, SQL, NoSQL, сравнение подходов.*

1. NoSQL

NoSQL (англ. not only SQL, не только SQL) - обозначает ряд подходов, проектов, направленных на реализацию моделей баз данных, имеющих существенные отличия от используемых в традиционных реляционных СУБД с доступом к данным средствами языка SQL. Описание схемы данных в случае использования NoSQL-решений может осуществляться через использование различных структур данных: хеш-таблиц, деревьев и других.

Основная идея при разработке такого типа систем — предоставление разработчикам более гибких средств по модификации схемы данных на этапе эксплуатации, отказе от транзакционного контроля в пользу более естественной поддержки распределённых вычислений.

Сторонниками концепции NoSQL подчёркивается, что она не является полным отрицанием языка SQL и реляционной модели, проект исходит из того, что SQL — это важный и весьма полезный инструмент, но при этом он не может считаться универсальным. Одной из проблем, которую указывают для классических реляционных БД, являются проблемы при работе с данными очень большого объема и в проектах с высокой нагрузкой. Основная цель подхода — расширить возможности БД там, где SQL недостаточно гибок, и не вытеснять его там, где он справляется со своими задачами. Пример реляционной модели можно увидеть на рисунке 1.

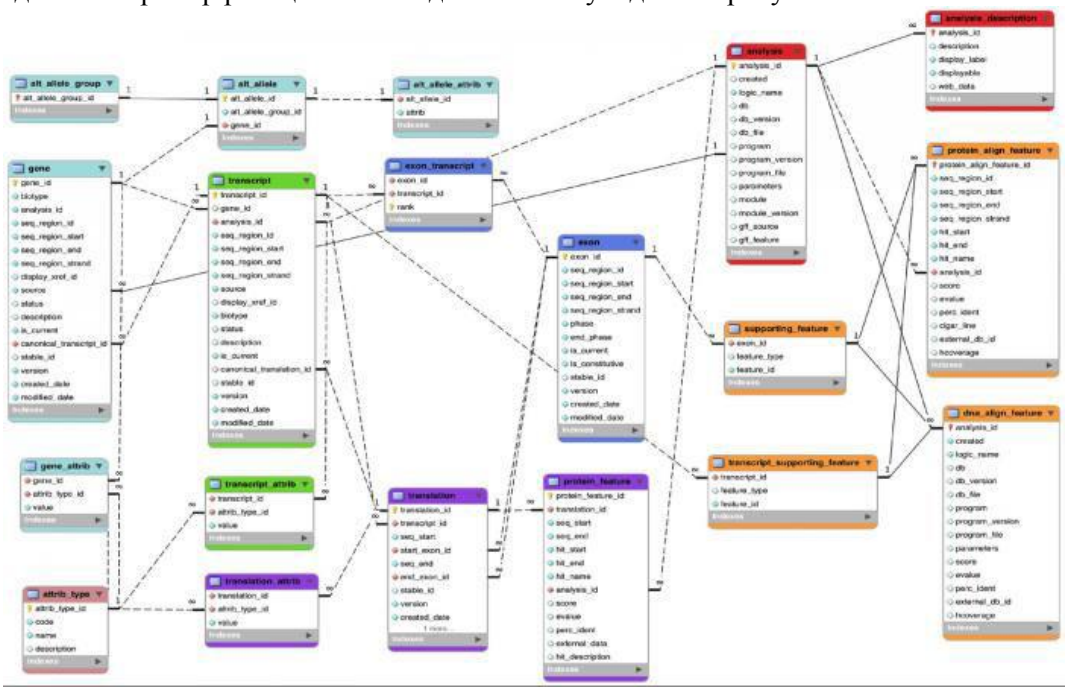

Рис. 1. Пример реляционной модели

2. Тренды в развитии технологий хранения данных

В последнее время мы можем наблюдать 4 тренда в развитии технологий хранения данных:

1) Увеличение объёмов данных

Сегодня хранилища достигли таких невероятных размеров, которые даже трудно себе представить.

2) Взаимосвязанность данных

Информация перестала быть изолированной. Каждый кусочек знаний как-то связан с данными в других хранилищах информации. Страницы в интернете ссылаются на другие страницы. Тэги связывают помеченную информацию из разных источников. Онтологии устанавливают взаимосвязи между различными терминами.

3) Использование слабоструктурированной информации

Возьмем простой пример: описание товара в магазине. Если раньше было достаточно 5-6 полей, чтобы описать мужскую сорочку (размер, цвет, материал, фотография товара, ...), то теперь количество параметров может доходить до нескольких десятков. Причем, для разных сорочек будет использован разный набор параметров. В таких условиях становится крайне сложно заранее определить структуру таблицы, в которой хранится описание товара.

4) Изменение архитектуры.

В 80-х годах прошлого века типичная архитектура использовала один большой компьютер (mainframe) и одну базу данных. В 90-х, распространение получила клиент-серверная архитектура. В новом веке активно используются web-сервисы, каждый со своим backend-ом (со своей базой данных) и другие распределенные решения. Оказалось, что в таких условиях у реляционных баз данных резко падает производительность. И если для большинства web-сайтов производительности еще хватает, то для таких приложений как современные социальные сети или поисковые сервисы SQL базы данных оказались несостоятельны.

3. Категории NoSQL баз

Существует четыре категории NoSQL баз данных:

1) Key-Value Базы Данных

Это очень простые по своей идее хранилища. Фактически это очень большие хэш-таблицы, где каждому ключу поставлено в соответствие значение. Такие базы могут очень быстро оперировать колоссальными объемами информации, но они имеют серьезные ограничения в языке запросов. В качестве примеров Key-Value баз данных можно привести Dynomite, Voldemort, Tokyo, Redis.

2) Клоны BigTable

BigTable — это база данных разработанная компанией Google для собственных нужд. Эта база представляет собой большую таблицу с тремя измерениями: колонки, строки и временные метки. Такая архитектура позволяет добиться очень высокой производительности, кроме того, она хорошо масштабируется на множество компьютеров. Но это не реляционная база, и она не поддерживает многие возможности реляционных баз. В частности в BigTable нет join-ов, нет сложных запросов и т.д. Компания Google не распространяет BigTable, поэтому на рынке появилось несколько независимо разработанных клонов этой базы. В частности это такие проекты, как Hadoop, Hypertable и Cassandra.

3) Документоориентированные Базы Данных

Такие базы немного напоминают Key-Value базы, но в данном случае, база данных знает, что из себя представляют значения. Обычно, значением является некоторый документ или объект, к структуре которого можно делать запросы. Примерами таких баз являются CouchDB и MongoDB. Переход от реляционной модели к нереляционной на примере MongoDB можно увидеть на рисунке 2.

4) Базы данных построенные на графах

Такие базы ориентированы на поддержку сложных взаимосвязей между объектами, и основываются на теории графов. Структура данных в таких базах представляет собой набор узлов, связанных между собой ссылками. При этом и узлы и ссылки могут обладать некоторым количеством атрибутов. В качестве примера можно привести такие базы данных, как Neo4j, AllegroGraph, Sones graphDB.

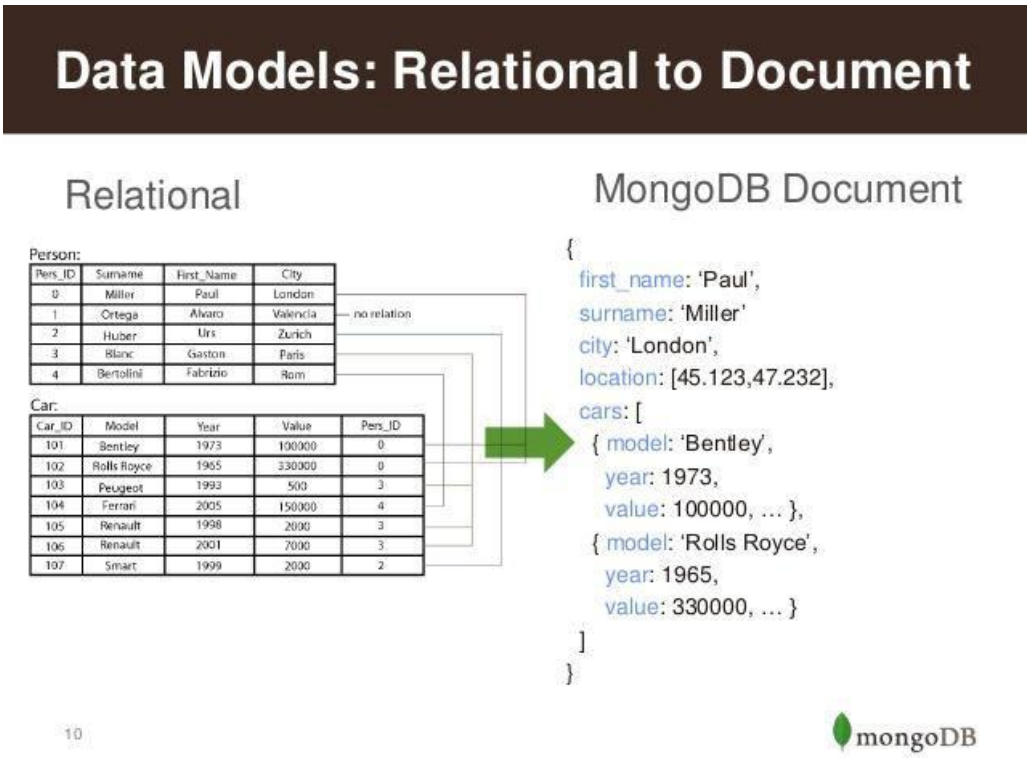

Рис. 2. Переход от реляционной модели к нереляционной на примере MongoDB

4. Заключение

В современном мире нет противостояния между реляционными и нереляционными базами данных. Вместо этого стоит говорить об их совместном использовании для решения задач, для которых та или иная технология показывает себя лучше всего. Кроме того, всё сильнее наблюдается интеграция этих технологий друг в друга. Например, Microsoft, Oracle и Teradata сейчас предлагают некоторые формы интеграции с Hadoop для подключения аналитических инструментов, основанных на SQL, к миру неструктурированных больших данных.

Вот признаки проектов, для которых идеально подойдут SQL-базы:

1) Имеются логические требования к данным, которые могут быть определены заранее.
2) Очень важна целостность данных.
3) Нужна основанная на устоявшихся стандартах, хорошо зарекомендовавшая себя технология, используя которую можно рассчитывать на большой опыт разработчиков и техническую поддержку.

А вот свойства проектов, для которых подойдёт что-то из сферы NoSQL:

1) Требования к данным нечёткие, неопределённые, или развивающиеся с развитием проекта.
2) Цель проекта может корректироваться со временем, при этом важна возможность немедленного начала разработки.
3) Одни из основных требований к базе данных — скорость обработки данных и масштабируемость.

Литература

1. SQL or NoSQL, That Is The Question! [Электронный ресурс]. -Режим доступа: https://blog.panoply.io/sql-or-nosql-that-is-the-question
2. SQL или NoSQL — вот в чём вопрос. [Электронный ресурс]. - Режим доступа: https://habrahabr.ru/company/ruvds/blog/324936/
3. SQL. [Электронный ресурс].-Режим доступа: https://ru.wikipedia.org/wiki/SQL
4. NoSQL. [Электронный ресурс].-Режим доступа: https://ru.wikipedia.org/wiki/NoSQL
5. Системы баз данных. Полный курс. [Электронный ресурс].-Режим доступа: http://padaread.com/?book=15175
6. Основы SQL. [Электронный ресурс].-Режим доступа:https://www.intuit.ru/studies/courses/5/5/info
7. Дейт К. Дж. - Введение в системы баз данных, 8-е издание.: Пер. с англ. — М.: Издательский дом "Вильяме", 2005. — 1328 с.: ил. — Парал. тит. англ.