# SQL или NoSQL — вот в чём вопрос

Автор оригинала: Alon Brody

Все мы знаем, что в мире технологий баз данных существует два основных направления: SQL и NoSQL, реляционные и нереляционные базы данных. Различия между ними заключаются в том, как они спроектированы, какие типы данных поддерживают, как хранят информацию.

Реляционные БД хранят структурированные данные, которые обычно представляют объекты реального мира. Скажем, это могут быть сведения о человеке, или о содержимом корзины для товаров в магазине, сгруппированные в таблицах, формат которых задан на этапе проектирования хранилища.

Нереляционные БД устроены иначе. Например, документо-ориентированные базы хранят информацию в виде иерархических структур данных. Речь может идти об объектах с произвольным набором атрибутов. То, что в реляционной БД будет разбито на несколько взаимосвязанных таблиц, в нереляционной может храниться в виде целостной сущности.

Внутреннее устройство различных систем управления базами данных влияет на особенности работы с ними. Например, нереляционные базы лучше поддаются масштабированию.

Какую технологию выбрать? Ответ на этот вопрос зависит от особенностей проекта, о котором идёт речь.

## О выборе SQL-баз данных

Не существует баз данных, которые подойдут абсолютно всем. Именно поэтому многие компании используют и реляционные, и нереляционные БД для решения различных задач. Хотя NoSQL-базы стали популярными благодаря

быстродействию и хорошей масштабируемости, в некоторых ситуациях предпочтительными могут оказаться структурированные SQL-хранилища. Вот две причины, которые могут послужить поводом для выбора SQL-базы:

1. Необходимость соответствия базы данных требованиям ACID (Atomicity, Consistency, Isolation, Durability — атомарность, непротиворечивость, изолированность, долговечность). Это позволяет уменьшить вероятность неожиданного поведения системы и обеспечить целостность базы данных. Достигается подобное путём жёсткого определения того, как именно транзакции взаимодействуют с базой данных. Это отличается от подхода, используемого в NoSQL-базах, которые ставят во главу угла гибкость и скорость, а не 100% целостность данных.

2. Данные, с которыми вы работаете, структурированы, при этом структура не подвержена частым изменением. Если ваша организация не находится в стадии экспоненциального роста, вероятно, не найдётся убедительных причин использовать БД, которая позволяет достаточно вольно обращаться с типами данных и нацелена на обработку огромных объёмов информации.

## О выборе NoSQL-баз данных

Если есть подозрения, что база данных может стать узким местом некоего проекта, основанного на работе с большими объёмами информации, стоит посмотреть в сторону NoSQL-баз, которые позволяют то, чего не умеют реляционные БД.

Вот возможности, которые стали причиной популярности таких NoSQL баз данных, как MongoDB, CouchDB, Cassandra, HBase:

1. Хранение больших объёмов неструктурированной информации. База данных NoSQL не накладывает ограничений на типы хранимых данных. Более того, при необходимости в процессе работы можно добавлять новые типы данных.

2. Использование облачных вычислений и хранилищ. Облачные хранилища — отличное решение, но они требуют, чтобы данные можно было легко распределить между несколькими серверами для обеспечения масштабирования. Использование, для тестирования и разработки, локального оборудования, а затем перенос системы в облако, где она и работает — это именно то, для чего созданы NoSQL базы данных.

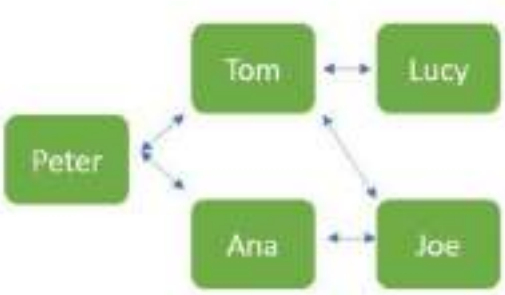
3. Быстрая разработка. Если вы разрабатываете систему, используя agile-методы, применение реляционной БД способно замедлить работу. NoSQL базы данных не нуждаются в том же объёме подготовительных действий, которые обычно нужны для реляционных баз.

В следующем разделе рассмотрим некоторые различия между технологиями SQL и NoSQL. А именно, сначала взглянем на простой пример, показывающий фундаментальное различие двух подходов к организации баз данных, потом поговорим о масштабируемости и индексации данных. А в итоге остановимся на

примере большой CRM-системы, нуждающейся в высокой производительности хранилища данных.

## SQL и NoSQL

Начнём с некоторых ключевых концепций реляционных и нереляционных баз данных. Ниже показана база данных, содержащая сведения о взаимоотношениях людей. Вариант **a** — это бессхемная структура, построенная в виде графа, характерная для NoSQL-решений. Вариант **b** показывает, как те же данные можно представить в структурированном виде, типичном для SQL.




| NODE | FRIENDS |
|---|---|
| Peter | {Tom, Ana} |
| Tom | {Joe, Lucy, Peter} |
| Ana | {Joe, Lucy, Peter} |
| Lucy | {Tom} |
| Joe | {Tom, Ana} |

(b)

*Два варианта представления данных*

Бессхемность означает, что два документа в структуре данных NoSQL не должны иметь одинаковые поля и могут хранить данные разных типов. Вот, например, массив объектов, набор полей которых не совпадает.

```
var cars = [
        { Model: "BMW", Color: "Red", Manufactured: 2016 },
        { Model: "Mercedes", Type: "Coupe", Color: "Black", Manufactured: "1-1-2017" }
];
```

При реляционном подходе данные надо хранить в заранее спроектированной структуре, из которой эти данные потом можно извлекать. Например, используя оператор `JOIN`при выборке из двух таблиц:

```
SELECT Orders.OrderID, Customers.Name, Orders.Date
FROM Orders
INNER JOIN Customers
ON Orders.CustID = Customers.CustID
```

Как более продвинутый пример, для демонстрации того, когда SQL предпочтительнее NoSQL, рассмотрим особенности применения в NoSQL-базах алгоритмов уплотнения. Проблема заключается в том, что в некоторых NoSQL-базах (например, в CouchDB и HBase) постоянно приходится формировать так называемые `sstables` — строковые таблицы в формате ключ-значение,

отсортированные по ключу. В такие таблицы, которые сохраняются на диск, данные попадают из таблиц, хранящихся в памяти, при их переполнении и в других ситуациях. При интенсивной работе с базой создание таблиц, со временем, приводит к тому, что подсистема ввода-вывода устройства хранения данных становится узким местом для операций чтения данных. Как результат, чтение в NoSQL-базе происходит медленнее, чем запись, что сводит на нет одно из главных преимуществ нереляционных баз данных. Именно для того, чтобы уменьшить этот эффект, системы NoSQL используют, в фоновом режиме, алгоритмы уплотнения данных, пытаясь объединить множество таблиц в одну. Но и сама по себе эта операция весьма ресурсоёмка, система работает под повышенной нагрузкой.

## Масштабируемость

Одно из основных различий рассматриваемых технологий заключается в том, что NoSQL-базы лучше поддаются масштабированию. Например, в MongoDB имеется встроенная поддержка репликации и шардинга (горизонтального разделения данных) для обеспечения масштабируемости. Хотя масштабирование поддерживается и в SQL-базах, это требует гораздо больших затрат человеческих и аппаратных ресурсов.

| **Тип хранилища данных** | **Сценарий использования** | **Пример** | **Рекомендации** |
|---|---|---|---|
| Хранилище типа ключ-значение | Подходит для простых приложений, с одним типом объектов, в ситуациях, когда поиск объектов выполняют лишь по одному атрибуту. | Интерактивное обновление домашней страницы пользователя в Facebook. | Рекомендовано знакомство с технологией memcached.<br>Если приходится искать объекты по нескольким атрибутам, рассмотрите вариант перехода к хранилищу, ориентированному на документы. |
| Хранилище, ориентированное на документы | Подходит для хранения объектов различных типов. | Транспортное приложение, оперирующее данными о водителях и автомобилях, работая с которым надо | Подходит для приложений, в ходе работы с которыми допускается реализация принципа «согласованность в конечном счёте» с ограниченными |

| | | | |
|---|---|---|---|
| | | искать объекты по разным полям, например — имя или дата рождения водителя, номер прав, транспортное средство, которым он владеет. | атомарностью и изоляцией. Рекомендуется применять механизм кворумного чтения для обеспечения своевременной атомарной непротиворечивости. |
| Система хранения данных с расширяемыми записями | Более высокая пропускная способность и лучшие возможности параллельной обработки данных ценой слегка более высокой сложности, нежели у хранилищ, ориентированных на документы. | Приложения, похожие на eBay. Вертикальное и горизонтальное разделение данных для хранения информации клиентов. | Для упрощения разделения данных используются HBase или Hypertable. |
| Масштабируемая RDBMS | Использование семантики ACID освобождает программистов от необходимости работать на достаточно низком уровне, а именно, отвечать за блокировки и непротиворечивость данных, обрабатывать устаревшие данные, коллизии. | Приложения, которым не требуются обновления или слияния данных, охватывающие множество узлов. | Стоит обратить внимание на такие системы, как MySQL Cluster, VoltDB, Clustrix, ориентированные на улучшенное масштабирование. |

Более подробное сравнение SQL и NoSQL можно найти в этом материале. Вот его основные положения. А именно, были проведены испытания трёх основных характеристик систем: параллельная обработка данных, работа с хранилищами информации, репликация данных. Возможности параллельной обработки оценивались путём анализа механизмов блокировки, управления параллельным доступом на основе многоверсионности, и ACID. Тестирование хранилищ

охватывало и физические носители, и хранилища использующие оперативную память. Репликацию испытывали в синхронном и асинхронном режимах.

Используя данные, полученные в ходе испытаний, авторы делают выводы о том, что SQL-базы с возможностью кластеризации показали многообещающие результаты производительности в расчёте на один узел, и, кроме того, обладают способностью масштабируемости, что даёт системам RDBMS преимущество перед NoSQL за счёт полного соответствия принципам ACID.

## Индексация

В системах RDBMS индексация используется для ускорения операций извлечения данных из баз. Отсутствие индекса означает, что таблица должна быть просмотрена целиком для того, чтобы выполнить запрос на чтение.

И в SQL, и в NoSQL-базах индексы служат одной и той же цели — ускорить и оптимизировать извлечение данных. Но то, как именно они работают — различается из-за разных архитектур баз данных и особенностей хранения информации в базе. В то время, как SQL-индексы представлены в виде B-деревьев, которые отражают иерархическую структуру реляционных данных, в NoSQL базах данных они указывают на документы, или на части документов, между которыми, в основном, нет никаких отношений. Вот подробный материал на эту тему.

## CRM-системы

CRM-приложения — это один из лучших примеров систем, для которых характерны огромные объёмы ежедневно обрабатываемых данных и очень большое количество транзакций. Все разработчики таких приложений используют и SQL, и NoSQL базы данных. И, хотя большая часть данных транзакций всё ещё хранится в SQL-базах, применение находят общедоступные системы класса DBaaS (data-base-as-a-service, база данных как сервис), наподобие AWS DynamoDB и Azure DocumentDB, в результате, серьёзная нагрузка по обработке данных может быть перенесена в облачные NoSQL-базы.

В то время, как использование подобных служб освобождает разработчика от решения задач по обслуживанию хранилищ, это, кроме того, область, где NoSQL базы применяются для того, для чего они, в основном, и были созданы, например, для глубинного анализа данных. Объёмы информации, хранимой в огромных CRM-системах финансовых и телекоммуникационных компаний, было бы практически невозможно проанализировать, используя инструменты вроде SAS или R. Это потребовало бы огромных аппаратных ресурсов.

Главное преимущество таких систем — использование неструктурированных данных, похожих на документы. Такие данные могут подаваться на вход статистических моделей, которые дают компаниям возможность выполнять различные виды анализа. CRM-приложения, кроме того, являются весьма удачным примером, в котором две системы баз данных выступают не конкурентами, а существуют в гармонии, играя каждая свою

роль в большой архитектуре управления данными.

## Итоги

Занимаясь поиском системы управления базами данных, можно выбрать одну технологию, а позже, уточнив требования, переключиться на что-то другое. Однако, разумное планирование позволит сэкономить немало времени и средств.

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

- Имеются логические требования к данным, которые могут быть определены заранее.
- Очень важна целостность данных.
- 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:

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

В итоге хочется сказать, что в современном мире нет противостояния между реляционными и нереляционными базами данных. Вместо этого стоит говорить об их совместном использовании для решения задач, на 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.