Реляционные и не реляционные базы данных

Сравним реляционные и не реляционные базы данных, разберемся в чем их отличия, плюсы и минусы каждого класса и приведем примеры. Для начала построим табличку со сравнением разных классов баз данных по основным характеристикам.

Дисклеймер: в данном эссе понятие «SQL базы данных» приравнено к понятию «реляционные базы данных», как и «NoSQL» приравнено к «не реляционные».

Класс БД	**Реляционная базы данных (SQL)**	**Не реляционная база данных (распределенная) (NoSQL)**
Схема	Статическая схема	Динамическая схема
Иерархическое хранение данных	Отсутствует	Присутствует
Сложные запросы	Подходит	Не подходит
Масштабируемость	По вертикали	По горизонтали
Свойства	ACID	CAP
Атомарность (Atomicity)	+	-
Согласованность (Consistency)	+	+
Изолированность (Isolation)	+	-
Надёжность (Durability)	+	-
Доступность (Availability)	-	+
Разделяемость (Partition tolerance)	-	+

Анализируя основные пункты сравнения, сразу становится понятно, что SQL и NoSQL базы данных отличаются практически по всем основным пунктам. Можно сразу предположить, что у этих баз данных абсолютно разная направленность использования и разные задачи, которые они должны решать (из общего, только хранить данные).

Преимущества и недостатки разных баз данных

Разберем основные плюсы SQL баз данных. Они напрямую вытекают из свойств ACID, которым удовлетворяет данный класс баз данных. **Во-первых**, атомарность гарантирует транзакционность то есть, что транзакция (группа запросов) будет выполнена и применена целиком, или не будет выполнена вовсе. **Во-вторых**, согласованность гарантирует, что база данных всегда (до и после транзакции) будет находится в корректном состоянии. **В-третьих**, изолированность гарантирует, что несколько транзакций не будут перемешиваться друг с другом, а будут выполнены независимо. **В-четвертых**, надежность – если транзакция прошла успешно, то она точно записана в базу данных на диск, а не во временную память. Следовательно, даже системный сбой (не затрагивающий диск) не может повредить эти данные.

Теперь перечислим основные плюсы NoSQL баз данных, которые аналогично выводятся из свойств CAP. **Во-первых**, согласованность, как и у SQL баз данных, но с оговоркой, что свойство гарантируется для всех копий баз данных (nodes). **Во-вторых**, доступность гарантирует, что каждая копия базы данных (node) сможет ответить на любой запрос за разумное количество времени. **В-третьих,** partion tolerance гарантирует, что если одна или несколько копий (nodes) вышли из «строя», то база продолжит корректно функционировать. При этом, когда отключенные копии (nodes) восстановят связь с системой, они «догонят» ее состояние.

Недостатками обоих классов баз данных являются отсутствующие преимущества, которые присутствуют у противоположного класса. Например, NoSQL базы в целом не гарантируют атомарность операций, хоть и существует частные примеры БД, где атомарность частично реализована (или может быть реализована с помощью дополнительных улучшений).

Какую базу данных выбрать?

Теперь возникает логичный вопрос – какой класс базы данных выбрать? SQL или NoSQL? Как уже было сказано выше, данные классы баз данных предназначены для решения разных задач, поэтому для корректного выбора нужно проводить анализ отдельно взятого проекта и для него уже подбирать базу данных.

Каким правилами можно руководствоваться при выборе базы данных?

1. В первую очередь, стоит обратить внимание – есть ли в проекте критическое требование, реализованное только в одном из классов баз данных. Если есть, тогда выбор очевиден. Например, проекту в силу каких-либо причин требуется база данных с динамической схемой и с возможность разделения. В таком случае, возможно выбирать только среди NoSQL баз данных (например, Mongo DB, как одна из самых распространённых).

2. Стоит обратить внимание на структурированность данных, которые требуется хранится. Для строго определённой структуры данных подойдут SQL базы данных, в противном случае лучше выбрать NoSQL.

3. Следующий фактор, который следует учитывать — это то, как часто вы будете запрашивать свои данные, как быстро вам нужно выполнять запросы, и кто будет отвечать за выполнение этих запросов. Естественно, что наличие строгой структуры позволит SQL выполнять запросы быстрее аналогичных в NoSQL, но при этом NoSQL базы предоставляют бóльший выбор типов для хранимых данных

4. Стоит подумать, как база данных может изменяться и увеличиваться в будущем. В этом плане SQL базы данных проигрывают NoSQL, так как рассчитаны на вертикальную масштабируемость, то есть расширение может происходить только за счет увеличения мощности сервера. В тоже время NoSQL базы данных могут размещаться на нескольких серверах и расширяться за счет добавления новых серверов.

Обобщая, нужно понять, как выглядят ваши данные, как вы будете их получать, и какая масштабируемость понадобится. Исходя из этих факторов выбирать между SQL и NoSQL, а далее уже конкретную базу данных. Среди самых популярных **SQL** решений можно перечислить: **MySQL, PostgreSQL, Oracle**. Среди **NoSQL: MongoDB, Redis.** Также, стоит упомянуть, что в рамках одного проекта может быть «поднято» множество разных баз данных, в том числе разных классов, для решения соответствующих задач.

Примеры проектов с разными классами баз данных

Представим, что перед нами стоит задача разработать базу данных, в которой будет хранится информации о товарах на складе одного из предприятий (номенклатура). Сразу становится понятно, что в данном случае не стоит ожидать резкого роста объема базы данных, так как база отражает физическое наличие товара на складе, то есть горизонтальная масштабируемость не требуется. Схема хранения подойдет статическая и данные структурированы. В данном случае SQL база данных сможет обеспечить атомарность, надежность и безопасность решения. Как уже упоминалось ранее, для ограниченного количества данных со строго закреплённой структурой, то есть **для большинства внутренних систем компаний и предприятий подойдет SQL база данных.**

Классические примеры использования NoSQL баз данных – это веб-сервисы и разработка связанная с big data. Анализ больших данных изначально подразумевает наличие огромного количества сохраненной информации, а многие веб-сервисы, даже если не пользуются популярностью на данный момент, должны быть готовы к резкому росту количества пользователей, а следовательно, и к резкому росту объема хранимой информации. В обоих вышеупомянутых случаях отсутствие возможности горизонтальной масштабируемости может привести к низкой производительности или вовсе к отказу базы данных. Следовательно, единственным верным решением в подобных ситуациях будут NoSQL базы данных.

Источники

1. https://www.geeksforgeeks.org/difference-between-sql-and-nosql/
2. https://www.geeksforgeeks.org/acid-properties-in-dbms/
3. https://www.geeksforgeeks.org/the-cap-theorem-in-dbms/
4. https://www.geeksforgeeks.org/sql-vs-nosql-which-one-is-better-to-use/
5. https://thorntech.com/sql-vs-nosql