Урок 5 > Выбор подходящих баз данных

> Оглавление

> Глоссарий

Требования ACID — это набор требований, которые обеспечивают сохранность ваших данных.

- **Atomicity** — Атомарность. Гарантирует, что каждая транзакция будет выполнена полностью или не будет выполнена совсем. Не допускаются промежуточные состояния.
- **Consistency** — Согласованность. Это свойство вытекает из предыдущего. Благодаря тому, что транзакция не допускает промежуточных результатов, база остается консистентной. Есть такое определение транзакции: «Упорядоченное множество операций, переводящих базу данных из одного согласованного состояния в другое». То есть до выполнения операции и после база остается консистентной.
- **Isolation** — Изолированность. Во время выполнения транзакции параллельные транзакции не должны оказывать влияния на её результат.
- **Durability** — Надёжность. Если пользователь получил подтверждение от системы, что транзакция выполнена, он может быть уверен, что сделанные им изменения не будут отменены из-за какого-либо сбоя. Обесточилась система, произошел сбой в оборудовании? На выполненную транзакцию это не повлияет.

> Основные виды БД

- **Реляционные** — базы данных, записи в которых хранятся в виде набора таблиц и заранее определенных связей между ними. К таким БД относятся все СУБД, а также базы удовлетворяющее ACID-требованиям.
- **БД типа key-value** — поддерживают и хранят только наиболее простые операции с записями по ключу.

- **Колоночные БД** — данные хранятся в виде набора колонок, не связанных между собой. Каждая запись, которую нужно сохранить в базу, разбивается на части, каждая часть записывается в свою колонку.
- **Документные БД** — хранят данные в виде сложных древовидных структур типа JSON.

Выделяют еще и другие типы БД, но они используются крайне редко и для очень специфичных задач:

- Временные (темпоральные) БД
- Графовые БД
- и другие

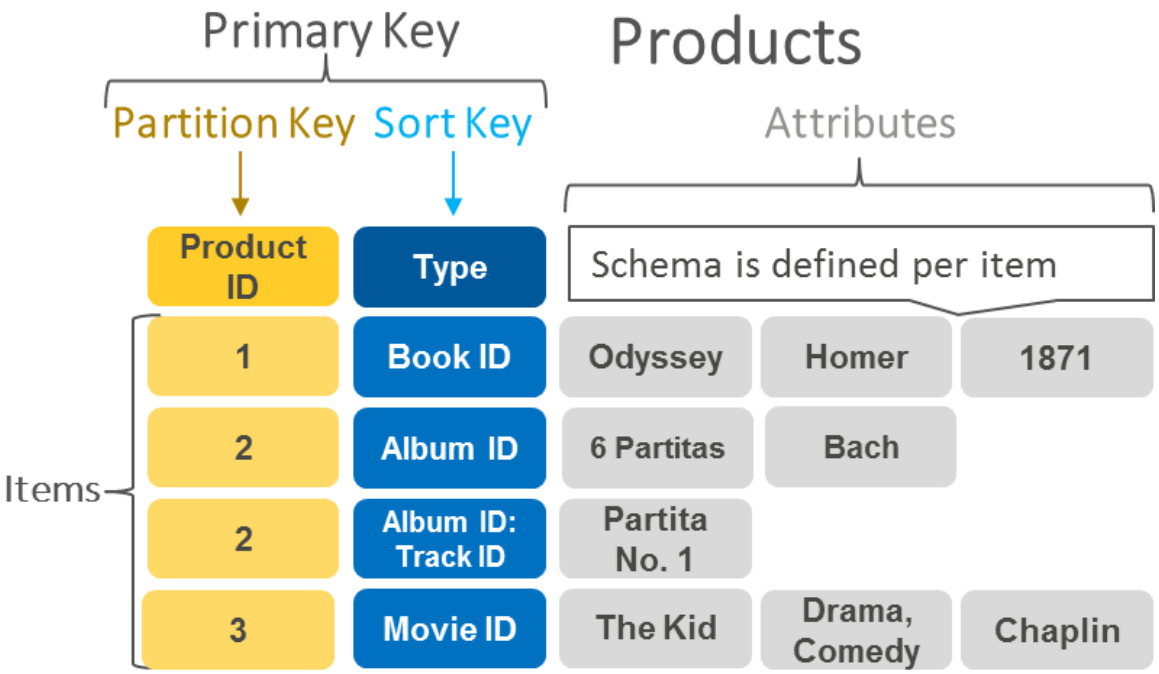
> Реляционные базы данных

Реляционные БД – базы данных, основанные на реляционной модели, подразумевающей организацию данных в виде таблиц (строк-записей и столбцов-атрибутов), каждая запись в которой имеет уникальный ключ и может в атрибутах ссылаться на ключи в других таблицах. Можно считать это «базой данных по умолчанию», т.к. при создании MVP вашего проекта вы наверняка подключите БД вроде SQLite, Postgres, MySQL и т.п. для хранения ваших записей.

Все реляционные базы данных соблюдают ACID-требования.

> БД вида key-value

База данных на основе пар «ключ-значение» – это тип нереляционных баз данных, в котором для хранения данных используется простой метод «ключ-значение». База данных на основе пар «ключ-значение» хранит данные как совокупность пар «ключ-значение», в которых ключ служит уникальным идентификатором. Как ключи, так и значения могут представлять собой что угодно: от простых до сложных составных объектов. Базы данных с использованием пар «ключ-значение» поддерживают высокую разделяемость и обеспечивают беспрецедентное горизонтальное масштабирование, недостижимое при использовании других типов баз данных. Например, Amazon DynamoDB выделяет дополнительные разделы на таблицу, если существующий раздел заполняется до предела и требуется больше пространства для хранения.



Примеры использования

Хранилище сессий

Основанное на сессиях приложение (например, интернет-приложение) запускает сессию, когда пользователь входит в систему. Оно активно до тех пор, пока пользователь не выйдет из системы или не истечет время сессии. В течение этого периода приложение хранит все связанные с сессией данные либо в основной памяти, либо в базе данных. Данные сессии могут включать информацию профиля пользователя, сообщения, индивидуальные данные и темы, рекомендации, таргетированные рекламные кампании и скидки. Каждая сессия пользователя имеет уникальный идентификатор. Данные сессий всегда запрашиваются только по первичному ключу, поэтому для их хранения отлично подходит быстрое хранилище пар «ключ-значение». В целом базы данных на основе пар «ключ-значение» могут снижать накладные расходы в расчете на страницу по сравнению с реляционными базами данных.

Корзина интернет-магазина

Во время праздничного сезона покупок сайт интернет-магазина может получать миллиарды заказов за считаные секунды. Используя базы данных на основе пар «ключ-значение», можно обеспечить необходимое масштабирование при существенном увеличении объемов данных и чрезвычайно интенсивных изменениях состояния. Такие базы данных позволяют одновременно обслуживать миллионы пользователей благодаря распределенным обработке и хранению данных. Базы данных на основе пар «ключ-значение» также обладают встроенной избыточностью, что позволяет справляться с потерей узлов хранилища.

Из примеров:

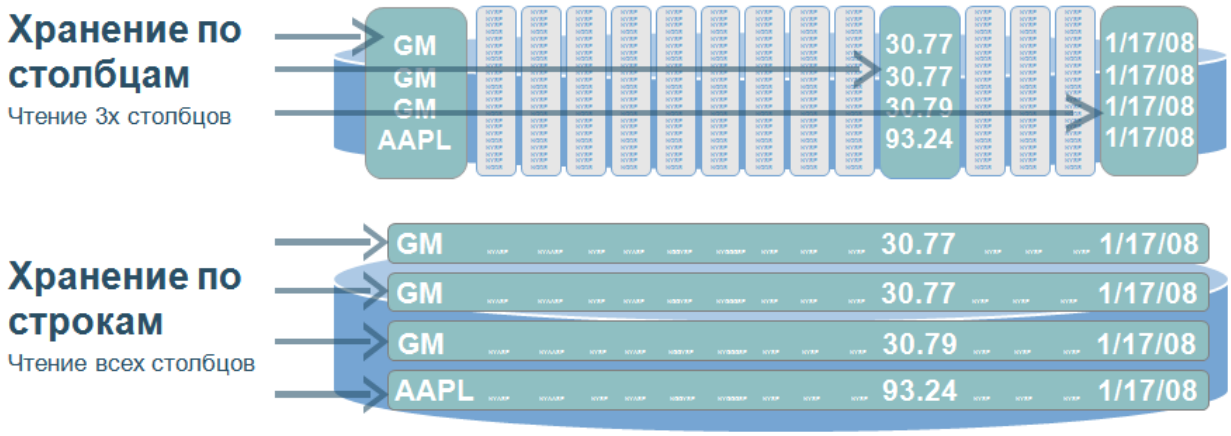
- Redis
- Memcache

> Колоночные базы данных

Колоночные БД призваны решить проблему неэффективной работы традиционных СУБД в аналитических системах и системах в подавляющим большинством операций типа «чтение». Они позволяют на более дешевом и маломощном оборудовании получить прирост скорости выполнения запросов в 5, 10 и иногда даже в 100 раз, при этом, благодаря компрессии, данные будут занимать на диске в 5-10 раз меньше, чем в случае с традиционными БД.

У колоночных БД есть и недостатки — они медленно работают на запись, не подходят для транзакционных систем и как правило, ввиду «молодости» имеют ряд ограничений для разработчика, привыкшего к развитым традиционным БД.

Колоночные БД применяются как правило в аналитических системах класса business intelligence (ROLAP) и аналитических хранилищах данных (data warehouses). Причем объемы данных могут быть достаточно большими — есть примеры по 300-500ТБ и даже случаи с >1ПБ данных.



Примеры БД:

- Hbase
- Vertica
- ClickHouse

Дополнительные ссылки:

Перевод статьи М. Стоунбрекера "«One Size Fits All»: An Idea Whose Time Has Come and Gone"

> Документные базы данных

Документная база данных – это тип нереляционных баз данных, предназначенный для хранения и запроса данных в виде документов в формате, подобном JSON. Документные базы данных позволяют разработчикам хранить и запрашивать данные в БД с помощью той же документной модели, которую они используют в коде приложения. Гибкий, полуструктурированный, иерархический характер документов и документных баз данных позволяет им развиваться в соответствии с потребностями приложений. Документная модель хорошо работает в таких примерах использования, как каталоги, пользовательские профили и системы управления контентом, где каждый документ уникален и изменяется со временем. Документные базы данных обеспечивают гибкость индексации, производительность выполнения стандартных запросов и аналитику наборов документов.

В следующем примере документ в формате, подобном JSON, описывает книгу.

```
[
    {
        "year" : 2013,
        "title" : "Turn It Down, Or Else!",
        "info" : {
            "directors" : [ "Alice Smith", "Bob Jones"],
            "release_date" : "2013-01-18T00:00:00Z",
            "rating" : 6.2,
            "genres" : ["Comedy", "Drama"],
            "image_url" : "http://ia.media-imdb.com/images/N/O9ERWAU7FS797AJ7LU8HN09AMUP908RLlo5JF90EWR7LJKQ7@@._V1_SX400_.jpg",
            "plot" : "A rock band plays their music at high volumes, annoying the neighbors.",
            "actors" : ["David Matthewman", "Jonathan G. Neff"]
        }
    },
    {
        "year": 2015,
        "title": "The Big New Movie",
        "info": {
            "plot": "Nothing happens at all.",
            "rating": 0
        }
    }
]
```

Примеры использования

Управление контентом

Документная база данных – отличный выбор для приложений управления контентом, таких как платформы для блогов и размещения видео. При использовании документной базы данных каждая сущность, отслеживаемая приложением, может храниться как отдельный документ. Документная база данных позволяет разработчику с удобством обновлять приложение при изменении требований. Кроме того, если необходимо изменить модель данных, то требуется обновление только затронутых этим изменением документов. Для внесения изменений нет необходимости обновлять схему и прерывать работу базы данных.

Каталоги

Документные базы данных эффективны для хранения каталожной информации. Например, в приложениях для интернет-коммерции разные товары обычно имеют различное количество атрибутов. Управление тысячами атрибутов в реляционных базах данных неэффективно. Кроме того, количество атрибутов влияет на производительность чтения. При использовании документной базы данных атрибуты каждого товара можно описать в одном документе, что упрощает управление и повышает скорость чтения. Изменение атрибутов одного товара не повлияет на другие товары.

Из примеров:

- MongoDB
- CouchDB
- Amazon DocumentDB

> CAP- и PACELC-теоремы

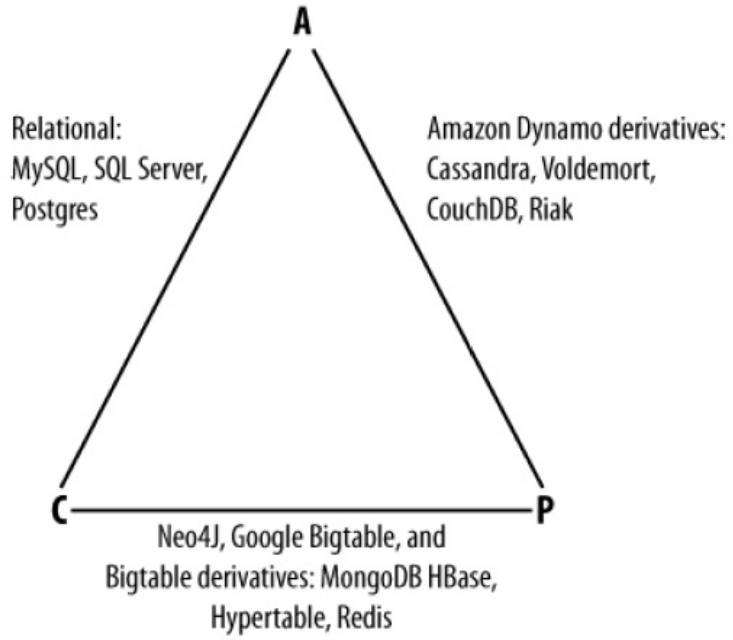
CAP теорема

Эта теорема была представлена на симпозиуме по принципам распределенных вычислений в 2000 году Эриком Брюером. В 2002 году Сет Гилберт и Нэнси Линч из MIT опубликовали формальное доказательство гипотезы Брюера, сделав ее теоремой.

В чем заключается теорема

В CAP говорится, что в распределенной системе возможно выбрать только два из трех свойств:

- C (consistency) — согласованность. Каждое чтение даст вам самую последнюю запись.
- A (availability) — доступность. Каждый узел (не упавший) всегда успешно выполняет запросы (на чтение и запись).
- P (partition tolerance) — устойчивость к распределению. Даже если между узлами нет связи, они продолжают работать независимо друг от друга.

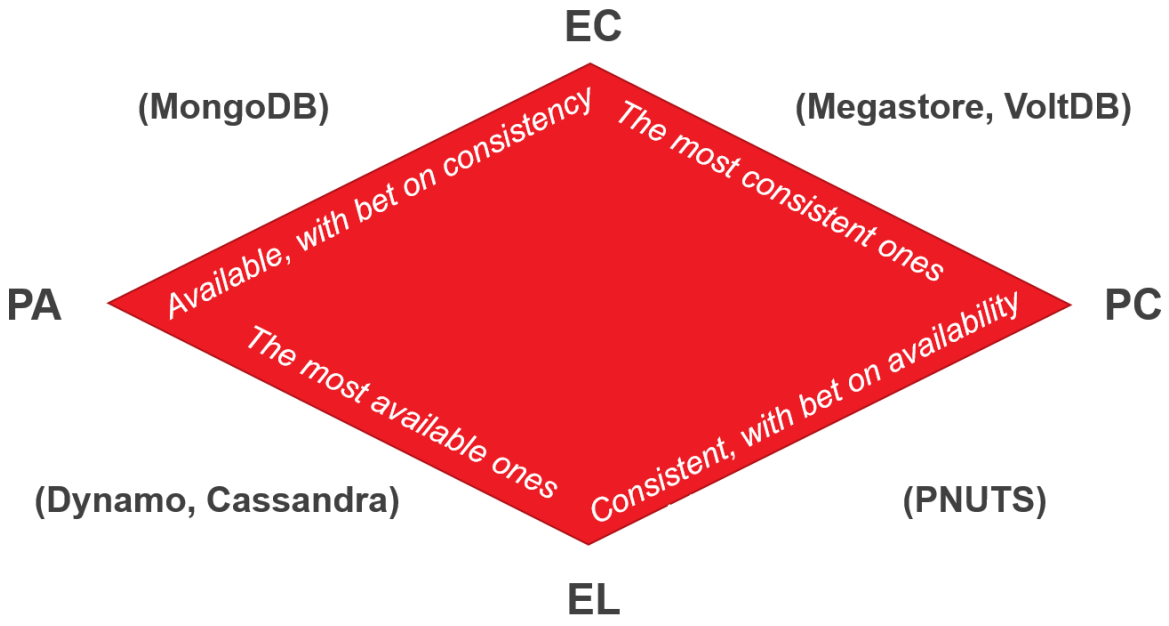


PACELC-теорема

Теорема PACELC была впервые описана и формализована Даниелом Дж. Абади из Йельского университета в 2012 году. Поскольку теорема PACELC основана на CAP, она также использует его определения.

Вся теорема сводится к IF P -> (C or A), ELSE (C or L).

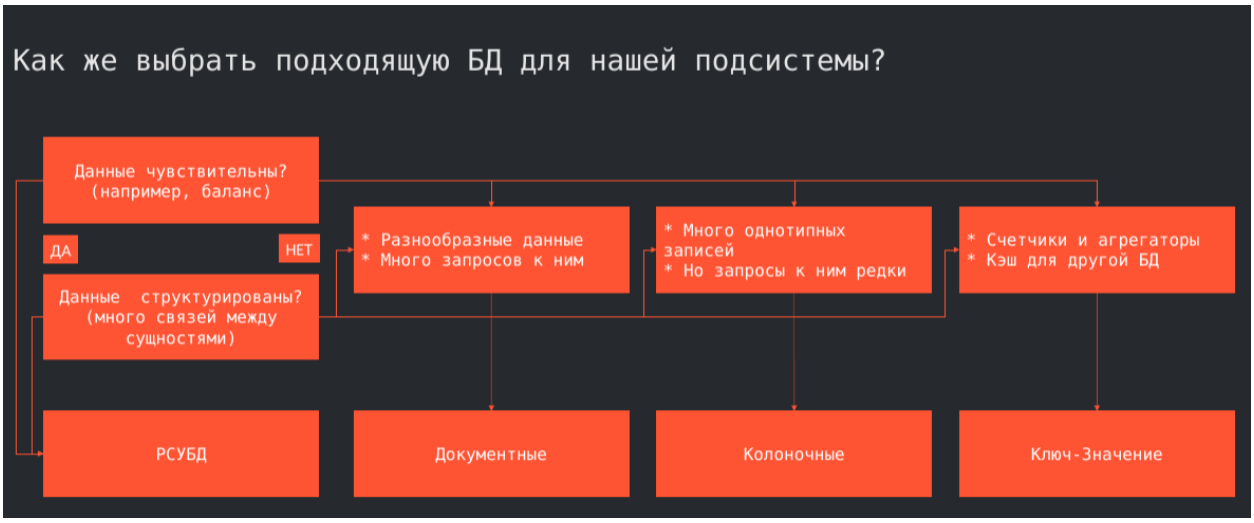
Latency — это время, за которое клиент получит ответ и которое регулируется каким-либо уровнем consistency. Latency (задержка), в некотором смысле представляет собой степень доступности.



> Выбор подходящей БД

При проектировании подсистемы требуется определиться, что из C, A и P нам важнее:

- У согласованных и доступных систем (CA) есть сложности с устойчивостью к разделению, которые частично решается созданием реплик, а также такие системы масштабируют вертикально (мощнее железо), чтобы изначально избежать разделения. Именно сюда попадают RDBMS с ACID, которые всегда кстати для хранения чувствительных данных.
- В согласованных и устойчивых к разделению системах (CP) могут быть проблемы с доступностью, на зато данные в разделенных нодах всегда корректны и выглядят одинаково. Примеры таких БД есть в разных семействах – колоночная HBase, документная Mongo и Redis.
- В доступных и устойчивых к разделению системах (AP) нет строгой согласованности, но она так или иначе достигается, чего бывает достаточно для крупных систем (например, чей-то новый пост на другом конце света увидят на минуту позже). Примеры опять же есть разные – колоночная Cassandra, документная CouchDB и Dynamo. В противовес к строгим ACID выделяют BASE (basic availability, soft state, eventual consistency).


Как же выбрать подходящую БД для нашей подсистемы?
Данные чувствительны? (например, баланс)
ДА
НЕТ
Данные структурированы? (много связей между сущностями)
* Разнообразные данные
* Много запросов к ним
* Много однотипных записей
* Но запросы к ним редки
* Счетчики и агрегаторы
* Кэш для другой БД
РСУБД
Документные
Колоночные
Ключ-Значение