Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина

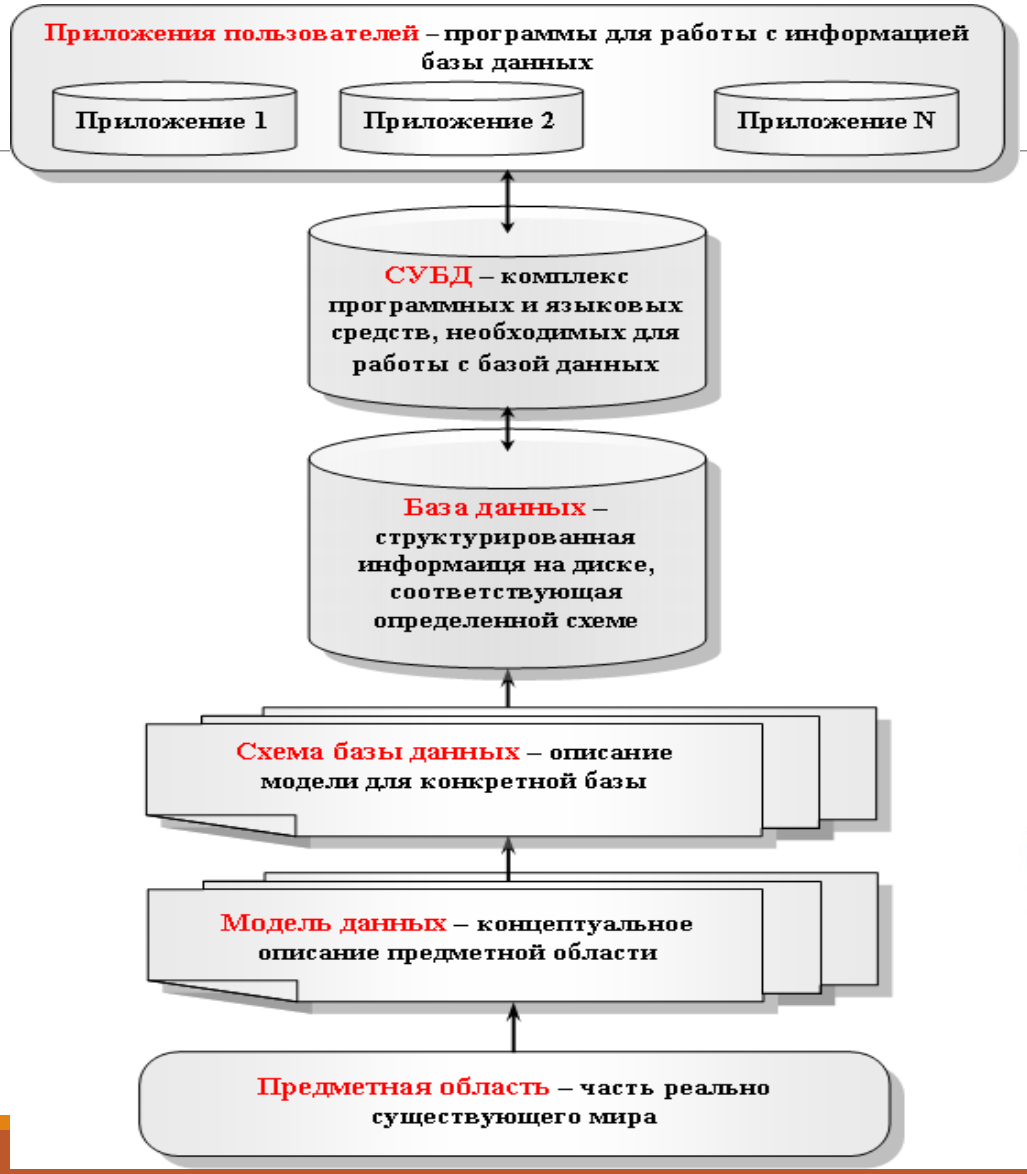
Системы управления базами данных. Современные типы баз данных

ЛЕКЦИЯ 1

ассистент кафедры Интеллектуальных информационных технологий УрФУ

Глухих Олег Юрьевич

Взаимосвязь основных терминов в области проектирования баз данных и работы с ними



Любая информационная система включает некоторую базу данных, так как, чтобы работать с информацией, нужно работать с данными.

Типы современных баз данных

I. Простейшие типы баз данных

- 1. Простые структуры данных
- 2. Иерархические базы данных
- 3. Сетевые базы данных

II. Реляционные БД

- 4. SQL базы данных

III. NoSQL базы данных

- 5. Базы данных «ключ-значение»
- 6. Документная база данных
- 7. Графовая база данных
- 8. Колоночные базы данных
- 9. Базы данных временных рядов

IV. Комбинированные типы

- 10. NewSQL базы данных
- 11. Многомодельные базы данных

Простые структуры данных

/etc/passwd в *nix системе

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
bob:x:1000:1000:Bob Smith,,,:/home/bob:/bin/bash
```

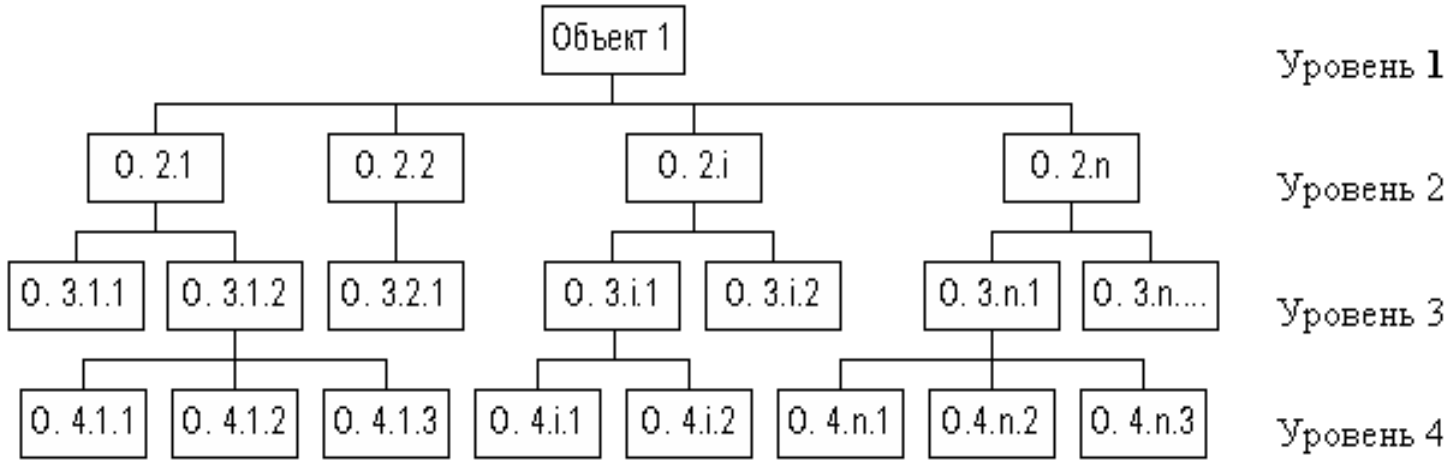
Следствия:

- ограничен тип и уровень сложности хранимой информации;
- трудно установить связи между компонентами данных;
- отсутствие функций параллелизма;
- практичны только для систем с небольшими требованиями к чтению и записи;
- используются для хранения конфигурационных данных;
- нет необходимости в стороннем программном обеспечении.

Примеры:

- /etc/passwd и /etc/fstab в *nix-системах
- csv-файлы

Иерархическая модель данных



Примеры:

- файловые системы
- DNS
- LDAP

В этой модели имеется один главный объект и остальные - подчиненные - объекты, находящиеся на разных уровнях иерархии. Взаимосвязи объектов образуют иерархическое дерево с одним корневым объектом.
Типичным представителем (наиболее известным и распространенным) является Information Management System (IMS) фирмы IBM. Первая версия появилась в 1968 г. До сих пор поддерживается много баз данных этой системы.

Иерархическая модель данных: достоинства и недостатки

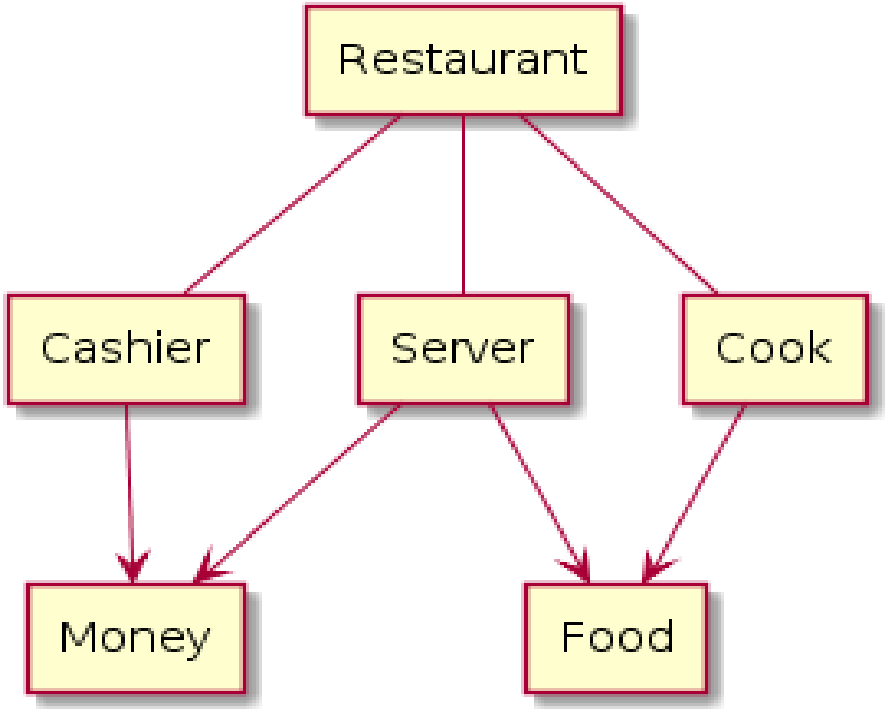
Достоинства модели

- *простота представления предметной области,*
- *наглядность,*
- *удобство анализа структур,*
- *простота их описания.*

Недостатки

сложность добавления новых и удаления существующих типов записей, невозможность отображения отношений, отличающихся от иерархических, громоздкость описания и информационную избыточность

Сетевая модель данных



В сетевой модели данных любой объект может быть одновременно и главным, и подчиненным, и может участвовать в образовании любого числа взаимосвязей с другими объектами. Сетевая БД состоит из набора записей и набора связей между этими записями.

Типичным представителем является **Integrated Database Management System (IDMS)** компании Cullinet Software, Inc. (1971 г.)

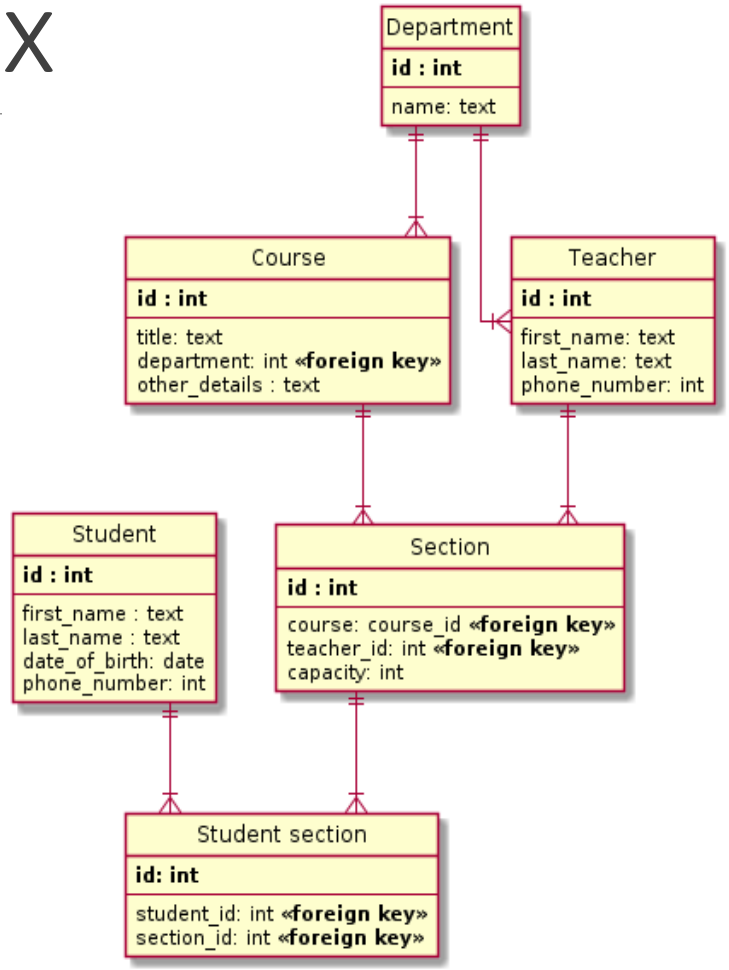
По сравнению с иерархическими сетевые модели обладают рядом существенных ***преимуществ****:*

- *возможность отображения практически всего многообразия взаимоотношений объектов предметной области,*
- *непосредственный доступ к любой вершине сети (без указания других вершин),*
- *малая информационная избыточность.*

Вместе с тем, в сетевой модели невозможно достичь полной независимости данных – с ростом объема информации сетевая структура становится весьма сложной для описания и анализа.

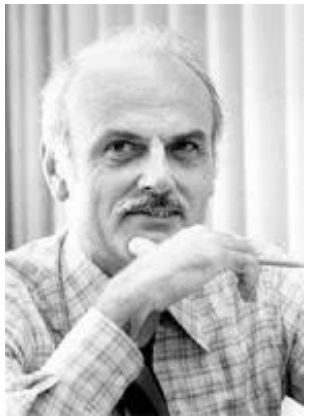
Реляционная модель данных

Реляционная модель данных основывается на математических принципах, вытекающих непосредственно из теории множеств и логики предикатов. Эти принципы впервые были применены в области моделирования данных в конце 1960-х гг. доктором *Е.Ф. Коддом*, в то время работавшим в IBM, а впервые опубликованы - в 1970 г.



Эдгар Франк «Тед» Кодд

(23 августа 1923 — 18 апреля 2003)



британский учёный, работы которого заложили основы теории реляционных баз данных. Работая в компании IBM, он создал реляционную модель данных.

12 правил Кодда

1. Реляционная СУБД должна быть способна полностью управлять базой данных через ее реляционные возможности.
2. **Информационное правило** - вся информация в реляционной БД (включая имена таблиц и столбцов) должна определяться строго как значения в таблицах.
3. **Гарантированный доступ** - любое значение в реляционной БД должно быть гарантированно доступно для использования через комбинацию имени таблицы, значения первичного ключа и имени столбца.
4. **Поддержка пустых значений** (null value) - СУБД должна уметь работать с пустыми значениями (неизвестными или неиспользованными значениями), в отличие от значений по умолчанию и независимо для любых *доменов*.
5. **Онлайновый реляционный каталог** - описание БД и ее содержания должны быть представлены на логическом уровне как таблицы, к которым можно применять запросы, используя язык базы данных.
6. **Исчерпывающий язык управления данными** - по крайней мере, один из поддерживаемых языков должен иметь четко определенный синтаксис и быть всеобъемлющим. Он должен поддерживать описание структуры данных и манипулирование ими, правила целостности, авторизацию и транзакции.
7. **Правило обновления представлений** (views) - все представления, теоретически обновляемые, могут быть обновлены через систему.

12 правил Кодда

8. **Вставка, обновление и удаление** - СУБД поддерживает не только запрос на отбор данных, но и вставку, обновление и удаление.
9. **Физическая независимость данных** - на программы-приложения и специальные программы логически не влияют изменения физических методов доступа к данным и структур хранилищ данных.
10. **Логическая независимость данных** - на программы-приложения и специальные программы логически не влияют, в пределах разумного, изменения структур таблиц.
11. **Независимость целостности** - язык БД должен быть способен определять правила целостности. Они должны сохраняться в онлайновом справочнике, и не должно существовать способа их обойти.
12. **Независимость распределения** - на программы-приложения и специальные программы логически не влияет, первый раз используются данные или повторно.
13. **Неподрывность** - невозможность обойти правила целостности, определенные через язык базы данных, использованием языков низкого уровня.

Реляционная модель данных: достоинства и недостатки

Табличная организация БД позволяет реализовать ее важнейшее ***преимущество*** *перед другими моделями данных:*

- *возможность использования точных математических методов манипулирования данными, и, прежде всего, аппарата реляционной алгебры и исчисления отношений,*
- *наглядность,*
- *простота изменения данных и организация разграничения доступа к ним.*

Основным ***недостатком*** *реляционной модели данных является:*
информационная избыточность, что ведет,
к перерасходу ресурсов вычислительных систем.

Примеры:

- MySQL
- MariaDB
- PostgreSQL
- SQLite

Основные понятия реляционных баз данных

Каждая строка, содержащая данные, называется ***кортежем***, каждый столбец отношения называется ***атрибутом*** (на уровне практической работы с современными реляционными БД используются термины "запись" и "поле").

Элементами описания реляционной модели данных на концептуальном уровне являются *сущности*, *атрибуты*, *домены* и связи.

Термины реляционной модели

Термин реляционной модели	Эквивалентный термин
Отношение	Таблица
Схема отношения	Строка заголовков столбцов таблицы (заголовок таблицы)
Кортеж	Строка
Сущность	Описание свойств объекта
Атрибут	Столбец, поле
Домен	Множество допустимых значений атрибута
Первичный ключ	Уникальный идентификатор
Кардинальность	Количество строк
Степень	Количество столбцов

NoSQL базы данных

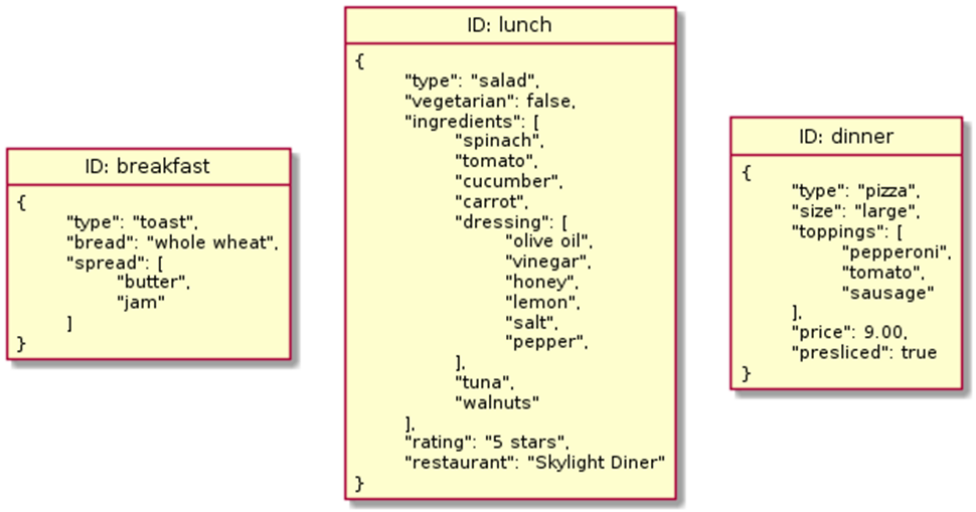
5. Базы данных «ключ-значение»

key:	value
user_id:	f5badc33-5bd7-4b65-a737-b5304675f476
color:	blue
repetitions:	3
text:	hello world
data:	{ … }

Примеры:
- Redis
- memcached
- etcd

Преимущества:
- хранилища обеспечивают быстрый и малозатратный доступ;
- часто хранят данные конфигураций и информацию о состоянии данных, представленных словарями или хэшем;
- нет жёсткой схемы отношения между данными, поэтому в таких БД часто хранят одновременно различные типы данных;
- разработчик отвечает за определение схемы именования ключей и за то, чтобы значение имело соответствующий тип/формат.

6. Документная база данных

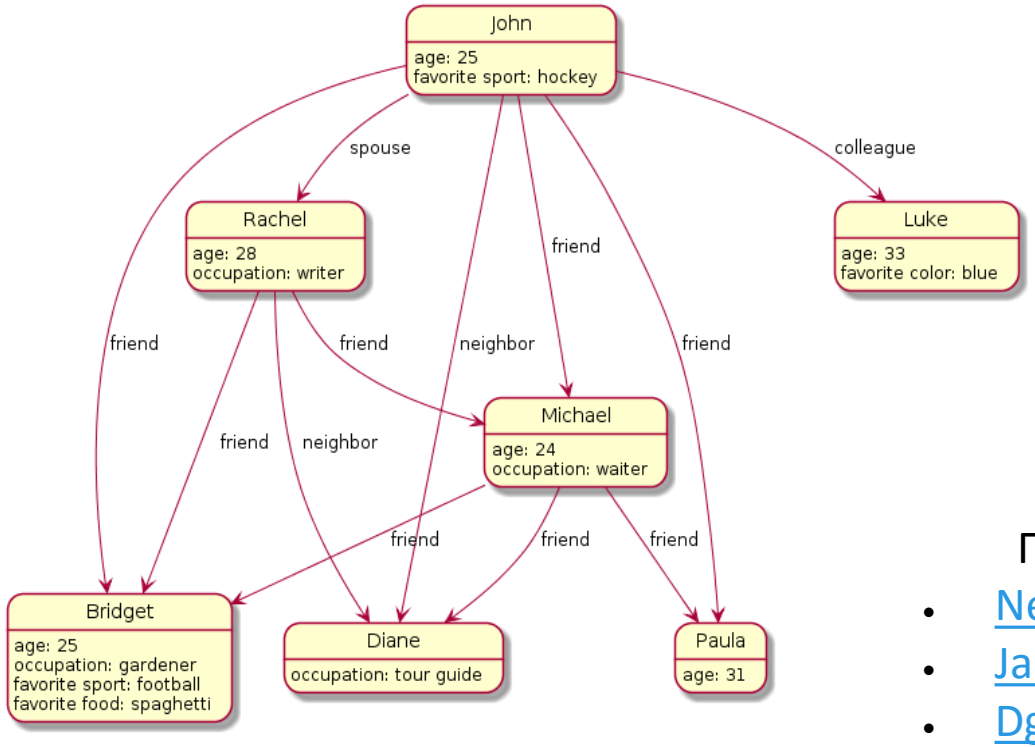


- база данных не предписывает определённый формат или схему;
- каждый документ может иметь свою внутреннюю структуру;
- документные БД являются хорошим выбором для быстрой разработки;
- в любой момент можно менять свойства данных, не изменяя структуру или сами данные.

Примеры:
- MongoDB
- RethinkDB

NoSQL базы данных

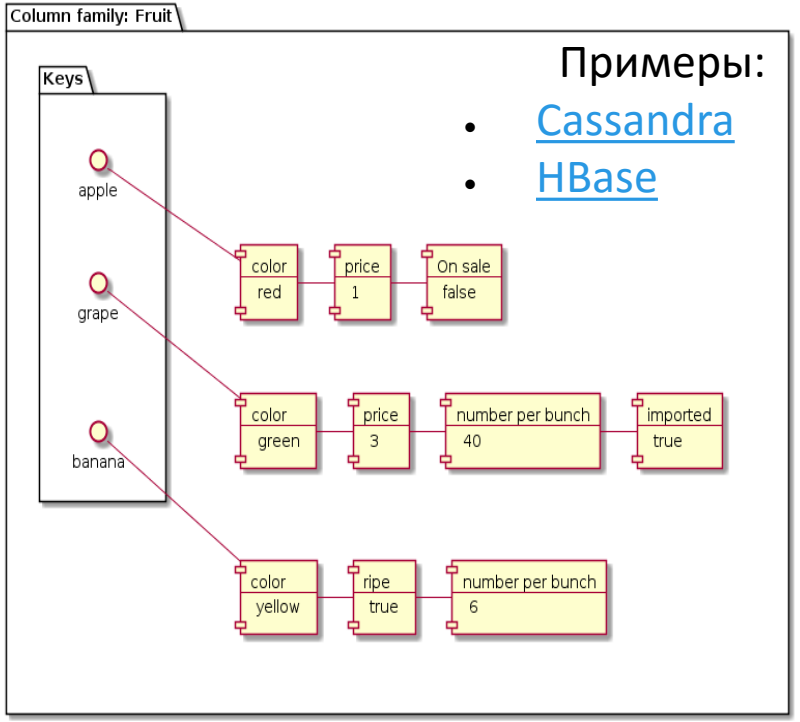
7. Графовая база данных



Примеры:

- Neo4j
- JanusGraph
- Dgraph

8. Колоночные базы данных



Примеры:

- Cassandra
- HBase

NoSQL базы данных

9. Базы данных временных рядов

Time	CPU Temp	System Load	Memory Usage %
2019-10-31T03:48:05+00:00	37	0.85	92
2019-10-31T03:48:10+00:00	42	0.87	90
2019-10-31T03:48:15+00:00	33	0.74	87
2019-10-31T03:48:20+00:00	34	0.72	77
2019-10-31T03:48:25+00:00	40	0.88	81
2019-10-31T03:48:30+00:00	42	0.89	82
2019-10-31T03:48:35+00:00	41	0.88	82

Преимущества

- ориентированы на запись;
- предназначены для обработки постоянного потока входных данных;
- производительность зависит от количества отслеживаемых элементов, интервала опроса между записью новых значений и фактической полезной нагрузки данных.

Примеры

- OpenTSDB
- Prometheus
- InfluxDB
- TimescaleDB

NewSQL базы данных

NewSQL базы данных наследуют реляционную структуру и семантику, но построены с использованием более современных, масштабируемых конструкций.

Цель – обеспечить большую масштабируемость, нежели реляционные БД, и более высокие гарантии согласованности, чем в NoSQL. Компромисс между согласованностью и доступностью является фундаментальной проблемой распределённых баз данных, описываемой теоремой CAP.

Преимущества

- возможность горизонтального масштабирования;
- высокая доступность;
- большая производительность и репликация;
- небольшой функционал и гибкость;
- немалое потребление ресурсов и необходимость специализированных знаний для работы с базой данных.

Примеры:

- MemSQL
- VoltDB
- Spanner
- Calvin
- CockroachDB
- FaunaDB
- yugabyteDB

Многомодельные базы данных

Многомодельные базы данных – базы, объединяющие функциональные возможности нескольких видов БД. Преимущества такого подхода очевидны – одна и та же система может использовать различные представления для разных типов данных.

Преимущества

- помогают уменьшить нагрузку на СУБД;
- позволяют расширяться до новых моделей по мере изменения потребностей без внесения изменений в базовую инфраструктуру;
- обеспечивают непрерывный доступ и простое распределение данных;
- имеют линейную масштабируемость и просты для разработки.

Примеры

- ArangoDB
- OrientDB
- Couchbase

Список источников

https://www.youtube.com/watch?v=wRaHyWMTlmw

https://mail.ru/search?search_source=mailru_desktop_safe&msid=1&suggest_reqid=94495414716034771051969969753778 3&serp_path=%2Fvideo%2Fpreview%2F8855136859856382252&type=video