МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

Требования к распределенных систем обработки больших данных, моделям данных и доступа

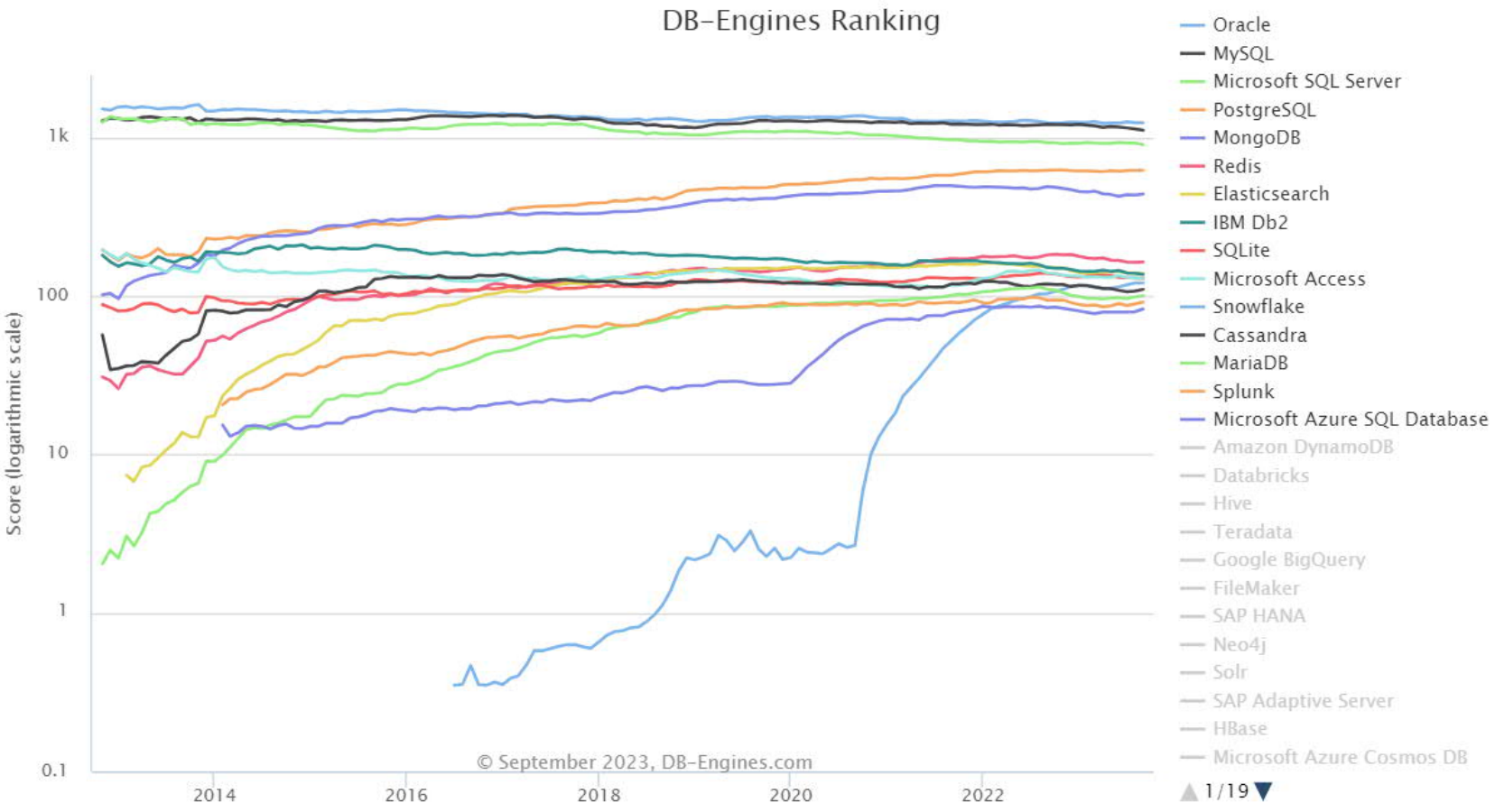
Краткий обзор рейтинговых систем

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

1. DB-Engines Ranking — рейтинг, который учитывает количество запросов в поисковых системах Google и Bing, позиции в Google Trends, упоминания в Stack Overflow и DBA Stack Exchange и другие показатели. На основе анализа формируется итоговый индекс популярности БД, определяющий ее позицию в общем рейтинге.

Краткий обзор рейтинговых систем

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ



https://db-engines.com/en/ranking_trend

Краткий обзор рейтинговых систем

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

422 systems in ranking, September 2023

Rank			DBMS	Database Model	Score		
Sep 2023	Aug 2023	Sep 2022			Sep 2023	Aug 2023	Sep 2022
1.	1.	1.	Oracle	Relational, Multi-model	1240.88	-1.22	+2.62
2.	2.	2.	MySQL	Relational, Multi-model	1111.49	-18.97	-100.98
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	902.22	-18.60	-24.08
4.	4.	4.	PostgreSQL	Relational, Multi-model	620.75	+0.37	+0.29
5.	5.	5.	MongoDB	Document, Multi-model	439.42	+4.93	-50.21
6.	6.	6.	Redis	Key-value, Multi-model	163.68	+0.72	-17.79
7.	7.	7.	Elasticsearch	Search engine, Multi-model	138.98	-0.94	-12.46
8.	8.	8.	IBM Db2	Relational, Multi-model	136.72	-2.52	-14.67
9.	↑ 10.	↑ 10.	SQLite	Relational	129.20	-0.72	-9.62
10.	↓ 9.	↓ 9.	Microsoft Access	Relational	128.56	-1.78	-11.47
11.	11.	↑ 13.	Snowflake	Relational	120.89	+0.27	+17.39
12.	12.	↓ 11.	Cassandra	Wide column, Multi-model	110.06	+2.67	-9.06
13.	13.	↓ 12.	MariaDB	Relational, Multi-model	100.45	+1.80	-9.70
14.	14.	14.	Splunk	Search engine	91.40	+2.42	-2.65
15.	↑ 16.	↑ 16.	Microsoft Azure SQL Database	Relational, Multi-model	82.73	+3.22	-1.69
16.	↓ 15.	↓ 15.	Amazon DynamoDB	Multi-model	80.91	-2.64	-6.51
17.	↑ 18.	↑ 20.	Databricks	Multi-model	75.18	+3.84	+19.56
18.	↓ 17.	↓ 17.	Hive	Relational	71.83	-1.52	-6.60

https://db-engines.com/en/ranking_trend

Популярные системы управления базами данных

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

1. Oracle

Oracle RDBMS (она же Oracle Database) на первом месте среди СУБД. Система популярна у разработчиков, проста в использовании, у нее понятная документация, поддержка длинных наименований, JSON, улучшенный тег списка и Oracle Cloud.

Особенности

- Обрабатывает большие данные.
- Поддерживает SQL, к нему можно получить доступ из реляционных БД Oracle.
- Oracle NoSQL Database с Java/C API для чтения и записи данных.

Популярные системы управления базами данных

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

2. MySQL

Эта система управления базами данных использует стандартную форму SQL. Утилиты для проектирования таблиц имеют интуитивно понятный интерфейс. MySQL поддерживает до 50 миллионов строк в таблице. Предельный размер файла для таблицы по умолчанию 4 ГБ, но его можно увеличить. Поддерживает секционирование и репликацию, а также Xpath и хранимые процедуры, триггеры и представления.

Особенности

Масштабируемость. Лёгкость использования. Безопасность.
Поддержка Novell Cluster. Скорость.
Поддержка многих операционных систем.

Популярные системы управления базами данных

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

3. Microsoft SQL Server

Самая популярная коммерческая СУБД. Она привязана к Windows, но это плюс, если вы пользуетесь продуктами Microsoft. Зависит от платформы. И графический интерфейс, и программное обеспечение основаны на командах. Поддерживает SQL, непроцедурные, нечувствительные к регистру и общие языки баз данных.

Особенности

Высокая производительность. Зависимость от платформы.
Возможность установить разные версии на одном компьютере.
Генерация скриптов для перемещения данных.

Популярные системы управления базами данных

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

4. PosgreSQL

Самая популярная коммерческая СУБД. Она привязана к Windows, но это плюс, если вы пользуетесь продуктами Microsoft. Зависит от платформы. И графический интерфейс, и программное обеспечение основаны на командах. Поддерживает SQL, непроцедурные, нечувствительные к регистру и общие языки баз данных.

Особенности

Поддержка табличных пространств, а также хранимых процедур, объединений, представлений и триггеров.
Восстановление на момент времени (PITR).
Асинхронная репликация.

Популярные системы управления базами данных

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

5. MongoDB

Самая популярная NoSQL система управления базами данных. Лучше всего подходит для динамических запросов и определения индексов. Гибкая структура, которую можно модифицировать и расширять. Поддерживает Linux, OSX и Windows, но размер БД ограничен 2,5 ГБ в 32-битных системах. Использует платформы хранения MMAPv1 и WiredTiger.

Особенности

Высокая производительность. Автоматическая фрагментация.
Работа на нескольких серверах. Поддержка репликации Master-Slave.
Данные хранятся в форме документов JSON.
Возможность индексировать все поля в документе.
Поддержка поиска по регулярным выражениям.

Популярные системы управления базами данных

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

6. DB2

Работает на Linux, UNIX, Windows и мейнфреймах. Эта СУБД идеально подходит для хост-сред IBM. Версию DB2 Express-C нельзя использовать в средах высокой доступности (при репликации, кластеризации типа active-passive и при работе с синхронизируемым доступом к разделяемым данным).

Особенности

Улучшенное встроенное шифрование.
Упрощённая установка и развёртывание.

Популярные системы управления базами данных

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

7. Cassandra

СУБД активно используется в банковском деле, финансах, а также в Facebook и Twitter. Поддерживает Windows, Linux и OSX. Для запросов к БД Cassandra используется SQL-подобный язык — Cassandra Query Language (CQL).

Особенности

Линейная масштабируемость.
Быстрое время отклика.
Поддержка MapReduce и Apache Hadoop.
Максимальная гибкость.
P2P архитектура.

Популярные системы управления базами данных

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

8. Redis

Redis или Remote Dictionary Server — СУБД с открытым исходным кодом, которая снабжена механизмами журналирования и снимков. Поддерживаются списки, строки, хэши, наборы. Используется для БД, брокеров сообщений и кэшей. Все операции в Redis атомарные. Система написана на языке C и поддерживается практически всеми языками программирования.

Особенности

Автоматическая обработка отказа. Транзакции.
Сценарии LUA. Вытеснение LRU-ключей.
Поддержка Publish/Subscribe.

Популярные системы управления базами данных

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

9. Elasticsearch

Легко масштабируемая поисковая система корпоративного уровня с открытым исходным кодом. Благодаря обширному и продуманному API обеспечивает чрезвычайно быстрый поиск, работает в том числе с приложениями для обнаружения данных. Используется такими компаниями, как Википедия, The Guardian, StackOverflow, GitHub. ElasticSearch позволяет создавать копии индексов и сегментов.

Особенности

Масштабируемость вплоть до нескольких петабайт структурированных и неструктурированных данных.
Многопользовательская поддержка.
Масштабируемый поиск, поиск в режиме реального времени.

Требования к системе данных

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Хранение данных: требуется хранить данные, а также иметь возможность найти их позже (база данных).

Запрос к данным: необходимо иметь возможность эффективно запрашивать и фильтровать данные определенными способами (транзакциями и индексами).

Сохранение и производительность. Требуются быстрые результаты, особенно в отношении дорогостоящих операций чтения (кэширования).

Обработка данных: хотим иметь возможность обрабатывать огромное количество данных (пакетная обработка), а также обрабатывать данные асинхронно (потоковая обработка).

Требования к системе данных

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- те же требования, что и для первой базы данных CODASYL (еще в 1960-х годах);
- хотя в прошлом было много баз данных - все они по-прежнему соответствуют одним и тем же требованиям.

- Эволюция:

- реляционные базы данных, способные обрабатывать данные NoSQL (например, такие как IBM DB2 или Oracle), а также базы данных NoSQL, способные обрабатывать традиционный SQL (например, ToroDB).
- базы данных становятся очередями сообщений (например, RethinkDB или Redis), и наоборот, системы очередей сообщений становятся базами данных (например, Apache Kafka).
- границы размыты.

Масштабируемость

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Масштабируемость — это способность системы обработки данных справляться с растущим объемом нагрузки (например, с большим объемом данных, необходимых для хранения или обработки запросов).

Система данных считается масштабируемой, если она способна увеличить общую пропускную способность/выход при увеличении нагрузки при добавлении ресурсов (обычно аппаратных).

Масштабируемость — нагрузка

Нагрузка системы данных — это измерение объема вычислительной работы, которую она выполняет (в зависимости от используемой архитектуры), например. количество (одновременных) операций чтения из хранилища данных, операций записи в хранилище данных или соотношение между операциями чтения и записи.

Максимальная нагрузка определяется самой слабой частью архитектуры (=**узким местом**).

- **запрос/секунду на сайт**;
- **запросы на чтение/запись в секунду в систему данных**;
- **отношение чтения/записи системы данных.**

Масштабируемость — производительность

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Производительность системы данных определяется пропускной способностью системы и временем отклика, например. количество транзакций (таких как операции чтения/записи), обработанных записей (таких как агрегация для аналитических целей) или даже системных команд (таких как обновление статистики или повторная балансировка нескольких узлов данных) **при заданной рабочей нагрузке и за определенное время – временной промежуток - фрейм.**

Обычно это зависит от множества как влияющих, так и не влияющих факторов самой системы, таких как задержка в сети, сбой страницы или поврежденный диск.

Масштабируемость — измерение производительности

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Пропускная способность:

- **чтение/запись в секунду** (в случае MongoDB до 100 000 чтений/пишет в секунду);
- **Обработка сообщений**(в случае Apache Kafka и Linkedin более 2 миллионов записей в секунду всего на 3 узлах);
- **Обработка данных** (в случае Apache Hadoop и Spark терабайты данных в течение нескольких секунд).

Время ответа:

- **чтение/запись в секунду** (в случае MongoDB до 100 000 чтений/пишет в секунду).

Масштабируемость — показатели производительности

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Среднее арифметическое:

- легко рассчитать,
- игнорирует коэффициенты,
- на него сильно влияют статистические выбросы, поэтому он не может сказать вам, сколько запросов, операций чтения или записи на самом деле имели худшую производительность.

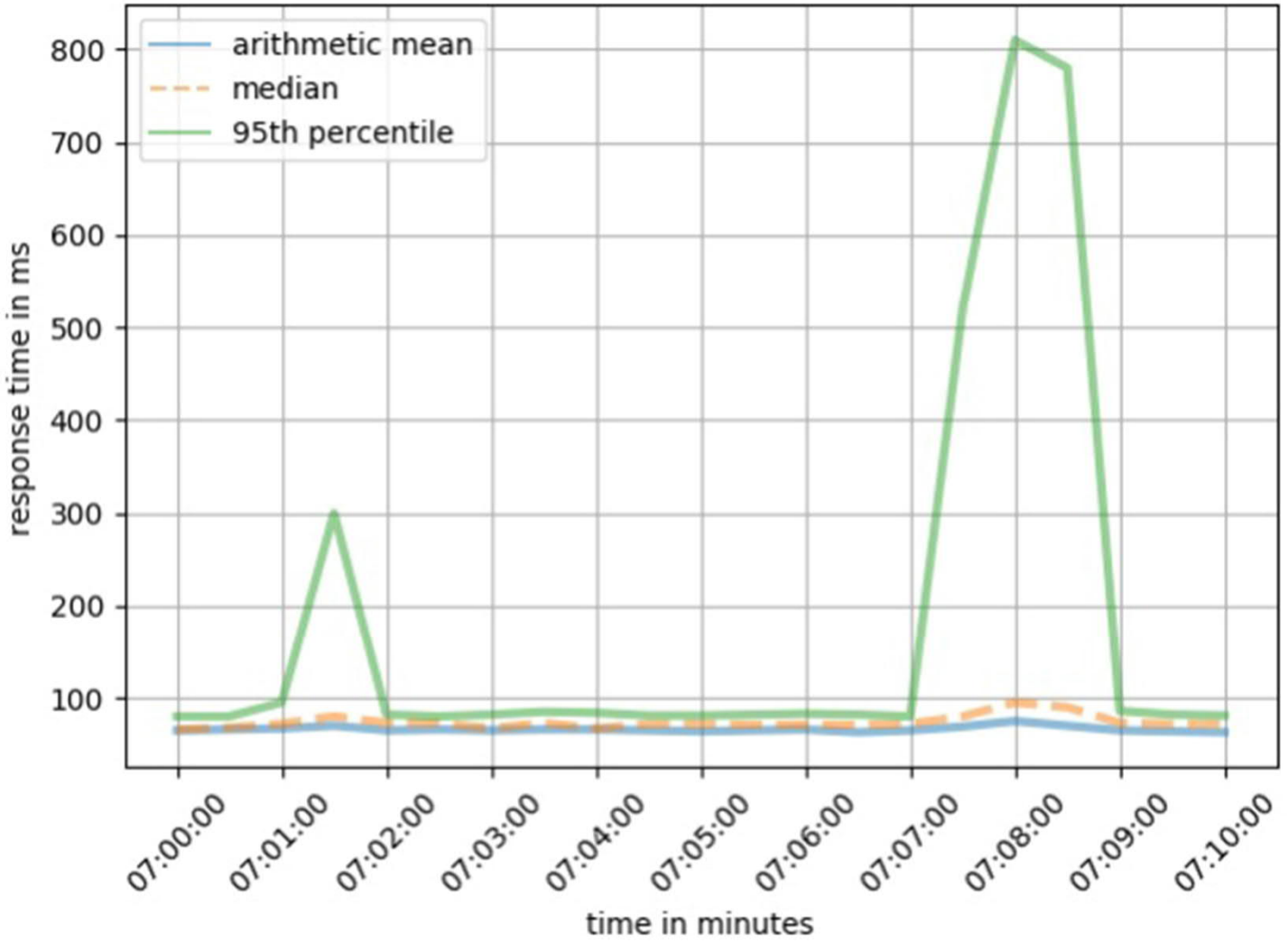
Медиана:

- легко рассчитать,
- меньше искажается выбросами,

Процентили (например, процентили p95, p99, p999):

- легко рассчитать,
- не искажается выбросами.

Масштабируемость — показатели производительности



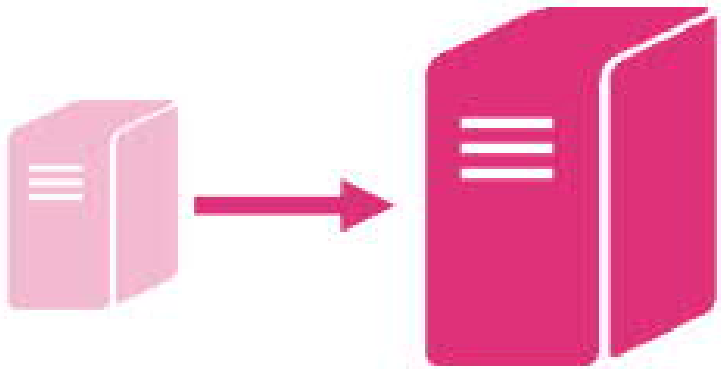
МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

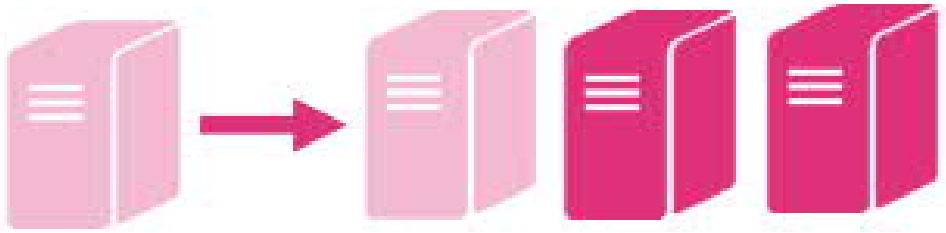
Масштабируемость — подходы к масштабированию

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Вертикальное масштабирование: заменить сервер на один более мощный

Горизонтальное масштабирование: распределять нагрузку на несколько серверов вместо одного





На практике – всегда гибридное масштабирование!

Надежность

Надежность с точки зрения аппаратного обеспечения, программного обеспечения или особенно систем данных может быть определена как **способность системы функционировать в соответствии с требованиями и ожиданиями**.

Надежная система данных также обнаруживает и допускает сбои из-за ошибок пользователей, оборудования или более низких частей самой системы данных, а также обеспечивает требуемую производительность при любой ожидаемой нагрузке.

- **Типичные неисправности**:
- **Аппаратные неисправности**
- **Программные сбои**
- **Человеческие недостатки**

Надежность — аппаратные сбои

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- сломанные жесткие диски или твердотельные накопители,
- неисправная оперативная память или процессоры,
- сломанные адаптеры питания, коммутаторы или сбои всей сети,
- отсоединены сетевые кабели или даже подключены не к тому порту.

Представьте кластер Hadoop из 100 узлов с 19 жесткими дисками на узел = 1,900.HDD в целом. По данным BackBlaze, среднегодовая частота отказов жестких дисков составляет около 2,11%, **примерно каждую неделю** HDD **выходит из строя**.

Надежность - Человеческие ошибки

Самый ненадежный фактор: люди

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- Подходы к тому, чтобы сделать систему данных надежной, даже с точки зрения человека:

- **Разделить окружения**, где совершается максимальное количество ошибок от мест, где потенциально может быть внесена ошибка, например. с использованием разработки среды или предоставление интерфейсов или фреймворков для API, вместо прямого доступа к API
- **использовать тестирование** (например, модульное тестирование, системные тесты, интеграционные тесты) и автоматизировать их
- **измерение и мониторинг** (например, показателей производительности, частоты ошибок) всех способов проверки того, не нарушаются ли допущения или ограничения на ранней стадии.

Ремонтопригодность

Обслуживание = одна из самых больших затрат
при разработке программного обеспечения

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Система данных должна быть спроектирована таким образом, чтобы свести к минимуму затраты на техническое обслуживание.

3 основных принципа, которым нужно следовать:

1. **Оперативность: упростить запуск системы данных**

- **хорошая документация и операционная модель** (система данных, которую легко понять, легче эксплуатировать)
- **прозрачность** (видимость системы данных и поведения во время выполнения, например, с помощью файлов журнала или инструментов мониторинга)
- **отсутствие зависимостей между отдельными службами или сервером** (разрешить отключение одного сервера для задач обслуживания, например, исправлений, обновлений или перезапусков)
- **самовосстановление**, если возможно, но также возможность переопределения для операторов

Ремонтопригодность

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

2. Простота: упростить понимание системы данных.

- используйте абстракцию и уменьшите сложность (менее сложная не требует снижения функциональности, это больше связано с устранением ненужной сложности),
- **четко определенные интерфейсы,**
- **отсутствие чрезмерной инженерии.**

3. Гибкость: упростить адаптацию/изменение системы данных

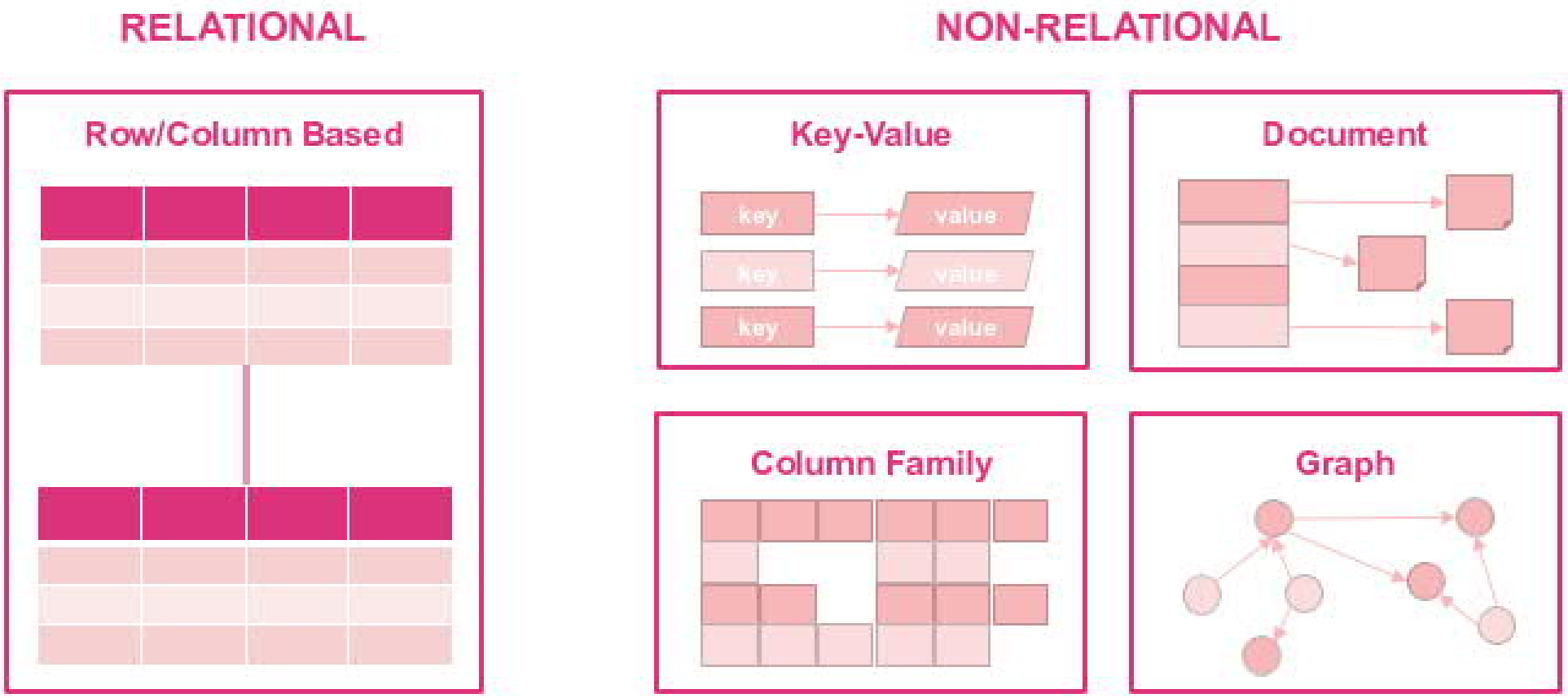
- продолжение интеграции,
- разработка через тестирование,
- парное программирование.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

Введение в модели данных и доступ

Модели данных (реляционные/нереляционные)

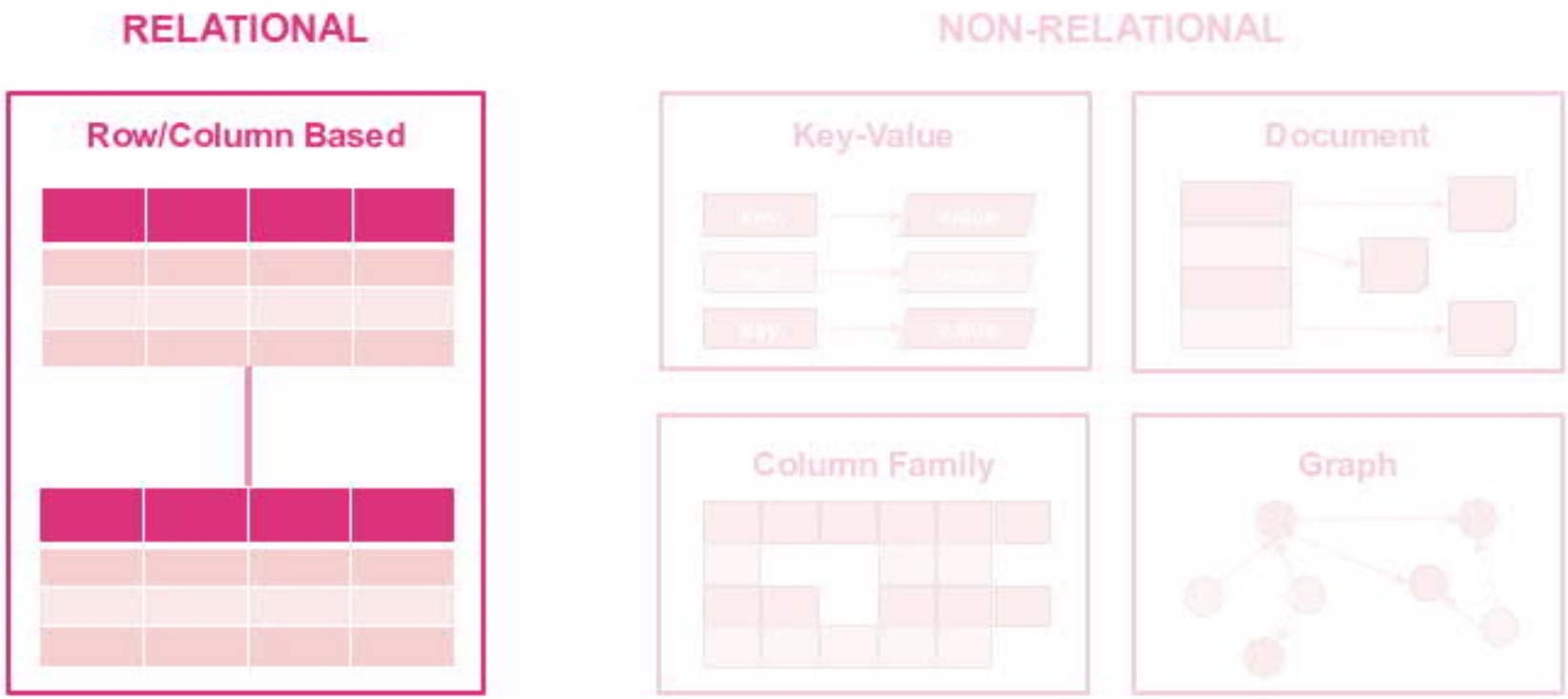
МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ



Реляционная модель данных

Впервые представлен Эдгаром Фрэнком Коддом в 1970

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ



Идея: скрыть детали реализации (представление данных в хранилище данных)
Путем предоставления: декларативного и читаемого на схеме интерфейса

Реляционная модель данных

Впервые представлен Эдгаром Фрэнком Коддом в 1970

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

Разработчики/пользователи могут легко указать, какую информацию содержит база данных и какую информацию они хотят получить.

База данных позаботится об описании, хранении и извлечении данных

Идея: скрыть детали реализации (представление данных в хранилище данных)
Путем предоставления: декларативного и читаемого на схеме интерфейса

Реляционная модель данных - Пример

-Пример: страница профиля Facebook как реляционная модель.

- Пример: страница профиля Facebook, как реляционная модель.

- table `user` = main entity
- stores *primary key* (`id`)
- stores basic information (e.g. *`first_name`*, *`last_name`*)

table: ***user***

id	first_name	last_name	is_verified	married_to	...
1	Mark	Zuckerberg	true	4711	
4711	Priscilla	Chan	true	1	

table: ***universities***

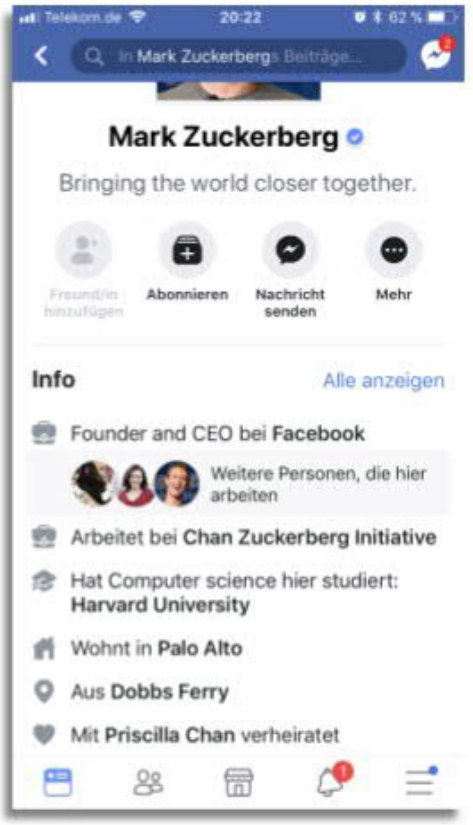
id	user_id	university_name	subject	...
1378	1	Harvard University	Computer Science	
1379	4711	University of California	Medicin	

table: ***companies***

id	user_id	company_name	...
1337	1	Facebook	
1338	1	Chan Zuckerberg Initiative	
1339	4711	UCSF Benioff Children's Hospital	

table: ***cities***

id	user_id	city_name	status	...
1378	1	Palo Alto	living	
1379	1	Dobbs Ferry	born	
1379	4711	Palo Alto	living	



Так как у людей может быть:

- работа в нескольких компаниях,
- учеба в нескольких университетах,
- Проживание в разных городах.

→Разделение таблиц через *foreign key* (`user_id`):

- `companies`
- `universities`
- `cities`

Реляционная модель данных — список программного обеспечения

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

List of Software [edit]

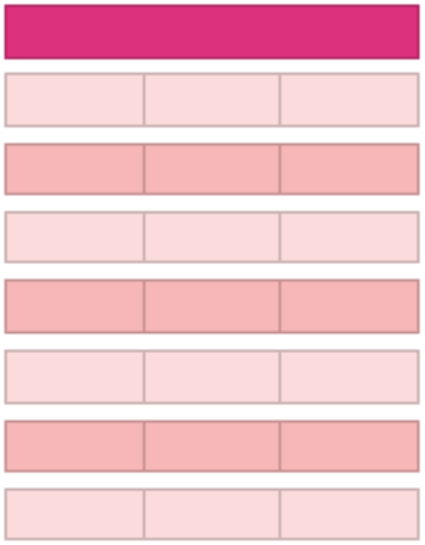
- 4th Dimension
- Adabas D
- Alpha Five
- Apache Derby
- Aster Data
- Amazon Aurora
- Altibase
- CA Datacom
- CA IDMS
- Clarion
- ClickHouse
- Clustrix
- CSQL
- CUBRID
- DataEase
- Database Management Library
- Dataphor
- dBase
- Derby aka Java DB
- Empress Embedded Database
- EXASolution
- EnterpriseDB
- eXtremeDB
- FileMaker Pro
- Firebird
- FrontBase
- Google Fusion Tables
- Greenplum
- GroveSite
- H2
- Helix database
- HSQLDB
- IBM DB2
- IBM Lotus Approach
- IBM DB2 Express-C
- Infobright
- Informix
- Ingres
- InterBase
- InterSystems Caché
- LibreOffice Base
- Linter
- MariaDB
- MaxDB
- MemSQL
- Microsoft Access
- Microsoft Jet Database Engine (part of Microsoft Access)
- Microsoft SQL Server
- Microsoft SQL Server Express
- SQL Azure (Cloud SQL Server)
- Microsoft Visual FoxPro
- Mimer SQL
- MonetDB
- mSQL
- MySQL
- Netezza
- NexusDB
- NonStop SQL
- NuoDB
- Omnis Studio
- Openbase
- OpenLink Virtuoso (Open Source Edition)
- OpenLink Virtuoso Universal Server
- OpenOffice.org Base
- Oracle
- Oracle Rdb for OpenVMS
- Panorama
- Pervasive PSQL
- Polyhedra
- PostgreSQL
- Postgres Plus Advanced Server
- Progress Software
- RDM Embedded
- RDM Server
- R:Base
- SAND CDBMS
- SAP HANA
- SAP Adaptive Server Enterprise
- SAP IQ (formerly known as Sybase IQ)
- SQL Anywhere (formerly known as Sybase Adaptive Server Anywhere and Watcom SQL)
- solidDB
- SQLBase
- SQLite
- SQream DB
- SAP Advantage Database Server (formerly known as Sybase Advantage Database Server)
- Teradata
- Tibero
- TimesTen
- Trafodion
- txtSQL
- Unisys RDMS 2200
- UniData
- UniVerse
- Vectorwise
- Vertica
- VoltDB

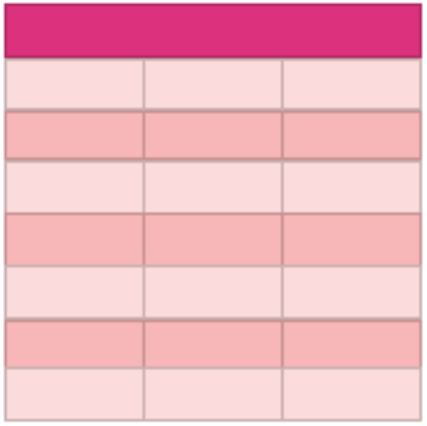
Реляционная модель данных — на основе строк(Row) и столбцов(Column-Based)

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

Row-Based:

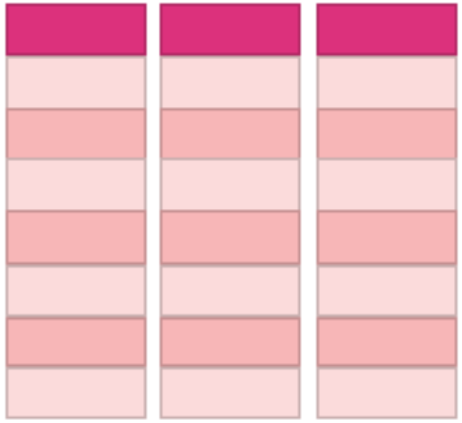
→ строки хранятся непрерывно
→ Oracle, IBM DB2, Microsoft SQL Server, MySQL





Column-Based:

→ столбцы хранятся непрерывно
→ Hbase, Parquet, SAP HANA, Teradata



Реляционная модель данных – строка или столбец

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Операция	На основе строк	На основе столбцов
Сжатие	Low	High
Сканирование столбцов	Slow (Multiple Reads)	Fast (One Read)
Вставка записей	Fast (One Insert)	Slow (Multiple Inserts)
Запросы по одной записи	Fast (One Read)	Slow (Multiple Reads)
Агрегация по одному столбцу	Slow (Full Table Scan)	Fast (Only Column Scan)
Типичные случаи использования	Транзакции	Аналитика

Реляционная модель данных — SQL

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Первоначально представлено как «SEQUEL» Дональдом Д. Чемберлином и Рэймондом Ф. Бойсом в1974 г., вдохновленный реляционной моделью данных Эдгара Фрэнка Коддса 1970 г.

- Стандарт ANSI 1958 г., ISO 1987 г. наиболее широко используемый на сегодняшний день язык баз данных даже системами данных NoSQL, например:
  - Apache Cassandra (CQL),
  - Apache Hadoop (HiveQL),
  - Apache Flink (FQL),

Реляционная модель данных — SQL

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Язык

SQL = декларативный
- не нужно беспокоиться о том, как данные хранятся и извлекаются,
- автоматическая оптимизация производительности (оптимизатор запросов),

императив = например MapReduce, Spark, …
- нужно подумать о хранении данных,
- нет оптимизатора запросов.

Операторы DML :

CREATE/DROP/ALTER TABLE…
INSERT INTO … VALUES…
UPDATE … SET … WHERE…
DELETE FROM .. WHERE…
…

Structure:

```
SELECT      -- attributes
FROM        -- tables
WHERE       -- conditions
GROUP BY    -- grouping attributes
HAVING      -- grouping conditions
ORDER BY    -- attributes
```

Keywords:

DISTINCT, LIMIT, AS
MIN, MAX, AVG, COUNT, SUM
AND, OR, NOT, IN, LIKE, ANY
UNION, JOIN
ASC, DESC
…

Реляционная модель данных — пример SQL

Query:

```
select m.tconst, m.original_title, m.start_year, r.average_rating, r.num_votes
from imdb_movies m JOIN imdb_ratings r on (m.tconst = r.tconst) WHERE r.average_rating > 6
and m.start_year > 2010 and m.title_type = 'movie' and r.num_votes > 100000
ORDER BY r.average_rating desc, r.num_votes desc LIMIT 200
```

Result:

	m.tconst	m.original_title	m.start_year	r.average_rating	r.num_votes
1	tt0816692	Interstellar	2.014	8,6	1.213.141
2	tt4154756	Avengers: Infinity War	2.018	8,6	492.952
3	tt1675434	Intouchables	2.011	8,5	635.669
4	tt2582802	Whiplash	2.014	8,5	571.172
5	tt5074352	Dangal	2.016	8,5	105.207
6	tt1345836	The Dark Knight Rises	2.012	8,4	1.331.811
7	tt1853728	Django Unchained	2.012	8,4	1.153.183
8	tt2380307	Coco	2.017	8,4	219.957
9	tt5311514	Kimi no na wa.	2.016	8,4	108.553
10	tt2106476	Jagten	2.012	8,3	226.819
11	tt1832382	Jodaeiye Nader az Simin	2.011	8,3	186.217
12	tt0993846	The Wolf of Wall Street	2.013	8,2	976.551
13	tt2096673	Inside Out	2.015	8,2	499.721
14	tt1291584	Warrior	2.011	8,2	389.935
15	tt3170832	Room	2.015	8,2	288.153
16	tt5027774	Three Billboards Outside Ebbing, Missouri	2.017	8,2	280.790

Реляционная модель данных — SQL

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Сильные стороны:

- ACID (Atomicity, Consistency, Isolation, Durability) — описание требований к СУБД, системе управления **базами данных**: атомарность, согласованность, изолированность и прочность. Следование требованиям ACID обеспечивает надежную и предсказуемую работу транзакционных систем.
- **Универсальность** – множество типов данных, связанные и несвязанные данные, «независимость» от РБД.
- **Строгая схема качества данных**, предотвращение ошибок, сжатие.

Недостатки:

- Строгая схема:
  Необходимо постоянно менять при любом изменении формата данных - данные необходимо перенести в другую ИС.
- Нужны ORM, обычно замедляет и усложняет доступ к данным.

Реляционная модель данных – преимущества/недостатки

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Извлечение данных:
строки можно легко вставлять и отображать, используя идентификатор или первичный/внешний ключ,
в отличие, например, от документно-ориентированной модели данных, в которых обычно необходимо:
- при доступе к данным требуется думать о вложенных структурах,
плохо структурируемые пустые поля, поскольку эти системы обычно считываются по схеме, или необходимо тщательно подумать о возможных проблемах с производительностью и о том, как система, вероятно, выполнит запрос, поскольку механизм выполнения обычно не настолько развит, как механизм реляционной системы,
сложнее понять, как работают системы, поскольку иногда даже нет функции запроса-пояснения, как в реляционных базах данных, поэтому не возможно заранее изучить, как она будет подробно вести себя во время выполнения.

Реляционная модель данных – преимущества/недостатки

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Объектно-ориентированность:

приложениям нужен уровень трансляции для системы данных: ***ORM Frameworks****, например:*

- ***Hibernate*** *в случае* ***Java.***
- ***SQLAlchemy*** *в случае* ***Python.***

Отношения «многие-ко-многим»:

- реляционная модель данных: недорогие ограничения внешнего ключа (индексы и т. д.).

- *Модель данных NoSQL: вложенные структуры, ссылки на документы и дорогостоящие операции* IO/CPU.

Записю/обновление:

-реляционная модель данных: эффективна, поскольку будут обновляться отдельные строки.

-Модель данных NoSQL: обычно требует перезапись всего документа.

Реляционная модель данных – преимущества/недостатки

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Ограничения:

-реляционная модель данных:

-Типы данных.

-Первичные/внешние ключи.

Обеспечивает целостность данных.

-Модель данных NoSQL:

- Нет или меньше ограничений

Свобода, но уязвимость перед входом и выходом мусора.

Операции:

-реляционная модель данных:

реляционная алгебра.

-Модель данных NoSQL:

зависит от системы данных.

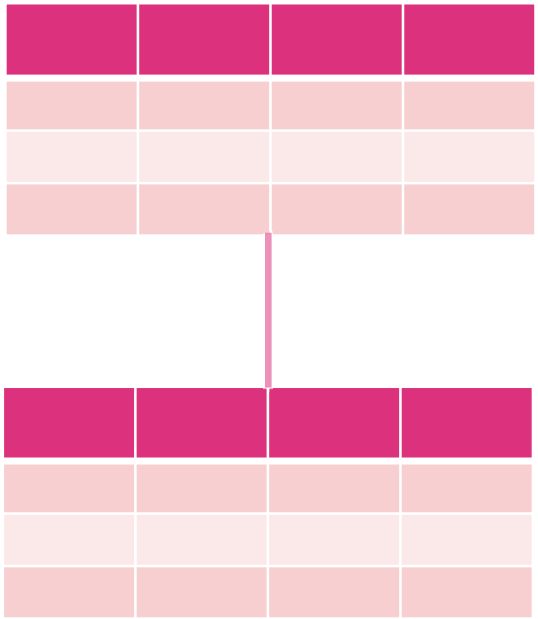
Нереляционная модель данных

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

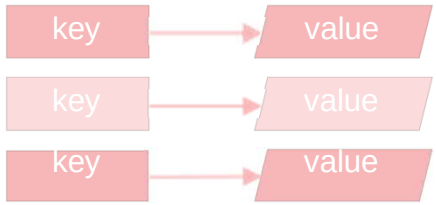
RELATIONAL

NON-RELATIONAL

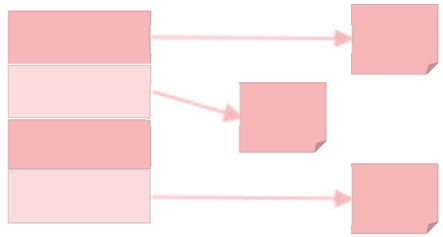
Row/Column Based



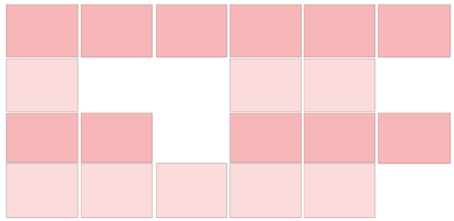
Key-Value



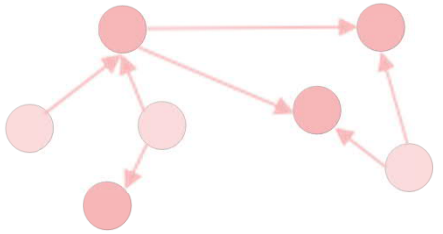
Document



Column Family

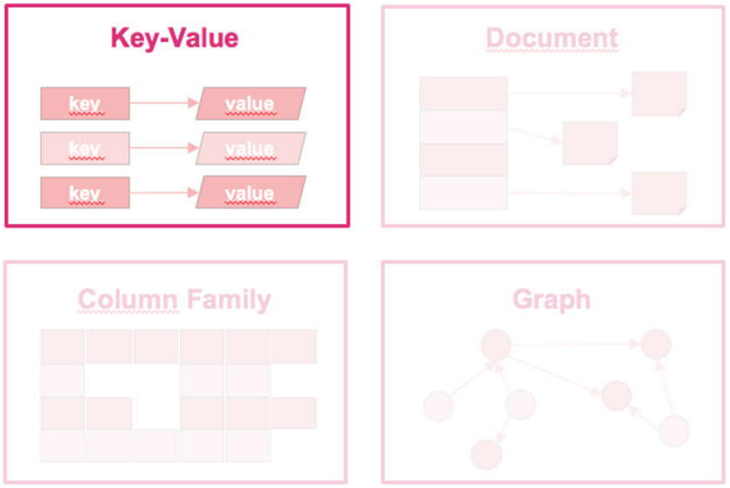


Graph



Нереляционная модель данных - Key-Value

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ



- Примеры:
  - Redis,
  - BerkeleyDB,
  - VoldemortDB,
  - ArangoDB,
  - Riak, …

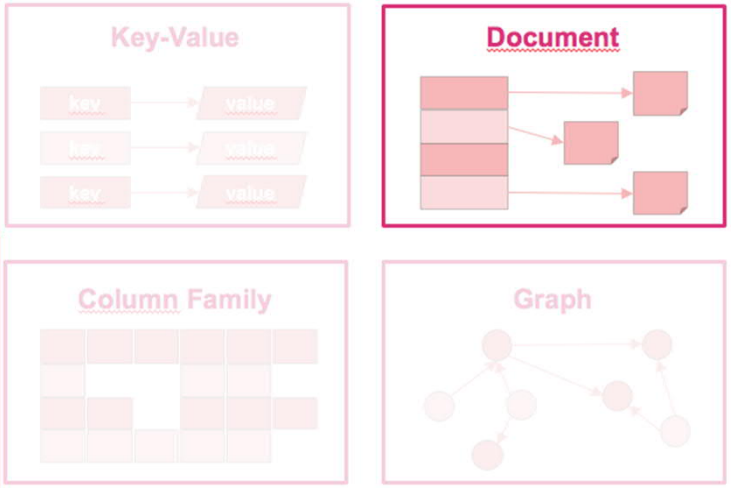
- Сильные стороны:
  - быстрые запросы,
  - быстрая вставка,
  - легкая репликация и распределение запросов

- Слабые стороны:
  - менее эффективные запросы:
    - агрегация
    - фильтр
    - соединение

    → Запросы делают через

    → внешние приложения

Нереляционная модель данных – Document



- Примеры:
  - MongoDB,
  - Couchbase,
  - RethinkDB,
  - ArangoDB,
  - DynamoDB,
  - CouchDB, …

- Сильные стороны :
  - гибкость схемы (документы могут иметь разные форматы),
  - быстрые вставки и единичные запросы запросы,
  - локальность данных,
  - легко тиражировать и распространять (например, для достижения балансировки нагрузки и отказоустойчивости),
  - простота кода приложения.
- Слабые стороны :
  - обновления документов обычно менее
  - эффективны/затратны на ввод-вывод,
  - менее эффективно и медленно:
    - агрегация
    - фильтр
    - соединение

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Нереляционная модель данных – Document

МОСКОВСКИЙ ГОРОДСКОЙ

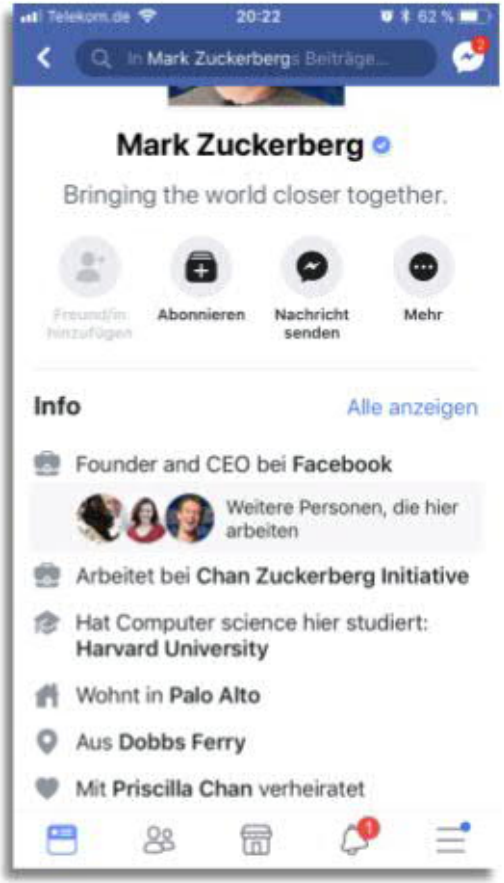
-Пример: страница профиля Facebook как модель документа.

- Нет необходимости получать несколько таблиц. (местоположение данных)

- Легко разбивать по приложениям (объектно-ориентированность)

- гибкость схемы

```
{
  "id": 1,
  "first_name": "Mark",
  "last_name": "Zuckerberg",
  "is_verified": "true",
  "married_to": 4711,
  "universities": [
    {
      "university_name": "Harvard University",
      "subject": "Computer Science"
    }
  ],
  "companies": [
    {
      "company_name": "Facebook"
    },
    {
      "company_name": "Chan Zuckerberg Initiative"
    }
  ],
  "cities": [
    {
      "city_name": "Palo Alto",
      "status": "living"
    },
    {
      "city_name": "Dobbs Ferry",
      "status": "born"
    }
  ]
}
```



Нереляционная модель данных – Document

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

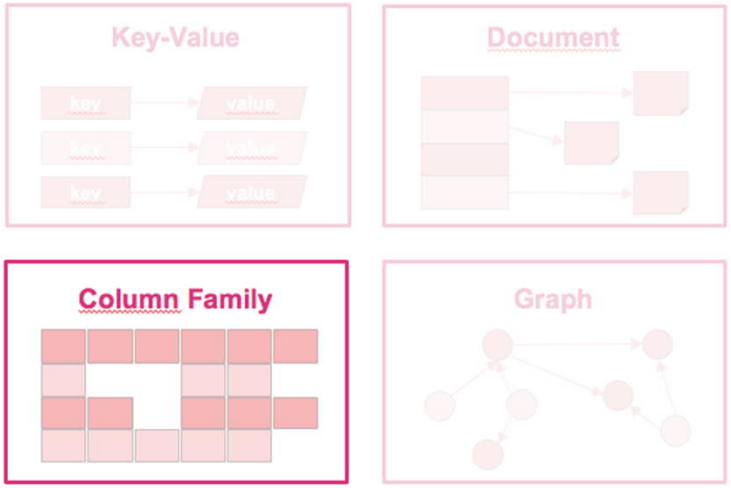
SQL	MongoDB
SELECT * FROM user	`db.user.find()`

SELECT id, first_name, last_name
FROM user WHERE is_verified = true
	`db.user.find(`
	`{ is_verified : true },`
	`{ id : 1, first_name: 1,`
	`last_name: 1 }`
	`)`

INSERT INTO user(id, first_name, last_name
VALUES (1, „Mark“, „Zuckerberg“)
	`db.user.insertOne(`
	`{ id : 1,`
	`first_name: “Mark“,`
	`last_name: “Zuckerberg“ }`
	`)`

Нереляционная модель данных –Колончатые

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

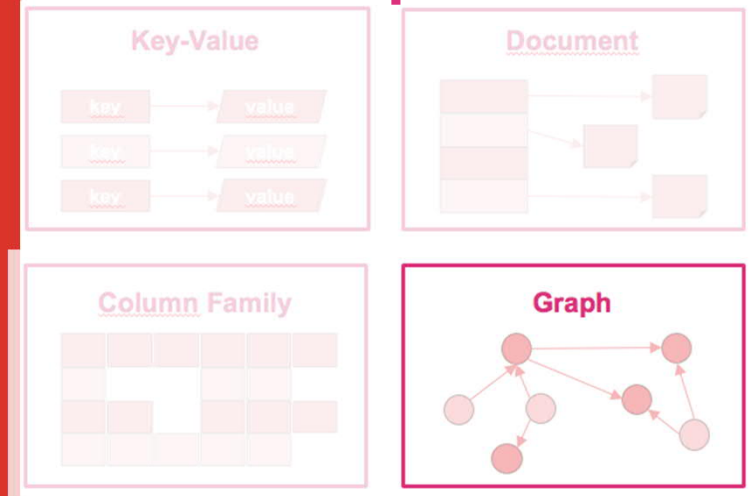


- Пример:
  - Cassandra,
  - BigTable
  - HBase

- Сильные стороны :
  - гибкость схемы (документы могут иметь разные форматы),
  - локальность данных,
  - легко тиражировать и распространять (например, для достижения балансировки нагрузки и отказоустойчивости),
  - простота кода приложения.
  - обновления документов обычно менее
  - эффективны/затратны на ввод-вывод,
  - менее эффективно и медленно:
    - агрегация
    - фильтр
    - соединение

Нереляционная модель данных – Graph

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ



- Examples:
  - Neo4j,
  - ArangoDB,
  - OrientDB,
  - Titan,
  - Giraph, …

- Сильные стороны :
  - отношения «многие ко многим» (в отличие от всех других моделей данных),
  - гибкость схемы (за счет ребер и определения свойств),
  - эффективные запросы отношений,
  - обычно поддерживают ACID.

- Weaknesses:
  - бесполезно для транзакционных данных или операций CRUD.

Нереляционная модель данных – MapReduce

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

MapReduce = парадигма программирования и связанная с ней реализация для параллельной обработки и создания больших наборов данных, распределенных в кластере, первоначально представленная Джеффри Дином и Санджаем Гемаватом (Google Inc.) в 2004 году.

MapReduce не является ни декларативным языком, ни императивным языком программирования, это нечто среднее между.

Парадигма основана на указании:

- функция map, которая выполняет фильтрацию и сортировку, в результате чего получается промежуточный набор пар ключ/значение и
- функция reduce, которая объединяет все промежуточные значения, связанные с одним и тем же ключом (например, суммирует все значения).
- Задания MapReduce автоматически распараллеливаются и выполняются на нескольких узлах кластера.

MapReduce

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Система контроля выполнения этапов MapReduce (например, YARN в случае Hadoop) заботится о:

-деталях предоставления и разделения входных данных,
-планировании выполнения кода приложения на всех узлах кластера,
-сохранении промежуточных состояний,
-обработке сбоев узлов,
-межузловой связи.

MapReduce – WordCount

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

```
1  map(String key, String value):
2    // key: document name
3    // value: document content
4    for each word w in value:
5      EmitIntermediate(w, 1);
6
7  reduce(String key, Iterator values):
8    // key: a word
9    // values: a list of counts
10   int result = 0;
11   for each v in values:
12     result += v;
13   Emit(key, result);
```

Функция map принимает одну пару данных с типом в одном домене данных и возвращает список пар в другом домене:

$$map(k1,v1) \rightarrow list(k2, v2)$$

Это создает список пар (с ключом k2) для каждого вызова.

После этого фреймворк MapReduce собирает все пары с одинаковым ключом (k2) из всех списков и группирует их вместе, создавая по одной группе для каждого ключа.

Функция reduce выполняется параллельно для каждой группы, которая, в свою очередь, создает коллекцию значений (v3) для связанного ключа (k2) в том же домене:

$$reduce(k2, list(v2)) \rightarrow list(k2, v3)$$

Возвраты всех процессов reduce собираются в виде списка требуемых результатов.

MapReduce – WordCount

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

```
1 document1 = "Da steh ich nun, ich armer Tor!";
2 document2 = "Und bin so klug als wie zuvor;";
3 document3 = "Heiße Magister, heiße Doktor gar";
4 document4 = "Und ziehe schon an die zehen Jahr";
5 document5 = "Herauf, herab und quer und krumm":
6 document6 = "Meine Schüler an der Nase herum.";
```

Code Snippet 2.2: MR Example - *Word Count (Input Documents)*

```
1 map(document1, "da steh ich nun ich armer tor");
2 map(document2, "und bin so klug als wie zuvor");
3 map(document3, "heiße magister heiße doktor gar");
4 map(document4, "und ziehe schon an die zehen jahr");
5 map(document5, "herauf, herab und quer und krumm");
6 map(document6, "meine schüler an der nase herum");
```

Code Snippet 2.3: MR Example - *Word Count (`map()` Calls)*

```
1 p1 = [ ("da",1), ("steh",1), ("ich",1), ("nun",1), ("ich",1),
2        ("armer",1), ("tor",1) ];
3 p2 = [ ("und",1), ("bin",1), ("so",1), ("klug",1), ("als",1),
4        ("wie",1), ("zuvor",1) ];
5 ...
6 p6 = [ ("meine",1), ("schüler",1), ("an",1), ("der",1), ("nase",1),
7        ("herum",1)];
```

Code Snippet 2.4: MR Example - *Word Count (Partial `map()` Results)*

MapReduce – WordCount

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

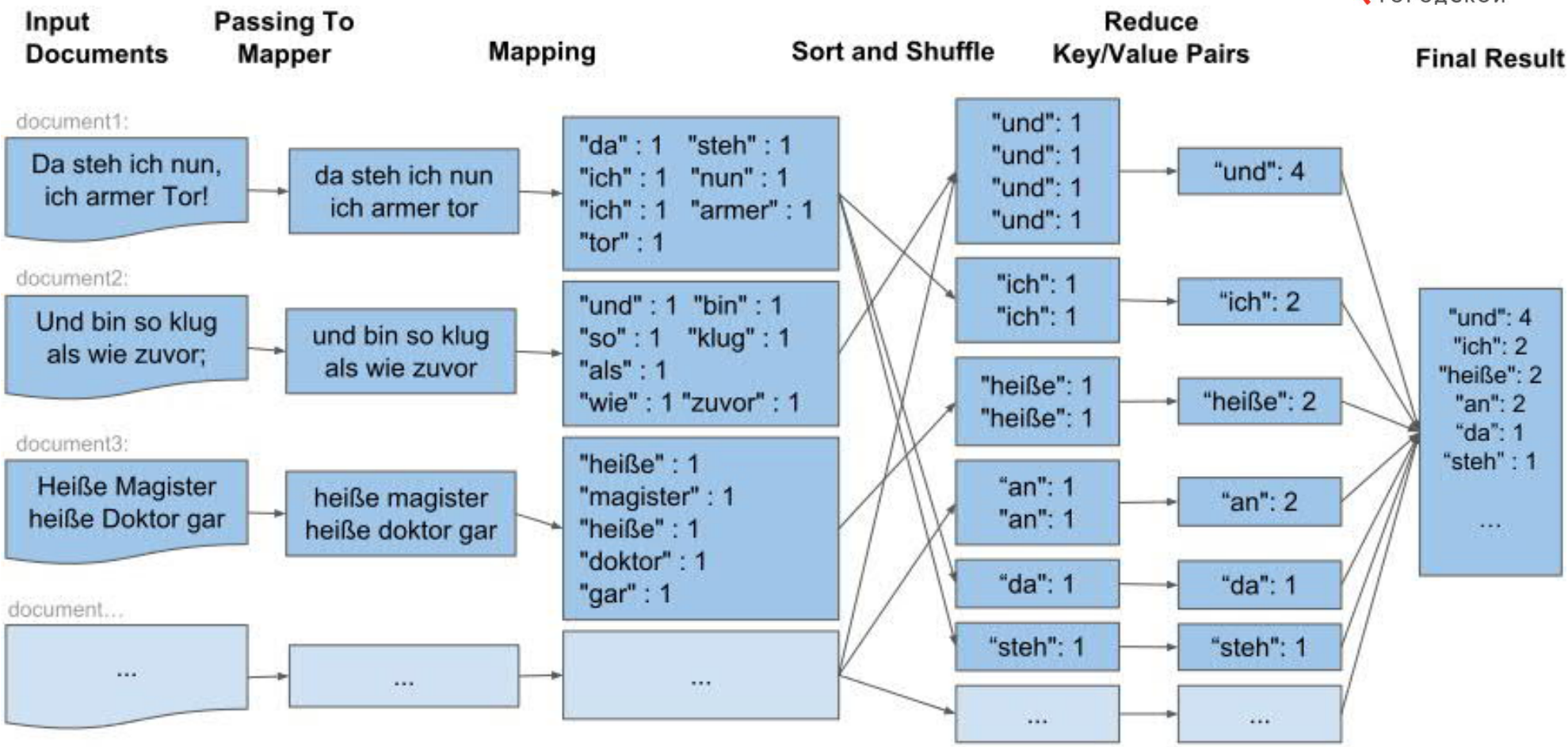
```
reduce("da", [1]); // = ("da", 1)
reduce("ich", [1,1]); // = ("ich", 2)
reduce("und", [1,1,1,1]); // = ("und", 4)
...
```

Code Snippet 2.5: MR Example - *Word Count (reduce() Calls)*

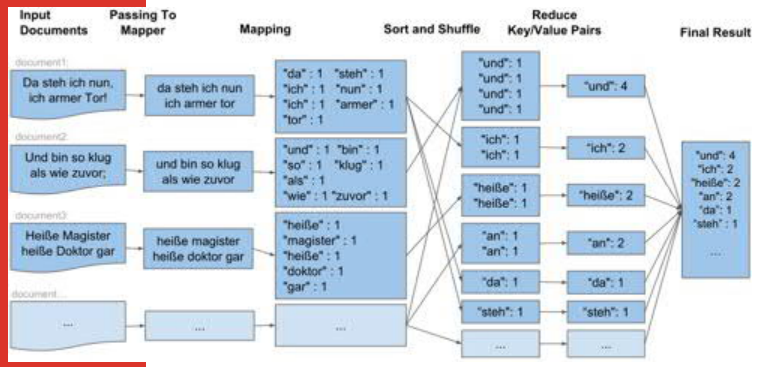
```
result = [ ("und", 4), ("ich",2), ("heiße",2), ("an", 2), ("da",1),
    ("steh",1), ("nun",1), ("armer",1), ("tor",1), ("bin",1), ("so",1),
    ("klug",1). ("als",1), ("wie",1), ("zuvor",1), ("magister",1),
    ("doktor",1), ("gar",1), ("ziehe",1), ("schon",1), ("die",1),
    ("zehen",1), ("jahr",1), ("herauf",1), ("herab",1), ("quer",1),
    ("krumm",1), ("meine",1), ("schüler",1), ("der",1), ("nase",1),
    ("herum",1) ]
```

Code Snippet 2.6: MR Example - *Word Count (Final Result)*

MapReduce – Phases



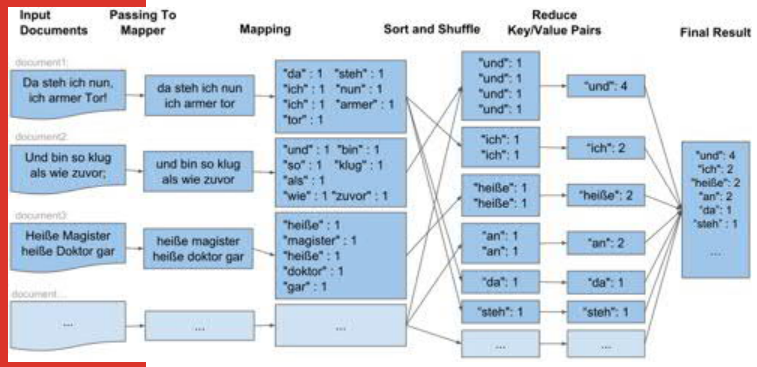
MapReduce – Phases



1. Фаза Map: каждая функция Map получает несколько пар ключ/значение и обрабатывает каждую из них отдельно. Все процессы Map выполняются независимо друг от друга, что позволяет одновременно выполнять всю обработку на нескольких узлах. В нашем примере это приводит к нескольким спискам слов со связанным числом, равным одному и представленным как ключ/значение этой пары.

2. Фаза сортировки и перетасовки: этап между Map и reduce с целью переноса данных с Map для эффективного сокращения процессов, что обычно выполняется автоматически с помощью платформы MapReduce. Весь вывод, создаваемый процессами Map, группируется и сортируется по ключу, разделяется и распределяется по процессам сокращения. Распространенный способ разделения — хешировать ключи и использовать хеш-значение по модулю количества процессов reduce для достижения равномерного распределения нагрузки между всеми узлами кластера (общее количество разделов равно количеству процессов reduce). Обычно не нужно использовать стандартные **функции Shuffle, Sort и Partitioning** по умолчанию.

MapReduce – Phases



3. Фаза Reduce. Функция Reduce вызывается один раз для каждого уникального ключа. Все процессы сокращения выполняются независимо друг от друга, что позволяет одновременно выполнять всю обработку на нескольких узлах. Таким образом, функции Reduce перебирают все значения, связанные с ключом, и выдают ноль или более результатов. В примере WordCount приращение происходит для каждого значения (счетчика), связанного с одним и тем же ключом (словом).

4. Output Phase: выходные данные всех процессов Reduce собираются и консолидируются инфраструктурой MapRedcue и обычно записываются в распределенную файловую систему. Помимо перечисленных этапов, большинство фреймворков MapReduce также используют этап, называемый объединение или консолидация Sort&Shuffle, который происходит между этапом Map и этапом Sort&Shuffle. Консолидатор также известен как «mini-reducer», он принимает промежуточные выходные данные процессов отображения — перемешивает, сортирует и частично сокращает их путем объединения пар ключ/значение с одним и тем же ключом.

МОСКОВСКИЙ
ГОРОДСКОЙ
УНИВЕРСИТЕТ
МГПУ

129226, г. Москва, 2-й Сельскохозяйственный проезд, 4
info@mgpu.ru
+7 (499) 181-24-62
www.mgpu.ru

СПАСИБО ЗА ВНИМАНИЕ