МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

Требования к распределенных систем, моделям данных и доступа

Требования к системе данных

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

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

Запрос данных: мы должны иметь возможность эффективно запрашивать и фильтровать данные определенными способами (транзакциями и индексами).

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

Обработка данных: мы 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);
- **сообщения обработаны** (в случае **Apache Kafka** и **Linkedin** более 2 миллионов записей в секунду всего на 3 узлах);
- **данные обработаны** (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

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.

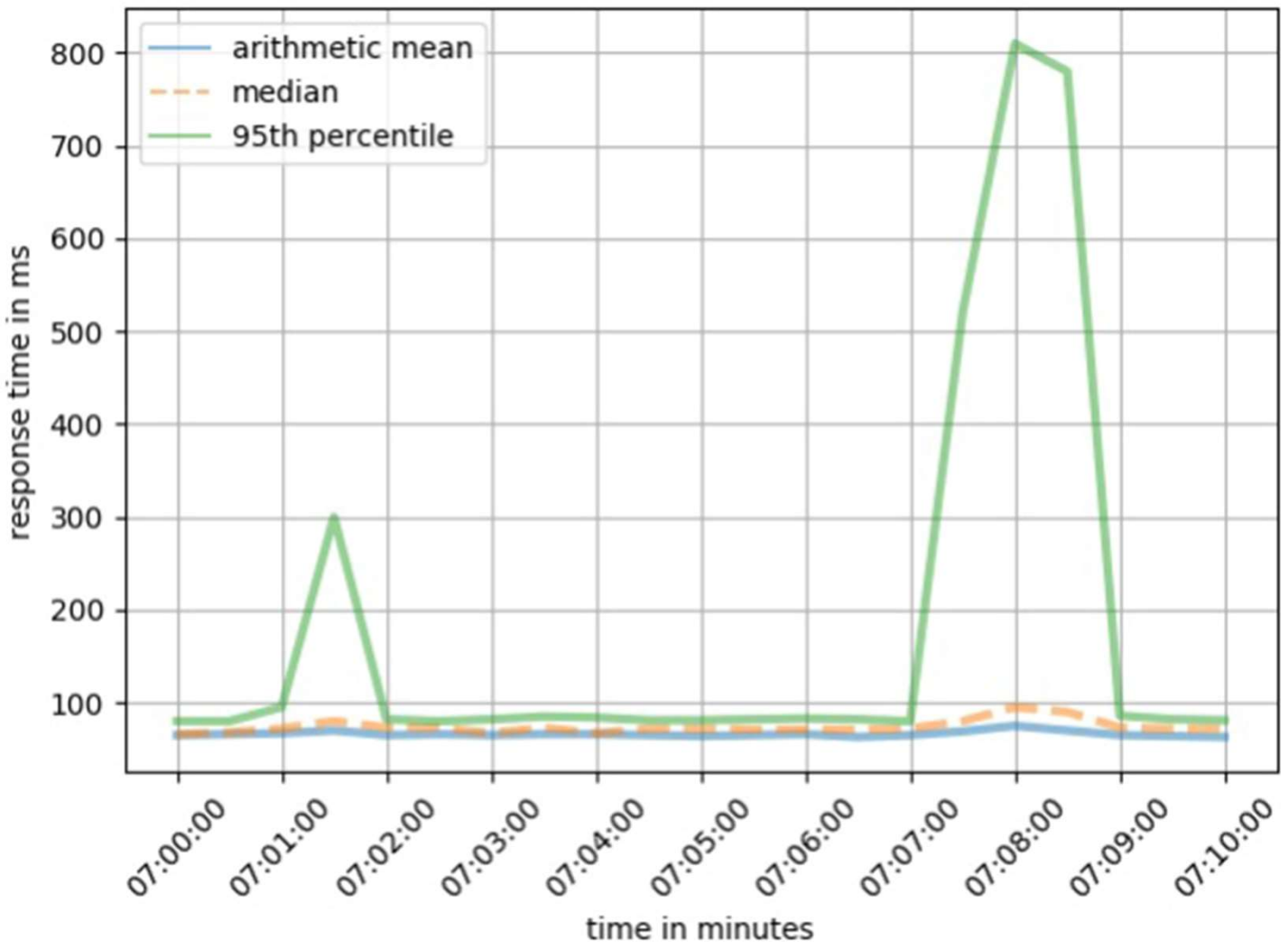
Медиана:

- легко рассчитать
- меньше искажается выбросами

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

- легко рассчитать
- не искажается выбросами

Масштабируемость — показатели производительности



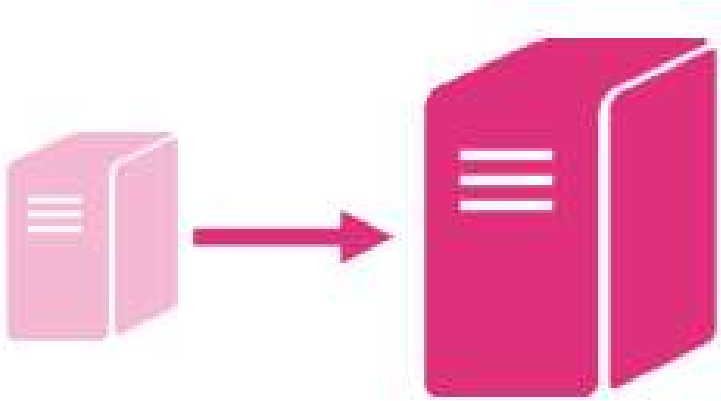
МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

Масштабируемость — подходы к масштабированию

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Вертикальное масштабирование: заменить сервер один более мощный

Горизонтальное масштабирование: распределять нагрузку на несколько серверов вместо одного



На практике – всегда гибридное масштабирование!

Надежность

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

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 около2,11%, **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

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Обслуживание = одна из самых больших затрат
при разработке программного обеспечения

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. Развиваемость: упростить адаптацию/изменение системы данных

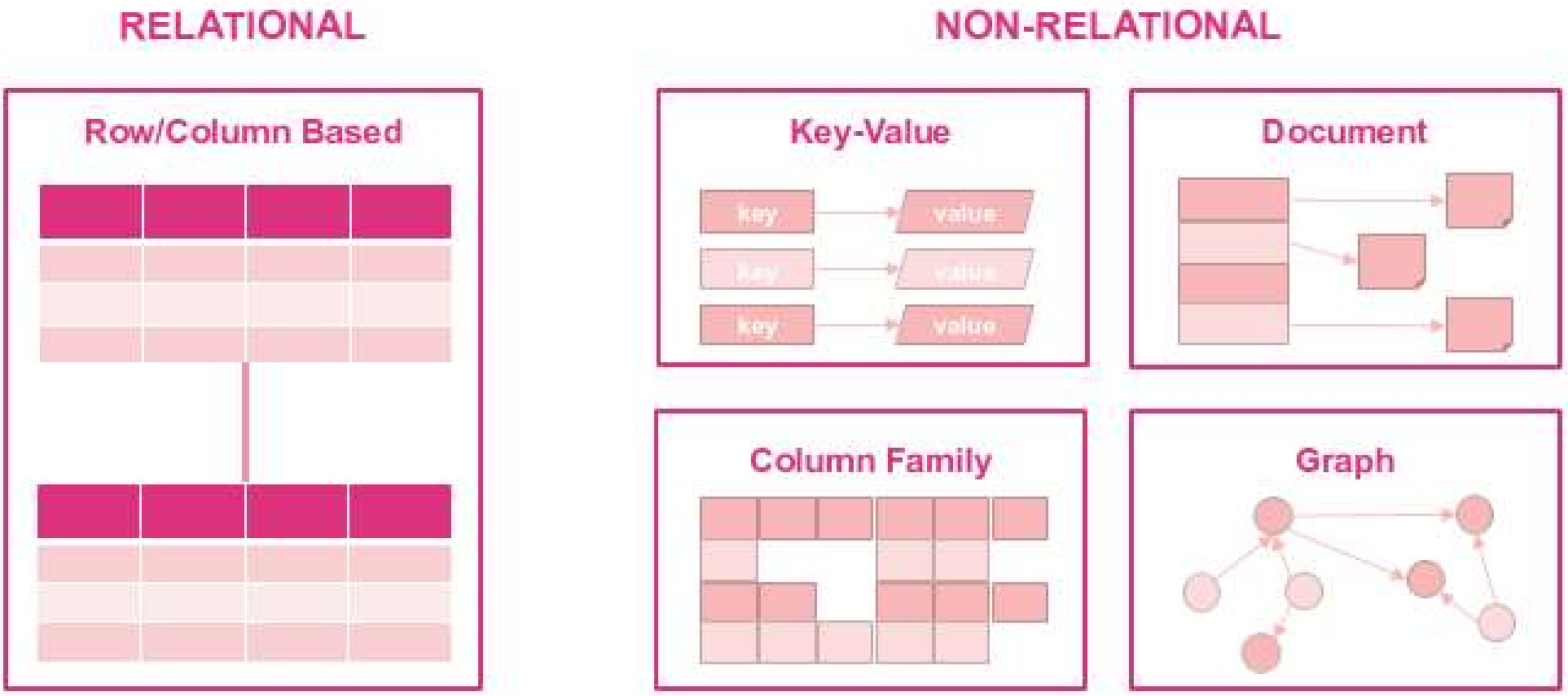
- продолжение интеграции
- разработка через тестирование
- парное программирование

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

Введение в модели данных и доступ

Модели данных (реляционные/нереляционные)

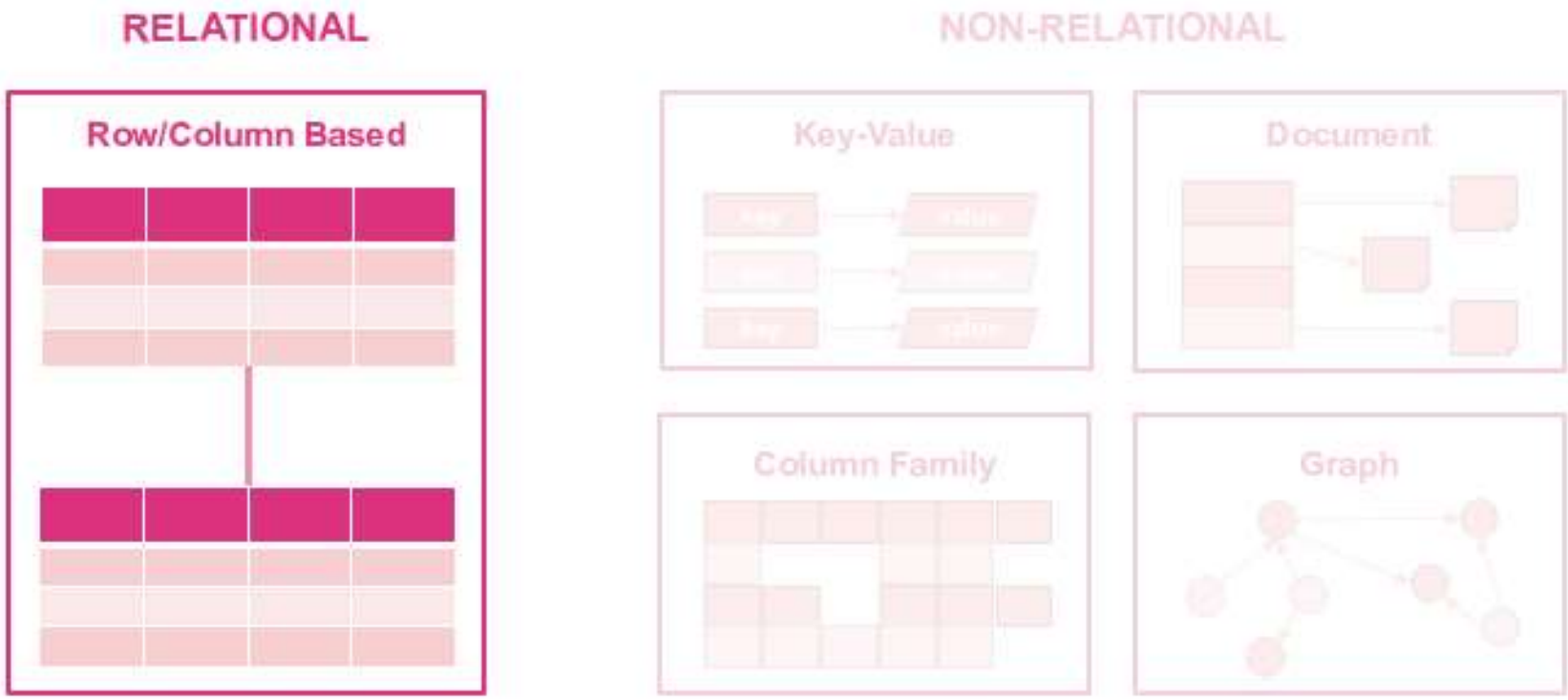
МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ



Реляционная модель данных

Originally introduced by Edgar Frank Codd in 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

Originally introduced by Edgar Frank Codd in 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 — 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

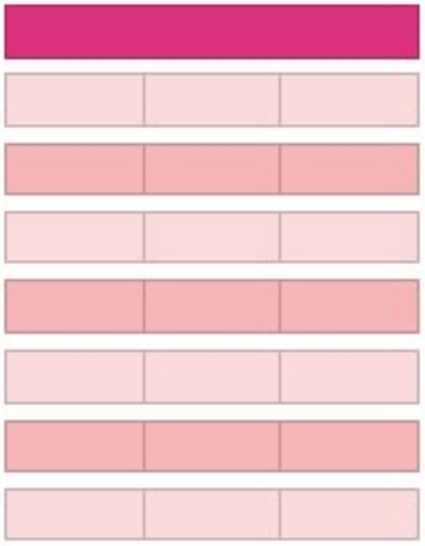
https://en.wikipedia.org/wiki/List_of_relational_database_management_systems

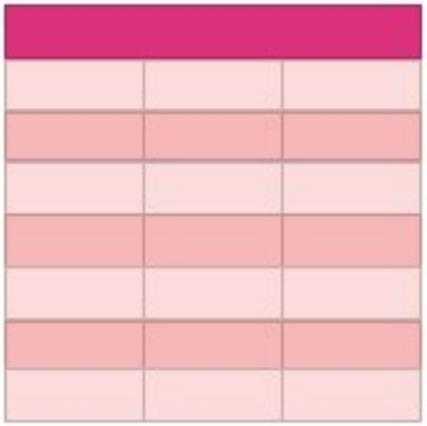
Реляционная модель данных — на основе строк(Row) и столбцов(Column-Based)

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Row-Based:

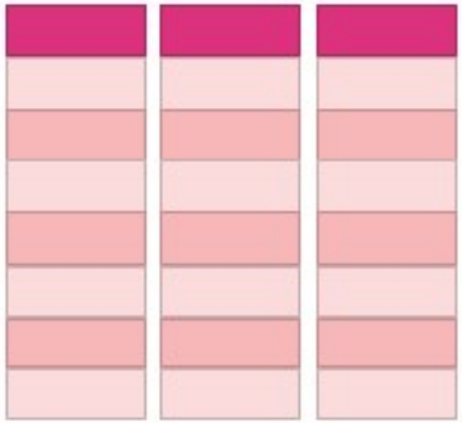
→ строки хранятся непрерывно
→ Oracle, IBM DB2, Microsoft SQL Server, MySQL





Column-Based:

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



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

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

Operation	Row-Based	Column-Based
Compression	Low	High
Column Scans	Slow (Multiple Reads)	Fast (One Read)
Insert Of Records	Fast (One Insert)	Slow (Multiple Inserts)
Single Record Queries	Fast (One Read)	Slow (Multiple Reads)
Single Column Aggregation	Slow (Full Table Scan)	Fast (Only Column Scan)
Typical Use Cases	Transactional	Analytical

Реляционная модель данных — SQL

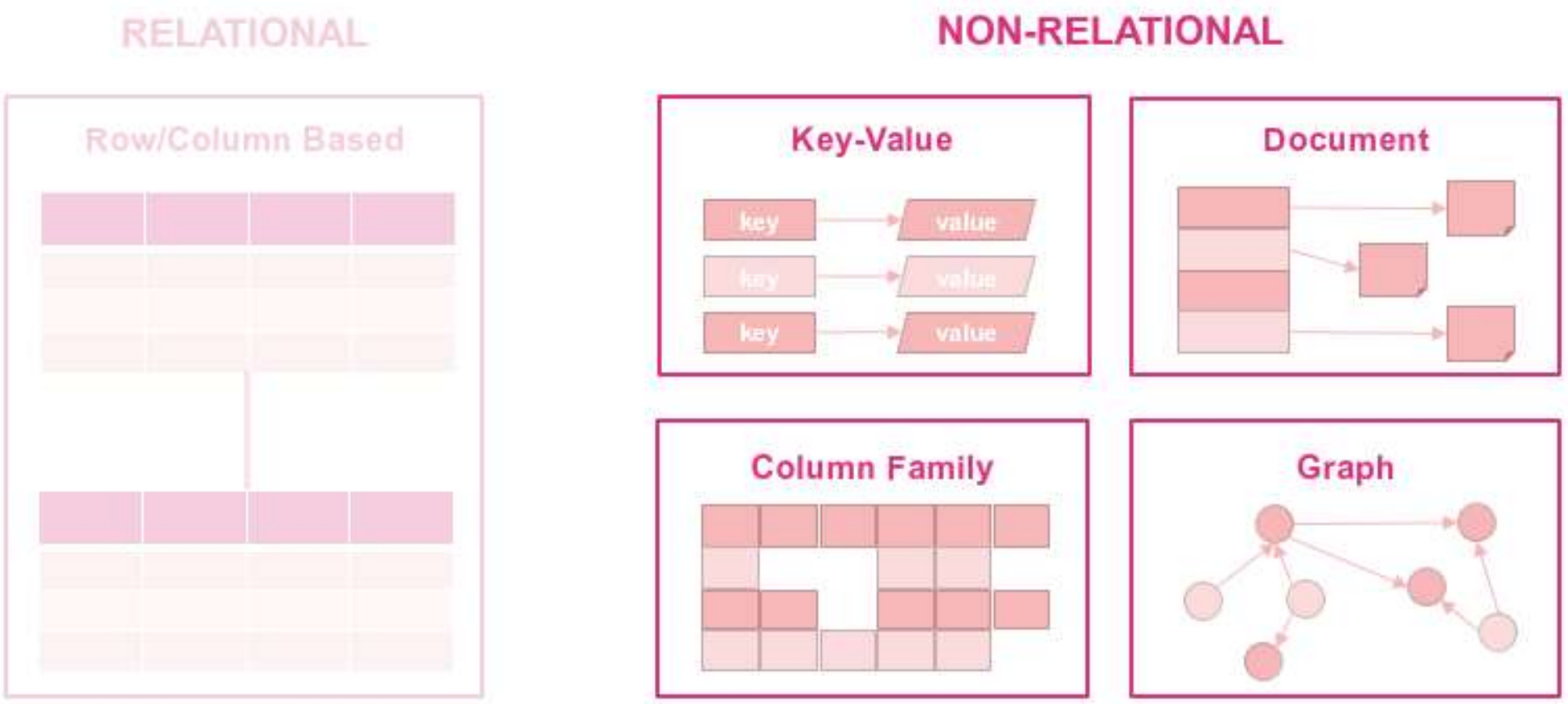
МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

- **Strenghts:**
  - Consistency → ACID
  - Universal → a lot of data types, linked and unlinked data, “Independance“ of RDBS
  - Strict Schema → Data Quality (*Garbage-In – Garbage-Out*), Error Prevention, Compression

- **Weaknesses:**
  - Strict Schema:
    → needs to be altered at any data format change
    → data needs to be migrated
  - Object-Relational-Impedance Mismatch:
    - E.g. objects, structs, ...
    - needs ORMs
    - usually slows and complicates data access

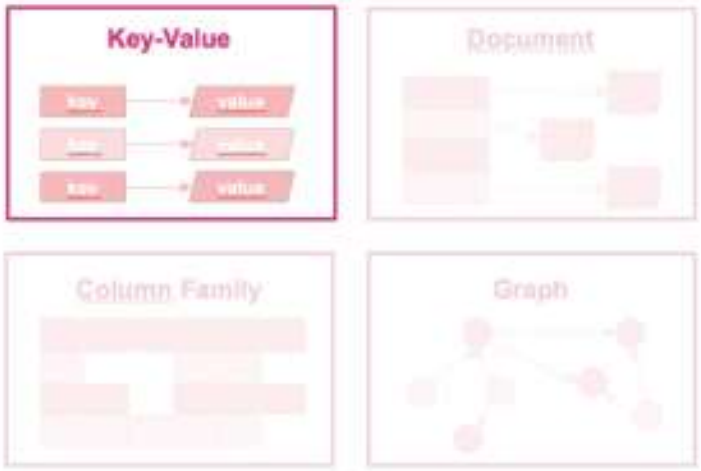
Нереляционная модель данных

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ



Нереляционная модель данных — ключ-значение

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ



- **Examples:**
  - Redis,
  - BerkeleyDB,
  - VoldemortDB,
  - ArangoDB,
  - Riak, …

- **Strenghts:**
  - **fast queries** (value lookups)
  - **fast inserts** (key-value pairs)
  - **easy to replicate** and **distribute** (*consistent hashing*)

- **Weaknesses:**
  - **no** or **less efficient** and slow:
    - aggregation
    - filtering
    - joining

    → needs to be done by application

Нереляционная модель данных — MapReduce

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

MapReduce = парадигма программирования и соответствующая реализация для параллельной обработки и генерации больших наборов данных, распределенных в кластере.

- первоначально представлено Джеффри Дином и Санджаем Гемаватом (Google Inc.) в2004 г.
- MapReduce не является ни декларативным языком, ни императивным языком программирования, это нечто среднее
- Парадигма основана на указании:
- map function, выполняющая фильтрацию и сортировку, в результате промежуточный набор пар ключ/значение объединяет
- reduce function все промежуточные значения, связанные с одним и тем же ключом.

Задания MapReduce автоматически распараллеливаются и выполняются на нескольких узлах кластера

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