KOÇ UNIVERSITY

Database Management Systems

NoSQL Databases

M. Emre Gürsoy

Assistant Professor
Department of Computer Engineering
www.memregursoy.com

NoSQL

- **NoSQL** = “Not only SQL”
  - An umbrella term for DBMSs that don’t follow the relational DBMS (RDBMS) principles
  - Emerged as a movement in late 2000s, early 2010s
  - Used to be a trending, hot topic

- Popular NoSQL DBMS examples:
  - MongoDB, Redis, Neo4J, BigTable (Google), DynamoDB (Amazon), Cassandra (Facebook)
- Check out popularity ranking:
  - https://db-engines.com/en/ranking

Main Characteristics

- Storage of semi-structured or unstructured data
  - Relaxed schema requirements (“schema-free”)
- High performance, availability, scalability
  - “Internet-scale” data management, variety of data types
  - ACID and transaction mindset are oftentimes not supported (in some NoSQL systems they are)
- Support data replication and distributed storage
  - Leverage the power of “cluster computing”
  - Give up consistency in favor of speed + availability

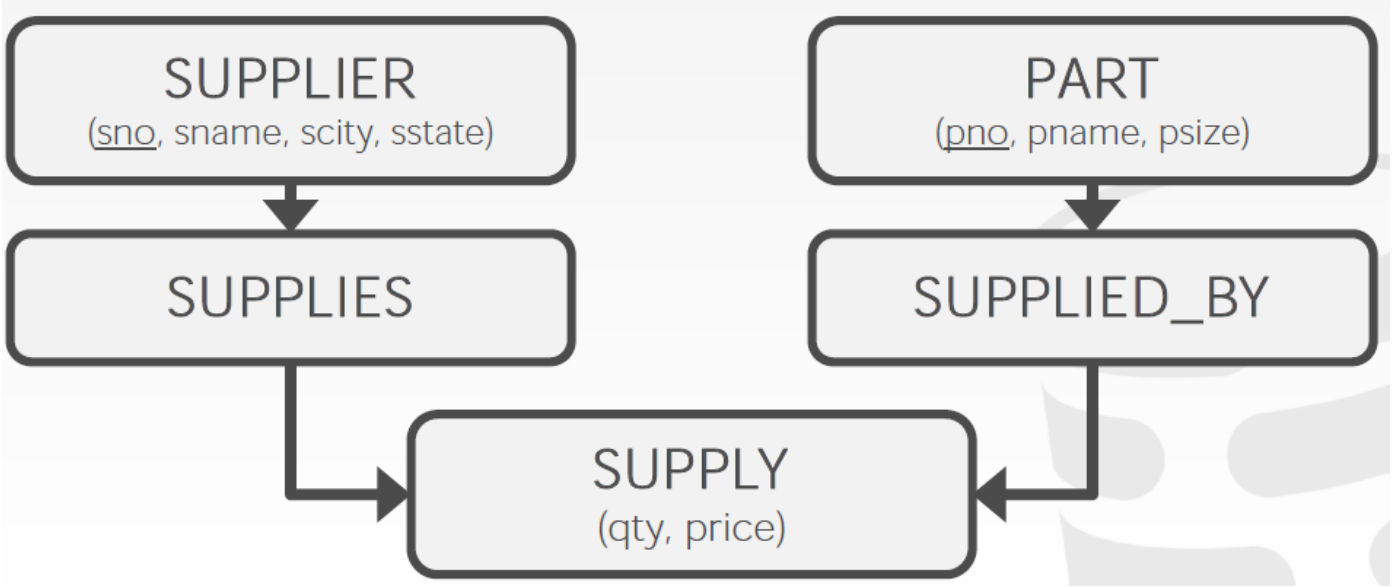
History of DBMS

- In order to better understand the NoSQL movement, it is important to understand the **history** of DBMS.

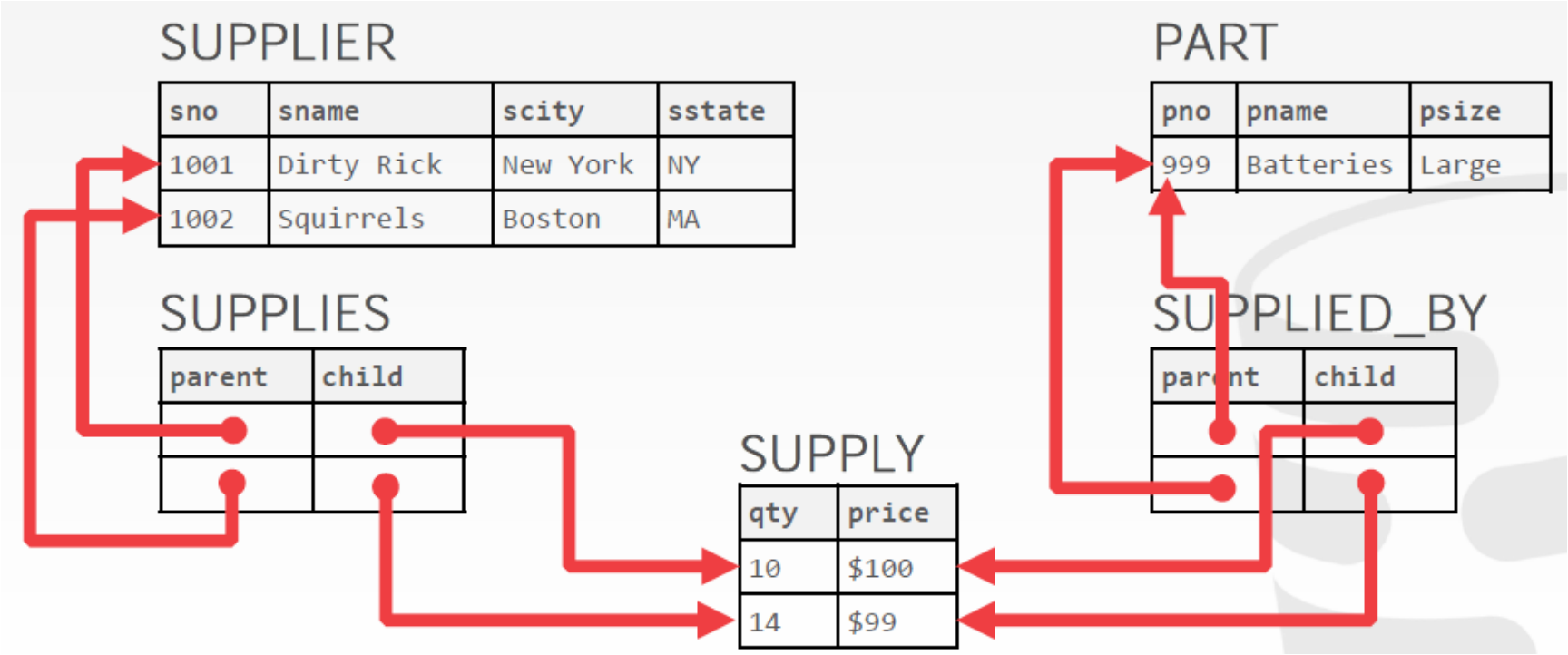
- Many of the ideas in today's DBMS are not new!
  - Some debates today (**SQL vs NoSQL**) are reminiscent of debates from many years ago.
  - But the conditions have changed, the world has evolved, and requirements and expectations from DBMSs are now different.

1960s – Network Data Model

- **IDS:** Integrated Data Store
  - Developed by GE in the 1960s
  - Led by Charles Bachman
    - Bachman won a Turing award for his contributions to databases
  - Supports the network data model



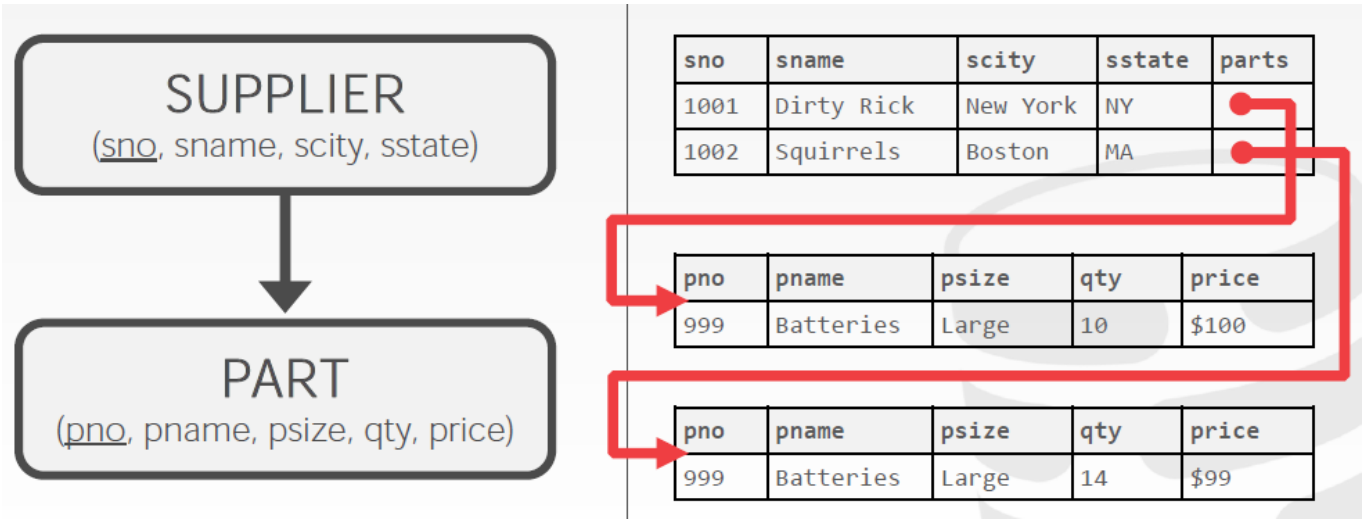
Network Data Model



- Shortcomings of the network data model:
  - Queries are complex (follow a lot of pointers)
  - Data is easily corrupted

1960s – Hierarchical Data Model

- **IMS:** Information Management System
  - Developed by IBM
  - Supports the hierarchical data model



- Shortcomings of the hierarchical data model:
  - Data duplication (e.g., repeated pno – pname pairs)
  - Lack of independence for children (e.g., parts)

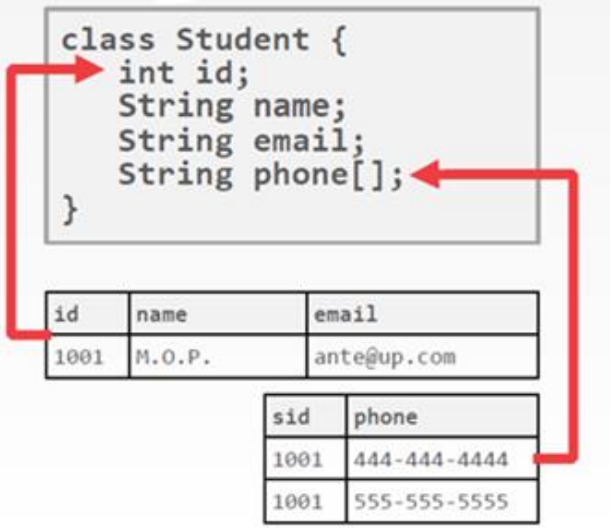
1970s – Relational Model

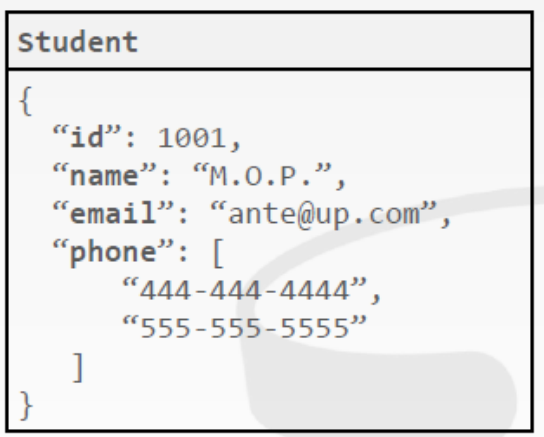
- Developed by E. F. Codd at IBM Research
  - Also a Turing award recipient (for contrib. to databases)
  - Motivations:
    - Address shortcomings of the network and hierarchical data models
    - Improve convenience for programmers – IMS and IDS programmers were having to rewrite programs every time the database layout or schema changed

- Relational model: Initially an abstract, mathematical model
  - Store the database in simple data structures (eg: tables)
  - Access data through a high-level language
    - Relational algebra, SQL

1980s – Object-Oriented Model

- Rise of object-oriented programming
- Avoid the relational-object impedance mismatch:





- Not many new DBMSs
- But some core ideas exist today in other forms (e.g., JSON, XML, object-oriented SQL, SQL Alchemy)

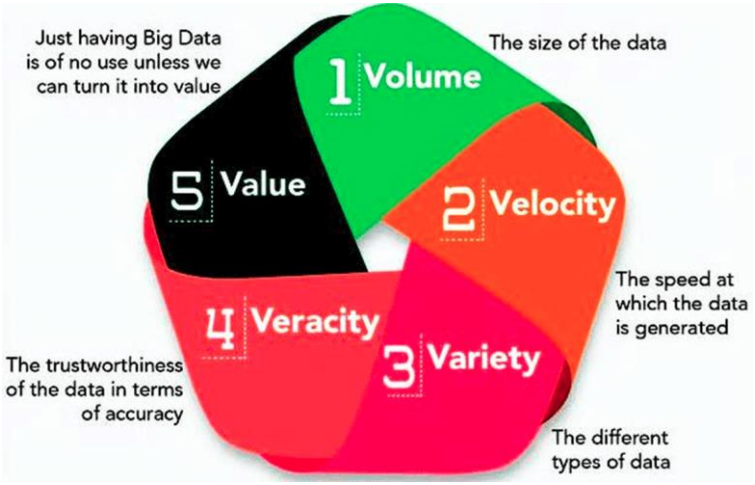
1980s and 1990s

- **Relational model** emerges as the clear winner
  - SQL becomes a standard (recognized by ANSI + ISO)
  - Many relational DBMSs are developed (academia + industry)

IBM DB2
ORACLE
Microsoft SQL Server
Informix
TANDEM
MySQL
SYBASE
TERADATA
PostgreSQL
INGRES
InterBase

2000s – Internet Boom

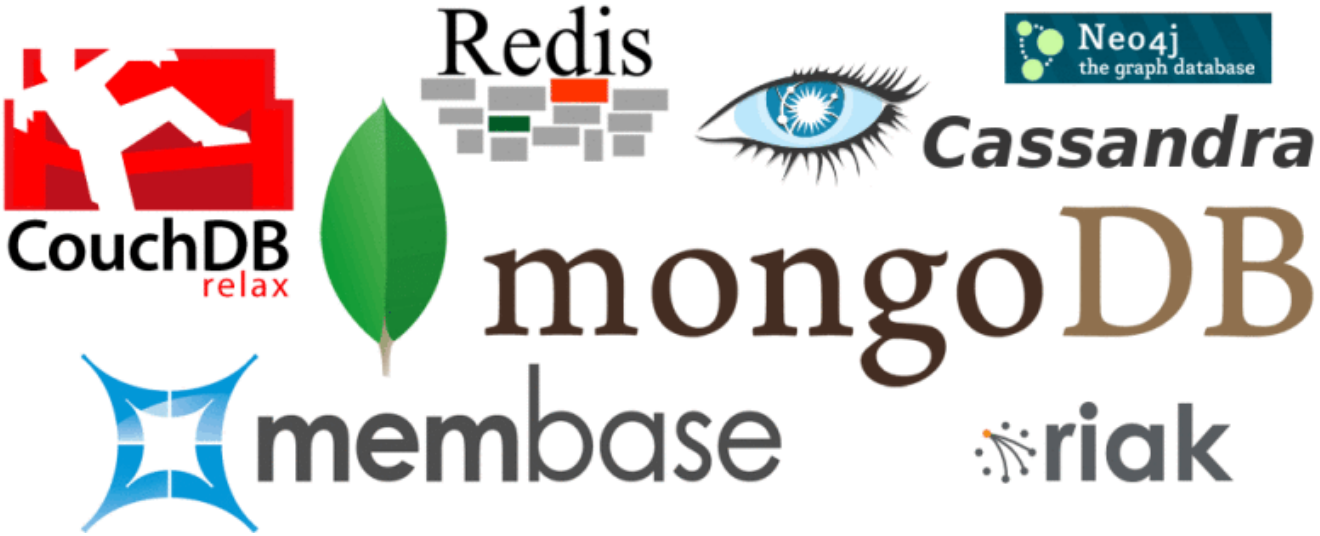
- DBMSs need to handle big data
  - Volume, velocity, variety
  - Unstructured data
  - RDBMSs are heavy-weight, expensive, mostly single-node



- Internet companies start writing their own custom middleware or custom databases
  - BigTable (Google), DynamoDB (Amazon), Cassandra (Facebook)
  - Academic publications describing these systems
  - Open source culture

Late 2000s – NoSQL

- High availability and scalability
  - “Schemaless” (“schema-last”)
  - Non-relational data models (documents, graphs, …)
  - No ACID transactions
  - Custom APIs and query languages for their data models
    - Graph data, document-oriented data, etc.



Scaling DBMSs

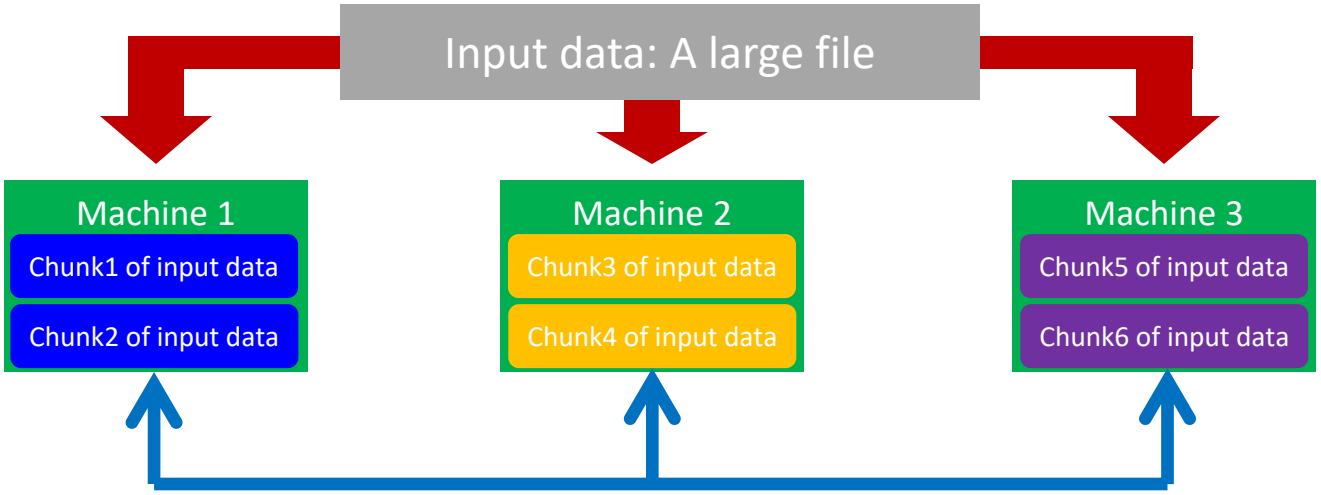
- DBMSs can be scaled vertically or horizontally.

- **Vertical Scaling**
  - More hardware (faster CPU, larger disk)
  - How much can you afford? How much can you fit?

- **Horizontal Scaling**
  - Many weak machines (e.g., commodity hardware)
  - Need proper orchestration of the machines: network communication + distributed system overheads
  - Data sharding and replication

Data Sharding

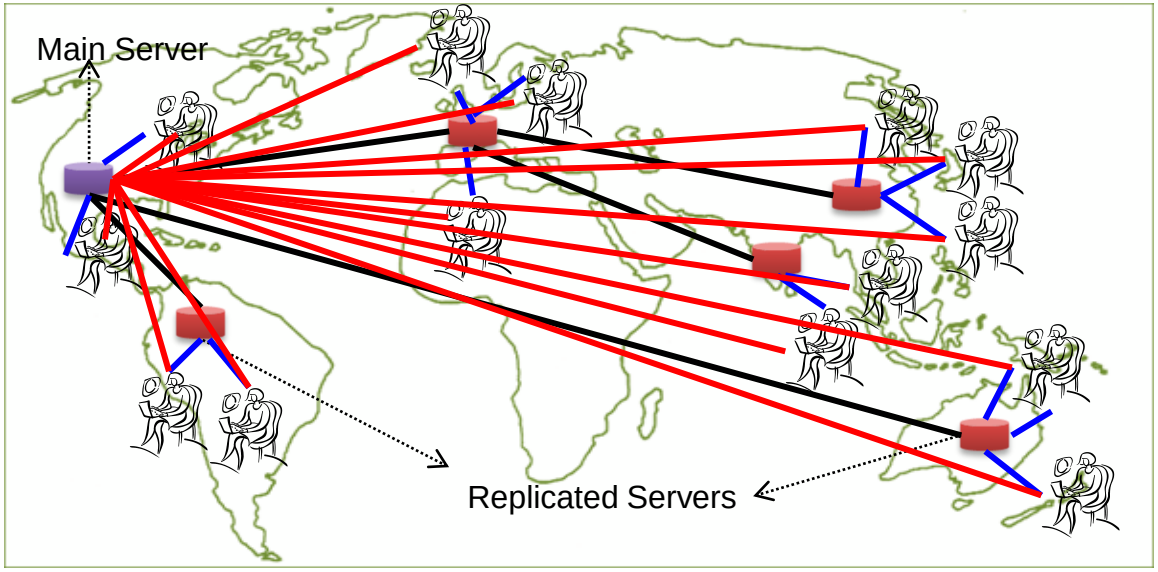
- A shard is a horizontal partition of the data.
- Shards can be distributed to multiple machines to allow for concurrent/parallel access.
- One way to do sharding: **hashing**



E.g., Chunks 1, 3 and 5 can be accessed in parallel

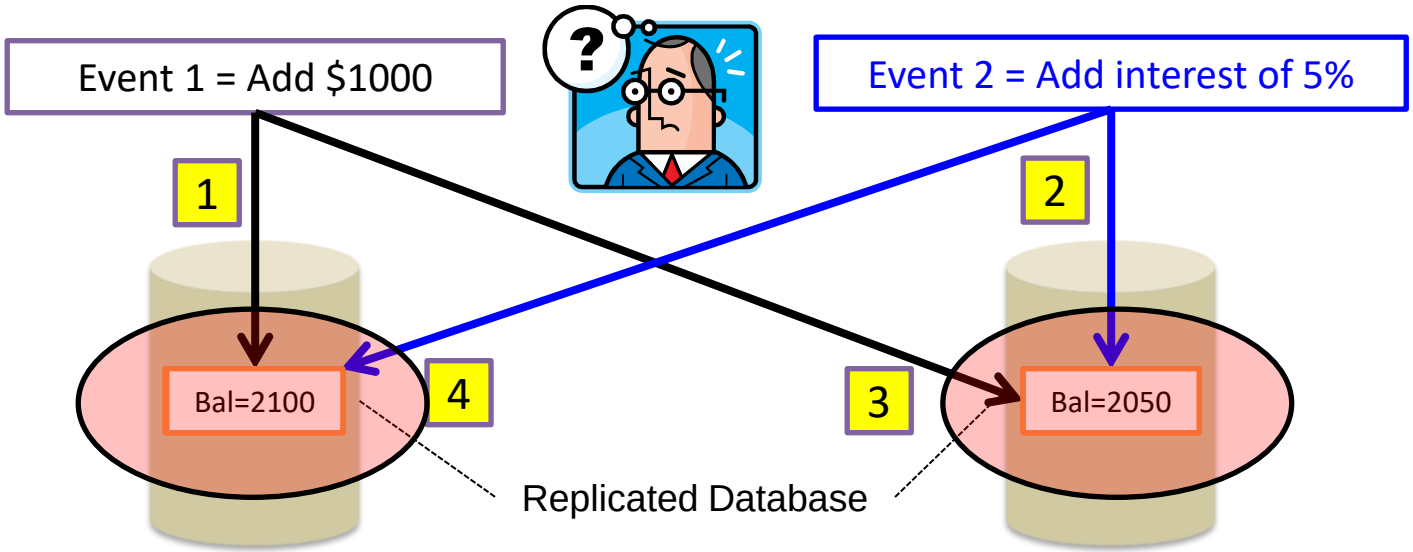
Data Replication

- Replication: Store the same data in multiple places.
- **Why?** Increase speed and availability
  - Avoid performance bottlenecks of machines
  - Avoid single point of failure
  - Geographical proximity to users



Consistency Challenge

- Maintaining consistency of replicated data is a challenge
- Example: E-commerce application
  - Bank database is replicated across two servers
  - Two transactions arrive

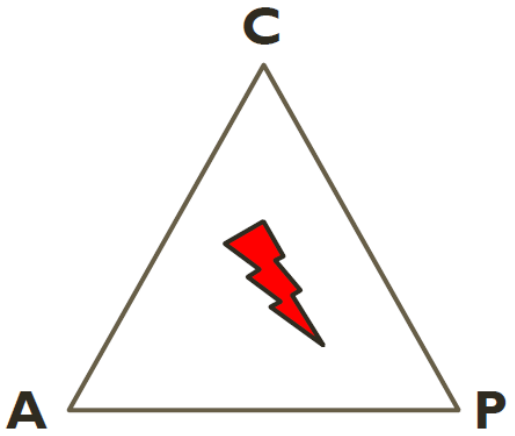


CAP Theorem

- **C**onsistency
  - All clients always have the same view of the data
- **A**vailability
  - The system is always highly available
  - Clients can read and write whenever they want to
- **P**artition tolerance
  - System works well in the presence of partitions

CAP Theorem

You can have at most two of C, A, P at the same time.



CAP and NoSQL

- When developing DBMS-powered services to clients during the Internet era:
  - Offering 24/7 availability and high speed is key – keep **A**
  - Scaling is horizontal (w/ partitions) – keep **P**
  - What do we sacrifice? **C**

- **Eventual Consistency:** weaker form of Consistency
  - Also known as: “good-enough consistency”
  - A database is eventually consistent if all replicas gradually become consistent over time
  - Often adopted in NoSQL

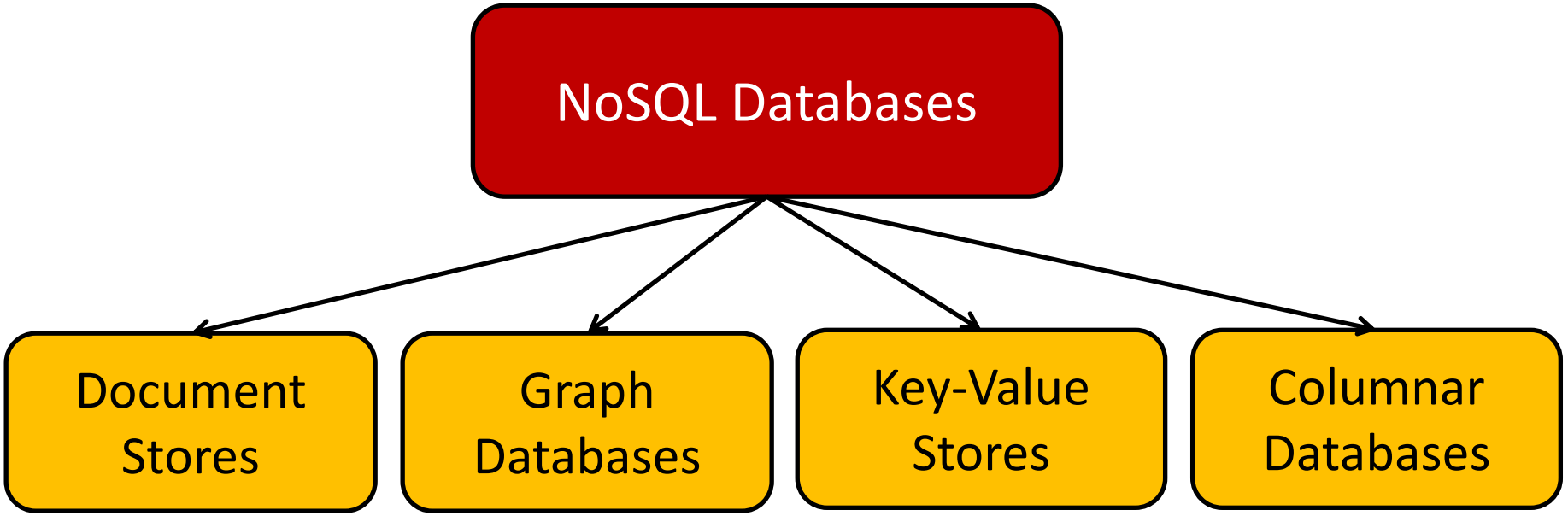
BASE Properties

- NoSQL typically supports **BASE**
  - The acronym became tightly connected to NoSQL
  - (Caution: Not every NoSQL DBMS supports BASE.)

- **B**asically **A**vailable
  - High speed, highly distributed, highly available
- **S**oft-State
  - State of the system may change over time
  - You shouldn’t rely on correctness of current state
- **E**ventually Consistent

Types of NoSQL

- Here is a (limited) taxonomy of different popular NoSQL database types:



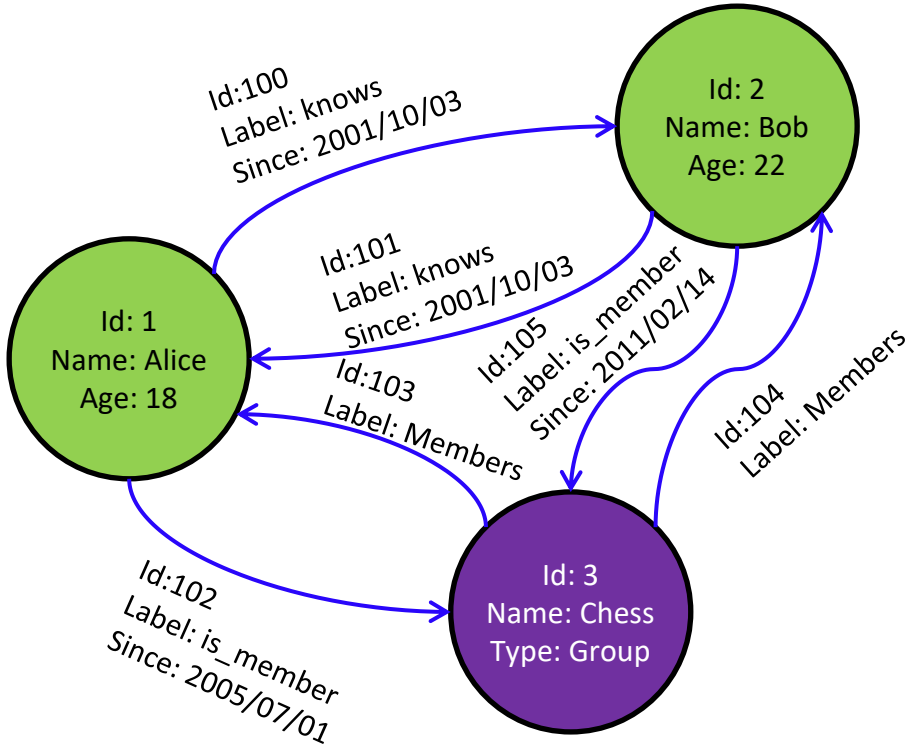
Document Stores

- Collections of similar documents
  - Documents come in some standard format such as XML, JSON, BSON
  - Documents can have different schema (relaxed)
- Similar to the object-oriented or hierarchical data models

```
{
    _id: ObjectId("51156a1e056d6f966f268f81"),
    type: "Article",
    author: "Derick Rethans",
    title: "Introduction to Document Databases with MongoDB",
    date: ISODate("2013-04-24T16:26:31.911Z"),
    body: "This arti…"
},
{
    _id: ObjectId("51156a1e056d6f966f268f82"),
    type: "Book",
    author: "Derick Rethans",
    title: "php|architect's Guide to Date and Time Programming with PHP",
    isbn: "978-0-9738621-5-7"
}
```

Graph Databases

- Good for storing data represented as vertices and edges
  - Have their own QL for graph-related queries

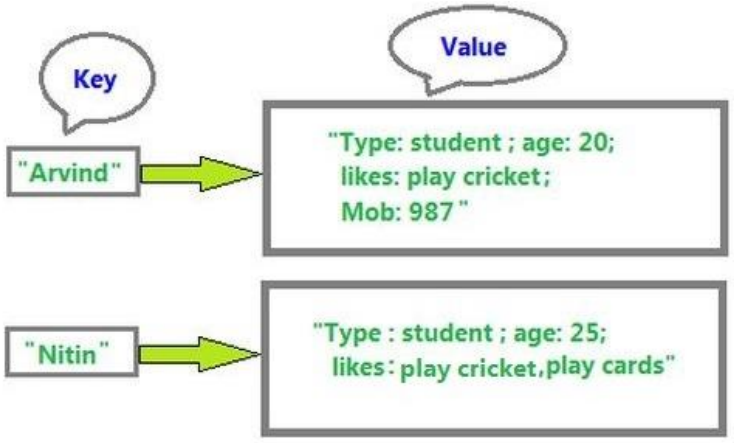


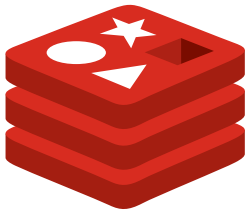
- Similar to which data model?

neo4j

Key-Value Stores

- **Key**: unique identifier associated with a data item
  - Stored in a hash table
  - Distributed to nodes easily
- **Value**: the data item itself (possibly complex)
- Typically only support **CRUD** operations
  - **C**reate, **R**ead, **U**pdate, **D**elete
  - No joins, no aggregate functions, no advanced QL





redis

Columnar Databases

- "Column-oriented databases"
  - Store data in column-order rather than row-order
  - Column analytics become faster, e.g., AVG(Salary)

RowId	EmpId	Lastname	Firstname	Salary
001	10	Smith	Joe	60000
002	12	Jones	Mary	80000
003	11	Johnson	Cathy	94000
004	22	Jones	Bob	55000

```
001:10,Smith,Joe,60000;
002:12,Jones,Mary,80000;
003:11,Johnson,Cathy,94000;
004:22,Jones,Bob,55000;
```

Row-order storage

```
10:001,12:002,11:003,22:004;
Smith:001,Jones:002,Johnson:003,Jones:004;
Joe:001,Mary:002,Cathy:003,Bob:004;
60000:001,80000:002,94000:003,55000:004;
```

Column-order storage

Example from Wikipedia

Final Remarks

- DBMSs have been around for many years
  - Historically, relational DBMSs have thrived
  - They are still more popular today, but NoSQL is also a popular trend
- Motivations behind using NoSQL?
  - Existence of semi-structured and unstructured data
    - Social networks are naturally represented as graphs
    - It's nice to store tweets/text/documents without worrying about schema
  - Desire to make good use of clusters & the "cloud"
  - High speed, scalability, availability
  - Optimized query languages and APIs for the kind of data/workloads they specialize in
    - Writing graph queries in SQL can be painful (and slow)