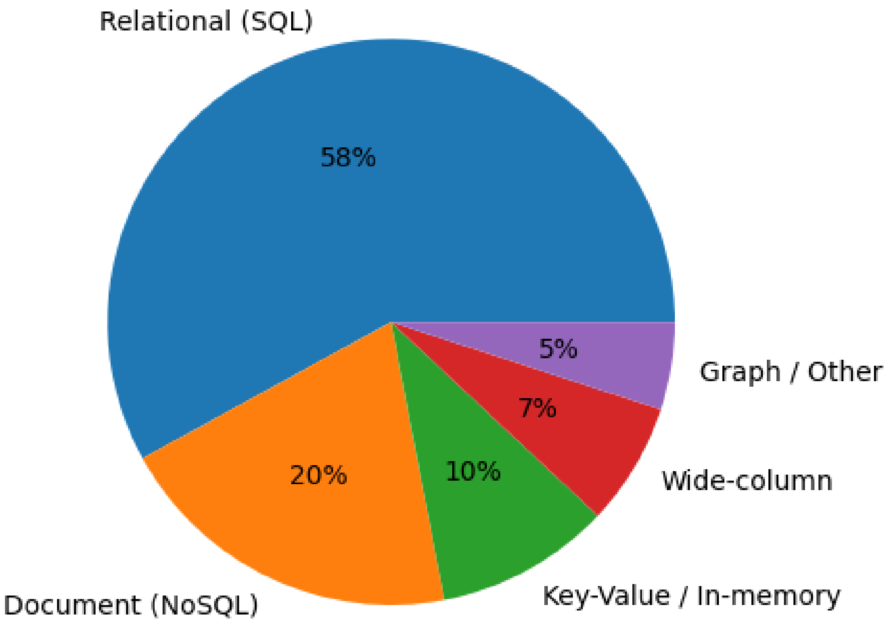
Big data management: CAP theorem and NoSQL

Relational database technology

- Invented in the 1970s
- Well-founded theoretically (relational algebra)
- Provides **high-consistency**
- Many added capabilities
    - BLOB storage
    - Document store
    - Text search
    - In-database code execution
    - Data warehousing (star schema)
    - GIS extensions
- Excellent to cope with centralised storage



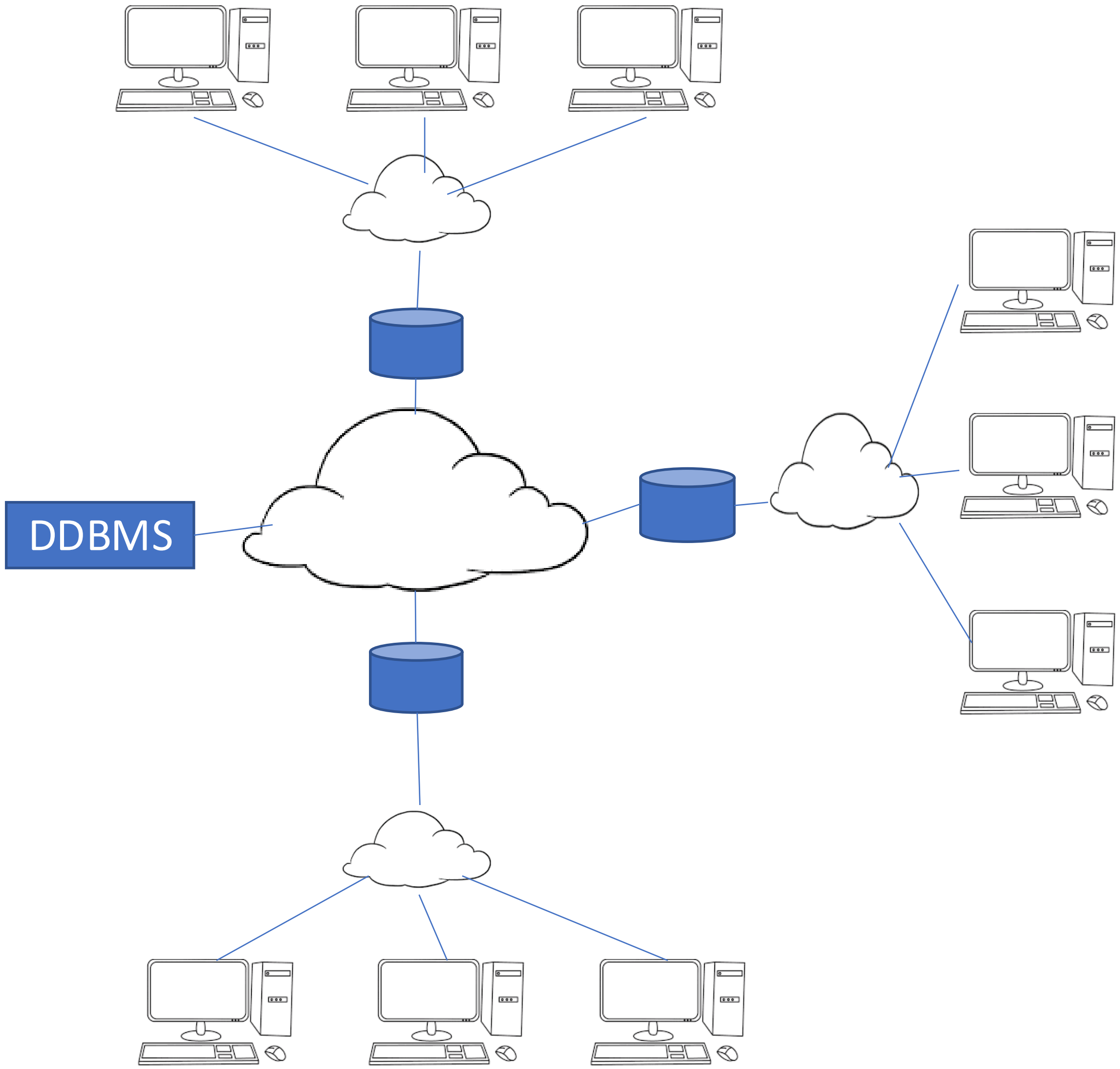
ACID properties in RDBMS

ACID

A	Atomicity	The entire transaction takes place at once or doesn't happen at all.
C	Consistency	The data must be consistent before and after the transaction.
I	Isolated	Multiple transactions occur independently without interference.
D	Durability	The changes of a successful transaction must persist (even if the system fails.)

RDB technology and Internet

- Intensive online access to data
- Requires
  - Availability, scalability, and speed
- Consistency cannot be ensured in all times



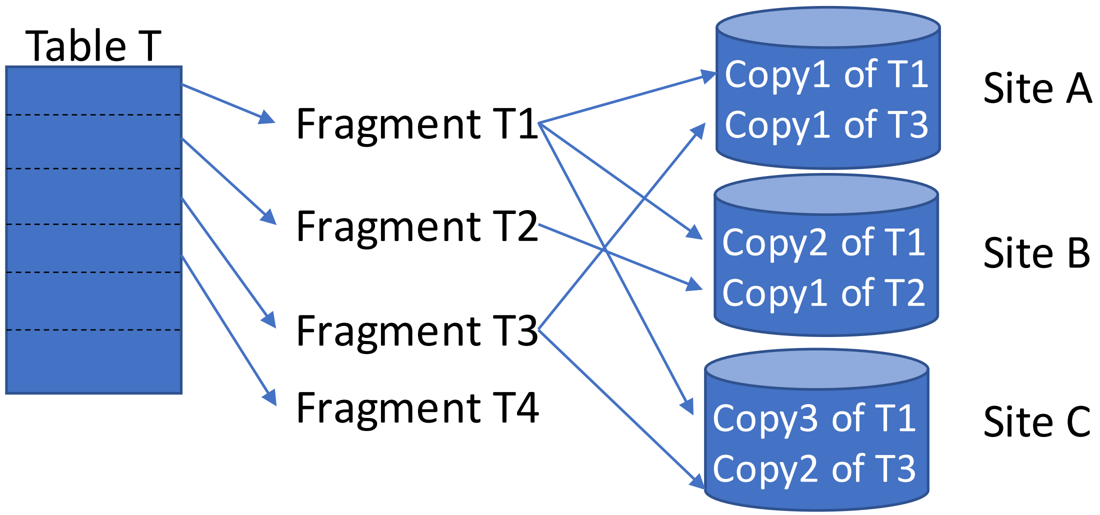
Limitations of relational databases

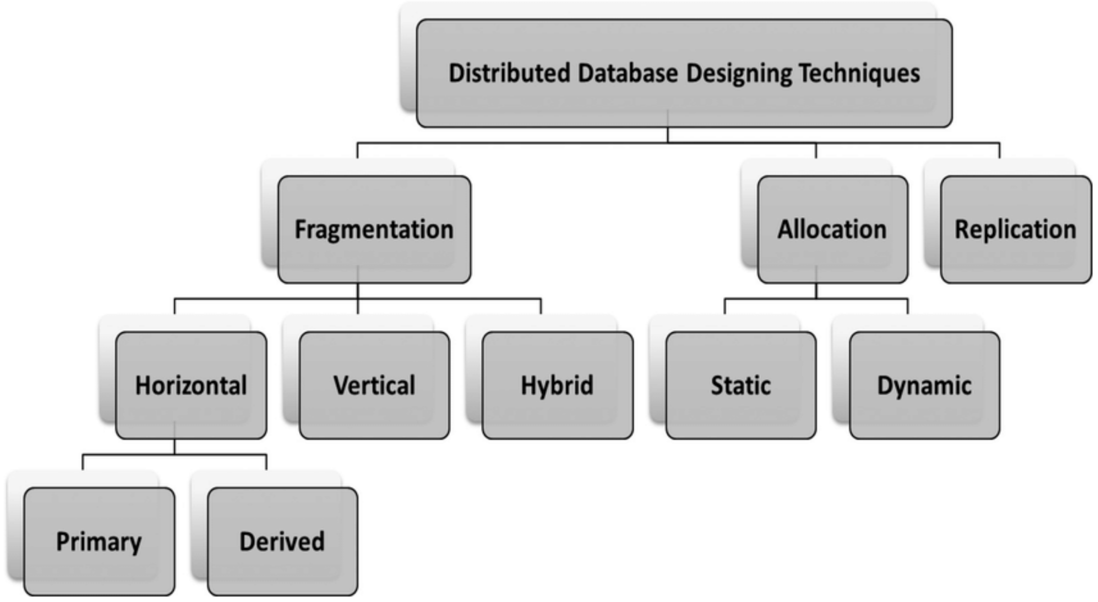
- Low response time with massive amounts of data
  - Fast querying difficult
- Expensive Scale up
  - Adding memory, CPU…
- Limited analytical capabilities
  - Need OLAP like schemas

- Appearance of **distributed databases**
  - Data stored in different nodes
  - Nodes are interconnected by a computer network
  - Nodes are databases
  - DDBMS software manages DDB (access, transparent distribution)

DDBMS: Fragmentation (or partitioning) and Replication

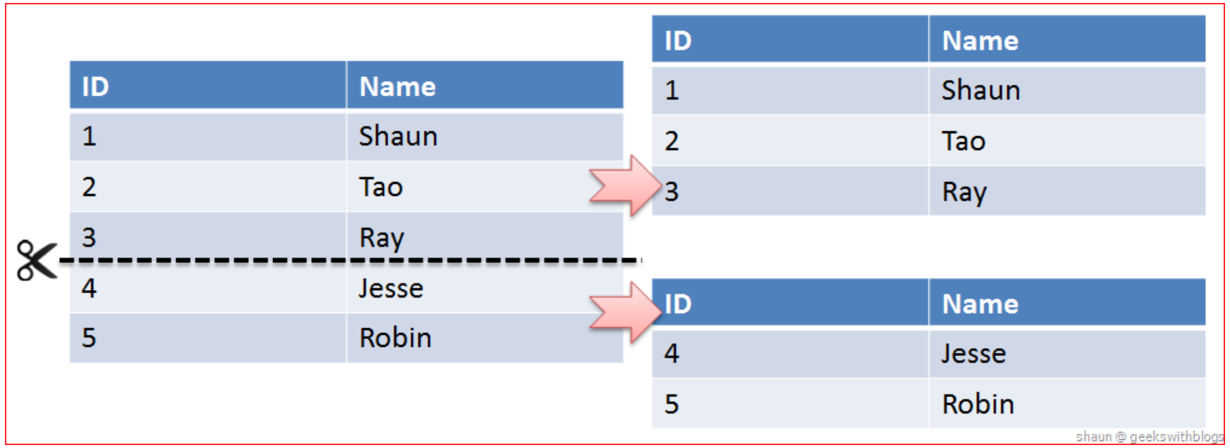
- User unaware of fragments and their replication
- Queries specified on the relations (original schema) and not on fragments





Example partitioning

Horizontal

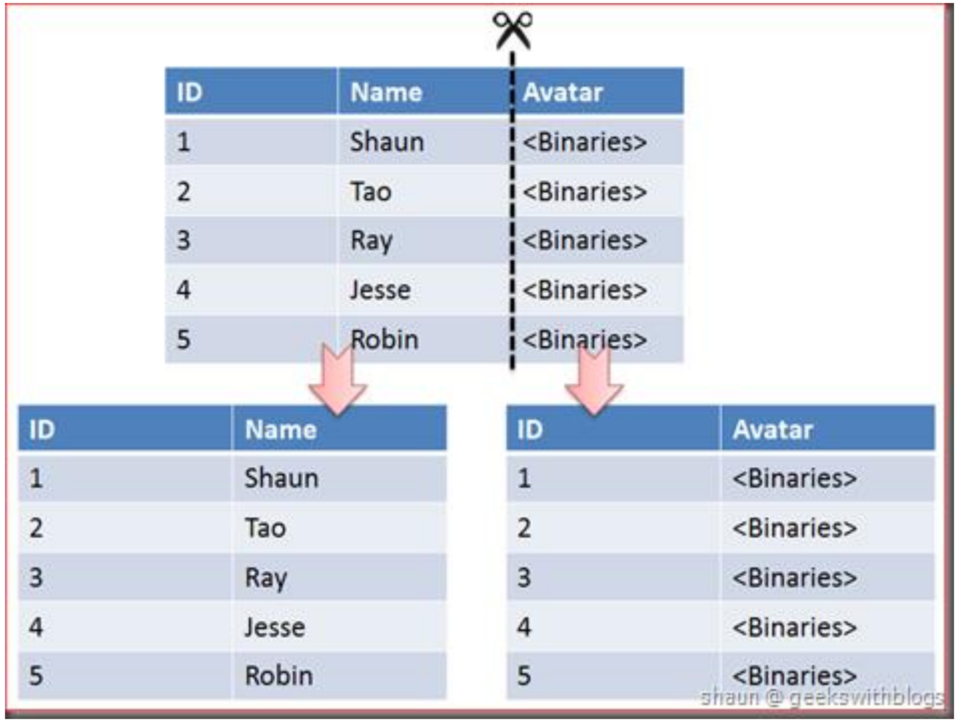


ID	Name
1	Shaun
2	Tao
3	Ray
4	Jesse
5	Robin

ID	Name
1	Shaun
2	Tao
3	Ray

ID	Name
4	Jesse
5	Robin

Vertical



ID	Name	Avatar
1	Shaun	<Binaries>
2	Tao	<Binaries>
3	Ray	<Binaries>
4	Jesse	<Binaries>
5	Robin	<Binaries>

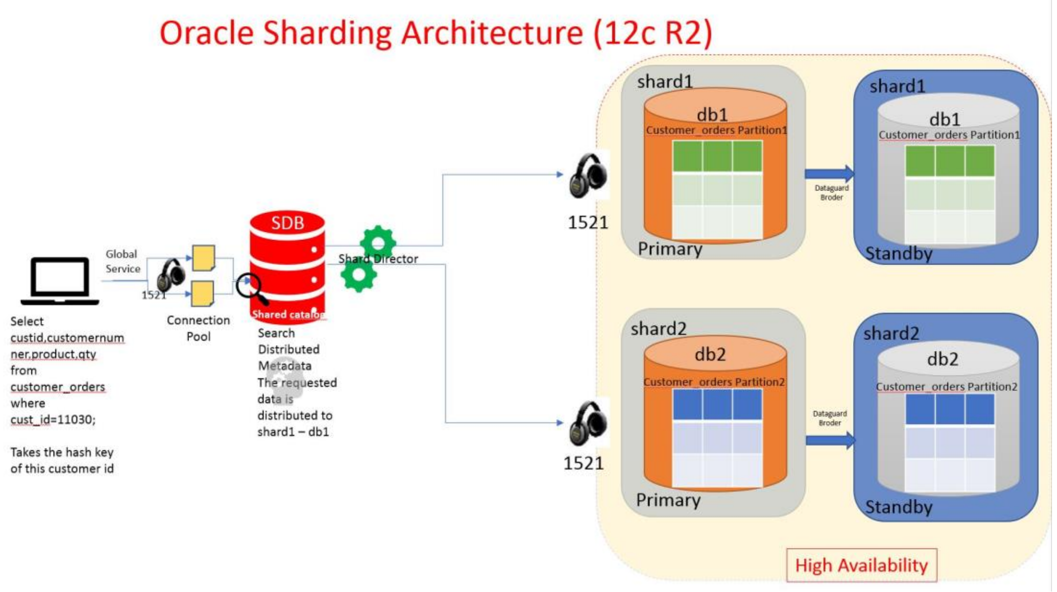
ID	Name
1	Shaun
2	Tao
3	Ray
4	Jesse
5	Robin

ID	Avatar
1	<Binaries>
2	<Binaries>
3	<Binaries>
4	<Binaries>
5	<Binaries>

http://gwb.blob.core.windows.net/shaunxu/Windows-Live-Writer/SQL-Azure-Federation--Why-We-Need-Federa_74F9/

Database sharding

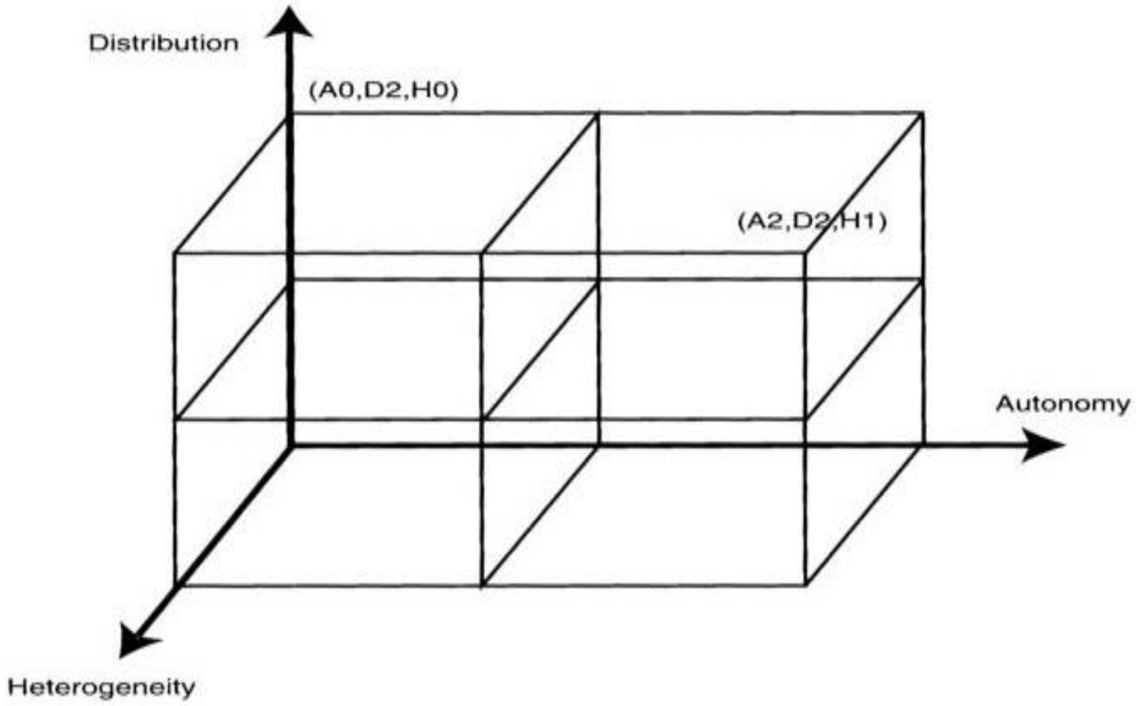
- Another way of naming partitioning
- Table portioned into shards
- Shards are necessarily stored on different computers
- Queries are run on separate shards



In 2011, Facebook split its MySQL database into **4,000 shards** in order to handle the site's massive data volume

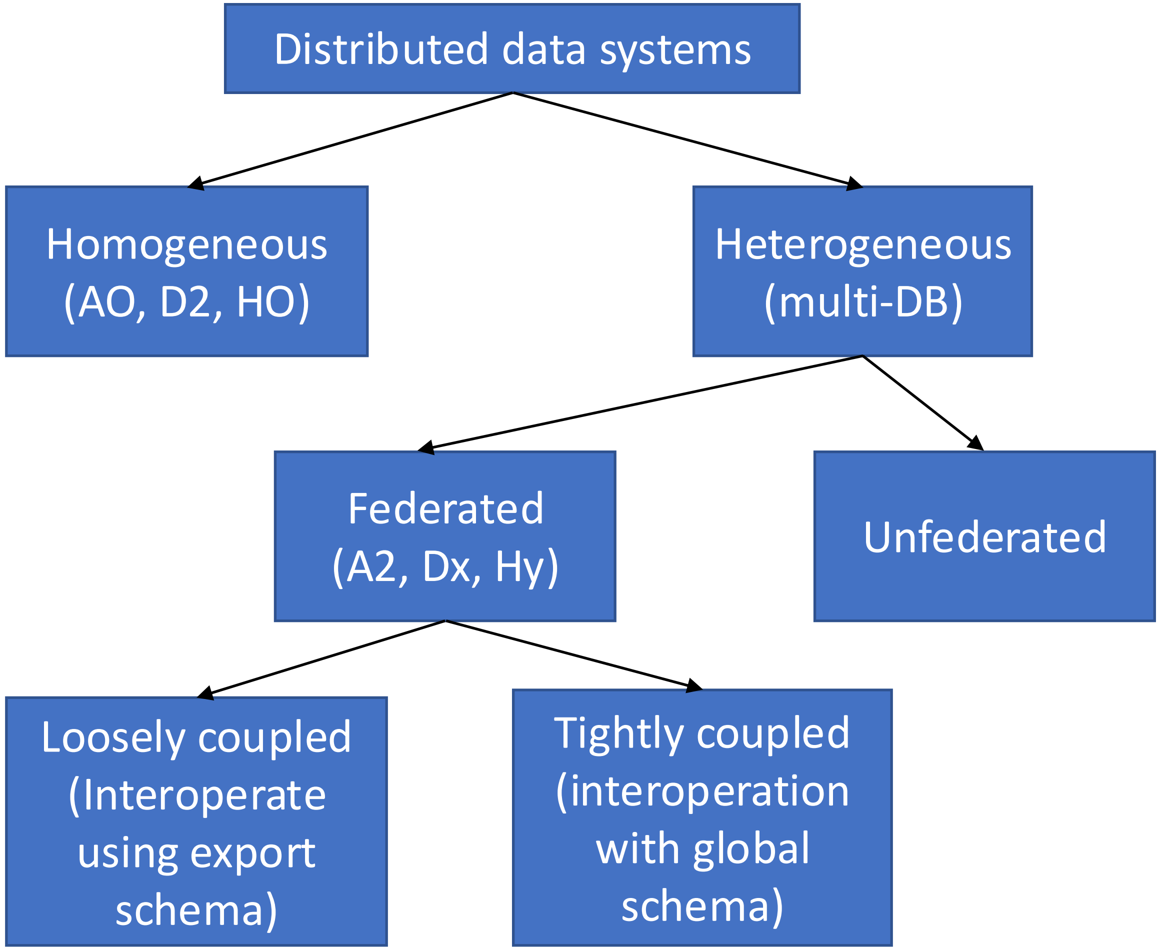
Dimensions of distributed database systems

- Distribution
  - What is the mode of data sharing?
  - D0: No-distribution
  - D1: client-server
  - D2: peer-to-peer
- Heterogeneity
  - Do local systems use same or different technologies?
  - H0: Homogeneous
  - H1: Heterogeneous
- Autonomy
  - How much local nodes operate independently ?
  - A0: tight integration, a single coordinator
  - A1: semi-autonomous, but implement sharing
  - A2: independent, sharing done by another software



Taxonomy of distributed database systems

- Distributed database is a logically integrated collection of shared data physical distributed across network nodes
- In DBMS, global schema is union of local schemas of all databases
- In multi-DB, global schema is a subset of the union of all local databases



Challenges of Distributed Database systems

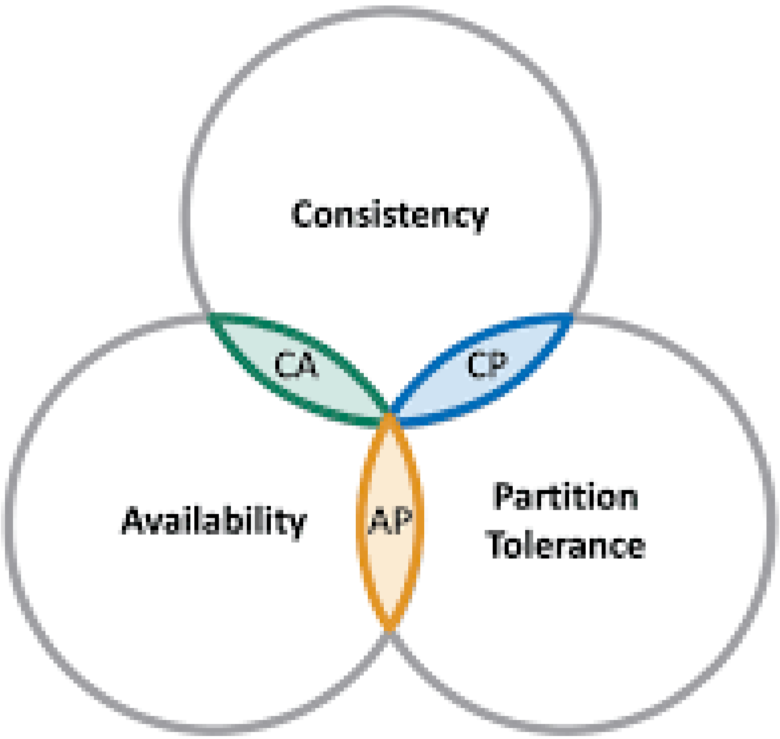
- Large-scale internet applications require scalable storage and support of concurrency
  - Increasable data capacity
  - Growing read/write throughput
  - Handle highly concurrent access
- Horizontal scale required to allow for <u>throughput</u> and <u>capacity</u> increases, however
  - Database system must maintain data <u>consistency</u> while read/write operations are conducted
  - In case of node failure, the overall system must maintain <u>availability</u>
  - Operations need low latency

Properties of distributed systems

- **C**onsistency
  - The system assures that operations are atomic, and changes are disseminated simultaneously to all nodes, with same results
  - A read is guaranteed to return the most recent write
- **A**vailability
  - The system must answer every request, even in case of failures
  - A non-failing node returns a reasonable response within a reasonable time, without guarantee of being the latest write
- **P**artition tolerance
  - Partition is any arbitrary split between nodes resulting in message loss in between
  - The system must be resilient to message losses between nodes
    - Consistency mechanisms must cope with message losses
    - Availability requires any partition to be able to answer a request

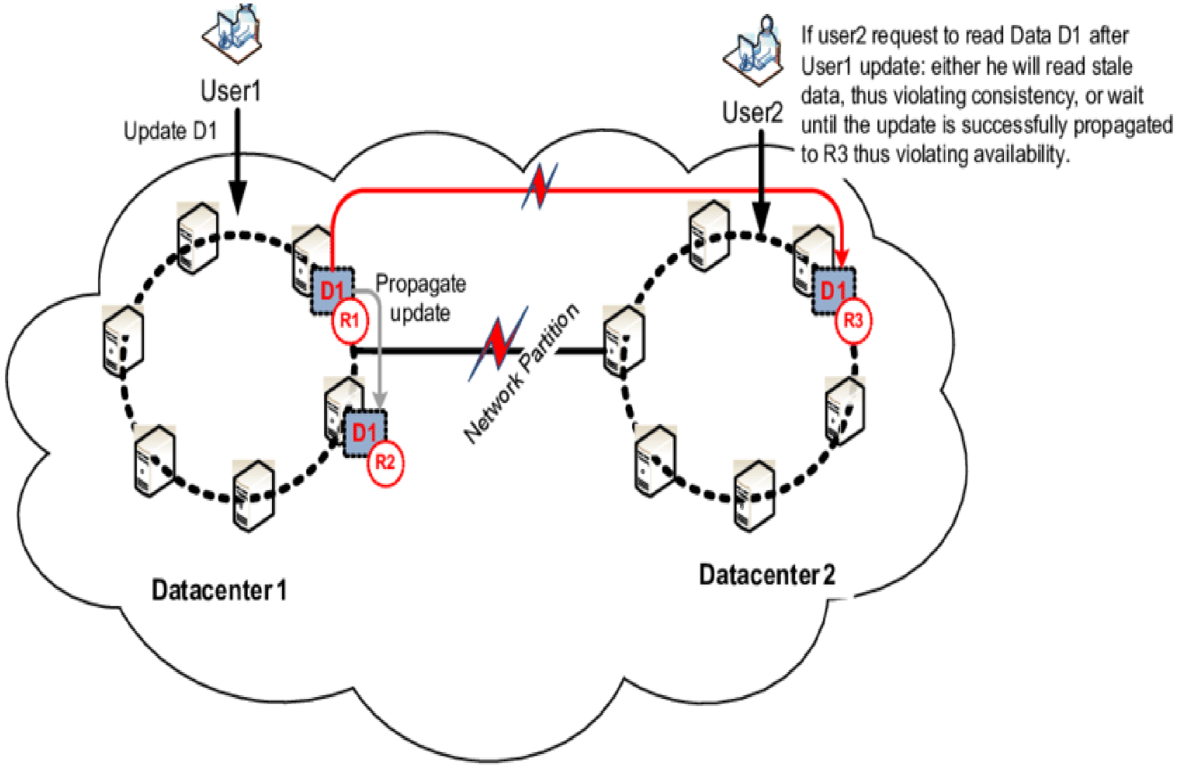
CAP theorem

- *A distributed system can only have at most two of the three properties:*
    - *CA, CP, AP*
- This was initially a Conjecture by Brewer 2000
- It has been demonstrated as a Theorem by Gilbert and Lynch 2002



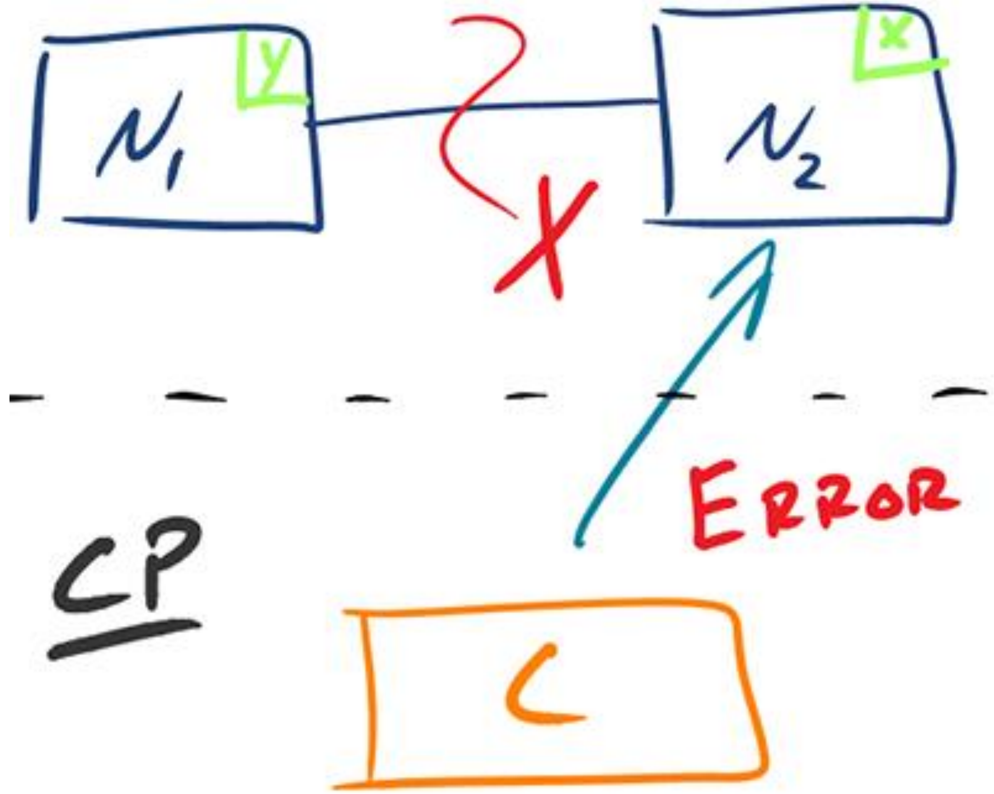
Consistency and Availability (CA)

- Such systems provide consistent and available services
- Do not tolerate partitions
  - In case of partition, systems become inconsistent
- RDMS follow this approach
  - Replication is used to achieve high availability
  - Transaction protocols (e.g. two-phase commit) ensure consistency
  - Partitions lead to conflicting replicas
- Two options to recover from such situations
  - Favour consistency (CP)
    - require consensus protocol
    - No answer to request if no consensus
    - sacrifice availability
  - Favour availability (AP)
    - Returned data is not guaranteed to be the latest update



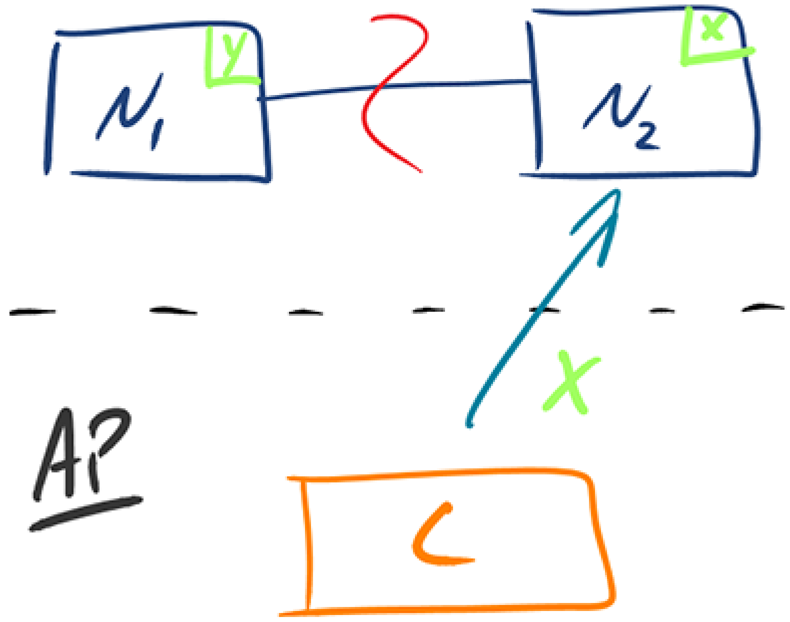
Consistency and Partition tolerance (CP)

- Systems with CP model provide strongly consistent service, even in the presence of partition
- Nodes must agree to ensure consistency
  - Use quorum and majority decision algorithms (e.g. Paxos protocol)
- In case of partition, consensus is delayed, therefore affecting availability (no answer possible)



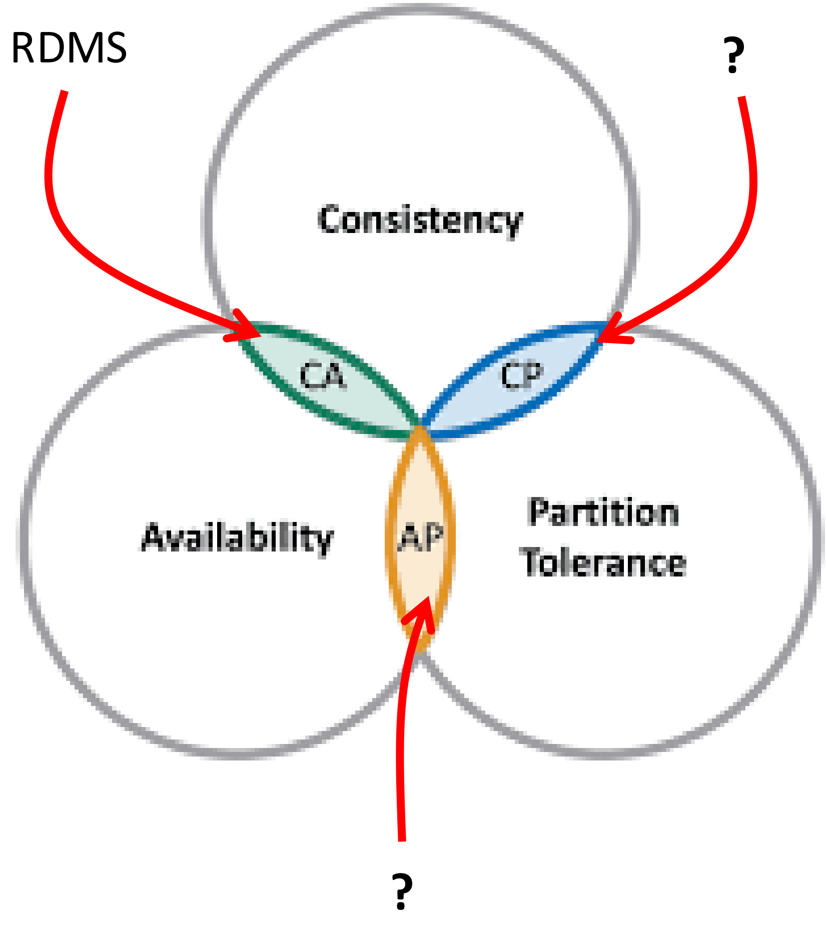
Availability and Partition tolerance (AP)

- Systems with AP model provide available service, even in the presence of partition
- Unreachable nodes hold temporarily inconsistent data
- Eventual consistency consists in ensuring the system converges to full consistency



Question

- Since RDMS have CA model, therefore cannot provide availability or consistency in the presence of partition
  - Which database systems can be designed according to CP or AP models?



NoSQL database systems

Many database systems

RDF
- *Virtuoso*
- *Jena*
- *Stardog*
- *RDF4J*
- *GraphDB*
- *Blazegraph*

Object
- *Caché*
- *Db4o*
- *Versant*
- *ObjectStore*

XML
- *MarkLogic*
- *Sedna*
- *Tamino*
- *BaseX*
- *eXist-db*

NoSQL

Key-Value
- Redis
- Memcached
- *Riak KV*
- *Aerospike*
- *SimpleDB*

Document
- MongoDB
- DynamoDB
- CouchBase
- Elasticsearch

Wide-Column
- Cassandra
- HBase
- *Accumulo*
- *HyperTable*

Graph
- Neo4J
- *Titan*
- *Giraph*
- *InfiniteGraph*

Relational
- Oracle
- MySQL
- MS SQL Server
- PostgreSQL
- DB2
- SQLite
- MS Access
- Teradata
- SAP Adaptive Server
- Hive
- FileMaker
- MariaDB
- Informix
- *Vertica*

NewSQL
- SAP HANA
- *Google Spanner*
- *Clustrix*
- *VoltDB*
- *MemSQL*
- *NuoDB*

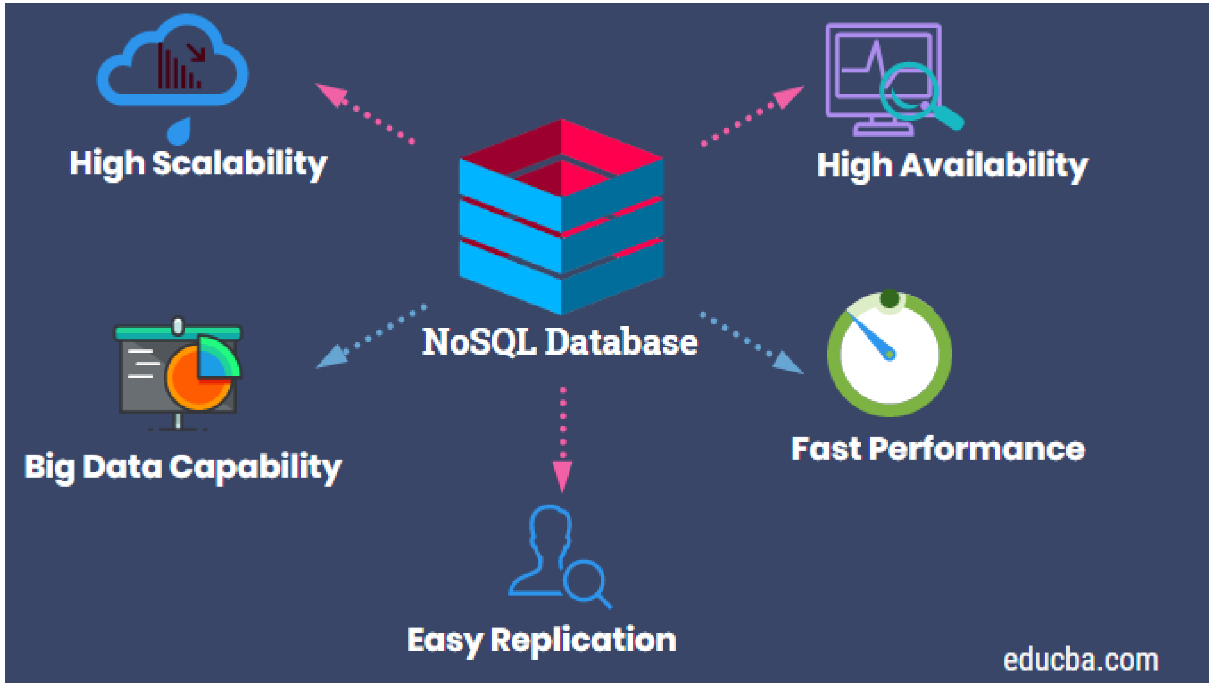
noSQL movement: history

- noSQL used first by Carlo Strozzi to query a RDB with no SQL (1998)
- NoSQL arbitrary name of a conference hosted by Rackspace (2009)
  - Presenting a collection of distributed non-relational databases
- Started as “no more SQL”, an anti SQL movement
- Became “not only SQL”, gentler definition
- Driven by the web
  - Minimally structured databases
  - Scale large amount of data across servers

2000	neo4j
2004	Google BigTable
2005	CouchDB
2007	amazon DynamoDB (research paper)
2008	cassandra (Facebook open source)

Characteristics of noSQL databases

- Do not respect the ACID principles
  - **Consistency** cannot be achieved while targeting high availability
  - Substituted by **eventual consistency** (in a parallel environment)
- No standard declarative query language
  - Most use variations of SQL
    - Cosmos DB, Cassandra CQL, Elasticsearch SQL, Cockraoch Labs, MongoDB
  - Couchbase uses N1QL (SQL like)
- Do not have predefined schema
  - Rely on dynamic schema
- Built to scale horizontally (vs. vertical scalability of RDBMS)

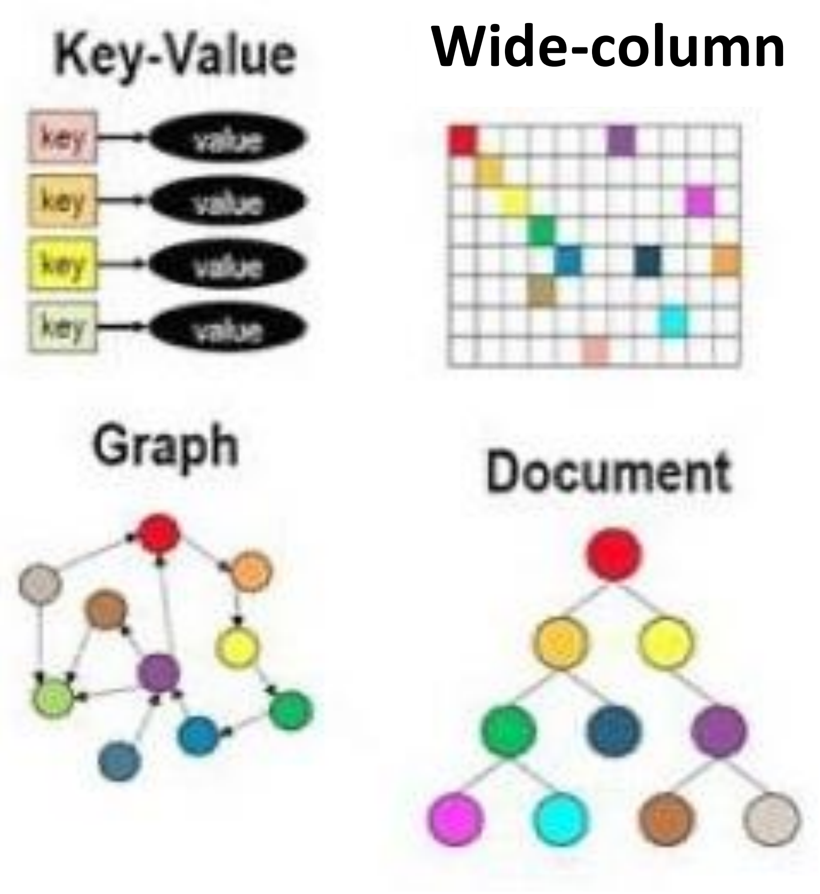


BASE principles

- Basic availability
  - Highly distributed DBM, with replication, to provide availability even with failures
- Soft State
  - Consistency is the problem of the developer not the DBMS
- Eventual consistency
  - Database should converge to consistent state in the future (operations can be executed without waiting for prior consistency)

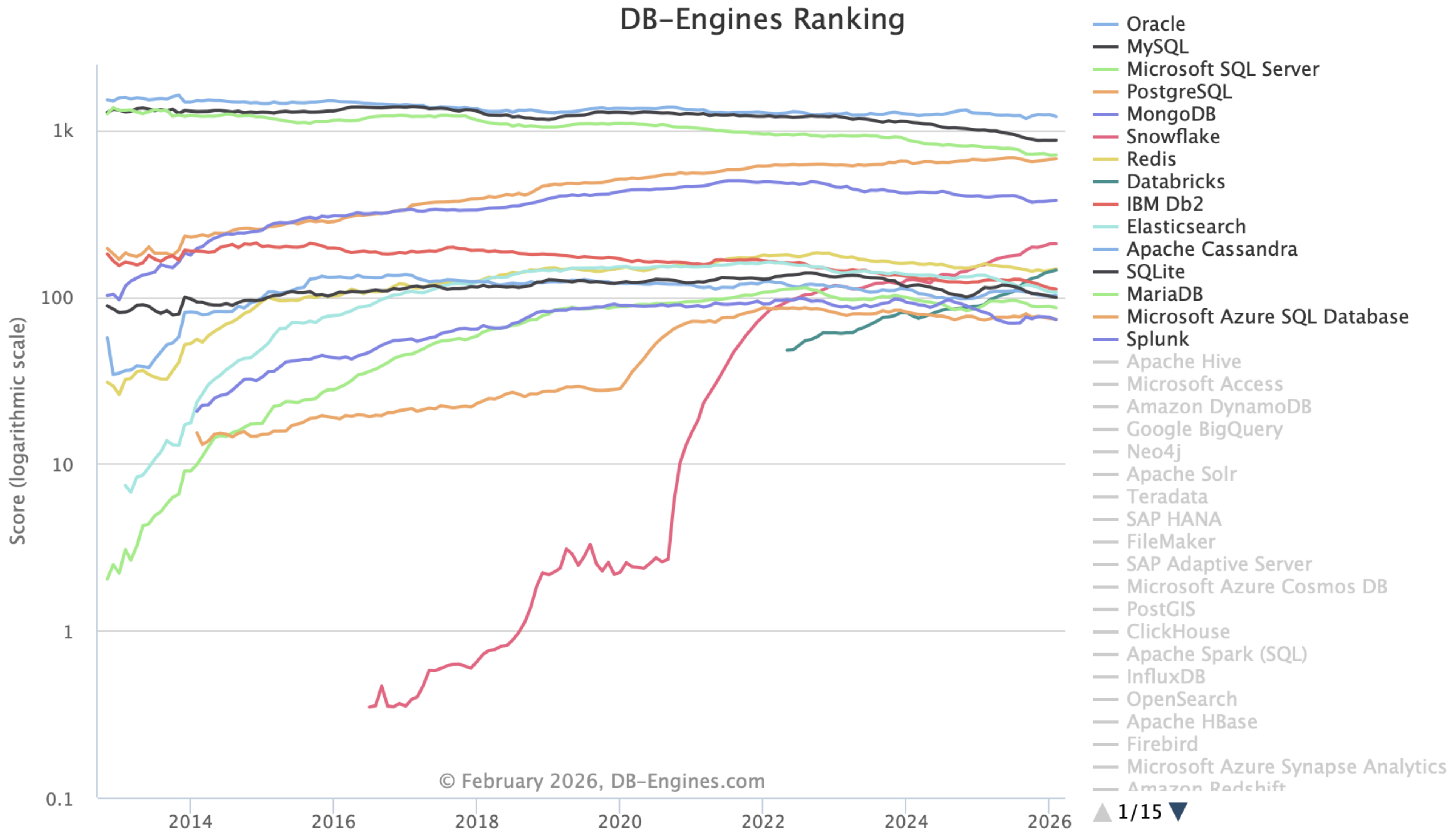
ACID	BASE
☐Strong Consistency	☐Weak Consistency
☐Isolation	☐Availability first
☐Focus on "commit"	☐Best effort
☐Nested transactions	☐Approximated answers
☐Less Availability	☐Aggressive(optimistic)
☐Conservative (pessimistic)	☐Simpler!
☐Difficult evolution (e.g. schema)	☐Faster
	☐Easier evolution

Four noSQL data models



Key-Value Store	redis	riak
Wide Column Store	HBASE	cassandra
Document Store	mongoDB	CouchDB relax
Graph Store	Neo4j	InfiniteGraph The Distributed Graph Database

Database systems usage



Key-value

- Based on the key-value pattern
- Data is stored in hashtable (dictionary or map) data structure as objects
- Objects are opaque to the database (uniquely read written)
- Objects are indexed by a key
- Schema-less
- Mostly in-memory

redis

riak KV

ORACLE NOSQL DATABASE

amazon DynamoDB
(also document)

Azure Cosmos DB
(also document)

Document

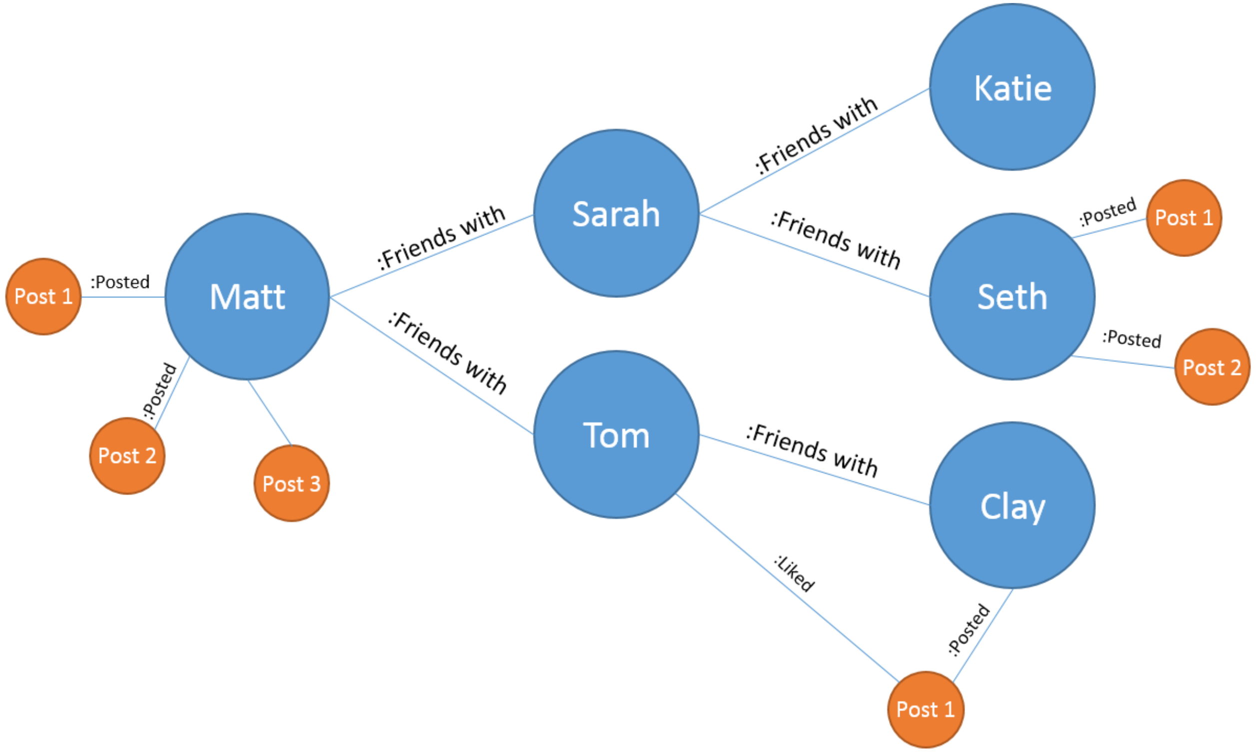
- Extend key-value model to key-document lookup
- Data is structured (set of key-value) and typically stored in JSON or XML
  - Hence the term document
- Documents are indexed by unique key
- Database has access to internal structure of document therefore provide access to fields…
- Indexes are usually used for fast query
- Apache CouchDB and MongoDB use JSON markup in a document data store
- Include search capabilities (search patterns)

Wide-column

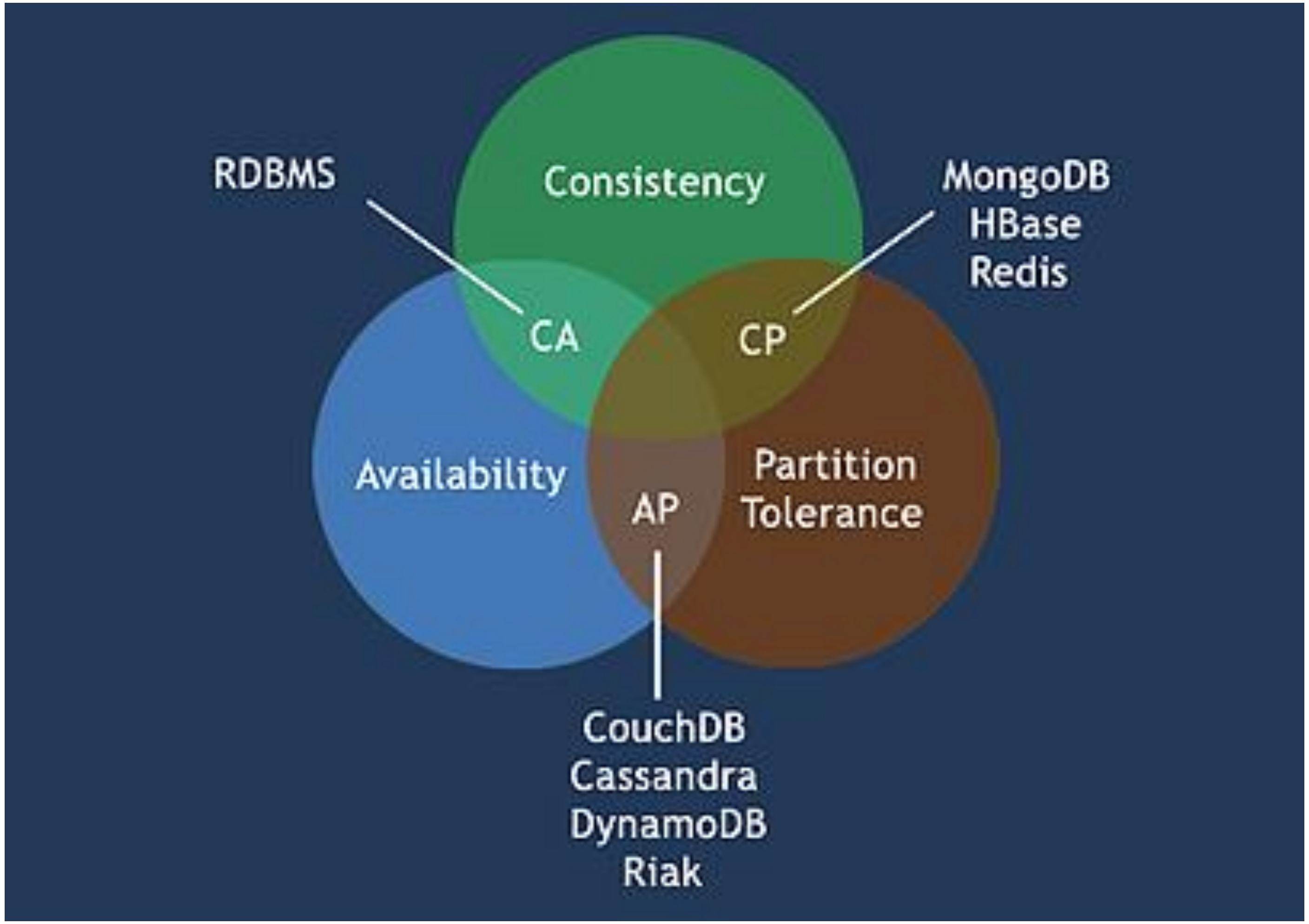
- Also called column family databases
- Use "flexible" tables (rows and columns)
  - <u>Sparse data matrix</u>
- Names and formats of columns can vary from a row to another
  - Not all columns have values
- Vertical sharding
  - Column families (set of columns) stored on separate computers
- Horizontal sharding
  - Within a column-family , rows are stored on different computers

Graph

- Focus on relationship between entities
  - Nodes are entities
  - Edges are relations (properties)
- Widely used by social networks



NoSQL and CAP properties



Critics of CAP theorem

- Trade-off between Consistency and Availability in case of partition is obvious
- Even in the absence of partition (normal operations), one has to choose between Latency and Consistency
- High availability requires **replication**

- Data updates sent to all replicas simultaneously
  - Without pre-processing layer -> Latency
  - With pre-processing layer -> consistency
- Data updates sent to agreed-upon location first
  - Synchronous -> consistency
  - Asynchronous -> latency (e.g. PNUTS)
- Data updates sent to an arbitrary location (e.g. Dynamo, Cassandra, Riak)
  - Synchronous -> consistency
  - Asynchronous -> latency

PACELEC theorem

- on **P**artition tradeoff between **A**vailability and **C**onsistency,
- **E**lse tradeoff between **L**atency and **C**onsistency

DDBMS	PA	PC	EL	EC
BigTable/HBase		✓		✓
Cassandra	✓		✓	
Cosmos DB	✓		✓	
Couchbase		✓	✓	✓
DynamoDB	✓		✓	
FaunaDB		✓	✓	✓
Hazelcast IMDG	✓	✓	✓	✓
Megastore		✓		✓
MongoDB		✓		✓
MySQL Cluster		✓		✓
PNUTS		✓	✓	
Riak	✓		✓	
VoltDB/H-Store		✓		✓

NoSQL aims

- Flexible
  - no fixed schema
- Highly available
  - replication across compute clusters
- Web-scale
  - automatic horizontal sharding

NoSQL do not support

- Schema: Data model can evolve
  - Data typically exchanged as JSON
- SQL: No standard query language
  - Proliferation of APIs
  - Simpler lighter-weight interactions
- Transaction processing: no ACID
  - BASE consistency: Basically Available, Soft state, Eventually consistent

NoSQL characteristics

- **Distributed!**
  - Sharding: splitting data over servers “horizontally”
  - Replication
- Lower-level than RDBMS/SQL
  - Simpler ad hoc APIs
  - Programmer ensures consistency (programming not querying)
  - Operations simple and cheap
- Different flavours (for different scenarios)
  - Different CAP emphasis
  - Different scalability profiles
  - Different query functionality
  - Different data models

RDB vs. noSQL

- Relational Databases don’t solve everything
  - SQL and ACID add overhead
  - Distribution not so easy
- NoSQL: what if you don’t need SQL or ACID?
  - Something simpler
  - Something more scalable
  - Trade efficiency against guarantees

Query language standardization