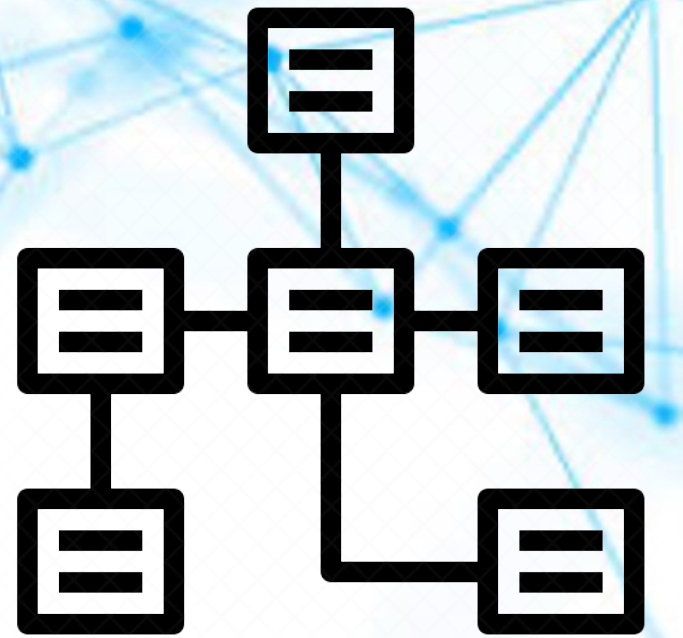
Introduction to NoSQL Databases

RELATIONAL DATABASES

Benefits of Relational databases:

- Designed for all purposes
- ACID
- Strong consistency, concurrency, recovery
- Mathematical background
- Standard Query language (SQL)
- Lots of tools to use with i.e: Reporting services, entity frameworks

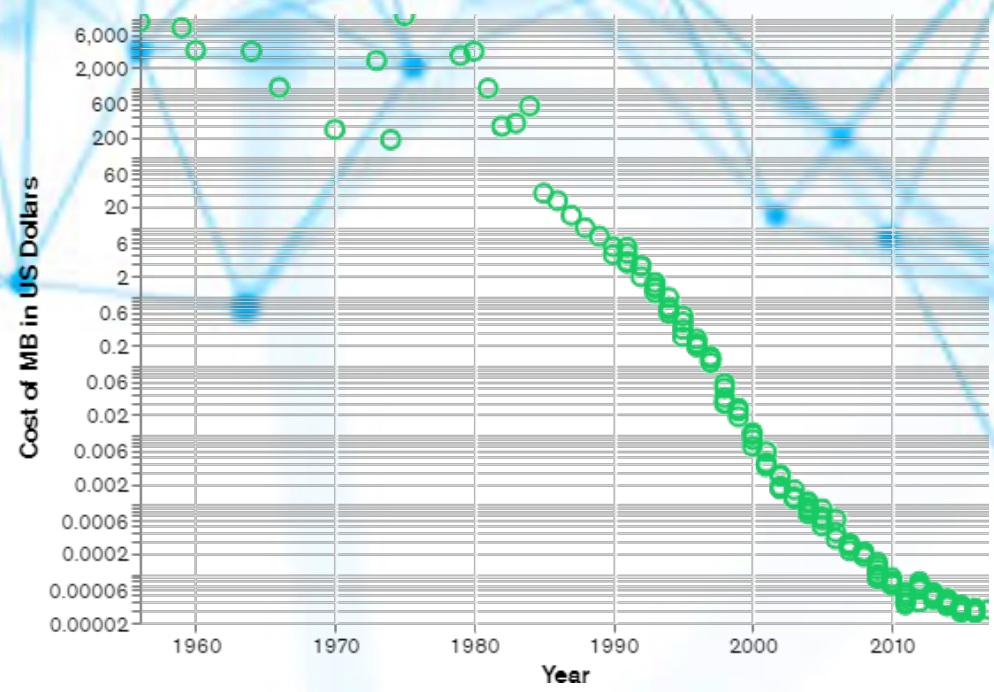


RELATIONAL DATABASES

In general, RDBMS systems have been considered as the **one-size-fits-all** data retrieval and persistence solution for decades

RELATIONAL DATABASES - CHALLENGES

- **Dramatic decrease in storage costs** - Exponential rise in data applications
- **Variations in Data**
- Continuously **evolving schema** with change in requirements



RELATIONAL DATABASES - CHALLENGES

Relational databases were not built for **distributed applications.**

Because...

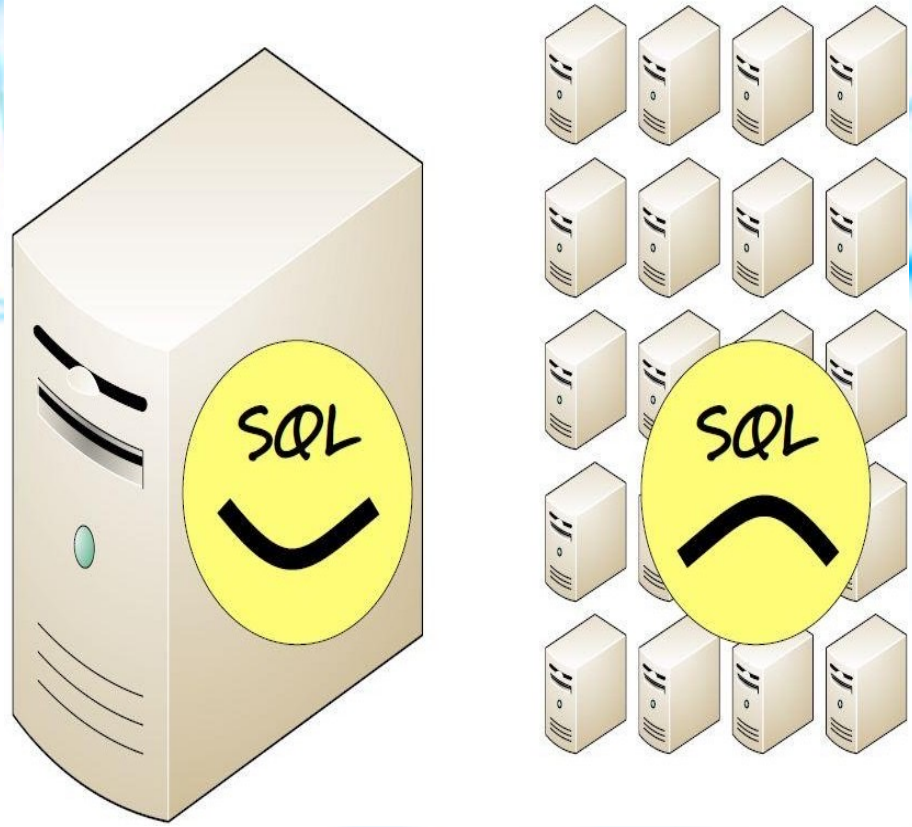
- Joins are expensive
- Hard to scale horizontally
- Expensive (product cost, hardware, maintenance)

RELATIONAL DATABASES - CHALLENGES

Relational databases were not built for **distributed applications.**

Because...

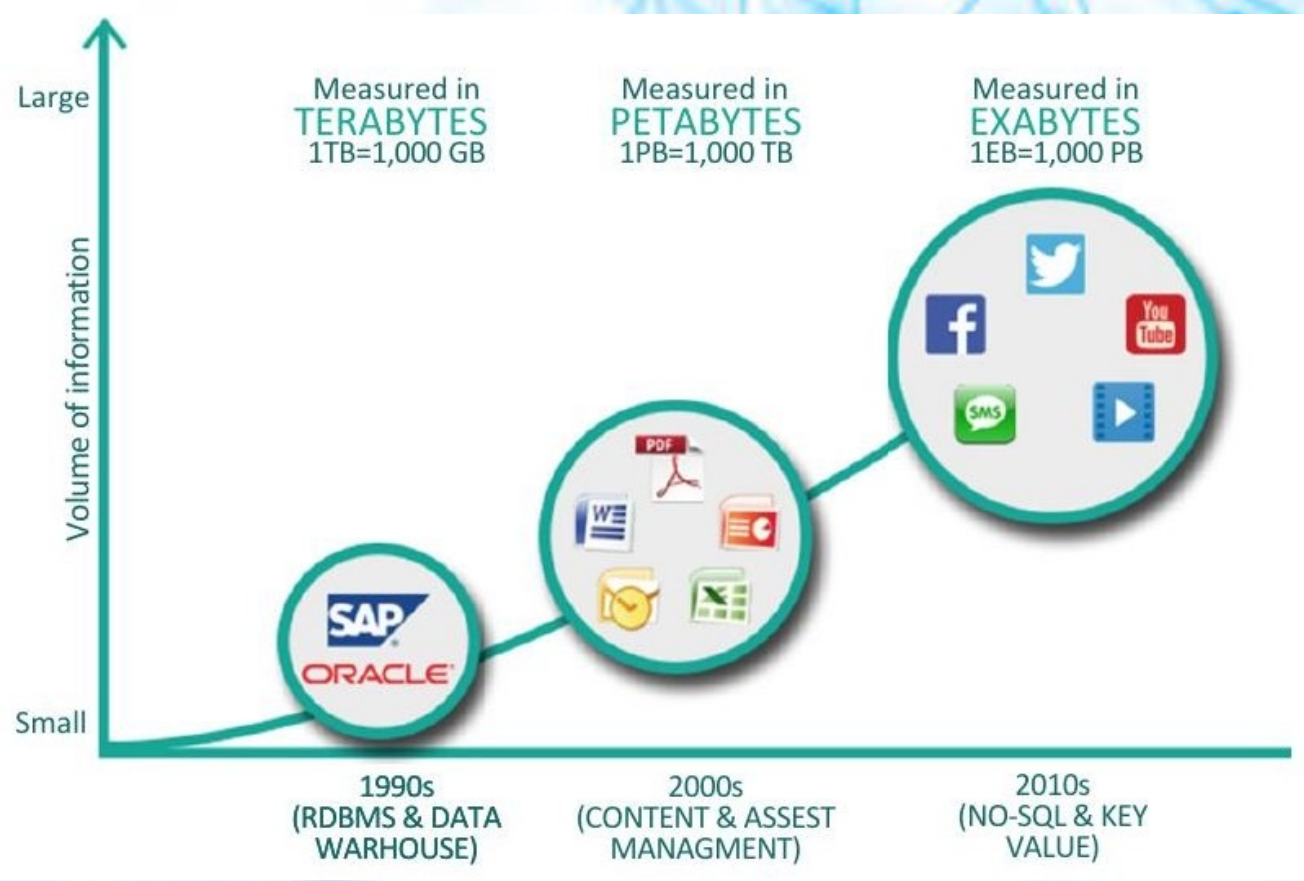
- Joins are expensive
- Hard to scale horizontally
- Expensive (product cost, hardware, maintenance)



MODERN DATA REQUIREMENTS

- Explosion of **social media sites** (Facebook, Twitter) with large data needs
- Rise of **cloud-based solutions** such as Amazon S3 (simple storage solution)
- Constantly **changing requirements**
- High-Velocity Data requiring fast query processing
- Increasingly sparse and semi-structured data

MODERN DATA REQUIREMENTS



NOSQL DATABASES

NoSQL stands for:

- No Relational
- No RDBMS
- **Not Only SQL**
  - Allows SQL-like query languages to be used.

NoSQL is an umbrella term for all databases and data stores that do not follow the RDBMS principles

NOSQL DATABASES

"Next Generation Database Management Systems mostly addressing some of the points: being **non-relational, distributed, open-source** and **horizontally scalable.**"

-- Nosql-database.org

The primary objective of a NoSQL Database is to have:

- Simplicity of design
- Horizontal scaling
- Finer control over availability

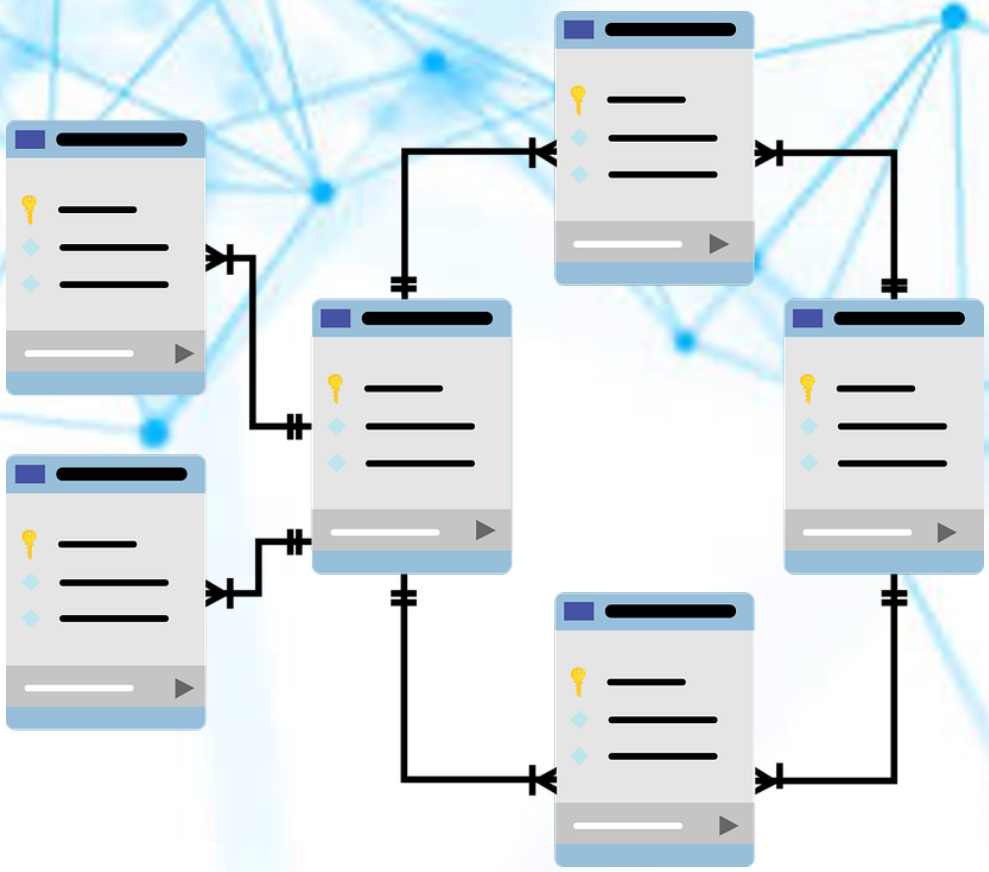
CHARACTERISTICS OF NOSQL DATABASES

THEY AVOID	THEY ALLOW
Overhead of ACID transactions	Easy and frequent changes to DB
Complexity of SQL query	Fast development
Burden of up-front schema design	Large data volume
DBA presence	Schema less
Transactions (It should be handled at application layer)	Distributed

SCHEMA BASED DATA MODELLING

- In RDBMS, a schema describes every functional element, including tables and rows

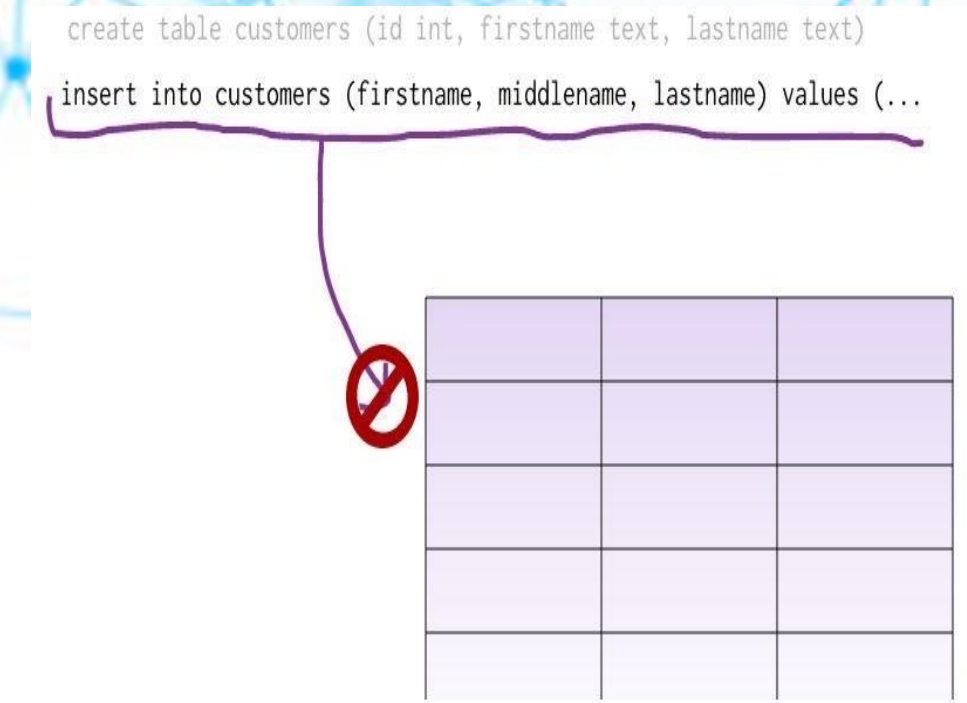
- Exerts a high degree of control and prevents capture of low-quality data



SCHEMA BASED DATA MODELLING

Problems

- Cannot add a record which does not fit the schema
- Need to add NULLs to unused items in a row
- Need to consider the datatypes - cannot add a string to an integer field
- Cannot add multiple items in a field



SCHEMALESS DATA MODEL

In NoSQL Databases:

- There is no schema to consider – No need to conform to a rigid schema
- There is no unused cell
- There are no datatype enforcements on columns
- Most of the considerations are done in the application layer
- Data can be rapidly transformed as requirements change
- Facilitates the storage of unstructured data as well as structured data

SCHEMALESS DATA MODEL

Information stored in **JSON-style** documents which can have **varying sets of fields** with **different data types** for each field

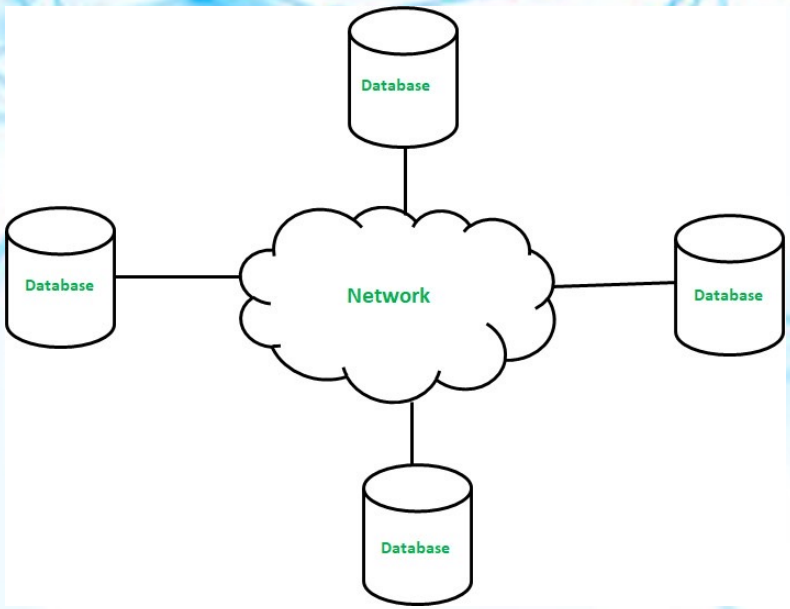
```
{
        "name":"Joe",
        "age":30,
        "interests":"football"
}
{
        "name":"Kate",
        "age":25
}
```

SCHEMALESS DATA MODEL - ADVANTAGES

- No pre-defined database schemas
- No data truncation
- Suitable for real-time analytics
- On demand scalability to meet extreme Volumes, Velocity and Variety of data

DISTRIBUTED ARCHITECTURE

- Multiple NoSQL databases can be created in a distributed fashion
- Data is physically stored across different sites
- Reaches Eventual consistency
- Offers auto-scaling and fail-over capabilities



BASE (NOT ACID)

Recall ACID for RDBMS desired properties of transactions

- Atomicity, Consistency, Isolation, and Durability

NoSQL systems provide BASE and do not provide ACID

- Basically Available
- Soft state
- Eventually consistent

ACID VS. BASE

The idea is that by giving up ACID constraints, one can achieve much higher performance and scalability

The systems differ in how much they give up

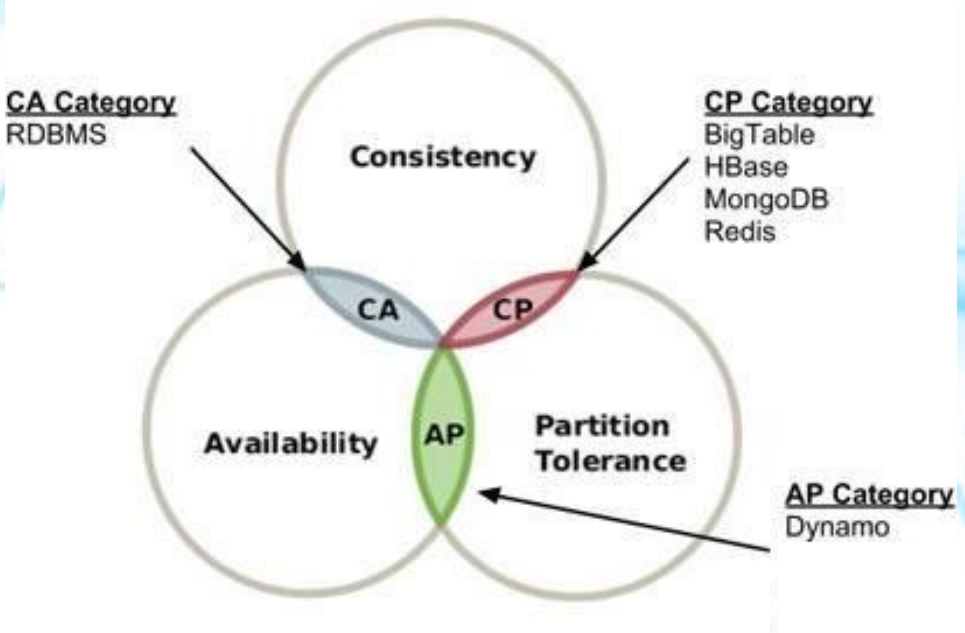
- e.g. most of the systems call themselves "**eventually consistent**", meaning that updates are eventually propagated to all nodes
- but many of them provide mechanisms for some degree of consistency, such as **multi-version concurrency control** (MVCC)

CAP THEOREM

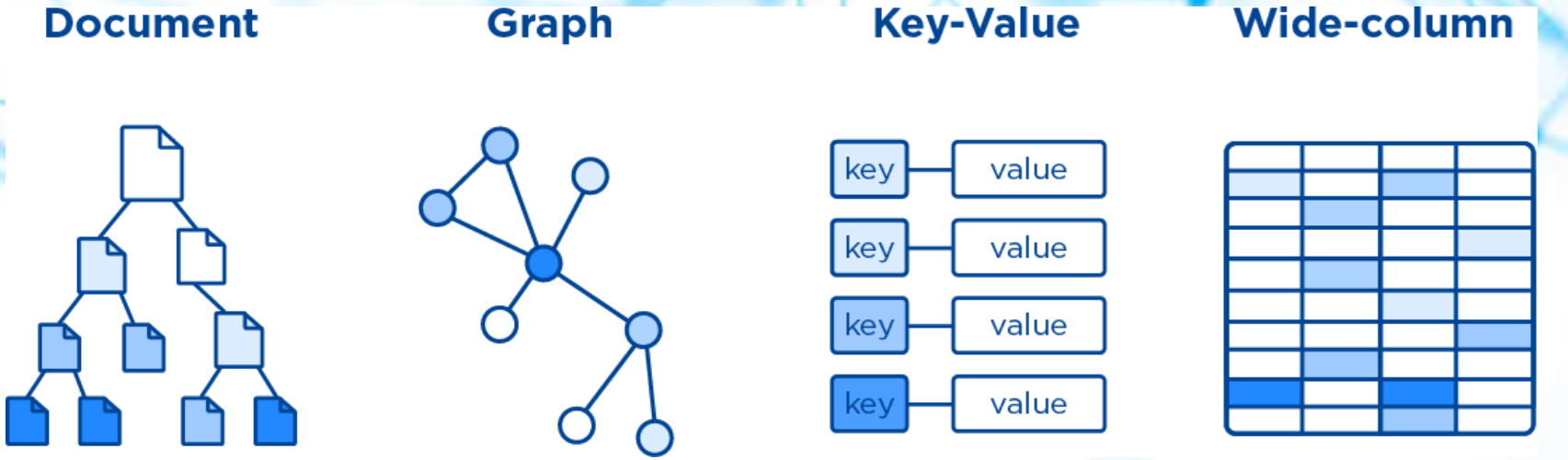
Often Eric Brewer's CAP theorem cited for NoSQL

A system can have only two out of three of the following properties:

- Consistency – Data is same across all sites even after updates and deletions
- Availability – Data is always immediately available
- Partition-Tolerance – System continues to work even in the presence of a partial network failure



TYPES OF NOSQL DATABASES

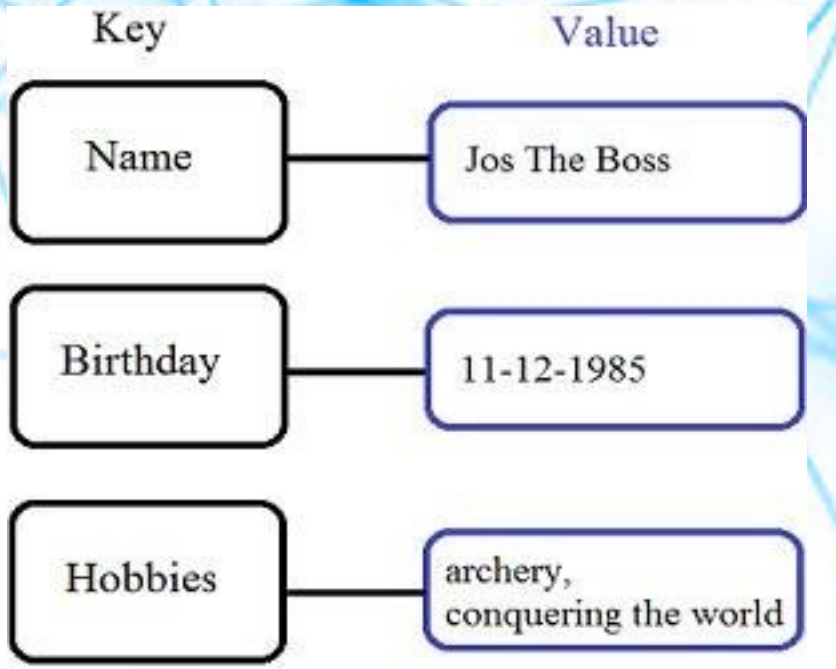


KEY-VALUE STORE

Simplest NoSQL Database

Data is stored in the form of a key-value hash table

- Each key is unique
- Its corresponding value can be any data type (string, JSON, Blob e.t.c.)
- Values can also contain nested key-value pairs

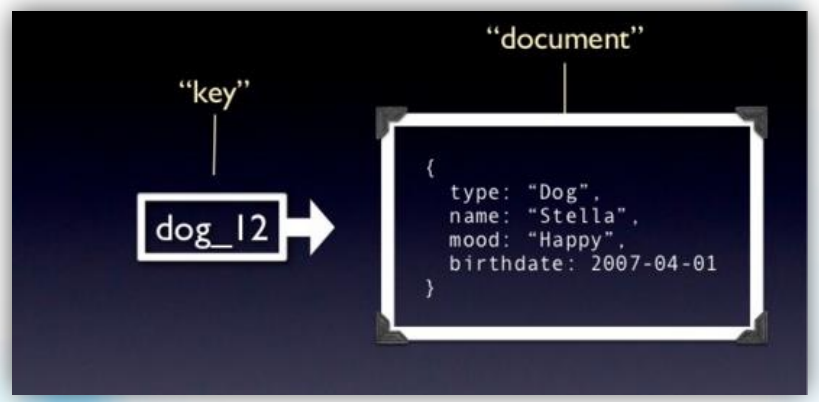


DOCUMENT-ORIENTED

Subclass of key-value store

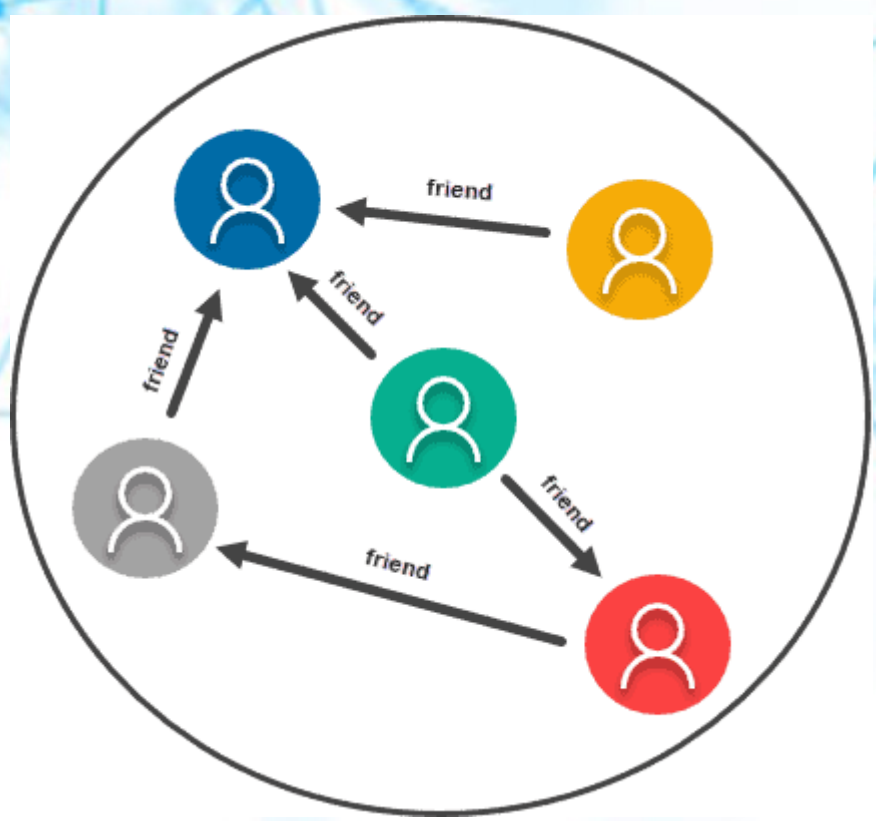
Assumes a certain internal document structure in the data

A query language provides the ability to perform queries based on this internal structure



GRAPH BASED

- Used to store fine-grained networks of inter-connected data
- Stores entities as well the relations amongst the entities
- Entity is stored as a node with the relationship as edges
- Traversing **persisted relationships** are faster



COLUMN BASED

- Based on the BigTable paper by Google
- Variable-width tables
- Rows do not need to have the same columns
- Columns can be added to any row without having to add them to other rows

ColumnFamily: Authors	
Key	Value
"Eric Long"	Columns Name: Value "email": "eric (at) long.com" "country": "United Kingdom" "registeredSince": "01/01/2002"
"John Steward"	Columns Name: Value "email": "john.steward (at) somedomain.com" "country": "Australia" "registeredSince": "01/01/2009"
"Ronald Mathies"	Columns Name: Value "email": "ronald (at) sodeso.nl" "country": "Netherlands, The" "registeredSince": "01/01/2010"

WHEN TO USE A NOSQL DATABASE?

- Large amounts of data
  - Terabytes and Petabytes of data
- Need horizontal scalability
- Need high throughput – fast reads
- Need a flexible schema
  - No fixed number of columns
- Need high availability
- Need to be able to store different data type formats
- Users are distributed – low latency
- Need redundancy in case of failures

WHEN NOT TO USE A NOSQL DATABASE?

- Need ACID Transactions
- Need ability to do JOINS
- Ability to do aggregations and analytics
- Have changing business requirements
- Queries are not available and need to have flexibility
- Have a small dataset

APACHE CASSANDRA

"**Apache Cassandra** is a free and open-source, **distributed, wide-column store**, NoSQL database management system designed to handle large amounts of data across many commodity servers, providing **high availability with no single point of failure.**"

Apache Cassandra uses its own query language CQL

FEATURES OF CASSANDRA

- **Elastic Scalability**
- **Always on architecture**
- **Fast linear scale performance**
- **Flexible data storage**
- **Fast writes**

COMPANIES USING CASSANDRA

Netflix uses Apache Cassandra to serve all their videos to customers.

Uber uses Apache Cassandra for their entire backend.

BASICS OF CASSANDRA

Keyspace

- Collection of Tables

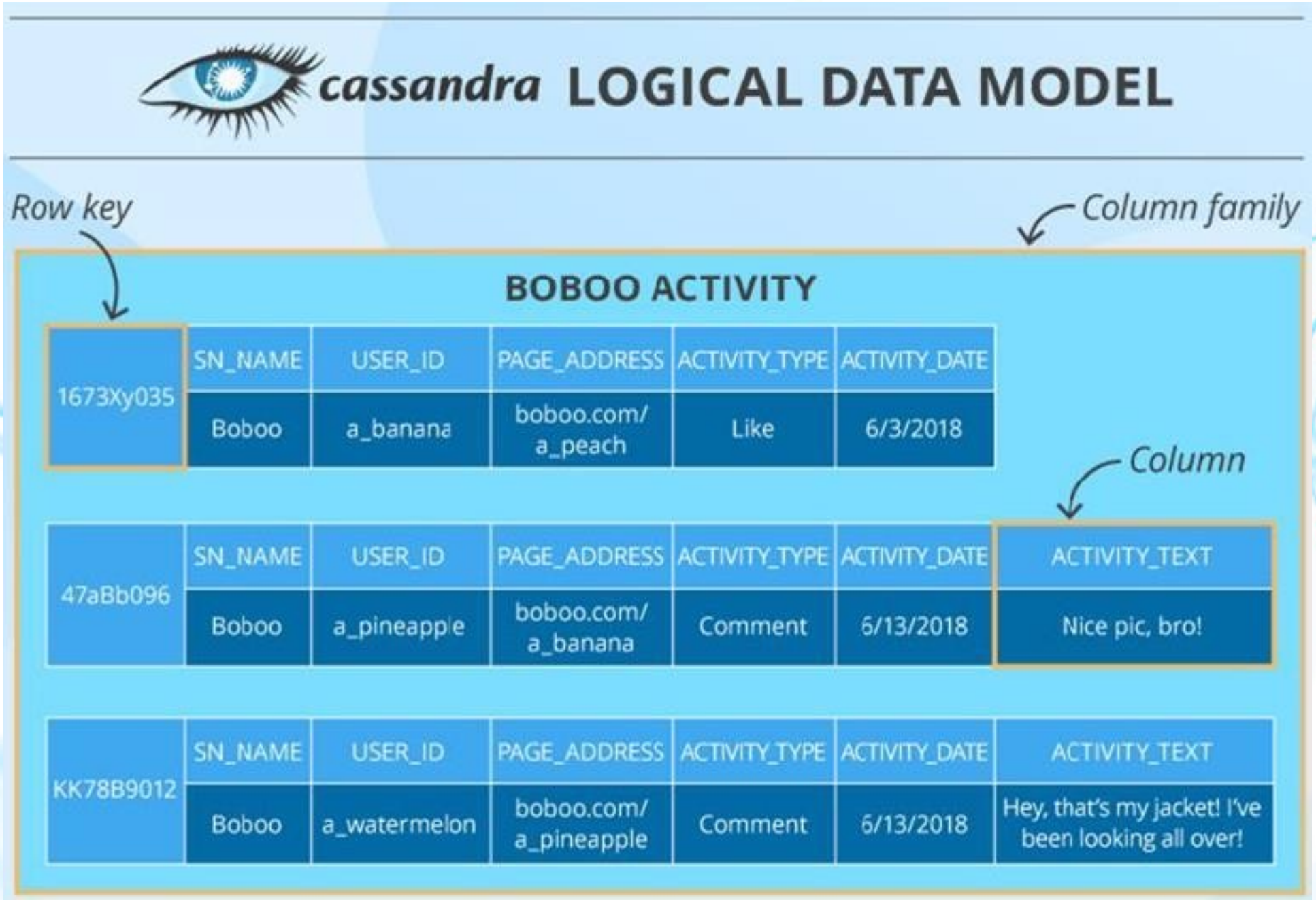
Table

- A group of partitions

Rows

- A single item

Last Name	First Name	Address	Email
Flintstone	Dino	3 Stone St	dino@gmail.com
Flintstone	Fred	3 Stone St	fred@gmail.com
Flintstone	Wilma	3 Stone St	wilm@gmail.com
Rubble	Barney	4 Rock Cir	brub@gmail.com


cassandra
LOGICAL DATA MODEL
Row key
Column family
BOBOO ACTIVITY
1673Xy035
SN_NAME
USER_ID
PAGE_ADDRESS
ACTIVITY_TYPE
ACTIVITY_DATE
Boboo
a_banana
boboo.com/ a_peach
Like
6/3/2018
Column
47aBb096
SN_NAME
USER_ID
PAGE_ADDRESS
ACTIVITY_TYPE
ACTIVITY_DATE
ACTIVITY_TEXT
Boboo
a_pineapple
boboo.com/ a_banana
Comment
6/13/2018
Nice pic, bro!
KK78B9012
SN_NAME
USER_ID
PAGE_ADDRESS
ACTIVITY_TYPE
ACTIVITY_DATE
ACTIVITY_TEXT
Boboo
a_watermelon
boboo.com/ a_pineapple
Comment
6/13/2018
Hey, that's my jacket! I've been looking all over!

THE BASICS OF APACHE CASSANDRA

Partition

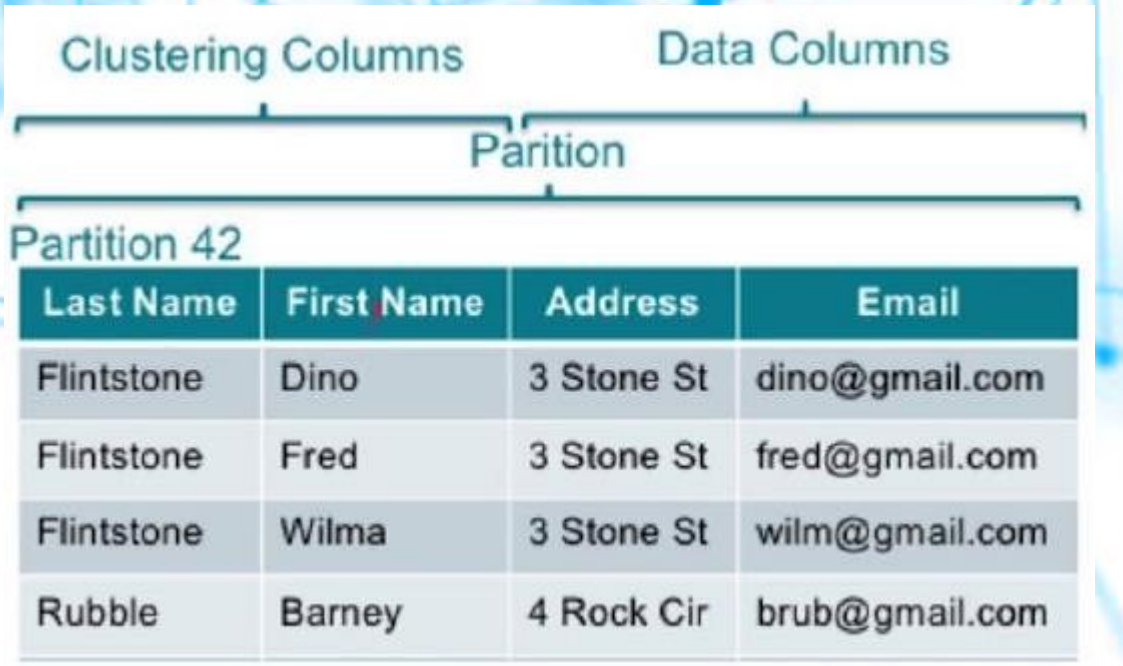
- Fundamental unit of access
- Collection of row(s)
- How data is distributed

Primary Key

- Primary key is made up of a partition key and clustering columns

Columns

- Clustering and Data
- Labeled element



Last Name	First Name	Address	Email
Flintstone	Dino	3 Stone St	dino@gmail.com
Flintstone	Fred	3 Stone St	fred@gmail.com
Flintstone	Wilma	3 Stone St	wilm@gmail.com
Rubble	Barney	4 Rock Cir	brub@gmail.com

Summary

SQL Databases

- Relational database (RDBMS)
- Table based with rows and columns and clearly defined schema
- Scales vertically by increasing server and hardware horsepower
- Well suited for high transactional based applications
- Common SQL DBs: MS SQL, MySQL, Oracle, PostgreSQL, DB2
- More mature application and support

NoSQL Databases

- Non-relational or distributed database
- Document based, key value pairs, graph or wide-column stores with dynamic schema
- Scales horizontally by adding more servers to the pool or resources
- Well suited for Big Data and large unstructured data sets
- Common NoSQL DBs: MongoDb, CouchDB, Google BigTable
- New offerings with less mature support