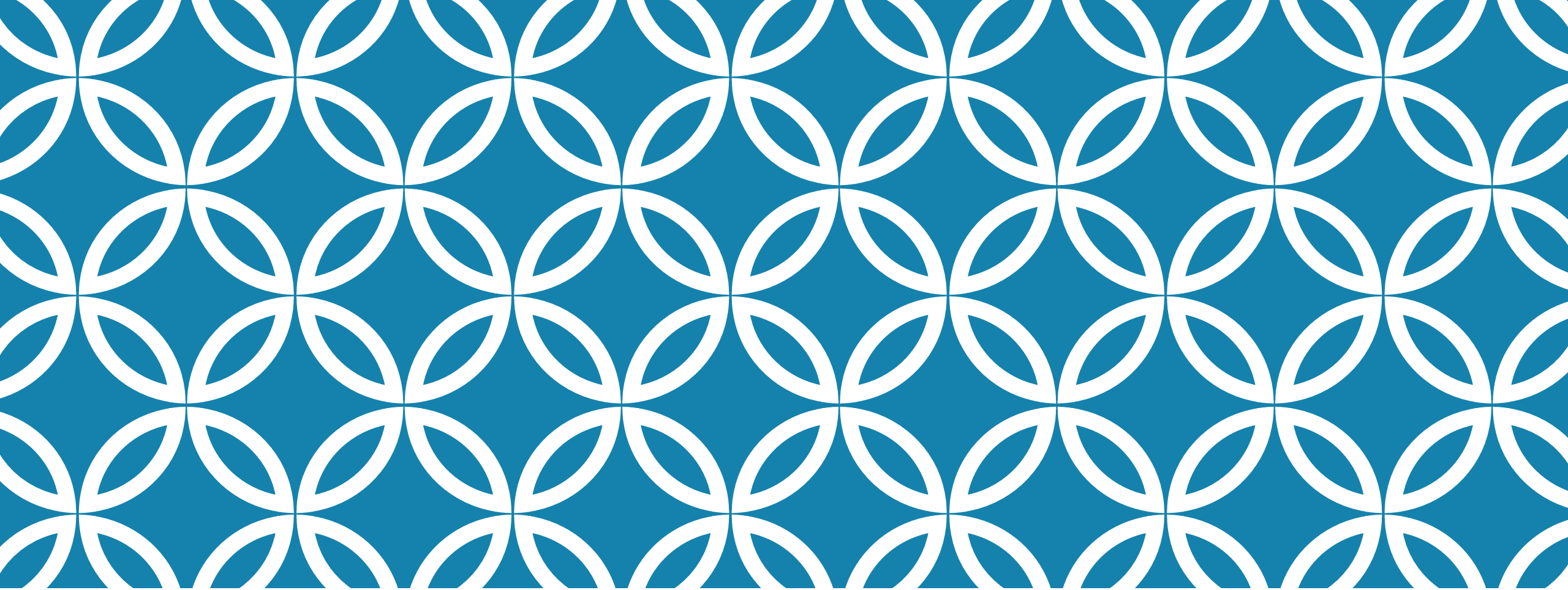
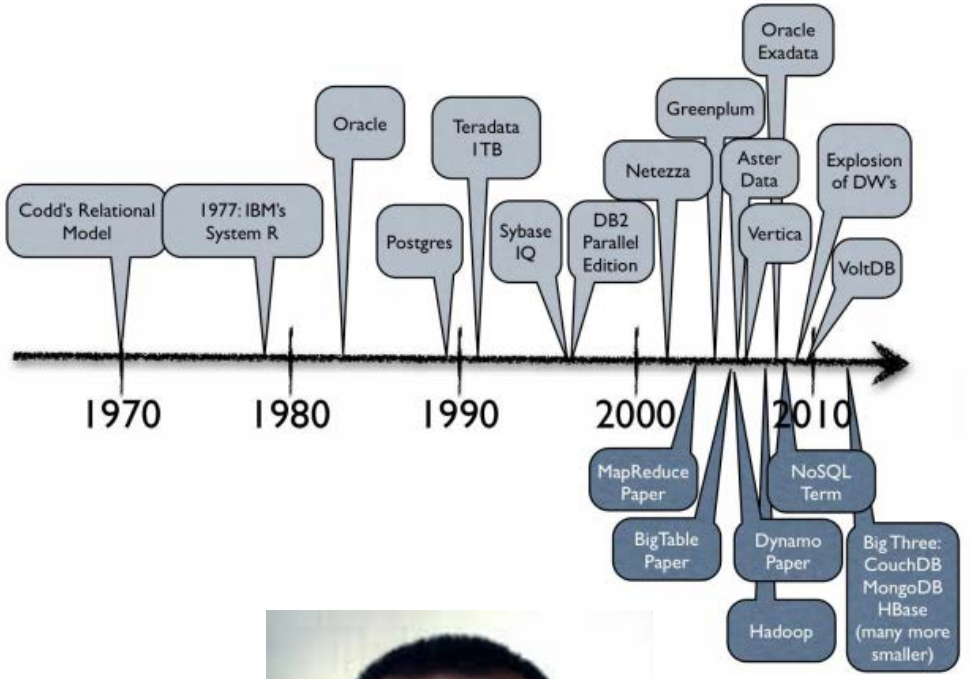


BIG DATA ANALYTICS
UNIT 3: NO SQL

niketamoda@gmail.com

NOT ONLY SQL

Evolution of No SQL



The term NoSQL was first used in 1998 by Carlo Strozzi while naming his lightweight, open-source "relational" database that did not use SQL.

Relational databases are often referred to as SQL systems. The term NoSQL can mean either "No SQL systems" or the more commonly accepted translation of "Not only SQL," to emphasize the fact some systems might support SQL-like query languages.

NOT ONLY SQL

DBMS has the following advantages

- **ACID properties**
- **Designed for all purpose**
- **Strong consistency, concurrency, recovery**
- **Standard Query language (SQL)**

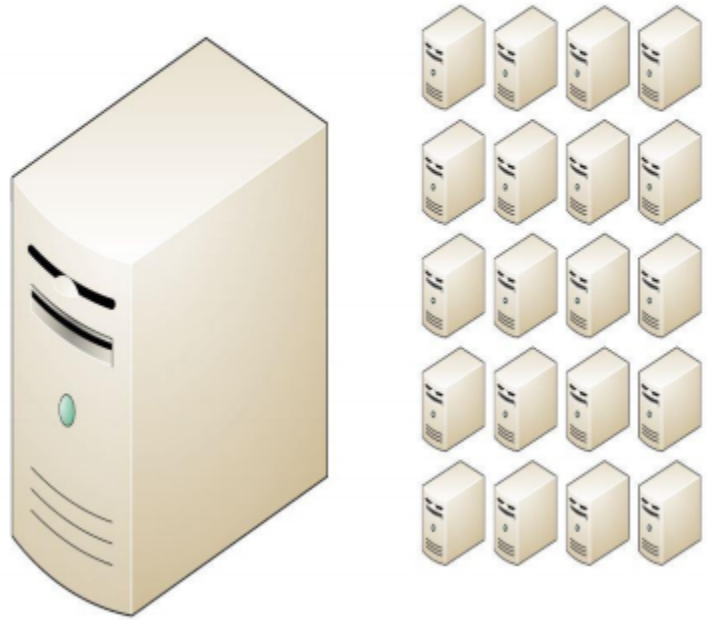
But...

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

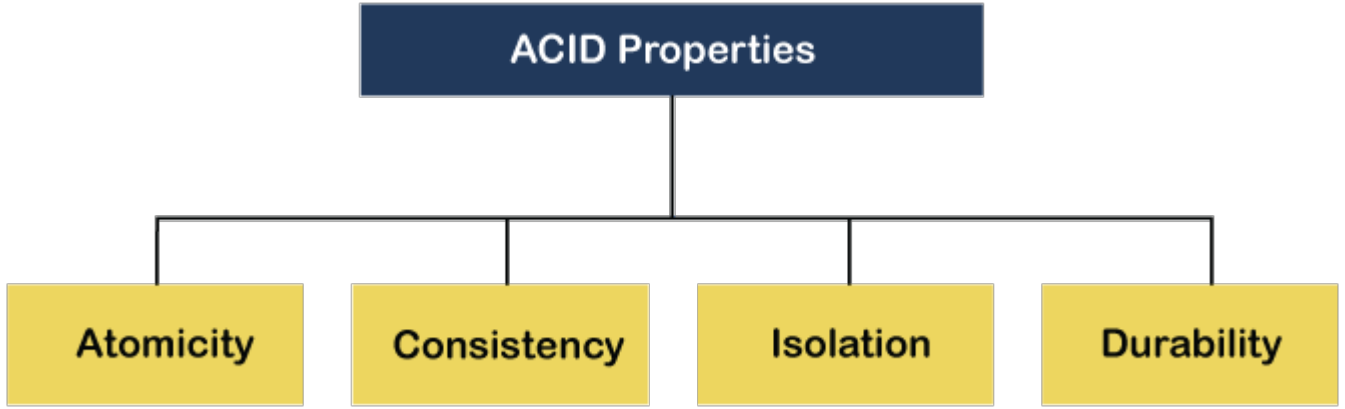
Because...

- ❑ Joins are expensive
- ❑ Hard to scale horizontally
- ❑ Impedance mismatch occurs
- ❑ Expensive (product cost, hardware, Maintenance)

Era of Distributed Computing



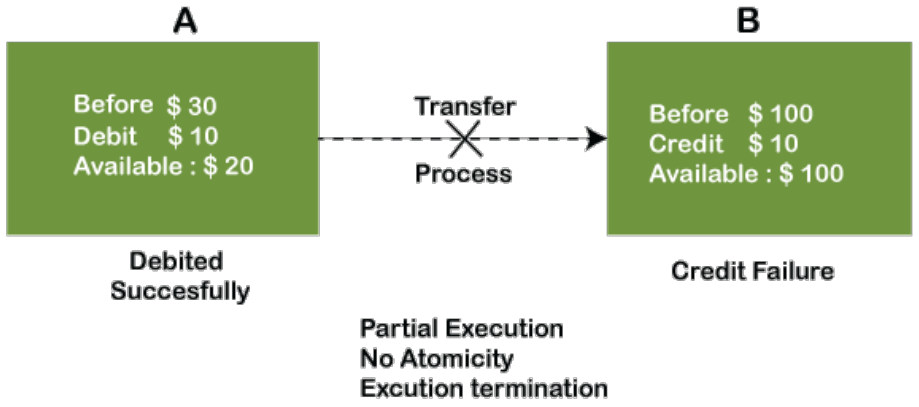
ACID PROPERTIES



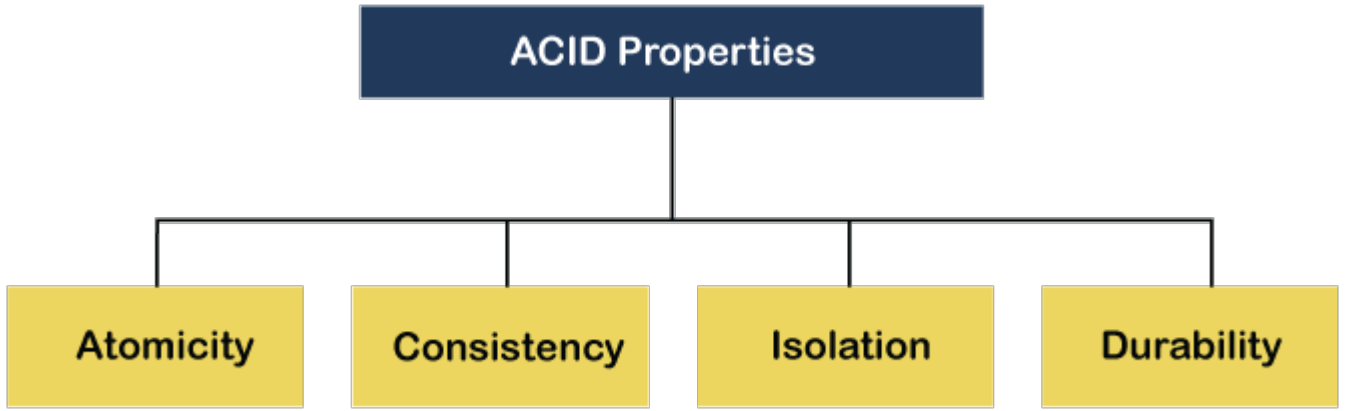
1) Atomicity:

It means if any operation is performed on the data, either it should be performed or executed completely or should not be executed at all.

It further means that the operation should not break in between or execute partially. In the case of executing operations on the transaction, the operation should be completely executed and not partially.



ACID PROPERTIES



2) Consistency:

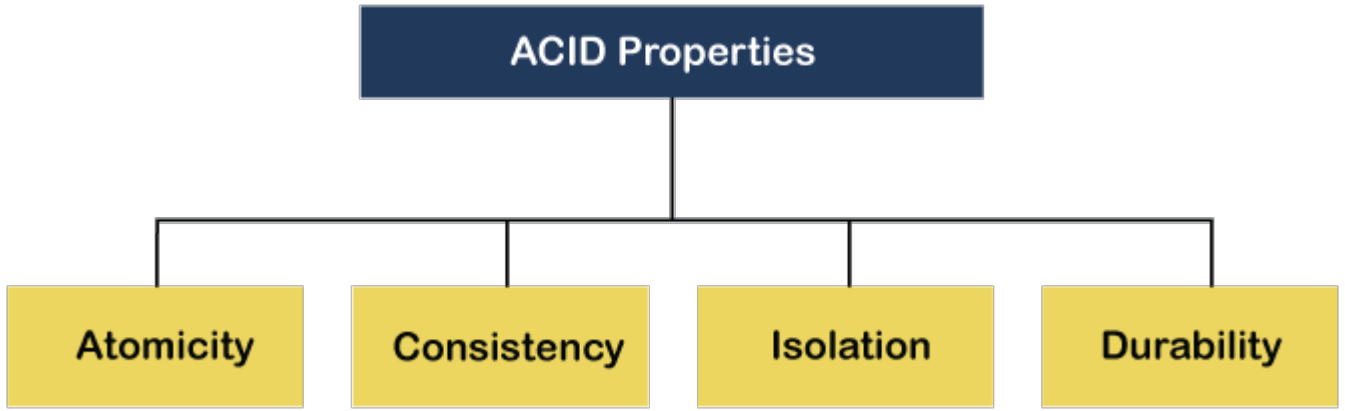
The word consistency means that the value should remain preserved always.

In DBMS, the integrity of the data should be maintained, which means if a change in the database is made, it should remain preserved always.

In the case of transactions, the integrity of the data is very essential so that the database remains consistent before and after the transaction.

The data should always be correct.

ACID PROPERTIES



3) Isolation: :

The term 'isolation' means separation.

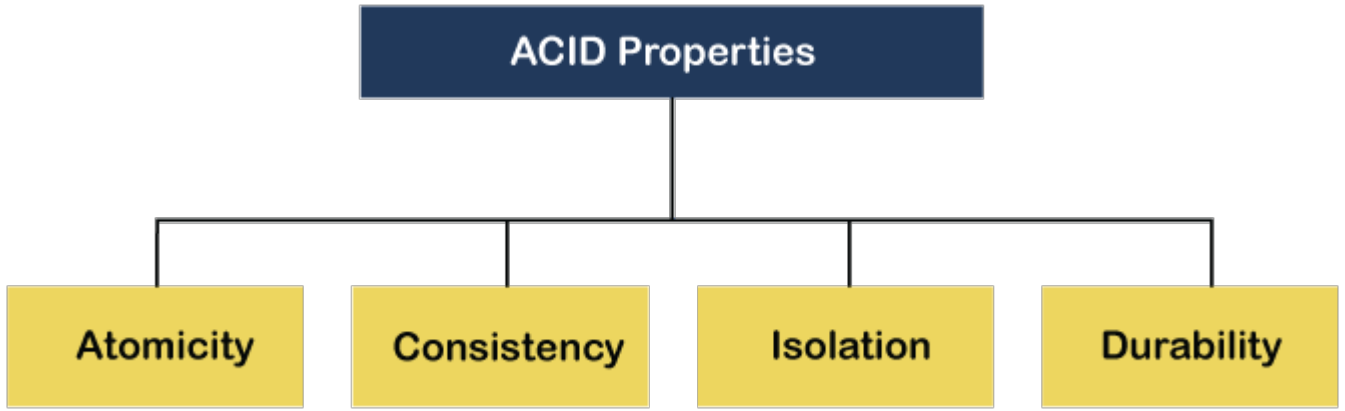
In DBMS, Isolation is the property of a database where no data should affect the other one and may occur concurrently.

In short, the operation on one database should begin when the operation on the first database gets complete.

It means if two operations are being performed on two different databases, they may not affect the value of one another.

In the case of transactions, when two or more transactions occur simultaneously, the consistency should remain maintained.

ACID PROPERTIES



4) Durability:

Durability ensures the permanency of something.

In DBMS, the term durability ensures that the data after the successful execution of the operation becomes permanent in the database.

The durability of the data should be so perfect that even if the system fails or leads to a crash, the database still survives.

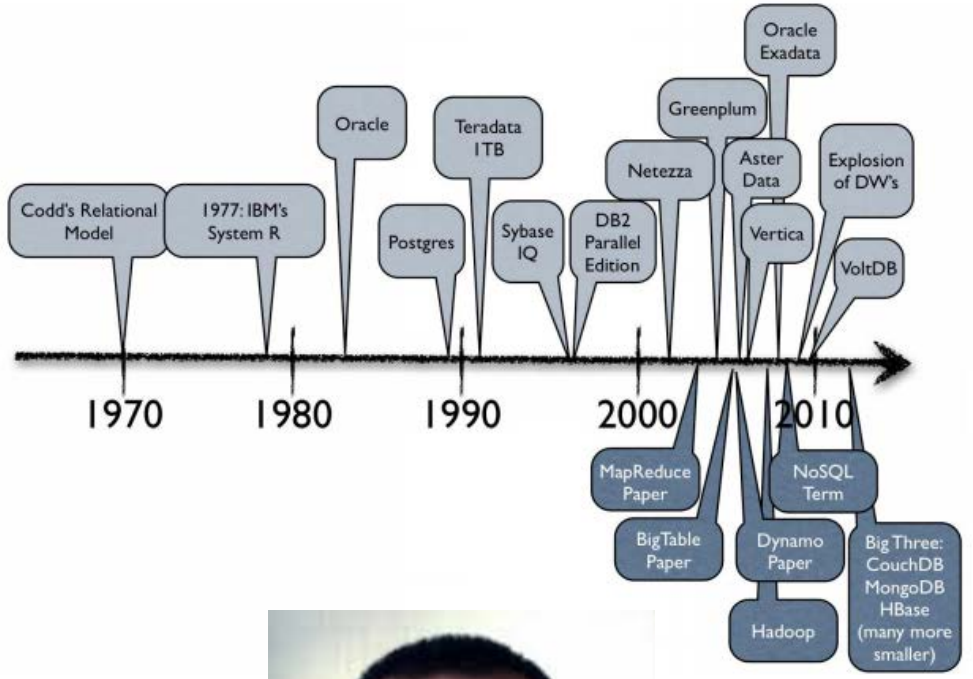
However, if gets lost, it becomes the responsibility of the recovery manager for ensuring the durability of the database. For committing the values, the COMMIT command must be used every time we make changes.

NOT ONLY SQL

NoSQL developed at least in the beginning as a response to web data, the need for processing unstructured data, and the need for faster processing.

The NoSQL model uses a distributed database system, meaning a system with multiple computers.

The non-relational system is quicker, uses an ad-hoc approach for organizing data, and processes large amounts of differing kinds of data.



NOT ONLY SQL

For general research, NoSQL databases are the better choice for large, unstructured data sets compared with relational databases due to their speed and flexibility.

Not only can NoSQL systems handle both structured and unstructured data, but they can also process unstructured Big Data quickly.

This led to organizations such as Facebook, Twitter, LinkedIn, and Google adopting NoSQL systems.

These organizations process tremendous amounts of unstructured data, coordinating it to find patterns and gain business insights. Big Data became an official term in 2005.

NOT ONLY SQL

Need

In real time data requirements are changed a lot.

Data is readily available with Facebook, Google, Twitter etc.

This data include the user information, social graphs, geographic locations etc.

To provide the quality services to the user we must be able to use the relevent technology which can operate on this data

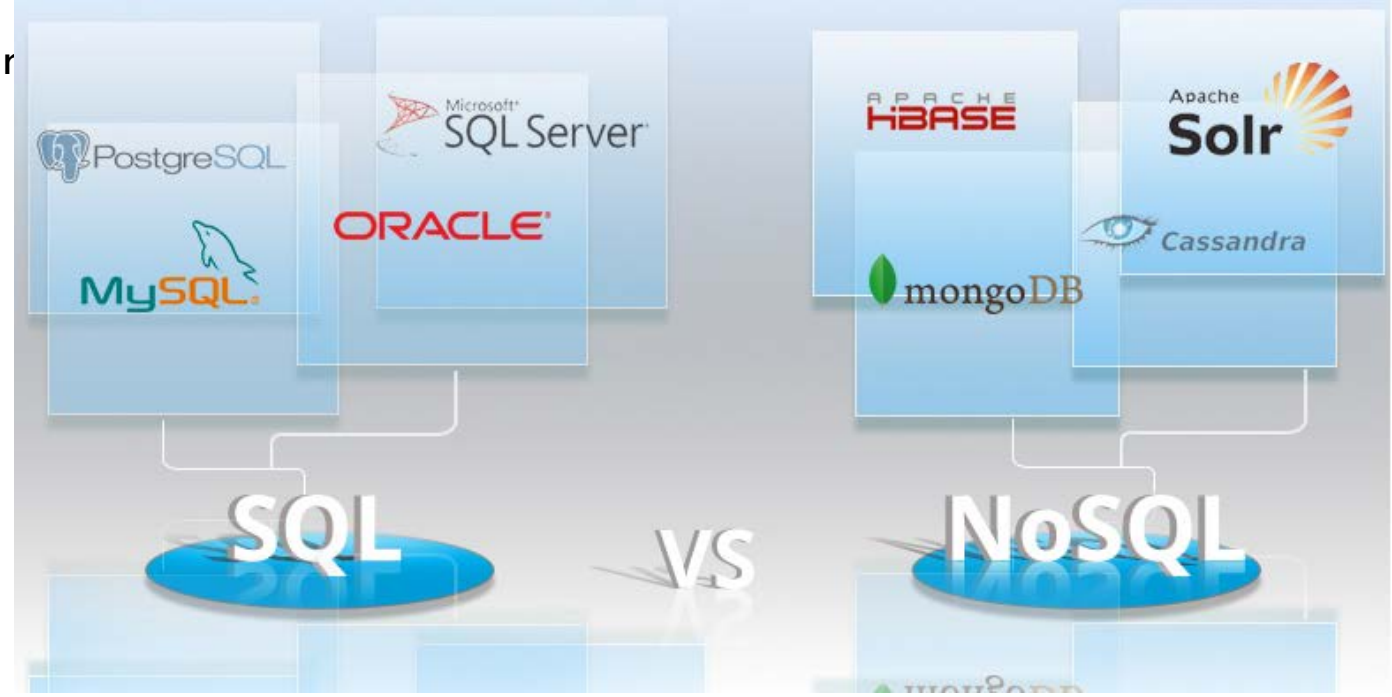
NOT ONLY SQL

Advantage of NoSQL

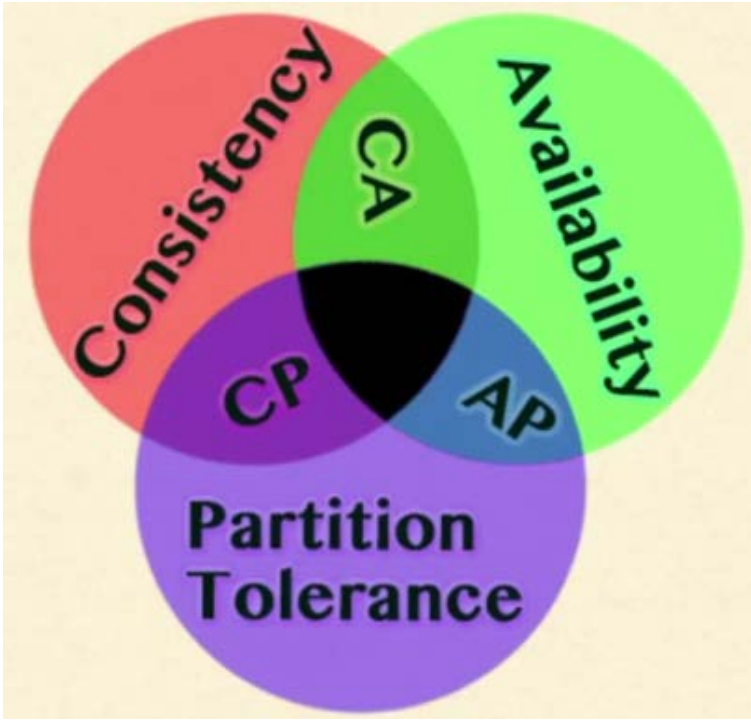
- Good resource scalable
- Lower operation cost
- Supports semi-structure data
- No static schema
- Supports distributed computing
- Faster data processing
- Relatively simple data model

Disadvantage of NoSQL

- Not a defined standard
- Limited query capabilities



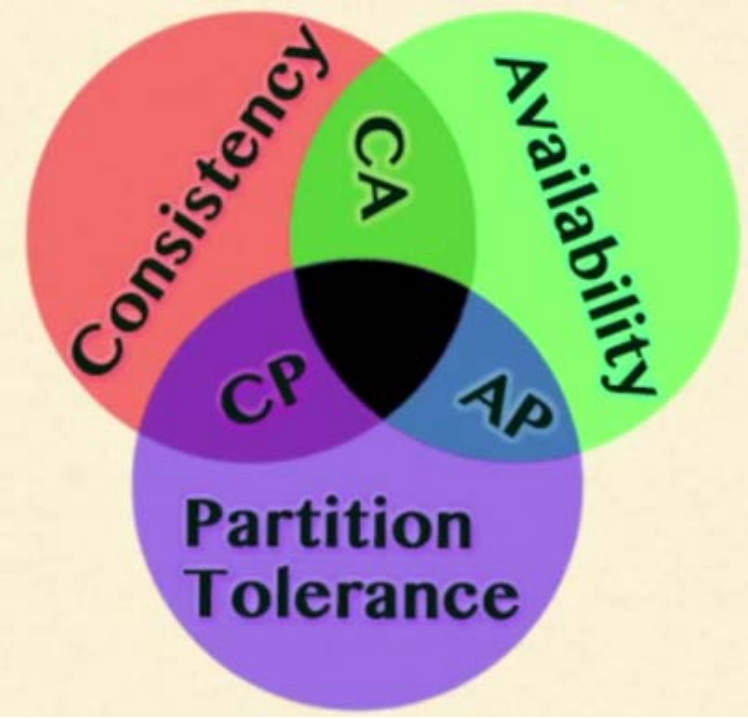
CAP / BREWER'S THEOREM



- **It is very important to understand the limitations of NoSQL database.**
- **NoSQL can not provide consistency and high availability together.**
- **This was first expressed by Eric Brewer in CAP Theorem.**
- **CAP theorem states that we can only achieve at most two out of three guarantees for a database: Consistency, Availability and Partition Tolerance.**

c. Explain CAP Theorem? how it is different from ACID Properties.

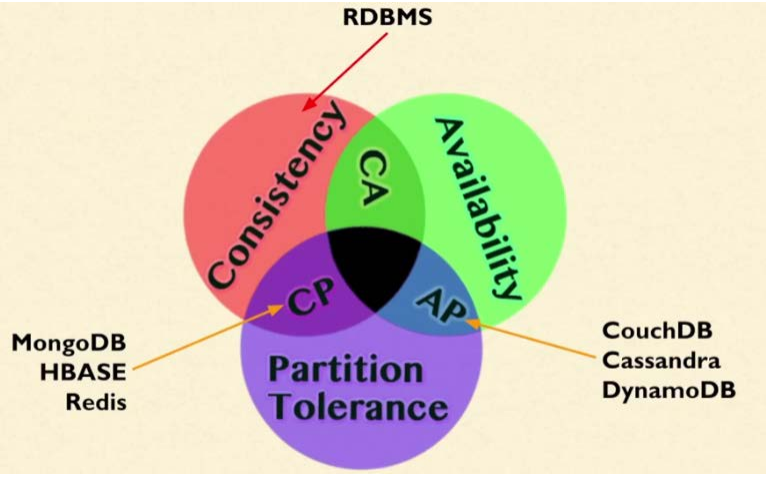
CAP / BREWER'S THEOREM



- **Consistency means that all nodes in the network see the same data at the same data.**
- **Availability is a guarantee that every request receives a response about whether it was successful or failed.**
- **Partition Tolerance is a guarantee that the system continues to operate despite arbitrary message loss or failure of part of the system.**
- **In other words, even if there is a network outage in the data center and some of the computers are unreachable, still the system continues to perform.**

c. Explain CAP Theorem? how it is different from ACID Properties.

CAP / BREWER'S THEOREM



- **Out of these three guarantees, no system can provide more than 2 guarantees.**
- **Since in the case of a distributed systems, the partitioning of the network is must, the tradeoff is always between consistency and availability.**
- **RDBMS can provide only consistency but not partition tolerance.**
- **MongoDB, HBASE and Redis can provide Consistency and Partition tolerance.**
- **CouchDB, Cassandra and Dynamo guarantee only availability but no consistency.**

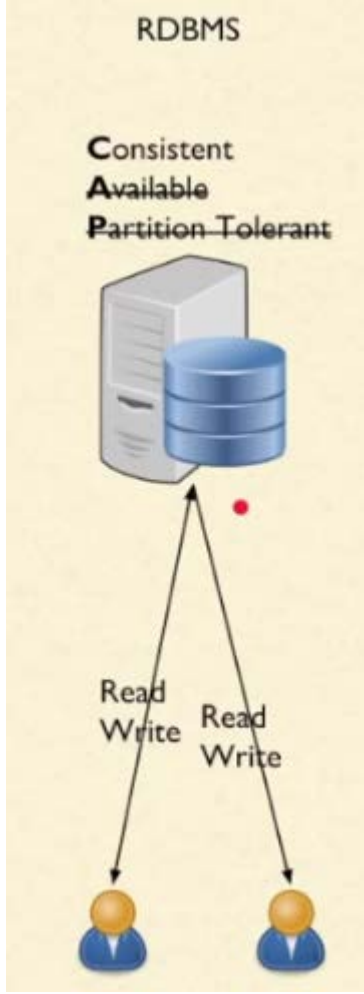
CAP / BREWER'S THEOREM

- **Let us take a look at various scenarios or architectures of systems to better understand the CAP theorem.**

- **The first one is RDBMs where Reading and writing of data happens on the same machine.**

- **Such systems are consistent but not partition tolerant because if this machine goes down, there is no backup.**

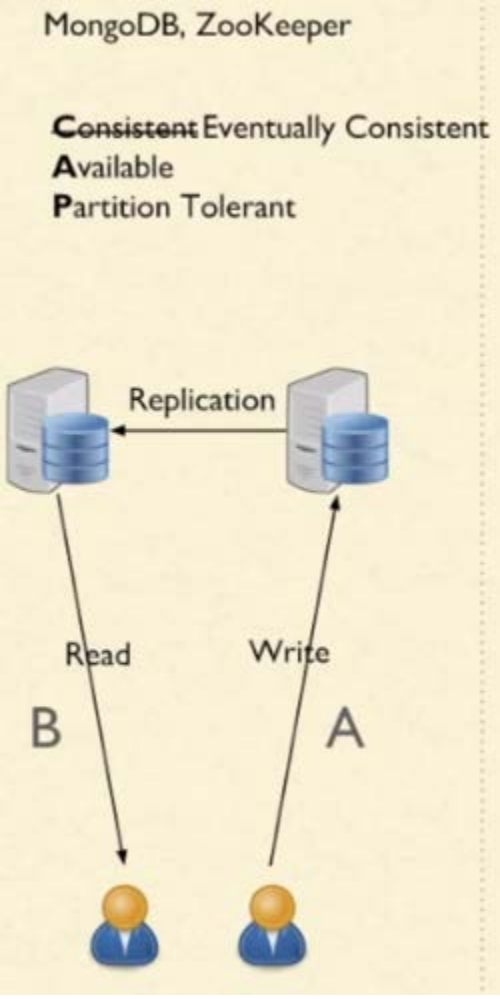
- **If one user is modifying the record, others would have to wait thus compromising the high availability.**

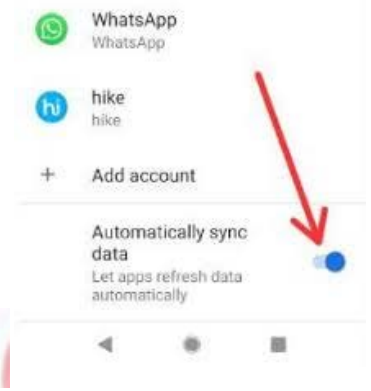


c. Explain CAP Theorem? how it is different from ACID Properties.

CAP / BREWER'S THEOREM

- **The second diagram is of a system which has two machines.**
- **Only one machine can accept modifications while the reads can be done from all machines.**
- **In such systems, the modifications flow from that one machine to the rest.**
- **Such systems are highly available as there are multiple machines to serve.**
- **Such systems are partition tolerant because if one machine goes down, there are other machines available to take up that responsibility.**





c. Explain CAP Theorem? how it is different from ACID Properties.

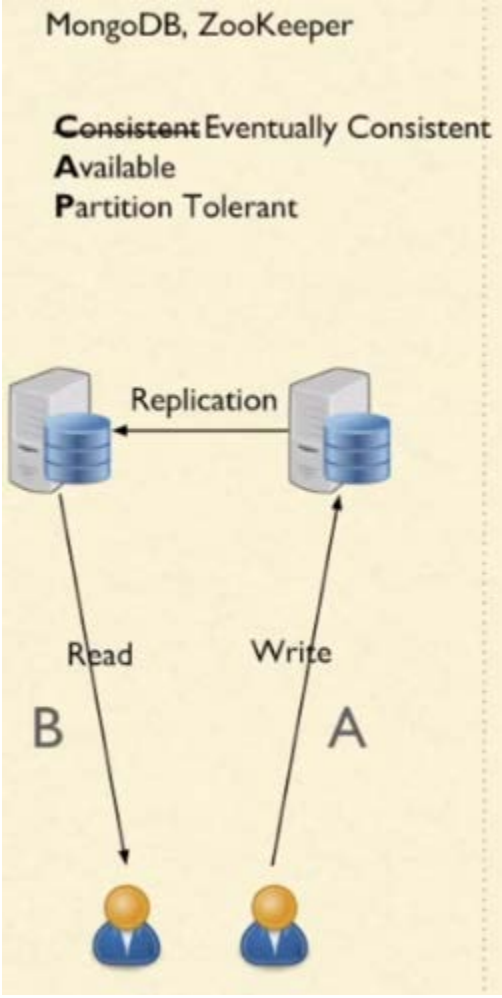
CAP / BREWER'S THEOREM

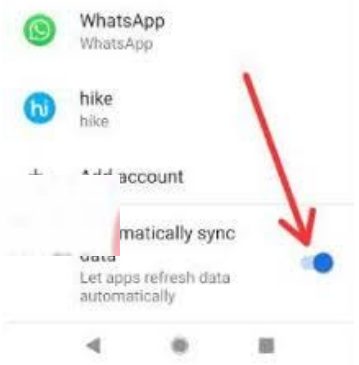
- **Since it takes time for the data to reach other machines from the node A, the other machine would be serving older data.**

- **This causes inconsistency. Though the data is eventually going to reach all machine and after a while, things are going to okay.**

- **There we call such systems eventually consistent instead of strongly consistent.**

- **This kind of architecture is found in Zookeeper and MongoDB.**





c. Explain CAP Theorem? how it is different from ACID Properties.

CAP / BREWER'S THEOREM

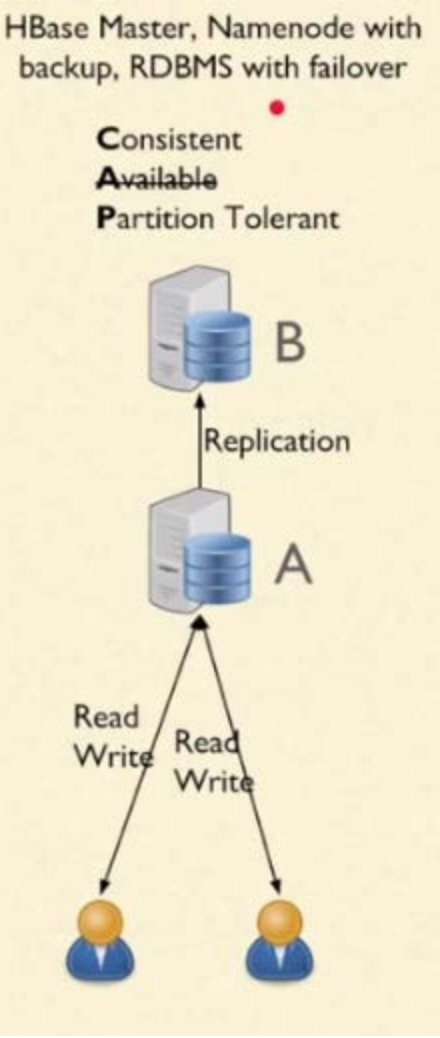
- **In the third design of any storage system, we have one machine similar to our first diagram along with its backup.**

- **Every new change or modification at A in the diagram is propagated to the backup machine B.**

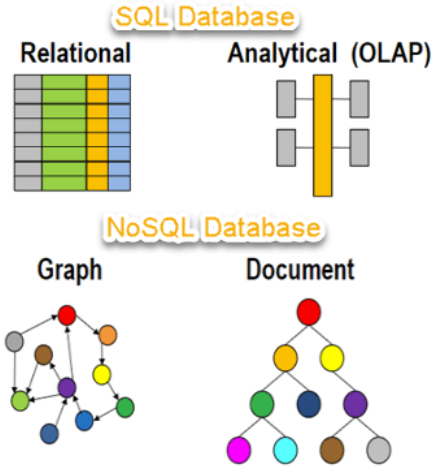
- **There is only one machine which is interacting with the readers and writers.**

- **So, It is consistent but not highly available. If A goes down, B can take A's place. Therefore this system is partition tolerant.**

- **Examples of such system we are HDFS having secondary Namenode and even relational databases having a regular backup.**



COMPARISON BETWEEN SQL AND NC



NoSQL is a non-relational DBMS

It does not require a fixed schema, avoids joins, and is easy to scale.

NoSQL database is used for distributed data stores with huge data storage needs.

NoSQL is used for Big data and real-time web apps. For example, companies like Twitter, Facebook, Google that collect terabytes of user data every single day.

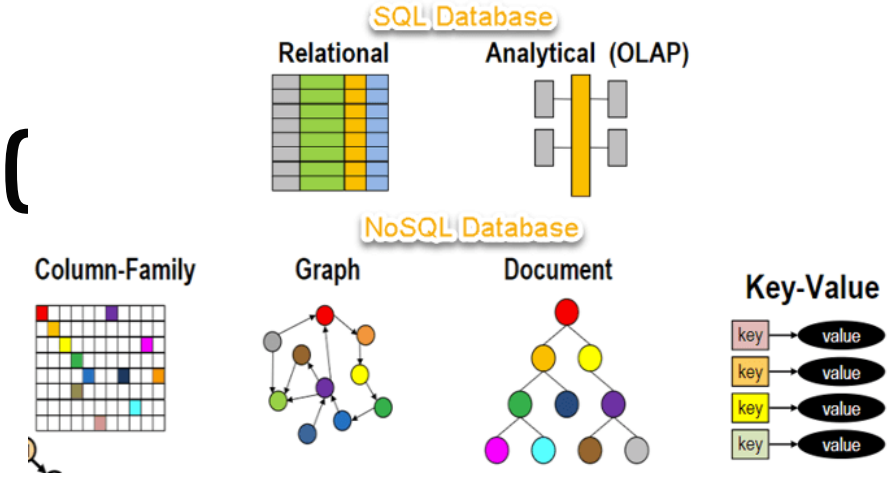
COMPARISON BETWEEN SQL AND NC

Traditional RDBMS uses SQL syntax to store and retrieve data for further insights.

NoSQL database system can store structured, semi-structured, unstructured data.

NoSQL databases became popular with Internet giants like Google, Facebook, Amazon, etc. who deal with huge volumes of data.

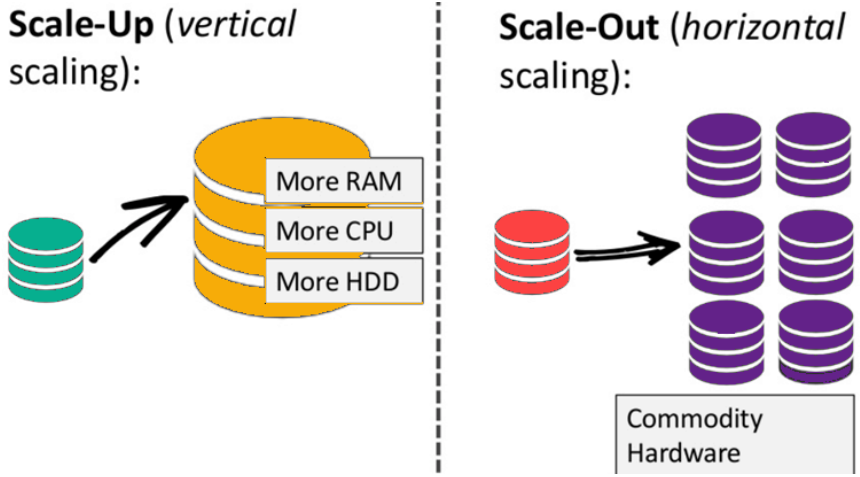
The system response time becomes slow when you use RDBMS for massive volumes of data.



COMPARISON BETWEEN SQL AND NO

To resolve this problem, we could "scale up" our systems by upgrading our existing hardware. This process is expensive.

The alternative for this issue is to distribute database load on multiple hosts whenever the load increases. This method is known as "scaling out.“



COMPARISON BETWEEN SQL AND NOSQL

May 2015

SQL	NoSQL
Full form Structured query language	Full form Not only SQL
Relational database	Non relational database
SQL is declarative query language	Non declarative query language
SQL database works on ACID properties Atomicity Consistency Isolation Durability	NoSQL follow Cap theorem Consistency Availability Partition Tolerace
Structured and organized data	Unstructured and replicable data
Relational database tables are used	Key-value pair storage, Column Store, Document Store, Graph Database
Tightly consistent	Eventually consistent
MySQL, Oracle, MS SQL, PostgreSQL, SQLite, DB2	Mongo DB, Big Table, Neo4j, Couch DB, Cassandra, HBase

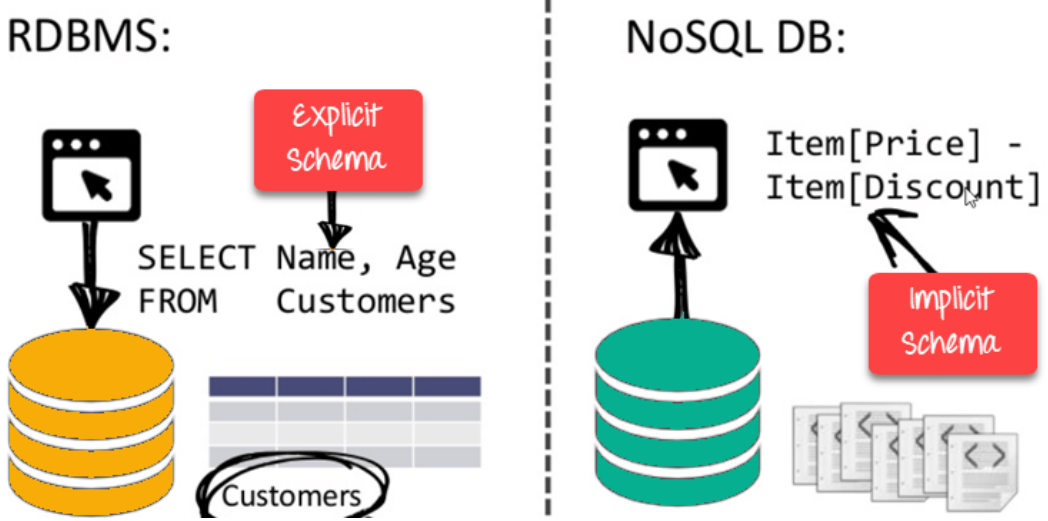
NOSQL DATABASES TIME LINE

- **1998- Carlo Strozzi use the term NoSQL for his lightweight, open-source relational database**
- **2000- Graph database Neo4j is launched**
- **2004- Google BigTable is launched**
- **2005- CouchDB is launched**
- **2007- The research paper on Amazon Dynamo is released**
- **2008- Facebooks open sources the Cassandra project**
- **2009- The term NoSQL was reintroduced**

FEATURES OF NOSQL

Non Relational

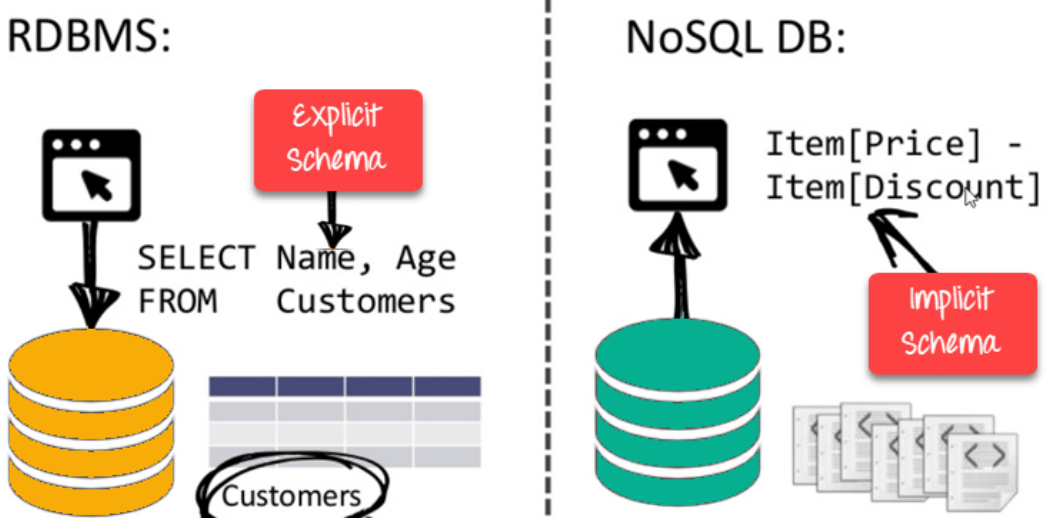
- **NoSQL databases never follow the relational model**
- **Never provide tables with flat fixed-column records**
- **Work with self-contained aggregates or BLOBs (Binary large object)**
- **Doesn't require object-relational mapping and data normalization**
- **No complex features like query languages, query planners, referential integrity joins, ACID**



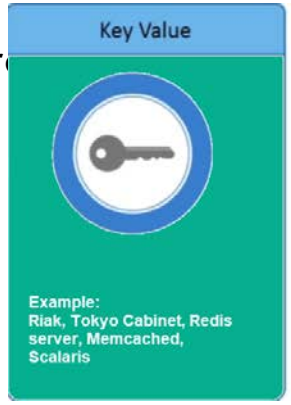
FEATURES OF NOSQL

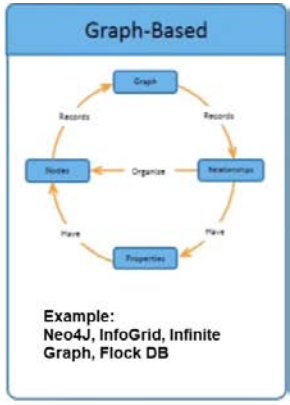
Schema-Free

- **NoSQL databases are either schema-free or have relaxed schemas**
- **Do not require any sort of definition of the schema of the data**
- **Offers heterogeneous structures of data in the same domain**



FEATURES OF NOSQL





There are mainly four categories of NoSQL databases.

Each of these categories has its unique attributes and limitations.

No specific database is better to solve all problems.

You should select a database based on your product needs.

DIFFERENT ARCHITECTURAL PATTERS ON NOSQL

1. Key Value Pair Based

This is very simple NoSQL database

Data is stored in key/value pairs.

It is designed to store as schema free data.

Such data is stored along with indexed key.

Example: Cassandra, DynamoDB, Azure Table Storage

Key	Value
Name	Joe Bloggs
Age	42
Occupation	Stunt Double
Height	175cm
Weight	77kg

Redis 

Riak 

Memcached 

Project Voldemort 

Couchbase 

What are the different data architecture patterns in NOSQL? Explain "key value" store and "Document" store patterns with relevant examples. **(10)**

IT DEC 2016

DIFFERENT ARCHITECTURAL PATTERS ON NOSQL

Use Case:

This type is generally used when you need quick performance for basic Create – Read – Update – Delate operations.

Example:

Storing and retrieving session information for Web pages
Storing user profile and preferences
Storing shopping cart data for ecommerce

Key	Value
Name	Joe Bloggs
Age	42
Occupation	Stunt Double
Height	175cm
Weight	77kg

Redis 

Riak 

Memcached 

Project Voldemort 

Couchbase 

What are the different data architecture patterns in NOSQL? Explain "key value" store and "Document" store patterns with relevant examples. **(10)**

DIFFERENT ARCHITECTURAL PATTERS ON NOSQL

Limitations:

It may not work for complex queries attempting to connect multiple relations

If data contain many to many relationship a key value pair is likely to show poor performance

Key	Value
Name	Joe Bloggs
Age	42
Occupation	Stunt Double
Height	175cm
Weight	77kg

Redis

Riak

Memcached

Project Voldemort

Couchbase

What are the different data architecture patterns in NOSQL? Explain "key value" store and "Document" store patterns with relevant examples. **(10)**

DIFFERENT ARCHITECTURAL PATTERS ON NOSQL

Column-based Database

Column-oriented databases work on columns and are based on BigTable paper by Google.

They deliver high performance on aggregation queries like SUM, COUNT, AVG, MIN etc. as the data is readily available in a column.

Instead of storing data in relational tuples (rows), it is stored in cells grouped in columns

ColumnFamily			
Row Key	Column Name		
	Key	Key	Key
	Value	Value	Value
	Column Name		
	Key	Key	Key
	Value	Value	Value

What are the different data architecture patterns in NOSQL? Explain "key value" store and "Document" store patterns with relevant examples. **(10)**

DIFFERENT ARCHITECTURAL PATTE

Example:
HBase, Hyper Table, Big Table.

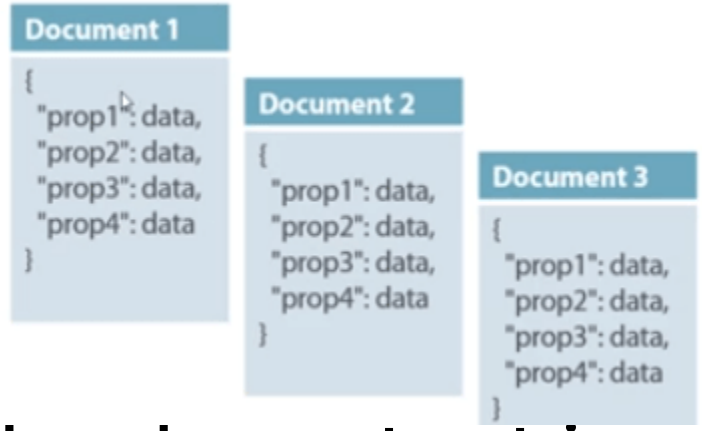
Use Cases:
It is used for storing blogs

ColumnFamily			
Row Key	Column Name		
	Key	Key	Key
	Value	Value	Value
	Column Name		
	Key	Key	Key
	Value	Value	Value

What are the different data architecture patterns in NOSQL? Explain "key value" store and "Document" store patterns with relevant examples. **(10)**

DIFFERENT ARCHITECT

Col1	Col2	Col3	Col4
Data	Data	Data	Data
Data	Data	Data	Data
Data	Data	Data	Data



Document-based database:

Document-Oriented NoSQL DB works on key value storage where document contain lot of complex data.

The document is stored in JSON or XML formats. The value is understood by the DB and can be queried.

JSON (JavaScript Object Notation) is a lightweight data-interchange format.

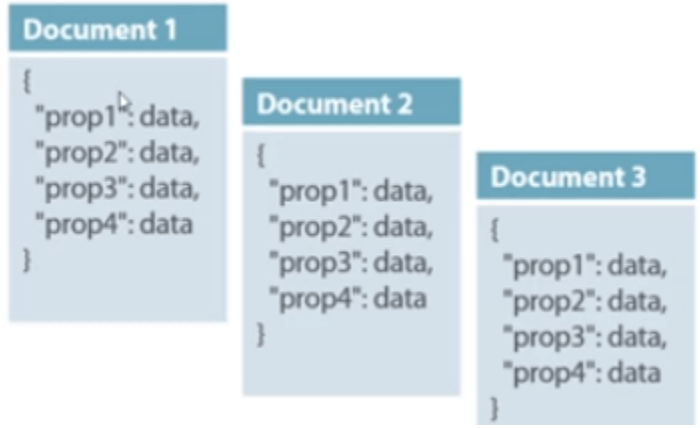
Every document contains a unique key to retrieve the document

Key is used for storing, retrieving and managing document – oriented information

What are the different data architecture patterns in NOSQL? Explain "key value" store and "Document" store patterns with relevant examples. **(10)**

DIFFERENT ARCHITECT

Col1	Col2	Col3	Col4
Data	Data	Data	Data
Data	Data	Data	Data
Data	Data	Data	Data




```
Example: (MongoDB) document
{Name:"SimpliLearn",
Address:" 10685 Hazelhurst Dr, Houston, TX 77043, United States",
Courses: ["Big Data","Python","Android","PMP","ITIL"],
Offices: [ "NYK","Dubai","BLR"],
}
```

Eg: Amazon SimpleDB, CouchDB, MongoDB, Lotus Notes are popular Document originated DBMS systems.

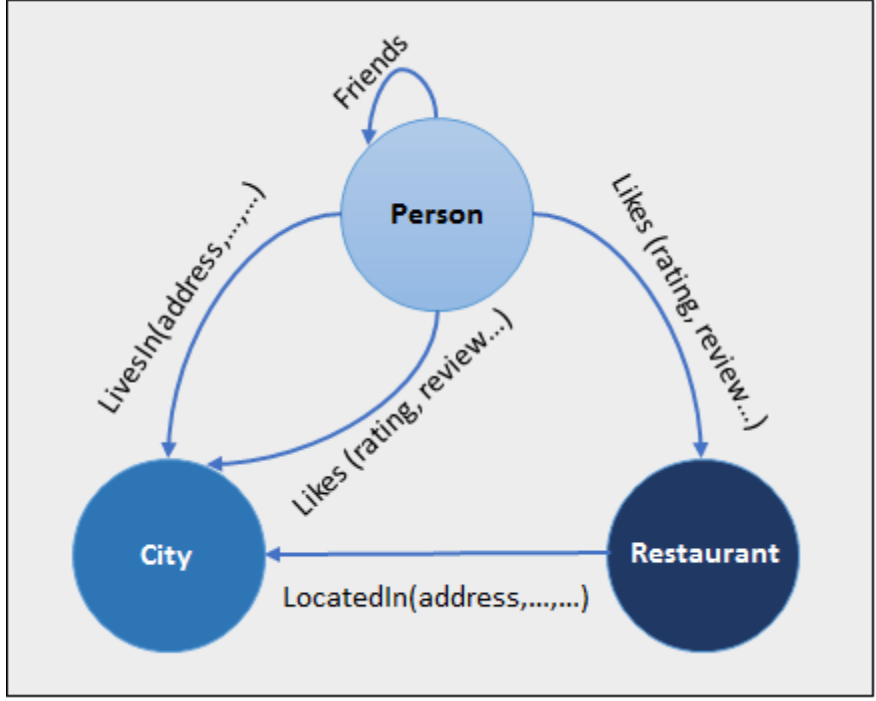
Use Cases:

Used for storing Event logging information and online blogging
All document would contain information about type of document, userid, post content, timestamp etc.

Limitation:
It may be good for blogging but not good for aggregation.

DIFFERENT ARCHITECTURAL PATTERS ON NOSQL

Graph-Based



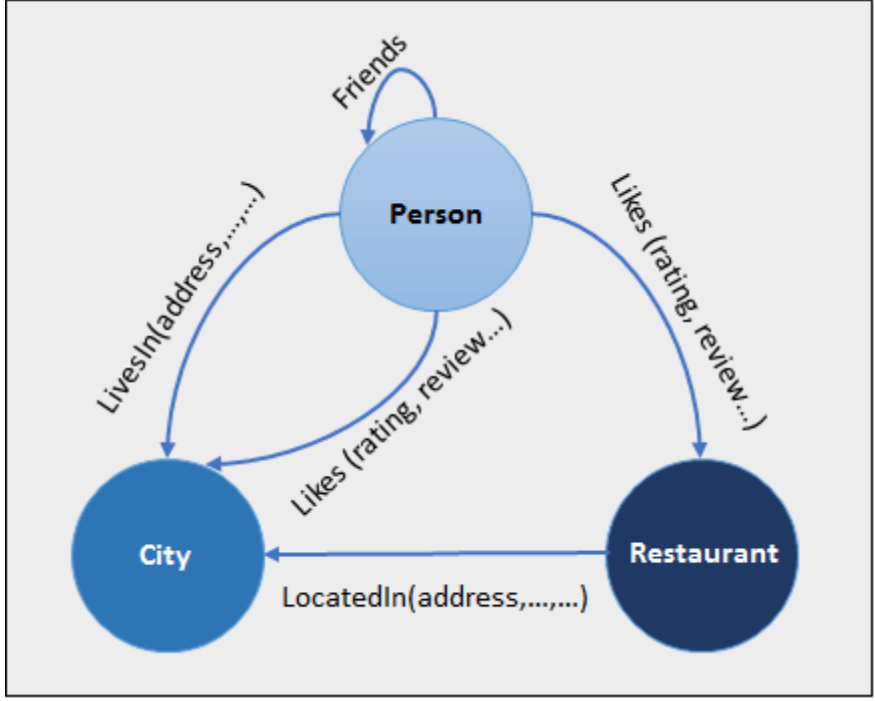
A graph type database stores entities as well the relations amongst those entities.

The entity is stored as a node with the relationship as edges.

An edge gives a relationship between nodes. Every node and edge has a unique identifier.

Compared to a relational database where tables are loosely connected, a Graph database is a multi-relational in nature.

DIFFERENT ARCHITECTURAL PATTERS ON NOSQL



Example:

Neo4J, Infinite Graph, OrientDB, FlockDB are some popular graph-based databases.

Use cases:

Very important application is social networking site, it can quickly locate friend, friends of friends

Google maps useful for navigarion and finding the closest location

What are the different data architecture patterns in NOSQL? Explain "key value" store and "Document" store patterns with relevant examples. **(10)**

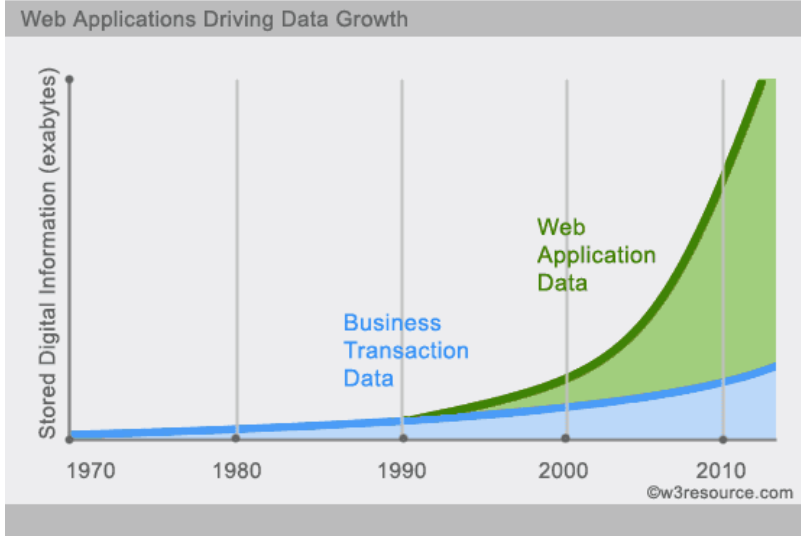
DIFFERENT ARCHITECTURAL PATTERS ON NOSQL

Database Model	Performance	Scalability	Flexibility
Key value store database	High	High	High
Column store database	High	High	Moderate
Document store database	High	Variable	High
Graph database	Variable	Variable	High

What are the different data architecture patterns in NOSQL? Explain "key value" store and "Document" store patterns with relevant examples. (10)

ADVANTAGES OF NOSQL

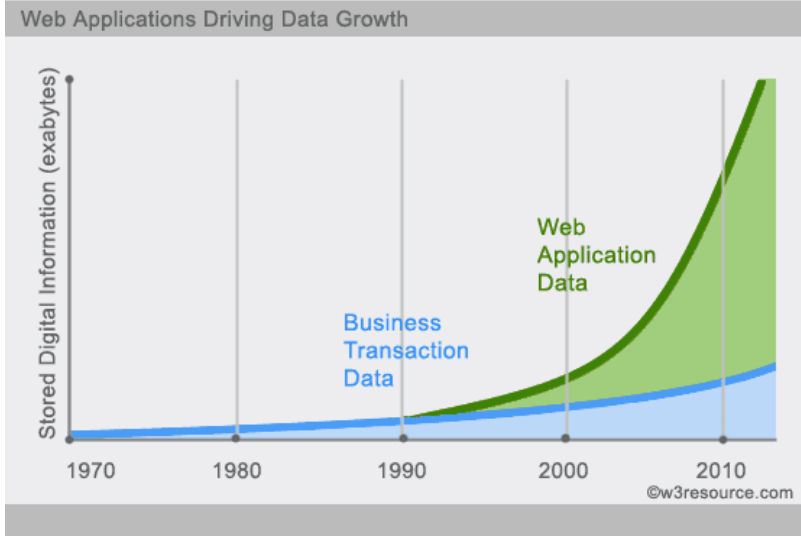
1. **Growth of Big Data**
    1. **Big data is one of the main driving factor of NoSQL for business**
    2. **Web data has increased exponentially within last two years**
2. **Continuous Availability of data**
    1. **Hardware failure are possible but NoSQL is built on distributed architecture which is robust.**
    2. **If data node goes down we have replication factor, if name node goes down we have secondary name node**
3. **Location Independent**
    1. **It is ability to read and write the database from anywhere**



Exabyte (18): Blankets West coast
Or 1/4th of India

ADVANTAGES OF NOSQL

1. **Flexible data models**
    1. **NoSQL has more flexible data models as compared to others which is schema less**
2. **Better Architecture**
    1. **NoSQL has more business oriented architecture for a particular application**
    2. **Organizations migrate their data to NoSQL platform which allows them to maintain very volume of data**
3. **Analytics and Business Intelligence**
    1. **Extracting meaningful information from vey high volume of data is very difficult task for RDBMS**
    2. **Modern NoSQL provides integrated data analysis and better understanding of complex data sets which facilitate flexible decision-making.**



Exabyte (18): Blankets West coast
Or 1/4th of India

IMPLEMENTATION OF KEY VALUE DATABASE

This is very simple NoSQL
It is designed for storing data as schema free
In this data is stored in the form of indexed key

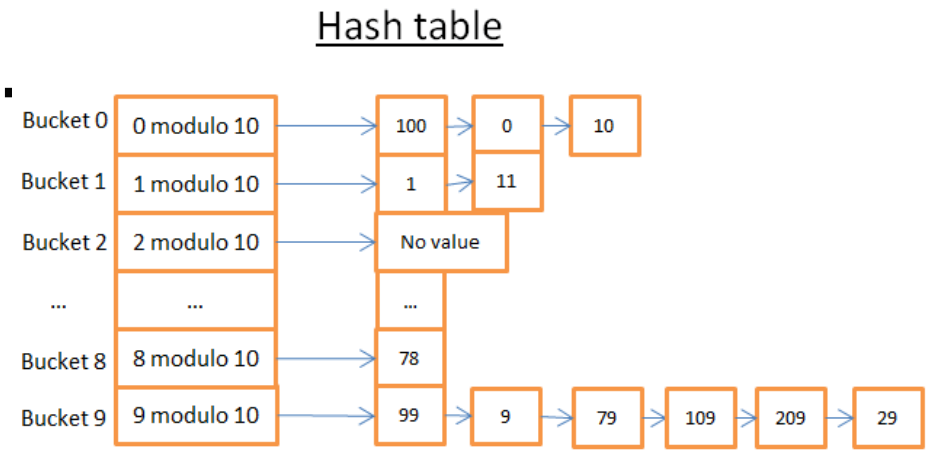
Key: 1	ID: 123	First Name: Ganesh

Key: 2	Email: abc@gmail.com	Location: Mumbai	Pin: 401209

Key: 3	Facebook ID: xyz	Password: *******	Name: Tom

Working
The schema less format of a key value database is required for data storage needs.
The key can be auto generated while the value can be string

IMPLEMENTATION OF KEY VALUE DA



Key value uses a hash table in which there exists a Unique key and pointer to each data item

The logical group of keys is known as bucket

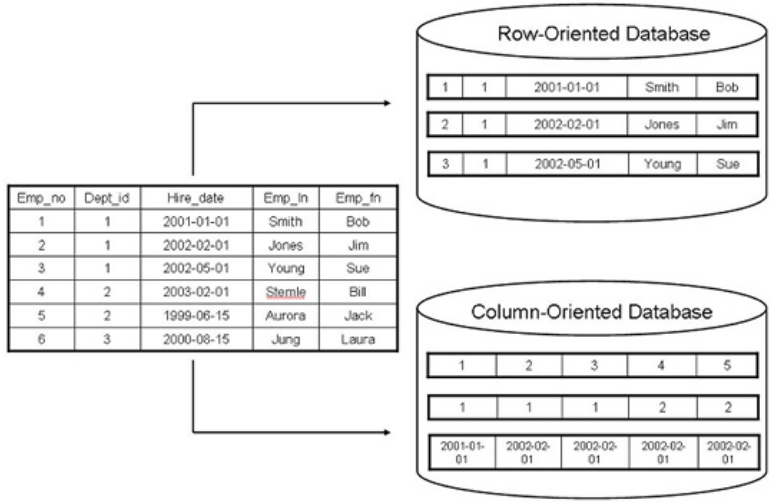
It will improve the performance because of cache mechanism

Read Write values

- **Get(key): It will return the value associated with the key**
- **Multi-get(key1, key2, …, keyN): It will return the list of values associated with the key**
- **Put(key, value): It will associate the value with the key**
- **Delete(key): it will delete entry for the key from the data store**

Alex Homes, Hadoop in Practice 2nd Edition Page 28

COLUMN STORE DATA



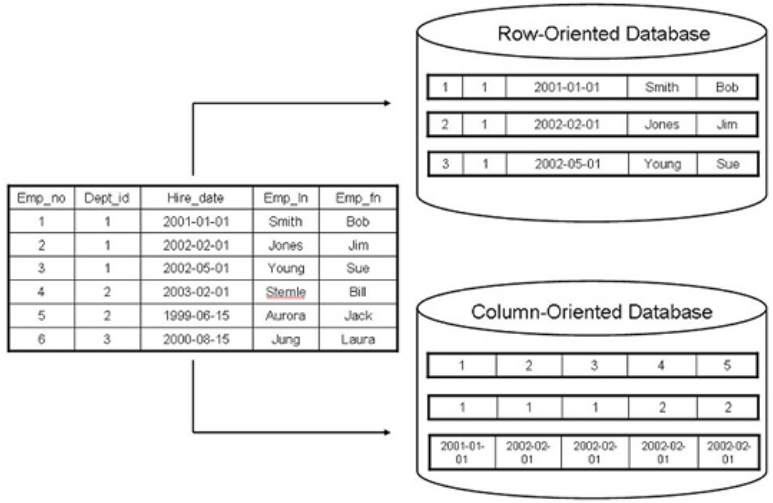
Instead of storing the data in in rows, it stored in cells grouped in columns
It offers high performance and high scalability

Working:

- **In column-oriented NoSQL database, data is stored in cells grouped in columns rather than rows**
- **Read and write is done using columns**
- **It offers fast search and access of data & Aggregation.**

Alex Homes, Hadoop in Practice 2nd Edition Page 28

COLUMN STORE DATA



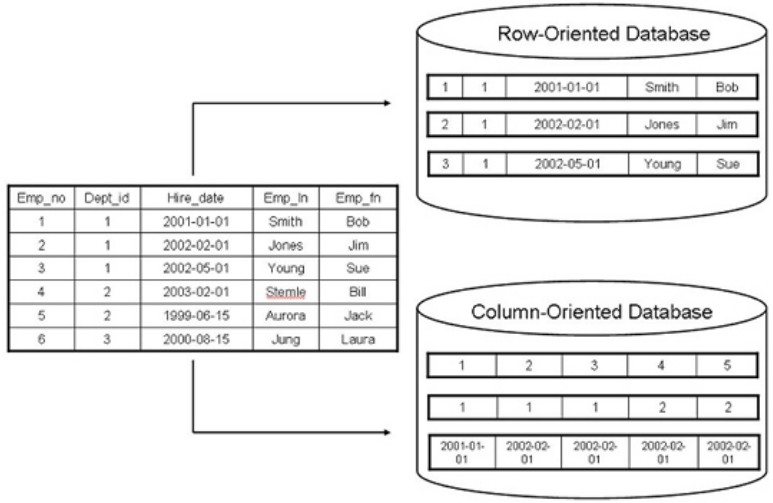
Data Model:

- **Column Family: Single structure that can group Columns**

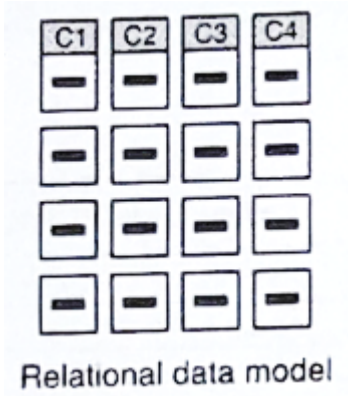
- **Example:**
  - **Hbase, BigTable, Hyper Table**

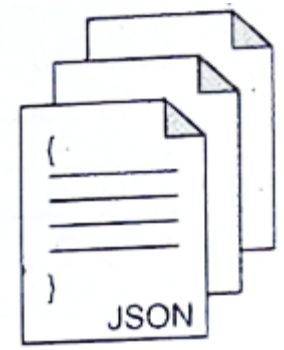
Alex Homes, Hadoop in Practice 2nd Edition Page 28

COLUMN STORE DATA



Emp_no	Dept_id	Hire_date	Emp_ln	Emp_fn
1	1	2001-01-01	Smith	Bob
2	1	2002-02-01	Jones	Jim
3	1	2002-05-01	Young	Sue
4	2	2003-02-01	Stemle	Bill
5	2	1999-06-15	Aurora	Jack
6	3	2000-08-15	Jung	Laura





Document Based:

- **It is based on the concept of key value store where "documents" contain a lot of complex data.**
- **Every document contain a unique key used to retrieve a document**
- **Key is used for managing, storing and retrieving document oriented information**

Working:

This type of data is collection of key-value pair where value is a compressed document

JSON and XML are commonly used documents

Eg: MongoDB , CoutchDB

Alex Homes, Hadoop in Practice 2nd Edition Page 28

COLUMN STORE DATA

Graph Database:

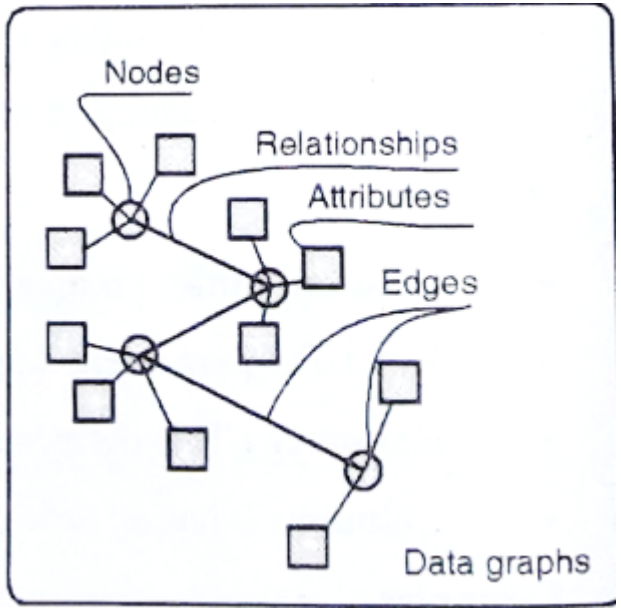
- **Data is stored l graph and their relationship are stored as a link where as entity act as a node.**

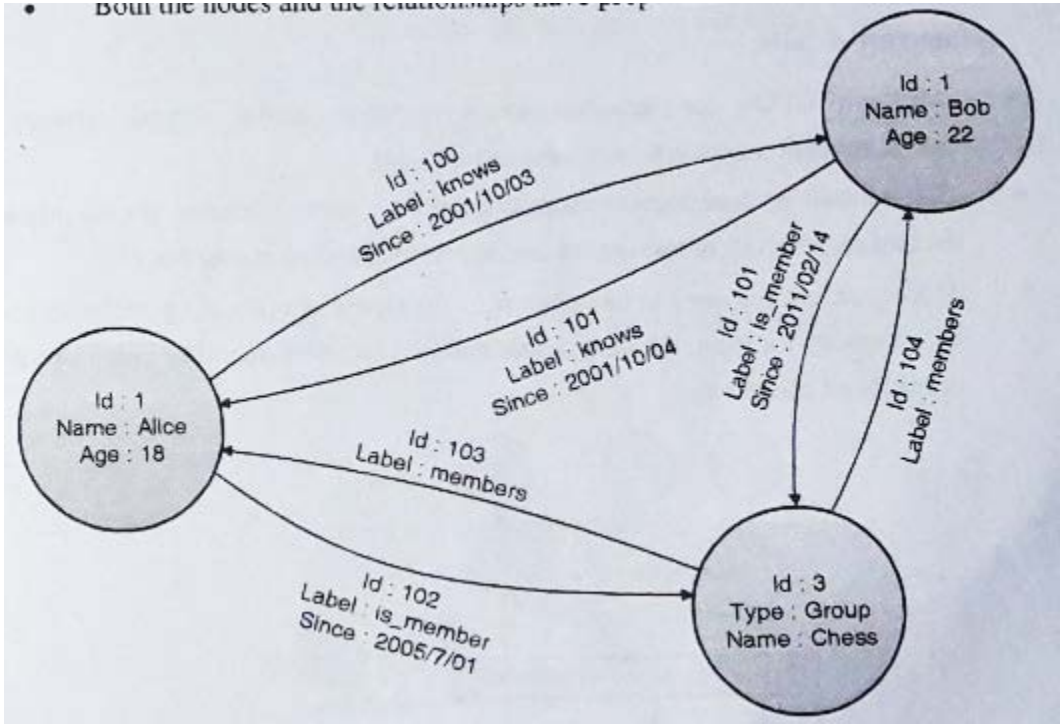
Working:

In this a flexible graphical notation is used with edges and nodes

Data can be easily transformed from one model to another model by using graph based NoSQL database

Eg: Neo4j, Polyglot





SQL CASE STUDIES

AmazonDB

It has the largest ecommerce operations in the world
Customers across the globe shop 24*7
Initially Amazon used RDBMS systems for shopping and checkout system
Amazon DynamoDB a NoSQL brought a turning point
DynamoDB addresses the core problem of RDBMS scalability and partition tolerance
Developers can store unlimited amount of data by creating a database table
DynamoDB saves the table in multiple servers
DynamoDB is a Key-Value store NoSQL

Salient features of key-value store are as follows:

Scalable: If the application requirement changes, AWS management console can scale up or scale down the services.

SQL CASE STUDIES

Automated storage scaling: More storage can be obtained when ever more storage is required.

Built-in fault tolerance: DynamoDB automatically replicates data across various nodes

Flexible: DynamoDB has a schema free format. Multiple data types can be used

Efficient Indexing: Every item is defined by a primary key. It allows secondary indexes on non key attributes.

Strong Consistency: DynamoDb ensures strong consistency on reads (reads only the latest value)

Secure: DynamoDB used cryptography to authenticate users

SQL CASE STUDIES

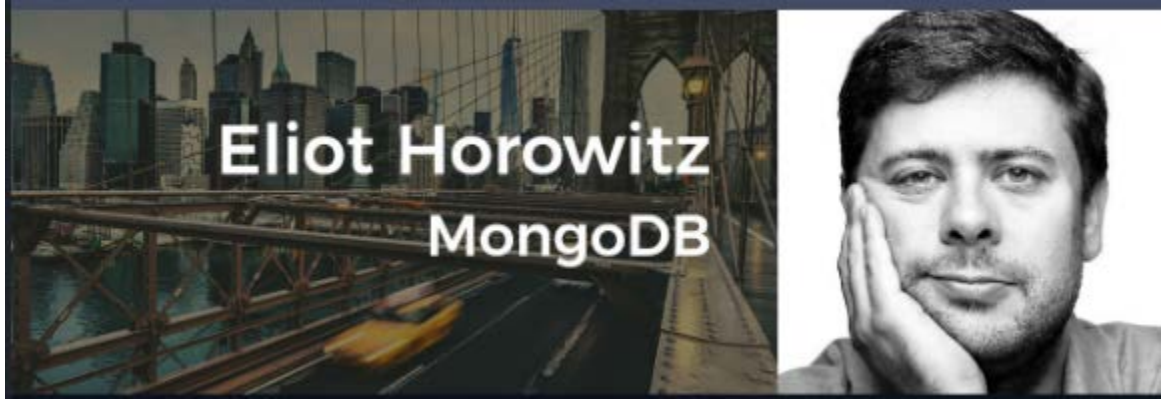
Google Big Table:

Motivation for developing BigTable is to achieve massive scalability, better performance and ability to run commodity hardware.

The volume of Google data is generally in petabytes and is distributed over 1,00,000 nodes

Big table is column based NoSQL.

SQL CASE STUDIES



MongoDB

MongoDB was designed by Eliot Horowitz
Mongo DB was designed for building large scale, high availability, robust systems
MongoDB changed the transformed the relational data to document based data to manage speed, agility, schema less databases
MongoDB is a document data model that stores data in JSON document

DATATYPES USED IN MONGO DB

Data Types	Description
String	String is the most commonly used datatype. It is used to store data. A string must be UTF 8 valid in mongodb.
Integer	Integer is used to store the numeric value. It can be 32 bit or 64 bit depending on the server you are using.
Boolean	This datatype is used to store boolean values. It just shows YES/NO values.
Double	Double datatype stores floating point values.
Min/Max Keys	This datatype compare a value against the lowest and highest bson elements.
Arrays	This datatype is used to store a list or multiple values into a single key.
Object	Object datatype is used for embedded documents.
Null	It is used to store null values.
Symbol	It is generally used for languages that use a specific type.
Date	This datatype stores the current date or time in unix time format. It makes you possible to specify your own date time by creating object of date and pass the value of date, month, year into it.

SQL CASE STUDIES

Neo4j

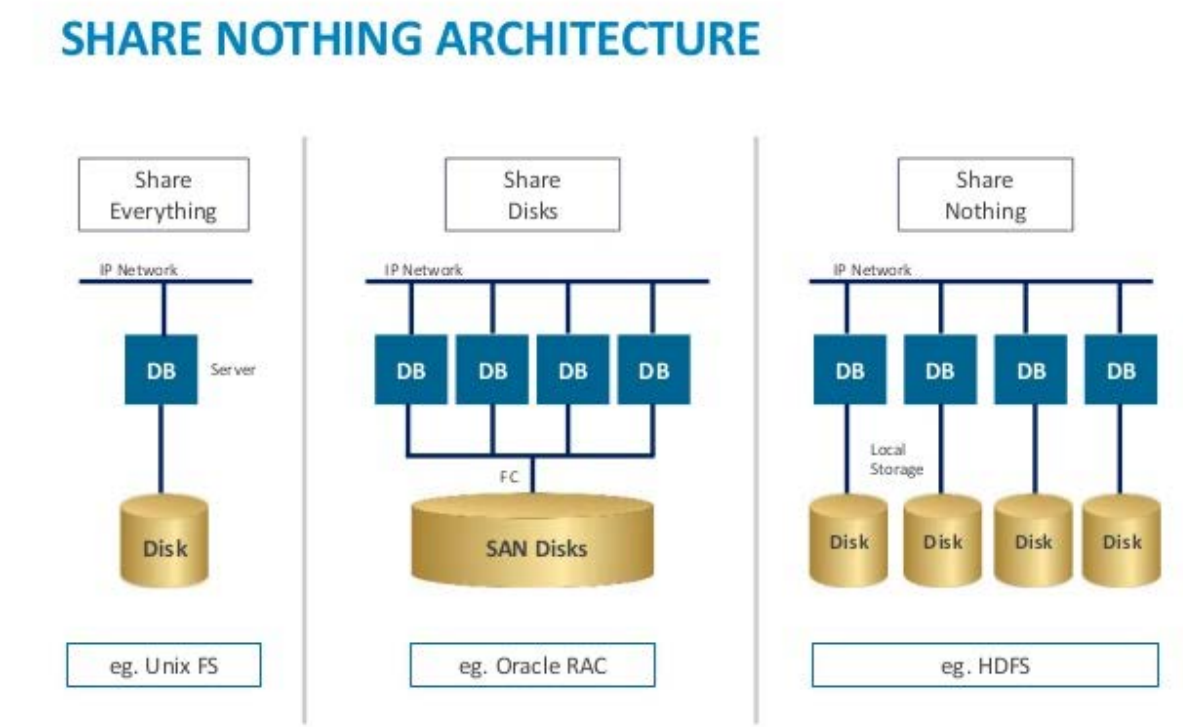
Neo4j is open source sponsored by Neo Technologies
It is graph based NoSQL which is implemented in Java and Scala
Its development was started in 2003 and was made public in 2007
Neo4j is used by many organizations for scientific research, routing, matchmaking, network management, recommendations, social networks, software analytics and project management

SHARED NOTHING" ARCHITECTURE

A Shared Nothing Architecture is one in which you have a number of nodes.

These nodes do not share resources like memory or storage with any one.

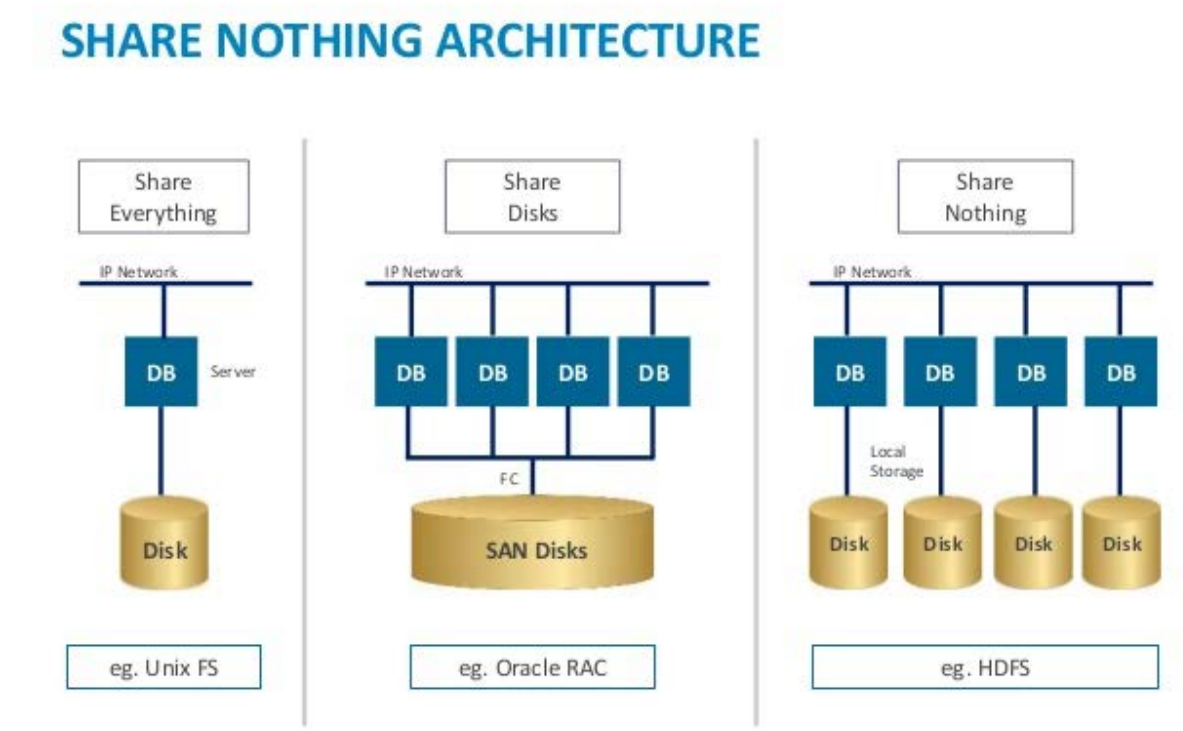
On the other hand One Alternative Architecture shares every resource when requested.



SHARED NOTHING" ARCHITECTURE

Advantages of shared nothing architecture:

- **easier scaling**
- **non-disruptive upgrades**
- **elimination of a single point of failure self-healing capabilities.**

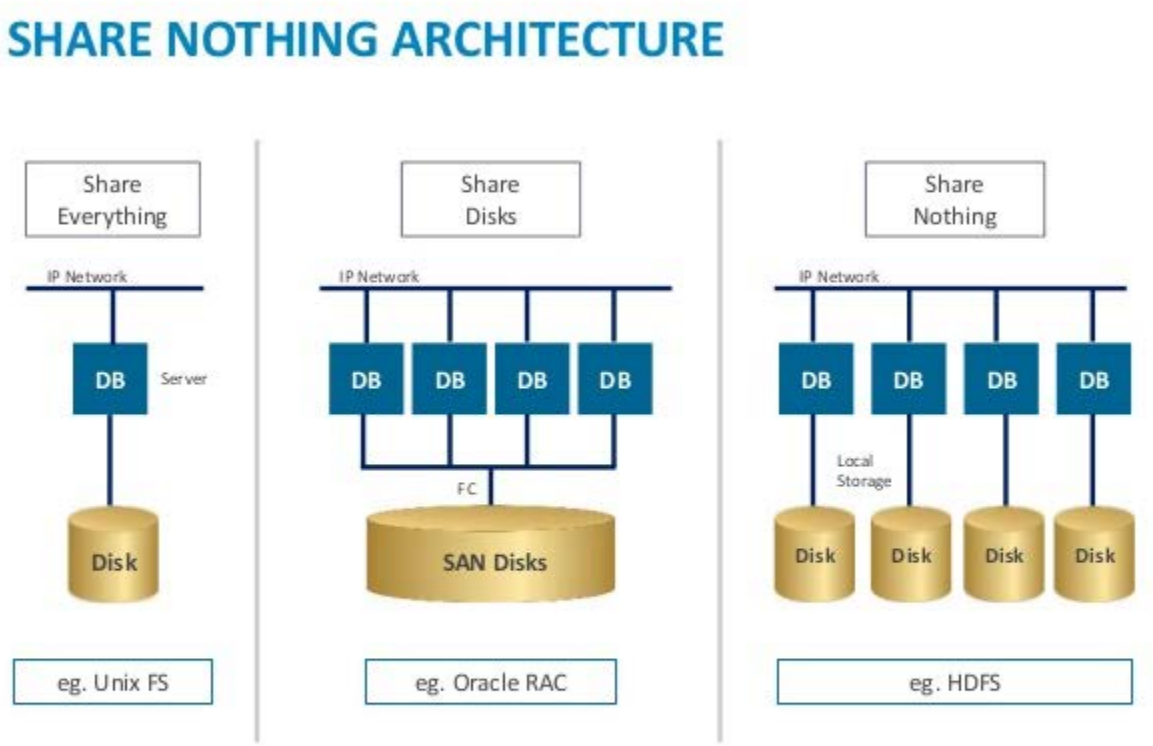


SHARED NOTHING" ARCHIT

Scaling becomes simpler when things such as disks are not shared.

For example, scaling up a single shared disk to get more storage space can lead to enormous problems if things do not go well.

On the other hand, if you are using several nodes that do not share the space, scaling up the disk space becomes quite a bit easier.

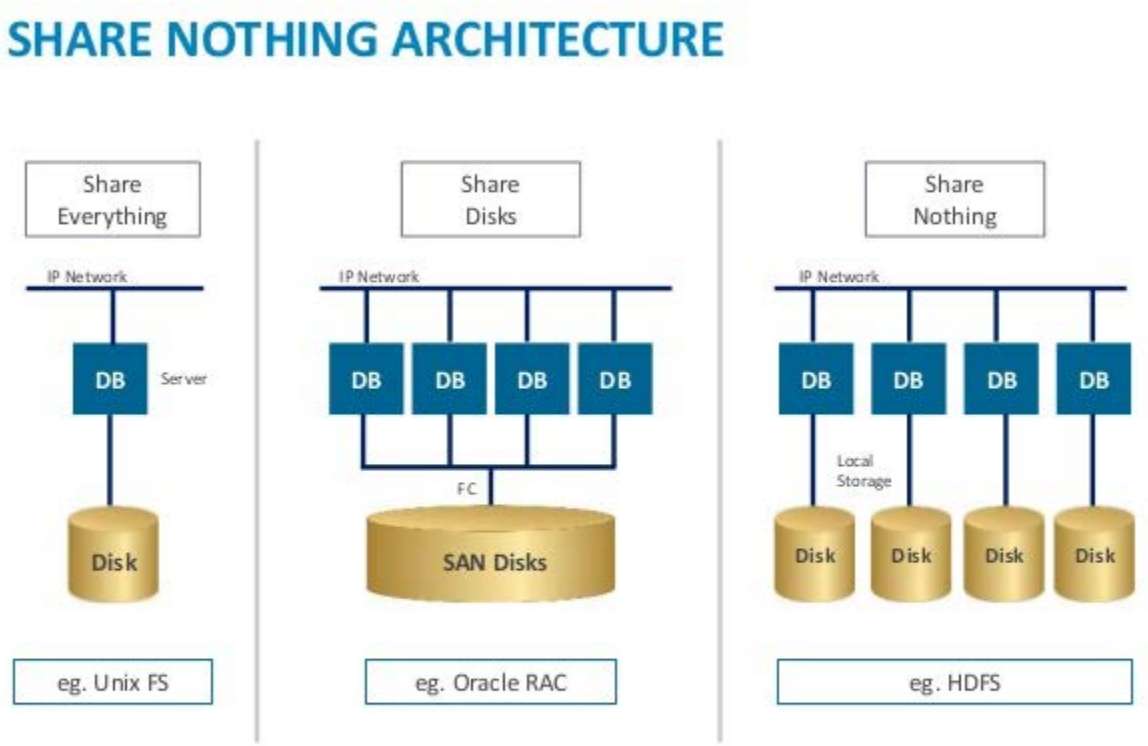


SHARED NOTHING" ARCHIT

If the scaling should fail on one of the resources, the others will still continue to do their work normally.

"This architecture is followed by essentially all high-performance, scalable, DBMSs, including Teradata, Netezza, Greenplum, as well as several Morpheus integrations.

It is also used by most of the high-end e-commerce platforms, including Amazon, Akamai, Yahoo, Google, and Facebook."



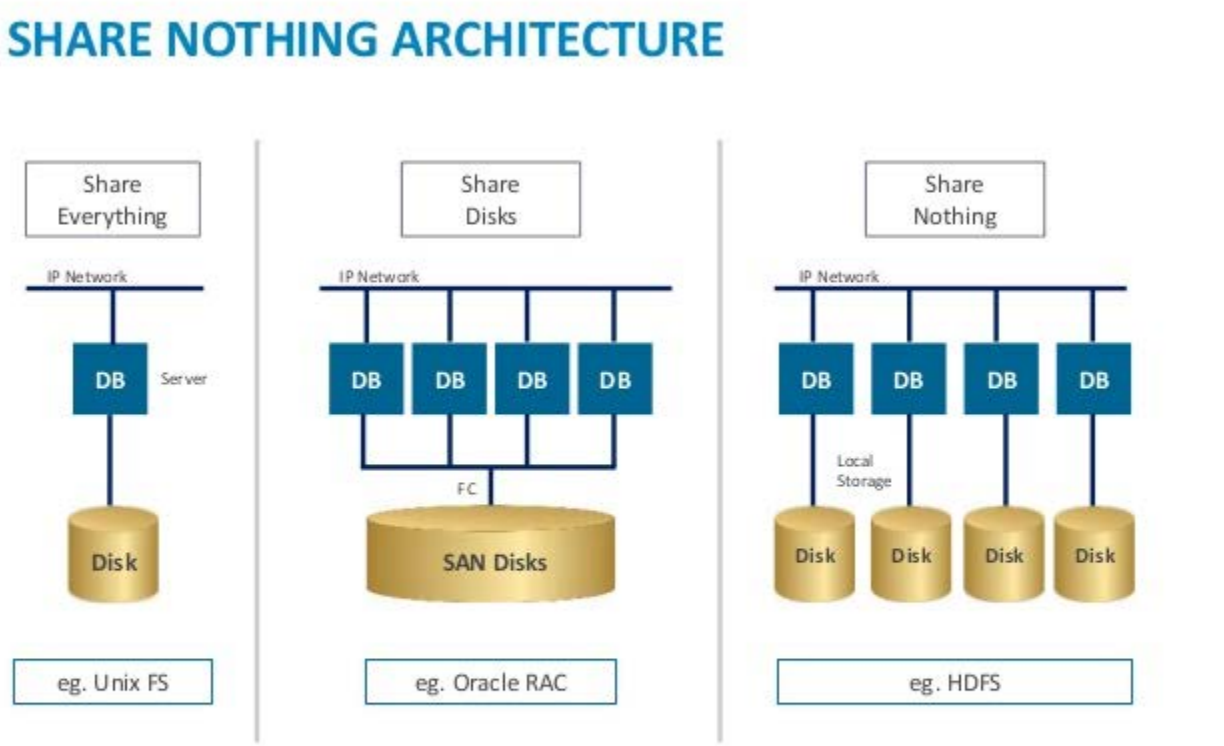
SHARED NOTHING" ARCHIT

Enables Non-disruptive Upgrades

Similar to the scaling advantages, you can use shared nothing architecture to perform non-disruptive upgrades to your services.

Instead of having a certain amount of downtime while you are upgrading an infrastructure with shared resources, you can upgrade a node at a time.

The redundancy in the other nodes will continue to run so that you do not need to shut everything down for the amount of time it takes to perform the upgrade.

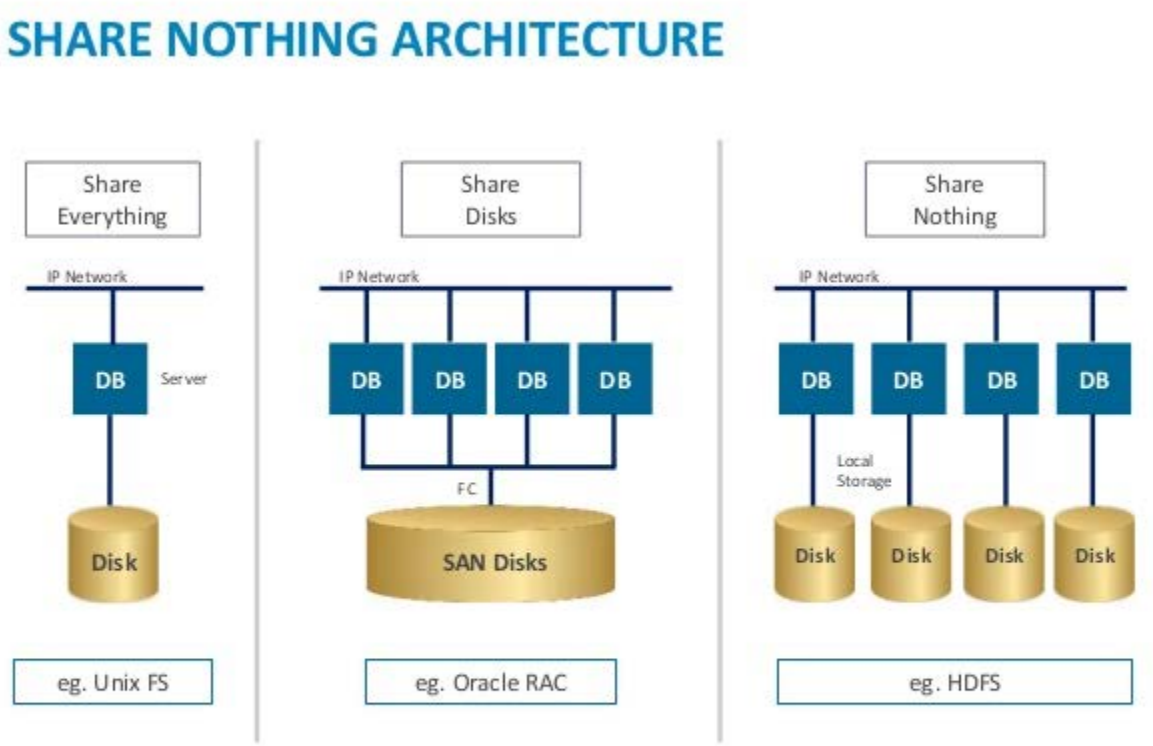


SHARED NOTHING" ARCHIT

Eliminates Single Point of Failure

With shared systems, a single point of failure can take down your site or app entirely.

As noted, the ability to have separate systems on separate nodes with redundancy can make things much easier while avoiding the disaster of a single failure causing unexpected downtime.

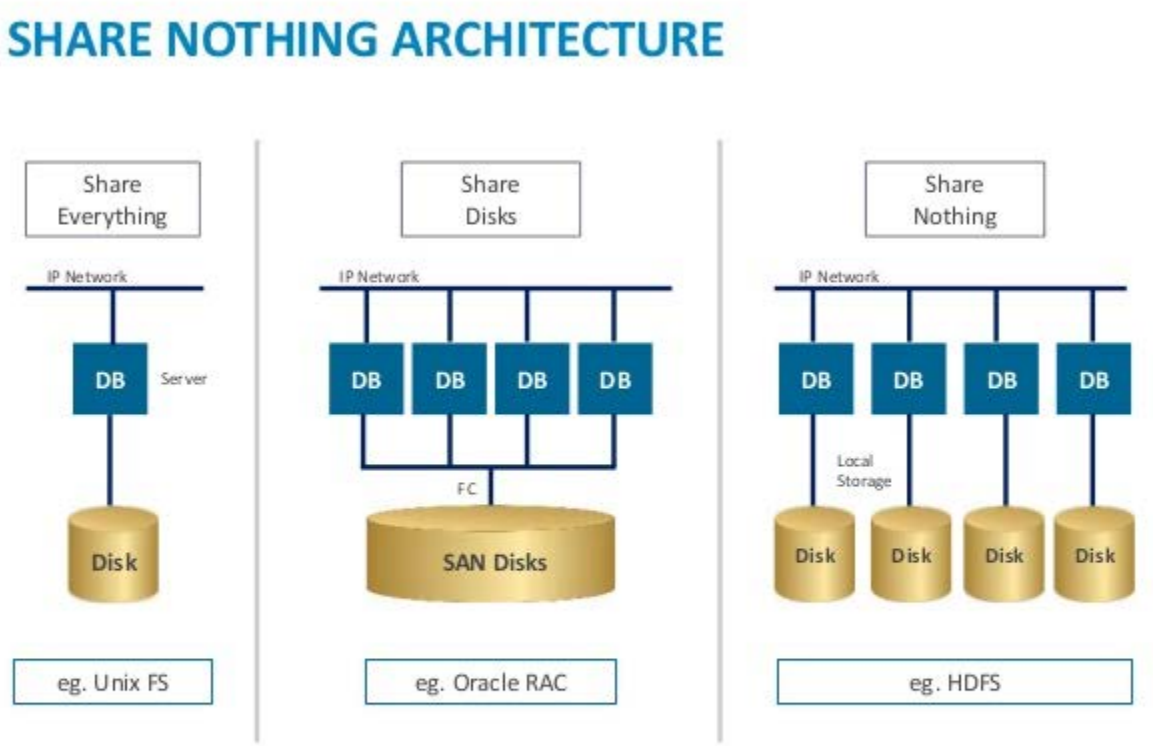


SHARED NOTHING" ARCHIT

Avoids Unexpected Downtime

Shared Nothing architecture allows for some amount of self-healing that can be another line of defense against unexpected downtime.

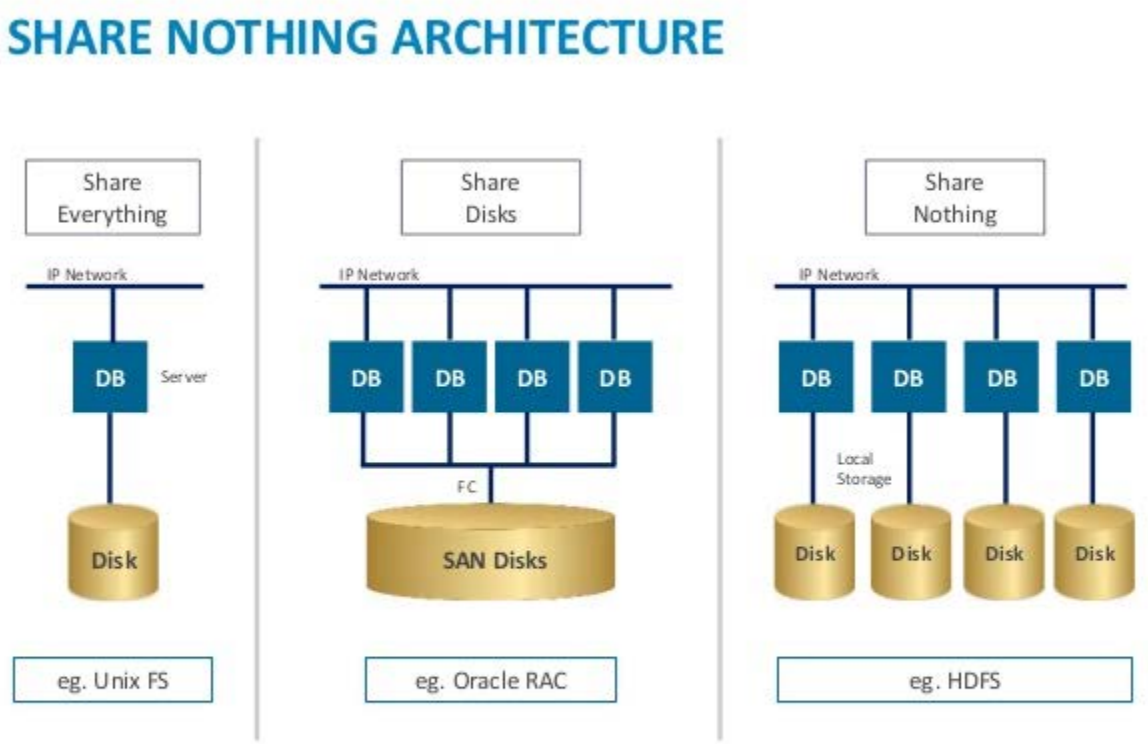
For example, when you have redundant copies of data or databases on different disks, a disk that loses data may be able to recover it when the redundancies are synced.



SHARED NOTHING" ARCHIT

Had it instead been a single, shared disk, the data would be lost and downtime would be indefinite.

As you can see, shared nothing architecture can be very helpful.



DISTRIBUTION MODELS

NoSQL has its ability to run databases on a large cluster.

The ability to process a greater read or write traffic, or more availability in the face of network slowdowns or breakages.

There are two paths to data distribution: Replication and Sharding.

- **Replication: Replication takes the same data and copies it over multiple nodes.**
- **Sharding: Sharding is a method for storing data across multiple machines.**

Replication comes into two forms:

- **Master-slave**
- **Peer-to-peer.**

DISTRIBUTION MODELS

NoSQL has its ability to run databases on a large cluster.

The ability to process a greater read or write traffic, or more availability in the face of network slowdowns or breakages.

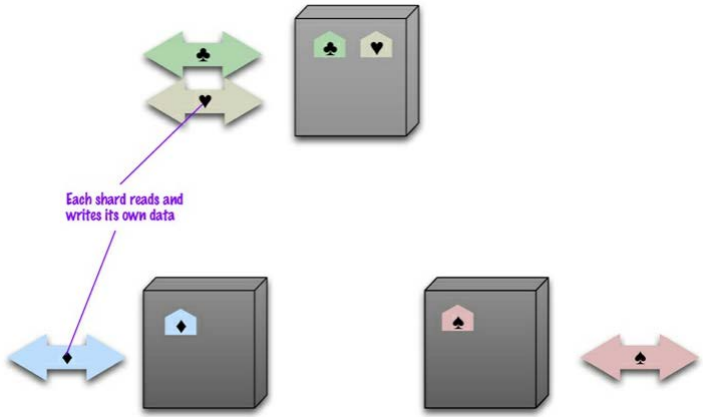
There are two paths to data distribution: Replication and Sharding.

- **Replication: Replication takes the same data and copies it over multiple nodes.**
- **Sharding: Sharding is a method for storing data across multiple machines.**

Replication comes into two forms:

- **Master-slave**
- **Peer-to-peer.**

SHARDING



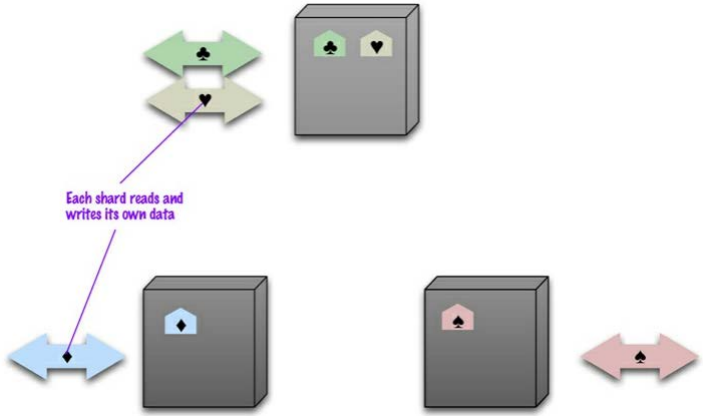
In a busy data store different people are accessing different parts of the dataset.

In these circumstances we can support horizontal scalability by putting different parts of the data onto different servers—a technique that's called sharding.

The load is balanced out nicely between servers—for example, if we have ten servers, each one only has to handle 10% of the load.

In order to do it we have to ensure that data that's accessed together is clumped together on the same node and that these clumps are arranged on the nodes to provide the best data access.

SHARDING



The first part of this question is how to clump the data up so that one user mostly gets her data from a single server.

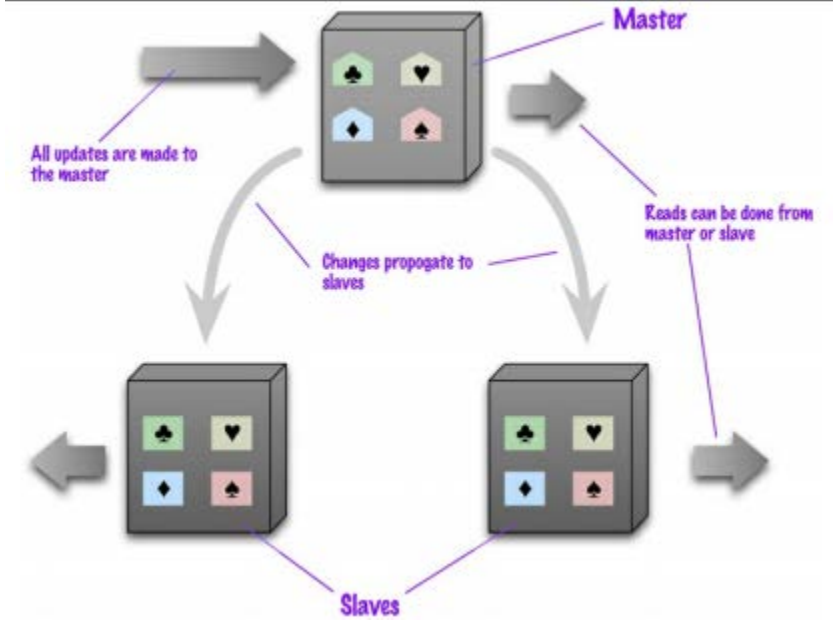
When it comes to arranging the data on the nodes, there are several factors that can help improve performance.

If you know that most accesses of certain aggregates are based on a physical location, you can place the data close to where it's being accessed. If you have orders for someone who lives in Boston, you can place that data in your eastern US data center.

MASTER-SLAVE REPLICATION

With master-slave distribution, you replicate data across multiple nodes.

One node is designated as the master, or primary. This master is the authoritative source for the data and is usually responsible for processing any updates to that data. The other nodes are slaves, or secondaries.



PEER-TO-PEER REPLICATION

Master-slave replication helps with read scalability but doesn't help with scalability of writes.

Peer-to-peer replication attacks these problems by not having a master.

All the replicas have equal weight, they can all accept writes, and the loss of any of them doesn't prevent access to the data store.