

# SQL, NoSQL, NewSQL

Prepared By : Prof. Shital Pathar

Prepared For :- DDU CE Semester-7

# Database Categories



Oracle  
MySQL  
MS SQL  
DB2  
Etc.

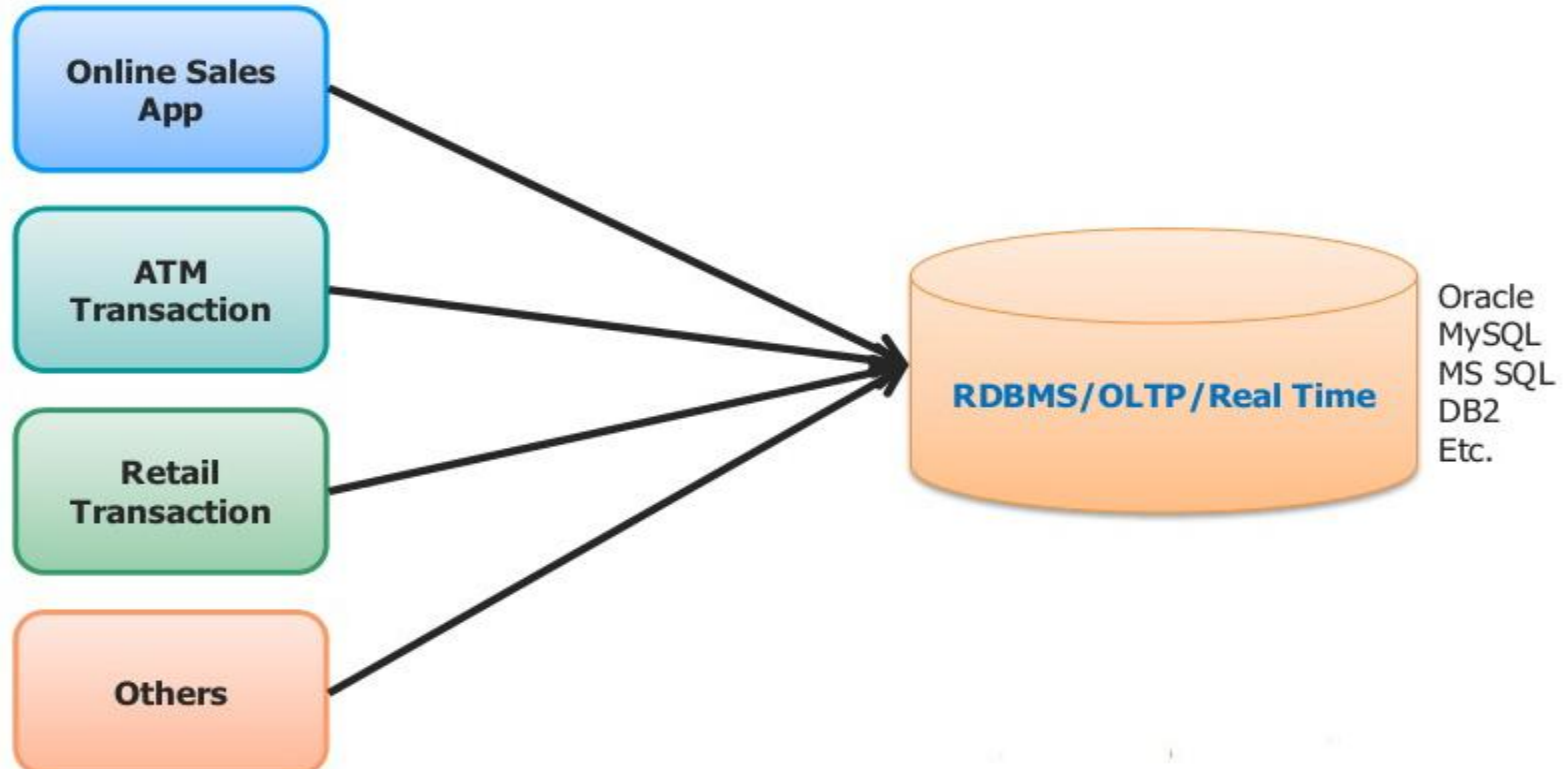


MongoDB  
Hbase  
Cassandra  
CauchDB  
Etc.



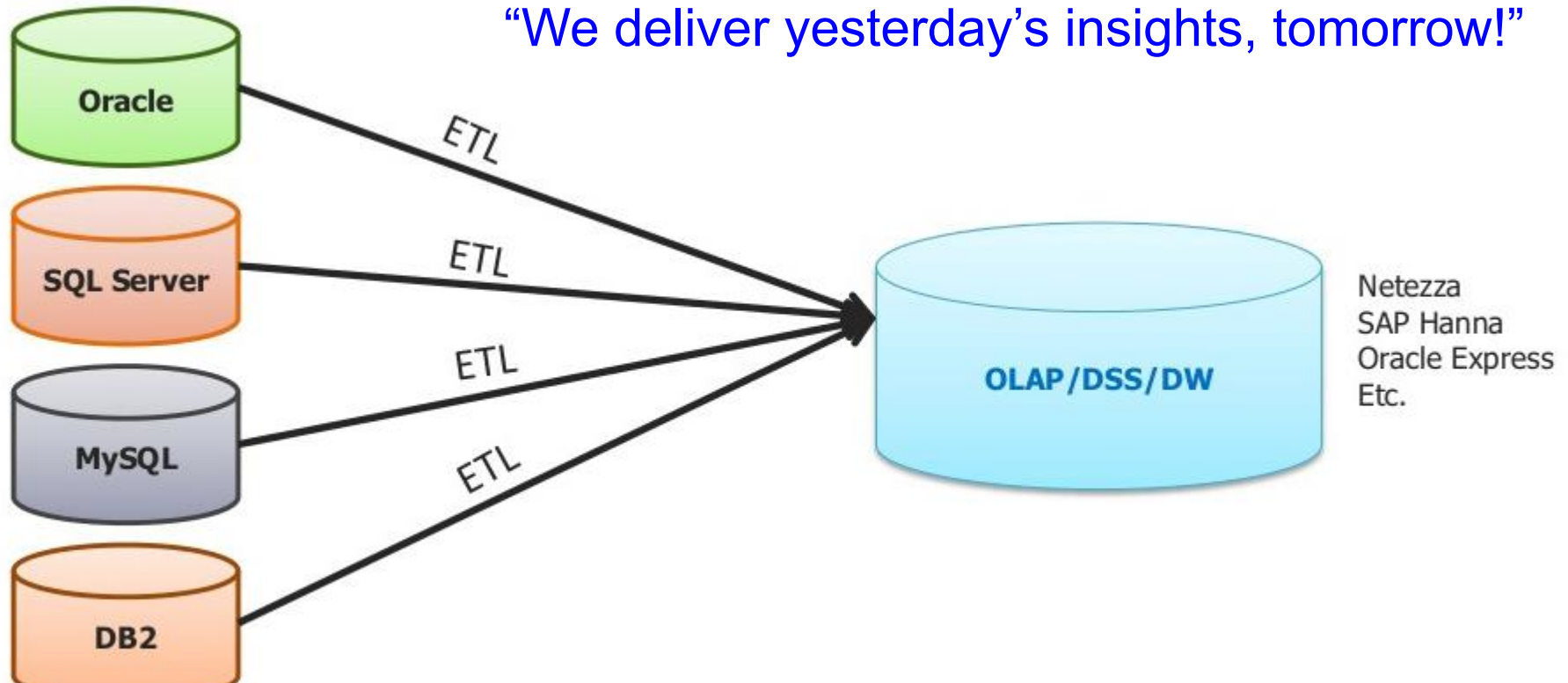
Netezza  
SAP Hanna  
Oracle Express  
Etc.

# OLTP Database

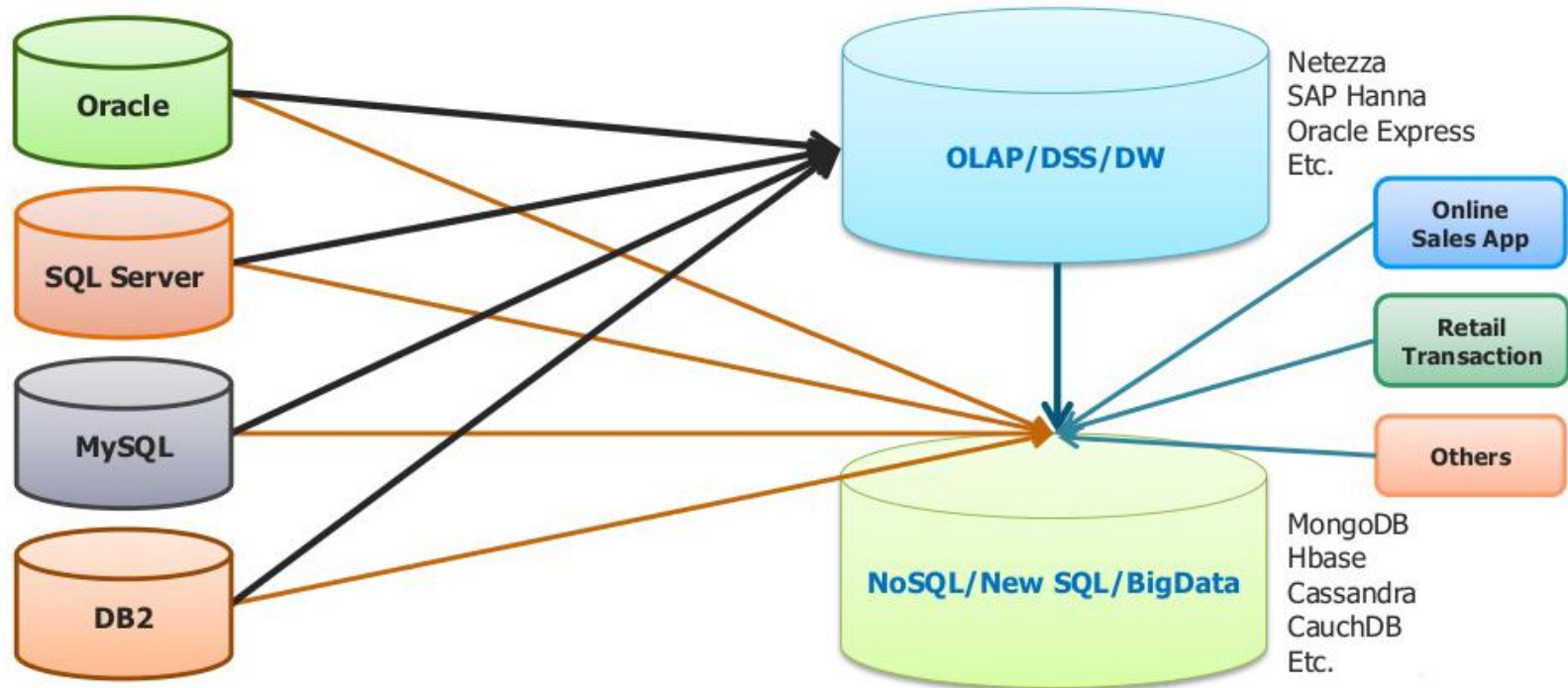


# OLTP to OLAP

“We deliver yesterday’s insights, tomorrow!”



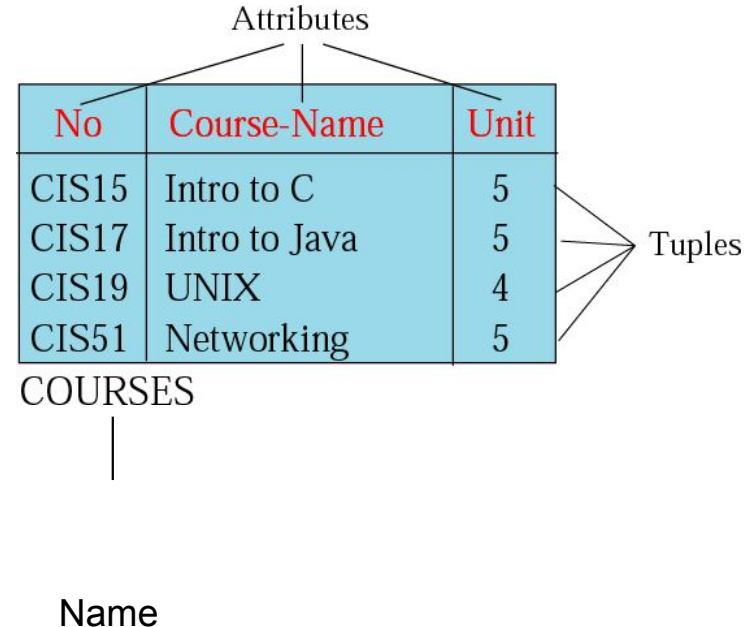
# NoSQL Databases



# What is RDBMS

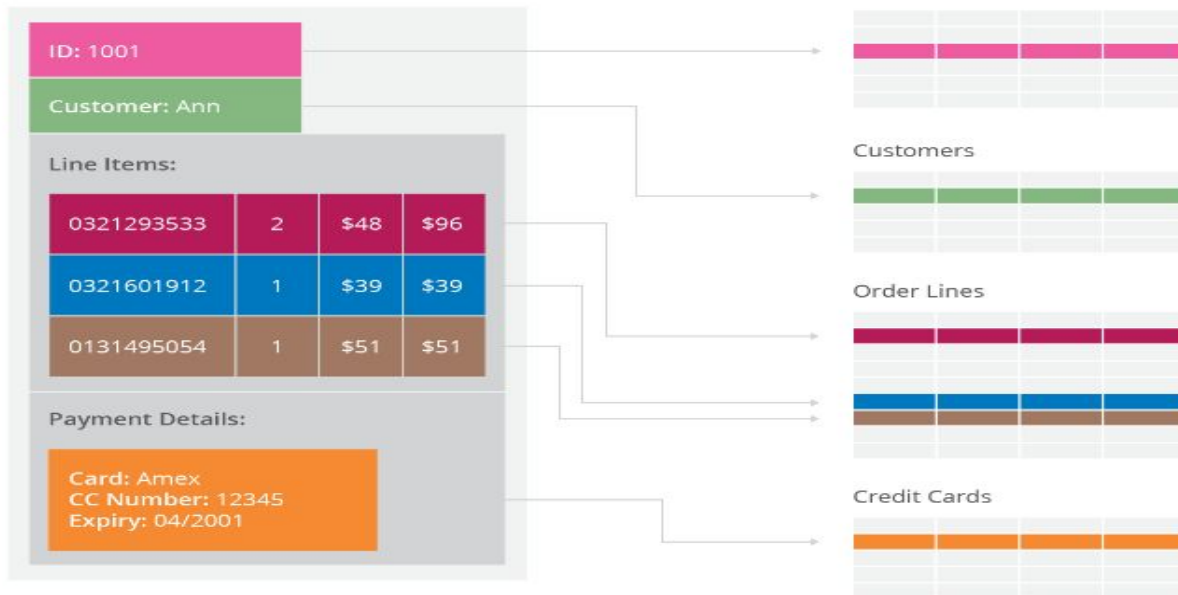
2

- ❑ RDBMS: the relational database management system.
- ❑ Relation: a relation is a 2D table which has the following features:
  - Name
  - Attributes
  - Tuples



# Issues with RDBMS- impedance mismatch

**impedance mismatch** between the relational data structures and the in-memory data structures of the application



# Issues with RDBMS– Scalability



- ❑ Issues with scaling up when the dataset is just too big e.g. Big Data.
- ❑ Not designed to be distributed.
- ❑ Looking at multi-node database solutions. Known as ‘horizontal scaling’.



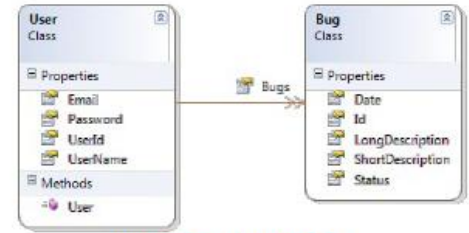
# What is NoSQL?



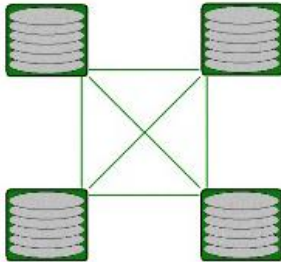
**Next Generation Databases**

**N**ot  
**O**nly SQL

**Not Only SQL**



**Non – Relational**



**Distributed Architecture**



**Open Source**



**Horizontally Scalable**

# What is NoSQL?

**Schema – Free !**

**Schema - Free**



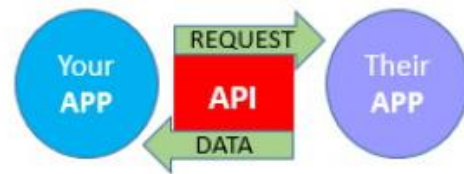
**Can Manage Huge  
Amount of Data**



**Easy – Replication**



**Can be implement on  
Commodity Hardware's**

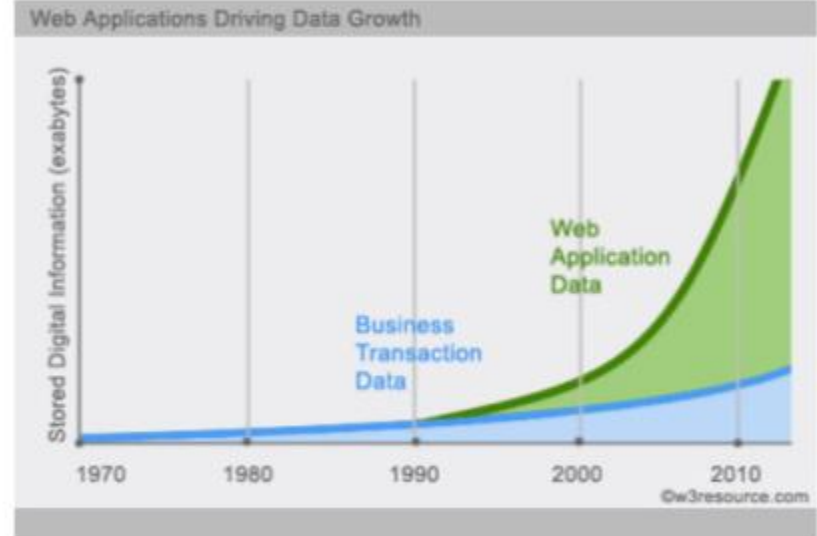
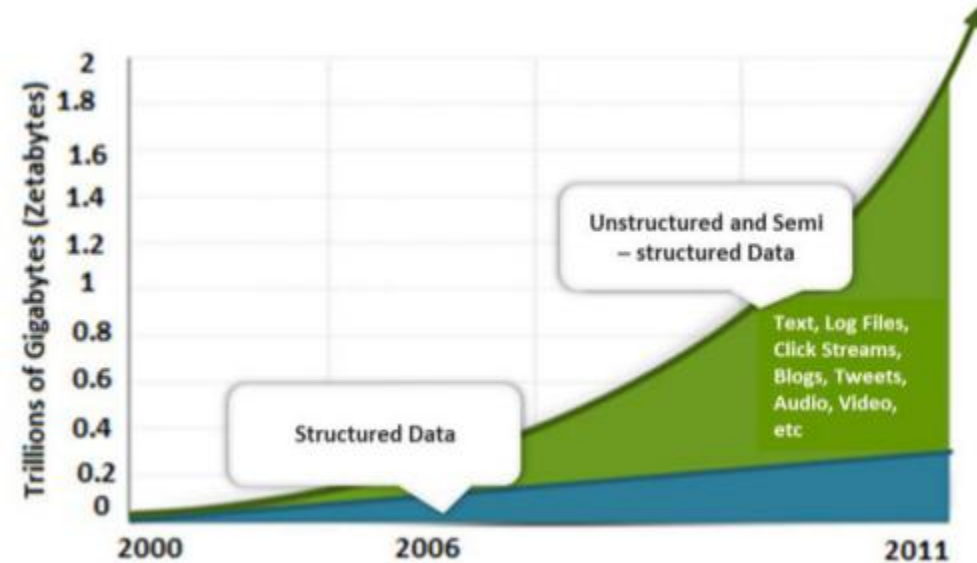


**Simple API**



**~ 150 No SQL Database  
are there in Market**

# Why NoSQL?



## Data Warehousing and Analytics

Source: IDC 2011 Digital Universe Study [<http://www.emc.com/collateral/demos/microsites/emc-digital-universe-2011index.html>]

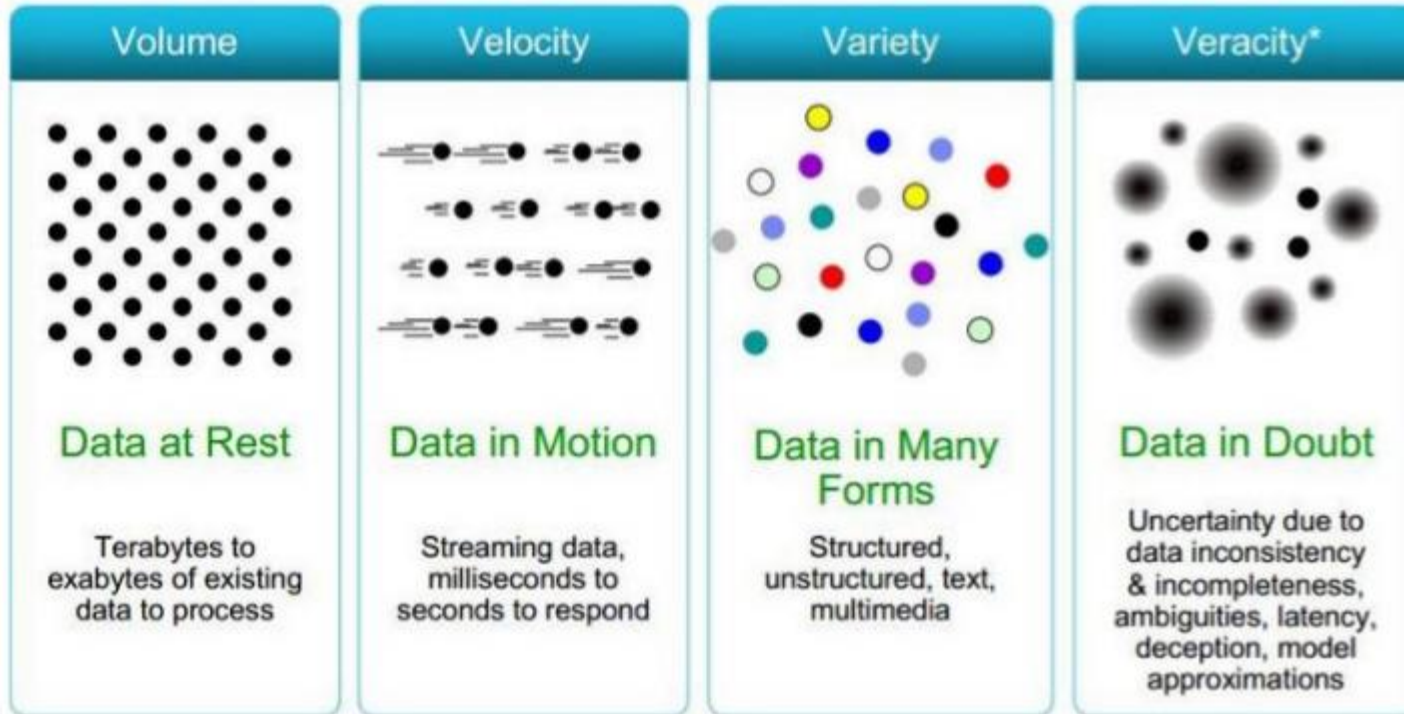
# Need of NoSQL

6

- Explosion of social media sites (Facebook, Twitter, Google etc.) with large data needs. (Sharding is a problem)
- Rise of cloud-based solutions such as Amazon S3 (simple storage solution).
- Just as moving to dynamically-typed languages (Ruby/Groovy), a shift to dynamically-typed data with frequent schema changes.
- Expansion of Open-source community.
- NoSQL solution is more acceptable to a client now than a years ago.

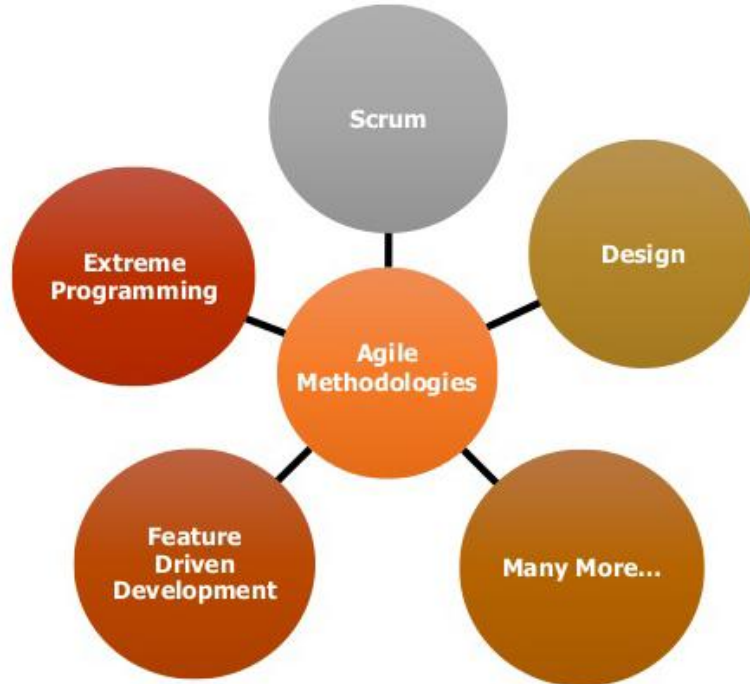
# Benefits of NoSQL

- ✓ Large volumes of structured, semi-structures, and unstructured data.



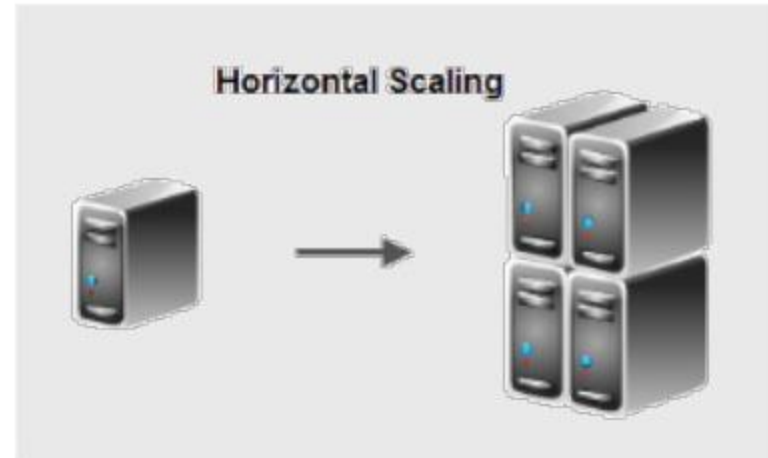
# Benefits of NoSQL

- ✓ Agile development, quick changes, and frequent code pushes.



# Benefits of NoSQL

- ✓ Object-oriented programming that is easy to use and flexible
- ✓ Horizontal scaling instead of expensive hardware's



# NoSQL Types

7

NoSQL database are classified into four types:

- Key Value pair based
- Column based
- Document based
- Graph based

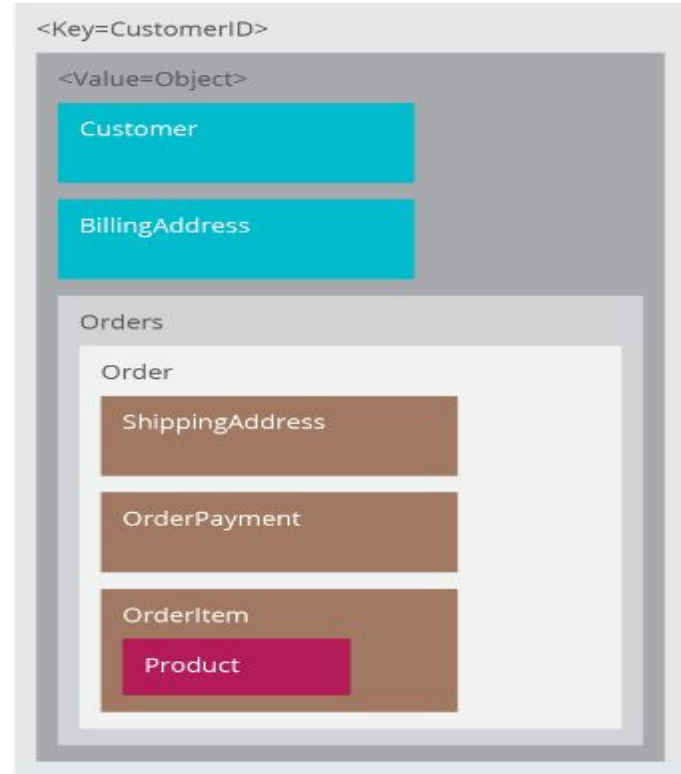
<https://www.thoughtworks.com/en-in/insights/blog/nosql-databases-overview>



# Key Value Pair Based

Key_1	Value_1
Key_2	Value_2
Key_3	Value_1
Key_4	Value_3
Key_5	Value_2
Key_6	Value_1
Key_7	Value_4
Key_8	Value_3

Key →  
Value →



# Key Value Pair Based

8

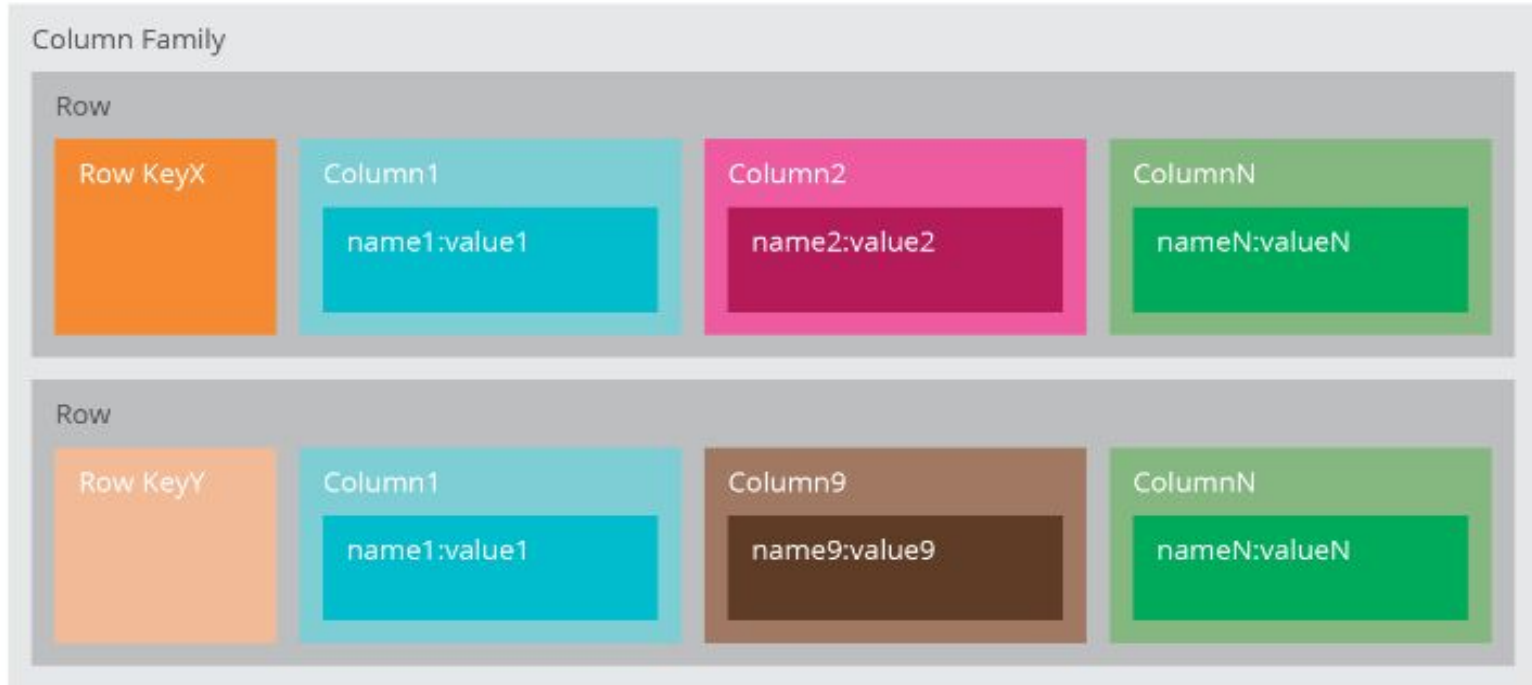
- Designed for processing dictionary. Dictionaries contain a collection of records having fields containing data.
- Records are stored and retrieved using a key that uniquely identifies the record, and is used to quickly find the data within the database.

**Example:** CouchDB, Oracle NoSQL Database, Riak etc.

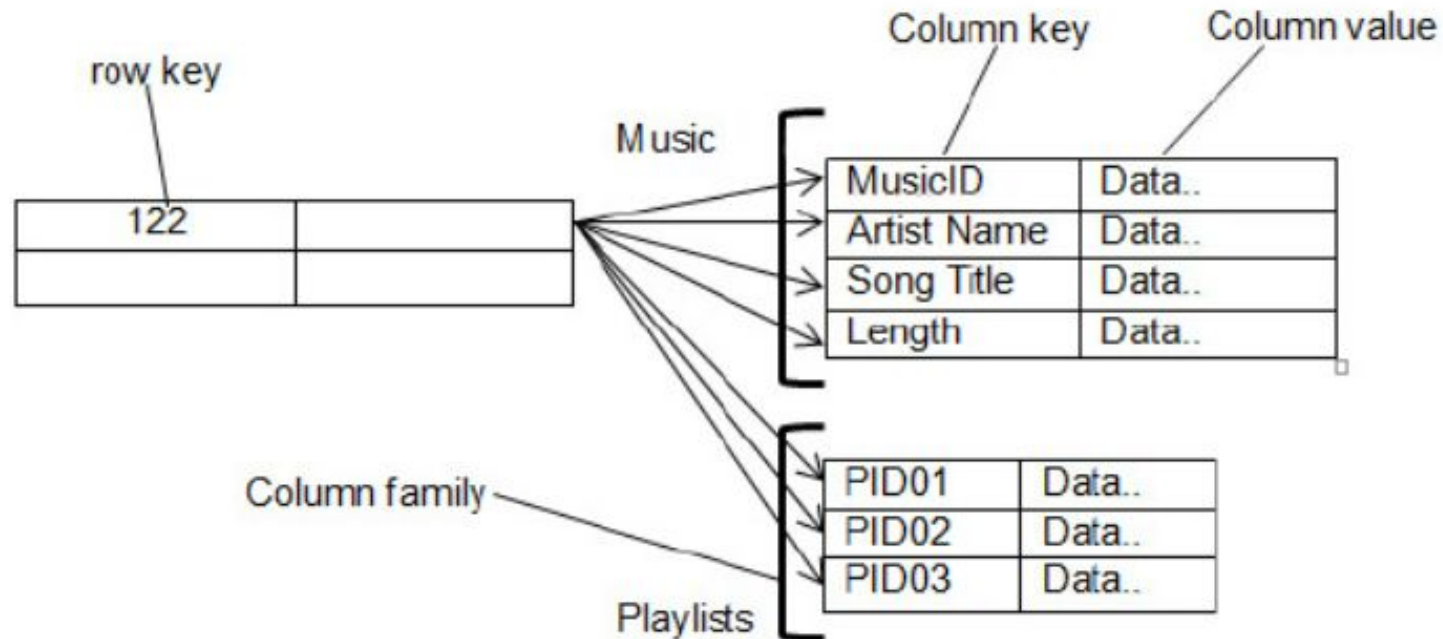
We use it for storing session information, user profiles preferences, shopping cart data.

We would avoid it when we need to query data having relationships between entities.

# Column based



# Column based(Example)



# Column based

9

It store data as Column families containing rows that have many columns associated with a row key. Each row can have different columns.

Column families are groups of related data that is accessed together.

**Example:** Cassandra, HBase, Hypertable, and Amazon DynamoDB.

We use it for content management systems, blogging platforms, log aggregation.

We **would avoid it** for systems that are in early development, changing query patterns.

# Document Based

10

The database stores and retrieves documents. It stores documents in the value part of the key-value store.

Self- describing, hierarchical tree data structures consisting of maps, collections, and scalar values.

**Example:** Lotus Notes, MongoDB, Orient DB, Raven DB.

We use it for content management systems, blogging platforms, web analytics, real-time analytics, e-commerce applications.

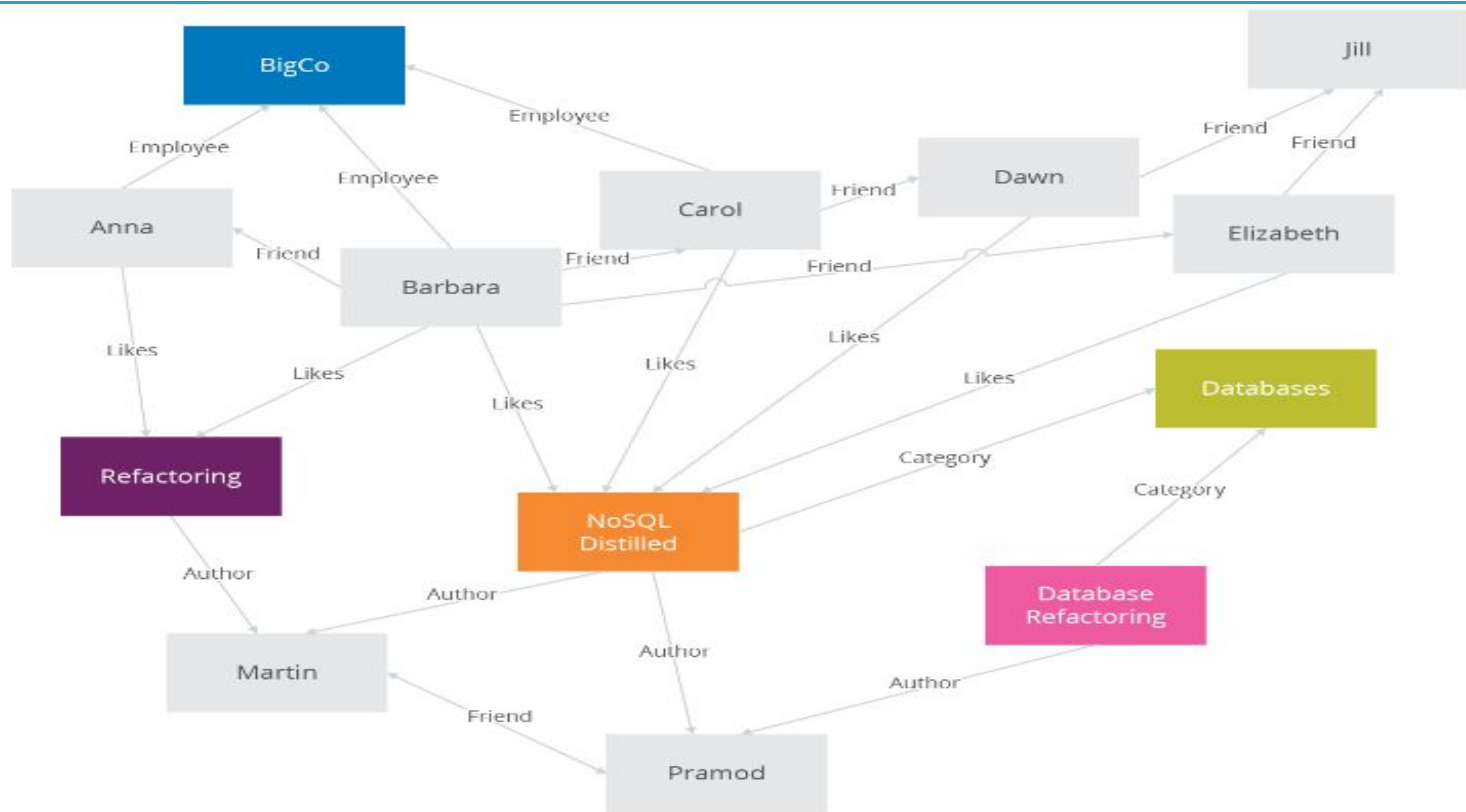
We would avoid it for systems that need complex transactions spanning multiple operations or queries against varying aggregate structures.

<Key=CustomerID>

```
{
  "customerid": "fc986e48ca6"
  "customer":
  {
    "firstname": "Pramod",
    "lastname": "Sadalage",
    "company": "ThoughtWorks",
    "likes": [ "Biking", "Photography" ]
  }
  "billingaddress":
  {
    "state": "AK",
    "city": "DILLINGHAM",
    "type": "R"
  }
}
```

← Key

# Graph Based



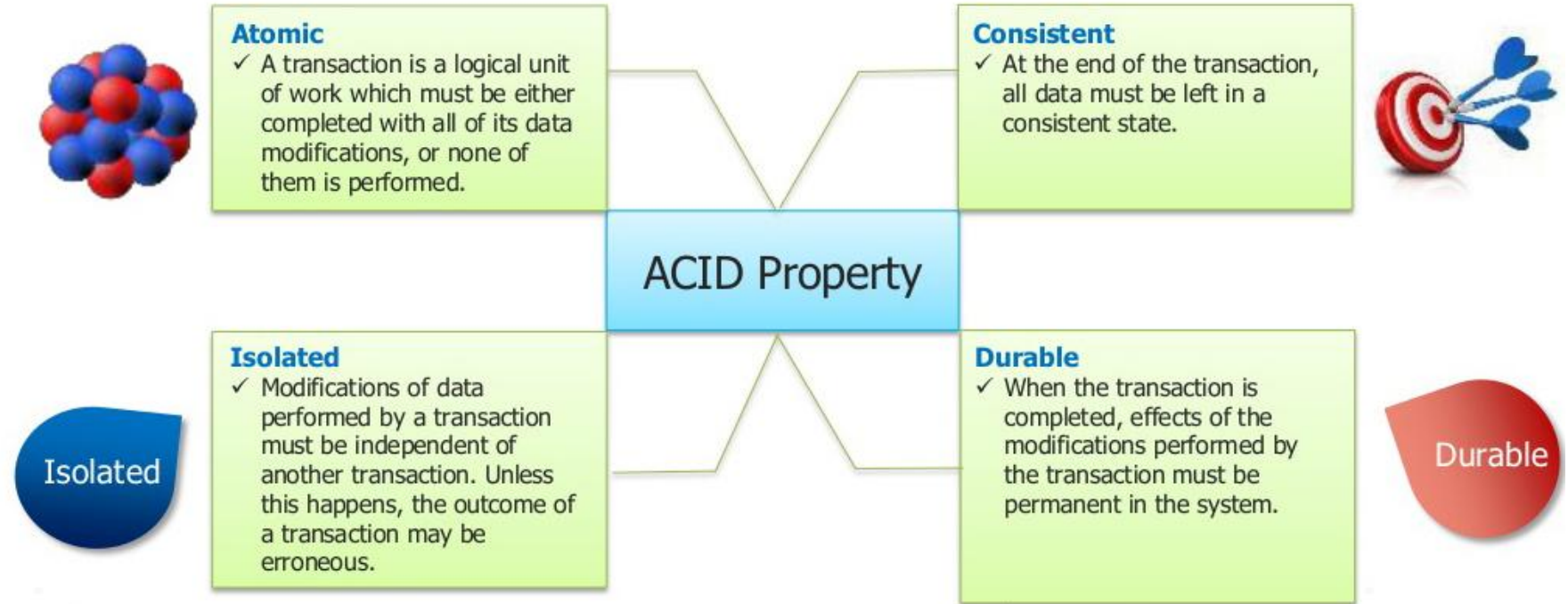
# Graph Based

11

- Store entities and relationships between these entities as nodes and edges of a graph respectively. Entities have properties.
- Traversing the relationships is very fast as relationship between nodes is not calculated at query time but is actually persisted as a relationship.
- **Example:** Neo4J, Infinite Graph, OrientDB, FlockDB.
- It is well suited for connected data, such as social networks, spatial data, routing information for goods and supply.



# SQL Databases (ACID Property)



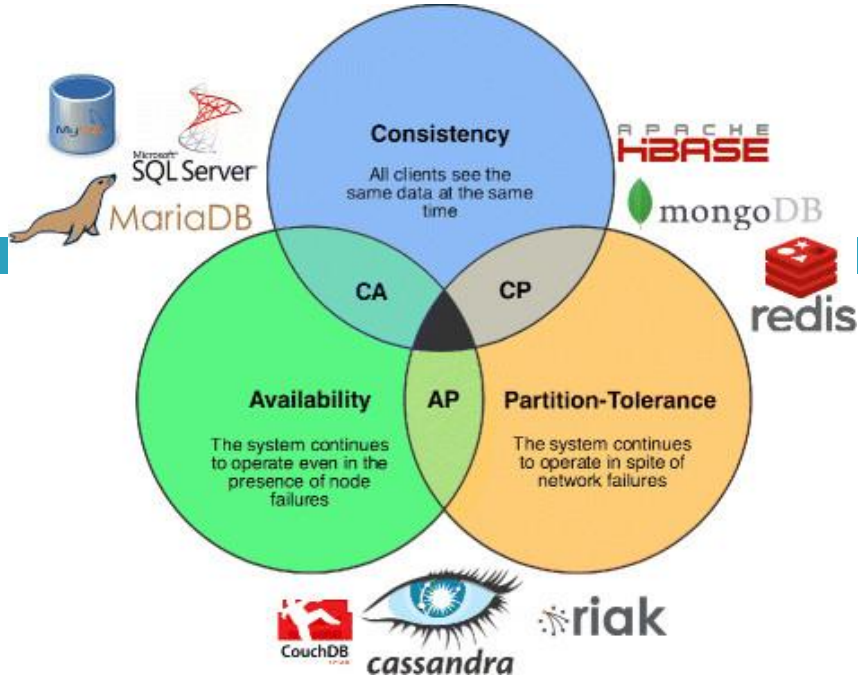
# Brewer's CAP Theorem

12

- According to Eric Brewer, a distributed system has 3 properties

- Consistency
- Availability
- Partitions

- We can have at most two of these three properties for any shared-data system
- To scale out, we have to partition. It leaves a choice between consistency and availability. (In almost all cases, we would choose availability over consistency)
- Everyone who wants to develop big applications, builds them on CAP : Google, Yahoo, Facebook, Amazon, eBay, etc.





## **Consistency**


Consistency means that all clients see the same data at the same time, no matter which node they connect to. For this to happen, whenever data is written to one node, it must be instantly forwarded or replicated to all the other nodes in the system before the write is deemed 'successful.'

## **Availability**

Availability means that any client making a request for data gets a response, even if one or more nodes are down. Another way to state this—all working nodes in the distributed system return a valid response for any request, without exception.


## **Partition tolerance**

A partition is a communications break within a distributed system—a lost or temporarily delayed connection between two nodes. Partition tolerance means that the cluster must continue to work despite any number of communication



**CP database:** A CP database **delivers consistency and partition tolerance at the expense of availability**. When a partition occurs between any two nodes, the system has to shut down the non-consistent node (i.e., make it unavailable) until the partition is resolved.

**CA database:** A CA database **delivers consistency and availability across all nodes**. It can't do this if there is a partition between any two nodes in the system, however, and therefore can't deliver fault tolerance.



**AP database:** An AP database **delivers availability and partition tolerance at the expense of consistency**. When a partition occurs, all nodes remain available but those at the wrong end of a partition might return an older version of data than others. (When the partition is resolved, the AP databases typically resync the nodes to repair all inconsistencies in the system.)

# Advantages of NoSQL

13

- ❑ Cheap and easy to implement (open source)
- ❑ Data are replicated to multiple nodes (therefore identical and fault-tolerant) and can be partitioned
  - When data is written, the latest version is on at least one node and then replicated to other nodes
  - No single point of failure
- ❑ Easy to distribute
- ❑ Don't require a schema

# What is not provided by NoSQL

14

- ❑ Joins
- ❑ Group by
- ❑ ACID transactions
- ❑ SQL
- ❑ Integration with applications that are based on SQL

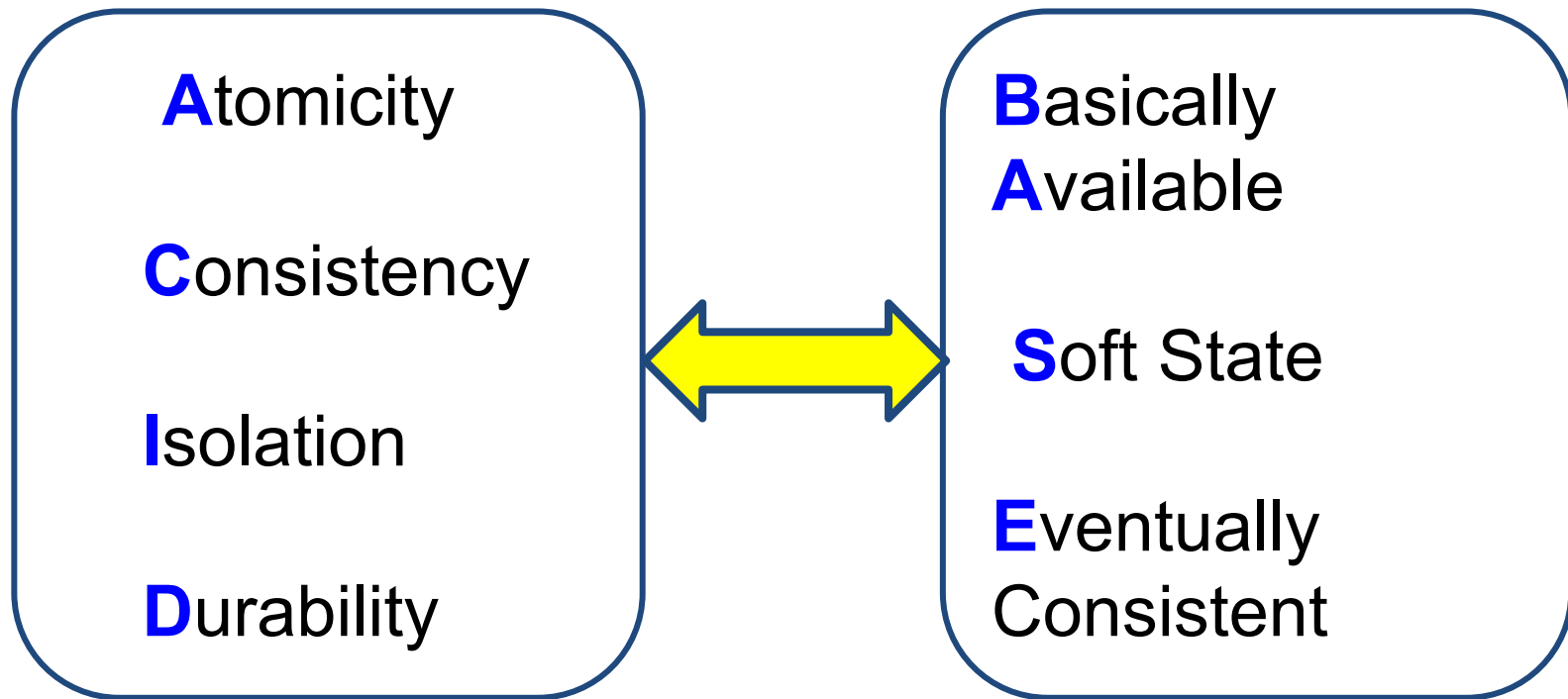
# Where to use NoSQL

15

- ❑ NoSQL Data storage systems makes sense for applications that process very large **semi-structured data** –like **Log Analysis, Social Networking Feeds, Time-based data**.
- ❑ To improve programmer productivity by using a database that better matches an application's needs.
- ❑ To improve data access performance via some combination of handling larger data volumes, reducing latency, and improving throughput.



# RDBMS ACID to NOSQL BASE



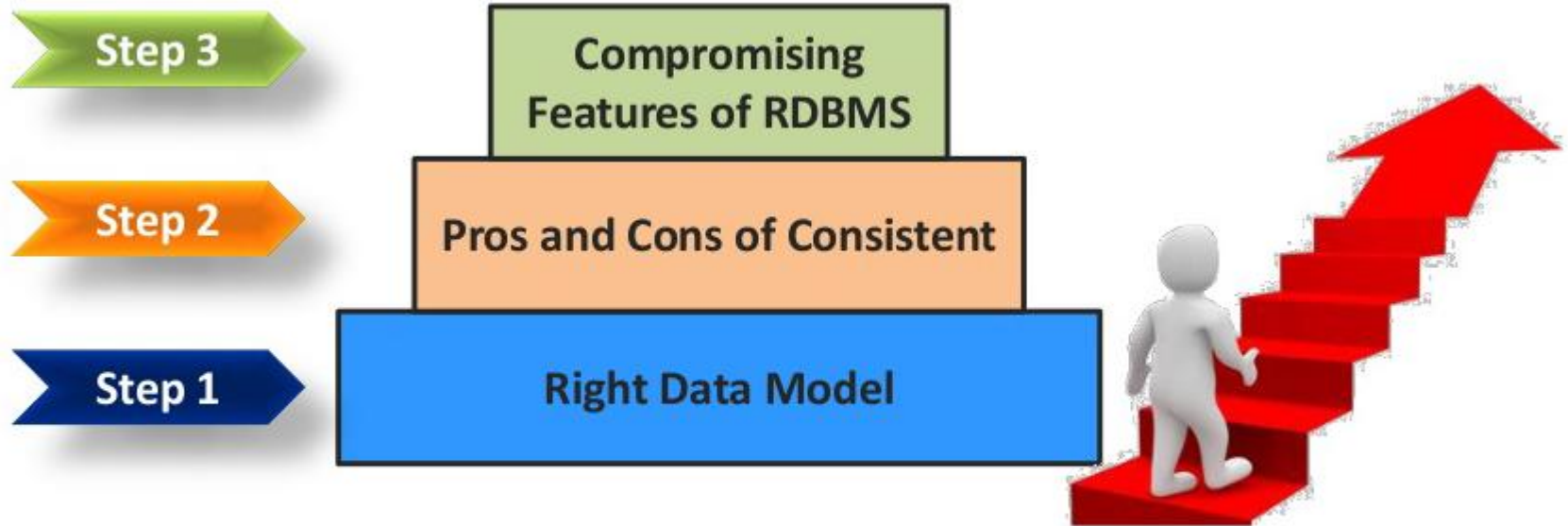
# ACID Model

- **Atomic** – Each transaction is either properly carried out or the process halts and the database reverts back to the state before the transaction started. This ensures that all data in the database is valid.
- **Consistent** – A processed transaction will never endanger the structural integrity of the database.
- **Isolated** – Transactions cannot compromise the integrity of other transactions by interacting with them while they are still in progress.
- **Durable** – The data related to the completed transaction will persist even in the cases of network or power outages. If a transaction fails, it will not impact the manipulated data.

# BASE Model

- **Basically Available** – Rather than enforcing immediate consistency, BASE-modelled NoSQL databases will ensure availability of data by spreading and replicating it across the nodes of the database cluster.
- **Soft State** – Stores don't have to be write-consistent, nor do different replicas have to be mutually consistent all the time. It means that the state of the system can change over time, even without input.
- **Eventually Consistent** – It means that the system will eventually become consistent, although there may be some inconsistency in the meantime.

# Before Selection and Implementation of NoSQL



# NoSQL Landscape



# NewSQL

- NewSQL is a unique database system that **combines ACID compliance with horizontal scaling**. The database system strives to keep the best of both worlds. OLTP-based transactions and the high performance of NoSQL combine in a single solution.
- Enterprises expect high-quality of data integrity on large data volumes. When either becomes a problem, an enterprise chooses to:
  - Improve hardware, or
  - Create custom software for distributed databases
- Both solutions are expensive on both a software and hardware level. NewSQL strives to improve these faults by creating consistent databases that scale.

# NewSQL Features

- **In-memory storage** and data processing supply fast query results.
- **Partitioning** scales the database into units. Queries execute on many shards and combine into a single result.
- **ACID** properties preserve the features of RDBMS.
- **Secondary indexing** results in faster query processing and information retrieval.
- **High availability** due to the database replication mechanism.
- A **built-in crash recovery** mechanism delivers fault tolerance and minimizes downtime.

# Difference between SQL vs NOSQL vs NewSQL

Feature	SQL	NoSQL	NewSQL
Schema	Relational (table)	Schema-free	Both
SQL	Yes	Depends on the system	Yes, with enhanced features
ACID	Yes	No (BASE)	Yes
OLTP	Partial support	Not supported	Full support
Scaling	Vertical	Horizontal	Horizontal
Distributed	No	Yes	Yes
High availability	Custom	Auto	Built-in
Queries	Low complexity queries	High complexity queries	Both



# References

1. “NoSQL Databases: An Overview”. Pramod Sadalage, thoughtworks.com(2014)
2. “Data management in cloud environments: NoSQL and NewSQL data stores”. Grolinger, K.; Higashino, W. A.; Tiwari, A.; Capretz, M. A. M. (2013). JoCCASA, Springer.
3. “Making the Shift from Relational to NoSQL”. Couchbase.com(2014).
4. <https://www.singlestore.com/blog/pre-modern-databases-oltp-olap-and-nosql/>
5. “NoSQL - Death to Relational Databases”. Scofield, Ben (2010).
6. <https://www.youtube.com/watch?v=UyGlgzc7i4w>



Thank you..!!