

NoSQL Programming

Trainer : Pradnyaa S Dindorkar

Email: pradnya@sunbeaminfo.com

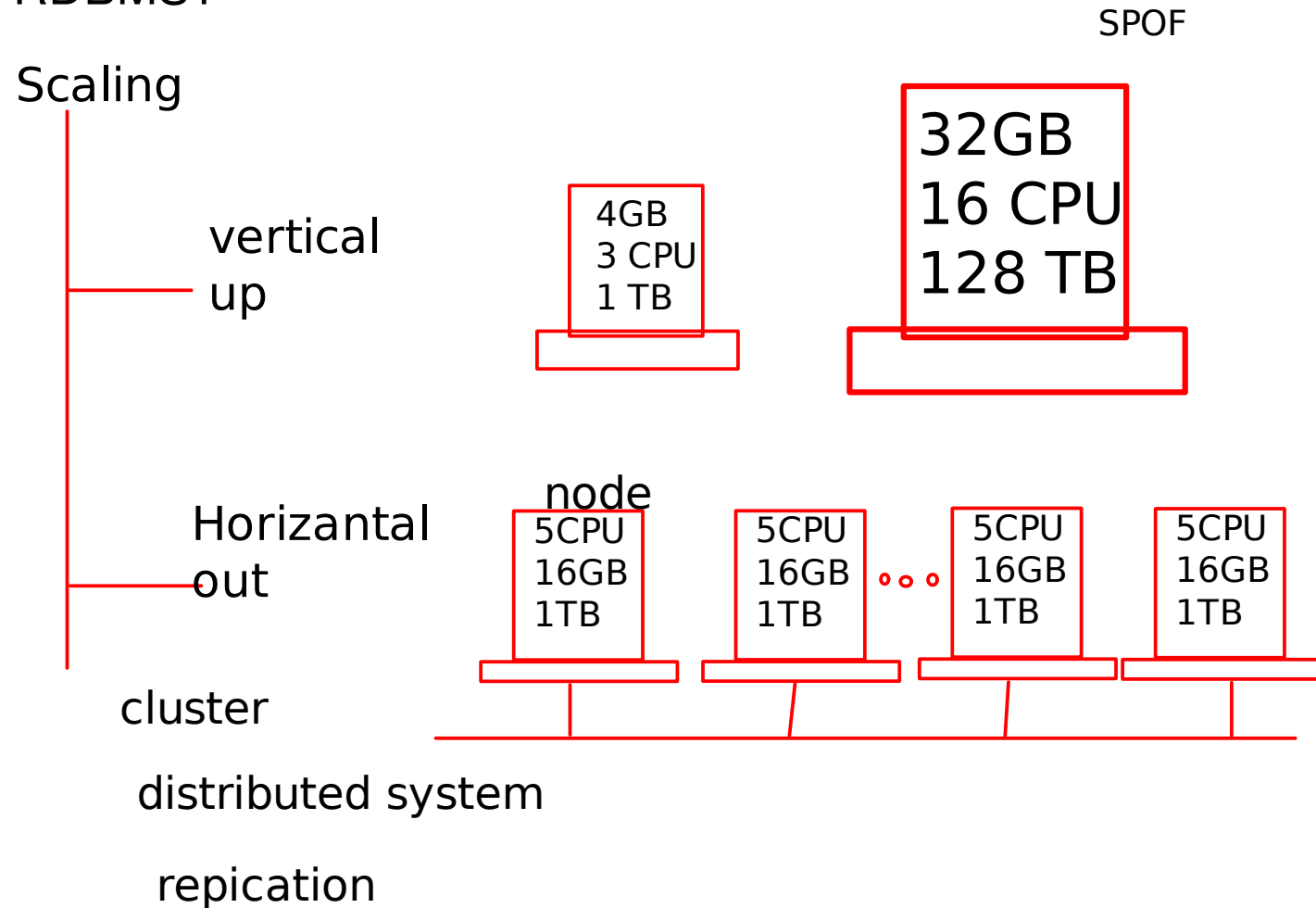


- Introduction to NoSQL ✓
- MongoDB ✓
- Redis ✓
- Cassandra ✓




What are we going to cover?

- What is NoSQL and how is it different from RDBMS?
- What is JSON?
- Introduction to MongoDB
- JSON vs BSON
- MongoDB Fundamentals
- Basic CRUD operations
- Performance optimization
- Data Modeling
- Aggregation Pipeline
- Introduction to Geospatial Queries



Todyas Topics

- 
- ✓ ■ What is database?
 - ✓ ■ Quick recap of sql,
 - ✓ ■ limitation of sql,
 - ✓ ■ intro of nosql ,
 - ✓ ■ CAP theorem and BASE theorem
 - ✓ ■ types of nosql
 - ✓ ■ Advantages of NoSQL
 - ✓ ■ Disadvantages of NoSQL
 - ✓ ■ NoSQL Scenarios,
 - SQL vs NoSQL →
- ✓



What is Database ?

- A database is an organized collection of data, generally stored and accessed electronically from a ✓ computer system
 - A database refers to a set of related data and the way it is organized
 - Database Management System
 - Software that allows users to interact with one or more databases and provides access to all of the data contained in the database
 - Types
 - RDBMS (SQL) ✓
 - NoSQL ✓
- NewSQL



RDMBS

- The idea of RDBMS was borne in 1970 by E. F. Codd
- The language used to query RDBMS systems is SQL (Sequel Query Language)
- RDBMS systems are well suited for structured data held in columns and rows, which can be queried using SQL
- Supports: DML, DQL, DDL, DTL, DCL
- The RDBMS systems are based on the concept of ACID transactions
 - **Atomic**: implies either all changes of a transaction are applied completely or not applied at all
 - **Consistent**: data is in a consistent state after the transaction is applied
 - **Isolated**: transactions that are applied to the same set of data are independent of each other
 - **Durable**: the changes are permanent in the system and will not be lost in case of any failures





- Scalability is the ability of a system to expand to meet your business needs.
- Scalability describes a elasticity of the system, ability to adapt to change and demand.
- Good scalability ensures the quality of your service.

Vertical Scaling



CPU: 3 , RAM: 6G



CPU: 2 , RAM:4G



CPU: 1 , RAM:2G



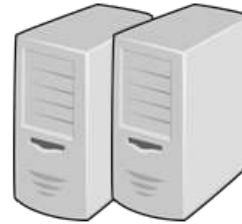
Vertical scaling describes adding more resources to your current machines.

- ✓ increase memory in the system
- ✓ expanding storage by adding hard drives
- ✓ upgrading the CPUs.
- ✓ upgrading network speed.

Horizontal Scaling



1 PC (CPU: 1, RAM:2G)



2 PC (CPU: 1, RAM:2G)



3 PC (CPU: 1, RAM:2G)



Horizontal Scaling refers to adding additional nodes or machines to your infrastructure to cope with new demands.

- ✓ adding a new computer to a distributed software application

Structured Data



Structured data is highly specific and is stored in a predefined format, well-defined data model.

Example: Employee table in the database ,
Banking transaction data .

Table: Transaction				+: Credit, -: Debit	
Transaction ID	Transaction Date Time	User ID	Account ID	Amount	Account Balance
1000001	01/04/2012 09:10:19	2	1	3100.00	4,300.21
1000002	01/04/2012 11:10:19	4	3	5800.00	6,412.44
1000003	01/04/2012 12:10:19	3	4	1200.00	307.85
1000004	01/04/2012 13:10:19	1	5	2500.00	229.87
1000005	02/04/2012 09:10:19	5	1	-50.00	4,250.21
1000006	02/04/2012 11:10:19	3	3	-100.00	612.44
1000007	02/04/2012 14:10:19	1	6	810.00	-99,119.91
1000008	03/04/2012 09:10:19	3	1	-50.00	4,200.21
1000009	03/04/2012 11:10:19	1	3	-100.00	512.44
1000010	03/04/2012 14:10:19	5	6	810.00	-98,309.91

Semi-structured Data



Semi-structured data contains some level of organization or structure, but does not conform to a rigid schema or data model

Example: XML data, JSON data

```
{
  "empoyee":
  [
    {
      "name": "sujay",
      "age": 56,
      "married": true,
      "contact": 9887667767 ,
      "Language": ["C++", "JAVA", "Python", "Kotlin", "Go"]
    },
    {
      "name": "ajay",
      "age": 28,
      "married": false,
      "Favorite-Sports": ["Football", "Basketball"]
    }
  ]
}
```

fb 2008

```
{
  txt:----
  img:--- jpg/png
  links:----
  video:----
  comm:----
  loc:-----
  tag:---
  feeling:----
}
```

Unstructured Data



Unstructured data is information that is not arranged according to a preset data model or schema. Unstructured data is very difficult to analyze. It required AI and machine learning , new software tools

Example: Videos, Audio files , Word, PDF, Text, Media logs.



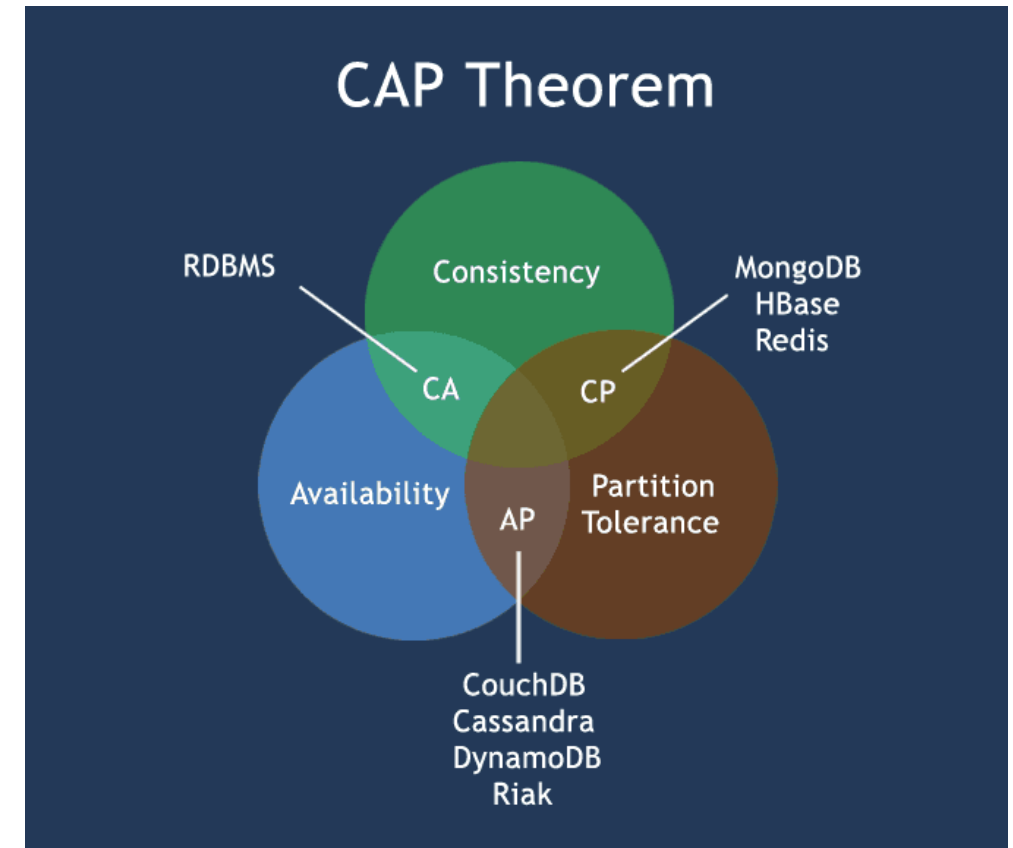
NoSQL

- Stands for Not Only SQL
- NoSQL is a term used to refer to non-relational databases
- The term NoSQL was coined by Carlo Strozzi in 1998 .
- Does not use any declarative query language
- No predefined schema → no restriction over the schema flexible schema
- Unstructured and unpredictable data
- Supports eventual consistency rather than ACID properties
- ✓ - Prioritizes high performance, high availability and scalability
- In contrary to the ACID approach of traditional RDBMS systems, NoSQL solves the problem using an approach popularly called as BASE



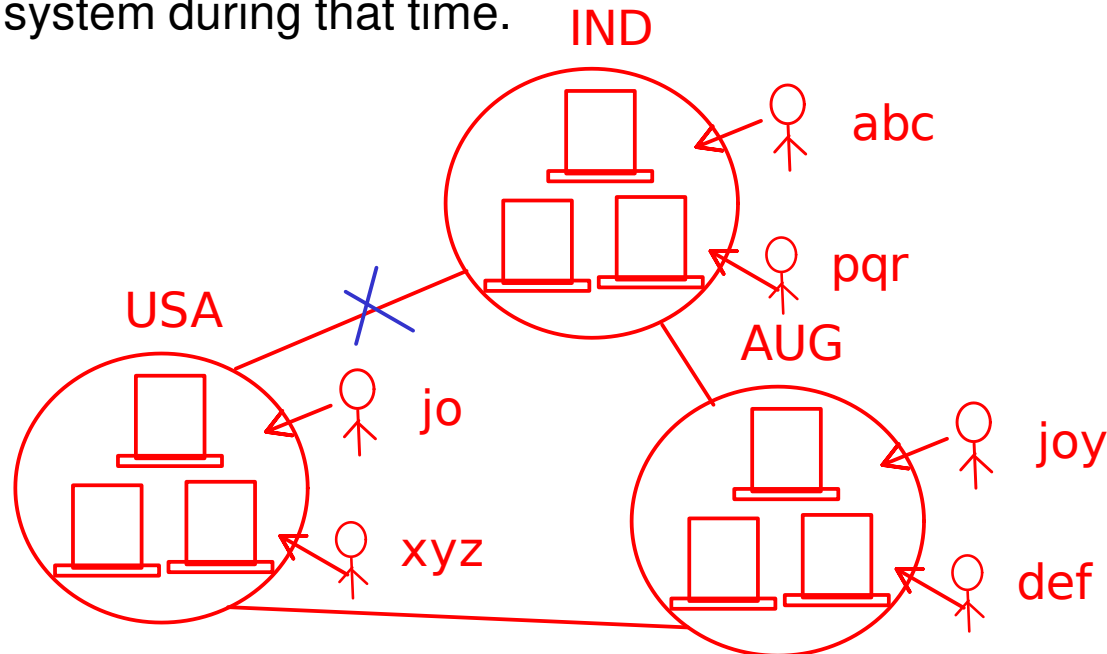
CAP Theorem

- Also known as Brewer's theorem states that it is impossible for a distributed data store to simultaneously provide more than two out of the following three guarantees
 - Consistency ✓
 - Data is consistent after operation
 - After an update operation, all clients see the same data
 - Availability ✓
 - System is always on (i.e. service guarantee), no downtime 24X7
 - Partition Tolerance ✓
 - System will continue to function even if it is partitioned into groups of servers that are not able to communicate with one another



BASE Theorm

- Eric Brewer coined the BASE acronym
- BASE means
 - **Basically Available:** means the system will be available in terms of the CAP theorem. 24X7
 - **Soft state** indicates that even if no input is provided to the system, the state will change over time. This is in accordance to eventual consistency.
 - **Eventual consistency:** means the system will attain consistency in the long run, provided no input is sent to the system during that time.



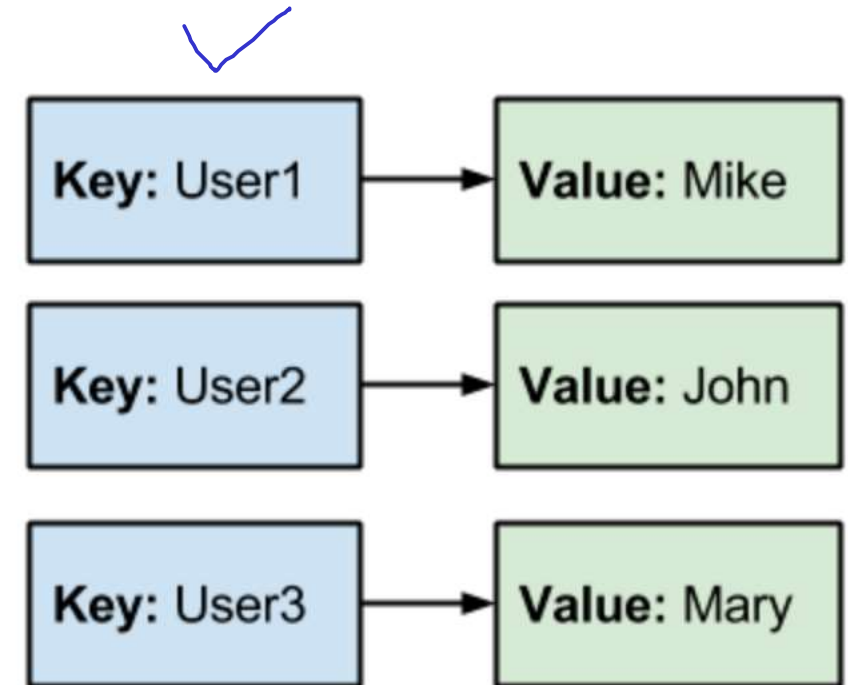
NoSQL Types

- Following are the types of NoSQL database based on how it stores the data
 - Key Value
 - Document DB
 - Column Based DB
 - Graph DB



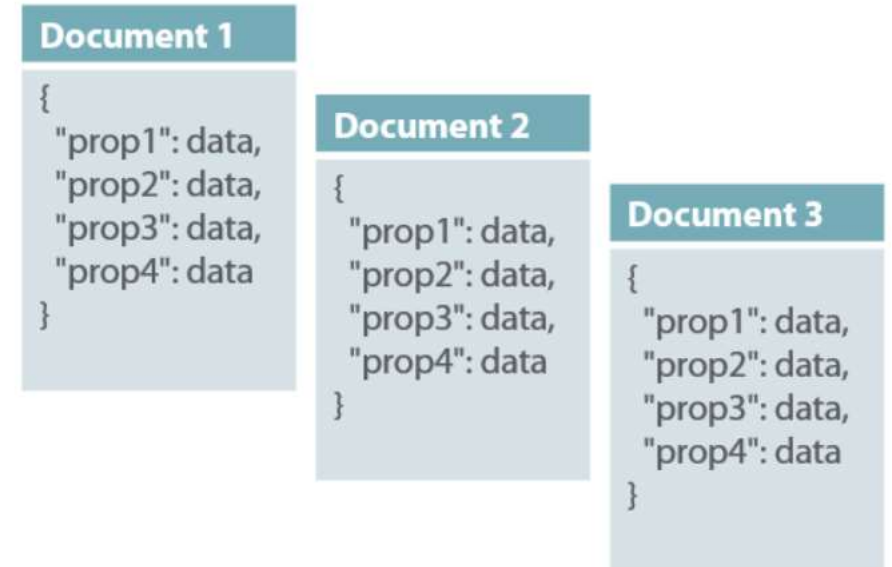
Key Value Database

- Data is stored as keys and values
- Provides **super fast queries** and ability to scale almost infinite data and **performance level**
- No relationships and weak/no schema
- E.g.
 - DynamoDB ✓
 - Redis ✓
 - Couchbase ✓



Document DB

- Data is stored as Documents
- Document: collection of key-value pairs with structure
- Great performance when interacting with documents
- E.g.
 - MongoDB
 - CouchDB ✓
 - RethinkDB ✓

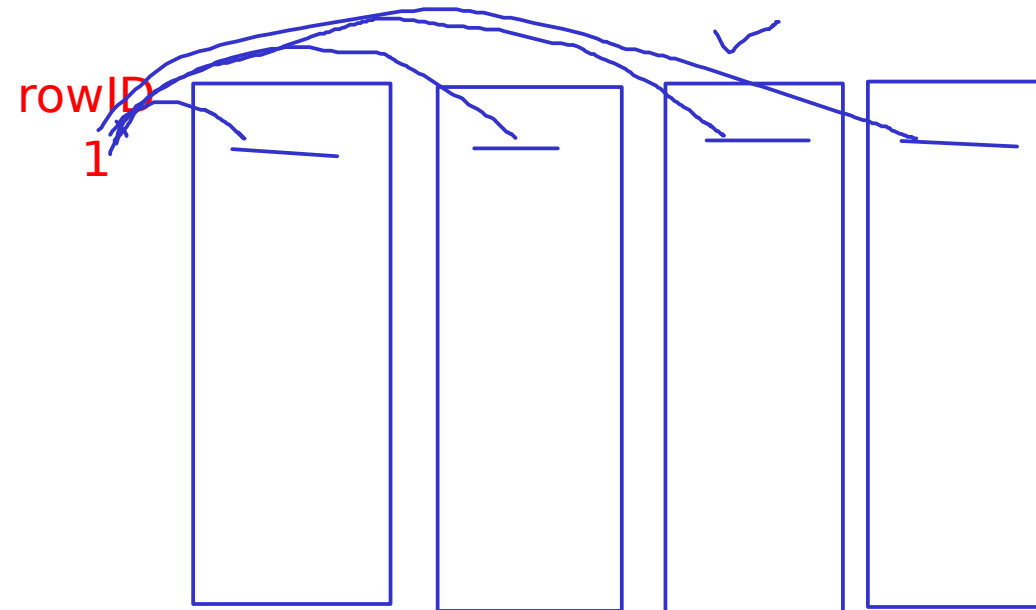


Column Based DB

50000X30

✓city
count pune=??

- Data is stored as columns rather than rows
- Fast queries for datasets
- Slower when looking at individual records
- Great for warehousing and analytics
- E.g.
 - Redshift
 - Cassandra
 - HBase
 - Vertica

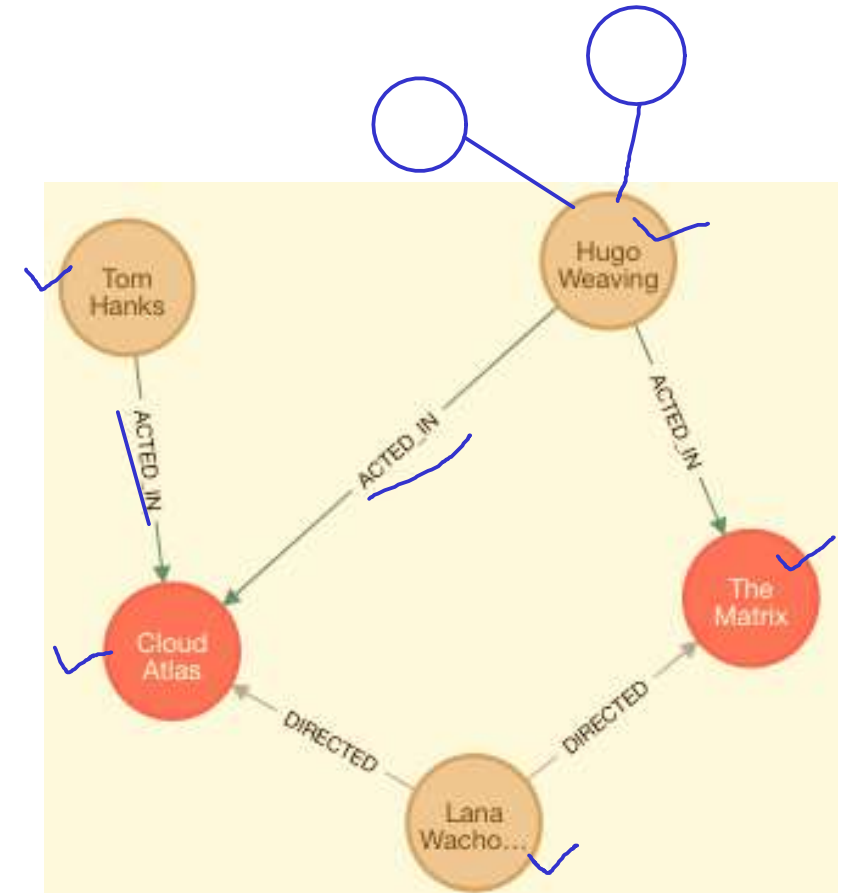


ID	Name
0001	Roffle
0002	Penny
0003	Winkie



Graph DB

- Designed for dynamic relationship
- Stores data as nodes and relationships between those nodes
- Ideal for human related data like social media
- E.g.
 - Neo4J ✓
 - Giraph
 - OrientDB



Advantages of NoSQL

■ High scalability

- This scaling up approach fails when the transaction rates and fast response requirements increase. In contrast to this, the new generation of NoSQL databases is designed to scale out (i.e. to expand horizontally using low-end commodity servers).

■ Manageability and administration

- NoSQL databases are designed to mostly work with automated repairs, distributed data, and simpler data models, leading to low manageability and administration.

■ Low cost

- NoSQL databases are typically designed to work with a cluster of cheap commodity servers, enabling the users to store and process more data at a low cost.

■ Flexible data models

- NoSQL databases have a very flexible data model, enabling them to work with any type of data; they don't comply with the rigid RDBMS data models. As a result, any application changes that involve updating the database schema can be easily implemented.



Disadvantages of NoSQL

- **Maturity**
 - Most NoSQL databases are pre-production versions with key features that are still to be implemented. Thus, when deciding on a NoSQL database, you should analyze the product properly to ensure the features are fully implemented and not still on the To-do list.
- **Support**
 - Support is one limitation that you need to consider. Most NoSQL databases are from start-ups which were open sourced. As a result, support is very minimal as compared to the enterprise software companies and may not have global reach or support resources.
- ✓ - **Limited Query Capabilities**
 - Since NoSQL databases are generally developed to meet the scaling requirement of the web-scale applications, they provide limited querying capabilities. A simple querying requirement may involve significant programming expertise.
- ✓ - **Administration**
 - Although NoSQL is designed to provide a no-admin solution, it still requires skill and effort for installing and maintaining the solution.
- **Expertise**
 - Since NoSQL is an evolving area, expertise on the technology is limited in the developer and administrator community.



SQL vs NoSQL

	SQL	NoSQL
Types	All types support <u>SQL</u> standard	Multiple types exists, such as <u>document stores</u> , <u>key value stores</u> , <u>column</u> databases, etc
History	Developed in <u>1970</u>	Developed in <u>2000s</u>
Examples	SQL Server, Oracle, MySQL	MongoDB, HBase, Cassandra
Data Storage Model	Data is stored in <u>rows and columns</u> in a table, where each column is of a specific type	The data model depends on the database type. It could be Key-value pairs, documents etc
Schemas	<u>Fixed structure</u> and schema	<u>Dynamic schema</u> . Structures can be accommodated
Scalability	<u>Scale up</u> approach is used	<u>Scale out</u> approach is used
Transactions	Supports <u>ACID</u> and transactions	Supports <u>partitioning</u> and <u>availability</u>
Consistency	Strong consistency ✓	Dependent on the product [Eventual Consistency] ✓
Support	High level of enterprise support ✓	Open source model ✓
Maturity	Have been around for a long time ✓	Some of them are mature; others are evolving ✓



NoSQL Scenarios

- When to use NoSQL ?

- ✓ ■ Large amount of data (TBs)
- ✓ ■ Many Read/Write operations
- ✓ ■ Economical scaling
- ✓ ■ Flexible Schema

- Examples

- ✓ ■ Social Media
- ✓ ■ Recordings
- ✓ ■ Geospatial analysis
- ✓ ■ Information processing

- ✓ ■ When NOT to use NoSQL ?

- ✓ ■ Need ACID transactions
- ✓ ■ Fixed multiple relations
- ✓ ■ Need joins
- ✓ ■ Need high consistency

- Examples

- ✓ ■ Financial transactions
- ✓ ■ Business operations

