Prof R. Madana Mohana

DATABASE MANAGEMENT SYSTEMS

NoSQL Databases

https://www.youtube.com/c/RASINENIMADANAMOHANA

NoSQL Databases: *Outline*

- Review of traditional Databases
- Need for NoSQL Databases

- **Columnar Databases**
- **Differences between SQL and NoSQL databases**

Review of Traditional Databases

Big Data

Prof R Madana Mohana

Traditional Databases

- **Traditional data** is the structured data that is being majorly maintained by all types of businesses starting from very small to big organizations.
- In a **traditional database system**, a centralized database architecture used to store and maintain the data in a fixed format or fields in a file.
- For managing and accessing the data **Structured Query Language (SQL)** is used.

Traditional Databases vs. Big Data

Traditional Database	Big Data
Traditional data is generated in enterprise level.	Big data is generated outside the enterprise level.
Its volume ranges from Gigabytes to Terabytes.	Its volume ranges from Petabytes to Zettabytes or Exabytes.
secondTraditional database system deals with structured data.	Big data system deals with structured, semi-structured, database, and unstructured data.
Traditional data is generated per hour or per day or more.	But big data is generated more frequently mainly per seconds.
Traditional data source is centralized and it is managed in centralized form.	Big data source is distributed and it is managed in distributed form.

Traditional Databases vs. Big Data

Traditional Database	Big Data
Data integration is very easy.	Data integration is very difficult.
Normal system configuration is capable to process traditional data.	High system configuration is required to process big data.
The size of the data is very small.	The size is more than the traditional data size.
Traditional data base tools are required to perform any data base operation.	Special kind of data base tools are required to perform any database schema-based operation.
Normal functions can manipulate data.	Special kind of functions can manipulate data.

Traditional Databases vs. Big Data

Traditional Database	Big Data
Its data model is strict schema based and it is static.	Its data model is a flat schema based and it is dynamic.
Traditional data is stable and inter relationship.	Big data is not stable and unknown relationship.
Traditional data is in manageable volume.	Big data is in huge volume which becomes unmanageable.
It is easy to manage and manipulate the data.	It is difficult to manage and manipulate the data.
Its data sources includes Enterprise Resource Planning (ERP) transaction data, Customer relationship management (CRM) transaction data, financial data, organizational data, web transaction data etc.	Its data sources includes social media, device data, sensor data, video, images, audio etc.

NoSQL: Introduction

Big Data

Prof R Madana Mohana

NoSQL

- **NoSQL** databases ("**not only SQL**") are non-tabular databases and store data differently than relational tables.
- **NoSQL** databases come in a variety of types based on their data model.
- The main types are *document*, *key-value*, *wide-column*, and *graph*.
- They provide flexible schemas and scale easily with large amounts of data and high user loads.

What is a NoSQL database?

- When people use the term "**NoSQL database**," they typically use it to refer to any non-relational database.
- Some say the term "**NoSQL**" stands for "**non SQL**" while others say it stands for "**not only SQL**."
- Either way, most agree that **NoSQL** databases are databases that store data in a format other than relational tables.

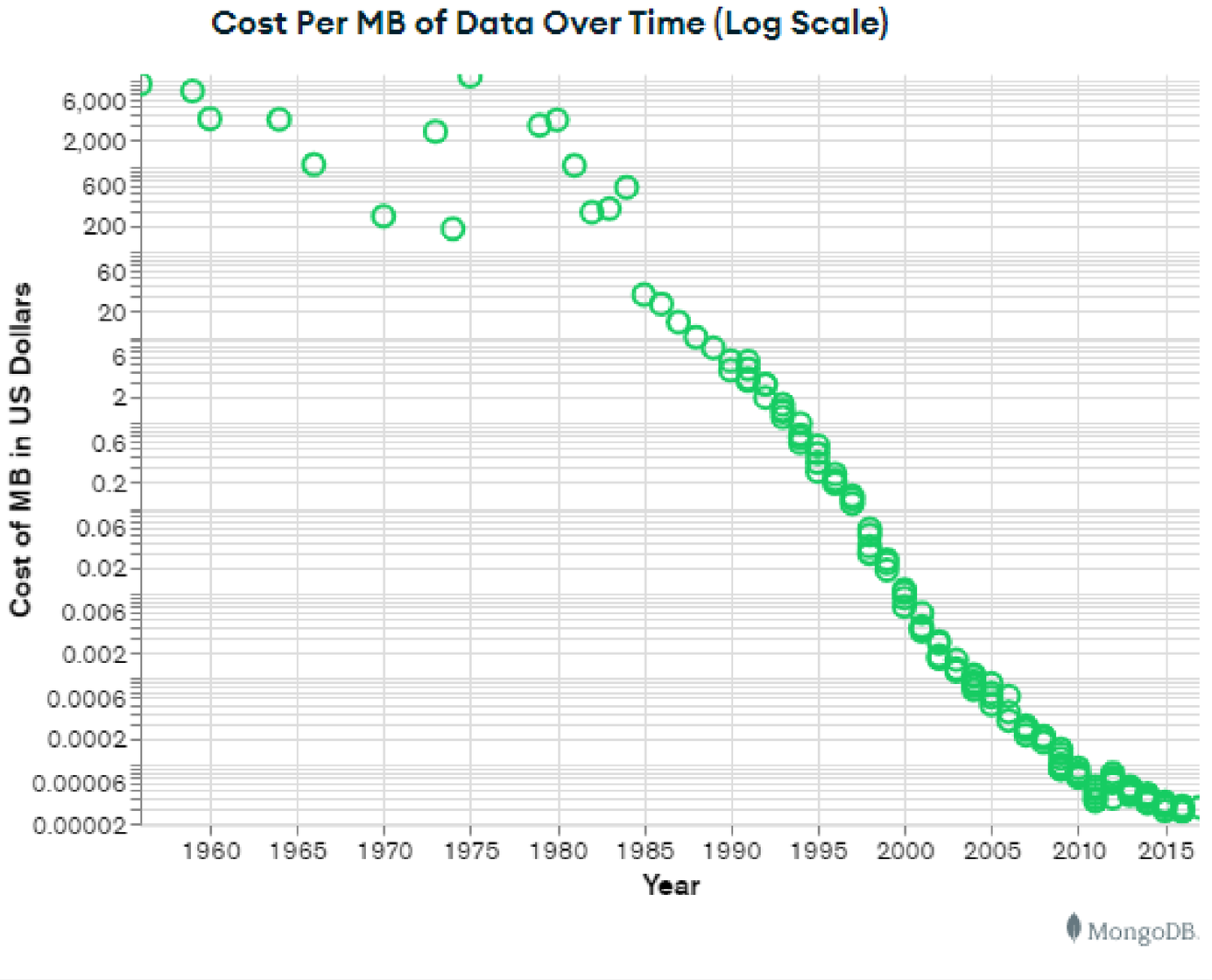
Brief History of NoSQL databases

- **NoSQL** databases emerged in the late **2000**s as the cost of storage dramatically decreased.
- Gone were the days of needing to create a complex, difficult-to-manage data model in order to avoid data duplication.
- Developers (rather than storage) were becoming the primary cost of software development, so **NoSQL** databases optimized for developer productivity.

Brief History of NoSQL databases

- **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.

Brief History of NoSQL databases



Brief History of NoSQL databases

- As storage costs rapidly decreased, the amount of data that applications needed to store and query increased.
- This data came in all shapes and sizes i.e., *structured*, *semi-structured*, and *polymorphic* and defining the schema in advance became nearly impossible.
- **NoSQL** databases allow developers to store huge amounts of *unstructured data*, giving them a lot of flexibility.

Brief History of NoSQL databases

- Additionally, the Agile Manifesto was rising in popularity, and software engineers were rethinking the way they developed software.
- They were recognizing the need to rapidly adapt to changing requirements.
- They needed the ability to iterate quickly and make changes throughout their software stack - all the way down to the database.
- **NoSQL** databases gave them this flexibility.

Brief History of NoSQL databases

- **Cloud computing** also rose in popularity, and developers began using *public clouds* to host their applications and data.
- They wanted the ability to distribute data across multiple servers and regions to make their applications resilient, to scale out instead of scale up, and to intelligently geo-place their data.
- Some **NoSQL** databases like **MongoDB** provide these capabilities.

Need for NoSQL Databases

Need for NoSQL Databases

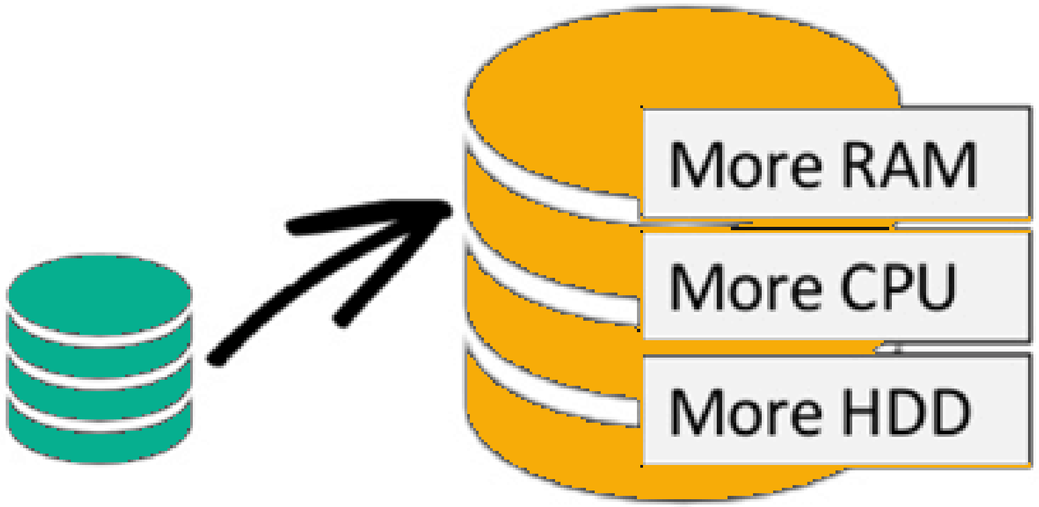
- The concept of **NoSQL** databases became popular with Internet giants like **Google**, **Facebook**, **Amazon**, etc. who deal with huge volumes of data.
- The system *response time* becomes slow when we use **RDBMS** for massive volumes of data.
- To resolve this problem, we could "**scale up**" our systems by upgrading our existing hardware.
- This process is expensive.

Need for NoSQL Databases

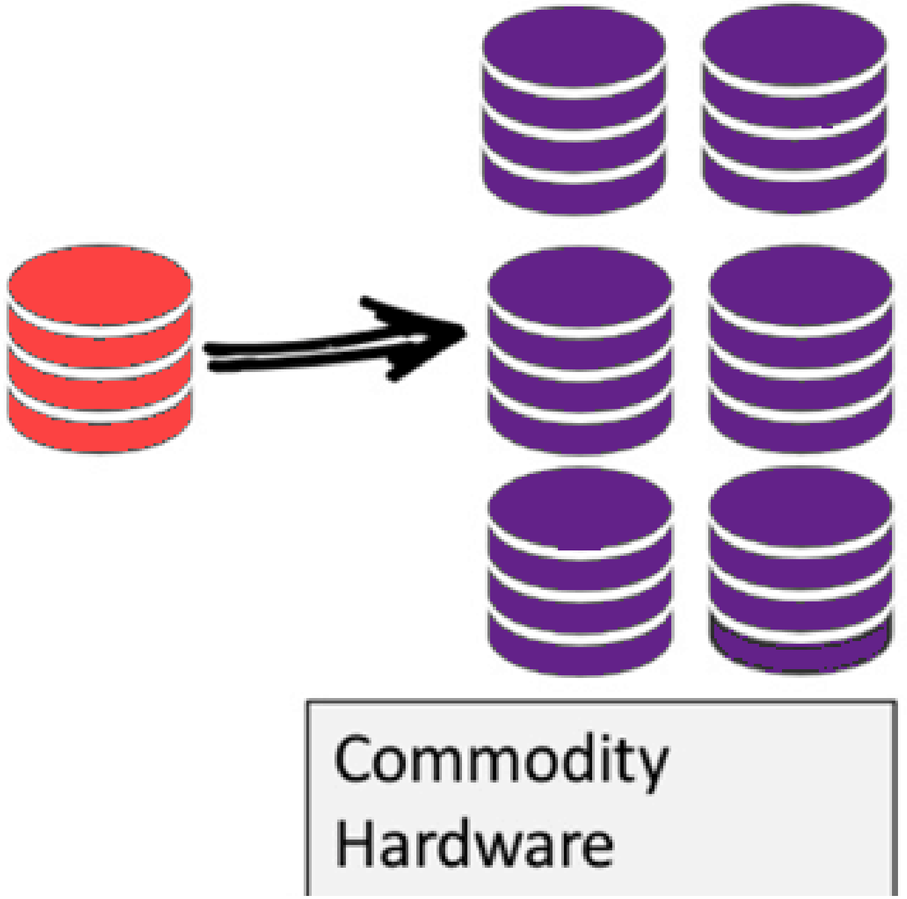
- The alternative for this issue is to distribute database load on multiple hosts whenever the load increases.
- This method is known as "**scaling out**."
- **NoSQL** database is non-relational, so it scales out better than relational databases as they are designed with web applications in mind.

Need for NoSQL Databases

Scale-Up (*vertical* scaling):



Scale-Out (*horizontal* scaling):



NoSQL database features

- Each **NoSQL** database has its own unique features.
- At a high level, many **NoSQL** databases have the following features:
  - `Non-relational`
  - `Flexible schemas`
  - `Horizontal scaling`
  - `Fast queries due to the data model`
  - `Ease of use for developers`

NoSQL database features

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, and **ACID** etc.

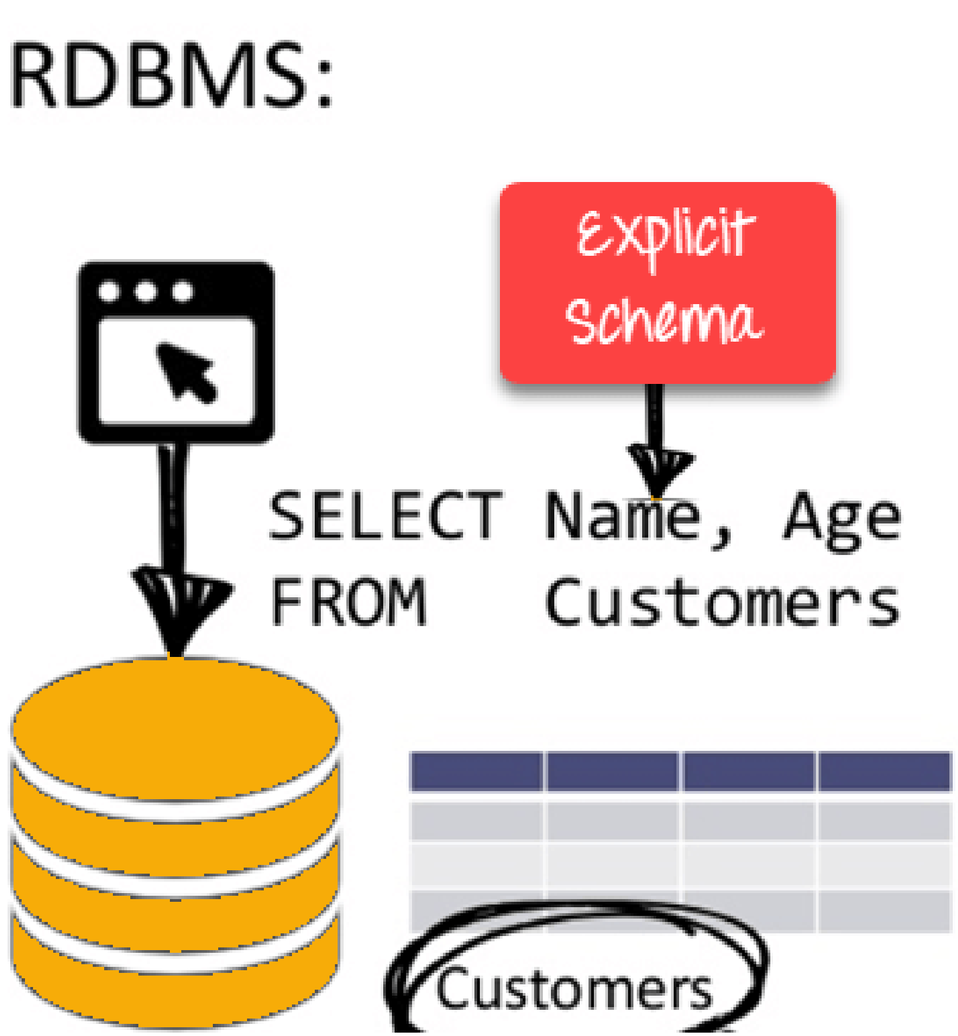
NoSQL database features

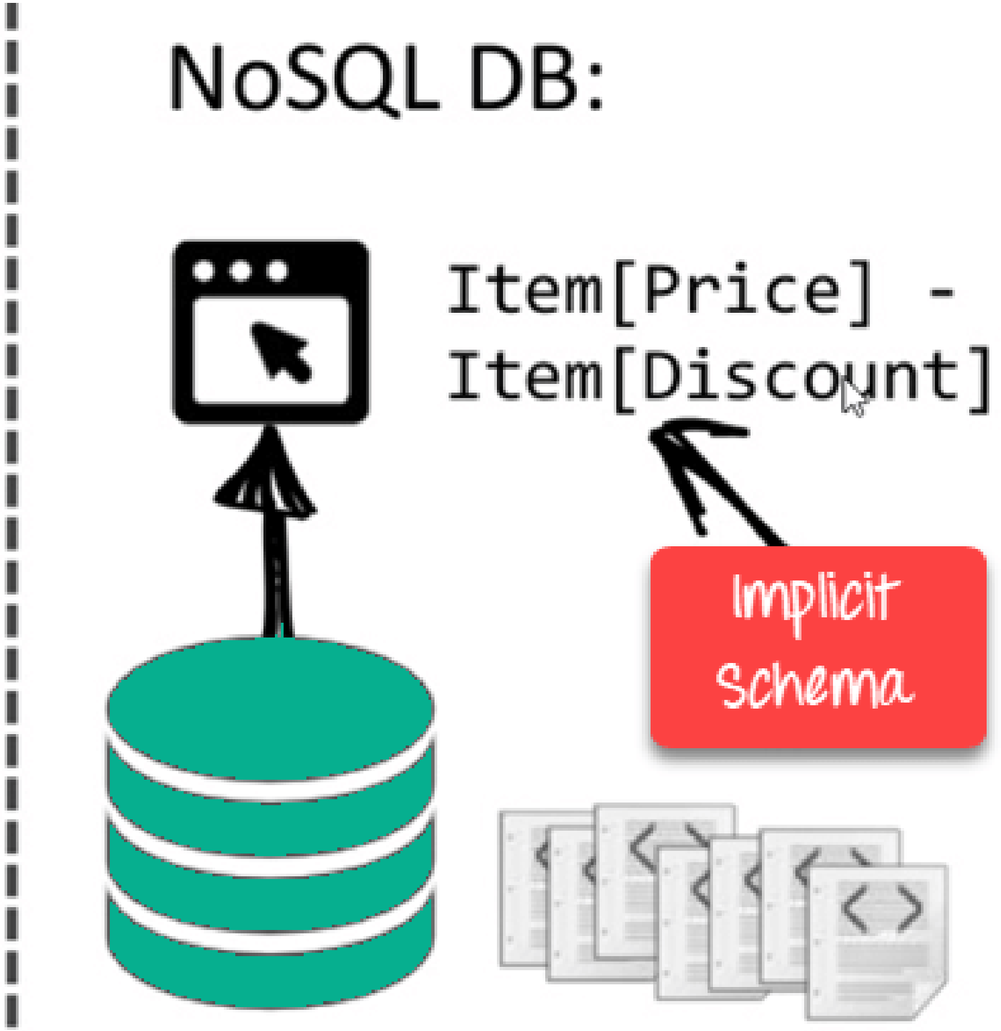
Flexible schemas:

- **NoSQL** databases typically have very flexible schemas.
- A flexible schema allows you to easily make changes to our database as requirements change.
- We can iterate quickly and continuously integrate new application features to provide value to our users faster.

NoSQL database features

Flexible schemas:





NoSQL database features

Horizontal scaling:

- Most **SQL** databases require us to scale-up vertically (migrate to a larger, more expensive server) when we exceed the capacity requirements of our current server.
- Conversely, most **NoSQL** databases allow us to scale-out horizontally, meaning we can add cheaper commodity servers whenever we need to.

NoSQL database features

Fast queries:

- Queries in **NoSQL** databases can be faster than **SQL** databases.
- *Why Data in SQL databases is typically normalized*, so queries for a single object or entity require us to join data from multiple tables.
- As our tables grow in size, the joins can become expensive.

NoSQL database features

Fast queries:

- However, data in **NoSQL** databases is typically stored in a way that is optimized for queries.
- The rule of thumb when you use **MongoDB** is data that is accessed together should be stored together.
- Queries typically do not require joins, so the queries are very fast.

NoSQL database features

Easy for developers:

- Some **NoSQL** databases like **MongoDB** map their data structures to those of popular programming languages.
- This mapping allows developers to store their data in the same way that they use it in their application code.
- While it may seem like a trivial advantage, this mapping can allow developers to write less code, leading to faster development time and fewer bugs.

Columnar Databases

Big Data

Prof R Madana Mohana

Columnar Databases

- The **Columnar Data Model** of **NoSQL** is important.
- **NoSQL** databases are different from **SQL** databases.
- This is because it uses a data model that has a different structure than the previously followed row-and-column table model used with **relational database management systems (RDBMS)**.
- **NoSQ**L databases are a flexible schema model which is designed to scale horizontally across many servers and is used in large volumes of data.

Columnar Data Model of NoSQL

- Basically, the relational database stores data in rows and also reads the data row by row, ***column store*** is organized as a set of columns.
- So if someone wants to run analytics on a small number of columns, one can read those columns directly without consuming memory with the unwanted data.
- **Columns** are somehow are of the same type and gain from more efficient compression, which makes reads faster than before.
- Examples of Columnar Data Model: C**assandra** and **Apache Hadoop Hbas**e.

Working of Columnar Data Model

- In **Columnar Data Model** instead of organizing information into rows, it does in columns.
- This makes them function the same way that tables work in relational databases.
- This type of data model is much more flexible obviously because it is a type of **NoSQL** database.

Working of Columnar Data Model

The below example will help in understanding the Columnar data model:

Row-Oriented Table:

S.No.	Name	Course	Branch	ID
01.	Tanmay	B-Tech	Computer	2
02.	Abhishek	B-Tech	Electronics	5
03.	Samriddha	B-Tech	IT	7
04.	Aditi	B-Tech	E & TC	8

Working of Columnar Data Model

Column – Oriented Table:

S.No.	Name	ID
01.	Tanmay	2
02.	Abhishek	5
03.	Samriddha	7
04.	Aditi	8

S.No.	Course	ID
01.	B-Tech	2
02.	B-Tech	5
03.	B-Tech	7
04.	B-Tech	8

S.No.	Branch	ID
01.	Computer	2
02.	Electronics	5
03.	IT	7
04.	E & TC	8

Working of Columnar Data Model

- **Columnar Data Model** uses the concept of keyspace, which is like a schema in relational models.

Advantages of Columnar Data Model

Well structured: Since these data models are good at compression so these are very structured or well organized in terms of storage.

Flexibility: A large amount of flexibility as it is not necessary for the columns to look like each other, which means one can add new and different columns without disrupting the whole database.

Advantages of Columnar Data Model

Aggregation queries are fast: The most important thing is aggregation queries are quite fast because a majority of the information is stored in a column. An example would be Adding up the total number of students enrolled in one year.

Scalability: It can be spread across large clusters of machines, even numbering in thousands.

Load Times: Since one can easily load a row table in a few seconds so load times are nearly excellent.

Disadvantages of Columnar Data Model

Designing indexing Schema: To design an effective and working schema is too difficult and very time-consuming.

Suboptimal data loading: incremental data loading is suboptimal and must be avoided, but this might not be an issue for some users.

Security vulnerabilities: If security is one of the priorities then it must be known that the Columnar data model lacks inbuilt security features in this case, one must look into relational databases.

Disadvantages of Columnar Data Model

Online Transaction Processing (OLTP): Online Transaction Processing (OLTP) applications are also not compatible with columnar data models because of the way data is stored.

Applications of Columnar Data Model

- **Columnar Data Model** is very much used in various Blogging Platforms.
- It is used in Content management systems like **WordPress**, **Joomla**, etc.
- It is used in Systems that maintain counters.
- It is used in Systems that require heavy write requests.
- It is used in Services that have expiring usage.

Column-oriented vs. Row-oriented Database

Operation	Column-Oriented Database	Row-Oriented Database
Aggregate Calulation of Single Column e.g. sum(price)	✔ fast	slow
Compression	✔ Higher. As stores similar data together	-
Retrieval of a few columns from a table with many columns	✔ Faster	has to skip over unnecessary data
Insertion/Updating of single new record	Slow	✔ Fast
Retrieval of a single record	Slow	✔ Fast

Differences between SQL and NoSQL databases

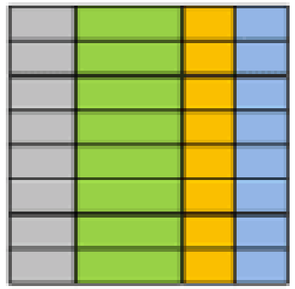
Big Data

Prof R Madana Mohana

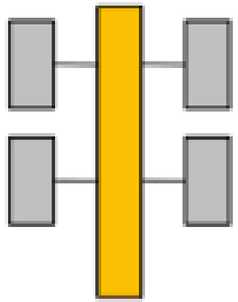
Differences between SQL and NoSQL databases

SQL Database

Relational

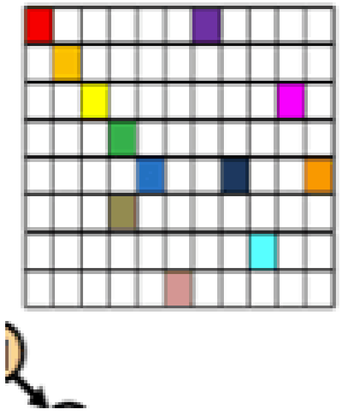


Analytical (OLAP)

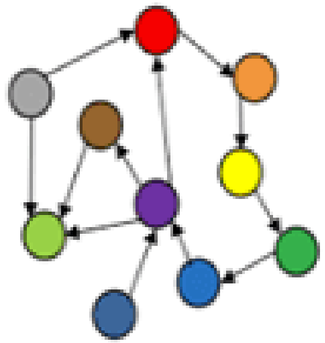


NoSQL Database

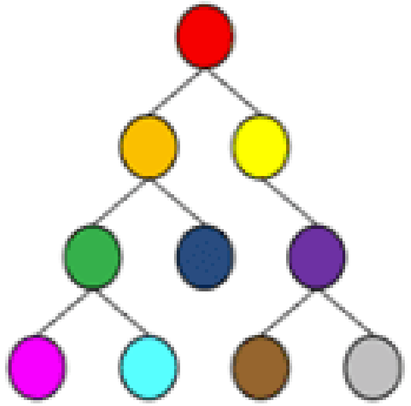
Column-Family



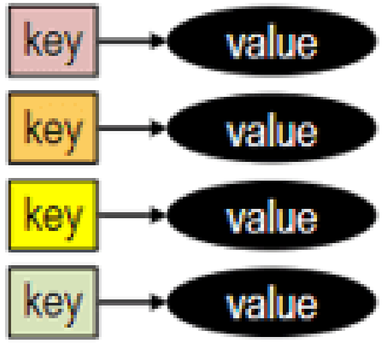
Graph



Document



Key-Value



Differences between SQL and NoSQL databases

	SQL Databases	NoSQL Databases
Data Storage Model	Tables with fixed rows and columns.	**Document:** JSON documents, **Key-value:** key-value pairs, **Wide-column:** tables with rows and dynamic columns, **Graph:** nodes and edges.
Development History	Developed in the 1970s with a focus on reducing data duplication.	Developed in the late 2000s with a focus on scaling and allowing for rapid application change driven by agile and DevOps practices.
Examples	Oracle, MySQL, Microsoft SQL Server, and PostgreSQL	**Document:** MongoDB and CouchDB, **Key-value:** Redis and DynamoDB, **Wide-column**: Cassandra and HBase, **Graph:** Neo4j and Amazon Neptune

Differences between SQL and NoSQL databases

	SQL Databases	NoSQL Databases
Primary Purpose	General purpose	**Document:** general purpose, **Key-value:** large amounts of data with simple lookup queries, **Wide-column:** large amounts of data with predictable query patterns, **Graph:** analyzing and traversing relationships between connected data.
Schemas	Rigid (not flexible)	Flexible
Scaling	Vertical (scale-up with a larger server)	Horizontal (scale-out across commodity servers)

Differences between SQL and NoSQL databases

	SQL Databases	NoSQL Databases
Multi-Record ACID Transactions	Supported	Most do not support multi-record ACID transactions. However, some - like **MongoDB** -do.
Joins	Typically required	Typically not required
Data to Object Mapping	Requires ORM (object-relational mapping)	Many do not require ORMs. **MongoDB** documents map directly to data structures in most popular programming languages.

Types of NoSQL databases

Types of NoSQL databases

Over time, *four major types* of **NoSQL** databases emerged:

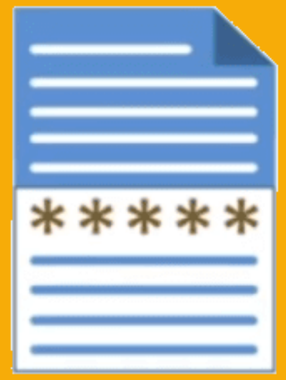
1. Document databases
2. Key-value databases
3. Wide-column stores (column-based), and
4. Graph databases

Types of NoSQL databases

Key Value

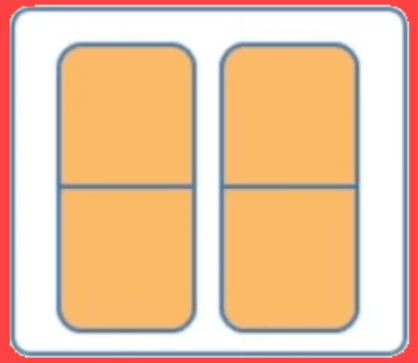
Example:
Riak, Tokyo Cabinet, Redis server, Memcached, Scalaris

Document-Based



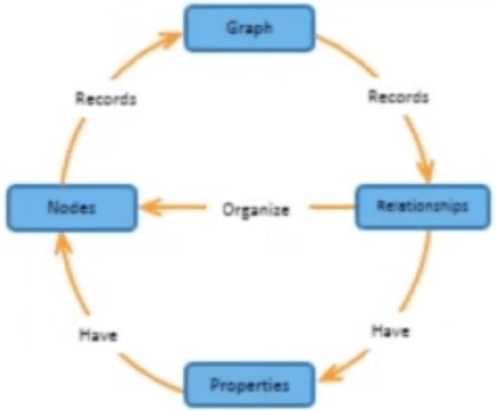
Example:
MongoDB, CouchDB, OrientDB, RavenDB

Column-Based



Example:
BigTable, Cassandra, Hbase, Hypertable

Graph-Based



Example:
Neo4J, InfoGrid, Infinite Graph, Flock DB

Types of NoSQL databases

Document databases:

- **Document databases** store data in documents similar to **JSON** (**JavaScript Object Notation**) objects.
- Each document contains pairs of fields and values.
- The values can typically be a variety of types including things like strings, numbers, booleans, arrays, or objects.

Types of NoSQL databases

Document databases:

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

Document 1

```
{
  "prop1": data,
  "prop2": data,
  "prop3": data,
  "prop4": data
}
```

Document 2

```
{
  "prop1": data,
  "prop2": data,
  "prop3": data,
  "prop4": data
}
```

Document 3

```
{
  "prop1": data,
  "prop2": data,
  "prop3": data,
  "prop4": data
}
```

Fig. *Relational Databases Vs. Document databases*

Types of NoSQL databases

Key-value databases:

- **Key-value databases** are a simpler type of database where each item contains ***keys*** and ***values***.
- Data is stored in **key/value pairs**. It is designed in such a way to handle lots of data and heavy load.
- **Key-value pair** storage databases store data as a hash table where each key is unique, and the value can be a **JSON**, **BLOB**(Binary Large Objects), string, etc.

Types of NoSQL databases

Key-value databases: *Example*

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

Types of NoSQL databases

Wide-column stores (or Column-oriented):

- **Wide-column store**s store data in tables, rows, and dynamic columns.
- **Column-oriented databases** work on columns and are based on **BigTable** paper by **Google**.
- Every column is treated separately.
- Values of single column databases are stored contiguously.

Types of NoSQL databases

Wide-column stores (or Column-oriented):

- They deliver high performance on aggregation queries like **SUM**, **COUNT**, **AVG**, **MIN** etc. as the data is readily available in a column.
- **Column-based NoSQL** databases are widely used to manage data warehouses, business intelligence, CRM, Library card catalogs, HBase, Cassandra.
- **HBase**, Hypertable are **NoSQL** query examples of column based database.

Types of NoSQL databases

Wide-column stores (or Column-oriented):

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

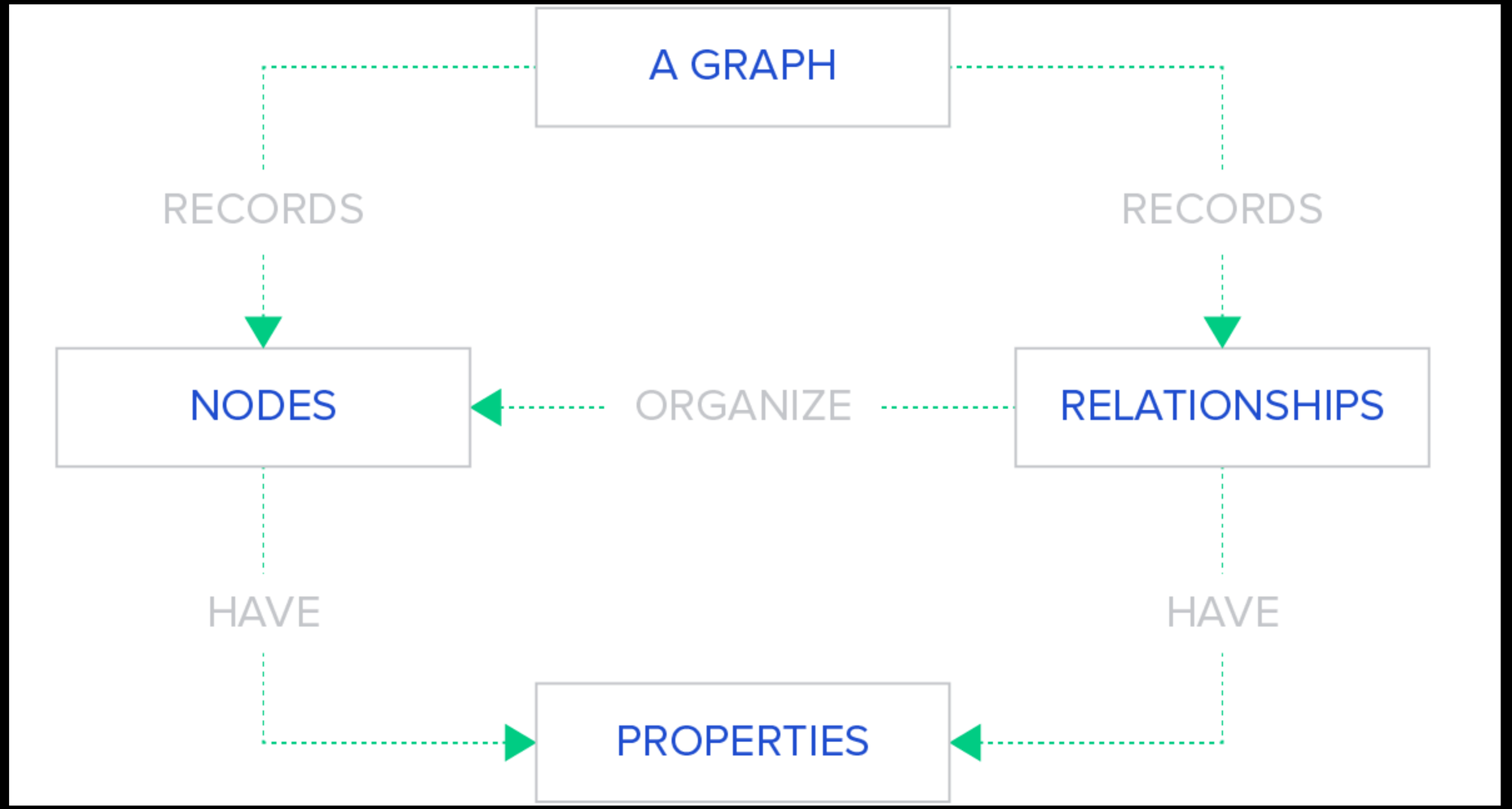
Types of NoSQL databases

Graph databases:

- **Graph databases** store data in nodes and edges.
- 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.
- Nodes typically store information about people, places, and things, while edges store information about the relationships between the nodes.
- Every node and edge has a unique identifier.

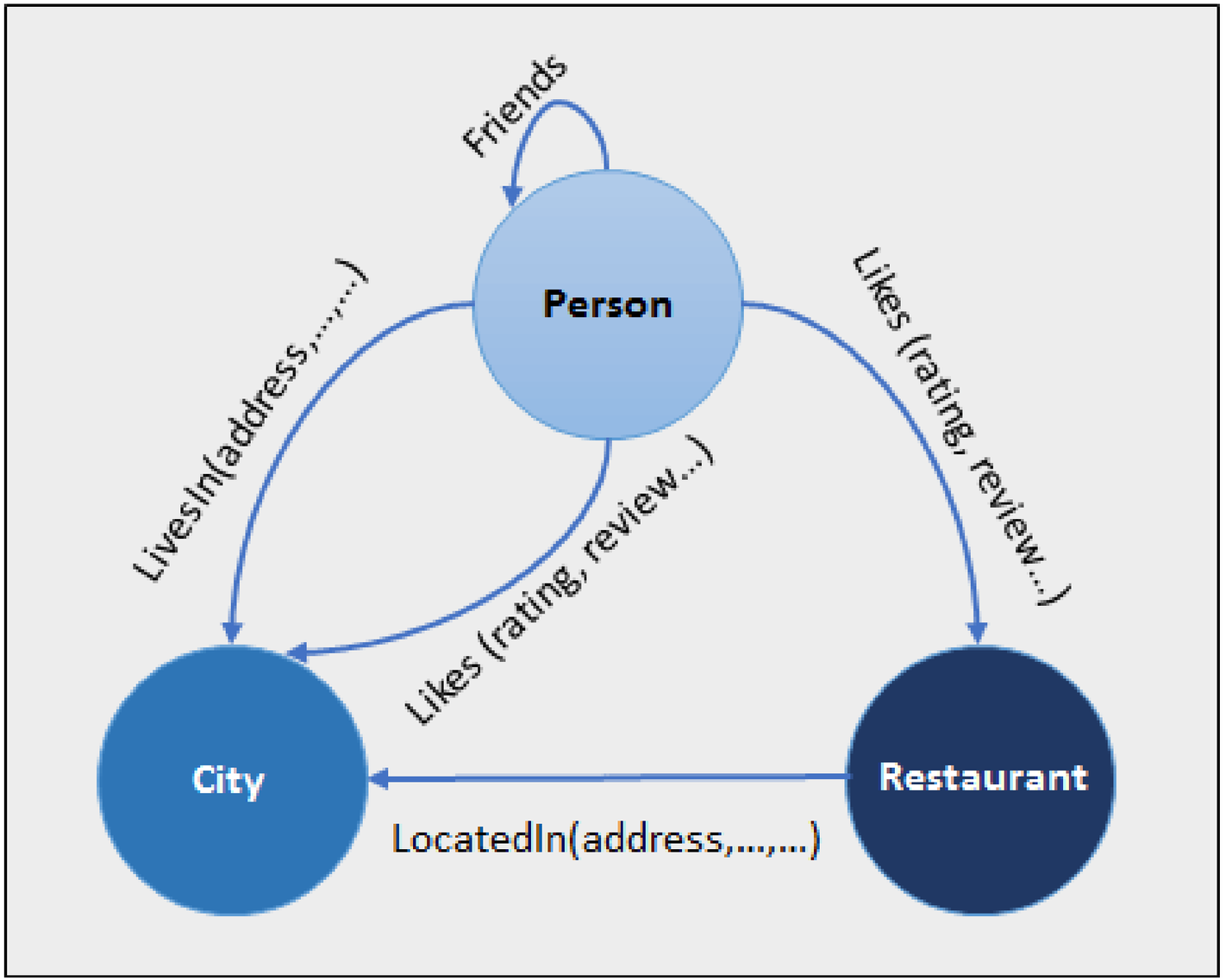
Types of NoSQL databases

Graph databases:



Types of NoSQL databases

Graph databases:



Types of NoSQL databases

Graph databases:

- Compared to a **relational database** where tables are loosely connected, a **Graph database** is a multi-relational in nature.
- Traversing relationship is fast as they are already captured into the **DB**, and there is no need to calculate them.
- **Graph base database** mostly used for social networks, logistics, spatial data.
- **Neo4J, Infinite Graph, OrientDB, FlockDB** are some popular **graph-based database**s.

Advantages of NoSQL

Advantages of NoSQL

- Can be used as Primary or Analytic Data Source.
- Big Data Capability.
- No Single Point of Failure.
- Easy Replication.
- No Need for Separate Caching Layer.
- It provides fast performance and horizontal scalability.
- Can handle structured, semi-structured, and unstructured data with equal effect.

Advantages of NoSQL

- Object-oriented programming which is easy to use and flexible.
- **NoSQL** databases don’t need a dedicated high-performance server.
- Support Key Developer Languages and Platforms.
- Simple to implement than using **RDBMS.**
- It can serve as the primary data source for online applications.

Advantages of NoSQL

- Handles **big data** which manages data velocity, variety, volume, and complexity.
- Excels at distributed database and multi-data center operations.
- Eliminates the need for a specific caching layer to store data.
- Offers a flexible schema design which can easily be altered without downtime or service disruption.

Disadvantages of NoSQL

Disadvantages of NoSQL

- No standardization rules.
- Limited query capabilities.
- **RDBMS** databases and tools are comparatively mature.
- It does not offer any traditional database capabilities, like consistency when multiple transactions are performed simultaneously.
- When the volume of data increases it is difficult to maintain unique values as keys become difficult.
- Doesn’t work as well with relational data.
- The learning curve is stiff for new developers.
- Open source options so not so popular for enterprises.

Reference Videos

Playlist: NoSQL Databases (Data Science / Big Data Analytics)

https://youtube.com/playlist?list=PLENQMW_c1dimQRQWd8dJynZqVaBHBVrUl&si=iYQSdsA0RL9DrO2l