MODULE III

CHAPTER 3

NoSQL

University Prescribed Syllabus w.e.f AcademicYear 2022-2023

3.1 Introduction to NoSQL, NoSQL Business Drivers

3.2 NoSQL Data Architecture Patterns: Key-value stores, Graph stores, Column family (Bigtable)stores, Document stores, Variations of NoSQL architectural patterns, NoSQL Case Study

3.3 NoSQL solution for big data, Understanding the types of big data problems; Analyzing big data with a shared-nothing architecture; Choosing distribution models: master-slave versus peer-to-peer; NoSQL systems to handle big data problems.

Scanned by CamScanner

Scanned by CamScanner

3.1 INTRODUCTION TO NoSQL, NoSQL BUSINESS DRIVERS

3.1.1 Introduction to NoSQL

A database is a systematic collection of data. And a database management system supports storage and manipulation of data which makes data management easy. For example, an online telephone directory uses a database to store data of people like phone numbers and other contact details that can be used by service provider to manage billing client related issues and handle fall data etc. That means A database management system provides the mechanism to store and retrieve the data.

There are different kinds of management systems:

RDBMS	OLAP	NoSQL
(Relational Database Management System)	(Online Analytical Processing)	(Not only SQL)

NoSQL refers to all databases and data stores that are not based on the relational database management system or RDBMS principles. NoSQL are the new set of databases that has emerged recent past as an alternative solution to relational databases.

Carl Strozzi introduced the term NoSQL to name his file-based database in 1998.

NoSQL does not represent single product or technology but it represents a group products and various related data concepts for storage and management. NoSQL is an approach to database management that can accommodate a wide variety of data models including key-value, document, column and graph formats. NoSQL database generally means that it is non-relational, distributed, flexible and scalable.So, we can bind it up as-

- NoSQL an approach to database design that provides flexible schemas for the storage and retrieval of data beyond the traditional table structures found in relational databases.
- It relates to large data sets accessed and manipulated on a Web scale.

3.1.2 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.

Scanned by CamScanner

- 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.

3.1.3 Why NoSQL?

The concept of NoSQL databases became popular with internet giants like Google, Facebook, Amazon etc. Who deals with huge volumes of data the system response time becomes slow when we use RDBMS for massive volumes of data so to resolve this problem, we could scale up our system by upgrading our existing hardware but this process is an expensive. So alternative for this issue is to distribute database load on multiple hosts whenever the load increases this method known as scaling out.

NoSQL databases are non-relational so they scale-out better than relational databases. As they designed with the web applications in mind. Now NoSQL database is exactly type pf database that can handle all sorts of semi structured data, unstructured data, rapidly changing data and bigdata. So, to resolve the problems related to large volume and semi structured data. NoSQL databases have emerged.

3.1.4 CAP Theorem

GQ. What is CAP Theorem? How it is applicable to NOSQL systems? **(4 Marks)**

It plays important role in NoSQL databases. CAP theorem is also called brewer's theorem which states that it is impossible for a distributed data store to offer more than two out of three guarantees:

So basically, some NoSQL databases offer consistency and partition tolerance. While some offer availability and partition tolerance. But partition tolerance is common as NoSQL databases are distributed in nature so based on requirement, we can choose NoSQL database has to be used. Different types of NoSQL databases are available based on data models.

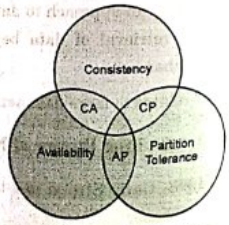


Fig 3.1.1 :CAP Property

Consistency

- This means that the data in the database remains consistent after the execution of an operation.
- For example, after an update operation all clients see the same data.

Availability

- This means that the system is always on (service guarantee availability), no downtime.

Partition Tolerance

- This means that the system continues to function even the communication among the servers is unreliable, i.e., the servers may be partitioned into multiple groups that cannot communicate with one another.

In theoretically it is impossible to fulfil all 3 requirements. CAP provides the basic requirements for a distributed system to follow 2 of the 3 requirements. Therefore, all the current NoSQL database follow the different combinations of the C, A, P from the CAP theorem.

Here is the brief description of **three combinations CA, CP, AP :**

- CA - Single site cluster, therefore all nodes are always in contact. When a partition occurs, the system blocks.
- CP -Some data may not be accessible, but the rest is still consistent/accurate.
- AP - System is still available under partitioning, but some of the data returned may be inaccurate.

The use of the word consistency in CAP and its use in ACID do not refer to the same identical concept.

In CAP, the term consistency refers to the consistency of the values in different copies of the same data item in a replicated distributed system. In ACID, it refers to the fact that a transaction will not violate the integrity constraints specified on the database schema

3.1.5 Characteristics / Features of NoSQL

UQ. Describe characteristics of a NoSQL database. **MU- Dec 17. 10 Marks**

1. Non-relational

- NoSQL databases never follow the relational model

Scanned by CamScanner

- Never provide tables with flat fixed-column records
- Work with self-contained aggregates or BLOBs
- Doesn't require object-relational mapping and data normalization
- No complex features like query languages, query planners, referential integrity joins, ACID

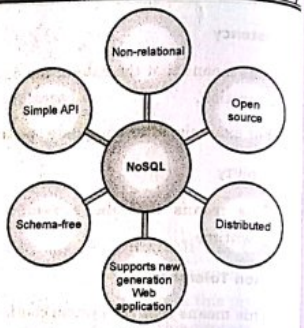


Fig. 3.1.2

2. Open-source

NoSQL databases don't require expensive licensing fees and can run on inexpensive hardware, rendering their deployment cost-effective.

3. 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

4. Simple API

- Offers easy to use interfaces for storage and querying data provided
- APIs allow low-level data manipulation & selection methods
- Text-based protocols mostly used with HTTP REST with JSON
- Mostly used no standard based query language
- Web-enabled databases running as internet-facing services

5. Distributed

- Multiple NoSQL databases can be executed in a distributed fashion
- Offers auto-scaling and fail-over capabilities
- Often ACID concept can be sacrificed for scalability and throughput
- Mostly no synchronous replication between distributed nodes Asynchronous MultiMaster Replication, peer-to-peer, HDFS Replication
- Only providing eventual consistency

- Shared Nothing Architecture. This enables less coordination and higher distribution.

3.1.6 Advantages and Disadvantagesof NoSQL

Advantages of NoSQL	Disadvantages of NoSQL
1. Scale(horizontal)	1. ACID transactions
2. SQL databases are vertically scalable. This means that you can increase the load on a single server by increasing things like RAM, CPU or SSD. But on the other hand, NoSQL databases are horizontally scalable. This means that you handle more traffic by sharding, or adding more servers in your NoSQL database	2. Cannot use SQL
3. Simple data model (fewer joins)	3. Cannot perform searches
4. Streaming/ volume	4. Data loss
5. Reliability	5. No referential integrity
6. Schema-less (no modelling or prototyping)	6. Lack of availability of expertise
7. Rapid development	
8. Flexible as it can handle semi-structured, unstructured and structured data.	
9. Cheaper than relational database	
10. Creates a caching layer	
11. Wide data type variety	
12. Uses large binary objects for storing large data	
13. Bulk upload	
14. Lower administration	
15. Distributed storage	
16. Real-time analysis	

Scanned by CamScanner

3.1.7 Difference between RDBMS and NoSQL

UQ. Differentiate between a RDBMS and NoSQL database. **MU- Dec-18, 10 Marks**

Sr. No.	RDBMS	NoSQL
1.	Have fixed or static or predefined schema	Have dynamic schema
2.	Not suited for hierarchical data storage	Best suited for hierarchical data storage
3.	Vertically scalable	Horizontally scalable
4.	Follow ACID property	Follows CAP (consistency, availability, partition tolerance)
5.	Relational Database supports transactions (also complex transactions with joins).	NoSQL databases don't support transactions (support only simple transactions).
6.	Relational database manages only structured data.	NoSQL database can manage structured, unstructured and semi-structured data.
7.	Relational databases have a single point of failure with failover.	NoSQL databases have no single point of failure.
8.	Relational Database supports a powerful query language.	NoSQL Database supports a very simple query language.
9.	It gives only read scalability.	It gives both read and write scalability.
10.	Transactions written in one location.	Transactions written in many locations.
11.	It supports complex transactions.	It supports simple transactions.
12.	It is used to handle data coming in low velocity.	It is used to handle data coming in high velocity.
13.	Examples- MySQL, Oracle, Sqlite, PostgreSQL and MS-SQL etc.	Examples- MongoDB, BigTable, Redis, RavenDB, Cassandra, Hbase, Neo4j, CouchDB etc.

Scanned by CamScanner

3.1.8 NoSQL Business Drivers

The scientist-philosopher Thomas Kuhn coined the term paradigm shift to identify a recurring process. He observed that in science, where innovative ideas came in bursts and impacted the world in nonlinear ways. We'll use Kuhn's concept of the paradigm shift as a way to think about and explain the NoSQL movement and the changes in thought patterns, architectures, and methods emerging today.

Many organizations are supporting to the single-CPU relational systems that have fulfilled the needs of theirorganizations as per the requirements.

Businesses have initiated the value in fast catching and examining huge quantityof adjustable data and making direct changes in their businesses based on the data that they obtain. As all of thesedrivers applies burden on single-processor relational model, its basis suits less steady and in time no extendedencounters the organization's needs.

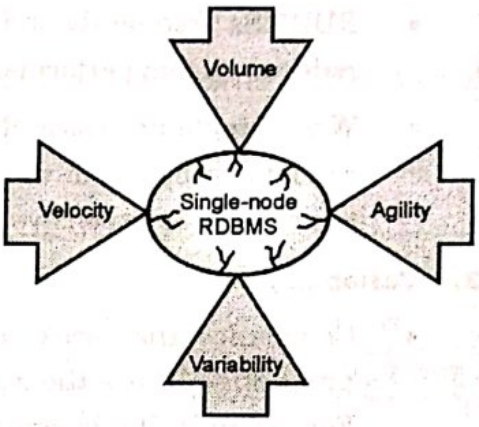


Fig. 3.1.3

Fig. 3.1.3 In this, we see how the business drivers volume, speed, variability, and agility create pressure on a single CPU system, resulting in cracks. Volume and velocity state to the capability to handle and managethe big datasets that appears early. Variability states to how various data types do not fit within the structured tables,and agility states to how much fast an organization replies to the business modification.

There are 4 major business drivers for NoSQL as:

(1) Volume	(2) Velocity	(3) Variability	(4) Agility.

1. Volume

- Undoubtedly, the main factor forcing organizations to look at alternatives to their existing RDBMS is to investigate big data using clusters of commodity processors.
- Until around 2005, performance problems were eliminated by purchasing faster processors. Over time, the ability to speed up the process is no longer an option.
- As the chip density increases, the heat can no longer dissolve rapidly enough without chip overheating. This phenomenon, known as the power wall, forced system designers to shift their focus from increasing speeds on a single chip to using more processors working together.
- The need to scale out (also known as horizontal scaling), rather than scale up (faster processors), moved organizations from serial to parallel processing where data problems are split into separate paths and sent to separate processors to divide and conquer the work.

Scanned by CamScanner

2. Velocity

- While large data issues are considered for many organizations moving away from RDBMSs, the ability of a single processor system to read and write data quickly is also important.
- Many single-processor RDBMS cannot meet the demand for online queries on databases created by real-time insert and public-facing websites.
- RDBMSs frequently index multiple columns of each new row, a process that reduces system performance.
- When single-processor RDBMSs are used as a back-end in front of a web store, random outbursts in web traffic reduce the response for everyone, and tuning the system can be expensive when both high read and write throughput is required.

3. Variability

- Companies that seek to capture and report exceptional data conflicts when attempting to use the rigorous database schema structure imposed by RDBMSs. For example, if a business unit wants to capture some custom fields for a specific customer, it must store this information even if it does not apply to all customer rows in the database.
- Adding new columns to RDBMS requires shutting down the system and running the ALTER TABLE command. When the database is large, this process can affect the availability of the system, costing time and money.

4. Agility

- Agility is ability to accept change easily and quickly.
- Among the variety of agility dimensions such as model agility (ease and speed of changing data modules), operational agility (ease models), operational agility (ease and speed of changing operational aspects), and programming agility (ease and speed of application development) – one of the most important is the ability to quickly and seamlessly scale an application to accommodate large amounts of data, users and connections.
- The most complex part of building applications using RDBMS is the process of putting data into and getting data out of the database.
- If your data has nested and repeated subgroups of data structures, you need to include an object-relational mapping layer.
- The responsibility of this layer is to generate the correct combination of Insert, update, delete and select SQL statement to move object data to and from the RDBMS persistence layer.

Scanned by CamScanner

- This process is not simple and is associated with the largest barrier to rapid change when developing new or modifying existing applications.
- Generally, object relational mapping requires experienced software developers.
- Even with experienced staff, small change requests can cause slowdowns in development and testing schedules.
- All these hurdles are best overcome by NOSQL database.
- These databases are schematic and can be scaled down easily.
- They can accommodate application changes easily and can handle any volume of data efficiently.
- This agility has become business driver for NOSQL databases.

3.2 NoSQL DATA ARCHITECTURE PATTERNS

UQ. What are the different architectural patterns in NoSQL? Explain Graph data store and Column Family Store patterns with relevant examples. **(MU - May 19, 10 Marks)**

NoSQL databases were born out of the rigidity of traditional relational or SQL databases, which use tables, columns, and rows to establish relationships across data. Developers welcomed NoSQL databases because they didn't require an upfront schema design; they were able to go straight to development. And it's this flexibility, this "ad-hoc" approach to organizing data, that has arguably been NoSQL's greatest selling point, which continues to appeal to organizations that need to store, retrieve, and analyze either unstructured or rapidly changing data.

The data stored in NoSQL follows any of the four data architecture patterns.

(A)	Key-Value Stores	(B)	Column family (Bigtable)Stores
(C)	Document Stores	(D)	Graph Stores

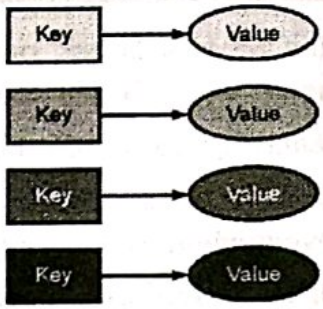
3.2.1 Key-Value Stores

UQ. Explain in detail key-value store NoSQL architectural pattern. Identify two applications that can use thispattern. **(MU - May 18, 5 Marks)**

- One of the most basic NoSQL database models is this model. The data is collected in the pattern of Key-Value Pairs, as the name implies. A series of strings, integers or characters is typically the key, but it can also be a more advanced form of data.
- Usually, the value is connected or co-related to the key. The databases for key-value pair storage typically store information as a hash table where each key is unique.

Scanned by CamScanner

- The value may be of any form (JavaScript Object Notation (JSON), Binary Large Object (BLOB), strings, etc.).
- **Application :** This style of architecture is commonly used in shopping websites or e-commerce applications and its important assets is its ability for wide management of data volumes, heavy loads and the ease with which keys are used to retrieve data.



Key	Value
user-123	"John Doe"
image-123.jpg	<binary image file>
http://webpage-123.html	<web page html>
file-123.pdf	<pdf document>

Fig 3.2.1: An example of Key-Value

Keys and values are flexible. Keys can be image names, web page URLs, or file path names that point to values like binary images, HTML web pages, and PDF documents

Constraints associated with the key-value store databases is its complexity in handling queries which will attempt to include many key-value pairs that may delay output and may cause data to clash with many-to-many relationships.

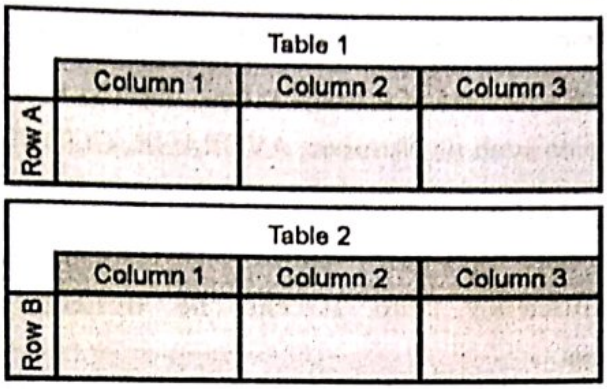
GQ. State example of any two key value databases **(2 Marks)**

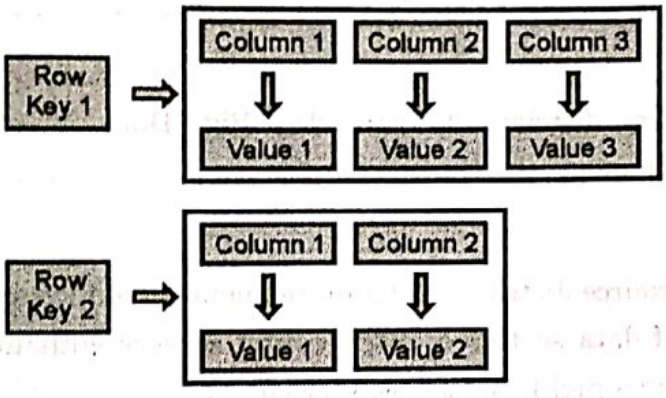
Examples here are:

- DynamoDB (developed by Amazon)
- Berkeley DB (developed by Oracle)
- **REDIS:** An advanced open-source key-value store, also referred to as a data structure server because keys can include strings, hashes, lists, sets and sorted sets. This product, written in C/C++, is searingly quick, which makes it perfect for data collection in real time.
- **Riak:** An open source that is powerful, distributed database that predictably scalescapability and simplifies creation by prototyping, developing, and deployingapplications quickly. Written in Erlang and C this technology gives transparent fault-tolerant/fail-over functionality, a comprehensive and versatile API perfect for point-of-sale and factory control systems.
- **VoltDB:** scalable database in memory that offers complete transactional ACID consistency and ultra-high throughput, self-referred to as the NewSQL. This technology relies on segmentation and replication to achieve high-availability data snapshots and durable command logging using Java stored processes (for crash recovery), making it ideal for capital markets, digital networks, network services, and for online gaming.

Scanned by CamScanner

3.2.2 Column Store Database





	company				super column family
	name	address		website	
1	DataX	city	San Francisco	protocol	https
		state	California	domain	datax.com
		street num	135	subdomain	www
		street	Kearny St		
2	Process-One	city	Arlington	protocol	https
		state	Virginia	domain	process1.com
		street num	3500	subdomain	www
		street	Wilson St		

column

row key column family

Fig 3.2.2 : An Example of Column Store

- This pattern employs data storage in individual cells that is further divided into columns, rather than storing data in relational tuples.
- Databases thatare column-oriented operate only on columns. They together store vast quantities of data in columns. The column format and titles will diverge fromone rowto another.

Scanned by CamScanner

- Each column is handled differently, but still, like conventional databases, each individual column will contain several other columns. (Niharika ,2020)
- Basically, columns are in this sort of storage mode. Data is readily available and it is possible to perform queries such as Number, AVERAGE, COUNT on columns easily.

The setbacks for this system includes: transactions should be avoided or not supported, queries can decrease high performance with table joins, record updates and deletes reduce storage efficiency, and it can be difficult to design efficient partitioning/indexing schemes.

GQ. State example of any two column store databases **(2 Marks)**

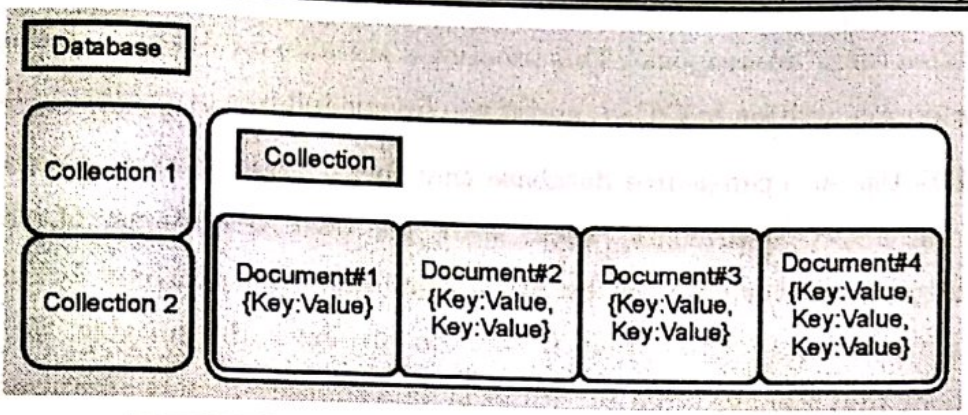
Examples here are:

- **HBase:** HBase is a distributed, portable, Big Data Store modelled after Google's BigTable technology, the Hadoop database.
- Google's BigTable
- Cassandra: An open-source distributed database management system built to manage very large volumes of data scattered over several servers without a single point of failure while delivering a highly accessible service.
- Written in Java, this product is best for non-transactional real-time data analysis with linear scalability and proven fault-tolerance combined with column indexes.

3.2.3 Document Database

- In the form of key-value pairs, the record database fetches and accumulates information, but here the values are called documents. A complicated data structure can be represented as a text.
- It is hierarchical version of key-value databases.
- The document can be in text form, arrays, strings, JSON (JavaScript Object Notation), XML (Extensible Markup Language) or any other format.
- The use of nested documents is immensely popular. It is highly efficient since most of the generated information is generally in the form of JSONs and is unstructured.

Scanned by CamScanner



Key	Document
document-1	{ "id":"1", "name":"John Smith", "isactive":"true", "birthdate":"08/30/1984" }
document-2	{ "id":"2", "fullname":"Sara Walker", "isactive":"false", "birthdate":"02/15/1971" }
document-3	{ "id":"3", "fullname": { "firstname":"Max", "lastname":"Johnson" "middleinitial":"B" } "isactive":"true", "birthdate":"04/02/1964" }

Fig 3.2.3 : An Example of Document

This format is extremely useful and appropriate for semi-structured data, and it is simple to retrieve and handle documents from storage. The drawbacks associated with this system includes the challenging factor of handling multiple documents and the inaccurate working of aggregation operations.

GQ. State example of any two document store databases **(2 Marks)**

Examples of such databases are:

- **MongoDB:** This scalable, high-performance, open-source NoSQL database features document-oriented (JSON-like) storage, full index support, replication, and fast on-

Scanned by CamScanner

site updates from "humongous". This product is suitable for dynamic queries, dynamic data structures, written in C/C++, and if you favour indexes over Map/Reduce.

- **CouchDB:**Also an open-source database that focuses on the ease of data storage in aseries of JSON documents, each with its own definitions of the schema. Eventualconsistency is enforced by ACID semantics that prevents locking database files duringwriting. This product, written in Java, is suitable for web-based applications that manage large quantities of data that are loosely organized.

3.2.4 Graph Database

GQ. Write a short note on Graph Databases. **(4Marks)**

- This pattern of architecture clearly deals with information storage and management in graphs.
- Graphs are essentially structures that represent relations between two or more objects in some data.
- Objects or entities are referred to as nodes and are connected with relationships known as edges. There is a unique identifier on each edge.

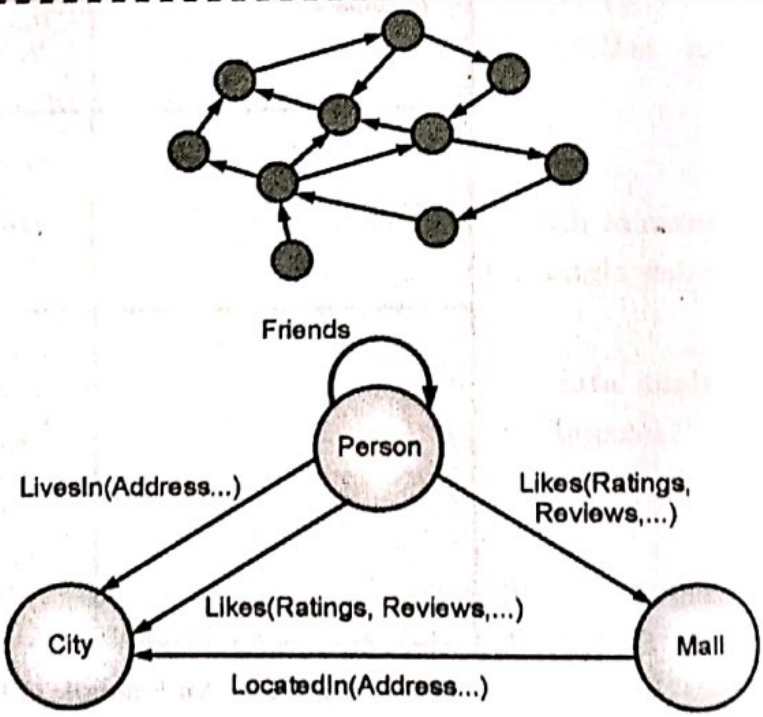


Fig 3.2.4 : An Example of Graph

- For the graph, each node serves as a point of touch. In social networks where there are many and large numbers of entities, this pattern is very widely used and each entity has one or many characteristics that are linked by edges.

There are loosely connected tables in the relational database pattern, whereas graphs are often strong and rigid in nature, have a faster traversal due to connections, and allow spatial data to be easily handled, but incorrect connections can lead to infinite loops. (Jan. 2016)

Scanned by CamScanner

GQ. State example of any two graph databases **(2 Marks)**

Examples of such databases are:

- **Neo4J:** The leading native graph database and graph platform is Neo4J: Neo4j. For enterprise levels of security and high performance and reliability by clustering, it is available both as open source and through a commercial license. Cypher, the graph query language of Neo4j, is very simple to learn and can use newly released open source toolkits, "Cypher on Apache Spark (CApS) and Cypher for Gremlin to operate across Neo4j, Apache Spark and Gremlin-based products."
- **FlockDB (Used by Twitter):** FlockDB is easier than other graph databases since it attempts to solve less problems. It fits horizontally and is optimized for on-line, low-latency, high throughput environments such as websites.
- **ArangoDB:** this type of graph database requires one database, One Query language, Three models for data. The Limitless Possibilities. ArangoDB is a fast-growing native multi-model NoSQL database, with more than one million downloads.
- **OrientDB:** OrientDB is the first Distributed DBMS multi-model with a True Graph Driver. Multi-Model means NoSQL 2nd generation that is capable of managing complex domains with amazing efficiency.
- **Titan:** Titan is a scalable graph database designed for storing and querying graphs spread over a multi-machine cluster comprising hundreds of billions of vertices and edges. Titan is a transactional database that can facilitate the real-time execution of complex graph traversals by thousands of concurrent users.
- **DataStax:** In a rapidly changing environment where aspirations are strong, DataStax helps businesses thrive and new technologies occur daily.
- **Amazon Neptune:** Amazon Neptune with a highly connected datasets to create and run applications is secure, fast and has a fully managed graph database service that is easy. (Niharika, 2020).

Scanned by CamScanner

NoSQL databases features

Types Features	Key-value store	Document store	Column-oriented store	Graph store
Characteristics	A simple hash tableindexed by key	Multiple key/value pairsform a document. Document stored generally in JSON format.	Store data in columnar format. Each key is associated with multiple attributes.	Focused on modelling the structure of the data.
Pros	Very fast and scalable. Simple model.	Schema free: Unstructured data can be stored easily. Simple, powerful data model. Horizontal scalability.	Better for complex read queries. Fast querying of data. Storage of very large quantities of data. Better analytic performance. Improved data compression.	Powerful data model. Locally connected data. Indexed data. Easy to query. Handling complex relational information.
Cons	Stored data have no schema. Poor for complex data. All joins must be done in code. No foreign key constraints. No triggers.	Query model limited to keys and indexes. No standard query syntax. Map Reduce for larger queries. Poor for interconnected data.	Very low-level API. Undefined data usage pattern. Increased disk seek time. Increased cost of inserts. Poor for interconnected data.	Travers entire graph to give correct results. Sharding.

Scanned by CamScanner

Types Features	Key-value store	Document store	Column-oriented store	Graph store
Suitable for	Storing session's information, user profiles, preferences, shopping cart data.	Content management systems, blogging platforms, web analytics, real-time analytic's, e-commerce applications.	Content management systems, blogging platforms, maintaining counters, expiring usage, heavy write volume such as log aggregation	Space problem and connected data, such as social networks, spatial data, routing information for goods and money, recommendation engines.
Examples	Riak Redis MemcacheDB Dynamo Voldemort	MongoDB CouchDB ArangoDB MarkLogic RzthinkDB	BigTable Habse Casandra Accumulo Hypertable	Neo4j OrientDB Allegro Virtuoso InfiniteGraph

3.2.5 Variations of NoSQL Architectural Patterns

GQ. How NoSQL data architecture patterns varies as you move from a single processor to multiple processors? **(10 Marks)**

The key-value store, graph store, Bigtable store, and document store patterns can bemodified by focusing on a different aspect of system implementation. Variations on the architectures that use RAM or solid state drives (SSDs), and the patterns can be used on distributed systems or modified to createenhanced availability. Finally, we'll look at how database items can be groupedtogether in different ways to make navigation over many items easier.

Customization for RAM or SSD stores

Some NoSQL products are designed to specifically work with one type of memory; for example, Memcache, a key-value store, was specifically designed to see ifitems are in RAM on multiple servers. A key-value store that only uses RAM is called a RAM cache; it's flexible and has general tools that application developers can use tostore global variables, configuration files, or intermediate results of document transformations.

Scanned by CamScanner

A RAM cache is fast and reliable, and can be thought of as another programming construct like an array, a map, or a lookup system. There are several things about them you should consider:

- Simple RAM resident key-value stores are generally empty when the server startsup and can only be populated with values on demand.
- You need to define the rules about how memory is partitioned between theRAM cache and the rest of your application.
- RAM resident information must be saved to another storage system if you want itto persist between server restarts.

The key is to understand that RAM caches must be re-created from scratch each time a server restart. A RAM cache that has no data in it is called a cold cache and is why some systems get faster the more they're used after a reboot.

SSD systems provide permanent storage and are almost as fast as RAM for read operations. The Amazon DynamoDB key-value store service uses SSDs for all its storage, resulting in high-performance read operations. Write operations to SSDs can often be buffered in large RAM caches, resulting in fast write times until the RAM becomes full.

Distributed Stores

- NoSQL data architecture patterns vary from a single processor to multiple processors that are distributed over data centers in different geographic regions. The ability to elegantly and transparently scale to a large number of processors is a core property of most NoSQL systems. Ideally, the process of data distribution is transparent to the user, meaning that the API doesn't require you toknow how or where your data is stored. But knowing that your NoSQL software can scale and how it does this is critical in the software selection process.
- If your application uses many web servers, each caching the result of a long-running query, it's most efficient to have a method that allows the servers to worktogether to avoid duplication. This mechanism, known as memcache. Whether we use NoSQL or traditional SQL systems, RAM continues to be the most expensive and precious resource in anapplication server's configuration. If you don't have enough RAM, your application won't scale.
- The solution used in a distributed key-value store is to create a simple, lightweightprotocol that checks whether any other server has an item in its cache. If it does, this item is quickly returned to the requester and no additional searching is

Scanned by CamScanner

required. The protocol is simple: each memcache server has a list of the other memcache servers it's working with. Whenever a memcache server receives a request that's not in its own cache, it checks with the other peer servers by sending them the key.

- The memcache protocol shows that you can create simple communication protocols between distributed systems to make them work efficiently as a group. This type of information sharing can be extended to other NoSQL data architectures such as Big table stores and document stores. You can generalize the key-value pair to other patterns by referring to them as cached items.
- Cached items can also be used to enhance the overall reliability of a data service by replicating the same items in multiple caches. If one server goes down, other servers quickly fill in so that the application gives the user the feeling of service without interruption. To provide a seamless data service without interruption, the cached items need to be replicated automatically on multiple servers. If the cached items are stored on two servers and the first one becomes unavailable, the second server can quickly returnthe value; there's no need to wait for the first server to be rebooted or restored from backup.

Grouping Items

- In the key-value store section, we looked at how web pages can be stored in a key-value store using a website URL as the key and the web page as the value. We can extend this construct to file systems as well. In a filesystem, the key is the directory or folder path, and the value is the file content. But unlike web pages, file systems have the ability to list all the files in a directory without having to open the files. If the file content is large, it would be inefficient to load all of the files into memory each time you want a listing of the files.
- To make this easier and more efficient, a key-value store can be modified to include additional information in the structure of the key to indicate that the key-value pair is associated with another key-value pair, creating a collection, or general-purpose structures used to group resources. Though each key-value store system might call it something different (such as folders, directories, or buckets), the concept is the same.
- The implementation of a collection system can also vary dramatically based onwhat NoSQL data pattern you use. Key-value stores have several methods to group similar items based on attributes in their keys. Graph stores associate one or moregroup identifiers with each triple. Big data systems use column families to group similar columns. Document stores use a concept of a document collection.

Scanned by CamScanner

- One approach to grouping items is to have two key-value data types, the first called resource keys and the second collection keys. We can use collection keys to store a list ofkeys that are in a collection. This structure allows you to store a resource in multiple collections and also to store collections within collections. Using this design poses some complex issues that require careful thought and planning about what should bedone with a resource if it's in more than one collection and one of the collections is deleted.
- To simplify this process and subsequent design decisions, key-value systems can include the concept of creating collection hierarchies and require that a resource bein one and only one collection. The result is that the path to a resource is essentially a distinct key for retrieval. Also known as a simple document hierarchy, the familiar concept of folders and documents resonates well with end users.

These are some examples where we can use the concept of a collection hierarchy in a key, use it to perform many functions on groups of key-value pairs:

- Associate metadata with a collection (who created the collection, when it was created, the last time it was modified, and who last modified the collection).
- Give the collection an owner and group, and associate access rights with the owner group and other users in the same way UNIX file systems use permissions.
- Create an access control permission structure on a collection, allowing only users with specific privileges the ability to read or modify the items within the collection.
- Create tools to upload and/or download a group of items into a collection.
- Set up systems that compress and archive collections if they haven't been accessed for a specific period of time.

3.2.6 NoSQL case studies

LiveJournal's Memcache

- Engineers working on the blogging system Live Journal started to look at how their systems were using their most precious resource: the RAM in each web server. Live-Journal had a problem. Their website was so popular that the number of visitors using the site continued to increase on a daily basis. The only way they could keep up with demand was to continue to add more web servers, each with its own separate RAM.

Scanned by CamScanner

- To improve performance, the LiveJournal engineers found ways to keep the results of the most frequently used database queries in RAM, avoiding the expensive cost of rerunning the same SQL queries on their database. But each web server had its own copy of the query in RAM; there was no way for any web server to know that the server next to it in the rack already had a copy of the query sitting in RAM.
- So the engineers at LiveJournal created a simple way to create a distinct "signature" of every SQL query. This signature or hash was a short string that represented a SQL SELECT statement. By sending a small message between web servers, any web server could ask the other servers if they had a copy of the SQL result already executed. If one did, it would return the results of the query and avoid an expensive round trip to the already overwhelmed SQL database. They called their new system Memcache because it managed RAM memory cache.
- Many other software engineers had come across this problem in the past. The concept of large pools of shared-memory servers wasn't new. What was different this time was that the engineers for LiveJournal went one step further. They not only made this system work (and work well), they shared their software using an open source license, and they also standardized the communications protocol between the web front ends (called the memcached protocol). Now anyone who wanted to keep their database from getting overwhelmed with repetitive queries could use their front end tools.

Google's MapReduce—use commodity hardware to create search indexes

- Google shared their process for transforming large volumes of web data content into search indexes using low-cost commodity CPUs. Though sharing of this information was significant, the concepts of map and reduce weren't new. Map and reduce functions are simply names for two stages of a data transformation
- The initial stages of the transformation are called the map operation. They're responsible for data extraction, transformation, and filtering of data. The results of the map operation are then sent to a second layer: the reduce function. The reduce function is where the results are sorted, combined, and summarized to produce the final result.
- The core concepts behind the map and reduce functions are based on solid computer science work that dates back to the 1950s when programmers at MIT implemented these functions in the influential LISP system. LISP was different than other programming languages because it emphasized functions that transformed isolated lists of data. This focus is now the basis for many modern functional programming languages that have desirable properties on distributed systems.

Scanned by CamScanner

- Google extended the map and reduce functions to reliably execute on billions of web pages on hundreds or thousands of low-cost commodity CPUs. Google made map and reduce work reliably on large volumes of data and did it at a low cost. It was Google's use of MapReduce that encouraged others to take another look at the power of functional programming and the ability of functional programming systems to scale over thousands of low-cost CPUs. Software packages such as Hadoop have closely modeled these functions.
- The use of MapReduce inspired engineers from Yahoo! and other organizations to create open source versions of Google's MapReduce. It fostered a growing awareness of the limitations of traditional procedural programming and encouraged others to use functional programming systems.

Google's Bigtable—a table with a billion rows and a million columns

Google also influenced many software developers when they announced their Bigtable system white paper titled A Distributed Storage System for Structured Data. The motivation behind Bigtable was the need to store results from the web crawlers that extract HTML pages, images, sounds, videos, and other media from the internet. The resulting dataset was so large that it couldn't fit into a single relational database, so Google built their own storage system. Their fundamental goal was to build a system that would easily scale as their data increased without forcing them to purchase expensive hardware. The solution was neither a full relational database nor a filesystem, but what they called a "distributed storage system" that worked with structured data.

By all accounts, the Bigtable project was extremely successful. It gave Google developers a single tabular view of the data by creating one large table that stored all the data they needed. In addition, they created a system that allowed the hardware to be located in any data center, anywhere in the world, and created an environment where developers didn't need to worry about the physical location of the data they manipulated.

Amazon's Dynamo—accept an order 24 hours a day, 7 days a week

- Google's work focused on ways to make distributed batch processing and reporting easier, but wasn't intended to support the need for highly scalable web storefronts that ran 24/7. This development came from Amazon. Amazon published another significant NoSQL paper: Amazon's 2007 Dynamo: A Highly Available Key-Value Store. The business motivation behind Dynamo was Amazon's need to create a highly reliable web storefront that supported transactions from around the world 24 hours a day, 7 days a week, without interruption.

Scanned by CamScanner

- Traditional brick-and-mortar retailers that operate in a few locations have the luxury of having their cash registers and point-of-sale equipment operating only during business hours. When not open for business, they run daily reports, and perform backups and software upgrades. The Amazon model is different. Not only are their customers from all corners of the world, but they shop at all hours of the day, every day. Any downtime in the purchasing cycle could result in the loss of millions of dollars. Amazon's systems need to be iron-clad reliable and scalable without a loss in service.
- In its initial offerings, Amazon used a relational database to support its shopping cart and checkout system. They had unlimited licenses for RDBMS software and a consulting budget that allowed them to attract the best and brightest consultants for their projects. In spite of all that power and money, they eventually realized that a relational model wouldn't meet their future business needs.
- Many in the NoSQL community cite Amazon's Dynamo paper as a significant turning point in the movement. At a time when relational models were still used, it challenged the status quo and current best practices. Amazon found that because key-value stores had a simple interface, it was easier to replicate the data and more reliable. In the end, Amazon used a key-value store to build a turnkey system that was reliable, extensible, and able to support their 24/7 business model, making them one of the most successful online retailers in the world.

MarkLogic

- In 2001 a group of engineers in the San Francisco Bay Area with experience in document search formed a company that focused on managing large collections of XML documents. Because XML documents contained markup, they named the company MarkLogic.
- MarkLogic defined two types of nodes in a cluster: query and document nodes. Query nodes receive query requests and coordinate all activities associated with executing a query. Document nodes contain XML documents and are responsible for executing queries on the documents in the local filesystem.
- Query requests are sent to a query node, which distributes queries to each remote server that contains indexed XML documents. All document matches are returned to the query node. When all document nodes have responded, the query result is then returned.

Scanned by CamScanner

- The MarkLogic architecture, moving queries to documents rather than moving documents to the query server, allowed them to achieve linear scalability with petabytes of documents.
- MarkLogic found a demand for their products in US federal government systems that stored terabytes of intelligence information and large publishing entities that wanted to store and search their XML documents. Since 2001, MarkLogic has matured into a general-purpose highly scalable document store with support for ACID transactions and fine-grained, role-based access control. Initially, the primary language of MarkLogic developers was XQuery paired with REST; newer versions support Java as well as other language interfaces.
- MarkLogic is a commercial product that requires a software license for any datasets over 40 GB. NoSQL is associated with commercial as well as open-source products that provide innovative solutions to business problems.

3.3 NOSQL SOLUTION FOR BIG DATA

A big data class problem is any business problem that's so large that it can't be easily managed using a single processor. Big data problems force you to move away from a single-processor environment toward the more complex world of distributed computing. Though great for solving big data problems, distributed computing environments come with their own set of challenges.

Some of the challenges you face when you move from a single processor to a distributed computing system. Moving to a distributed environment is a nontrivial endeavour and should be done only if the business problem really warrants the need to handle large data volumes in a short period of time. This is why platforms like Hadoop are complex and require a complex framework to make things easier for the application developer.

These are challenges for one or many databases:

NoSQL includes concepts and use cases that can be managed by a single processor and have a positive impact on agility and data quality. But we consider big data problems a primary use case for NoSQL.

Scanned by CamScanner

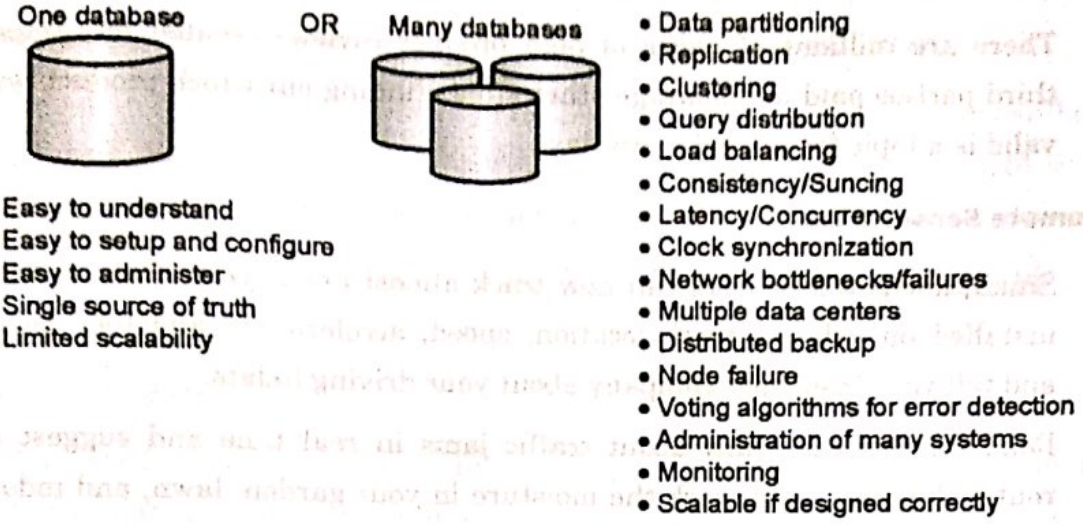


Fig. 3.3.1

Here are some typical big data use cases:

(1) Bulk Image Processing	(2) Public Web Page Data
(3) Remote Sensor Data	(4) Event Log Data
(5) Mobile Phone Data	(6) Social Media Data
(7) Game Data	(8) Open Linked Data

(1) Bulk Image Processing

- Organizations like NASA regularly receive terabytes of incoming data from satellites or even rovers on Mars. NASA uses a large numberof servers to process these images and perform functions like image enhancement and photo stitching.
- Medical imaging systems like CAT scans and MRIs need to convert raw image data into formats that are useful to doctors and patients. Custom imaging hardware has been found to be more expensive thanrenting a large number of processors on the cloud when they're needed.
- For example, the New York Times converted 3.3 million scans of old newspaper articles into web formats using tools like Amazon EC2 and Hadoop for a few hundred dollars.

(2) Public Web Page Data

- Publicly accessible pages are full of information that organizations can use to be more competitive. They contain news stories, RSS feeds, new product information, product reviews, and blog postings. Not all of the information is authentic.

Scanned by CamScanner

- There are millions of pages of fake product reviews created by competitors or third parties paid to disparage other sites. Finding out which product reviews are valid is a topic for careful analysis.

(3) Remote Sensor Data

- Small, low-power sensors can now track almost any aspect of our world. Devices installed on vehicles track location, speed, acceleration, and fuel consumption, and tell your insurance company about your driving habits.
- Road sensors can warn about traffic jams in real time and suggest alternate routes. You can even track the moisture in your garden, lawn, and indoor plants to suggest a watering plan for your home.

(4) Event Log Data

- Computer systems create logs of read-only events from web page hits (also called clickstreams), email messages sent, or login attempts.
- Each of these events can help organizations understand who's using what resources and when systems may not be performing according to specification.
- Event log data can be fed into operational intelligence tools to send alerts to users when key indicators fall out of acceptable ranges.

(5) Mobile Phone Data

- Every time users move to new locations; applications can track these events. You can see when your friends are around you or when customers walk through your retail store.
- Although there are privacy issues involved in accessing this data, it's forming a new type of event stream that can be used in innovative ways to give companies a competitive advantage.

(6) Social Media Data

- Social networks such as Twitter, Facebook, and LinkedIn provide a continuous real-time data feed that can be used to see relationships andtrends.
- Each site creates data feeds that you can use to look at trends in customer mood or get feedback on your own as well as competitor products.

Scanned by CamScanner

(7) Game Data

- Games that run on PCs, video game consoles, and mobile deviceshave back-end datasets that need to scale quickly. These games store and sharehigh scores for all users as well as game data for each player.
- Game site backends must be able to scale by orders of magnitude if viral marketing campaign scatch on with their users.

(8) Open Linked Data

- Not only is this data large, butit may require complex tools to reconcile, remove duplication, and find invaliditems.
- This includes problems like image and signal processing. Theirfocus is on the efficient and reliable data transformation at scale. These use cases don't need the query or transactions support provided by many NoSQL systems.

They read and write to key-value stores or distributed filesystems like *Amazon's Simple StorageService (S3)* or *Hadoop Distributed File System (HDFS)* and may not need the advanced features of a document store or an RDBMS. Other use cases are more demanding and need more features. Big data problems like event log data and game data do need to store their data directly into structures that can be queried and analysed, so they will need different NoSQL solutions.

To be a good candidate for a general class of big data problems, NoSQL solutionsshould :

1. Be efficient with input and output and scale linearly with growing data size.
2. Be operationally efficient. Organizations can't afford to hire many people to run the servers.
3. Require that reports and analyses be performed by nonprogrammers using simple toolsnot every business can afford a full-time Java programmer to write on-demand queries.
4. Meet the challenges of distributed computing, including consideration of latency between systems and eventual node failures.
5. Meet both the needs of overnight batch processing economy-of-scale and time-critical event processing.

Scanned by CamScanner

6. RDBMS can, with enough time and effort, be customized to solve some big data problems. Applications can be rewritten to distribute SQL queries to many processors and merge the results of the queries. Databases can be redesigned to remove joins between tables that are physically located on different nodes. SQL systems can be configured to use replication and other data synchronization processes. Yet these steps all take considerable time and money.

3.4 UNDERSTANDING THE TYPES OF BIG DATA PROBLEMS

There are many types of big data problems, each requiring a different combination of NoSQL systems. After you've categorized your data and determined its type, you'll find there are different solutions.

Fig.3.4.1 is a good example of a high-level big data classification system.

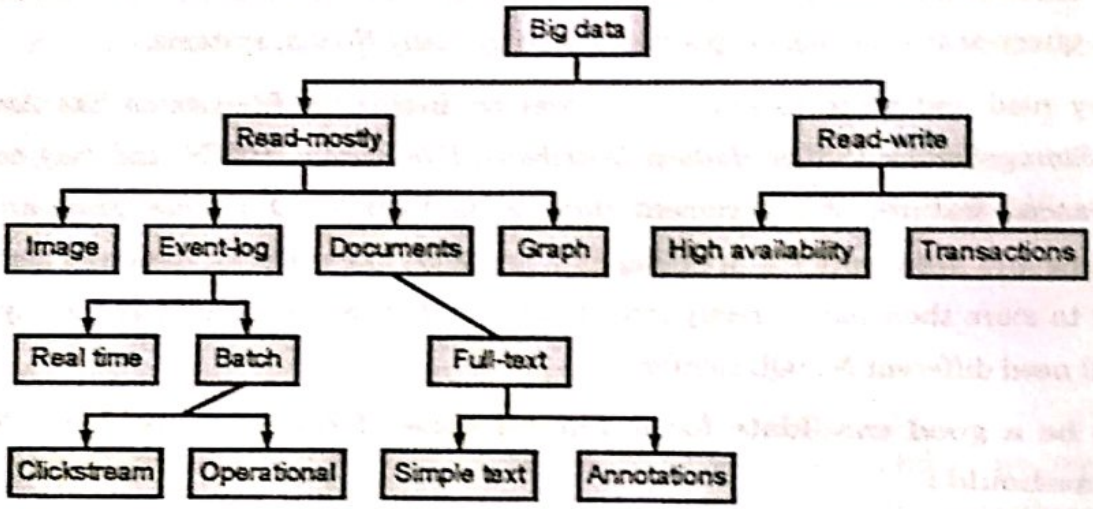


Fig.3.4.1:A sample of a taxonomy of big data types

Some ways you classify big data problems and see how NoSQL systems are changing the way organizations use data.

(1) **Read-mostly :** Read-mostly data is the most common classification. It includes data that's created once and rarely altered. This type of data is typically found in data warehouse applications but is also identified as a set of non-RDBMS items like images or video, event-logging data, published documents, or graph data. Event data includes things like retail sales events, hits on a website, system logging data, or real-time sensor data.

(2) **Log events :** When operational events occur in your enterprise, you can record it in a log file and include a timestamp so you know when the event occurred. Log events

Scanned by CamScanner

may be a web page click or an out-of-memory warning on a disk drive. In the past, the cost and amount of event data produced were so large that many organizations opted not to gather or analyse it. Today, NoSQL systems are changing companies' thoughts on the value of log data as the cost to store andanalyse it is more affordable.

The ability to cost-effectively gather and store log events from all computers in your enterprise has lead to BI operational intelligence systems. Operational intelligence goes beyond analysing trends in your web traffic or retail transactions. It can integrate information from network monitoring systems so you can detect problems before they impact your customers. Cost-effective NoSQL systems can be part of good operations management solutions.

(3) **Full-text documents :** This category of data includes any document that contains natural-language text like the English language. An important aspect of document stores is that you can query the entire contents of your office document inthe same way you would query rows in your SQL system.

This means that you can create new reports that combine traditional data inRDBMSs as well as the data within your office documents. For example, youcould create a single query that extracted all the authors of titles of PowerPoint slides that contained the keywords NoSQL or big data. The result of this list of authors could then be filtered with a list of titles in the HR database to show which people had the title of Data Architect or Solution Architect.

This is a good example of how organizations are trying to tap into the hidden skills that already exist within an organization for training and mentorship. Integrating documents into what can be queried is opening new doors in knowledge management and efficient staff utilization.

3.5 ANALYZING BIG DATA WITH A SHARED-NOTHING ARCHITECTURE

GQ. Explain Shared Nothing architecture in detail. **(10 Marks)**

There are three ways that resources can be shared between computer systems: shared RAM, shared disk, and shared-nothing. Fig.3.5.1 shows a comparison of these three distributed computing architectures. Of the three alternatives, a shared-nothing architecture is most cost effective interms of cost per processor when you're using commodity hardware.

Scanned by CamScanner

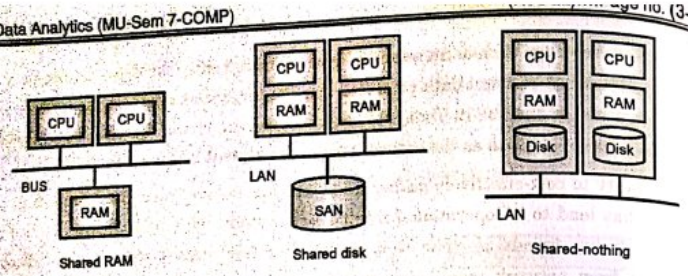


Fig.3.5.1 :Three ways to share resources

In Fig 3.5.1, The left panel shows a shared RAM architecture, where many CPUs access a single shared RAM over a high-speed bus. This system is ideal for large graph traversal. The middle panel shows a shared disk system, where processors have independent RAM but share disk using a storage area network (SAN). The right panel shows an architecture used in big data solutions: cache-friendly, using low-cost commodity hardware, and a shared-nothing architecture.

Of the architectural data patterns row store, key-value store, graph store, document store, and Bigtable store, only two (key-value store and document store) lend themselves to cache-friendliness. Bigtable stores scale well on shared-nothing architectures because their row-column identifiers are similar to key-value stores. But row stores and graph stores aren't cache-friendly since they don't allow a large BLOB to be referenced by a short key that can be stored in the cache. For graph traversals to be fast, the entire graph should be in main memory. This is why graph stores work most efficiently when you have enough RAM to hold the graph.

3.6 CHOOSING DISTRIBUTION MODELS: MASTER-SLAVE VERSUS PEER-TO-PEER

GQ. Explain Distributing models in details. **(10 Marks)**

- From a distribution perspective, there are two main models: master-slave and peer-to-peer. Distribution models determine the responsibility for processing data when a request is made.
- Understanding the pros and cons of each distribution model is important when you're looking at a potential big data solution. Peer-to-peer models may be more resilient to failure than master-slave models. Some master-slave distribution models have single points of failure that might impact your system availability, so you might need to take special care when configuring these systems.

- In the master-slave model, one node is in charge (master). When there's no single node with a special role in taking charge, you have a peer-to-peer distribution model.
- Fig. 3.6.1 shows how these models each work.

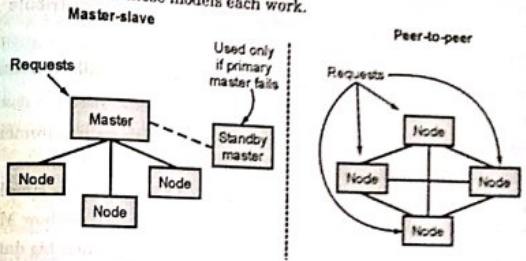


Fig.3.6.1 : Master-slave versus peer-to-peer

- In Fig.3.6.1 on the left illustrates a master-slave configuration where all incoming database requests (reads or writes) are sent to a single master node and redistributed from there. The master node is called the NameNode in Hadoop.
- This node keeps a database of all the other nodes in the cluster and the rules for distributing requests to each node. The panel on the right shows how the peer-to-peer model stores all the information about the cluster on each node in the cluster. If any node crashes, the other nodes can take over and processing can continue.
- With a master-slave distribution model, the role of managing the cluster is done on a single master node. This node can run on specializedhardware such as RAID drives to lower the probability that it crashes. The cluster canalso be configured with a standby master that's continually updated from the masternode.
- The challenge with this option is that it's difficult to test the standby master without jeopardizing the health of the cluster. Failure of the standby master to takeover from the master node is a real concern for high-availability operations.
- Peer-to-peer systems distribute the responsibility of the master to each node in the cluster. In this situation, testing is much easier since you can remove any node in the cluster and the other nodes will continue to function.
- The disadvantage of peer-to-peer networks is that there's an increased complexity and communication overhead that must occur for all nodes to be kept up to date with the cluster status.

Scanned by CamScanner

- The initial versions of Hadoop (frequently referred to as the 1.x versions) were designed to use a master-slave architecture with the NameNode of a cluster being responsible for managing the status of the cluster. NameNodes usually don't deal with any MapReduce data themselves. Their job is to manage and distribute queries to thecorrect nodes on the cluster.
- Hadoop 2.x versions are designed to remove single points of failure from a Hadoop cluster.
- Using the right distribution model will depend on your business requirements: if high availability is a concern, a peer-to-peer network might be the best solution. If you can manage your big data using batch jobs that run in off hours, then the simpler master-slave model might be best. As we move to the next section, you'll see how Map-Reduce systems can be used in multiprocessor configurations to process your big data.

3.7 NOSQL SYSTEMS TO HANDLE BIG DATA PROBLEMS

1. Moving queries to the data, not data to thequeries

- With the exception of large graph databases, most NoSQL systems use commodity processors that each hold a subset of the data on their local shared-nothing drives.
- When a client wants to send a general query to all nodes that hold data, it's more efficient to send the query to each node than it is to transfer large datasets to a central processor. This may seem obvious, but it's amazing how many traditional databases still can't distribute queries and aggregate query results.
- This simple rule helps you understand how NoSQL databases can have dramatic performance advantages over systems that weren't designed to distribute queries to the data nodes.
- Consider an RDBMS that has tables distributed over two different nodes. In order for the SQL query to work, information about rows on one table must all be moved across the network to the other node.
- Larger tables result in more data movement, which results in slower queries. Think of all the steps involved. The tables can be extracted, serialized, sent through the network interface, transmitted over networks, reassembled, and then compared on the server with the SQL query.
- Keeping all the data within each data node in the form of logical documents means that only the query itself and the final result need to be moved over a network. This keeps your big data queries fast.

2. Using hash rings to evenly distribute data on a cluster

- One of the most challenging problems with distributed databases is figuring out a consistent way of assigning a document to a processing node.
- Using a hash ring technique to evenly distribute big data loads over many servers with a randomly generated 40-character key is a good way to evenly distribute a network load.
- Hash rings are common in big data solutions because they consistently determine how to assign a piece of data to a specific processor. Hash rings take the leading bits of a document's hash value and use this to determine which node the document should be assigned. This allows any node in a cluster to know what node the data lives on and how to adapt to new assignment methods as your data grows.
- Partitioning keys into ranges and assigning different key ranges to specific nodes is known as keyspace management. Most NoSQL systems, including MapReduce, use keyspace concepts to manage distributed computing problems.
- The concept of a hash ring can also be extended to include the requirement that an item must be stored on multiple nodes.
- When a new item is created, the hash ring rules might indicate both a primary and a secondary copy of where an item is stored. If the node that contains the primary fails, the system can look up the node where these secondary item is stored.

3. Using replication to scale reads

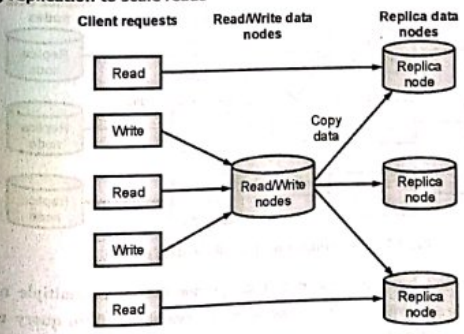


Fig 3.7.1 : Duplication of the Data to increase the performance of NoSQL system

Scanned by CamScanner

- The Fig 3.7.1 show how you can replicate data to speed read performance in NoSQL systems. All incoming client requests enter from the left. All reads can be directed to any node, either a primary read/write node or a replica node. All write transactions can be sent to a central read/write node that will update the data and then automatically send the updates to replica nodes. The time between the write to the primary and the time the update arrives on the replica nodes determines how long it takes for reads to return consistent results.
- There are only a few times when you must be concerned about the lag time between a write to the read/write node and a client reading that same record from a replica. One of the most common operationsafter a write is a read of that same record. If a client does a write and then an immediate read from that same node, there's no problem. The problem occurs if a readoccurs from a replica node before the update happens. This is an example of an inconsistent read.
- The best way to avoid this type of problem is to only allow reads to the same write node after a write has been done. This logic can be added to a session or state management system at the application layer. Almost all distributed databases relax database consistency rules when a large number of nodes permit writes. If your application needs fast read/write consistency, you must deal with it at the application layer.

4 Letting the database distribute queries evenly to data nodes

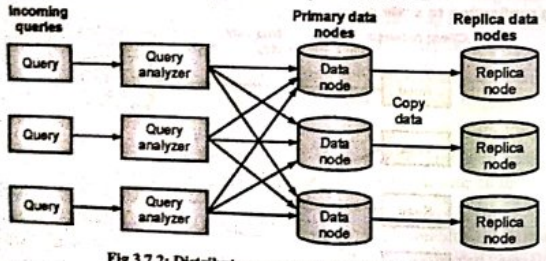


Fig 3.7.2: Distribute query to all the data nodes

- In order to get high performance from queries that span multiple nodes, it's important to separate the concerns of query evaluation from query execution. Fig.3.7.2 shows this structure.

- In this figure NoSQL systems move the query to a data node, but don't move data to a query node. In this example, all incoming queries arrive at query analyzer nodes. These nodes then forward the queries to each data node. If they have matches, the documents are returned to the query node.
- The query won't return until all data nodes (or a response from a replica) have responded to the original query request. If the data node is down, a query can be redirected to a replica of the data node.
- The approach shown in Fig. 3.7.2 is one of moving the query to the data rather than moving the data to the query. This is an important part of NoSQL big data strategies. In this instance, moving the query is handled by the database server, and distribution of the query and waiting for all nodes to respond is the sole responsibility of the database, not the application layer.
- This approach is somewhat similar to the concept of federated search. Federated search takes a single query and distributes it to distinct servers and then combines the results together to give the user the impression they're searching a single system. In some cases, these servers may be in different geographic regions. In this case, sending query to a single cluster that's not only performing search queries on a single local cluster but also performing update and delete operations.

ChapterEnds...

□□□

Scanned by CamScanner