MODULE - V

BIG DATA STORAGE TECHNOLOGY

Q-1 Differentiate between horizontal scaling and vertical scaling.

Ans:

Horizontal scaling	Vertical scaling
It is also known as scaling out.	It is also known as scaling up.
It means that enhancing the performance of server or node by adding more node/ machine to pool of machines.	It means that increasing the capacity of machine by adding more resources, such as more memory or more processors or adding RAM.
Horizontal scaling can be achieved with the help of clustering, distributed file system and load balancing.	Vertical scaling can be acheived with the help of RDBMS.
It provides out of box redundancy and high availability.	It doesn't provide out of the box redundancy and fault tolerance.
It provides simple, fast access data storage that is capable of storing large datasets.	RDBMS are good for handling transactional workloads involves small amounts of data.

It provides fast read/write capability	It provides random read/write capability
Distributed File system are not ACID complaint	But RDBMS are ACID complaint
It is do not provide the ability to search the contents of files	It provides better backup and recovery procedures and ability to search the contents.

Q-3 Write a note on Distributed File System?

Ans: Distributed File system support schema-less data storage. It provides out of box redundancy and high availability by copying data to multiple location via replication. A storage device that is implemented with a distributed file system provides simple, fast access data storage. It provides fast read/write capability.

It is not ideal for datasets comprising a large number of small-files. Multiple smaller files are generally combined into a single file. This allows the distributed file systems to have increased performance when data must be accessed in streaming mode with no random reads and writes.

It provides an inexpensive storage option for storing large amounts of data over a long period of time that needs to remain online. The main disadvantage of distributed file systems do not provide the ability to search the contents of files as standard out-of-the box capability.

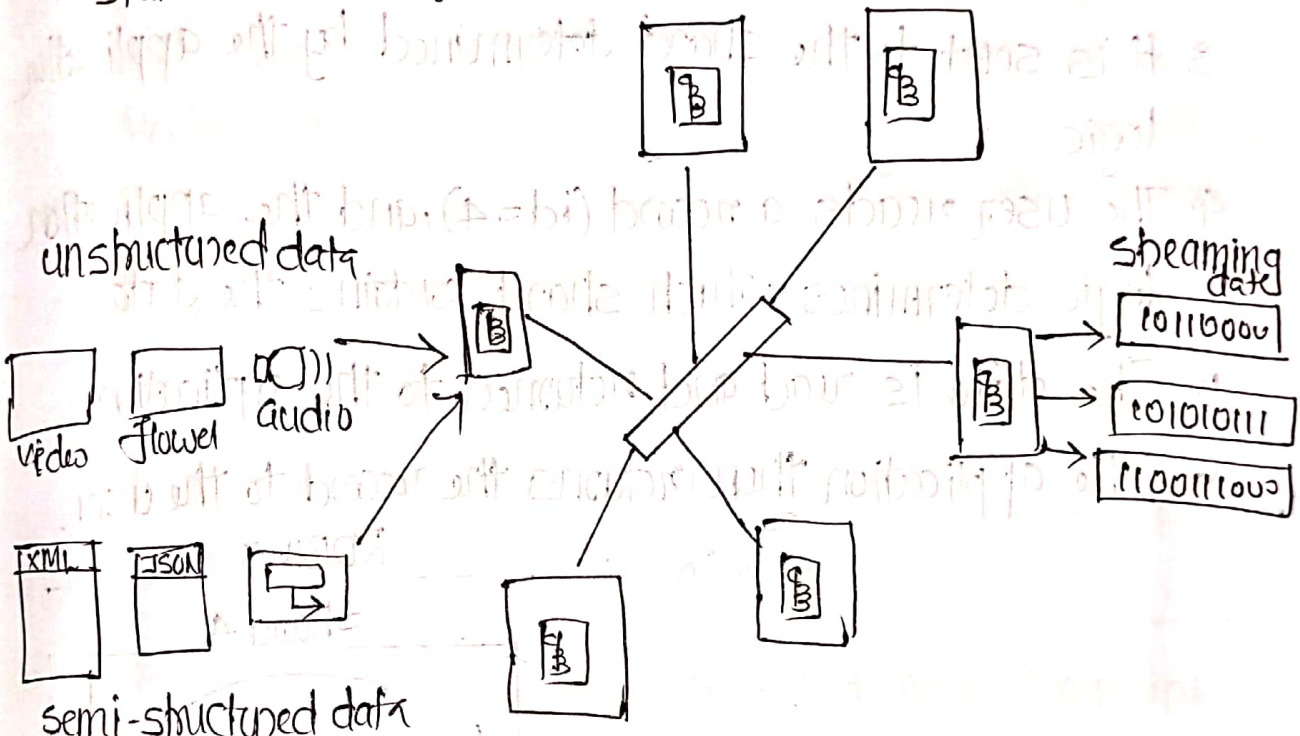


Fig: A distributed file system accessing data in streaming mode with no random reads and writes.

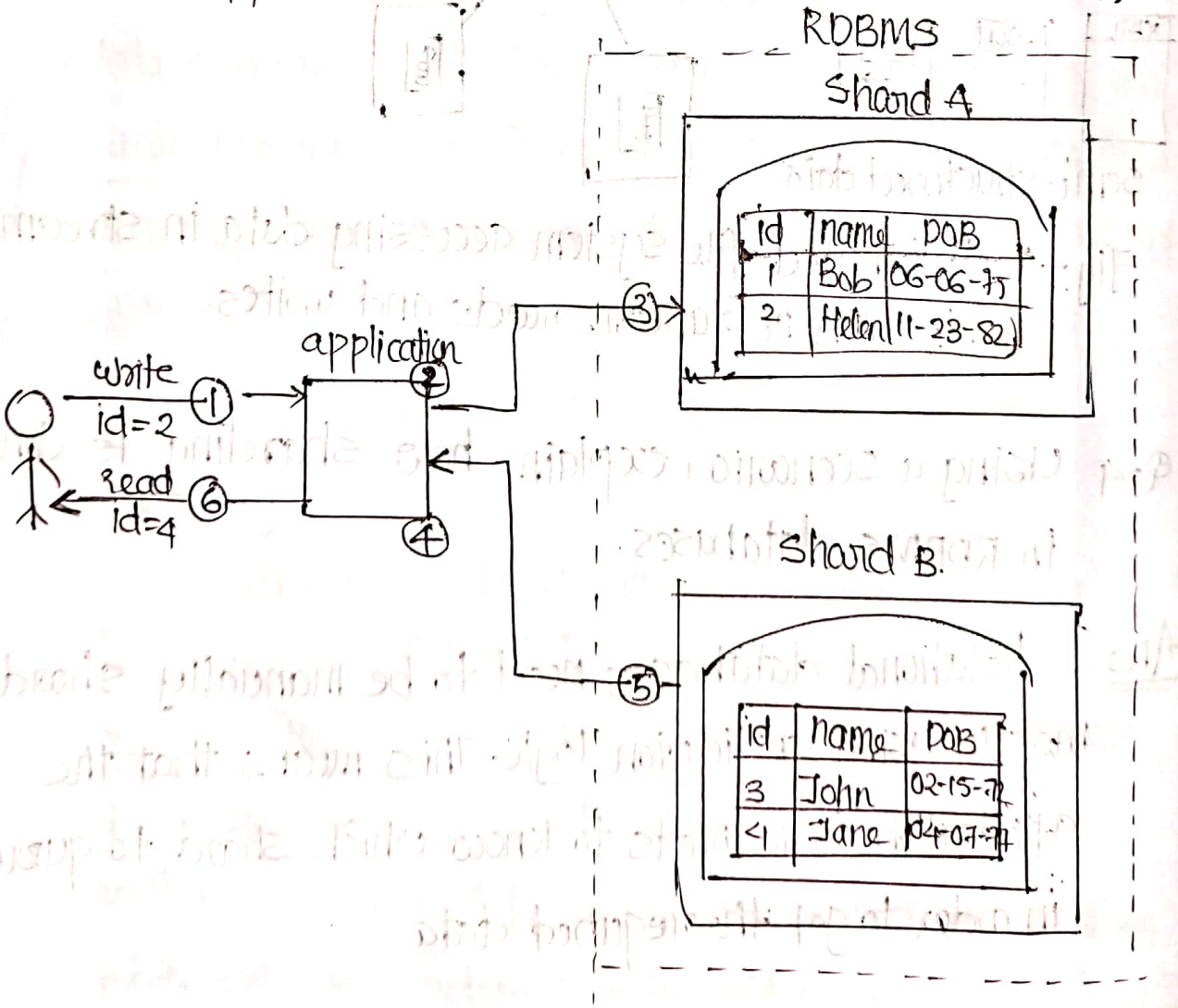
Q-4 Using a scenario, explain how sharding is done in RDBMS databases.

Ans: Relational databases need to be manually sharded, mostly using application logic. This means that the application logic needs to know which shard to query in order to get the required data.

Data processing when data from multiple shards

The following steps are shown in fig:

1. A user writes a record (id=2)
2. The application logic determines which shard it should be written to.
3. It is sent to the shard determined by the application logic.
4. The user reads a record (id=4), and the application logic determines which shard contains the data.
5. The data is read and returned to the application.
6. The application then returns the record to the user.



Q5 Differentiate between RDBMS and NoSQL databases.

Ans:

RDBMS	NoSQL
RDBMS stands for Relational Database Management Systems	NoSQL stands for Not-Only SQL
Data stored in a relational model, with rows and columns.	Data stored in a host of different databases each with different data storage models
It has a fixed schema.	It has no fixed schema (it follows dynamic schema).
It supports a powerful query language.	It supports a very simple query language.
RDBMS is ACID compliant	It is not ACID complaint
Relational database manages only structured data.	NoSQL database can manage structured, unstructured and semi-structured data
RDBMs do not provide out-of-the box redundancy and fault tolerance.	It provide highly scalabale and fault-tolerant.
It has random read/write properties.	It gives both read and write properties.
It supports vertical scaling	It supports horizontal scaling

Figure:

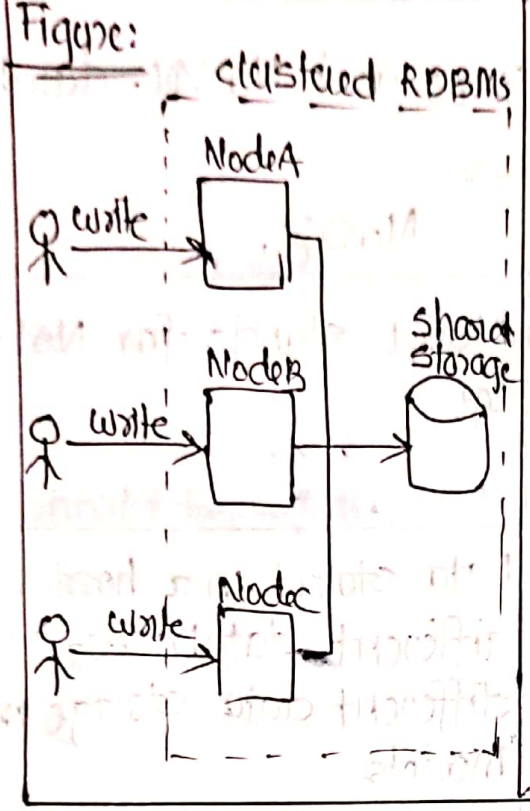
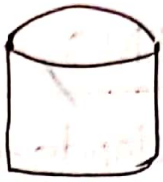


Figure:



Q-6 What are the features of NoSQL storage devices.

Ans: Not-only SQL (NoSQL) refers to technologies used to develop next generation non-relational databases that are highly scalable and fault-tolerant.

Features

1. Schema-less data model - Data can exist in its raw form.
2. Scale out rather than scale up - More nodes can be added to obtain additional storage with a NoSQL database.
3. Highly available - This is built on cluster-based technologies that provide fault tolerance out-of the box.

4. Lower operational costs - Many NoSQL databases are built on open source platforms with no licensing costs.

5. Eventual consistency - Data reads across multiple nodes but may not be consistent immediately after a write. However, all nodes will eventually be in a consistent state.

6. Base, not ACID - BASE compliance requires a database to maintain high availability in the event of network/node failure, while not requiring the database to be in a consistent state whenever an update occurs.

7. API driven data access - Data access is generally supported via API based queries, including RESTful APIs, whereas some implementations may also provide SQL-like query capability.

8. Auto sharding and replication - To support horizontal scaling and provide high availability, a NoSQL storage device automatically employs sharding and replication.

9. Integrated caching - This removes the need for a third-party distributed caching layer.

10. Distributed query support - NoSQL storage devices maintain consistent query behavior across multiple shards

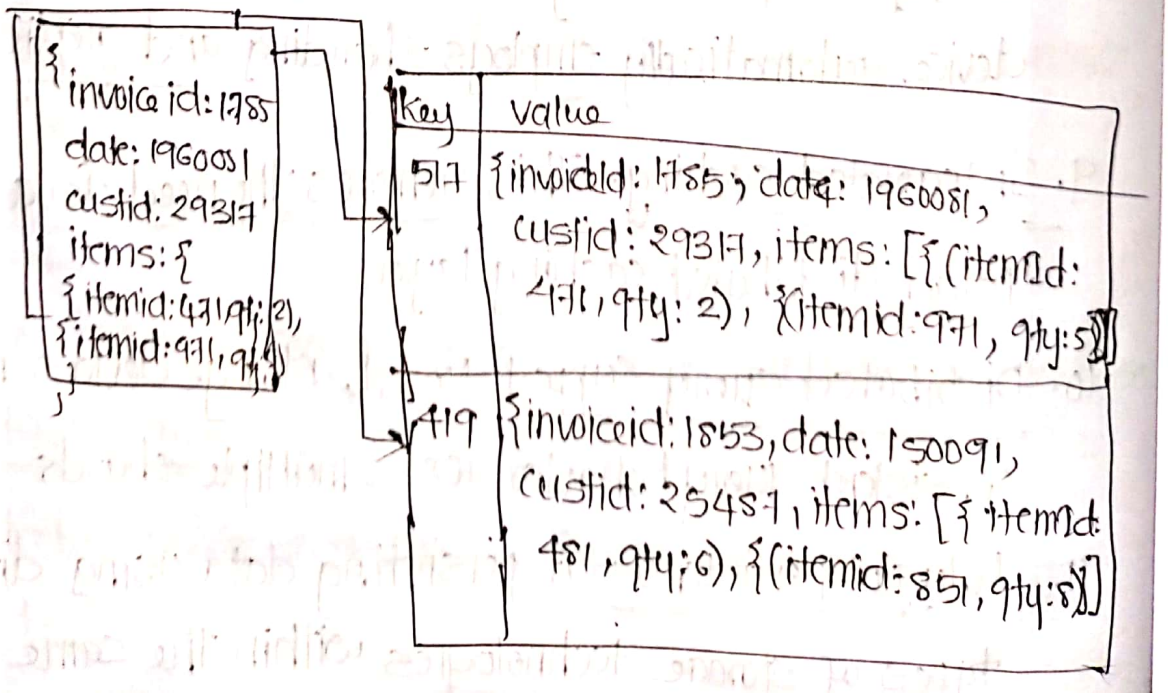
11. Polyglot persistence - If persisting data using different types of storage technologies within the same solution

architecture.

12. Aggregate-focused - NoSQL storage devices store de-normalized aggregate data thereby eliminating the need for joins and extensive mapping between application objects and the data stored in the database.

Q-07 Briefly write about the NoSQL storage device used by MongoDB.

Ans: Document storage device also store data as key-value pairs. However, unlike key-value storage devices, the stored value is a document. These documents can have a complex nested structure.



* A document storage device is appropriate when:
→ storing semi-structured document-oriented data comprising flat or nested schema.
→ schema evolution is a requirement as the structure of the document is either unknown or is likely to change
→ applications require a partial update of the aggregate stored as a document.
→ searches need to be performed on different fields of the documents.
→ storing domain objects, such as customers in serialized object form.
→ query patterns involve insert, select, update and delete operations.

* A document storage device is inappropriate when:
→ multiple documents need to be updated as part of a single transaction.
→ performing operations that need joins between multiple documents or storing data that is normalized.
→ schema enforcement for achieving consistent query design.
→ the stored value is not self-describing.
→ binary data needs to be stored.

Examples of document storage devices include MongoDB, CouchDB and Terrastore.

Q.8 Explain the different types of NoSQL databases

Ans: NoSQL storage devices can mainly be divided into four types:-

1. Key-value
2. document
3. column-family
4. graph.

1. Key-value

Key-value storage devices store data as key-value pairs and act like hash tables. The table is a list of values where each value is identified by a key. value look-up can only be performed via the keys. Partial updates are not possible. Key value storage devices generally do not maintain any indexes, therefore writes are quite fast. Based on a simple storage model; key-value storage devices are highly scalable.

* A key-value storage device is appropriate when:
  - → unstructured data storage is required.
  - → high performance read/writes are required.
  - → the value is fully identifiable via the key alone.
  - → the value is a standalone entity that is not dependent on other values.

* A key-value storage device is inappropriate when:

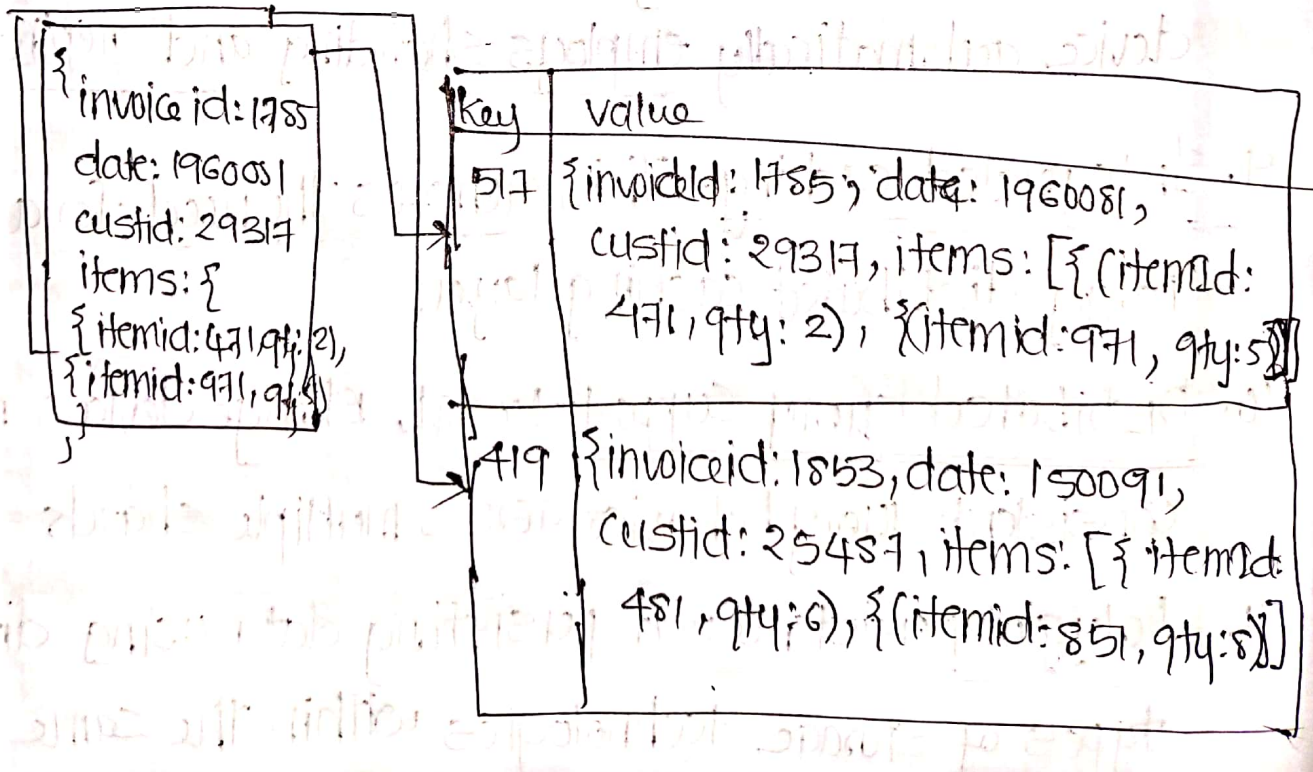
→ applications require searching or filtering data using attributes of the stored value

→ relationships exist between different key-value entries

→ multiple-keys require manipulation in a single operation.

2. Document.

Document storage device also store data as key-value pairs. However, unlike key-value storage devices the stored value is a document. These documents can have a complex nested structure.



* A document storage device is appropriate when:
→ storing semi-structured document-oriented data comprising flat or nested schema.
→ schema evolution is a requirement as the structure of the document is either unknown or is likely to change
→ applications require a partial update of the aggregate stored as a document.
→ searches need to be performed on different fields of the documents.
→ storing domain objects, such as customers in serialized object form.
→ query patterns involve insert, select, update and delete operations.

* A document storage device is inappropriate when:
→ multiple documents need to be updated as part of a single transaction.
→ performing operations that need joins between multiple documents or storing data that is normalized.
→ schema enforcement for achieving consistent query design.
→ the stored value is not self-describing.
→ binary data needs to be stored.

Examples of document storage devices include MongoDB, CouchDB and Terrastore.

3. Column-family

It stored data much like a tradition RDBMS but group related columns together in a row, resulting in column-families. Each column can be a collection of related columns itself, refered to as a super-column. Each row consists of multiple column families. It provide fast data access with random read/write capability.

* A column-family storage device is appropriate when
→ realtime random read/write capability.
→ data represents a tabular structure.
→ support for schema evaluation
→ effience use of storage
* A column-family storage device is inappropriate when.
→ relational data access is required
→ ACID transactional support is required.
→ SQL-complaint queries need to be executed
→ query patterns are likely to change frequently.

4. Graph

Graph storage devices are used to persist interconnected entities. If emphasis is on the structure of the entities, graph storage devices place emphasis on storing the linkages between entities. Entites are stored as nodes and are also called vertices

* A graph storage device is appropriate when:

→ interconnected entities need to be stored

→ querying entities based on the type of relationship

→ finding groups of interconnected entities

→ mining data with a view toward finding pattern.

* A graph storage device is inappropriate when:

→ Involves searching for nodes or edges.

→ Binary storage is required

→ queries based on the selection of node/edge attribute dominate node traversal queries.

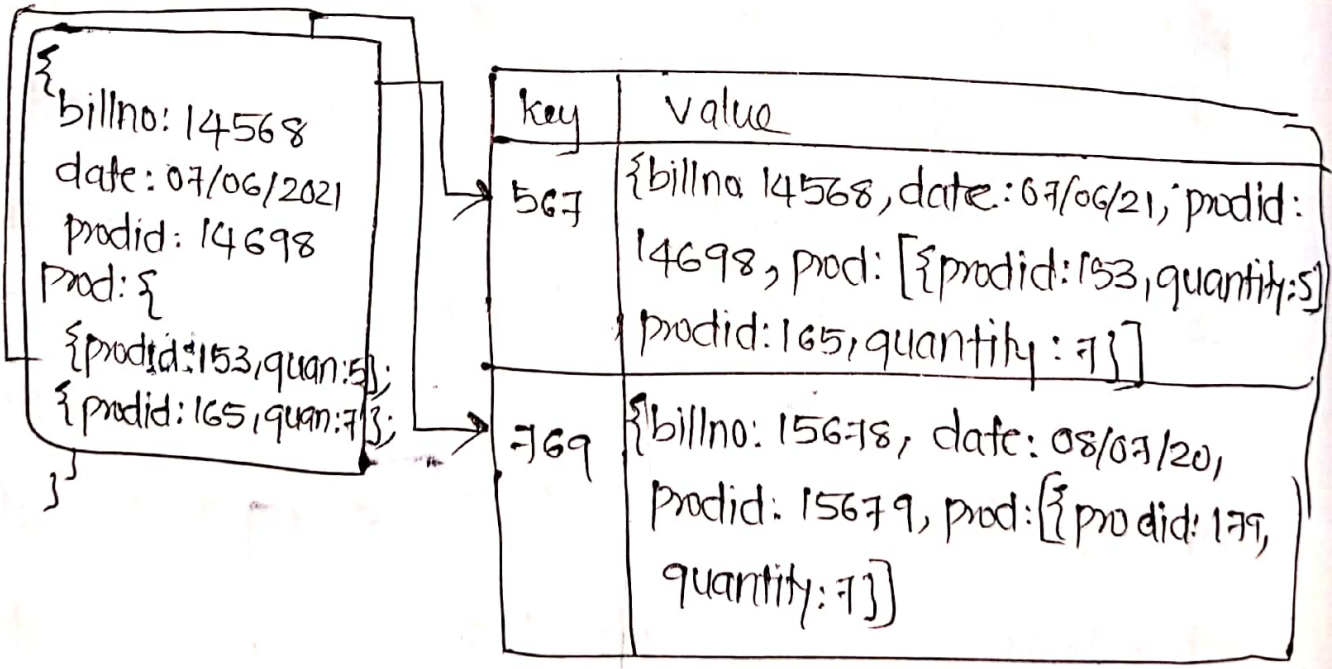
Eg: Neo4J, OrientDB.

Types of NoSQL - Eg:

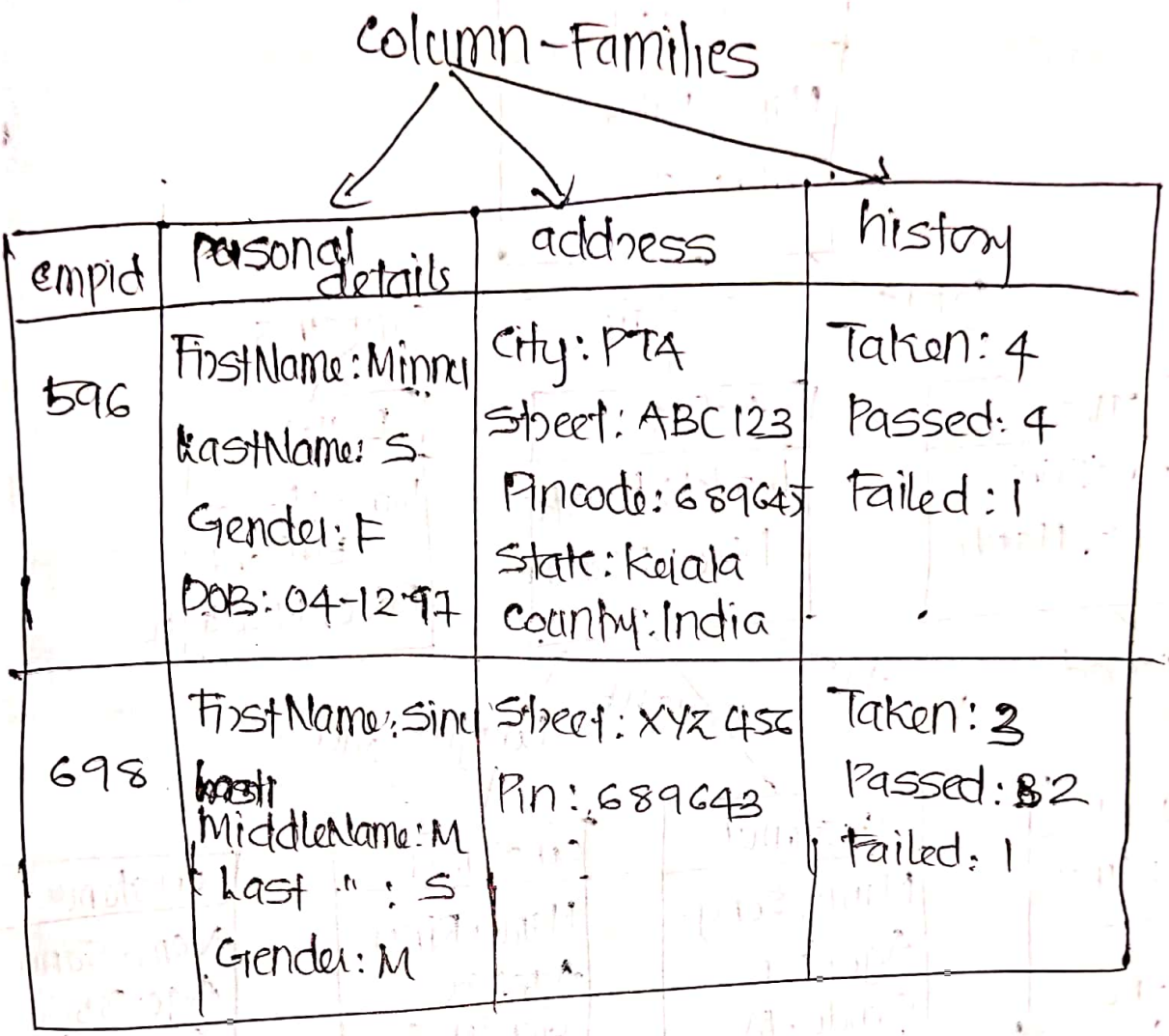
1) Key-Value

Key	value	
859	Miller, 12-10-45-30, Good Product Service	← Text
465	10101100111011110000 11100110	← image
785	<prod_id>1567</prod_id><Total>1495</Total>	← XML

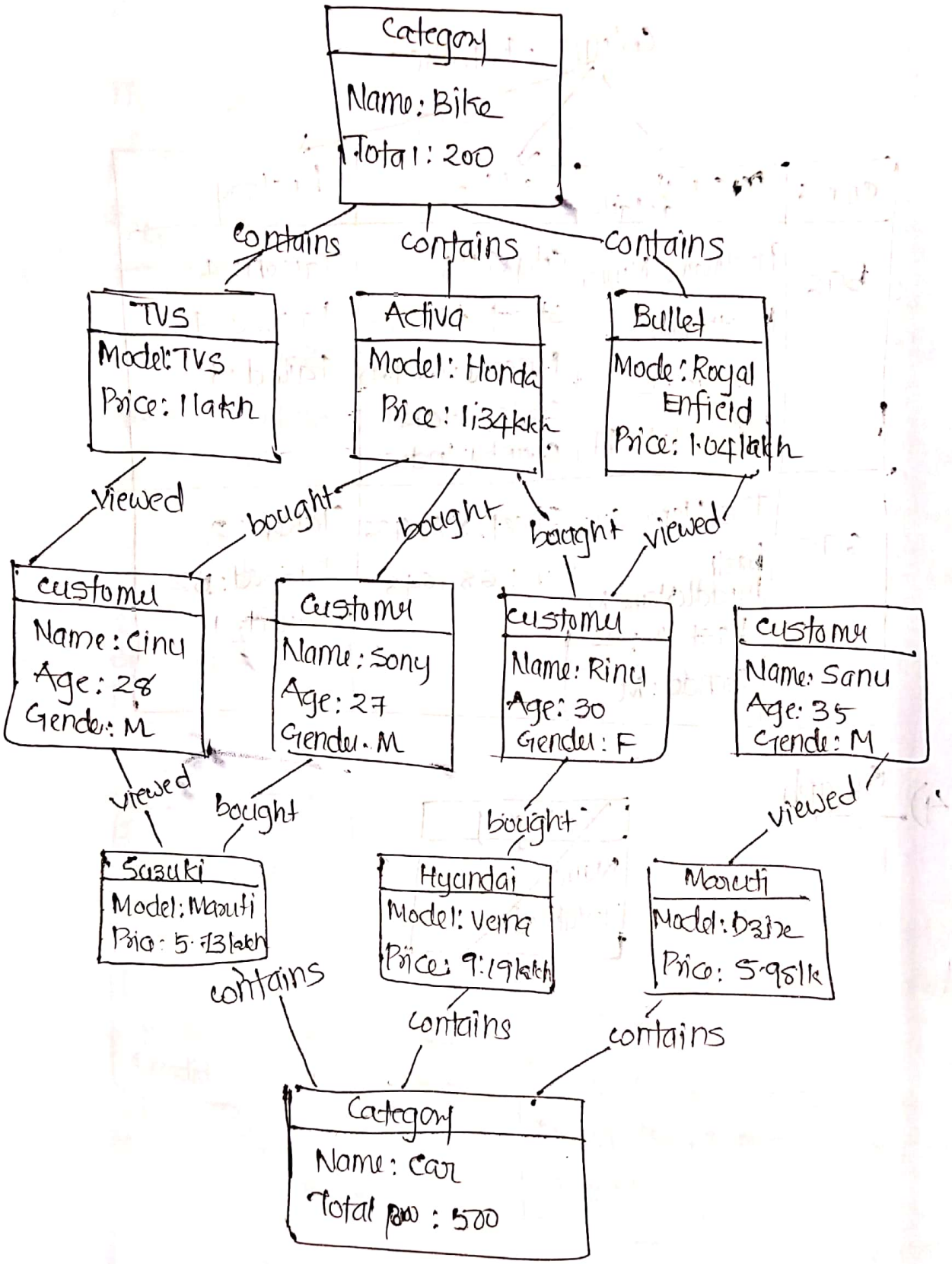
2) Document



3) Column-Family



4) Graph



Q-9 Differentiate between Document storage device and key value storage devices.

Ans: → Document storage devices are value aware
→ the stored value is self-describing.
→ A select operation can reference a field inside the aggregate value.
→ A select operation can retrieve a part of the aggregate value
→ Partial updates are supported, therefore a subset of the aggregate can be updated.
→ indexes that speed up searches are generally supported.

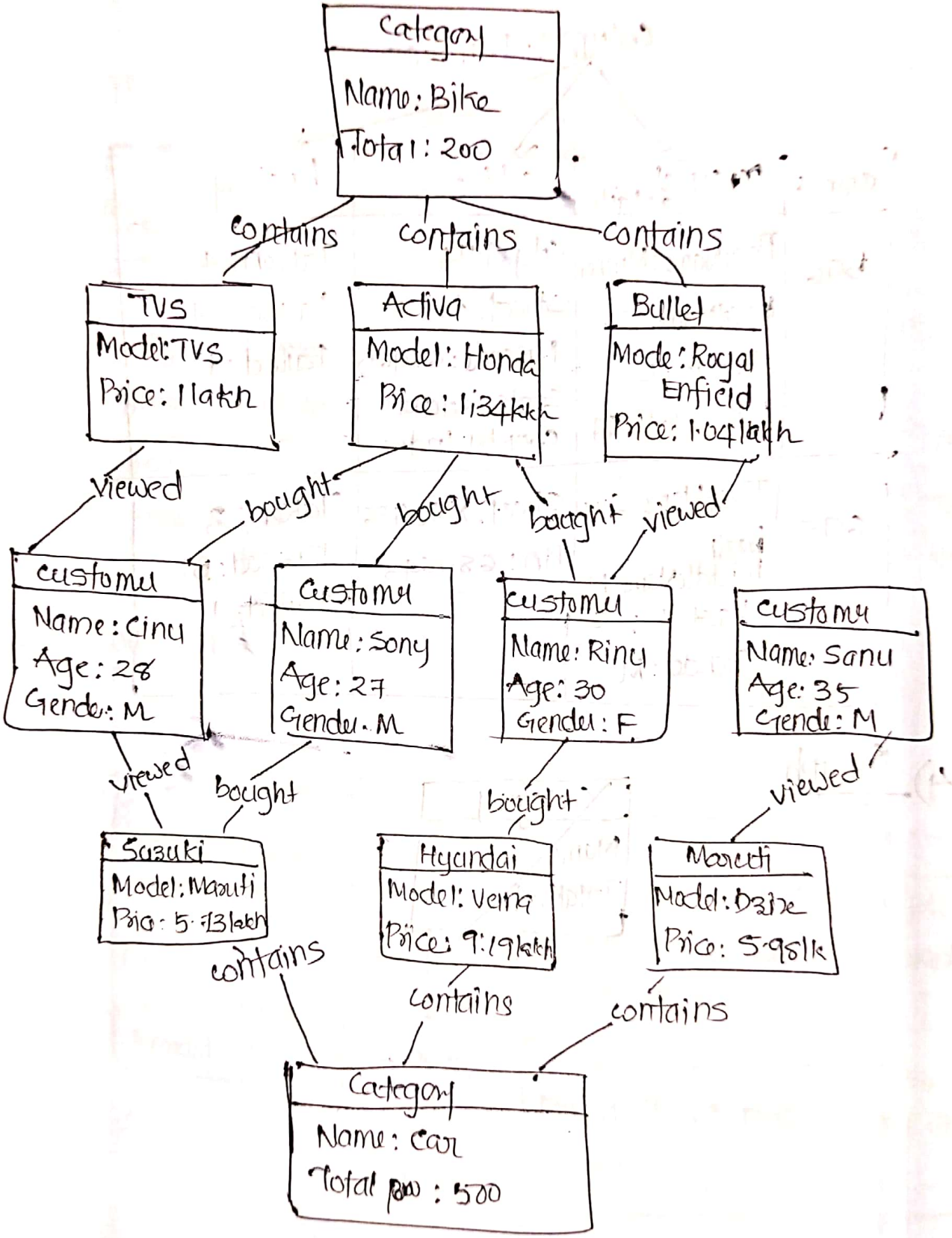
Q-10 Write a short note on Graph based storage device. When it is appropriate to use?

A: Graph storage devices are used to persist interconnected entities. If emphasis is on the structuring the entities, graph storage devices place emphasis on storing the linkages between entities. Entites are stored as nodes and are also called vertices

* A graph storage device is appropriate when:

→ interconnected entities need to be stored
→ querying entities based on the type of relationship
→ finding groups of interconnected entities
→ mining data with a view toward finding pattern.

4) Graph - Eg:



Q-11 Differentiate between NewSQL and NoSQL databases.

Ans:

NoSQL	NewSQL
NoSQL is a schema-free database	NewSQL is schema-fixed as well as a schema free database
It is horizontally scalable.	It is also horizontally scalable.
It possesses automatically high-availability	It possesses built-in high availability.
It promotes CAP properties	It promotes ACID properties
It is not used for developing OLTP systems (Online Transactional Processing)	It is used for developing OLTP systems with very high volumes of transactions.

There are low-security concerns Examples of NoSQL databases include DynamoDB, MongoDB	There are moderate security concerns. Examples of NewSQL databases include VoltDB, NuoDB, InnoDB

Q-12. List out the advantages of NewSQL databases.

Ans:→ It is less complex applications, greater consistency.

→ It support SQL compliant syntax for data definition and data manipulation operations.

→ It is use a logical relational data model for data storage.

→ It introduces new implementations to traditional relational databases.

Q-13. What are NewSQL databases?

Ans: NewSQL storage devices combine the ACID properties of RDBMS with the scalability and fault tolerance offered by NoSQL storage devices. It generally support SQL compliant syntax for data definition and data manipulation operations, and they often use a logical

relational data model for data storage.

NewSQL databases can be used for developing OLTP systems with very high volumes of transactions.

Eg: Banking system.

They can also be used for realtime analytics.

Eg: operational analytics, as some implementations leverage in-memory storage.

Q-2 Explain the different methods for implementing on disk storage devices.

Ans: Refer 3rd, 4th, 5th, 6th, 8th, & 13th answers.

Q-14 Write a short note on the in-memory storage device

Ans: In-memory storage device generally utilizes RAM, the main memory of a computer, as its storage medium to provide fast data access. The main advantage is to storage of data in memory eliminates the latency of disk I/O and the data transfer time between the main memory and the hard drive.

Cluster-based memory enables storage of large amounts of data, including Big data datasets, which can be accessed considerably faster when compared with an on-disk storage device. Its access time is very low.

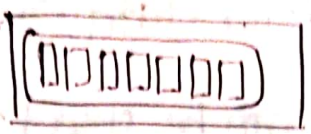


Fig: The symbol used to represent an in-memory storage device.

In-memory analysis of data, such as generating statistics by executing queries on data that is stored in memory instead of on disk. In-memory analytics enable operational analytics and operational BI through fast execution of queries and algorithms.

* An in-memory storage device is appropriate when:
  - → data arrives at a fast pace and requires realtime analytics or event stream processing.
  - → continuous or always-on analytics is required, such as operational BI and operational analytics.
  - → Interactive query processing.
  - → The same dataset is required by multiple data processing jobs.
  - → Performing exploratory data analysis.
  - → Developing low latency Big Data solutions

* An in-memory storage device is inappropriate when:
  - → when data processing consists of batch processing.
  - → very large amounts of data need to be persisted in-memory for a long time in order to perform in-depth

analysis.

→ Datasets are extremely large and do not fit into the available memory

→ An enterprise has a limited budget.

Q-15 Explain the two methods used to implement in-memory storage.

Ans: In-memory storage devices can be implemented as:

(i) In-Memory Data Grid (IMDG)

(ii) In-Memory Database.

(i) In-Memory Data Grids

IMDGs store data in memory as key-value pairs across multiple nodes. This supports schema-less data storage through storage of semi/unstructured data.

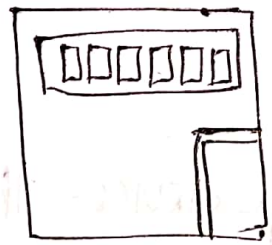


fig: The symbol used to represent an IMDG.

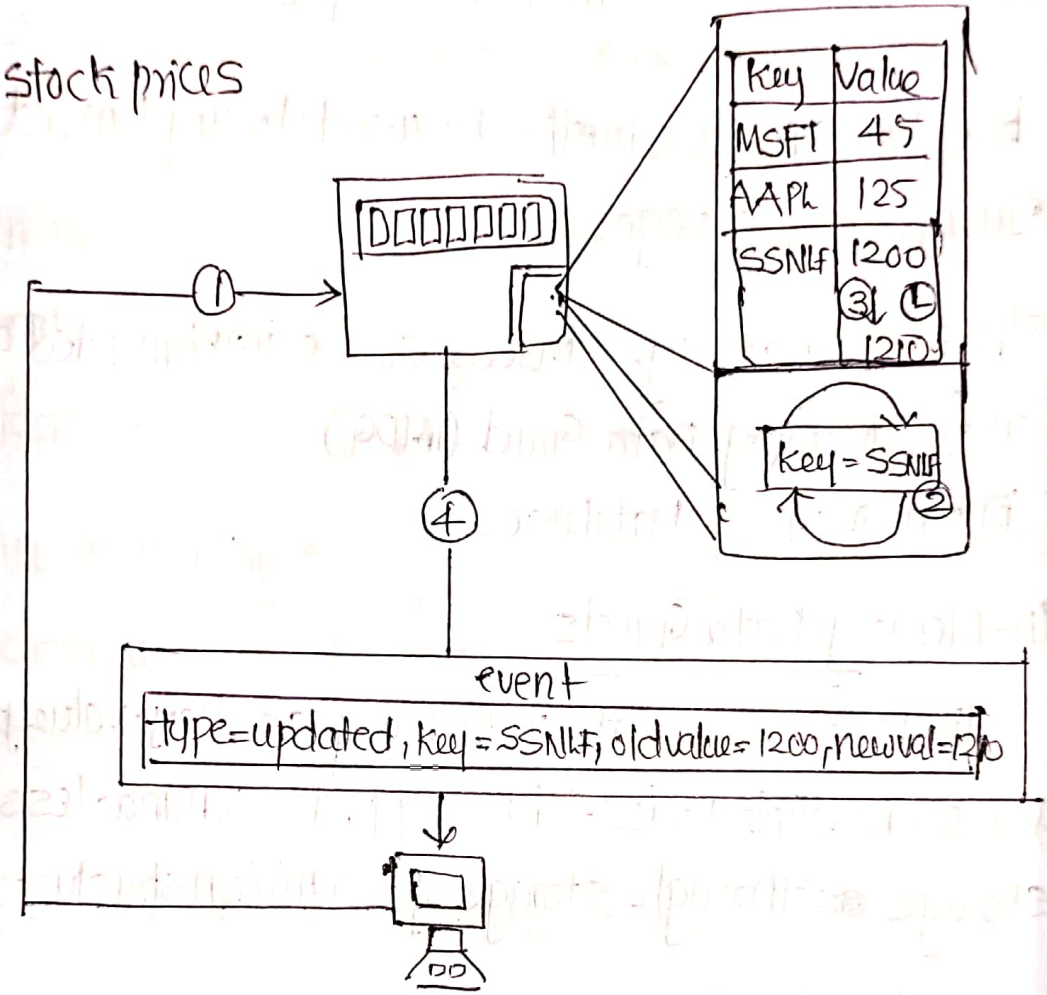
Advantage

IMDG provide faster data access. Nodes in IMDGs keep themselves synchronized and collectively provide high availability, fault-tolerance and consistency.

IMDGs scale horizontally by implementing data partitioning and data replication. IMDGs are heavily used for realtime analytics, because they support complex

Event Processing (CEP) via the publish-subscribe messaging model. This is achieved through a feature called continuous querying:

Eg: Stock prices



(ii) In Memory Database

IMDBs are in-memory storage devices that employ database technology. An IMDB can be relational in nature for the storage of semi structured data and unstructured data. Relational IMDBs make use of the more familiar SQL language. NoSQL based IMDBs generally provide API-based access.

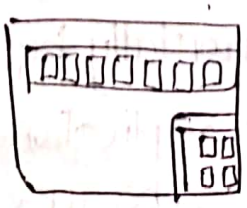
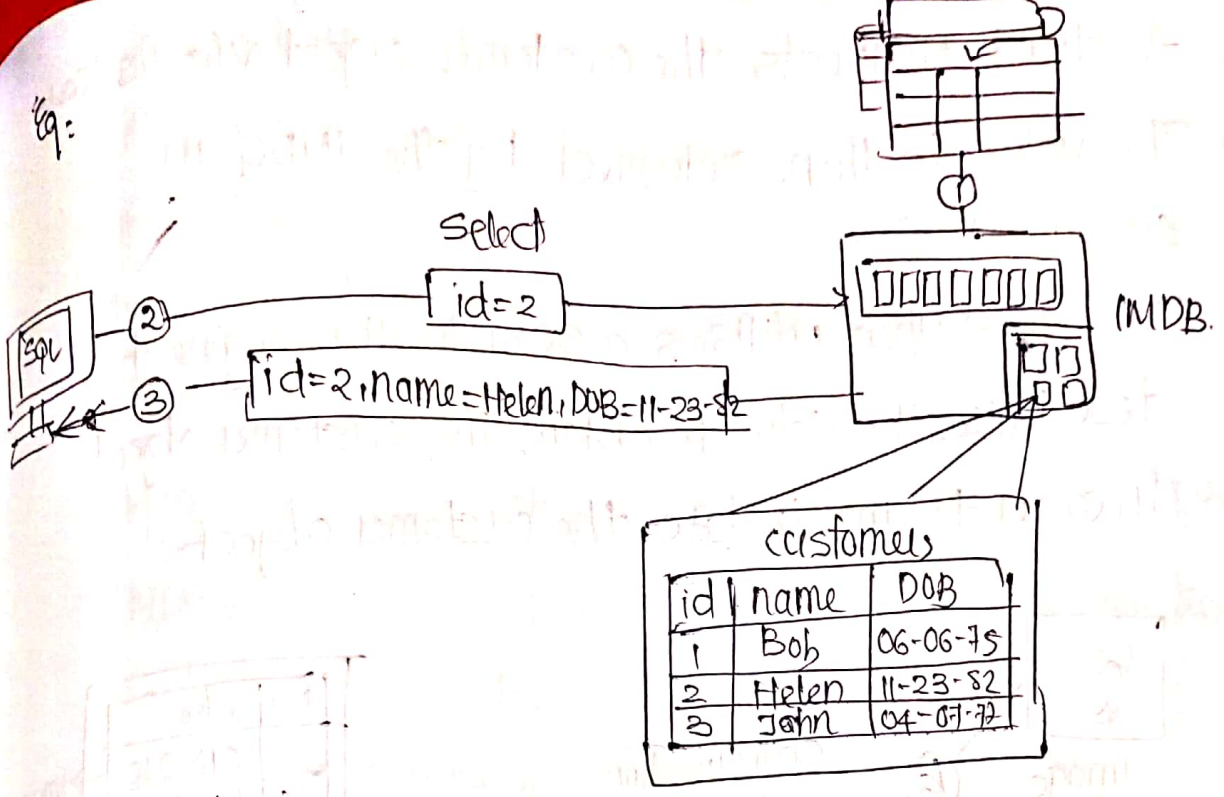


Fig: The symbol used to represent an IMDB.

Eg:



Strategies:

→ Use of Non-volatile RAM (NVRAM) for storing data permanently

→ Database transaction logs can be periodically stored to a non-volatile medium, such as disk.

→ Snapshot files, which capture database state at a certain point in time, are saved to disk.

→ An IMDB may leverage sharding and replication to support increasing availability and reliability.

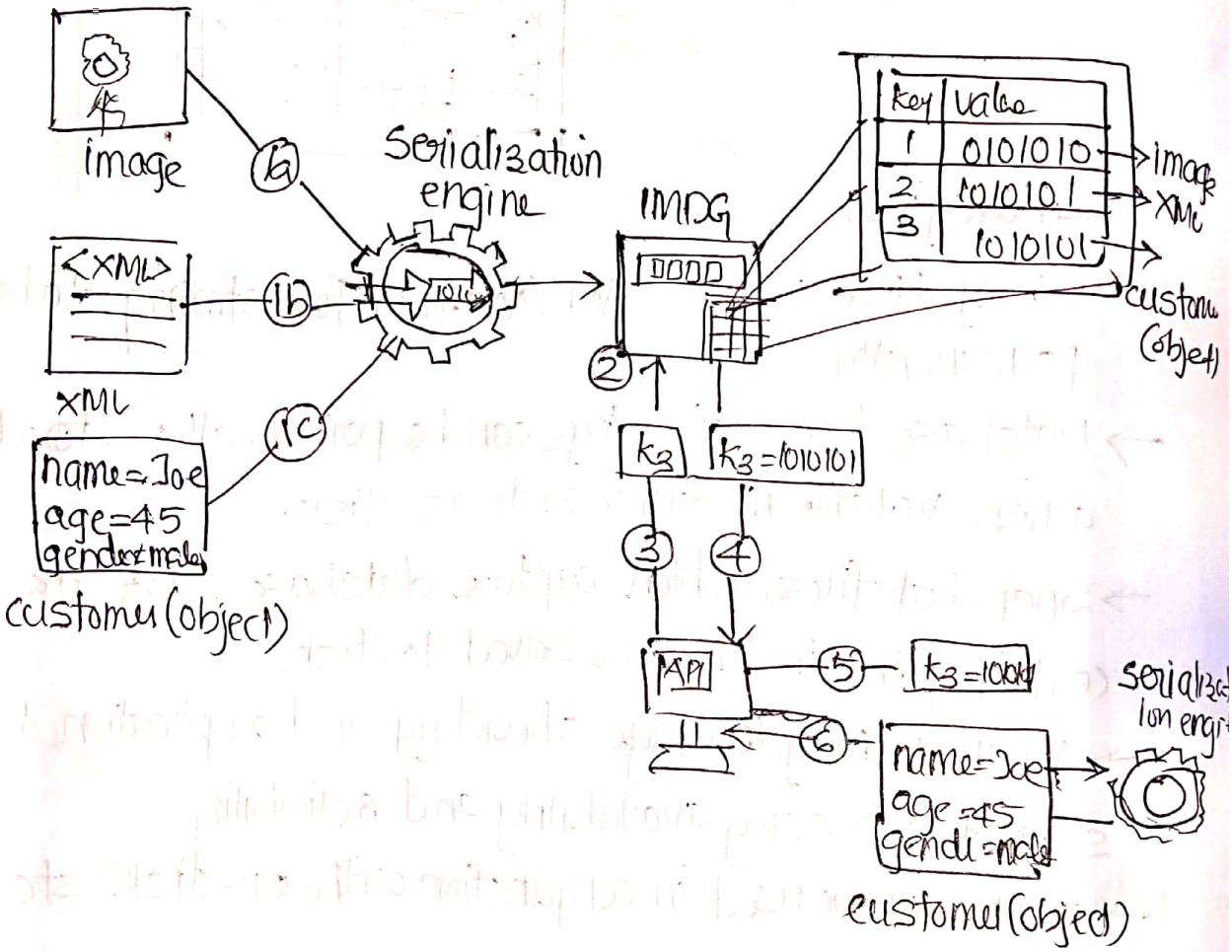
→ IMDBs can be used in conjunction with on-disk storage devices.

Q-16 Explain the role of serialization engine in IMDG using an example.

Ans: 1. An image (a), XML data (b) and a customer object (c) are first serialized using serialization engine.

2. They are then stored as key-value pairs in an IMDG.

3. A client requests the customer object via its key.
4. The value is then returned by the IMDG in serialized form.
5. The client then utilizes a serialization engine to deserialize the value to obtain the customer object
6. Finally to manipulate the customer object



Q-17 Explain the various approaches used by the IMDG used to support the read/write performance in the Big Data environment.

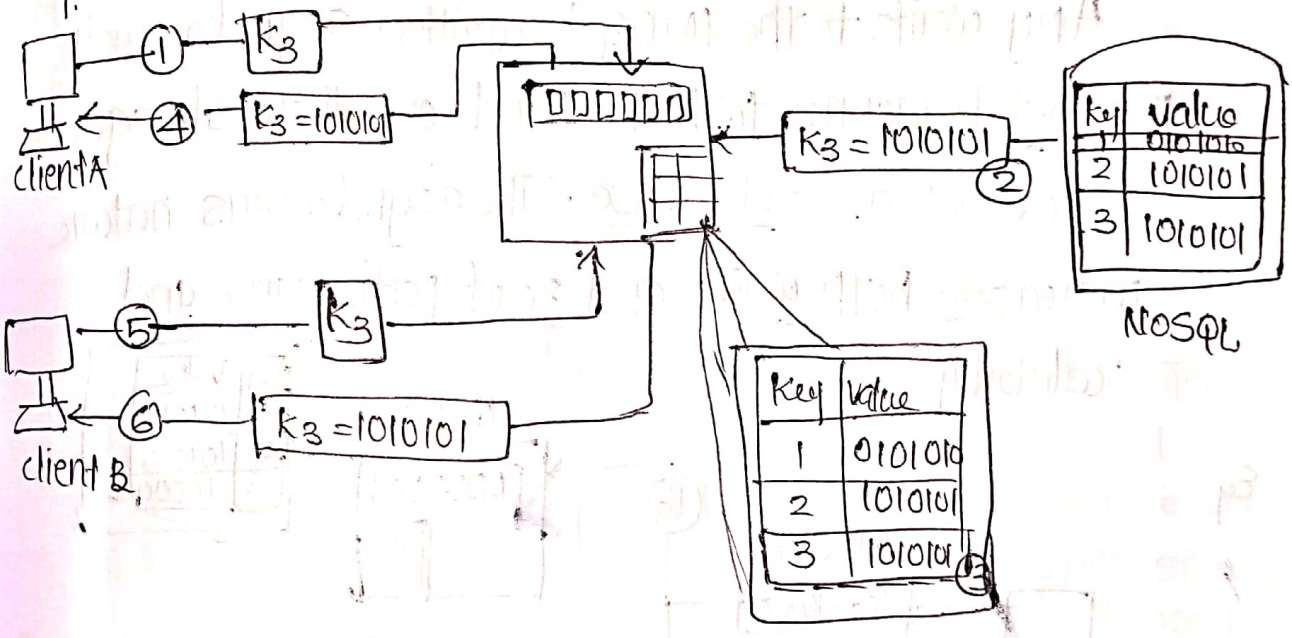
Ans: The various approaches that can be combined as necessary to support read/write performance consistency and simplicity requirements:

(i) read-through
(ii) write-through
(iii) write-behind
(iv) refresh-ahead

(i) Read-through

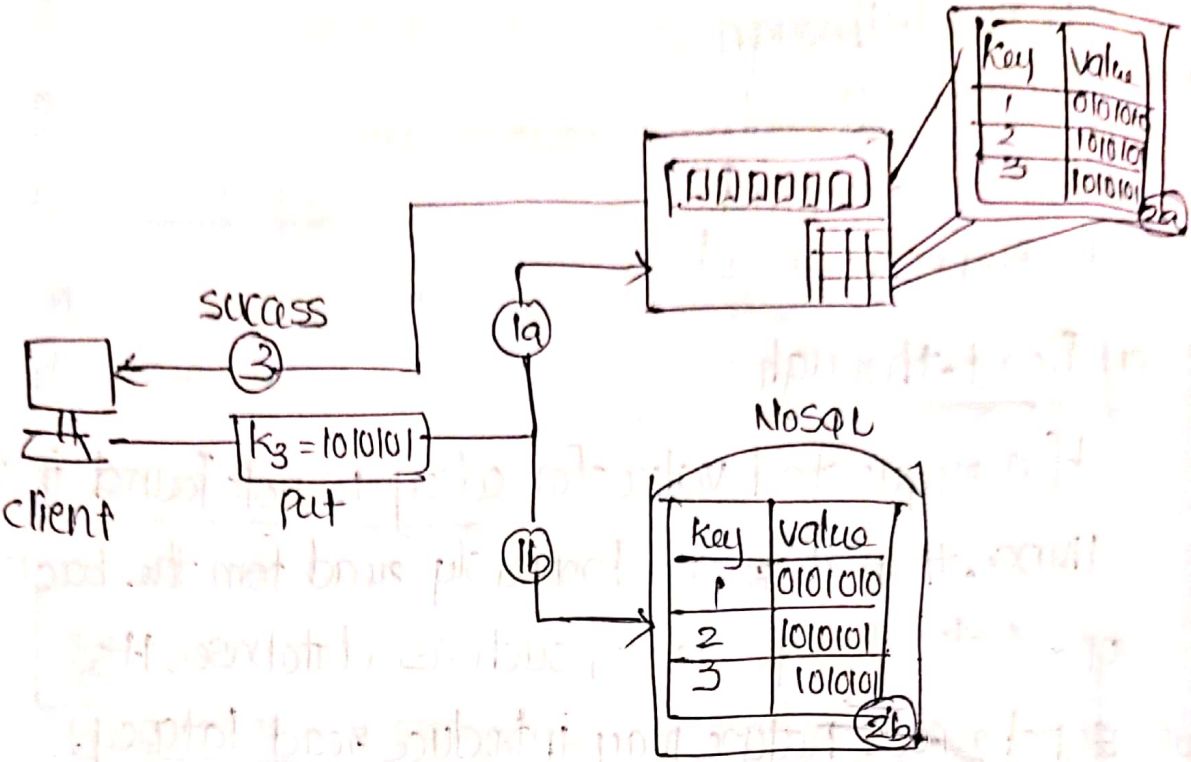
If a requested value for a key is not found in the IMDG, then it is synchronously read from the backend on-disk storage device, such as database. Its synchronous nature may introduce read latency.

Eg:



(ii) Write-through

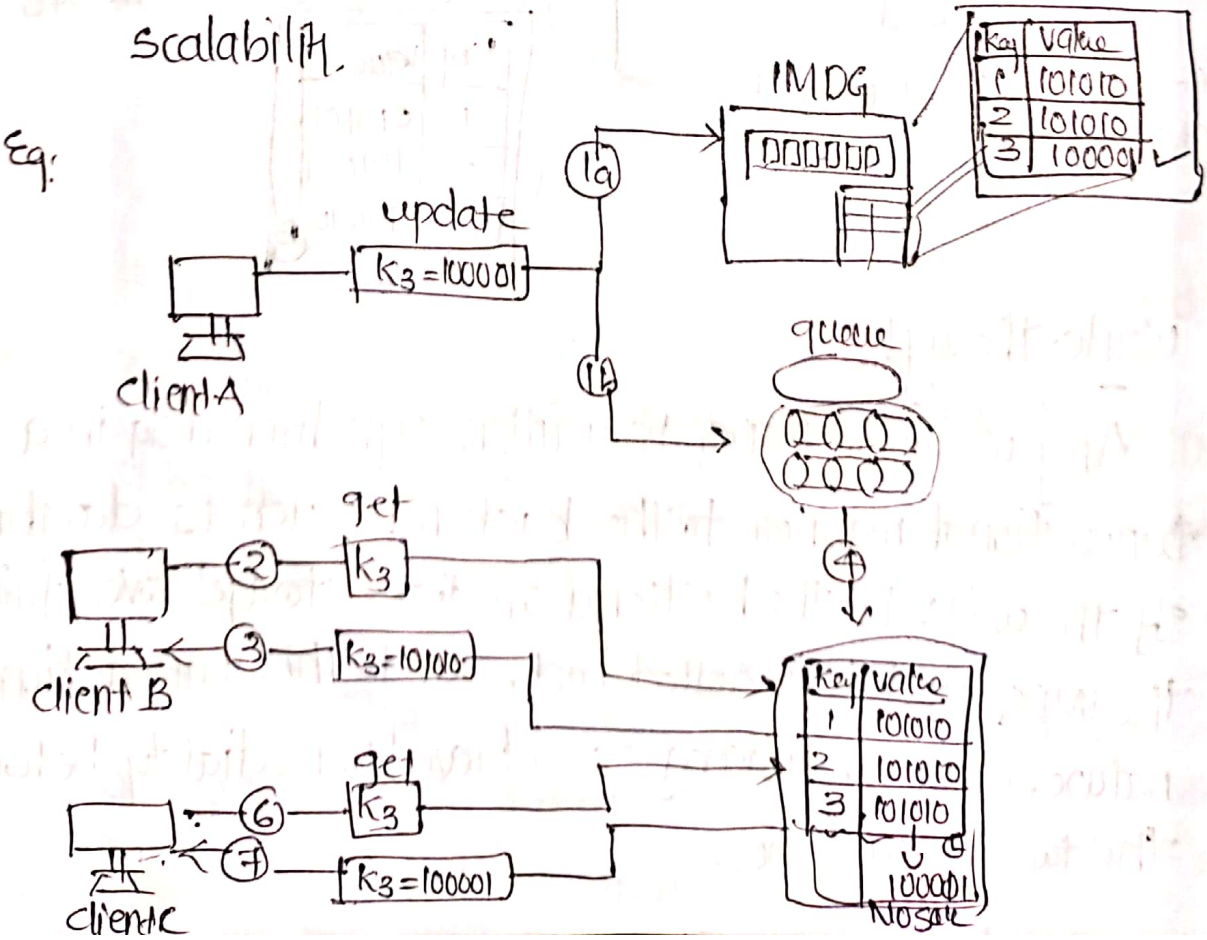
Any write to IMDG is written synchronously in a transactional manner to the backend such as database. If the write to the backend on disk storage device fails the IMDG update is rolled back. Due to this transactional nature, data consistency is achieved immediately between the two data stores.



(iii) Write-behind

Any write to the IMDG is written asynchronously in a batch manner to the backend on-disk storage device, such as a database. The asynchronous nature increases both write and read performance and scalability.

Eg:



(iv) Refresh-ahead OR Q-22 Write an eg of IMDG leveraging the refresh-ahead approach.

Refresh-ahead is a proactive approach where any frequently accessed values are automatically, asynchronously refreshed in the IMDG, provided that the value is accessed before its expiry time as configured in the IMDG.

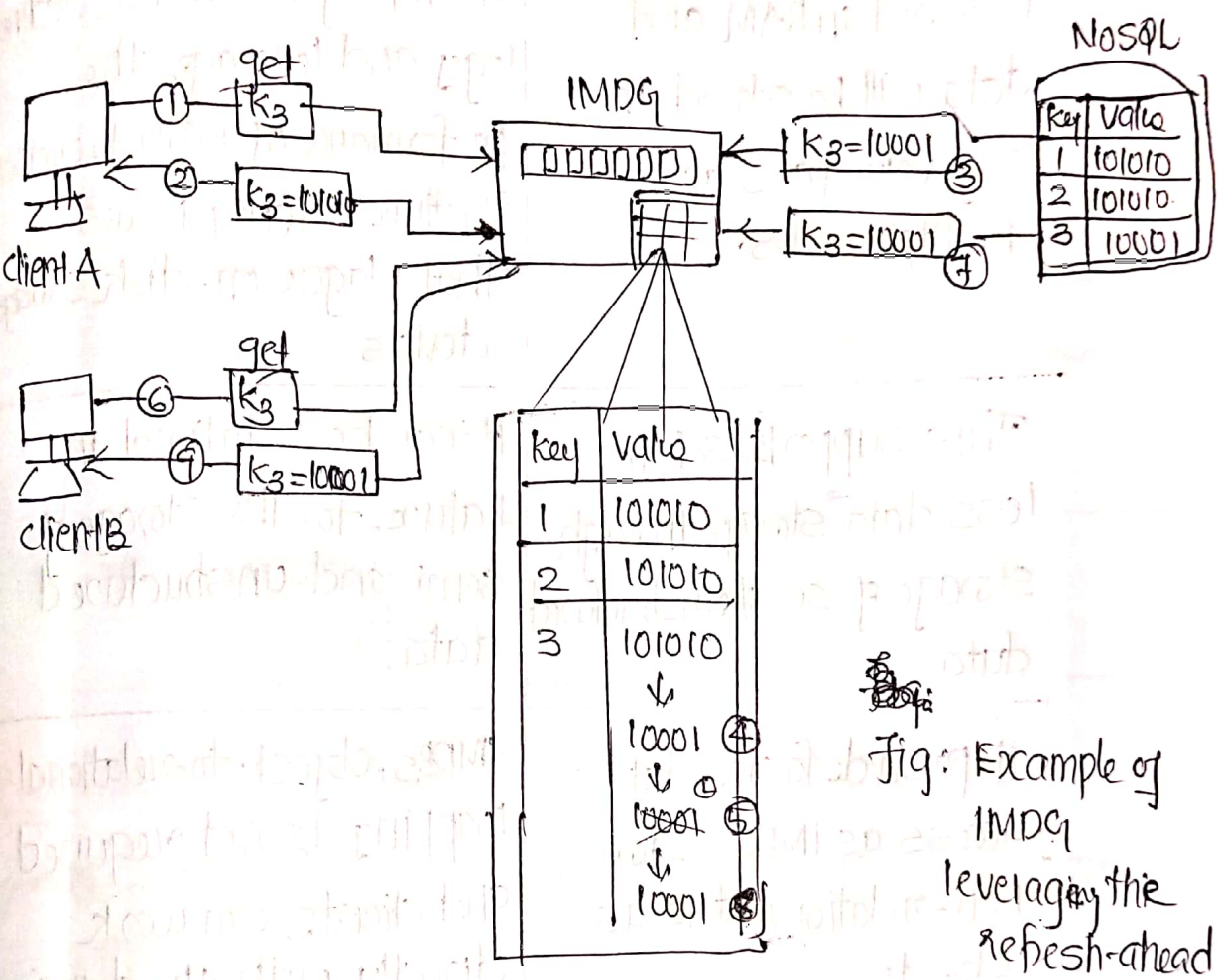


Fig: Example of IMDG leveraging the refresh-ahead approach.

Q-18 Explain the write through and write behind approach used by IMDG.

A: Refer 17th ans - write through, write behind

Q-19 Write an example of IMDG storage configured with a continuous query.

IMDGs store data in memory as key-value pairs across multiple nodes. This supports schema-less data storage through storage of semi/unstructured data.

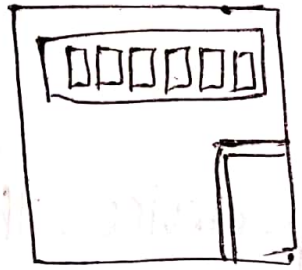


fig: the symbol used to represent an IMDG.

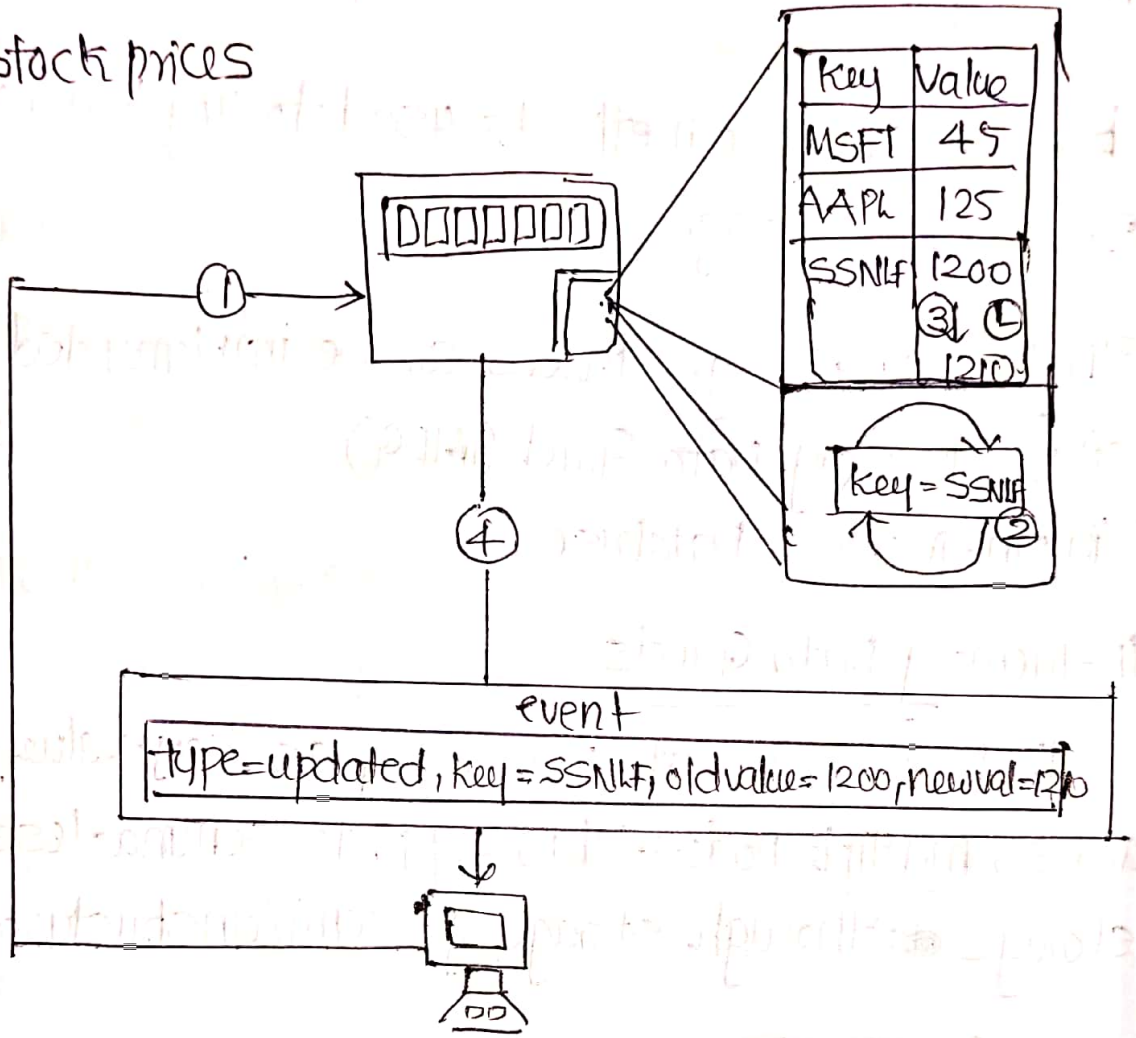
Advantage

IMDG provide faster data access. Nodes in IMDGs keep themselves synchronized and collectively provide high availability, fault-tolerance and consistency.

IMDGs scale horizontally by implementing data partitioning and data replication. IMDGs are heavily used for realtime analytics, because they support complex

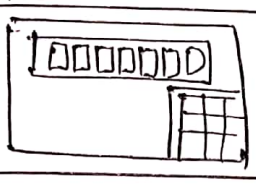
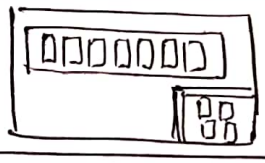
Event Processing (CEP) via the publish-subscribe messaging model. this is achieved through a feature called continuous querying :

eg: stock prices



Q-20 Differentiate between IMDG and IMDB.

IMDG	IMDB
IMDG stands for In-Memory Data Grid.	IMDB stands for In-memory Database.
It stored in RAM and data will be stored in key-value pairs across multiple nodes.	It employ database technology and leverage the performance of RAM to overcome runtime latency issues that plague on-disk storage devices
This supports schema-less data storage through storage of semi/unstructured data.	It can be relational in nature for the storage of semi and unstructured data.
It provide faster data access as IMDGs store non-relational data as objects.	IMDBs, object-to-relational mapping is not required and clients can work directly with the domain specific objects.
It scale horizontally by implementing data partitioning and data replication.	It also scale horizontally depending on the underlying implementation.
It support reliability by replicating data to atleast one extra node.	Not all IMDB implementations directly support durability, but instead leverage various

	strategies for providing durability.
IMDG support the continuous querying feature.	IMDB may also support the continuous query feature.
IMDGs are heavily used for realtime analytics	It's also heavily used in realtime analytics.
IMDG provide faster data access	IMDBs provide an easy to set-up in-memory data storage.
Examples include Hazelcast and Oracle Coherence	Examples include Memsql, extreme DB.
Fig: 	Fig: 

Q21 When is an IMDG storage device appropriate?

Ans:→ Data needs to be readily accessible in object form with minimal latency.

→ Data being stored in non-relational in nature such as semi-structured and unstructured data.

→ Adding realtime support to an existing Big Data solution currently using on-disk storage.

→ The existing storage device cannot be replaced but the data access layer can be modified.