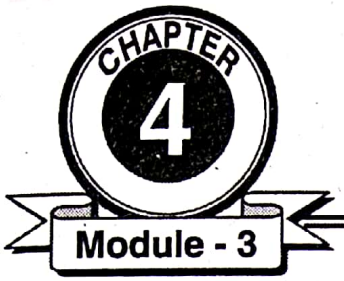


NoSQL

Syllabus

Introduction to NoSQL, NoSQL Business Drivers, NoSQL Data Architecture Patterns: Key-value stores, Graph stores, Column family (Bigtable)stores, Document stores, Variations of NoSQL architectural patterns, NoSQL Case Study, 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.

4.1 NoSQL (What is NoSQL?)

MU - May 17

Q. What is NoSQL? **(May 17, 2 Marks)**

1. History

- The term NoSQL was first used by Carlo Strozzi in the year 1998.
- He mentioned this name for his Open Source Database system in which there was no provision of SQL Query interface.
- In the early 2009, at conference held in USA, NoSQL was comes into picture and actually comes in practice.

2. Overview

- NoSQL is a not a RDBMS (Relational Database Management System).
- NoSQL is specially designed for large amount of data stored in distributed environment.
- The important feature of NoSQL is, it is not bounded by table schema restrictions like RDBMS. It gives options to store some data even if there is no such column is present in table.
- NoSQL generally avoids join operations.

3. Need

- In real time, data requirements are changed a lot. Data is easily available with Facebook, Google+, Twitter and others.
- The data that includes user information, social graphs, geographic location data and other user-generated content.
- To make use of such abundant resources and data, it is necessary to work with a technology which can operate such data.
- SQL databases are not ideally designed to operate such data.
- NoSQL databases specially designed for operating huge amount of data.

4. Advantages

(i) Good resource scalability.

(ii) Lower operational cost.

(iii) Supports semi-structure data.

(iv) No static schema.

(v) Supports distributed computing.

(vi) Faster data processing.

(vii) No complicated relationships.

(viii) Relatively simple data models.

5. Disadvantages

(i) Not a defined standard.

(ii) Limited query capabilities.

6. Companies working with NoSQL

(i) Google (ii) Facebook

(iii) LinkedIn (iv) McGraw-Hill Education

4.2 NoSQL Basic Concepts

Theorem (Brewer's Theorem) for NoSQL

CAP theorem states three basic requirements of NoSQL databases to design a distributed architecture.

(a) Consistency

Database must remain consistent state like before, even after the execution of an operation.

(b) Availability

It indicates that NoSQL system is always available without any downtime.

(c) Partition Tolerance

This means that the system continues to function even the communication failure happens between servers i.e. if one server fails, other server will take over.

Note : It is very difficult to fulfill all the above requirements.

There are many combinations of NoSQL rules :

1. CA

- It is a single site cluster.
- All nodes are always in contact.
- Partitioning system can block the system.

2. **CP**

   Some data may not be accessible always still it may be consistent or accurate.

3. **AP**

   - System is available under partitioning.
   - Some part of the data may be inconsistent.

4. **BASE model**

   - Relational databases have some rules to decide behaviour of database transactions.
   - ACID model maintains the atomicity, consistency, isolation and durability of database transactions.
   - NoSQL turns the ACID model to the BASE model.

5. **BASE offers some guidelines**

   - Basic availability
   - Soft state
   - Eventual consistency

6. **Data storage**

   - NoSQL databases use the concept of a key / value store.
   - There are no schema restrictions for NoSQL database.
   - It simply stores values for each key and distributes them across the database, it offers efficient retrieval.

7. **Redundancy and Scalability**

   - To add redundancy to a database, we can add duplicate nodes and configure replication.
   - Scalability is simply a matter of adding additional nodes. There can be hash function designed to allocate data to server.

4.3 Case Study NoSQL (SQL vs NoSQL)

SQL databases are Relational Databases (RDBMS); whereas NoSQL database are non-relational database.

Data storage

- SQL databases stores data in a table whereas NoSQL databases stores data as document based, key-value pairs, graph databases or wide-column stores.
- SQL data is stored in form of tables with some rows.
- NoSQL data is stored as collection of key-value pair or documents or graph based data with no standard schema definitions.

Database schema

SQL databases have predefined schema which cannot be change very frequently, whereas NoSQL databases have dynamic schema which can be change any time for unstructured data.

Complex queries

- SQL databases provides standard platform for running complex query.
- NoSQL does not provide any standard environment for running complex queries.
- NoSQL are not as powerful as SQL query language.

Comparison between SQL and NoSQL

Sr. No.	SQL	NoSQL
1.	Full form is Structured Query Language.	Full form is Not Only SQL or Non-relational database.
2.	SQL is a declarative query language.	This is Not a declarative query language.
3.	SQL databases works on ACID properties, Atomicity Consistency Isolation Durability	NoSQL database follows the Brewers CAP theorem, Consistency Availability Partition Tolerance
4.	Structured and organized data	Unstructured and unreplicable data
5.	Relational Database is table based.	Key-Value pair storage, Column Store, Document Store, Graph databases.
6.	Data and its relationships are stored in separate tables.	No pre-defined schema.
7.	Tight consistency.	Eventual consistency rather than ACID property.
8.	**Examples :** MySQL Oracle MS SQL PostgreSQL SQLite DB2	**Examples :** MongoDB Big Table Neo4j Couch DB Cassandra HBase

4.4 Business Drivers of NoSQL

MU - May 17

Q. What are the business drivers for NoSQL? **(May 17, 4 Marks)**

1. The growth of big data

- Big Data is one of the main driving factor of NoSQL for business.
- The huge array of data collection acts as driving force for data growth.

2. Continuous availability of data

- The competition age demands less downtime for better company reputation.
- Hardware failures are possible but NoSQL database environments are built with a distributed architecture so there are no single points of failure.
- If one or more database servers goes down, the other nodes in the system are able to continue with operations without any loss of data.
- So, NoSQL database environments are able to provide continuous availability.

3. **Location independence**
   - It is ability to read and write to a database regardless of where that I/O operation is done.
   - The master/slave architectures and database sharding can sometimes meet the need for location independent read operations.
4. **Modern transactional capabilities**

   The transactions concept is changing and ACID transactions are no longer a requirement in database systems.
5. **Flexible data models**
   - NoSQL has more flexible data model as compared to others.
   - A NoSQL data model is schema-less data model not like RDBMS.
6. **Better architecture**
   - The NoSQL has more business oriented architecture for a particular application.
   - So, Organizations adopt a NoSQL platform that allows them to keep their very high volume data.
7. **Analytics and business intelligence**
   - A key driver of implementing a NoSQL database environment is the ability to mining data to derive insights that offers a competitive advantage.
   - Extracting meaningful business information from very high volumes of data is a very difficult task for relational database systems.
   - Modern NoSQL database systems deliver integrated data analytics and better understanding of complex data sets which facilitate flexible decision-making.

4.5 NoSQL Database Types

MU - May 16, Dec. 16, May 17, May 18, May 19

Q. What are the different data architecture patterns in NOSQL? Explain Graph Store and Column Family Store patterns with relevant examples. **(May 16, May 19, 10 Marks)**

Q. Explain different NoSQL data architecture patterns. **(Dec. 16, 10 Marks)**

Q. Discuss any two architectural patterns of NoSQL. **(May 17, 4 Marks)**

Q. Explain in detail any one NOSQL architecture pattern. Identify two applications that can use this pattern. **(May 18, 5 Marks)**

Different Architectural Patterns in NoSQL

- Key-Value databases examples : Riak,Redis, Memcached, BerkeleyDB, upscaledb, Amazon DynamoDB.
- Document databases examples : MongoDB, CouchDB, Terrastore, OrientDB , RavenDB
- Column family stores examples : Cassendra, HBase, HyperTable.
- Graph Databases examples : Neo4j, InfiniteGraph, FlockDB.

1. Key-value store databases

- This is very simple NoSQL database.
- It is specially designed for storing data as a schema free data.
- Such data is stored in a form of data along with indexed key.

Examples

- Cassandra
- Azure Table Storage (ATS)
- DyanmoDB

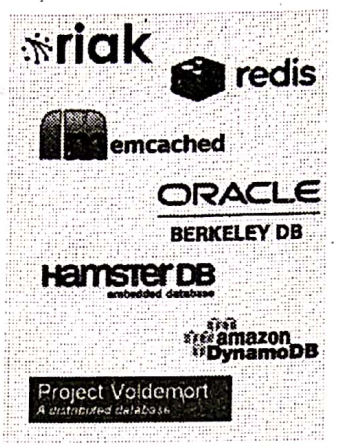


Fig. 4.5.1

Use Cases

This type is generally used when you need quick performance for basic Create-Read-Update-Delete operations and data is not connected.

Example

- Storing and retrieving session information for a Web pages.
- Storing user profiles and preferences
- Storing shopping cart data for ecommerce

Limitations

- It may not work well for complex queries attempting to connect multiple relations of data.
- If data contains lot of many-to-many relationships, a Key-Value store is likely to show poor performance.

2. Column store database

- Instead of storing data in relational tuples (table rows), it is stored in cells grouped in columns.
- It offers very high performance and a highly scalable architecture.

Fig. 4.5.2

Examples

(i) HBase (ii) Big Table (iii) Hyper Table

Use Cases

- Some common examples of Column-Family database include event logging and blogs like document databases, but the data would be stored in a different fashion.
- In logging, every application can write its own set of columns and have each row key formatted in such a way to promote easy lookup based on application and timestamp.
- Counters can be a unique use case. It is possible to design application that needs an easy way to count or increment as events occurs.

3. Document database

- Document databases works on concept of key-value stores where "documents" contains a lot of complex data.
- Every document contains a unique key, used to retrieve the document.
- Key is used for storing, retrieving and managing document-oriented information also known as semi-structured data.

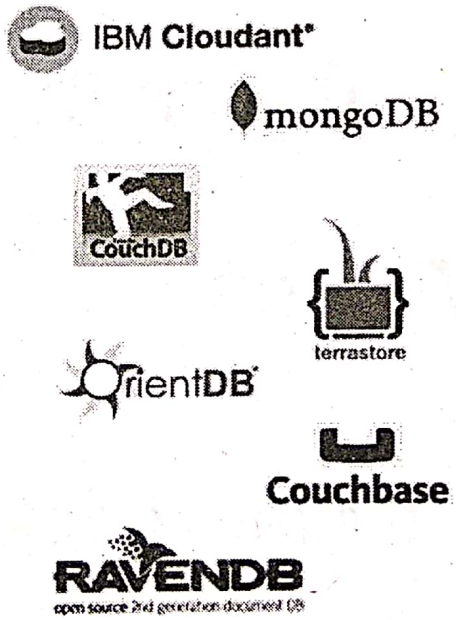


Fig. 4.5.3

Examples

(i) MongoDB (ii) Couch DB

Use Cases

- The example of such system would be event logging system for an application or online blogging.
- In online blogging user acts like a document; each post a document; and each comment, like, or action would be a document.

- All documents would contain information about the type of data, username, post content, or timestamp of document creation.

Limitations

- It's challenging for document store to handle a transaction that on multiple documents.
- Document databases may not be good if data is required in aggregation.

4. Graph database

Data is stored as a graph and their relationships are stored as a link between them whereas entity acts like a node.

Examples

(i) Neo4j (ii) Polyglot

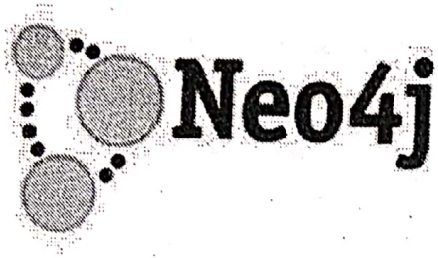


Fig. 4.5.4

Use Cases

- The very important and popular application would be social networking sites can benefit by quickly locating friends, friends of friends, likes, and so on.
- The Google Maps can help you to use graphs to easily model their data for finding close locations or building shortest routes for directions.
- Many recommendation systems makes effective use of this model.

Limitations

- Graph Databases may not be offering better choice over other NoSQL variations.
- If application needs to scale horizontally this may introduces poor performance.
- Not very efficient when it needs to update all nodes with a given parameter.

5. Comparison of NoSQL variations

Database model	Performance	Scalability	Flexibility
Key value store database	High	High	High
Column store database	High	High	Moderate
Document store database	High	Variable (High)	High
Graph database	Variable	Variable	High

4.6 Benefits of NoSQL

1. Big data analytics

- Big data is one of main feature promotes growth and popularity of NoSQL.
- NoSQL has good provision to handle such big data.

2. Better data availability

- NoSQL database works with distributed environments.
- NoSQL database environments should provide good availability across multiple data servers.
- NoSQL databases supply high performance.

3. Location independence

NoSQL data base can read and write database regardless of location of database operation.

4.7 Introduction to Big Data Management

- We all are surrounded by huge data. People upload/download videos, audios, images from variety of devices. Sending text messages, multimedia messages, updating their Facebook, WhatsApp, Twitter status, comments, online shopping, online advertising etc. generates huge data.
- As a result machines have to generate and keep huge data too. Due to this exponential growth of data the analysis of that data becomes challenging and difficult.
- The term 'Big Data' means huge volume, high velocity and a variety of data. This big data is increasing tremendously day by day.
- Traditional data management systems and existing tools are facing difficulties to process such a Big Data.
- R is one of the main computing tools used in statistical education and research. It is also widely used for data analysis and numerical computing in other fields of scientific research.

4.8 Big Data

- We all are surrounded by huge data. People upload/download videos, audios, images from variety of devices.
- Sending text messages, multimedia messages, updating their Facebook, WhatsApp, Twitter status, comments, online shopping, online advertising etc.
- Generates huge data. As a result machines have to generate and keep huge data too. Due to this exponential growth of data the analysis of that data becomes challenging and difficult.
- The term 'Big Data' means huge volume, high velocity and a variety of data. This big data is increasing tremendously day by day. Traditional data management systems and existing tools are facing difficulties to process such a Big Data.
- Big data is the most important technologies in modern world. It is really critical to store and manage it. Big is a collection of large datasets that cannot be processed using traditional computing techniques.
- Big Data includes huge volume, high velocity and extensible variety of data. The data in it may be structured data, Semi Structured data or unstructured data. Big data also involves various tools, techniques and frameworks.

Four Important V of Big Data

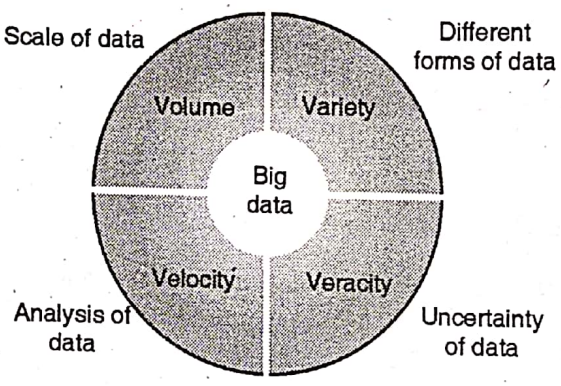


Fig. 4.8.1 : Four V of big data

1. **Volume**

   Huge amount of data is generated during big data applications.

2. **Velocity**

   For time critical applications the faster processing is very important. E.g. share marketing, video streaming

3. **Variety**

   The data may be Structured, Semi Structured or Unstructured.

4. **Veracity**

   Data is not certain. Data captured can vary greatly. So accuracy of analysis depends on the veracity of the source data.

4.8.1 Tools Used for Big Data

1. **Map Reduce**

   Hadoop, Hive, Pig, Cascading, Cascalog, mrjob, Caffeine, S4, MapR, Acunu, Flume, Kafka, Azkaban, Oozie, Greenplum

2. **Storage**

   S3, Hadoop Distributed File System

3. **Servers**

   EC2, Google App Engine, Elastic, Beanstalk, Heroku

4. **NoSQL**

   Zookeeper, MongoDB, Cassandra, Redis, Big Table, Hbase, Hyper table, Voldemort, Riak, Couch DB

5. **Processing**

   R, Yahoo! Pipes, Mechanical Turk, Solr/Lucene, ElasticSearch, Datameer, BigSheets, Tinkerpop

4.8.2 Understanding Types of Big Data Problems

1. **Acquiring data**

   High volume of data and transactions are the basic requirements of big data. Infrastructure should support the same. Flexible data structures should be used for the same. The amount of time required for this should be as less as possible.

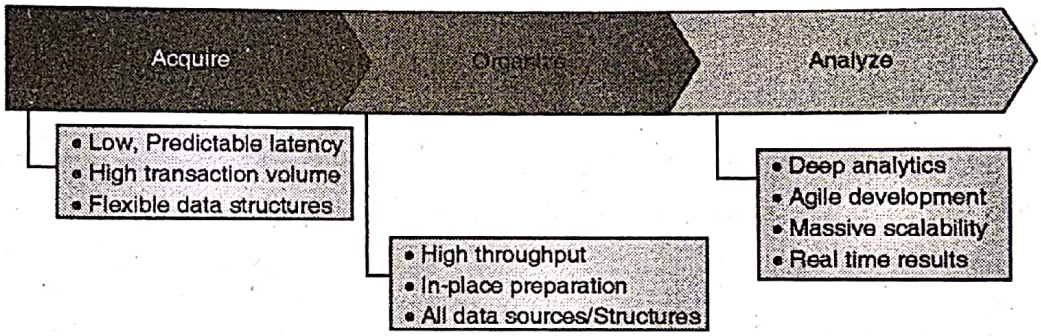


Fig. 4.8.2

2. **Organizing data**

   As the data may be structured, semi structured or unstructured it should be organized in a fast and efficient way.

3. **Analysing data**

   Data analysis should be faster and efficient. It should support distributed computing.

4.9 Four Ways of NoSQL to Operate Big Data Problems

1. **Key-value store databases**

   - This is very simple NoSQL database.
   - It is specially designed for storing data as a schema free data.
   - Such data is stored in a form of data along with indexed key.

Key : 1	ID : 123	First Name : Ganesh	
Key : 2	Email : abc@gmail.com	Location : Mumbai	Age : 37
Key : 3	Facebook ID : wea	Password : xxx	Name : Max

Fig. 4.9.1 : Example of unstructured data for user records

Working

- The schema-less format of a key value database is required for data storage needs.
- The key can be auto-generated while the value can be String.
- The key value uses a hash table in which there exists a unique key and a pointer to a particular item of data.
- A logical group of keys called as bucket, There can be identical keys in different buckets.
- It will improve performance because of the cache mechanisms that accompany the mappings.
- To read any value you need to know both the key and the bucket because the real key is a hash (Bucket+ Key).

Read Write values

- **Get(key) :** It will returns the value associated with key.
- **Put(key, value) :** It will associates the value with the key.
- **Multi-get(key1, key2, .., keyN) :** It will returns the list of values associated with the list of keys.
- **Delete(key) :** It will delete entry for the key from the data store.

2. Column store database

- Instead of storing data in relational tuples (table rows), it is stored in cells grouped in columns.
- It offers very high performance and a highly scalable architecture.

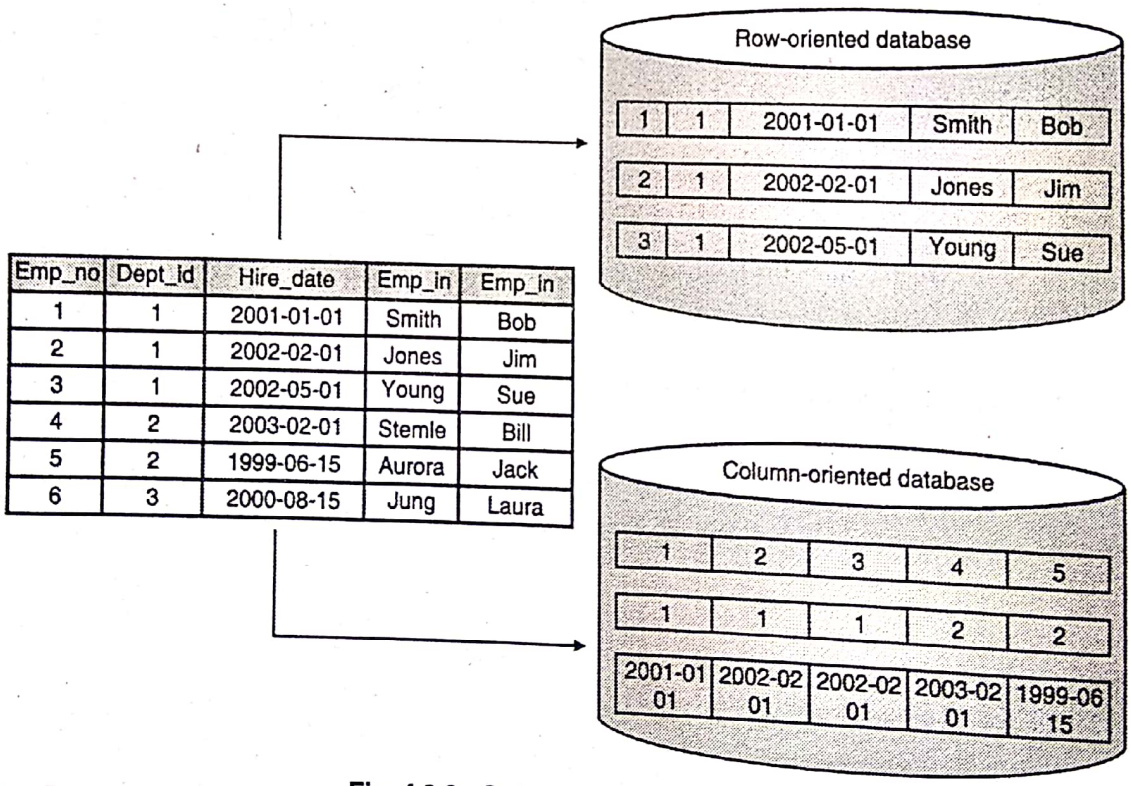


Fig. 4.9.2 : Column store database

Working

- In column-oriented NoSQL database, data is stored in cells grouped in columns of data rather than as rows of data.
- Columns are logically grouped into column families which are a virtually unlimited number of columns that can be

created at runtime.

- Read and write is done using columns.
- It offers fast search/ access and data aggregation.

Data Model

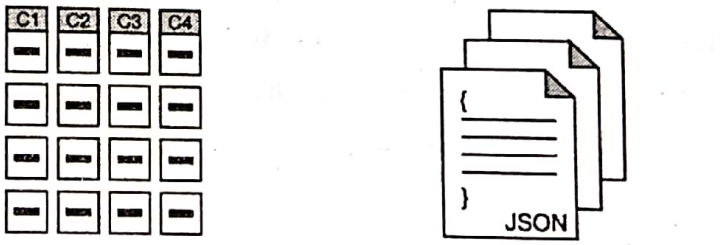
- **ColumnFamily :** Single structure that can group Columns and SuperColumns
- **Key :** The permanent name of the record having different numbers of columns.
- **Keyspace :** This defines name of the application.
- **Column :** It has an ordered list of elements with a name and a value defined.

Examples

(i) HBase (ii) Big Table (iii) Hyper Table

3. Document database

- Document databases works on concept of key-value stores where "documents" contains a lot of complex data.
- Every document contains a unique key, used to retrieve the document.
- Key is used for storing, retrieving and managing document-oriented information also known as semi-structured data.



Relational data model

Highly-structured table organization with rigidly-defined data formats and record structure.

Document data model

Connection of complex documents with arbitrary, nested data formats and varying "record" format.

Fig. 4.9.3 : Document database

Working

- This type of data is a collection of key value pairs is compressed as a document store quite similar to a key-value store, but the only difference is that the values stored is known as "documents" has some defined structure and encoding.
- The above example shows data values collected in Colum family.
- JSON and XML are common encoding as above.
- It is schemaless data makes easy for JSON to handle data.

Examples

(i) Mongo DB

(ii) Couch DB

4. Graph database

Data is stored as a graph and their relationships are stored as a link between them whereas entity acts like a node.

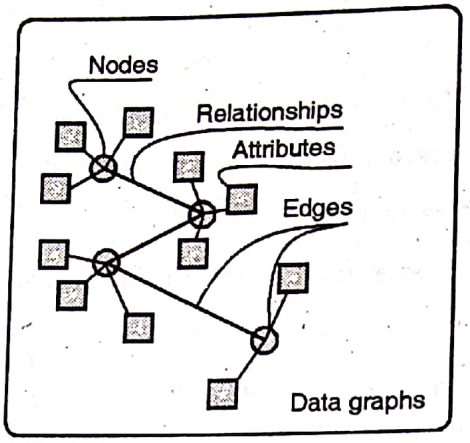


Fig. 4.9.4 : Graph database

Working

- In a Graph NoSQL Database, a flexible graphical representation is used with edges, nodes and properties which provide index-free adjacency.
- Data can be easily transformed from one model to the other using a Graph Base NoSQL database.
- These databases use edges and nodes to represent and store data.
- These nodes are organised by some relationships, which is represented by edges between the nodes.
- Both the nodes and the relationships have properties.

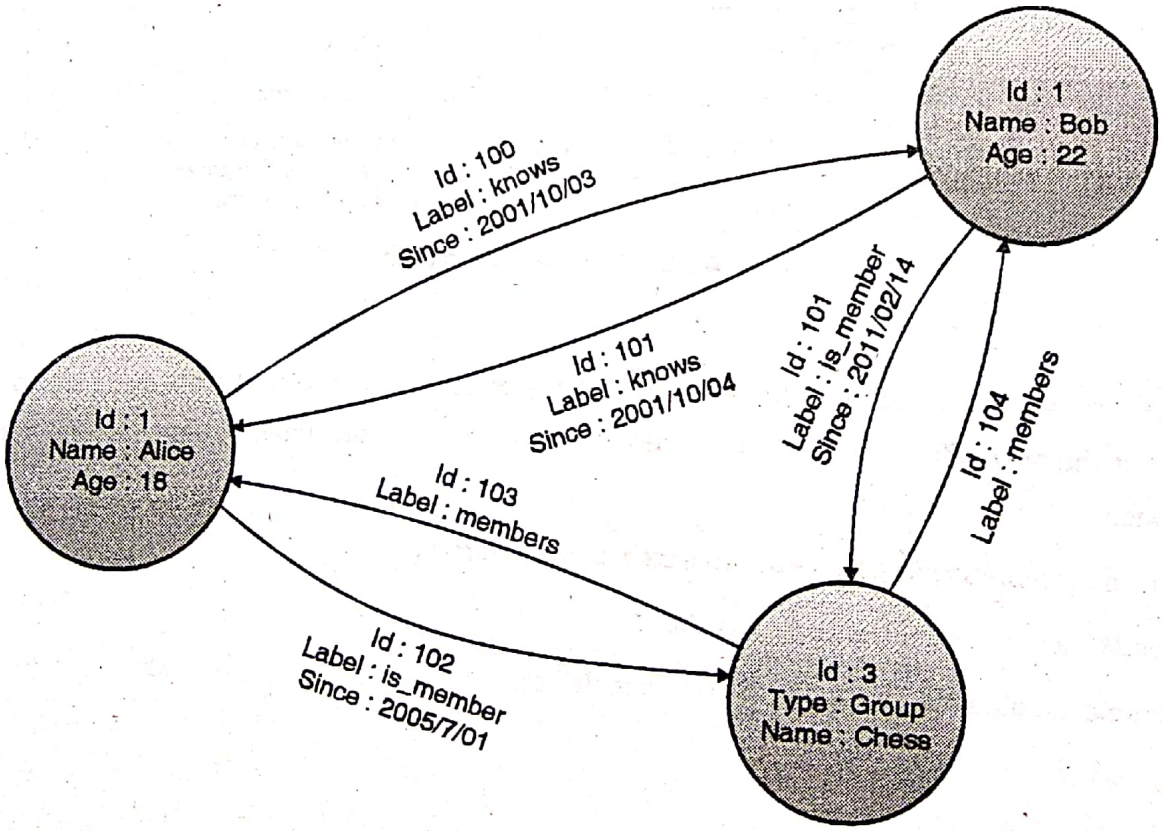


Fig. 4.9.5

Examples

(i) Neo4j (ii) Polyglot

4.10 Analyzing Big Data with a Shared-Nothing Architecture

Parallelism in databases represents one of the most successful instances of parallel computing system.

Types :

1. Shared Memory System (UNIX FS)
2. Shared Disk System (ORACLE RAC)
3. Shared Nothing System (HDFS)
4. Hierarchical System

4.10.1 Shared Memory System

(a) Architecture details

- Multiple CPUs are attached to a common global shared memory via interconnection network or communication bus.
- Shared memory architectures usually have large memory caches at each processor, so that referencing of the shared memory is avoided whenever possible.
- Moreover, caches need to be coherent. That means if a processor performs a write to a memory location, the data in that memory location should be either updated at or removed cached data.

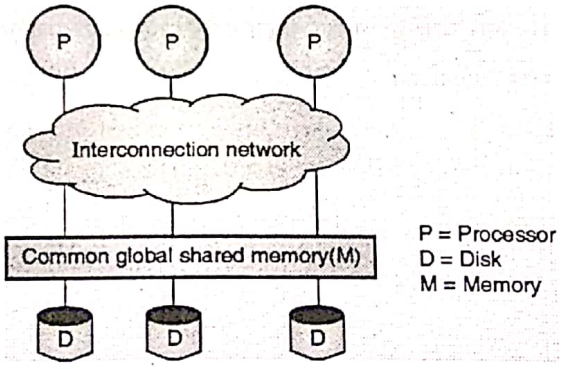


Fig. 4.10.1 : Shared memory system architecture

(b) Advantages

- Efficient communication between processors.
- Data can be accessed by any processor without being moved from one place to other.
- A processor can send messages to other processors much faster using memory writes.

(c) Disadvantages

- Bandwidth problem
- Not scalable beyond 32 or 64 processors, since the bus or interconnection network will get into a bottleneck.
- More number of processors can increase waiting time of processors.

4.10.2 Shared Disk System

(a) Architecture details

- Multiple processors can access all disk directly via inter communication network. But, every processor has local memory.
- Shared disk has two advantages over shared memory.
- Each processor has its own memory; the memory bus is not a bottleneck.
- System offers a simple way to provide a degree of **fault tolerance**.
- The systems built around this architecture are called **clusters**.

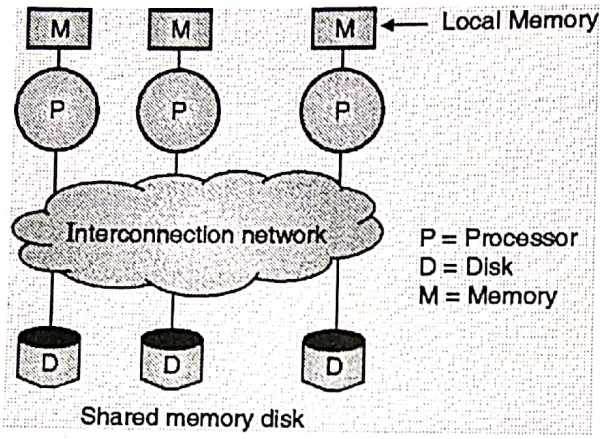


Fig. 4.10.2 : Shared memory disk architecture

(b) Advantages

- Each CPU or processor has its own local memory, so the memory bus will not face bottleneck.
- High degree of fault tolerance is achieved.
- **Fault tolerance** : If a processor (or its memory) fails, the other processor can take over its tasks, since the database is present on disks and are accessible to all processors.
- If one processor fails, other processors can take over its tasks, since database is on shared disk that can be accessible from all processors.

(c) Disadvantages

- Some Memory load is added to each processor.
- **Limited Scalability :** Not scalable beyond certain point. The shared-disk architecture faces this problem because large amounts of data are shipped through the interconnection network. So now the interconnection to the disk subsystem is a bottleneck.
- The basic problem with the shared-memory and shared-disk architecture is interference. As more CPUs are added, existing CPUs are slowed down because of the increased contention for memory accesses and network bandwidth.

(d) Applications

Digital Equipment Corporation : DEC cluster running Relational Databases were one of the early commercial user of shared disk database architecture. Now this is owned by Oracle.

Note : This observation has motivated the development of the shared nothing architecture, which is now widely considered to be the best architecture for large parallel database systems.

4.10.3 Shared Nothing Disk System

(a) Architecture details

- Each processor has its own local memory and local disk.
- A processor at one node may communicate with another processor using high speed communication network.
- Any terminal can act as a Node which functions as Server for data that is stored on local disk.
- Moreover, the interconnection networks for shared nothing systems are usually designed to be scalable, so that we can increase transmission capacity as more nodes are added to the network.

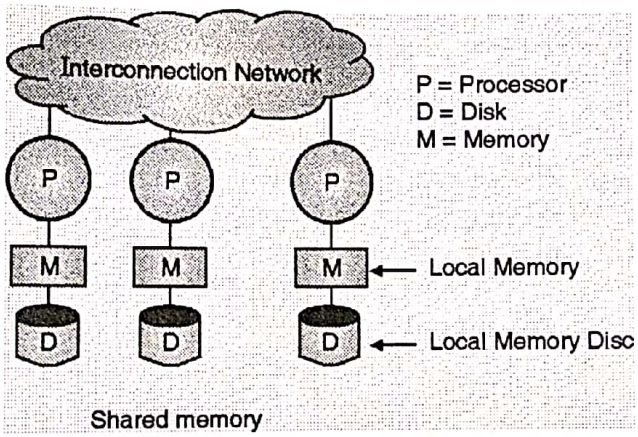


Fig. 4.10.3 : Shared nothing architecture

(b) Advantages

- In this type of architecture no need to go through all I/O, Only a single interconnection network queries which access non local disk can pass through the network.
- We can achieve High degree of parallelism. i.e. Number of CPU and disk can be connected as desired.
- Shared nothing architecture systems are more scalable and can easily support a large number of processors.

(c) Disadvantages

- Cost of communication and of non local disk access is higher than other two architectures since sending data involves software interaction at both ends.
- Requires rigid data partitioning.

(d) Applications

- The teradata database machine uses shared nothing database architecture.
- Grace and the Gamma research prototypes.

4.10.4 Hierarchical System

Architecture details

- The hierarchical architecture comes with combined characteristics of shared memory, shared disk and shred nothing architectures.
- At the top level, the system consists of nodes connected by an interconnection network and they do not share disks or memory with one another.
- This architecture is attempts to reduce the complexity of programming such systems yields to distributed virtual-memory architectures, where logically there is a single shared memory, the memory mapping hardware coupled with system software, allows each processor to view the disjoint memories as a single virtual memory.
- The hierarchical architecture is also referred to as **nonuniform memory architecture (NUMA).**

Hierarchical architecture

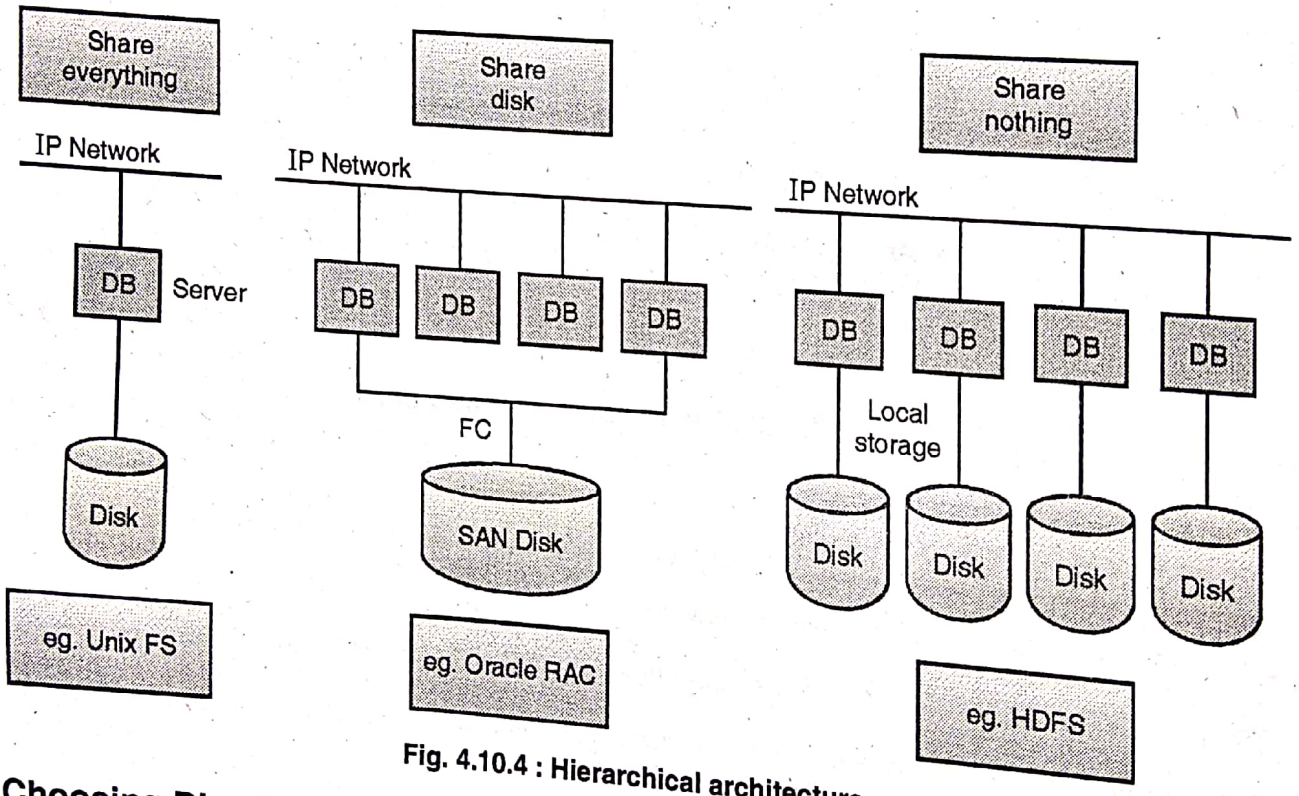


Fig. 4.10.4 : Hierarchical architecture

4.11 Choosing Distribution Models : Master-Slave versus Peer-to-Peer

Master Slave vs. Peer to Peer Model

- The data bases like HBase follow the master slave model In Master slave model one node out of all participating nodes will acts as the master i.e. the master node will responsible for governing all the decisions related to the job assignment, task assignment, data storage, data retrieval, data manipulation etc. The Master will delegate the jobs and tasks to the other working nodes in a given cluster. The main advantage is the working nodes cannot manipulate the things without permission of the master node.
- Fig. 4.11.1 shows Master slave model.

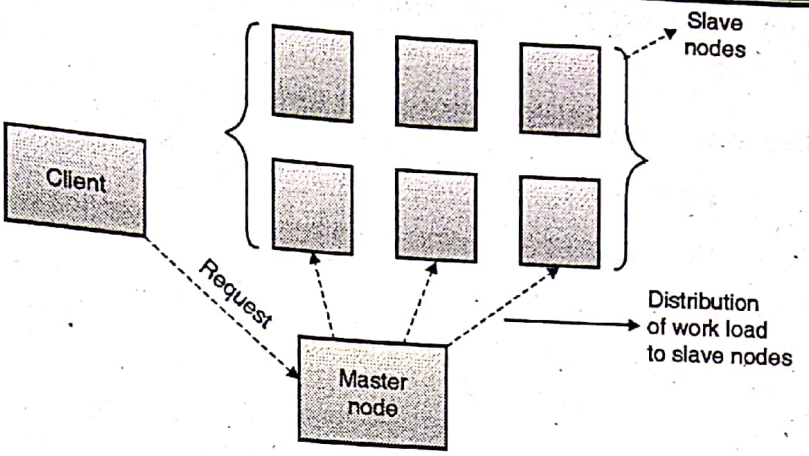


Fig. 4.11.1 : Master slave model

- The data bases like cassendra will follow the Peer- to-peer model. In peer to peer model all participating nodes has same rights. It means every node can perform all operations requested by the database user.
- These operations include Create, Read, Update, Delete as well as operations related to configuration of such databases.
- As all nodes has the same priority, so the requests from data base users will be received by any of the nodes irrespective of work load distribution.
- The peer to peer model will exhibit a data replication mechanism where data packets will be replicated for certain number of times to avoid the data loss if some part of the data base will get crashed.
- Fig. 4.11.2 shows Peer to Peer model.

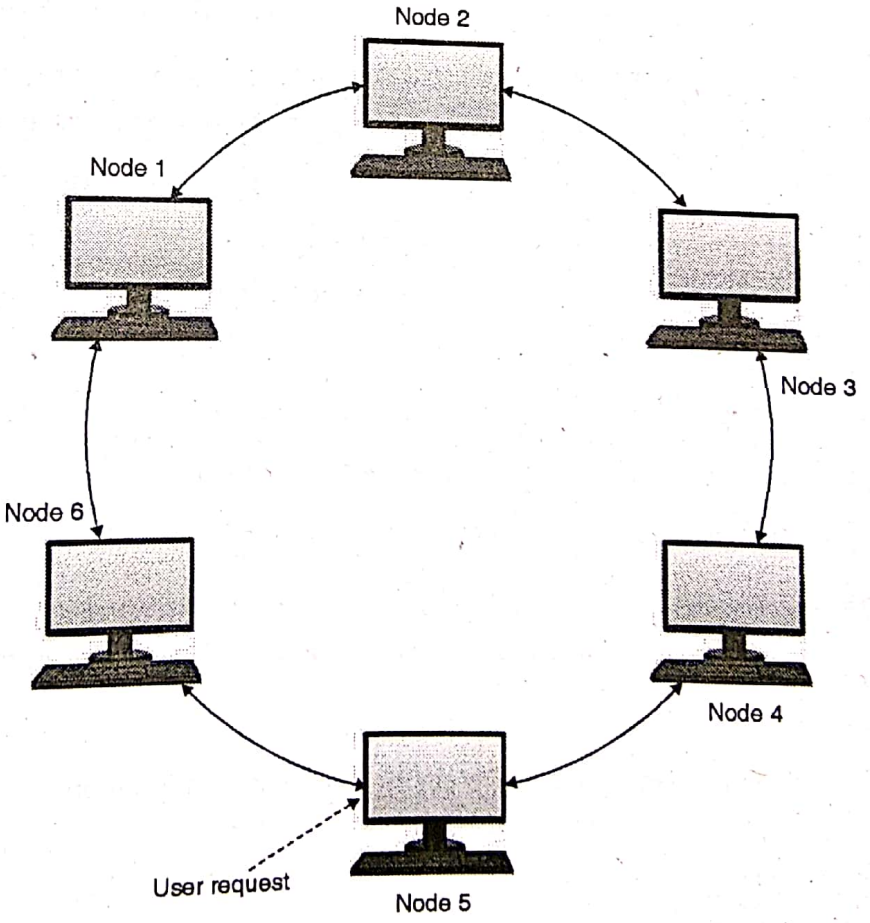


Fig. 4.11.2 : Peer to peer model

4.11.1 Big Data NoSQL Solutions

There are many NoSQL solutions to Bigdata problems as listed below,

1. Cassendra
2. DynamoDB
3. Neo4J
4. MongoDB
5. Hbase
6. Big Tables

4.11.1(A) Cassendra

1. Introduction

- Cassendra is a distributed storage system mainly designed for managing large amount of structured across multiple servers.
- Cassandra run with hundreds of servers and manages them.
- Cassandra provides a simple data model that supports dynamic control over data.
- Cassandra system was designed to run with economic hardware to handle large amount of data.
- For Example, Facebook runs on thousands of servers located in many data centres using Cassandra.
- The cassendra fall in the category of NOSQL data bases. In NOSQL the data bases doesn't have the schema i.e. a schema less data bases.
- The Main aim behind the development of cassesndra is to deal with the complexities involved in the Big Data which makes use of no. of nodes as well as it tries to avoid the critical part known as Single Point Failure.
- Cassendra has Data Replication as its major features, in which the data packet can duplicated no. of times. This duplication is generally limited to a pre defined threshold value. For Replication a Gossip Protocol is used.
- Nodes or work stations participating in the database will have equal priority, access privileges yet they all are connected to each other with dedicated communication links.
- When user of a data base requests to have read / write operation then any available node can receive this request and process it.

Basic building blocks of cassesndra

- **A node :** The Node represents the place where the data is actually stored.
- **The data center :** The Data center is the set of nodes which process same category of data.
- **The cluster :** A group of data centers is known as Cluster.
- **The commit log :** The Commit log is a maintenance tool which recovers the data if some failure occurs.

2. Types of Databases

- Besides Cassandra system there are few NOSQL databases very popular among users,
- Apache's HBase
- MongoDB

3. Features of Cassandra

(i) Scalability

Cassandra is highly scalable system; it also allow to add more hardware as per data requirement.

(ii) 24X7 Availability

- Cassandra less chances of failure.
- It is always available for business applications.

(iii) Good performance

Cassandra system can be scaled up linearly, which means, it increases your outputs as you increase the number of nodes in the cluster.

(iv) Good Data storage

- Cassandra can store almost all possible data formats including: structured, semi-structured, and unstructured.
- It can manage all change to your data structures as per your need.

(v) Data distribution

Cassandra system provides flexible data distribution, as per need of data replication across multiple data centers.

(vi) Transaction support

- Cassandra supports properties like Atomicity, Consistency, Isolation, and Durability which is properties of transaction i.e.ACID.
- Faster write Operations
- Cassandra system was mainly designed for using low cost.
- It is designed faster write operations and can store huge amount of data, with very good read efficiency.

4. History of Cassandra

- Cassandra was developed at Facebook for searching facebook inbox.
- Cassandra system was open-sourced by Facebook in July 2008.
- Cassandra was accepted into Apache Incubator in 2009.

4.11.1(B) Dynamo DB

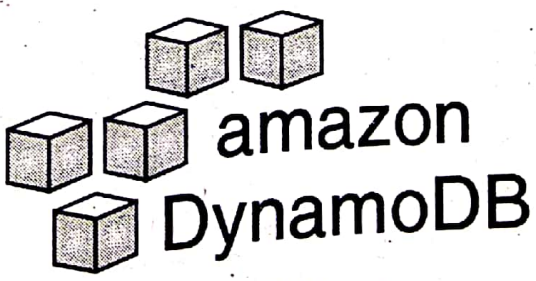


Fig. 4.11.3

1. Data Model

- Amazon's NOSQL database called DynamoDB in form of table is a collection of various items and each item is a collection of attributes.

- In a relational database, a table has a fixed schema of tables with primary key and list of its columns with data types.
- All tuples are of same schema.
- DynamoDB Data model contains,
  - Table
  - Items
  - Attributes
- DynamoDB requires only a primary key and does not require to define all of the attribute names and data types in advance.
- Each attribute of DynamoDB in an item is a name-value pair.

Primary Key

In order to create table we must specify primary key column name which identifies item set in table uniquely.

Partition Key

- A simple primary key has only one attribute known as the partition key.
- DynamoDB uses the partition key as input to hash function for internal use.

Partition Key and Sort Key

- A composite primary key is made of two attributes.
- The first attribute is the partition key, and the second attribute is the sort key.
- All items with the similar partition key are stored together, sorting is done in sorted order using sort key value.

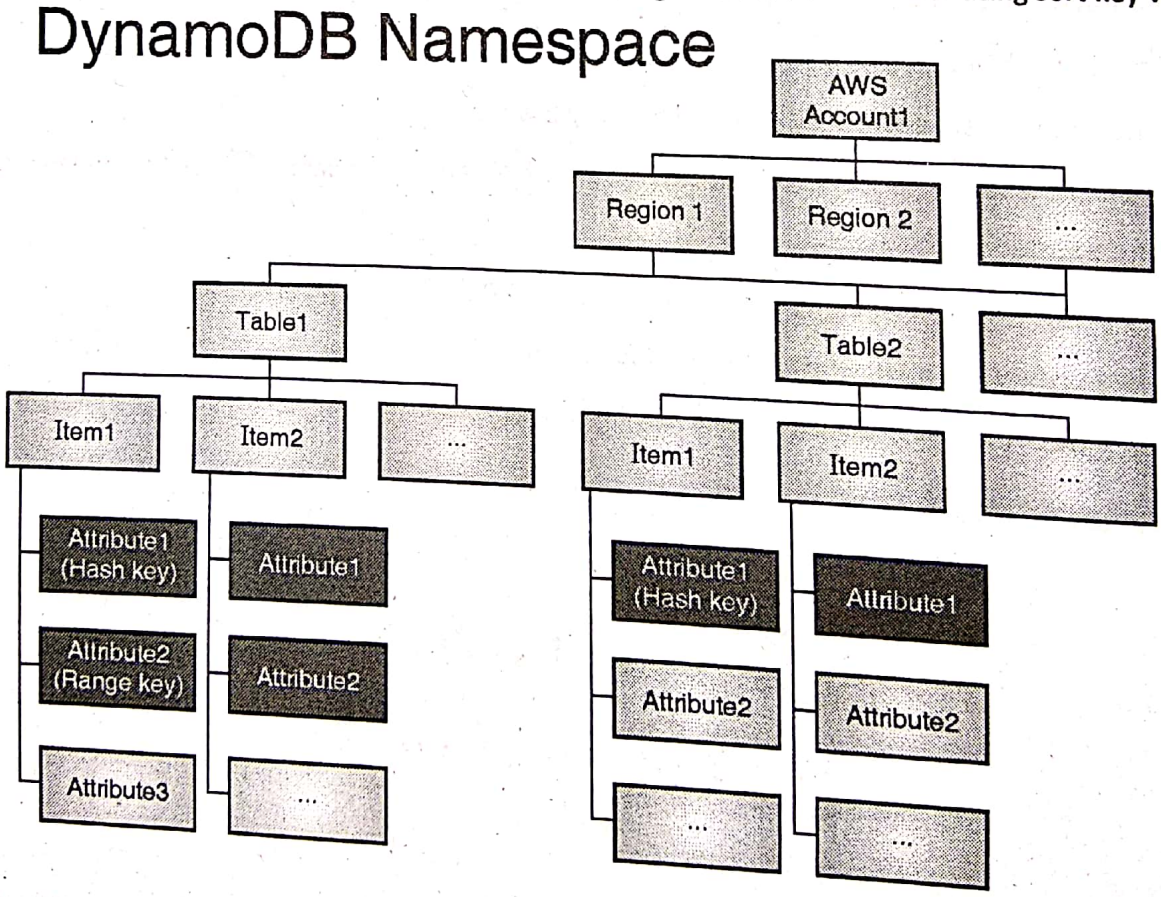


Fig. 4.11.4

2. Example

Consider a class of students with ID column as primary key,

StudentClass{ID,....}

```
{
  Id = 1011
  SubjectName = "Java"
  ISBN = "111-1111111111"
  Authors = [ "Author 1", "Author 2" ]
  Price = 150
  PageCount = 500
  Publication = TechMax
}

{
  Id = 1015
  SubjectName = "JDB"
  ISBN = "111-1111111111"
  Authors = ["Author 3"]
  Price = 1543
  PageCount = 5000
  Publication = TechMax
}
```

3. Data Type

- Amazon DynamoDB supports various data types
- **Scalar types :** Number, String, Binary, Boolean, and Null.
- **Document types :** List and Map.
- **Set types :** String Set, Number Set, and Binary Set.

4. CRUD Operations

(a) Table Operations

(i) CreateTable

- It is used to create new table on your account.
- To check status of table we can use DescribeTable command.

```
{
  "AttributeDefinitions": [
    {
```

```
        "AttributeName": "string",
        "AttributeType": "string"
    }
],


"GlobalSecondaryIndexes": [
    {
        "IndexName": "string",
        "KeySchema": [
            {
                "AttributeName": "string",
                "KeyType": "string"
            }
        ],
        "Projection": {
            "NonKeyAttributes": [
                "string"
            ],
            "ProjectionType": "string"
        },
        "ProvisionedThroughput": {
            "ReadCapacityUnits": number,
            "WriteCapacityUnits": number
        }
    }
],
    "TableName": "string"
}
```

(ii) Readtables

The readtable operation used to read the tables which are created by create table command with the help o diagnostic tools such as list tables. Returns list of table names associated with the current account.

(iii) DeleteTable

The *DeleteTable* operation deletes a table and items associated with it.

(iv) UpdateTable

Modifies throughput, global secondary indexes for a given table.

(b) Item Operation

(i) BatchGetItem

The BatchGetItem operation returns the attributes of one or more items from tables.

(ii) BatchWriteItem

The BatchWriteItem operation deletes multiple items in tables.

(iii) DeleteItem

It will delete a single item with help of its primary key.

(iv) GetItem

GetItem operation returns a set of attributes for item with the given primary key.

(v) PutItem

It will create a new item, or replaces an old item with a new item.

(vi) UpdateItem

Edits an existing item's attributes to add new item to the table if it does not already exist.

(c) Others

(i) *Query*

A *Query* operation uses the primary key of a table to directly access items from that table.

(ii) *Scan*

The *Scan* operation returns one or more items and item attributes.

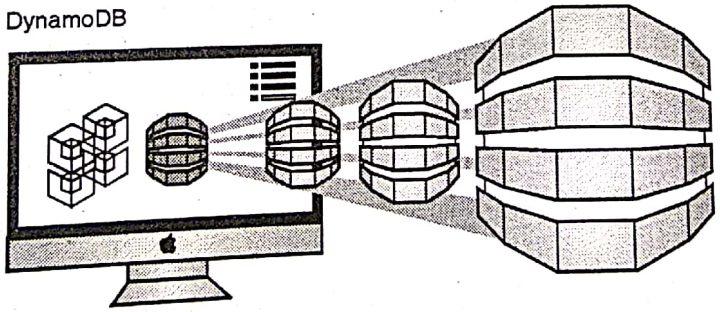


Fig. 4.11.5

5. Data Access

- Amazon DynamoDB is a web service uses HTTP and HTTPS as a transport layer services.
- JavaScript Object Notation (JSON) can be used as a message serialization format.
- Application code makes requests to the DynamoDB web service API.
- It is possible to use AWS Software Development Kits (SDKs).
- DynamoDB API libraries take care of request authentication, serialization, and connection management.

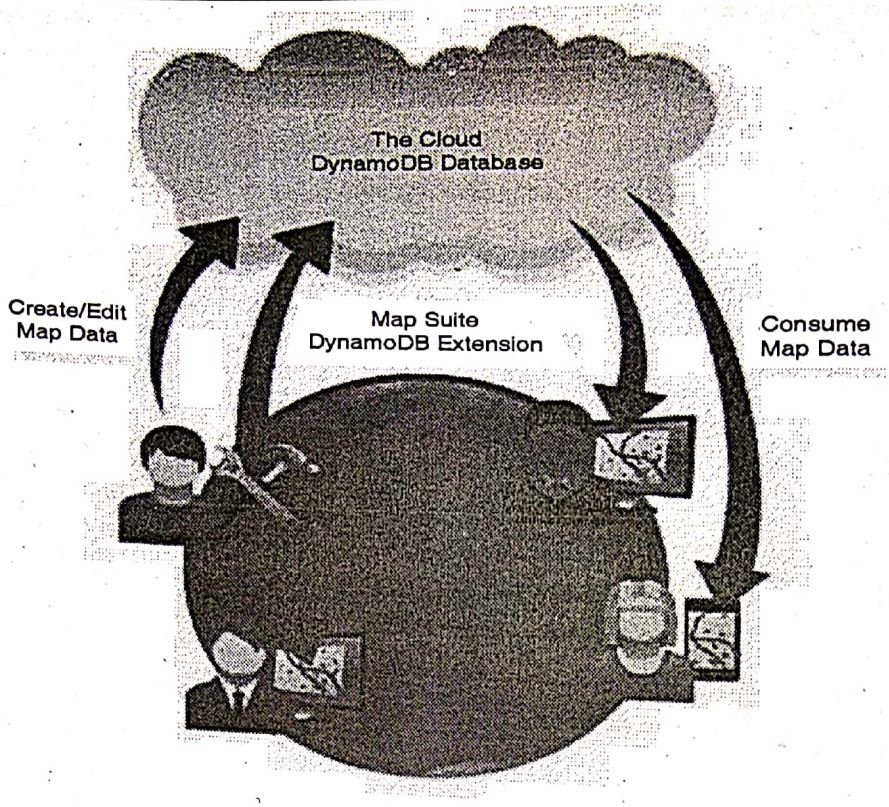


Fig. 4.11.6

6. Data Indexing

- In order to have efficient access to data in a table, Amazon DynamoDB creates and maintains indexes for the primary key attributes.
- Indexing allows applications to quickly retrieve data by specifying primary key.
- Applications may get good benefits from having one or more secondary keys available, for efficient data access with attributes other than the primary key.

Secondary Indexes

- In order to create table we must specify primary key column name which identifies item set in table uniquely.
- DynamoDB can create a table with a composite primary key it also supports one or more secondary indexes on table.
- A secondary index allows you to query data in the table using an alternate key like primary key.

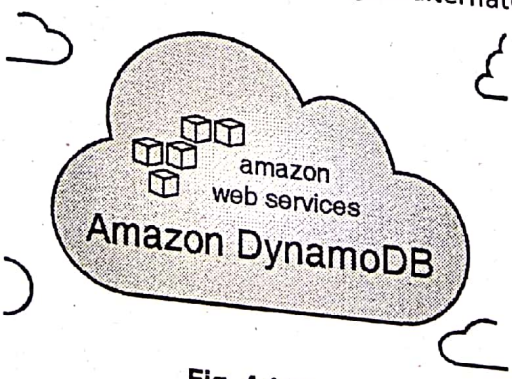


Fig. 4.11.7

Types of secondary indexes :

(i) Global Secondary Index

An index with a partition key and sort key, different from index on the table.

(ii) Local Secondary Index

An index that has the same partition and different sort key.

Review Question

Q. 1 Write a short note on Big Data.

Q. 2 Compare SQL and NoSQL databases.

Q. 3 Write a short note on Casssendra.

Q. 4 Write a short note on DynamoDB.

□□□

CHAPTER 5

Module - 4

Mining Data Streams

Syllabus

The Stream Data Model: A Data-Stream-Management System, Examples of Stream Sources, Stream Queries, Issues in Stream Processing, Sampling Data techniques in a Stream, Filtering Streams: Bloom Filter with Analysis, Counting Distinct Elements in a Stream, Count-Distinct Problem, Flajolet-Martin Algorithm, Combining Estimates, Space Requirements, Counting Frequent Items in a Stream, Sampling Methods for Streams, Frequent Itemsets in Decaying Windows, Counting Ones in a Window: The Cost of Exact Counts, The Datar-Gionis-Indyk-Motwani Algorithm, Query Answering in the DGIM Algorithm, Decaying Windows.

5.1 The Stream Data Model

Q. What is data stream? Explain data stream operations in the context of Big Data.

- A data stream is a flow of data which arrives at uncertain intervals of time.
- The data available from a stream is fundamentally different from the data stored in a conventional database in the sense that the data available in a database is complete data and can be processed at any time we want it to be processed. On the other hand, stream data is not completely available at any one point of time. Instead only some data is available with which we have to carry on our desired processing.
- Another problem with stream data is the storage of the entire data received from the stream. The rate of arrival is so rapid as well as the volume of data is so huge that it is not practically possible to save the entire data in a database. Rather it is necessary to summarize the stream data by taking appropriate samples.
- Generally, we have to perform the following fundamental operations to handle input data stream :
  (i) Gather information from the input data stream,
  (ii) Clear or filter the data,
  (iii) Apply standard modeling techniques,
  (iv) Deploy the generated solutions.

Diagrammatically we may represent the above four operations as shown in Fig. 5.1.1.

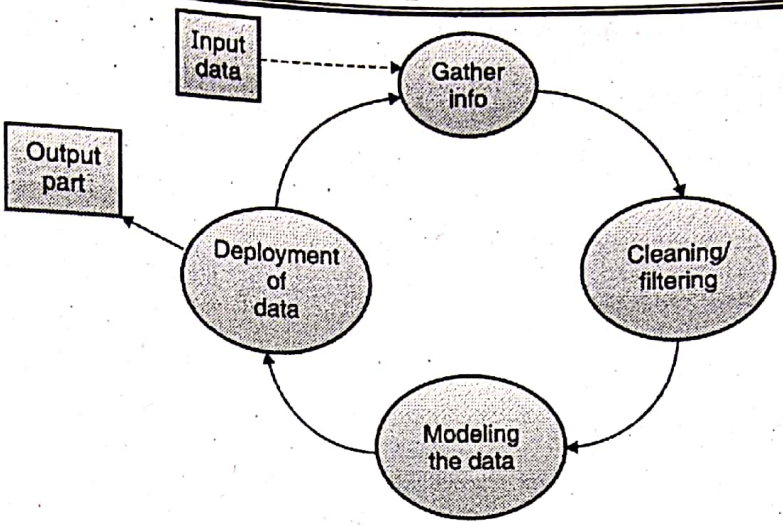


Fig. 5.1.1 : Stream data operations

- When data stream or data flow arrives at the compute node, then either it has to be stored in a database or processed immediately. Otherwise the data is lost forever.
- The major issues in this storage/processing task are :
  (i) The incoming rate of arrival of data is tremendous, and
  (ii) Different streams containing different categories of data items (images, text, alphanumeric, encoded, video data etc.) may be coming to the same compute node.
- It is practically not feasible to store the entire stream data as the storage will never be sufficient to accommodate it.
- Thus, it becomes necessary to summarize the incoming stream data in some way or other. Every algorithm dealing with stream data uses summarization. For this purpose, samples of the stream are taken and the stream is filtered to remove those portions which are not required for the computations.
- The next step is the estimation of all the different elements in the stream using just the samples obtained in the previous step. This drastically reduces the storage requirements.
- The summarization algorithms may be classified into the following two categories :
  (i) The first category involves algorithms described above which make use of good sampling techniques, efficient filtering (removal of irrelevant data items), and estimation of the stream elements.
  (ii) In the second category, algorithms are based on the concept of window. A window is nothing but the most recent n elements of the stream. Queries are then executed on the window considering it as a relation in a database. But the size and/or the number of windows may be large, so it is necessary to even summarize the windows.

5.1.1 A Data-Stream-Management System

MU - Dec. 16, May 17

Q. Explain abstract architecture of Data Stream Management system (DSMS). **(Dec. 16, 10 Marks)**

Q. What is a Data stream Management System? Explain with Block Diagram. **(May 17, 10 Marks)**

Q. Explain the data-stream-management system with neat diagram.

- A data-stream-management system is very similar to a conventional relational database-management system. Fig. 5.1.2 represents the organization of a data-stream-management system.

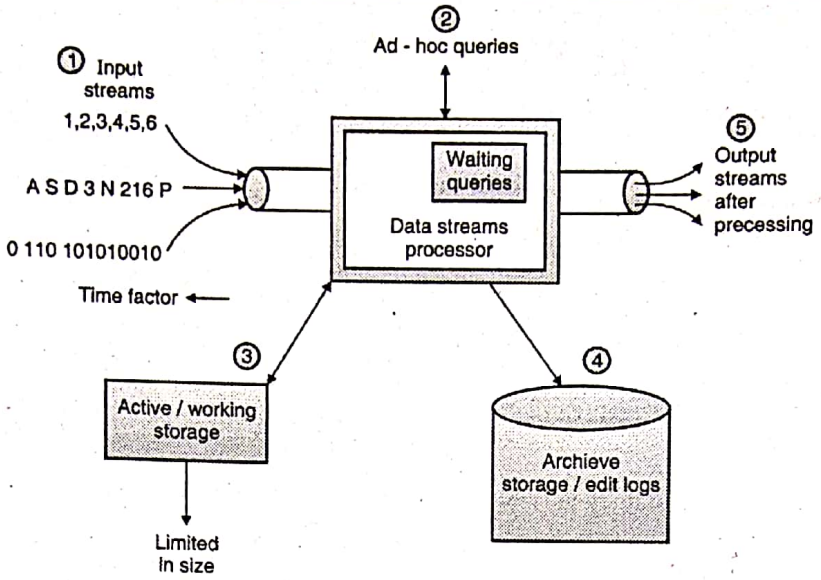


Fig. 5.1.2 : A data-stream-management system

– Let us understand the purpose of each of the components of the system :

1. **Input streams :** The input streams have the following characteristics:
   - There can be one or more number of input streams entering the system.
   - The streams can have different data types.
   - The rate of data flow of each stream may be different.
   - Within a stream the time interval between the arrival of data items may differ. For example, suppose the second data item arrives after 2 ms from the arrival of the first data item, then it is not necessary that the third data item will also arrive after 2 ms from the arrival of the second data item. It may arrive earlier or even later.
2. **Stream processor :** All types of processing such as sampling, cleaning, filtering, and querying on the input stream data are done here. Two types of queries are supported which are standing queries and ad-hoc queries. We shall discuss both the query types in details in the upcoming section.
3. **Working storage :** A limited memory such as a disk or main memory is used as the working storage for storing parts or summaries of streams so that queries can be executed. If faster processing is needed, main memory is used otherwise secondary storage disk is used. As the working storage is limited in size, it is not possible to store all the data received from all the streams.
4. **Archival storage :** The archival store is a large storage area in which the streams may be archived but execution of queries directly on the archival store is not supported. Also, the fetching of data from this store takes a lot of time as compared to the fetching of data from the working store.
5. **Output streams :** The output consists of the fully processed streams and the results of the execution of queries on the streams.

– The difference between a conventional database-management system and a data-stream-management system is that in case of the database-management system all of the data is available on the disk and the system can control the rate of data reads. On the other hand, in case of the data-stream-management system the rate of arrival of data is not in the control of the system and the system has to take care of the possibilities of data getting lost and take the necessary precautionary measures.

5.1.2 Examples of Stream Sources

Q. List and explain various data stream sources.

Following are some of the primary sources of stream data :

1. Sensor data

- Sensors are the devices which are responsible for reading and sending the measurements of various kinds of physical parameters such as temperature, wind speed, pressure, moisture content, humidity, pollution level, surface height, amount of light, etc.
- Consider a network of millions of sensors on the ocean surface to provide an early warning system for Tsunami by studying the behaviour of the ocean. This can save a lot of lives. But the amount of data received from all the sensors combined is simply huge.

2. Data streams related to Images

- High resolution image streams are relayed to the earth stations by the satellites which can amount to many terabytes of image data per day. Many such high-resolution images are released for the public from time to time by NASA as well as ISRO.
- Lower resolution image streams are produced by the CCTV cameras placed in and around important places and shopping centres. Now a days most of the public places and some of the private properties are under the surveillance of CCTV cameras 24×7.

3. Internet services and web services traffic

- The networking components such as switches and routers on the Internet receive streams of IP packets and route them to the proper destination. These devices are becoming smarter day by day by helping in avoiding congestion and detecting denial-of-service-attack, etc.
- Websites receive many different types of streams. Twitter receives millions of tweets, Google receives tens of millions of search queries, Facebook receives billions of likes and comments, etc. These streams can be studied to gather useful information such as the spread of diseases or the occurrence of some sudden event such as catastrophe.

5.1.3 Stream Queries

MU - May 17, Dec. 18

Q. With respect to data stream querying, give example of Ad-hoc queries. (May 17, Dec. 18, 2 Marks)

Q. What are stream queries? Explain different categories of stream queries.

In the previous section we have mentioned that the data-stream-management system supports two different types of queries:

1. Standing queries
2. Ad-hoc queries

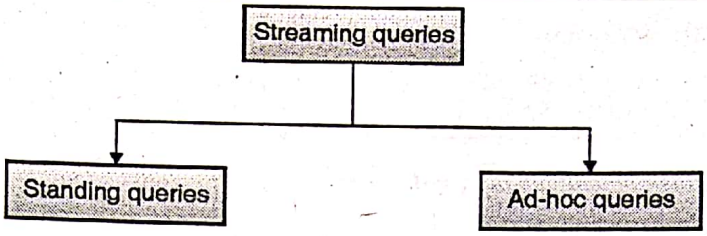


Fig. 5.1.3 : Query types

1. Standing queries

- A standing query is a query which is stored in a designated place inside the stream processor. The standing queries are executed whenever the conditions for that particular query becomes true.
- For example, if we take the case of a temperature sensor then we might have the following standing queries in the stream processor :
  - Whenever the temperature exceeds 50 degrees centigrade, output an alert.
  - On arrival of a new temperature reading, produce the average of all the readings arrived so far starting from the beginning.
  - Output the maximum temperature ever recorded by the sensor, after every new reading arrival.

2. Ad-hoc queries

- An ad-hoc query is not predefined and is issued on the go at the current state of the streams. The nature of the ad-hoc queries cannot be determined in advance.
- To allow a wide range of arbitrary ad-hoc queries it is necessary to store a sliding window of all the streams in the working storage. A sliding window is nothing but the most recent elements in the stream. The number of elements to be accommodated in the sliding window has to be determined beforehand. As and when new elements arrive, the oldest ones will be removed from the window and hence the name sliding window.
- Instead of determining the size of the sliding window in advance we may also take another approach based on the unit of time. In this approach the sliding window may be designed to accommodate say all the stream data for an hour or a day or a month, etc.
- For example, a social networking website like Facebook may want to know the number of unique active users over the past one month.

5.1.4 Issues in Stream Processing

Q. Discuss different issues in data stream processing.

- The issues in stream processing mainly arise because of the following two basic reasons :
  1. The rapid rate of arrival of stream data, and
  2. The huge size of the data when all of the input streams are considered.
- Because of the rapid arrival of stream data, the processing speed also must match the arrival speed of the data. To achieve this, the entire stream processing algorithm must reside in main memory and there should be minimal secondary memory accesses. This is because the secondary memory accesses are many times slower than the main memory accesses.

- In the case of a slow stream it may be possible to process the data in that stream using a small portion of main memory. But if the number of such slow streams becomes large then again, the problem of shortage of main memory arises.
- Thus, one way to solve the issues related to stream processing is by having a large amount of main memory. But in real life compute nodes the amount of main memory available is usually limited which makes it unfeasible. So, we have to resort to some other way for solving the issues. One such way is to use the technique of sampling.

5.2 Sampling Data Techniques in a Stream

Q. What is sampling of data in a stream? How do we obtain representative sample ?

- A sample is a subset of a stream which adequately represents the entire stream. The answers to the queries on the sample can be considered as though they are the answers to the queries on the whole stream.
- Let us illustrate the concept of sampling with the example of a stream of search engine queries. A search engine may be interested in learning the behaviour of its users to provide more personalized search results or for showing relevant advertisements. Each search query can be considered as a tuple having the following three components: (user, query, time)

Obtaining a representative sample

- The first step is to decide what will constitute the sample for the problem in hand. For instance, in the search query stream we have the following two options :
  - Take the sample of queries of each user, or
  - Take the sample of users and include all the queries of the selected users.
- Option number 2 as a sample is statistically a better representation of the search query stream for answering the queries related to the behaviour of the search engine users.
- The next step is to decide what will be the size of the sample compared to the overall stream size. Here we will assume a sample size of 1/10th of the stream elements.
- When a new user comes into the system, a random integer between 0 and 9 is generated. If the number is 0, then the user's search query is added to the sample. If the number is greater than 0, then the user's search query is not added to the sample. A list of such users is maintained which shows which user's search query is to be included in the sample.
- When a new search query comes in the stream from any existing user, then the list of users is consulted to ascertain whether the search query from the user is to be included in the sample or not.
- For this method to work efficiently the list of users has to be kept in the main memory because otherwise disk access will be necessary to fetch the list which is a time-consuming task.
- But as the list of users grows it will become a problem to accommodate it into the main memory. One solution to this problem is to use a hash function as the random number generator. The hash function will map a user to a number between 0 to 9. If a user hashes to 0 then the search query of the user is added to the sample and otherwise it is not added.
- In general we can create a sample size of any rational fraction a/b by using a hash function which maps a user to a number between 0 and b-1 and the user's query is added to the sample if the hash value is less than a.

The general sampling problem

Q. Explain general sampling problem? What is effect on stream if we vary the sample size?

- Each tuple in the stream consists of n components out of which a subset of components called key on which the selection for the sample is based.
- For instance, in the search query example the key consists of only one component user out of the three components user, query and time. But it is not always necessary to consider only user as the key, we could even make query as the key or even the pair (user, query) as the key.
- So, to produce a sample of size a/b, we have to use a hash function which can map the key of a tuple to a value between 0 and b-1. The tuple will be added to the sample if the hash value is less than a.
- In case of a multi-component keys such as (user, query), the hash function has to do the extra work of combining the values of all the components in order to produce a single hash value.

Variation in sample size

- With the passage of time the sample size will keep on growing as new users as well as new search queries from existing users will be added to the sample. If we have a limit on the number of tuples that can be stored in the sample, we can follow the technique described below.
- We select such a hash function h which maps the keys to a very large number of values

$$0, 1, 2, \ldots, B-1.$$

We also initialize a threshold,

$$t = B-1$$

- A tuple from the stream is allowed into the sample if and only if

$$h(\text{key}) \leq t$$

- When the number of stored tuples reach the maximum limit, we can remove all those tuples whose key hash to the value $B-1$ and modify the threshold as,

$$t = t-1$$

- We can even reduce t by more than 1 and remove all those tuples associated with the hash values higher than t.

5.3 Filtering Streams

Q. Explain the filtering process of data streams with suitable example.

- Filtering also known as selection is another important process which is applied to the streams. Through filtering only those tuples are accepted which satisfy the given criteria and other tuples are rejected.
- There are two categories of selection criteria :
  - The criteria based on any property of the tuple which can be calculated from the tuple itself. This is a case of easy filtering. For example, the selection criteria could be whether the tuple contains a query having the term "Big Data" in it.

- The criteria which involve the looking up of set membership. In this case the filtering becomes harder if the set is huge and cannot be stored in the main memory.

- Bloom filtering is a filtering technique which is used for eliminating or rejecting most of the tuples which do not satisfy the criteria.

Example of filtering

- Let us consider the example of spam email filtering. Let S be the set of safe and trusted email addresses. Assume that the size of S is one billion email addresses and the stream consist of the pairs (email address, email message).
- The set S cannot be accommodated in main memory because on average an email address is of minimum 20 bytes in size. So, to test for set membership in S, it becomes necessary to perform disk accesses. But as discussed earlier a disk access is many times slower than main memory access.
- We can do the spam filtering using only the main memory and no disk accesses with the help of Bloom filtering. In this technique the main memory is used as a bit array.
- Say we have 1 GB of main memory available for the filtering task. Since each byte consists of 8 bits, so 1 GB memory contains 8 billion bits. This means we have a bit array of 8 billion locations.
- We now need a hash function h which will map each email address in S to one of the 8 billion locations in the array. All those locations which get mapped from S are set to 1 and the other locations are set to 0.
- As there are 1 billion email addresses in S and 8 billion bits in main memory, so approximately 1/8th of the total available bits will be set to 1. The exact count of bits that are set to 1 will be less than 1/8th because more than one email address may hash to the same bit location.
- When a new stream element arrives, we simply need to hash it and check the contents of the bit location to which it is hashed. If the bit is 1, then the email is from a safe and trusted sender. On the other hand, if the bit is 0, then the email is a spam.

5.3.1 Bloom Filter with Analysis

MU - Dec. 17, Dec 18

Q. Explain the concept of a Bloom Filter using an example. (Dec. 17, 10 Marks)

Q. How Bloom filter is useful for big data analytics. Explain with one example. (Dec. 18, 10 Marks)

Q. What is Bloom filter? Explain Bloom filtering process with neat diagram.

The components of a Bloom filter are as follows :

- An array of n bits initialized to 0's.
- A set H of k hash functions each of which maps a key to one of the n bit locations :

  $h_1, h_2, ..., h_k$
- A set S consisting of m number of keys.
- Fig. 5.3.1 illustrates the block diagram of Bloom filter and its process.

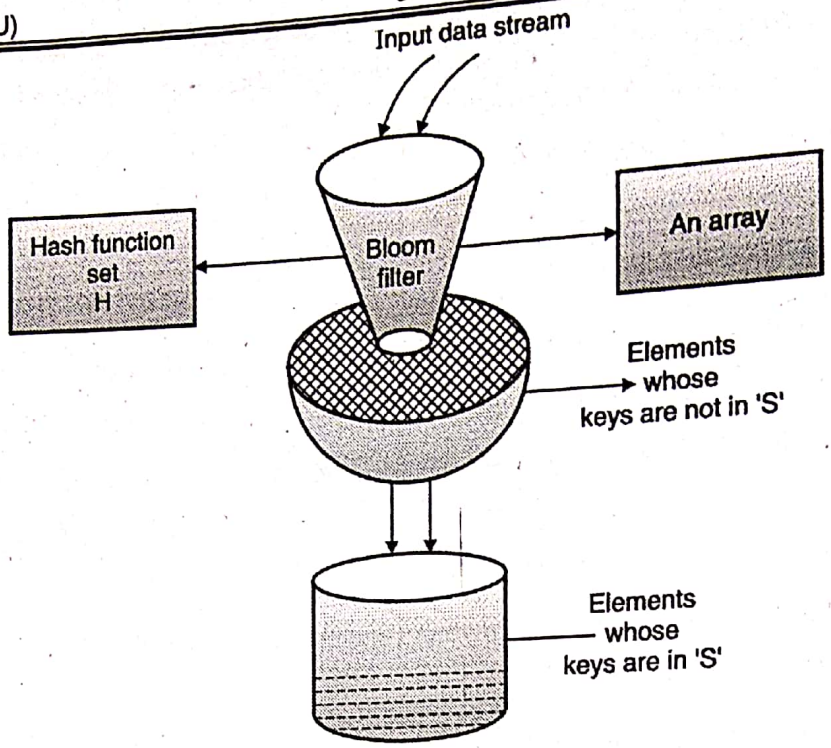


Fig. 5.3.1: Bloom filter

- Bloom filter accepts all those tuples whose keys are members of the set S and rejects all other tuples.
- Initially all the bit locations in the array are set to 0. Each key K is taken from S and one by one all of the k hash functions in H are applied on this key K. All bit locations produced by $h_i(K)$ are set to 1.

  $h_1(K), h_2(K), ..., h_k(K)$
- On arrival of a test tuple with key K, all the k different hash functions from H are once again applied on the test key K. If all the bit locations thus produced are 1, the tuple with the key K is accepted. If even a single bit location out of these turns out to be 0, then the tuple is rejected.

Analysis of bloom filtering

- The Bloom filter suffers from the problem of false positives. This means even if a key is not a member of S, there is a chance that it might get accepted by the Bloom filter.
- To find the probability of a false positive we need to use the model of a dart game in which darts are thrown at targets. Assume that there are x darts and y targets and the probability of any dart hitting any target is equally likely, then according to the theory of probability :
  - The probability of a given dart not hitting a given target is $\frac{x-1}{x}$
  - The probability of no dart hitting a given target is $\left(\frac{x-1}{x}\right)^y = \left(1-\frac{1}{x}\right)^{x\left(\frac{y}{x}\right)}$
- As we know, $(1-\epsilon)^{\frac{1}{\epsilon}} = \frac{1}{\epsilon}$ when ϵ is small. Thus, we can say that the probability of no dart hitting a given target is $\left(1-\frac{1}{x}\right)^{x\left(\frac{y}{x}\right)} = \left(\frac{1}{4}\right)^{y/x} = e^{-y/x}$

5.4 Counting Distinct Elements in a Stream

MU - Dec. 17

Q. Explain any one algorithm to count number of distinct elements in data stream. (Dec. 17, 10 Marks)

- Apart from sampling and filtering, one more simple and important processing on the stream data is that of counting the number of elements which are distinct in the given data stream.
- This also requires a large amount of main memory. But with the help of hashing and randomized algorithm we can achieve a good approximation of the count using only a small amount of memory.

5.4.1 Count - Distinct Problem

Q. Explain count distinct problem with suitable example.

- Let us illustrate the problem of count-distinct with the example of a website like Amazon or Facebook or Google which wants to know the number of monthly active unique users. These numbers are useful in preparing their server and other infrastructure for efficient load handling as well as for generation of advertising revenues.
- Here we assume that the elements of the stream belong to a universal set. The universal set for sites which have the login facility will be the set of all logins (usernames and passwords). Examples of such sites are Facebook, Twitter, Amazon, etc. But for sites like Google which do not require a login for searching, the universal set will be the set of all IP addresses.
- One way of performing the distinct user count will be to keep the entire list of elements (users) that have appeared on the stream in the main memory arranged in some efficient structure such as hash table or search tree.
- Whenever a new user visits the website, it is checked whether she is already there in the list or not. If she is not there in the list, she is added to the list. If she is already in the list, then no action is taken.
- This approach works fine till the number of users can easily fit the available main memory. If the number of users grows or there are a number of streams to be processed simultaneously, then it starts becoming a problem.
- This problem can be solved in a number of different ways :
  - By using a greater number of compute nodes but it increases the complexity of implementation.
  - By storing the list structure in secondary memory but it increases the time complexity by a huge factor.
- Thus, instead of trying to find the exact count of the distinct elements we can try to find an approximate count. This will ensure that the task is finished quickly using only a small portion of the main memory.

5.4.2 The Flajolet- Martin Algorithm

MU - Dec. 16, May 17

Q. Give problem in Flajolet-Martin (FM) algorithm to count distinct elements in a stream. (Dec. 16, 5 Marks)

Q. How to count distinct elements in a stream? Explain Flajolet-Ma1tin Algorithm. (May 17, 5 Marks)

Q. Explain Flajolet-Martin algorithm in detail.

- The Flajolet-Martin algorithm is used for estimating the number of unique elements in a stream in a single pass. The time complexity of this algorithm is O(n) and the space complexity is O(log m) where n is the number of elements in the stream and m is the number of unique elements.

- The major components of this algorithm are :
  - A collection of hash functions, and
  - A bit-string of length L such that $2^L > n$. A 64-bit string is sufficient for most cases.
- Each incoming element will be hashed using all the hash functions. Higher the number of distinct elements in the stream, higher will be the number of different hash values.
- On applying a hash function h on an element of the stream e, the hash value h(e) is produced. We convert h(e) into its equivalent binary bit-string. This bit string will end in some number of zeroes. For instance, the 5-bit string 11010 ends with 1 zero and 10001 ends with no zeroes.
- This count of zeroes is known as the tail length. If R denotes the maximum tail length of any element e encountered thus far in the stream, then the estimate for the number of unique elements in the stream is 2^R.
- Now to see that this estimate makes sense we have to use the following arguments using probability theory :
  - The probability of h(e) having a tail length of at least r is 2^{-r}.
  - The probability that none of the m distinct elements have tail length of at least r is $(1-2^{-r})^m$.
  - The above expression can also be written as $.((1-2^{-r})^{2r})^{m2-r}$
  - And finally, we can reduce it to e^{-m2-r} as $(1-2-r)^{2r} = e^{-1}$
- If $m >> 2^r$, the probability of finding a tail of length at least r approaches 1.
- If $m << 2^r$, the probability of finding a tail of length at least r approaches 0.
- So, we can conclude that the estimate of 2^R is neither going to be too low nor too high.
- Let us now understand the working of the algorithm with an example :

Stream: 5, 3, 9, 2, 7, 11

Hash function :

$$h(x) = 3x + 1 \bmod 32$$

$$h(5) = 3(5) + 1 \bmod 32 = 16 \bmod 32 = 16 = 10000$$

$$h(3) = 3(3) + 1 \bmod 32 = 10 \bmod 32 = 10 = 01010$$

$$h(9) = 3(9) + 1 \bmod 32 = 28 \bmod 32 = 28 = 11100$$

$$h(2) = 3(2) + 1 \bmod 32 = 7 \bmod 32 = 7 = 00111$$

$$h(7) = 3(7) + 1 \bmod 32 = 22 \bmod 32 = 22 = 10110$$

$$h(11) = 3(11) + 1 \bmod 32 = 34 \bmod 32 = 2 = 00010$$

Tail lengths: {4, 1, 2, 0, 1, 1}

$$R = \max(\text{Tail length}) = 4$$

$$\text{Estimate of } m = 2^R = 2^4 = 16.$$

- The above estimate is obtained from a single hash function. Similarly, the other hash functions will produce one estimate each. We will need a method of combining the estimates to arrive at the overall estimate.

5.4.3 Combining Estimates

Q. Explain the process of combining the Estimates.

- There are three approaches for combining the estimates from the different hash functions :
  - Average of the estimates, or
  - Median of the estimates, or
  - The combination of the above two.
- If we take the average of the estimates to arrive at the final estimate then it will be problematic in those cases where one or a few estimates are very large as compared to the rest of the others. Suppose the estimates from the various hash functions are 16, 25, 18, 32, 900, 23. The occurrence of 900 will take the average estimate to the higher side although most of the other estimates are not that high.
- The median is not affected by the problem described above. But a median will always be a power of 2. So, for example the estimate using a median will jump from $2^8 = 256$ to $2^9 = 512$, and there cannot be any estimate value in between. So, if the real value of m is say, 400, then neither 256 nor 512 is a good estimate.
- The solution to this is to use a combination of both the average and the median. The hash functions are divided into small groups. The estimates from the groups are averaged. Then the median of the averages is calculated which is the final estimate.
- Now even if a large value occurs in a group and makes its average large, the median of the averages will nullify its effect on the final estimate. The group size should be a small multiple of $\log_2 m$ so that any possible average value is obtained and this will ensure that we get a close estimate by using a sufficient number of hash functions.

5.4.4 Space Requirements

Q. Comment on space requirements in count distinct problem.

- We do not need to store the elements of the stream in main memory.
- The only data that needs to be stored in the main memory is the largest tail length computed so far by the hash function on the stream elements.
- So, there will be as many tail lengths as the number of hash functions and each tail length is nothing but an integer value.
- If there is only a single stream, millions of hash functions can be used on it. But a million hash functions are far more than what is necessary to arrive at a close estimate.
- Only when there are multiple streams to be processed simultaneously, we have to limit the number of hash functions per stream. Even in this case the time complexity of calculating the hash values is a bigger concern than the space constraint.

5.5 Counting Frequent Items in a Stream

Q. How frequent items in a stream are counted?

- There are two main differences between a stream and a file :
  - A stream has no end while every file ends at some point.

- The time of arrival of a stream element cannot be predicted in advance while the data in a file is already available.

- Moreover, the frequent items in a stream at some point of time may be different from the frequent items in the same stream at some other point of time.
- To continue our discussion, we need to understand the concept of an itemset. In the market-basket model of data we have two types of objects. One is items and the other is baskets. The set of items in a basket is called the itemset.
- In the next section we consider some of the sampling methods available for counting the frequent items in a stream. We will consider the stream elements as baskets of items.

5.5.1 Sampling Methods for Streams

- The simplest technique for estimating the frequent itemsets in a stream is to collect a few baskets and save them in a file. On this file we can run any frequent-itemsets algorithm. This algorithm will produce the estimate of the current frequent itemsets in the stream. The other stream elements which arrive during the execution of the algorithm can be stored in a separate file for processing later.
- After the completion of the first iteration we can run another iteration of the frequent-itemsets algorithm with :
  - A new file of baskets, or
  - The old file collected during the execution of the first iteration.
- We can go on repeating the above procedures for more iterations. This will result in a collection of frequent itemsets. If a particular itemset occurs in a fraction of the baskets that is lower than the threshold, it can be dropped from the collection.
- It should also be ensured that new frequent itemsets are added to the collection. For this to happen the following can be done :
  - A new segment of the baskets from the stream can be used as an input to the algorithm in some iteration.
  - Adding a few random itemsets in the collection and continuing the iterations.

5.5.2 Frequent Itemsets in Decaying Windows

- To use the concept of decaying windows for finding the frequent itemsets we need to keep the following points in mind :
  - The stream elements are not individual items, rather they are baskets of items. This means that many items appearing together is considered a single element here.
  - The target here is to find all of the frequent itemsets and not just limit ourselves to finding singleton itemsets. In other words, for each itemset received we need to initiate the count for not just that particular itemset but also all of its subsets. The problem here is that the number of subsets of even a not so big itemset may become unmanageable.
- To solve the first issue, we have to consider all the items appearing together as current items and do the following for each such item :
  - If the item has no current score, then its score is initialized to 1.
  - Otherwise, multiply the current score by the term $(1 - c)$ and add 1 to the product. Here c is a small constant such as 10^{-9}.

- To solve the second issue, only those itemsets are scored whose all immediate proper subsets are already being scored.

5.6 Counting Ones in a Window

One of the most important operations performed on a stream is the operation of counting of the number of occurrences of a particular element in a stream. If we consider a binary stream and we have a window of size N bits on this stream, we may want to find the number of 1's in the last k bits of the window where $k \leq N$. Our main focus here is the case in which we cannot fit the entire window in the main memory.

5.6.1 The Cost of Exact Counts

- Again, there are two major approaches to the counting problem :
  1. Exact count
  2. Approximate count
- For the exact count approach, we need to store the entire N-bit window in the main memory. Otherwise it will not be possible to compute the exact count of the desired elements. Let us try to understand it with the following arguments.
- Let us suppose instead of storing the N-bit window in main memory, we store an n-bit representation of the N-bit window where N> n.
- Now, number of possible N-bit window sequences = 2^N.

  and, number of possible n-bit window representations = 2^n.

  Since, N > n, it implies $2^N > 2^n$.
- Clearly we can see that the number of representations are not sufficient to represent all possible window sequences. Hence by the pigeon-hole principle there exists at least two different window sequences p and q which are represented by the same window representation in main memory.
- As $p \neq q$, it means they must differ in at least one bit position. But since both of them are having the same representation, the answer to the query of the number of 1's in the last k bits is going to be the same for both p and q, which is clearly wrong.
- Thus, from the above discussion we can conclude that it is totally necessary to store the entire N-bit window in memory for exact counts.

5.6.2 The DGIM Algorithm (Datar – Gionis – Indyk - Motwani)

MU – May 16, May 18, Dec. 18, May 19

Q. Using an example bit stream explain the working of the DGIM algorithm to count number of 1's (Ones) in a data stream. **(May 16, May 18, 5/10 Marks)**

Q. Explain DGIM algorithm for counting ones in stream with example. **(Dec. 18, May 19, 10 Marks)**

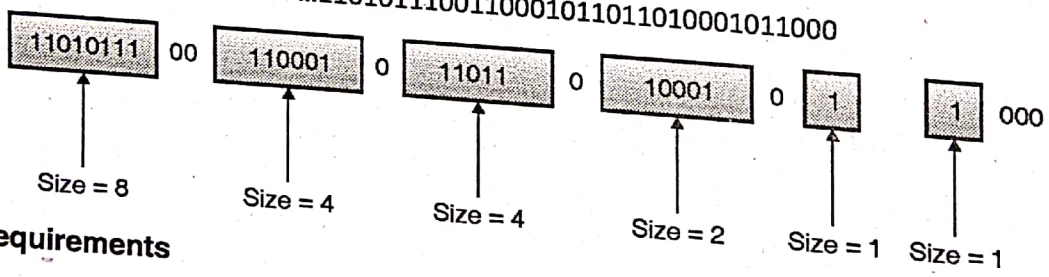
- In the simplest version of DGIM algorithm an N-bit window is represented using $O(\log^2 N)$ bits.
- The maximum error in the estimation of the number of 1's in the window in case of this algorithm is 50%.

- The two basic components of this algorithm are :
  - Timestamps, and
  - Buckets.
- Each bit arriving in the stream is assigned a timestamp. The timestamp is nothing but the sequence number of the bit in the order of arrival. So, the first bit is assigned timestamp 1, the second bit is assigned timestamp 2, the third bit is assigned timestamp 3 and each subsequent bit is assigned a timestamp one higher than the previous bit.
- The window is divided into buckets. Each bucket is described by :
  - The timestamp of the right most bit, and
  - The size of the bucket which is the number of 1's inside the bucket. The size must be a power of 2.

The following six conditions must be satisfied by the buckets :

1. The right end of every bucket must be occupied by a 1.
2. Every 1 should be inside some bucket.
3. A bit cannot be inside more than one bucket. In other words the buckets cannot overlap.
4. The number of buckets of a particular size can be either one or two. There will also be a limit on the maximum size of the bucket for a particular stream.
5. The size of a bucket is always a power of 2.
6. On moving from right to left the size of the buckets will increase.

Consider the following stream of bits : ...1101011100110001011011010001011000



Storage space requirements

The storage space required for the DGIM algorithm can be determined as follows :

1. A single bucket is represented using O(log N) bits.
2. Number of buckets is O (log N).
3. Total space required = O ($\log^2$ N).

5.6.3 Query Answering in the DGIM Algorithm

- If the query is to calculate the number of 1's in the last k bits of the window then :
  - Find the oldest timestamp bucket b which has at least one bit of the most recent k bits.
  - The estimate of the number of 1's = Half of the size of the oldest bucket + sizes of all the newer buckets.

- Let us apply the above mentioned steps to the buckets in the previous example. Suppose we want to estimate the number of 1's in the most recent 16 bits, then we observe that starting from the rightmost side the latest 16 bits fall inside both the size 1 buckets, one size 2 bucket and partially the size 4 bucket. This means in this case the oldest timestamp bucket is the size 4 bucket.
- Thus the estimate of the number of 1's in the latest 16 bits = (4/2) + 2 + 1 + 1 = 6. But the actual number of 1's is 7.

5.6.4 Decaying Windows

Q. What is Decaying Windows? Explain in detail.

- The problem with the sliding window of fixed size is that it does not take into account the older elements which are outside the window.
- A solution to this is the use of exponentially decaying windows which take into account all of the elements in the stream but assign different weightages to them. The recent elements are given more weightage compared to the older elements.
- This type of window is suitable for answering the queries on the most common recent elements. For example the most popular current movies, or the most popular items bought on Flipkart recently, or the top most current tweets, etc.
- Suppose the elements of a stream are :

  $e_1, e_2, e_3, \ldots, e_t$

  where e_1 is the oldest element and e_t is the most recent.

  Then the exponentially decaying window is the following sum :

  $$\sum_{i=0}^{t-1} e_{t-i} (1-c)^i$$

  where c is a small constant such as 10^{-9}.

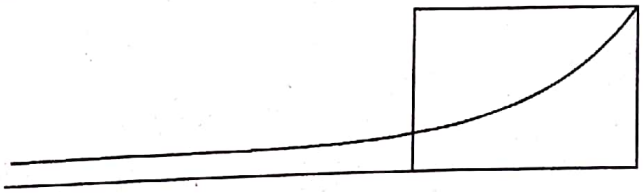


Fig. 5.6.1 : Fixed length window vs decaying window

- Fig. 5.6.1 shows the difference between a fixed length sliding window and an exponentially decaying window of equal weight. The rectangular box represents the fixed length window.
- When a new element e_{t+1} arrives in the stream, the following steps are taken :
  - Multiply the current sum by the term $(1-c)$, and
  - Add e_{t+1}.
- The major advantage of the decaying window is that we don't need to worry about the older elements going out of the window on arrival of new elements.

Review Questions

Q.1 What is data stream ? Explain data stream operations in the context of Big Data.

Q.2 What is stream Data Model ? Explain in detail.

Q.3 Explain the data-stream-management system with neat diagram.

Q. 4 List and explain various Data stream sources.

Q. 5 What are stream Queries ? Explain different Categories of stream Queries.

Q. 6 Discuss different issues in Data stream processing.

Q. 7 What is sampling of Data in a stream ? How do we obtain representative sample ?

Q. 8 Explain General Sampling problem. What is effect on stream if we vary the sample size?

Q. 9 Explain the filtering process of data streams with suitable example.

Q. 10 What is bloom filter ? Explain Bloom filtering process with neat diagram.

Q. 11 What is bloom filter ? Analyze the Bloom filter for performance.

Q. 12 Explain count distinct problem with suitable example.

Q. 13 Explain Flajolet-Martin algorithm in detail.

Q. 14 Explain the process of combining the Estimates. Also comment on space requirements.

Q .15 How frequent items in a stream are counted ?

Q. 16 What is cost of exact counts ?

Q. 17 Explain Datar-Gionis-Indyk-Motwani Algorithm in detail.

Q. 18 What is Decaying windows ? Explain in detail.

CHAPTER 6

Module - 5

Finding Similar Items

Syllabus

Distance Measures : Definition of a Distance Measure, Euclidean Distances, Jaccard Distance, Cosine Distance, Edit Distance, Hamming Distance.

6.1 Distance Measures

MU - May 16, Dec. 18

Q. Write a note on different distance measures that can be used to find similarity/ dissimilarity between data points in a big data set. **(May 16, 10 Marks)**

Q. What are distance measures? Brief any two distance measures. **(Dec. 18, 5 Marks)**

Q. What is a distance measure? Explain different criteria's regarding distance measures.

- A set of points is called a space. A space is necessary to define any distance measure. Let x and y be two points in the space, then a distance measure is defined as a function which takes the two points x and y as input, and produces the distance between the two points x and y as output. The distance function is denoted as :

  $d(x, y)$

- The output produced by the function d is a real number which satisfies the following axioms:

1. **Non-negativity :** The distance between any two points can never be negative.

   $d(x, y) \geq 0$

2. **Zero distance :** The distance between a point and itself is zero.

   $d(x, y) = 0$ iff $x = y$

3. **Symmetry :** The distance from x to y is same as the distance from y to x.

   $d(x, y) = d(y, x)$

4. **Triangle inequality :** The direct distance between x and y is always smaller than or equal to the distance between x and y via another point z. In other words, distance measure is the length of the shortest path between two points x and y.

   $d(x, y) \leq d(x, z) + d(z, y)$

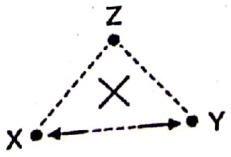


Fig. 6.1.1 : Triangle inequality

In this section we shall discuss about the following distance measures in details :

(1) Euclidean Distance (2) Jaccard Distance

(3) Cosine Distance (4) Edit Distance

(5) Hamming Distance

6.1.1 Euclidean Distances

Q. What do you mean by Euclidean distance ? Explain with example.

- The Euclidean distance is the most popular out of all the different distance measures.
- The Euclidean distance is measured on the Euclidean space. If we consider an n-dimensional Euclidean space then each point in that space is a vector of n real numbers. For example, if we consider the two-dimensional Euclidean space then each point in the space is represented by (x_1, x_2) where x_1 and x_2 are real numbers.
- The most familiar Euclidean distance measure is known as the L_2– norm which in the n-dimensional space is defined as :

$$d([x_1, x_2,\ldots x_n], [y_1, y_2,\ldots y_n]) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- For the two-dimensional space the L_2– norm will be :

$$d([x_1, x_2], [y_1, y_2]) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

- We can easily verify all the distance axioms on the Euclidean distance :
  1. **Non-negativity :** The Euclidean distance can never be negative as all the sum terms $(x_i - y_i)$ are squared and the square of any number whether positive or negative is always positive. So the final result will either be zero or a positive number.
  2. **Zero distance :** In case of the Euclidean distance from a point to itself all the x_i's will be equal to the y_i's. This in turn will make all the sum terms $(x_i - y_i)$ equal to zero. So the final result will also be zero.
  3. **Symmetry :** $(x_i - y_i)^2$ will always be equal to $(y_i - x_i)^2$. So the Euclidean distance is always symmetric.
  4. **Triangle inequality :** In Euclidean space, the length of the side of a triangle is always less than or equal to the sum of the lengths of the other two sides.
- Some other distance measures that are used on the Euclidean space are :
  1. L_r– norm where r is a constant :

$$d([x_1, x_2,\ldots x_n], [y_1, y_2,\ldots y_n]) = \left(\sum_{i=1}^{n} |x_i - y_i|^r\right)^{\frac{1}{r}}$$

  2. L_1– norm which is commonly known as Manhattan distance :

$$d([x_1, x_2,\ldots x_n], [y_1, y_2,\ldots y_n]) = \sum_{i=1}^{n} |x_i - y_i|$$

  3. L_∞– norm which is defined as :

$$d([x_1, x_2,\ldots x_n], [y_1, y_2,\ldots y_n]) = \max (|x_i - y_i|) \; \forall \; i$$

Ex. 6.1.1 : Consider the two points (10, 4) and (6, 7) in the two-dimensional Euclidean space. Find the Euclidean distance between them.

Soln. :

(1) L_2– norm $= \sqrt{(10-6)^2 + (4-7)^2}$

$= \sqrt{4^2 + 3^2}$

$= \sqrt{16+9}$

$= \sqrt{25}$

$= 5$

(2) L_1 – norm $= |10-6| + |4-7|$

$= 4 + 3$

$= 7$

(3) L_∞– norm $= \max(|10-6|, |4-7|)$

$= \max(4, 3)$

$= 4$

6.1.2 Jaccard Distance

MU - May 18

Q. What do you mean by Jaccard Similarity? (May 18, 2 Marks)

- Jaccard distance is measured in the space of sets. Jaccard distance between two sets is defined as :

  $d(x, y) = 1 - SIM(x, y)$

  SIM(x, y) is the Jaccard similarity which measures the closeness of two sets. Jaccard similarity is given by the ratio of the size of the intersection and the size of the union of the sets x and y.

- We can verify the distance axioms on the Jaccard distance :

1. **Non-negativity :** The size of the intersection of two sets can never be more than the size of the union. This means the ratio SIM(x, y) will always be a value less than or equal to 1. Thus d(x, y) will never be negative.
2. **Zero distance :** If x = y, then $x \cup x = x \cap x = x$. In this case SIM(x, y) = x/x = 1. Hence, d(x, y) = 1 – 1 = 0. In other words the Jaccard distance between the same set and itself is zero.
3. **Symmetry :** As both union as well as intersection are symmetric $x \cup y = y \cup x$ and $x \cap y = y \cap x$, hence Jaccard distance is also symmetric d(x, y) = d(y, x).
4. **Triangle inequality :** Jaccard distance can also be considered as the probability that a random minhash function does not map both the sets x and y to the same value.

   $P[h(x) \neq h(y)] \leq P[h(x) \neq h(z)] + P[h(z) \neq h(y)]$

   Where h is the random minhash function.

6.1.3 Cosine Distance

Q. What is Cosine distance? Explain with suitable example.

- The cosine distance is measured in those spaces which have dimensions. Examples of such spaces are :
  1. Euclidean spaces in which the vector components are real numbers, and
  2. Discrete versions of Euclidean spaces in which the vector components are integers or Boolean (0 and 1).
- Cosine distance is the angle made by the two vectors from the origin to the two points in the space. The range of this angle is between 0 to 180 degrees.
- The steps involved in calculating the cosine distance given two vectors x and y are :
  1. Find the dot product x.y :

$$([x_1, x_2,\ldots x_n], [y_1, y_2,\ldots y_n]) = \sum_{i=1}^{n} x_i y_i$$

  2. Find the L_2– norms of both x and y,
  3. Divide the dot product x.y by the L_2– norms of x and y to get cos θ where θ is the angle between x and y.
  4. Finally, to get θ use the $\cos^{-1}$ function.
- Let us illustrate the concepts with an Ex. 6.1.2.
- The distance axioms on the Cosine distance may be verified as follows :
  1. **Non-negativity :** The range of the possible values is from 0 to 180 degrees. So there is no question of negative distance.
  2. **Zero distance :** If two vectors are the same direction, only then the angle between them will be zero.
  3. **Symmetry :** The angle between x and y will always be equal to the angle between y and x.
  4. **Triangle inequality :** The sum of the rotations from x to z and then z to y can never be less than the direct rotation from x to y.

Ex. 6.1.2 : Consider the following two vectors in the Euclidean space :
x = [1, 2, – 1], and y = [2, 1, 1].
Calculate the cosine distance between x and y.

Soln. :

Given $x = [1,2,-1]$; $y = [2, 1, 1]$

(i) $$x \cdot y = [1\times2] + [2\times1] + [(-1)\times1]$$
$$= 2 + 2 + (-1) = 2 + 2 - 1$$
$$= 4 - 1 = 3$$

(ii) L_2 norm for x $= \sqrt{(1)^2 + (2)^2 + (-1)^2} = \sqrt{6}$

L_2 norm for y $= \sqrt{(2)^2 + (1)^2 + (1)^2} = \sqrt{6}$

(iii) $$\cos\theta = \frac{x,y}{(L_2 - \text{norm of x}), (L_2 - \text{norm of y})} = \frac{3}{\sqrt{6},\sqrt{6}} = \frac{3}{6} = \frac{1}{2}$$

Now,

$$\text{Cos}^{-1} = 60°$$

∴ The angle between the two vectors x and y is 60°.

6.1.4 Edit Distance

Q. What is Edit distance? Explain with classical method.
Q. What is Edit distance? Explain with Longest Common Sequence (LCS) method.

- The points in the space for edit distance are strings.
- There are two different methods for defining and calculating edit distances :
  1. The classical method
  2. The Longest Common Subsequence (LCS) method

(I) Classical method

- The Edit distance between two strings x and y is the least number of edit operations required to convert x into y. Only two edit operations are allowed :
  1. Insertion of a single character, and
  2. Deletion of a single character.
- To illustrate let us take the following two strings :

x = JKLMN

y = JLOMNP

- For calculating the Edit distance between x and y we have to convert string x into string y using the edit operations of insertion and deletion.
- At first compare the character sequences in both the strings :

x	=	J	K	L	M	N	
y	=	J	L	O	M	N	P
		↓	↓	↓	↓	↓	↓
		①	②	③	④	⑤	⑥ → Positions

- Clearly, positions②,③and⑥are having different characters in x and y. So we need to make the necessary insertions and deletions at these three positions.

K	L	–
L	O	P
↓	↓	↓
②	③	⑥

- From position ② of string x, we have to delete the character K. The characters following K will be shifted one position to the left.

- After the first edit operation (deletion)the status of the string x is :

x =	J	L	M	N
	①	②	③	④

- Now the character O has to be inserted at position 3 i.e. after the character L and before the character M.
- After the second edit operation (insertion) the status of the string x is :

x =	J	L	O	M	N
	↓	↓	↓	↓	↓
	①	②	③	④	⑤

- In the final step, the character P has to be inserted in the string x at position 6.
- After the third and final edit operation (insertion) the status of the string x is :

X =	J	L	O	M	N	P
	↓	↓	↓	↓	↓	↓
	①	②	③	④	⑤	⑥

- So the Edit distance between x and y is :

$$d(x, y) = \text{Number of deletions} + \text{Number of insertions}$$
$$= 1 + 2 = 3$$

(ii) Longest Common Subsequence (LCS)

- The longest common subsequence of two strings x and y is a subsequence of maximum length which appears in both x and y in the same relative order, but not necessarily contiguous.
- Let us illustrate the concept of finding the Edit distance using LCS method with the same set of strings as in the previous method :

x =	J	K	L	M	N
y =	J	L	O	M	N	P

- The longest common subsequence in x and y = JLMN.
- The formula of Edit distance using LCS is :

$$d(x, y) = \text{length of string } x + \text{length of string } y - 2 \times (\text{length of LCS})$$

Here,

$$\text{length of string } x = 5,$$
$$\text{length of string } y = 6,$$
$$\text{length of LCS} = 4,$$

So,

$$d(x, y) = 5 + 6 - 2 \times (4)$$
$$= 11 - 8$$
$$= 3$$

- The distance axioms on the Edit distance may be verified as follows :

1. **Non-negativity :** To convert one string into another string at least zero or more insertions and/or deletions are necessary. So, the Edit distance can never be negative.
2. **Zero distance :** Only in the case of two identical strings, the Edit distance will be zero.
3. **Symmetry :** The edit distance for converting string x into string y will be the same for converting string y into string x as the sequence of insertions and deletions can be reversed.
4. **Triangle inequality :** The sum of the number of edit operations required for converting string x into string z and then string z into string y can never be less than the number of edit operations required for converting the string x directly into the string y.

6.1.5 Hamming Distance

MU - Dec. 17

Q. Explain Hamming distance measure with an example. (Dec. 17, 5 Marks)

- Hamming distance is applicable in the space of vectors. Hamming distance between two vectors is the number of components in which they differ from each other.
- For example, let us consider the following two vectors :

x	=	1	0	0	0	1	1
y	=	1	1	1	0	1	0
		↓	↓	↓	↓	↓	↓
		①	②	③	④	⑤	⑥ → Positions

- The Hamming distance between the above two vectors is 3 because components at positions 2, 3 and 6 are different.
- The distance axioms on the Hamming distance may be verified as follows :

1. **Non-negativity :** Any two vectors will differ in at least zero or more component positions. So, the Hamming distance can never be negative.
2. **Zero distance :** Only in the case of two identical vectors, the Hamming distance will be zero.
3. **Symmetry :** The Hamming distance will be the same whether x is compared with y or y is compared with x.
4. **Triangle inequality :** The number of differences between x and z, plus the number of differences between z and y can never be less than the number of differences between x and y.

Review Questions

Q. 1 What is Jaccard similarity ?

Q. 2 What is distance Measure ? Explain different criteria's regarding distance measures.

Q. 3 What do you mean by Euclidean distance ? Explain with example.

Q. 4 What is Cosine distance ? Explain with suitable example.

Q. 5 Consider following are the two vectors in Euclidean space X = [1, 2, – 1] and Y = [2,1,1]. Calculate the cosine distance between X and Y.

Q. 6 What is Edit distance ? Explain with classical method.

Q. 7 What is Edit distance ? Explain with Longest Common Subsequence (LCS) method.