University of Mumbai

Program: **Electronics and Telecommunications**

Curriculum Scheme: Rev 2016

Examination: BE Semester:VII

Course Code:ECCDL7032 and Course Name: Big Data Analytics

Time: 2 hour 30 minutes Max. Marks: 80

==========

Q1. Choose the correct option for following questions. All the Questions are compulsory and carry equal marks

Question Number	**Correct Option (Enter either 'A' or 'B' or 'C' or 'D')**
Q1.	**D**
Q2.	**B**
Q3.	**C**
Q4	**D**
Q5	**D**
Q6	**B**
Q7	**B**
Q8.	**B**
Q9.	**A**
Q10.	**C**

Q2.

A. Write the Map Reduce Pseudo code for any two of the following relational algebra operations

i) Projection ii) Selection iii) Natural Join iv) Matrix-Vector Multiplication v) Grouping & Aggregation.

Expected Answer :

Relational Algebra Operations:

i) projection: **2 Marks**

for some subset s of the attribute of the relation, produce from each tuple only the components for the attributes in S.

The result of this projection is denoted TTs (R)

Projection is performed similarly to selection.

As projection may cause the same tuple to appear several times, the reduce function eliminate duplicates.

The pseudo code for projection is as follows :

Map (key, valve)

for tuple in valve :

ts = tuple with only the components for the attributes in S.

emit (ts, ts)

Reduce (key, values)

emit (key, key)

ii) Selection: **2 Marks**

Apply a condition c to each taple in the relation and produce as output only those tuples that satisfy c.

The result of this selection is denoted by 6c(R)

Selection really do not need the full power of MapReduce.

They can be done most conveniently in the map portion alone, although they could also be done in the reduce portion also.

The pseudo code is as follows :

Map (key, valve)

for tuple in valve :

if tuple satisfies C :

emit (tuple, tuple)

Reduce (key, valves)

emit (key, key)

Similarly write for rest 3

iii) Natural Join **2 Marks**

iv) Matrix-Vector Multiplication **2 Marks**

v) Grouping & Aggregation. **2 Marks**

Q.2

B. Explain any one clustering algorithm for mining Social Network Graph.

Expected Answer:

Explain distant measures of Social network graphs –**2 Marks**

Explain Standard clustering technique – **2 Marks**

Measure for graph clustering – **2 Marks**

Girvan-Newman Algorithm – **2 Marks**

Clique percolation Method- **2 Marks**

Q.2

C. Explain Column family store and Graph Store NoSQL Architectural pattern with examples.

Expected Answer:

A data architecture pattern is a consistent way of representing data in a regular structure that will be stored in memory. Architectural patterns allow you to give precise names to recurring high level data storage patterns. When you suggest a specific data architecture pattern as a solution to a business problem, you should use a consistent process that allows you to name the pattern, describe how it applies to the current business problem, and articulate the pros and cons of the proposed solution. It's important that all team members have the same understanding about how a

particular pattern solves your problem so that when implemented, business goals and objectives are met. **2 marks**

Column Family Architectural Pattern:**4 marks**

Column family systems are important NoSQL data architecture patterns because they can scale to manage large volumes of data. They're also known to be closely tied with many MapReduce systems.

Column family stores use row and column identifiers as general purposes keys for data lookup. They're sometimes referred to as data stores rather than databases, since they lack features you may expect to find in traditional databases. For example, they lack typed columns, secondary indexes, triggers, and query languages. Almost all column family stores have been heavily influenced by the original Google Bigtable paper. HBase, Hypertable, and Cassandra are good examples of systems that have Bigtablelike interfaces, although how they're implemented varies.

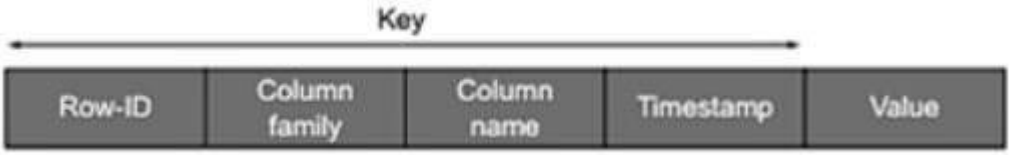


Figure: The key structure in column family stores is similar to a spreadsheet but has two additional attributes. In addition to the column name, a column family is used to group similar column names together. The addition of a timestamp in the key also allows each cell in the table to store multiple versions of a value over time.

Graph Store Architectural Pattern:**4 marks**

A graph store is a system that contains a sequence of nodes and relationships that, when combined, create a graph. a graph store has three data fields: nodes, relationships, and properties. Some types of graph stores are referred to as triple stores because of their node-relationship-node structure

Graph nodes are usually representations of real-world objects like nouns. Nodes can be people, organizations, telephone numbers, web pages, computers on a network, or even biological cells in a living organism. The relationships can be thought of as connections between these objects and are typically represented as arcs (lines that connect) between circles in diagrams.

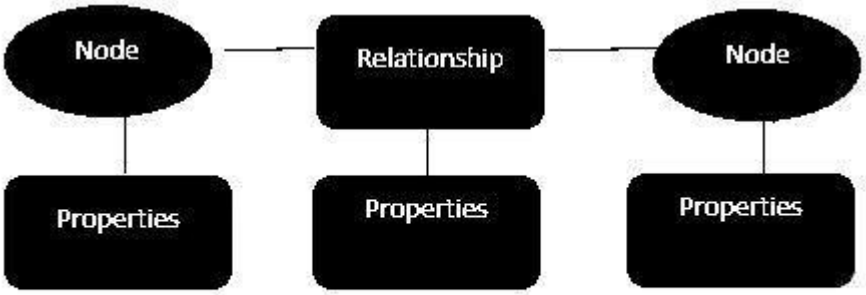


Figure: A graph store consists of many node-relationship-node structures. Properties are used to describe both the nodes and relationships.

Graph stores are important in applications that need to analyze relationships between objects or visit all nodes in a graph in a particular manner (graph traversal). Graph stores are highly optimized to efficiently store graph nodes and links, and allow you to query these graphs. Graph databases are useful for any business problem that has complex relationships between objects such as social networking, rules-based engines, creating mashups, and graph systems that can quickly analyze complex network structures and find patterns within these structures.

Q.3

A. Compare Content based Recommendation system with Collaborative filtering based Recommendation system

Expected Answer:

Collaborative filtering system**: 2 Marks**

• It uses community data from peer group for recommendation.

• These exhibits all those things that are popular among the peers.

• These filtering systems recommended items based on similarity measure between users and / or items

• Here user profile and contextual parameters along with the community data are used by the recommender systems to personalize the recommendation list.

• This is the most prominent approach in e-commerce site.

Example: **1 Mark**

Consider a movie rating system

The basic assumption for collaborative filtering is:

User gives ratings to items in the catalog

• Customers who had similar taste in the past will have similar taste in future

• Users who agreed in their subjective evaluations in the past will agree in the future too.

• To find out similarity we can use Pearson's correlation co-efficient as:

prediction and Pearson's correlation coefficient **–3 Marks**

We can apply this to movie rating system and based on that we can predict rating for movie as well as to whom the movie should be recommended.

Collaborative vs content based -4 Marks

- Content based systems examine the properties of the item examined or recommended.
- These systems takes input from the user profile and the contextual parameters along with product features to make the recommendation list.
- Similarity of items is determined by measuring the similarity in their properties.
- These systems need some information related to the content of available items, but no complete information.
- It also requires user profiling describing what user likes.
- In movie recommendation system- content based filtering systems will recommend the movie based on users profile information like age, gender, etc as well as properties of movie like genre, actors, etc.

Q.3

B. What is MapReduce? Explain how map and reduce works?

Expected Answer:

MapReduce is a style of computing that has been implemented in several systems, including Google's internal implementation (simply called MapReduce) and the popular open-source implementation Hadoop which can be obtained, along with the HDFS file system from the Apache Foundation. You can use an implementation of MapReduce to manage many large-scale computations in a way that is tolerant of hardware faults. All you need to write are two functions, called Map and Reduce, while the system manages the parallel execution, coordination of tasks that execute Map or Reduce, and also deals with the possibility that one of these tasks will fail to execute. **2Marks**

In brief, a MapReduce computation executes as follows:**4 Marks**

Some number of Map tasks each are given one or more chunks from a distributed file system. These Map tasks turn the chunk into a sequence of key-value pairs. The way key-value pairs are produced from the input data is determined by the code written by the user for the Map function.

The key-value pairs from each Map task are collected by a master controller and sorted by key. The keys are divided among all the Reduce tasks, so all key-value pairs with the same key wind up at the same Reduce task.

The Reduce tasks work on one key at a time, and combine all the values associated with that key in some way. The manner of combination of values is determined by the code written by the user for the Reduce function.

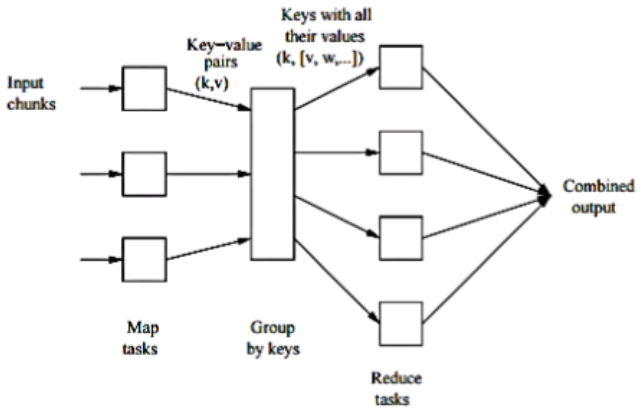
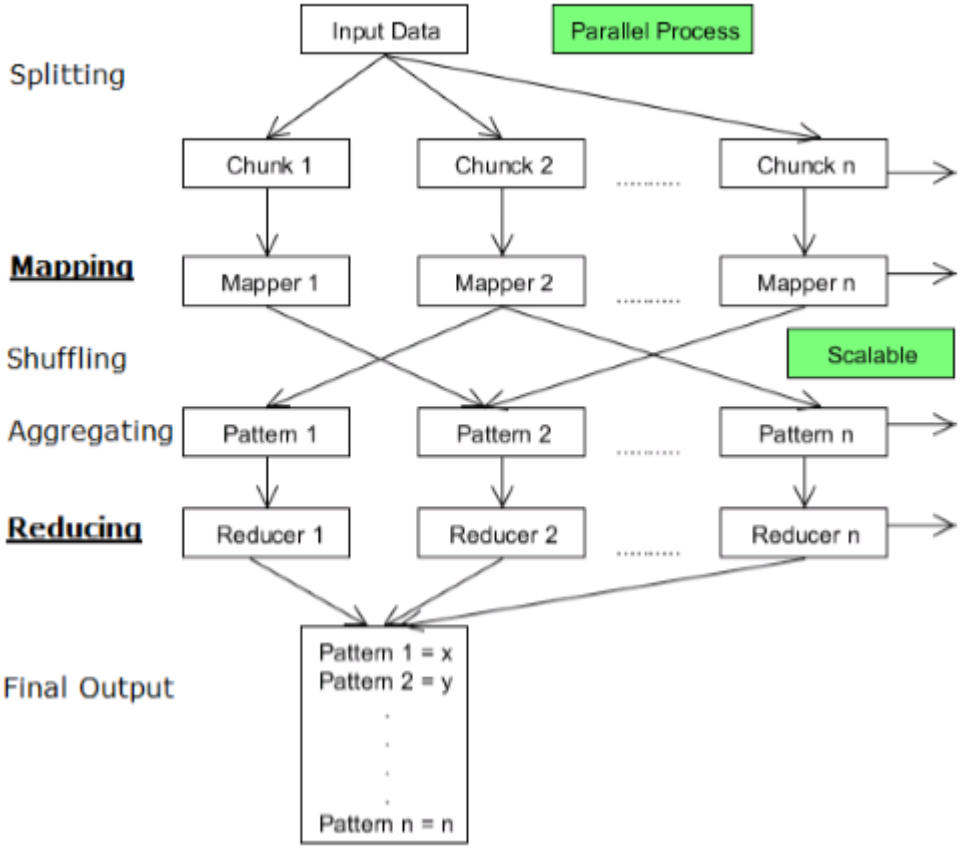


Figure: **Schematic of a MapReduce computation**

Flow of Map Reduce Algorithm : **4 Marks**

To following diagram summarizes the flow of Map reduce algorithm:



In the above map reduce flow:

1. The input data can be divided into n number of chunks depending upon the amount of data and processing capacity of individual unit.
2. Next, it is passed to the mapper functions. Please note that all the chunks are processed simultaneously at the same time, which embraces the parallel processing of data.
3. After that, shuffling happens which leads to aggregation of similar patterns.
4. Finally, reducers combine them all to get a consolidated output as per the logic.
5. This algorithm embraces scalability as depending on the size of the input data, we can keep increasing the number of the parallel processing units.

While processing large set of data, we should definitely address scalability and efficiency in the application code that is processing the large amount of data.

Map reduce algorithm (or flow) is highly effective in handling big data.

Let us take a simple example and use map reduce to solve a problem. Say you are processing a large amount of data and trying to find out what percentage of your user base where talking about games.

First, we will identify the keywords which we are going to map from the data to conclude that its something related to games.

Next, we will write a mapping function to identify such patterns in our data. For example, the keywords can be Gold medals, Bronze medals, Silver medals, Olympic football, basketball, cricket, etc.

Let us take the following chunks in a big data set and see how to process it.

“Hi, how are you”

“We love football”

“He is an awesome football player”

“Merry Christmas”

“Olympics will be held in China”

“Records broken today in Olympics”

“Yes, we won 2 Gold medals”

“He qualified for Olympics”

Mapping Phase

So our map phase of our algorithm will be as follows:

1. Declare a function “Map”
2. Loop: For each words equal to “football”
3. Increment counter
4. Return key value “football”=>counter

In the same way, we can define n number of mapping functions for mapping various words:

“Olympics”, “Gold Medals”, “cricket”, etc.

Reducing Phase

The reducing function will accept the input from all these mappers in form of key value pair and then processing it. So, input to the reduce function will look like the following:

- reduce(“football”=>2)
- reduce(“Olympics”=>3)

Our algorithm will continue with the following steps:

i. Declare a function reduce to accept the values from map function.

ii. Where for each key-value pair, add value to counter.

iii. Return “games”=> counter.

At the end, we will get the output like “games”=>5.

Q.3

C. Explain the recommendation system model along with the applications of it.

Expected Answer:

The recommendation system model uses the utility matrix and the concept of "long-tail" which explains the advantage of online vendors over conventional, brick-and-mortar vendors.

The utility Matrix: 4 Marks

In a recommendation-system application there are two classes of entities, which we shall refer to as users and items. Users have preferences for certain items, and these preferences must be teased out of the data. The data itself is represented as a utility matrix, giving for each user-item pair, a value that represents what is known about the degree of preference of that user for that item. Values come from an ordered set, e.g., integers 1–5 representing the number of stars that the user gave as a rating for that item. We assume that the matrix is sparse, meaning that most entries are "unknown." An unknown rating implies that we have no explicit information about the user's preference for the item.

Example: In Fig. A we see an example utility matrix, representing users' ratings of movies on a 1–5 scale, with 5 the highest rating. Blanks represent the situation where the user has not rated the movie. The movie names are HP1, HP2, and HP3 for Harry Potter I, II, and III, TW for Twilight, and SW1, SW2, and SW3 for Star Wars episodes 1, 2, and 3. The users are represented by capital letters A through D.

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Figure A: A utility matrix representing ratings of movies on a 1–5 scale

Notice that most user-movie pairs have blanks, meaning the user has not rated the movie. In practice, the matrix would be even sparser, with the typical user rating only a tiny fraction of all available movies.

The goal of a recommendation system is to predict the blanks in the utility matrix. For example, would user A like SW2? There is little evidence from the tiny matrix in Fig. A. We might design our recommendation system to take into account properties of movies, such as their producer, director, stars, or even the similarity of their names. If so, we might then note the similarity between SW1 and SW2, and then conclude that since A did not like SW1, they were unlikely to enjoy SW2 either. Alternatively, with much more data, we might observe that the people who rated both SW1 and SW2 tended to give them similar ratings. Thus, we could conclude that A would also give SW2 a low rating, similar to A's rating of SW1.

It is not necessary to predict every blank entry in a utility matrix. Rather, it is only necessary to discover some entries in each row that are likely to be high.

The Long Tail: 4 Marks

Physical delivery systems are characterized by a scarcity of resources. Brick-and-mortar stores have limited shelf space, and can show the customer only a small fraction of all the choices that exist. On the other hand, on-line stores can make anything that exists available to the customer.

Thus, a physical bookstore may have several thousand books on its shelves, but Amazon offers millions of books. A physical newspaper can print several dozen articles per day, while on-line news services offer thousands per day.

Recommendation in the physical world is fairly simple. First, it is not possible to tailor the store to each individual customer. Thus, the choice of what is made available is governed only by the aggregate numbers. Typically, a bookstore will display only the books that are most popular, and a newspaper will print only the articles it believes the most people will be interested in. In the first case, sales figures govern the choices, in the second case, editorial judgment serves.

The distinction between the physical and on-line worlds has been called the long tail phenomenon, and it is suggested in Fig. B. The vertical axis represents popularity (the number of times an item is chosen). The items are ordered on the horizontal axis according to their popularity. Physical institutions provide only the most popular items to the left of the vertical line, while the corresponding on-line institutions provide the entire range of items: the tail as well as the popular items.

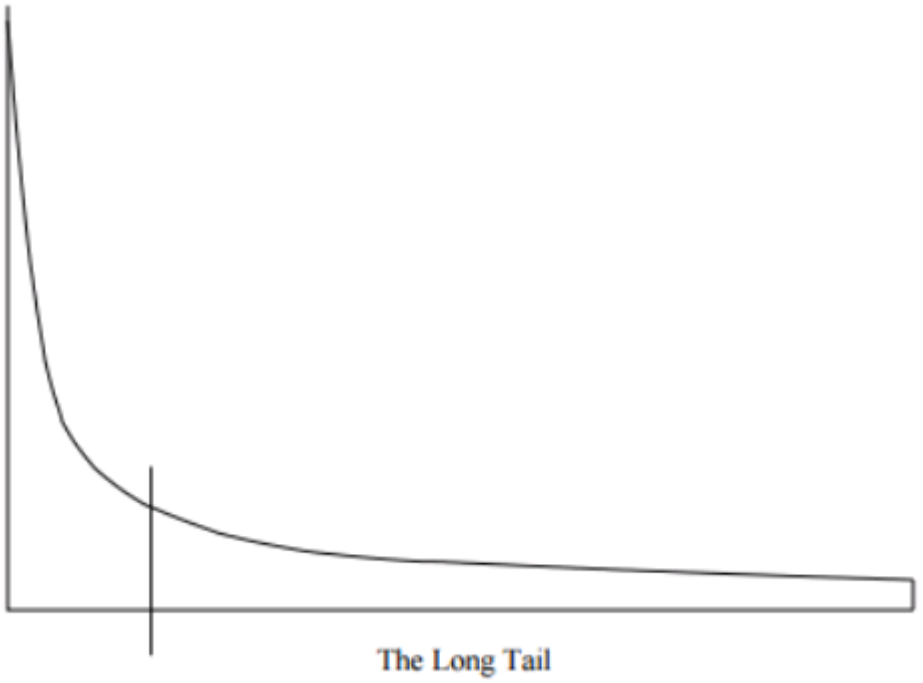


Figure B: The long tail: physical institutions can only provide what is popular, while on-line institutions can make everything available

The long-tail phenomenon forces on-line institutions to recommend items to individual users. It is not possible to present all available items to the user, the way physical institutions can. Neither can we expect users to have heard of each of the items they might like.

Applications of Recommendation Systems: 2 Marks

1. Product Recommendations: Perhaps the most important use of recommendation systems is at on-line retailers. We have noted how Amazon or similar on-line vendors strive to present each returning user with some suggestions of products that they might like to buy. These suggestions are not random, but are based on the purchasing decisions made by similar customers or on other techniques.

2. Movie Recommendations: Netflix offers its customers recommendations of movies they might like. These recommendations are based on ratings provided by users.
3. News Articles: News services have attempted to identify articles of interest to readers, based on the articles that they have read in the past. The similarity might be based on the similarity of important words in the documents, or on the articles that are read by people with similar reading tastes. The same principles apply to recommending blogs from among the millions of blogs available, videos on YouTube, or other sites where content is provided regularly.
4. Mobile Recommender Systems: One example of a mobile recommender system is one that offers potentially profitable driving routes for taxi drivers in a city. This system takes as input data in the form of GPS traces of the routes that taxi drivers took while working, which include location (latitude and longitude), time stamps, and operational status (with or without passengers). It then recommends a list of pickup points along a route that will lead to optimal occupancy times and profits. This type of system is obviously location-dependent, and as it must operate on a handheld or embedded device, the computation and energy requirements must remain low.

Q.4

A. Explain with diagrams the Park Chen Yu (PCY) Algorithm.

Expected Answer:

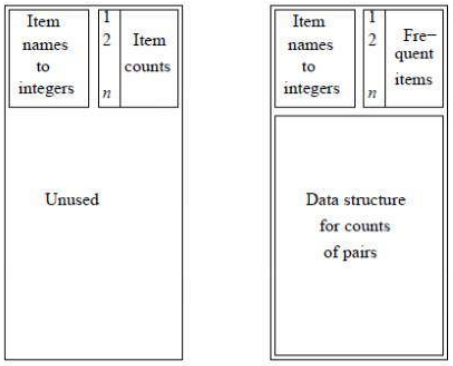


Fig. 1– a translation table from item names to small integers and an array to count those integers.

5 Marks

This algorithm, which we call PCY after its authors, exploits the observation that there may be much unused space in main memory on the first pass. If there are a million items and gigabytes of main memory, we do not need more than 10% of the main memory for the two tables suggested in above Figure.

The PCY Algorithm uses that space for an array of integers that generalizes the idea of a Bloom filter.

The idea is shown schematically in Fig. 2.Think of this array as a hash table, whose buckets hold integers rather than sets of keys (as in an ordinary hash table) or bits (as in a Bloom filter).

Pairs of items are hashed to buckets of this hash table. As we examine a basket during the first pass, we not only add 1 to the count for each item in the basket, but we generate all the pairs, using a double loop.

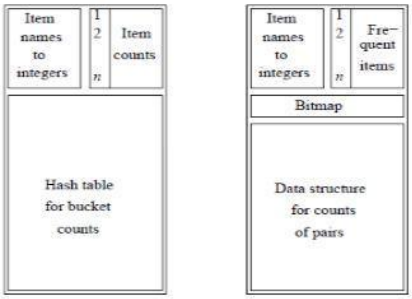


Figure 2: Organization of main memory for the first two passes of the PCY Algorithm **5Marks**

We hash each pair, and we add 1 to the bucket into which that pair hashes. Note that the pair itself doesn't go into the bucket; the pair only affects the single integer in the bucket.
At the end of the first pass, each bucket has a count, which is the sum of the counts of all the pairs that hash to that bucket. If the count of a bucket is at least as great as the support threshold s, it is called a frequent bucket.
We can say nothing about the pairs that hash to a frequent bucket; they could all be frequent pairs from the information available to us. But if the count of the bucket is less than s (an infrequent bucket), we know no pair that hashes to this bucket can be frequent, even if the pair consists of two frequent items.
That fact gives us an advantage on the second pass. We can define the set of candidate pairs C2 to be those pairs {i, j} such that:

- i and j are frequent items.
- {i, j} hashes to a frequent bucket.

Q. 4

B. What is NoSQL Data Architectural Pattern? What are its different types?

Expected Answer:

Definition: 2 Marks

a data architecture pattern is a consistent way of representing data in a regular structure that will be stored in memory. Although the memory you store data in is usually long-term persistent

memory, such as solid state disk or hard drives, these structures can also be stored in RAM and then transferred to persistent memory by another process.

There are two types of architectural Patterns:

- High level Architecture Pattern
- Low level Architecture Pattern

Architectural patterns allow you to give precise names to recurring high level data storage patterns. When you suggest a specific data architecture pattern as a solution to a business problem, you should use a consistent process that allows you to name the pattern, describe how it applies to the current business problem, and articulate the pros and cons of the proposed solution. It's important that all team members have the same understanding about how a particular pattern solves your problem so that when implemented, business goals and objectives are met.

Types of Architecture Patterns:

1. Key Value Store:2 Marks

A key-value store is a simple database that when presented with a simple string (the key) returns an arbitrary large BLOB of data (the value). Key-value stores have no query language; they provide a way to add and remove key-value pairs (a combination of key and value where the key is bound to the value until a new value is assigned) into/from a database. A key-value store is like a dictionary. A dictionary has a list of words and each word has one or more definitions

The key in a key-value store is flexible and can be represented by many formats:

- Logical path names to images or files
- Artificially generated strings created from a hash of the value
- REST web service calls
- SQL queries

2. Graph stores 2 Marks

A graph store is a system that contains a sequence of nodes and relationships that, when combined, create a graph. A graph store has three data fields: nodes, relationships, and properties. Some types of graph stores are referred to as triple stores because of their node-relationship-node structure

Graph nodes are usually representations of real-world objects like nouns. Nodes can be people, organizations, telephone numbers, web pages, computers on a network, or even biological cells in a living organism. The relationships can be thought of as connections between these objects and are typically represented as arcs (lines that connect) between circles in diagrams.

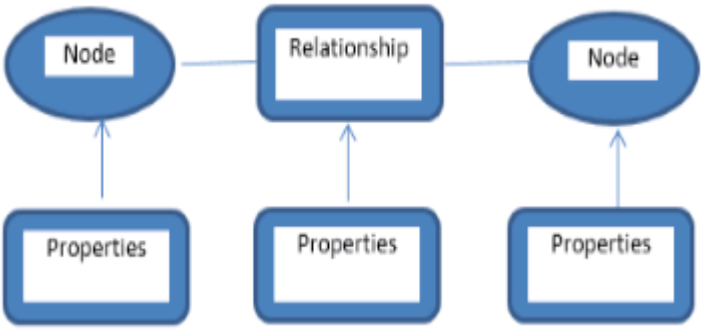


Figure: A graph store consists of many node-relationship-node structures. Properties are used to describe both the nodes and relationships.

Graph stores are important in applications that need to analyze relationships between objects or visit all nodes in a graph in a particular manner (graph traversal). Graph stores are highly optimized to efficiently store graph nodes and links, and allow you to query these graphs. Graph databases are useful for any business problem that has complex relationships between objects such as social networking, rules-based engines, creating mashups, and graph systems that can quickly analyze complex network structures and find patterns within these structures.

3. Column Family (BigTable) Stores: 2 Marks

Column family systems are important NoSQL data architecture patterns because they can scale to manage large volumes of data. They're also known to be closely tied with many MapReduce systems. Column family stores use row and column identifiers as general purposes keys for data lookup. They're sometimes referred to as data stores rather than databases, since they lack features you may expect to find in traditional databases. For example, they lack typed columns, secondary indexes, triggers, and query languages. Almost all column family stores have been heavily influenced by the original Google Bigtable paper. HBase, Hypertable, and Cassandra are good examples of systems that have Bigtablelike interfaces, although how they're implemented varies.

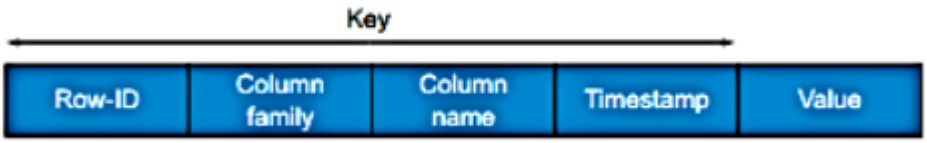


Figure: The key structure in column family stores is similar to a spreadsheet but has two additional attributes. In addition to the column name, a column family is used to group similar column names together. The addition of a timestamp in the key also allows each cell in the table to store multiple versions of a value over time.

4. Document Stores: 2 Marks

Think of a document store as a tree-like structure, as shown in figure. Document trees have a single root element (or sometimes multiple root elements). Beneath the root element there is a sequence of branches, sub-branches, and values. Each branch has a related path expression that shows you how to navigate from the root of the tree to any given branch, sub-branch, or value. Each branch may have a value associated with that branch. Sometimes the existence of a branch in

the tree has specific meaning, and sometimes a branch must have a given value to be interpreted correctly.

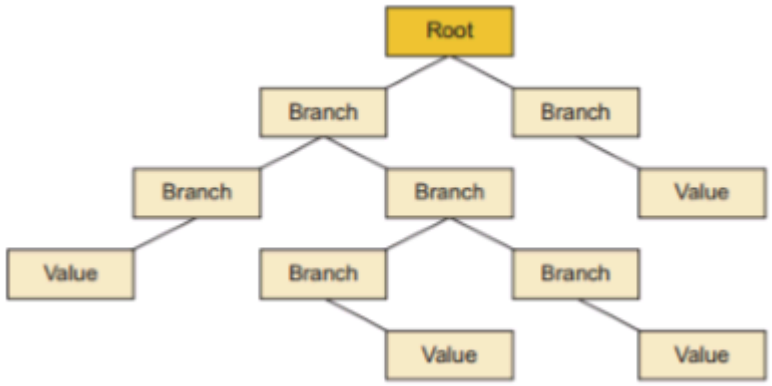


Fig: 1 Document stores use a tree structure that begins with a root node, and have subbranches that can also contain sub-branches. The actual data values are usually stored at the leaf levels of a tree.

Q,4

C.What is search engine? Discuss in detail Algorithm-Based Ranking System.

Expected Answer:

Search Engine: **2 Marks**

- Search Engine is an application that is used to search for information on the Internet according to a specified criterion.
- Without search engines, finding the desired information from the colossus Internet would be painstaking and cumbersome task.
- In other words, the search engine searches the Internet for the information you want and displays it on your computer screen, all within a few seconds.
- Therefore, the task that would have taken hours to be completed by you if you do it manually can be now done in seconds by the search engine.
- Some popular search engines are Google, Yahoo, MSN, AltaVista, Look Smart and Netscape.

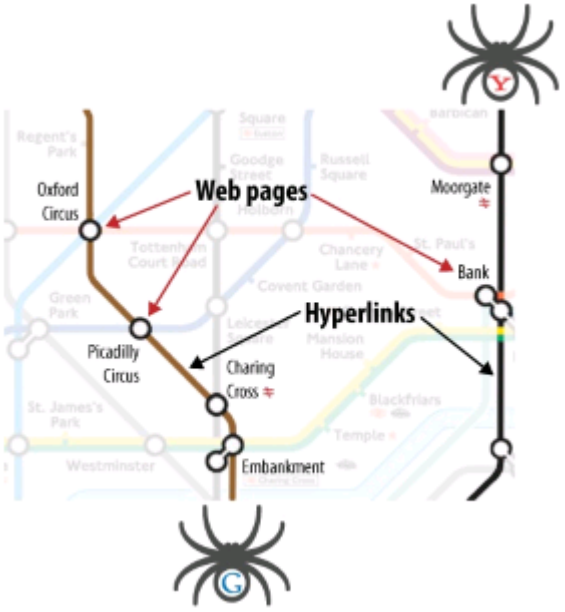
Algorithm-Based Ranking System: 4marks

Understanding how crawling, indexing, and ranking works is helpful to SEO practitioners, as it helps them determine what actions to take to meet their goals. This primarily covers the way Google and Bing operate. The search engines must execute multiple tasks very well to provide relevant search results. Put simplistically, you can think of these as:

- Crawling and indexing billions of documents (pages and files) on the Web (note that they ignore pages that they consider to be “insignificant,” perhaps because they are perceived as adding no new value or are not referenced at all on the Web).
- Responding to user queries by providing lists of relevant pages.

Crawling and Indexing:

- The web has link structure which serves to bind all the pages that were made public during linking with other pages.
- The search engine automated robots also known as crawlers or spiders have capability to reach billions of into connected documents.
- Once the engine finds these pages, its next job is to parse the code and store selected pieces in the form of array on hard drives. The retrieval is done when it is called through query.



Retrieval and Ranking: 4 marks

- After crawling and indexing the search engine returns a list of relevant pages on the web to satisfy user requirement.
- Two things are involved in this process:
    - Only related result should be returned which is asked in users query.
    - Rank the result as per importance taking into account the trust and authority associated with the site.
- Relevance:

  It is degree of matches found in any document returned by search or the amount of data related to the users query intention. Relevance is the first step if the web pages are not relevant to the search query as the search engine does not consider them for search results.

- Importance or popularity:

  It refers to the relative importance which is measured through citation. It can also be considered as out of one work referencing another. If a given document matches with users query the citation increases with number of other document that references it. Popularity and relevance cannot be determined manually, to achieve these search engines which have algorithms and several mathematical equations.