CELEBRATING 25 YEARS
CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A+ GRADE BY NAAC

Internal Assessment Test 2 Scheme and solution – Dec 2021

Sub:	Big Data Analytics					Sub Code:	18CS72	Branch:	CSE	
Date:	21/12/21	Duration:	90 mins	Max Marks:	5(	Sem / Sec:	VII/A,B,C		OBE	

	Answer any 5 FIVE FULL Questions	MARKS	CO	RBT
1	Explain working of JobTracker and TaskTracker. diagram-2 marks working of JobTracker and TaskTracker-8 marks	[10]	CO2	L2
2	Explain read and write operation on Hadoop. diagram-2 marks read and write operation-8 marks	[10]	CO2	L2
3	Explain Graph databases with characteristics, use and limitations. Graph database- 2 marks Characteristics including working-4marks Use- 2 marks Limitations-2 marks	[10]	CO2	L2
4	Explain working of Flume agent. Explain importance AVRO in pipelining Flume. Working of Flume agent- 5 marks Diagram -1 mark AVRO with diagram-4+1 marks	[10]	CO2	L2
5	Explain Oozie features and its workflow in detail. Oozie features- 3 marks Workflow with diagram- 6+1 marks	[10]	CO2	L2
6	Explain NoSQL data store structures i) Parquet File Formats ii) ORC file format Each data store – 5 marks	[10]	CO2	L2
7	Explain Mongo DB features, replication and autosharding process MongoDB features- 6 marks Replication -2 marks Autosharding- 2 marks	[10]	CO2	L2

1. Explain working of JobTracker and TaskTracker.

MapReduce runs as per assigned Job by JobTracker, which keeps track of the job submitted for execution and runs TaskTracker for tracking the tasks. MapReduce programming enables job scheduling and task execution as follows:

A client node submits a request of an application to the JobTracker. A JobTracker is a Hadoop daemon (background program). The following are the steps on the request to MapReduce: (i) estimate the need of resources for processing that request, (ii) analyze the states of the slave nodes, (iii) place the mapping tasks in queue, (iv) monitor the progress of task, and on the failure, restart the task on slots of time available. The job execution is controlled by two types of processes in MapReduce:

1. The Mapper deploys map tasks on the slots. Map tasks assign to those nodes where the data for the application is stored. The Reducer output transfers to the client node after the data serialization using AVRO.

2. The Hadoop system sends the Map and Reduce jobs to the appropriate servers in the cluster. The Hadoop framework in turns manages the task of issuing jobs, job completion and copying data around the cluster between the slave nodes. Finally, the cluster collects and reduces the data to obtainthe result and sends it back to the Hadoop server after completion of the given tasks.

The job execution is controlled by two types of processes in MapReduce. A single master process called JobTracker is one. This process coordinates all jobs running on the cluster and assigns map and reduce tasks to run on the TaskTrackers. The second is a number of subordinate processes called TaskTrackers. These processes run assigned tasks and periodically report the progress to the JobTracker.

Figure 2.4 showed the job execution model of MapReduce. Here the JobTracker schedules jobs submitted by clients, keeps track of TaskTrackers and maintainsthe available Map and Reduce slots. The JobTracker also monitors the execution of jobs and tasks on the cluster. The TaskTracker executes the Map and Reducetasks, and reports to the JobTracker.

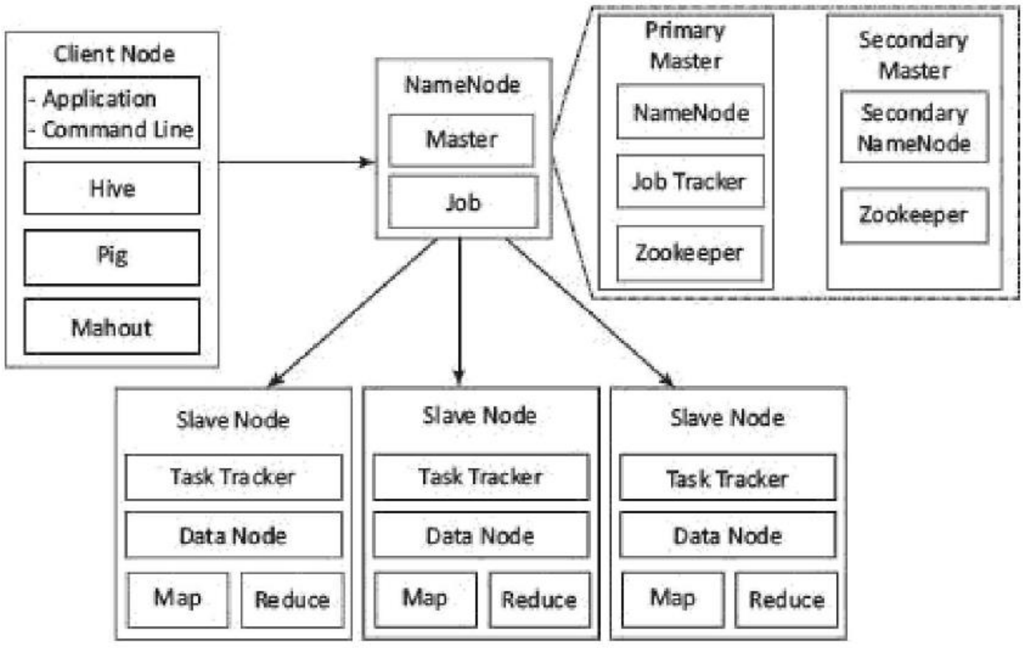
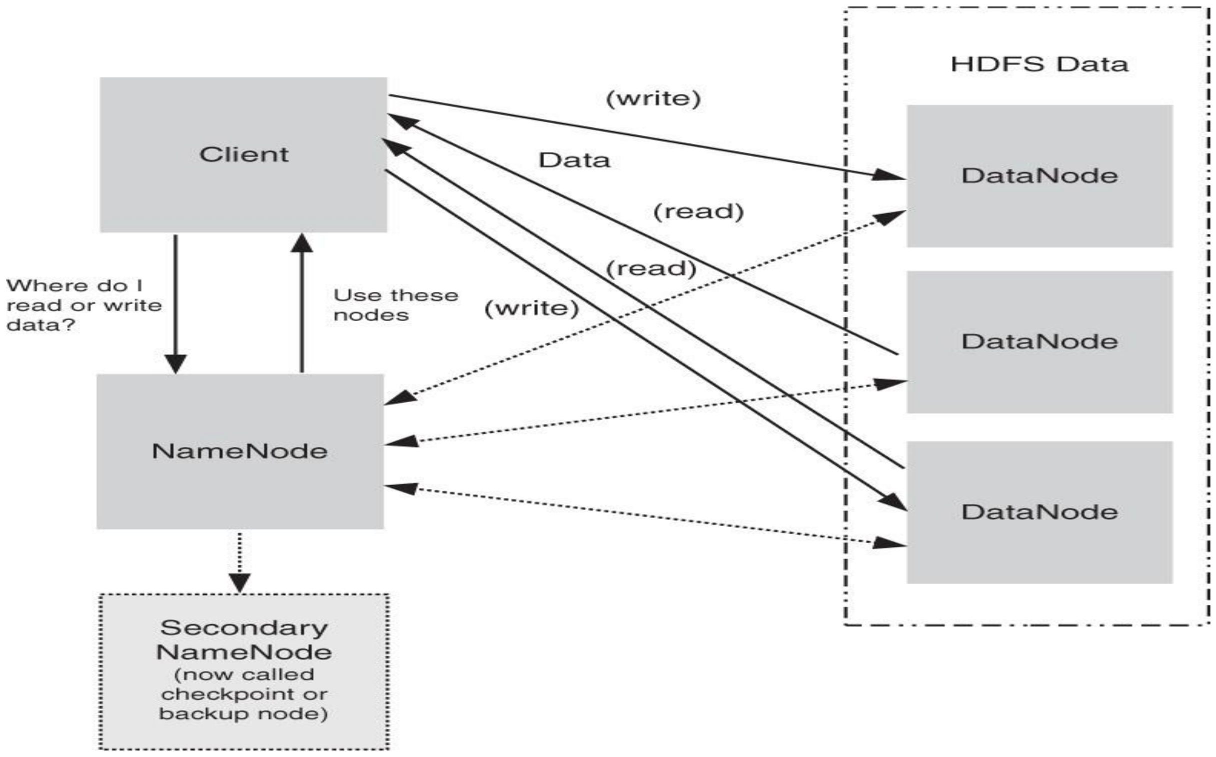


Figure 2.4 The client, master NameNode, MasterNodes and slave nodes

Q 2. Explain read and write operation on Hadoop.



- When a client writes data, it first communicates with the NameNode and requests to create a file. The NameNode determines how many blocks are needed and provides the client with the

DataNodes that will store the data.

- Supports Replication
- After the DataNode acknowledges that the file block replication is complete, the client closes the file and informs the NameNode that the operation is complete. Note that the NameNode does not write any data directly to the DataNodes. It does, however, give the client a limited amount of time to complete the operation. If it does not complete in the time period, the operation is canceled.
- Reading data happens in a similar fashion. The client requests a file from the NameNode, which returns the best DataNodes from which to read the data. The client then accesses the data directly from the DataNodes.
- Thus, once the metadata has been delivered to the client, the NameNode steps back and lets the conversation between the client and the DataNodes proceed. While data transfer is progressing, the NameNode also monitors the DataNodes by listening for heartbeats sent from DataNodes.
- The lack of a heartbeat signal indicates a potential node failure. NameNode finds the alternative blocks.
- The mappings between data blocks and the physical DataNodes are not kept in persistent storage on the NameNode. For performance reasons, the NameNode stores all metadata in memory. Upon startup, each DataNode provides a block report (which it keeps in persistent storage) to the NameNode. The block reports are sent every 10 heartbeats.

Q3. Explain Graph databases with characteristics, use and limitations.

- A characteristic of graph is high flexibility.
- Any number of nodes and any number of edges can be added to expand a graph.
- The complexity is high and the performance is variable with scalability.
- Data store as series of interconnected nodes.
- Graph with data nodes interconnected provides one of the best database system when

relationships and relationship types have critical values.

- Data Store focuses on modeling *interconnected* structure of data. Data stores based on graph theory relation G = (E, V), where E is set of edges e_1, e_2,…. and V is set of vertices, v_1, v_2, ... , vn·
- Nodes represent entities or objects. Edges encode relationships between nodes. Some operations become simpler to perform using graph models.
- Examples of graph model usages are social networks of connected people. The connections to related persons become easier to model when using the graph model.
- The yearly sales compute by path traversals from nodes for weekly sales to yearly sales data.

- The path traversals exhibit BASE properties because during the intermediate paths, consistency is not maintained. Eventually when all the path traversals complete, the data becomes consistent.

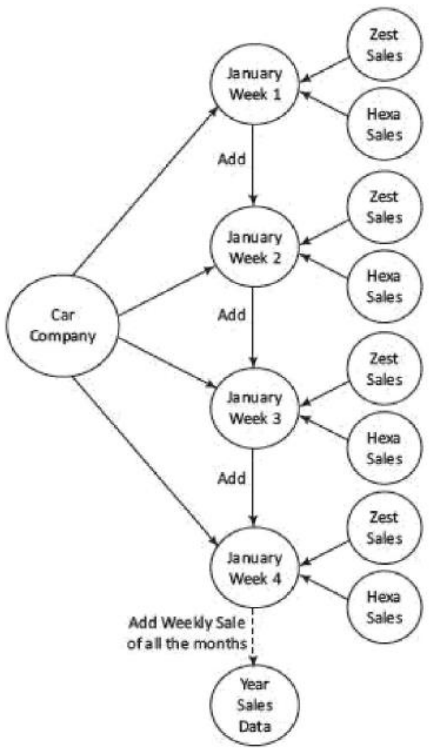


Figure 3.8 Section of the graph database for car-model sales

- Graph databases enable fast network searches. Graph uses linked datasets, such as social media data.
- Data store uses graphs with nodes and edges connecting each other through relations, associations and properties.
- Querying for data uses graph traversal along the paths. Traversal may use single-step, path expressions or full recursion.
- A relationship represents key. A node possesses property including ID. An edge may have a label which may specify a role.
- The task of adding relations in graph database is simpler. The nodes assign internal identifiers to the nodes and use these identifiers to join the network.
- Traversing the joins or relationships is fast in graph databases. It is due to the simpler form of graph nodes. The graph data may be kept in RAM only.

Q4. Explain working of Flume agent. Explain importance AVRO in pipelining Flume.

- Apache Flume is an independent agent designed to collect, transport, and store data into HDFS.
- Often data transport involves a number of Flume agents that may traverse a series of machines and locations. Flume is often used for log files, social media-generated data, email messages, and just about any continuous data source.

- Flume agent is composed of three components.

✓ ***Source***. The source component receives data and sends it to a channel. It can send the data to more than one channel. The input data can be from a real-time source (e.g., weblog) or another Flume agent.

✓ ***Channel.*** A channel is a data queue that forwards the source data to the sink destination. It can be thought of as a buffer that manages input (source) and output (sink) flow rates.

✓ ***Sink.*** The sink delivers data to destination such as HDFS, a local file, or another Flume agent.

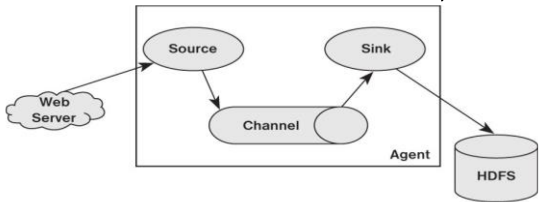


Figure 7.3 Flume agent with source, channel, and sink

A Flume agent can have several sources, channels, and sinks. Sources can write to multiple channels, but a sink can take data from only a single channel. Data written to a channel remain in the channel until a sink removes the data. By default, the data in a channel are kept in memory but may be optionally stored on disk to prevent data loss in the event of a network failure.

As shown in Figure 7.4, Sqoop agents may be placed in a pipeline, possibly to traverse several machines or domains. This configuration is normally used when data are collected on one machine (e.g., a web server) and sent to another machine that has access to HDFS.

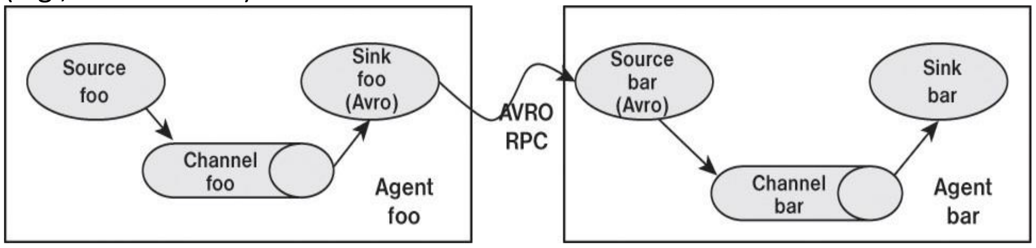


Figure 7.4 Pipeline created by connecting Flume agents

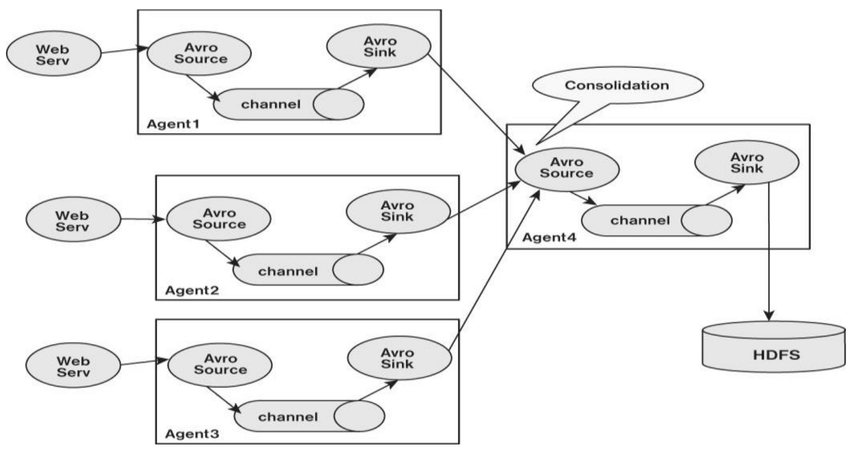
In a Flume pipeline, the sink from one agent is connected to the source of another. The data transfer format normally used by Flume, which is called Apache Avro, provides several useful features.

1) Avro is a data serialization/deserialization system that uses a compact binary format.

2) The schema is sent as part of the data exchange and is defined using JSON

3) Avro also uses remote procedure calls (RPCs) to
send data. That is, an Avro sink will contact an Avro source to send data.

Another useful Flume configuration is shown in Figure 7.5. In this configuration, Flume is used to consolidate several data sources before committing them to HDFS.



Q5. Explain Oozie features and its workflow in detail.

- Oozie is a workflow director system designed to run and manage multiple related Apache Hadoop jobs.
- Oozie is designed to construct and manage these workflows. Oozie is not a substitute for the YARN scheduler.
- YARN manages resources for individual Hadoop jobs, and Oozie provides a way to connect and control Hadoop jobs on the cluster.
- Oozie workflow jobs are represented as directed acyclic graphs (DAGs) of actions. Three types of Oozie jobs are permitted:

➢ Workflow—a specified sequence of Hadoop jobs with outcome-based decision points and control dependency. Progress from one action to another cannot happen until the first action is complete.

➢ Coordinator—a scheduled workflow job that can run at various time intervals or when data become available.

➢ Bundle—a higher-level Oozie abstraction that will batch a set of coordinator jobs.

➢ Oozie is integrated with the rest of the Hadoop stack, supporting several types of Hadoop jobs out of the box

Figure 7.6 depicts a simple Oozie workflow. In this case, Oozie runs a basic MapReduce operation. If the application was successful, the job ends; if an error occurred, the job is killed. Such workflows contain several types of nodes:

Control flow nodes define the beginning and the end of a workflow. They include start, end, and optional fail nodes.

Action nodes are where the actual processing tasks are defined. When an action node finishes, the remote systems notify Oozie and the next node in the workflow is executed. Action nodes can also include HDFS commands.

Fork/join nodes enable parallel execution of tasks in the workflow. The fork node enables two or more tasks to run at the same time. A join node represents a rendezvous point that must wait until all forked tasks complete.

Control flow nodes enable decisions to be made about the previous task. Control decisions are based on the results of the previous action (e.g., file size or file existence). Decision nodes are essentially switch-case statements that use JSP EL (Java Server Pages—Expression Language) that evaluate to either true or false.

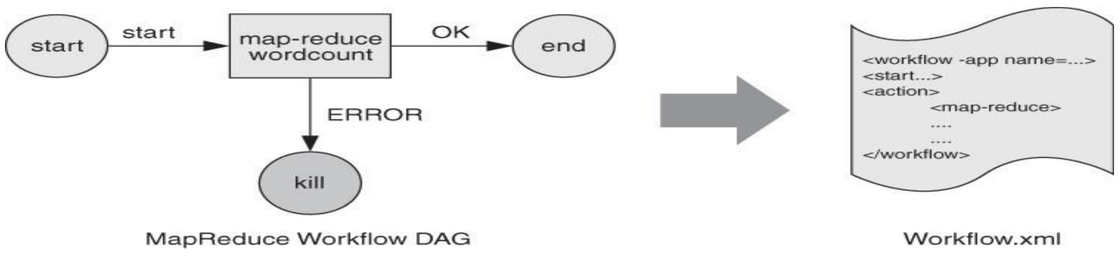
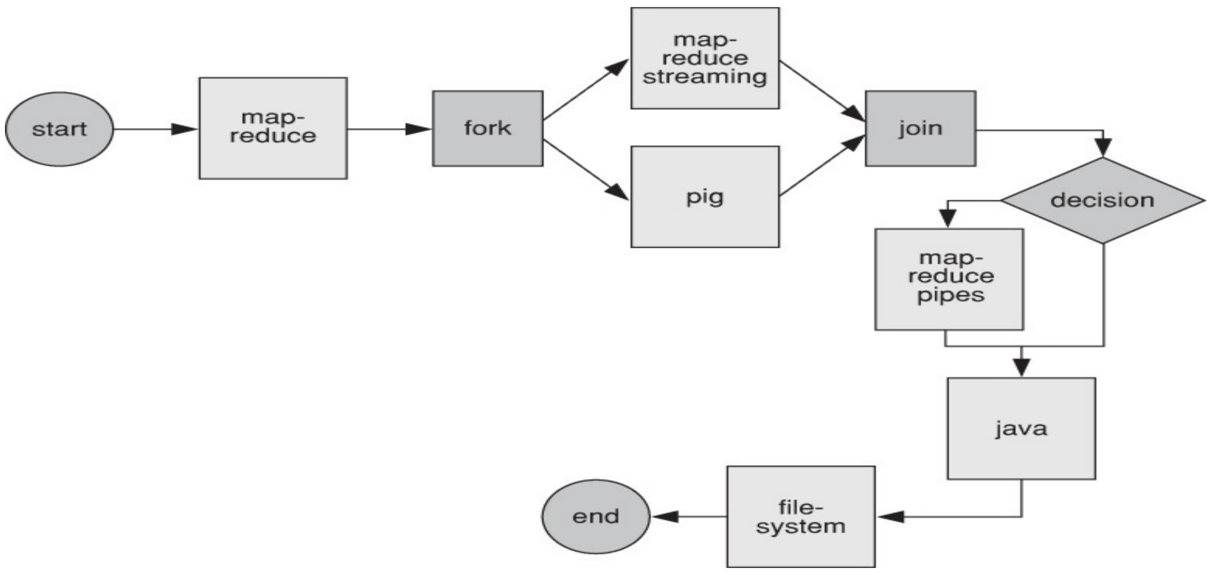


Figure 7.6 A simple Oozie DAG workflow

Figure 7.7 depicts a more complex workflow that uses all of these node types



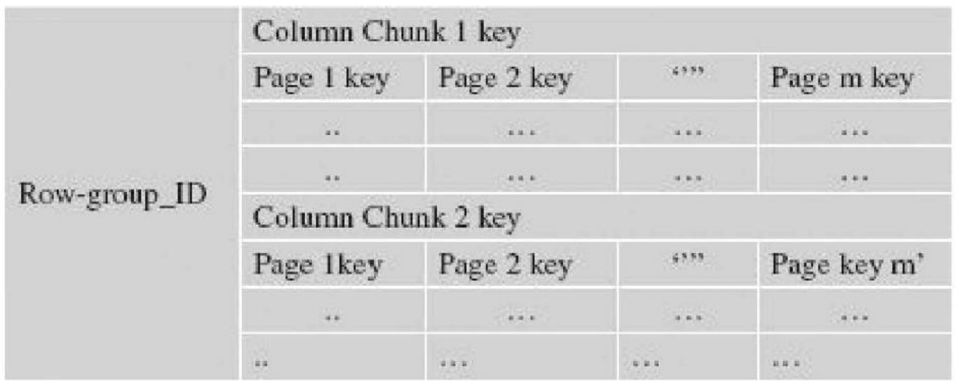
Q6. Explain NoSQL data store structures i) Parquet File Formats ii) ORC file format

Parquet File Formats

- Parquet is nested hierarchical columnar-storage concept.
- Nesting can be applied to sequence is the table, row group, column chunk and chunk page.
- Apache Parquet file is columnar-family store file.
- Apache Spark SQL executes user defined functions (UDFs) which query the Parquet file columns.
- A programmer writes the codes for an UDF and creates the processing function for big long queries.
- A Parquet file uses an HDFS block. The block stores the file for processing queries on Big Data.
- The file must consists of metadata, though the file need not consist of data.
- The Parquet file consists of row groups. A row-group columns data process in-memory.

- Each row group has a number of columns. Each column chunk consists of values saved in each column of each row group.
- A column chunk can be divided into pages and thus, consists of one or more pages. The column chunk consists of a number of interleaved pages, Npg
- A page is a conceptualized unit which can be compressed or encoded together at an instance.
- The unit is minimum portion of a chunk which is read at an instance for in-memory analytics.
- An ORC array <int> has two columns, one for array size and the other for contents.
- ORC has two columns, one is for each Map, List size, min, max and the second is for the contents.
- Parquet format file does not consist of extra column per nesting level, just one column per leaf in the schema.
- Table 3.6 gives the keys used to access or skip the contents page. Three keys are: (i) row-group_ID, (ii) column-chunk key and (iii) page key.

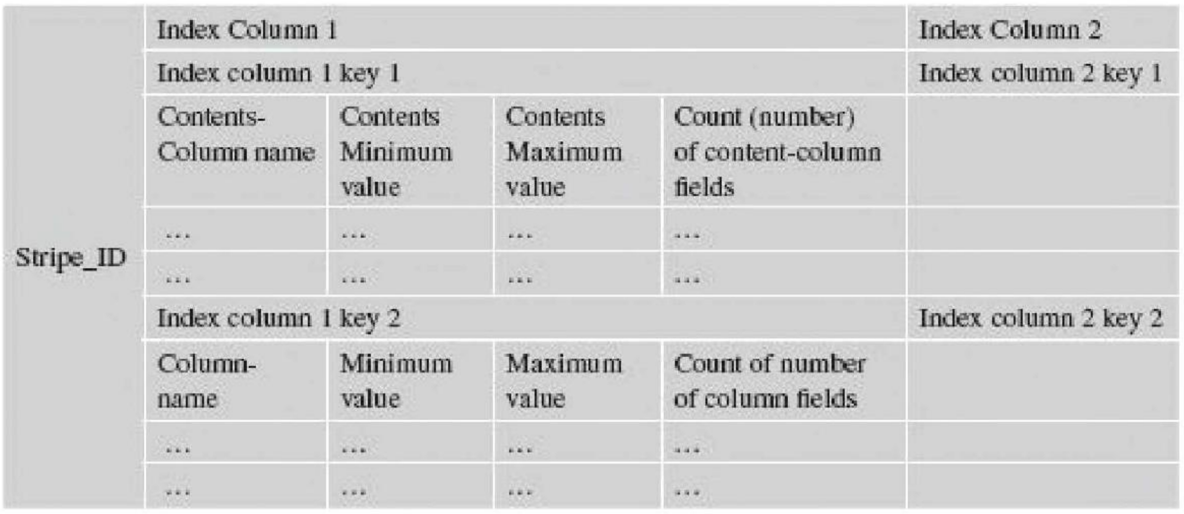
Table 3.6 Combination of keys for content page in the Parquet file format



Row-group_ID	Column Chunk 1 key			
	Page 1 key	Page 2 key	‘’’	Page m key
	..	...	...	...
	..	...	...	...
	Column Chunk 2 key			
	Page 1key	Page 2 key	‘’’	Page key m’
	..	...	...	...
	..	...	...	...

- **An ORC (Optimized Row Columnar)** file consists of row-group data called stripes.
- ORC enables concurrent reads of the same file using separate RecordReaders. Metadata store uses Protocol Buffers for addition and removal of fields.
- ORC is an intelligent Big Data file format for HDFS and Hive.
- An ORC file stores a collections of rows as a row-group. Each row-group data store in columnar format. This enables parallel processing of multiple row-groups in an HDFS cluster.
- An ORC file consists of a stripe the size of the file is by default 256 MB.
- Stripe consists of indexing (mapping) data in 8 columns, row-group columns data (contents) and stripe footer (metadata).

- An ORC has two sets of columns data instead of one column data in RC. One column is for each map or list size and other values which enable a query to decide skipping or reading of the mapped columns.
- A mapped column has contents required by the query. The columnar layout in each ORC file thus, optimizes for compression and enables skipping of data in columns. This reduces read and decompression load.
- Lightweight indexing is an ORC feature.
- Each index includes the aggregated values of minimum, maximum, sum and count using aggregation functions on the content columns.
- Therefore, contents column key for accessing the contents from a column consists of combination of row-group key, column mapping key, min, max, count (number) of column fields of the contents column.
- Table 3.5 gives the keys used to access or skip a contents column during querying. The keys are Stripe_ID, Index-column key, and contents-column name, min, max and count.
- The throughput increases due to skipping and reading of the required fields at contents-column key. Reading less number of ORC file content-columns reduces the workload on the NameNode.



Stripe_ID	Index Column 1				Index Column 2
	Index column 1 key 1				Index column 2 key 1
	Contents-Column name	Contents Minimum value	Contents Maximum value	Count (number) of content-column fields	
	...	...	...	...	
	...	...	...	...	
	Index column 1 key 2				Index column 2 key 2
	Column-name	Minimum value	Maximum value	Count of number of column fields	
	...	...	...	...	
	...	...	...	...	

Q7. Explain Mongo DB features, replication and autosharding process

- **MongoDB** is

(i) non-relational,
(ii) NoSQL,
(iii) distributed,
(iv) open source,
(v) document based,
(vi) cross-platform,
(vii) Scalable,
(viii) flexible data model,
(ix) Indexed,

(x) multi-master and

(xi) fault tolerant. Document data store in JSON-like documents. The data store uses the dynamic schemas.

Features:

- The typical MongoDB applications **are content management and delivery systems, mobile applications, user data management, gaming, e-commerce, analytics, archiving and logging**.

1. MongoDB data store is a physical container for collections. Each DB gets its own set of files on the file system. A number of DBs can run on a single MongoDB server. The database server of MongoDB is mongod and the client is mongo.
2. Collection stores a number of MongoDB documents. It is analogous to a table of RDBMS. A collection exists within a single DB to achieve a single purpose. Documents of the collection are schema-less. Thus, it is possible to store documents of varying structures in a collection.
3. Document model is well defined. Document is the unit of storing data in a MongoDB database. Insert, update and delete operations can be performed on a collection. Documents have dynamic schema.
4. MongoDB is a document data store in which one collection holds different documents. Number of fields, content and size of the document can differ from one document to another.
5. Storing of data is flexible, and data store consists of JSON-like documents. This implies that the fields can vary from document to document and data structure can be changed over time; JSON has a standard structure, and scalable way of describing hierarchical data.
6. Storing of documents on disk is in BSON serialization format. BSON is a binary representation of JSON documents. The mongo JavaScript shell and MongoDB language drivers perform translation between BSON and language-specific document representation.
7. Querying, indexing, and real time aggregation allows accessing and analyzing the data efficiently.
8. Deep query-ability-Supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.
9. No complex Joins.
10. Distributed DB makes availability high, and provides horizontal scalability.
11. Indexes on any field in a collection of documents: Users can create indexes on any field in a document. Indices support queries and operations. By default, MongoDB creates an index on the _id field of every collection.
12. Atomic operations on a single document can be performed even though support of multi-document transactions is not present. The operations are alternate to ACID transaction requirement of a relational DB.
13. Fast-in-place updates: The DB does not have to allocate new memory location and write a full new copy of the object in case of data updates. This results into high performance for frequent update use cases. For example, incrementing a counter operation does not fetch the document from the server. Here, the increment operation can simply be set.
14. No configurable cache: MongoDB uses all free memory on the system automatically by way of memory-mapped files. The most recently used data is kept in RAM. If indexes are created for queries and the working dataset fits in RAM, MongoDB serves all queries from memory.
15. Conversion/mapping of application objects to data store objects not needed