# **Segment 1**

Module 6: HBase

Session 1: HBase

Platform Text

In the previous module, you learnt how PIG simplifies the implementation (programming) of some operations compared to the MapReduce. In this module, you will learn about the HBase - NoSQL database component of Hadoop Ecosystem. Some of the limitations of Hadoop like only confined to batch processing of data and unable to perform random lookups on data and unable to update the existing data laid the foundation for HBase.

## **Guidelines for in-module questions**

The in-video and in-content questions for this module are not graded. Note that graded questions are given on a separate page labelled 'Graded Questions' at the end of this session. These graded questions will adhere to the following guidelines:

|  |  |  |
| --- | --- | --- |
|  | First Attempt Marks | Second Attempt Marks |
| Question with 1 Attempt | 10 | 5 |

## 

## **People you will hear from in this module**

**Subject Matter Expert**

Prof. Chittaranjan Hota

Professor & Associate Dean, BITS Hyderabad

Prof. Chittaranjan Hota is a senior professor of Computer Science and Engineering at BITS Hyderabad. He was the founding head of the Computer Science and Engineering department (2008-2012) and is currently the Associate Dean (Admissions) at the institute. At BITS Pilani, Prof. Hota has conducted courses on computer networks, operating systems, distributed computing, systems programming, big data systems, artificial intelligence, and more. He is the professor in charge of BITSAT (university-wide) at BITS Pilani and has also been administering the IT division at BITS Hyderabad, as its unit chief for the past nine years.

**Industry Expert**

Gowtham Ram

Gowtham Ram is a Big Data Leader working at a leading investment banking firm. He leads the delivery of data warehousing, big data, and data science projects and has more than 13 years of experience in this industry. Ram has also completed his MBA in Analytics from IIM Bangalore.

**Presenter**

Sandeep Thilakan

# **Segment 2**

# 

# **HBase**

HBase

Platform Text

## **Introduction: HBase**

Welcome to the session on ‘HBase’.

## **In this session**

This session will cover a few basic concepts related to HBase - a distributed NoSQL database of Hadoop, which allows performing random lookups on data. HBase also allows updating the existing data by maintaining multiple versions of data which was not the case with HDFS.

The topics that will be covered in this session are as follows:

* Introduction to NoSQL databases
* CAP Theorem
* Introduction to HBase
* Data Model of HBase
* Perform Operations in HBase: Shell commands and Java API

# **Segment 3**

# **Introduction to NoSQL Database**

Introduction to NoSQL Database

Before looking into HBase, let’s look into what a NoSQL database is and why it is needed.

Video and in-video questions

<video 1 appears>

Platform Text

Though Hadoop overcame some of the challenges faced by SQL, it introduced new challenges. MapReduce of Hadoop (used for processing the data stored in HDFS) is well suited for batch processing, where the whole data is accessed sequentially. But the drawback of using MapReduce for processing big data is that it is not suited for all use cases like performing random lookups on data.. Hence, apart from Hadoop, there was a need for some solution for these use cases. These Limitations of Hadoop led to the inception of NoSQL datastores.

Some of the reasons for increasing popularity of NoSQL databases discussed in the video are:

* NoSQL datastores are efficient in storing and handling Big Data. Based on the targeted use cases, every NoSQL database has its data model for storing data.
* NoSQL data stores provide scalability, i.e. in case of space crunch extra space can be easily created by just adding additional nodes to the cluster.
* NoSQL data stores are flexible and do not restrict themselves to a fixed schema. Hence, NoSQL data stores can adapt to changes in the schema of data dynamically.

Out-of-Video Questions

Q1. Let’s consider an RDBMS table that has the following schema:

|  |  |
| --- | --- |
| **Column Name** | **Data Type** |
| Roll No | NUMBER (Used to represent numbers) |
| NAME | VARCHAR2 (Same as string in Java) |
| Age | NUMBER (Used to represent numbers) |
| Year | NUMBER (Used to represent numbers) |
| Department | VARCHAR2 (Same as string in Java) |
| Sex | VARCHAR2 (Same as string in Java) |

In SQL, the INSERT command is used for inserting data into RDBMS tables. Consider the syntax of the INSERT command given below and select the correct Insert command that will execute without giving any errors.

INSERT INTO <TABLE\_NAME> VALUES (‘STRING’, NUMBER ….).

Anything enclosed within single quotes is considered as VARCHAR2. Let’s assume that there is no implicit type conversions when inserting data into a relational database.

1. INSERT INTO <TABLE\_NAME> VALUES (‘20’, ‘Siddharth’, 28, 2, ‘CSE’, ‘M’ )
2. **INSERT INTO <TABLE\_NAME> VALUES (20, ‘Siddharth’, 28, 2, ‘3’, ‘M’ )**
3. INSERT INTO <TABLE\_NAME> VALUES (20, ‘Siddharth’, 28, 2, ‘CSE’, ‘M’, 50.5 )
4. INSERT INTO <TABLE\_NAME> VALUES (‘20’, ‘Siddharth’, ‘28’, ‘2’, ‘3’, ‘M’ )

Ans b

* Option a is incorrect. The values to be inserted into the table do not adhere to the schema. The first field is ‘Roll No’ which is of type ‘Number’, whereas in this option, we are treating the field as a String as it is enclosed within single quotes.
* **Option b is correct. All the values adhere to the schema of the table.**
* Option c is incorrect. The values to be inserted into the table do not adhere to the schema. This command is trying to insert seven values, whereas the table has only six columns.
* Option d is incorrect. The values to be inserted into the table do not adhere to the schema. This command is trying to insert six values, all enclosed within single quotes. Numeric data values should not be enclosed within quotes.

Q2. Which data format can be stored and processed efficiently using RDBMSs?

1. Image
2. Video
3. **Data arranged in rows and columns**
4. Log files

Ans c.

* Option a is incorrect. Image files contain unstructured data, and RDBMS is not designed for storing images and processing them.
* Option b is incorrect. Video files contain unstructured data, and RDBMS is not designed for storing videos and processing them.
* **Option c is correct. Data arranged in rows and columns represent structure, and RDBMS is well-suited for storing data in a tabular format and processing them.**
* Option d is incorrect. Text files contain unstructured data, and RDBMS is not designed for storing text files in their original format and then processing them.

Q3) State whether the following statement is true or false:  
“NoSQL was designed because SQL systems were incompetent in processing transactional data.”

1. True
2. **False**

The ans is False because SQL systems are designed to process transactional data. But, NoSQL databases were designed to accommodate high volumes of unstructured data in distributed storage.

Q4) Select the most suitable use case where NoSQL should be used:

1. When the data security is of the utmost priority
2. When data is arranged in rows and columns
3. **When data storage in the range of TB is required**
4. When the retrieval speed of data is not critical

Ans c

* Option a is incorrect. NoSQL databases are not used specifically to provide high security to the stored data.
* Option b is incorrect. NoSQL databases are not specifically designed to store structured data.
* **Option c is correct. NoSQL databases are designed to store high volumes of data in the range of TB distributed across clusters of machines.**
* Option d is incorrect. NoSQL databases are designed to provide quick access to data.

Q5) Select the option that is not valid:

1. SQL databases make use of a fixed schema, whereas NoSQL datastores do not make use of a fixed schema
2. **The query language used for querying NoSQL datastores adds a layer of abstraction on top of MapReduce. They convert the query into a series of MapReduce jobs.**
3. NoSQL refers to “Not Only SQL”
4. MapReduce scans the entire dataset for any query, whereas NoSQL has the ability to fetch a single record without scanning the entire dataset

* Option a is incorrect. It is a valid statement. SQL databases indeed make use of a fixed schema. On the other hand, NoSQL datastores make use of dynamic schemas, which can be modified on the fly.
* **Option b is correct. This statement is False. The queries are not converted to MapReduce because NoSQL datastores are designed to perform low latency data lookups.**
* Option c is incorrect. It is a valid statement. NoSQL datastores are indeed called as “Not only SQL”.
* Option d is incorrect. It is a valid statement. MapReduce scans the entire dataset. NoSQL datastores are designed to perform low latency data lookups.

**Segment 4**

# **CAP Theorem**

## **CAP Theorem**

Platform Text

Now that you have a brief idea about the need for NoSQL database, let’ us look into a specific theorem known as the CAP theorem which clearly states the trade-offs in the design of networked shared-data systems.

Video and in-video questions

<video 2 appears>

Platform Text

The CAP(Consistency, Availability, Partition tolerance) theorem is a significant result related to distributed systems. This theorem was first published as the CAP principle in 1999 and later presented as a conjecture by Eric Brewer at the 2000 symposium on Principles of Distributed Computing (PODC). In 2002, Seth Gilbert and Nancy Lynch of MIT published a formal proof of Brewer’s conjecture. The CAP theorem states that it is impossible for a distributed data store to simultaneously provide more than two out of the three guarantees - Consistency, Availability and Partition tolerance. Some of the known systems and the guarantees provided by them are mentioned below:

* **RDBMS Systems (support SQL):** Consistency and Availability.
* **NoSQL Systems:** They store data in a distributed manner across a cluster of interconnected machines and provide network partitioning. Hence, there are two flavours of NoSQL databases which provide a different set of guarantees:

1) Consistency and Partition tolerance(Ex - HBase, MongoDB).

2) Availability and Partition tolerance(Ex - Cassandra, DynamoDB).

Depending upon the type of application, while designing networked shared-data systems, the system designers should make the trade-off between consistency, availability and partition tolerance accordingly.

## **Additional Reading:**

[CAP Theorem](https://dzone.com/articles/understanding-the-cap-theorem)

Out-of-video Questions

Q1) Which of the following is **not** a reason why NoSQL has become a popular solution in some organisations?

1. Better scalability
2. Faster data lookup
3. Allows data to be stored across multiple nodes
4. **Ability to store consistent data eternally**

* Option a is incorrect. NoSQL datastores provide scalability by various mechanism such as horizontal scalability and sharding.
* Option b is incorrect. Some NoSQL datastores such as HBase provide low latency data lookup.
* Option c is incorrect. NoSQL datastores are designed to store data across multiple nodes.
* **Option d is correct. NoSQL datastores do not guarantee consistency all the time. In case of network partition, some datastores may not return the recent value.**

Q2) Which of the following use cases will give priority to consistency over availability?

1. Prices of air tickets on a travel portal
2. Notifications on facebook
3. **A customer’s bank account details that includes all latest transactions**
4. None of these

* Option a is incorrect. The prices of the air tickets on a travel portal may not be the latest at the time when we are checking. Sometimes, the prices are updated just before the payment. Here, availability is given a higher priority over consistency.
* Option b is incorrect. Delaying of notifications is completely fine. But it is not acceptable if facebook.com goes down because of the delay in notifications. Here, availability is given a higher priority over consistency.
* **Option c is correct. Bank account details are sensitive information and they have to be consistent and correct. So, if the latest accurate information is not available, then its better to declare an outage than showing wrong information to the customer. Sometimes there is a delay in updating the latest transactions, those delays are due to regulations in the financial system and that the transactions get immediately accounted under unbilled/uncleared funds.**
* Option d is incorrect. Go through the use cases again. One of the remaining options is the correct answer.

Q3) Pick the correct statement regarding CAP theorem?

1. A in CAP refer to Accuracy
2. **If partition tolerance is not considered then the datastore is an RDBMS system**
3. Consistency guarantees that every node in the distributed system must have identical configuration
4. None of these

* Option a is incorrect. A in CAP refers to availability
* **Option b is correct. RDBMS systems are not used to process data using distributed computing. Hence, partition tolerance is out of context for RDBMS systems**
* Option c is incorrect. The definition of consistency mentioned in this option is incorrect. Consistency means every node in the distributed system returns the same, most successful, recent write.
* Option d is incorrect. Please go through the use cases again, one of the remaining options is the correct answer.

Q4) There is a need of storing transactional data generated by a Bank’s ATM. The data is to be stored in tabular format. According to CAP theorem, which type of datastore will be used for this:

1. **CA**
2. CP
3. AP
4. None of these

* **Option a is correct. RDBMS systems will be required to store transactional data in structured format. As per CAP theorem, RDBMS systems support Consistency and Availability.**
* Option b is incorrect. RDBMS systems will be required to store transactional data in structured format. As per CAP theorem, RDBMS systems do not support Consistency and Partition Tolerance.
* Option c is incorrect. RDBMS systems will be required to store transactional data in structured format. As per CAP theorem, RDBMS systems do not support Availability and Partition Tolerance together.
* Option d is incorrect. Please go through the use cases again, one of the remaining options is the correct answer.

Q5) For a social networking website, there is a need for storing users and their friends’ data in a data store. Let’s assume the data is stored in a distributed storage system. Here, due to system outage, if the latest total count of the friends for a user is not available the portal can still be available and reflect older stats. According to CAP theorem, which type of datastore can be used:

1. CA
2. CP
3. **AP**
4. None of these

* Option a is incorrect. This answer is incorrect as partition tolerance is not been considered.
* Option b is incorrect. This answer is incorrect as consistency is given priority over availability.
* Option c is correct. This answer is correct because the system needs to be highly available in case of a partition tolerance.
* Option d is incorrect. Please go through the use cases again, one of the remaining options is the correct answer.

# 

# 

# **Segment 5**

# **Introduction to HBase**

## **Introduction to HBase**

Platform Text

Now that you have learnt about NoSQL databases and the guarantees they provide, let’s take a look at HBase and its features.

Video and in-video questions

<Video 3 appears>

Platform Text

HBase is a distributed data store built on top of HDFS and can leverage all the benefits provided by Hadoop or HDFS. HBase has the ability which allows a user to query for individual records as well as derive aggregate analytic reports across a massive amount of data. HBase was first released as a code for an open source BigTable implementation.

Some of the prominent features of HBase that distinguishes it from Hadoop and other relational or NoSQL data stores discussed in the video are:

* HBase stores data internally as key-value pairs where the rowkey is the key and the rest of the data is value.
* HBase records are sorted by RowKey.
* HBase Columns do not have any specific data type and all the data in HBase is stored in the form of bytes.
* HBase do not follow a strict schema (Schema-less and dynamic architecture) which means any number of columns can be added dynamically.

Out-of-video Questions

**Q1)** Which among the following is an advantage of HBASE over HDFS??

1. **Low latency retrieval of data**
2. Distributed storage
3. Horizontal scalability
4. Fault tolerance

* **This option is correct. HDFS does not provide low latency retrieval data. MapReduce is used to read data directly from HDFS in a sequential manner.**
* This option is incorrect. HDFS provides distributed storage.
* This option is incorrect. HDFS is horizontally scalable. Capacity of HDFS can be increased by simple adding extra nodes to the cluster.
* This option is incorrect. HDFS is fault tolerant and ensures there is minimal data loss by replication

Q2) Pick the correct tabular representation of HBase. In the give tables, RowId is the RowKey.

|  |  |  |
| --- | --- | --- |
| **RowID** | **Field1** | **Field2** |
| **1** | A | 10 |
| **3** | B | 20 |
| **2** | C | 30 |
| **5** | D | 40 |
| **9** | E | 50 |

B.

|  |  |
| --- | --- |
| **Field1** | **Field2** |
| A | 10 |
| B | 20 |
| C | 30 |
| D | 40 |
| E | 50 |

C.

|  |  |  |
| --- | --- | --- |
| **RowID** | **Field1** | **Field2** |
| **1** | A | 10 |
| **2** | B | 20 |
| **3** | C | 30 |
| **5** | D | 40 |
| **9** | E | 50 |

D. None of these

* Option ‘a’ is incorrect. Because the rows are not sorted according to row id.
* Option ‘b’ is incorrect. The table does not have the row id column which is impossible in HBase
* **Option ‘c’ is correct. The rows in the given table are sorted as per the rowkey i.e rowId**
* Option ‘d’ is incorrect. Please check the options again. One of the options is correct.

**Segment 6**

# **Data Model of HBase**

# **Data Model of HBase**

Platform Text

Now that you have learnt about the features of the HBase, let’ us now take a look at how data is stored in HBase.

Video

<Video 4 appears>

Platform Text

Let’s recap the data model of HBase discussed in the video:

* In HBase, data is stored in tables which are nothing but a collection of rows. In an HBase table, a row is a collection of column families. A column family is a collection of related columns known as column qualifiers, and there can be any number of columns in a single column family.
* Every entry in an HBase table is identified and indexed by a RowKey and for every RowKey, an unlimited number of columns can be stored. This feature ensures that the schema of an HBase table is flexible and the table can scale linearly.
* Each column can have a configurable number of versions and there is a provision for selecting data from a particular version. In HBase, each version is identified by a timestamp and each column has one or more versions.

Out-of-video Questions

Q1) Pick the option which is **not** true with respect to column families?

1. Column families comprises of more than one columns
2. **The number of columns present in the column family is fixed**
3. Column families allows the table to scale linearly
4. None of these

* Option ‘a’ is incorrect. This statement is true. Column families are collection of columns in HBase table.
* **Option ‘b’ is correct. This statement is incorrect. The number of columns in column families is not fixed. This gives flexibility to HBase table.**
* Option ‘c’ is incorrect. This statement is true. Column families allows the tables to scale linearly. This feature ensures that HBase does not
* Option ‘d’ is incorrect. Please check the options again. One of the options is correct.

Q2) Pick the invalid statement(s). (Multiple statements may be invalid) ?

1. Rows are sorted by rowkey in HBase whereas in RDBMS rows are not sorted by any keys
2. HBase tables have column families which comprises of multiple columns whereas RDBMS tables do not have the concept of column families
3. **HBase tables follow a strict schema whereas RDBMS does not follow a strict schema and is flexible**
4. HBase tables are known as NoSQL databases whereas RDBMS are known as SQL databases

* Option ‘a’ is incorrect. Rows are sorted by rowkey in HBase whereas in RDBMS rows are not stored in any specific order.
* Option ‘b’ is incorrect. HBase tables comprises of column families which are a collection of columns. RDBMS tables do not have anything called column families.
* **Optionc ‘c’ is correct. HBase tables do not follow a strict schema where as RDBMS tables follow a well defined schema**
* Option ‘d’ is incorrect. HBase tables are known as NoSQL databases whereas RDBMS tables are known as SQL databases

Let’s assume the syntax for retrieving data present in a single cell of a HBase table:

get ‘<table name>’, ‘rowkey’, ‘<column name>’, ‘version’

The column name is represented as column family : column name

In the get command, table name, row key and column name are the mandatory parameters. The version field is optional. If you don’t provide any version in the command then by default latest value is picked.

Let’s consider a sample table below whose name is ‘Employee’. This table can record a maximum of two versions for each column. Version “V1” refers to the latest information. The column families are “Personal Details” and “Professional Details”. The columns in column family Personal Details are “Name” and “Age”. The columns in column family Professional Details are “Company Name” and “Designation”:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ROWID** | **Personal Details** | | **Professional Details** | | | |
|  | **Name** | **Age** | **Company Name** | | **Designation** | |
|  | **V1** | **V1** | **V1** | **V2** | **V1** | **V2** |
| **1** | Siddharth | 27 | Wipro | Infosys | Software Engineer | Systems Engineer |
| **2** | Sandeep | 27 | ITC Infotech | Accenture | Senior Business Analyst | Business Analyst |
| **3** | Jayadeep | 22 | Symantec |  | Intern |  |
| **4** | Onkar | 28 | Wipro | ITC Infotech | Senior Project Manager | Project Manager |

Please note that blank cells are used if there is no entry, i.e. for Rowid 3, no value has been recorded in column ‘Professional Details: Company Name’ under version V2.

Answer the questions based on the information provided above:

Q3)

What would be the correct get command to retrieve version V2 of company name for RowID 2.

1. get ‘Employee’, ‘2’, ‘Company Name’, ‘V2’
2. get ‘Employee’, ‘Company Name’, ‘2’, ‘V2’
3. **get ‘Employee’, ‘2’, ‘Professional Details:Company Name’, ‘V2’**
4. get ‘Employee’, ‘2’, ‘Professional Details:Company Name’, ‘V1’

* Option ‘a’ is incorrect. The column name is not preceded by the qualifier name.
* Option ‘b’ is incorrect. The column name is not preceded by the qualifier name and rowkey is written after column name which is incorrect. Please refer to the syntax for more clarity
* **Option ‘c’ is correct. The command is in accordance with the syntax mentioned above.**
* Option ‘d’ is incorrect. In the question it is asked to retrieve the data for V2 and not V1.

Q4)

What would be the output of the below mentioned command:

get ‘Employee’, ‘4’, ‘Professional Details:Designation’

1. Project Manager
2. **Senior Project Manager**
3. Command will not run and flag an error
4. Senior Business Analyst

* Option ‘a’ is incorrect. If the version is not mentioned in the query then by default latest value is picked. In this table the latest value is denoted by V1.
* **Option ‘b’ is correct. If the version is not mentioned in the query then by default latest value is picked. In this table the latest value is denoted by V1.**
* Option ‘c’ is incorrect. The command is correct. If the version is not mentioned in the query then by default latest value is picked. In this table the latest value is denoted by V1.
* Option ‘d’ is incorrect. Senior Business Analyst is the latest designation for Row Id 2.

Q5) What will be the output if the below mentioned command is executed on Hbase:

get ‘Employee’, ‘3’

1. Jayadeep
2. 22
3. **Command will not run and flag an error**
4. None of these

* Option ‘a’ is incorrect. This not the correct answer. Please read the entire problem description again to answer this question correctly.
* Option ‘b’ is incorrect. This not the correct answer. Please read the entire problem description again to answer this question correctly.
* **Option ‘c’ is correct. This command will not run as the mandatory field i.e. the column family followed by the column name is missing.**
* Option ‘d’ is incorrect. This not the correct answer. Please read the entire problem description again to answer this question correctly.

**Segment 7**

# **Perform Operations in HBase Using Shell Commands**

## **Perform Operations in HBase Using Shell Commands**

Platform Text

Now that you have learnt about how the data is stored in the HBase, let's perform the basic CRUD (Create, Read, Update and Delete) operations in HBase using shell commands.

**Note:**

1. The video shows how to perform basic CRUD operations in HBase using shell commands on a Cloudera VM running on localhost, i.e. a personal computer. If you have access to a machine hosting the Cloudera VM, it is highly recommended to use your local machine as shown in the video. The instructions for performing basic CRUD operations in HBase using the shell commands on a Cloudera VM are documented below.
2. If you are facing troubles running the Cloudera VM on your local machine, please follow the document "HBase basic operations using shell commands on AWS EC2" attached below the video. The document explains how to use the HBase shell commands on an Amazon EC2 instance with Cloudera deployed on it.

Video

<Video 5 appears>

## 

Platform Text

The shell commands of HBase demonstrated in the video are as follows:

* **List:** This command is used to check all the tables present in your HBase instance.

**Syntax:** hbase> list

* **Create:** This command is used to create a table. To create a table you must name the table and define its schema. As a part of the schema, you are required to specify the column families. However, columns are defined later while inserting records into the HBase table.

**Syntax:** hbase> Create ‘<table\_name>’, ‘<column\_family\_name>’

**Example:** Create a table named Students with three column families named Personal Details, Contact Details and Marks.

**Command:** Create ‘Students’, ‘Personal Details’, ‘Contact Details’, ‘Marks’

* **Scan:**  This command is used to view the contents of a table created. The optional parameters in syntax include TIMERANGE, FILTER, TIMESTAMP, LIMIT, MAXLENGTH, COLUMNS, CACHE, STARTROW and STOPROW.

**Syntax:** hbase> Scan ‘<table\_name>’ {Optional parameters}

* **Put:**  This command is used to insert the records into the table and can also be used to update the records in the table.

**Syntax:** hbase> put ‘<table\_name>’, ‘<row\_key>’, ‘<column\_value>’, ‘<value>’

**Example:** Insert a “Name” record (student name “Sandeep”) into the “Students” table in a row ‘students1’ and in the column ‘Personal Details’ that was created earlier. After insertion, update the earlier record by adding the “email id” of Student1.

**Command:** Put ‘Students’, ‘Student1’, ‘Personal Details:Name’, ‘Sandeep’

**Update Command:** Put ‘Students’, ‘Student1’, ‘Personal Details:Email’, ‘Sandeep@pqr.com’

**Note:** If we observe the “Put command”, we can understand that columns are defined while inserting records into the HBase table. (Column “Name” is defined while inserting the name “Sandeep”)

* **get:** This command is used to fetch data from HBase. There are various ways to make use of the get command.

**Syntax:** hbase> get ‘<table\_name>’, ‘<row\_key>’, {‘<Additional Parameters>’}

**Note: “**<Additional Parameters>” mentioned in the syntax include TIMERANGE, TIMESTAMP, VERSIONS and FILTERS.

**Example:** Fetch data from “Students” table created in three ways as mentioned below:

1. Fetch entire “Student1” row data
2. Fetch data from the column family “Personal Details”
3. Fetch data from the particular column “Name”

**Commands:**

1. get ‘Students’, ‘Student1’
2. get ‘Students’, ‘Student1’, {COLUMN => ‘Personal Details’}
3. get ‘Students’, ‘Student1’, {COLUMN => ‘Personal Details:Name’}

* **delete:**  This command is used to delete a cell in HBase table.

**Syntax:** hbase> delete ‘<table\_name>’, ‘<row\_key>’, ‘<column\_value>’, ‘<value>’

**Example:** Delete the “email id” updated to the “Student1” record..

**Command:** delete ‘Students’, ‘Student1’, ‘Personal Details:Email’

**Note:**  “scan” command helps to verify whether the contents of a table are deleted or not.

Out-of-video Questions

Q1) Pick the command which is not used in HBase?

1. get
2. put
3. **insert**
4. scan

* Option ‘a’ is incorrect. Get is a valid HBase command and is used to get data from a single cell in an HBase table
* Option ‘b’ is incorrect. Put is a valid HBase command and is used to enter data into an HBase table
* **Option ‘c’ is correct. Insert is an invalid command**
* Option ‘d’ is incorrect. scan is a valid HBase command and is used to display all the records present in HBase table.

Q2) Pick the correct command to create table ‘Employee’ having below mentioned structure:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Personal Details** | | **Professional Details** | | | |
| **Name** | **Age** | **Company Name** | | **Designation** | |

1. create ‘Employee’, ‘Personal Details:Name’, ‘Personal Details:Age’, ‘Professional Details:Company Name’, ‘Professional Details:Designation’
2. **create ‘Employee’, ‘Personal Details’, ‘Professional Details’**
3. create ‘Employee’, ‘RowKey’, ‘Personal Details’, ‘Professional Details’
4. None of these

* Option ‘a’ is incorrect. While creating a table it is not required to specify the columns along with their column families
* **Option ‘b’ is correct. For creating a table only the column families are specified in the create command.**
* Option ‘c’ is incorrect. This command will run without giving any errors but an extra column family will be created with the name as ‘RowKey’. As per the table structure provided, a column family named ‘RowKey’ is not required
* Option ‘d’ is incorrect. Please check the options again. One of the options is correct.

Q3) What will be the ouput if the following command is executed after creating Employee table and before inserting any data:

**scan ‘Employee’**

1. Command does not run and flags an error
2. Default values are returned
3. **Command runs without giving any errors**
4. None of these

* Option ‘a’ is incorrect. The command runs without giving any errors.
* Option ‘b’ is incorrect. HBase does not store default values in empty tables
* **Option ‘c’ is correct. Command runs without giving any errors. Zero rows are returned as the table is empty.**
* Option ‘d’ is incorrect. Please check the options again. One of the options is correct.

Q4) What is NOT True regarding the put command?

1. put command works similar to the put method of HashMap in Java
2. put command is used to add data to HBase table
3. put command can be used to update the existing values
4. **A single put command can be used to add multiple cell value**

* Option ‘a’ is incorrect. This is a valid statement. Put command works same as the put method in HashMap. In HashMap, the put method is used to add a value for a given key. Similarly, the put command is used to add a single data value for the given key. Here, the key is a combination of rowkey and the column.
* Option ‘b’ is incorrect. This is a valid statement. Put command is used to add data to the HBase table.
* Option ‘c’ is incorrect. This is a valid statement. Put command is used to update the existing values in HBase table
* **Option ‘d’ is correct. A single put cannot be used to add multiple cell values.**

Q5) What is not a mandatory parameter in a delete command

1. Table name
2. **Timestamp version**
3. Row key
4. Column Name

* Option ‘a’ is incorrect. In every delete command table name has to be there.
* **Option ‘b’ is correct. The version parameter in the delete command is optional. Every delete command must have the table name, row key and column name.**
* Option ‘c’ is incorrect. In every delete command row key is mandatory.
* Option ‘d’ is incorrect. In every delete command column name is mandatory.

Q6) Let’s assume the current state of the Employee table is

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ROWID** | **Personal Details** | | **Professional Details** | | | |
|  | **Name** | **Age** | **Company Name** | | **Designation** | |
|  | **V1** | **V1** | **V1** | **V2** | **V1** | **V2** |
| **1** | Siddharth | 27 | Wipro | Infosys | Software Engineer | Systems Engineer |
| **2** | Sandeep | 27 | ITC Infotech | Accenture | Senior Business Analyst | Business Analyst |

What is the correct command to delete only V2 company name for employee ‘Siddharth’

1. delete ‘Employee’, ‘Siddharth’,’Professional Details:Company Name’, V2
2. delete ‘Employee’, ‘Siddharth’,Personal Details:Company Name’, V2
3. **delete ‘Employee’, ’1’, ‘Professional Details:Company Name’, V2**
4. delete ‘Employee’, ‘1’, ’Siddharth’, ‘Professional Details:Company Name’, V2

* Option ‘a’ is incorrect. The second parameter in the delete command has to be a rowid.
* Option ‘b’ is incorrect. The second parameter in the delete command has to be a rowid.
* **Option ‘c’ is correct. The command is syntactically correct. This will run fine.**
* Option ‘d’ is incorrect. An additional field i.e ‘Siddharth’ is included which is wrong

**Segment 8**

# **Perform Operations in HBase Using Java API**

## **Perform Operations in HBase Using Java API**

Platform Text

Let us now learn how to perform basic CRUD (Create, Read, Update and Delete) operations in HBase using Java API.

**Note:**

1. The video shows how to write and execute a Java program to perform basic CRUD operations in HBase on a Cloudera VM running on localhost, i.e. a personal computer. If you have access to a machine hosting the Cloudera VM, it is highly recommended to use your local machine as shown in the video. The instructions for performing basic CRUD operations in HBase using the Java API on a Cloudera VM are documented below.
2. If you are facing troubles running the Cloudera VM on your local machine, please follow the document "HBase basic operations using Java API on AWS EC2" attached below the video. The document explains how to use the Java API on an Amazon EC2 instance with Cloudera deployed on it.

Video

<Video 6 appears>

## 

Platform Text

Out-of-video Questions

**Segment 9**

# **Additional Shell Commands in HBase**

## **Additional Shell Commands in HBase**

Platform Text

Now that you have learnt about how to perform basic operations in HBase, Let us now learn how to modify and view the specifications of the HBase table using shell commands.

Video

<Video 7 appears>

Platform Text

The shell commands of HBase demonstrated in the video are as follows:

* **describe:** This command is used to check the schema of the HBase tables.

**Syntax:** hbase> describe <table\_name>

* **exists:**  This command is used to verify whether a given table is present in the HBase storage or not.

**Syntax:** hbase> exists <table\_name>

* **alter:** This command is used to modify the specifications of a HBase table like adding column families, updating the version number of column families. It can also be used to delete the column family by applying delete method to it.

**Syntax (Modify the number of versions):**

hbase> alter ‘<table\_name>’, Name=>’<column\_family\_name>’, VERSIONS => <new\_version\_number>

**Syntax (Modify the number of versions for multiple column families):**

hbase> alter ‘<table\_name>’, {Name=>’<column\_family\_name>’, VERSIONS => <new\_version\_number>}, {Name=>’<column\_family\_name>’, VERSIONS => <new\_version\_number>}

**Syntax (To delete):**

hbase> alter ‘<table\_name>’, Name=>’<column\_family\_name>’, METHOD => <’delete’> or

hbase> alter ‘<table\_name>’, ‘delete’ => ‘<column\_family\_name>’

* **drop:**  This command is used to delete a cell in HBase table. But this operator cannot be applied directly to the table. Instead, the table is first disabled. And then it is dropped.

**Syntax:**

Step 1: hbase> disable ‘<table\_name>’

Step 2: hbase> drop ‘<table\_name>’

* **truncate:**  This command is used to remove all the data from the table (Note that we do not intend to delete the table, just the data that is stored in the table). Internally, this command disables the table, drops it and then again recreates it but for us, the end result is that the table’s data has been removed.

**Syntax:** hbase> truncate ‘<table\_name>’

**Note:**  “Scan” command helps to verify whether the contents of a table are deleted or not.

Now that you have learnt about how how to modify and view the specifications of the HBase table using shell commands, Let us now learn how HBase stores multiple versions of the data present in a single cell and how we can filter the data stored in HBase tables.

Video

<Video 8 appears>

Platform Text

Let’s recap the operations and shell commands of HBase demonstrated in the video:

* **Get data based on Timestamp: The** “get” command can also be used to retrieve past versions of records based on timestamp.

**Syntax:** hbase> get ‘<table\_name>’, <row\_key>, {COLUMN => ‘<column\_family\_name>’, TIMESTAMP => value}

* **Get data based on filter condition:** In HBase, fetching data based on a filtering condition is achieved by using Filters. In HBase, filters are like java methods which take two input parameters that are, a logical operator and a comparator. The logical operator specifies the type of the test i.e. equals, less than, etc. The comparator is the number/value against which you wish to compare your record.

Some commonly used filter functions are:

1. **ValueFilter:** A ValueFilter takes a comparison operator and a comparator as the parameter. It compares each value with the comparator using the comparison operator. If the check is true then result is displayed on the console.

**Syntax:** ValueFilter (<compareOp>, ‘<value\_comparator>’)

**Example of using ValueFilter with Scan command:** Let us consider an example of an HBase table named “Companies” maintained by the placement cell of a college , which contains the details of all the companies that visit the college every year. Check whether the company “UpGrad” exists in that table or not.

**Command**: hbase> scan ‘Companies’, {FILTER => “ValueFilter(=, ‘binary:UpGrad’)”}

1. **QualifierFilter:** A QualifierFilter also takes two parameters they are comparison operator and comparator. Each qualifier name is compared with the comparator using the compare operator and if the comparison is true, it returns the key-values in that column.

**Syntax:** QualifierFilter (<compareOp>, ‘<qualifier\_comparator>’)

**Example of using QualifierFilter with Scan command:** Let us consider an example of an HBase table named “Companies” maintained by the placement cell of a college , which contains the details of all the companies that visit the college every year. Do etch all the names of the companies present in the table(In the given HBase table,the names of the companies are present in the column named as “Name”).

**Command**: hbase> scan ‘Companies’, {FILTER => “QualifierFilter(=, ‘substring:Name’)”}

1. **FamilyFilter:** A FamilyFilter is used to fetch key-values for a specified column family.

**Syntax:** FamilyFilter (<compareOp>, ‘<family\_comparator>’)

**Example of using FamilyFilter with Scan command:** Let us consider an example of an HBase table named “Companies” maintained by the placement cell of a college, which contains the details of all the companies that visit the college every year. Fetch the contact details (which include the mobile number, Email ID etc) of all the companies present in the table(In the given HBase table,the names of the companies are present in the column family named as “Contact Details”).

**Command**:hbase> scan ‘<table\_name>’, {FILTER => “FamilyFilter(=, substring:Contact Details’)”}

* **count:**  This command is used to count the number of rows present in the table.

**Syntax:** hbase> count ‘<table\_name>’

## **Additional Reading:**

[HBase Filtering](https://www.cloudera.com/documentation/enterprise/5-5-x/topics/admin_hbase_filtering.html)

Out of video questions

**Segment 10**

# **Summary**

## **Summary**

Platform Text

In this session, you got to know the limitations of SQL and HDFS. You were then introduced to NoSQL databases. After looking at the reasons for their increasing popularity, you saw the CAP theorem and its consequences, in particular the tradeoffs between consistency, availability and partition tolerance in the design of networked shared-data systems.

Later, you were introduced to HBase and the features it provides that differentiate it from Hadoop and other relational or NoSQL data stores. Then, you looked at how data is stored in HBase and understood its data model.

You also understood how to perform basic CRUD operations in HBase using the shell commands, and explored an alternative method of doing the same using the Java API. Finally, you learnt about the commands that help you view and modify the specifications of the HBase tables. You also learnt how HBase is capable of storing multiple versions of the data using timestamp and also learnt to filter the data stored in HBase tables.

The next session contains a few graded questions to check your knowledge of HBase. Good luck!

Out-of-video Questions

Q1. What are your top three takeaways from this session?