Overview

HBase or Hadoop Database, is a **distributed** and **non-relational** database management system that runs on top of the Hadoop Distributed File System (HDFS). HBase can handle large amounts of data while providing **low latency** via parallel processing.

Introduction

The HBase Shell is a command-line interface for interacting with the HBase database.

When the shell is launched, it establishes a connection with the hbase client which is responsible for interacting with the HBase Master for various operations.

The HBase shell translates **user-entered commands into appropriate API calls,** allowing users to interact with HBase in a more user-friendly and interactive manner.

There are several types of **commands** in HBase.

They are,

- General Commands.
- Data Definition Language (DDL) Commands.
- Data Manipulation Language (DML) Commands.
- Security Commands.
- Other HBase Shell Commands.

The **HBase Shell is built on top of the HBase Java API and provides a simplified interface for executing HBase commands without the need for extensive coding.**

General Commands

HBase shell offers a range of general commands to get information regarding the database. The general HBase commands are,

- Whoami
- Status
- Version
- Table Help
- Cluster Help
- Namespace Help

1. whoami

The whoami HBase command is used to retrieve the current user's information.

Syntax:

```
whoami
```

Output:

```
user: hari
```

Explanation:

This output indicates that the user executing the command is logged in as hari in the HBase Shell.

2. Status

The status command provides **information** about the HBase cluster, including the cluster's overall health and the number of servers and regions.

Syntax:

```
status
```

Output:

```
1 active master, 3 region servers, 6 regions
```

Explanation:

- **active master:**
  Number of active master servers in the HBase cluster.
- **region servers:**
  Number of region servers that are currently running.
- **regions:**
  Total number of regions in the cluster.

3. Version

The version command retrieves the **version** information of HBase. **Syntax:**

```
version
```

Output:

```
HBase 2.4.0
```

Explanation:

This output displays the version of HBase installed is 2.4.0.

4. Table Help

The table_help command provides a list of available **commands** related to table operations.

Syntax:

```
table_help
```

Output:

```
TABLE COMMANDS:
  create 'table_name', {NAME => 'column_family_name'}, ...
    Create a table with the specified table_name and column families.

  alter 'table_name', {NAME => 'column_family_name'}, ...
    Alter the schema of an existing table, adding or modifying column families.

  describe 'table_name'
    Display the detailed schema of the specified table.


...
```

5. Cluster Help

The cluster_help command provides information on commands related to **cluster** management. **Syntax:**

```
cluster_help
```

Output:

```
status
version
...
```

6. Namespace Help

The namespace_help command provides guidance on commands for managing **namespaces** in HBase.

Syntax:

```
namespace_help
```

Output:

```
create_namespace 'namespace_name'
describe_namespace 'namespace_name'
...
```

Data Definition Language in HBase

The Data Definition Language (DDL) allows users to define and manipulate the structure of the HBase tables.

Some of the DDL HBase commands are,

- Create
- Describe
- Alter
- Disable
- Enable
- Drop
- Exist

1. Create

The create command is used to **create** a new table in HBase. **Syntax:**

```
create 'table_name', 'column_family1', 'column_family2', ...
```

Explanation:

A table can have multiple-column families. A column family is a group of columns that share common characteristics.

Example:

```
create 'employees', 'personal', 'professional'
```

Output:

```
0 row(s) in 1.4860 seconds
```

This command creates a table named employees with two column families: personal and professional.

In HBase, a **namespace** is a logical container that provides a way to group related tables together, similar to how directories for files.

We can create a namespace in HBase using the create_namespace your_namespace command and switch to a namespace using the following command,

```
namespace 'your_namespace'
```

All tables created after this command will be in the your_namespace namespace.

We can also add various configurations to the HBase commands while creating the column family in a table. **Example:**

```
create 'products', {NAME => 'details', VERSIONS => 5, TTL => '86400'}, {NAME =>
'inventory', COMPRESSION => 'GZ'}
```

Explanation:

- **{}:**
  Used to specify details of a column family.
- **NAME:**
  Specifies the name of the column family.
- **VERSION:**
  Sets the maximum number of versions to keep for each cell.
- **TTL:**
  Sets the Time-to-Live (TTL).
- **COMPRESSION:**
  Specifies the compression algorithm to be used.
- GZ means Gzip compression.

The value 86400 represents the number of seconds (1 day) that cells will be kept before being automatically expired and deleted.

2. Describe

The describe command provides **information** about the specified table, including column families and region details.

Syntax:

```
describe 'table_name'
```

Example:

```
describe 'employees'
```

Output:

```
Table employees is ENABLED
```

```
employees
COLUMN FAMILIES DESCRIPTION
{NAME => 'personal', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY =>
'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE',
TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0',
BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}

{NAME => 'professional', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY
=> 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING =>
'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0',
BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
2 row(s) in 0.0310 seconds
```

Explanation:

This command provides a detailed description of the employees table, including its enabled status and the configuration of its column families.

3. Alter

The alter command allows users to **modify** the schema of an existing table.

Syntax:

```
alter 'table_name', {NAME => 'new_column_family'}
```

Example:

```
alter 'employees', {NAME => 'contact'}
```

Output:

```
0 row(s) in 1.2040 seconds
```

Explanation:

This command adds a new column family named contact to the employees table.

4. Disable

The disable command is used to disable a table. Once disabled, the table becomes **read-only**, and no further modifications can be made to it.

Syntax:

```
disable 'table_name'
```

Example:

```
disable 'employees'
```

Output:

```
0 row(s) in 1.2410 seconds
```

Explanation:

This command disables the employees table, preventing any further modifications.

5. Enable

The enable command is used to enable a previously disabled table. Once enabled, the table becomes **writable** again. **Syntax:**

```
enable 'table_name'
```

Example

```
enable 'employees'
```

Output:

```
0 row(s) in 0.8760 seconds
```

Explanation: This command enables the 'employees' table, allowing modifications to be made.

6. Drop

The drop command in HBase Shell is used to **delete** an existing table from the HBase database. **The table to be deleted should be disabled first using the disable command.**

Syntax:

```
drop 'table_name'
```

Example:

```
drop 'employees'
```

Output:

```
0 row(s) in 1.2300 seconds
```

Explanation:

This command deletes the employees table from the database.

7. Exist

The exist command in HBase is used to **check** the existence of a table or a column family within a table.

Syntax:

```
exists 'table_name'
```

Example:

```
exists 'employees'
```

Output:

```
Table table_name does not exist
```

Explanation:

Since we have deleted the table, the output shows that the table doesn't exist.

Data Manipulation Language in HBase

HBase Shell provides a Data Manipulation Language (DML) that enables users to insert, update, and delete data within HBase tables. Some of the DML HBase commands are,

- Put
- Get
- Scan
- Delete
- Truncate

1. Put

The put command is used to **insert** or update data in a table. **Syntax:**

```
put 'table_name', 'row_key', 'column_family:column_qualifier', 'value'
```

Explanation:

- **table_name:**
  Name of the HBase table.
- **row key:**
  Unique identifier for a specific row in the HBase table.
- Each row in an HBase table is indexed and accessed using its row key.
- **column family:**
  Represents the group to which a specific column belongs.
- **column qualifier:**
  Specifies the specific column within the column family.
- **value:**
  Data that will be stored

Example:

```
put 'employees', '1001', 'personal:name', 'Sample user'
```

Output:

```
0 row(s) in 0.4560 seconds
```

Explanation:

This command inserts the value Sample user into the personal:name column of the row with key 1001 in the employees table.

2. Get

The get command is used to **retrieve** data from a table based on the row key. **Syntax:**

```
get 'table_name', 'row_key'
```

The command requires specifying the table name and row key. **Example:**

```
get 'employees', '1001'
```

Output:

```
COLUMN                          CELL
 personal:name                   timestamp=1628272254000, value=John Doe
1 row(s) in 0.0390 seconds
```

Explanation:

This command retrieves the data associated with the row key 1001 in
the employees table, displaying the column name, timestamp, and cell value.

Command Options

We can also use different options with the GET command to get specific information in the table. The options are,

- **FILTER:**
  accepts a filter string that defines the filter conditions. You can use various filter types like SingleColumnValueFilter, PrefixFilter, ColumnPrefixFilter, etc.

Example:

```
get 'employees', 'row1', {FILTER => "SingleColumnValueFilter('personal', 'age', >=,
'binary:25')"}
```

Explanation:

The above command retrieves the cell values from the row with the key row1 in the employees table. The filter condition SingleColumnValueFilter is applied to fetch only those cells where the age column in the personal column family is greater than or equal to 25.

- **COLUMN:**
  Can specify the name of the column or column family you want to retrieve.
- **TIMERANGE:**
  Takes a single timestamp value and fetches cell values based on a specific timestamp.
- **TIMERANGE:**
  Takes a start time and an end time in the format yyyy-MM-dd HH:mm:ss and retrieves cell values within a specific time range. For instance, TIMERANGE => ['2023-01-01 00:00:00', '2023-01-31 23:59:59'].

3. Scan

The scan command is used to **retrieve** multiple rows or a range of rows from a table. **Syntax:**

```
scan 'table_name'
```

Example:

```
scan 'employees'
```

Output:

```
ROW                              COLUMN+CELL
```

```
 1001                          column=personal:name, timestamp=1628272254000,
value=Sample user
 1002                          column=personal:name, timestamp=1628272296250,
value=New User
2 row(s) in 0.0560 seconds
```

Explanation:

This command retrieves all the rows and associated column families and cells from the employees table.

4. Delete

The delete command is used to **delete** data from a table based on the row key, column family, and column qualifier. **Syntax:**

```
delete 'table_name', 'row_key', 'column_family:column_qualifier'
```

Explanation:

If we want to delete a specific cell, the column family and column qualifier must also be specified. **Example:**

```
delete 'employees', '1002', 'personal:name'
```

Output:

```
0 row(s) in 0.4980 seconds
```

Explanation: This command deletes the data in the personal:name column of the row with key 1002 in the employees table.

5. Truncate

The truncate command in HBase Shell is used to **delete all** data from a specific table while retaining the table structure. **Syntax:**

```
truncate 'table_name'
```

Example:

```
truncate 'employees'
```

Output:

```
Truncated table employees
```

Explanation:

Through the above command, all the data from the employees table are removed.

Security Commands

Security commands in HBase Shell are used to manage user permissions, and access control, and ensure the security of the HBase database. Some of the security HBase commands are,

- Create
- Grant
- Revoke
- User Permissions
- Disable Security

1. Create

The create command is used to create a **new user** with a specified username and password. **Syntax:**

```
create 'user', 'password'
```

Example:

```
create 'bob', 'hbasecommand'
```

Output:

```
0 row(s) in 1.5680 seconds
```

Explanation:

This command creates a new user named bob with the password hbasecommand.

2. Grant

The grant command is used to grant **permissions** to a user on a specific table. **Syntax:**

```
grant 'user', 'permissions', 'table'
```

Explanation:

The permissions can be any of read, write, execute, create, and admin. The permission is applied for a particular table. **Example:**

```
grant 'bob', 'read', 'employees'
```

Output:

```
0 row(s) in 0.7150 seconds
```

Explanation:

This command grants the read permission to the user bob on the employees table.

The execute permission allows the user to execute coprocessor functions on the specified table. **Coprocessors** are custom code modules that run on HBase Region Servers and can perform operations on the server side. The admin permission provides full administrative privileges over the specified table.

3. Revoke

The revoke command is used to **revoke permissions** from a user on a specific table. **Syntax:**

```
revoke 'user', 'permissions', 'table'
```

Example:

```
revoke 'bob', 'read', 'employees'
```

Output:

```
0 row(s) in 0.7890 seconds
```

Explanation:

This command revokes the read permission from the user bob on the employees table.

4. User Permissions

The user_permission command is used to **list** the permissions assigned to a specific user. **Syntax:**

```
user_permission 'user_name'
```

Example:

```
user_permission 'bob'
```

Output:

```
User                          Table, Family, Qualifier, Permission
 bob
 bob                         employees:, [read]
1 row(s) in 0.0710 seconds
```

Explanation:

The output displays the permissions assigned to the user bob.

5. Disable Security

The disable_security command is used to disable **security** in HBase, which removes all user and permission-related restrictions.

Syntax:

```
disable_security
```

Example:

```
disable_security
```

Output:

```
0 row(s) in 1.2230 seconds
```

Explanation:

The output indicates that the security in HBase has been disabled.

Starting HBase Shell

Starting the HBase Shell provides a command-line interface to interact with the HBase database through HBase commands.

The following steps should be followed to start the HBase Shell,

1. Navigate to the directory where HBase is installed through the terminal.
2. The HBase shell can be started using the following command:

```
bin/hbase shell
```

Once the HBase Shell is launched, you will see a command prompt that indicates you are in the HBase Shell environment. From here, you can start entering various HBase commands to interact with the HBase database.