Querying of data from HBase

Now that we have added two rows of data that is with RowKey 1 and RowKey 2 into the table called as employees which consists of two column families. The first column family is going to be called as basic info, the second column family is going to be called as personal info.

Let's quickly describe the schema of the employees table. Here it is. If you perform a scan operation on the employees table, it's going to consist of all this data.

Now, let's try to query this data that is how do we retrieve a row value based on a RowKey. Suppose I want to get all the columns belonging to the row with RowKey 1. Basically, I want to retrieve the entire row identified by RowKey 1. How do I do it?

So the command for that is going to be get get, name of the table which is going to be employees in our case. We have to specify the RowKeys for which we need the entire row. here we get.

For RowKey 1, for the employees table, we basically get all the data that is we have two columns under the basic info and two columns under the personal info column families. The

same applies to RowKey 2 as well. Instead of one, we just mentioned it as two over here, you would be getting all the data corresponding to RowKey 2.

In case, if we don't want the entire row then how do we actually retrieve a particular column family? For that we have to use something like get employees followed by the RowKey and then we just have to specify the column family from which we need to extract the data. I just want the basic info.

So since the basic info column family of the employees table is having two columns for the employees table, we are going to get two column values out here. See we get value as 101 for the employee ID and we get the name as Alice for the name column. Since both of them belong to the basic info column. Now suppose, if we want one of the columns from the basic info column family and if we want one of the columns from the personal info column family, this is the way of doing it. I want the name from the basic info and from the personal info column family, I want the age. So, selectively we can actually pick from which particular column family, which column data we need to query.

We can also bolt this down to a cell level data extraction that is if I don't want the entire

column family, if I just want only one column data, even that is possible. I just have to specify; I just want the name from the basic info column family. This is how we go about getting it. So basically we have seen how to actually store the data and also to retrieve the data based on the RowKeys. Here the RowKey is going to be very important and this is actually going to be the notation or simply the syntax using which we are going to retrieve the data back from the HBase tables.

What I have basically done is, with the existing employees table which originally consisted of just two rows that is two RowKeys, I have now added four more rows that is in total I have actually added 6 rows. If you notice that for rows starting with RowKey 3 until 6, I have just added the data corresponding to the basic info column.

Now, I have actually done a scan of the employees table. Now, what if in my scan operation, I want to restrict the output what I see on the screen that is I don't want to see all the rows.

What if I want to see the rows only from 2 to 5 and ignore the rest? Such kind of filtering option is also available in HBase and this is actually the command to execute this query. So, if you notice in this query that I am using a

scan operation on the employees table where in the brackets, I just mention that start row is from two and the end row is from three. Instead of 3, I can actually make it 5. So we will be getting the rows starting from only 2 to 5. This is how we can actually limit the range of rows what we basically want to see in the output. Here we go.

So the fifth row is actually ignored. So, basically it's 2 to 4, one less than the number out here. If you just do a scan operation, it's going to list out all the rows. So this is very useful in case if you want to filter out only certain rows based on the range of RowKeys when your table is going to be literally having billions of rows.

Now that in the employees table what I have try to do is I just try to retrieve the information present under the personal info column family for the RowKey is equal to 1. It just says that the salary of the person is almost like 4000, you can actually assume denomination to be in any currency which you want and then the age actually is going to be 37.

If you want to know the information corresponding to the entire row, we just have to execute the same query without this personal info. Here, it says that this row with

RowKey 1 belongs to the person named Alice whose employee ID is going to be 101. The name and the employee ID is going to belong to the basic info column family whereas the age and the salary is going to belong to the personal info column family of the same row.

Now, what if I want to update the age of Alice from 37 to 38. Next year, the age actually is going to be updated. So there is no explicit update operation or update command actually available in HBase, instead we just have to use the same put. So we would say, put into employees. Here it is. So, what I have done is basically used the put command to update the age column value to 38 under the personal info column family for the RowKey is equal to 1.

Now, let's try to retrieve the data corresponding to the RowKey 1 and let's see whether the age of Alice has been updated from 37 to 38. We execute the get employees for RowKey1 and here we go. So it's been updated to 38. Previously it was 37 and now it's updated to 38. This is one of the very important features of HBase.

That is the reason why it is being very popular because we cannot do an update operation in Hive or in PIG because the very nature of HDFS is that once data is written in to the HDFS, it

cannot be updated. However, in HBase, it is possible. But it is not directly possible, it is indirectly possible because of the HFiles being stored into the HDFS ultimately.

Indirectly, HBase actually does a lot of bookkeeping activity behind the scenes which we discussed in the theory part. So we are not going to again discuss how it's going to allow updates but it's not directly happening, it's indirectly happening. But for the end user, we see that data is updated naturally. from 37, it got updated to 38. So we just use the put command instead of doing anything else.

In the employees table, for the row corresponding to the RowKey 1, especially under the personal info column family, the age field has been updated twice. Initially when we created the table, the age field for Alice was 37. To demonstrate the update operation using the put command, it is updated to the 38. Now recently I have updated it to 39.

So what has happened is three times, the data has been changed under the age column of the personal info column family for RowKey is equal to one under the employees table. Now that, we had mentioned during the theory part discussion that HBase is going to retain three

versions of every given cell value, meaning based on the timestamp, it's just going to retrieve and display only the latest value for any given field. However, internally it will be retaining the last two changed values as well. So, this actually makes up for a total three versions to be maintained. That is what you actually see here when you try to describe the employees table. By default, it tries to actually maintain three versions. You would see here, versions are going to be equal to three.

Now what if I want to see the previous two versions as well. There is a command to go about it that is I can actually see the values 37 and 38 still being retain by HBase for this particular column family personal info and for the age field which I can actually get it using the get command itself. The command for that would actually look like get from the employees table, the details for RowKey is equal to 1. The particular column which I want to retrieve is going to be personal_info column family under which I want to get the information about the age column for all the three previous versions which actually is stored internally.

You press the enter key, you would see the three values which actually is still retained. So, the latest one is going to be displayed

when you actually scan the table or when you try to retrieve it using the get command based on the most recent timestamp.

Now, let me try to actually update the same age column once again for the RowKey is equal to 1 in the employees table. Another personal info column family, I just need to update the age to 40. Yes, so what would happen in this case is if I try to retrieve using the get command-get. So I would eventually get only the latest updated value that is going to be 40.

Now, what if, I am going to re-execute the same command if I want to see all the last 3 versions. Notice that previously, the last three versions stored was 37, 38, 39. Now that since I have actually stored 40 and the number of versions that is going to be retain by default is going to be only 3 versions based on a timestamp. The oldest one will be automatically deleted from the HBase entry that is the value corresponding to the age of 37 will now be removed which we can verify by re-executing the same query that is if we try to get the age field for up to the 3 versions for RowKey is equal to one from the employees table, we would not be able to see 37, instead we will see 40, 38 and 39.

Let’s quickly re-execute the same query. It’s going to be stored in the buffers. So I'm just going to keep pressing the up arrow Key to get the same command.

If you notice over that it's going to be get employees with RowKey is equal to 1 where I want the particular column information where the personal info column and then the versions is going to be up to 3. Personal info specifically I want the age field under the personal info column family. press the enter key and you would notice that 37 is missing because, now, it cannot maintain 4 entries, it can maintain only 3 entries because the number of versions of every cell it can maintain based on the timestamp is going to be only 3.

But, however, when you try to scan the table, it will display only the most recently updated one based on the timestamp that is the most recent timestamp will be displayed, when you just perform a scan operation on the employees table. Here we go. So we would not see the previous versions, we would just see 40.

Now what during the scan itself, we want all the versions. How do we do that? Just like we specifically try to retrieve based on a particular RowKey, now, we are going to scan

on the entire range of keys on a given table but we want to make sure, for every row, for every column, we want all the 3 versions which is going to be stored. How do we implement that? This is the command to actually get a details during the scan operation itself.

Scan employees table where we want the data corresponding to all the RowKeys for all the columns, we want all the 3 versions. Just press the enter key. Here you will actually see only for those columns where the old data is applicable, it would be displayed.

For example, for RowKeys is equal to one, one particular column family that is personal info column family has three variances for age 38, 39 and 40. All the 3 is going to be displayed. Now, in case, if you don't want the three versions, if you just want two versions, you can just change the number to 2 over here. And you would see only the last two updated values. This is how you play around with the versions option for every scan or even get commands or get options within the HBase tables.