Agent Plugin for BigQuery

Contents

BigQuery location is natively supported (agent plugin is not required) since HVR version 5.7.5/5. For more information, see Requirements for BigQuery.

Name

hvrbigqueryagent.py

Synopsis

hvrbigqueryagent.py *mode chn loc* [*–options*]

Description

The agent plugin **Hvrbigqueryagent** enables HVR to replicate data into BigQuery database. This agent plugin should be defined in the HVR channel using action **AgentPlugin**. The behavior of this agent plugin depends on the *–options* supplied in **/UserArgument** field of **AgentPlugin** screen.

This agent plugin supports replication of data in Avro format only.

For better performance it is recommended to install HVR remote listener on VM (virtual machine) located in Google Cloud and use the HVR transfer protocol with compression when using BigQuery for replication.

Options

This section describes the parameters that can be used with **Hvrbigqueryagent**:

Parameter	Description
-r	Truncates existing data from target and then recreates table and insert new rows. If this option is not defined, appends data into table.
-s *col_name*	Soft deletes the column *col_name*.

Environment Variables

The **Environment** variables listed in this section should be defined when using this agent plugin:

Environment Variable Name	Description

$HVR_GBQ_CREDFILE	The directory path for the credential file. The default directory path in - • Linux: **$HOME/.config/gcloud/application_default_credentials.json** • Windows: **Users/***<user name>***/.config/gcloud /application_default_credentials.json**
$HVR_GBQ_PROJECTID	The Project ID in BigQuery. This is the Project ID of the dataset being used for replication.
$HVR_GBQ_DATASETID	The Dataset ID in BigQuery. This dataset should belong to the Project ID defined in $HVR_GBQ_PROJECTID.

Installing Python Environment and BigQuery Client

Google BigQuery client is required for uploading data into BigQuery from local source and convert it into tables.

To enable data upload into BigQuery using HVR, perform the following on HVR **Integrate** machine:

1. Install Python 2.7.x +/3.x. Skip this step if the mentioned python version is already installed in the machine.
2. Install the following python client modules:

   ```
   pip install google_cloud_bigquery
   pip install enum34
   ```

   Installing enum34 is not required for python versions 3.4 and above.
3. Pass authorization process with Google:

   ```
   gcloud auth application-default login
   ```

4. Copy configuration file (location is differ on different platforms, see below) into integration side.

   Linux, **$HOME/.config/gcloud/application_default_credentials.json**

   Windows, **Users/***user_name***/.config/gcloud/application_default_credentials.json**

BigQuery Date and Timestamp Limitations

1. BigQuery maps all Avro date and Avro timestamp-millis/micros data types into one common TIMESTAMP type
2. BigQuery has the following limitations:
   a. the minimum date value is 0001-01-01 00:00:00.000000
   b. the maximum date value is 9999-12-31 23:59:59.999999
3. BigQuery only recognizes dates of the Gregorian calendar, even if the date is less than 1582-10-15. Note that this calendar is named the Proleptic Gregorian calendar.
4. By default, the Avro file contains dates calculated according to the rules of the Julian calendar if the date is less than 1582-10-04; otherwise, the Gregorian calendar is used. Such difference leads to incorrect dates translation while uploading Avro files into BigQuery. To resolve this issue, set action **Column Properties** with specific **/DataTypeMatch** parameters (the following example is for an Oracle source):

Group	Table	Action
FILE	*	FileFormat /Avro

FILE	*	ColumnProperties /DatatypeMatch=date /Datatype="avro date gregorian"
FILE	*	ColumnProperties /DatatypeMatch=timestamp (oracle)[prec < 4] /Datatype="avro timestamp millis gregorian"
FILE	*	ColumnProperties /DatatypeMatch=timestamp (oracle)[prec > 3] /Datatype="avro timestamp micros gregorian"

5. Some dates in the Julian calendar are not present in the Gregorian calendar and vice versa.

   Example:

Julian	Proleptic Gregorian
1500-02-29	Not present, replaced by 1500-03-01
Not present	Range: 1582-10-(05-14)

   If the source has such dates, it could lead to data inconsistency. To resolve this issue, dates must be converted to strings.

Use Case

Use Case 1: BigQuery tables with timekey column (No burst table idiom).

Group	Table	Action
FILE	*	Integrate /ReorderRows=SORT_COALESCE /RenameExpression="{hvr_integ_tstamp}-{hvr_tbl_name}.avro"
FILE	*	FileFormat /Avro
FILE	*	ColumnProperties /Name=hvr_op_val /Extra /IntegrateExpression={hvr_op} /Datatype=integer
FILE	*	ColumnProperties /Name=hvr_integ_tstamp /Extra /IntegrateExpression={hvr_integ_tstamp} /Datatype=datetime
FILE	*	ColumnProperties /Name=hvr_integ_key /Extra /IntegrateExpression={hvr_integ_seq} /Datatype=varchar /Length=36 /Key /TimeKey
FILE	*	AgentPlugIn /Command=hvrbigqueryagent.py /Context=!recreate
FILE	*	AgentPlugIn /Command=hvrbigqueryagent.py /UserArgument="-r" /Context=recreate
FILE	*	Environment /Name=HVR_GBQ_CREDFILE /Value=<path>
FILE	*	Environment /Name=HVR_GBQ_DATASETID /Value=<dataset_id>
FILE	*	Environment /Name=HVR_GBQ_PROJECTID /Value=<proejct_id>

In this use case, during the execution of mode **refr_write_end**,

- If option **-r** is not defined, then HVR appends new data into table.
- If option **-r** is defined, then HVR re-creates table and insert new rows.

Use Case 2: BigQuery tables with soft delete column (using burst table).

Group	Table	Action

FILE	*	Integrate /ReorderRows=SORT_COALESCE /RenameExpression="{hvr_integ_tstamp}-{hvr_tbl_name}.avro"
FILE	*	FileFormat /Avro
FILE	*	ColumnProperties /Name=hvr_is_deleted /Extra /SoftDelete /Datatype=integer
FILE	*	ColumnProperties /Name=hvr_integ_tstamp /Extra /IntegrateExpression={hvr_integ_tstamp} /Datatype=datetime
FILE	*	AgentPlugIn /Command=hvrbigqueryagent.py /UserArgument="-s hvr_is_deleted" /Context=!recreate
FILE	*	AgentPlugIn /Command=hvrbigqueryagent.py /UserArgument="-r -s hvr_is_deleted" /Context=recreate
FILE	*	Environment /Name=HVR_GBQ_CREDFILE /Value=<path>
FILE	*	Environment /Name=HVR_GBQ_DATASETID /Value=<dataset_id>
FILE	*	Environment /Name=HVR_GBQ_PROJECTID /Value=<proejct_id>

In this use case, during the execution of mode **refr_write_end**, burst table is not used. Data is uploaded directly into base table,

- If option **-r** is not defined, then HVR appends new data into table.
- If option **-r** is defined, then HVR recreates table and insert new rows.

During the execution of mode **integ_end**, HVR

- Updates all rows in base table if rows with corresponding keys are present in temporal burst table.
- Inserts all rows into base table from burst table if they are missed in base table.
- Drops burst table on BigQuery.