Hvrswitchtable

Since v5.3.1/4

Contents

Name

hvrswitchtable - Schedule switch of one channel's tables into another channel without interrupting replication.

Synopsis

hvrswitchtable [*-options*] *hubdb chn1 chn2*

hvrswitchtable [**-m**] [**-i** *latency*] [**-k**] [**-k** *table*]... [**-h** *class*] [**-u** *user*[*/pwd*]] *hubdb chn1 chn2*

hvrswitchtable [**-a** | **-A** *integ_loc*] [**-s**] [**-h** *class*] [**-u** *user*[*/pwd*]] *hubdb chn1 chn2*

Description

Command **hvrswitchtable** encapsulates a series of steps required to prepare a switch of some or all tables of channel *chn1* into channel *chn2* at a given 'switch moment'. This is a moment in the future which is relative to **hvr_cap_tstamp** value and not the job's processing time. Switching tables means moving all the specified tables including actions which are explicitly defined on *chn1* and these tables (i.e. actions where channel is *chn1* and table is not '*****') from *chn1* into *chn2*. The prepared switch will be performed without interrupting replication of any table in *chn1* and *chn2*.

The argument *hubdb* specifies the connection to the hub database. For more information about supported hub databases and the syntax for using this argument, see Calling HVR on the Command Line.

This command can be used to add tables to an existing channel without having to stop capture or integrate.

Requirements for a successful channel merge are:

- All locations of *chn1* and *chn2* must be the same
- The location groups of *chn1* and *chn2* must have the same names and members
- Table names of the tables being switched from *chn1* should not be used in *chn2*.
- All tables in *chn1* must be refreshed
- Both *chn1* and *chn2* must have running capture and integrate jobs with a low latency

Options

This section describes the options available for command **hvrswitchtable**.

Parameters	Description

-a	Abort switch. Removes newly added tables and actions from *chn2* and let *chn1* continue with the replication. This is not possible if switch has already succeeded for one or more integrate locations. HVR Scheduler must be running to abort switch.
-A*integ_loc*	Abort switch only for specific integrate location.
-h*class*	Location *class* of the hub database. Valid values for *class* are **db2**, **db2i**, **ingres**, **mysql**, **oracle**, **postgresql**, **sqlserver**, or **teradata**. For more information, see Calling HVR on the Command Line.
-i*latency*	Specify an initial minimum *latency*. Valid input for *latency* are time formats *HH***:***MM* or *HH***:***MM***:***SS* or an integer (in seconds). Switch of tables will not be performed if one of the involved jobs has a initial latency higher than *latency*.
-k	Keep *chn1* integrate jobs. Do not delete integrate jobs of *chn1* after successful switch.
-m*moment*	Specify future switch *moment* relative to **hvr_cap_tstamp**. Valid input for *moment* are date-time formats *YYYY-MM-DD [HH:MM:SS]* (in local time) or *YYYY-MM-DD***T***HH:MM:SS+TZD* or *YYYY-MM-DD***T***HH:MM:SS***Z** or **now***[±S ECS]* (current time, not **hvr_cap_tstamp**) or an integer (seconds since **1970-01-01 00:00:00 UTC**).
-s	Do not suspend and restart integrate jobs when aborting.
-t*table*	Specify table scope. Only switch table *table* from *chn1* to *chn2*. This option can be supplied multiple times. If not specified, all tables will be switched and integrate jobs of *chn1* will be deleted afterwards (unless **-k** is specified). It is not possible to switch all tables using option **-t**.
-u*user*[*/pwd*]	Connect to hub database using DBMS account *user*. For some databases (e.g. SQL Server) a password must also be supplied.

Examples

Example 1: Adding tables to a running channel via a temporary channel

Consider a running channel **chn_main** replicating tables **tab_main1** and **tab_main2** from capture location **cap** to integrate location **integ**. Location **cap** is member of location group **CAP** and location **integ** is member of location group **INTEG**. **chn_main** is running with a low latency and without errors. Now **hvrswitchtable** can be used to add new tables **tab_new1** and **tab_new2** without interrupting capture or integrate. First a new channel **chn_tmp** is created with the same location groups **CAP** and **INTEG**. Then new tables **tab_new1** and **tab_new2** are added using **hvradapt,chn_tmp** is initialize using **hvrinit** and after starting the capture job the new tables are refreshed with an online **hvrrefresh**. After **hvrrefresh** completed successfully the integrate job can be started and HVR will start replicating all changes made to the new tables. After ensuring that **chn_tmp** runs fine with a low latency, a switch if the tables of **chn_tmp** into **chn_main** can be prepared.

Executing the following command will first check that all capture and integrate jobs in **chn_tmp** and **chn_main** run with a latency lower than 10 seconds. If so it adds tables **tab_new1** and **tab_new2** to **chn_main** and prepares to switch the tables in 1 minute.

```
$ hvrswitchtable -m now+60 -i 10 hubdb chn_tmp chn_main
```

Note that the 'now' in the command line refers to the current time (not relative to **hvr_cap_tstamp**). After **chn_tmp** has captured and integrated all changes that were made to **tab_new1** and **tab_new2** up to 1 minute after **hvrswitchtable** was run, **chn_main** will capture and integrate all future changes and the integrate job of **chn_tmp** will be deleted. Now all tables have been switched and **chn_tmp** can be deleted.

Example 2: Schedule and abort a merge of two channels

The scenario is similar to Example 1 but now location group **INTEG** contains a second integrate location **intg2** and **chn_tmp** is already running fine with a low latency. In this example the current time is '2017-12-10T14:30:00+01:00'.

Executing the following command prepares the switch for '2017-12-10T15:00:00+01:00', i.e., in 30 minutes and the integrate jobs of **chn_tmp** will not be deleted after the switch:

```
$ hvrswitchtable -m 2017-12-10T15:00:00+01:00 -k hubdb chn_tmp chn_main
```

If you want to abort the switch before '2017-12-10T15:00:00+01:00', execute the following command:

```
$ hvrswitchtable -a -s hubdb chn_tmp chn_main
```

Aborting the switch removes tables **tab_new1** and **tab_new2** from **chn_main** and the replication of both channels will continue without any interruption.

Assume the switch was not aborted and at '2017-12-10T14:58:00+01:00' the integrate job for location **intg2** in channel **chn_tmp** (**chn_tmp-integ-intg2**) failed due to some integration error. At '2017-12-10T15:00:00+01:00' tables have been switched successfully for location **integ**, but **chn_tmp-integ-intg2** is still failing. The integrate job for **intg2** in **chn_main** (**chn_main-integ-intg2**) will wait for **chn_tmp-integ-intg2** to integrate all changes up to '2017-12-10T15:00:00+01:00', so now it is hanging. To abort the switch only for location **intg2**, execute the following command:

```
$ hvrswitchtable -A intg2 hubdb chn_tmp chn_main
```

This will not remove tables **tab_new1** and **tab_new2** from **chn_main**, but will restart **chn_main-integ-intg2** and ensure that **chn_main** will not replicate any changes for **tab_new1** and **tab_new2**. After integration issues in **chn_tmp** have been resolved, integration of **tab_new1** and **tab_new2** into **intg2** will continue in **chn_tmp**.

Example 3: Switch some tables between two running channels

Consider a situation where two identically configured channels **chn_from** and **chn_to** are moving tables at a low latency. For this example **chn_from** is replicating tables **tab1**, **tab2** and **tab3** and **chn_to** is replicating tables **tab4** and **tab5**. To switch tables **tab2** and **tab3** from **chn_from** to **chn_to** in 1 hour execute the following command.

```
$ hvrswitchtable -m now+3600 -t tab2 -t tab3 hubdb chn_to chn_from
```

This will add **tab2** and **tab3** and all their associated actions to **chn_to**. After all involved integrate jobs are passed the given switch moment (i.e. **hvr_cap_tstamp** has gone passed now +1 hour), **chn_to** will replicate all changes for **tab2** and **tab3** and the tables and their associated actions will be removed from **chn_from**. That means **chn_from** will now only replicate **tab1** and **chn_to** will replicate **tab2**, **tab3**, **tab4** and **tab5**. No further actions are required and **chn_from** and **chn_to** will continue with their replication. Note that aborting of the switch works as in Example 2, only the table scope (option **-t**) must be provided for the abort command as well.

Example 4: Moving a portion of tables to a new secondary channel

Consider a situation where the main channel **chn_main** is running with many tables being replicated from capture location **cap** to integrate location **integ**. Location **cap** is a member of location group **CAP** and location **integ** is a member of location group **INTEG**. Channel **chn_main** is running with low latency and without errors. A new secondary channel **chn_sec** needs to be created that would take 50% of the load from the main channel **chn_main**. It is assumed that the source is always active/live and that the main channel **chn_main** has been running for some time. All checkpoints, including the capture and integrate checkpoints, should be maintained.

To implement this, first a secondary channel **chn_sec** is created with the same location groups **CAP** and **IN TEG**. Then a baseline table is added to the secondary channel using **hvradapt**, **chn_sec** is initialized using **hvrinit**, and after starting the capture job, the baseline table is refreshed with an online **hvrrefresh**. After **hvrrefresh** completed successfully, the integrate job can be started.

To move half of the tables from **chn_main** to **chn_sec** in 1 hour, execute the following command:

```
$ hvrswitchtable -m now+3600 -t tab1 -t tab2 – tab3 hubdb chn_sec chn_main
```

If there are multiple tables in **chn_main**, each table to be moved to **chn_sec** needs to be listed with **-t** in the command

After the switch completed successfully, the baseline can be deleted from **chn_sec** (optional).

Checkpoints

No manual action is required to maintain correct checkpoints for the new secondary channel **chn_sec**. Up to the switch point, the capture state, capture checkpoints and integrate state for the tables being moved are handled by the original channel **chn_main**. After the switch (which is always on the transaction boundary), the capture state, capture checkpoints and integrate state will automatically be handled by the secondary channel **chn_sec**.