Quick Start for HVR - DB2 for LUW

Contents

This quick start guide helps you to get started with HVR for replicating data between DB2 databases.

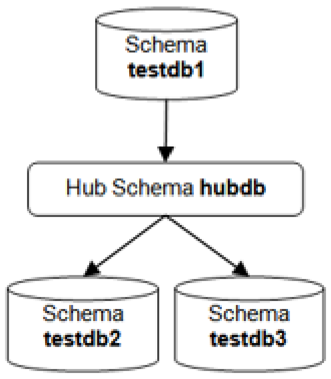
To proceed with this replication you must have basic understanding about HVR's architecture and terminolo gies like Hub, Location, Channel, Location Groups, Actions etc.

The example here demonstrates how to replicate tables from one DB2 database (source location) to another two DB2 databases (target locations).

In real-life scenarios, the source location(s) and the target location(s) reside on different machines and the HVR hub can reside on source or target or a separate machine. However, in this example, for simplicity we have the source, target, and HVR hub database on the same machine. If the HVR hub is installed on a Unix /Linux machine, to manage HVR hub and the replication activities from a remote location (for example a PC), you need to install HVR GUI on your PC.

Before proceeding with this example ensure that the requirements for using HVR with DB2 are met.

For information about access privileges and advanced configuration changes required for performing replication using DB2, see Requirements for DB2 for Linux, UNIX and Windows.



Create Test Databases and Tables

Create three test databases, each containing two empty tables named **dm01_product** and **dm01_order**. If replication is configured between existing databases and tables then this step should be skipped.

```
$ db2 create database testdb1
$ db2 create database testdb2
$ db2 create database testdb3
```

Create the test tables:

```
$ cd $HVR_HOME/demo/hvr_demo01/base/db2
$ db2 connect testdb1
$ db2 '-td;;' <hvr_demo01.cre
$ db2 '-td;;' <hvr_demo01.mod
$ db2 connect reset
$ db2 connect testdb2
$ db2 '-td;;' <hvr_demo01.cre
$ db2 '-td;;' <hvr_demo01.mod
$ db2 connect reset
$ db2 connect testdb3
$ db2 '-td;;' <hvr_demo01.cre
$ db2 '-td;;' <hvr_demo01.mod
$ db2 connect reset
$ db2 update db cfg for testdb1 using logarchmeth1 logretain
$ db2 update db cfg for testdb1 using logarchmeth2 off
$ db2 backup db testdb1
```

Install HVR

First read section Architecture Overview which explains the HVR's terminology and architecture. In particular this explains the importance of a hub database.

Then install the HVR software on the hub machine by following the installation steps in section Installing HVR on Windows or Installing HVR on Unix or Linux. If the hub machine is a Unix machine then HVR can either be installed on a Windows PC (so the HVR GUI can run on the PC and connect to the Unix hub machine) or the HVR GUI can be run on the Unix hub machine and connect back to an X server running on a PC.

Create the Hub Database

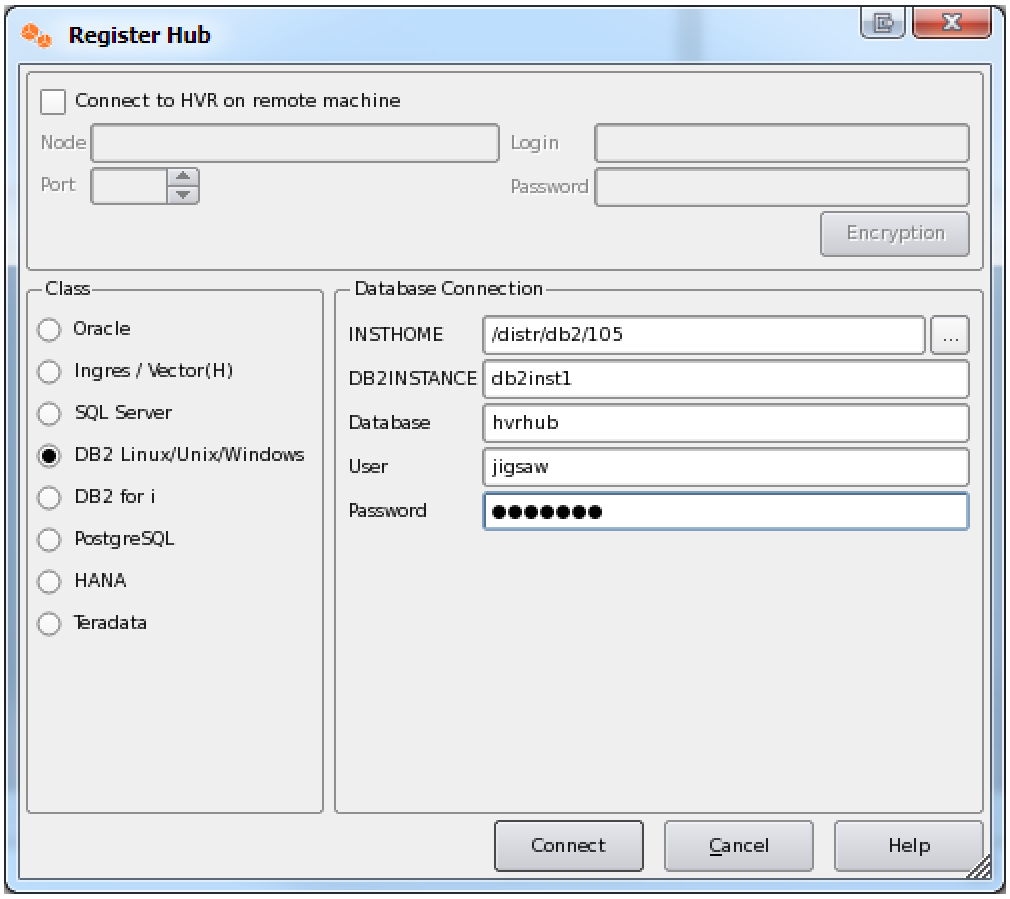
Create the hub database, in which the HVR GUI will store the channel definition.

```
$ db2 create database hvrhub
```

Connect to Hub Database

This section describes how to connect **HVR GUI** to the hub database.

When **HVR GUI** is launched for the first time, the **Register Hub** dialog is displayed automatically. The **Regi ster Hub** dialog can also be accessed from the main menu **File Register Hub**.


Register Hub

Connect to HVR on remote machine

Node		Login	
Port		Password	

Class
- Oracle
- Ingres / Vector(H)
- SQL Server
- [✓ DB2 Linux/Unix/Windows]
- DB2 for i
- PostgreSQL
- HANA
- Teradata

Database Connection

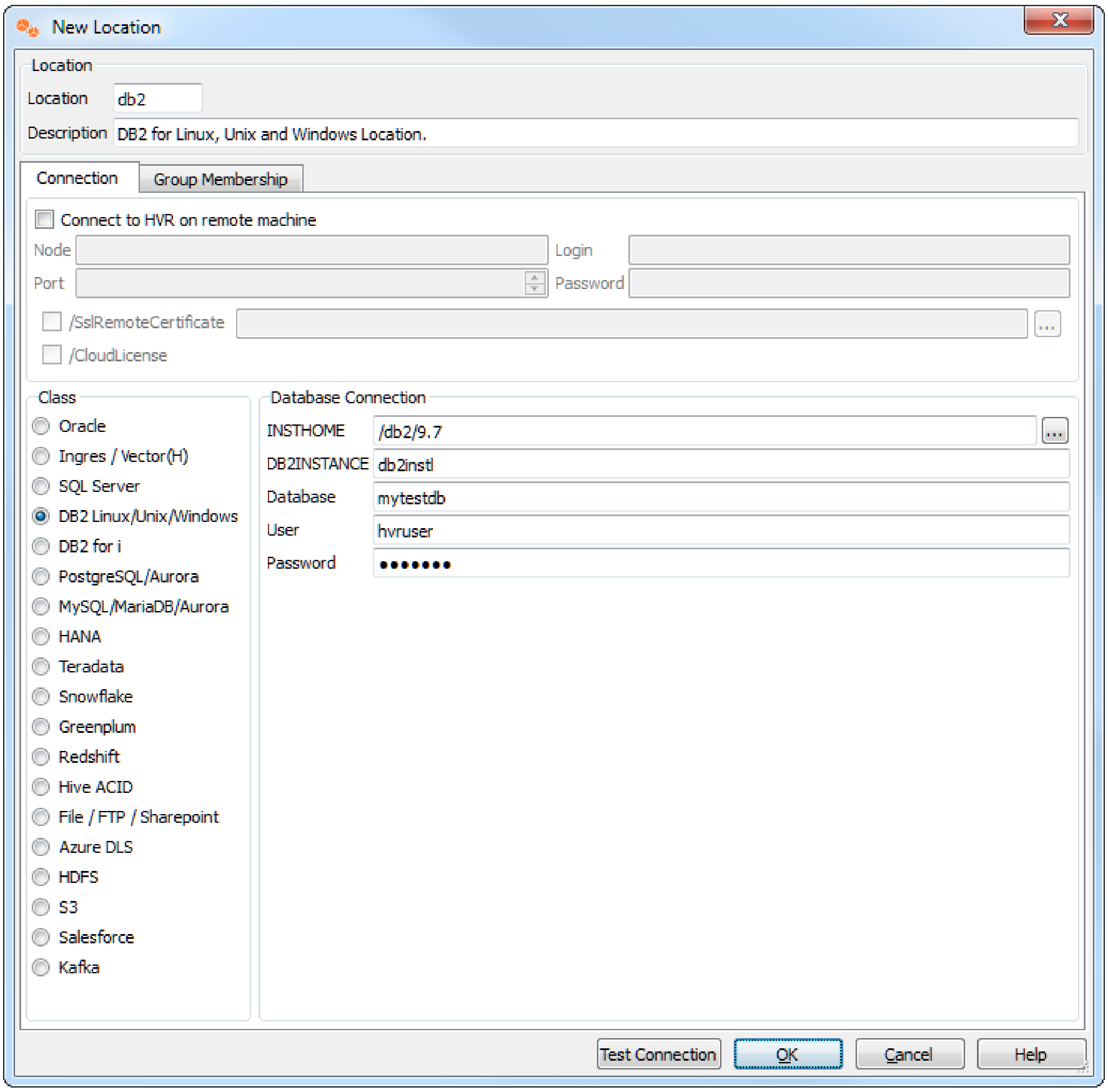
INSTHOME	/distr/db2/105
DB2INSTANCE	db2inst1
Database	hvrhub
User	jigsaw
Password	●●●●●●●

Enter connection details and click **Connect**.

For a new hub database a dialog will prompt **Do you wish to create the catalogs?**; click **Yes**.

Create DB2 Locations

Next, create three locations (one for each test database) by right-clicking **Location Configuration New Location**.



Location: db2
Description: DB2 for Linux, Unix and Windows Location.

[ ] Connect to HVR on remote machine
Node:
Login:
Port:
Password:
[ ] /SslRemoteCertificate
[ ] /CloudLicense

Class: Oracle, Ingres / Vector(H), SQL Server, [✓ DB2 Linux/Unix/Windows], DB2 for i, PostgreSQL/Aurora, MySQL/MariaDB/Aurora, HANA, Teradata, Snowflake, Greenplum, Redshift, Hive ACID, File / FTP / Sharepoint, Azure DLS, HDFS, S3, Salesforce, Kafka

Database Connection

INSTHOME	/db2/9.7
DB2INSTANCE	db2instl
Database	mytestdb
User	hvruser
Password	●●●●●●●

In this example, there is no need to check **Connect to HVR on remote machine** because **testdb1** is on the same machine as the hub.

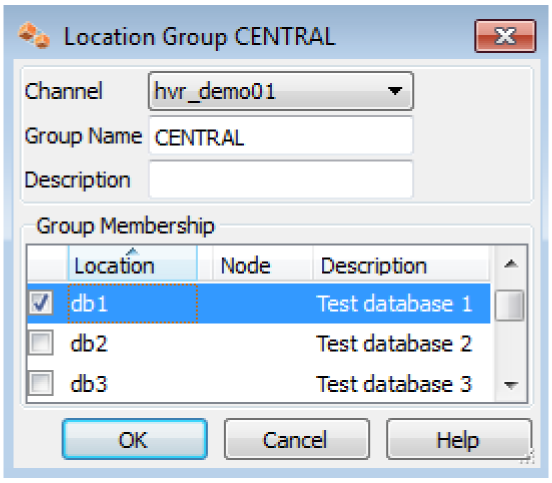
Ignore the **Group Membership** tab for now.

Make locations for **testdb2** and **testdb3** too.

Now define a channel using **Channel Definitions New Channel**.

Create Location Groups

The channel needs two location groups. Under the new channel, right-click **Location Groups New Group**. Enter a group name (for instance **CENTRAL**).



Channel: hvr_demo01
Group Name: CENTRAL
Description:

Group Membership

	Location	Node	Description
✓	db1		Test database 1
	db2		Test database 2
	db3		Test database 3

Add location **db1** as a member of this group by checking the box for **db1**.

Then create a second location group, called **DECENTRAL** that has members **db2** and **db3**.

The new channel also needs a list of tables to replicate. This can be done as follows: right-click **Tables** **Table Explore**.

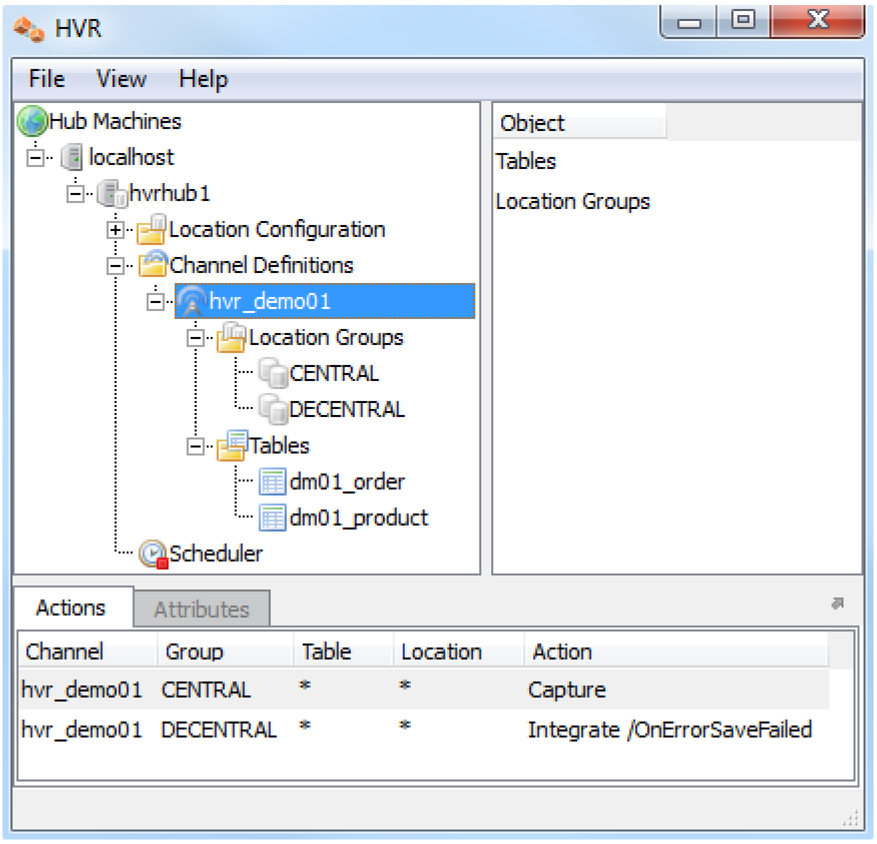
- Choose the first of the three locations **Connect**.
- In the **Table Explore** window, click on both tables and click **Add**.
- In new dialog **HVR Table Name** click **OK**.
- Close the **Table Explore** window.
- Perform table select again on one of the other locations and confirm that all tables to be replicated have value **Same** in column **Match**.

Define Actions

The new channel needs two actions to indicate the direction of replication.

- Right-click group **CENTRAL** **New Action** **Capture**.

- Right-click **Group DECENTRAL New Action Integrate**. Check **/OnErrorSaveFailed**, this affects how replication errors are handled.



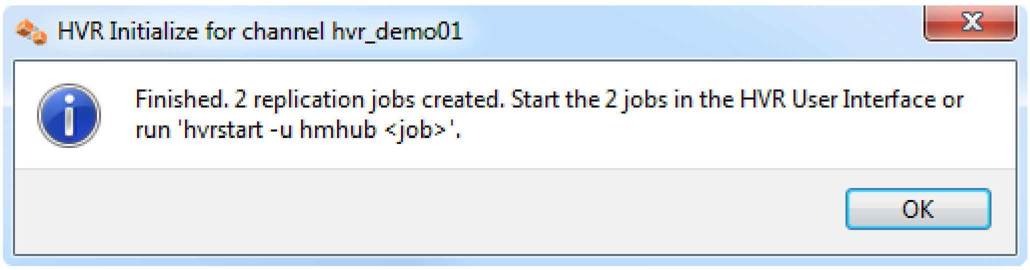
Channel	Group	Table	Location	Action
hvr_demo01	CENTRAL	*	*	Capture
hvr_demo01	DECENTRAL	*	*	Integrate /OnErrorSaveFailed

Note that the **Actions** pane only displays actions related to the objects selected in the left-hand pane. So click channel **hvr_demo01** to see both actions.

Enable Replication with HVR Initialize

Now that the channel definition is complete, create the runtime replication system.

1. Right-click channel **hvr_demo01 HVR Initialize**.
2. Choose **Create or Replace Objects** and click **HVR Initialize**.



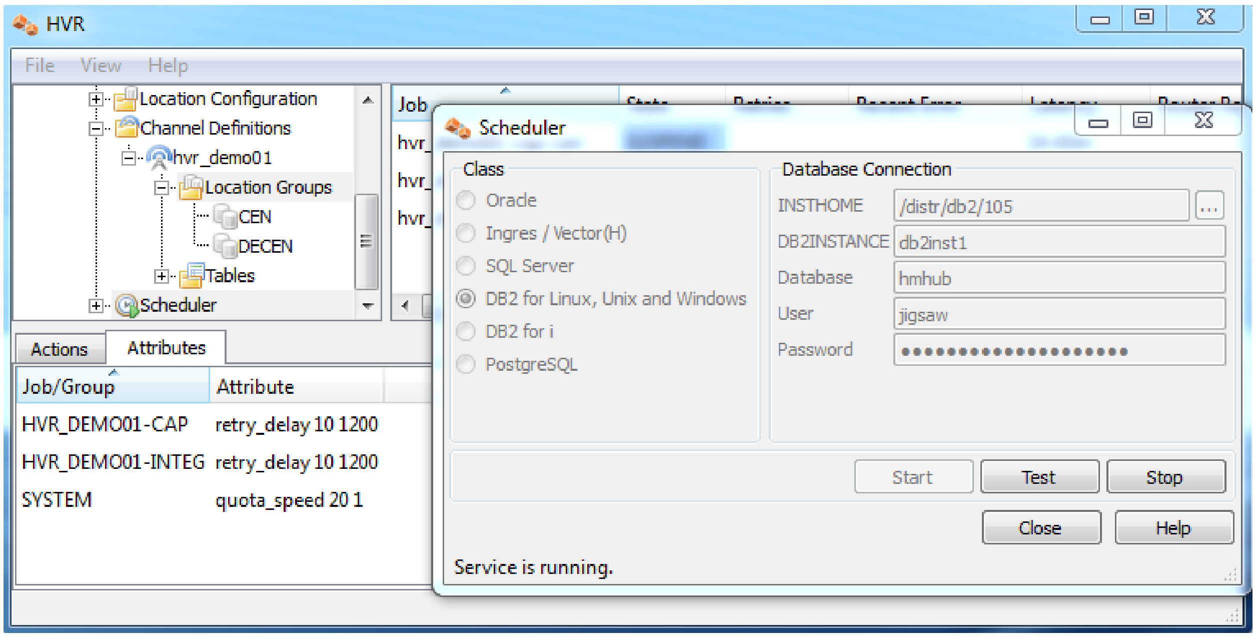
Finished. 2 replication jobs created. Start the 2 jobs in the HVR User Interface or run 'hvrstart -u hmhub <job>'.

From the moment that HVR Initialize is done, all changes to database **sourcedb** will be captured by HVR when its capture job looks inside the logging.

HVR initialize also creates three replication jobs, which can be seen under the **Scheduler** node in the GUI.

Start Scheduling of Replication Jobs

Start the Scheduler on the hub machine by clicking in the HVR GUI on the **Scheduler** node of the hub database.



Job/Group	Attribute
HVR_DEMO01-CAP	retry_delay 10 1200
HVR_DEMO01-INTEG	retry_delay 10 1200
SYSTEM	quota_speed 20 1

Class: Oracle, Ingres / Vector(H), SQL Server, [✓ DB2 for Linux, Unix and Windows], DB2 for i, PostgreSQL

Database Connection

INSTHOME	/distr/db2/105
DB2INSTANCE	db2inst1
Database	hmhub
User	jigsaw
Password	•••••••••••••••••••

Service is running.

Next, instruct the HVR Scheduler to trigger the replication jobs.

The replication jobs inside the Scheduler each execute a script under **$HVR_CONFIG/job/hvrhub /hvr_demo01** that has the same name as the job. So job **hvr_demo01–cap–db1** detects changes on database **testdb1** and stores these as transactions files on the hub machine.

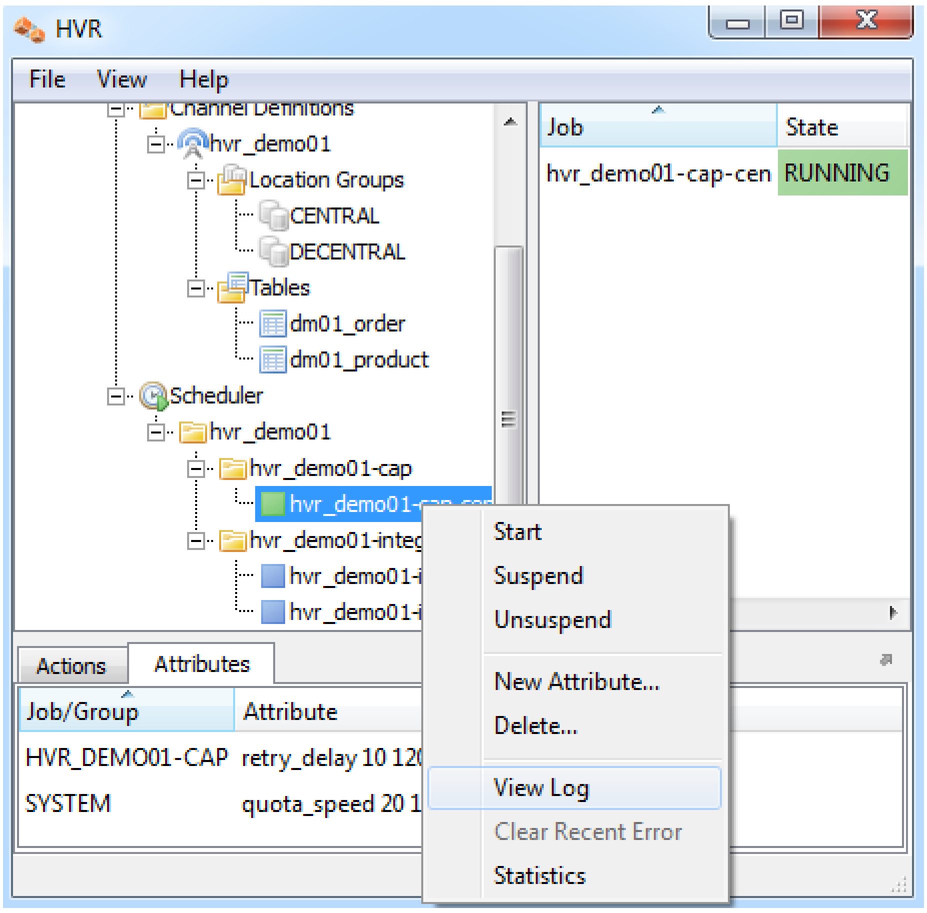
The other two jobs (**hvr_demo01–integ–db2** and **hvr_demo01–integ–db3**) pick up these transaction files and perform inserts, updates and deletes on the two target databases.

Test Replication

To test replication, make a change in **testdb1**:

```
db2 => connect to testdb1
db2 => insert into dm01_product values (1, 19.99, 'DVD')
db2 => commit
```

In the HVR log file you can see the output of the jobs by clicking on **View Log**. This log file can be found in **$HVR_CONFIG/log/***hubdb***/hvr_demo01–cap–db1**.



Job	State
hvr_demo01-cap-cen	RUNNING

Job/Group	Attribute
HVR_DEMO01-CAP	retry_delay 10 120
SYSTEM	quota_speed 20 1

The job output looks like this:

```
 hvr_demo01-cap-db1: Scanned 1 transaction containing 1 row (1 ins) for 1
table.
 hvr_demo01-cap-db1: Routed 215 bytes (compression=40.6%) from 'db1' into 2
locations.
 hvr_demo01-cap-db1: Capture cycle 3.
 hvr_demo01-integ-db2: Integrate cycle 2 for 1 transaction file (215 bytes).
 hvr_demo01-integ-db2: Integrated 1 change from 'dm01_product' (1 ins).
 hvr_demo01-integ-db2: Integrate used 1 transaction and took 0.004 seconds.
 hvr_demo01-integ-db3: Integrate cycle 2 for 1 transaction file (215 bytes).
 hvr_demo01-integ-db3: Integrated 1 change from 'dm01_product' (1 ins).
 hvr_demo01-integ-db3: Integrate used 1 transaction and took 0.013 seconds.
 hvr_demo01-integ-db3: Waiting...
```

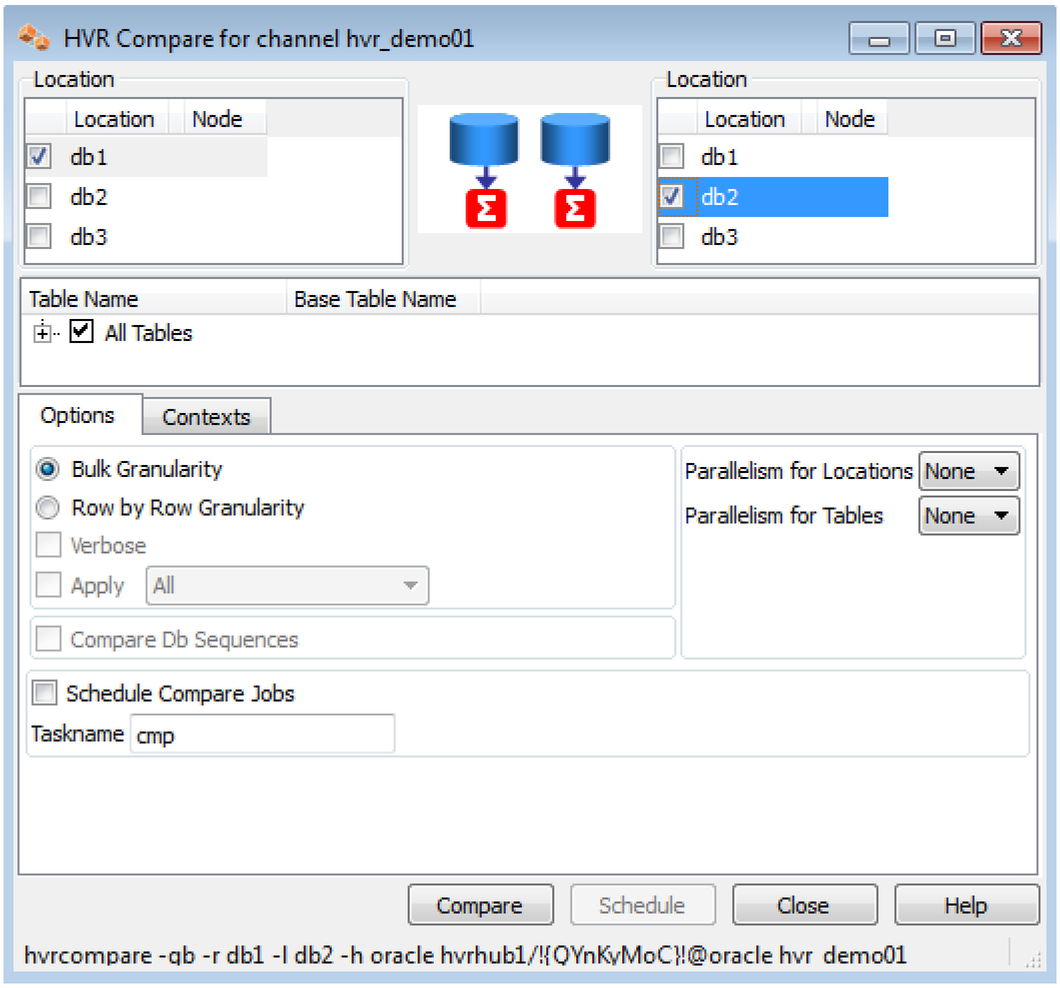
This indicates that the jobs replicated the original change to **testdb2** and **testdb3**. A query on **testdb2** confi rms this:

```
db2 => select * from dm01_product
```

prod_id	prod_price	prod_descrip
1	19.99	DVD

HVR Compare and Refresh

HVR Compare checks whether two locations have identical rows, and HVR Refresh copies the content of one location to the second location. In the HVR GUI, right-click on a channel and select **HVR Compare** (or **HVR Refresh**). Choose source and target locations.



HVR Compare for channel hvr_demo01

Location

	Location	Node
✓	db1	
	db2	
	db3	

Location

	Location	Node
	db1	
✓	db2	
	db3	

Table Name	Base Table Name
✓ All Tables	

Options | Contexts

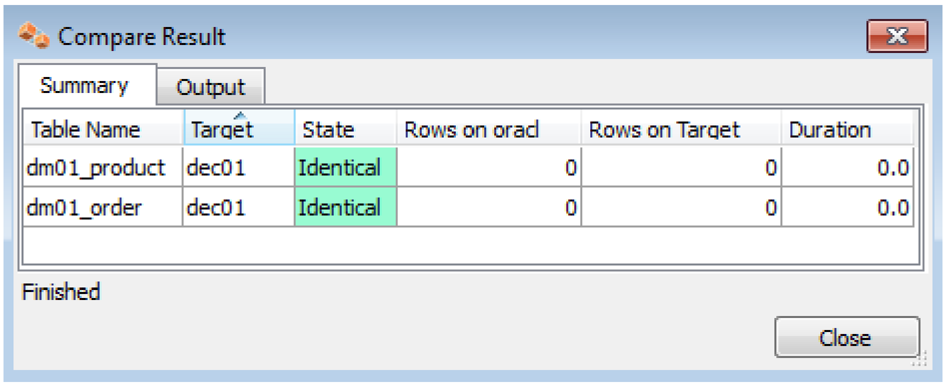
- ● Bulk Granularity
- ○ Row by Row Granularity
- ☐ Verbose
- ☐ Apply: All
- ☐ Compare Db Sequences

Parallelism for Locations: None
Parallelism for Tables: None

☐ Schedule Compare Jobs
Taskname: cmp

```
hvrcompare -gb -r db1 -l db2 -h oracle hvrhub1/!{QYnKvMoC}!@oracle hvr_demo01
```

The outcome of the comparison is displayed below;



Compare Result

Summary | Output

Table Name	Target	State	Rows on orad	Rows on Target	Duration
dm01_product	dec01	Identical	0	0	0.0
dm01_order	dec01	Identical	0	0	0.0

Finished