MATERIALIZED VIEWS IN CONVERIS

DEPLOYMENT GUIDE

Date of issue: Oct 30 2015

THOMSON REUTERS™

REVISION HISTORY

Name	Date	version	Summary of changes
Georgi Mechev	Oct 30 2015	0	Document Creation
Georgi Mechev	Dec 16 2015	1	Update
Georgi Mechev	Mar 08 2016	2	Update

REVIEW HISTORY

Reviewer Name	Doc Version Reviewed	Date Sent for review	Date Review Filed	Approved/Rejected (with Reasons)

Note	It is the author's responsibility to send this document out for review, making entries into columns 1-3

Note	It is the reviewers' responsibility to complete columns 4-5 and return the reviewed document to the author

CONTENTS

Intended readership 4
In this guide 4
Chapter 1 introduction 5
Chapter 2 Hardware and Software Requirements 6
Chapter 3 deployment 7
Creating MV Framework 7
Chapter 4 materialized view creation / Refresh 8
Chapter 5 JDBC resource 9
Chapter 6 limitations 10
Appendix A 11
refresh scripts 11

ABOUT THIS DOCUMENT

INTENDED READERSHIP

This document is intended for assisting in the deployment and maintenance of materialized views in Converis Servers on Postgres / Oracle database run on Linux OS.
Basic to middle knowledge of the Linux environment is expected.

IN THIS GUIDE

This deployment guide walks through a typical setup of materialized views in Converis server on Postgres / Oracle database and includes required commands for the Linux command line interface.

This document does not cover in-depth topics of the Postgres / Oracle database, as these are widely used and extensive documentation can be found on the Internet.

CHAPTER 1 INTRODUCTION

Up to Converis 5.5 our database had an easy to follow /human-readable data model. Entities were store in IOT_ tables, relations in REL_ tables. It was easy to fetch the attribute value for given Publication id f.i. Database model was changed in Converis 5.6. This leads to a different or more complex approach in querying the DB for reporting. As an intermediate solution for all current Converis clients on 5.5 and lower we introduced a solution on a DB level – materialized views.

Materialized view is a database object (you can think of it as a virtual table) that contains the results of a query.

After deploying materialized views, we have all entities and relations in IOT_ and REL_ tables as well language columns.

Main goals:

- ease migration efforts for legacy integrations (Converis 5.5 and lower)
- enable reporting without accessing Converis main DB (SLA constraints)
- ease simple reporting actions for end users
- enabling end users to get data from the database while 'hiding' the existing database model

In Postgres database materialized views are created in a new schema in the database – reporting and new user/role is created for accessing materialized views – reportviewer.

In Oracle database materialized views are created in a new schema/user (in Oracle schema and user are practically mutual, so the schema is user itself).

Thus views are separated from Converis tables in schema public (PG) or Converis schema (Oracle).

This solution is not intended to serve as a long term solution for reporting.

CHAPTER 2 HARDWARE AND SOFTWARE REQUIREMENTS

As Materialized views (MV) are created in a database used by existing Converis server, all hardware and software requirements for Converis server have to be fulfilled.
For details refer to the Converis installation guide on Postgres / Oracle database.

In order to speed up the 'create view' process, the Postgres parameter work_mem has to be set to 64MB. By default this parameter is not set. It could be found in postgres.conf file. After the change, postgres service must be restarted.
Without this parameter, time needed for MV creation is ~2.5 longer.

Parameter sgared_buffers also must be set appropriately. If you have a system with 1GB or more of RAM, a reasonable starting value for shared_buffers is 1/4 of the memory dedicated for Postgres in your system.

New login role will be created for MV access – reportviewer. This user must be allowed to connect to the database from the localhost - corresponding changes has to be done in pg_hba.conf file. By default it is enabled, but check are the following lines present in file –

```
host    all         all         127.0.0.1/32          md5

host    all         all         ::1/128               md5
host    all         all         0.0.0.0/0             md5
```

If these settings are changes, it has to be discussed with the customer that new line is needed in the file – allowing reportviewer to connect to the database locally.

CHAPTER 3 DEPLOYMENT

Login as the user under which credentials run database and extract the downloaded mviews.tar.gz file in the designated directory. Designated directory is the directory where all scripts for MV framework reside – f.i. /opt/converis/Postgres. This shouldn't be a temporary directory. OS user is usually 'postgres', but it could be some other user.

```
tar -xvzf mviews.tar.gz
```

CREATING MV FRAMEWORK

MV framework consists of schema reporting, user reportviewer, functions for MV create/refresh (and some additional functions for benchmarking/reporting) and all necessary grants to the user reportviewer.

Scripts for MV refresh a generated dynamically.

All scripts reside in directory **scripts**. Switch to the directory **scripts** and execute the script **create_user**

```
cd scripts
./create_user
```

The following information has to be provided.

```
Enter the full pathiname to psql( f.i. /opt/converis/postgres/9.3/bin/psql )
```

If the database is Oracle you will be asked to provide the path to sqlplus.

You have to provide the full path to the binary psql(including the psql, not only the directory where psql is). If no path is specified, default value /opt/converis/postgres/9.3/bin/psql will be used.

Similarly, if database is Oracle, you have to provide the full path to sqlplus; otherwise default value will be used

```
User reportviewer will be created. Please enter password for user reportviewer
```

Enter the password for the new user reportviewer.

```
Email for notification by MV refreshing problem
```

An email will be sent to this user in case of troubles by MV refresh.

After providing this information the schema reporting and user reportviewer will be created and all required privileges will be granted to reportviewer (for Oracle user reportviewer will not be created).

As a next step user reportviewer will logon to the database and will create the following function:

```
ent_mviews
rel_mviews
ent_mviews_refresh
rel_mviews_refresh
create_table_count_type
calculate_percent
```

Functions create_table_count_type and calculate_percent are used for creation of benchmark reports.

CHAPTER 4 MATERIALIZED VIEW CREATION / REFRESH

By execution of the script create_user, following scripts will be generated

mviews_refresh
ent_mviews_refresh
rel_mviews_refresh

mviews_refresh refreshes MV for both entities and relations.

Create a new cronjob which run at convenient time (recommended during the night) and executes the script mviews_refresh. F.i, to have the script executed every night at 1am, add the following to the crontab –

00 1 * * * /opt/converis/postgres/scripts/mviews_refresh

where /opt/converis/postgres/scripts is the path to the mviews_refresh (use your own path here, it would be probably different, especially if database is Oracle).

For more flexibility, there are two other scripts - ent_mviews_refresh – for refreshing entities MV and rel_mviews_refresh - for refreshing relations MV. They could be called with a parameter, thus only one or a few MV to be refreshed (refer to Appendix A).

Crontab job could be created for any of those script (similarly like for mviews_refresh) if separate refresh is preferred.

If MV are still not created, first execution of any of the refresh scripts will create the corresponding MV.

Time for creation/refresh vary from the number of entities and type/usage of the attributes.

It is recommended to create materialized views out of peak hours(please note that it depends again on the database size – by systems with relatively small number of entities, it would take just few minutes, so high CPU wouldn't be a problem).

CHAPTER 5 JDBC RESOURCE

Copy the file jdbc_pool to the machine where Converis Glassfish server is running and run the script - either as the user who run Glassfish server or as root.

As the user who run Glassfish server

./jdbc_pool

As root
sudo –u glassfish ./jdbc_pool
(here is assumed that the user is glassfish)

New jdbc resource converis-reporting and jdbc connection pool converis-reporting-pool are created for accessing the MV in schema reporting.

Do the same on the machine where DIS is deployed (if DIS and Converis are on different Glassfish servers/hosts).

CHAPTER 6 LIMITATIONS

MV's are not intended to be a long term solution for the reporting.
This solution is not recommended for huge databases, because of the long time needed for views creation/refresh. This makes MV not applicable for environments with huge amount of data which rely on real/close to real reporting results.

During the creation of the materialized views, there is more intensive CPU usage. It is recommended MV creation/refresh to be done during the night or out of the business hours as well to refresh only MV needed for reporting (see Appendix A for information how to refresh simple MV).

APPENDIX A

REFRESH SCRIPTS

mviews_refresh

This script is calling the functionis reporting.ent_mviews_refresh, reporting.rel_mviews_refresh which are refreshing all materialized views

ent_mviews_refresh

This script is calling the function reporting.ent_mviews_refresh which is refreshing the entities materialized views. If empty string is used as a parameter for the function reporting.ent_mviews_refresh, all MV will be refreshed. If one entity is sent as a parameter, only MV for this entity will be refreshed.

% can be used in the parameter - thus F% means that MV's for all entities starting with F will be refreshed

rel_mviews_refresh

This script is calling the function reporting.rel_mviews_refresh which is refreshing the relations materialized views.

If empty string is used as a parameter for the function reporting.rel_mviews_refresh, all MV will be refreshed. If one relation is sent as a parameter, only MV for this relation will be refreshed.

% can be used in the parameter - thus F% means that MV's for all relations starting with F will be refreshed