Actian Corporation

dbmv – Database Schema Conversion Tool

1.0 Dbmv.py

The dbmv.py tool helps you to move one database schema to another database if you have a "live" connection available to both source and target databases. ODBC is used both to manage metadata about the schemas and to transfer data (if required), so Python-accessible ODBC drivers are required on the execution machine.

1.1 Source Schemas

The source schema can be from any of the following database products:

- ORACLE Mysql
- MICROSOFT SqlServer
- ORACLE
- POSTGRES
- SAP Sybase IQ (V15+)
- TERADATA

Schema description can easily be extended to any database in the destination list. You must add missing description for "tbDefinition", etc., in file "dbmv.xml".

SQL output, field names, and SELECT descriptions must respect the pattern used for other existing databases.

1.2 Target Schemas

The target schema can be any of the following:

- ACTIAN VectorH, Vector

- ACTIAN Ingres
- EMC Greenplum
- IBM Netezza (UDB, not sure about iseries and Zos)
- IBM Netezza (4.xx ?)
- ORACLE MySQLl
- MICROSOFT SQL Server
- ORACLE
- POSTGRES
- PROGRESS Database
- SAP Sybase IQ (V15+)
- SAP Sybase ASA (Anywhere)
- SAP Sybase ASE
- SAP Hana
- SAP MaxDB
- TERADATA

1.3 Syntax

Without parameters, dbmv.py displays simple help about its syntax.

```
$ ./dbmv.py
```

```
usage: dbmv_new.py [-h] [--src SRC] [--dest DEST] [--loadata] [--cretab]
                   [--creview] [--creall] [--loaddl] [--loadtest]
                   [--batchsize BATCHSIZE] [--maxrows MAXROWS] [--truncate]
                   [--parfile PARFILE] [--fdelim FDELIM] [--unload]
                   [--translation TRANSLATION] [--quote QUOTE]
                   [--cmdsep CMDSEP] [--charmax CHARMAX] [--creindex]
                   [--ownsrc OWNSRC] [--owntgt OWNTGT] [--add_drop]
                   [--on_error ON_ERROR] [--source_schema SOURCE_SCHEMA]
                   [--target_schema TARGET_SCHEMA] [--unsupported]
                   [--exclude EXCLUDE] [--include INCLUDE] [--tables TABLES]
                   [--insertmode INSERTMODE] [--trial]
                   [--loadmethod LOADMETHOD] [--threads THREADS]
```

Optional arguments are as follows:

-h, --help

Shows the help message.

--on_error=*ON_ERROR*

Specifies action to take when an error occurs: **continue** or **abort.**

--src=*SRC*

source DB used as a data source for migration, copy, etc.. For more information, see Specifying Source and Destination Databases.

--dest=*DEST*

Destination DB used as a destination for migration. Copy , etc. For more information, see Specifying Source and Destination Databases.

--cretab

Creates only tables in the @dest from @source schema.

Note: Use cretab and creall to build a full schema.

--creview

Creates only views in the @dest from @source schema.

Note: Several iterations may be required if nested views exist. Depends on how the source database returns the definitions.

--creindex

Creates only indexes in @dest schema.

--creall

Creates all objects of type INDEX, PRIMARY KEY, REFERENTIAL CONSTRAINTS, and VIEWS in a @dest from @source schema.

Note: creall does **not** create objects of type TABLE; use cretab and creall to build a full schema. Such an operation may perform slower when loading data into the tables in addition to creating all the objects.

--loaddl

Loads DDL from @source to @dest. If not specified, a file containing all objects is created in the current directory. In this case, you can reload.

--add_drop

Drops table at the beginning.

--truncate

Removes existing rows from @dest table.
Default: False

--unsupported

Whether to skip unsupported types or not.
Default: False

--loadata

Indicates whether to load the table data from @source to @dest. For more information, see Unload and Load Capabilities.

--loadtest

Performs predefined INSERT queries to @dest (used for performance testing).
Default: False

--batchsize=*BATCHSIZE*

Sets custom batch size for INSERT queries.

Default: 500

Limits: 1 to 10000

--maxrows=*MAXROWS*

Sets total limit for INSERT queries.

Default: 100000

Limits: 1 to 100000

--parfile=*PARFILE*

Specifies a file with dbmv parameters. Parameters are stored as a parfile. Parfiles examples can be found in the 'wrk' directory.

--fdelim=*FDELIM*

Sets field delimiter for data or statements written to file. Used for data unloads.

Default: \t

--unload

Unloads table structure from @source to file as CREATE statements. For more information, see Unload and Load Capabilities.

--translation=*TRANSLATION*

Uses translation table/constraints/indexes rules in @dest. Use `schema-name` translation pairs for tables, constraints, and indexes.

Example:

```
'''--translation=scname:public,vw_user,postgres,vw_user;
iscname:public,vw_user,postgres,vw_user; rscname:public,vw_user,postgres,vw_user'''
```

Explanation:

scname: Tables owned by "public" are moved to schema "vw_user", and from "postgres" to "vw_user".

iscname: Indexes owned by "public" are moved to schema "vw_user" and from "postgres" to "vw_user".

rscname: Constraints owned by "public" are moved to schema "vw_user" and from "postgres" to "vw_user".

--quote=*QUOTE*

Uses specified symbol as quotation symbol for DDL. Example:

```
--quote='"' => CREATE TABLE "my_table"
```

--cmdsep=*CMDSEP*

Specifies command separator used in DDL file.

Default=';'

Examples:

```
--cmdsep='\g'
```

```
--cmdsep='go'
```

--charmax=*CHARMAX*

Specifies maximum column precision that is acceptable for translation.

Default: 6400

Limits: 1 to 30000

--ownsrc=*OWNSRC*

Specifies single source owner.

--owntgt=*OWNTGT*

Specifies single new target owner.

--source_schema=*SOURCE_SCHEMA*

Specifies @source schema to process.

--target_schema=TARGET_SCHEMA

Converts @source_schema to the specified schema.

--exclude=*EXCLUDE*

Excludes specified tables or columns from processing.

--include=INCLUDE

Includes only specific tables or columns for processing.

--tables=TABLENAME

Process only specified tables

--insertmode=*INSERTMODE*

Specifies mode for inserting data:

bulk - Insert data in chunks (buffered insertion) (Default).

row - Insert data row-by-row, ignoring batchsize.

--trial

Tests the operation (trial mode).

--loadmethod=*LOADMETHOD*

Specifies number of threads to use when loading data:

serial - 1 thread (Default)

parallel - 4 threads

multitable - CPU dependent

--threads=*THREADS*

Specifies custom number of threads to use when loading data.

Default: 1

Limits: 1 to 32 (CPU dependent)

1.3.1 Specifying Source and Destination Databases

The --src and --dest arguments are specified as follows:

```
--src=URL
--dest=URL
```

where

```
URL: db_driver[-odbc]://db_host[:port]/dbname?db_login&login_password
```

And where db_driver can be one of the following driver names:

Driver Name	Product
vector	ACTIAN Vector
vectorh	ACTIAN VectorH
ingres	ACTIAN Ingres
greenplum	EMC Greenplum
netezza	IBM Netezza
mysql	ORACLE MySQL
mssql	MICROSOFT SQL Server
oracle	ORACLE
postgres	POSTGRES
progress	PROGRESS Database
iq	SAP Sybase IQ (V15+)
asa	SAP Sybase ASA
ase	SAP Sybase ASE
hana	SAP Hana
maxdb	SAP MaxDB
teradata	TERADATA

1.3.1.1 About ODBC

Linux:

The path and configuration for the ODBC database driver must be added in file "../etc/driverTools.xml". (See in-file examples.)

Windows:

ODBC drivers and Datasource must exist. No additional configuration is required. The DSN is provided in place of db_host:

```
URL : db_driver[-odbc]://DSN[:port]/dbname?db_login&login_password
```

Note: port and dbname are required but ignored for Windows DSN use case.

When using Azure and Microsoft SQL Server:

```
--src=mssql-
odbc://your_azure_access.database.windows.net/dbname?db_login&login_password
```

Note: The .database.windows.net is a generic value provided by Azure.

On all operating systems, Microsoft SQL Server ODBC driver Version 13 or higher is required.

1.3.2 Unload and Load Capabilities

The --unload and --loadata parameters provide basic facilities for unloading and loading data. These operations are inefficient for massive data but can be useful for small tables (less than 100,000 records).

1.4 Supported Python Drivers

Supported Python drivers are as follows:

Module	**Product**
pyodbc	ODBC (Used for many databases including Actian Vector, VectorH, Ingres, Microsoft SQL Server on premises, and Azure.)
cx_Oracle	Oracle
Sybase	Sybase ASE
pymssql	MS-SQL
MySQLdb	My SQL
psycopg2, psycopg2.extensions	Postgres
import DB2	DB2

1.5 Logging

The dbmv tool provides JSON logging and is configured to use a relative path by default from the DBMV_HOME/bin folder.

To override this, use the LOG_CFG environment variable. Define the LOG_CFG variable to be the full path to the JSON logging configuration file:

Windows:

```
set LOG_CFG=C:\Actian\dbmv\etc\logging.json
```

Linux:

```
export LOG_CFG=/opt/Actian/dbmv/etc/logging.json
```

1.6 Examples

To run correctly, all parameters must be on one line. For clarity, the parameters are shown here on separate lines.

1. Load all tables for dbo schema, excluding columns not required, specifying the source and destination in the parameter file ms_vw.par.

   Note: The image columns and binary datatype columns are automatically ignored for Vector and VectorH destinations.

   ```
   C:\DBMV_HOME> python.exe bin\dbmv_new.py
   --quote=\"
   --parfile=C:\DBMV_HOME\wrk\ms_vw.par
   --cretab
   ```

```
--source_schema=dbo
--maxrows=3000
--target_schema=vw_user
--batchsize=700
--loaddl
--add_drop
--loadata
--exclude="Employees"."Photo","Categories"."Picture"
--threads=10
--maxrows=2000
--creall
```

2. Add additional views if there were load failures.

   Note: Load failures can happen when some views rely on others and the migration order is not aligned to the creation of the views. This is a small limitation of dbmv, depending on the source database.

```
C:\DBMV_HOME> python.exe bin\dbmv_new.py
--quote=\"
--parfile=C:\DBMV_HOME\dbmv\wrk\ms_vw.par
--source_schema=dbo
--target_schema=vw_user
--loaddl --exclude="Employees"."Photo","Categories"."Picture"
--creview
```

EXAMPLE of Oracle odbc based source
--src=oracle-odbc://SD02/SD02?db_login&login_password

EXAMPLE of Vector Based destination
--dest=vector-odbc://topcoder64:II/topcoder?db_login&login_password

2.0 Your Support Options

Enterprise customers with active maintenance and support contracts have full access to Actian Support, including online use of our case management system and knowledge base, at the Customer Portal (https://communities.actian.com).

If you have an active support contract and want to register for access to the Customer Portal, use the enrollment form at http://supportservices.actian.com/user/register.php. (Your Account Number ID is required.)

If you do not have a support agreement for Actian Corporation products and are interested in purchasing support, contact us at sales@actian.com.

For more information about support options, visit https://www.actian.com/support-services/.