



Scott Auge
sauge@amduus.com

Amduus Information Works, Inc.
<http://www.amduus.com>

LICENSE

This is your typical BSD license. Basically it says do what you want with it - just don't sue me.

Written by Scott Auge scott_auge@yahoo.com
Copyright (c) 2006 Amduus Information Works, Inc.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement:
This product includes software developed by Amduus Information Works and its contributors.
4. Neither the name of Amduus nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AMDUUS AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AMDUUS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Contributors

Peter Brown, pbrown at beasleyfoodservice. com

Updates to handle arrayed fields.

Shane Dunn, shane_dunn at debortoli. com. au

Updates to handle arrayed fields in schema and data

Updates to handle output in SQL or CSV for data dumps

Updates to handle cleaning % in table and field names

Updates to handle null dates

Introduction

This document is oriented towards Progress and PostgreSQL programmers and database administrators familiar with working in the UNIX operating system environment.

This is a conversion kit to aid with moving data from a Progress RDBMS to a PostgreSQL RDBMS. It has tools to migrate database schemas as well as to move data from a Progress database to the PostgreSQL database.

There are some test routines you can view to see actual testing that has occurred on the system to use as examples for your own scripts using the tool kit.

Be sure to see the CHANGELOG file found in the base directory of the ZIP file to see what has changed between builds.

Amduus uses a build number system to keep track of versions. Basically it is made up of the following:

name.yyyyjjhhmmss.zip

where

name = name of the package

yyyy = year of the build

jjj = day of the build (1-365)

hh = hour of the build

mm = minute of the build

ss = second of the build

Creating A Schema To Load Into PostgreSQL

The first step to moving data from the Progress DB to the PostgreSQL database is to transfer the Progress database's schema into the PostgreSQL database cluster.

This is done with the `cnv.bash` script. The `cnv.bash` script expects the following arguments:

<code>-output</code>	The file to put the schema SQL into
<code>-pf</code>	The Progress pf file to use to connect to the Progress DB
<code>-dlc</code>	Where to find Progress
<code>-owner</code>	Who should own the file in PostgreSQL
<code>-db</code>	Name of the DB in PostgreSQL
<code>-ini</code>	Choose an ini file (defaults to <code>config.ini</code> in <code>PROPATH</code>)

The program can only handle connecting to one database at a time. If you have multiple databases you wish to transfer, you must connect to them individually.

An example call can be seen in the `src/tstcnv.bash` script:

```
cnv.bash -dlc /usr/dlc -pf /db/amduus/client.pf -output /  
tmp/db.sql -db test1 -owner sauge
```

The above command will create a `/tmp/db.sql` file that can be used to create the PostgreSQL database in the database cluster.

Note that in the `config.ini` file, there are many other configuration options. See the later section and notes in the `config.ini` file for more information.

Command To Load Table Schema

Once one has a schema, one will need to load it into the PostgreSQL database cluster so that you can use it with PostgreSQL. A quick way to do this is with a command similar to:

```
psql -d template1 < /tmp/db.sql
```

You must execute this command as a PostgreSQL privileged user, that is one who can manipulate the SCHEMA.

Whats up with amduus_rsvp?

This field has two reasons for existing. One was this software came out of my work that required another field. The other reason is that it makes the “,” logic in the CREATE TABLE routines work out quite nicely.

Creating A Data Dump To Load Into PostgreSQL

Once you have the schema prepared, you will need to export data from the Progress database into the PostgreSQL database. This can be done with the `export.bash` script. This script expects the following arguments:

<code>-output</code>	File to place the SQL into
<code>-pf</code>	Progress PF file to use to connect to the Progress DB with
<code>-dlc</code>	Where to find Progress
<code>-table</code>	Name of table to export
<code>-where</code>	Where clause applicable to a FOR EACH
<code>-delete</code>	Set to yes or no to include a DELETE FROM table statement to reset
<code>-ini</code>	Choose an ini file (defaults to config.ini in PROPATH)

The program can only handle connecting to one database at a time. If you have multiple databases you wish to transfer, you must connect to them individually.

An example call might be:

```
export.bash -dlc /usr/dlc -pf /db/amduus/client.pf -output /  
tmp/customer.sql -table customer -where "update_date > TODAY - 1"  
-delete
```

This invocation would create a file of SQL in `/tmp/customer.sql` that can be loaded into the PostgreSQL database. It generates data from the customer table and selects only those fields where the `update_date` is greater than yesterday.

Command To Load The Table Data

After creating a data extract file, you can apply that file to the PostgreSQL database like any other SQL file with the psql command.

For example:

```
psql -d costacct < /tmp/customer.sql
```

You would need to do this as a privileged user with INSERT and DELETE permissions on the customer table.

Programmers API

Currently the only supplied code is for a CLI (Command Line Interface) to the conversion routines.

These routines have been made into APIs so as GUI, CHUI, or WWW developers can create those interfaces to the routines. This helps with developing on Windows.

cnv2pgsql.p

This routine will take the files found in the currently connected Progress database and create an output file of SQL to be applied to a PostgreSQL database cluster to create a new database.

It's arguments are:

```
DEFINE INPUT PARAMETER cOutputFile AS CHARACTER NO-UNDO.  
DEFINE INPUT PARAMETER cOwner AS CHARACTER NO-UNDO.  
DEFINE INPUT PARAMETER cDbName AS CHARACTER NO-UNDO.
```

The output file is where you wish the SQL to be written on the machine running the code.

The Owner is who would own the database and tables in the PostgreSQL cluster.

The DBName is what you wish the database to be named in the PostgreSQL cluster.

The final parameter cConfigIni is used to define where the config.ini file is found. It is up to the implementing programmer default this value.

This routine takes additional information from the config.ini file.

exportsql.p

This routines will generate SQL insert statements from the data found in the Progress DB it is run against.

It's arguments are:

```
DEFINE INPUT PARAMETER cOutputFile AS CHARACTER NO-UNDO.  
DEFINE INPUT PARAMETER cTable      AS CHARACTER NO-UNDO.  
DEFINE INPUT PARAMETER cWhere      AS CHARACTER NO-UNDO.  
DEFINE INPUT PARAMETER cDelete     AS CHARACTER NO-UNDO.  
DEFINE INPUT PARAMETER cConfigIni  AS CHARACTER NO-UNDO.
```

The output file is where the SQL will be written to. This is the file that should be loaded into PostgreSQL to actually move the data.

The routine is designed to work with one table at a time so one can be judicious in it's use – often one doesn't need to repeatedly export data from every table in the Progress DB.

The where clause (as found in a FOR EACH) is used to help isolate those records that will be put into INSERT statements.

The Delete is a flag of values “yes” and “no” as values. It is used to determine if the ENTIRE TABLE SHOULD BE EMPTIED in the PostgreSQL database before the inserts are applied. *A value of yes will delete the table contents and a value of no will leave the table contents.*

The final parameter cConfigIni is used to define where the config.ini file is found. It is up to the implementing programmer default this value.

This routine takes additional information from the config.ini file.

The config.ini file

The config.ini file can be used to control the base operating parameters of the conversion routines.

```
# Various configurations

# This is used to map from Progress types to PostgreSQL types

[MapTypes]
INTEGER=int
CHARACTER=varchar
LOGICAL=boolean
DECIMAL=float(35)
DATE=date

[Logs]
ErrorLog=/tmp/pro2pgsql_error
ProcessLog=/tmp/pro2pgsql_log

[Transformations]
CleanName=yes
Encoding=LATIN1
OutputSQL=yes

[ArrayConversions]
Type=Array
```

The first section are MapTypes. This is what data type the Progrss type will be converted to in the PostgreSQL version of the table. Note that when a varchar is used without a limit, it is generally unlimited in PostgreSQL 8.0+. Be sure to see the PostgreSQL documentation for more information.

The next section are for Logs. There are logs kept of what is happening in the routines. You can set these here.

The Transformations section is very important. Some Progress Dbs have illegal SQL characters in them. The CleanName parameter is used in the `cleanname.i` code – you will see that it:

- Converts all # into “no” in the object name.
- Converts all – into _ in the object name.
- Converts all the upper case letters into lower case letters in the object name.
- Converts all % into pcnt in the object name.

The encoding option determines the character set that the DB will be using. See the PostgreSQL documentation for more information about this.

The `outputsql` option is used for exporting data. If `OutputSQL` is yes, then SQL statements will be created. Otherwise a CSV file will be created.

The section, `DateConversion`, helps handle the ? In Progress for data values. You can use it set the value to null, a special PostgreSQL value, or a ?, or a defined date.

The following section, `ArrayConversions`, determines how to handle array fields in the export of the schema or the data. When `Type=Array`, then progress arrays will be transformed into PostgreSQL arrays. When `Type=SingleField`, then the array will be expanded out to individual fields prefixed with the field name + “_” + extent number, like `Month_01`, `Month_02`, `Month_03`, etc.

Troubleshooting

Problem: I receive an error on creating an index:

```
ERROR:  syntax error at or near ")" at character 55
LINE 1: CREATE  INDEX testbed_default ON testbed USING btree ();
                                     ^
ERROR:  relation "public.testbed_default" does not exist
```

Solution: All tables **MUST** have an index on them.

Changes, Updates, And Omissions

This document is provided in PDF and OpenOffice format. If you see some changes that can be added to it, please send your updated version to the author Scott Auge sauge@amduus.com.

If the source code can be tidied up or additional goodies added to it, be sure to send those to the author also for consideration and insertion to the next kit release.

If you find a bug, I think a lot of us using the tool would appreciate you attempting to find the problem and correcting the code. Then send the code along to the author for consideration and inclusion for the next release.