

# Cross-Database Copying in PostgreSQL

Published [Jan 16, 2006 @ 14:25:16](#)

## The Need

Often times it is necessary to copy data from one database to another, but not necessarily make a full copy. A good real-world example of this is when a new piece of software has been developed to replace an older system. A big issue in the final "go-live" is converting existing data over to your new system. You can write a bunch of tiny script in your favorite language to do the task, but that's often a lot of needless work when the structure of the data is similar, but not close enough to do straight alter statements on a database to get it up to speed.

Often times, in this situation, the ideal thing to do is simply create a new database and simply use sql (or plpgsql) to copy data from the old system into the new one. For example: If we have a table in database **a** named employees, and a table in database **b** named employees, it would be nice to be able to simply do this (given that both tables resides in their respective databases's "public" schema):

```
insert into a.public.employees
select * from b.public.employees;
```

However, PostgreSQL does not allow cross-database queries like this. The solution to this problem (at least the one I use) is simpler then you would think.

## The Steps

Basically, what we need to do is emulate cross-database queries. This is easier then it sounds. Here are the basic steps we're going to take:

1. Create a new database and populate it with your new database schema and objects (I'll call it 'target' database).
2. Create another new database as a copy of the old source database (I'll call this the 'source' database, as it is an exact copy of the real source)
3. Alter the tables in the source database putting their tables in a different schema.
4. Dump the objects out of that specific schema to disk (using `pg_dump`).
5. Import the dump into the target database. All the old data will now be in the target database, but under a different schema.
6. Execute the steps necessary to copy data from the old schema to the new table structure.
7. Drop the old schema from the new target database.
8. Drop the new-source database (the one created in step 2.)

## Steps In Detail

Here is a sample of all the steps listed above. Refer the number above for an explanation of what this is doing:

```
# Databases used:
source=ProdDB1
tmp_source=ProdTmp1 (a copy of the real source -- we don't actually want to touch the old source database)
target=ProdDBTarget (the final destination for the data)

# login to the psql command line
$ psql -Umyuser ProdDB1

# Step 1
# Create a new 'target' database:
psql> create database ProdDBTarget;

# Step 2
```

## Cross-Database Copying in PostgreSQL

```
# For this to work, all users must be out of ProdDB1
# if this isn't an option, you can create the new database
# and populate it in the typical pg_dump/pg_restore fashion.
pgsql> create database ProdTmp1 template = ProdDB1;

# connect to the newly created database.
pgsql> \c ProdTmp1

# Step 3
# create a new schema
pgsql> create schema oldsys;

# create a bunch of alter statements to move tables into the new schema
pgsql> select 'alter table ' || tablename || 'set schema oldsys;'
        from pg_tables where schemaname = 'public';

# You'll want to copy these statements and run them:
pgsql> alter table employee set schema oldsys;
...

# Exit psql command line
exit;

# Step 4
# From the shell, run:
$ pg_dump -n oldsys ProdTmp1 > oldsys.sql

# Step 5
# Again, from the shell, run:
$ psql ProdDBTarget < oldsys.sql

# Step 6
# You can now copy data from the old tables
# (now stored in the schema 'oldsys') to the
# new tables. These command would be specific to what you need
# to do:
$ psql -Umyuser mydatabase
pgsql> insert into tday
    > select * from oldsys.tday;

pgsql> insert into tmonth
    > select * from oldsys.tmonth;
# ... and the rest of your commands

# Step 7
# Once you're finished, you can drop all the old data
# out of the new database:
pgsql> drop schema oldsys cascade;

# Step 8
# Drop the temporary database:
pgsql> drop database ProdTmp1;
```