Exporting Postgres DB to SQLite DB

This is a step-by-step procedure for exporting all tables and data from a Postgres database to a new SQLite database:

This procedure assumes that there is a fully populated Postgres database running on a specific port and that you are logged into the server as a user that has "***super user***" permissions to the Postgres database. This procedure is using code and functions stored in the GitHub tomcat repository - https://github.com/ml4ai/tomcat/tree/master/human_experiments/datasette_interface
It is recommended that after you ***ssh*** into the server where the Postgres DB lives, you run ***git pull*** to update to the latest code in the repository, check to make you have enough available disk space where the SQLite DB is going to be created, and enter a ***tmux*** session before running any of the ***datasette interface*** commands.

Installed apts used in this procedure:

- **tmux**
- **mibicondo3**

Examples used in this procedure are for a Postgres database running on ***gauss*** and that the SQLite database will be created on gauss.

1. **SSH to the server where the Postgres DB lives using a user account that has "*super user*" permissions DB:**

   ```
   ssh gauss
   ```

2. **CD to the directory where the local repository is cloned and run a "*git pull*" to get the latest code:**
   Example:

   ```
   cd ~/tomcat/human\_experiments/datasette\_interface
   git pull
   ```

3. **Start or attach to a "*tmux session*":**
   (You may have to install "***tmux***" on the server if it is not already installed)
   It is recommended to use an app like "***tmux***" so that if you get disconnect from the server while running a long-running ***datasette interface*** command", the command will continue

to run in a "***tmux session***" and you will be able to reconnect to that "***tmux session***" later.

```
# Install tmux if not already installed on server:
apt install tmux

# Start a new tmux session:
tmux

# List current tmux sessions running:
tmux ls

# Attach to a running tmux session by session\_number:
tmux attach -t <session_number>
```

4. **Verify that the Postgres DB is running on your specified <port>:**
   (If the Postgree DB is not started, start it)

```
### Example Variables:
port="5436"                                  # Port the DB is running on.
run_path="/fast/rchamplin/postgres/run"              # Path to DB run directory.
db_path="/fast/rchamplin/postgres/data/11/rchamplin" # Path to the database directory.
db_name="tomcat"                             # The DB name.
pg_ctl_path="/usr/lib/postgresql/11/bin"             # Path to where pg_ctl is installed on server.

# Check to see if you can connect to the running Postgres DB by command line:
psql -p $port -h $run_path -d $db_name
### Example output:
### psql (11.20 (Debian 11.20-0+deb10u1))
### Type "help" for help.
### tomcat2-#
\dt<return> # to list tables.
\q<return>  # to exit command line interface.

# If the Postgres DB is not running, start it:
$pg_ctl_path/pg_ctl -D $db_path -l logfile start
```

5. **Using "*minicondo3*", activate the tomcat-database environment:**
   (if "***minicondo3***" is not installed, install it and create a tomcat-database environment)

```
### <datasette_interface_clone_path> = Path to where "datasette_interface" github code is cloned.
### Example: ~/tomcat/human_experiments/datasette_interface
```

```
# If not already installed, install "minicondo3" on server:
mkdir -p ~/miniconda3 wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86\_64.sh
-O ~/miniconda3/miniconda.sh
bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3
rm ~/miniconda3/miniconda.sh

# After installing minicondo3, tell Linux to source ~/.bashrc
# so that it will have PATH to condo commands:
source ~/.bashrc
cd <datasette\_interface\_clone\_path>
# condo will now change prompt to show what environment is active (defalt is base)
# Example: (base) rchamplin@gauss:~/tomcat/human\_experiments/datasette\_interface$

# If not already created, create a "conda" python=3.9 environment named "tomcat-database":
cd <datasette\_interface\_clone\_path>
conda create --name tomcat-database python=3.9
### Example output:
### Retrieving notices: ...working... done
### WARNING: A conda environment already exists at '/work/rchamplin/miniconda3/envs/tomcat-database'
### Remove existing environment (y/\[n\])? n
### *** To remove a conda environment: conda remove -n tomcat-database --all ***

# Activate the conda "tomcat-database" environment:
source ~/.bashrc
cd <datasette\_interface\_clone\_path>
conda activate tomcat-database
### Example output:
### (tomcat-database) rchamplin@gauss:~/tomcat/human_experiments/datasette_interface$
```

6. **Install all requirements from "*requirements.txt*":**
   The "*requirements.txt*" file is created when the "tomcat-database" conda environment is created. You install this file on the server using *pip install*. The files installs any prerequisites needed to setup your "tomcat-database" environment.

```
pip install -r requirements.txt
```

7. **Export all postgres tables to SQLite db:**

```
### For "TBS=" you can specify a list of tables, by name, to export:
### db_pass="tomcat" db_port=5436 db_name="tomcat" working_env="production"
```

```
### TBS="affective_task_event, audio_vocalics, data_validity, eeg_device, eeg_raw, eeg_sync,
ekg_sync,
### finger_tapping_task_observation,fnirs_raw, fnirs_sync, gaze_raw, group_session, gsr_sync,
### minecraft_mission, minecraft_testbed_message, modality, participant,
### ping_pong_competitive_task_observation, ping_pong_cooperative_task_observation,
post_game_survey,
### rest_state_task, screen_capture, station, task"
### make to_sqlite

### Or, you can specify "all" for all tables:
### db_pass="tomcat" db_port=5436 db_name="tomcat" working_env="production" TBS="all"
make to_sqlite

db_pass="tomcat" db_port=5436 db_name="tomcat" working_env="production" TBS="all" make
to_sqlite
```

Revision #62
Created 10 April 2025 20:41:07 by Rick Champlin
Updated 6 November 2025 14:15:13 by Rick Champlin