# RepoDbSync

Helps to maintain (synchronize) local development database instances and one main database storage in a shared source code repository (in form of database creation SQL scripts). Each run does these 2 steps:

1. Applying scripts from the repository to your local database.

2. Helping with creating scripts in the repository which reflect changes in your local database.

Applying of these 2 steps (maybe in more iterations) makes the local database synchronized with the repository.

**Version:** 0.3

**License:** Public domain. Use at your own risk. There is absolutely no warranty of any kind.

## BUILD

* implementation language: C++11
* does not depend on any platform
* latest source code available at:<https://github.com/milan11/repodbsync>
* source code distribution: as CMake project - uses (and includes) cotire CMake module: <https://github.com/sakra/cotire>

build dependencies:

* boost

build steps:

* git clone git@github.com:milan11/repodbsync.git
* cd repodbsync
* cmake
* make

possible CMake arguments:

* **debug build:** -DCMAKE\_BUILD\_TYPE=Debug
* **force using gcc compiler:** -DCMAKE\_CXX\_COMPILER=g++
* **force using clang compiler:** -DCMAKE\_CXX\_COMPILER=clang++
* **build executable running tests instead of the normal executable:** -DTESTMODE=1

## SUPPORTED ENVIRONMENT

* database support: MySQL, PostgreSQL, SQLite
* OS support: Linux tested only (but should be multiplatform)
* depends on these database tools (must be in path):
  + for MySQL: mysql, mysqldump
  + for PostgreSQL: psql, pgdump
  + for SQLite: sqlite3
* needs the following libraries (is dynamically linked with):
  + boost (system, filesystem)

## SHORT USAGE DESCRIPTION

Just run it in some source code repository directory where you want the sql scripts and some configuration to be placed. It will guide you through the configuration and synchronization process. The executable does not process any arguments.

After following each instruction printed by RepoDbSync, you will have to run it again. It will continue in the process of setup and synchronization. After everything is correctly set up and after the databases are synchronized, it should print nothing and quit immediately.

Please note that the following descriptions refer to the state when everyone already has their development database in some state. However, the instructions can be applied to the cases of creating new development databases, too. Just imagine that you already have a development database, but it is empty.

### A. I have a local development database and want to put its state to a shared repository (e.g. a GIT repository) to share/synchronize the database with others (who maybe have their own development databases already).

#### 1. Create an empty directory in your repository working copy.

* this directory will serve as a main source for the database state
* if you have more development databases (e.g. one for a core system and one for a web interface), you have to deal with each database separately, thus creating such directory in your repository for each database

#### 2. Set up (fill the configuration file).

* run the RepoDbSync in that empty directory for the first time
* it will complain about missing settings
* fill the settings file; see: C. Common instructions - settings file

#### 3. Set up database versioning.

* run the RepoDbSync with settings filled in; see: D. Common instructions - database versioning

#### 4. Add database scripts to the repository.

* see E. Common instructions - adding database scripts to the repository

#### 5. Commit.

* commit the directory dedicated to the database to the shared repository
* the directory now contains scripts (in the "scripts" directory) which can be used to create new database in the desired state or to synchronize an existing database

### B. Someone has already put their repository state to our shared repository. I want to synchronize my own development database to reflect the state in the repository.

#### 1. Checkout the repository and find the database directory.

* the database directory should contain a directory called "scripts" and maybe .gitignore, ignore\_data.txt and ignore\_tables.txt files

#### 2. Set up (fill the configuration file).

* run the RepoDbSync in that directory for the first time
* it will complain about missing settings
* fill the settings file; see: C. Common instructions - settings file

#### 3. Set up database versioning.

* run the RepoDbSync with settings filled in; see: D. Common instructions - database versioning

#### 4. Apply scripts to your local database.

* review the scripts you checked out from the repository
* RepoDbSync will ask you if it can apply each script automatically; but maybe you will have to make some changes in your local development database manually and let the RepoDbSync increment the database version only
* after all scripts are applied, RepoDbSync helps you to add scripts reflecting your own database changes to the repository; see: E. Common instructions - adding database scripts to the repository

#### 5. Commit.

* commit new .sql scripts to the shared repository

## MANUAL

### Settings file

* you have to set up two databases:
  + the "local" database is your development database
  + the "temp" database is a (empty) temporary database - it will be used by RepoDbSync for simulating a database reflecting the repository state and will be cleaned before each run (so be careful to set it to an empty database; not to some database you are using for something)

### Database versioning

* if your local development database does not contain version information, the tool will ask you if it can add the version information - that is adding one table to your database where the version info will be stored

### Adding database scripts to the repository

* RepoDbSync will tell you what add to the repository so that the repository will exactly reflect your local development database
* you will have to repeat this step until RepoDbSync prints nothing (RepoDbSync prints nothing only if everything is in a correct, synchronized state - if the repository exactly reflects your local development database)
* note the contents of the "outs" directory after each run, there are some created scripts that you can use when creating the scripts in the repository
* the scripts must be named in the following way: \_.sql
  + version is the version which will the database have after applying this script (a number must be aligned to six digits, first version is 000001, no version can be left out)
  + description is your own description (e.g. create\_table\_user, fill\_users, delete\_table\_user)
* note that after adding scripts to the repository and running RepoDbSync again, you will have a version mismatch (the repository version will be higher that your local development database version)
  + RepoDbSync asks you, if you want to apply the repository scripts to upgrade your local development database to the repository state
  + because you already have the desired state provided by the scripts in your local development database (because you created the scripts from the state of the database), you can ignore this mismatch and answer with "nall" (this only sets the version of the local development database to the version provided by the scripts)
* of course, you will have some specific tables or data in your local development database, which you do not want to share (which do not form the base database state), this can be e.g. data about users which you have added while testing etc.
  + list tables you want to ignore in the ignore\_tabes.txt file (one table name on one line)
  + list tables which data have to be ignored in the ignore\_data.txt file (one table name on one line)
    - additionally, you can list conditions for ignored data (e.g. "user id = 1 OR id > 4") will ignore records in the table user which have ID 1 or ID greater than 4
  + the \*local.txt files are there for the ignore lists which you do not want to share in the repository

## TODO

* more concise output (maybe detailed help shown on user demand)
* provide downloadable binaries
* real-world examples (e.g. handling a standard Drupal database)