September 2015 Features Update

Table of Contents

Illustration Index

Global

The biggest change this month is under the hood. When I started this project several years ago there was no standard way to access SQL databases from the PHP language the site is written in. I used one of the most functional technologies that was available at the time, PEAR MDB2, but now a powerful standard interface has been established, PHP Data Objects (PDO), so I have moved the entire project over to that new implementation. This required making a set of changes to about 60% of the scripts in the project. This change also encouraged a global improvement to the arrangement of the code. Because PDO reports errors in a different way from MDB2 it is now easier to put the code for handling the result of a successful database request immediately after making the request and ahead of the code for handling errors. This makes the code much easier to understand. PDO also has a standard way of retrieving a set of responses from the database as an array, where MDB2 required repetitively issuing a request for one record at a time. This should make many of the scripts faster.

This global change combined with starting a new contract has limited the time I had available to work on my hobby.

The `User::getUsers` method of the class that manages registered contributors has been enhanced to support searching for a user by the whole value of the user name identified by the regular expression `'^username$'`, and to retrieve all users except a specified user by the `'!username'` pattern. This eliminated a couple of places where scripts accessed the SQL database table directly.

Family Tree Enhancements

Most of the work here was done as part of the migration from PEAR MDB2 to PDO. One of the design objectives is that the application specific scripts will access the database only through an object-oriented interface. For example the scripts do not access the SQL table `tblIR`, but rather the `LegacyIndiv` class which encapsulates the information recorded in that table as well as the supporting table `tblER` which records each event or fact pertaining to an individual, the table `tblCR` which records the individual's membership as a child, and the table `tblMR` which records the individual's role as a spouse.

The static member method `LegacyAddress::getWhere` which is used to identify those instances to be returned by the method `LegacyAddress::getAddresses`, deleted by `LegacyAddress::deleteAddresses`, or updated by `LegacyAddress::updateAddresses`, is enhanced to support selection using any of the fields in the class and to support lists of values for all of the integer fields. This eliminated the need for the script `FamilyTree/LegacyAddresses.php` to access the database using SQL.

The request to the database to obtain the information to fill in the nominal index is slow and getting slower as the database gets larger. It used to search from the requested name all the way to the end of the table, requiring examining tens of thousands of records, even though a maximum of 50 records are displayed. This has been reduced somewhat this month, but more needs to be done to make this script responsive.

The base class Record, which implements the common functionality for all classes which represent a record from the SQL database, is enhanced to use prepared statements, which improves performance and protection against SQL insertion attacks.

Census Database Enhancements

The script `CensusUpdateStatus.php`, which summarizes the progress of the transcription effort for a particular census and optionally a province, is improved to also display the French name of the districts.

Vital Statistics Enhancements

The functionality for determining the default place name used when creating a new birth, marriage, or death transcription has been moved into a shared common routine, which makes the code easier to maintain.

When using the back-door into Ron Demaray's Ontario Cemetery Finding Aid (OCFA) database it took a couple of seconds for the list of townships to become available after selecting a county. I found this was because the SQL server determined it had to read 161,424 out of the 1,101,579 records from the OCFA table in order to respond to the request. So I created a separate table `OcfaTownships` which permits the script to obtain the information in a small fraction of a second.

Bug Fixes

- The method `LegacyCitation::save` did not return 1 for a successful insert, or 0 for an insert that was ignored because it was for a duplicate of an existing record.
- In `LegacySource` the table to interpret values of the source type was missing the type for web-sites.
- `Record::save` did not set the value of the numeric index field in the internal copy of the record contents for a newly inserted record.
- Clicking on the preferred event checkbox cleared all preferences, not just the preferences for the type of event.
- Clicking on the preferred event checkbox to clear the existing preference did not update the database.
- The page number was not inserted into the citation for a match against the Wesleyan Methodist Baptisms table in `legacyIndivid.php`.
- The display of the contents of an individual birth registration did not have proper CSS definitions for left-aligned fields reporting an error. The fields were displayed with the wrong width.
- The search for matches in the database to a death registration did not include matches on married name.