

Database Schema Upgrade

Introduction

Database schema & PHP update files require a patch file being placed in `installer/schema/`. This file handles incremental updates from SVN or updates. The `db/tiki.sql` file also needs to be updated with the same change; it handles all new installs. The database schema from an update and a new install must be the same. The `tiki.sql` file defines the bulk of the database. All patch files can be verified before committing by running `php console.php -n database:update`. A list of successful patches executed will be given in the terminal. Its worth noting that the installer-console stores the name of each patch in the `tiki_schema` table so for testing you need to remove that row to get it to run a second time. (usually the last one installed obviously).

A upgrade can be handled in three different ways:

- From command line, by running: `php console.php -n database:update` (this also runs PHP upgrade files)
- From the web interface, by running `tiki-install.php`
- From command line, by running `sh doc/devtools/sqlupgrade.sh` (supports [MultiTiki](#))
 - *I have found it somewhat buggy on MultiTiki sites in that it (or something) seems to loose some of the dirs needed for MultiTiki operation. (jonnyb Sept 2009)*
 - maybe related to these two bug reports?: (Xavi)
 - [upgrading db with tiki-install.php breaks multitiki installs \(removes /templates_c/ ...\)](#)
 - [multitiki in subdirs broken in tiki3 proposed branch due to deleting templates_c at install time](#)

For shell scripts, the PHP interpreter must be available from command line.

Create a New Database Patch

Each file is stored in `installer/schema` and the naming convention is **YYYYMMDD_patch_description_tiki.sql**. The SQL file contains a list of semi-colon separated SQL statements.

Example:

```
installer/schema/20080922_new_structures_tiki.sql
```

```
DROP TABLE tiki_structures;  
CREATE TABLE tiki_structures (  
    -- ...  
) ENGINE=MyISAM;
```

```
installer/schema/20080922_new_structures_tiki.php
```

```
function pre_20080922_new_structures_tiki($installer) {  
    // Store current structures  
}  
  
function post_20080922_new_structures_tiki($installer) {
```

```
// Restore current structures in new model  
}
```

Update tiki.sql file

The db/tiki.sql file has to be updated to reflect the changes for new installations if the patch is to be made on standard Tiki distribution.

Create New PHP Patch

A PHP patch file may also be defined. The file naming is very similar to that of a SQL update, **YYYYMMDD_patch_description_tiki.php**. One may simply swap out the .sql for .php. The file can define a `pre_` and `post_` function (or both) to be executed before and after the patch. These functions can be used to perform operations to be made during the upgrade that would otherwise be too complicated to do with SQL, or for anything more suited to PHP. A function named **upgrade_YYYYMMDD_patch_description_tiki(\$installer)** will be automatically executed. `$installer` may be used to run SQL queries e.g. `$installer->query($query, array());` etc.

Examples of patches

- <https://sourceforge.net/p/tikiwiki/code/HEAD/tree/trunk/installer/schema>

How it works

For updates:

1. The script scans for patches available in the folder
2. Previously installed patches are pulled from the database
3. New patches are ordered for installation
4. For each patch,
 1. Pre-function is executed if available
 2. SQL statements are run
 3. Post function is executed if available
 4. Patch installation is saved
5. Global scripts are applied

For new installs:

1. Runs the usual tiki.sql-derived file to create the base install
2. Mark all patches with the `_tiki` suffix as installed
3. Performs update (as above) to apply other patches

Third Party Modifications and Customization

Effectively, any third party modification can add database patches simply by adding the patch file in the patch folder. If the patch does not have the `_tiki` suffix, it will be applied to new installations. Custom deployments no longer need to maintain the `tiki.sql` file and will not risk conflicts when updating from the source.

Global scripts

The process also defines global scripts. These scripts are stored in `installer/script/` and are PHP scripts. Each file in that folder is executed after the update. The files can be used to perform various operations, like clearing cache or any operation that should be executed after a normal update.

Schema Update History

- Before [Tiki3](#), database schema patches were handled in a single file. Separating them allows for branch merging and altogether smoother upgrade process.
- `tiki.sql` is taken from a shell script and modified to work with various different databases: SQLite, PostgreSQL so after modifying `tiki.sql`, make sure you run `convertsqls.sh` (while you are in its directory, `db/convertscripts`), and commit back all the affected files. Starting in [Tiki5](#), no need to use `convertscripts` as [Database independence](#) has been dropped.
- Patches and custom coding, in the past, always had to go through this path, now they do not, which is much better for core development. Also, it allows any custom changes you make to the database to survive through an upgrade.

Alias

- [DatabaseSchemaUpgrade](#)
- [Data Migration Path](#)
- [Database Patch](#)
- [Database Upgrade](#)
- [Code Howtos: Adding fields or tables to the database](#)