**The Webb-site database editing system**

**Background**

I built an online editing system which allows for an unlimited number of volunteers to edit the system. We started with the “pay” database and haven’t taken that further, but the infrastructure is here!

In order not to have a proliferation of MySQL users, we instead built our own user and privileges system for editing the database. We created a “wsrole” system which grants permissions to users in our own users table, to edit certain groups of tables based on their rank for that role.

**Tables**

|  |  |
| --- | --- |
| wsroles | A table with the MySQL user and encrypted password corresponding to the role. The cred is encrypted with:  SELECT aes\_encrypt(‘password’,’<wskey>’) and decrypted with  CAST(SELECT aes\_decrypt(cred,’<wskey>’) AS CHAR).  The value of wskey must be protected: we store that in a separate schema (“private”) in a table (“keys”) on the master server. See “Setting up the Webb-site database” for its structure and data (excluding the values, of course).  Many editing pages in the Webb-site dbeditor folder (for example, the pages for academic projects with the HKUteam role) use the wsrole to connect to the master database, using Sub prepRole in the prepMaster.inc include file. |
| wsroletabs | A list of all enigma tables which can be edited by that role. A table may belong to more than one role. For example, the “people” role can only edit tables related to humans (alias, people, person etc) but the HKUteam role has all of those and more. |
| wsprivs | A table with the roles of each user, their ranking in that role (from 0 to 255) and whether they are still live. A ranking is normally 1 or higher, but we can set it to 0 if the user is dead and we want all live users to be able to correct their work. A user cannot be live and have a 0 ranking. |
| users | A table with userID of those users (in mailvote.livelist) which have volunteered as editors, along with their chosen username which matches the name in mailvote.livelist. Any subsequent changes to the username (using the Webb-site user pages) are applied to both the livelist and users tables.  The userID is contained in any table which they have edited, so don’t delete a user from enigma.users. When they stop working on a project, set them to live=FALSE in wsprivs (see “User Administration” below). |

**MySQL users**

For each row in wsroles, create a MySQL user on the master server with the name in the MSuser column and a password which is stored (encrypted as above with your wskey) in the wsroles cred column.

On the master server, GRANT the necessary privileges to the MySQL user for editing relevant tables. For our existing such wsroles, we have included CSV dumps of their current grants. Find these in the “Setting up the Webb-site database/ Master grants/ Role grants” folder.

**Stored procedures for MySQL triggers**

|  |  |
| --- | --- |
| insertUserID (‘tablename’,new.userID) | Call for each row in BEFORE\_INSERT triggers. Determine the correct userID (including if sent from Access rather than web) and whether that user has write-privilege. Rejects the INSERT if not. |
| updUserID (‘tablename’,old.userID,new.userID) | Call for each row in BEFORE\_UPDATE triggers. Determine the correct userID and whether that user has write-privilege and rank 255 or outranking the existing userID in the record. Rejects the UPDATE if not. |
| NB there is no way to check userID in a DELETE query. |  |

Example: triggers in the people table

BEFORE\_INSERT

Call insertUserID (‘people’,new.userID)

BEFORE\_UPDATE

Call updUserID(‘people’,old.userID,new.userID)

BEFORE\_DELETE

We cannot send a userID through a query “DELETE FROM table WHERE…” – so our privileges system won’t work for that! We just user server-side code to guard against deletions by users who are out-ranked by the prior editor.

**MySQL functions**

|  |  |
| --- | --- |
| hasRole (roleID,userID) | Check whether the user has live write privileges on all the tables needed for a role, whether via that role or other roles. For example, a user may have access to all the tables of the “people” role if they have the “HKUteam” role. Used in the corresponding VBS to display menu buttons for a page. |
| maxRank (‘tablename’,userID) | Find the maximum ranking that a user has on a table in all the roles of that user (whether the privilege is live or not). A user may have a higher privilege in some roles than others. This function is used in VBS to determine the ranking of existing userID in a record. Returns zero if no ranking. |
| maxRankLive (‘tablename’,userID) | Like maxRank, but only using live privileges. Determine the ranking of the editing user. Returns zero if no ranking. Used by insertUserID and upUserID |

**MySQL roles**

|  |  |
| --- | --- |
| core | This role includes all needed SELECT privileges for tables and EXECUTE privileges for functions, and we GRANT it to all editor roles. |

**VBScript functions/procedures in prepMaster.inc include file**

|  |  |
| --- | --- |
| Function rankingRs(rs,uRank) | Returns Boolean, whether the user has ranking to edit this row. Recordset rs must include fields userID and uRank (an alias from calling the maxRank MySQL function). Replaces the canChange(rs) function. |
| Sub prepRole(ByVal roleID,ByRef conRole,ByRef conRs,ByRef userID,ByRef uRank) | Prepares a connection and recordset on the master database for editing using the MySQL account corresponding to the wsrole. Returns the user rank (uRank) for that role. The MySQL account has a set of GRANTs covering the necessary tables. |
| Sub prepMaster | A lot of tables such as stocklistings can only be edited by a few users (historically at Webb-site HQ) and we don’t use rankings. For these pages, prepMaster establishes a database connection with that user’s username and password which are contained in session cookie after login (only visible to the server). |

**UserID in tables**

All userID columns are “NOT NULL” and we replaced the previous null values with 4 (David) even though we actually don’t know who entered them as some were entered before we had a userID column.

We’ve set all userID columns in tables modified by Access or our scripts to default 0. If there were no default then when we INSERT from VB.net or from Access without specifying a userID, the query would cause an error: “<column> doesn’t have a default value”. By setting the default to 0, the query passes to the Trigger, which can then calculate the actual userID from either the role or the local connection.

**Handling local users with MS Access**

These users (David, Cynthia, etc) use MS Access or run VB.net scripts, which do not send updates to the userID columns in the tables. However, each is connected via a user DSN to ODBC, logged in with their username and password. The MySQL trigger looks for the connection-user in wsroles and if it doesn’t find it then it must be a local user, so then it finds the userID from the users table.

Then the privileges test on wsprivs proceeds as before – does the local or remote userID have write privilege to insert to the table, or sufficient ranking to update it?

**User Administration**

The Webb-site editor folder contains useradmin.asp. A user with ranking of at least 128 (out of 255) in a wsrole can use this to add users or edit the ranking of a lower-ranking user in that role and change their “live” status if they are removed from the role. A user cannot edit their own rank or live status.

**Adjusting rankings**

The wsprivs table has rankings for users in roles. But a user may lose status. If we downgrade the user’s ranking, then others can outrank them and correct their entries.

If a user’s entries are considered unreliable then: downgrade their rank to 0 so that everyone else can correct them; and set the “live” flag to FALSE.

**Authentication**

For online editors, all authentication is done using a shared module “webbmail/authentic.asp” used by the User-login (webbmail/login.asp) and by the editor login on the editing pages (dbeditor/default.asp). See “Notes on the Webb-site login system”.