Multiuser Bibliography – Database Design

Michael Wojcik

WRA 410: Advanced Web Authoring

25 February 2008 – 28 April 2008

Revised 28 April 2008:

- Superseded entries now have their own table
- Additional information regarding some fields (user password, entry IDs)
- Additional comments about possible changes for the next release (relational table for comments)
- Some formatting changes for readability

Current entries stored in the bibliography

Entries		
id	INTEGER	primary key, autoincrement
supersedes	INTEGER	ID of entry in **Oldentries** table
dateadded	DATETIME	
contributor	INTEGER	ID of user who added entry
bibkey	VARCHAR (40) NOT NULL	key for bibliographic sorting
worktype	ENUM	book, article, ..., other
primaryauthor	VARCHAR (80)	typically lastname, firstname
otherauthors	VARCHAR (160)	typically firstname lastname
title	VARCHAR (160)	
containedin	VARCHAR (160)	title of journal, collection, etc
editor	VARCHAR (160)	and/or translator, etc
volume	VARCHAR (40)	and/or edition, etc
workinfo	VARCHAR (160)	any other info about the work
publisher	VARCHAR (80)	
pubyear	VARCHAR (10)	also used for [name year] cites
pubdate	VARCHAR (40)	for periodicals, etc
pubinfo	VARCHAR (160)	any other publication info
pages	VARCHAR (40)	

author is not separated into “first name” and “last name” fields because many documents are written by authors whose names should not be separated that way: authors from cultures where surnames are

written first, authors with a single name, institutional authors, etc.

dateadded should be augmented with a *dateupdated* field, which would indicate when any modification had happened which did not create a new version of the entry (moving the old one to the **Oldentries** table—see below). Currently, the only such action is adding a comment. This would allow the "Recently Added or Updated" view (the application's home view) to show entries that had recently been commented on.

A more sophisticated approach would be to employ a greater normal form and extract *author*, *editor*, and similar fields into their own tables. That would potentially allow unifying work authors, where multiple entries by the same author exist. There's a substantial body of research on the general problem of generating canonical names and identifying duplicates in processing citations (the "citation-matching problem"). See for example Dongwon Lee et al, "Are your citations clean?", *Communications of the ACM* 50.12 (2007) 33-38.

Superseded entries

This table contains entries which were in the **Entries** table but were edited, causing them to become *superseded*. Superseded entries are not displayed in normal views, but they can be retrieved using an option in Advanced Search, and they are available to Detail View (by ID).

Note that an entry can be edited multiple times, creating multiple versions in the **Oldentries** table. Versions after the first will have both a *supersedes* and a *supersededby* value.

Oldentries		
id	INTEGER	primary key, autoincrement
supersededby	INTEGER	entry ID of superseding entry
supersedes	INTEGER	entry ID of superseded entry
dateadded	DATETIME	
contributor	INTEGER	ID of user who added entry
bibkey	VARCHAR (40) NOT NULL	key for bibliographic sorting
worktype	ENUM	book, article, ..., other
primaryauthor	VARCHAR (80)	typically lastname, firstname
otherauthors	VARCHAR (160)	typically firstname lastname
title	VARCHAR (160)	
containedin	VARCHAR (160)	title of journal, collection, etc
editor	VARCHAR (160)	and/or translator, etc
volume	VARCHAR (40)	and/or edition, etc
workinfo	VARCHAR (160)	any other info about the work
publisher	VARCHAR (80)	
pubyear	VARCHAR (10)	also used for [name year] cites
pubdate	VARCHAR (40)	for periodicals, etc
pubinfo	VARCHAR (160)	any other publication info
pages	VARCHAR (40)	

The *id* fields of the **Entries** and **Oldentries** tables are common; IDs are unique across both tables, and when an entry is moved from **Entries** to **Oldentries** its ID remains the same. This lets old Detail View URLs (for example, in user bookmarks) continue to work even when an entry is updated. The *supersededby* and *supersedes* field values are IDs from this set, and may refer to entries in either table.

Users defined for the application

Users		
id	INTEGER	primary key, autoincrement
email	VARCHAR (60) NOT NULL	also serves as username
password	VARCHAR (80) NOT NULL	actually a hash, in hex
dateadded	DATETIME	
fullname	VARCHAR (80)	

As with the *author* field of the **Entries** table, the *fullname* field is not split into first and last name. This properly accommodates users with non-European names, users who wish to use a single-word alias, and "users" that don't represent a single person (a group account, for example). Having surname and personal name separated actually isn't all that useful, especially for a small set of users. There's no need to do it for lookup purposes (as with a phone book)—we can search the database. There's no need to correlate these user names with any other database; we don't even have any reason to think people are using their real names. I don't anticipate ever using the user names as anything but a single unit; they're just for display purposes. So I don't need to know how users might break their names up, and the database doesn't either. (Note the long history of treating user's "actual" or "display" names this way, going back at least as far as the "GECOS" field in the Unix *passwd* user database.)

Currently, the *password* field actually contains a string specifying password representation type and additional data, plus the password verifier itself. The only supported representation is salted MD5, and the additional data is the salt. This is more secure than bare MD5 hashes, but only marginally more secure than using a fixed salt, and unnecessarily cumbersome. It should be revisited for the next release.

Comments attached to entries as annotations

Comments		
id	INTEGER	primary key, autoincrement
entryid	INTEGER NOT NULL	key of entry this applies to
contributor	INTEGER NOT NULL	ID of contributing user
dateadded	DATETIME	
comment	TEXT	

Because each comment has an *entryid* field that associates it with an entry, a comment can only be attached to a single entry. That means that in the current version, when an entry is edited (moving the previous version of that entry to the **Oldentries** table), any comments are moved to the new version of the entry, and no longer displayed with the old version. It would be better if comments were shared by all versions of an entry that they apply to (ie, with the one that was current when the comment was added, and all future ones). That would require removing *entryid* and creating a relational table to associate comments with entries.

Future Features

These additional tables may be used to implement a “custom list” feature in a future version of the application. Users could create lists of entries that would be stored across sessions. Lists would essentially be subject bibliographies, and the entire bibliography would be a superset of all subject bibliographies. Entries could belong to multiple bibliographies.

User-created lists

EntryLists		
id	INTEGER	primary key, autoincrement
listname	VARCHAR (80)	
creator	INTEGER	user ID
dateadded	DATETIME	

Relational table to associate entries with lists

ListEntries		
listid	INTEGER	
entryid	INTEGER	