BTeV Document Database Design and Interface

Eric W. Vaandering, Vanderbilt University
Lynn Garren, Fermilab

BTeV-doc-140

27 November 2001
Revised: 8 June 2002

Abstract

In this memo, we describe the design and implementation of the BTeV document database.

1 Design considerations

BTeV needs a document database that satisfies a number of requirements:

- A single database to store talks from group meetings, conference talks and proceedings, and publications *and* to present these special cases in ways that makes sense for those special case.
- All documents are kept on the BTeV web server, not personal machines. This way we don't suffer from link rot (linked-to files disappearing from the web).
- Each document can have multiple revisions and old revisions are still available.
- Each revision of each document can have multiple files. This accommodates multiple file types (source and presentable) and/or child files (especially useful for trees of HTML documents).
- The ability to upload documents by browser upload or forcing the document database to download via http.

There are several security issues we want to resolve too:

- The ability to not only have public and private documents but also documents that are accessible to subgroups (like RTES) which don't know the BTeV password. We also want to have documents that are only accessible to sub-groups (like the executive council).
- Users should only know that a document exists if they have privileges to view it.
- The ability to easily move documents, or just certain versions of documents, from restricted to public and vice-versa.

The document database system consists of three pieces which work together: a web-based user interface, a relational database server to store information about the documents, and a file system to store the documents. The BTeV web server serves files from the file system to the users.

2 Interface

The document database is completely accessible via the web. HTML and HTML forms provide the user interface and CGI scripts handle all user access to the underlying database, both for placing information in the database and retrieving it.

All access and data entry scripts are written in Perl using the Perl::CGI and Perl::DBI database access modules. Additional access programs can be written using PHP, C++, or any other language with an SQL interface.

2.1 Adding information to the database

There are five different ways to add or update information in the database. Each is used for a particular circumstance:

1. Reserve a document #: This is used for documents that don't yet exist, but a number is needed. This allows the writer to know the document number before the real document is placed in the database.
2. New document: This is for documents that have not been entered or reserved in the database. This is the way the first version of a document is entered.

3. Update a document: This is the way updated versions of existing documents are placed in the database. The document number remains the same, but the version number is incremented.
4. Update database information: This is used when the document hasn't changed, but the information about it has. For example, it's now published so that information has to be added. Security settings on documents are also changed this way. The version number is unchanged.
5. Add files to a document: This option is used to add new files to an existing version of a document, perhaps files that were forgotten or an additional presentation format of an old document. If the content of the document has changed at all, the Update, not the Add file, option should be used. The version number is unchanged.

Retrieved information from the database is presented in two ways, lists of documents that meet specific criteria and a display of all the information available about a document. The information presented in the list is intended to provide enough information for the user to decide if they want more information.

Currently, documents can be listed by author, subtopic, type, and date of last modification. While a more comprehensive search engine is not yet included in the product, one is planned.

2.2 Security

When you access any part of the interface, you will be asked by your web browser to supply a username and password. Which username you use will influence which documents you have access too. You will not be able to see any information about any documents you can't access.

3 Database

The database design is essentially finished, although small additions will probably be needed for refereed publications (see Section 5).

The document database is implemented as a number of different tables. Because each document can have multiple revisions and each revision can have multiple files (and authors and topics), there are three main tables Document, DocumentRevision, and DocumentFile. The hierarchy of these tables, their elements, and the auxiliary tables are shown in Figure 1.

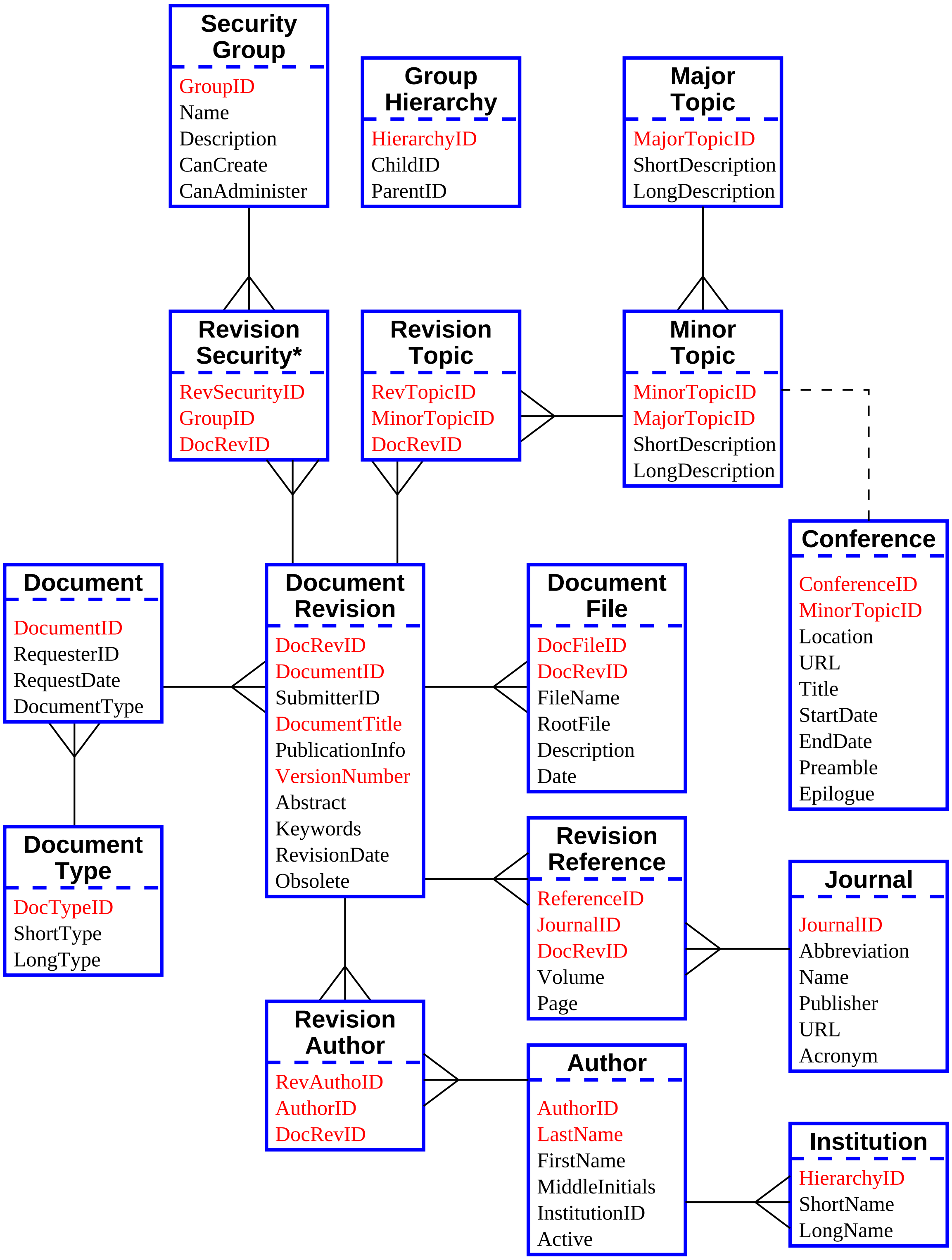

Figure 1: The design of the document database. Indexed fields are shown in red and the symbols are described in the text.

3.1 Database design considerations

The database design we are using is in the "third normal form." For a longer discussion of what this means and how a database is properly designed see [1]. Essentially it boils down to three requirements:

1. Each field of a database contains only one piece of information
2. Any field which has values in common across multiple entries should use a unique identifier and another table
3. No two fields in a table should depend on each other

To take a simple example, the field containing the document type should contain a unique identifier, not the name of the type because the name might have been misspelled (rule 2). The institution of the person who wrote the document something is not a property of the document, but of the person (rule 3).

The "tree"-like symbols indicate the one-to-many relationships within the database. For instance, there can (and will) be many revisions of a single document. Rule 1 mandates that there can be no "trees" with branches on both ends. For example, because each document revision can have multiple authors and an author can be associated with multiple documents, an intermediate table RevisionAuthor is used. There is one entry in this table for each author of each document revision.

Fields shown in red are indexed for fast look up. Other fields must be searched on which is more time consuming. Each index takes up some amount of disk space, so the number of indices should be kept to a reasonable number. The ID numbers in each entry indicate what other entries they are associated with. In addition, each table has a timestamp which is not shown.

3.2 Explanation of tables and fields

While most of the tables and fields in Figure 1 should be self explanatory, a few should be described further.

Document: It might be surprising that very little information is stored about the actual document (just who requested it and the document type). This is because most of the information about a document can change and is properly stored with each document revision rather than the document.

DocumentRevision: This is where most of the information about a document is stored. The Obsolete field is set when the database information is updated but the document itself is not. A new version number is not generated in this case, so the DocumentID and VersionNumber fields are not sufficient to specify a unique DocRevID.

DocumentFile: This table stores the information about each file in a document revision. RootFile is used to distinguish between primary and supporting files within the document. Under some circumstances only the main files are listed or the two types are segregated.

Linking tables: The tables RevisionSecurity, RevisionTopic, and RevisionAuthor are all used to link multiple settings for security, topics, and authors to a single document revision.

Topics: Topics are divided into MajorTopics and MinorTopics (or sub-topics). Each minor topic is associated with a single major topic. The document revision is associated with one or more minor topics, not major topics.

Conference: For conferences, the short and long names of the conference are stored in MinorTopic. Additional information about the conference, such as location and dates are stored in the Conference table. Each row from conference is linked to a row from MinorTopic.

SecurityGroups and GroupHierarchy: Each document revision can be viewable by an arbitrary number of groups. A group can be the collaboration as whole, a sub-group, or a group associated with BTeV like the Real Time Embedded Systems (RTES) group. The table GroupHierarchy implements a hierarchy of groups within the system. For instance, BTeV is currently defined as a "parent" of "RTES" so that those with BTeV access can read all documents under the RTES security, but not vice versa. This is primarily a convenience measure.

3.3 Implementation

This database is implemented using a MySQL server running on `fnsimu1.fnal.gov`. The use of non-standard SQL features in the interface has been kept to a minimum to ease porting to another another database if needed. (MySQL has a very useful but non-standard function `last_insertid` which returns the unique identifier of the last item inserted into the database. We use this because it is safer and more elegant than keeping track of all these numbers in a separate table.)

4 File system and security

All documents are placed into a single directory tree. There is no division based on security or topic since these are fluid designations (and each document can have multiple values for these settings). The file system is arranged only by document number. Each revision of each document is placed in its own directory, which means that most directories will only have a small number of files. While this is somewhat inefficient, the advantage of this scheme is two-fold. First, each of the files of a particular revision need not be renamed. Second, web access to each revision is controlled with a potentially unique .htaccess file. (These .htaccess files are written by the interface software at the time files placed in the file system.)

Once the database system has placed a document in the file system, it has no further interaction with the file. (It never checks to make sure it is still there.) From this point on, sending the actual files to the user is the sole responsibility of the web server.

To avoid possible issues with large numbers of files in a single directory, the document directories are further divided into groups of 100. For example, imagine that the 3rd revision of the 1029th document contains one file, text.ps. Then the full path of the file is:

```
$DOCROOT/0010/001029/003/text.ps
```

As you can see, the file system is designed for up to 999,999 documents each of which can have up to 999 versions.

5 Special Types of Documents

In addition to generic type of documents, extra information exists about documents which are associated with group meetings, conferences, or refereed journals. With small extensions to the database structure, this additional information can be easily captured and represented.

5.1 Group meeting talks

Up until this point, talks given at group meetings outnumber the total number of documents entered into the old document database. We would like to completely integrate the group meeting talks into the document database. They should receive special treatment since they will be a large fraction of the documents and people have certain expectations of how the information is presented.

Associating a talk with a meeting is done by choosing, as one of the document's subtopics, the correct meeting. Then a search on that topic will return all the talks for the meeting. Additionally, meetings are sorted in reverse chronological order rather than alphabetically wherever they appear.

Making a usable list of meeting talks involves making a minor change to the listing format by listing the files associated with a talk rather than the modification date.

Together these changes yield an interface that is similar to what collaborators are used to. Combined with the instant update nature of the database and its indexing capabilities should make the overall experience much better.

5.2 Conference talks and proceedings

Conference talks and proceedings pose a slightly different problem. As with group meetings, associating a document with a conference will be done by choosing an appropriate subtopic. Listing of documents will be by conference date, not modification date, in reverse chronological order.

A full listing, however, should also contain information about the conference itself such as location, dates, a URL, etc. This was accomplished by adding an additional table to the database linked to the conference's MinorTopicID.

This functionality has been partially implemented. Listing by conference is possible, with slightly different options. The sort is currently done on the modification rather than conference date.

Yet to be decided is the issue of how to handle a conference talk and its associated proceedings. Are these filed as two documents or just one? If just one, what document type is to be used?

5.3 Journal publications

Publications in journals have a slightly enhanced interface as well. While we have a Publication Information field in the database, what is placed there is completely up to the user and is intended more as a place to put notes rather than information in a well defined form.

The publication information field will remain to allow arbitrary notes to be added by the users.

To support journal publications,additional fields have been added to the database to contain this information in a very well defined manner for a set of predefined journals. This will

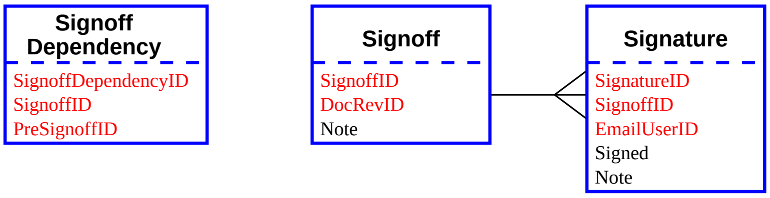

Figure 2: DocDB tables used for signoffs. Indexed fields are shown in red. A Signoff is satisfied by obtaining one of the signatures in the Signature table. SignoffDependency controls the order in which signoffs are collected. A Signoff can not be signed until all PreSignoffs have been completed.

allow a coherent presentation of a list of publications and advanced features such as links to the publisher's web site, automatic generation of BibTeX entries, etc.

This is also partially implemented. The data is collected and stored in the database. It is also displayed on the full document view page, but there is no code written to just list public documents and their references, etc.

6 Signoff system

An optional component of DocDB is to allow some documents to be "signed" by a group of people before becoming "approved." People with personal accounts can sign documents. The list of people needing to approve a document is editable by the same people that can edit the document itself.

A typical requirement of such a system is the ability to, at certain points determined by policy, not the DocDB, "freeze" a document and its meta-information such that only "managers" can modify it or unfreeze it. BTeV does this by making the document editable only by the "Executive" group.

When displaying document version(s) in a list, there are obvious indications of which documents are approved, which are unapproved, and which are obsolete (even if they were approved at some time). All information about who "signed" each version of each document is kept.

DocDB contains tables to allow any number of approval "topologies." These tables are shown in Figure 2. For instance, person A or person B might be allowed to sign at the first step, followed by person C at the second step. Or, person A and person B may have to sign (but in parallel) before person C can sign. However, the initial code only allows one topology (an ordered list). When a document under control is updated, the signoff list structure is preserved, but the approvals themselves are cleared.

The signoff system provides a number of additional convieniences:

- Email notifications to signatories when a document is ready for their signature
- A way to list all controlled documents

A number of other features are planned and will be added as needed:

- Email reports of outstanding signatures needed (to desired signatory and other signatories of documents)
- List of all documents a person is a signatory (actual or requested) on
- More complicated approval topolgies (OR's, parallel paths, etc.)
- Reminders if a document goes unsigned for a while
- Restricting the list of people who may sign documents to a sub-set of those with personal accounts

References

[1] R. J. Yarger, G. Reese, and T. King, *MySQL & mSQL*, O'Reilly & Associates, Inc., 1999.