BookBrainz Developer Docs

Release 0.1

Ben Ockmore

December 16, 2014

Contents

This documentation is intended to act as a guide for developers working on or with the BookBrainz project, describing the system, its modules and functions. The BookBrainz webservice is also fully documented here.

For a description of the BookBrainz and end-user oriented documentation, please see the BookBrainz User Guide.

CHAPTER 1

Contents

1.1 BookBrainz Schema

1.1.1 Introduction

The BookBrainz schema describes how the data used by BookBrainz is stored. It's quite important to have a good idea of this before you look at any of our code. If you're coming from a MusicBrainz background, our schema is similar, but there are some key differences - see *Coming from MusicBrainz*.

1.1.2 Definition

The BookBrainz schema is defined in the python-bbschema package, which you can access at https://github.com/BookBrainz/python-bbschema. When the schema is more stable, there'll be a nice schema diagram here to help you understand how things fit together, but for now, you'll have to make do with some words.

1.2 Entities

In BookBrainz, an entity is a container for some data, associated with some globally unique identifiers (GIDs). BookBrainz represents a number of different objects as entities, and each of these has its own particular associated data. In the following section, we'll go through the database structures used to represent entities, and talk about each type of entity in more detail.

1.2.1 Generic Entity Tables

Entity and Entity Redirect

All entity GIDs are stored in a single table in the database, **entity**.

A second table, **entity_redirect**, allows redirection of GIDs. For example, if one entity was merged into a second entity, then a row would be created in the **entity_redirect** table to indicate a mapping to the merge target.

Entities in BookBrainz are versioned, which means that the entire entity history is stored in the database. To make this possible, we have an additional table in the database, called **entity_tree**.

Entity Tree

The **entity_tree** table is the place where information is actually stored. You can think of this table like a folder on a computer. It points to the various bits of data related to a particular version of the entity. For example, the most recent annotation ID, disambiguation comment ID and entity-specific data ID will usually be stored at this level.

When a new version of an entity is created, a corresponding row in **entity_tree** is created, which will indicate that data was updated by modifying one of the stored IDs.

Entity Data

Each entity tree will point to a particular row in the **entity_data** table. We use joined table inheritance to represent the different entities in BookBrainz, and this single **entity_data** table represent the base object in this inheritance hierarchy. It contains an ID and a field to determine the type of entity data stored, known as the *discriminator*.

Additional Tables

There are some additional tables related to all types of entity. We've already mentioned annotations and disambiguations, so let's talk a little more about those.

An **annotation** is a way of making notes about an entity, for other editors to read. It stores some content associated with an ID. Disambiguation comments, stored in the **disambiguation** table, have a similar data structure but are intended to contain a short description to allow editors to easily differentiate between similarly-named entities.

An **alias** represents a name or title. Each alias will store some text along with a language, and a couple of flags to indicate whether the alias is *primary* and whether it is *native*. An entity can only have one *native* alias, which indicates its original name. It can have many *primary* aliases, which give the most common names in particular languages. *Native* aliases will usually also be *primary*.

1.2.2 Specific Entities

Publication

Publications represent the books, magazines, articles, newspapers, novels and other published materials catalogued in BookBrainz. The table **publication_data** stores the entity specific data for publications, and represents an object derived from **entity_data**.

1.3 Coming from MusicBrainz

This page describes the key differences between the BookBrainz and MusicBrainz schemas, for developers already familiar with the MusicBrainz schema.

1.3.1 Entity Base Object

In BookBrainz, there is an "Entity" base object. What this means is that all entities share some common data, and store this data in a single table in the database. This greatly simplifies the rest of the database compared to MusicBrainz - instead of having a set of tables defined for each type of entity, we only need a single set of tables referencing the base table for entities.

The following fields are stored in the base table:

- gid
- master_revision_id
- last_updated

1.4 BookBrainz Webservice