Michael Prelich
mprelich
20424478

CS348
Assignment 2

Design Document

Bibauthor:

My bibauthor program is split up into multiple steps. First, I query the database for all the publications written by the supplied author and get their pubids. I then query the database for all books, and articles with a matching pubid, as proceedings and journals cannot be written by an author. I also query for all publications that have a pubid that matches an article's appearsin attribute that has been written by the author. The name of the first author for each publication is also returned in these queries for sorting purposes (it is returned as "" for journals and proceedings). The year of each article is taken as the year of the publication that the article appears in. These three main queries (books, articles, articlies_appear_in) are unioned together to form a full list of publications written by the author, and they are ordered by year and name. This query is stored in a cursor, which is then iterated through. Each variable in the cursor returns the: pubid, year, type, and name of the first author for each publication. The pubid and type are stored in host variables. As we iterate through the cursor, we take the pubid and type and use this information to print out the details for each publication. The printing process for each publication type is pretty self-explanatory from the code.

Bibmaint:

My bibmaint program first reads in a line from standard in. It uses strtok to parse through each line accordingly. I first parse to the first "(" and use this string to see what kind of operation needs to be done (ie. author, book, etc.). The appropriate update function is then called, each of them working similarly. I then parse the rest of the string into host variables. If the update call is for an author, I first see if that author exists, with a simple count(*) sql query. If they do, I just update their name, and if they do not, then they are inserted as a new tuple. For the authorurl update, the author's url is only updated if the author specified exists. For a publication update, I first update the publication tuple. Similairly, I first check if the publication exists and if it does, I update it accordingly or else just insert it as a new tuple. I then have to update the corresponding proceedings, book, journal, or article tuple. Again, it is updated accordingly if it exists, or inserted if it does not. For articles, and books, I also update the corresponding author's wrote tuples. I first delete all the wrote tuples corresponding to the pubid for the specified publication. I then insert new wrote tuples for each author specified in the call (as they are stored in host variables from parsing the line).