Subjects/Names Database

Description:

Names: The cooperative is organized into projects, with each project focused on the papers/letters/diary of a historical figure or family. Each project can add to the names and subjects as a whole; each project can also assign to itself a subset of all names/subjects that are relevant to it (**project_uses** table). Further, each project creates lists, which are yet smaller subsets of names/subjects. So for example, an editor might be working on a specific period in time of a person's life, and so creates lists of subjects and names that are particularly relevant to that period. The alias table allows tracking of alternate spellings of people, or their titles and roles. Each name has any number of links associated with it (independent of project).

Subjects:

Each project may select which subjects are associated with their project (project_uses table). ? Should we have a separate associations table for subjects?

Overall requirements:

Mysql database storage backend for Primary Source Cooperative's subjects and names, written in PHP, with a URL-based API to query database and receive JSON data in response. Should implement full CRUD capabilities, including the following queries that perform joins:

- get all subjects for a specific project
- get all children/ancestors of a specific subject
- get all names and their aliases and their links for a specific project
- get all lists for a project
- get all authority_links for a specific name/subject
- get all names in a list
- delete a list (and purge list_has rows for list)
- copy a list's associated rows to new list

Queries should paginate, that is they should be able to return results in arbitrary sets of 10, 20 rows etc., and should provide as part of the JSON response the URL for the continuation of the results.

Timeline:

Part1: End of April 2020: specification of API request URI and response JSON structure, to allow simultaneous work on frontend UI by other parties.

Part2: Early July 2020: completion.

Required columns for tables

TABLE: subject

id	autoincrement; primary key
subject_name	String
display_name	String
staff_notes	text
first_created_by	String
creation_date	datetime
child_of	int; refers to another record's id column
keywords	String, space-separated additional keywords for search
loc	String (ID of library of congress equivalent field)

TABLE: name

id	autoinc, primary key
name_key	String, unique: auto created as lowercase concat of [family name]-[given name]-[middle name]-[date_of_birth]
family_name	String
given_name	String
middle_name	String
maiden_name	String
suffix	String
keywords	String
date_of_birth	String (follow ISO YYYY-MM-DD leaving out unknown parts)
date_of_death	String (follow ISO YYYY-MM-DD leaving out unknown parts)
public_notes	Text
staff_notes	Text
bio_filename	String
first_created_by	populated with username from credentials API
creation_date	datetime

TABLE: alias

id	autoinc, primary key
name_id	foreign key to name table
type	String (one of: spelling|role)
family_name	String
given_name	String

middle_name	String
maiden_name	String
suffix	String
title	String
role	String (possible future tie-in to controlled vocab, maybe an ID)
public_notes	Text
staff_notes	Text

TABLE: link

id	autoinc; primary key
foreign_key	int, ref to either subject or name table row
type	string, from choices: source|authority
Authority	string, from choices: snac|loc (other's to follow?)
authority_id	string, id from the authority's system
display_title	string
url	string
notes	text

TABLE: project

project_id	string, primary key, comes from site-name from user-roles API
name	string
description	text

TABLE: project_uses

This table is how subjects/names are assigned to projects

id	autoincrement, primary key
project_id	(foreign key to "project" table)
foreign_key	(foreign key to associated table row)
table	name of table, subject|name

TABLE: list

d	string, primary key
project_id	string, primary key, comes from site-name from user-roles API
name	string

type	subject or name
description	text

TABLE: list_has

This table is how subjects/names are assigned to lists

id	autoincrement, primary key
list_id	(foreign key to "project" table)
foreign_key	(foreign key to associated table row)
table	name of table, subject|name

How to get User Roles from the MHS System

Use this PHP:

```
include($_SERVER['DOCUMENT_ROOT'] ."/publications/lib/classes/publications/staffuser.php");
$data = \Publications\StaffUser::currentUser();
```

returns: array with "username", "role", and "sitename" elements.

"username" should be used to directly populate the "first created by" columns
"sitename" should be used to directly populate project.project_id columns

the roles returned are one from: author, editor, admin, super

"username" and "role" will both be false if the user is not logged in.

How permissions should work:

- all users && public (i.e. API returns false for username/role) have full read/view access for all tables, but public should not see staff notes columns
- authors can only read/view, for any project
- editors can insert/update subjects, names, aliases
- editors can create associations (project_uses, link, list, list_has) for their project only
- editors can delete associations tables, but cannot delete subjects or names
- admins have the one addition privilege to update (not delete, insert) their project row only
- super has full access to all (only super can create projects, or delete subjects/names/projects)