MetPetDB Database Schema Report

This document was put together by Patrick West and represent his findings after learning the schema for the system. It includes notes, suggestions, and more.

Empty tables

The users table will be subsumed by the Drupal CMS. We can keep track of users, their roles, changes to their roles, comment on their roles and on the person themselves, etc... We can also keep track of passwords, organizations, and more. The following three tables were empty, and I can find no information on the current trac wiki that says anything about these tables. Not sure what they are or what they are for. So, considering we'll be handing user, role, organization maintenance over to Drupal, we can remove these tables.
* admin_users
* role_changes
* users_roles

According to Sibel, and others (can't remember who mentioned it, and it's not in the notes for the meeting that I read), these x_archive tables are no longer needed and can be removed. They were supposed to be for a new feature, but they aren't being used, and we were told we could remove them.
* chemical_analyses_archive
* chemical_analyses_elements_archive
* chemical_analyses_oxides_archive
* sample_metamorphic_grades_archive
* sample_minerals_archive
* sample_reference_archive
* sample_regions_archive
* samples_archive
* subsamples_archive

These tables were empty as well, but more than likely they can be used for further information about images, comments on images, any references that reference the images, etc... Although, I feel that image_reference should be referred to reference_image, where the reference has a reference to the image. The image doesn't have a reference to the reference. That just doesn't make sense. So I recommend that we change image_reference to reference_image. These two tables, though they are empty, are probably going to be used.
* image_comments
* image_reference

Composite or No Primary Keys

Tables with composite or no primary keys, and need one. What this means is that a new column needs to be created that is auto_increment (django does this automagically). Then the current composite primary keys need to be, together, be uniq, or nothing needs to happen if they are just foreign keys and not primary.

* oxide_mineral_type
* chemical_analysis_elements
* chemical_analysis_oxides
* element_mineral_types
* oxide_mineral_types
* mineral_relationships
* sample_metamorphic_regions
* sample_regions
* sample_metamorphic_grades
* sample_reference
* project_members
* project_samples

The xray_image table uses a foreign key as its primary key. This probably won't work in django. We'll have to create a new primary key for xray_image, and then reference the image_id.

Table Notes

users
** has one primary key
** why there's a restriction on email addresses, not sure? It's a single string anyway. No restrictions necessary. Though, even if we could have multiple emails, shouldn't have a limitation on the email addresses. Could be a email for contact information. But a person can, and do, have multiple email addresses. I have 5 or so, can't even keep track. And, since we have to create multiple users for one person, each with a different role, that means that each new user for the one person has to have a different email address. Then again, you probably only want one anyway in this case.
** there are multiple representations of institutions in this table. Do we ever see the need to show contributions per organization within the system. I recommend we add this to the task list.
** this will be subsumed by Drupal. We add a URI field to reference a foaf document, and everything else can reference the foaf document
** roles, authentication and authorization could be handled by drupal instead of by the developed system
** Drupal has a web service available to authenticate and authorize users. So our search/facet system can utilize this service.

role
** has one primary key
** is basically a role type, not representative of a uniq role a specific person plays
** This is an unneccesary table and can be subsumed by Drupal
** Not of much use as in order for a user to have multiple roles we have to create a new user. And that doesn't make sense. Perhaps that's what users_roles table was going to be, but never got around to it.
** In a foaf document we would have multiple Role instances for each role a person plays, what organization they are affiliated in that role, the time range they play the role, etc... A person can play the same role, but associated with different organizations. We need to keep track of that.

samples
** has one primary key
** no changes here

** should sesar numbers be uniq? No two samples should have the same sesar number, right?

subsamples
** has one primary key
** I thought that even subsamples had a sesar number? Is this not the case?

rock_type
** has one primary key
** it is noted in the trac wiki (http://wiki.cs.rpi.edu/trac/metpetdb/wiki/RockType) is that the types listed here is not an exhaustive list of rock types. Is there one? Can we utilize that taxonomy? Would such a taxonomy be hierarchical in nature? On the page, it looks like types are grouped together. That grouping doesn't seem to be described on that page or represented in the database.

minerals
** has one primary key
** BUT ... has a real_mineral_id which basically allows for the creation of aliases for minerals. Can be called different things, can be referenced as a different thing, but really means the same thing. I recommend that we create a mineral_alias table instead for this purpose. There is only one mineral and it has different names, which means there should only be one row in the table with one primary key. I recommend that we create a mineral_alias table where we can list different aliases for a mineral. But a chemical analysis would reference the actual mineral, not an alias of the mineral.

elements
** has one primary key

oxides
** has one primary key
** references a single element. Guessing that since it's called an oxide that the other element is inferred
** weight, but no units? Do we need unites here?

oxide_mineral_type
** has two primary keys
** would have to create a new primary key with two foreign keys which together must be uniq

chemical analyses
** All of the chemical analysis tables are tied together, uses the same primary key and then have a second key that makes it a composite key
** each of the other ca tables will need to have a primary key associated with them. We still use the chemical_analyses_id and x_id to be uniq and to do searches
** the one thing I noticed about these tables is that a chemical analysis is actually based on a subsample, not the sample itself. An analysis is made of a subsample, right? Not on the sample itself. The subsample is itself part of a sample. So it can be inferred that the analysis is in fact of the larger sample? Is it possible that multiple analysis are done on different subsamples of the same sample?

element_mineral_types
** has two primary keys
** will have to create an id

mineral_types
** has one primary key

oxide_mineral_types
** has two primary keys
** will have to create a new primary key

mineral_relationships
** has two primary keys
** will have to create a new primary key
** I'm guessing the queries that we use to search for a sample based on a mineral also searches for any child minerals?

images, image_format, image_type
** these three image tables have a single primary key each, no problems here

xray_image
** has one primary key, which is also a foreign key. Guessing this is allowed.
** Is xray_image like a subclass of image? In other words, an instance of image is created (row) and then an instance of xray_image is created (row) that reference the image that it actually is with just the additional information.

metamorphic_regions
** has one primary key

* regions
** has one primary key

* metamorphic_grades
** has one primary key

* sample_metamorphic_regions, sample_regions, sample_metamorphic_grades
** these three tables have composite primary keys
** need to create a new primary key and have the old composite primary keys be uniq

* uploaded_files
** this table seems to represent when a particular file was uploaded, though there is no reference in the table as to what was added to the database from the file. You'd have to go to the file and see what was uploaded. Also, there is nothing that says whether the file was successful or not, what succeeded and what failed, what needed to be edited, what needed to be added, or any sort of provenance.
** has a single primary key, nothing to be done here

* reference
** has a single primary key
** the column 'name' seems to be the name of a file, but not always. Sometimes there's more information in the name. Might it be good to reference an uploaded file instead, where we have information about the file, who uploaded or used the file, and other information.

* georeference

** has a single primary key
** this is what is pulled from the text files, which are referenced via the reference_id
** the name of this table is rather confusing. It has nothing to do with, and has no column related to, geology. The site that the references come from is called georeference. These are documents. I also don't care for the name reference. Is it a reference for something else. No, it's a table of documents, with authors, publication references (journal), titles, text, etc...
** no authors actually reference any user id's that are in this system. This is something that we could/should add
** Could either set up separate tables for authors. The author table would have the name, what reference they are an author for, and an index to specify whether first, second, third, fourth ... author. Currently, the second authors column is a
** Could also have an organization for each author, just as you have in many documents.
** There is a column for the full_text, but is it possible that the document has an abstract?
** Also, a column for the full text? I would prefer to see a link to a document on the system, or a place to retrieve the document. If these are internal documents, then there could be version information in the documents as well.
** Or, this could be the first concept that we translate over to the semantic knowledge base. Each person has an instance, with an authorship being yet another role that they play. Then, in the drupal page we could display each reference (paginated list), the list of authors (all links to their instance data), and on the person page, what references they are author of. And, in the future, these pages could also list what roles they play, what samples they've owned, what projects they are on, etc...

* sample_reference
** has composite primary key
** would need to create a new primary key, and the two old primary keys become uniq constraints.
** the table name actually implies, in my opinion, that a sample references a reference. Instead, a reference (document) references a sample.

* projects
** has one primary key
** should description be limited to 300 characters? Or made into text field
** do projects have URLs? Someplace that has more information about the project, funding information, a place where people can find out information about the project, what the project is for, etc...?
** Also, is there a need to keep track of who funded the projects, the digs, the samples, etc... Funding information might be of interest, with representation of funding organizations, departments, groups, award information, etc... In the future, funding organizations might want to come in and find what projects they are funding, what samples are available for the projects they are funding, who to contact, and that they are being sited in the research.

* project_members
** has composite primary key
** would need to create a new primary key and have the old primary keys have constraint of uniq
** isn't it possible that a single person could be a member of one project affiliated with one organization, and then a member of a later project affiliated with a different organization? This might be interesting to note, and model for the future.

* project_invites

** has one primary key
** the status should probably be an enumeration instead of just a character field. New, Accepted, Rejected
** need to have a comment field in this table to say why the invite was rejected, or notes on the acceptance, or notes from the invited or invitee, etc...

* project_samples
** has a composite primary key
** would need to create a new primary key and have the old primary keys have constraint of uniq

General Comments

In some situations where a foreign key is specified we simply say x_id, where x_id is the same column name in that foreign table. But this isn't descriptive at all. We might know what it means, but it doesn't help new people. Column names need to have a descriptive column name. For example, projects has a user_id. What's it for? The owner of the project? The PI? The lead on the project? The person who created the project instance in the database? Who is this user.

Some of the table names seem to be backwards. I've mentioned this in some cases, but not all cases as I started to notice it later in the process. For example, sample_reference makes it sound like a sample makes reference to a document. But, what we actually mean is that a reference (document) makes reference to a sample. When/if we switch to semantic representation of the model we would need to make this distinction.

Documents. In this case, I'm referring to uploaded files, reference files (the files that the reference information was pulled from), the reference documents (the physical documents that are represented by a reference), images. I believe that this would be a good case for having document URI's. We have a lot of information about documents. Documents where the information is pulled from georeference site. Documents that were uploaded that contain information about samples, chemical analysis, images, etc... The images themselves can be considered documents. There is a lot of information that we can provide for any kind of document. Who uploaded them? Author information? Provenance of the document? And we can provide different representations of the documents. If someone browses to the URI then we can display a page that describes the document. If someone browses to the URI using an RDF browser, or request RDF, then we can return the knowledge of the file (another reason for semantic representation). On the html page we can have the information about the document, and then either allow users to download the document, or point them to where they could download the document.

Users. There are some instances where users are simply text fields with names. And, even in that case, there are probably multiple representations of the same user. For example, Peter Fox, Peter A. Fox, P.A. Fox, P. Fox, Dr. Peter Fox, and so on. Same person, different label. I recommend that we start a person hub, where we have a single URI to reference a person, and then have different labels for that person. Then, when we're displaying information about a person, we can search the person hub, find the actual reference to the person, and point people to that single representation.

Same with Organizations. Create an organization hub, a single representation of the organization with multiple possible labels for the organization. In the current database there are multiple entries for the same organization, just with different names.

Don't know what these tables are for

* geometry_columns
* grids
* image_on_grid
* sample_ref_sys