Gene Curation Tool
Technical Paper

Christian T. Brewton
University of Alabama, ctbrewton@crimson.ua.edu

1. Introduction

The word ontology was coined back in 1613; its practice however can be dated as far back as to Aristotle [3,4]. Philosophical ontology has "sought the definitive and exhaustive classification of entities in all spheres of being" [3]. Since the scientific revolution, the idea of ontology has been adopted by various domains (professions) one of the prominent being computer science [3]. Computer science uses ontologies to form relationships between concepts and by doing so it allows them to see the ontologies semantic structure viz. to see what it looks like. In addition, it allows for computer scientist to efficiently search and run "computational reasoning" on these structures [5]. Alternatively, ontologies can be classified into two major types: "domain ontologies" and "upper-level ontologies" [2]. Domain ontologies represent a vocabulary (concepts) for a domain of knowledge, while upper-level ontologies describe general knowledge that carries across various disciplines [2]. Not to mention, by adding annotations (information about a gene) to these concepts it allows for us to make connections between things that we may not have seen otherwise. By using ontologies and annotation, one can make collaborating easier, because we are able to share "meaning" using their structure [2]. In the following paragraphs, one will be acquainted with various types of ontologies that are used in the domain of biology and why visualizing these domain concepts as an ontology is crucial.

One of the most vital ontologies in biology is gene ontologies, which is often abbreviated GO. Gene Ontology essentially sets out to create a unified vocabulary for genes and to use the relationship between these concepts to see how species relate. For example, one could use this ontology to conclude that a specific protein found in eukaryotic cells relates to "core biological process" and is common among all eukaryotic cells [1]. Furthermore, one can see that understanding how genes among various species work and how they relate, can be vital when trying to understand the prolific amount of species that are around today.

Another vital type of ontology is the plant ontology, commonly abbreviated PO. The plant ontology is "a structured vocabulary and database resource that links plant anatomy and development to gene expression and phenotypic datasets from all areas of plant ontology" [7]. Just like other ontologies it uses relationships (e.g. is_a and part_of are the most common) to interpret how things relate [7]. Furthermore, there are various other ontologies that relate to plants. For example, the trait ontology (abbreviated TO). The trait ontology basic idea is that one can represent a trait as its own unique feature or characteristic. For example, plant height is a type of vocabulary that is associated with the trait ontology [6]. These are just to name a few ontologies, so keep in mind there are many other ontologies that encompasses many other domains.

In conclusion, ontologies helps us understand the meaning of things and how they relate. By assisting one with understanding meaning, it helps speed up the discovery process. In any case, one can conclude that with the rate of discoveries increasing so will innovation, and I believe innovation is the backbone of any profession.

1.1 Purpose

With the human population growing at such a rapid rate there is also a growing need to research plants, because they are the primary food source for many organisms on Earth [8]. Our hope is to speed up the plant research process by combining all the various plant ontology groups into one source. This will allow researchers who have various different domains of knowledge to collaborate with one another.

In addition, users will be able to easily search annotations that are associated with a gene, an ontology, etc. In conclusion, by giving researchers an easier way to collaborate with one another and by giving them an easy way to search through and manage annotations, we will speed up the overall research process.

1.2 Scope

The scope of this project is to design a database that will merge the many branches of ontology. We will be creating a collaborative web portal that will interact with our database and allow users to easily share results (e.g. annotation data). This web portal will act as a wiki so to speak, in that it allows the users to easily manage annotations. Furthermore, we will implement our database in such a way that users can easily search annotations associated with a gene, an ontology, etc. In addition, we will be making API calls to SOLR, which is the data store AmiGo is using, to query various information for the many ontologies; this is to ensure that we always have the most up-to-date version. Lastly, our web portal will allow us to easily bulk upload our approved annotation to AmiGo, to ensure that they always have our most up-to-date annotation information.

2. Design Overview

The design overview will be split up into seven parts. The first part being what features our web portal will have. Secondly, we will be describing the user level hierarchy. Thirdly, we display our Role Based Access Control (RBAC) design. Fourthly, we will present you with a couple of our use case diagrams. Fifthly, we will show several of our basic UI designs. Sixthly, we will talk about the Entity-Relationship model (ER) diagram and how it relates to our web portal. Lastly, we will be showing the table diagram for our web portal.

2.1 Features

This web portal will have a plethora of features. Here is a list of the various features:

- Relational Database System
  - We plan to use a relational database because in this stage of development our client (Planteome group) doesn't full know the type of queries they want to make, and to take full advantage of NoSQL databases a user has to have a good idea of what type of queries they will be querying.
- Role Based Access Control (RBAC)
  - This will be used to distinguish the permissions of the various user levels and what they will have access to. It will also ensure that we have a well structured user level/role design which increases security.
- User Management System
  - We want the managers and admins to easily be able to manage the users (e.g. banning users)
- Users will be able to have specialties in a variety of species
  - This feature will be added to better assign annotations to a manager so they can either approve or disapprove the annotation. The idea is to give them annotations in areas they specialize in.
- Users can can flag an annotation as valid or invalid
  - The idea behind this is once an annotation's invalid/valid flag ratio hits a certain threshold a manager specializing in that annotation will be alerted to review this annotation.
- Efficient ways to search and manage annotations
  - This is a crucial feature to our system, because it is the core backbone of what we wanna do. We want users to be able to easily search and manage annotations.
- Save annotation draft for later
  - This is to ensure that if it is late in the night or the user has to do something, so they can save the annotation and finish it at a later date.
- Two types of comments on annotations
  - The two types of comments on annotations (private and public comments). The private comments will be seen just by the approving manager and the public comment that all users will be able to see on the annotation.
- Easy way to bulk upload to AmiGO
  - We will create a system that bulk uploads all our approved annotations to

AmiGO's database every so often (every week or so) to ensure that AmiGO has our most up-to-date revisions. Furthermore, this means that we will have to design our database to work with AmiGO and be able to easily plug our data into their database.

2.2 User Level Hierarchy

There will be four user levels. The first user level will be unregistered users. These are the users who have yet to register for the site and will have very basic permissions (e.g. just being able to browse approved annotations). The second user level will be registered users. They are users who have registered on the site and they will have significantly more permissions than the unregistered users (e.g. seeing unapproved annotations). The third user level is the manager they will have significantly more power than the registered user (e.g. being able to ban a user and approve/disapprove annotations). The last user level is the admin and the only thing that distinguishes it from the manager is having the ability to promote a user to a user level. See **figure 1** for a more visual look of how user levels will work.

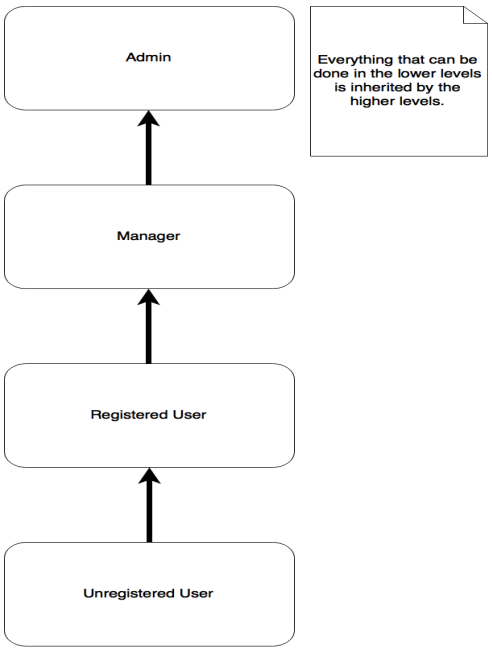

Figure 1: *This diagram essentially shows the various user levels a user can have and based on these user levels a user will have different permissions and access. The idea is that the lowest level user can be found at the bottom and as you go up in the hierarchy the user gains access to more permissions and roles. Furthermore, it also retains all the permissions and roles defined in previous layers. So this diagram serves as a gateway for the user to essentially understand the flow of user levels.*

2.3 Role Based Access Control Design

The Role Based Access Control (RBAC) design is yet another crucial stage in our design process, because designing a well structured RBAC design will make expanding user roles easier and will make our database much more secure, because access permissions are more separated.

David F. Ferraiolo and D. Richard Kuhn state that the primary descriptions are as followed:

- *User_Roles(database) = {Roles for the database}*
- *Role_Access(database) = {Access permissions of each role}*
- *Transaction_Access(Role) = {transaction permissions for a role}*
- *exec(database, transaction) = {returns true or false depending if the user has access to executing the transaction to the database}*

Furthermore, David F. Feraiolo and D. Richard Kuhn also state that the three following rules must apply:

1. ∀ d:database,t:tran,(exec(d,t) ⇒ User_Roles(d) ≠ empty set).
    - That all active users must have a role.
2. ∀ d:database,(User_Roles(d) ⊆ Role_Access(d)).
    - This essentially says that users can only take on roles that are defined.
3. ∀ s:database,t:tran,(exec(d,t) ⇒ t ∈Transaction_Access(User_Roles(d))).
    - This pretty much just ensures users can only execute commands they are authorized to execute.

Both of these sets of bulletins were taken from source [9]. Using the rules above I have developed a basic RBAC diagram example for our web portal, which is shown in **figure 2.3.1**.

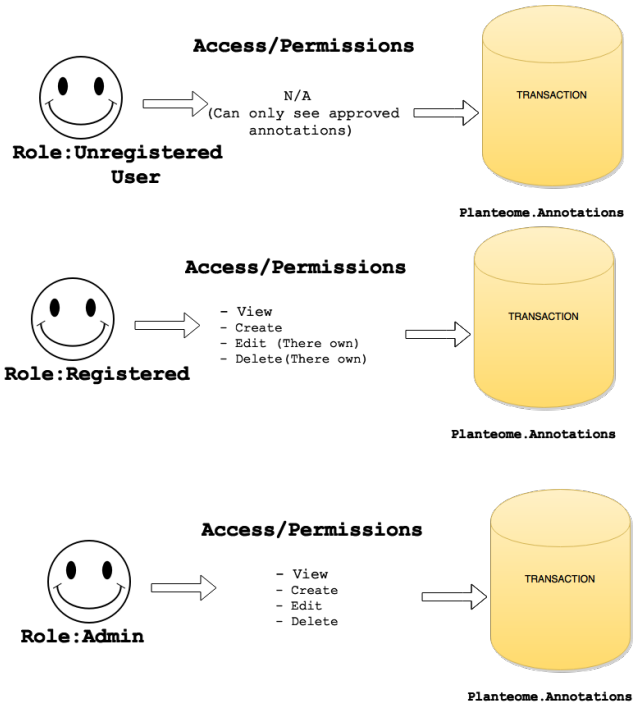

Figure 2.3.1: *This diagram shows the basic example of how our RBAC system was designed for our database/web portal and is for the annotations table (this table shows all the annotations, so not just approved).*

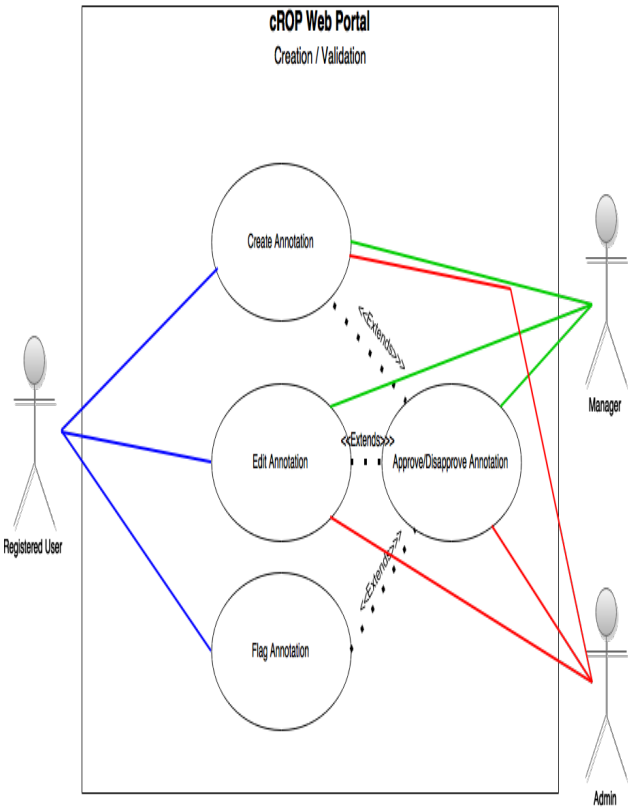

Figure 2.4.1: *This diagram shows the basic process of creating, editing, and flagging (e.g. All three of these actors can create, edit, and flag annotations, but only the managers and admin can approve/disapprove an annotation). Purple lines representing unregistered users, blue line representing registered users, green line representing managers, and the red line representing admins. One can also see that each time an annotation is edited, flagged (a certain number of times) and an annotation is created it will be prompted to be approved or disapproved (represented in the use case diagram by extend) by a manager or admin with a specialty in that area.*

2.4 Use Cases

Use case diagrams are crucial in software engineering, because they allow for you to create basic diagrams that tell you exactly what every users or system can do. Then since these diagrams are basic it allows for clients to easily understand what you believe each user or system should be able to do. In our web portal there are four main actors. The four main actors are the four user levels. In **figure 2.4.1** & **figure 2.4.2** you can see the various actions that each user level is has the permission to do.

Figure 2.4.2: *This diagram shows the various actions that the various levels of users can do. Purple lines representing unregistered users, blue line representing registered users, green line representing managers, and the red line representing admins.*

2.5 User Interface Design

User Interface (UI) design is a very crucial part of the design process. It helps one ensure that the user can do and view everything that he or she wants to. When designing the view you have to think about what would be intuitive for the user and as well as what will be best for the user. You can see in **figure 2.5.1** - **figure 2.5.7** various UI designs for our web portal.

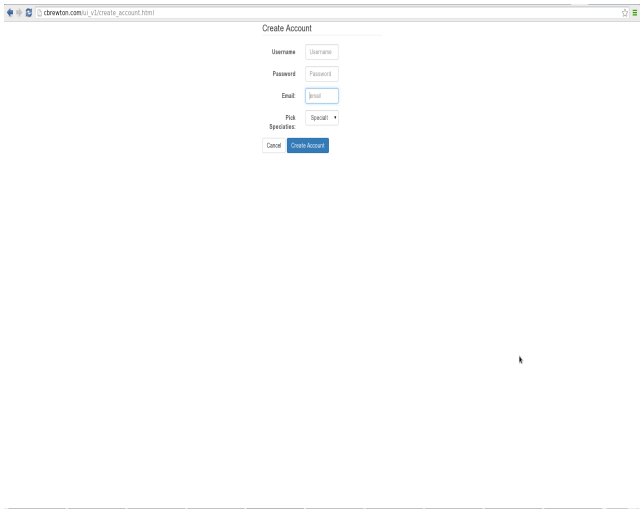

Figure 2.5.1: *This UI diagram shows the basic UI design for the create new account screen. It allows the user to put in a username, password, email, and them to select a specialty. The user level will be automatically set to 1, when a user registers and both the last_login_ip and last_login_timestamp will be set to the current IP and time stamp respectively, so these table attributes are not needed to be manually inputted by the user. Once the user hits create if all field inputs are valid, the users account will be created and they will be prompted to login. The user can click the cancel button and it will redirect them to the login screen, where they can proceed as an unregistered user (can only see approved annotations).*

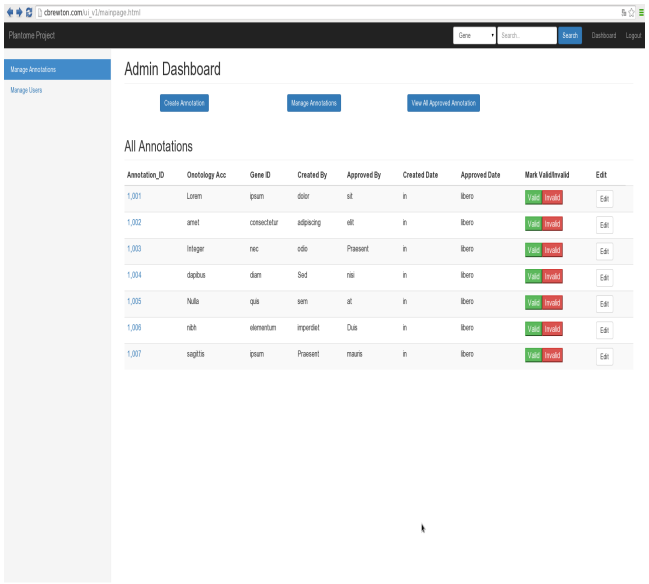

Figure 2.5.2: *This UI diagram shows the basic admin dashboard where all annotations are*

being displayed. One can see the annotations attributes as well as a clickable ID link, invalid/valid flag buttons, and an edit button. The clickable ID link for each row will take you to a more detailed description of that annotation. The invalid/valid flag buttons are used to either flag an annotation as either valid or invalid. The edit button allows you to edit an annotation. Furthermore, one can also see three buttons at the top. A create annotation button, a manage annotation button and a view all annotation button. The create annotation button will take you to the create annotation screen. The manage annotation screen will take you to all the annotations that need to be reviewed. The view all will allow you to see all annotations. There is also a left side bar that has manage annotations and manage users. The manage annotation tab will take you to this view, while the manage user will take you to a similar view, but deal with users rather than annotations.

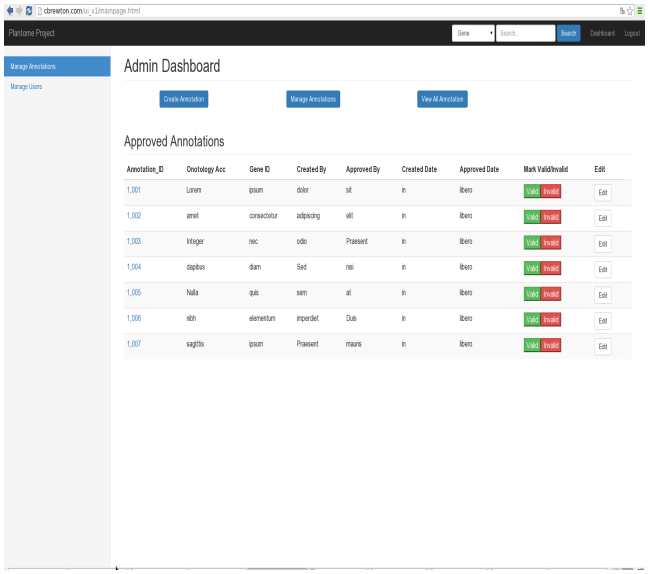

Figure 2.5.3: *This UI diagram is only different from the previous UI diagram in that it shows all approved annotations after the approved annotation button has been pressed.*

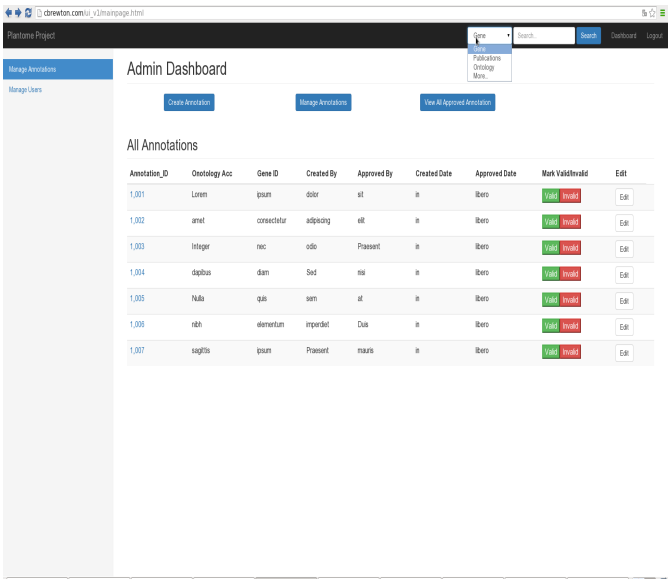

Figure 2.5.4: *This figure is only different from the previous picture in that it shows that one can use the drop down menu to search by particular things (e.g. publication, annotations, ontology, etc.).*

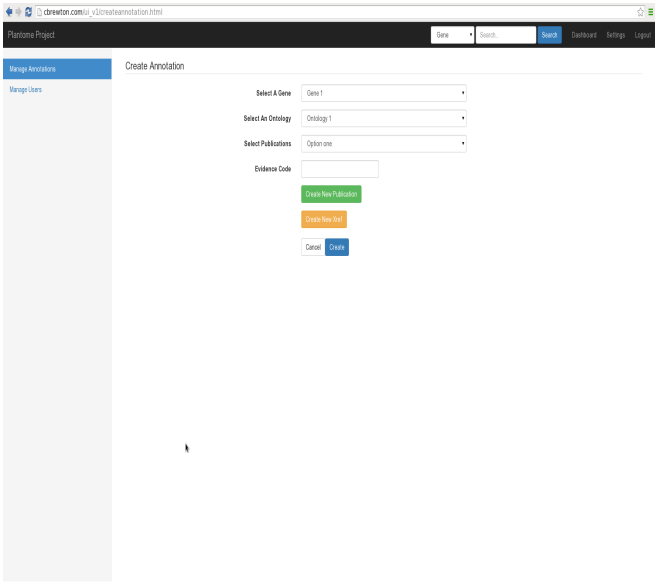

Figure 2.5.5: *This UI diagram shows what one will see when they click the create annotation button. The user can select a gene, select an ontology, select a publication, add an evidence code, and create both publications and xrefs. Once all fields have been entered a user can click create and an admin will be flagged to review this annotations for approval or disapproval.*

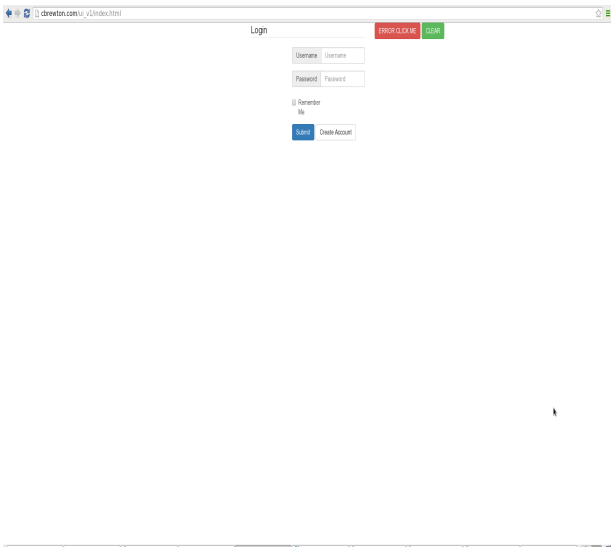

Figure 2.5.6: *This UI diagram shows the basic login page. (Please disregard both the error and clear buttons. The error button was to show when an error occurred on login and the clear button was to clear the error shown.) The user will be prompted for their username and password to login. If all fields are entered correctly the user will be logged in otherwise the user will be alerted they entered their information incorrectly. If the user does not wish to login they can proceed to just viewing approved annotations with very basic functionality.*

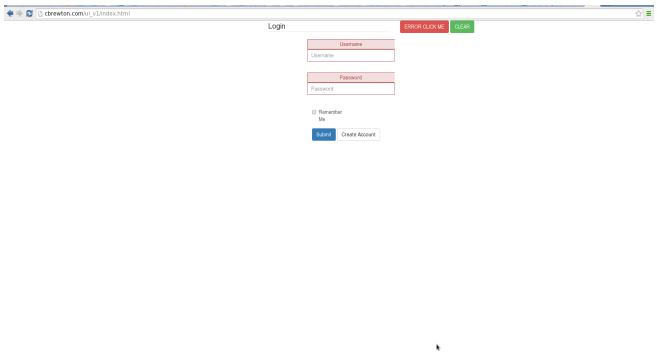

Figure 2.5.6: *This UI diagram shows what happens when a user inputs their username or password incorrectly.*

2.6 ER Diagram

ER Diagrams are crucial to the database design process, because it allows for a simple and concise way to show the relationships between tables. It allows for even the most basic of designers to understand how everything relates. In **figure 2.6.1** you will see the ER diagram for our database.

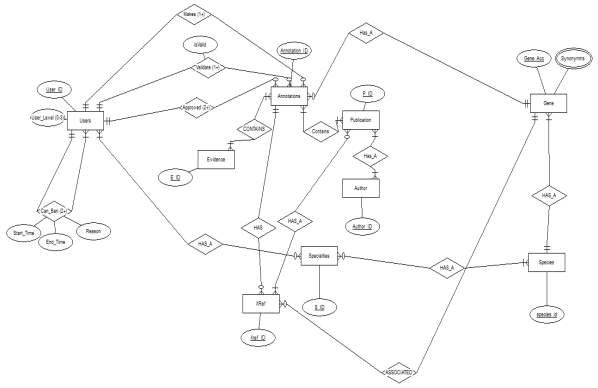

Figure 2.6.1: *This diagram is a basic ER diagram of our web portal. Where we show the relationships (diamond shaped) between entities (box shaped). One can read a relationship like this, the entity gene has to have one species and that species is mandatory. Another example could be that the entity gene has many annotations or no annotations, so we call that optional to many.*

2.7 Table Diagram

The table design for our web portal consist of eighteen tables. Each one of these tables correlates to something on our ER diagram. Whether it be an entity, relationship or a multi-valued attribute. The following tables represent the tables in our database, while the first column is the attribute name and the second column is the attribute description.

Users

ID	Unique ID
Username	Unique Username
Password	Password
Email	User's email

Last_IP	The IP of last login
Last_Login_Timestamp	The time of last login
User_Level	Can be 0,12, or 3 (Defines the level of the user)
Credit	The credit for the contribution of the system (maybe split it into its own table)

This table represents the user table. The primary key is the ID and it is used to distinguish each user in the table, since from what I can tell the username attribute is not unique.

User_banned

Banned_id	Unique ID
Manager_id	Manger who banned the user
Start_time	Start ban time
End_time	End ban time
Reason	Reason for banning

This table is used to keep track of user bans. It references both the manager ID (ID from User table) and the banned users ID (also the ID from the user table). Both the user ids in this table combine to make the unique key.)

Specialty

id	Specialty ID
name	Specialty Name
Species_id	Species ID

This is the specialties table. It keeps track of the various specialties that are referenced to a particular species. The specialty ID is the unique attribute in this table.

User_Specialty

User_id	User ID
Specialty_id	Specialty ID

This table is used to reference the (many-to-many) relationship between users and specialties, since a specialty can have many users and a user can have many specialties. The user_id and the specialty_id combine to make a unique ID.

Gene

Gene_ACC	Unique Gene ID
Gene_name	Gene Name
Gene_description	Gene Description
Chromosome number	Genome_location
Start	Chromosome Start
End	Chromosome End

This is the gene table and it contains the various attributes one would need to know about the gene. The unique ID is the Gene_ACC, which stands for the gene accession ID. This is referenced in various tables (e.g. The synonym table, gene_species table, the annotation table).

Synonyms

Gene_ACC	Unique Gene ID
Synonym_name	Synonym Name

This table is used to store the one-to-many relationship between gene's and synonyms. The Gene_id and the synonym name together make a unique id.

Species

Species_id	Species ID
Species_name	Species Name

This table represents the species entity. It contains the data that is associated with species. The species ID is the unique attribute in this table.

Gene_Species

Gene_ACC	Gene ID
Species_id	Species ID

This table represents the Species (one) to Gene (many) relationship. We could have just stored the species in the gene table, but decided that just in case we want to expand the information we had on species to keep them as their own separate entities.

Annotation

Index	Unique ID
Annotation_ID	Annotation ID
Ontology_ACC	Ontology ID
Gene_ACC	Gene ID
Created_by	Who added the annotation (user_ID)
Approved_by	Who approved (user_ID)
Created_date	Date submitted
Approved_date	Date Annotation Approved (Can be null)
Comment	every people can read comment
Note	Private and not readable for others
Draft_flag	0: in processing 1: submitted

This table is the annotation table. The annotation ID is not the unique ID, because there could be multiple edits of the same annotation, so we added an index attribute to be the unique ID for this table.

Annotation_Validation

Annotation_ID	Annotation ID
User_ID	User ID
isValid	-1 invalid or 1 valid
date	Date
reason	Reason why flagged

This table is used to keep track of the valid to invalid ratio of annotations and to also keep track of what users flagged this annotation invalid/valid. The idea is that invalid is -1 and valid is 1, so essentially they will cancel one another out and if the value is negative (more people flagged invalid) and if it is positive (more people flagged it valid). The User_ID and the annotation ID make a unique ID.

Approved_Annotations

Annotation_ID	Annotation ID
Ontology_ACC	Ontology ID
Gene_ACC	Gene ID
Created_by	Who added the annotation (user_ID)
Approved_by	Who approved (user_ID)
Created_date	Date submitted
Approved_date	Can be NULL who approved it
Comment	Everyone can read this comment
Note	Private and not readable for others

This table just contains the approved annotations. It is the same as the annotation table except that it doesn't contain the index since the annotation_id is unique in this case. We decided to split up the approved and all annotations to speed up query time if we just wanted to get just the approved annotations, which could happen quite often because unregistered users can only see approved annotations.

Evidence

Evidence_ID	Evidence ID
Evidence_Code	Evidence Code

This table is the evidence table. It contains the evidence ID and the evidence code. The evidence ID is the unique attribute in this table.

Annotation_Evidence

Annotation_ID	Annotation ID
Evidence_ID	Evidence ID

This table is the relationship between annotation and evidence (many-to-many). The annotation ID and the evidence ID make a unique ID.

Publications

title	Name of Publication
Publication_ID	Publication ID
Source	Source
Abstract	Abstract Text

This table contains all the information for publications. The publication ID is the unique attribute for this table.

Author

Author_ID	Author ID
Author_first_name	Author's First Name
Author_last_name	Author's Last Name

This table contains information on the author of the publications. We decided to split up the author and publications so users could easily search all publications by the authors first name and/or last name. The author ID is the unique attribute for this table.

Author_Publication

Author_ID	Author ID
Publication_ID	Publication ID

This table contains the many-to-many relationship between author's and publications where an author and publication make a unique ID since we assume an author cannot be on the same publication twice.

Xref

XRef_name	XRef Name
Xref_ ID	Xref_Object_ID

This table contains all the data associated with xrefs (external databases). The unique ID for the table is the xref_ID.

Xreference_relation

XRef_ID	XRef ID
Object_ID	Link to Publication or annotation

This table contains the many-to-many relationship of xref to the object_id, which in this case is a publication ID or an annotation ID, since an xreference can be on an annotation or a publication. Together the xref_id and the object_id make a unique id.

3. Future Work

There could be numerous things added to this gene curation web portal. For example, once we know exactly how they will be querying the data, we could design the database in a NoSQL database to possibly increase efficiency. Furthermore, we could use MapReduce to speed up the run time of our NoSQL database queries, because it would allow things to run in parallel. In addition, we could further better our web portal by creating an application programming interface (API). This would allow other users to make there own applications that take advantage of our data. Lastly, we would like to add a system that takes full advantage of the credit attribute that we have in the user table. We could use our RBAC system to maybe even give the more contributing users (users with more credit) more access permissions or be more apt to get promoted to either admin or manager.

4. Acknowledgment

I would like to personally thank the Distributed Research Experience for Undergraduates (DREU) for giving me this amazing opportunity to research. In addition, I would like to thank Dr. Eugene Zhang, professor at Oregon State, for guiding me in my research endeavor. Furthermore, I would like to thank Dr. Pankaj Jaiswal and Justin Elser and the rest of the Planteome team for providing me with the design requirements for their web portal. Lastly, I would like to thank both Botong Qu (PhD student at Oregon State University), and

Marquis Hackett (Undergraduate from University of North Carolina at Chapel Hill) for helping me improve my the database design by actively giving feedback and insight.

5. References

1. Ashburner M, *et al.*, "Gene Ontology: tool for unification of biology" *Nature Genetics,* Vol 25, May 2000, 25-29. [Publication Link]
2. El-Diraby T, *et al.,* Domain Taxonomy for Construction Concepts: Toward a Formal Ontology for Construction Knowledge *Journal of Computer in Civil Engineering,* October 2005, 394-406. [Publication Link]
3. Smith B, *et al*, Ontology: Towards a New Synthesis, October 2001, 3-9. [Publication Link]
4. Welty, C, "Ontology Research" AI Magazine*,* Vol 24,#3, 2003, 11-12. [Publication Link]
5. Tweedie S, *et al*, "FlyBase: enhancing Drosophila Gene Ontology Annotation" *Nucleic Acids research,* Vol. 37,2009, 555-559. [Publication Link]
6. Jaiswal P, *et al,* "Gramene: development and integration of trait and gene ontologies for rice" *Comparative and Functional genetics,* March 2002, 132-136. [Publication Link]
7. Lens F, *et al*,"An Extension of the Plant Ontology Project Supporting Wood Anatomy and Development Research" *IAWA Journal,* Vol. 33, #2, 2012, 113-117. [Publication Link]
8. Walls R, *et al,* "A plant disease extension of the Infectious Disease Ontology", 2012, 1-5. [Publication Link]
9. Ferraiolo D, *et al,* "Role-Based Access Controls", 1992, 554-563. [Publication Link]

Author Information

Christian Brewton, Student, Department of Computer Science, The University of Alabama.