LawsForMe

Steven Landau, Tory Leo, Talha Azhar

September 29, 2017

* Overview

LawsForMe™ is a web application that will inform users (via email) of laws that exist and that are being passed/proposed that might aﬀect their interests. When someone registers an account on the LawsForMe website their address will be used to determine their state and county. The user is assumed to be in the United States. The user will then enter their interests which will result in Federal, State, and County laws being scanned for possible laws that aﬀect their interest (positively or negatively). There will also be a forum section of the website where users will be able to discuss the impacts of various laws on various hobbies.

* Queries

The queries implemented in phase two are as follows:

* The Account query has been implemented to create a new account for a user - INSERT INTO "user"(username, first\_name, last\_name, password, uuid) VALUES (%s, %s, %s, %s, %s)
* The Address query has also been implemented. The query allows for the insertion of an address into the Address table - INSERT INTO address(street\_1, street\_2, city, state, postal\_code, belongs\_to) VALUES (%s, %s, %s, %s, %s, (SELECT id FROM "user" WHERE uuid = %s))
* The IsUsernameTaken query. This query allows the program to check whether or not a username is already registered in the database - SELECT 1 FROM "user" WHERE username=%s'
* The VerifyCredentials query allows the program to determine if the credentials a user has entered are correct. - SELECT 1 FROM "user" WHERE username=%s AND password=%s

All of the queries implemented so far allow the program to accurately and smoothly handle user account creation and login.

* Requirements
  + Users must be able to register, giving a minimum of their email, and state.
  + Federal and Statelaws need to parsed in search of data relating to users interersts.
  + The website must have a fully functional forum section where users can discuss various laws.
  + Users must be emailed whenever a law is being proposed that could aﬀect their interests.

4 Progression

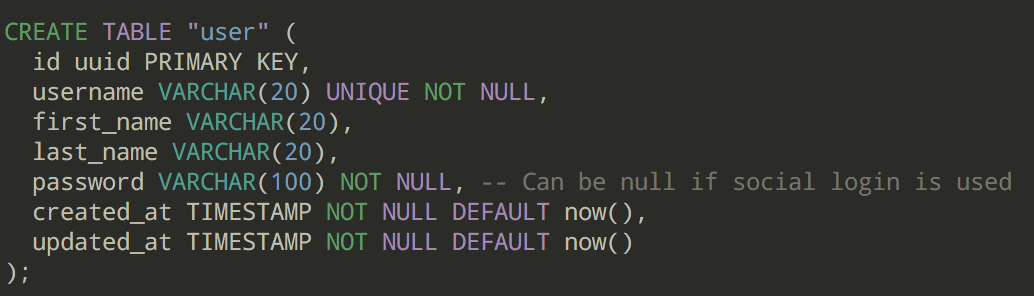
* + GitHub created for LawsForMe project
  + Database created on Reddwarf with tables
  + All team members have logged into Reddwarf
  + Created UML to lay out the design of the program
  + CSS made for the project
  + HTML written for the main page. Starting writing Javascript for login
  + HTML written for the account creation. Started writing Javascript for account creation
  + Account registration queries written
  + Account validation queries written
  + Backend switched to Django
  + Backend for user registration/login written and tested

5 Use Cases

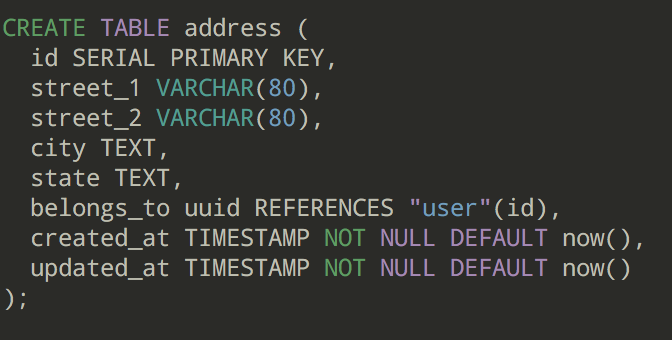
* + User enters Email/Password to login
  + User enters invalid Email/Password and notification is brought to login page with a warning, notifying them that it is invalid
  + To create an account, the user enters Email, Password, Address, if the credentials fail, user will be notified that the credentials are invalid and prompted to enter them again
  + Once the user logs in, they can select what interests, out of a list, by clicking on the elements of the list
  + Based on the user’s interests, laws will be listed, that effect their interests
  + User can click “OK” button to be brought to the law list page from the interest list page
  + User can click on the law and be taken to a page with more information related to the given law
  + User can hit the “back” button when on law page, to return to the page that lists laws
  + User can click on an icon on the law list page to be presented with a list of interests, which they can interact with
  + User can go to an information update screen to update their address
  + User can click the “Logout” button to be signed out of their account

6 Table Normalization

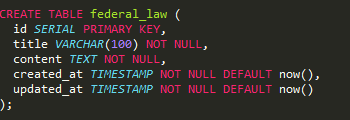
There is no need to normalize our tables for they are already in the optimal state.



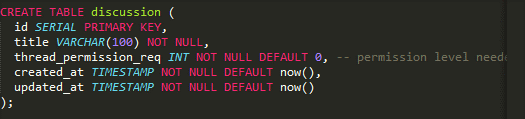
As shown above, the main way to access a row in the “user” table is via the users id. All values in this table are atomic and no value infers anything else (aside from the id and username granting identification to the row, however all attributes can be accessed from either key).



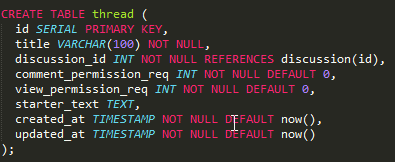
The address table is also as “normalized” as possible for nothing in it implies any other part except for the id which is used to determine the row of the table.



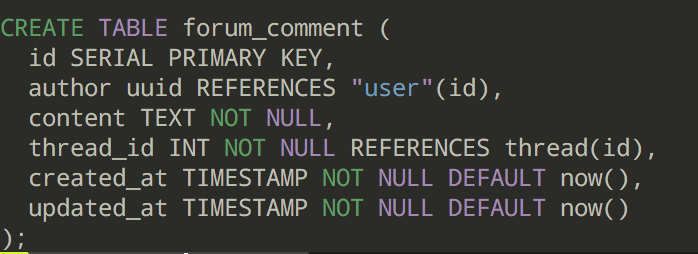
The federal law table is extremely straight forward whereas an id uniquely identifies a row in the table. Each row contains the title of a law and the content (description) of that law. There is no way this can be reduced any farther.



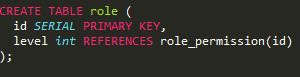
The discussion table is used to house data and organize a group of threads. Threads (coming up) have a reference to discussion that allows for determining what discussion a group of threads belong to. A discussion itself has an id which is used to uniquely identify a row, along with the title of a discussion and the permission required to create a thread in that discussion.



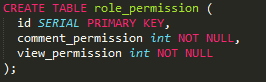
An individual thread is part of a discussion and is used to house a discussion about a given topic. The id of the thread uniquely identifies it. A thread has a title as well as a discussion\_id which is used to identify the discussion the thread belongs to. Threads also contain attributes that house the permission required to view and comment on them, as well as the text that the thread has been created with.



A thread can house numerous comments. Comments are created by “users”. Each comment is uniquely identified by it’s id. Each comment maintains a reference to the author that posted/created the comment as well as the content of the comment itself (words the user wrote). The thread\_id is also referenced to identify the thread of which this comment belongs to.



The role table is used to identify the role of a user (normal or admin). Each row in the role table is uniquely identified by it’s id. The level integer corresponds with the id of the role\_permission table which houses the permissions at which a given users role grants them. The role table has yet to be fully implemented with the “user” table.

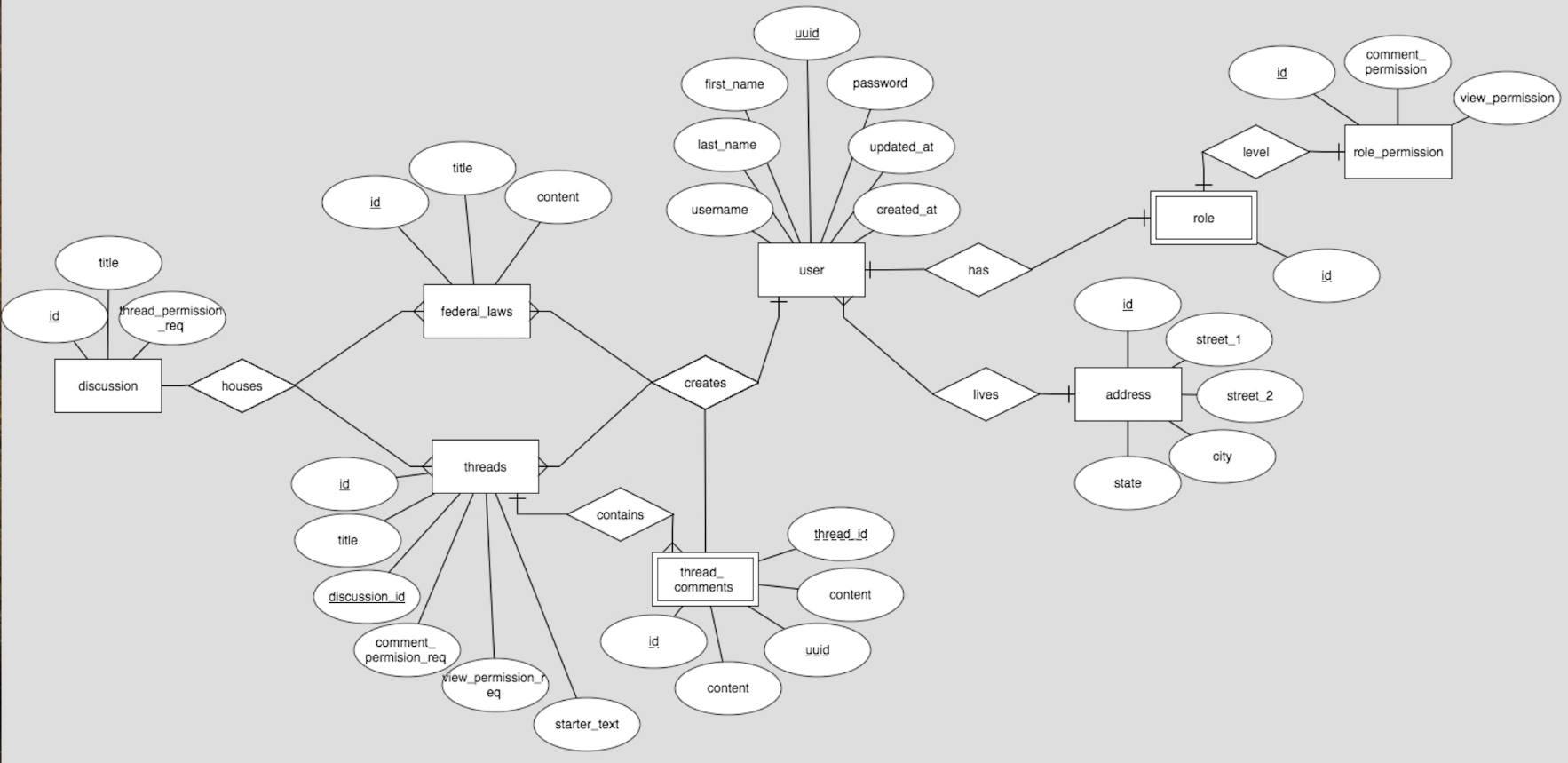


The role\_permission table specifies permission levels for each role. The id of each row doubles as the unique and sole identifier of the table as well as the number indicating the overall level of the role. The other permission attributes contain information about what content the role is allowed access to.

Since there are no transitive dependencies among keys in any table of the database, the database is in 3NF, and since there are no transitive dependencies among non-keys in the database, then it is in BCNF. There are some tables that have two attributes which can identify a row however only one of them are needed to access a row, making the minimum candidate key either of the attributes (typically the id is chosen).

7 Table Reduction

The following diagrams show how the database was conceptualized:



The ER diagram was reduced to the following tables:

User = (uuid, first\_name, last\_name, username, password, updated\_at, created\_at)

Permission = (id, level)

Role\_permission = (id, view\_permission, comment\_permission)

Address = (id, street\_1, street\_2, city, state, belongs\_to)

Federal\_laws = (id, content, title, thread\_permission\_req)

Thread = (id, title, discussion\_id, comment\_permission\_req, view\_permission\_req, starter\_text)

Forum\_comment (thread comment above) = (id, content, author, thread\_id)

Discussion = (id, title, thread\_permission\_req)