Project Final Report

Ian Waldschmidt [isw5@case.edu](mailto:isw5@case.edu) EECS 341 - Gurkan Bebek

Justin Lee [jpl88@case.edu](mailto:jpl88@case.edu) Introduction to Database Systems

Jacob Alspaw [jaa134@case.edu](mailto:jaa134@case.edu)

**1. In at least two of the operations in your system, use transactions to ensure that the applications work correctly even if they encounter unexpected events. Document the transactions in the applications.**

--Creating a new user

START TRANSACTION

INSERT INTO Users (username, password, email, country, birthdate, gender)

VALUES(‘ISW5’, ‘myPass’, ‘isw5@case.edu’, ‘United States’, ‘1996-26-03’, ‘M’)

IF @@ERROR <> 0

ROLLBACK

ELSE

COMMIT

--Updating user password

START TRANSACTION

UPDATE Users

SET password = ‘newPass’

WHERE username = ‘ISW5’

IF @@ERROR <> 0

ROLLBACK

ELSE

COMMIT

--Adding a comment

START TRANSACTION

INSERT INTO History

VALUES(id, username, mediaID, accessed)

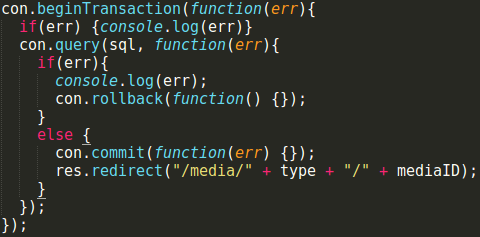
IF @@ERROR <> 0

ROLLBACK

ELSE

COMMIT

An example transaction looks like this:

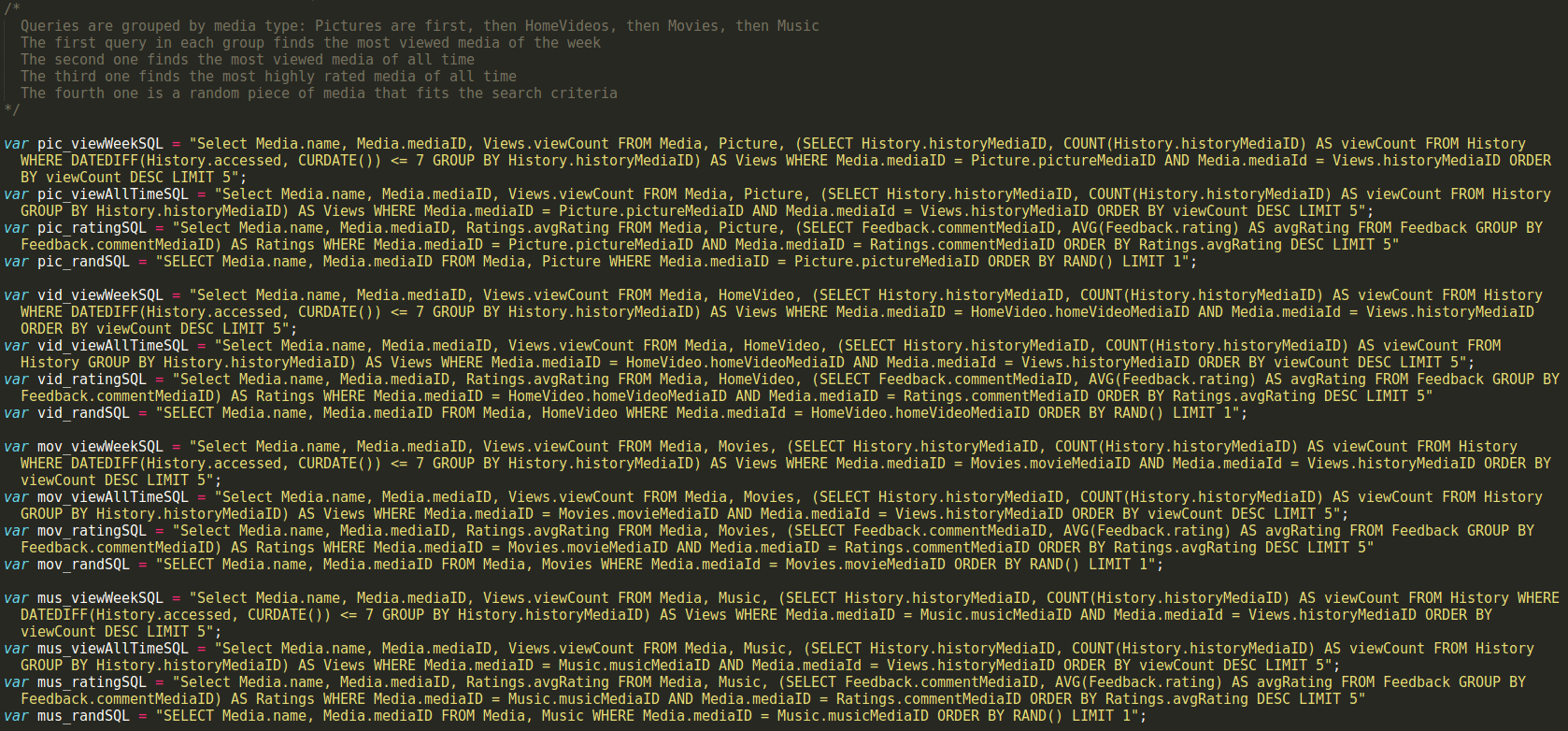


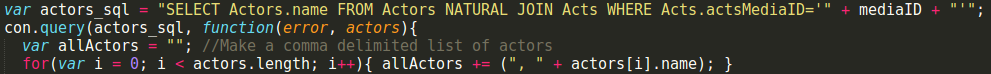
We tried using SQL transactions, but those attempts failed because Node.js handles transactions differently. Instead, the MySQL plugin to Node.js uses the above structure for handling transactions. There are built in functions on connection objects that handle transaction processes such as BEGIN, ROLLBACK, and COMMIT.

**2. Document your program. Highlight design decisions. Provide a user manual for your software.**

Our code is commented thoroughly for extra readability. We have no new design decisions to report because all design decisions we made recently are noted in the earlier documents since we changed our project a bit towards the end. A separate user manual is attached with the rest of our submission.

The TA suggested that we put some examples of queries we used to give you a better idea of what we do with the database in our code. The following pictures are examples of a couple queries involving aggregate functions and joins.





**3. Provide database assessment, i.e. validation of database against client/user/database objectives, list of weaknesses and limitations, list of potential future upgrades etc.**

Our current version of Night Owl serves the core purpose that we set out to achieve which was to create a place where user’s can upload and share media of different forms with others. Furthermore, users are allowed to give feedback to this uploaded media in the form of comments and ratings. However, due to a limited amount of time we weren’t able to make the final project as perfect as we had hoped, so there are a couple weaknesses and limitations. The first is that, for user authentication and login sessions we had hoped to use Passport.js and Express-Session.js which are very secure in what they do, but we had a fair amount of trouble implementing these this with a MySQL database because almost all the implementation guides available were with MongoDB (A NoSQL database). In lieu of Passport.js and Express-Session.js, we used cookie,js which allowed us to send the username between pages in the form of a HTTP request cookie header. While effective for our final project this method is not secure enough to actually post to the internet. Secondly, when it came to the user interface of the website, while our final project doesn’t look terrible it would be nice to flesh out the website to look more professional and clean; however, because this wasn’t the main focus of the project and we had a limited amount of time to finish the final product we put more of our time into trying to perfect the functionality over the design. Lastly, one problem we came across late in the development process was determining what type of media (Music, HomeVideo, Picture, or Movie) something was based off of their mediaID. This is very necessary for sending the user to the correct URL to load the media (ie. /media/music/…. vs /media/movie/…..), while we did get this to work in roundabout ways it would have been better to add a attribute into the database for the mediaType for easier access. One feature we had planned on implementing but didn’t complete (we edited our previous reports to reflect this after emailing the Prof about it) was a subscription based premium user. A premium user would need to pay a monthly fee and would have access to uploading private media. This would require secure and encrypted billing tables and permissions to media based off of ownership and user “level”. Another potential future upgrade would be to implement more interactive user pages which would allow for friending people and displaying a feed, similar to facebook, of activity done by your friends, such as media viewed, uploaded, or commented. Another, thing we would want to add is to add more database validation to our queries. We would do this using the built in Node.js transactions that we used for the required 2 queries for most of our queries to ensure if the first query works and the second one fails we rollback. Lastly we would like to upgrade the visual design of the website as to make it look more modern and clean.

**4. List of Project Members and a brief description of their roles.**

We met collectively to finalize the project over many days in the North Residential Village. The website design was split evenly among the group members. The team collaborated and communicated successfully to choose which components we should use in the database’s design and the web service’s implementation. Jacob tended to create the queries, lead the documentation of our design, handle graphic design elements, and populated the database. Ian and Justin focused on creating the database, finding third party tools to develop the NightOwl service, and connecting the front-end web service to our database. Everyone contributed to the front-end system.