Introduction

ISTM 622 has been the class in which I have learned the most. From Perusall videos, in-class discussions, and most importantly this project, I have understood the basics of SQL like never before. I have worked on database management before. I have extensively worked on ETL, Electronic Data Interchange, and Application databases but I never knew the term for it or the concepts behind it. I have only learned in this class what this means, how to use them, and why I used them in my work.

At the start of the semester, when we were first discussing normalization, I struggled with understanding why normalization was so important in relational databases since I have used denormalized data extensively in a relational-like model. Now, at the end of this course, I have landed on the infamous answer – it depends 😄. When I was using denormalized data in SAP's application database it was because we were relaxing consistency to speed up queries. Surprisingly, that was the ask from the client and I never realized back then that consistency was always the trade-off for speed. But the number of times I rewrote queries to optimize joins or simply denormalized data to optimize queries makes me realize that I have known these concepts for a while but just never knew their name and theoretical connotations.

The milestone project has not only solidified these concepts for me but also helped me learn so much more about how databases at large work. To begin with, I had never worked on EC2 or Linux systems before. After all the struggle with the clustering milestone, I can say I'm at least at an intermediate level for both of those tools. It's been a fun experience exploring the various functionalities of EC2 as well. I would never have had such an opportunity if not for this project.

I believed the JSON milestone would be the easiest for me since REST APIs were my bread and butter at my work till I realized that MongoDB only takes one JSON file at a time and I had created 4077 JSON files. In classic me style, I got stuck at the easy part after finishing the difficult part in a mere 30 minutes. I took an unnecessarily long path to solve this issue, which I will talk about later, but at least I know how to run loop commands in the Windows command prompt now.

What I mean to say is that over the objectives of the milestone, I also learned a lot about how different OS work with MariaDB, how EC2 operates, and even peculiar things about Windows.

What I learned from each milestone

Lessons Learned: Infrastructure

The infrastructure milestone was our introduction to the world of EC2 and Linux operations. As I mentioned in the Introduction, I have never worked on either so I was very nervous about getting something wrong. The Server Infrastructure Demonstration video was my guide and bible to follow. I also followed the demonstration given in the class side-by-side with Dr. G so that I would not have to struggle later on. Well, I got lost in between and so the video was the go-to solution.

Learning Linux commands was easier than I thought. They are very intuitive and you can easily google them if you do not know what to do. The part that was tough to understand was the difference between the dgomillion Linux user and the MariaDB user. It was only till the clustering milestone that I fully understood how these users worked and why we needed them.

Since we followed a step-by-step tutorial for this milestone, I don't think I understood any of the commands properly while creating the infrastructure. I only followed the steps and hoped that the auto-grader would recognize the connection. That being said, I think the video was needed as there were many students like me who had never even seen the interfaces for these tools, and we did learn how to properly configure everything later on anyways.

Lessons Learned: ETL

The ETL milestone was a little easier for me. It was in the realm of what I had worked with before, thus I was glad to be using familiar commands like create table, insert into table, etc. Yet, there were a lot of commands I had never worked with.

It was interesting to see how files from the auto-grader were being downloaded into a cloud environment for us to use and how zip files behave in Linux. There was also the matter of understanding what the data looked like. I think we were all terrified of unknowingly changing the data when opening it on our local systems. But nothing like that happened. The data was comprehensive enough in the way that I understood what we were trying to achieve with it. The challenge came in extracting and transforming it to fit the structure provided by Dr. G.

'Load data infile' was a new command I learned. I initially tried the entire milestone on my local system before working on the EC2 instance. On the local system, the 'Load

data local infile' command doesn't work properly which was the instructed command in the milestone preparation document. Thus, I had to learn what the difference between the two commands was and why the local was not working in my personal setup. After figuring that out, the milestone was mostly a piece of cake.

Another thing that slightly surprised me was learning that I could nest REPLACE() functions inside one other. There were obviously many ways to crack that particular problem but to me, the nested replace statement seemed the easiest and most concise.

Another thing I learned was the difference between the 'Insert into … Select' statement and the 'Select into' statement. I initially thought they were the same command and should work the same but that was not true. The former simply copy data from one table to a different one whereas the latter creates a new table and transfers the data in it. Thus, 'Select into' was not the right command for our needs,

Lessons Learned: Views

The views milestone was the easiest in my opinion. The ask was self-explanatory and there weren't a lot of unexpected obstacles. Sure, there were some complex joins to take care of but it was still pretty straightforward.

The first thing we learned from the customer names view was working with aliases. I have worked on views before so this was easy enough. Next came a single join, which was also easy enough to achieve. I think at this point we were all feeling very proud of ourselves not aware of the absolute confusion we were about to face in the next view.

Welcome to a quick lesson on the difference between concat() and group_concat() that if I had known earlier would have made my life easier. "The difference here is while CONCAT is used to combine values across columns, GROUP_CONCAT gives you the capability to combine values across rows."[1] A single statement that took me too long to find. Nevertheless, like all the other problems faced during the time of this milestone, Google, my best friend, came through in the end and I solved the silliest issue of all time.

Creating materialized views was an easy task and indexing was barely a sentence. Working with the joins was laidback as well because there were no tricks like using a right join or focusing on where to use a left join and where to use an inner join. A simple left join worked wonderfully to connect the tables and that was it.

Lessons Learned: Procedures

The Procedures Milestone was definitely trickier than the Views Milestone. It was good practice in learning how alter statements work. I learned what a virtual column is. I never knew we could create virtual columns as well. It makes sense though considering that there is no need to store calculated values in the table.

I also learned about naming conventions for foreign keys. I hadn't realized that a unique name is given to identify them. The dropping of the status table was a nice way to study how these foreign key pairs work.

The fillUnitPrice procedure was easy since we were only updating a single column of a table. But the FillOrderTotal procedure was where I faced the most issues in the milestone. I could not figure out how to join the two tables in an update statement. Funnily enough, I had forgotten the concept of subqueries and could not for the life of me figure out how to use a join inside update. After much unnecessary struggle and many hours on google, I saw a subquery and had a Eureka + I'm so dumb! Moment. I wrote my subquery beside the join statement, creating a virtual table to use to calculate the total. Could still have just run the subquery inside the set statement but I like taking the road not taken and making my life harder.

Lessons Learned: Triggers

This milestone was the worst one. Not because of its complexity but because I scored the lowest in it after successfully cracking everything. Another great lesson to learn when working with scripts; always write 'Use Pos' in your script. I lost way too many points for this very silly error.

The Triggers Milestone was definitely the most difficult. There was a lot to learn and a lot to figure out. There were no comprehensive steps or hints given in the project documentation, and the only way to solve this one was to rely on our own analytical skills. There were so many possibilities and outcomes to consider and so many triggers to create. The trickiest one was the trigger on the OrderLine table.

I learned about using if statements, and signals, and how to refresh materialized views. I had too many arguments with my classmates about using the delete and insert commands to refresh materialized views instead of a direct update command. Ultimately it was a matter of what worked best for our system, but we did a lot of research on which one is better and why.

Lessons Learned: Clustering

The clustering milestone properly taught me the various tools in EC2 instances and how to use them. Here I properly understood the steps we had taken in the infrastructure milestone. I learned about subnets, security groups, what opening a port means, how virtual private clouds behave, and how connected servers look like. I also realized I had not paid attention when I learned about Networking in my undergraduate degree. A lot of these concepts were new to me when they should not have been.

I think I also learned the most about Linux commands in this milestone. During setting up standard replication, my MariaDB repository just wasn't installing properly. I had to learn about uninstalling MariaDB from Linux itself and reinstalling it. Even after it got successfully installed, my grant statements would not work. I had apparently gotten some statements wrong so even though I had corrected my statement, the incorrect one was stored in the cache. I had to use the 'Flush privileges' command to make the correct command work.

Another struggle was figuring out how to change the server id for the replica server. Apparently, the instance needs to be restarted after writing the server id in the server.cnf file for it to take effect. Alas, I did not know that and just used the 'Global Set server_id =2' command instead. It worked the first time. Not the second time I recreated my instance and then finally, at my third attempt I figured I just needed to restart my server.

Galera clustering was easier as there weren't too many steps to follow. It did most of the clustering automatically and there was no worry about checking if I was getting the 'Change Master' command correctly. The only pain point I felt for this was that once the server closed, it would not start back up properly. I'm too impatient to wait a couple of minutes between stopping my instances, so I had to create the Galera cluster 3 times before I submitted it for grading.

This milestone is also where I learned what different users in MariaDB look like and do. Like I accGUIntally found out that apparently the user 'dgomillion@%' does not work for localhost commands. It opens requests from everywhere but the *localhost*. Obviously, this was not a need for the milestone but I explored and experimented a lot on the command line and discovered such surprising facts.

Lessons Learned: JSON

I learned the most in this milestone. There were a lot of new concepts in this one that I had never encountered before. I did not know that MariaDB has in-built functions to generate JSON files directly. I got to learn about the differences between JSON_Object, JSON_arrayagg, and JSON_array. It was actually really fun to play around with the JSON structures and how they nest within each other.

Having worked with JSON files before, I knew how I wanted my final data to look so there was no confusion there. The difficult part was figuring out how json_arrayagg works with json_objects. Once I figured out the arguments that these functions take, it was easy enough to create the correct JSON, at least for a single customer. The next doubt was on how to create multiple files without them overwriting each other.

I thought that prepared statements could only be used in applications but I learned that it is possible in MariaDB too. It also allowed me to use concat() so that I could generate multiple customer files with different names. All I had to do after that was run a simple while loop and the MariaDB part was done.

I had also thought that MariaDB would be the tough part but working with MongoDB was equally difficult especially since there were few resources for the GUI that I found online. A lot of queries available on the internet worked on the mongo console but remaking them for the GUI took some time. I had to learn each aspect of the GUI and understand what different functions do. It took me an entire day and multiple failed queries to fully figure out the components of MongoDB. But it's a very interesting database and I wish we got to explore it a little more in class.

Lessons Learned: Reflection Milestone

To be honest, this milestone is the hardest one yet. Putting into words everything you have learned and done is a hard task. Especially since I come from a programming background, I hate documentation.

It is definitely a good revision of everything we have achieved this semester. It reminds me why I enjoyed the class so much. I got to revise all the new commands we learned during the course of this class. I also got to reminisce about all the steps and struggles I had taken to complete each milestone.

This milestone is a good practice to end the semester but I would much rather write queries than cohesive words about what I learned on the last day of classes 😊.

Most Difficult

The triggers milestone was the most difficult for me. Very few hints were given and solving it relied more on our problem-solving skills. We were expected to understand how the database behaves and interpret which of our commands would work best to answer the questions laid down by Dr. G. It was also the one where I fully understood my data properly and how the tables were interacting with each other.

The trigger to update qtyOnHand was one of the hardest ones. Many end cases had to be considered and tested for this question. There were too many questions that I was asking myself which led to nothing but more confusion. After racking my brain for way too long, I put my programming skills to task and just wrote the code without thinking about anything. Turns out that the best answer is always the simplest one. My code handled each end case that I had been thinking about, all I had to do was stop confusing myself.

There was some ambiguity present with how many triggers should be created as well, especially for the trigger that had to update the materialized views. It was important to understand how eager updates could be done on the materialized views without fully refreshing the table. As I said before, I had too many arguments on what was the best way to achieve this. Should you refresh from the view itself or rewrite the entire select statement? Should you use delete, insert or update? I found the easiest answer was to use delete and insert using the view to update my materialized view. It worked for all my test cases and it was easier to read.

At some point, I remember getting confused between all the triggers I wrote on OrderLine. The after-update and after-insert trigger on OrderLine would have been the death of me and at multiple points, I almost wrote code in one thinking it was the other. After that, I realized I need comments on my script and bifurcated each trigger properly.

As difficult as the triggers milestone was, I finished the milestone pretty early and then watched multiple people cry over it. It is definitely tough for people who have never worked on SQL or programming before. The only reason I did not struggle as much was because of my experience in both. Though I think I will forever be salty about how the majority of my marks go deducted because of a foolish mistake like not writing ‘Use POS’.

Most Surprising

The most surprising part of the project was the JSON milestone for me. I did not expect it to be challenging, yet it provided issues I never imagined. The freedom to generate our JSON formats meant we had to figure out what components will be needed from our database to answer all relevant questions. I struggled to make the decision to put all my JSON structures in one file or many. I ultimately went with many files because I kept getting confused with how to join the tables so that all my customer aggregates would be generated in one file. I would figure it out one second, and then forget it the next.

I also did not expect working with JSON functions would be so confusing yet so fun. I genuinely enjoyed figuring out how to nest JSON objects inside one other. Understanding the JSON_arrayagg() function was essential to nesting the products inside the order structure. My biggest advantage in the milestone was my understanding of JSON structures. I knew how to read and write them. Hence, I knew exactly how I wanted my data to look from the start. Then, it was just a matter of trial and error till I was satisfied with the final product.

I believed the JSON milestone would be the easiest for me since REST APIs were my bread and butter at my work till I realized that MongoDB only takes one JSON file at a time and I had created 4077 JSON files. In classic me style, I got stuck at the easy part after finishing the difficult part in a mere 30 minutes. I took an unnecessarily long path to solve this issue. I was working on my local machine for this milestone and did not have the luxury of using the ‘cat’ command (not that I had thought of it). Instead, like all things mysterious, I decided to use trusty Google’s help. I love Google but sometimes it really likes to complicate things. Instead of simply giving me the ‘type’ command (which is the Windows alternate for cat), I ended up downloading mongo database tools and using a long-winded command that looped on the folder where I had stored my files and imported each one into MongoDB. It took a total of 30 minutes for this task to complete. 30 minutes of my life that could have been solved in a mere minute. But hey! At least I know how to run loop commands in the Windows command prompt now.

Another surprising thing to me was just how interesting MongoDB is. I wish we had gotten more time to play around on it.

Favorite

My favorite milestone was the ETL milestone because it was the easiest. No, I'm kidding. My favorite was JSON itself. I loved working on that one because it made me feel really smart. I was able to complete the MariaDB part within an hour and then I played around with the MongoDB GUI for 2 days straight.

I learned a lot about document databases and in-built functions in MongoDB and MQL. It was interesting and interactive and gave a different perspective to our data. I loved understanding more about how indexing works inside JSON files, how to query aggregate functions inside an aggregate, how to filter aggregates, and how to add limits to the number of aggregates I want to receive. Even though we only had to answer one question for the milestone, I tried different queries to answer questions like which customer had the maximum number of orders, which customer spent the most, how much each customer has spent, what products each customer has bought, etc.

In fact, learning MongoDB and DGraph, for my class presentation was the most fun and insightful part of this class. We have been studying NoSQL concepts but only while working on them personally did I understand how different it is from a relational database and yet the data remains the same. I finally understood what it means to say that the way we perceive our data changes because ultimately the data we were trying to store remained the same, the only thing that changed was how we accessed it.

Why I think this project was useful

This project was useful because it allows us to dive hands-first into the world of SQL and experience data management outside the realm of texts and videos. Only while doing the project by myself did I understand the difference between relational and NoSQL databases. It helped me improve my SQL commands and learn about new technologies and tools like EC2, MongoDB, and Linux.

This project has been the true highlight of this semester. It was rightfully challenging but I got to learn so much more than I could ever have thought of. As I mentioned in the introduction, I have worked on a lot of these concepts before but I never had the right term for them nor did I understand why we were doing what we did. I did not know when I had created a materialized view that that's what it is called. I did not know that I was working on an application database and not an oracle database. I did not know that when I extracted data from my database and created JSON structures from it to send across APIs it is called Electronic Data Interchange and ETL. This class and project taught me that.

This project is the only way I can think of that allows students to put into practice what we learned in class. I do wish though that we had worked with CTE, group by and some other aggregate commands a little more so that we could have mastered the skill of querying as well. But the class is known as a Database Management class and not Advanced SQL so that makes sense.

I would have loved to work a little more on MongoDB or some other NoSQL databases, to get a better idea of how they differ from a relational database. That is my only feedback: I wish we had another milestone that dived a little more on MongoDB queries. Other than that, this project has been a pleasure to work on and no I did not cry even once while working on it. Ok, maybe once when I did not write 'Use Pos' (I will never let that go), but never because of the difficulty of the project.

How I will describe this class to recruiters

For my ADBMS class, I simulated a POS system to track & calculate sales amounts by creating & managing a cluster of MariaDB Databases on AWS EC2 instances. I used triggers, stored procedures, views, and materialized views to achieve the same. I also migrated data from the relational model to MongoDB by creating JSON files in MariaDB to reduce query execution by 20%.

References

1. https://www.softwaretestinghelp.com/mysql-concat-function/#:~:text=The%20difference%20here%20is%20while,combined%20to%20return%20desired%20results.