Project Reflection

Introduction

In this course, we built a POS (point of sale) system and also inventory management, The data was extracted from the CSV files, which had customer details, product details, order details. Using this data, we started off building on it. We used relational database management system for this, we set up MariaDB from scratch using AWS. We also performed some queries and features to modify, update and insert the data. For quick retravel of the data we explored concepts like triggers and transactions. We performed some analysis to know the customer purchasing trend, products being sold, who was the best customer, where do the maximum customer come from.

We saw how clustering and replication can be done. By forming clusters, the work load will be distributed. And in this we saw that happen, it acts as a data backup as well. The risk of data loss will be less. peer to peer clustering was also used in this project, this concept is particularly used in distributed system. Used for fault- tolerant architecture is essential.

We also learnt about different types of NoSQL datatype and we also learnt about mongo dB, we converted the data to Json files and then used it in MongoDB to retrieve data from it. Overall, I have learnt about a whole new system from the scratch. In companies mostly the database will already be established and we have to just by querying it or working over it, because of this, I did not really have the knowledge of how exactly a database will be set. And the data extraction and cleaning happen. I learnt a lot from this.

What I learned from each milestone

Infrastructure Milestone

In this milestone we are setting up the infrastructure, foundation for the rest of our milestones. This milestone was all about installing MariaDB on an AWS instance, firstly we create a AWS instance, we also set the firewall, we need keep the port 22 open.

Then we gave connection to the instance through the SSH using the private key. The Dgomillion user was created. Then we had to set up MariaDB repository, installed MariaDB using yum. The installation was successfully completed. We have setup public private key authentication for the autograder to establish a communication channel with my AWS server.

Infrastructure milestone exposed me to new ways of doing things like installation, setup, repo files, yum package management tool, and many more. I learned how the database user is different from the system user in linux machine which is wrapped around my database.

There were many first time experiences in this milestione. For example, I have used AWS to create linux machine and got myself familiar with Putty, SSH, and interacting with command line in this milestone. All these experiences setup the foundation for the subsequent ones.

ETL Milestone

ETL stands for Extract, Transform, Load. With this milestone, I not only learned data transformation techniques, but also more about Linux and SSH connections. Firstly, I realized how the public-private authentication we setup in the first milestone helped us to push the CSV files in this milestone also. LOAD DATA LOCAL INFILE command taught me that the LOCAL keyword stresses to the terminal that my desired files are read from the client machine itself.

Coming to the data transformation itself, I have seen in real-time how the raw data is transformed into the format we want in the destination we want. The raw data which was extracted and given to us in CSV format had some inconsistencies on purpose. It included random junk characters like $ symbol to donate currency, where as the databases are currency agnostic usually. So we had to remove that by updating the data in temporary tables. Similarly, we removed commas in price columns to format it according to our needs. We have converted the dates also from literal string to date datatype in the database to store dates in a way our destination database understands. Finally, after this cleaning and transformation, we have loaded the clean data in our desired format to the database tables. This experience taught me how we should always look out for inconsistencies in the data, and our goal always should be make our data make sense once it goes to the database layer.

Database Views and Indexes Milestone

In views milestone, I came to learn how views can be used to improve our read speeds. Since it is a virtual table built on top of the data coming from underlying tables, views will help us predefine joins and help us construct the virtual table for the data of our interest. This saves us time in getting to the information relevant to us. For example, we have used views to relate customers with the orders they bought, products and list of customers who bought each product.

Materialized views are usually stored on the disk. However, in this milestone, I was surprised to learn that MariaDB does not support materialized views and we are making a work around to take the experience of materialized views in this milestone. However, when we took the data from views and loaded it to the tables which we treated as materialized views by following the naming convention mv_tablename for them, I could corelate it with actual materialized views and understood why they are like view, but store snapshots of the results.

Finally, when it came to indexing, the last part of indexing, I realized it is not as hard to implement them as I thought when I first heard of them in the class. All I needed was a simple command to create index and also make sure that I use them for the data of my interest to read my data in a faster and efficient way.

Prepared statements and Transactions Milestone

When I think about concurrent user data access, ACID compliance and Isolated sessions, this milestone was a stepping stone for me. I also learnt how transactions can be leveraged to safely do my work in my own session and see if my changes are taking place or not in the database. It was a big learning moment for me to realize that there is a big undo button inform of ROLLBACK when we use transactions in case of any blunders. I saw how transactions are isolated and do not impact the activities done by other users in different transactions in other sessions until it is committed.

After I learned of the prepared statements, I felt excited to see an efficient way of doing the same things over and again. It provided a reusable template saving the time taken for users to enter the data, and also the data sent over the console to database is less now as you do not enter the same query separately over and again now. Prepared statements help reduce the

load on database massively. Also, calculating the total number of characters it took through prepared statements and separate queries to enter the same number of entries, visually confirmed to me how prepared statements can be more efficient.

From a data security standpoint, prepared statements can be used to protect our database against SQL injections.

Stored Procedures Milestones

After I learned prepared statements, I wondered if there are any effective means to execute multiple SQL statements in sequence. Then I came across stored procedures as a way to do repetitive work easily. Calling the stored procedures was a new thing I learned in this milestone, I wrote the stored procedures in my script and ran it. I was surprised at first and thought I should troubleshoot to make the logic and code in my stored procedures to take effect in the database. Then, I realized that all I had to do was just call the stored procedures I wrote in my script.

In this milestone, I was also reminded that the mariadb does not actually the materialized views features. Thus, we cannot use any direct commands to refresh a whole materialized view which may be available in other databases. To get around this, I learnt that we are deleting the entire data from our mv_tables and loading the new data into them again, technically with the new underlying data. I understand that it is essentially same as refreshing a whole materialized view.

Triggers Milestones

Before I started solving trigger milestones, at one point, it got too theoretical and a little tricky to understand BEFORE, and AFTER keywords for triggers and how they are impacted by inserts, updates, and delete operations in a table.

When I first heard of denormalization, though it is an easier way for faster reads. I thought it would also need more manual work to keep the data updated everywhere it is present. For example, we have a certain price assigned to each product in the product table. But, this same price is present in order line table. Before triggers milestone, I used to think how complex and

messy it can get to update this price in both the places at a time, whenever there is a change in at the origin table efficiently.

But after implementing triggers successfully, I realized that we have an innovative way in the form of triggers to detect any changes in the origin table and update the same in our target tables.

I felt that the scenarios we ran into with the triggers milestone are very much common and possible in a point-of-sale database system. For example, we need self-check constraints like quantity needs to be at least one at minimum if a product is ordered. Triggers can be used to do this check in the order line table, like we did in our milestone.

One big takeaway for me in triggers milestone is that it is highly difficult to debug when triggers do not work. They were fine as long as they were working, but when the triggers ran into issues, I could not figure out if the issue was from the trigger itself or from the logic or from the choice between BEFORE or AFTER trigger for me.

It took me few attempts and had to go through my code line by line multiple times, before I finally realized some fundamental elements of using triggers like, why we should only use a before trigger for delete operations, and when it does not matter to use with before or after. Overall, this milestone stressed the need to do and break things before I learn and inculcate the lessons, I learnt to retain them for a longer period.

Replication milestone

Replication milestone is the turning point in this course for me when it comes to thinking strategically about how we manage data in large scale. I am the kind of person who maintains backups to my backups when it comes to personal files. The idea of learning how it is done at an organizational level in the production systems has always been a topic of interest to me.

It was interesting to learn that through standard replication using primary-replica architecture, we can not only achieve backup systems, but also, we can have multiple instances of databases to read the data from. Though there is a slight delay for the data to flow from primary servers to the replicas, since it will surely come, I have realized it is a good way to split the user traffic to different instances, thereby spreading the traffic across different instances.

After making configuration changes by having my primary server for write operations, and marking replicas for read only, I got the hands-on experience of setting up the stage for standard replication. Followed by this, when I tested my data and checked that the data is flowing from my primary to replicas after every new write I did in primary, I could visually see the delay and also felt assured that data will be there at the end of it. All in all, this milestone is where I got the feel and taste for advanced data management. This took me deeper into managing data, since I was only used to manipulate and manage the pos database till this point.

Peer to peer Clustering Milestone

After successfully finishing my replication milestone, I could not help but wonder about performance advantage once can have if we can write to all three servers involved in replication milestone. Replication delay was also a thing that stayed in my head, as there must be some applications out in the real world which do not tolerate inconsistent reads. Then, I quickly realized that's achieved through something called peer-to-peer clustering. As the name suggests peer-to-peer meaning all the machines involved in the cluster are equals or peers with equal level of access to perform any kind of operations at the server level.

In peer-to-peer clustering, we can write in all the servers involved in the cluster, thereby making it an ideal choice for write-heavy applications. Instead of burdening one server with all the writes, we can split our writes into different instances. Coming to the milestone, virtually synchronous replication offered by MariaDB offers minimal delay in replicating though there is a commit delay and it lets the cluster members know that a write is about to reach them before writing to the servers.

To achieve the clustering in MariaDB, it felt quick and easy, since Galera is already installed along with the MariaDB itself. My biggest key takeaway from this milestone was that I should remember the order in which my cluster members are started and follow the reverse order for shutting them down. When I missed the order once, I had to redo the whole thing and I felt it was also easier to just redo it because of the scripts that were available. The scripting approach also made me realize that we should always optimize for repeatability and consistency even for the things and infrastructure that is wrapped around our database itself.

JSON Milestone

In the JSON milestone, I had my expectations set that I would learn how to format the data to import it into a document database called MongoDB later on. But, I also learned many more things that were still pertinent to the relational database, or MariaDB itself in our project.

First new thing I learnt here was about default database directory, which in this case was /var/lib/mysql/pos/filename. After finishing the script and running it, my script gave me an insufficient access error. This got me thinking that it had to do with user permissions at the database level or linux system level. But I kept thinking I gave all the access to my users at the time of setup itself. After giving it a deep thought, I came to realize that it was an access issue, but the issue was because I was trying to write my output file using OUTFILE command into the database directory as a non-root user. The issue magically disappeared when I ran the same things from the same location as a root user.

I have seen the application of JSON_OBJECT and JSON_ARRAYAGG functions. To understand why it's needed, I went through a lot of documentation and then came to realize it had to do with the JSON file structure itself. JSON_OBJECT is used to generate key value pairs like it should be in a Json document and JSON_ARRAYAGG felt easier to understand after understanding the JSON_OBJECT function, as it's just wrapping together several Json objects as it's value and a separate key to the whole array itself.

I enjoyed learning about JSON printers and it made my job easier to verify my JSON outflies.

MONGODB MILESTONE

When I was doing my json milestone, I felt like what I did was still impartial mostly because I wanted to visually see and confirm how my documents would look like in the mongodb itself. Installing mongodb was a bit familiar to me this time, as I was used to yum package management tool and command line. In this milestone, I learned Mongo shell is the terminal connecting me with the MongoDB and query against my database.

I have seen denormalization coming handy when querying against my customerorders aggregate which maps the customers and the respective orders placed by them. Though the subquery approach did not occur to me at the beginning, after thinking through step by step,

I understood that subquery is the way to go for querying. I came to appreciate the ability to query through the aggregate partially, unlike key-value stores.

As much as I appreciated the technicalities involved in mongodb milestone, It exposed me to the various scenarios that occur to people in the business. I personally liked the question where we were asked to come up with the queries to handle the products recalled and fraudulent orders, as these are some common issues in the world of business that often make the news too. Overall, I felt like it is complete and whole after finishing my MongoDB milestone and migrating the data from relational database to MongoDB, I learned that there is always new and innovative ways coming up to deal with the same old issues in a novel way.

The Most difficult Section

The Hardest part in the whole course was the research that went behind each milestone, I learnt a lot while I was doing it, but at one point of time I honestly wanted to give up. One of the biggest challenges I faced was the triggers milestone, as i knew only basics and completing complicated milestones in just a week of research was a bit challenging for me. I had to spend a lot more time in understanding the concepts and then putting it into live action. One such milestone was triggers for me. In my previous organisation I had very wage knowledge on triggers. I dint really understand the trigger logic, I remember sitting down for the whole day to figure it out. As trigger operates in response to specific events like insert, update and delete writing logics for these were challenging. The other part of it was the order of execution, Triggers needs to be executed in specific order. I still remember Dr. G talking about how important the sequence of trigger execution is crucial, this was very challenging to figure out initially. As I was handling triggers for the first time, I was learning syntax for it for the first time as well. I initially faced a lot of syntax error at the start, and to correct it was hard. Now the elephant in the room. With already triggers being a new concept for me. The logic to think about while making a trigger was hard, the one-week time frame was not enough for me. The Rounding rules for the sales tax needed careful attention. I remember getting stuck there at some point. Choosing appropriate names for the triggers was also challenging at this point because, I almost got confused at the end. I was one of the people who ended up submitting the milestone at the end moment and I got 77. In summary completing this milestone has

provided me with practical experience in using triggers. Now I cannot forget the SQL Syntax for triggers. A great lesson learnt.

Because I concentrated a lot on how to crack triggers, I forgot to concentrate on small things like alter statements which is where I lost most of my marks. So, lesson learnt here is, never ignore the small changes you make, even if you know how to write them well.

In the triggers milestone I was stuck at this part, mv and tax calculation. How did I overcome that, triggers calling stored procedures inside it.

The Most Surprising thing I have learned

Each milestone thought me a new value. I was very scared to start a milestone, scared of failing of getting a zero on each milestone. I remember the first milestone was challenging for me because more then writing a code setting up the environment is always was challenging for me. The first milestone completing surprised me with my own ability to work on rest of it. One of the milestones that surprised me the most is the Mongo dB Milestone, this milestone was challenging because the question given by dr. G was a bit confusing, we had multiple ways to think around it. The syntax was also new and it was whole new Nosql we were looking at. I was clueless at the start, but then the documentation helped me a lot, one of the coolest parts about writing code in this milestone was in the question who is the best customer? and what is my best product? This almost had the similar answer for me the limit was set to -1 for the ascending order and the limit set to 1 for the other, both of them had the similar code. Just had to change the limits. It was a fun to discover that. I learnt about this by just looking at the documentation and then thinking of the logic. I tried to use different techniques but there was always some error in it. Then realised the logic can be written either way, just a simple change in code can get you the output.

Another surprising part was database normalization. I know normalization sounds very basic and normal; it might not be the as interesting as other topics. So, normalization is nothing but set of rules that help you organize your data in a way that makes sense. It's about putting things in order smartest way possible. I thought database had data, and as long as data is not haywire its all good. But no normalization is showed me a cure to this madness, it can make databases super-efficient. Sometimes I wonder what if normalization actually never existed,

Surprisingly learning about normalization was not boring at all. And it was not like I had to read the whole manual and sit and research on this topic for days, it happened so organically while working on the milestone. I liked the cleaning data and normalising it. I also tried doing normalization on one of my personal projects that I was trying where I had taken a set of data and started breaking down the table into smaller, more manageable pieces. I saw how this reduced redundancy.

With this knowledge I can organize information in databases like a pro, making sure it's neat and efficient. I can break down big chunk of data into smaller, manageable pieces. This will help me in avoiding repeating the same information all over the place, free from unnecessary repetition.

Favourite Section

My favourite section was learning about different types of NoSQL databases. The MongoDB milestone was my favourite, I liked it because, it was challenging we had to do it from scratch from installation to write the code on our own just in a week. I got a bit confused when I read use the mongo dB shell to run queries against a database. But later I figured out it means you use mongosh. The fun part was the installation. I failed to install it first because, I dint really find a proper documentation that would help me do that. Not even a YouTube video which would suggest me how to do it. After figuring out it was a big relief. The prerequisites for this milestone were also interesting but I dint enjoy it much, it needed us to set four Json documents from the Jason milestone, in this milestone I remember mismatching few columns in it, which affect me later on with the MongoDB milestone.

Another good part about this milestone is not a lot was given away, we had to critically think how to solve each and every step, for example in the question who is the best customer? this might have more than two approaches to think around. Which made it challenging but also interesting playing around the data.

The other question that I almost dint get a clarity about was the how do I find all the customers that purchased a product in case of a recall? I took a random product id, and considered that

product as the one I am recalling, now I build the queries to fetch the list of customers who purchased the product. This was fun.

Another favourite section of mine was, Replication, I felt this was the easiest milestone to come up with, we had to configure one server to be a copy of our database. We had to establish two serves as replicas and then one primary server. The process was also very simple, 3 instances should have been created in the Aws, we had to configure replication to work from the primary to secondary. This milestone was a relief after the heavy advance topics, so upon completion we clustered the MariaDB servers. The changes that are made in primary we were able to observed in the replicas. The irony is that the concept is heavy but it turned out to be easy, maybe in the real time world it will be difficult to cluster a large database.

concept 1: the ETL- I liked ETL because it involves extracting data from a source system, transforming it to suit the requirement of a new system. Understand the data you're working with. In real world data you get from various places will be often messy, inaccurate, and incomplete. So, this was a nice milestone to work on to prepare us for the challenges of dealing with the real-world data in our careers, I also learnt about data extraction, the CSV files (comma separated value) files. The prerequisites for this milestone were also interesting. ETL needs a well-prepared environment. Creating the temporary tables was a good idea in this milestone. As it avoids errors and you will have clean data to put into your tables. There were challenges as in data in CSV files had to be cleaned, such as handling quantity in order Line, managing dates, and addressing missing or invalid data. These were interesting to solve. The best part was after performing the ETL your data getting displayed after a successful implementation. Through this project, I will be able to close the knowledge gap between their theoretical education and practical experience as database administrators and data engineers. I will have real-world experience and practical skills by the end that will help me manage data effectively.

concept 2: Transactions and prepared statements, this was the milestone was the introduction the advanced concepts, unlike previous milestone this task demanded a little more effort from me. Working independently, I had to read the whole documentation and get my syntax on point. Creating transaction to add new Order Line and deleting an Order Line was a bit challenging for me. I remember sitting for hours in this section. But the best part to witness was the rollback statement on truncate operation. Prepared statements for adding a new

customer were easy. The SQL injection part was a bit tricky, creating something that could potentially harm your database was crazy, but it taught us the importance of safeguarding against malicious data.

concept 3: Replication: Understanding how databases operate, particularly when it comes to configuring a MariaDB cluster, is like going on an adventure with the Clustering Milestone. If I Consider a cluster as a group of friends cooperating to make things go more smoothly, in this case, the team consists of servers. Setting up one boss server, referred to as the primary, and two helper servers, referred to as secondary servers or replicas was fun. The whole milestone was easy and not much effort was required. We created 3 instances in AWS setting one as primary and other 2 as replica. When we copied data to the primary and everything was ready and set up. It was exciting to see changes happen in the replica as well. The teamwork between the clusters was a good experience to have.

concept 4: NoSQL models, this was one of the best part of the classes, I likes how everyone teamed up in 3 and presented different NoSQL models, I learned a lot about different types of database. The peer evaluation questions also helped me to judge the databases accordingly. We learned about important facts that support any NoSQL system. Determining whether the selected database was open source or not. We considered how well it fit into the paradigms described in the NoSQL book and examined how well it aligned with NoSQL models. Assessing the system's ability to support strong consistency, simple clustering, and transactional functions, The business use case and the QA session at the end gave me lot more insight on each and every NoSQL database.

Why I think this project was useful

I think project helped me a lot in critical thinking, As I was doing my own research. I made a lot of mistakes and gave up many times. But then this project thought me a lot of patience. All the new concepts that I learned in this milestone and on perusal is worth adding to my knowledge base. I felt the flow of the course or the project was set properly, we learnt a new thing each time. And for the next elevation we were building on the things we learnt.

This will help me a lot in building my resume as well, I can now put skills like AWS, MariaDB, ETL in my skills section. The projects made me like data.

I learned how a data is managed in the organisational level, production level, how the real-life data is actually used in the real time projects. Another important observation that I made was world runs on data, so each and every company will have a data team, from the small start to a big organization. So learning this skill will always enhance me to the new opportunities.

The project was a holistic view of the whole data related concepts, from setting up, etl, using advanced concepts to dive deep into it, clustering, and even using a NoSQL database to add to our challenges.

Course Evaluation

What is this class really about

The first thing I learnt in this class was the basics of data, how a real-world data model is actually set up. Managing data is complicated but with the right tools it can be easy. Each milestone taught me a new lesson, I started with a clean slate at first without a lot of knowledge but by the end of the semester, I am so happy to say this was one of the most I have learnt. I learnt some concepts like, normalization, clustering, triggers which were advanced and it was always tough for me to understand them, the best part was perusal tool helped me a lot in understanding concepts in deep.

For example, I was stuck in the ETL milestone, this was the toughest for me, but this milestone teaches how important it is to clean data before using it. The real-world data will never be this clean. This course also focused on data security not only the building part of it.

I learnt a lot about new components like putty, SSH. It was fun to research about them and use them. I enjoyed that setting up the work environment, knowledge about Linux operating system, yum packages can help us set up a good environment, now I can apply this knowledge for the future projects as well. The best part was we discussed a lot about consistency of data in the class, the different consistency levels in data in relational and NoSQL databases.

NoSQL as a part of course not only thought me how important is traditional database and why companies use it a lot. It also thought me a database which complies with ACID properties is better and not being ACID compliant. Consistency is very important in the data world today. With this comes the security as well. The NoSQL database in my point of view is less secure.

Overall, this course was a great combination of all of important content. It taught me a lot about how to select a database for myself. Now I am way confident in performing the queries and writing my own code. Each decision I take in the data is they might affect the whole entire table. Now I got some confidence is doing some small projects based on data.

The best part about participating in class

The best part about participating in class is learning was perusal, I feel learning the content while discussing with your peers was fun. The content that was put of perusal was interesting and had depth. When I got stuck in any of the places in the milestone, I always referred back to the perusal. The discussions in perusal were always interesting. There were few questions asked by my classmate that made me think otherwise. We can also provide important links in the comments section, which was actually helpful. Most of my doubts which were concept related were cleared. The peers also commented insightful comments on my doubts and also suggested me to read a particular section to get more clarity.

The next best part of the participation was the presentations, I felt students explained it very well. I learnt a lot about different types of NoSQL databases. We got to collaborate with good mindsets, each and every database has its own offering. The peer evaluation was a good concept, I like the question in it that said, “how confident are you in discussing this database with the recruiter”, I always took a pause just to answer this. The Q&A section of the presentation was interactive. The gifts parts of it were the best. The class was interactive.

The code discussion after each milestone also created some interesting questions in the class, and every time dr.g answered “it depends” makes more sense now. Most of my milestone doubts were solved in the class milestone participation. When I ran into any complex issues and dint really understand the concepts behind it clearly, the milestone code discussion would always cover it.

Dr.g always took all our questions and answered them with at most detail, by not giving away a lot, but at the same time dropping a hint of how to proceed.

Even when the classes were not conducted on some Wednesdays, we could visit Dr. g for doubts in the class. This was very helpful for all of us who were stuck in a milestone.

Overall, the class was very interactive and it made me learn more and more every class.

How to select the best persistence layer

While choosing a persistence layer a lot of factors will be involved, we should evaluate if the data fits better in relational or non- relational model. We should also consider scalability needs of the application. The read, write speed also counts, data Consistency is also important so we can choose between consistency and high performance. Here are few persistence layer and its types.

File

This type of persistence can be chosen when data is simple and it does not require any complex relationships. For structured data, flat files will be a good option. As Json and XML files are choose for semi-structured data. Let us have a use case for the file system. File system is easy to organize, you can create as many folders as you want to arrange the data. This way you can quickly find it without searching all of the information on the computer.

The system system names can also be simple and easy, this makes it clear to tell what each file is about. Just like how we give names to each thing in our room. The file system can also keep restriction to access control for your file. You can actually decide if only you can view the file or anyone else can also view this. So, the controls will be in your hand. The data can be retrieved faster in the file system. As the system is not that complicated. But file system is not that good if you are finding something quickly, by mistake if you forget where the file is stored then it will cause a problem.

File system can also have some security issues, if you forget to lock the files with proper permissions, anyone can mess with the files. One more issue would be, if we are working with more than on person on the same file, the changes reflection will be hard to reflect.

Consider a logging system, in this system each log entry contains time stamp, user ID, and action. The file can easily store this, without a lot of steps involved.

Managing a to-do list to save and retrieve, is one of the best real- world example. One of the examples of database system that uses a file-based system is SQLite. But most of the companies will opt for other and better databases.

Relational Database

Relational database is very famous. Many people will opt for this, few reasons are the data storage is more structured. This type of layer uses rows and columns to store the data, this kind of format is easier to understand and makes our work easier. The concept of primary key and foreign key will make sure of data integrity. One of the main things is they support SQL; it is a very powerful language to query data. The ACID properties will also be valid here. this will be useful for the financial systems. The relational database can handle more data and it is very good option if the user is going for scaling. But there are some downsides to this type of layer, it is very complex, as the tables increase the complexity and the searching of the tables will also be difficult. Next as the data is growing the maintenance problem will occur, this will require more technology.

This cannot handle any semi-structured data. The cost of it is high, the maintenance and support for this database is high.

One of the best real world use cases is managing information about customers, products, orders etc, like how we did in our mile stone. When there is a lot of data, the structured nature of the relational database will help in maintaining relation between the tables and get us the information correctly. Amazon, known for its e-commerce uses this type of database in managing the big amount of data that comes in every day. Some of the examples of relational databases is oracle dB, PostgreSQL, etc.

Key value database

Key value stores are simple and straightforward. it just contains of keys and the values related to the keys. It is easy to understand, key value is commonly used where quick access to data is necessary some of the key value databases are used as cache (like Redis) because of the fast retrieval ability. The key value stores are very suitable for real time analysis. One of the main advantages of the key value is the high performance for read and write operations.

But in key value, the querying is difficult. And if the key is lost the data that is stored will also be lost. So, you should keep a separate track of the keys. The key values do not support the relationship between data. To connect all the data will be more difficult. In key value database transactions make sure that the series of operations are complete or leave the database

unchanged if any error occurs. Consistency in key-value is minimal. Because as the key value database mainly focuses of the fast retrieval, it will compromise on the consistency.

One of the real-world examples is, let's take online shopping cart, each product will have its unique id. This is the key for each product. Many companies use key value database like twitter, snapshot etc. But most of these companies use them as a cache. Some of the popular key value databases are Amazon DynamoDB, Redis, Cassandra etc.

Document databases

This is one of the most popular type of NoSQL database, this type of data organises in semi-structured format. In this type each document will contain key value pair. The values can be same data types as well. The most common format used is Json document, which we saw in one of out milestones. It does not require a fixed schema. One of the best features about this is, it supports wide range of data types in a single document.

When there is a situation of handling high volume of data, it is a best option for horizontal scaling. High volume of read and write is possible.

But these don't give enough support for transactions. The queries are also complex is what I felt, So because of this they are not always a first choice for selecting a database.

One of the use cases we can take is, imagine a large e-commerce platform which offers products. Because of the flexible schema, the platform can store product information in a Json format. No data structure is required. The growing products can be handled correctly.

Companies like BBC, cisco etc will use this type of database, some of the popular document dB is MongoDB, CouchDB, Cassandra, Firebase etc.

Graph database

Graph database basically contains nodes, edges to store data. The connection is different in graph database. This model is suited when complex relationships or interconnections. One of the main advantages of the graph database is the entities navigation. It is one of the strengths of the database. This makes the developers and others work more easily on the data. Query performance is good in graph database compared to others, it also provides flexible schema,

which makes addition of nodes easy. There are certain types of queries a graph database can perform better than the relational database.

But the consistency is eventual in this case, they follow ACID properties and the transactions are generally supported in this database, they either fully succeed or roll back if any part fails.

One of the real times use cases of the graph database is social networks, because of the graph model, it enables efficient querying in discovery of new people, for any mutual friend's connections, and any recommendations, because activities are represented as nodes.

Some of the companies that uses graph data base is Facebook, LinkedIn, twitter. And the graph database that I recommend is the Amazon Neptune because it is fully managed by AWS, which makes it suitable for range of use cases.

Column-family database

This type of database organises data into columns not rows, the data will be grouped into column. This kind of database is mainly used when it can handle large amounts of data. High volume of write operations. Horizontal scaling is good as the data volume increases, adding more nodes to the cluster. They are suitable for high transactional loads. Columns can be added or modified without affecting the existing data.

But the queries are complex for even simple queries, so understanding and learning it will take a lot of time. They may even lack support for complex transactions. The querying capacity is also very limited. This database is also not set for all use cases. Many companies do not use it because of its complexity,

I prefer HBase, Amazon DynamoDB out of all because these are easy to install, and has the wide range of documentation, Amazon DynamoDB has a good support group, loss of data can be easily recovered.

Elevator pitch

In ADM course, I learnt a about managing real world data, we started with simple milestone of setting up the infrastructure, in this we had to set up the environment for Linux environment for installing the MariaDB. Then with ETL we loaded the data and started using advanced concepts for performing action son it. We used views, stored procedures, triggers.

We also learned advances concepts like clustering and replication. These concepts were the most interesting. The course also thought us about the non-traditional databases, we had a lot of interactive sessions on types of NoSQL databases. Overall, I am confident in telling I know the concepts of advanced database management systems by the end of this course.