**CA2 Report on choice of mySQL for the project and comparison to other DBMS**

**Glossary**

* **1.Transaction processing of mySQL**
* **2.Concurrency control of mySQL**
* **3.Security features of mySQL**
* **4.Scalability of mySQL**
* **5.Comparisons of DBMS**
* **6.Summary of the report**
* **7.References**

**Transaction processing of mySQL**

**1**

MySQL provides support for ACID which stands for Atomicity, Consistency, Isolation and Durability this determines that the database in my project is complete or it has fully stopped via through **rollback;** if it encounters an error or **commit;** if there are none which assists in error handling in this project.

**Start transaction;** starts the transaction that executes requirements that the user wishes to see and ends in **commit;** and in between **start transaction;** and **commit;** is where the **write** function comes in.

**Write** operations in MySQL are methods that **lock tables** which blocks from other operations running while the write operations are undergoing their tasks and other running checks and operations must wait until the lock is lifted through **unlock tables;** in my project to write changes into my tables I implemented the example line of code below in between **start transaction;** and **commit;**

**Example: update table1 set row1 = ‘text’ where id =’123’**

**Read** operations lock **write** operations after **write** is finished and locks other operations from performing through lock tables too and is unlocked through **unlock tables;** which allows me to put in commands to read the displayed changes I made to tables with the example code line below and is used below where I write the changes I wish to make.

**Example: select \* from table1 where id = 1**

**Commit;** signals the end of the transaction and begins to apply the changes the user wishes to make permanently which are then written to the database and are now visible to every other transaction that may be in place and is used where the project prepares to write my changes to the affected tables.

**Rollback;** is used at the end of the program to catch errors that may be present that the user may have not noticed or has missed, if there are errors rollback displays where the error is occurring for example an incorrect column name is present the changes will not be committed and the rollback will stop the transaction entirely.

**Concurrency Control of mySQL**

**2**

In my project concurrency control is implemented through **lock** and **unlock** queries.

Concurrency control is essentially controlling the flow of data within the database, this allows data to go in and out without performance issues especially for large amounts of data.

This is implemented through various means this can be **locking** or **multiversion concurrency control** or **transaction isolation levels** but the most basic ones are **lock** and **unlock** queries which is what were used in my project to control the flow of data.

In my project **Lock** was used in write and read queries which allows for other operations running to stop and lets the writing and reading of changes made to tables made below is an example of how it was used in my project.

**Example:** **lock table1 write;**

Below **lock** were example changes that were gonna be written and read

**lock tables Grades write;**

**update Courses set roomLocation = 'CL045' where lecturerID = 'L919';**

**update Courses set roomLocation = 'Online' where lecturerID = 'L269';**

**update Courses set roomLocation = 'CL010' where lecturerID = 'L406';**

**update Courses set roomLocation = 'Online' where lecturerID = 'L649';**

**lock tables Grades read;**

**select studentID, grades\_pass, passed\_module\_ID from Grades where grades\_pass < 80;**

**select studentID, grades\_fail, failed\_module\_id from Grades where grades\_fail < 40;**

**Concurrency Control of mySQL**

**2**

**(Continuation)**

Unlock is used to unlock tables that are targeted by the lock so that it can be used again, in this case it was used to unlock tables to test if my changes were displayed as I wrote them which was done through the code below:

**lock tables Feedback read;**

**select studentID, feedback from Feedback where studentID = 'S0001';**

**select studentID, feedback from Feedback where studentID = 'S0002';**

**select studentID, feedback from Feedback where studentID = 'S0003';**

**unlock tables;**

Locked tables cannot be accessed so therefore if there is no unlock and you try and update the data it will display a relevant error stating that the **Table 'table1’ was locked with a READ lock and can't be updated** if it was accessed through read lock and **Table 'table1’ was locked with a WRITE lock and can't be updated** if it was through write lock.

And to unlock the tables you have to run the query script unlock tables to which changes can be made freely and without errors.

**Security Features of mySQL**

**3**

The security features in MySQL can range down from implementing **primary keys** and implementing **foreign keys** which gives each table a unique identifier.

Implementing a **transaction management** which makes sure that each group of queries is identified as a single unit this maintains integrity of data and consistency.

Implementing **table locking** which can ensure table integrity but if it is not used appropriately and correctly it can increase risk of a bottleneck within the database.

Introducing **Data access control** through read or write locks which controls access to sensitive data in between updates and reads queries which stops conflict of operations operating concurrently

These are the four security features that were implemented in my project.

Of course, there are other security features that should be taken into consideration when implementing security features into MySQL such as;

**Authentication and authorization** of users which means who gets to access your data and when.

**Data encryption** which protects data in transit and data that is yet to be in transit.

**SQL injection prevention** would be a very important security feature in terms of MySQL as it prevents malicious code from being injected into your queries.

**Logging and auditing** activity within your database to prevent tampering and security issues within your database.

**Network security** for securing database via through firewall configurations, network segmentations and secure connections like SSL and TLS.

**Security Features of mySQL**

**3**

**(Continuation)**

For security in my project, I implemented the **primary keys** and **foreign keys** to give the values unique identities which allowed me to grab and change different parts of the table(s) without violating relationships between the tables.

**create table Courses(**

**programID varchar(255),**

**programName varchar(500),**

**programPopulation int(255),**

**moduleID varchar(500) not null primary key,**

**moduleName varchar(500),**

**modulePopulation int(255),**

**roomLocation varchar(500),**

**lecturerID varchar(500)**

**);**

**create table Students(**

**studentName varchar(100),**

**studentID varchar(100) not null primary key**

**);**

**Security Features of mySQL**

**3**

**(Continuation)**

I implemented **transaction management** in my project using queries like start transaction, commit and rollback which allowed me to make changes to specific parts of the table(s) which makes sure that I’m not working with large amounts of data to deal with as it treats it as a one small group rather than a large one.

**start transaction;**

**lock tables Grades write;**

**update Grades set grades\_pass = 99 where studentID = 'S0001';**

**update Grades set grades\_pass = 90 where studentID = 'S0002';**

**update Grades set grades\_pass = 95 where studentID = 'S0003';**

**unlock tables;**

**commit;**

**lock tables Grades read;**

**select \* from Grades where studentID = 'S0001';**

**select \* from Grades where studentID = 'S0002';**

**select \* from Grades where studentID = 'S0003';**

**unlock tables;**

**rollback;**

I implemented **table locking** so as to make sure that there is controlled access when trying to interact with the data and committing them permanently to the database and also to make any changes I see fit to demonstrate to check if it has changed anything in the database tables which is shown above.

**Security Features of mySQL**

**3**

**(Continuation)**

And finally, **data access** was demonstrated in my project by using read and write in the MySQL code to ensure there are no conflicts within the data encouraging concurrency control inside the database which is shown above in **transaction management**.

**Scalability of mySQL**

**4**

In Scalability MySQL can provide sharding, clustering and replication to lessen workload and large bundles of data that users work with, MySQL can be scaled vertically in terms of scalability meaning you add more resources to a single server rather than multiple but horizontally working with data you add more servers to the workload.

In my project I dealt with small sets of data so it would be very less noticeable in terms of scalability but if I dealt with large sets of data It would be more prominent and would most likely result in poor performance as MySQL is designed for short term data rather than long term data, in the project 100 observations were conducted per table totaling up to 500 observations across 5 tables so as to compensate for small data.

**Scalability of mySQL**

**4**

**(Continuation)**

The data that was being extracted was embedded in CSV files which contained details relevant to the tables or how I envisioned the tables would look like, this ranged from course details which contained what lecture is currently available, who runs the lecture, what their role is and what is the population of students attending this module and students details that contained their student number, first name and second name.

**create table Courses(**

**CourseID int(30) not null primary key,**

**CourseName varchar(100),**

**CourseProgramme varchar(100),**

**CourseNumOfStudentsEnrolled int(100),**

**CourseLecturerFirstName varchar(100),**

**CourseLecturerSecondName varchar(100),**

**CourseLecturerRole varchar(100),**

**CourseRoomNum int(100)**

**);**

**create table Students(**

**StudentFirstName varchar(100),**

**StudentSecondName varchar(100),**

**StudentID int(30) not null primary key**

**);**

**Comparisons of DBMS**

**5**

In comparison from mySQL and to other DBMS such as PostgreSQL, Oracle and MongoDB they differ quite a lot from each other.

We will start with **mySQL** and what it can offer first:

- MySQL offers support for transactions which adhere to the ACID properties, this comes with features like **start transaction, commit** and **rollback**. This gives assurance that your data integrity is safe and secure.

- In Concurrency Control MySQL uses locking and unlocking functions and transaction isolation levels to handle concurrency in the database, the isolation levels that it has are **read committed** and **repeatable read** which gives users and developers options to balance between consistency and concurrency.

- In terms of security MySQL features basic security like **user authentication and authorization**, **encrypted connections (SSL/TLS)** and **access control lists** for enforcing security in connections. MySQL also has data encryption for data stored in the database.

- MySQL in regard to Scalability has horizontal scalability techniques which are **replication**, **sharding** and **clustering**. MySQL can take medium sized to large volumes of data efficiently.

Next is **PostgreSQL**

- PostgreSQL has similarities to MySQL when introducing **ACID** transactions and assures data integrity and consistency same way as MySQL

- In concurrency control PostgreSQL uses **Multiversion Concurrency Control** as a concurrency control, this offers better reading consistency within the database and is perfect for high concurrency requirements situations.

- PostgreSQL in security gives features that have **role-based access control**, **transparent data encryption** and **row-level security** to ensure that any sensitive data that is contained within the database is secure and is used on projects with more stringent security.

**Comparisons of DBMS**

**5**

**(Continuation)**

* In terms of Scalability PostgreSQL has options like **table partitioning**, **sharding**, **parallel query execution** and **replication** to help hold large amounts of data and divert a large amount of traffic across database.

**Oracle** differs from MySQL in some respects as such:

* It does not differ much from **ACID** compliance, but it is able to handle more complex transactions in a database with its transaction management.
* In concurrency Oracle offers **locking**, **more advanced concurrency control queries** and **Multi-version Concurrency Control** to operate concurrent transactions more effectively.
* Oracle has security features like **FGAC or fine-grained access control**, **comprehensive security**, **auditing** and **encryption** which is widely used for more enterprise graded applications.
* Oracle has scalable options like **RAC** or **real application clusters**, **sharding**, **table partitioning** and **replication** which makes oracle lighter on performance and more accessible and operational to users.

Finally on how DBMS differ from MySQL is **MongoDB**

* In MongoDB it offers a **multi-document ACID transaction** and a **traditional document-based transaction** in its current stage, this safeguard data integrity for transactions that have more than one step required to permanently commit any data to the database.
* MongoDB uses more optimistic level of **Concurrency Control** when it comes down to writing data to the database this enables a multiple number of users to modify data at the same time while effectively resolving any conflicts that may occur during the process.

**Comparisons of DBMS**

**5**

**(Continuation)**

* MongoDB has security features like **role-based access control**, **data encryption** in **transit** and at **rest** and **auditing**. This ensures the security of the data that is in the database and also encrypts data that is in transit or is yet to be in transit.
* MongoDB supports Scalability horizontally via **distribution of data** which flows through multiple servers and scales with the size requirements as size of data starts to expand and also has **replication** of data sets and **sharding**.

**Summary**

**6**

To summarize MySQL differs from the rest of DBMS in many aspects and also has the same functionality as some of the DBMS, MySQL is a good choice due to its scalability, security, transaction processing and concurrency control but it does have its downsides where some DBMS have covered, MySQL is commonly used for short-term data so that means it is able to handle medium to large quantity of data as long as it does not overwhelm the program itself.

PostgreSQL and Oracle are always used for long-term data as they can hold and compensate for large quantities of data without performance issues or conflict, MongoDB is also used for large data storage and is flexible in regard to document format.

Though they differ in some respect, the DBMS do have common traits for example they share same functionality in regard to Scalability they have sharding, replication and clustering of data and data sets and both support ACID properties in terms of transaction control.

**References**

**7**

Hallie, L. (n.d.). *Comparing MySQL, PostgreSQL, and MongoDB*. [online] vercel.com. Available at: <https://vercel.com/guides/mysql-vs-postgresql-vs-mongodb>.

‌ KITRUM. (n.d.). *Difference Between MongoDB, MySQL, and PostgreSQL*. [online] Available at: <https://kitrum.com/blog/difference-between-mongodb-mysql-and-postgresql/> [Accessed 24 Mar. 2024].

‌ Gillis, A. (n.d.). *What is MongoDB? A definition from WhatIs.com*. [online] SearchDataManagement. Available at: <https://www.techtarget.com/searchdatamanagement/definition/MongoDB>.

‌ Oracle.com. (2023). *What is MySQL?* [online] Available at: <https://www.oracle.com/ie/mysql/what-is-mysql/>.

‌www.solarwinds.com. (n.d.). *What Is an Oracle Database? - IT Glossary | SolarWinds*. [online] Available at: <https://www.solarwinds.com/resources/it-glossary/oracle-database>.

‌ SQL Easy Tutorial. (2023). *How to Scale PostgreSQL: A Comprehensive Guide for Rapid Growth - SQL Knowledge Center*. [online] Available at: <https://www.sql-easy.com/learn/how-to-scale-postgresql/#:~:text=Understanding%20PostgreSQL%20and%20Its%20Scalability&text=Well%2C%20scalability%2C%20in%20simple%20terms> [Accessed 24 Mar. 2024].

‌ PostgreSQL Tutorial. (n.d.). *What is PostgreSQL*. [online] Available at: <https://www.postgresqltutorial.com/postgresql-getting-started/what-is-postgresql/>.

‌

planetscale.com. (2023). *What does ACID compliance mean? — PlanetScale*. [online] Available at: <https://planetscale.com/learn/articles/what-does-acid-compliance-mean>.

**References**

**7**

**(Continuation)**

www.solarwinds.com. (n.d.). *What is Database Concurrency? - IT Glossary | SolarWinds*. [online] Available at: <https://www.solarwinds.com/resources/it-glossary/database-concurrency>.

www.continuent.com. (2021). *Continuent Blog: What is MySQL Data Replication versus Clustering?* [online] Available at: <https://www.continuent.com/resources/blog/mysql-replication-vs-mysql-clustering#:~:text=But%20there%20is%20a%20big> [Accessed 24 Mar. 2024].

‌

Lord, N. (2023). *Data Protection: Data In transit vs. Data At Rest*. [online] Digital Guardian. Available at: <https://www.digitalguardian.com/blog/data-protection-data-in-transit-vs-data-at-rest>.

‌

Liquid Web. (2023). *What is a Database Audit?* [online] Available at: <https://www.liquidweb.com/kb/database-audit/#:~:text=A%20database%20audit%20requires%20analysis> [Accessed 24 Mar. 2024].

‌

planetscale.com. (n.d.). *undefined — PlanetScale*. [online] Available at: <https://planetscale.com/learn/articles/sharding-vs-partitioning-whats-the-difference>.

‌

Zhang, E. (2023). *What Is Role-Based Access Control (RBAC)? Examples, Benefits, and More*. [online] Digital Guardian. Available at: https://www.digitalguardian.com/blog/what-role-based-access-control-rbac-examples-benefits-and-more.

**References**

**7**

**(Continuation)**

Devane, H. (2021). *What is Fine-Grained Access Control? (And Why It’s So Important)*. [online] Immuta. Available at: <https://www.immuta.com/blog/what-is-fine-grained-access-control-and-why-its-so-important/#:~:text=Fine%2Dgrained%20access%20control%20is> [Accessed 24 Mar. 2024].

‌

www.tutorialspoint.com. (n.d.). *Multiversion Concurrency Control Techniques*. [online] Available at: <https://www.tutorialspoint.com/multiversion-concurrency-control-techniques>.

‌

TECHCOMMUNITY.MICROSOFT.COM. (n.d.). *Introduction to Parallel Query Execution*. [online] Available at: <https://techcommunity.microsoft.com/t5/sql-server-blog/introduction-to-parallel-query-execution/ba-p/383166>.

‌

rwestMSFT (2023). *Transparent data encryption (TDE) - SQL Server*. [online] learn.microsoft.com. Available at: <https://learn.microsoft.com/en-us/sql/relational-databases/security/encryption/transparent-data-encryption?view=sql-server-ver16>.

‌

www.linkedin.com. (n.d.). *What is Oracle Real Application Cluster (RAC)*. [online] Available at: <https://www.linkedin.com/pulse/what-oracle-real-application-cluster-rac-ipspecialistofficial/>.

‌

Google Cloud. (n.d.). *Introduction to BigQuery row-level security*. [online] Available at: <https://cloud.google.com/bigquery/docs/row-level-security-intro#:~:text=and%20other%20details.-> [Accessed 24 Mar. 2024].

‌

**References**

**7**

**(Continuation)**

WilliamDAssafMSFT (n.d.). *SET TRANSACTION ISOLATION LEVEL (Transact-SQL) - SQL Server*. [online] learn.microsoft.com. Available at: <https://learn.microsoft.com/en-us/sql/t-sql/statements/set-transaction-isolation-level-transact-sql?view=sql-server-ver16>.

‌