W11 Paper: Case Study Working with Transactions.

You have had a profitable tenth week at your new company. This week you mastered how to write transactions across two or more tables.

Your boss decided to see if you now have the skills to learn how to write logic that supports web application development. Your manager is keen to have you explain transaction management and scope because some other web developers are struggling with the concept. You should explain:

* How multiuser databases perform read and write locks?
* How multiuser databases provide lock granularity, like table, page, and row locks?
* How and why transactions exist when you want to insert, update, or delete data from a series of tables?

You should return and report with a 3–5 paragraph report that clearly explains what you learned while mastering transaction management in MySQL. This paper should qualify what you learned by experimenting with the technology.

Report:

During this week, I learned how multi-user databases manage read and write locks to control concurrent access to data. In MySQL, when a user wants to modify data, the system grants a write lock that prevents other users from performing reads or writes to that data until the lock is released. Read locks, on the other hand, allow multiple users to read data simultaneously without interfering with each other, maintaining data integrity and ensuring that transactions are processed correctly without conflicts.

In addition, I discovered that lock granularity is a crucial strategy that databases employ to improve performance and concurrency. Databases can implement locks at the table, page, or row level. Table locks are the simplest but can reduce concurrency, while row locks, while requiring more management, allow higher concurrency by locking only specific rows that are being modified. Page locks, which affect blocks of data, represent a balance between table and row locks, allowing multiple users to work with the same table efficiently and securely.

Additionally, it is important to understand transactions in database management, especially when manipulating data across multiple tables. Transactions ensure that a set of SQL operations executes completely or not at all, guaranteeing atomicity. If an operation in the set fails, the system will roll back all previous operations in the transaction, thus maintaining data integrity. This concept is critical when updating, inserting or deleting data in multiple tables, where data consistency between tables must be meticulously maintained to avoid discrepancies that could negatively affect the application logic and user experience.