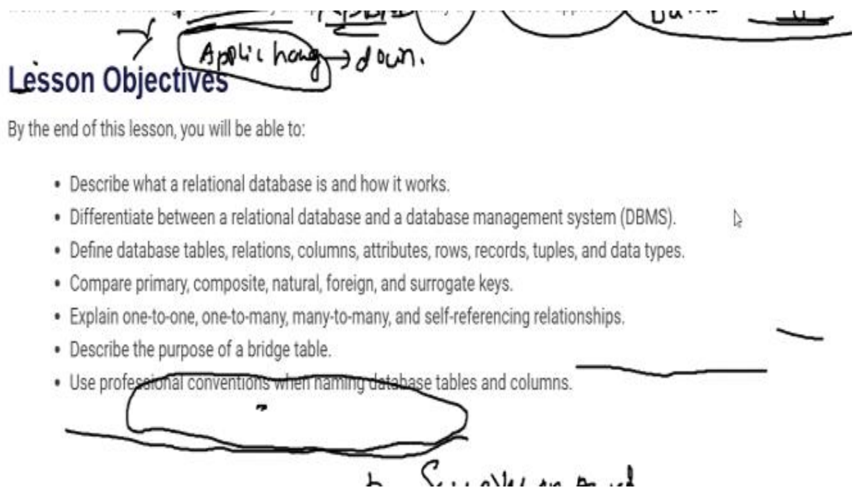
Day 1 SRE basics RDMS SQL

10 February 2025 11:27

By end of the lesson aka RDMS:

1) We can tell what a relational db is and its working
2) Differentiating between RDBMS and DBMS
3) Define database tables, relations, columns, attritbutes, rows, rec etc..
4) Compare all the typa keys
5) Tell relations entity relations like 1-1, 1-M, M-1, M-M
6) Describe purpose of **BRIDGE-TABLE**
7) Use proper conventions while naming db tables and colums



Applic hang → down.

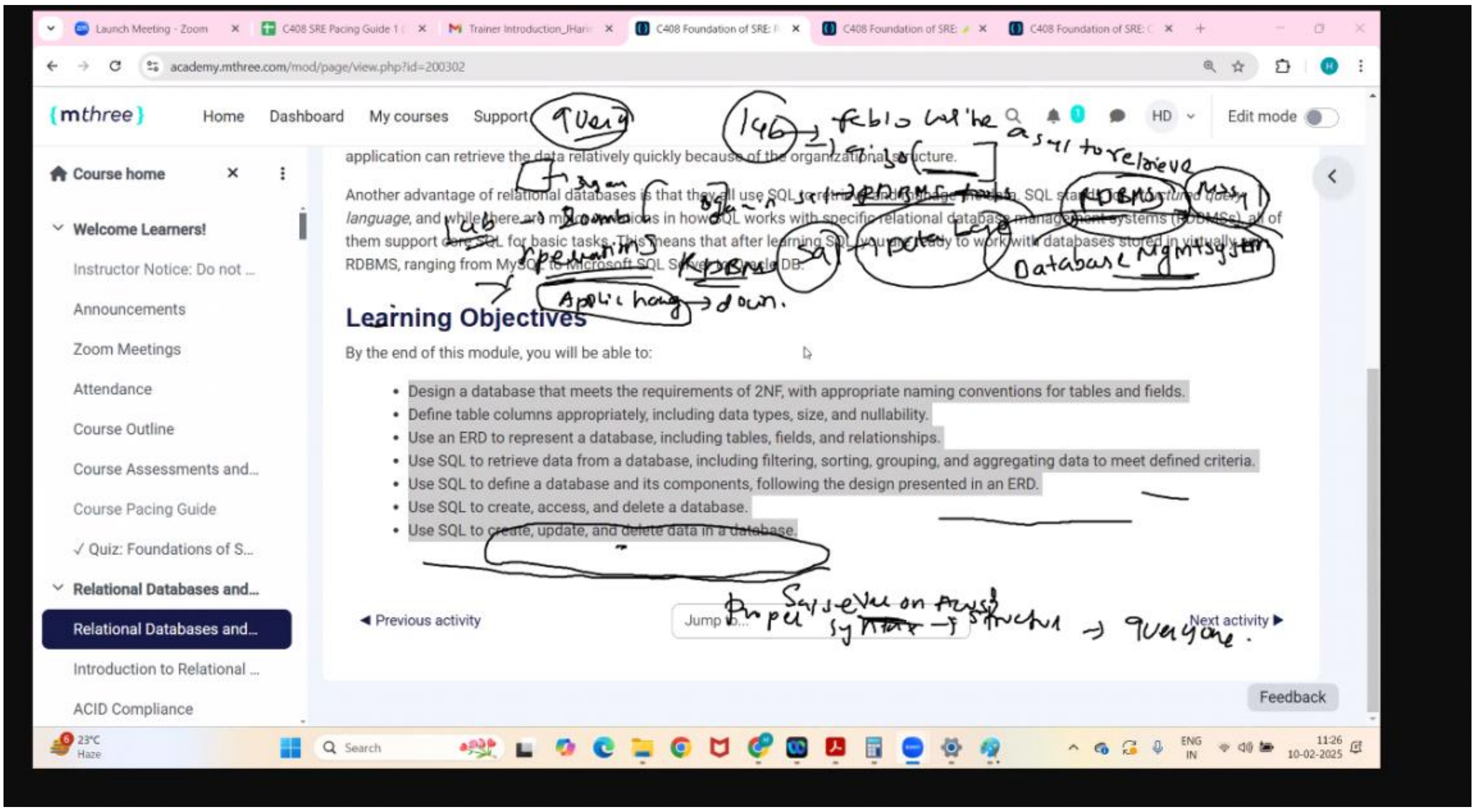
Lesson Objectives

By the end of this lesson, you will be able to:

- Describe what a relational database is and how it works.
- Differentiate between a relational database and a database management system (DBMS).
- Define database tables, relations, columns, attributes, rows, records, tuples, and data types.
- Compare primary, composite, natural, foreign, and surrogate keys.
- Explain one-to-one, one-to-many, many-to-many, and self-referencing relationships.
- Describe the purpose of a bridge table.
- Use professional conventions when naming database tables and columns.

What SQL outcomes!!

1) design a DB that meets requirements of 2NF and all
2) Define table columns and data types and sizes with all keys
3) Use an ERD to represent visually
4) Use SQL to perform CRUD operations



Query

146 → tables we'll have a sql to relieve

application can retrieve the data relatively quickly because of the organizational structure.

Another advantage of relational databases is that they all use SQL to retrieve and manage the data. SQL stands for *structured query language*, and while there are minor variations in how SQL works with specific relational database management systems (RDBMSs), all of them support core SQL for basic tasks. This means that after learning SQL, you are ready to work with databases stored in virtually any RDBMS, ranging from MySQL to Microsoft SQL Server to Oracle DB.

RDBMS MSS

Database Mgmt system

Lab Room

Applic hang → down.

Learning Objectives

By the end of this module, you will be able to:

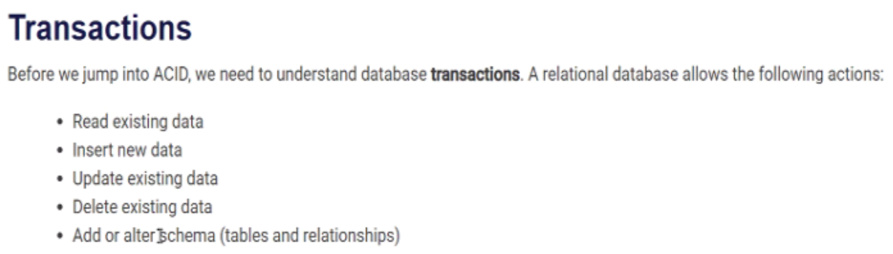
- Design a database that meets the requirements of 2NF, with appropriate naming conventions for tables and fields.
- Define table columns appropriately, including data types, size, and nullability.
- Use an ERD to represent a database, including tables, fields, and relationships.
- Use SQL to retrieve data from a database, including filtering, sorting, grouping, and aggregating data to meet defined criteria.
- Use SQL to define a database and its components, following the design presented in an ERD.
- Use SQL to create, access, and delete a database.
- Use SQL to create, update, and delete data in a database.

Proper Syntax → structure → query done.

ACID COMPLIANCE:

1) Define db transaction
2) Identify **ACID** properties **Atomicity, Consistency, Isolation and Durability**
3) Describe purpose of a transaction log
4) Discuss db backup strategies

TRANSACTIONS



Transactions

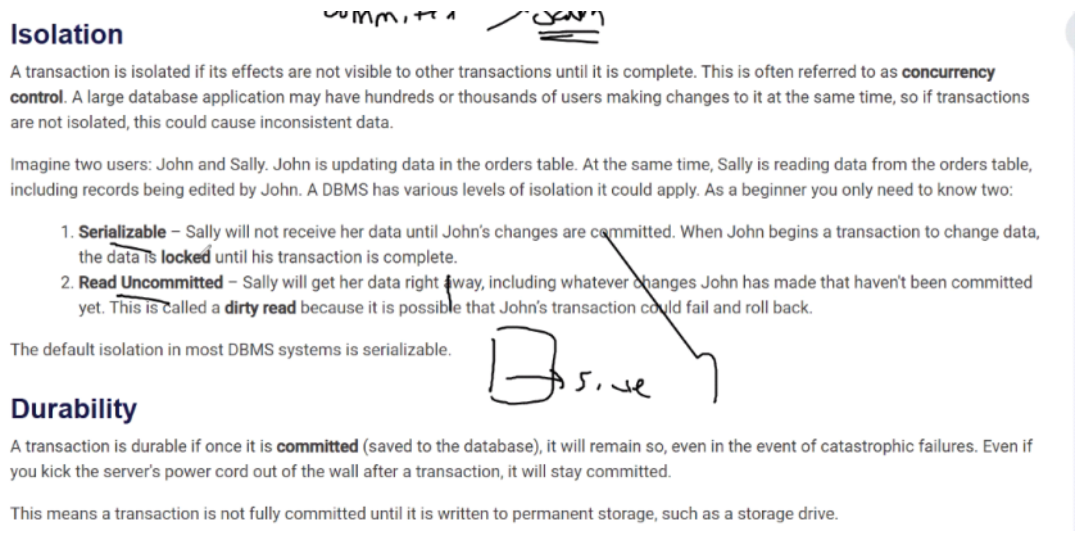
Before we jump into ACID, we need to understand database **transactions**. A relational database allows the following actions:

- Read existing data
- Insert new data
- Update existing data
- Delete existing data
- Add or alter schema (tables and relationships)

TRANSACTIONS is a set of one or more actions that represents a single logical unit of work. It means like an operation like every SQL query we write comes under a transaction.

ATOMICITY (all or nothing rule) no partial transactions ..complete unique and separate
CONSISTENCY a transaction is consistent if it can only move the DB from one valid state to another one, a consistent DB enforces constraints on the types of datatype and sizes of the data that can be allowed.

It also enforces primary and foreign key concepts,
A properly configured relational schema will prevent deleting the primary one if its linked in other tables. CASCADING delete is where it removes data from all related tables unlike normal deletion not possible here where it prevents from deletion if its available in other tables. Generally orphan row is not allowed by most DBMS here for foreign keys..



Isolation

A transaction is isolated if its effects are not visible to other transactions until it is complete. This is often referred to as **concurrency control**. A large database application may have hundreds or thousands of users making changes to it at the same time, so if transactions are not isolated, this could cause inconsistent data.

Imagine two users: John and Sally. John is updating data in the orders table. At the same time, Sally is reading data from the orders table, including records being edited by John. A DBMS has various levels of isolation it could apply. As a beginner you only need to know two:

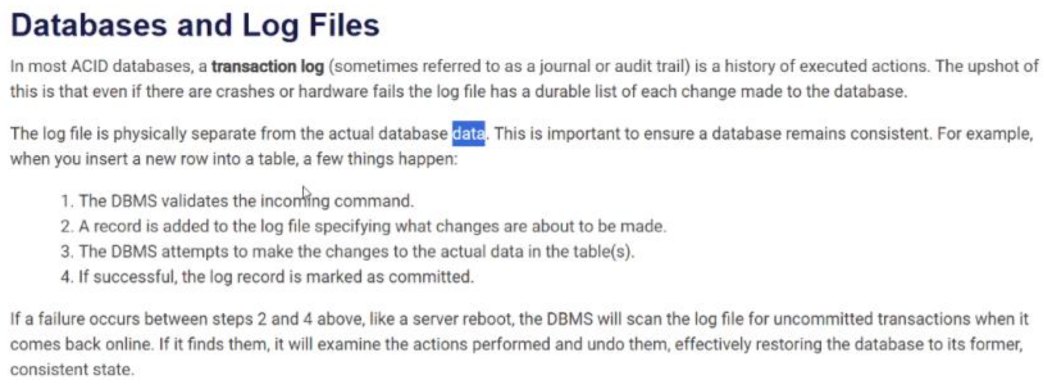
1. **Serializable** – Sally will not receive her data until John's changes are committed. When John begins a transaction to change data, the data is **locked** until his transaction is complete.
2. **Read Uncommitted** – Sally will get her data right away, including whatever changes John has made that haven't been committed yet. This is called a **dirty read** because it is possible that John's transaction could fail and roll back.

The default isolation in most DBMS systems is serializable.

Durability

A transaction is durable if once it is **committed** (saved to the database), it will remain so, even in the event of catastrophic failures. Even if you kick the server's power cord out of the wall after a transaction, it will stay committed.

This means a transaction is not fully committed until it is written to permanent storage, such as a storage drive.



Databases and Log Files

In most ACID databases, a **transaction log** (sometimes referred to as a journal or audit trail) is a history of executed actions. The upshot of this is that even if there are crashes or hardware fails the log file has a durable list of each change made to the database.

The log file is physically separate from the actual database data. This is important to ensure a database remains consistent. For example, when you insert a new row into a table, a few things happen:

1. The DBMS validates the incoming command.
2. A record is added to the log file specifying what changes are about to be made.
3. The DBMS attempts to make the changes to the actual data in the table(s).
4. If successful, the log record is marked as committed.

If a failure occurs between steps 2 and 4 above, like a server reboot, the DBMS will scan the log file for uncommitted transactions when it comes back online. If it finds them, it will examine the actions performed and undo them, effectively restoring the database to its former, consistent state.

Backup Strategies

A lot of time and effort is put into the backup and recovery of database systems. In some businesses, losing access to the database can cost thousands of dollars per minute as orders can no longer be taken or customer data could be lost or compromised.

For this reason, it is important that the database administrator has backup and recovery options for both data and log files. Because logs contain all transaction information, they provide point-in-time restoration information. Full data backups tend to be very large and are only done periodically. Log backups tend to be much smaller.

As an example, we might perform a nightly data backup and a log backup every 10 minutes. If our data backup occurs at midnight and the server fails at 2:55 PM, we would restore the last data backup, then restore all the logged transactions until 2:50 PM. We would only lose changes between 2:50 PM and 2:55 PM. (Not great, but better than the alternative.)

To further reduce losses, we could use multiple database servers and execute transactions on each. If one server fails, another can takes its place. This is called a database **cluster**. As you approach a true lossless solution, the cost of servers and software increases exponentially. An experienced database administrator has the job of matching budget to loss tolerance for a business.

Summary

When reliable data is essential, using an ACID compliant database is a requirement. Only databases that are atomic, consistent, isolated, and durable are going to handle all the errors and failures that can occur while protecting the quality of the data.

No business should be without a backup and recovery plan that suits their budget and risk tolerance.