Data Model:
Relational data model
SQL --> MySQL/Postgres
4 basic queries: Select, Insert, Update, Delete
One is SQL and all others that are not SQL are NoSQL

SQL vs NoSQL
SQL: MySQL, Postgres
In Relational database, what is a relation? Say we have data stored in multiple tables and we have relation between tables. Mapping between entities is known as Relation. Relation = Mapping between tables. Table is a way of presenting our entity into structured format. Table = representation of entity.
Row = one record
Column = one of the fields/attributes of the record
Constraints: There are different constraints in our database. Rules or conditions we apply to our database in order to achieve acceptable data quality.
Add / Update --> After validating the data we update
Different constraints:

- Unique (we can put the constraints only on Column level like username should meet this condition etc not on Row level since Row level is same as putting constraints on the entire data, which doesn't make sense). Say of the user wants to update Username, before it actually updates, it needs to check in the database whether the same username already exists or not
- Not null: There are some values like email etc, we cant have null values. So that's like required field
- Primary key: Primary is a combination of unique and not-null. Email could be a Primary key. StudentID
- Check: value check like Passwords should have 6-12 characters, Age must be a positive integer only
- Foreign key: A key used to establish relations between different entities or tables. Normally Foreign key is the Primary key of some other table
- Default: Whenever we cannot have a null value, that time we could have a default value for that field

Join: 1:M join, M:1, M:M

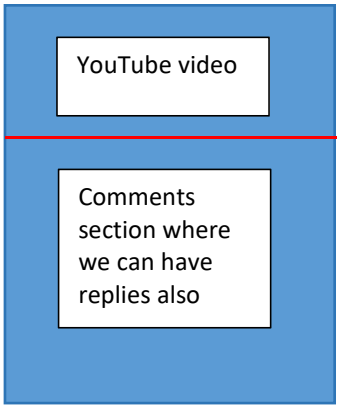
ACID: Atomicity, Consistency, Isolation, Durability
Atomicity (sending money is a transaction) Transaction is a high-level functionality through which we are completing a unit task. There can be multiple steps within a transaction also.
A is sending 1000$ to B
Transaction has multiple steps

- Check Balance of A
- Deduct 1000$ amount from A
- Add $1000 to B
- Commit

Atomicity is All or Nothing. Even if one operation cannot be completed, all the previous operations in the sequence are rolled back
Consistency
Before and After Transaction, Balance should match.
Isolation
Every transaction in the system must be able to work in parallel and must not overlap with other transactions. All requests must be handled in isolation
Durability
If we are sending $1000 to B, the resulting account balance must be permanent. Say 5k and it is permanent even if the system crashes. Only after permanently saving the data in the database, we call it done

Designing Database or Designing a feature of a Database
Design 'comments section of YouTube'

Video/Post: Content table



Content table

ID	Title	Video.ID
1	System design	1

Video

ID	Address
1	Video_address

Post

ID	Address
1	

Comments

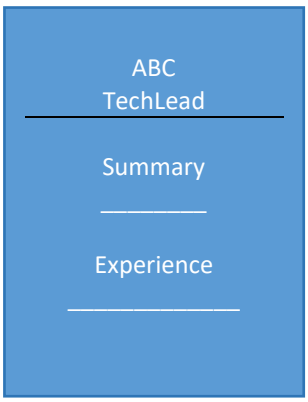
ID	Comment_text	Parent_Comment_ID	User.ID
1	I like this video	0	1
2	Correct, it is a good video	1	2
3	This is my video	0	3

User

ID	Name
1	ABC
2	XYZ
3	PQR

NoSQL: key-value pair database, column, graph, document (MongoDB, Cassandra) database

Homework:
Resume builder



Database design for Resume builder