Episode-12 Databases - SQL and NoSQL

Database → database is an organized collection of data or a type of data store based on the use of database management system (DBMS)

DBMS → The software that interacts with the end users, applications and the database itself to capture and analyze the data. DBMS has the core functionality to administer the database.

Database System → The sum total of the database, the DBMS and the associated applications can be referred to as Database system.

> **Note:** sometimes "Database" is also used loosely to refer to any of the DBMS, the database system or an application associated with database.

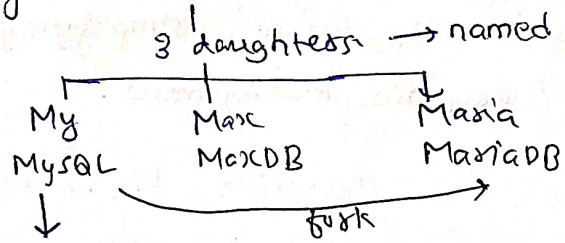
- **Types of Database** →

1. Relational DB – MySQL, PostgreSQL
2. NoSQL DB – MongoDB
3. In Memory DB – Redis
4. Distributed SQL DB – Cockroach DB
5. Time series DB – Influx DB
6. OO DB – db40
7. Graph DB – Neo4j
8. Hierarchical DB – IBM IMS
9. Cloud DB – Amazon DRS
10. Network DB – IDMS

- **RDBMS [MySQL, PostgreSQL]** SQL → Structured Query Language.

Relational Database Management System – developed by EFCODD
he defined rules for RDMS called as CODD's 12 Rules [0 – 12]

MySQL → History: → developed by Michael Widenius



PostgreSQL → History: Michael Stonebreaker
he was working on project ingres in (University of California)
Later called it Post Ingres and Hence the name- PostgreSQL

NoSQL [MongoDB] (Not only SQL)

↳ Types → ① Document DB ② Key value DB ③ GraphDB ④ Wide Column DB ⑤ MultiModel

MongoDB → History → made in 2009 (same year as Nodejs)
→ developed by 10gen company, later renamed itself MongoDB Inc.

↳ flexible, scalable both horizontally and Vertically
↳ very much compatible with Nodejs, increases developer's productivity.

RDBMS vs NoSQL (Document)

RDBMS	NoSQL
1) Stores data in tabular formats with rows and columns. i.e. structured	1) Stores data in various formats like documents, key-value pairs or graphs. i.e. Unstructured
2) Vertical scalability (increasing hardware resources)	2) Horizontal scalability (Adding more servers)
3) uses structured query language (SQL) for complex queries	3) Query Language vary (like MongoDB (MQL))
4) Keeps data clean And Organized	4) Sacrifices some Accuracy for flexibility and speed
5) Slower for large datasets, especially with complex relationships	5) Generally faster for read/write operations with large volumes of data.
6) Ensures strict data accuracy (ACID) rules.	6) More relaxed rules, focus on availability and speed.
7) Need for Joins	7) No need for Joins
8) Need for Data-Normalization	8) No need for Data-Normalization
9) Example → MySQL, PostgreSQL	9) Example → MongoDB, Cassandra, Redis

10) <table><tr><th>Name</th><th>Gender</th><th>Age</th></tr><tr><td>Rahul</td><td>Male</td><td>22</td></tr><tr><td>Hermione</td><td>Female</td><td>18</td></tr><tr><td>Ron</td><td>Male</td><td>19</td></tr></table>	10) <pre>{ "_id": 1
	"name": "Rahul"
	"Gender": "Age"
	"Age": "22"
	}</pre> document

11) Fixed Schema	11) No fixed Schema
12) Relationships - Foreign keys + joins	12) Nested (Relationships)
13) Read Heavy Apps, transaction workloads	13) Real Time, Big Data, distributed Computing
14) Ex → Banking Apps	Ex → Real Time Analytics, social Media.