Base Your Database

Session 1

Introduction to Databases & NoSQL with MongoDB

By : Abdalrahman Khaled

Table of contents

01

Introduction

02

What Is a Database?

03

Types of Databases

04

Introduction to NoSQL

05

Why MongoDB?

06

MongoDB Core Concepts

Table of contents

07

Schema:
SQL vs MongoDB

08

Basic CRUD Operations

09

MongoDB in Real Applications

10

Common Mistakes & When NOT to Use MongoDB

11

What's Next?

12

Closing

Final Outcome of This Session

By the end of this session, attendees will:

- ★ Understand what databases are
- ★ Know why NoSQL exists
- ★ Understand MongoDB concepts
- ★ Be ready to mentally switch to SQL next

01 Introduction

About the Event

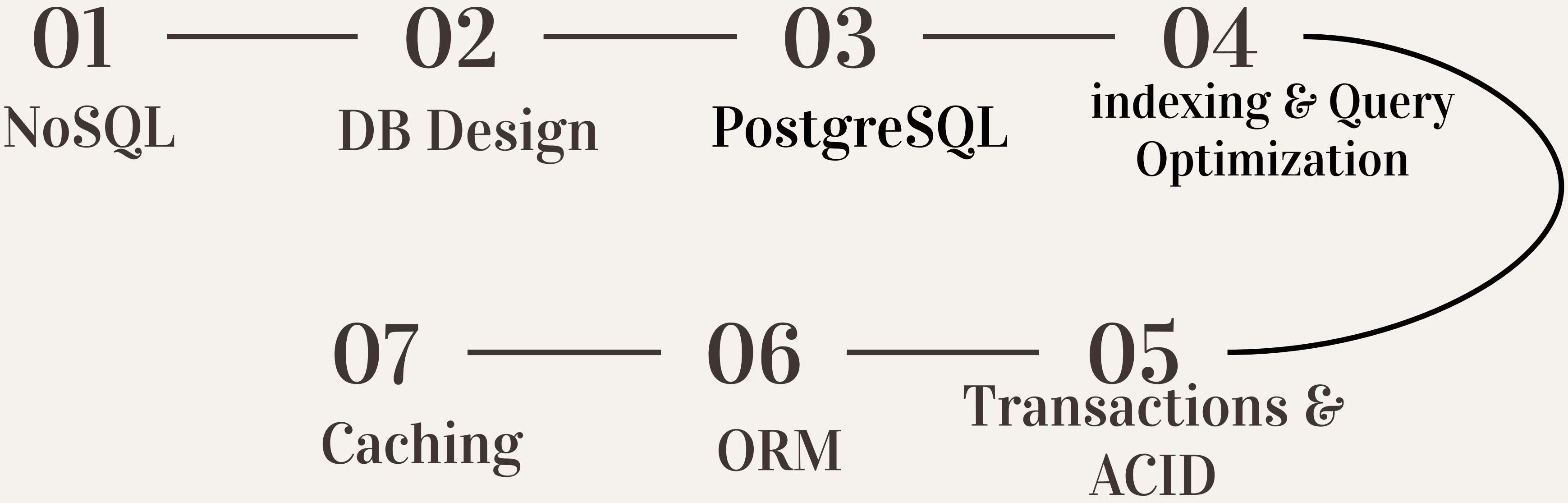
[redacted]

What this database event is about

- ❖ Understanding how data is stored
- ❖ Exploring different database models
- ❖ Learning when to use each technology
- ❖ Building a strong foundation

This is not about tools only , it’s about mindset.

The Event Roadmap



01 NoSQL

02 DB Design

03 PostgreSQL

04 indexing & Query Optimization

05 Transactions & ACID

06 ORM

07 Caching

Rules

- Not attending a session without good excuse **BEFORE** the session -> ***warning***

- Missing a Task without good excuse -> ***warning***

- Arriving late after 7 mins -> pay the fee
  - Fee -> 1 min = 1 pound (max 15 pound)

Rules

- **Best of the week** will be announced at the end of each week on the community and discord

- **Best of the camp** will be awarded at the end of the camp!

Table of contents

~~01~~

~~Introduction~~

02

What Is a Database?

03

Types of Databases

04

Introduction to NoSQL

05

Why MongoDB?

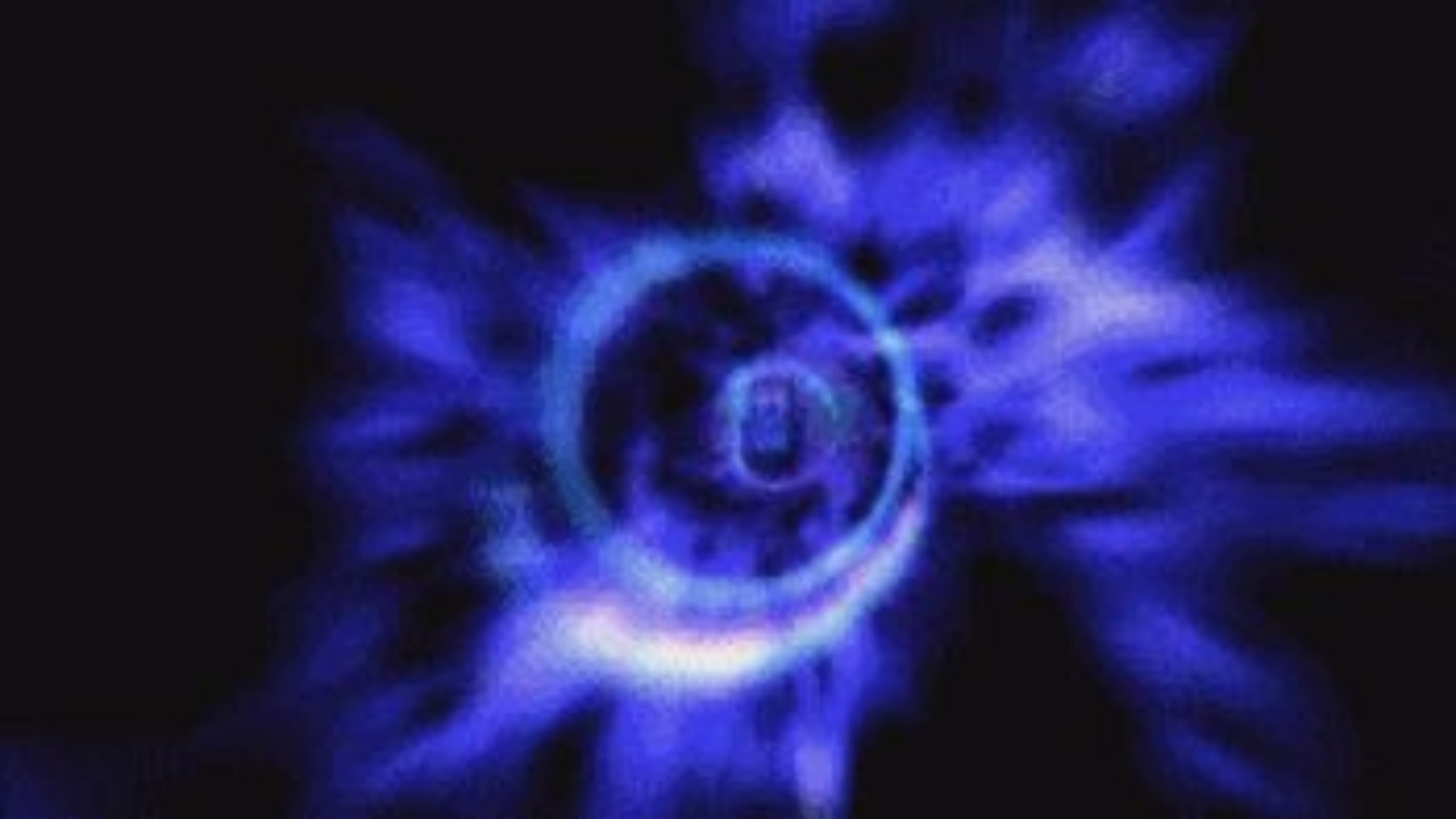
06

MongoDB Core Concepts

02

What is DataBase?

Let’s go back a little bit …
Before computer existed!



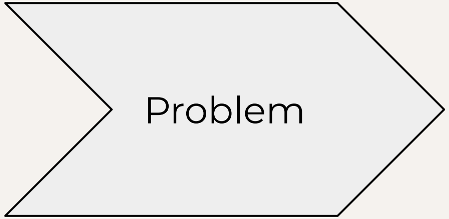
COS DE VVLSINIENSIBVS K NOV
M' VALERIVS M F M N MAXIMVS AN CDXC
MESSALLA COS DE POENEIS ET REGE SICVLOR
C DVILIVS M F M N COS PRIMVS AN CDXCIII
NAVALEM DE SICVLEI ET CLASSE POENICA EGIT
K INTERKALAR
L CORNELIVS L F CN N SCIPIO COS AN CDXCIV
DE POENEIS ET SARDIN CORSICA V ID MART
C AQVILLIVS M F C N FLORVS AN CDXCV
PRO COS DE POENEIS IIII NON OCT
C SVLPICIVS Q F Q N PATERCVLVS AN C
COS DE POENEIS ET SARDEIS
A ATILIVS A F C N CALATINVS
EX SICILIA DE POENEIS XIII K
C ATILIVS M F M N REGVLVS COS
DE POENEIS NAVALEM EGIT
L MANLIVS A F P N VVLSO LONGA
COS DE POENEIS NAVALEM EGIT
SER FVLVIVS M F M N PAETINVS
NOBILIOR PRO COS DE COSSVRENSIB
ET POENEIS NAVALEM EGIT XIII K FEBR
M AEMILIVS M F L N PAVLLVS AN CDXCIX
PRO COS DE COSSVRENSIBVS ET POENIS

31

FROM TEXT

How Was Data Stored Back Then?

- Ancient Egyptian walls
- Mesopotamian / Iraq clay tablets
- Paper records
- Ledgers and notebooks
- Filing cabinets
- Index cards



Problem

- Searching took a long time
- Data was duplicated
- Easy to lose or damage
- Only one person at a time

CTRL + F did not exist :)

Let’s go to the time after computers were invented!

Flat File Model

Enter Computers

- ★ Data moved from paper to digital
- ★ Files replaced folders
- ★ Faster processing

Same problems , new medium.

Flat File Model

	Route No.	Miles	Activity
Record 1	I-95	12	Overlay
Record 2	I-495	05	Patching
Record 3	SR-301	33	Crack seal

What is the Flat File Model?

- ➔ You have one big file (or many separate files), and all data is written directly inside them.
- ➔ EX: Text files, CSV files
- ➔ Flat = no layers, no logic, no connections, no relationships.
- ➔ Same problems , new medium.
- ➔ Computers changed the tool, not the problem

Problems with Flat Files

- ➔ Data duplication
- ➔ Hard to update
- ➔ Inconsistent data
- ➔ No structure enforcement

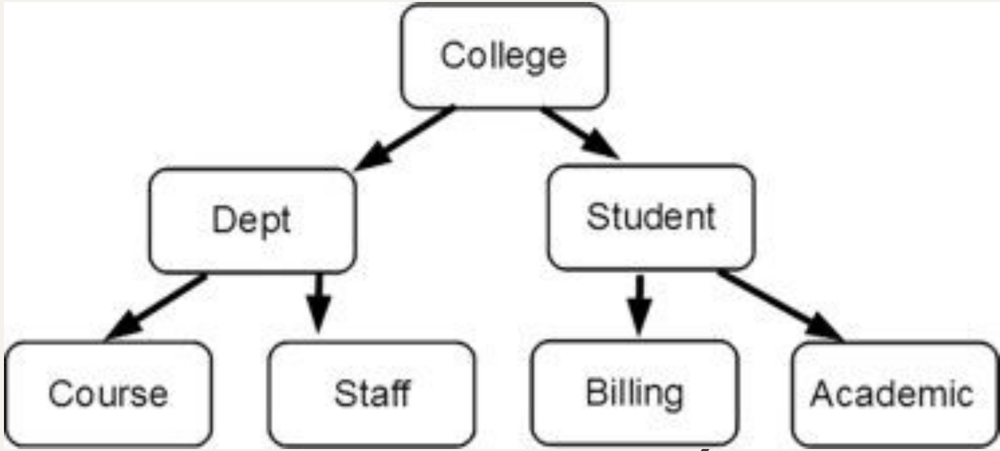
Hierarchical and Network Models (Mid-1960s)

Hierarchical Model

- IBM introduced the Information Management System (IMS).

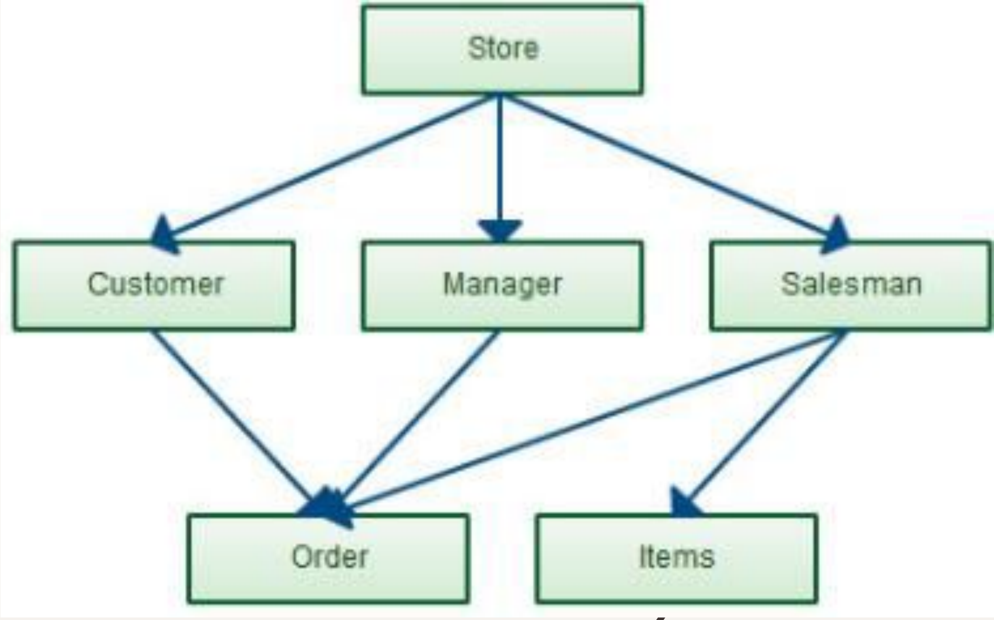
- which used a hierarchical "tree" structure where parent nodes pointed to child nodes.

- While successful for projects like NASA’s lunar lander, it was rigid.



Network Model

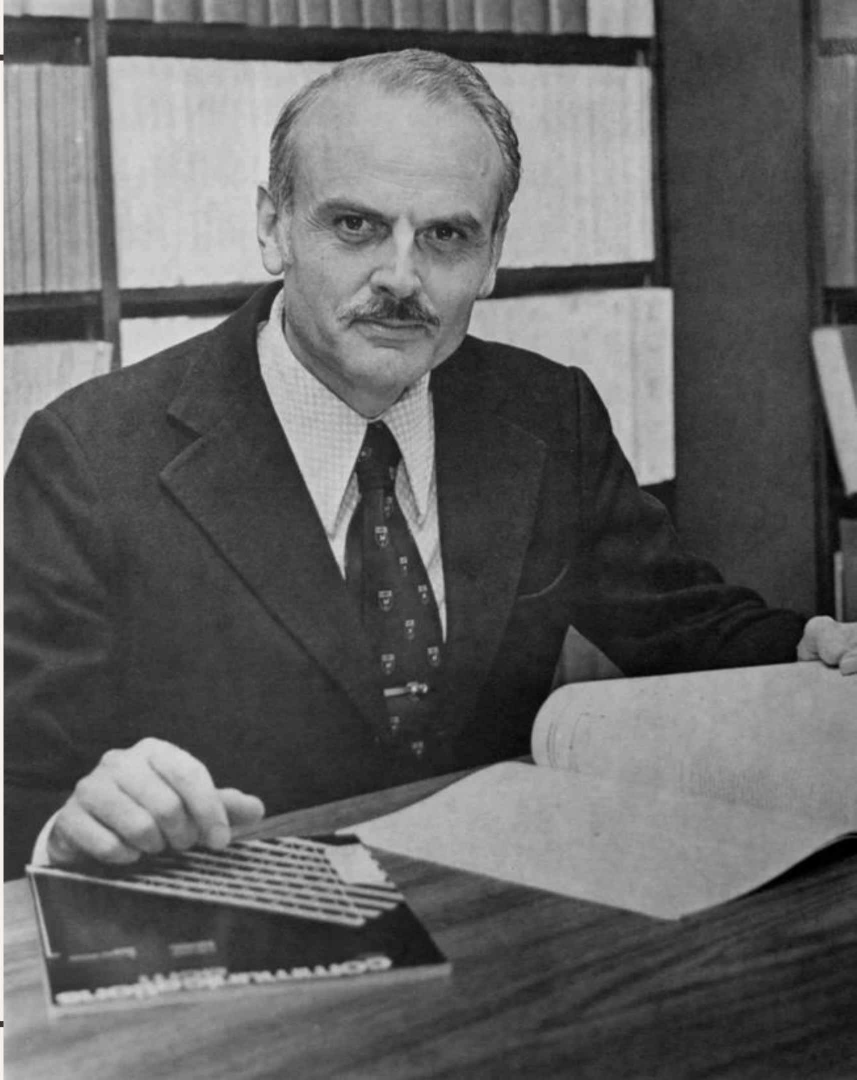
- A more flexible network model was developed by Charles Bachmann at GE.
- allowing child nodes to have multiple parents.
- but it became too difficult to manage as the pointers between data grew complicated



The Relational Breakthrough (1970)

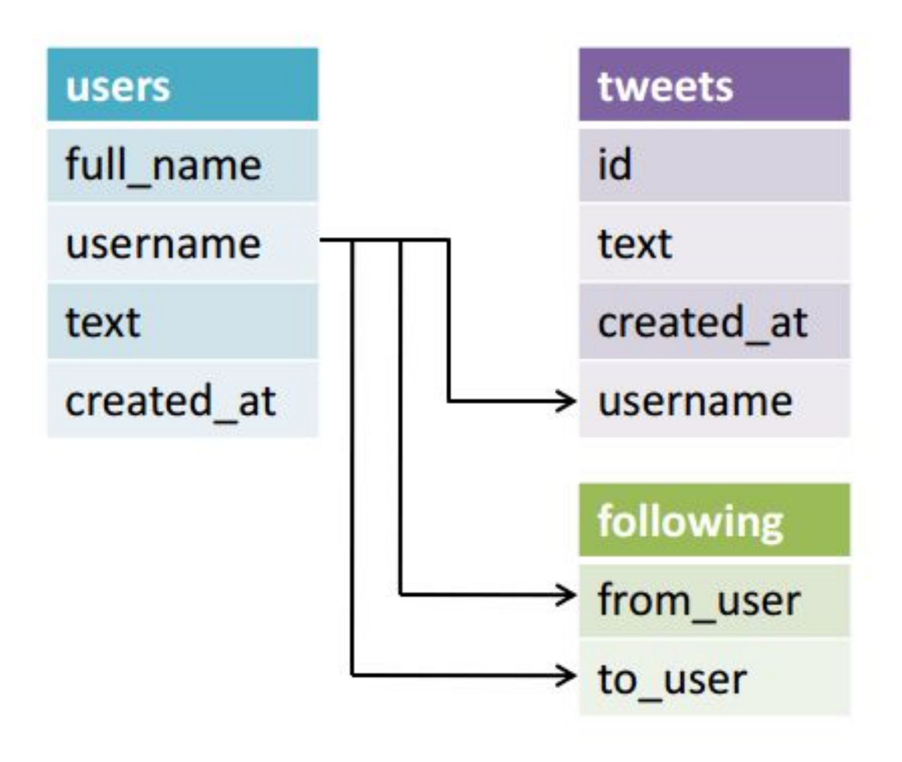
The Relational DB

- **IBM** scientist **Ted Codd** proposed the **relational database** model
- which organized data into simple **tables**.
- This eliminated the need for complex pointers because tables connected through matching data fields, making it much easier to access and change information.



The Relational DB

- Despite its brilliance, IBM was slow to adopt it because it competed with their profitable IMS product



users
full_name
username
text
created_at

tweets
id
text
created_at
username

following
from_user
to_user

Commercialization and Competition (1973–1979)

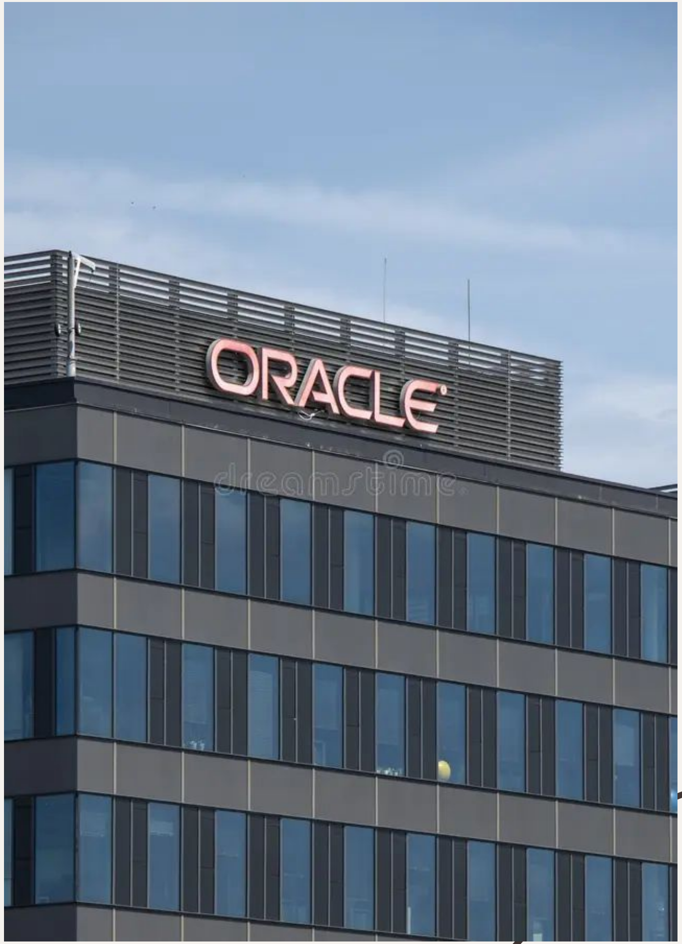
Commercialization and Competition

- ***1973 (The Foundation):*** Researchers at UC Berkeley created **Ingress**, a database that many other companies used to build their own products.
- ***1975 (The Language):*** IBM created an experimental system called **System R**, which introduced **SQL**, the standard language still used today to search and change data.
- ***1977–1979 (The Competition):***
  - **Larry Ellison** saw the opportunity and started his own company to build a compatible database.
  - He released **Oracle** in 1979, beating IBM to the public market by several years

Modern Dominance (1983–Present)

Oracle

- **1983 (The Market Shift):**
  - Oracle had updated its software to run on almost every computer, including IBM's own machines.
  - By the time IBM finally released its own commercial version **(DB2)**, Oracle had already captured the market and was selling to IBM's own customers.
- **Today:**
  - Because of this early competition, relational databases became the global standard.
  - They now organize the data we use for almost everything, including shopping, working, and communicating



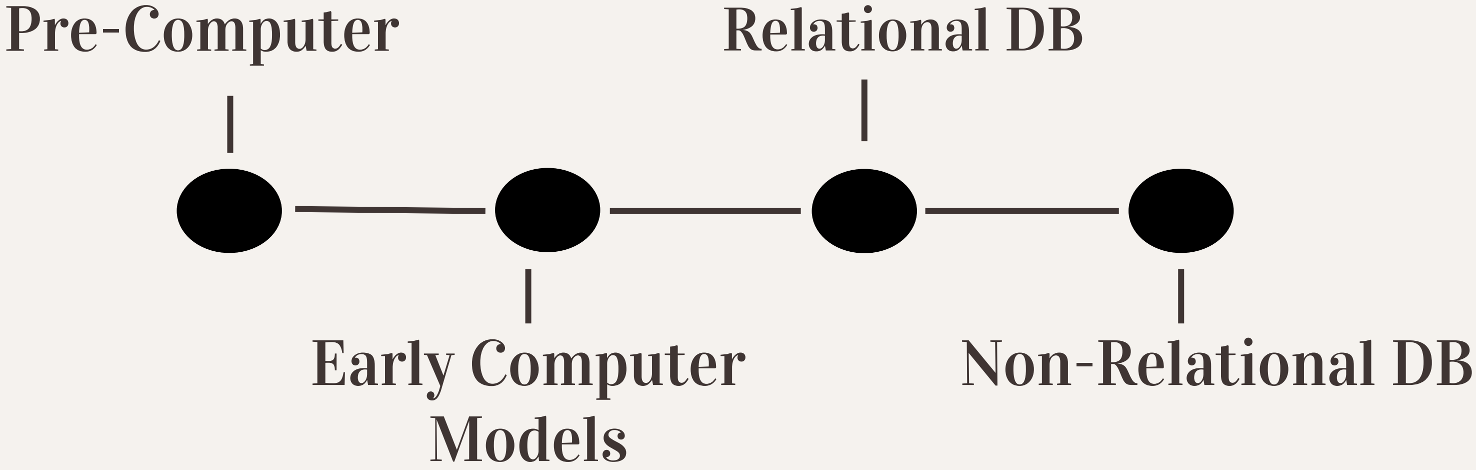
The Rise of NoSQL (Late 2000s)

NoSQL

- As the internet grew, companies like Google and Amazon needed to handle massive amounts of unstructured data (like social media posts or images) that didn't fit easily into relational tables.

- NoSQL ("Not Only SQL") databases emerged to provide a more flexible way to store data that could be easily spread across thousands of servers, prioritizing speed and scale over the strict table structures of the relational model.

Summary



Pre-Computer

Early Computer Models

Relational DB

Non-Relational DB

Summary

- ★ Databases existed long before computers
- ★ Computers changed the medium, not the problem
- ★ Flat files stored data, but didn’t manage it
- ★ Databases were invented to:
  - ■ Organize data
  - ■ Prevent duplication
  - ■ Enable fast search
  - ■ Support multiple users

Table of contents

~~01~~

~~Introduction~~

~~02~~

~~What is a Database?~~

03

Types of Databases

04

Introduction to NoSQL

05

Why MongoDB?

06

MongoDB Core Concepts

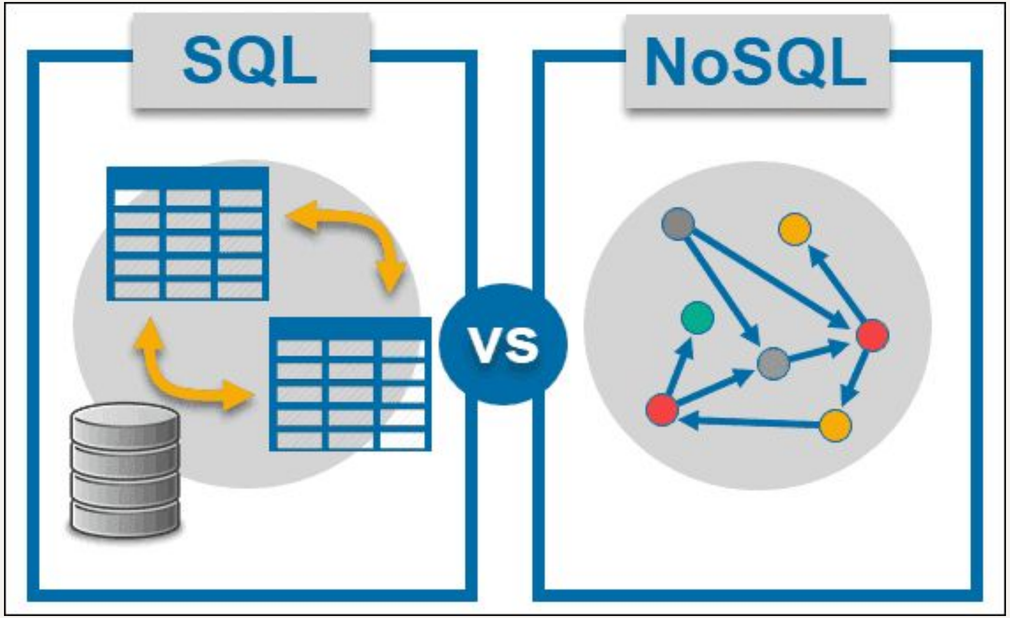
Are All Databases the Same?

- Different problems
- Different data shapes
- Different scale requirements

❖ Main Types of Databases:

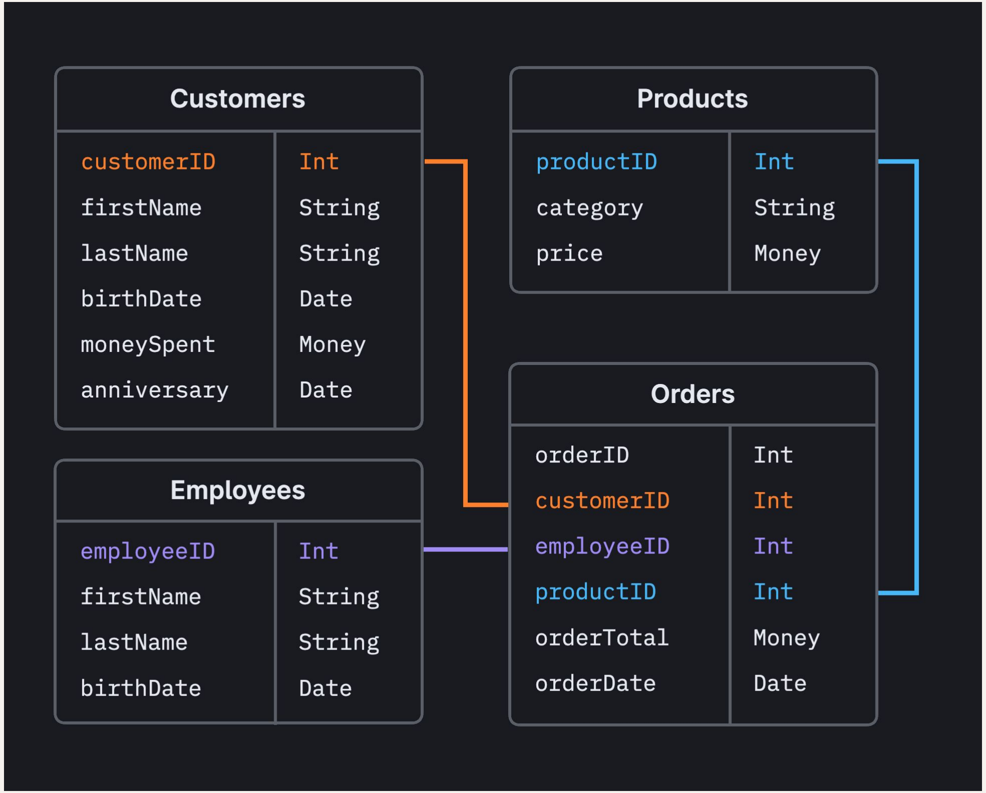
★ Relational Databases (SQL)

★ Non-Relational Databases (NoSQL)



Relational Databases (SQL)

- ★ Data stored in tables
- ★ Fixed schema
- ★ Strong relationships
- ★ ACID guarantees



Customers

customerID	Int
firstName	String
lastName	String
birthDate	Date
moneySpent	Money
anniversary	Date

Products

productID	Int
category	String
price	Money

Employees

employeeID	Int
firstName	String
lastName	String
birthDate	Date

Orders

orderID	Int
customerID	Int
employeeID	Int
productID	Int
orderTotal	Money
orderDate	Date

What Is SQL?

- ★ Structured Query Language
- ★ SQL is not a database. It’s a language.
- ★ A language to talk to databases
- ★ Used to store, read, update, and delete data
- ★ Works with relational databases
- ★ Not a programming language like Java or Python

Why Was SQL Invented?

- ★ Data stored in tables
- ★ Humans need a simple way to ask questions
- ★ Databases need a standard way to understand requests

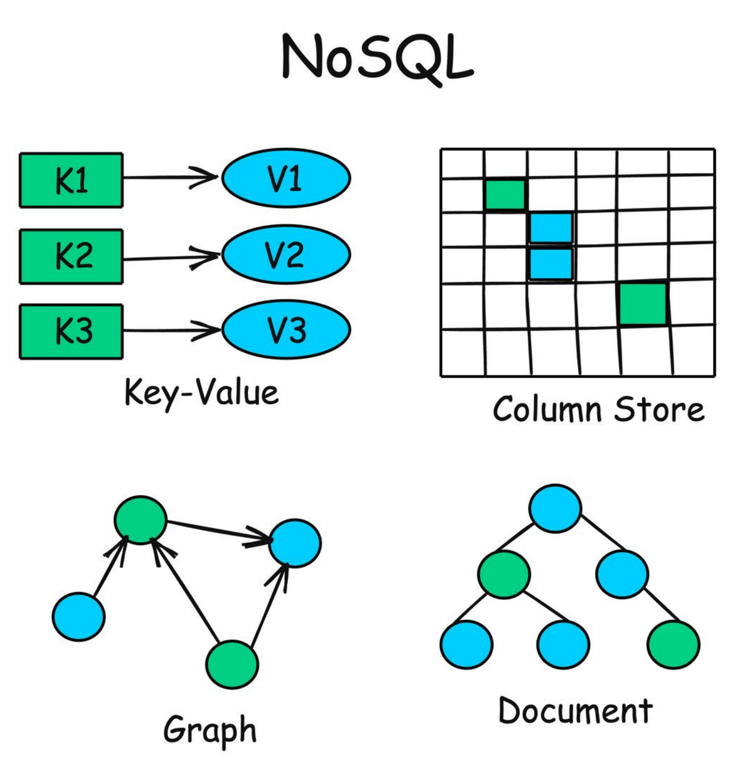
SQL exists to ask questions about data.

SQL DBMS Examples

- PostgreSQL
- MySQL
- SQLite
- Oracle

Non-Relational Databases (NoSQL)

- ★ Flexible schema
- ★ Different data models
- ★ Designed for scale
- ★ High availability



What Is NoSQL?

- ★ A way to store and access data
- ★ Designed for flexibility and scale
- ★ Uses different data models
- ★ Often used in modern applications

NoSQL is a category, not a tool.

Why Was NoSQL Invented?

★ Data became large and diverse

★ Schemas changed frequently

★ Systems needed to scale horizontally

★ Performance under high traffic mattered

NoSQL exists to handle scale and flexibility.

NoSQL DBMS Examples

- Document → **MongoDB**
- Key-Value → **Redis**
- Column → **Cassandra**
- Graph → **Neo4j**

INSIDE OUT 2

Terminologies Alert

What is the difference between :

(**Relational-non Relational**)**Databases** and **DBMS** and **SQL** and **NoSQL**

Terms

Databases

Is the Data.

A database is just organized data stored somewhere.

DBMS

is the software that manages that data.

This is the software that:

- Stores the database
- Protects it
- Organizes it
- Controls access to it

DBMS = the manager of the data

Terms

SQL

is the language.

- Used to communicate with the DBMS
- To tell it what you want to do with the data

SQL = how you talk to the DBMS

NoSQL

is a category of databases.

- Not Only SQL
- It refers to databases that are not based on the relational table model.

So, When to use What?!

Use a Relational DB (SQL) when…

- Data has **strong relationships**
  - Example:User → has many Orders Order → has many Products
- Need **strict consistency** (**ACID**)
  - Example:
    - Banking system, Payment system, Inventory system
  - cannot afford:
    - double payments
    - lost transactions
- Data structure is **stable**
  - If your tables:
    - rarely change structure
    - have clear schema follow
    - predictable patterns

Use Non-Relational DB (NoSQL) when…

- Data is **flexible** or **unstructured**
  - Example:
    - Posts with different fields
    - User profiles with optional data
- Need massive **horizontal scaling** If you're building:
  - Social media
  - Real-time analytics
  - IoT system
  - Logging platform
- Don't rely heavily on JOINs
  - If your app mostly:
    - fetches full objects stores nested
    - data doesn’t require complex relational queries

Side Quest:

Search about Hybrid Database architecture!

Another Side Quest:

Create a practical decision file that helps any developer decide:

Should I use a Relational Database or a Non-Relational Database?

Summary

- Database design didn’t start with just two types,Early models were structured, but rigid.
- In 1970, the relational model was introduced,Tables and logical queries changed everything.
- Relational databases dominated for decades.
- Then data grew bigger and more complex,Applications needed more flexibility and scale.
- New database models emerged,They became known as NoSQL.
- Databases evolved as data evolved.

Table of contents

~~01~~

~~Introduction~~

~~02~~

~~What is a Database?~~

~~03~~

~~Types of~~ Databases

04

Introduction to NoSQL

05

Why MongoDB?

06

MongoDB Core Concepts

The internet changed, so databases had to change, too

NOSQL (Not Only SQL)

★ A category of databases designed for massive data volume and flexible structures.

★ moved away from one big server (vertical scaling) to distributing over a team of servers (horizontal scaling)

★ No complex tables linked by keys

★ You don't need to define every column before adding data

When to Use NOSQL?

- Rapid development
- Frequently changing data structure
- High traffic systems
- Large-scale distributed apps

NoSQL Is Not One Thing

Types of NoSQL Databases

- Document
- Key-Value
- Column-Family
- Graph

NoSQL is an umbrella term.

NoSQL Is Not One Thing

Today’s Focus: Document Databases

Among NoSQL models,
we’ll focus on the Document model ,using MongoDB.

What Is the Document Model? Data is stored as documents
Instead of rows in tables

Each document is:

- Self-contained
- JSON-like
- Flexible in structure

Table of contents

~~01~~

~~Introduction~~

~~02~~

~~What is a Database?~~

~~03~~

~~Types of Databases~~

~~04~~

~~Introduction to NoSQL~~

05

Why MongoDB?

06

MongoDB Core Concepts

Why MongoDB?

- MongoDB is the most popular DBMS for NoSQL Document model databases
- Easy to learn and developer-friendly
- Uses JSON-like structure
- Widely used in modern web applications
- Strong community & ecosystem

Table of contents

~~01~~

~~Introduction~~

~~02~~

~~What is a Database?~~

~~03~~

~~Types of Databases~~

~~04~~

~~Introduction to NoSQL~~

~~05~~

~~Why MongoDB?~~

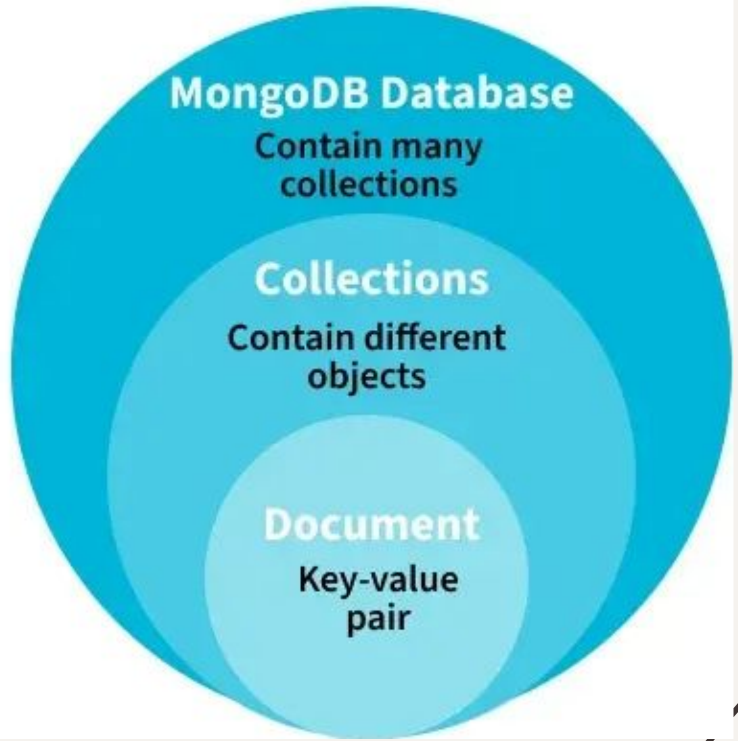
06

MongoDB Core Concepts

MongoDB Structure

How MongoDB Organizes Data:

- →Database
  - → Collections : Tables
    - → Documents : Rows
      - → Fields : Columns



MongoDB Structure

- Documents can contain other documents -> **Nested Document**
- Documents can store **lists (Arrays)**-> Relational databases would need separate tables
- **Flexible** Schema
- Every document has: `"_id"` This is MongoDB’s primary key
  - Unique identifier
  - Automatically generated
  - Used to retrieve documents
- MongoDB stores data as flexible **JSON-like** documents.

Table of contents

~~01~~

~~Int[illegible]on~~

~~02~~

~~What[illegible]tabase?~~

~~03~~

~~[illegible] Databases~~

~~04~~

~~Intro[illegible]ion to NoSQL~~

~~05~~

~~Why [illegible]DB?~~

~~06~~

~~Mon[illegible]Core Concepts~~

Table of contents

07

Schema:
SQL vs MongoDB

08

Basic CRUD Operations

09

MongoDB in Real Applications

10

Common Mistakes & When NOT to Use MongoDB

11

What’s Next?

12

Closing

Schema : SQL vs MongoDB

What is a Schema?

A schema defines:

- What fields exist
- Their data types
- Required vs optional
- Relationships

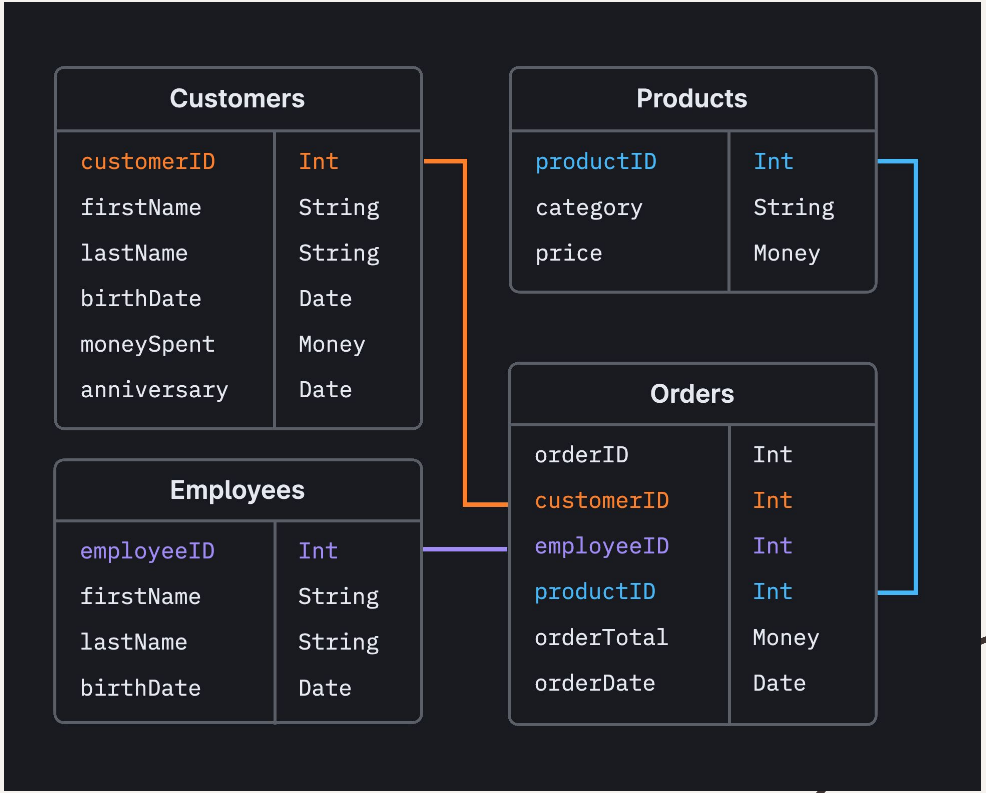
A schema is the structure blueprint of your data.

SQL Schema (Strict)

Before inserting data, you must:

- Define table
- Define columns
- Define data types
- Define constraints

Structure first, data second.



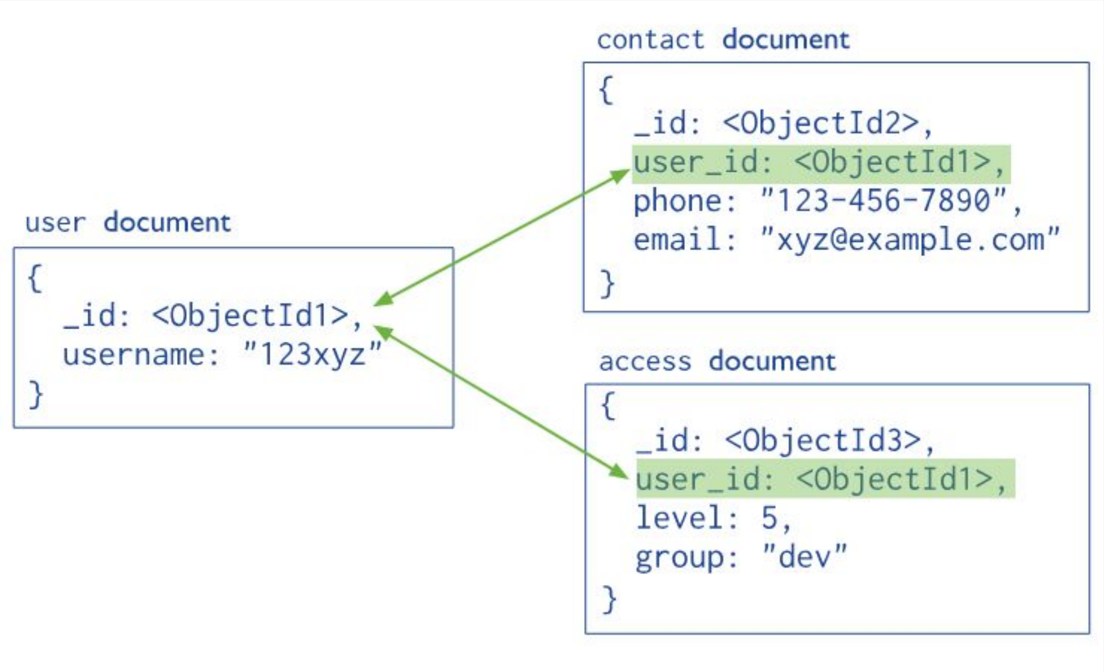
Customers	
customerID	Int
firstName	String
lastName	String
birthDate	Date
moneySpent	Money
anniversary	Date

Products	
productID	Int
category	String
price	Money

Employees	
employeeID	Int
firstName	String
lastName	String
birthDate	Date

Orders	
orderID	Int
customerID	Int
employeeID	Int
productID	Int
orderTotal	Money
orderDate	Date

MongoDB Schema (Flexible)



contact document

```
{
  _id: <ObjectId2>,
  user_id: <ObjectId1>,
  phone: "123-456-7890",
  email: "xyz@example.com"
}
```

user document

```
{
  _id: <ObjectId1>,
  username: "123xyz"
}
```

access document

```
{
  _id: <ObjectId3>,
  user_id: <ObjectId1>,
  level: 5,
  group: "dev"
}
```

- No need to predefine structure.
- Data first, structure optional.

MongoDB Schema (Flexible)

- IMPORTANT NOTE:
  - MongoDB is schema-flexible,not schema-less.

You can:

- Enforce schema using validation

- Define structure at application level

Table of contents

07

S[illegible]
SQL vs MongoDB

08

Basic CRUD
Operations

09

MongoDB in Real
Applications

10

Common Mistakes &
When NOT to Use
MongoDB

11

What’s Next?

12

Closing

Basic CRUD Operations

What is CRUD operations?

The Basic Operations that can be applied to Data

- C → Create → Insert
- R → Read → Find
- U → Update → Update
- D → Delete → Delete

CODE!

Table of contents

~~07~~

~~S[illegible]~~
~~SQL vs MongoDB~~

~~08~~

~~B[illegible] [illegible]UD~~
~~Operations~~

09

MongoDB in Real
Applications

10

Common Mistakes &
When NOT to Use
MongoDB

11

What’s Next?

12

Closing

Where does MongoDB actually shine

MongoDB is strong when:

- Data structure changes frequently
- Rapid development is required
- Applications scale horizontally
- Data is naturally document-like

Where does MongoDB actually shine

Examples:

- Social media platforms
- Chat applications
- Content management systems
- Analytics dashboards

Table of contents

~~07~~

~~S[illegible]~~
~~SQL vs MongoDB~~

~~08~~

~~B[illegible]UD~~
~~Operations~~

~~09~~

~~Mong[illegible] Real~~
~~Applications~~

10

Common Mistakes &
When NOT to Use
MongoDB

11

What’s Next?

12

Closing

Common Mistakes

- Ignoring schema design
- Overusing embedding
- No proper indexing
- Treating it like SQL
- Assuming it replaces relational databases

When NOT to Use MongoDB

- Complex joins across many entities
- Strict ACID financial systems
- Strong relational integrity requirements
- Heavy transactional systems

Banking systems → better suited for relational databases.

Trade-offs

MongoDB gives:

- Flexibility
- Horizontal scalability

But you give up:

- Strong enforced relationships
- Strict schema control
- Traditional joins

Choose based on the problem

MongoDB is not better than SQL.
It’s better for certain problems

Table of contents

07

S[illegible] SQL vs MongoDB

08

B[illegible]UD Operations

09

Mong[illegible] Real Applications

10

Commo[illegible]takes & Whe[illegible] Use MongoDB

11

What's Next?

12

Closing

What’s Next?

Next session :
Relational Database (SQL)

Closing!

Next session will be next week , Study Well till then!

Do you Have any Questions?

Resources

- History of DBMS (GeeksforGeeks)
- Database Types (Youtube video)
- Ted Cod Paper: Relational Model of Data (pdf)
- How do NoSQL databases work? (Youtube video)
- What is a Document Database? (MongoDb Document)
- What is a NoSQL? (MongoDb Document)
- Types of databases (GeeksforGeeks)
- IBM DOC about Relational Databases
- Blog about the development of DB
- MongoDB Document

Thanks!

Kahoot