

Data Modelling

Sancheeta Kaushal | Ex-EM Blinkit

Disclaimer

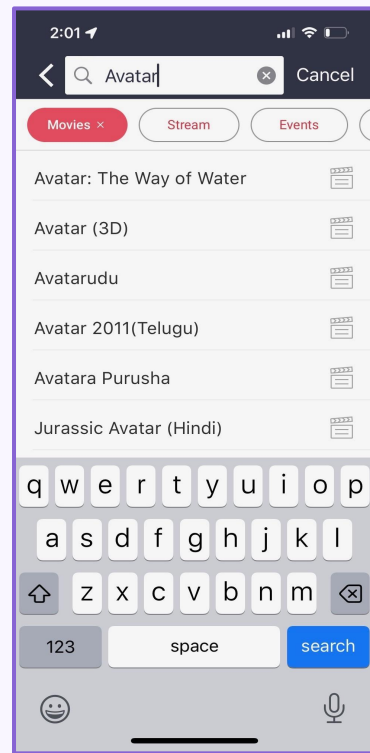
1. What is the single most important skill you need to learn while learning databases ?
2. Lot of coding bias.
3. Databases are 90% about know what to use in which scenario.
4. Today you'll read about 7-8 different types of NoSQL databases. It's going to be overwhelming but don't panic, we will do a quiz and 3 practice case studies in lecture 7 for you to practice things better.
5. Keep in mind, what matters is are you able to evaluate a scenario against the factors for deciding which database to choose.

Problem Statement

Problem Solving Case: Bookmyshow

Continuing with our Bookmyshow case study.

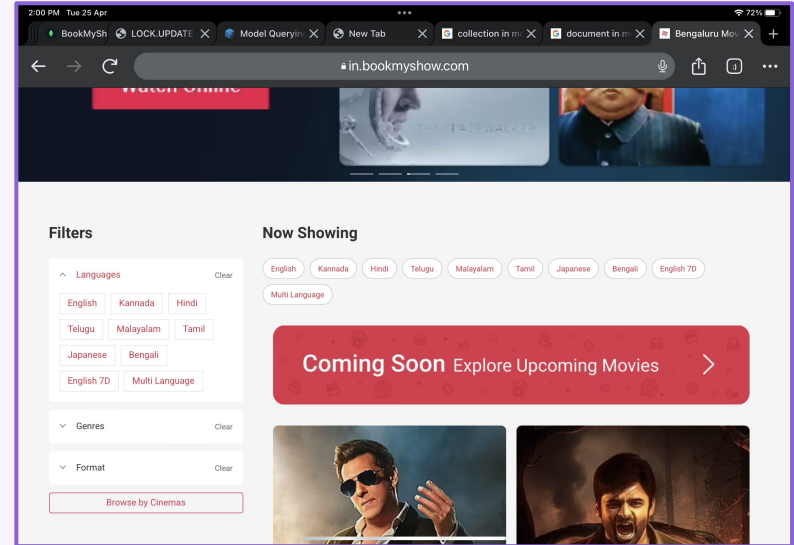
1. Implement code for quick retrieval of static movie data - Name, Cast, Crew, Movie Plot, Runtime, Language, Genre, etc.
2. Build an endpoint for storing and retrieval of comments and ratings for a given movie. Which database would you choose and why ?
3. Build an endpoint where one can search for movies and theaters. Refer image on the right.



Problem Statement

Problem Solving Case: Bookmyshow

4. Implement language (Hindi, English, Kannada, Telugu etc.) and genre (Romantic, Action, Scifi) and format (2D, 3D, IMAX) filters.



What all skills you need to learn to build this ?

- Factors to consider when deciding on a database
- Learn different types of databases that exist
- Understand what type of database should be used in what scenario

Agenda

- Factors to consider when deciding on a database
- Limitations of SQL Databases
- NoSQL Databases
- Types of NoSQL Databases

Quick Recap

For the following scenarios which type of database would you choose and why ?

- You are trying to create an order on an e-commerce app.
- You want to store data for static product attributes.
- Your company uses proprietary Oracle Database and wants to save cost.
- You are trying to cache data for faster retrieval.

Challenges with databases

- Scaling
- Availability
- Performance
- Consistency
- Security and privacy



**But what are the true limitations of SQL
Databases ?**

Limitation of SQL Databases

1. Need for well defined schema
2. Hard to scale horizontally
3. Poor performance as data grows
4. Inability to solve for scenarios other than transactions
5. Management overhead

SQL to NoSQL Databases

1. Need for well defined schema → No fixed schema - Unstructured or Semi-structured
2. Hard to scale horizontally → Horizontally distributed architecture
3. Poor performance as data grows → No constraints leads to higher performance
4. Inability to solve for scenarios other than transactions → Supports various data models
 - graph based, document based, key value pairs, column based, time series based.
5. Management overhead → Data management taken care by adding configuration

Factors to consider for choosing DB

- Type of data
 - Structured vs Unstructured
 - Well defined vs Loosely defined relationships
- Scalability
 - Horizontal vs Vertical
- Performance/Query Patterns
 - Read vs Write vs Both
- Flexibility
 - Ability to get started quickly and build quick prototypes

Factors to consider for choosing DB

- Consistency
 - Strong vs Eventual Consistency
 - ACID vs BASE (Basically Available, Soft State, Eventually Consistent)
- Availability vs Fault Tolerance
- Community
 - Open Source vs Commercial
 - Integration effort
 - Primary use case
- Cost - Licensing vs Operational

Factors to consider for choosing DB

- Security and Regulatory Compliance
- Geographical Distribution
- Maintenance and Management
- X factors - What's special in your scenario ?

What are the types of NoSQL Databases ?

Types of NoSQL Databases

- Document Stores
- Key Value Stores
- Time series Database
- Search databases
- Graph Databases
- Column Databases
- Vector Databases
- File Systems

Document Based Database

- Type of NoSQL database that manages data in a document-oriented format.
- "document" refers to a complex data structure that may contain nested fields and arrays, and is usually represented in a standard format like JSON or BSON.
- Great for dynamic schemas including hierarchical relationships
- Great for handling large amounts of data
- MongoDB, Couchbase, CouchDB, Amazon DocumentDB

Document Based Database – Use Case

- Content Management System as it can handle varied and dynamic content, including text, images, and multimedia assets. → Invision, Adobe
- Product Merchandising and Attribute Details → Blinkit, Ebay
- IoT sensor data → Bosch
- Policy details with transactional support → Metlife Insurance
- User Activity Log Data for compliance → Kalarna

MongoDB

- Type of data - Unstructured
 - Each document can have its own schema, allowing for varying attributes within the same collection.
 - Rich support for nested structures.
- Scalability & Flexibility
 - Use horizontal scaling model uses sharding.
 - Great for building quick prototypes due to schema flexibility.

MongoDB

- Transactional support - Decent support for transactions.
 - ACID Compliant across documents, collections and databases.
 - Transactional operations are slower than non-transactional operations as it takes lock on the data being modified. Max time limit of 1 mint.
 - Drivers and libraries support this change.
- Query Patterns
 - Great for heavy reads
 - Okay for heavy writes, better options available like DynamoDB, Cassandra

Limitations of Document Databases

- Limited Transaction Support
- Inability to model complex relationships
- Suboptimal for complex analytics and aggregations - Columnar databases

Imagine an e-commerce platform that serves millions of customers worldwide. The website has a variety of products, each with detailed descriptions, images, reviews, and pricing information. Users frequently browse, search for products, and check availability.

To ensure a fast and responsive user experience, the system must deliver this information quickly under 30 ms. However, repeatedly querying the primary database for the same information can be slow and resource-intensive.

Key – Value Stores

Key Value Stores

- A type of NoSQL database that store data in a simple key-value pair format. They are known for their simplicity, performance, and scalability.
- Redis, Amazon DynamoDB (in key-value mode)
- High read and write throughput enabling fast retrievals.
- Session management, caching, distributed configuration management, real time analytics

Key Value Stores – Exercise

1. Type of data - Structured vs Unstructured
2. Scalability & Flexibility (Prototyping)
3. Consistency i.e. Transactional support
4. Read vs Write - Query patterns (Performance)
5. Community & Ecosystem
6. X factors - What's special in your scenario ?

Special case of Dynamo DB

- Type of data
 - Supports both key-value and document data models
- Scalability
 - Fully managed by AWS. So, automated scaling.
- Performance
 - Single-digit millisecond response times
 - Can handle more than 10 trillion requests per day, making it suitable for applications that require low-latency and high-throughput.

Special case of Dynamo DB

- Consistency
 - Strong Consistency
 - Strong support for transactions
- Availability
 - Provides 99.999% availability and replicates data across three geographically separated Availability Zones
- X-factor
 - Real time Event Driven Programming

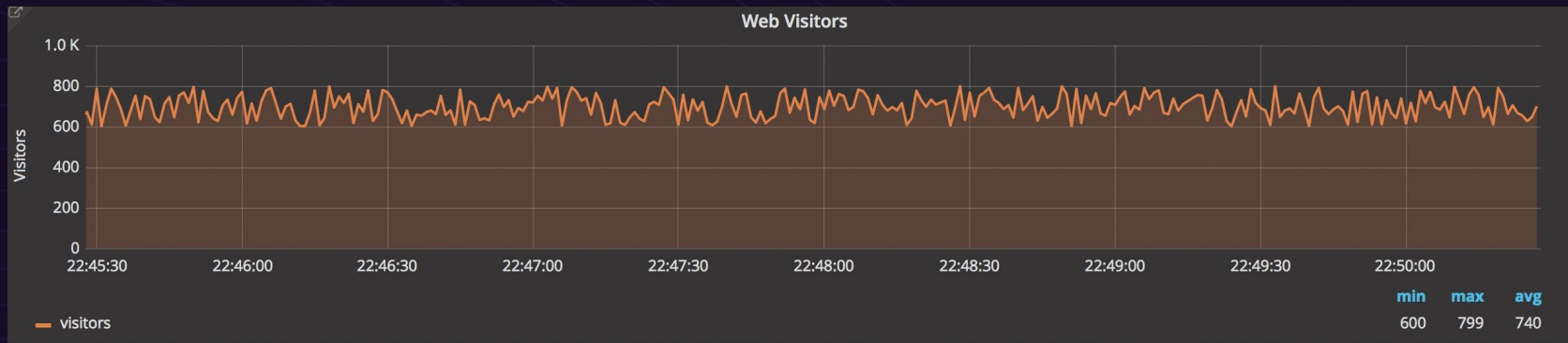
Imagine a smartwatch that continuously tracks (every minute) various health metrics, such as heart rate, blood pressure, oxygen saturation, sleep patterns, step counts, and calorie burn rate. The company wants to analyze this data over time to understand trends, suggest lifestyle adjustments, set goals, and even share this information with healthcare providers on behalf of user when they give permission.

Time Series Database

Time Series Databases

- Type of data
 - Data is organized and stored based on timestamps, allowing for efficient retrieval and analysis of data points over time
- Timestamp, Value, Tags/Metadata, Sampling Rate
- Aggregation & Downsampling to analyse data
- Algorithms like moving averages, exponential smoothing, and Fourier transforms
- InfluxDB, TimescaleDB, OpenTSDB, Graphite, and Apache Druid
- IoT, Financial Services, Log Analysis, Infra Monitoring, Energy & Env Data Monitoring

Time Series Databases




Imagine a global social networking platform with billions of users engaging in activities like posting updates, commenting, liking posts, sharing media, and forming connections. The platform must manage a vast, diverse, and highly interconnected set of data. Users expect quick and seamless experiences, regardless of their location.

Wide Column Stores

Wide Column Stores Database

- Type of NoSQL database that organizes data in a column-oriented manner.
- Instead of storing entire rows together, each column is stored separately, allowing for efficient retrieval and analysis of specific columns of data.
- Related columns are grouped into column families.
- Designed to scale horizontally by distributing data across multiple nodes in a cluster. Fast read and write across multiple data centers.
- Social media, gaming, Online streaming
- Apache Cassandra, Google BigTable, ScyllaDB, Apache Hbase

Apache Cassandra

- Consistency model
 - Both strong and eventual consistency on a per-operation basis.
 - Lightweight transactions
- Scalability
 - Easily scale horizontally by adding new machines to the system without any downtime by expanding across multiple nodes or even across different data centers.
- Performance
 - Write optimised. Handle heavy write loads without degrading read efficiency.
 - Efficient storage and querying of large amounts of distributed data,  Airtribe

Wide Column Stores vs Document Based

Mongo DB	Cassandra
Flexible schema which can store data in hierarchies.	Flexible schema with write intensive workloads.
Manual configuration & explicit choice of shards values.	Automatic Partitioning using consistent hashing.
Strong Consistency - Supports document level transactions.	Eventual Consistency - Limited support for transactions.
Exceptional Querying Capabilities.	Exceptional scalability and fault tolerance.

Imagine a large financial institution that offers various services like banking, credit cards, loans, and investments. Fraudulent activities within financial systems can be complex and interconnected, involving multiple accounts, transactions, and entities.

Traditional relational databases may struggle to represent and analyze the intricate web of relationships that signify fraudulent behavior. Detecting fraud requires understanding not just individual transactions but the patterns and connections between them.

Graph Databases

Graph Databases

- Designed to store and manage data using graph structures.
- Data entities are represented as nodes, and the relationships between entities are represented as edges or arcs.
- Network like structure enables deep insights and patterns about data.
- Social Networks, Content Management for Recommendation Engines, Knowledge Graphs, Network Operations, Genome Analysis etc.
- Neo4j, Amazon Neptune, JanusGraph, and Apache Giraph

An e-commerce company wants to enhance its content recommendation engine. They wish to provide personalized recommendations to users by finding products that are similar to those the users have previously interacted with. To do this, they represent each product as a high-dimensional vector based on various features such as category, brand, price, user ratings, textual descriptions, and images. These vectors encapsulate the essential characteristics of the products.

Vector Databases

Vector Databases

- Type of data
 - Specialized databases designed to handle vector data efficiently. High dimensional data with 1000s of dimensions with real time requirements.
 - Used in applications for similarity search and machine learning
- Recommendation Engines for e-commerce platforms, streaming services and social media
- Chatbots, Image and Video Retrieval, Text Similarity, Bioinformatics
- Faiss by Facebook, Milvus, Annoy by Spotify, Vearch

Filestore – S3

Mental Model for deciding database

- Look at all the factors to consider while choosing database.
- First think if the data has well defined relationships in real world.
- Type of consistency required in the problem you are trying to solve.
- Is the data structured vs unstructured.
- Scalability requirements of the system.
- Is the system going to be read-heavy or write-heavy ?

Problem Solving

Example Case:

For each of the following scenarios, which database would you use to model the data and why ?

Scenario 1: Instamart wants to create an offers system. An offer is attached with expiry date at the time of creation itself. You can't edit the offer values once created, you can only expire it.

RDBMS

Scenario 2: The definition of different offers have different fields. Offer can be a bundle(pack of x), combo(buy x and y at reduced price), flat price off.

MongoDB

Scenario 3: You are trying to build a recommendation system to suggest new products to users based on their past purchases ?

Graph Database/Document Based Database

Scenario 4: You are building an inhouse tool for analysing user logs data.

Document Based Database/TimeSeries Database

Problem Solving

Example Case:

For each of the following scenarios, which database would you use to model the data and why ?

Scenario 5: You want to store vectorised input data for processing machine learning models.

Vector Database

Scenario 6: An online streaming service that requires storing large volumes of video metadata, user preferences, and playback history.

Wide column store(Cassandra)

Scenario 7: You want to quickly access player profiles, achievements, and in-game statistics in a multiplayer video game.

Redis

Scenario 8: You are building an inhouse tool for analysing server logs data.

Time Series Database

Thank You!