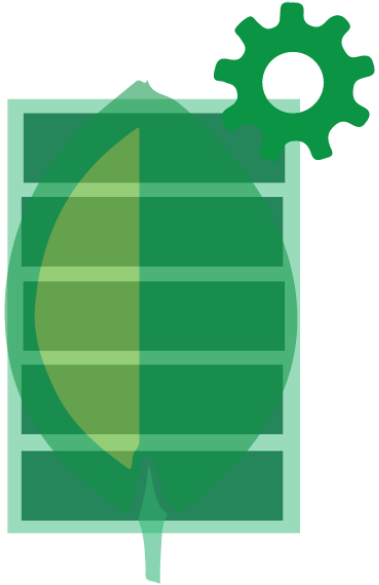
edureka!



MODULE-3
SCHEMA DESIGN AND DATA MODELING

Course Topics

edureka!

- → Module 1
  - » Design Goals, Architecture and Installation
- → Module 2
  - » CRUD Operations
- → Module 3
  - » **Schema Design and Data Modelling**
- → Module 4
  - » Administration
- → Module 5
  - » Scalability and Availability
- → Module 6
  - » Indexing and Aggregation Framework
- → Module 7
  - » Application Engineering and MongoDB Tools
- → Module 8
  - » Project, Additional Concepts and Case Studies

Objectives

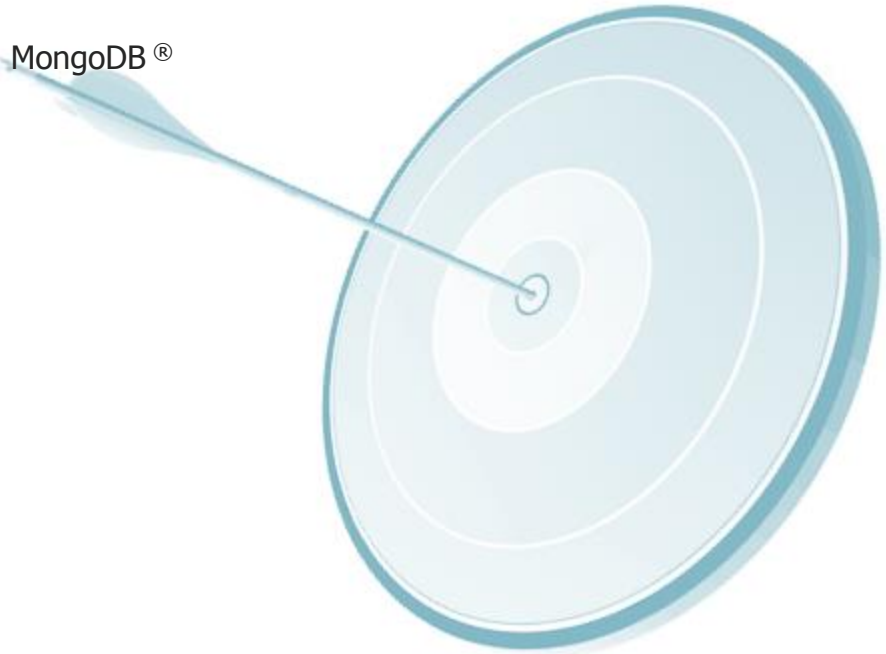
At the end of this module, you will be able to

→ Understand different concepts of data modeling in MongoDB®

→ Understand different types of data model

→ Understand the challenges of designing data model in MongoDB ®

→ Apply the knowledge in a real world use case



Annie's Question

How to close Inactive Cursors?

Annie's Answer

In the mongo shell, you can set the no Timeout flag
var myCursor=db.inventory.find().addOption(DBQuery.Option.noTimeout);

- → By default, the server will automatically close the cursor after 10 minutes of inactivity or if client has exhausted the cursor.
- → To override this behavior, you can specify the noTimeout wire protocol flag in your query.
- → However, you should either close the cursor manually or exhaust the cursor.

Annie's Question

What is capped collection?

Annie's Answer

A fixed-sized collection that automatically overwrites its oldest entries when it reaches its maximum size.

Annie's Question

How to check whether collection is capped or not?

Annie's Answer

db.Collection_Name.isCapped()

Annie's Question

What is Record Padding?

Annie's Answer

The extra space allocated to document on the disk to prevent moving a document when it grows as the result of update() operations.

Data Modeling Concepts

edureka!

→ It represents the nature of data, business rules governing the data and how it will be organized in the database.

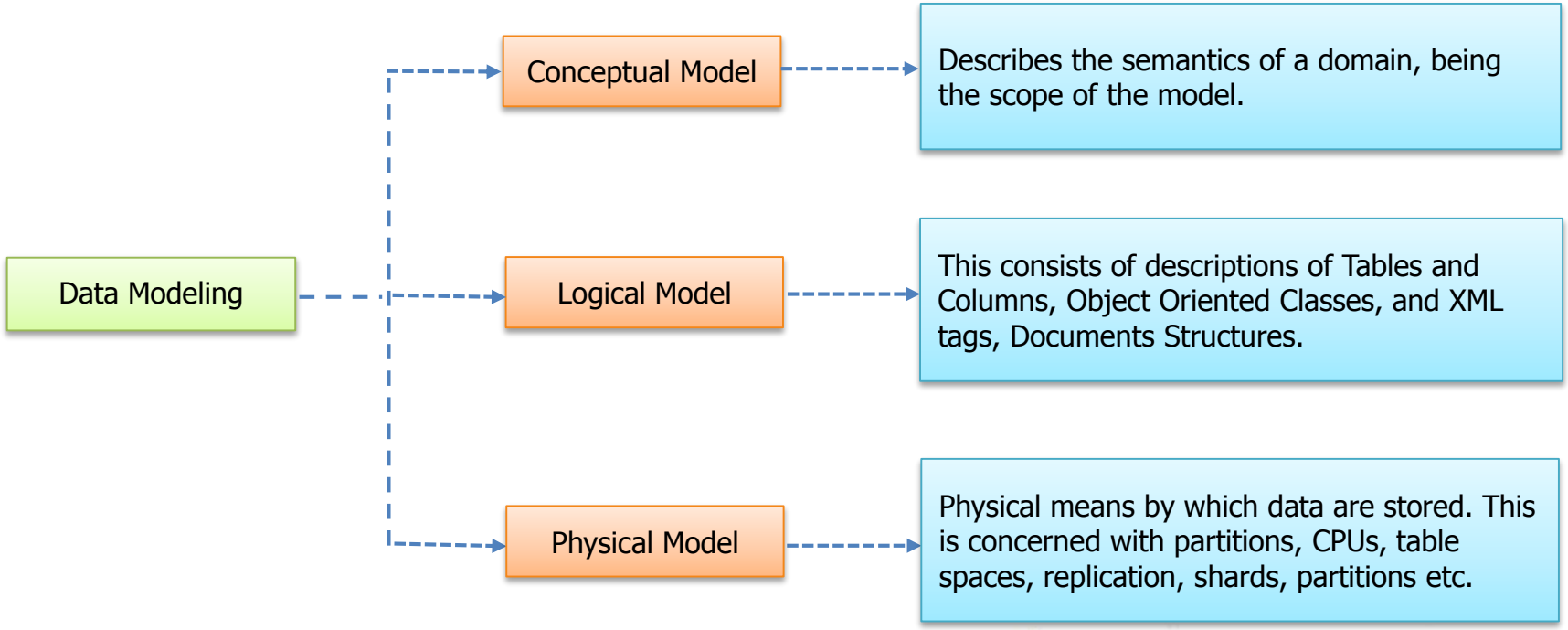
→ A data model explicitly determines the structure of data.

→ A data model can be sometimes referred to as a data structure.

→ It is used to communicate between the business people defining the requirements for computer system and the technical people defining the design in response to those requirements.

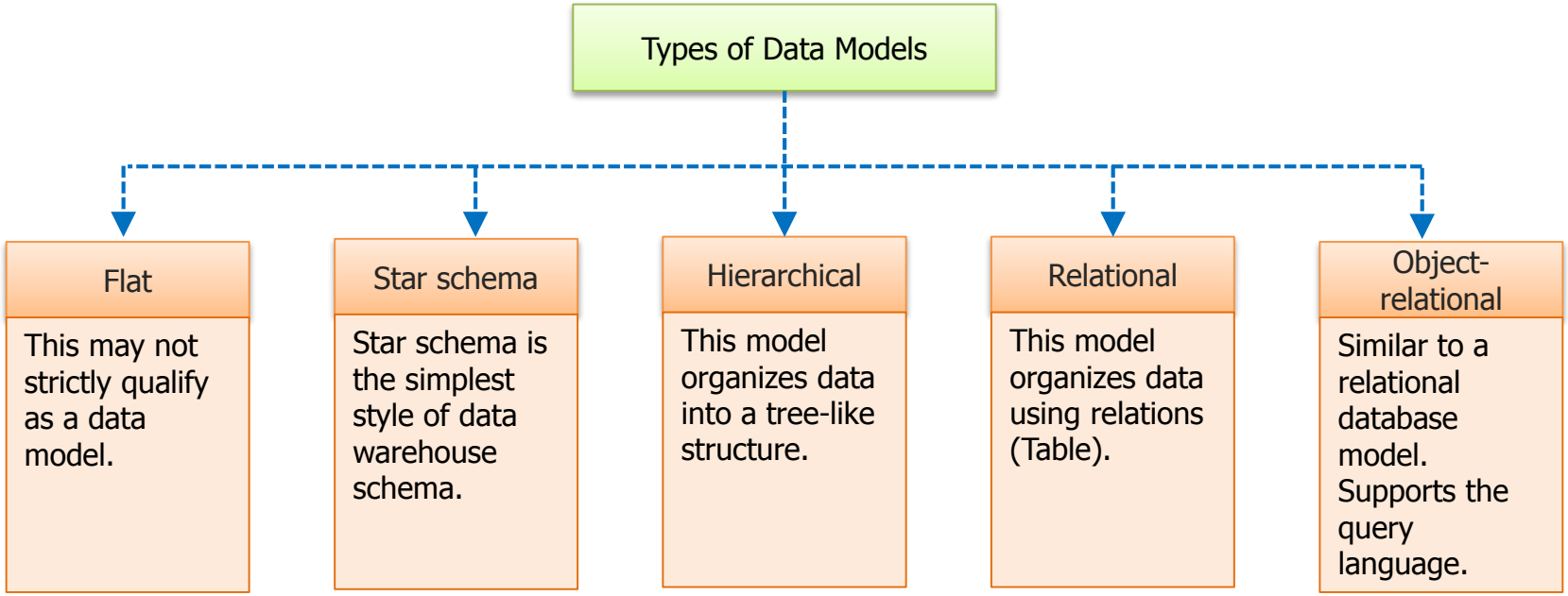
→ They are used to show the data needed and are created by business processes.

Data Model Perspectives (Levels)

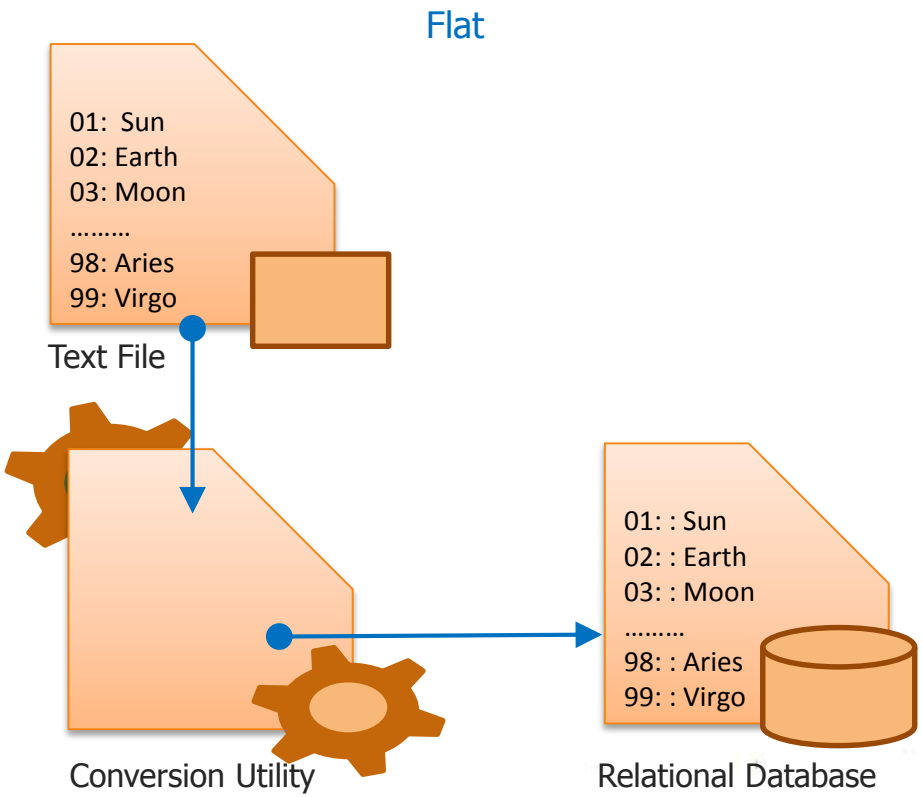


Types of Data Models

edureka!



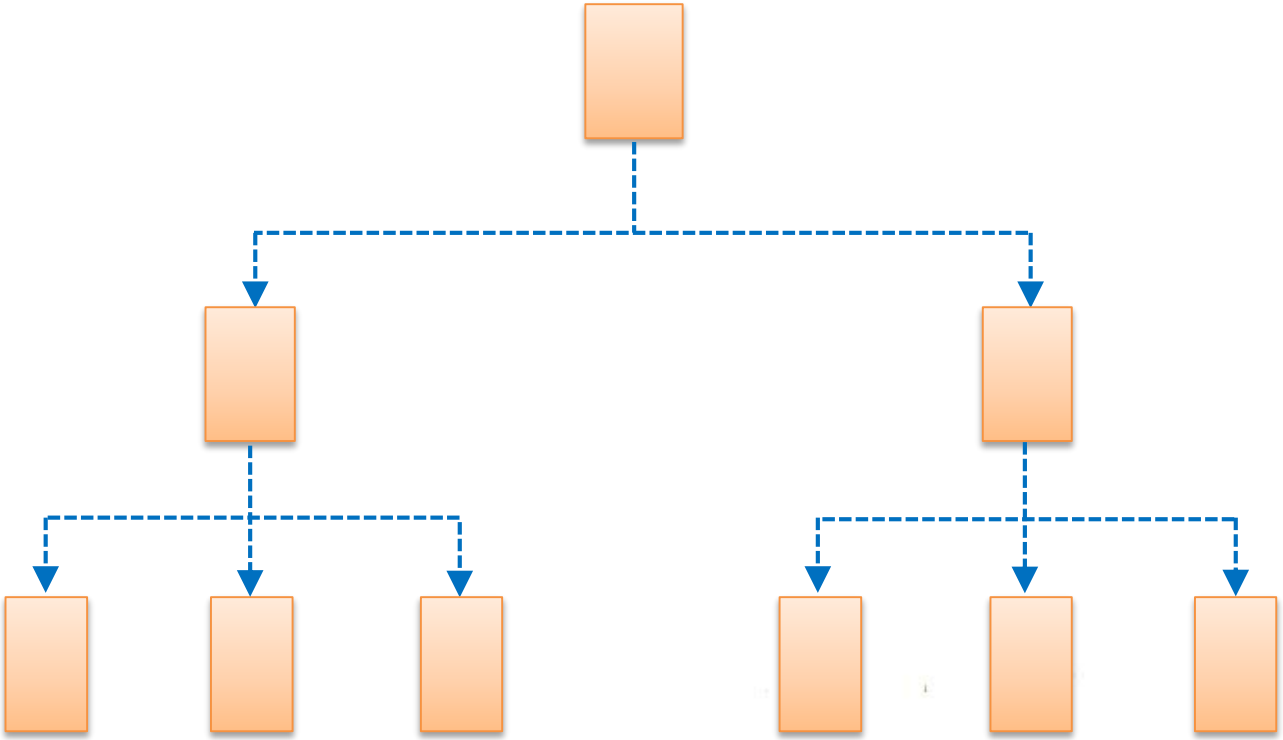
Types of Data Models (Contd.)



Types of Data Models (Contd.)

edureka!

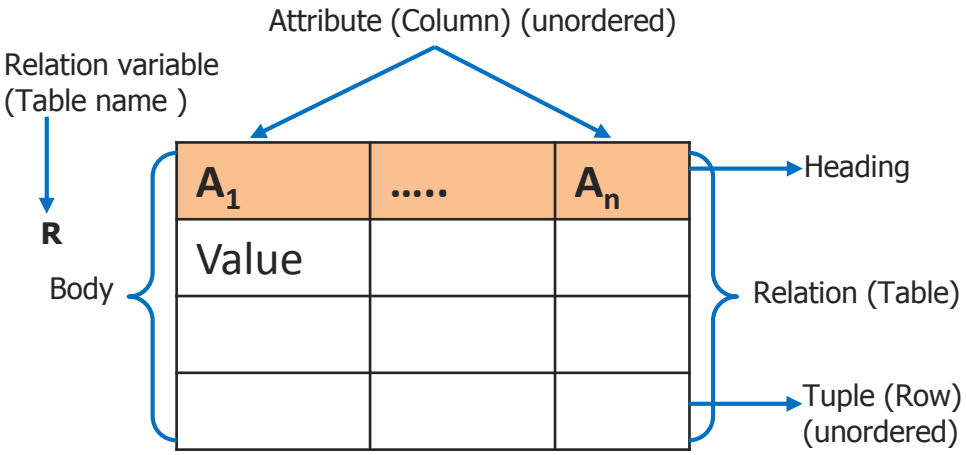
Hierarchical



Types of Data Models (Contd.)

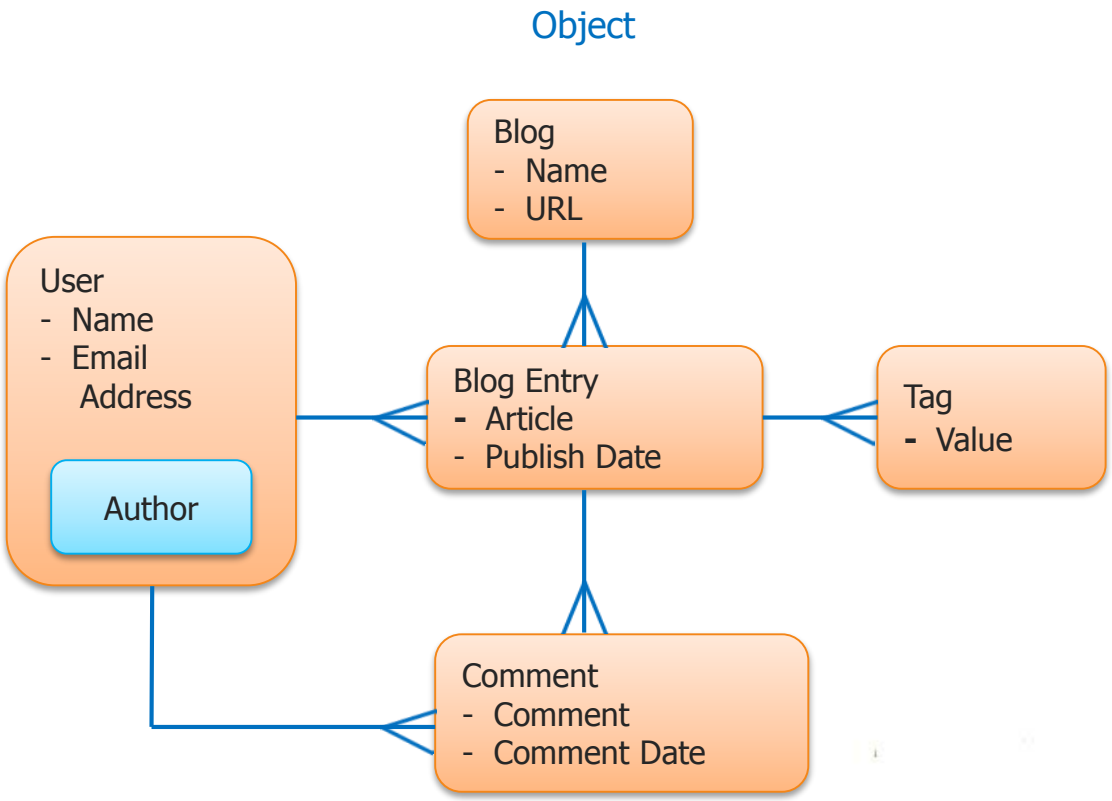
edureka!

Relational



Types of Data Models (Contd.)

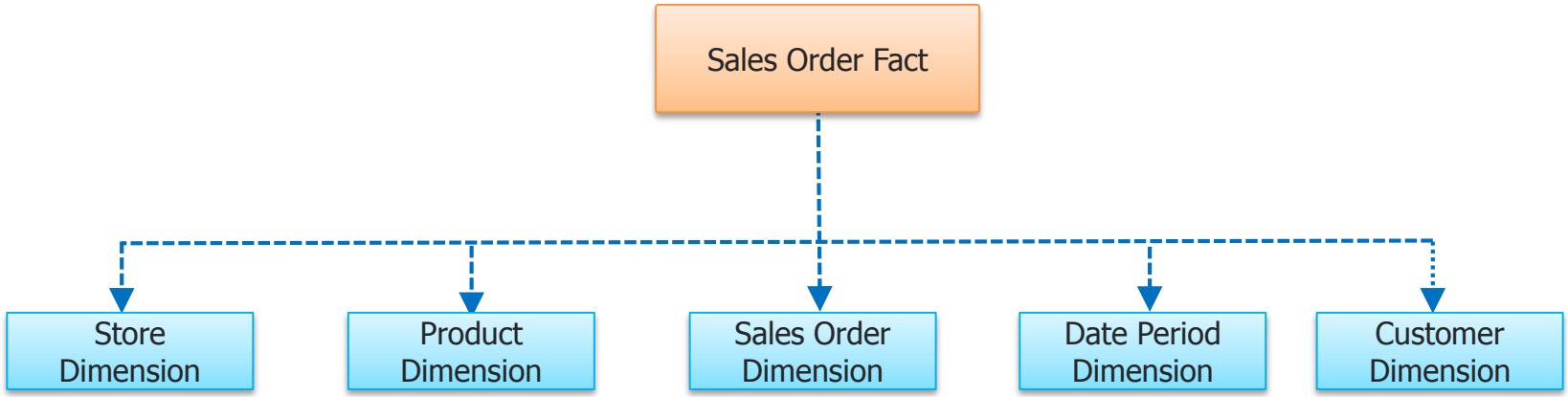
edureka!



Types of Data Models (Contd.)

edureka!

Star Schema



Why Data Modeling ?

edureka!

→ Database should be catered to fit the needs of the company or business using the database.

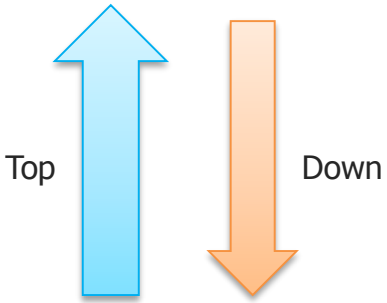
→ A database needs to be user friendly. It should provide means to users for getting and storing their information.

→ It also needs to be secure against outside attacks.

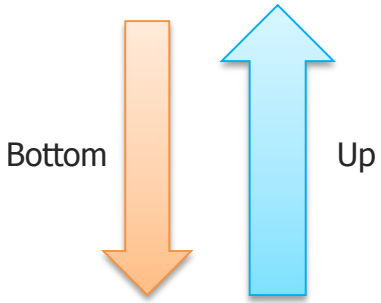
→ Since model of functioning is different for every business, database also differs accordingly.

Database Model should be driven by the business needs.

Database Model – Approach



Top Down Approach



Bottom Up Approach

Analogy b/w RDBMS and MongoDB® Data Model

edureka!

RDBMS	MongoDB®
Database Server	Database Server (mongod cluster)
Database	mongod
Table	Collections
Row & Column	Documents/Key
Primary Key	_ID
Indexes	Indexes
Replication	Replication
Partitioning	Sharding
Joins	Embedding & Linking (References)

MongoDB® Data Model

edureka!

→ Data in MongoDB® has a flexible schema.

→ Collections do not enforce document structure.

→ In practice, however, the documents in a collection share a similar structure.

→ This flexibility gives you data-modeling choices to match your application and its performance requirements.

Key Challenges for Data Modeling in MongoDB®

edureka!

→ Balancing the needs of the application

→ Performance characteristics of the database engine, and the data retrieval patterns

→ Queries, Updates and processing of the data

→ Paradigm shift from the traditional approach

→ No ACID transactions, No Joins

Considerations while Data Modeling in MongoDB®

edureka!

→ Design your schema according to user requirements.

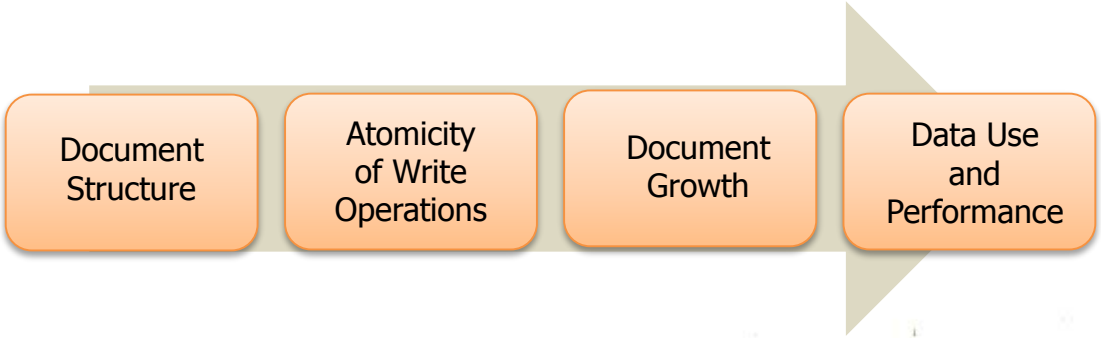
→ Combine objects into one document if you will use them together. Otherwise separate them (but make sure there should not be need of joins).

→ Duplicate the data (but limited) because disk space is cheap as compare to compute time.

→ Do joins while write, not on read.

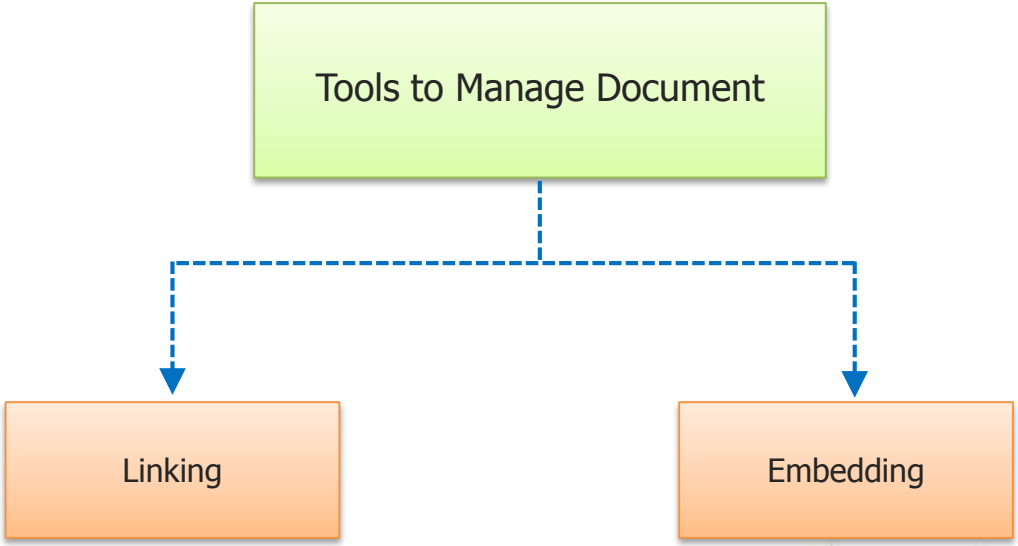
→ Optimize your schema for most frequent use cases.

→ Do complex aggregation in the schema.



Document Structure

→ The key decision in designing data models for MongoDB® applications revolves around the structure of documents and how the application represents relationships between data.



Atomicity of Write Operations

edureka!

- In MongoDB®, write operations are atomic at the document level, and no single write operation can atomically affect more than one document or more than one collection.
- A de-normalized data model with embedded data combines all related data for a represented entity in a single document.
- This facilitates atomic write operations since a single write operation can insert or update the data for an entity.
- Normalizing the data would split the data across multiple collections and would require multiple write operations that are not atomic collectively.
- However, schemas that facilitate atomic writes may limit ways that applications can use the data or may limit ways to modify applications.

Document Growth

edureka!

→ Some updates, such as pushing elements to an array or adding new fields, increase a document's size.

→ If the document size exceeds the allocated space for that document, MongoDB® relocates the document on disk.

→ The growth consideration can affect the decision to normalize or denormalized data.

Data Use and Performance

edureka!

→ When designing a data model, consider how applications will use your database.

→ For instance, if your application only uses recently inserted documents, consider using Capped Collections.

→ Or if your application needs are mainly read operations to a collection, adding indexes to support common queries can improve performance.

Data Modeling Concept

edureka!

→ When constructing a data model for MongoDB® collection, there are various options we can choose from, each of which has its strengths and weaknesses.

→ Key design decisions and detail various considerations for choosing the best data model for your application needs.

Data Model Design

- Presents the different strategies that you can choose from when determining your data model, their strengths and their weaknesses.

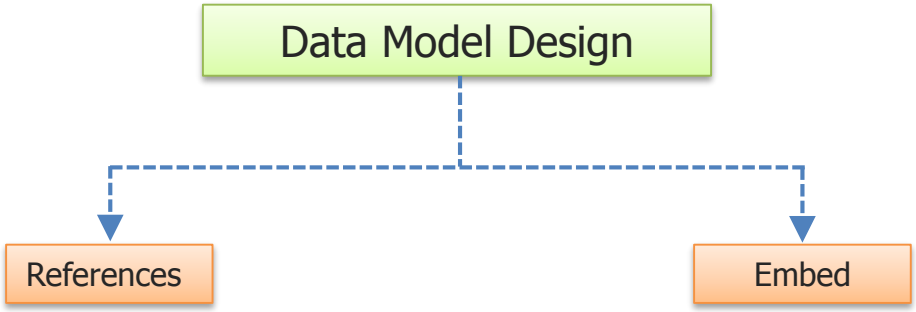
Operational Factors and Data Models

- Details features you should keep in mind when designing your data model, such as lifecycle management, indexing, horizontal scalability, and document growth.

GridFS

- GridFS is a specification for storing documents that exceeds the BSON-document size limit of 16MB.

Data Model Design



Embedding

edureka!

→ Embedded documents capture relationships between data by storing related data in a single document structure.

→ MongoDB® documents make it possible to embed document structures as sub-documents in a field or array within a document.

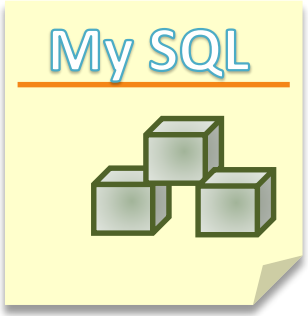
→ These denormalized data models allow applications to retrieve and manipulate related data in a single database operation.

```
_ID:<ObjectID1>,
username:'Shukla',
logintime:'9AM-5PM'
Department:'Training',
contact:  {
          Phone:9739205326,
          email:'narendra@edureka.in',
          ExtNo:2156
          },
access:
          {
          user_id:<ObjectID1>,
          skill:[{DB:'MongoDB',Lng:'.Net'}]
          }
}
```

Embedded sub- document

Embedded sub- document

Embedding (Contd.)



tags

id
name

posts_tags

id
post_id
tag_id

posts

id
title
slug
body
published
created
updated

comments

id
post_id
author
email
body
created

Embedding (Contd.)

posts
id
title
slug
body
published
created
updated

comments
author
email
body
created

tags

Embedding (Contd.)

```
{
“_id”:object ID(“4c03e856e258c271930c091”).
“title”:welcome to MongoDB”,
“slug”:”welcome –to – Mongodb”.
“boby”: “Today, we’re gonna totally rock your world……”,
“publish” : true,
“created” : “Mon May 31 2010 12:48:22 GMT – 0400 (EDT)”,
“updated” : “Mon May 31 2010 12:48:22 GMT – 0400 (EDT)”,
“comments” : [
        {
          “author”:”Bob”,
          “email” : bod@example.com “,
          “body”  :“My mind has been totally blown!”,
          “created”:“Mon May 31 2010 12:48:22 GMT – 0400 (EDT)”,
        }
        ].
“tags”: [
“databases” , “MongoDB” , “awesome”
        ]
}
```

References (Linking)

edureka!

→ References store the relationships between data by including links or references from one document to another.

→ Applications can resolve these references to access the related data.

→ Broadly, these are normalized data models.

→ In the picture data model, references are used to link documents.

References (Linking)

→ Both the contact document and the access document contain a reference to the user document.

User Document
{
_ID:<ObjectID1>,
username:'Shukla',
logintime:'9AM-5PM'
Department:'Training'
}

Contact Document
{
_ID:<ObjectID2>,
Userid:<ObjectID1>,
Phone:9739205326,
email:'narendra@edureka.in',
ExtNo:2156
}

Access Document
{
_ID:<ObjectID3>,
user_id:<ObjectID1>,
skill:[{DB:'MongoDB ',Lng:'.Net'}]
}

Annie's Question

Which kind of schema MongoDB® supports?

a. Static
b. Dynamic

Annie's Answer

Dynamic Schema

Annie's Question

Does MongoDB® supports Atomic Operation?

Annie's Answer

Yes, but at the single document. No single write operation can change more than one document.

Annie's Question

What is difference in embedding & linking?

Annie's Answer

Embedding → Nesting one document with other.
Linking → Referencing one or more document to get desire output.

Operational Factors and Data Models

GridFS

edureka!

GridFS is a specification for storing and retrieving files that exceed the BSON-document size limit of 16MB.

Instead of storing a file in a single document, GridFS divides a file into parts, or chunks and stores each of those chunks as a separate document.

By default GridFS limits chunk size to 256k. GridFS uses two collections to store files. One collection stores the file chunks, and the other stores file metadata.

Annie's Question

How much size is required for each Index in MongoDB® ?

Annie's Answer

Each index requires at least 8 KB of data space.

Annie's Question

What is GridFS in MongoDB® ?

Annie's Answer

GridFS is a specification for storing and retrieving files that exceed the BSON-document size limit of 16MB.

Annie's Question

In which collections GridFS stores information in MongoDB® ?

Annie's Answer

GridFS stores files in two collections:

1. chunks stores the binary chunks.
2. files stores the file's metadata.

Data Model Examples and Patterns

edureka!

→ Model Relationships Between Documents
- » Model One-to-One Relationships with Embedded Documents
- » Model One-to-Many Relationships with Embedded Documents
- » Model One-to-Many Relationships with Document References

→ Model Tree Structures
- » Model Tree Structures with Parent References
- » Model Tree Structures with Child References
- » Model Tree Structures with an Array of Ancestors
- » Model Tree Structures with Materialized Paths
- » Model Tree Structures with Nested Sets

→ Model Specific Application Contexts
- » Model Data for Atomic Operations
- » Model Data to Support Keyword Search

1–to–m Relationships with Embedded Documents

edureka!

- If the contact data is frequently retrieved with the trainer_id information, then with referencing, your application needs to issue multiple queries to resolve the reference.
- The better data model would be to embed the trainer_contact document in the trainer_id document, as in the following document.
- With the embedded data model, your application can retrieve the complete contact information with one query.

Trainer_id Document

```
{
_id:trainer_id,
username:'Shukla',
logintime:'9AM-5PM'
Department:'Training'
}
```

Trainer_Contact Document

```
{
address_id:trainer_id,
Location:[{City:'Bangalore',Place:'Kormangala'}]
Phone:9739205326,
email:'narendra@edureka.in',
ExtNo:2156
}
```

Embedding Documents

New_Collection_Name

```
{
_ID:trainer_id,
username:'Shukla',
logintime:'9AM-5PM'
Department:'Training'
}
{
Address:[
{
Location:[{City:'Bangalore',Place:'Kormangala'}]
Phone:9739205326,
email:'narendra@edureka.in',
ExtNo:2156
}
```

1–to–m Relationships with Embedded Documents

edureka!

- If the contact data is frequently retrieved with the trainer_id information, then with referencing, your application needs to issue multiple queries to resolve the reference.
- The better data model would be to embed the trainer_contact document in the trainer_id document, as in the following document.
- With the embedded data model, your application can retrieve the complete contact information with one query.

Trainer_id Document

```
{
_id:trainer_id,
username:'Shukla',
logintime:'9AM-5PM'
Department:'Training'
}
```

Trainer_Contact Document

```
{
address_id:trainer_id,
Location:[{City:'Bangalore',Place:'Kormangala'}]
Phone:9739205326,
email:'narendra@edureka.in',
ExtNo:2156
}
```

Trainer_Contact Document

```
{
address_id:trainer_id,
Location:[{City:'Hyderabad',Place:'Secundrabad'}]
Phone:8050126646,
email:'narendra@gmail.com',
ExtNo:7325
}
```

Embedding Documents

New_Collection_Name

```
{
_ID:trainer_id,
username:'Shukla',
logintime:'9AM-5PM'
Department:'Training'
}
{
Address:[
{
Location:[{City:'Bangalore',Place:'Kormangala'}]
Phone:9739205326,
email:'narendra@edureka.in',
ExtNo:2156
},
{
Location:[{City:'Hyderabad',Place:'Secundrabad'}]
Phone:8050126646,
email:'narendra@gmail.com',
ExtNo:7325
}
}
```

Embedded Documents

1–to–m Relationships with Reference Documents

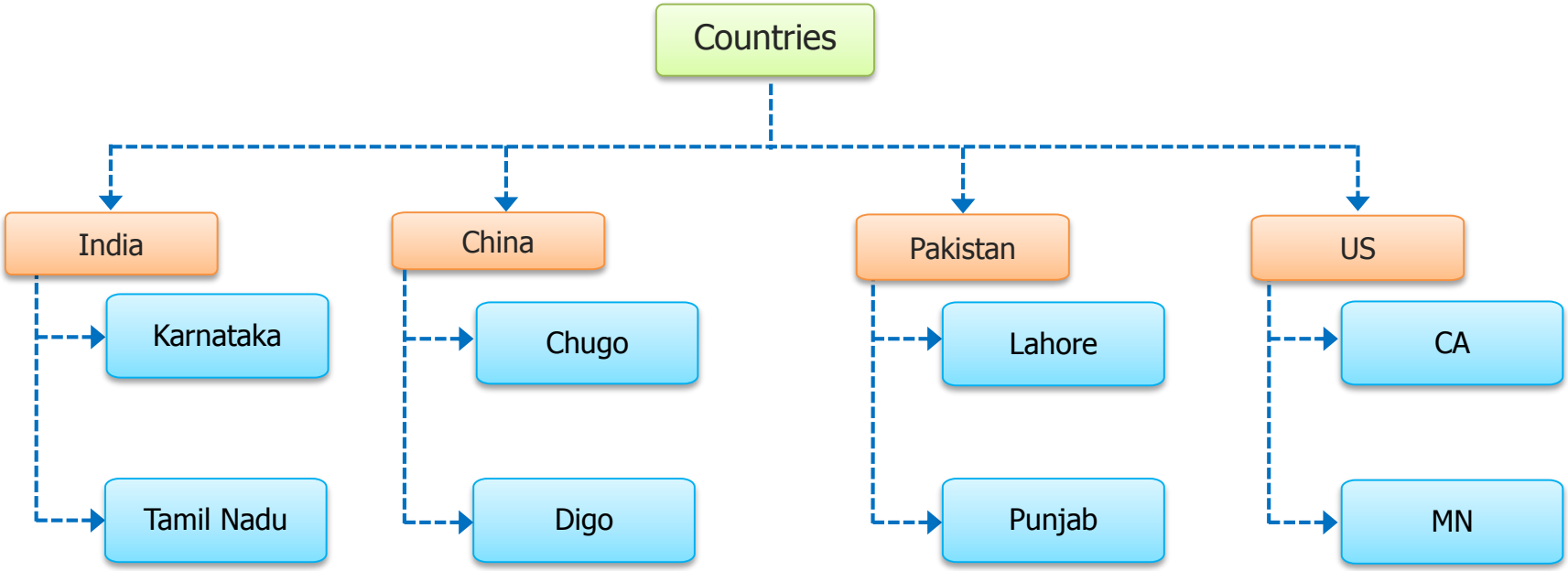
edureka!

- → The example illustrates the advantage of referencing over embedding to avoid repetition of the Producer information
- → Embedding the Producer document inside the Movie document would lead to repetition of the Producer data, as shown in documents.
- → To avoid repetition of the Producer data, use references and keep the Producer information in a separate collection from the Movie collection.

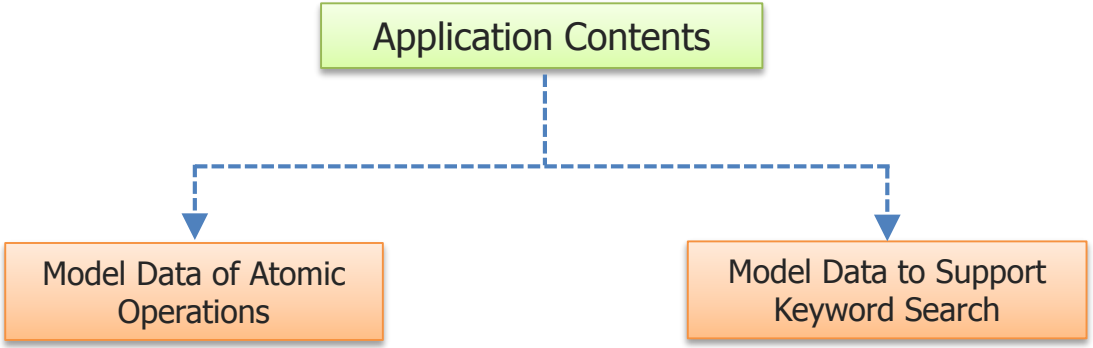
```
{
{
  Movie_Name title:    "Dhoom3",
  Actor:    author: ["Amir Khan", "Salman Khan"],
  Release_Date:        ISODate("2014-0-24"),
  language:            "Hindi",
  Producer:            {
            Director_Name:  "Shukla",
            Start_Year: 2012,
             location:   "India"
             }
}

{
  Movie_Name:          "The proposal",
  Actor:    "Wilson Jacky",
  Release_Date :       ISODate("2013-05-06"),
  language:            "English",
  Producer:            {
            Director_Name:  "Shukla",
            Start_Year: 2012,
             location:   "India"
            }
}
}
```

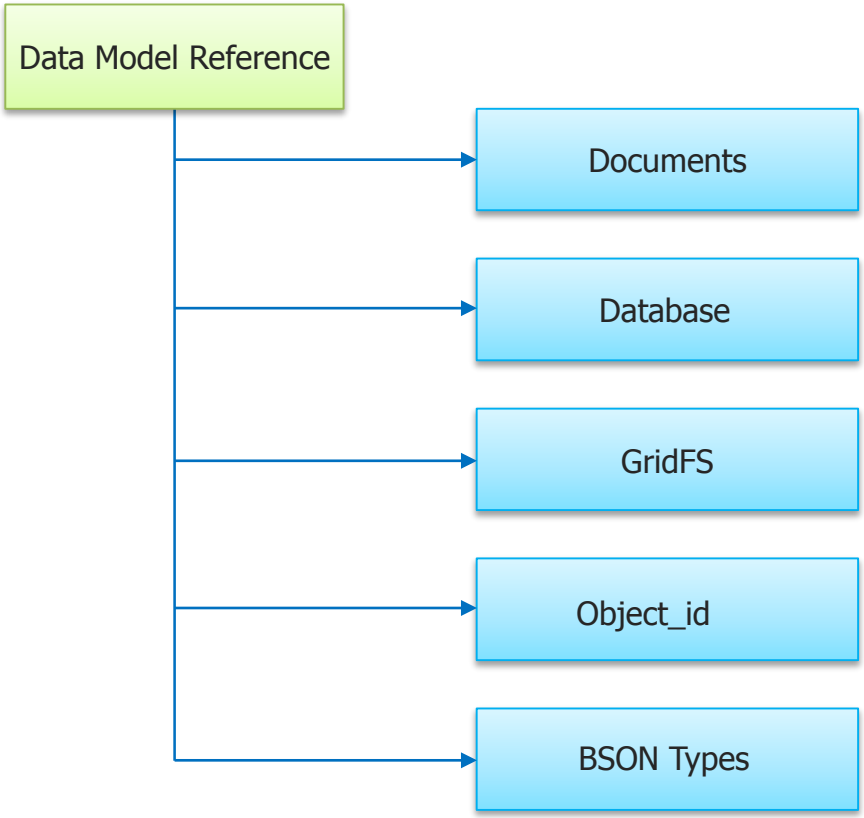
Model Tree Structure



Model Specific Application Contexts



Data Model Reference



Manual Reference and DBRef

edureka!

Manual Reference

```
original_id = ObjectId()

db.places.insert({
        "_id": original_id,
        "name": "Broadway Center",
        "url": "bc.example.net"
})

db.people.insert({
        "name": "Erin",
        "places_id": original_id,
        "url": "bc.example.net/Erin"
})
```

When a query returns the document from the people collection you can, if needed, make a second query for the document referenced by the **places_id** field in the places collection.

DBRef

```
{
        "_id" : ObjectId("5126bbf64aed4daf9e2ab771"),
        "name": "Broadway Center",
        "url": "bc.example.net"
        "creator" : {
                "$ref" : "creators",
                "$id" :
ObjectId("5126bc054aed4daf9e2ab772"),
                "$db" : "users"
        }
}
```

The **DBRef** in this example points to a document in the creators collection of the users database that has **ObjectId("5126bc054aed4daf9e2ab772")** in its **_id** field.

Annie's Question

What all the limitations are there for Documents in MongoDB®?

Annie's Answer

1. The maximum BSON document size is 16 megabytes.
2. The field names cannot start with the $ character.
3. The field names cannot contain the . character.
4. The field name _id is reserved for use as a primary key; its value must be unique.

Annie's Question

What is DBRefs in MongoDB®?

Annie's Answer

DBRefs are references from one document to another using the value of the first document's _id field collection, and optional database name.

Data Modeling Example in RDBMS and MongoDB®

Suppose a client needs database design for his blog website with below features, then what could be difference between RDBMS and MongoDB® schema design approach?

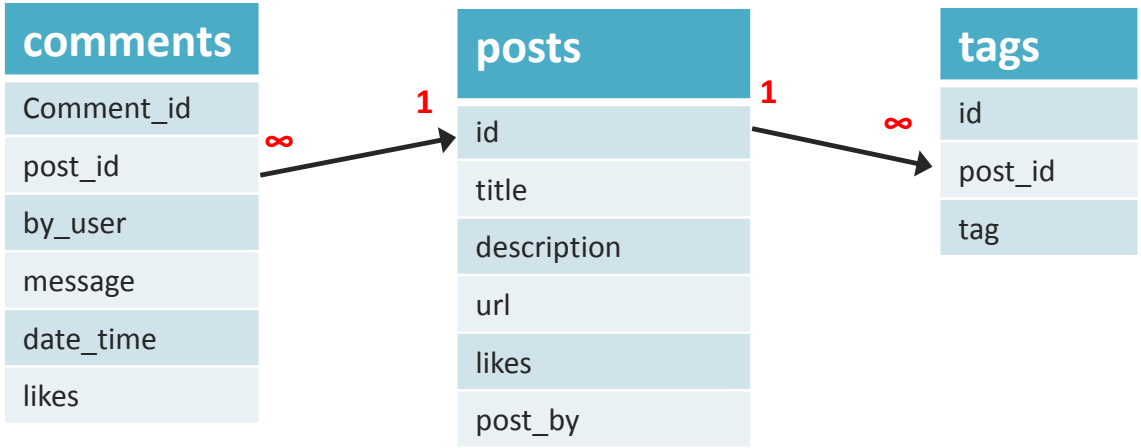
Website has the following requirements.

→ Every post has the unique title, description and url.
→ Every post can have one or more tags.
→ Every post has the name of its publisher and total number of likes.
→ Every post have comments given by users along with their name, message, data-time and likes.
→ On each post there can be zero or more comments.

Example in RDBMS

edureka!

→ In RDBMS schema design for above requirements will have minimum 3 tables.



Example in MongoDB®

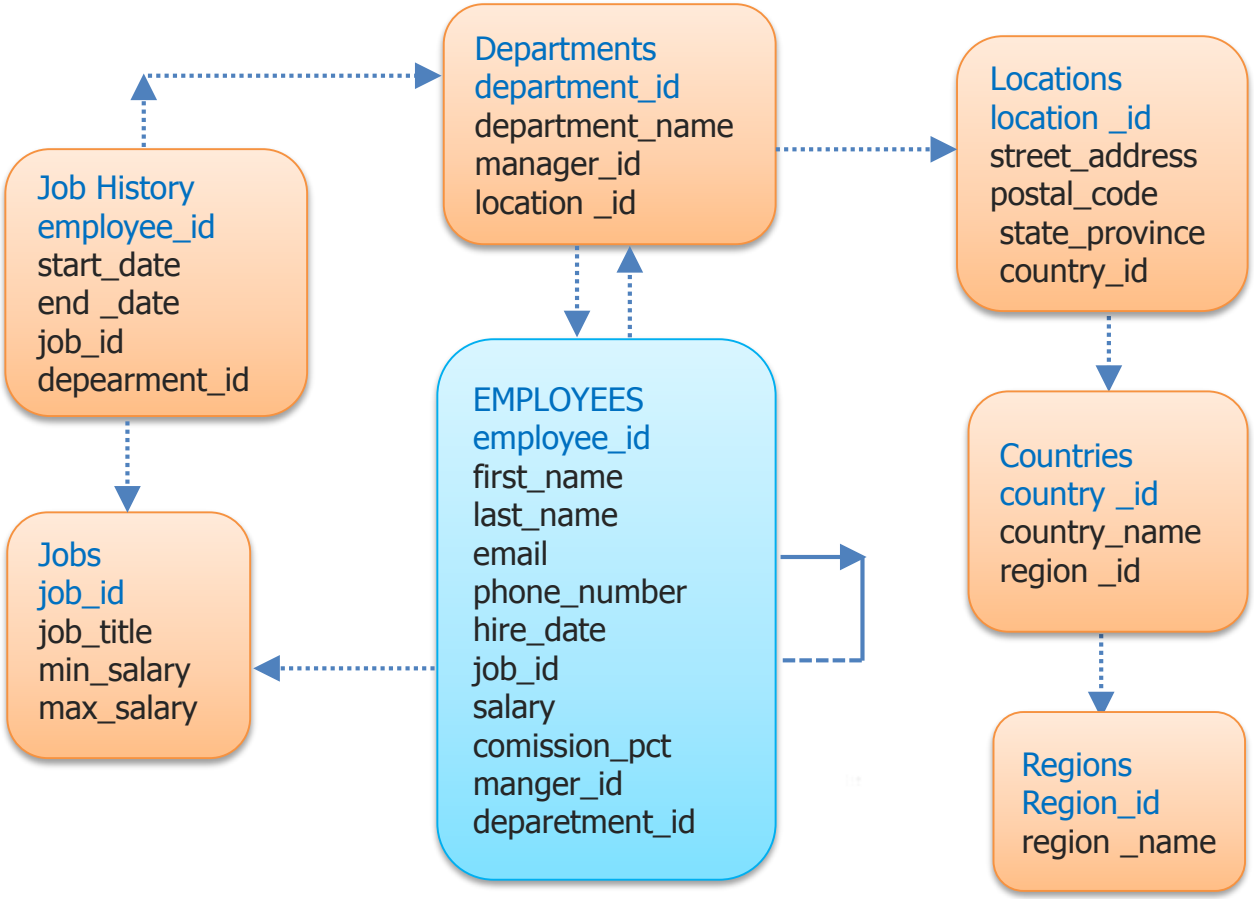
edureka!

→ While in MongoDB® schema design will have one collection post and has the following structure.

→ So while showing the data, in RDBMS we need to join three tables and in MongoDB® data will be shown from one collection only.

```
{
  _id:        POST_ID
  title:      TITLE_OF_POST,
  description:            POST_DESCRIPTION,
  by:         POST_BY,
  url:        URL_OF_POST,
  tags: [     TAG1, TAG2, TAG3],
  likes:      TOTAL_LIKES,
  comments: [
              {
                    user:'COMMENT_BY',
                    message: TEXT,
                    dateCreated: DATE_TIME,
                    like: LIKES
               },
              {
                    user:'COMMENT_BY',
                    message: TEXT,
                    dateCreated: DATE_TIME,
                    like: LIKES
              }
                ]
}
```

HR Data Modeling in RDBMS



HR Data Modeling in RDBMS (Contd.)

edureka!

Demo

Annie's Question

How to design a CMS which will have posts?
Post will have authors and I would like to support commenting & voting, and also would like to tag posts for searching.

Annie's Answer

I will show you live demo for the above case

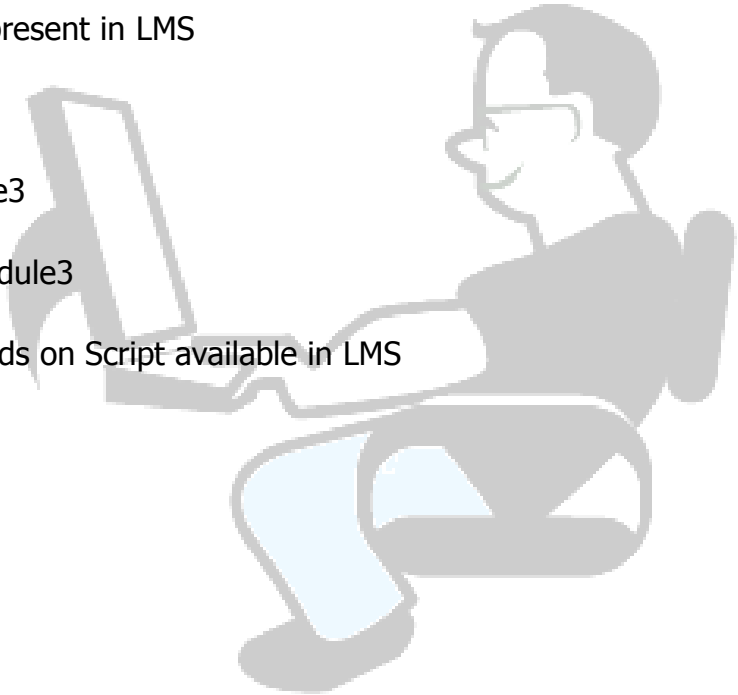
Assignment

edureka!

- → What do you mean by flexible schema in MongoDB® ?
- → How MongoDB® handles Atomicity?
- → How does MongoDB® handles Document Growth?
- → What is embedding & linking in MongoDB®?
- → How MongoDB® handles normalized data?
- → What is constrain for Atomicity in MongoDB®?
- → What is _id field in MongoDB®?
- → What is sharding in MongoDB®?
- → What is GridFS in MongoDB® ?
- → What all limitations are there with Documents in MongoDB® ?

Pre-work

- Design HR Schema on MongoDB® Database
- Execute all Module3 Script present in LMS
- Read FAQ Module3 in LMS
- Take Quiz in LMS for Module3
- Complete assignment of Module3
- Try out Schema Design Hands on Script available in LMS



Agenda for Next Class

edureka!

- → Administration Concepts
- → Operational Strategies
- → Data Management
- → Optimization Strategies for MongoDB®
- → Administration Tutorials
- → Administration Reference

Survey

Your feedback is important to us, be it a compliment, a suggestion or a complaint. It helps us to make the course better!

Please spare few seconds to take the survey after the webinar.

edureka!