Enterprise Z is developing a content management system that will handle a variety of material kinds such as “articles, photos, and videos”.

• Creating an effective data model to manage a variety of content kinds.

• Improving query performance for faster content retrieval.

• Ensuring data consistency across several content kinds.

• Adapting to changes in content structure.

Solution:

1. MongoDB Non-Relational Database Option: Use a document-based model with embedded or linked content.

**Considerations:**

1. **Indexes:**
   * Employ indexes judiciously to optimize query performance.
2. **Syntax:**
   * For MySQL, use SQL; for MongoDB, utilize its versatile JSON-like syntax.

Enterprise Z's goal of developing a content management system (CMS) that handles various types of content like articles, photos, and videos involves key considerations in designing the system.

**1. Data Model:**

* **Option:** MongoDB Non-Relational Database
* **Model:** Utilize a document-based approach.
* **Implementation:** Embed or link content within documents.

**2. Query Performance:**

* **Indexing:** Implement appropriate indexes to enhance query speed.

**3. Data Consistency:**

* **Approach:** Leverage ACID transactions for relational databases.
* **NoSQL Consistency:** In NoSQL (like MongoDB), follow eventual consistency principles.

**4. Adaptability to Changes:**

* **Flexibility:** MongoDB's JSON-like syntax allows easy adaptation to changes in content structure.

**Transaction Handling:**

* **Relational (ACID):** Ensures data consistency through Atomicity, Consistency, Isolation, and Durability.
* **NoSQL (MongoDB):** Adheres to eventual consistency, allowing flexibility at the cost of immediate consistency.

By employing MongoDB as a non-relational database, utilizing effective indexing strategies, and considering the syntax nuances, Enterprise Z can build a CMS that efficiently manages diverse content types, ensures data consistency, and adapts to evolving content structures.

Here I’ve Implemented with both mongosh and mongodb Compass for this project.

In mongoshell create content\_management\_system with following command:

use content\_management\_system; /// Creates a database with above name

Since we’re making a content management system let us consider 3 types of collections in our CMS which are articles, photos and videos with following command:

db.createCollection("articles");

db.createCollection("photos");

db.createCollection("videos");

**Define document structure**

**Articles collection:**

{

\_id: ObjectId("654f71ee466af9c091a04cd6"),

title: 'Intro to Java',

content: 'New content has been updated',

author: 'Joshua',

tags: [ 'hello-world', 'Data\_types' ],

published\_date: ISODate("2023-11-10T00:00:00.000Z"),

photo\_id: ObjectId("654f72d1466af9c091a04cda"),

video\_id: ObjectId("654f731d466af9c091a04cdc")

},

{

\_id: ObjectId("654f8f67466af9c091a04cde"),

title: 'Intro to MongoDB',

content: 'MongoDB is used to store and manage Data',

author: 'Christan',

tags: [ 'compass', 'mongosh' ],

published\_date: ISODate("2023-11-11T00:00:00.000Z"),

photo\_id: ObjectId("654f8f88466af9c091a04ce0"),

video\_id: ObjectId("654f8fe1466af9c091a04ce5")

},

{

\_id: ObjectId("654f8fc8466af9c091a04ce3"),

title: 'Intro to AWS',

content: 'Aws is a cloud service provider!',

author: 'Dan lee',

tags: [ 'ec2', 'IAM' ],

published\_date: ISODate("2023-11-12T00:00:00.000Z"),

photo\_id: ObjectId("654f8fa0466af9c091a04ce1"),

video\_id: ObjectId("654f8ff8466af9c091a04ce6")

}

Created 3 articles like above with json syntax with phot\_id and video\_id for linked content as a reference between document’s

**Photos Collection:**

{

\_id: ObjectId("654f72d1466af9c091a04cda"),

title: 'New Wallpapers',

image\_url: 'https://example.com/sunset.jpg',

photographer: 'Jordan willey',

published\_date: ISODate("2023-11-01T00:00:00.000Z")

},

{

\_id: ObjectId("654f8f88466af9c091a04ce0"),

title: 'Desktop Wallpapers',

image\_url: 'https://example.com/desktopwallpaper1.jpg',

photographer: 'Michel Phillip',

published\_date: ISODate("2023-11-02T00:00:00.000Z")

},

{

\_id: ObjectId("654f8fa0466af9c091a04ce1"),

title: 'Lockscreen Wallpapers',

image\_url: 'https://example.com/lockscreenwall.jpg',

photographer: 'windows 11',

published\_date: ISODate("2023-11-04T00:00:00.000Z")

}

Above are 3 photos collection

**Videos Collection:**

{

\_id: ObjectId("654f731d466af9c091a04cdc"),

title: 'Installing java Step by Step',

video\_url: 'https:example.com/java\_installation.mp4',

uploaded\_by: 'Joshua',

published\_date: ISODate("2023-11-05T00:00:00.000Z")

},

{

\_id: ObjectId("654f8fe1466af9c091a04ce5"),

title: 'Installing MongoDB Step by Step',

video\_url: 'https:example.com/mongodb\_installation.mp4',

uploaded\_by: 'Jonh',

published\_date: ISODate("2023-11-06T00:00:00.000Z")

},

{

\_id: ObjectId("654f8ff8466af9c091a04ce6"),

title: 'Installing AWS CLI Step by Step',

video\_url: 'https:example.com/awscli\_installation.mp4',

uploaded\_by: 'Denny kole',

published\_date: ISODate("2023-11-07T00:00:00.000Z")

}

Above are 3 video collections

**Indexing for Query performance**

let’s create index for faster queries using following commands in mongosh and we can cross verify created index under mongodb compass

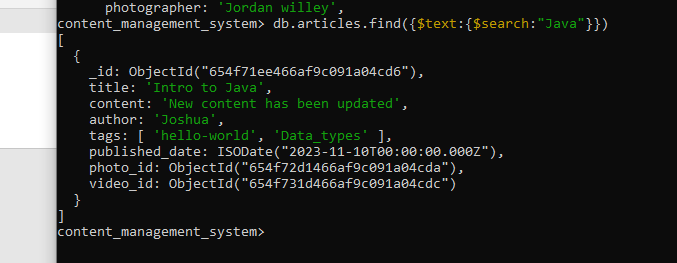
db.articles.createIndex({ title: "text", content: "text" });

db.photos.createIndex({ title: "text" });

db.videos.createIndex({ title: "text" });

Now let’s Implement some sample Index queries

**1.Text Search Query**



Assuming we have index on” title” and “content” fields of “articles” collection.

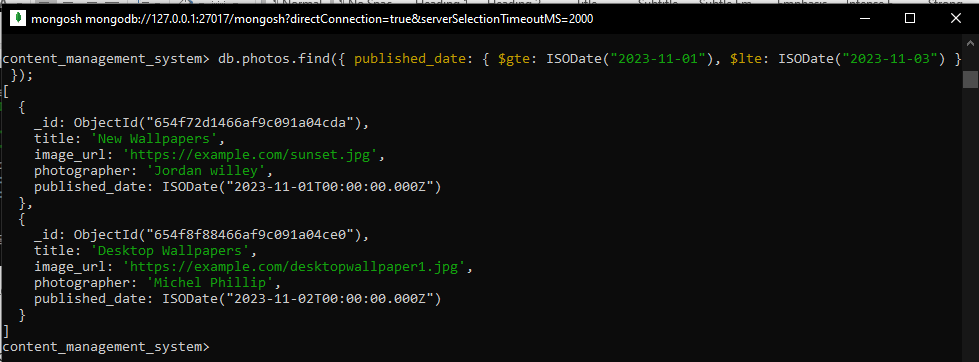
**2.Equality Query:**

If you have an index on the author field of the articles collection:



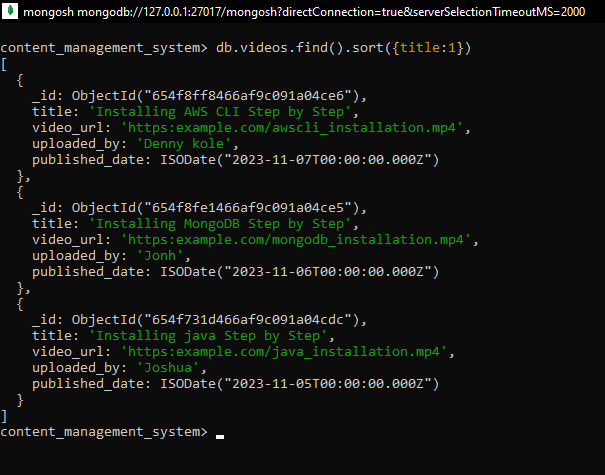
**3. Range Query:**

Assuming an index on the published\_date field of the photos collection:



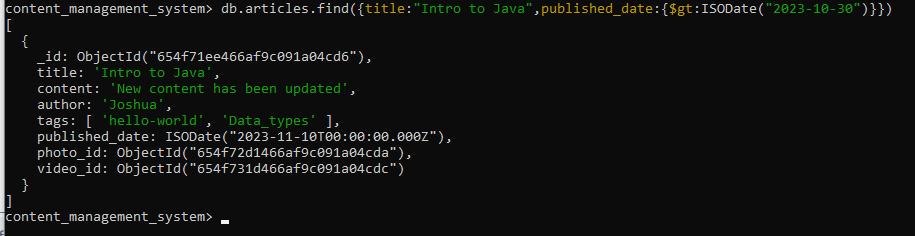
**4. Sorting with Index:**

If there's an index on the title field of the videos collection:



**5. Combining Multiple Fields:**

If you have a compound index on the category and price fields of the products collection:



**6. Covered Query:**

Assuming an index on the tags field of the articles collection:

content\_management\_system> db.articles.find({$text:{$search:"Java"}},{\_id:0,tags:1}).explain("executionStats")

A covered query is one where all the fields needed are in the index, and MongoDB can fulfill the query without examining the actual documents. When a query is covered, MongoDB can satisfy the query by only accessing the index, avoiding the need to load the entire document into memory. This is particularly beneficial for read-heavy workloads where the goal is to retrieve specific data without loading unnecessary information.

Linked content Queries: We have added links to each document in article as photo\_id and video\_id with corresponding Photos and videos collection

db.articles.aggregate([

{

$lookup: {

from: "photos",

localField: "photo\_id",

foreignField: "\_id",

as: "photo"

}

},

{

$lookup: {

from: "videos",

localField: "video\_id",

foreignField: "\_id",

as: "video"

}

},

{

$project: {

title: 1,

content: 1,

author: 1,

published\_date: 1,

photo: { $arrayElemAt: ["$photo", 0] },

video: { $arrayElemAt: ["$video", 0] }

}

}

]);

The above script will return all the articles with linked photo and video information.

This is output for above linked query:

[

{

\_id: ObjectId("654f71ee466af9c091a04cd6"),

title: 'Intro to Java',

content: 'New content has been updated',

author: 'Joshua',

published\_date: ISODate("2023-11-10T00:00:00.000Z"),

photo: {

\_id: ObjectId("654f72d1466af9c091a04cda"),

title: 'New Wallpapers',

image\_url: 'https://example.com/sunset.jpg',

photographer: 'Jordan willey',

published\_date: ISODate("2023-11-01T00:00:00.000Z")

},

video: {

\_id: ObjectId("654f731d466af9c091a04cdc"),

title: 'Installing java Step by Step',

video\_url: 'https:example.com/java\_installation.mp4',

uploaded\_by: 'Joshua',

published\_date: ISODate("2023-11-05T00:00:00.000Z")

}

},

{

\_id: ObjectId("654f8f67466af9c091a04cde"),

title: 'Intro to MongoDB',

content: 'MongoDB is used to store and manage Data',

author: 'Christan',

published\_date: ISODate("2023-11-11T00:00:00.000Z"),

photo: {

\_id: ObjectId("654f8f88466af9c091a04ce0"),

title: 'Desktop Wallpapers',

image\_url: 'https://example.com/desktopwallpaper1.jpg',

photographer: 'Michel Phillip',

published\_date: ISODate("2023-11-02T00:00:00.000Z")

},

video: {

\_id: ObjectId("654f8fe1466af9c091a04ce5"),

title: 'Installing MongoDB Step by Step',

video\_url: 'https:example.com/mongodb\_installation.mp4',

uploaded\_by: 'Jonh',

published\_date: ISODate("2023-11-06T00:00:00.000Z")

}

},

{

\_id: ObjectId("654f8fc8466af9c091a04ce3"),

title: 'Intro to AWS',

content: 'Aws is a cloud service provider!',

author: 'Dan lee',

published\_date: ISODate("2023-11-12T00:00:00.000Z"),

photo: {

\_id: ObjectId("654f8fa0466af9c091a04ce1"),

title: 'Lockscreen Wallpapers',

image\_url: 'https://example.com/lockscreenwall.jpg',

photographer: 'windows 11',

published\_date: ISODate("2023-11-04T00:00:00.000Z")

},

video: {

\_id: ObjectId("654f8ff8466af9c091a04ce6"),

title: 'Installing AWS CLI Step by Step',

video\_url: 'https:example.com/awscli\_installation.mp4',

uploaded\_by: 'Denny kole',

published\_date: ISODate("2023-11-07T00:00:00.000Z")

}

}

]

And we can also update and delete using following commands

db.articles.update({\_id:ObjectId(“our article id goes here ”)},{$set:{content:” content we want to update here”}}) /// **to update data**

db.articles.deleteOne({\_id:ObjectId(“object if we want to delete goes here”)}) ///**delete an article**

**Conclusion:**

This outlines the setup of a MongoDB-based content management system, focusing on effective data modeling, query performance, data consistency, and adaptability to changes.

**Submitted by**

**Shiva Kumar Gorasa**

**shivagor@amazon.com**