**ENTERPRISE - Z**

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:** Select appropriate indexes to improve query performance.

2. **Syntax:** SQL for MySQL, and MongoDB uses versatile JSON-like syntax.

ACID transactions in relational databases; eventual consistency in NoSQL databases.

Enterprise Z is developing a CMS that will handle a variety of material kinds such as articles, photos and videos which involves key considerations in designing the system.

* **Data Model:** MongoDB is a Non-Relational document-based Database. Which can be implemented with embedded or reference content.
* **Query Performance:** Query performance can be enhanced by implementing appropriate indexes.
* **Data Consistency:** MongoDB follow eventual consistency, and partition tolerance principle from CAP Theorem.
* **Adaptability:** MongoDB’s JSON like syntax allows easy adaptability to change in content structure.

Implementing the project using mongosh.

In mongosh create the database using below mentioned command.

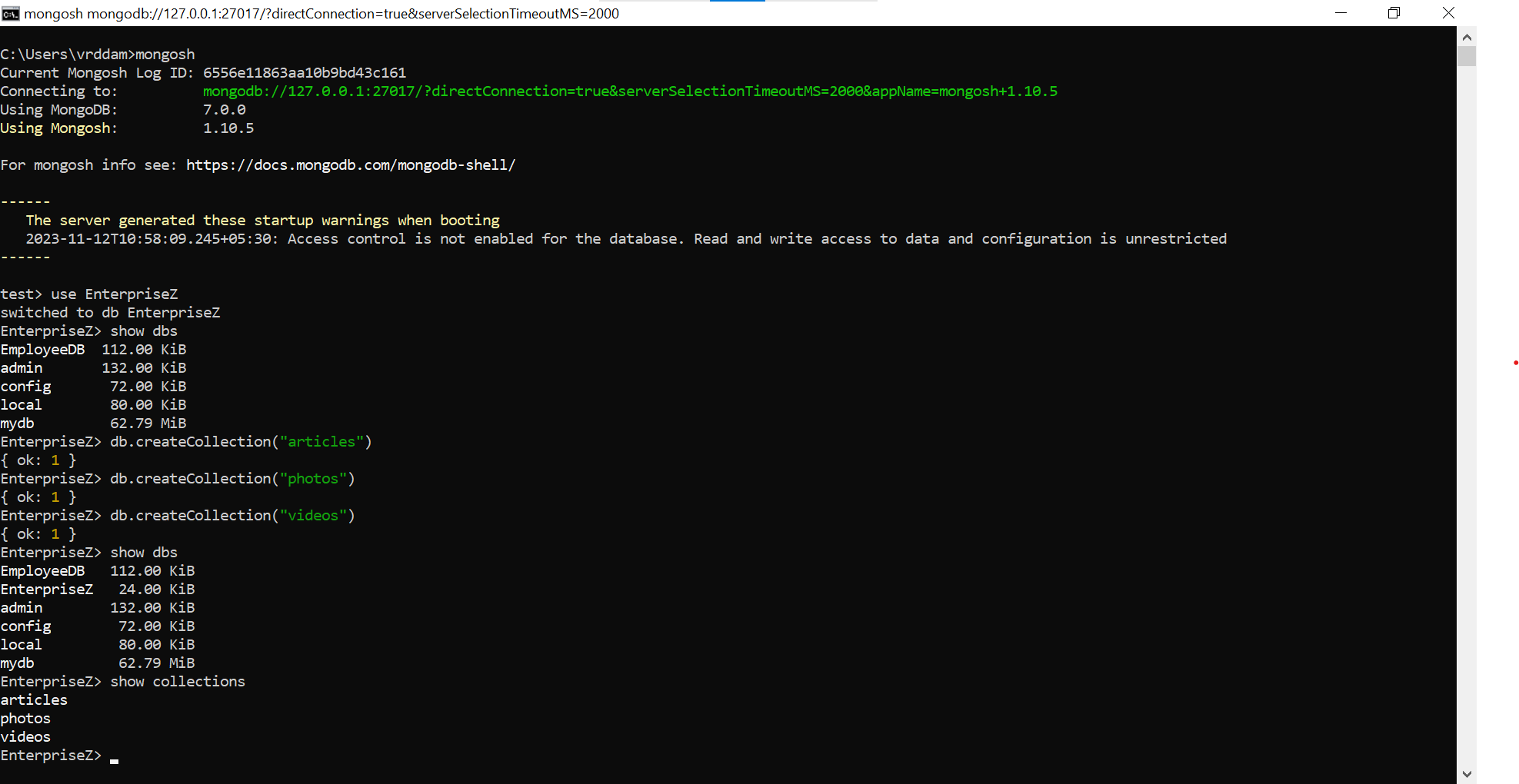
use EnterpriseZ;

Since we are using CMS let’s create three types of collections.

db.createCollection(“articles”);

db.createCollection(“photos”);

db.createCollection(“videos”);



**Define document structure:**

Photos Collection:

db.photos.insertMany([

{"title":"java", "image\_url":"C:\java.jpg","photographer":"abc"},

{"title":"python", "image\_url":"C:\python.jpg", "photographer":"def"},

{"title":"javascript", "image\_url":"C:\javascript.jpg", "photographer":"ghi"}

]);

Created three dummy photo documents.

Videos Collection:

db.videos.insertMany([

{"title":"Basics of java", "image\_url":"C:\coreJava.mp4","photographer":"xxx"},

{"title":"Basics of python", "image\_url":"C:\corePython.mp4", "photographer":"yyy"},

{"title":"Basics of javascript", "image\_url":"C:\coreJavasSript.mp4", "photographer":"zzz"}

]);

Created three dummy video documents.

Article Collection:

db. articles.insertMany([

{"title":"Core Java", "author":"AAA","level":5, "photo\_id":ObjectId("6556edc963aa10b9bd43c17f"), "video\_id":ObjectId("6556ea1763aa10b9bd43c173")},

{"title":"Core Python", "author":"BBB", "level":2, "photo\_id":ObjectId("6556edc963aa10b9bd43c180"), "video\_id":ObjectId("6556ea1763aa10b9bd43c174")},

{"title":"Core JavaScript", "author":"CCC", "level":3, "photo\_id":ObjectId("6556edc963aa10b9bd43c181"), "video\_id":ObjectId("6556ea1763aa10b9bd43c175")}

]);

Created three dummy article documents with photo\_id and video\_id for linking the content.







**Enhancing Query performance using Index:**

Let’s create index for fastening query performance and we can validate them using mongosh.

Below is the command to create ascending order index.

db.articles.createIndex({"level":1});

Below is the command to create descending order index.

db.articles.createIndex({"level":-1});





1) Filtered Query:



Above command will fetch all the documents which matches given filter criteria.

2) Range Query:



Above command will fetch all the documents range between 2 and 4.

3) Sort Query:

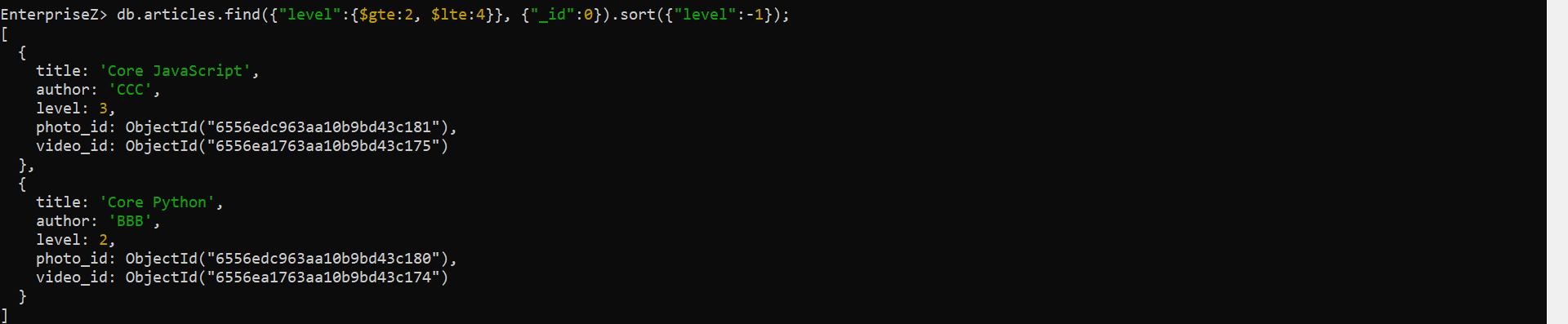


Above command will sort the documents in ascending order.



Above command will sort the documents in descending order.

4) Combined Query:



Linked Content Query:

db.articles.aggregate([

{$lookup:{from:"photos", localField:"photo\_id",foreignField:"\_id",as:"photo"}},

{$lookup:{from:"videos", localField:"video\_id",foreignField:"\_id",as:"video"}},

{$project:{title:1, author:1, level:1, image\_url:"$photo.image\_url", video\_url:"$video.image\_url"}}

]);



Above mentioned is the output for link content query.

Below are the following commands to drop the collection and database.

db.photos.drop();

Command Is used to drop a particular collection.

db.dropDatabase();

Command is used to drop entire database.

**Submitted By:**

**Rohith Reddy**

**vrddam@amazon.com**