Design a Database for a Product Catalog

Estimated duration: 30 mins

Learning Objectives

After completing this activity, you will be able to:

- Identify entities required for your database
- Define the attributes of an entity
- Define the relationship between two or more entities
- Design the schema for MongoDB

In this hands-on activity, you are going to design a product catalog database and develop it. This activity involves the Logical DB creation identifying and defining the schema. You can use a physical notebook and pen or use Word or other text editors for this activity. In real-life projects often specific tools for database schema design are used. In succeeding labs, you will create the database in MongoDB.

Step 1

Identify the different *objects* or *entities*, as they are referred to in data modelling. These entities are what you need to take into consideration to create a product catalog. You can draw inspiration from real-life product catalogs that you may see in your daily lives.

- You have `Catalog` which by itself is an entity
- The catalog will have one or more categories of products. `Category` will be one entity.
- In each category you can have one or more products. `Product` will be one entity.

The entities you will use to design the database for the product catalog are

1. Catalog
2. Category
3. Product

Databases usually have many entities. But for simplicity in this activity, we will only use these three.

Unlike Relational Database Management Systems, MongoDB doesn't follow any formal process, rules, or algorithms. The most important thing that matters is you design a database that works well for your application.

Step 2

Identify the information that you need to store about each entity. These are the *attributes* of each entity.

For each entity you need:

- Entity name
- The attributes in the entity
- The data type of each of the attributes (string, int, list, etc.)
- Description of the attribute
- Whether this attribute is required
- Whether this attribute is unique

The schema for `Catalog` entity can be represented in a table format as below:

Attribute name	Data Type	Description	Required	Unique
_id	String	unique identifier	true	true
name	String	name of the catalog	true	true

If you don't specify `_id`, an ObjectId is automatically generated and allocated. All the entities in MongoDB will have an `_id` attribute which can be a string or a number. This will be unique so the specific entity can be easily identified.

The data in MongoDB is defined in JSON format. The `type` is the only mandatory attribute. The rest of the attributes can be included as per need. The `description` attribute is good to have for readability.

The schema or the definition for the `Catalog` entity will be as follows:

```
{
  _id: {
    type: String,
    required: [true],
    description: "unique identifier",
    unique: true
  },
  name: {
    type: String,
    required: [true, 'The catalog must have a unique name'],
    description: "name of the catalog",
    unique: true
  }
}
```

Sample `Catalog` data can be represented in JSON format, as below.

```
{
  "_id":"clog1",
  "name":"product catalog"
}
```

3. Now take into consideration the `Category` entity. It will have `_id` attribute, it will have a `name`. Each category might have sub-categories. This will be a list of ids, each of which uniquely identifies a category. In the same way, if the category you are defining can be a sub-category to one or more parent categories. For example, `iPhone 14` can be listed under `smart phones` category and `Apple products` category.

The schema for `Category` entity can be represented in a table format as below:

Attribute name	Data Type	Description	Required	Unique
_id	String	unique identifier	True	True
category_name	String	name of the category	True	True
child_categories	Array	list of ids of child categories	False	False
parent_categories	Array	list of ids of parent categories	False	False

The schema or the definition for the `Category` entity will be as follows:

```
{
  _id: {
    type: String,
    required: [true],
    description: "unique identifier",
    unique: true
  },
  category_name: {
    type: String,
    required: [true, 'The category must have a unique name'],
    description: "name of the category",
    unique: true
  },
  child_categories: {
    type: Array,
    description: "list of ids of child categories"
  },
  parent_categories: {
    type: Array,
    description: "list of ids of parent categories"
  }
}
```

Sample `Category` data can be represented in JSON format, as shown in the example below.

```
[{
  "_id":"ct001",
  "category_name":"Smart Phone",
  "child_categories":[],
  "parent_categories":[],
},
{
  "_id":"ct002",
  "category_name":"Apple Products",
  "child_categories":[],
  "parent_categories":[],
},
{
  "_id":"ct003",
  "category_name":"Apple Phones",
  "child_categories":[],
  "parent_categories":["ct001","ct002"],
}]
```

4. Now do the same activity for `Product` entity. It will have `_id` attribute, it will have a `product_name`. Each product belongs to one or more categories. This will be a list of ids, each of which uniquely identify a category.

   The product should have a price, availability count, an image of the product, and reviews.

   The schema for `Product` entity can be represented in a table format as below.

Attribute name	Data Type	Description	Required	Unique
_id	String	unique identifier	True	True
product_name	String	name of the product	True	True
categories	Array	list of ids of categories the product belongs to	True	False
price	Number	price at which the product is sold	True	False
available_count	Number	number of pieces avaiable for sale	True	True
image_url	String	url link to the image which represents the product	False	False
reviews	Array	list of reviews for this product	False	False

The schema or the definition for the `Product` entity will be as follows:

```
{
  _id: {
    type: String,
    required: [true],
    description: "unique identifier",
    unique: true
  },
  product_name: {
```

```
      type: String,
      required: [true, 'The product must have a unique name'],
      description: "name of the product",
      unique: true
    },
    categories: {
      type: Array,
      description: "list of ids of categories the product belongs to",
      required: [true, "The product should belong to a category"]
    },
    price: {
      type: Number,
      description: "price at which the product is sold",
      required: [true, 'The product must have a price'],
    },
    available_count: {
      type: Number,
      description: "number of pieces avaiable for sale",
      required: [true, 'The product must have number of items available for sale'],
    },
    image_url:{
      type: String,
      description: "url link to the image which represents the product",
    },
    reviews:{
      type: Array,
      description: "list of reviews for this product",
    }
  }
```

Sample `Product` data can be represented in JSON format, as shown in the example below.

```
[{
  "_id":"pd001",
  "product_name":"iPhone 14",
  "categories":["ct001","ct002","ct003"],
  "price":850.00,
  "available_count":243,
  "image_url":"https://store.storeimages.cdn-apple.com/4982/as-images.apple.com/is/iphone-14-model-unselect-gallery-1-202209",
  "reviews":["It is value for money","It is the best iPhone after iPhone 8","It is a great phone"]
}]
```

You can have a schema validator and further restrict the data that is contained in the database. That is beyond the scope of this course.

Practice task:

Try to design and develop an employee database. Can you identify the entities and their attributes?

Summary:

In this activity, you learned how to design and develop a database for MongoDB.

Author:

Lavanya T S