NAME : Nevetha S K
EMAIL ID : nevethasenthilkumar@gmail.com

CODING ASSESSMENT – UNDERSTANDING RELATIONS IN MONGODB

In traditional RDBMS, relationships are maintained via **foreign keys**. In MongoDB, a **NoSQL database**, relationships are handled via:

- **Embedding (Denormalization)** – Nest related data inside documents.
- **Referencing (Normalization)** – Use ObjectIds to reference other collections.

MongoDB supports three main types of relationships:

- One-to-One
- One-to-Many
- Many-to-Many

One-to-One Relationship :

In a one-to-one relationship, one document is related to only one other document. For example, each student can have one unique profile with personal details like address and phone number. In MongoDB, this can be done by either **embedding** the profile directly inside the student document if the data is small and always accessed together, or by **referencing** it in a separate collection using an ID if the profile is large or needs to be accessed separately. Embeddig makes reading faster, while referencing keeps the data more flexible and clean.

Example: A student has one profile

Insert into students collection (embedding) and Find student and profile

```
> use relationdb
< switched to db relationdb
> db.students.insertOne({
    name: "Alice",
    age: 22,
    profile: {
      address: "123 Main Street",
      phone: "9876543210"
    }
  })
< {
    acknowledged: true,
    insertedId: ObjectId('68830ac371f9869c2cc36183')
  }
> db.students.find()
< {
    _id: ObjectId('68830ac371f9869c2cc36183'),
    name: 'Alice',
    age: 22,
    profile: {
      address: '123 Main Street',
      phone: '9876543210'
    }
  }
```

Example: A student has one profile (referencing)

Insert student and profile

```
> db.students.insertOne({
    _id: 1,
    name: "Bob",
    age: 23
  })
< {
    acknowledged: true,
    insertedId: 1
  }
> db.profiles.insertOne({
    student_id: 1,
    address: "456 Park Avenue",
    phone: "9998887770"
  })
< {
    acknowledged: true,
    insertedId: ObjectId('68830c2871f9869c2cc36184')
  }
```

Find student and profile

```
> db.students.find()
< {
    _id: ObjectId('68830ac371f9869c2cc36183'),
    name: 'Alice',
    age: 22,
    profile: {
      address: '123 Main Street',
      phone: '9876543210'
    }
  }
  {
    _id: 1,
    name: 'Bob',
    age: 23
  }
> db.profiles.find({ student_id: 1 })
< {
    _id: ObjectId('68830c2871f9869c2cc36184'),
    student_id: 1,
    address: '456 Park Avenue',
    phone: '9998887770'
  }
```

One-to-Many Relationship :

In a one-to-many relationship, one document is related to many documents. A good example is a student enrolling in multiple courses. You can embed the list of courses directly inside the student document if the list is small and changes rarely. But if the student can have many courses or if the courses are large or updated often, it's better to store them in a separate collection and use the student's ID to connect them. This referencing method is more efficient when the number of related items grows large.

Example: A student has many courses (embedding)

Insert student with course list and Find student and course

```
> db.students.insertOne({
    name: "Carol",
    age: 21,
    courses: [
      { name: "Math", code: "MTH101" },
      { name: "Physics", code: "PHY101" }
    ]
  })
< {
    acknowledged: true,
    insertedId: ObjectId('68830da271f9869c2cc36186')
  }
> db.students.find()
< {
    _id: ObjectId('68830da271f9869c2cc36186'),
    name: 'Carol',
    age: 21,
    courses: [
      {
        name: 'Math',
        code: 'MTH101'
      },
      {
        name: 'Physics',
        code: 'PHY101'
      }
    ]
  }
```

Example: A student has many courses (referencing)

Insert student and Insert enrollments (one-to-many)

```
> db.students.insertOne({
    _id: 2,
    name: "David",
    age: 24
  })
< {
    acknowledged: true,
    insertedId: 2
  }
> db.enrollments.insertMany([
    { student_id: 2, course: "Math", code: "MTH101" },
    { student_id: 2, course: "Physics", code: "PHY101" }
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('68830e3d71f9869c2cc36187'),
      '1': ObjectId('68830e3d71f9869c2cc36188')
    }
  }
```

Find student and student's courses

```
> db.students.find({ _id: 2 })
< {
    _id: 2,
    name: 'David',
    age: 24
  }
> db.enrollments.find({ student_id: 2 })
< {
    _id: ObjectId('68830e3d71f9869c2cc36187'),
    student_id: 2,
    course: 'Math',
    code: 'MTH101'
  }
  {
    _id: ObjectId('68830e3d71f9869c2cc36188'),
    student_id: 2,
    course: 'Physics',
    code: 'PHY101'
  }
```

Many-to-Many Relationship :

A many-to-many relationship means multiple documents are connected to multiple others. For example, many students can take many courses, and each course can have many students. In MongoDB, this is handled by creating a **third collection** (like “enrollments”) that holds references to both the student and the course. This is the only proper way to model many-to-many relationships in MongoDB. You should not embed this kind of relationship because the data would become repetitive and hard to manage.

Example: Students and courses (many-to-many using enrollments)

Insert students and courses

```
> db.students.insertMany([
    { _id: 3, name: "John" },
    { _id: 4, name: "Emma" }
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': 3,
      '1': 4
    }
  }
> db.courses.insertMany([
    { _id: 101, name: "Math" },
    { _id: 102, name: "Physics" }
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': 101,
      '1': 102
    }
  }
```

Insert enrollments (many-to-many)

```
> db.enrollments.insertMany([
    { student_id: 3, course_id: 101 },
    { student_id: 3, course_id: 102 },
    { student_id: 4, course_id: 102 }
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('688313fa71f9869c2cc36189'),
      '1': ObjectId('688313fa71f9869c2cc3618a'),
      '2': ObjectId('688313fa71f9869c2cc3618b')
    }
  }
```

Find enrollments of Emma (student_id = 4)

```
> db.enrollments.find({ student_id: 4 })
< {
    _id: ObjectId('688313fa71f9869c2cc3618b'),
    student_id: 4,
    course_id: 102
  }
```