CECS 323: Database Fundamentals

Project 3: The Mongo Mash

by

Quan Nguyen and Lee Nguyen

12/10/2022

Schemas Created and Modified (films and actors)

Films schema -

```
{
  $jsonSchema: {
    title: 'film',
    description: 'A film shown at theaters.',
    bsonType: 'object',
    required: [
      'title',
      'duration',
      'releaseDate',
      'rating',
      'genres',
      'actors'
    ],
    properties: {
      _id: {
        bsonType: 'objectId'
      },
      title: {
        bsonType: 'string'
      },
      duration: {
        bsonType: 'int',
        minimum: 1
      },
      releaseDate: {
        bsonType: 'date'
      },
      rating: {
        bsonType: 'string',
        'enum': [
          'NR',
          'G',
          'PG',
          'PG-13',
          'R',
          'NC-17'
        ]
      },
      genres: {
        bsonType: 'array',
        minItems: 1,
        items: {
          bsonType: 'string'
        }
      },
      actors: {
        bsonType: 'array',
        items: {
          bsonType: 'object',
          required: [
            'actor_id',
            'actor_name',
```

```
            'actor_sex',
            'actor_birthday'
          ],
          properties: {
            actor_id: {
              bsonType: 'objectId'
            },
            actor_name: {
              bsonType: 'string'
            },
            actor_sex: {
              bsonType: 'string'
            },
            actor_birthday: {
              bsonType: 'date'
            }
          }
        }
      }
    }
  }
}
```

Actors schema -

```
{
  $jsonSchema: {
    title: 'actor',
    description: 'An actor who stars in films.',
    bsonType: 'object',
    required: [
      'firstName',
      'lastName',
      'birthday',
      'imdbUrl'
    ],
    properties: {
      _id: {
        bsonType: 'objectId'
      },
      firstName: {
        bsonType: 'string'
      },
      lastName: {
        bsonType: 'string'
      },
      birthday: {
        bsonType: 'date'
      },
      imdbUrl: {
        bsonType: 'string'
      }
    }
  }
}
```

Justification for Schemas

While adding the new actors schema, the only schema modified was the film collections. A film can have many actors, and an actor can star in many films. In this enterprise, an actor object is not likely to be needed on its own very often, as customers are generally interested in purchasing tickets for specific films rather than looking up actors. Therefore, it would be sufficient to store just a few details for each actor in the film document, such as their name, sex, birthday, and reference to their full actor document (actor_id). This would be an example of denormalization and redundancy, reducing the need for joins between collections, where details about the actor are only shown in the actors collections and not in the films collections. The redundancy would be that the actor's name and birthday are in both the films and actors schema but it is a small expense in exchange for avoiding joins for looking at a few details about an actor in a particular film. If you wanted to find more information about an actor that stars in a particular film, then a join would be used to access an actor's IMDb page, which should also be possible to quickly find all films starring a particular actor.

Json Docs (4 files)

actors.json -


```
[{
  "_id": {
    "$oid": "639324dad1319e98bdbd6f9d"
  },
  "firstName": "Letitia",
  "lastName": "Wright",
  "birthday": {
    "$date": {
      "$numberLong": "783561600000"
    }
  },
  "imdbUrl": "https://www.imdb.com/name/nm4004793/"
},{
  "_id": {
    "$oid": "639324dad1319e98bdbd6f9e"
  },
  "firstName": "Lupita",
  "lastName": "Nyong'o",
  "birthday": {
    "$date": {
      "$numberLong": "415324800000"
    }
  },
  "imdbUrl": "https://www.imdb.com/name/nm2143282/"
},{
  "_id": {
    "$oid": "639324dad1319e98bdbd6f9f"
  },
  "firstName": "Danai",
  "lastName": "Gurira",
  "birthday": {
    "$date": {
      "$numberLong": "256262400000"
    }
  },
  "imdbUrl": "https://www.imdb.com/name/nm1775091/"
},{
  "_id": {
    "$oid": "639324dad1319e98bdbd6fa0"
  },
  "firstName": "Angela",
  "lastName": "Bassett",
  "birthday": {
    "$date": {
      "$numberLong": "-359078400000"
    }
  },
  "imdbUrl": "https://www.imdb.com/name/nm0000291/"
}]
```

films.json -


```
[{
  "_id": {
    "$oid": "63930e8cfa8f8cca749c00a2"
  },
  "title": "Black Panther: Wakanda Forever",
  "duration": 161,
  "releaseDate": {
    "$date": {
      "$numberLong": "1668038400000"
    }
  },
  "rating": "PG-13",
  "genres": [
    "Action",
    "Adventure",
    "Superhero"
  ],
  "actors": [
    {
      "actor_id": {
        "$oid": "639324dad1319e98bdbd6f9d"
      },
      "actor_name": "Letitia Wright",
      "actor_sex": "female",
      "actor_birthday": {
        "$date": {
          "$numberLong": "783561600000"
        }
      }
    },
    {
      "actor_id": {
        "$oid": "639324dad1319e98bdbd6f9e"
      },
      "actor_name": "Lupita Nyong'o",
      "actor_sex": "female",
      "actor_birthday": {
        "$date": {
          "$numberLong": "415324800000"
        }
      }
    },
    {
      "actor_id": {
        "$oid": "639324dad1319e98bdbd6f9f"
      },
      "actor_name": "Danai Gurira",
      "actor_sex": "female",
      "actor_birthday": {
        "$date": {
          "$numberLong": "256262400000"
        }
      }
    },
    {
      "actor_id": {
        "$oid": "639324dad1319e98bdbd6fa0"
      },
```

```
      "actor_name": "Angela Bassett",
      "actor_sex": "female",
      "actor_birthday": {
        "$date": {
          "$numberLong": "-359078400000"
        }
      }
    }
  ]
},{
  "_id": {
    "$oid": "63930ea4fa8f8cca749c00a4"
  },
  "title": "Akeelah and the Bee",
  "duration": 112,
  "releaseDate": {
    "$date": {
      "$numberLong": "1142467200000"
    }
  },
  "rating": "PG",
  "genres": [
    "Family",
    "Drama"
  ],
  "actors": [
    {
      "actor_id": {
        "$oid": "639324dad1319e98bdbd6fa0"
      },
      "actor_name": "Angela Bassett",
      "actor_sex": "female",
      "actor_birthday": {
        "$date": {
          "$numberLong": "-359078400000"
        }
      }
    }
  ]
},{
  "_id": {
    "$oid": "63930de5fa8f8cca749c009f"
  },
  "title": "No Time to Die",
  "duration": 163,
  "releaseDate": {
    "$date": {
      "$numberLong": "1633651200000"
    }
  },
  "rating": "PG-13",
  "genres": [
    "Action",
    "Adventure",
    "Thriller"
  ],
  "actors": []
}]
```


showings.json -

```
[{
  "_id": {
    "$oid": "639317e9fa8f8cca749c00b4"
  },
  "theaterId": {
    "$oid": "639307fbfa8f8cca749c007f"
  },
  "roomTitle": "Screen 1",
  "filmId": {
    "$oid": "63930e8cfa8f8cca749c00a2"
  },
  "filmTitle": "Black Panther: Wakanda Forever",
  "filmRating": "PG-13",
  "showTime": {
    "$date": {
      "$numberLong": "1668095100000"
    }
  },
  "format": "Standard",
  "tickets": [
    {
      "_id": {
        "$oid": "3c4f626a656374496428293e"
      },
      "price": 18,
      "seatNumber": "1"
    }
  ]
},{
  "_id": {
    "$oid": "639318a3fa8f8cca749c00b9"
  },
  "theaterId": {
    "$oid": "6393189efa8f8cca749c00b8"
  },
  "roomTitle": "Screen 1",
  "filmId": {
    "$oid": "63930e8cfa8f8cca749c00a2"
  },
  "filmTitle": "Black Panther: Wakanda Forever",
  "filmRating": "PG-13",
  "showTime": {
    "$date": {
      "$numberLong": "1668106800000"
    }
  },
  "format": "IMAX",
  "tickets": [
    {
      "_id": {
        "$oid": "3c4f626a656374496428293e"
      },
      "price": 22,
      "seatNumber": "1"
    },
    {
      "_id": {
        "$oid": "3c4f626a656374496428293e"
      },
      "price": 15,
      "seatNumber": "5"
    }
```

```
  ]
},{
  "_id": {
    "$oid": "63931915fa8f8cca749c00bb"
  },
  "theaterId": {
    "$oid": "639307fbfa8f8cca749c007f"
  },
  "roomTitle": "Screen 1",
  "filmId": {
    "$oid": "63930e8cfa8f8cca749c00a2"
  },
  "filmTitle": "Black Panther: Wakanda Forever",
  "filmRating": "PG-13",
  "showTime": {
    "$date": {
      "$numberLong": "1668258000000"
    }
  },
  "format": "Standard",
  "tickets": []
}]
```


theaters.json -


```
[{
  "_id": {
    "$oid": "639307fbfa8f8cca749c007f"
  },
  "name": "Regal Edwards Long Beach",
  "address": "7501 E Carson St",
  "city": "Long Beach",
  "state": "CA",
  "zipcode": "90808",
  "rooms": [
    {
      "title": "Screen 1",
      "capacity": 5,
      "formats": [
        "Standard",
        "IMAX"
      ],
      "seats": [
        {
          "number": "1",
          "labels": [
            "reclining"
          ]
        },
        {
          "number": "2",
          "labels": [
            "reclining"
          ]
        },
        {
          "number": "3",
          "labels": [
            "reclining"
          ]
```

```
        },
        {
          "number": "4",
          "labels": [
            "reclining"
          ]
        },
        {
          "number": "5",
          "labels": [
            "accessible seating",
            "non-reclining"
          ]
        }
      ]
    },
    {
      "title": "Screen 2",
      "capacity": 1,
      "formats": [
        "IMAX",
        "ScreenX"
      ],
      "seats": [
        {
          "number": "1"
        }
      ]
    }
  ]
}]
```

Queries

1. Select all films that have "Action" as one of their genres, sorted by film title.

```
db.films.aggregate([
  {
    $match: {
      genres: "Action",

    }
  },
  {
    $sort: {
      title: 1
    }
  }
])
```

2.Select the showtime of all Black Panther: Wakanda Forever showings that are on November 11. You don't need to use joins; you can hard-code the filmId of this film. Hint: you can use a filter with both $lt and $gt comparisons. A date is on November 11 2022 if it is greater than midnight of November 11 and less than ____.... To put a date into a query, you must write ISODate("..."), where the parameter is the date in YYYY-MM-DD format.

```
db.showings.aggregate([
  {
    $match: {
      filmId: ObjectId("63930e8cfa8f8cca749c00a2"),
      showTime: {
        $gt: ISODate("2022-11-11T00:00:00Z"),
        $lt: ISODate("2022-11-12T00:00:00Z")
      }
    }
  },
```

```
  {
    $project: {
      showTime: 1
    }
  }
])
```

3. For each theater, select the name of the theater and also a list of all the film titles and showtimes of all showings scheduled in the theater. This requires a join from theater to showing. (There's 2 more lines of code on the last page.)

```
db.theaters.aggregate([
  {
    $lookup: {
      from: "showings",
      localField: "_id",
      foreignField: "theaterId",
      as: "showings"
    }
  },
  {
    $project: {
      theaterName: "$name",
      allShowings: {
        $map: {
          input: "$showings",
          as: "showing",
          in: {
            filmTitle: "$$showing.filmTitle",
            showTime: "$$showing.showTime"
          }
        }
      }
    }
```

```
  }
])
```