MongoDB – Data Modelling

What Is Data Modelling?

In MongoDB, data modelling is the process of designing the structure of your documents and collections to match your app's requirements and performance needs.

Project Context: E-Learning Platform with Gamification

1.Identify Collections

|  |  |
| --- | --- |
| Collection | Purpose |
| users | All users of the platform (reader, author, admin) |
| blogs | Articles published by users |
| comments | Comments on blogs |
| leaderboard | Gamified user scores and rankings |
| notifications | Alerts for blog actions, badges, updates |
| learning\_paths | Personalized learning routes for users |
| badges | Earnable achievements by users |
| ai\_suggestions | AI-generated blog recommendations |

2. Define Relationships

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Relationship | Type | Description | Field  Reference | Modelling Approach | Justification |
| 1 | User ↔ Blogs | One-to-Many | A user can author many blogs | blogs.authorId → users.\_id | 🔗 Reference | Blogs grow over time; separation is scalable |
| 2 | User ↔ Comments | One-to-Many | A user can write many comments | comments.userd → users.\_id | 🔗 Reference | Comments span blogs; better separated |
| 3 | Blog ↔ Comments | One-to-Many | Each blog can have multiple comments | comments.blogid → blogs.\_id | 🔗 Reference | Enables fast comment retrieval |
| 4 | User ↔ Leaderboard | One-to-One | One user has one leaderboard record | leaderboard.userId → users.\_id | 🔗 Reference | Gamification is an add-on |
| 5 | User ↔ Notifications | One-to-Many | Each user can receive many notifications | notifications.userId → users.\_id | 🔗 Reference | Independent stream of updates |
| 6 | User ↔ Learning Paths | One-to-One | Each user has a single learning path document | learning\_paths.userId → users.\_id | 🔗 Reference | User-specific, grows slowly |
| 7 | Blog ↔ Tags | One-to-Man | Blogs may have tags for categorization | blogs.tags[] | ✅ Embed | Tags are lightweight |
| 8 | AI Suggestions ↔ Blogs | Many-to-Many | Each user gets a list of suggested blogs | ai\_suggestions.userId & blogId | 🔗 Reference | Decouples suggestions logic |
| 9 | User ↔ Badges | One-to-Many | Users can earn multiple badges | users.badges[] | ✅ Embed | Lightweight and rarely updated |

3.Embed vs Reference

|  |  |
| --- | --- |
| Use case | Recommended Modeling |
| Blog authored by a user | Reference authorId |
| Blog comments | Reference blogId |
| AI suggestions to users | Reference |
| Tags for blogs | Embed |
| Badges earned by users | Embed |

4.Sample Schemas

a. users

{

"\_id": ObjectId("..."),

"name": "Alice",

"email": "alice@elearn.com",

"role": "reader",

"points": 120,

"badges": ["bronze", "gold"],

"createdAt": ISODate("2025-07-01T10:00:00Z"),

"updatedAt": ISODate("2025-07-08T15:00:00Z")

}

b. blogs

{

"\_id": ObjectId("..."),

"title": "Intro to AI",

"authorId": ObjectId("..."),

"tags": ["AI", "beginner"],

"views": 100,

"createdAt": ISODate("2025-07-01T10:00:00Z"),

"updatedAt": ISODate("2025-07-08T15:00:00Z")

}

c. comments

{

"\_id": ObjectId("..."),

"blogId": ObjectId("..."),

"userId": ObjectId("..."),

"text": "Great post!",

"rating": 5,

"createdAt": ISODate("2025-07-07T10:00:00Z"),

"updatedAt": ISODate("2025-07-07T10:00:00Z")

}

d. leaderboard

{

"\_id": ObjectId("..."),

"userId": ObjectId("..."),

"points": 120,

"rank": 1,

"createdAt": ISODate("2025-07-05T10:00:00Z"),

"updatedAt": ISODate("2025-07-08T15:00:00Z")

}

e. learning\_paths

{

"\_id": ObjectId("..."),

"userId": ObjectId("..."),

"blogs": [ObjectId("..."), ObjectId("...")],

"completed": [ObjectId("...")],

"createdAt": ISODate("2025-07-01T10:00:00Z"),

"updatedAt": ISODate("2025-07-08T15:00:00Z")

}

f. ai\_suggestions

{

"\_id": ObjectId("..."),

"userId": ObjectId("..."),

"suggestedBlogs": [ObjectId("..."), ObjectId("...")],

"createdAt": ISODate("2025-07-06T10:00:00Z")

}

5.Query Scenarios to Think About

|  |  |
| --- | --- |
| Scenario | Fields to Use |
| Show all blogs by an author | Filter by authorId in blogs |
| Show comments on a blog | Filter comments.blogId |
| Show user rank and points | Lookup from leaderboard |
| Get user’s AI blog suggestions | Filter ai\_suggestions.userId |
| Display user learning progress | Read from learning\_paths |

6.Practice Task

After reviewing these examples, try designing the data model for your project.

* Design the collections and define relationships
* Add createdAt and updatedAt in each schema
* Seed sample documents in MongoDB
* Write 5 common queries your app will use