GRADUAÇÃO

UniCesumar

Banco de Dados NoSql

Professor Msc. Aparecido Vilela Junior

aparecido.vilela@unicesumar.edu.br

Modelagem de Dados

GRADUAÇÃO
UniCesumar

- Agenda da Aula
  - Parte 1: A Filosofia da Modelagem em MongoDB (O "Porquê")
  - Parte 2: Modelando Relacionamentos (Embedded vs. References)
  - Parte 3: Aprofundando em Validação de Schema (Passo a Passo)
  - Parte 4: Desafios Práticos de Validação

A Filosofia do MongoDB

- **"Dados que são acessados juntos, devem ser armazenados juntos."**
- **Modelo Relacional (SQL):**
  - Dados divididos em múltiplas tabelas.
  - Requer JOINs para juntar informações.
  - Analogia: Uma enciclopédia com verbetes separados.
- **Modelo de Documento (MongoDB):**
  - Dados relacionados agrupados em um único documento.
  - Consultas mais rápidas, pois evitam JOINs.
  - Analogia: Um dossiê completo sobre um único assunto.

As Duas Estratégias Principais

- **Modelo Incorporado (Embedded Documents): Colocar um documento dentro de outro.**
- **Modelo de Referência (Document References): Armazenar o _id de um documento em outro.**
- A escolha depende do tipo de relacionamento e do padrão de acesso aos dados.

Estratégia 1: Modelo Incorporado (Embedded)

- O que é: Aninhar documentos ou arrays de documentos.
- **Quando usar:**
- Relacionamentos **"contém" ou "parte de".**
- Relacionamentos **um-para-um ou um-para-poucos.**
- {
- _id: "prod456",
- nome: "Teclado Mecânico Gamer",
- avaliacoes: [
- **{ usuario: "Bia", nota: 5, comentario: "Excelente!" },**
- **{ usuario: "Carlos", nota: 4, comentario: "Bom, mas barulhento." }**
-]
- }

- **Vantagem**: Performance máxima. Todos os dados são lidos em uma única operação.

Estratégia 2: Modelo de Referência

- O que é: Usar IDs para conectar documentos em coleções diferentes (similar a uma FK).
- **Quando usar:**
- Relacionamentos **um-para-muitos ou muitos-para-muitos.**
- Quando os dados aninhados seriam muito grandes ou cresceriam indefinidamente.
- Exemplo: Autores e Livros.
- **Coleção autores:**
- { _id: "autor_01", nome: "George R. R. Martin" }

- **Coleção livros:**
- {
- _id: "livro_101",
- titulo: "A Guerra dos Tronos",
- autores_ids: ["autor_01"] // Referência
- }
- **Vantagem**: Evita redundância e documentos excessivamente grandes.

Estratégia 2: Modelo de Referência

- O que é: Usar IDs para conectar documentos em coleções diferentes (similar a uma FK).
- **Quando usar:**
- Relacionamentos **um-para-muitos ou muitos-para-muitos.**
- Quando os dados aninhados seriam muito grandes ou cresceriam indefinidamente.
- Exemplo: Autores e Livros.
- **Coleção autores:**
- { _id: "autor_01", nome: "George R. R. Martin" }

- **Coleção livros:**
- {
- _id: "livro_101",
- titulo: "A Guerra dos Tronos",
- autores_ids: ["autor_01"] // Referência
- }
- **Vantagem**: Evita redundância e documentos excessivamente grandes.

Exercícios

GRADUAÇÃO
UniCesumar

- Aula 04 - Exercícios Relacional_MogoDB.pdf
- Studeo

Validação de Schema

- **Por que usar Validação?**
- A flexibilidade do MongoDB é poderosa, mas no mundo profissional, precisamos de regras para:
- Garantir a consistência dos dados.
- Evitar erros de digitação e dados incompletos.
- Manter a integridade da base de dados.
- **Como funciona?**
- O MongoDB verifica as regras a cada insert ou update. Se uma regra for violada, a operação falha.

Anatomia de um Validador

- **Anatomia de um Validador**
- Usamos o comando db.createCollection com o operador $jsonSchema.

```
db.createCollection("nome_da_colecao", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: [ "campo1", "campo2" ],
      properties: {
        // Regras para cada campo aqui...
      }
    }
  }
})
```

Anatomia de um Validador

- **bsonType**: Define o tipo de dado.
- **required**: Lista de campos obrigatórios.
- **properties**: Onde as regras de cada campo são definidas.

Guia de Restrições - Tipos e Presença

- **Restrições de Tipo (bsonType)**
- Garante que um campo tenha o tipo de dado correto.
- Exemplo: nome: { bsonType: "**string**" }, idade: { bsonType: "**int**" }
- **Restrições de Presença (required)**
- Define quais campos são obrigatórios em todos os documentos.
- Exemplo: **required**: ["nome", "email"]

Guia de Restrições - Números e Strings

- **Restrições Numéricas (minimum, maximum)**
- Define um intervalo de valores para números.
- Exemplo: idade: { bsonType: "int", minimum: 18, maximum: 100 }

- **Restrições de String (minLength, pattern)**
- Controla o tamanho ou valida o formato com expressões regulares (regex).
- Exemplo: cep: { **bsonType: "string", pattern: "^[0-9]{5}-[0-9]{3}$" }**

- **Restrições de Enumeração (enum)**
- Limita um campo a uma lista de valores permitidos.
- Exemplo: departamento: { enum: ["Tecnologia", "RH", "Financeiro"] }

Guia de Restrições - Arrays

GRADUAÇÃO UniCesumar

- **Restrições de Array (minItems, uniqueItems)**
- **Controla o tamanho do array e garante que os itens sejam únicos.**
  - Exemplo: tags: { bsonType: "array", minItems: 1, uniqueItems: true }
  - Validando Itens Dentro do Array (items)

- Permite definir regras para os documentos dentro do array.
- **Exemplo:**
- avaliacoes: {
- bsonType: "**array**",
- items: {
- bsonType: "object",
- required: ["usuario", "nota"],
- properties: {
- nota: { bsonType: "int", minimum: 1, maximum: 5 }
- }
- }
- }

Desafios Práticos de Validação

- **Desafio 1 (Nível Fácil)**
- Cenário: Um sistema de e-commerce precisa de garantir que todos os produtos cadastrados tenham um padrão mínimo de qualidade de dados.
- Sua Tarefa: Crie uma coleção produtos com as seguintes regras:
- **nome_produto**: Deve ser uma string e é obrigatório.
- **preco**: Deve ser um double e é obrigatório.
- **em_estoque**: Deve ser um booleano e é obrigatório.

Desafios Práticos de Validação

- **Desafio 2 (Nível Médio)**
- Cenário: Um sistema de gestão de eventos precisa de validar os dados dos participantes no momento da inscrição.

- Sua Tarefa: Crie uma coleção inscricoes com as seguintes regras:

- nome_participante: String obrigatória com no mínimo 5 caracteres.
- idade: Inteiro obrigatório com valor mínimo de 18.
- tipo_ingresso: String obrigatória que só pode ser um dos seguintes valores: "VIP", "Pista" ou "Camarote".

Desafios Práticos de Validação

- **Desafio 3 (Nível Difícil)**
- Cenário: Um sistema de e-commerce precisa de validar a estrutura de um pedido, incluindo os seus itens.

- Sua Tarefa: Crie uma coleção pedidos com as seguintes regras:

- cliente_id: String obrigatória.
- data_pedido: Date obrigatório.
- itens: Deve ser um array e não pode estar vazio (minItems: 1).
- Cada elemento dentro do array itens deve ser um objeto que contenha:
- produto_id: String obrigatória.
- quantidade: Inteiro obrigatório com valor mínimo de 1.
- preco_unitario: Double obrigatório.