# Atividade Prática - MegaArena eSports: MongoDB

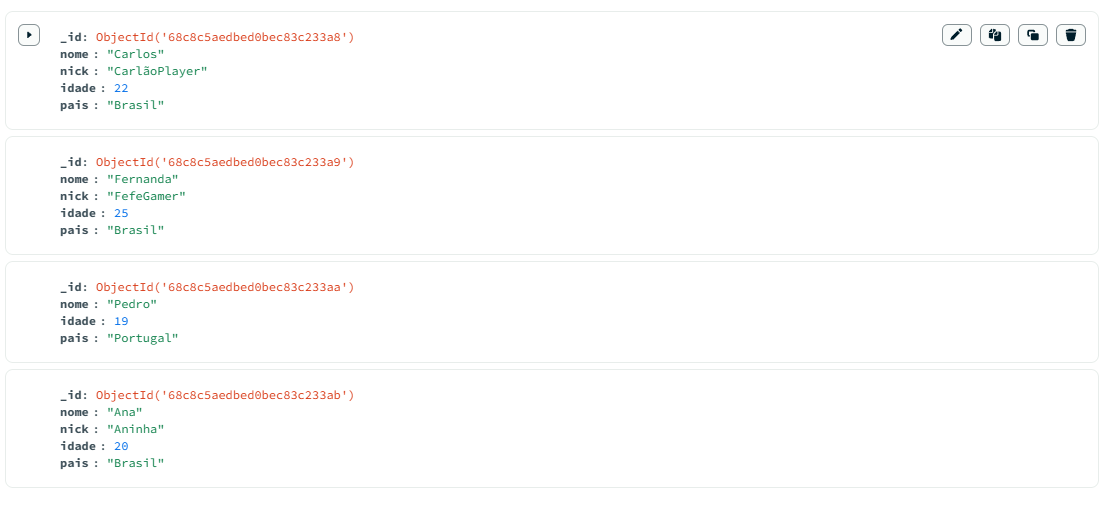
Este documento contém as respostas para a atividade prática de Banco de Dados Não Relacional, focada na plataforma MegaArena eSports utilizando MongoDB.

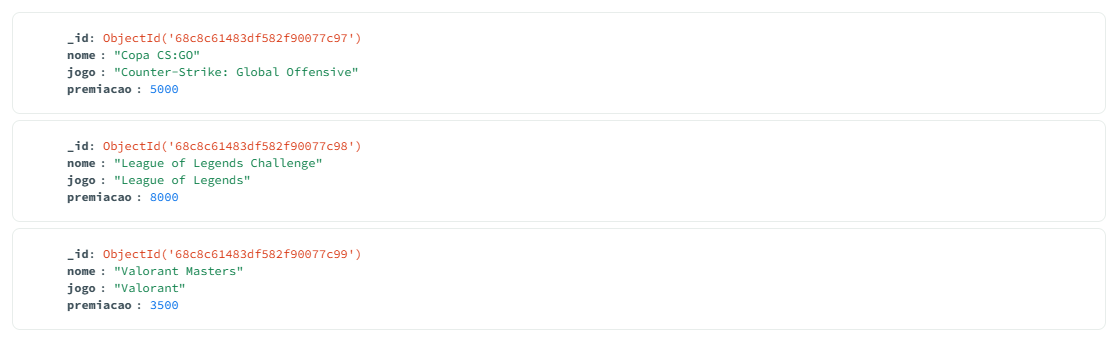
### Questão 1: Criação do Banco

Para criar o banco de dados e as coleções, e inserir alguns dados de exemplo para as próximas questões, podemos usar os seguintes comandos:

// Conecta ou cria o banco de dados megaArena  
use megaArena

  
  
// Cria a coleção 'jogadores' e insere alguns documentos  
db.jogadores.insertMany([  
 {  
 nome: "Carlos",  
 nick: "CarlãoPlayer",  
 idade: 22,  
 pais: "Brasil"  
 },  
 {  
 nome: "Fernanda",  
 nick: "FefeGamer",  
 idade: 25,  
 pais: "Brasil"  
 },  
 {  
 nome: "Pedro",  
 idade: 19,  
 pais: "Portugal"  
 },  
 {  
 nome: "Ana",  
 nick: "Aninha",  
 idade: 20,  
 pais: "Brasil"  
 }  
]);

  
  
// Cria a coleção 'torneios' e insere alguns documentos  
db.torneios.insertMany([  
 {  
 nome: "Copa CS:GO",  
 jogo: "Counter-Strike: Global Offensive",  
 premiacao: 5000  
 },  
 {  
 nome: "League of Legends Challenge",  
 jogo: "League of Legends",  
 premiacao: 8000  
 },  
 {  
 nome: "Valorant Masters",  
 jogo: "Valorant",  
 premiacao: 3500  
 }  
]);



// Cria a coleção 'partidas’  
db.createCollection("partidas")



### Questão 2: Embedding

Nesta modelagem, os detalhes dos jogadores inscritos são incorporados (embedded) diretamente no documento do torneio.

{  
 "\_id": ObjectId("632a1c1d0a8e3e4a3c1d9e3a"),  
 "nome": "Copa de Inverno",  
 "jogo": "FIFA 23",  
 "premiacao": 7000,  
 "jogadores\_inscritos": [  
 {  
 "nome": "Carlos",  
 "nick": "CarlãoPlayer",  
 "pais": "Brasil"  
 },  
 {  
 "nome": "Fernanda",  
 "nick": "FefeGamer",  
 "pais": "Brasil"  
 }  
 ]  
}

### Questão 3: Referencing

Nesta abordagem, o documento do torneio armazena apenas as referências (os \_ids) dos jogadores, que são mantidos em sua própria coleção.

// Documento do Torneio  
{  
 "\_id": ObjectId("632a1c1d0a8e3e4a3c1d9e3b"),  
 "nome": "Copa de Verão",  
 "jogo": "Street Fighter 6",  
 "premiacao": 6000,  
 "jogadores\_ids": [  
 ObjectId("632a1b0b0a8e3e4a3c1d9e2a"), // ID do Carlos  
 ObjectId("632a1b0b0a8e3e4a3c1d9e2b") // ID da Fernanda  
 ]  
}  
  
// Documentos na coleção 'jogadores'  
// { \_id: ObjectId("632a1b0b0a8e3e4a3c1d9e2a"), nome: "Carlos", ... }  
// { \_id: ObjectId("632a1b0b0a8e3e4a3c1d9e2b"), nome: "Fernanda", ... }

### Questão 4: Justificativa de Modelagem

A escolha entre *embedding* e *referencing* depende dos padrões de acesso e da natureza dos dados.

* **Usar Embedding (incorporação):**
  + Melhora o desempenho de leitura, pois todas as informações necessárias podem ser recuperadas com uma única consulta ao banco de dados, evitando a necessidade de múltiplas buscas (operações de $lookup ou join).
* **Usar Referencing (referência):**
  + Evita a duplicação de dados e mantém os documentos menores. Se um jogador atualiza seu nick, a mudança só precisa ser feita em um único lugar (na coleção jogadores), em vez de em todos os torneios onde ele foi incorporado. Isso também evita que os documentos de torneio fiquem muito grandes.

### Questão 5: Consulta

Listar todos os torneios cujo prêmio seja maior ou igual a 4000.

db.torneios.find({ premiacao: { $gte: 4000 } })



### Questão 6: Consulta Avançada com $and

Listar os jogadores do Brasil com idade maior que 21 anos.

db.jogadores.find({  
 $and: [  
 { pais: "Brasil" },  
 { idade: { $gt: 21 } }  
 ]  
})



### Questão 7: Operador $exists

Listar todos os jogadores que já possuem o campo nick.

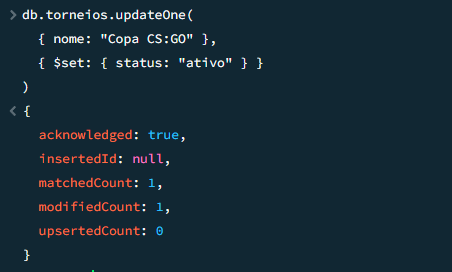
db.jogadores.find({ nick: { $exists: true } })



### Questão 8: Atualização com $set

Atualizar o torneio "Copa CS:GO" adicionando o campo status: "ativo".

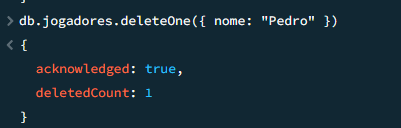
db.torneios.updateOne(  
 { nome: "Copa CS:GO" },  
 { $set: { status: "ativo" } }  
)



### Questão 9: Exclusão

Remover o jogador com nome "Pedro".

db.jogadores.deleteOne({ nome: "Pedro" })



### Questão 10: Paginação

Listar os torneios ordenados pela premiação (decrescente), exibindo apenas o segundo torneio mais bem pago.

db.torneios.find().sort({ premiacao: -1 }).skip(1).limit(1)



* sort({ premiacao: -1 }): Ordena os torneios pela premiação em ordem decrescente.
* skip(1): Pula o primeiro resultado (o mais bem pago).
* limit(1): Limita o resultado a apenas um documento (o segundo mais bem pago).