Prova I

Disciplina: Banco de Dados Não Relacional

Período: 4°

Valor da Prova: 07 Pontos

Professor: Diego Ramos Inácio

Curso de Engenharia de Software Município: Saquarema - RJ

Aluno:_____

Matricula:_____

Definições:

INSTRUÇÕES GERAIS

- Leia atentamente cada questão antes de responder
- Marque apenas UMA alternativa por questão
- Questões rasuradas serão anuladas
- Não é permitido o uso de dispositivos eletrônicos durante a prova
- Duração da prova: 2 horas

PARTE 1: QUESTÕES TEÓRICAS (2,0 pontos)

Questão 1 - História dos Bancos NoSQL (0,5 pontos)

De acordo com a linha do tempo apresentada nas aulas, qual foi o principal marco que iniciou a era NoSQL e suas motivações técnicas?

( ) **a)** O surgimento do modelo relacional de Edgar Codd na IBM em 1970, que estabeleceu SQL e normalização

( ) **b)** A publicação dos papers do Google (GFS, MapReduce, Bigtable) e do Amazon Dynamo entre 2003-2007, que apresentaram soluções para problemas de escala web

( ) **c)** O desenvolvimento de Data Warehouses e sistemas OLAP nos anos 90 para Business Intelligence

( ) **d)** A criação dos bancos hierárquicos e de rede (IMS, CODASYL) nos anos 60-80 com navegação por ponteiros

( ) **e)** O amadurecimento do ecossistema Hadoop com MongoDB, Neo4j e Cassandra em 2010

Questão 2 - Teorema CAP e Filosofias ACID vs BASE (0,5 pontos)

Sobre o Teorema CAP e as filosofias ACID vs BASE, analise as afirmativas:

I. O Teorema CAP estabelece que em sistemas distribuídos é possível garantir simultaneamente Consistência,

Disponibilidade e Tolerância a Partições

II. ACID (Atomicity, Consistency, Isolation, Durability) é característico de bancos relacionais e prioriza consistência forte e transações complexas

III. BASE (Basically Available, Soft state, Eventual consistency) ganhou espaço quando throughput e disponibilidade tornaram-se mais críticos que consistência imediata

IV. Na realidade prática, os trade-offs do CAP são binários e permanentes, aplicando-se igualmente a todas as operações

Estão corretas apenas:

( ) **a)** I e II
( ) **b)** II e III
( ) **c)** I, II e IV
( ) **d)** II, III e IV
( ) **e)** I, III e IV

Questão 3 - Evolução Histórica dos Bancos de Dados (0,5 pontos)

Sobre a evolução histórica dos bancos de dados apresentada nas aulas, analise as afirmativas:

I. Na era pré-relacional (1960-1970), os modelos hierárquico (IBM IMS) e de rede (CODASYL) apresentavam limitações como navegação complexa e dependência de aplicação

II. A Revolução Relacional iniciada por Edgar F. Codd em 1970 trouxe o modelo de tabelas com propriedades ACID, consolidando-se entre 1990-2000 com 95% do mercado

III. O termo "NoSQL" significa "No SQL" (Não SQL) e foi criado para substituir completamente os bancos relacionais em todas as aplicações

IV. O crescimento da Internet (2000-2005) trouxe novos desafios como explosão de dados, necessidade de escalabilidade para milhões de usuários simultâneos e variedade de tipos de dados

Estão corretas:

( ) **a)** I, II e III apenas
( ) **b)** I, II e IV apenas
( ) **c)** II, III e IV apenas
( ) **d)** I, III e IV apenas
( ) **e)** Todas as afirmativas

Questão 4 - Tipos de Bancos NoSQL (0,5 pontos)

Sobre os tipos de bancos NoSQL e seus casos de uso, assinale a alternativa que faz a correspondência CORRETA:

( ) **a)** Chave-Valor (Redis) - ideal para redes sociais e detecção de fraudes com relacionamentos complexos

( ) **b)** Documento (MongoDB) - otimizado para data warehousing e consultas analíticas com compressão eficiente

( ) **c)** Coluna (Cassandra, HBase) - estrutura de dados organizados por famílias de colunas, vantajoso para analytics e séries temporais

( ) **d)** Grafo (Neo4j) - armazena pares chave-valor simples, ideal para cache e contadores de alta performance

( ) **e)** Documento (CouchDB) - utiliza nós e arestas, adequado para consultas de roteamento em mapas

---

PARTE 2: QUESTÕES PRÁTICAS - MONGOSH (3,0 pontos)

CONTEXTO: Você é desenvolvedor de um sistema de e-commerce chamado "TechStore" que vende produtos eletrônicos. O banco de dados techstore_db possui a coleção produtos com a seguinte estrutura de documentos:

```
{
  _id: ObjectId("..."),
  nome: "Notebook Dell Inspiron",
  categoria: "Informática",
  preco: 3500.00,
  estoque: 15,
  fabricante: {
    nome: "Dell",
    pais: "EUA"
  },
  especificacoes: {
    ram: "16GB",
    processador: "Intel i7",
    armazenamento: "512GB SSD"
  },
  avaliacoes: [8.5, 9.0, 7.5],
  tags: ["notebook", "gaming", "trabalho"],
  ativo: true,
  data_cadastro: ISODate("2024-01-15")
}
```

Questão 5 - CREATE (0,75 pontos)

SITUAÇÃO: Sua empresa fechou parceria com três fornecedores e você precisa cadastrar os seguintes produtos **DE UMA ÚNICA VEZ** no banco:

Produto 1: Smartphone Samsung Galaxy S23

- Categoria: "Smartphones"
- Preço: 4200.00
- Estoque: 25 unidades
- Fabricante: Samsung, Coreia do Sul
- Especificações: tela "6.1 polegadas", camera "50MP", bateria "3900mAh"
- Avaliações: [9.0, 8.5]
- Tags: ["smartphone", "5G", "flagship"]
- Ativo: true

Produto 2: Mouse Logitech MX Master 3

- Categoria: "Periféricos"
- Preço: 650.00
- Estoque: 50 unidades
- Fabricante: Logitech, Suíça
- Especificações: conexao "Bluetooth", dpi "4000", botoes 7
- Avaliações: [9.5, 9.0, 8.8]
- Tags: ["mouse", "wireless", "produtividade"]
- Ativo: true

Produto 3: Monitor LG UltraWide 29"

- Categoria: "Monitores"
- Preço: 1850.00
- Estoque: 8 unidades
- Fabricante: LG, Coreia do Sul
- Especificações: resolucao "2560x1080", taxa_atualizacao "75Hz", tipo_painel "IPS"
- Avaliações: [] (array vazio)
- Tags: ["monitor", "ultrawide", "profissional"]
- Ativo: false

Assinale o comando CORRETO:

( ) **a)**

```
db.produtos.insertMany([
  {
    nome: "Smartphone Samsung Galaxy S23",
    categoria: "Smartphones",
    preco: 4200.00,
    estoque: 25,
    fabricante: {nome: "Samsung", pais: "Coreia do Sul"},
    especificacoes: {tela: "6.1 polegadas", camera: "50MP", bateria: "3900mAh"},
    avaliacoes: [9.0, 8.5],
    tags: ["smartphone", "5G", "flagship"],
    ativo: true
  },
  {
    nome: "Mouse Logitech MX Master 3",
    categoria: "Periféricos",
    preco: 650.00,
```

```
    estoque: 50,
    fabricante: {nome: "Logitech", pais: "Suíça"},
    especificacoes: {conexao: "Bluetooth", dpi: "4000", botoes: 7},
    avaliacoes: [9.5, 9.0, 8.8],
    tags: ["mouse", "wireless", "produtividade"],
    ativo: true
  },
  {
    nome: "Monitor LG UltraWide 29\"",
    categoria: "Monitores",
    preco: 1850.00,
    estoque: 8,
    fabricante: {nome: "LG", pais: "Coreia do Sul"},
    especificacoes: {resolucao: "2560x1080", taxa_atualizacao: "75Hz", tipo_painel: "IPS"},
    avaliacoes: [],
    tags: ["monitor", "ultrawide", "profissional"],
    ativo: false
  }
])
```

( ) **b)**

```
db.produtos.insertOne([
  {nome: "Smartphone Samsung Galaxy S23", categoria: "Smartphones", preco: 4200.00},
  {nome: "Mouse Logitech MX Master 3", categoria: "Periféricos", preco: 650.00}
])
```

( ) **c)**

```
db.produtos.insert({
  produtos: [
    {nome: "Smartphone Samsung Galaxy S23", preco: 4200.00},
    {nome: "Mouse Logitech MX Master 3", preco: 650.00}
  ]
})
```

( ) **d)**

```
db.produtos.insertMany([
  {nome: "Smartphone Samsung Galaxy S23", preco: "4200.00", estoque: 25},
  {nome: "Mouse Logitech MX Master 3", preco: "650.00", estoque: 50}
```

])

Questão 6 - READ (0,75 pontos)

SITUAÇÃO: O gerente de vendas solicitou um relatório com produtos que atendam **SIMULTANEAMENTE** todos os critérios:

✓ Categoria seja "Smartphones" **OU** "Monitores"
✓ Preço entre R$ 1000,00 e R$ 5000,00 (valores **INCLUSIVE**)
✓ Estoque **MAIOR QUE** 10 unidades
✓ Produto ativo (ativo = true)

O relatório deve exibir **APENAS**: nome, preço, estoque e nome do fabricante (**SEM** mostrar _id).

Assinale o comando CORRETO:

( ) **a)**

```
db.produtos.find(
  {
    categoria: {$in: ["Smartphones", "Monitores"]},
    preco: {$gte: 1000, $lte: 5000},
    estoque: {$gt: 10},
    ativo: true
  },
  {
    nome: 1,
    preco: 1,
    estoque: 1,
    "fabricante.nome": 1,
    _id: 0
  }
)
```

( ) **b)**

```
db.produtos.find(
  {
    categoria: ["Smartphones", "Monitores"],
    preco: {$between: [1000, 5000]},
    estoque: {$gte: 10},
    ativo: true
  },
  {nome: 1, preco: 1, estoque: 1, fabricante: 1}
)
```

( ) **c)**

```
db.produtos.find(
  {
    $or: [{categoria: "Smartphones"}, {categoria: "Monitores"}],
    preco: {$gte: 1000, $lt: 5000},
    estoque: {$gt: 10},
    ativo: true
  },
  {nome: 1, preco: 1, estoque: 1, "fabricante.nome": 1, _id: 0}
)
```

( ) **d)**

```
db.produtos.findOne(
  {
    categoria: {$in: ["Smartphones", "Monitores"]},
    preco: {$gte: 1000, $lte: 5000},
    estoque: {$gt: 10}
  },
  {nome: 1, preco: 1, estoque: 1, "fabricante.nome": 1, _id: 0}
)
```

Questão 7 - UPDATE (0,75 pontos)

SITUAÇÃO: Durante uma promoção de Black Friday, você precisa executar **EM SEQUÊNCIA** as seguintes operações:

1) Aumentar em 20 unidades o estoque de **TODOS** os produtos da categoria "Periféricos"

2) Aplicar 15% de desconto (multiplicar por 0.85) no preço de **TODOS** os produtos com estoque menor que 10

3) Adicionar a tag "black-friday" em **TODOS** os produtos ativos

4) Adicionar a avaliação 9.2 no produto "Mouse Logitech MX Master 3"

Assinale a sequência de comandos CORRETA:

( ) **a)**

```
db.produtos.updateMany({categoria: "Periféricos"}, {$inc: {estoque: 20}})
db.produtos.updateMany({estoque: {$lt: 10}}, {$mul: {preco: 0.85}})
db.produtos.updateMany({ativo: true}, {$push: {tags: "black-friday"}})
db.produtos.updateOne({nome: "Mouse Logitech MX Master 3"}, {$push: {avaliacoes: 9.2}})
```

( ) **b)**

```
db.produtos.updateMany({categoria: "Periféricos"}, {$set: {estoque: 20}})
db.produtos.updateMany({estoque: {$lte: 10}}, {$set: {preco: preco * 0.85}})
```

db.produtos.updateMany({ativo: true}, {$set: {tags: "black-friday"}})

db.produtos.updateMany({nome: "Mouse Logitech MX Master 3"}, {$push: {avaliacoes: 9.2}})

( ) **c)**

db.produtos.update({categoria: "Periféricos"}, {$inc: {estoque: 20}})

db.produtos.update({estoque: {$lt: 10}}, {$multiply: {preco: 0.85}})

db.produtos.update({ativo: true}, {$addToSet: {tags: "black-friday"}})

db.produtos.updateOne({nome: "Mouse Logitech MX Master 3"}, {$add: {avaliacoes: 9.2}})

( ) **d)**

db.produtos.updateMany({categoria: "Periféricos"}, {$inc: {estoque: 20}, $push: {tags: "black-friday"}})

db.produtos.updateMany({estoque: {$lt: 10}, ativo: true}, {$mul: {preco: 0.85}})

db.produtos.updateOne({nome: "Mouse Logitech MX Master 3"}, {$push: {avaliacoes: 9.2}})

Questão 8 - DELETE (0,75 pontos)

SITUAÇÃO: A empresa fará limpeza no estoque seguindo este procedimento:

1) Primeiro: CONTAR quantos produtos atendem a **QUALQUER UM** dos critérios:

- Produtos inativos (ativo = false) **OU**
- Produtos com estoque zerado **OU**
- Produtos sem nenhuma avaliação (array vazio) **OU**
- Produtos com preço acima de R$ 10.000,00

2) Segundo: REMOVER todos os produtos que atendem aos critérios acima

3) Terceiro: VERIFICAR quantos produtos restaram no banco

Assinale a sequência CORRETA:

( ) **a)**

```
db.produtos.countDocuments({
  $or: [
    {ativo: false},
    {estoque: 0},
    {avaliacoes: {$size: 0}},
    {preco: {$gt: 10000}}
  ]
})


db.produtos.deleteMany({
  $or: [
    {ativo: false},
```

```
    {estoque: 0},
    {avaliacoes: {$size: 0}},
    {preco: {$gt: 10000}}
  ]
})

db.produtos.countDocuments({})
```
( ) **b)**
```
db.produtos.count({
  ativo: false,
  estoque: 0,
  avaliacoes: [],
  preco: {$gt: 10000}
})

db.produtos.deleteMany({
  ativo: false,
  estoque: 0,
  avaliacoes: [],
  preco: {$gt: 10000}
})

db.produtos.count()
```
( ) **c)**
```
db.produtos.find({
  $or: [
    {ativo: false},
    {estoque: {$eq: 0}},
    {avaliacoes: {$exists: false}},
    {preco: {$gte: 10000}}
  ]
}).count()

db.produtos.deleteOne({
  $or: [
```

```
    {ativo: false},
    {estoque: {$eq: 0}},
    {avaliacoes: {$exists: false}},
    {preco: {$gte: 10000}}
  ]
})

db.produtos.countDocuments({})
```

( ) **d)**

```
db.produtos.countDocuments({
  ativo: false || estoque: 0 || avaliacoes: [] || preco: {$gt: 10000}
})

db.produtos.deleteMany({
  ativo: false || estoque: 0 || avaliacoes: [] || preco: {$gt: 10000}
})

db.produtos.countDocuments({})
```

PARTE 3: QUESTÕES PRÁTICAS - PYTHON (2,0 pontos)

IMPORTANTE: Considere que a conexão com MongoDB já está estabelecida:

```
from pymongo import MongoClient
from datetime import datetime

client = MongoClient("mongodb://localhost:27017")
db = client["techstore_db"]
produtos = db["produtos"]
```

Questão 9 - READ Avançado em Python (1,0 ponto)

SITUAÇÃO: O departamento de marketing solicitou um relatório detalhado. Você deve criar a função gerar_relatorio_fabricantes() que realize:

REQUISITOS:

1. Buscar **TODOS** os produtos ativos com preço **superior a R$ 500,00**
2. Agrupar os resultados por **país do fabricante**
3. Para cada país, calcular:
    - **Quantidade total** de produtos

- **Preço médio** dos produtos (arredondado para 2 casas decimais)
- **Lista com os nomes** dos produtos

4. **Ordenar** os resultados por quantidade de produtos (do maior para o menor)
5. **Retornar** um dicionário Python com a estrutura:

```
{
  "Brasil": {
    "quantidade": 5,
    "preco_medio": 2500.00,
    "produtos": ["Produto A", "Produto B", ...]
  },
  "EUA": {...}
}
```

Assinale a implementação COMPLETA e CORRETA:

( ) a)

```
def gerar_relatorio_fabricantes():
    pipeline = [
        {"$match": {"ativo": True, "preco": {"$gt": 500}}},
        {"$group": {
            "_id": "$fabricante.pais",
            "quantidade": {"$sum": 1},
            "preco_medio": {"$avg": "$preco"},
            "produtos": {"$push": "$nome"}
        }},
        {"$sort": {"quantidade": -1}}
    ]

    resultado = produtos.aggregate(pipeline)

    relatorio = {}
    for item in resultado:
        pais = item["_id"]
        relatorio[pais] = {
            "quantidade": item["quantidade"],
            "preco_medio": round(item["preco_medio"], 2),
            "produtos": item["produtos"]
```

```
        }

    return relatorio
```

( ) **b)**

```
def gerar_relatorio_fabricantes():
    resultado = produtos.find(
        {"ativo": True, "preco": {"$gt": 500}},
        {"nome": 1, "preco": 1, "fabricante.pais": 1, "_id": 0}
    )

    relatorio = {}
    for doc in resultado:
        pais = doc["fabricante"]["pais"]
        if pais not in relatorio:
            relatorio[pais] = {
                "quantidade": 0,
                "preco_medio": 0,
                "produtos": []
            }
        relatorio[pais]["quantidade"] += 1
        relatorio[pais]["produtos"].append(doc["nome"])

    return relatorio
```

( ) **c)**

```
def gerar_relatorio_fabricantes():
    cursor = produtos.find({"ativo": True, "preco": {"$gte": 500}})

    from collections import defaultdict
    relatorio = defaultdict(lambda: {"quantidade": 0, "preco_total": 0, "produtos": []})

    for produto in cursor:
        pais = produto["fabricante"]["pais"]
        relatorio[pais]["quantidade"] += 1
        relatorio[pais]["preco_total"] += produto["preco"]
        relatorio[pais]["produtos"].append(produto["nome"])
```

```
    for pais in relatorio:
        relatorio[pais]["preco_medio"] = round(
            relatorio[pais]["preco_total"] / relatorio[pais]["quantidade"], 2
        )
        del relatorio[pais]["preco_total"]

    return dict(sorted(relatorio.items(),
                       key=lambda x: x[1]["quantidade"],
                       reverse=True))
```

( ) **d)**

```
def gerar_relatorio_fabricantes():
    resultado = list(produtos.find(
        {"ativo": True, "preco": {"$gt": 500}}
    ))

    relatorio = {}
    for produto in resultado:
        pais = produto.get("fabricante", {}).get("pais", "Desconhecido")
        nome = produto.get("nome")

        if pais in relatorio:
            relatorio[pais]["produtos"].append(nome)
        else:
            relatorio[pais] = {"produtos": [nome]}

    return relatorio
```

Questão 10 - UPDATE, DELETE e Validação em Python (1,0 ponto)

SITUAÇÃO: Implemente a função processar_ajuste_estoque() que execute:

OPERAÇÕES OBRIGATÓRIAS:

1) ATUALIZAÇÃO DE PREÇOS:
Aumentar em 10% o preço de **TODOS** os produtos do fabricante "Samsung" que tenham **avaliação média maior ou igual a 8.0**

2) GESTÃO DE ESTOQUE:
Para produtos com estoque **entre 1 e 5 unidades** (inclusive):

- Adicionar a tag "estoque-baixo"

- Simular envio de e-mail (usando print)

3) LIMPEZA:
Deletar produtos **inativos** que estão há **mais de 6 meses** sem estoque (estoque = 0)

4) RELATÓRIO:
Retornar dicionário contendo:

- Quantidade de produtos com preço atualizado
- Lista de nomes dos produtos com estoque baixo
- Quantidade de produtos deletados

Assinale a implementação COMPLETA e CORRETA:

( ) **a)**

```
def processar_ajuste_estoque():
    from datetime import datetime, timedelta

    # 1. Atualizar Samsung com boa avaliação
    produtos_samsung = produtos.find({"fabricante.nome": "Samsung"})
    atualizados = 0

    for produto in produtos_samsung:
        if produto.get("avaliacoes"):
            media = sum(produto["avaliacoes"]) / len(produto["avaliacoes"])
            if media >= 8.0:
                produtos.update_one(
                    {"_id": produto["_id"]},
                    {"$mul": {"preco": 1.10}}
                )
                atualizados += 1

    # 2. Marcar estoque baixo
    estoque_baixo = []
    produtos_baixo_estoque = produtos.find({
        "estoque": {"$gte": 1, "$lte": 5}
    })

    for produto in produtos_baixo_estoque:
        produtos.update_one(
            {"_id": produto["_id"]},
```

```
            {"$addToSet": {"tags": "estoque-baixo"}}
        )
        estoque_baixo.append(produto["nome"])
        print(f"EMAIL: Alerta de estoque baixo para {produto['nome']}")

    # 3. Deletar produtos antigos sem estoque
    data_limite = datetime.now() - timedelta(days=180)
    resultado_delete = produtos.delete_many({
        "ativo": False,
        "estoque": 0,
        "data_cadastro": {"$lt": data_limite}
    })

    return {
        "precos_atualizados": atualizados,
        "produtos_estoque_baixo": estoque_baixo,
        "produtos_deletados": resultado_delete.deleted_count
    }
```

( ) **b)**

```
def processar_ajuste_estoque():
    # 1. Atualizar Samsung
    resultado_update = produtos.update_many(
        {"fabricante.nome": "Samsung"},
        {"$mul": {"preco": 1.10}}
    )

    # 2. Adicionar tag
    produtos.update_many(
        {"estoque": {"$gte": 1, "$lte": 5}},
        {"$push": {"tags": "estoque-baixo"}}
    )

    lista_baixo = list(produtos.find(
        {"estoque": {"$gte": 1, "$lte": 5}},
        {"nome": 1, "_id": 0}
```

```
    ))

    # 3. Deletar
    resultado_delete = produtos.delete_many({
        "ativo": False,
        "estoque": 0
    })

    return {
        "precos_atualizados": resultado_update.modified_count,
        "produtos_estoque_baixo": [p["nome"] for p in lista_baixo],
        "produtos_deletados": resultado_delete.deleted_count
    }
```

( ) **c)**

```
def processar_ajuste_estoque():
    # 1. Atualizar Samsung
    produtos.update_many(
        {
            "fabricante.nome": "Samsung",
            "avaliacoes": {"$exists": True, "$ne": []}
        },
        {"$set": {"preco": {"$multiply": ["$preco", 1.10]}}}
    )

    # 2. Estoque baixo
    resultado_baixo = produtos.find({"estoque": {"$between": [1, 5]}})
    nomes_baixo = []

    for p in resultado_baixo:
        produtos.update_one(
            {"_id": p["_id"]},
            {"$set": {"tags": ["estoque-baixo"]}}
        )
        nomes_baixo.append(p["nome"])
```

```
    # 3. Deletar
    resultado_del = produtos.delete_many({"ativo": False})

    return {
        "precos_atualizados": "N/A",
        "produtos_estoque_baixo": nomes_baixo,
        "produtos_deletados": resultado_del.deleted_count
    }
```

( ) **d)**

```
def processar_ajuste_estoque():
    import datetime

    # 1. Update Samsung
    count_samsung = 0
    for produto in produtos.find({"fabricante.nome": "Samsung"}):
        avaliacoes = produto.get("avaliacoes", [])
        if avaliacoes and (sum(avaliacoes) / len(avaliacoes)) >= 8.0:
            produtos.update_one(
                {"_id": produto["_id"]},
                {"$inc": {"preco": produto["preco"] * 0.10}}
            )
            count_samsung += 1

    # 2. Estoque baixo
    baixo = list(produtos.find({"estoque": {"$in": [1, 2, 3, 4, 5]}}))

    for item in baixo:
        produtos.update_one(
            {"_id": item["_id"]},
            {"$push": {"tags": "estoque-baixo"}}
        )

    # 3. Delete
    deleted = produtos.delete_many({
        "ativo": False,
```

```
    "estoque": {"$eq": 0}
})

return {
    "precos_atualizados": count_samsung,
    "produtos_estoque_baixo": [b["nome"] for b in baixo],
    "produtos_deletados": deleted.deleted_count
}
```

FOLHA DE RESPOSTAS

Essa prova deve ser apresentada na conclusão e os testes da parte prática também devem ser apresentados em anexo

Questão	Resposta	Questão	Resposta
1	( )	6	( )
2	( )	7	( )
3	( )	8	( )
4	( )	9	( )
5	( )	10	( )

Boa prova! Lembre-se de revisar suas respostas antes de entregar.