Atividade Final - NoSQL
XPTO Clube de Compras

Case:

A XPTO Clube de Compras permite que usuários façam compras online em lojas de parceiros e também diretamente dos fornecedores, ocasionando em cash-back e sorteios de diversos brindes, descontos.
Empresas podem oferecer para seus funcionários cartões e tickets.
Uma das funcionalidades chave da plataforma, é a possibilidade de inclusão de amigos na rede, assim, aumentando descontos coletivos para determinados volumes de compras. Assinaturas coletivas também são possíveis.

1. Problemas de escalabilidade da arquitetura transacional atual

A arquitetura atual utiliza Postgres, que é excelente para transações relacionais, mas pode apresentar problemas de escalabilidade em plataformas de alto volume, como:

- **Carga de leitura/escrita alta**: Bancos relacionais podem sofrer com degradação de desempenho quando o volume de transações cresce exponencialmente.
- **Complexidade em joins**: A obtenção de dados de tabelas relacionadas pode se tornar lenta à medida que os dados aumentam.
- **Dificuldade para escalar horizontalmente**: Postgres não foi projetado nativamente para sharding ou replicação em larga escala.

---

2. Proposta de dois modelos com NoSQL

1. **Modelo baseado em MongoDB**:
    - **Armazenamento**:
        - Dados de usuários, compras, e produtos seriam armazenados em documentos JSON.

Exemplo de documento (coleção `usuarios`):

```
{
  "id_usuario": "123",
  "nome": "João Silva",
  "aquisicoes": [
    {"id_produto": "101", "quantidade": 2, "data": "2024-12-01"},
    {"id_produto": "102", "quantidade": 1, "data": "2024-12-02"}
  ]
}
```

Vantagem:

        - Dados relacionados armazenados juntos, evitando joins.
        - Escalabilidade horizontal simplificada.
    - **Limitação**:
        - Duplicação de dados, exigindo mecanismos de consistência.
2. **Modelo baseado em Cassandra**:
    - **Armazenamento**:
        - Dados particionados por chave (`id_usuario`) para alta disponibilidade e leitura rápida.
    - **Vantagem**:
        - Alta performance em leituras e escritas em larga escala.
        - Escalabilidade linear.
    - **Limitação**:
        - Menor flexibilidade em consultas complexas.

---

3. Limitações do modelo relacional e vantagens do NoSQL

Problemas do modelo relacional:

1. **Joins complexos**: Consultas que integram múltiplas tabelas podem ser lentas.
2. **Rigidez do esquema**: Adicionar novos campos ou alterar o modelo pode ser custoso.
3. **Dependência de transações**: A garantia de ACID pode ser um gargalo em sistemas de grande escala.

Como o NoSQL ajuda:

- Armazenamento flexível de dados com esquemas dinâmicos.
- Consultas rápidas para leitura/escrita de dados densamente relacionados (e.g., MongoDB).
- Distribuição e escalabilidade nativas em modelos como Cassandra.

---

4. Coexistência de modelos relacional e NoSQL

- **Proposta híbrida**:
    - **Relacional**: Usar Postgresql para dados críticos que exigem consistência transacional, como informações de pagamentos.
    - **NoSQL**: Utilizar MongoDB ou Cassandra para dados altamente escaláveis e de leitura frequente, como histórico de compras ou conexões sociais.
- **Arquitetura**:
    - Integração via APIs intermediárias que coordenam as chamadas aos diferentes bancos.

5. Nova arquitetura proposta

A nova arquitetura combinaria os dois tipos de banco:

- **Bancos**:
    - MongoDB para dados de usuários e compras.
    - Redis para cache de dados frequentes (top 10 produtos mais vendidos).
    - Postgres para gestão de transações financeiras.
- **Infraestrutura**:
    - Uso de microservices desacoplados para cada funcionalidade.
    - Serviços de streaming, como Kafka, para sincronização entre bancos.

---

6. Problemas evitáveis com NoSQL desde o início

- Evitar gargalos em joins complexos.
- Melhor suporte para escalar horizontalmente sem necessidade de reestruturações significativas.
- Maior flexibilidade para mudanças de esquema, essencial em startups ou projetos dinâmicos.