Neo4j APOC: Casos de Uso Reais

Caso 1: Sistema de Reputação de Usuários

Problema

Manter um score de reputação que se atualiza automaticamente quando um usuário interage na plataforma.

Solução

```
-- 1. Preparação: Criar usuário com reputação inicial
CREATE (u:User {
  username: "joao_silva",
  reputation: 0,
  createdAt: datetime()
})

-- 2. Trigger: Aumentar reputação ao responder pergunta
CALL apoc.trigger.install(
  'neo4j',
  'increaseReputationOnAnswer',
  'UNWIND $createdNodes AS node
   WHERE "Answer" IN labels(node)
   MATCH (user:User)-[:CREATED]->(node)
   SET user.reputation = user.reputation + 10,
       user.lastActivityAt = datetime()',
  {phase: 'before'}
)

-- 3. Procedure: Obter usuários por reputação
CALL apoc.custom.installProcedure(
  'topUsers',
  'MATCH (u:User)
   RETURN u
   ORDER BY u.reputation DESC
   LIMIT $limit',
  'read',
  [['limit', 'INTEGER']],
  [['user', 'NODE']]
)

-- 4. Testar
CREATE (u:User {username: "joao_silva"})-[:CREATED]->(a:Answer {text:
"Resposta correta"})
MATCH (user:User {username: "joao_silva"})
RETURN user.reputation
-- Resultado: 10
```

Caso 2: E-commerce com Sincronização de Estoque

Problema

Quando um produto é vendido, o estoque deve ser atualizado e um alerta enviado se for baixo.

Solução

```
-- 1. Preparação
CREATE (p:Product {sku: "LAPTOP001", name: "Laptop", stock: 50, minStock:
5})

-- 2. Trigger: Atualizar estoque e gerar alerta
CALL apoc.trigger.install(
  'neo4j',
  'updateStockOnSale',
  'UNWIND $createdNodes AS sale
   WHERE "Sale" IN labels(sale)
   MATCH (p:Product {sku: sale.productSku})
   SET p.stock = p.stock - sale.quantity
   WITH p, sale
   WHERE p.stock <= p.minStock
   CREATE (alert:StockAlert {
     productId: p.sku,
     message: p.name + " está com estoque baixo (" + p.stock + " itens)",
     createdAt: datetime(),
     severity: "HIGH"
   })',
  {phase: 'before'}
)

-- 3. Procedure: Registrar venda
CALL apoc.custom.installProcedure(
  'registerSale',
  'CREATE (s:Sale {
     productSku: $sku,
     quantity: $qty,
     amount: $amount,
     timestamp: datetime()
   })
   RETURN s',
  'write',
  [['sku', 'STRING'], ['qty', 'INTEGER'], ['amount', 'FLOAT']],
  [['sale', 'NODE']]
)

-- 4. Testar
CALL custom.registerSale("LAPTOP001", 3, 2997.0) YIELD sale

-- Verificar estoque
MATCH (p:Product {sku: "LAPTOP001"})
RETURN p.name, p.stock
```

```
-- Verificar alertas
MATCH (alert:StockAlert {productId: "LAPTOP001"})
RETURN alert.message
```

Caso 3: Rede Social com Feed em Tempo Real

Problema

Quando um utilizador segue outro, deve receber automaticamente os posts desse utilizador no seu feed.

Solução

```
-- 1. Preparação
CREATE (u1:User {username: "alice", createdAt: datetime()})
CREATE (u2:User {username: "bob", createdAt: datetime()})

-- 2. Trigger: Adicionar posts ao feed quando alguém segue
CALL apoc.trigger.install(
  'neo4j',
  'updateFeedOnFollow',
  'UNWIND $createdRelationships AS rel
   WHERE type(rel) = "FOLLOWS"
   WITH rel.end AS follower, rel.start AS followee
   MATCH (followee)-[:POSTED]->(post:Post)
   CREATE (follower)-[:IN_FEED]->(post)',
  {phase: 'before'}
)

-- 3. Procedure: Criar post
CALL apoc.custom.installProcedure(
  'createPost',
  'CREATE (post:Post {
     content: $content,
     author: $author,
     likes: 0,
     createdAt: datetime()
   })
   WITH post
   MATCH (u:User {username: $author})
   CREATE (u)-[:POSTED]->(post)
   RETURN post',
  'write',
  [['content', 'STRING'], ['author', 'STRING']],
  [['post', 'NODE']]
)

-- 4. Procedure: Obter feed de um utilizador
CALL apoc.custom.installProcedure(
  'getUserFeed',
```

```
  'MATCH (u:User {username: $username})-[:IN_FEED]->(post)
   RETURN post
   ORDER BY post.createdAt DESC
   LIMIT $limit',
  'read',
  [['username', 'STRING'], ['limit', 'INTEGER']],
  [['post', 'NODE']]
)

-- 5. Testar
-- Alice segue Bob
MATCH (a:User {username: "alice"}), (b:User {username: "bob"})
CREATE (a)-[:FOLLOWS]->(b)

-- Bob cria um post
CALL custom.createPost("Olá mundo!", "bob") YIELD post

-- Ver feed da Alice
CALL custom.getUserFeed("alice", 10) YIELD post
RETURN post.content
```

Caso 4: Sistema de Aprovação de Documentos

Problema

Documentos passam por um fluxo de aprovação e precisam rastrear cada mudança de status.

Solução

```
-- 1. Preparação
CREATE (doc:Document {
  id: "DOC001",
  title: "Proposta Financeira",
  status: "DRAFT",
  createdAt: datetime()
})

-- 2. Trigger: Registrar alterações de status
CALL apoc.trigger.install(
  'neo4j',
  'auditStatusChange',
  'UNWIND $assignedNodeProperties["status"] AS prop
   WHERE "Document" IN labels(prop.node)
   AND prop.old <> prop.new
   CREATE (audit:AuditLog {
     docId: prop.node.id,
     previousStatus: prop.old,
     newStatus: prop.new,
     changedAt: datetime(),
     changedBy: "system"
```

```
    })',
  {phase: 'after'}
)

-- 3. Trigger: Notificar aprovadores quando documento está pronto
CALL apoc.trigger.install(
  'neo4j',
  'notifyApprovers',
  'UNWIND $assignedNodeProperties["status"] AS prop
   WHERE "Document" IN labels(prop.node)
   AND prop.new = "PENDING_APPROVAL"
   MATCH (approver:Approver {role: "MANAGER"})
   CREATE (notif:Notification {
     type: "DOCUMENT_AWAITING_APPROVAL",
     docId: prop.node.id,
     docTitle: prop.node.title,
     targetApprover: approver.username,
     createdAt: datetime()
   })',
  {phase: 'after'}
)

-- 4. Procedure: Enviar documento para aprovação
CALL apoc.custom.installProcedure(
  'submitForApproval',
  'MATCH (doc:Document {id: $docId})
   SET doc.status = "PENDING_APPROVAL",
       doc.submittedAt = datetime()
   RETURN doc',
  'write',
  [['docId', 'STRING']],
  [['document', 'NODE']]
)

-- 5. Procedure: Obter histórico de um documento
CALL apoc.custom.installProcedure(
  'getDocumentHistory',
  'MATCH (audit:AuditLog {docId: $docId})
   RETURN {
     previousStatus: audit.previousStatus,
     newStatus: audit.newStatus,
     changedAt: audit.changedAt
   } AS change
   ORDER BY change.changedAt DESC',
  'read',
  [['docId', 'STRING']],
  [['change', 'MAP']]
)

-- 6. Testar
CALL custom.submitForApproval("DOC001")

-- Ver histórico
```

```
CALL custom.getDocumentHistory("DOC001") YIELD change
RETURN change
```

Caso 5: Recomendação de Produtos com Machine Learning Ready

Problema

Rastrear interações de utilizadores com produtos para depois gerar recomendações.

Solução

```
-- 1. Preparação
CREATE (p:Product {id: "PROD001", name: "Monitor 4K", price: 399.99})
CREATE (u:User {username: "carlos", createdAt: datetime()})

-- 2. Trigger: Registrar visualizações
CALL apoc.trigger.install(
  'neo4j',
  'trackProductView',
  'UNWIND $createdRelationships AS rel
   WHERE type(rel) = "VIEWED"
   SET rel.viewedAt = datetime(),
       rel.score = 1',
  {phase: 'before'}
)

-- 3. Trigger: Aumentar score se usuário compra
CALL apoc.trigger.install(
  'neo4j',
  'increaseViewScore',
  'UNWIND $createdRelationships AS rel
   WHERE type(rel) = "PURCHASED"
   WITH rel.start AS user, rel.end AS product
   MATCH (user)-[viewed:VIEWED]->(product)
   SET viewed.score = viewed.score + 5',
  {phase: 'before'}
)

-- 4. Procedure: Registrar visualização
CALL apoc.custom.installProcedure(
  'viewProduct',
  'MATCH (u:User {username: $username}), (p:Product {id: $productId})
   MERGE (u)-[v:VIEWED]->(p)
   ON CREATE SET v.count = 1, v.firstViewedAt = datetime()
   ON MATCH SET v.count = v.count + 1, v.lastViewedAt = datetime()
   RETURN v',
  'write',
  [['username', 'STRING'], ['productId', 'STRING']],
  [['view', 'RELATIONSHIP']]
)
```

```
-- 5. Procedure: Recomendações baseadas em visualizações
CALL apoc.custom.installProcedure(
  'getRecommendations',
  'MATCH (u:User {username: $username})-[viewed:VIEWED]->(p:Product)
   WITH p.category AS category
   MATCH (other:Product {category: category})
   WHERE NOT (u)-[:VIEWED|:PURCHASED]->(other)
   RETURN other
   ORDER BY other.popularity DESC
   LIMIT $limit',
  'read',
  [['username', 'STRING'], ['limit', 'INTEGER']],
  [['recommended', 'NODE']]
)

-- 6. Testar
CALL custom.viewProduct("carlos", "PROD001")
CALL custom.viewProduct("carlos", "PROD001")
CALL custom.viewProduct("carlos", "PROD001")

-- Ver recomendações
CALL custom.getRecommendations("carlos", 5) YIELD recommended
RETURN recommended
```

Caso 6: Gestão de Versões com Histórico Completo

Problema

Manter um histórico completo de todas as versões de um documento ou configuração.

Solução

```
-- 1. Trigger: Criar snapshot de versão anterior
CALL apoc.trigger.install(
  'neo4j',
  'createVersionSnapshot',
  'UNWIND $assignedNodeProperties AS props
   UNWIND props.value AS prop
   WHERE prop.key IN ["content", "title", "config"]
   WITH prop, prop.node AS node
   WHERE "Document" IN labels(node) AND prop.old IS NOT NULL
   CREATE (snap:DocumentSnapshot {
     docId: node.id,
     version: coalesce(node.version, 1),
     field: prop.key,
     oldValue: prop.old,
     newValue: prop.new,
     changedAt: datetime()
   })
```

```
   WITH node
   SET node.version = coalesce(node.version, 1) + 1',
  {phase: 'before'}
)

-- 2. Procedure: Restaurar versão anterior
CALL apoc.custom.installProcedure(
  'revertToVersion',
  'MATCH (snap:DocumentSnapshot {docId: $docId, version: $version})
   WITH snap.docId AS docId, collect({field: snap.field, value:
snap.oldValue}) AS changes
   MATCH (doc:Document {id: docId})
   SET doc += apoc.map.fromPairs([[c.field, c.value] | c IN changes]),
       doc.version = $version
   RETURN doc',
  'write',
  [['docId', 'STRING'], ['version', 'INTEGER']],
  [['document', 'NODE']]
)

-- 3. Procedure: Listar histórico de versões
CALL apoc.custom.installProcedure(
  'getDocumentVersions',
  'MATCH (snap:DocumentSnapshot {docId: $docId})
   WITH snap.version AS version, collect({
     field: snap.field,
     oldValue: snap.oldValue,
     newValue: snap.newValue,
     changedAt: snap.changedAt
   }) AS changes
   RETURN {
     version: version,
     changes: changes
   } AS versionInfo
   ORDER BY version DESC',
  'read',
  [['docId', 'STRING']],
  [['versionInfo', 'MAP']]
)

-- 4. Testar
CREATE (d:Document {id: "DOC123", title: "Relatório Q1", version: 0})

-- Fazer alterações
MATCH (d:Document {id: "DOC123"})
SET d.title = "Relatório Q1 2025"

MATCH (d:Document {id: "DOC123"})
SET d.title = "Relatório Financeiro Q1"

-- Ver histórico
CALL custom.getDocumentVersions("DOC123") YIELD versionInfo
RETURN versionInfo
```

Caso 7: Cache de Agregações (Contadores)

Problema

Manter contadores atualizados sem fazer queries custosas de contagem.

Solução

```
-- 1. Preparação
CREATE (team:Team {name: "DevOps", memberCount: 0})

-- 2. Trigger: Incrementar contador ao adicionar membro
CALL apoc.trigger.install(
  'neo4j',
  'incrementTeamSize',
  'UNWIND $createdRelationships AS rel
   WHERE type(rel) = "MEMBER_OF"
   WITH rel.end AS team
   MATCH (t:Team)
   WHERE id(t) = id(team)
   SET t.memberCount = t.memberCount + 1',
  {phase: 'before'}
)

-- 3. Trigger: Decrementar contador ao remover membro
CALL apoc.trigger.install(
  'neo4j',
  'decrementTeamSize',
  'UNWIND $deletedRelationships AS rel
   WHERE type(rel) = "MEMBER_OF"
   WITH rel.end AS team
   MATCH (t:Team)
   WHERE id(t) = id(team)
   SET t.memberCount = max(0, t.memberCount - 1)',
  {phase: 'before'}
)

-- 4. Testar
CREATE (emp:Employee {name: "João"})
MATCH (emp:Employee {name: "João"}), (t:Team {name: "DevOps"})
CREATE (emp)-[:MEMBER_OF]->(t)

-- Verificar contador
MATCH (t:Team {name: "DevOps"})
RETURN t.memberCount
-- Resultado: 1
```

Caso 8: Validação Complexa com Regras de Negócio

Problema

Impedir criação de dados que violem regras de negócio específicas.

Solução

```
-- 1. Trigger: Validar limite de crédito
CALL apoc.trigger.install(
  'neo4j',
  'validateCreditLimit',
  'UNWIND $createdNodes AS order
   WHERE "Order" IN labels(order)
   MATCH (customer:Customer {id: order.customerId})
   WHERE customer.creditLimit < order.amount
   CALL apoc.util.validate(
     false,
     "❌ Limite de crédito excedido! Limite: " + customer.creditLimit + "
| Pedido: " + order.amount
   ) YIELD value
   RETURN value',
  {phase: 'before'}
)

-- 2. Trigger: Validar datas
CALL apoc.trigger.install(
  'neo4j',
  'validateProjectDates',
  'UNWIND $createdNodes AS proj
   WHERE "Project" IN labels(proj)
   AND date(proj.startDate) > date(proj.endDate)
   CALL apoc.util.validate(
     false,
     "❌ Data de início não pode ser posterior a data de fim!"
   ) YIELD value
   RETURN value',
  {phase: 'before'}
)

-- 3. Testar validação
CREATE (c:Customer {id: "CUST001", creditLimit: 1000})
CREATE (o:Order {customerId: "CUST001", amount: 2000})
-- Isto vai falhar com mensagem de erro
```

Checklist para Implementação em Produção

```
1. Análise
   □ Identificar eventos que precisam disparar triggers
   □ Definir ações que devem acontecer automaticamente
   □ Documentar fluxos de negócio
```

```
2. Desenvolvimento
   □ Criar triggers em ambiente de teste
   □ Testar com dados reais
   □ Verificar performance
   □ Criar procedures de suporte

3. Validação
   □ Revisar código com equipa
   □ Testar casos extremos
   □ Verificar loops infinitos
   □ Teste de carga

4. Deployment
   □ Backup da base de dados
   □ Instalar triggers em produção
   □ Monitorar logs
   □ Ter plano de rollback

5. Manutenção
   □ Monitorar performance
   □ Revisar logs regularmente
   □ Documentar mudanças
   □ Treinar equipa
```

Versão: 1.0 | **Casos de Uso Reais** | **Neo4j 5.x com APOC**