Guia Completo: Triggers e Procedures em Neo4j com APOC

Índice

Introdução

Triggers e **Procedures** são ferramentas poderosas do APOC que permitem automatizar operações em Neo4j:

- **Triggers**: Executam automaticamente quando há alterações nos dados (criar, atualizar, eliminar)
- **Procedures**: Funções reutilizáveis que podem ser chamadas a qualquer momento

---

Preparação do Ambiente

1. Instalar APOC

Opção A: Neo4j Desktop

1. Abrir Neo4j Desktop
2. Ir a "Project" > "DBMS" > "Plugins"
3. Procurar e instalar "APOC"
4. Reiniciar a base de dados

Opção B: Docker

```
docker run \
  -p 7474:7474 -p 7687:7687 \
  -e NEO4JLABS_PLUGINS='["apoc"]' \
  neo4j:latest
```

Opção C: Manualmente

1. Descarregar o ficheiro APOC da versão correspondente
2. Colocar em: `NEO4J_HOME/plugins/`
3. Reiniciar o Neo4j

2. Configurar Triggers

Criar/editar ficheiro `apoc.conf`:

```
apoc.trigger.enabled=true
apoc.trigger.refresh=60000
```

3. Verificar Instalação

```
CALL apoc.help('apoc.trigger')
```

Triggers: Estrutura Fundamental

Sintaxe Básica

```
CALL apoc.trigger.install(
  'database_name',
  'trigger_name',
  'CYPHER_QUERY_HERE',
  {phase: 'PHASE_TYPE'}
)
```

Parâmetros Disponíveis

Variável	Descrição	Tipo
`$createdNodes`	Nós criados	Lista de nós
`$createdRelationships`	Relações criadas	Lista de relações
`$deletedNodes`	Nós eliminados	Lista de nós
`$deletedRelationships`	Relações eliminadas	Lista de relações
`$assignedNodeProperties`	Propriedades atribuídas	Map
`$removedNodeProperties`	Propriedades removidas	Map
`$assignedLabels`	Labels atribuídos	Map
`$removedLabels`	Labels removidos	Map
`$transactionId`	ID da transação	Integer
`$commitTime`	Tempo de commit	Long
`$meta`	Metadados	Map

Fases de Execução

Fase	Quando Executa	Modificações
`before`	Antes de commit	✓ Pode modificar dados
`after`	Depois de commit	✗ Apenas leitura (Neo4j 5+)
`afterAsync`	Depois de commit (assíncrono)	✗ Apenas leitura
`rollback`	Quando há rollback	✗ Apenas leitura

Exemplo 1: Trigger Simples - Adicionar Timestamp

```
CALL apoc.trigger.install(
  'neo4j',
  'addTimestamp',
  'UNWIND $createdNodes AS n
   SET n.createdAt = datetime()',
  {phase: 'before'}
)
```

Teste:

```
CREATE (p:Person {name: "João"})
RETURN p
```

Exemplo 2: Trigger com Filtragem - Apenas para Label Específico

```
CALL apoc.trigger.install(
  'neo4j',
  'logMovieCreation',
  'UNWIND $createdNodes AS n
   WHERE "Movie" IN labels(n)
   CREATE (log:MovieLog {
     title: n.title,
     createdAt: datetime()
   })',
  {phase: 'before'}
)
```

Teste:

```
CREATE (m:Movie {title: "The Matrix", released: 1999})
MATCH (log:MovieLog) RETURN log
```

Exemplo 3: Trigger com Validação - Rejeitar Dados Inválidos

```
CALL apoc.trigger.install(
  'neo4j',
  'validatePrice',
  'UNWIND $assignedNodeProperties AS props
   WHERE props.key = "price" AND props.new < 0
   CALL apoc.util.validate(false, "Price cannot be negative") YIELD value
   RETURN value',
  {phase: 'before'}
)
```

Exemplo 4: Trigger com Relacionamentos - Conectar Nós Automaticamente

```
CALL apoc.trigger.install(
  'neo4j',
  'autoConnect',
  'UNWIND $createdNodes AS newNode
   WHERE "Product" IN labels(newNode)
   MATCH (cat:Category {name: newNode.category})
   CREATE (newNode)-[:BELONGS_TO]->(cat)',
  {phase: 'before'}
)
```

Exemplo 5: Trigger com Contadores - Atualizar Estatísticas

```
CALL apoc.trigger.install(
  'neo4j',
  'incrementCounter',
  'UNWIND $createdNodes AS n
   WHERE "User" IN labels(n)
   MATCH (stats:Stats {name: "UserCount"})
   SET stats.count = stats.count + 1',
  {phase: 'before'}
)
```

Procedures Customizadas

Sintaxe Básica

```
CALL apoc.custom.installProcedure(
  'procedure_name',
  'CYPHER_QUERY',
  'read|write',
```

```
  [['param_name', 'TYPE']],
  [['return_name', 'TYPE']],
  'description'
)
```

Tipos Suportados

- `STRING` - Texto
- `INTEGER` - Número inteiro
- `FLOAT` - Número decimal
- `BOOLEAN` - Verdadeiro/Falso
- `NODE` - Um nó do grafo
- `RELATIONSHIP` - Uma relação
- `MAP` - Um mapa/dicionário
- `LIST OF STRING`, `LIST OF INTEGER`, etc.

Exemplo 1: Procedure Simples - Buscar por Nome

```
CALL apoc.custom.installProcedure(
  'findPerson',
  'MATCH (p:Person {name: $name})
   RETURN p',
  'read',
  [['name', 'STRING']],
  [['p', 'NODE']]
)
```

Usar:

```
CALL custom.findPerson("João") YIELD p
RETURN p.name
```

Exemplo 2: Procedure com Múltiplos Parâmetros - Criar com Validação

```
CALL apoc.custom.installProcedure(
  'createUser',
  'MERGE (u:User {email: $email})
   ON CREATE SET u.name = $name, u.createdAt = datetime()
   RETURN u',
  'write',
  [['email', 'STRING'], ['name', 'STRING']],
  [['user', 'NODE']]
)
```

Usar:

```
CALL custom.createUser("joao@email.com", "João Silva") YIELD user
RETURN user
```

Exemplo 3: Procedure com Retorno de Múltiplos Valores

```
CALL apoc.custom.installProcedure(
  'getMovieStats',
  'MATCH (m:Movie {title: $title})
   WITH m, [(m)-[:ACTED_IN|<-:ACTED_IN]-(a) | a] AS actors
   RETURN m.title AS title,
          m.released AS year,
          size(actors) AS actorCount',
  'read',
  [['title', 'STRING']],
  [['title', 'STRING'], ['year', 'INTEGER'], ['actorCount', 'INTEGER']]
)
```

Usar:

```
CALL custom.getMovieStats("The Matrix") YIELD title, year, actorCount
RETURN *
```

Exemplo 4: Procedure Complexa - Análise Detalhada

```
CALL apoc.custom.installProcedure(
  'analyzeActor',
  'MATCH (a:Person {name: $name})-[:ACTED_IN]->(m:Movie)
   WITH a, m, m.released AS year
   RETURN {
     actor: a.name,
     movieCount: count(m),
     movies: collect(m.title),
     years: collect(year),
     avgYear: apoc.math.round(avg(year), 0),
     releaseCategory: CASE
       WHEN avg(year) < 2000 THEN "classic"
       WHEN avg(year) < 2010 THEN "modern"
       ELSE "contemporary"
     END
   } AS analysis',
  'read',
  [['name', 'STRING']],
  [['analysis', 'MAP']]
)
```

Usar:

```
CALL custom.analyzeActor("Tom Hanks") YIELD analysis
RETURN analysis
```

Exemplos Práticos Combinados

Cenário: Sistema de E-commerce

1. Trigger: Adicionar Timestamp e Slug a Produtos

```
CALL apoc.trigger.install(
  'neo4j',
  'productDefaults',
  'UNWIND $createdNodes AS n
   WHERE "Product" IN labels(n)
   SET n.createdAt = datetime(),
       n.slug = toLower(replace(n.title, " ", "-"))',
  {phase: 'before'}
)
```

2. Trigger: Atualizar Categoria quando Produto é Adicionado

```
CALL apoc.trigger.install(
  'neo4j',
  'productCategory',
  'UNWIND $createdNodes AS prod
   WHERE "Product" IN labels(prod)
   MATCH (cat:Category {id: prod.categoryId})
   SET cat.productCount = cat.productCount + 1',
  {phase: 'before'}
)
```

3. Procedure: Buscar Produtos por Preço

```
CALL apoc.custom.installProcedure(
  'getProductsByPrice',
  'MATCH (p:Product)
   WHERE p.price >= $minPrice AND p.price <= $maxPrice
   RETURN p
   ORDER BY p.price ASC',
  'read',
```

```
  [['minPrice', 'FLOAT'], ['maxPrice', 'FLOAT']],
  [['product', 'NODE']]
)
```

Usar:

```
CALL custom.getProductsByPrice(10.0, 100.0) YIELD product
RETURN product.title, product.price
```

4. Procedure: Recomendar Produtos Similares

```
CALL apoc.custom.installProcedure(
  'recommendSimilar',
  'MATCH (p:Product {id: $productId})-[:IN_CATEGORY]->(cat)
   MATCH (other:Product)-[:IN_CATEGORY]->(cat)
   WHERE other.id <> p.id
   AND abs(other.price - p.price) < (p.price * 0.2)
   RETURN other
   LIMIT 5',
  'read',
  [['productId', 'STRING']],
  [['recommended', 'NODE']]
)
```

Boas Práticas

✓ O que Fazer

1. **Use Fases Apropriadas**

   ```
   -- Para modificações: use 'before'
   -- Para logs/audits: use 'afterAsync'
   ```

2. **Sempre Filtre por Label**

   ```
   UNWIND $createdNodes AS n
   WHERE "MyLabel" IN labels(n)
   -- seu código aqui
   ```

3. **Evite Loops Infinitos**

```
-- Adicionar condição para evitar recursão
WHERE NOT 'processado' IN apoc.node.labels(n)
SET n.processado = true
```

4. **Use Nomes Descritivos**

```
-- Bom
'userCreationLogging'

-- Ruim
'trig1'
```

5. **Teste Antes de Implantar**

```
-- Sempre teste em desenvolvimento primeiro
CREATE (test:Test {name: "test"})
-- Verificar se o trigger funcionou
MATCH (log:TestLog) RETURN log
```

✗ O que Evitar

1. **Não Crie Triggers Sem Condições**

```
-- Ruim: afeta TUDO
UNWIND $createdNodes AS n

-- Bom: apenas nodes específicos
UNWIND $createdNodes AS n
WHERE "Product" IN labels(n)
```

2. **Não Modifique Dados em Fase 'after'**

```
-- Erro em Neo4j 5+
SET n.newProperty = "value"  -- Não funciona em 'after'
```

3. **Não Use Queries Muito Complexas**

   - Triggers devem ser rápidos
   - Mantenha lógica simples

4. **Não Ignore Tratamento de Erros**

```
```

```
CALL apoc.util.validate(
  condition,
  'mensagem de erro'
)
```

Operações Úteis

Listar Triggers

```
CALL apoc.trigger.show('neo4j')
```

Parar um Trigger

```
CALL apoc.trigger.stop('neo4j', 'trigger_name')
```

Retomar um Trigger

```
CALL apoc.trigger.start('neo4j', 'trigger_name')
```

Eliminar um Trigger

```
CALL apoc.trigger.drop('neo4j', 'trigger_name')
```

Eliminar Todos os Triggers

```
CALL apoc.trigger.dropAll('neo4j')
```

Listar Procedures Customizadas

```
CALL apoc.custom.list()
```

Troubleshooting

Problema: Trigger não dispara

Solução:

- Verificar se `apoc.trigger.enabled=true` em `apoc.conf`
- Reiniciar Neo4j
- Verificar fase correta

Problema: Loop infinito

Solução:

- Adicionar condição de parada
- Usar flag ou propriedade para marcar processamento

Problema: "Phase 'after' not allowed"

Solução:

- Em Neo4j 5+, não é permitido modificar dados em fase 'after'
- Usar fase 'before' ou 'afterAsync' apenas para leitura

Problema: Procedure não aparece

Solução:

- Verificar se APOC está instalado: `CALL apoc.help('')`
- Reiniciar Neo4j após instalar APOC
- Usar namespace correto: `CALL custom.procedure_name()`

Resumo: Fluxo Completo

```
1. Instalar APOC
   ↓
2. Configurar apoc.conf
   ↓
3. Reiniciar Neo4j
   ↓
4. Criar Triggers (se necessário)
   ↓
5. Criar Procedures Customizadas
   ↓
6. Testar cada componente
   ↓
7. Monitorar em produção
```

Versão: 1.0 | **Última Atualização**: 2025 | **Neo4j 5.x**