Neo4j APOC: Referência Rápida e Troubleshooting

Referência Rápida - Cheat Sheet

Sintaxe Triggers

```
-- Criar trigger
CALL apoc.trigger.install(
  'neo4j',                          -- nome da BD
  'trigger_name',                   -- nome do trigger
  'CYPHER QUERY WITH $variables', -- código Cypher
  {phase: 'before'}                 -- fase de execução
)

-- Listar triggers
CALL apoc.trigger.show('neo4j')

-- Parar trigger
CALL apoc.trigger.stop('neo4j', 'trigger_name')

-- Retomar trigger
CALL apoc.trigger.start('neo4j', 'trigger_name')

-- Eliminar trigger
CALL apoc.trigger.drop('neo4j', 'trigger_name')

-- Eliminar tudo
CALL apoc.trigger.dropAll('neo4j')
```

Sintaxe Procedures

```
-- Criar procedure
CALL apoc.custom.installProcedure(
  'procedure_name',
  'MATCH ... RETURN ...',
  'read|write',
  [['param1', 'STRING'], ['param2', 'INTEGER']],
  [['result', 'NODE']],
  'Descrição opcional'
)

-- Chamar procedure
CALL custom.procedure_name(param1, param2) YIELD result
RETURN result

-- Listar procedures
CALL apoc.custom.list()
```

```
-- Eliminar procedure
CALL apoc.custom.deleteFunction('procedure_name')
```

Variáveis de Trigger Disponíveis

Variável	Usado em	Conteúdo
`$createdNodes`	CREATE	Nós criados
`$createdRelationships`	CREATE	Relações criadas
`$deletedNodes`	DELETE	Nós eliminados
`$deletedRelationships`	DELETE	Relações eliminadas
`$assignedNodeProperties`	SET	Propriedades atribuídas
`$removedNodeProperties`	REMOVE	Propriedades removidas
`$assignedLabels`	ADD	Labels atribuídos
`$removedLabels`	REMOVE	Labels removidos
`$transactionId`	--	ID da transação
`$commitTime`	--	Tempo do commit (ms)
`$meta`	--	Metadados da transação

Fases de Execução

Fase	Quando	Modificável	Uso
`before`	Antes do commit	✓ SIM	Validação, modificação de dados
`after`	Depois do commit	✗ NÃO	Logs, audits (leitura)
`afterAsync`	Depois (assíncrono)	✗ NÃO	Logs pesados, notificações
`rollback`	Se houver erro	✗ NÃO	Limpeza, rollback

Tipos de Parâmetros

```
STRING       -- Texto
INTEGER      -- Número inteiro
FLOAT        -- Número decimal
BOOLEAN      -- true/false
NODE         -- Um nó do grafo
RELATIONSHIP -- Uma relação
MAP          -- {chave: valor}
LIST OF STRING, LIST OF INTEGER, etc.
```

Padrões Comuns

Padrão 1: Adicionar Timestamp em Todas as Criações

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

Padrão 2: Incremental ID Counter

```
-- Criar node contador
CREATE (counter:Counter {lastId: 0})

-- Trigger para incrementar
CALL apoc.trigger.install(
  'neo4j',
  'incrementId',
  'UNWIND $createdNodes AS n
   WHERE "Document" IN labels(n)
   MATCH (c:Counter)
   SET c.lastId = c.lastId + 1,
       n.docId = c.lastId',
  {phase: 'before'}
)
```

Padrão 3: Audit Trail (Log de Alterações)

```
CALL apoc.trigger.install(
  'neo4j',
  'auditTrail',
  'UNWIND $assignedNodeProperties["status"] AS prop
   CREATE (audit:AuditLog {
     entityId: id(prop.node),
     oldValue: prop.old,
     newValue: prop.new,
     changedAt: datetime()
   })',
  {phase: 'after'}
)
```

Padrão 4: Prevenir Loops Infinitos

```
CALL apoc.trigger.install(
  'neo4j',
  'safeProcessing',
  'UNWIND $createdNodes AS n
   WHERE "Task" IN labels(n) AND NOT "Processed" IN labels(n)
   SET n:Processed
   -- processamento aqui
   ',
  {phase: 'before'}
)
```

Padrão 5: Busca Simples com Procedure

```
CALL apoc.custom.installProcedure(
  'findByName',
  'MATCH (n {name: $name})
   RETURN n',
  'read',
  [['name', 'STRING']],
  [['node', 'NODE']]
)
```

Padrão 6: Agregação com Procedure

```
CALL apoc.custom.installProcedure(
  'stats',
  'MATCH (n:Item)
   RETURN {
     total: count(n),
     avgPrice: apoc.math.round(avg(n.price), 2),
     maxPrice: max(n.price)
   } AS summary',
  'read',
  [],
  [['summary', 'MAP']]
)
```

Troubleshooting e Erros Comuns

❌ ERRO: "Phase 'after' not allowed to write"

Problema:

```
CALL apoc.trigger.install(
  'neo4j',
```

```
  'bad',
  'UNWIND $createdNodes AS n
   CREATE (log:Log)',  -- ❌ Criar é escrita em 'after'!
  {phase: 'after'}
)
```

Solução:

```
-- Usar fase 'before'
{phase: 'before'}
-- OU usar fase 'afterAsync' para leitura apenas
{phase: 'afterAsync'}
```

❌ ERRO: "Trigger not executing"

Verificar:

1. **Triggers habilitados?**

```
-- Verificar apoc.conf:
-- apoc.trigger.enabled=true

-- Testar se APOC funciona:
CALL apoc.help('') YIELD name
RETURN count(name)
```

2. **Reiniciar depois de configurar apoc.conf**

```
# Docker
docker restart neo4j-container

# Desktop: DBMS > Restart
```

3. **Verificar se trigger existe**

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

4. **Verificar se está pausado**

```
-- Se paused = true, retomar:
CALL apoc.trigger.start('neo4j', 'trigger_name')
```

❌ ERRO: "Loop infinito - processo pendurado"

Problema:

```
UNWIND $createdNodes AS n
CREATE (log:Log)  -- Log também dispara trigger!
```

Solução 1: Usar label diferente

```
UNWIND $createdNodes AS n
WHERE "Document" IN labels(n)
CREATE (audit:AuditLog)  -- AuditLog não tem trigger
```

Solução 2: Usar flag de controle

```
UNWIND $createdNodes AS n
WHERE "Document" IN labels(n) AND NOT n.processed
SET n.processed = true,
    n.timestamp = datetime()
-- resto do código...
```

❌ ERRO: "Property cannot be accessed by cypher"

Problema:

```
UNWIND $deletedNodes AS n
SET n.reason = "deleted"  -- ❌ Não pode modificar deleted!
```

Solução: Usar apoc.any.properties()

```
UNWIND $deletedNodes AS n
WITH apoc.any.properties(n) AS props
CREATE (archive:Archive)
SET archive += props
```

❌ ERRO: "Procedure not found"

Problema:

```
CALL myProcedure("test")  -- ❌ Falta 'custom.' no namespace
```

Solução:

```
CALL custom.myProcedure("test") YIELD result
RETURN result
```

❌ ERRO: "apoc.custom.installProcedure not found"

Problema: Versão antiga de APOC ou APOC Extended não instalado

Solução:

1. Verificar versão: `CALL apoc.version()`
2. Descarregar APOC Extended correspondente
3. Colocar em plugins folder
4. Reiniciar

Dicas de Performance

1️⃣ Usar UNWIND para Processar Listas

```
-- ❌ Lento
MATCH (nodes:Node)
RETURN nodes

-- ✅ Rápido
UNWIND $createdNodes AS n
RETURN n
```

2️⃣ Adicionar Índices

```
-- Indexar propriedades frequentemente consultadas
CREATE INDEX ON :User(email)
CREATE INDEX ON :Product(sku)

-- Verificar índices
SHOW INDEXES
```

3️⃣ Limitar Operações em Triggers

```
-- ❌ Pesado
UNWIND $createdNodes AS n
MATCH (other:Product)
CREATE (n)-[:RELATED_TO]->(other)

-- ✅ Melhor
UNWIND $createdNodes AS n
MATCH (cat:Category {name: n.category})
CREATE (n)-[:IN_CATEGORY]->(cat)
```

4️⃣ Usar Fase afterAsync para Operações Pesadas

```
CALL apoc.trigger.install(
  'neo4j',
  'heavyProcessing',
  'UNWIND $createdNodes AS n
   -- operações pesadas aqui',
  {phase: 'afterAsync'}  -- não bloqueia outros commits
)
```

Checklist de Implementação

- ☐ APOC instalado e funcionando
- ☐ `apoc.trigger.enabled=true` em apoc.conf
- ☐ Neo4j reiniciado após config
- ☐ Trigger testado com dados pequenos
- ☐ Nenhum loop infinito detectado
- ☐ Fase correta escolhida (before/after/afterAsync)
- ☐ Labels/tipos filtrados corretamente
- ☐ Mensagens de erro claras documentadas
- ☐ Procedure testada com vários inputs
- ☐ Performance aceitável (< 1 segundo)

Comandos Úteis de Monitoramento

Ver todos os triggers ativos

```
CALL apoc.trigger.show('neo4j')
RETURN name, selector, paused, installed
```

Ver todas as procedures

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

Contar quantidade de logs/audits

```
MATCH (log:AuditLog)
RETURN count(log) AS totalLogs,
       min(log.changedAt) AS firstChange,
       max(log.changedAt) AS lastChange
```

Verificar triggers pausados

```
CALL apoc.trigger.show('neo4j')
WHERE paused = true
RETURN name
```

Exemplo Completo: Blog com Comments

1. Preparação

```
CREATE (blog:Blog {name: "MyBlog", postCount: 0})
CREATE (stats:Stats {type: "BlogStats", commentCount: 0})
```

2. Triggers

```
-- Trigger 1: Auto-timestamp em posts
CALL apoc.trigger.install(
  'neo4j',
  'blogPostTimestamp',
  'UNWIND $createdNodes AS n
   WHERE "BlogPost" IN labels(n)
   SET n.createdAt = datetime()',
  {phase: 'before'}
)

-- Trigger 2: Contar comentários
CALL apoc.trigger.install(
  'neo4j',
  'countComments',
  'UNWIND $createdNodes AS n
   WHERE "Comment" IN labels(n)
   MATCH (stats:Stats {type: "BlogStats"})
   SET stats.commentCount = stats.commentCount + 1',
```

```
  {phase: 'before'}
)
```

3. Procedures

```
-- Procedure 1: Criar post
CALL apoc.custom.installProcedure(
  'createPost',
  'CREATE (p:BlogPost {title: $title, content: $content, author: $author})
   RETURN p',
  'write',
  [['title', 'STRING'], ['content', 'STRING'], ['author', 'STRING']],
  [['post', 'NODE']]
)

-- Procedure 2: Adicionar comentário
CALL apoc.custom.installProcedure(
  'addComment',
  'MATCH (p:BlogPost {title: $postTitle})
   CREATE (c:Comment {text: $text, author: $author})-[:ON]->(p)
   RETURN c',
  'write',
  [['postTitle', 'STRING'], ['text', 'STRING'], ['author', 'STRING']],
  [['comment', 'NODE']]
)
```

4. Uso

```
-- Criar posts
CALL custom.createPost("Aprender Neo4j", "Conteúdo...", "João") YIELD post

-- Adicionar comentários
CALL custom.addComment("Aprender Neo4j", "Muito bom!", "Maria") YIELD
comment

-- Ver estatísticas
MATCH (stats:Stats {type: "BlogStats"})
RETURN stats
```

Versão: 3.0 | **Referência Rápida** | **Neo4j 5.x**