Apuntes de Neo4j – Ampliación de Bases de Datos

> **Objetivo** → Tener reunida en un solo sitio toda la teoría esencial de grafos + patrones Cypher + ejercicios resueltos semejantes a los que suelen caer en los exámenes y la práctica logística LocalStore.

---

1. Fundamentos del modelo de grafos

- **Nodo** = entidad.
- **Relación** = arista dirigida entre nodos, siempre tiene **tipo** y puede tener propiedades.
- **Propiedades**: pares clave-valor en nodos y relaciones (números, texto, boolean, listas).
- **Etiquetas** (labels) clasifican nodos para consultas rápidas e índices.

```
cypher (:Ciudad {nombre:'Madrid', poblacion:3.3})
(p:Persona {nombre:'Ana'})-[:AMIGO_DESDE {desde:2015}]-
>(q:Persona)
```

---

2. CRUD básico en Cypher

| Operación | Sintaxis mínima | Ejemplo | |-----------|-----------------|---------| | Crear nodo |
```
CREATE (n:Label {prop: val})
```
|
```
CREATE (:Ciudad {nombre:'Sevilla'})
```
| | Crear relación |
```
MATCH ... CREATE (a)-[:REL]->(b)
```
|
```
MATCH (a:Ciudad {nombre:'Madrid'}), (b:Ciudad
{nombre:'Sevilla'}) CREATE (a)-[:CARRETERA]->(b)
```
| | Leer |
```
MATCH patrón RETURN …
```
|
```
MATCH (c:Ciudad) RETURN c.nombre
```
| | Actualizar |
```
MATCH … SET n.prop = val
```

|

```
MATCH (c:Ciudad) SET c.tipo='provincia'
```

| | Borrar |

```
MATCH (n) DETACH DELETE n
```

|

```
MATCH (:Ciudad {nombre:'Valencia'}) DETACH DELETE n
```

|

2.1 Crear varios nodos en una sola consulta (examen)

```
cypher CREATE (:Ciudad {nombre:'Madrid'}), (:Ciudad
{nombre:'Valencia'}), (:Ciudad {nombre:'Sevilla'}),
(:Ciudad {nombre:'Barcelona'});
```

2.2 Añadir atributo a todas las ciudades (examen – solución correcta)

```
cypher MATCH (c:Ciudad) SET c.tipo = 'provincia';
```

2.3 Crear carreteras entre cada par de ciudades (sin bucles)

```
cypher MATCH (c1:Ciudad), (c2:Ciudad) WHERE id(c1) <
id(c2) CREATE (c1)-[:CARRETERA]->(c2);
```

> Si necesitas ambos sentidos: repite con
> `MERGE (c1)-[:CARRETERA_BIDIR]-(c2)`
> o usa APOC
> `apoc.create.relationship`
> .

---

3. Consultas de lectura avanzadas

- **Patrones de longitud variable**:
  `(a)-[:CARRETERA*1..3]->(b)`
- **Funciones de ruta**:
  `shortestPath`
  ,
  `allShortestPaths`
  ,

`length`, `nodes()`, `relationships()`.

- **Agregación**: `COUNT(*)`, `collect()`, `avg()`, etc.
- **PIPELINE** con `WITH` y `UNWIND` para pasos intermedios.

Ejemplo – ciudades alcanzables en ≤2 saltos:

```
cypher MATCH (m:Ciudad {nombre:'Madrid'})-[:CARRETERA*1..2]->(c) RETURN DISTINCT c.nombre;
```

---

4. Índices y restricciones

| Instrucción | Propósito | Ejemplo | |---|---|---| | `CREATE INDEX` | Acelera búsquedas por propiedad | `CREATE INDEX ciudad_nombre IF NOT EXISTS FOR (c:Ciudad) ON (c.nombre)` | | `DROP INDEX` | Eliminar índice | `DROP INDEX ciudad_nombre` | | `CREATE CONSTRAINT` | Único, existencia, punto |

```
CREATE CONSTRAINT unico_nombre_ciudad IF NOT EXISTS FOR
(c:Ciudad) REQUIRE c.nombre IS UNIQUE
```
|

---

5. Operaciones masivas y utilidades

| Comando | Uso | Ejemplo | |---------|-----|---------| |

```
MERGE
```
| Crea si no existe |

```
MERGE (c:Ciudad {nombre:'Bilbao'})
```
| |

```
FOREACH
```
| Actualizar colección |

```
MATCH p=(:Ciudad)-[*]->(:Ciudad) FOREACH (n IN nodes(p)|
SET n.visitado=true)
```
| |

```
LOAD CSV
```
| Importar datos |

```
LOAD CSV WITH HEADERS FROM 'file:///ciudades.csv' AS row
CREATE (:Ciudad {nombre:row.nombre})
```
| |

```
apoc.periodic.iterate
```
| Batch processing grandes |

```
CALL apoc.periodic.iterate('MATCH (c:Ciudad) RETURN
c','SET c.index = id(c)',{batchSize:1000})
```
|

---

6. Práctica LocalStore logística – modelo sugerido

```
cypher (:Almacen {id:'MAD'}) (:Plataforma {id:'ZAR'})
(:Destino {id:'VLC'}) (:Almacen)-[:RUTA {km:315,
t_transporte:'Carretera', tiempo:190, coste:3.15}]-
>(:Destino)
```

Consultas frecuentes

```
cypher MATCH p=(a:Almacen {id:'MAD'})-[r:RUTA*]-
>(d:Destino {id:'VLC'}) WITH p, reduce(t=0, rel IN r |
t+rel.tiempo) AS tiempo, reduce(c=0, rel IN r |
```

```
c+rel.coste) AS coste WHERE tiempo < 60*14 RETURN p,
tiempo, coste ORDER BY coste LIMIT 1;
```

---

7. Ejercicios de preparación para el examen

... [reinsertar si necesario] ...

---

8. Buenas prácticas

- Crear índices antes de cargar datos masivos.
- Usar MERGE para evitar duplicados.
- Proyectar sólo lo necesario.
- Usar propiedades únicas para identificar nodos.
- Revisar planes con PROFILE.

¡Listo! Con estos apuntes y ejercicios estás preparado para cualquier examen de Neo4j.

Apuntes de Neo4j – Ampliación de Bases de Datos

...[contenido original conservado]...

---

9. Ejercicios adicionales estilo examen (ampliación)

9.1 Operaciones de creación y modificación

```
// Crear 3 personas con sus edades
CREATE (:Persona {nombre:'Ana', edad:30}),
       (:Persona {nombre:'Luis', edad:25}),
       (:Persona {nombre:'Marta', edad:40});

// Añadir campo 'activo' a todas las personas
MATCH (p:Persona)
SET p.activo = true;

// Eliminar campo 'activo'
MATCH (p:Persona)
REMOVE p.activo;
```

9.2 Relaciones y patrones

```
// Relación AMIGO entre personas mayores de 18
MATCH (a:Persona), (b:Persona)
WHERE a <> b AND a.edad > 18 AND b.edad > 18
MERGE (a)-[:AMIGO]->(b);

// Crear una relación bidireccional de colaboración
MATCH (a:Persona {nombre:'Ana'}), (b:Persona {nombre:'Luis'})
MERGE (a)-[:COLABORA]-(b);

// Crear relación con propiedades
MATCH (a:Persona {nombre:'Marta'}), (b:Persona {nombre:'Luis'})
CREATE (a)-[:CONOCIDO_DESDE {año:2010, evento:'Universidad'}]->(b);
```

9.3 Consultas con filtros y funciones

```
// Personas mayores de 30
MATCH (p:Persona)
WHERE p.edad > 30
RETURN p.nombre;
```

```
// Personas cuyo nombre empieza por 'A'
MATCH (p:Persona)
WHERE p.nombre STARTS WITH 'A'
RETURN p;

// Número de relaciones por persona
MATCH (p:Persona)-[r]->()
RETURN p.nombre, COUNT(r) AS numRelaciones;
```

9.4 Consultas con path y agregaciones

```
// Caminos hasta 3 saltos
MATCH path=(a:Ciudad {nombre:'Madrid'})-[:CARRETERA*1..3]->(b:Ciudad)
RETURN b.nombre, length(path) AS hops;

// Ruta más corta y coste total
MATCH p=shortestPath((a:Ciudad {nombre:'Madrid'})-[:CARRETERA*]-(b:Ciudad
{nombre:'Barcelona'}))
RETURN p, reduce(c=0, rel IN relationships(p) | c + rel.coste) AS costeTotal;
```

9.5 Agregaciones avanzadas

```
// Media de edad por tipo de persona
MATCH (p:Persona)
RETURN labels(p)[0] AS tipo, AVG(p.edad) AS mediaEdad;

// Nodos más conectados
MATCH (n)-[r]-()
RETURN n.nombre AS entidad, COUNT(r) AS conexiones
ORDER BY conexiones DESC LIMIT 5;
```

9.6 Actualizaciones condicionales

```
// Si edad > 30 → añadir etiqueta 'Senior'
MATCH (p:Persona)
WHERE p.edad > 30
SET p:Senior;

// Eliminar nodos sin relaciones
MATCH (n)
WHERE NOT (n)--()
DELETE n;
```

9.7 Importaciones y APOC

```
// Cargar nodos desde CSV
LOAD CSV WITH HEADERS FROM 'file:///datos.csv' AS row
CREATE (:Ciudad {nombre: row.nombre, poblacion: toInteger(row.poblacion)});

// Crear relaciones desde CSV
LOAD CSV WITH HEADERS FROM 'file:///rutas.csv' AS row
MATCH (a:Ciudad {nombre: row.origen}), (b:Ciudad {nombre: row.destino})
CREATE (a)-[:CARRETERA {km: toInteger(row.km)}]->(b);

// Usar APOC para iterar y modificar
CALL apoc.periodic.iterate(
  'MATCH (n:Persona)',
  'SET n.hash = apoc.util.md5(n.nombre)',
  {batchSize:1000}
);
```

Estos ejemplos amplían aún más el espectro de preguntas que podrías encontrar en el examen: desde consultas con filtros y relaciones hasta importación de datos, uso de APOC y caminos con funciones de agregación. Practícalos y adáptalos a tus casos reales o a los ejercicios propuestos por tu profesor.