

## 001-generic-relationship-1

chapter01\$

```
MATCH (n {id:"0eLgWXRdFvQWb5Ch6Hm33Z"})-[:HAS_TRACK]->(t:Track)
      -[:ARTIST]->(a:Artist) //this n matches an album
RETURN t.name AS trackName, a.name AS artistName
```

```
1 MATCH (n {id:"0eLgWXRdFvQWb5Ch6Hm33Z"})-[:HAS_TRACK]->(t:Track)
2      -[:ARTIST]->(a:Artist) //this n matches an album
3 RETURN t.name AS trackName, a.name AS artistName
```

Table RAW

trackName	artistName
1 "Always"	"Bon Jovi"

## 002-generic-relationship-2

chapter01\$

```
MATCH (t:Track {id: "0kish3Tobj6Wq0we74343q"})<[:-HAS_TRACK]-(n)
RETURN n.name, labels(n)
```

● Instance: neo4j://localhost:7687 ▾ Database: chapter01 CYPHER 5 ▾ User: neo4j ▾

**Database information**

Nodes (127,932)  
Album Artist Playlist Track User

Relationships (233,263)  
ARTIST HAS\_TRACK OWNS SIMILAR

Property keys  
id name notSamePosition position  
samePosition uri

chapter01\$

```
1 MATCH (t:Track {id: "0kish3Tobj6Wq0we74343q"})<[:-HAS_TRACK]-(n)
2 RETURN n.name, labels(n)
```

Table RAW

n.name	labels(n)
1 "Cross Road"	["Album"]
2 "Road Trip Punt a Fuego"	["Playlist"]

Started streaming 2 rec

## 003-generic-relationship-3

chapter01\$

```
MATCH (t:Track {id:"15vzANxN8G9wWfwAJLLMCg"})<[:-HAS_TRACK]-(a:Album)
RETURN t,a
```

Instance: neo4j://localhost:7687 Database: chapter01 CYPHER 5 User: neo4j

### Database information

Nodes (127,932)

- \*  Album
- Artist
- Playlist
- Track
- User

Relationships (233,263)

- \*  ARTIST
- HAS\_TRACK
- OWNS
- SIMILAR

Property keys

- id
- name
- notSamePosition
- position
  
- samePosition
- uri

```
chapter01$
```

```
1 MATCH (t:Track {id: "0kish3Tobj6Wq0we74343q"})->[:HAS_TRACK]-(n)
2 RETURN n.name, labels(n)
```

Table
RAW

n.name	labels(n)
1 "Cross Road"	["Album"]
2 "Road Trip Punt a Fuego"	["Playlist"]

Started 8ms ago

## 004-find-similar

chapter01\$

```
MATCH (a:Playlist)-[:SIMILAR]->(b:Playlist)
RETURN a,b
```

Instance: neo4j://localhost:7687 Database: chapter01 CYPHER 5 User: neo4j

### Database information

Nodes (127,932)

- \*  Album
- Artist
- Playlist
- Track
- User

Relationships (233,263)

- \*  ARTIST
- HAS\_TRACK
- OWNS
- SIMILAR

Property keys

- id
- name
- notSamePosition
- position
  
- samePosition
- uri

```
chapter01$
```

```
1 MATCH (a:Playlist)-[:SIMILAR]->(b:Playlist)
2 RETURN a,b
```

Graph
Table
RAW

Results overview

Nodes (82)

\* (82)

Playlist (82)

Started streaming 616 records after 2 ms ago

## **005-double-traversal**

chapter01\$

```
MATCH path=(track1:Track {id:"0BB9eUBBaaX6GALSYNCep7"})-[*3]-(track2:Track {id:"2KmEgiY8fQs0G6WNxtzQKr"})
RETURN path
```

- `[*]` means "any number of relationships/hops"
- `[*3]` means "exactly 3 relationships/hops"
- So the pattern will match paths where there are exactly 3 relationships between `track1` and `track2`

**What does this query do:**

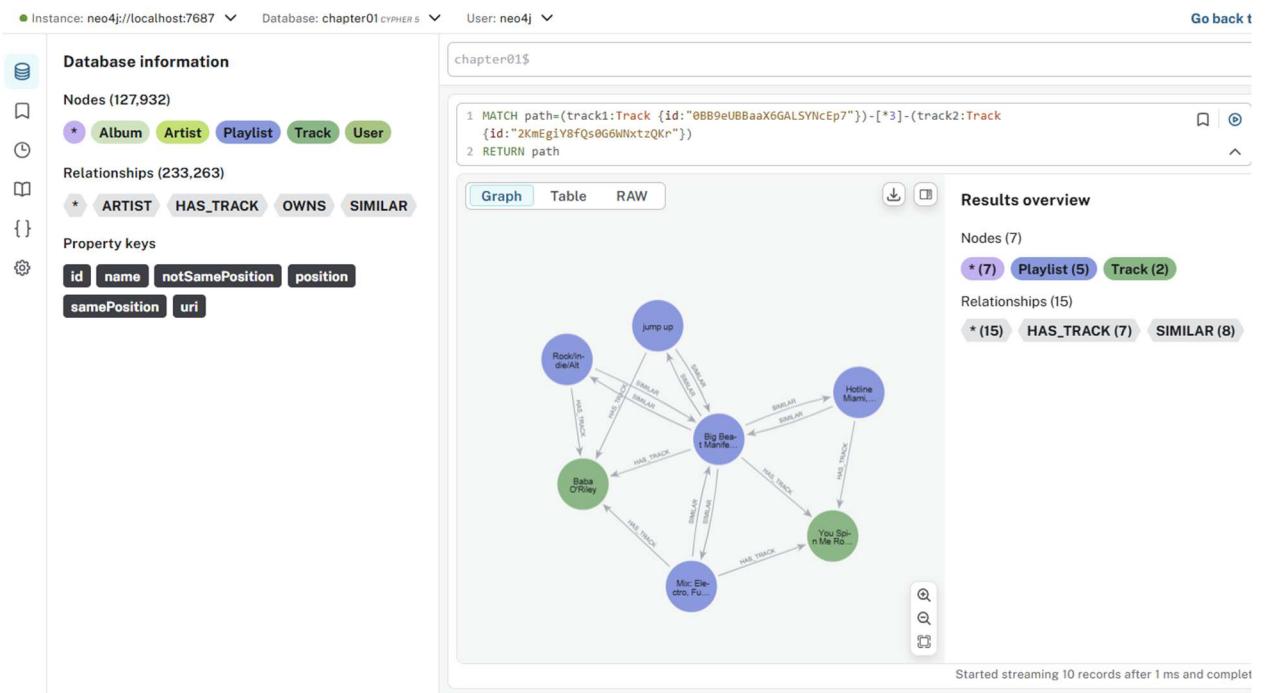
- Finds all paths that connect these two specific tracks
- Each path must have exactly 3 relationships (edges) between them
- The path could go through other nodes (likely other tracks, artists, albums, etc.)

**Other examples of variable-length relationships:**

- `[*1..3]` = 1 to 3 hops
- `[..5]` = up to 5 hops
- `[*3..]` = 3 or more hops
- `[*]` = any number of hops

In your music graph context, this might find connections like:

- `Track1 → Artist → Album → Track2`
- `Track1 → Genre → Playlist → Track2`
- Or any other 3-step path through the graph structure



## 006-drop-similar

chapter01\$

```
MATCH ()-[r:SIMILAR]-()
DELETE r
```

The screenshot shows the Neo4j browser interface. On the left, there's a sidebar with 'Database information' showing 127,932 nodes and 232,955 relationships. A 'SIMILAR' relationship type is highlighted. The main panel shows the command 'chapter01\$ MATCH ()-[r:SIMILAR]-() DELETE r' entered into the query field. Below it, a message indicates 'Deleted 308 relationships'.

## 007-single-similar-rel

chapter01\$

```
MATCH path=(p:Playlist)-[r1:HAS_TRACK]-(track)<-[r2:HAS_TRACK]-(other:Playlist)
WITH p AS playlistLeft, other AS playlistRight, collect({track: track, positionLeft: r1.position, positionRight: r2.position}) AS commonTracks
WHERE size(commonTracks) > 5
WITH playlistLeft, playlistRight, size([track in commonTracks WHERE track.positionLeft = track.positionRight]) AS tracksWithSamePosition,
size([track in commonTracks WHERE NOT track.positionLeft = track.positionRight]) AS tracksAtDifferentPosition
MERGE (playlistLeft)-[r:SIMILAR]-(playlistRight)
SET r.samePosition = tracksWithSamePosition, r.notSamePosition = tracksAtDifferentPosition
```

The screenshot shows the Neo4j browser interface. On the left, there's a sidebar with 'Database information' showing 127,932 nodes and 233,109 relationships. A 'SIMILAR' relationship type is highlighted. The main panel shows the complex Cypher query for creating single similar relationships between playlists based on shared tracks. A message at the bottom indicates 'Created 154 relationships, set 616 properties'.