

## 001-create-database

neo4j\$ CREATE DATABASE testload WAIT;

The screenshot shows the Neo4j Browser interface. On the left, the 'Database information' sidebar is visible, showing 'Nodes (0)', 'Relationships (0)', and 'Property keys' with a 'HAS\_TRACK' relationship. The main panel displays the command 'neo4j\$ CREATE DATABASE testload WAIT;' and its execution results in a table format.

address	state	message	success
"localhost:7687"	"CaughtUp"	"caught up"	true

Below the table, it indicates '1 system update' and 'Started streaming 1 record after 9,486 ms and completed after 9,487 ms.'

## 002-create-constraint

testload\$  
CREATE CONSTRAINT FOR (n:Track) REQUIRE n.id IS NODE KEY;

The screenshot shows the Neo4j Browser interface. The left sidebar shows 'Nodes (0)', 'Relationships (0)', and 'Property keys' with an 'id' property key. The main panel displays the command 'testload\$ CREATE CONSTRAINT FOR (n:Track) REQUIRE n.id IS NODE KEY;' and its execution results, indicating 'Added 1 constraint' and 'Completed after 118 ms'. Below this, the 'SHOW DATABASES;' command is also shown, displaying a table of databases.

name	type	aliases	access	address	role	writer	requestedStatus
"chapter01"	"standard"	[""]	"read-write"	"localhost:7687"	"primary"	true	"online"
"neo4j"	"standard"	[""]	"read-write"	"localhost:7687"	"primary"	true	"online"
"system"	"system"	[""]	"read-write"	"localhost:7687"	"primary"	true	"online"
"testload"	"standard"	[""]	"read-write"	"localhost:7687"	"primary"	true	"online"

## 003-load-tracks

testload\$  
LOAD CSV WITH HEADERS FROM "file:///medium/sample\_tracks\_medium.csv" AS row  
MERGE (track:Track {id: row.track\_id})  
SET track.uri = row.track\_uri,  
track.name = row.track\_name;

Instance: neo4j://localhost:7687 Database: testload\_cypher User: neo4j

Go back to old Browser

Database information

Nodes (69,878)

Track

Relationships (0)

Property keys

id name uri

testload\$

```
1 LOAD CSV WITH HEADERS FROM "file:///medium/sample_tracks_medium.csv" AS row
2 MERGE (track:Track {id: row.track_id})
3 SET track.uri = row.track_uri,
4   track.name = row.track_name;
```

Created 69,878 nodes, set 257,724 properties, added 69,878 labels

Completed after 4.732 ms

testload\$ CREATE CONSTRAINT FOR (n:Track) REQUIRE n.id IS NODE KEY;

Added 1 constraint

Completed after 118 ms

neo4j\$ SHOW DATABASES;

Instance: neo4j://localhost:7687 Database: testload\_cypher User: neo4j

Go back to old Browser

Database information

Nodes (69,878)

Track

Relationships (0)

Property keys

id name uri

testload\$

testload\$ MATCH (n) RETURN n ;

Graph Table RAW

Results overview

Nodes (1,000)

Initial display limit hit at 1,000 nodes. [Edit settings](#)

(1000) Track (1000)

Fetch limit hit at 5,000 records. Started streaming after 25 ms and completed after 105 ms.

## 004-call-in-transactions

testload\$

```
LOAD CSV WITH HEADERS FROM "file:///medium/sample_tracks_medium.csv" AS row
CALL(row) {
  MERGE (track:Track {id: row.track_id})
  SET track.uri = row.track_uri,
    track.name = row.track_name
} IN TRANSACTIONS OF 10_000 ROWS;
```

// CALL { ... }: Executes a subquery for each row

// IN TRANSACTIONS OF 10\_000 ROWS: Processes in batches of 10,000 rows

Instance: neo4j://localhost:7687 Database: testload cypher User: neo4j

**Database information**

Nodes (69,878)

- Track

Relationships (0)

Property keys

- id name uri

testload\$

```
1 LOAD CSV WITH HEADERS FROM "file:///medium/sample_tracks_medium.csv" AS row
2 CALL(row) {
3 MERGE (track:Track {id: row.track_id})
4 SET track.uri = row.track_uri,
5 track.name = row.track_name
6 } IN TRANSACTIONS OF 10,000 ROWS;
```

Set 187,846 properties

testload\$ MATCH (n) RETURN n ;

## 005-create-database-locking

neo4j\$ CREATE DATABASE locking WAIT;

Instance: neo4j://localhost:7687 Database: neo4j cypher User: neo4j

**Database information**

Nodes (0)

- Playlist Track User

Relationships (0)

HAS\_TRACK

Property keys

- id name position uri

neo4j\$

neo4j\$ CREATE DATABASE locking WAIT;

Table RAW

address	state	message	success
"localhost:7687"	"CaughtUp"	"caught up"	true

## 006-merge-wait-spinning

locking\$

```
MERGE (n:Track {id: 1})
WITH n
// the transaction will be paused for 60 seconds
CALL apoc.util.sleep(60000)
// the transaction now continues
SET n.name = 'Creep'
RETURN n
```

Instance: neo4j://localhost:7687 Database: locking cypher User: neo4j

**Database information**

Nodes (1)

- Track

Relationships (0)

Property keys

- id name

locking\$

```
1 MERGE (n:Track {id: 1})
2 WITH n
3 // the transaction will be paused for 60 seconds
4 CALL apoc.util.sleep(60000)
5 // the transaction now continues
6 SET n.name = 'Creep'
7 RETURN n
```

Graph Table RAW

Results overview

Nodes (1)

- Track (1)

Creep

## 007-create-node-1

locking\$ **CREATE (t:Track {id: 1})**

The screenshot shows the Neo4j Cypher Shell interface. The top bar indicates the instance is neo4j://localhost:7687, the database is locking cypher, and the user is neo4j. The left sidebar shows the database information, including 2 nodes (one labeled Track), 0 relationships, and property keys id and name. The main area displays the Cypher query locking\$ MATCH (n) RETURN n LIMIT 25; in the RAW tab, showing a graph with two nodes: 1 and Creep. The right sidebar shows the results overview with 2 nodes (one labeled Track). The bottom status bar shows the query locking\$ CREATE (t:Track {id: 1}) and a success message: Created 1 node, set 1 property, added 1 label.

## 008-tab-1-1

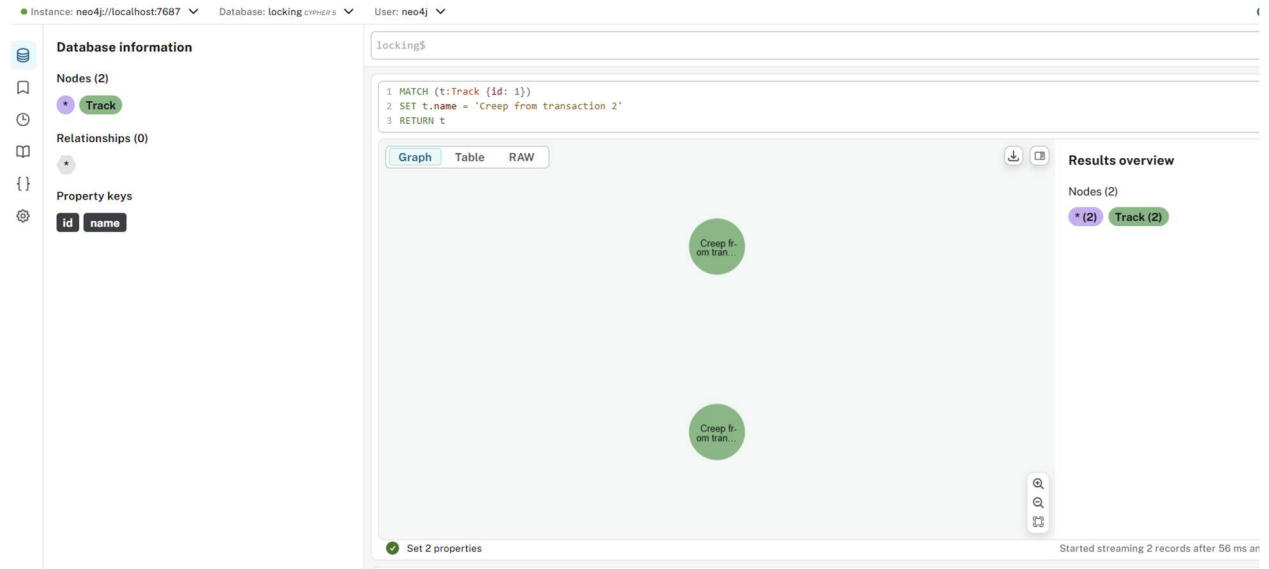
locking\$  
**MATCH (t:Track {id: 1})**  
**SET t.name = 'Creep'**  
**WITH t**  
**CALL apoc.util.sleep(60000)**  
**RETURN t**

The screenshot shows the Neo4j Cypher Shell interface. The top bar indicates the instance is neo4j://localhost:7687, the database is locking cypher, and the user is neo4j. The left sidebar shows the database information, including 2 nodes (one labeled Track), 0 relationships, and property keys id and name. The main area displays the Cypher query locking\$ MATCH (t:Track {id: 1}) SET t.name = 'Creep' WITH t CALL apoc.util.sleep(60000) RETURN t in the RAW tab, showing a graph with two nodes: Creep and Creep. The right sidebar shows the results overview with 2 nodes (one labeled Track). The bottom status bar shows the query locking\$ MATCH (t:Track {id: 1}) SET t.name = 'Creep' WITH t CALL apoc.util.sleep(60000) RETURN t and a success message: Updated 1 node, set 1 property, added 1 label.

## 009-tab-1-2

locking\$

```
MATCH (t:Track {id: 1})
SET t.name = 'Creep from transaction 2'
RETURN t
```

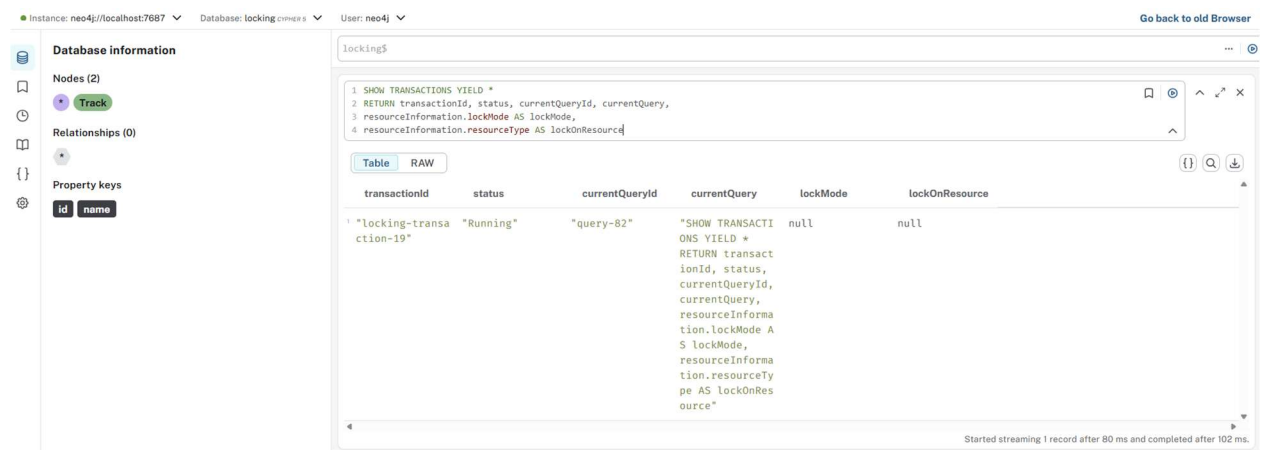


The screenshot shows the Neo4j interface with the query: `1 MATCH (t:Track {id: 1})`, `2 SET t.name = 'Creep from transaction 2'`, `3 RETURN t`. The results are displayed in a graph view with two nodes, both labeled "Creep from tran...". The left sidebar shows "Database information" with "Nodes (2)" and "Track (2)". The right sidebar shows "Results overview" with "Nodes (2)" and "Track (2)".

## 010-transactions-info

locking\$

```
SHOW TRANSACTIONS YIELD *
RETURN transactionId, status, currentQueryId, currentQuery,
resourceInformation.lockMode AS lockMode,
resourceInformation.resourceType AS lockOnResource
```



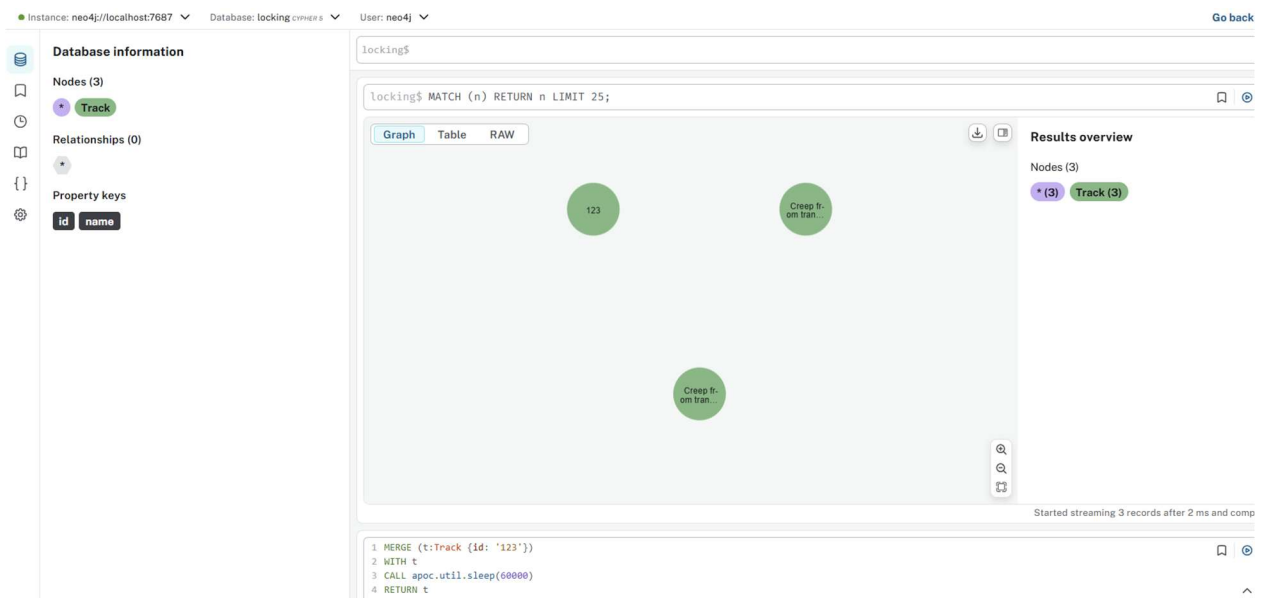
The screenshot shows the Neo4j interface with the query: `1 SHOW TRANSACTIONS YIELD *`, `2 RETURN transactionId, status, currentQueryId, currentQuery,`, `3 resourceInformation.lockMode AS lockMode,`, `4 resourceInformation.resourceType AS lockOnResource`. The results are displayed in a table view with columns: transactionId, status, currentQueryId, currentQuery, lockMode, and lockOnResource. The table contains one row of data.

transactionId	status	currentQueryId	currentQuery	lockMode	lockOnResource
"locking-transa ction-19"	"Running"	"query-82"	"SHOW TRANSACTIONS YIELD * RETURN transactionId, status, currentQueryId, currentQuery, resourceInformation.lockMode AS lockMode, resourceInformation.resourceType AS lockOnResource"	null	null

## 011-tab-2-1

locking\$

```
MERGE (t:Track {id: '123'})  
WITH t  
CALL apoc.util.sleep(60000)  
RETURN t
```



## 012-tab-2-2

locking\$

```
MERGE (t1:Track {id: '123'})  
MERGE (t2:Track {id: '234'})  
MERGE (t1)-[:SIMILAR_TO]->(t2)
```

Instance: neo4j://localhost:7687 Database: locking CYPHER 5 User: neo4j Go b

Database information

Nodes (4)

Track

Relationships (1)

SIMILAR\_TO

Property keys

id

name

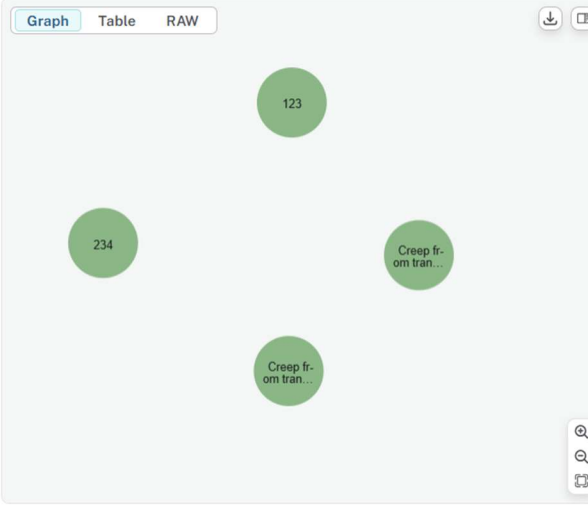
locking\$

locking\$ MATCH (n) RETURN n LIMIT 25;

Graph

Table

RAW



Results overview

Nodes (4)

\* (4)

Track (4)

Started streaming 4 records after 2 ms and c

1 MERGE (t1:Track {id: '123'})

2 MERGE (t2:Track {id: '234'})

3 MERGE (t1)-[:SIMILAR\_TO]->(t2)

Created 1 node, created 1 relationship, set 1 property, added 1 label

Co

Instance: neo4j://localhost:7687 Database: locking CYPHER 5 User: neo4j Go back

Database information

Nodes (4)

Track

Relationships (1)

SIMILAR\_TO

Property keys

id

name

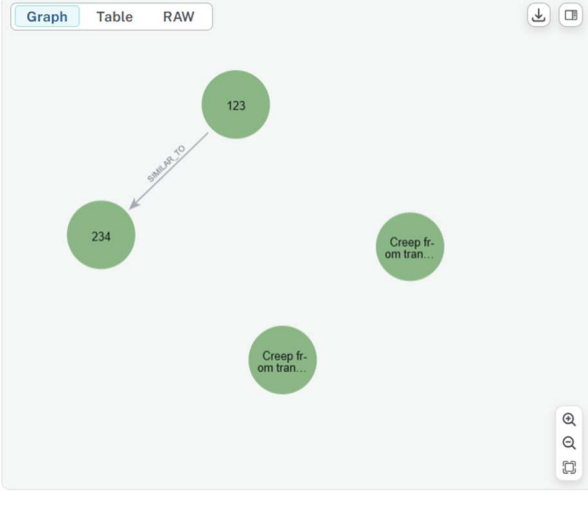
locking\$

locking\$ MATCH (n) RETURN n LIMIT 25;

Graph

Table

RAW



Results overview

Nodes (4)

\* (4)

Track (4)

Relationships (1)

\* (1)

SIMILAR\_TO (1)

Started streaming 1 record after 47 ms and comp

1 MERGE (t1:Track {id: '123'})

2 MERGE (t2:Track {id: '234'})

3 MERGE (t1)-[:SIMILAR\_TO]->(t2)

Started streaming 4 records after 2 ms and com

## 013-unique-constraint

locking\$ **MATCH (n) DETACH DELETE n;**

The screenshot shows the Neo4j Cypher console with the following commands and results:

```
locking$  
locking$ MATCH (n) RETURN n LIMIT 25;  
No changes, no records  
locking$ MATCH (n) DETACH DELETE n;  
Deleted 4 nodes, deleted 1 relationship  
locking$ MATCH (n) RETURN n LIMIT 25;
```

The left sidebar shows the database information for 'locking' with 0 nodes and 0 relationships. The 'Property keys' section shows 'id' and 'name'.

locking\$

**CREATE CONSTRAINT track\_uk FOR (t:Track) REQUIRE t.id IS UNIQUE;**

The screenshot shows the Neo4j Cypher console with the following commands and results:

```
locking$ show CONSTRAINTS  
locking$ CREATE CONSTRAINT track_uk FOR (t:Track) REQUIRE t.id IS UNIQUE;  
Added 1 constraint
```

The 'show CONSTRAINTS' command returns a table with the following data:

id	name	type	entityType	labelsOrTypes	properties
3	"track_uk"	"UNIQUENESS"	"NODE"	["Track"]	["id"]

The left sidebar shows the database information for 'locking' with 0 nodes and 0 relationships. The 'Property keys' section shows 'id' and 'name'.

## 014-tab-3-1

locking\$

**MATCH (t1:Track {id: 1})  
SET t1.popularity = 0.9  
WITH t1  
CALL apoc.util.sleep(60000)**

The screenshot shows the Neo4j Cypher console with the following commands and results:

```
locking$  
1 MATCH (t1:Track {id: 1})  
2 SET t1.popularity = 0.9  
3 WITH t1  
4 CALL apoc.util.sleep(60000)  
No changes, no records  
locking$ MATCH (n) RETURN n LIMIT 25;  
No changes, no records
```

The left sidebar shows the database information for 'locking' with 0 nodes and 0 relationships. The 'Property keys' section shows 'id' and 'name'.



## 015-tab-3-2

locking\$

```
MATCH (t1:Track {id: 1})  
MATCH (t2:Track {id: 2})  
CREATE (t1)-[r:SIMILAR]->(t2)
```

Instance: neo4j://localhost:7687 Database: locking cypher User: neo4j [Go back to old Browse](#)

**Database information**

Nodes (0)

- Track

Relationships (0)

locking\$

```
1 MATCH (t1:Track {id: 1})  
2 MATCH (t2:Track {id: 2})  
3 CREATE (t1)-[r:SIMILAR]->(t2)
```

No changes, no records

Completed after 99 ms