Cypher Shell Neo4j graph CLI

CONCEPTS

Statements typed in the shell must end with a semicolon; lines starting with : are shell commands and need none.

cypher-shell your terminal
↓ Bolt
Neo4j server port 7687
↓ routes to
Database neo4j / system
↓ holds
Graph nodes + relationships

Patterns are drawn in ASCII art: (node)-[:REL]→(node). Labels start with a colon, properties go in braces.

CONNECT

`cypher-shell -u <user> -p <password>`
Connect to localhost

`cypher-shell -a <uri>`
Connect to another server

`cypher-shell -d <database>`
Start in a given database

`cypher-shell --access-mode read`
Open a read-only session

Environment variables

Variable	Replaces
NEO4J_USERNAME	-u
NEO4J_PASSWORD	-p
NEO4J_ADDRESS	-a
NEO4J_DATABASE	-d

SHELL COMMANDS

`:help`
List shell commands

`:exit`
Quit the shell

`:use <database>`
Set the active database

`:begin`
Open an explicit transaction

`:commit`
Commit the open transaction

`:rollback`
Roll back the open transaction

`:param { alias: 'Robin' }`
Set a query parameter

SHELL COMMANDS (cont.)

`:source <file>`
Run statements from a file

Arrows browse history, Tab autocompletes commands, Home and End jump along the line.

SCRIPT & OUTPUT

`cypher-shell "<cypher>"`
Run one statement and exit

`cypher-shell -f <file>`
Run a Cypher file and exit

`cypher-shell --format plain`
Minimal output for scripts

`cypher-shell --format verbose`
Tables plus statistics

`cypher-shell -P '{n: 3}'`
Set a session parameter

`cypher-shell --fail-at-end`
Report errors after the whole file

READ

`MATCH (n) RETURN n LIMIT <k>`
Sample some nodes

`MATCH (n:<Label>) RETURN n`
Nodes with a label

`MATCH (n:<Label> {<prop>: <value>}) RETURN n`
Match by property

`MATCH (a)-[r:<TYPE>]→(b) RETURN a, r, b`
Follow a relationship type

`WHERE n.<prop> > <value>`
Filter matches

`RETURN n.<prop> AS <alias>`
Rename a returned value

`RETURN DISTINCT n.<prop>`
Remove duplicate rows

`ORDER BY n.<prop> DESC`
Sort the rows

`SKIP <n> LIMIT <m>`
Page through results

`OPTIONAL MATCH (n)⟶(m)`
Match, or give null

`MATCH (a)-[*1..3]→(b)`
Paths of one to three hops

READ (cont.)

`count(n), sum(n.<prop>), collect(n)`
Aggregate rows

WRITE

`CREATE (:<Label> {<prop>: <value>})`
Create a node

`CREATE (a)-[:<TYPE>]→(b)`
Create a relationship

`MERGE (n:<Label> {<prop>: <value>})`
Create only if missing

`SET n.<prop> = <value>`
Set a property

`SET n += {<prop>: <value>}`
Update several properties

`DELETE r`
Delete a relationship

`DETACH DELETE n`
Delete a node and its relationships

PIPELINES

`WITH n, count(*) AS c`
Pass results to the next part

`UNWIND <list> AS x`
Turn a list into rows

SCHEMA & ADMIN

`SHOW DATABASES`
List databases

`SHOW INDEXES`
List indexes

`SHOW CONSTRAINTS`
List constraints

`CREATE INDEX FOR (n:<Label>) ON (n.<prop>)`
Index a property

`CREATE CONSTRAINT FOR (n:<Label>) REQUIRE n.<prop> IS UNIQUE`
Enforce unique values

TUNING

`EXPLAIN <query>`
Show the plan without running

`PROFILE <query>`
Run and show rows and db hits