Neo4j & Graph Data Science Cheat Sheet

Calendar-versioned `2026.x` line (Neo4j + GDS). Quick reference only — verify against the official docs before production use.

CYPHER — CORE CLAUSES

```
CREATE (a:Person {name:$n})-[:ACTED_IN {roles:$r}]->(m:Movie)
MATCH (a:Person)-[:ACTED_IN]->(m:Movie) WHERE a.name=$n
RETURN a,m
MERGE (a:Person {name:$n}) ON CREATE SET a.since=date()
SET a.age = 40   REMOVE a.age
DETACH DELETE a
```

Escaping: in prose, write literal braces as `\{ \}` outside `[source,cypher]` blocks.

Params: `$name` — never inline user input into query text.

ADVANCED QUERYING

```
WITH a, count(*) AS c WHERE c > 1
CALL { MATCH (x) RETURN x LIMIT 5 }
UNION ALL
UNWIND $list AS item MERGE (:Tag {name:item})
OPTIONAL MATCH (a)-[:KNOWS]->(b)
MATCH p=shortestPath((a)-[:ACTED_IN*1..3]-(b))
MATCH allShortestPaths((a)-[*]-(b))
```

List/map comprehensions: `[x IN list WHERE x.age>18 | x.name]`

PROPERTY GRAPH MODEL

- Nodes: labels (multi), properties
- Relationships: always directed & typed, can carry properties
- Types: string, number, boolean, temporal (date/datetime/duration), spatial `Point`, `LIST/MAP`, native `VECTOR`
- Schema-optional — constraints enforce structure, not required upfront
- Avoid supernodes: use intermediate nodes / relationship properties

IMPORTING DATA

```
LOAD CSV WITH HEADERS FROM 'file:///p.csv' AS row
CALL { WITH row CREATE (:Person{name:row.name}) }
IN TRANSACTIONS OF 1000 ROWS

neo4j-admin database import full <db>
neo4j-admin database import incremental <db>
```

APOC: `apoc.load.json` / `apoc.load.jdbc` for external sources.

INDEXES & CONSTRAINTS

Index types: range, text, point, composite, full-text, vector.

```
CREATE INDEX FOR (p:Person) ON (p.name)
CREATE VECTOR INDEX FOR (p:Product) ON p.embedding
CREATE CONSTRAINT FOR (p:Person)
  REQUIRE p.email IS UNIQUE
SHOW INDEXES  SHOW CONSTRAINTS  DROP INDEX x
```

Constraints: uniqueness, existence, node key, relationship key.

TRANSACTIONS & DRIVERS

ACID + causal consistency & bookmarks across a cluster. Bolt driver family: Java, Python, JS, .NET, Go.

```
driver.executeWrite(tx ->
  tx.run("MERGE (p:Person{name:$n})", params))
# python
driver.execute_write(tx_fn)
```

Transaction functions auto-retry on transient errors. Backs Spring Data Neo4j's `Neo4jClient`.

APOC HIGHLIGHTS

- Enable: `apoc.import.file.enabled` + allowlists in `neo4j.conf`
- `apoc.refactor.mergeNodes` / `rename*`
- `apoc.load.json` / `apoc.load.jdbc`
- `apoc.path.*` path utilities
- `apoc.periodic.iterate` batching
- `apoc.export.csv` / `export.json`
- Ecosystem: GraphQL Library, Kafka Connector, Spark Connector

GRAPH DATA SCIENCE — PROJECTIONS & MODES

```
CALL gds.graph.project('g', 'Person', 'ACTED_IN')
// native
CALL gds.graph.project.cypher('g', nodeQ, relQ)
// cypher
CALL gds.graph.project.estimate(...)
// memory est.
CALL gds.graph.drop('g')
```

Execution modes (every algorithm): `stream` results, `mutate` in-memory graph, `write` back to DB, `stats` summary only.

PATHFINDING & CENTRALITY

- Dijkstra, A*, Yen's k-shortest paths
- `gds.pageRank.stream` (+ personalized)
- Degree, Betweenness, Closeness, Eigenvector centrality

```
CALL gds.pageRank.stream('g')
YIELD nodeId, score
```

COMMUNITY DETECTION

- WCC / SCC (weak / strong components)
- Triangle Count, Local Clustering Coefficient
- Louvain, Label Propagation
- Leiden — more stable Louvain successor

SIMILARITY, EMBEDDINGS & ML

- Node Similarity, KNN
- FastRP / Node2Vec (structural)
- GraphSAGE (inductive)
- Node classification & link prediction pipelines

VECTOR SEARCH & GRAPHRAG

```
MATCH (p:Product)
SEARCH p.embedding NEAR $q APPROXIMATE
RETURN p LIMIT 5
```

Native `VECTOR` type + vector index; modern `SEARCH` clause supersedes `db.index.vector.queryNodes`.

GraphRAG: vector retrieval + graph traversal for LLM context. `neo4j-graphrag-python`, LangChain4j, Spring AI.

CLUSTERING & MULTI-DATABASE

Raft-based cluster: **primary**/**secondary** roles (formerly CORE/READ_REPLICA), leader election.

```
CREATE DATABASE sales
SHOW DATABASES
```

Composite databases (formerly Fabric) span multiple constituent DBs. Aura manages both.

SECURITY & ADMINISTRATION

Auth: native users, LDAP, OIDC. RBAC with built-in roles.

```
GRANT MATCH {*} ON GRAPH * TO reader
DENY WRITE ON GRAPH * TO reader
```

Plus: backup/restore, Bolt+TLS encryption, security event log auditing.