Neo4J Cheatsheet

1 Why are RDBMS bad ?

- Contrived model; interested in connections more than data itself;
- Relationships are inferred from foreign keys;
- Intermediary table introduce unnecessary complexity;
- Struggles with highly connected application domains;
- Schema is rigid, does not scale well.

Why graph databases are better?

- Relationships are explicit;
- Model is easier to understand by a wide variety of people;
- Queries expressed as examples (patterns);
- **Additivity property:** Easy to add new relationships and nodes to graph without disturbing outcomes of existing queries.
- No schema!
- Connected data is stored as connected data.

2 Data model

No schema (schemaless database) – a simple graph.
Node: A single entity within the database. *May contain 0..n properties.*
Relationships: Directed edge associating two nodes. *Has a type (name), may 0..n properties.*
Properties: Simple dictionary text association.

```
{ attributeName: attributeValue, ... }
```

Labels: Acts as a "tag" for nodes. *A node may have 0..n associated labels.*

3 Cypher Query Language

Is a declarative query language;
Focuses on the *what* instead of the *how*;
Pattern oriented: Use examples to traverse the graph.
Placeholders: may be included in pattern (*optionally* bundled with variable names) to capture and manipulate data within the query;

3.1 MATCH clause (data retrieval)

- Specify pattern on *how* to traverse the graph (an example);

Selects a user named 'Jim' that is friend with any other user:

```
MATCH
(a:User {name: 'Jim'})-[:FRIEND]->(b:User)
RETURN a
```

Selects the customers' name along with the number of product they purchased that are part of a category named 'Produce':

```
MATCH (cust:Customer)-[:PURCHASED]->
        (:Order)-[o:ORDERS]->(p:Product),
      (p)-[:PART_OF]->
        (c:Category {categoryName:"Produce"})
RETURN
DISTINCT cust.contactName as CustomerName,
SUM(o.quantity) AS TotalProductsPurchased
```

3.2 Data manipulation clauses

CREATE/DELETE. Create or delete all part of a pattern:

Creates a new :Bar node and a :FOO relationship associated with every node having id = 1:

```
MATCH (n) WHERE id = 1 CREATE (n)-[:FOO]->(b:Bar)
MATCH (n) WHERE id = 1 DELETE (n)-[:FOO]->(b:Bar)
```

MERGE. Acts as a create clause iff node does not exists.

Indexes. Uniquely identify a node within a label. Mainly for efficieny –not as involved as in RDBMS.

4 Built-in tools and APIs

Neo4J Browser. Test and vizualize queries and manage database within your browser.

REST APIs: Query remotely server over HTTP: POST, GET, PUT, DELETE.