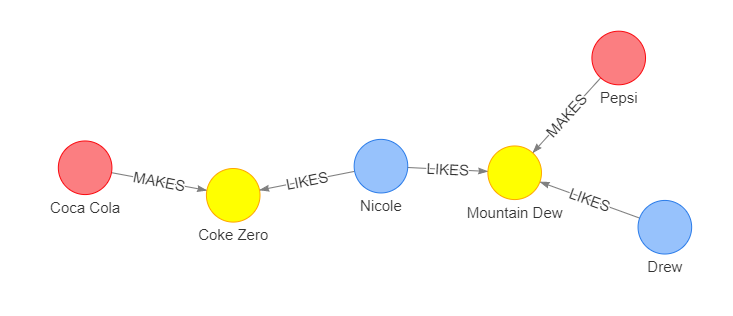
**Neo4J Graph Database 15 Minute Tutorial**

**What is:** Graph databases are the same as SQL databases in that they store data, however all data comes in the form of *nodes* or *edges* on a web, as opposed to tables. Advantages include frequently faster performance for high process time queries (depending on setup), shorter and more simple queries that eliminate the concept of table-joins, visualization through Neo4j, and better conceptual representation of data in *some scenarios.* Disadvantages include barrier to entry of understanding for a userbase, the additional planning required for data insertion strategy, and inability to store BLOB data types efficiently (though storing location pointers instead of BLOBs obviously skirts this).

**Free?** Community edition, yes.

**Cap-sensitive?** Yes

****

**Nodes:**

Nodes are easily thought of as values in a dictionary where the key is the internal ‘id’ assigned by Neo4j and the value is an object:

* Nodes have a ‘type’, a ‘value’, and an internal ‘id’ assigned by neo4j for internal tracking reasons
  + ‘types’ shown above are probably ‘company’, ‘drink’, and ‘people’ [‘Drink’] (yellow)
  + ‘values’ are the strings you see, though they could also be any type of object [‘Coke Zero’]
  + ‘id’ is used in neo4j’s backend, and should be completely ignored [<4>] (not shown)
* Nodes, unlike rows in SQL, do not need to have values stored for all fields, even among the same ‘type of node’, however to maintain sanity it is normal practice to have consistent fields across similar types.
* Multiple nodes can have identical values, because they are identified by their ‘id’ key in the backend

**Relationships**

Relationships connections between any types of node. Relationships can be entire objects themselves with multiple values, just like nodes.

* Relationships cannot exist without at least one node (a node can have a relationship with itself, but there’s no reason to ever use this feature)
* Relationships do not have a ‘type’ like nodes do. However, have polarity in a single specified direction
* Relationships can be objects as well, though in practice and to maintain sanity, they usually are just strings.

**Basic Building Steps:**

1. Insert all nodes (for row in table, insert node ’type’ and its child values)
2. Apply relationships. For example you may say ‘point coca cola to all drink node types with ‘coke’ in their name
   1. I suggest adding values to nodes that can be easily used to connect relationships later on.
3. Done

**Query Language:**

Haven’t researched that much yet.