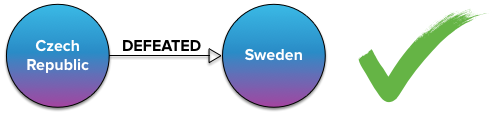
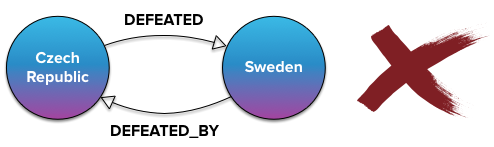
**Directed Relationships**

Relationships in Neo4j must have a type, giving the relationship a semantic meaning, and a direction. Frequently, the direction becomes part of the relationship’s meaning. In other words, the relationship would be ambiguous without it. For example, the following graph shows that the Czech Republic *DEFEATED* Sweden in ice hockey. Had the direction of the relationship been reversed, [the Swedes](http://neotechnology.com/) would be much happier. With no direction at all, the relationship would be ambiguous, since it would not be clear who the winner was.



Note that the existence of this relationship implies a relationship of a different type going in the opposite direction, as the next graph illustrates. This is often the case. To give another example, the fact that Pulp Fiction was *DIRECTED\_BY* Quentin Tarantino implies that Quentin Tarantino *IS\_DIRECTOR\_OF* Pulp Fiction. You could come up with a huge number of such relationship pairs.

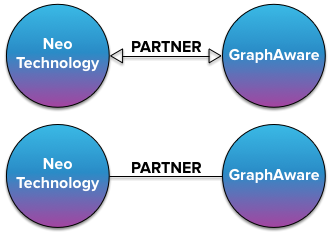


**One common mistake** people often make when modelling their domain in Neo4j is creating both types of relationships. Since one relationship implies the other, this is wasteful, both in terms of space and traversal time. Neo4j can traverse relationships in both directions. More importantly, thanks to the way Neo4j organizes its data, the **speed of traversal does not depend on the direction** of the relationships being traversed.

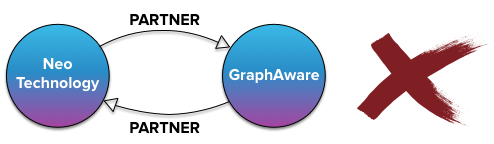
**Bidirectional Relationships**

Some relationships, on the other hand, are naturally bidirectional. A classic example is Facebook or real-life friendship. This relationship is mutual - when someone is your friend, you are (hopefully) his friend, too. Depending on how we look at the model, we could also say such relationship is undirected.

GraphAware and Neo Technology are partner companies. Since this is a mutual relationship, we could model it as bidirectional or undirected relationship, respectively.



But since none of this is directly possible in Neo4j, beginners often resort to the following model, which suffers from the exact same problem as the incorrect ice hockey model: an extra unnecessary relationship.



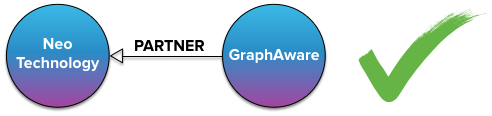
Neo4j APIs allow developers to completely **ignore relationship direction** when querying the graph, if they so desire. For example, in Neo4j’s own query language, Cypher, the key part of a query finding all partner companies of Neo Technology would look something like

MATCH (neo)-[:PARTNER]-(partner)

The result would be the same as executing and merging the results of the following two different queries:

MATCH (neo)-[:PARTNER]->(partner) and MATCH (neo)<-[:PARTNER]-(partner)

Therefore, the **correct** (or at least most efficient) way of modelling the partner relationships is using a **single** *PARTNER* **relationship with an arbitrary direction**.



**Conclusion**

Relationships in Neo4j can be traversed in both directions with the same speed. Moreover, direction can be completely ignored. Therefore, there is no need to create two different relationships between nodes, if one implies the other.