ONTOTEXT ANSWER:

For those of you who are just finding this for the first time, GraphDB operates a forward chaining algorithm, where statements are inferred and materialized at insertion time. This means some statements that are present in physical storage are explicit and some are implicit. All implicit statements are stored inside the default graph.

When sending any query towards GraphDB, you can use the `includeInferred` parameter to toggle if you want to include inferred statements or not. This parameter can also be configured on the UI via a button inside the SPARQL view. However, when you want to combine results, this isn't viable – you rarely want to run the query twice and compare results!

This is why GraphDB comes with multiple special "named graphs". In this case, we would want to use the implicit and explicit graphs. Those special "pseudo-graphs" aren't materialized like all other graphs, so you can't use them with all standard SPARQL constructs. Filtering and the GRAPH statement don't always work as expected. Those statements are optimized for working with the FROM keyword.

Let us illustrate this, with the Graph Path Search algorithm. The algorithm allows you to look for all paths between two RDF nodes. So, let's make a small data file about parts of the body and how they are connected.

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX : <http://test#>
INSERT data {
    :hand :partOf :arm .
    :arm :partOf :body .
    :partOf a owl:ObjectProperty ;
            a owl:TransitiveProperty .
}
```

You want to use this on a repository with some inference enabled. RDFS-plus is required at the least!

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

Then, you can query the data, **using FROM NAMED twice**.

```
PREFIX path: <http://www.ontotext.com/path#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX : <http://test#>
SELECT DISTINCT ?source ?target ?graph ?index ?path
FROM NAMED <http://www.ontotext.com/implicit>
FROM NAMED <http://www.ontotext.com/explicit>
WHERE {
    VALUES (?src ?property) {
        ( <http://test#hand> <http://test#partOf>)
    }
    SERVICE <http://www.ontotext.com/path#search> {
        <urn:path> path:findPath path:allPaths ;
                   path:sourceNode ?src ;
                   path:destinationNode ?dst ;
                   path:maxPathLength 1 ;
                   path:startNode ?source;
                   path:exportBinding ?graph ;
                   path:exportBinding ?property ;
                   path:endNode ?target ;
                   path:resultBindingIndex ?index ;
                   path:pathIndex ?path .
         SERVICE <urn:path> {
            ?source ?property ?target .
             OPTIONAL { GRAPH ?graph { ?source ?property ?target } }
        }
    }
}
```

And get as result:

source	target	graph	index
http://test#hand	http://test#arm	http://www.ontotext.com/explicit	0
http://test#hand	http://test#body	http://www.ontotext.com/implicit	0

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

So, both paths starting at the hand are of length 0, i.e., there are no intermediate steps. However, one of these is explicit and the other one – implicit.

You can use FROM NAMED in any SPARQL query to get reliable results about your explicit and implicit statements.

Help us get better: review GraphDB on Gartner

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com