The documentation of SPARQL 1.1 can be found at:

- https://www.w3.org/TR/sparql11-query/
- *https://www.w3.org/TR/sparql11-update/*

The following SPARQL update inserts some (ground) triples into the default graph of the graph store (i.e. the *default graph* in the GraphDB *graphs overview*).

```
PREFIX : <http://example.org/>

PREFIX foaf: <http://xmlns.com/foaf/0.1/>


INSERT DATA {

    :tizio a foaf:Person .

    :gino a foaf:Person .

    :pino a foaf:Person .


    :megaCompany a foaf:Organization , foaf:Group .

    :littleCompany a foaf:Organization , foaf:Group .


    :tizio foaf:knows :gino .

    :pino foaf:knows :tizio .

    :pino foaf:knows :gino .


    :megaCompany foaf:member :tizio .

    :megaCompany foaf:member :gino.

    :littleCompany foaf:member :pino .

}
```

The following SPARQL update deletes the default graph from the graph store, and restores it afterwards (because there must be a default graph in a graph store).

```
DROP DEFAULT
```

It is thus equivalent to the following update:

```
CLEAR DEFAULT
```

The following query counts the unique persons defined in the triple store:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

select (COUNT(DISTINCT ?x) as ?personCount) {

    ?x a foaf:Person .

}
```

The following query returns people and the number of people who know them, provided that this number is higher than or equal to a given threshold (in this case 1).

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

select (COUNT(?x) as ?cnt) ?y {

    ?x foaf:knows ?y .

}

group by ?y

having (?cnt >= 1)

order by DESC(?cnt)
```

The query above may not return people known by no one, even if the threshold is set to zero. The reason lies in the fact that the graph pattern must match. In the following query, the use of the OPTIONAL keyword allows us to overcome this limitation:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

select (COUNT(?x) as ?cnt) ?y {

    ?y a foaf:Person .

    OPTIONAL {

      ?x foaf:knows ?y .

    }

}

group by ?y

having (?cnt >= 0)

order by DESC(?cnt)
```

Below you can find the structure of an update for removing ground triples from a specific named graph.

```
DELETE DATA {

    GRAPH <...> {

        ...
    }
}
```

Below you can find the structure of an update that queries the underlying triples store, and then for each solution instantiates the provided graph patterns to generate the triples (actually the quadruples) to be removed and added.

```
DELETE {


}
INSERT {


}
WHERE {
    ....
}
```

Below you can find a Boolean (ASK) query to the triple store.

```
PREFIX : <http://example.org/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
ask {
    :gino a foaf:Organization .
}
```

A CONSTRUCT allows us to instantiate triples without asserting them. The following CONSTRUCT allows us to express the semantics of functional properties in OWL. The FILTER is necessary to avoid that the two triple patterns `?s ?p ?o1` and `?s ?p ?o2` match the same triple .

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
CONSTRUCT {
    ?o1 owl:sameAs ?o2 .
} WHERE {
    ?p a owl:FunctionalProperty .
    ?s ?p ?o1 .
    ?s ?p ?o2 .
    FILTER(?o1 != ?o2)
}
```

This intuition has eventually lead to the development of SPIN (SPARQL Inferencing Notation): http://spinrdf.org/