How to use global variables in gsql

Summary

GSQL use ACCUM clause to do aggregation with accumulators. GSQL has a set of built-in accumulator types. Each accumulator has a "+=" operator, which is used to accumulate result. The semantic for each "+=" operator is part of the built-in accumulator type.

A global accumulator can be viewed as global state variable, while user traverse the topology, anywhere user can update the global accumulator.

All sample queries below are based on GSQL 101 sample data set – social graph. Its schema is listed below.

Notation: we use GSQL code block to indicate the code should be invoked from GSQL shell. Comments start with # sign.

```
CREATE VERTEX person (PRIMARY_ID name STRING, name STRING, gender
STRING, age INT, state STRING)
CREATE UNDIRECTED EDGE friendship (FROM person, TO person, connect_day
DATETIME)
CREATE GRAPH social (person, friendship)
```

How to declare a global accumulator

Global accumulator must be declared first before any SELECT query block. Global accumulator's name is always prefixed with "@@", followed by an user-chose identifier. Just think any variable name with "@@" to indicate this variable is global visible. Here is an example.

```
USE GRAPH social
CREATE QUERY testGV (VERTEX<person> p) FOR GRAPH social{
  #declare an integer global sum accumulator, where += means
  # arithmetic plus
  SumAccum<int>  @@cnt = 0;
}
```

How to accumulate values into it

In this example, we have two SELECT query block. The first one find and mark input vertex p's first-hop neighbors. And the second SELECT query block find the second-hop neighbors. In both query blocks, we count the encountered vertices in @@cnt. And last, we decrement it to take out the starting vertex count.

```
USE GRAPH social
CREATE QUERY totalNeighbor (VERTEX<person> p) FOR GRAPH social{
  #global accumulator to accumulate neighbor count
  SumAccum<int>  @@cnt = 0;
  #local accumulator to mark which vertex has been seen
  OrAccum  @visited = false;

  Start = {p};

  #Below, in POST-ACCUM, for each vertex that has an accumulator
calculation in ACCUM,
  # the post-accum statement will be invoked once.
  # We add 1 to the global accumulator @@cnt in post-accum.
  # Note we add 1 to @@cnt for the starting point s, and all its
first-hop neighbors,
  # since all of them has updated @visited accumulator in ACCUM clause
  FirstNeighbors = SELECT tgt
                   FROM Start:s-(friendship:e)->person:tgt
                   ACCUM tgt.@visited += true, s.@visited += true
                   POST-ACCUM @@cnt +=1;

  # for each second-hop neighbors, we add 1 to @@cnt
  SecondNeighbors = SELECT tgt
                    FROM FirstNeighbors-(friendship:e)->person:tgt
                    WHERE tgt.@visited == false
                    POST-ACCUM @@cnt +=1;

  #decrement 1 from @@cnt, as we don't count starting point as a
neighbor.
  @@cnt += -1;

  PRINT @@cnt;
}
INSTALL QUERY totalNeighbor
RUN QUERY totalNeighbor("Tom")
RUN QUERY totalNeighbor("Dan")
```

Supported global accumulator

https://doc.tigergraph.com/GSQL-Language-Reference-Part-2---Querying.html#GSQLLanguageReferencePart2-Queryingv2.0REL-Accumulators