MAS DSE 250 Beyond Relational Data Models

Homework 4

Due Fri, Dec 7 by midnight

This homework is a programming assignment. Please turn it in electronically, by github, in form of a single text file containing your queries.

In this assignment, we will develop some GSQL queries over the movie graph provided by your installation of neo4j, translated to TigerGraph form conforming to the following schema:

```
CREATE VERTEX Person (PRIMARY_ID name STRING, name STRING, born INT)
CREATE VERTEX Movie (PRIMARY_ID title STRING, title STRING, released INT, tagline STRING)
CREATE VERTEX Role (PRIMARY_ID role STRING, role STRING)

CREATE UNDIRECTED EDGE DirectorOf (FROM Person, TO Movie)
CREATE UNDIRECTED EDGE ActorRole (FROM Person, TO Role)
CREATE UNDIRECTED EDGE RoleMovie (FROM Role, To Movie)

CREATE GRAPH MovieGraph(Person, Movie, Role, DirectorOf, ActorRole, RoleMovie)
USE GRAPH MovieGraph
BEGIN
CREATE LOADING JOB LoadMovieGraph FOR GRAPH MovieGraph {
  DEFINE FILENAME file_p="/home/tigergraph/movie/person.csv";
  DEFINE FILENAME file_m="/home/tigergraph/movie/movie.csv";
  DEFINE FILENAME file_r="/home/tigergraph/movie/role.csv";
  DEFINE FILENAME file_d="/home/tigergraph/movie/director_of.csv";
  DEFINE FILENAME file_a="/home/tigergraph/movie/actor_role.csv";
  DEFINE FILENAME file_c="/home/tigergraph/movie/role_movie.csv";
  LOAD file_p TO VERTEX Person VALUES ($0, $0, $1) USING header="true", separator=",";
  LOAD file_m TO VERTEX Movie VALUES ($0, $0, $1, $2) USING header="true", separator=",";
  LOAD file_r TO VERTEX Role VALUES ($0, $0) USING header="true", separator=",";
  LOAD file_d TO EDGE DirectorOf VALUES ($0, $1) USING header="true", separator=",";
  LOAD file_a TO EDGE ActorRole VALUES ($0, $1) USING header="true", separator=",";
  LOAD file_c TO EDGE RoleMovie VALUES ($0, $1) USING header="true", separator=",";
}
END
RUN LOADING JOB LoadMovieGraph
```

Answer the following queries in GSQL:

1. **Find the name of directors who have never acted. (hint: counting movies everyone acted in is easy)**

```
create query dir_never_acted() for graph MovieGraph {
        ArrayAccum<SumAccum<INT>> @edgesByType[2];
        per = {Person.*};
        Persons = SELECT s FROM per:s -(:e)-> :t
                  ACCUM
                        CASE e.type
                             WHEN "DirectorOf" THEN s.@edgesByType[0] += 1
                             WHEN "ActorRole"  THEN s.@edgesByType[1] += 1
                        END;

        PRINT Persons[Persons.@edgesByType] where Persons.@edgesByType[0]>0 and
Persons.@edgesByType[1]==0;
}
```

2. **For each director, find the number of roles they have directed over their entire career (display director name and role number).**

```
create query role_count() for graph MovieGraph {
        MapAccum<STRING, INT> @@movieaccum, @@diraccum;
        roles = {Role.*};
        dirs = {Person.*};
        role_count = SELECT m FROM roles:s-(RoleMovie)->Movie:m ACCUM @@movieaccum += (m.title-
>1);
        dir_num_role_count = SELECT m FROM dirs:d-(DirectorOf)->Movie:m ACCUM @@diraccum +=
(d.name->@@movieaccum.get(m.title));
        PRINT @@diraccum;
 }
```

3. **Find the name of actors who have worked with every director born after 1966 (hint: again, counting will be easiest)**

```
create query actors_worked() for graph MovieGraph {
        MapAccum<STRING, SetAccum<string>> @@role_to_person, @@movie_to_person, @@actors;
        SetAccum<String> @@directors;
        INT flag=0;
        SetAccum<String> @@actors_set, @@temp;
        actors = {Person.*};
        dirs = {Person.*};
        role = {Role.*};
        actor_role = SELECT r FROM actors:a-(ActorRole)->Role:r ACCUM @@role_to_person +=
(r.role->a.name);
        movie_to_person = Select r from role:r-(RoleMovie)->Movie:m ACCUM @@movie_to_person +=
(m.title->@@role_to_person.get(r.role));
        directors = SELECT d from dirs:d-(DirectorOf)->Movie:m where d.born>1966
                ACCUM @@actors += (d.name->@@movie_to_person.get(m.title)), @@directors +=
d.name;
        FOREACH key IN @@directors
                DO
```

```
                        IF flag==0 THEN @@actors_set=@@actors.get(key);flag=1;
                        ELSE @@temp=@@actors.get(key);@@actors_set=@@actors_set INTERSECT @@temp;
                        END;
                END;
        PRINT @@actors_set;
}
```

4. Finally we can try the Kevin Bacon query without running out of resources when we go past 3 degrees of separation (the limitation in neo4j): denote with b(p) the bacon degree of separation for person p, defined as
• b(p)= 0 if p is Kevin Bacon
• b(p)= 1 if p worked on the same movie as Kevin Bacon (in any capacity,
either as actor or as director)
• in general,
b(p)= min {1 + b(q) | q and p are distinct and worked on the same movie}.

Find the name and degree of separation of every person connected to Kevin Bacon.

```
create query actors_connected() for graph MovieGraph {
        MapAccum<STRING, SetAccum<string>> @@role_to_person, @@person_to_role, @@movie_to_person,
@@person_to_movie;
        MapAccum<INT, SetAccum<string>> @@person_connections;
        SetAccum<String> @@movie_list, @@movie_set, @@temp_movie_set, @@person_set,
@@temp_person_set;
        SumAccum<INT> @@degree_of_separation=0;
        INT stop=0;
        actors = {Person.*};
        dirs = {Person.*};
        role = {Role.*};

        actor_role = SELECT r FROM actors:a-(ActorRole)->Role:r
        ACCUM
                @@role_to_person += (r.role->a.name);

movie_to_person = SELECT r FROM role:r-(RoleMovie)->Movie:m
        ACCUM
                @@movie_to_person += (m.title->@@role_to_person.get(r.role)),
                @@movie_list += m.title;

FOREACH movie in @@movie_list DO
        FOREACH person in @@movie_to_person.get(movie) DO
                @@person_to_movie += (person->movie);
        END;
END;

director_to_movie = SELECT d FROM dirs:d-(DirectorOf)->Movie:m
        ACCUM
                @@movie_to_person += (m.title->d.name),
                @@person_to_movie += (d.name->m.title);

#PRINT @@movie_to_person;
#PRINT @@person_to_movie;

@@movie_set = @@person_to_movie.get("Kevin Bacon");
@@person_set = "Kevin Bacon";
@@person_connections += (@@degree_of_separation->@@person_set);

WHILE stop==0 DO
        FOREACH movie in @@movie_set DO
                FOREACH person in @@movie_to_person.get(movie) DO
                        @@temp_person_set += person;
                END;
        END;

        @@person_set = @@temp_person_set MINUS @@person_set;
        @@temp_person_set.clear();
```

```
                IF
                        @@person_set.size() == 0 THEN stop=1;
                ELSE
                        @@degree_of_separation += 1;
                        @@person_connections += (@@degree_of_separation->@@person_set);
                END;

                FOREACH person in @@person_set DO
                        FOREACH movie in @@person_to_movie.get(person) DO
                                @@temp_movie_set += movie;
                        END;
                END;

                @@movie_set = @@temp_movie_set MINUS @@movie_set;
                @@temp_movie_set.clear();
        END;

        PRINT @@person_connections;
}
```