PostgreSQL Triggers Lab II

Tom Kelliher, CS 417

The purpose of this lab is for you to gain some understanding of how triggers and stored procedures are used in PostgreSQL to implement integrity constraints. You will create a PL/pgSQL function which will implement a semantic constraint for a payroll database. The constraint is: No employee should earn more than his or her manager. You will then implement a second trigger and stored procedure. This trigger will maintain the constraint that no manager should earn more than the sum of the salaries of those she or he manages. These will be separate, but concurrent, triggers.

1. Download `triggerLab2.sql` from the course web site to one of your directories on phoenix.
2. Open this file in an editor. Notice that it consists of three parts: a clean-up section; a section which creates the payroll table, PL/pgSQL function, and trigger; and a section which tests the trigger function.
3. You will first be writing the code for `CheckEmployeeFunc()`. If the salary in question is greater than the manager's salary, adjust the salary to be the same as the the manager's salary and raise a notice. If the employee has no manager, raise a notice to this effect.
4. When you're ready to run the code, run `psql` to open your personal database and execute the SQL code in your file. The output from the run should be very similar to:

```
DROP TRIGGER
DROP FUNCTION
DROP TRIGGER
DROP FUNCTION
DROP TABLE
CREATE TABLE
CREATE FUNCTION
CREATE TRIGGER
CREATE FUNCTION
psql:triggerLab2Soln.sql:91: NOTICE:  Jim has no manager.
INSERT 0 1
psql:triggerLab2Soln.sql:93: NOTICE:  Frank has no manager.
INSERT 0 1
psql:triggerLab2Soln.sql:95: NOTICE:  Beth has no manager.
INSERT 0 1
UPDATE 1
UPDATE 1
psql:triggerLab2Soln.sql:100: NOTICE:  Salary adjusted for Frank.
UPDATE 1
 id | name  | mid | salary
----+-------+-----+--------
```

```
  1 | Beth  |     | 100000
  2 | Jim   |   1 |  75000
  3 | Frank |   1 | 100000
(3 rows)

psql:triggerLab2Soln.sql:104: NOTICE:  Beth has no manager.
UPDATE 1
 id | name  | mid | salary
----+-------+-----+--------
  2 | Jim   |   1 |  75000
  3 | Frank |   1 | 100000
  1 | Beth  |     | 200000
(3 rows)
```

5. Create a trigger, `CheckManagerTrigger`, and stored procedure, `CheckManagerFunc()`, to enforce the constraint that no manager earn more than the sum of his or her employees' salaries. When adjusting a salary, raise a notice. This trigger should enable on insert or update events, should be a row-level trigger, and the condition should be checked prior to the occurrence of the event. You should also raise a notice if the employee manages no one.

6. When you're ready to run the code, run `psql` to open your personal database and execute the SQL code in your file. The output from the run should be very similar to:

```
DROP TRIGGER
DROP FUNCTION
DROP TRIGGER
DROP FUNCTION
DROP TABLE
CREATE TABLE
CREATE FUNCTION
CREATE TRIGGER
CREATE FUNCTION
CREATE TRIGGER
psql:triggerLab2Soln.sql:93: NOTICE:  Jim has no manager.
psql:triggerLab2Soln.sql:93: NOTICE:  Jim manages no one.
INSERT 0 1
psql:triggerLab2Soln.sql:95: NOTICE:  Frank has no manager.
psql:triggerLab2Soln.sql:95: NOTICE:  Frank manages no one.
INSERT 0 1
psql:triggerLab2Soln.sql:97: NOTICE:  Beth has no manager.
psql:triggerLab2Soln.sql:97: NOTICE:  Beth manages no one.
INSERT 0 1
psql:triggerLab2Soln.sql:99: NOTICE:  Jim manages no one.
UPDATE 1
psql:triggerLab2Soln.sql:100: NOTICE:  Frank manages no one.
UPDATE 1
psql:triggerLab2Soln.sql:102: NOTICE:  Salary adjusted for Frank.
psql:triggerLab2Soln.sql:102: NOTICE:  Frank manages no one.
```

```
UPDATE 1
 id | name  | mid | salary
----+-------+-----+--------
  1 | Beth  |     | 100000
  2 | Jim   |   1 |  75000
  3 | Frank |   1 | 100000
(3 rows)

psql:triggerLab2Soln.sql:106: NOTICE:  Beth has no manager.
psql:triggerLab2Soln.sql:106: NOTICE:  Salary adjusted for Beth.
UPDATE 1
 id | name  | mid | salary
----+-------+-----+--------
  2 | Jim   |   1 |  75000
  3 | Frank |   1 | 100000
  1 | Beth  |     | 175000
(3 rows)
```