PostgreSQL Triggers Lab II

Tom Kelliher, CS 318

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. On phoenix, copy the file `/home/kelliher/pub/cs317/triggerLab2.sql` to one of your directories.

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, cap the salary to the manager's salary and raise a notice. The cap is ignored if the employee has no manager, but a notice should be raised.

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:

```
psql:triggerLab2.sql:5: ERROR:  relation "payroll" does not exist
psql:triggerLab2.sql:6: ERROR:  function checkemployeefunc() does not exist
psql:triggerLab2.sql:9: ERROR:  table "payroll" does not exist
psql:triggerLab2.sql:23: NOTICE:  CREATE TABLE / PRIMARY KEY will create
implicit index "payroll_pkey" for table "payroll"
CREATE TABLE
CREATE FUNCTION
CREATE TRIGGER
psql:triggerLab2.sql:56: NOTICE:  Tom has no manager.
INSERT 0 1
psql:triggerLab2.sql:58: NOTICE:  Justin has no manager.
INSERT 0 1
psql:triggerLab2.sql:60: NOTICE:  Jill has no manager.
INSERT 0 1
UPDATE 1
UPDATE 1
```

(Continued on next page.)

```
psql:triggerLab2.sql:65: NOTICE:  Adjusting salary for Justin.
UPDATE 1
 id |  name  | sid | salary
----+--------+-----+--------
  1 | Jill   |     | 100000
  2 | Tom    |   1 |  75000
  3 | Justin |   1 | 100000
(3 rows)

psql:triggerLab2.sql:69: NOTICE:  Jill has no manager.
UPDATE 1
 id |  name  | sid | salary
----+--------+-----+--------
  2 | Tom    |   1 |  75000
  3 | Justin |   1 | 100000
  1 | Jill   |     | 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.

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
psql:triggerLab2.sql:23: NOTICE:  CREATE TABLE / PRIMARY KEY will create
implicit index "payroll_pkey" for table "payroll"
CREATE TABLE
CREATE FUNCTION
CREATE TRIGGER
CREATE FUNCTION
CREATE TRIGGER
psql:triggerLab2.sql:80: NOTICE:  Tom has no manager.
INSERT 0 1
psql:triggerLab2.sql:82: NOTICE:  Justin has no manager.
INSERT 0 1
psql:triggerLab2.sql:84: NOTICE:  Jill has no manager.
INSERT 0 1
```

(Continued on next page.)

```
UPDATE 1
UPDATE 1
psql:triggerLab2.sql:89: NOTICE:  Adjusting salary for Justin.
UPDATE 1
 id |  name  | sid | salary
----+--------+-----+--------
  1 | Jill   |     | 100000
  2 | Tom    |   1 |  75000
  3 | Justin |   1 | 100000
(3 rows)

psql:triggerLab2.sql:93: NOTICE:  Jill has no manager.
psql:triggerLab2.sql:93: NOTICE:  Adjusting salary for Jill
UPDATE 1
 id |  name  | sid | salary
----+--------+-----+--------
  2 | Tom    |   1 |  75000
  3 | Justin |   1 | 100000
  1 | Jill   |     | 175000
(3 rows)
```