CS1555 Recitation 8

Objective: To understand how triggers work in PostgreSQL and to practice writing triggers.

Before we start, download and run the script **Bank_DB.sql** from the course website to setup the database. The database instance is shown below:

Account

acc_no	**Ssn**	**Code**	**open_date**	**Balance**	**close_date**
123	123456789	1234	2008-09-10	500	null
124	111222333	1234	2009-10-10	1000	null

Loan

Ssn	**Code**	**open_date**	Amount	close_date
111222333	1234	2010-09-15	100	null

Bank

Code	Name	Addr
1234	Pitt Bank	111 University St

Customer

Ssn	Name	Phone	Addr	num_accounts
123456789	John	555-535-5263	100 University St	1
111222333	Mary	555-535-3333	20 University St	1

Alert

Alert_date	Balance	Loan

Notes:

- Triggers are defined on a single table in PostgreSQL.
- With the "*for each row*" option, the trigger is row-level. In this mode, there are 2 special variables **new** and **old** to refer to new and old tuples, respectively.
- If "for each row" is not specified, then the trigger is a statement trigger- i.e., the trigger is fired only once, when the triggering event is met, if the optional trigger constraint is met.
- The statements in the trigger function need to be properly ended with "**;**"
- In Oracle, in the trigger body, if you select or update the table that the trigger is being defined on, you would get an error saying "*table ... is mutating, trigger/function may not see it*". This is OK in PostgreSQL as long as you avoid indefinite recursion.
- PL/pgSQL is SQL enhanced with control statement like any high-level programming languages. Examples include: If-Then-Else, Loops, etc.

Part 1: Triggers

1. Create a trigger that, when a customer opens new account (s), updates the corresponding num_accounts, to reflect the total number of accounts this customer has.

2. To test how the trigger works, insert a new account for customer '123456789', then display the num_accounts of that customer. An example tuple may be with values ('333', '123456789', '1234', '2010-10-10', 300, null).

3. Similarly, create a trigger that, upon deleting an account, updates the corresponding num_accounts. To test the trigger, delete from the account entries for ssn='123456789'. Then check the value of num_accounts.

4. To test the trigger, delete from the account entries for ssn='123456789'. Then check the value of num_accounts.

5. Create a trigger that upon updating an account's balance, if the new balance is negative then sets the balance to 0 and create a new loan for the negative amount (for this database, assume that this can happen only once per day).

6. To test how the trigger works, update the balance of the account '124' to -50, then check the data in the Loan table.

7. Create two triggers for Account and Loan tables that upon any changes in the two tables, if the sum of balance amount over all accounts is less than double the sum of loan amount over all loans, create a new alert with current date, total balance amount and total loan amount (for this database, assume that this can happen only once per day).

8. To test the trigger, update the balance of the account '124' to 50, then check the data in the Alert table.