Maman 11 – 2016c

This document and the sql queries can be found here:

<https://github.com/EliArad/OOP/tree/master/20277%20Database%20systems/2016c/maman11>

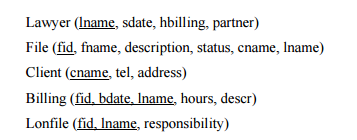
First, I need to create the tables.

I created a new database name law firm

Using DDL SQL queries, starting the following creation:

In Postgres there is no use like in sql server, so in order to write a query on the desired database do the following:

1. Connect to the database
2. Once it connected, select it and then select to open a query. This will cause the query to run on this database



Layer table:

CREATE TABLE Lawyer(

lname varchar(100) PRIMARY KEY NOT NULL,

sdate Date NOT NULL,

hbilling INTEGER NOT NULL,

partner Date

);

If I want to change the data type of a column for example:

ALTER TABLE Lawyer ALTER COLUMN lname TYPE varchar(150);

File table

CREATE TABLE File(

fid INT PRIMARY KEY NOT NULL,

fname varchar (150) NOT NULL,

description varchar (500) NOT NULL,

status Date ,

cname varchar (150) NOT NULL references Client(cname ) ,

lname varchar (150) NOT NULL references Lawyer(lname)

);

Two constrains:

1. foreign key to Layer ( lname column)
2. foreign key to Client (cname column)

CREATE TABLE Client(

cname varchar (150) PRIMARY KEY NOT NULL,

tel varchar (100) NOT NULL,

address varchar (500) NOT NULL

);

CREATE TABLE Lonfile(

fid INT NOT NULL references File(fid ),

lname varchar (100) NOT NULL references Lawyer(lname),

responsibility varchar (500) NOT NULL,

PRIMARY KEY(fid,lname)

);

Here we can see a post that shows how to do composite primary key in postgres:

<http://stackoverflow.com/questions/1285967/postgres-how-to-do-composite-keys>

ALTER TABLE Lonfile ADD CONSTRAINT Lonfile\_fid\_lname\_unique UNIQUE (fid, lname)

Also we can add constraint like that

After the table was created or during the create table

CREATE TABLE Billing(

fid INT NOT NULL references File(fid ),

bdate Date NOT NULL,

lname varchar (500) NOT NULL references Lawyer(lname),

hours INT NOT NULL ,

descr varchar (500) NOT NULL ,

PRIMARY KEY(fid, bdate, lname),

CONSTRAINT Billing\_fid\_bdate\_lname\_unique UNIQUE (fid, bdate, lname)

);

Now we need to create a trigger that do two things:

1. Inform when a new Billing row is inserted
2. If a lname in Billing row is updated than it check it possible and if so give message

To start working on this task I need some data in various table which are linked ( foreign) to Billing table

-- Lawyer

INSERT INTO Lawyer VALUES('Alex', '12-1-2005', 100, NULL);

INSERT INTO Lawyer VALUES('Eli', '3-2-2010', 190, '1-1-2001');

INSERT INTO Lawyer VALUES('Amit', '1-1-2002', 190, '1-1-2008');

INSERT INTO Lawyer VALUES('Ronen', '1-1-2004', 190, NULL);

INSERT INTO Lawyer VALUES('Ben', '1-1-1996', 190, '1-1-2001');

INSERT INTO Lawyer VALUES('Shalom', '1-1-2012', 190, NULL);

INSERT INTO Lawyer VALUES('Danny', '1-1-2015', 190, '2-2-2016');

INSERT INTO Lawyer VALUES('Amir', '1-1-2016', 190, NULL);

INSERT INTO Lawyer VALUES('Yonaton', '1-1-1984', 190, NULL);

INSERT INTO Lawyer VALUES('Shely', '1-1-2002', 190, '10-2-2004');

INSERT INTO Lawyer VALUES('Mor', '1-1-2003', 190, NULL);

INSERT INTO Lawyer VALUES('Dana', '1-1-2005', 190, '8-5-2008');

INSERT INTO Lawyer VALUES('Karin', '1-1-2009', 190, '2-3-2015');

For now we will insert one client:

INSERT INTO Client VALUES('alphi', '0504123133', 'Ramat-gan');

And create an open file for this client , handle by lawyer Eli:

INSERT INTO File VALUES(110323, 'Stolen', 'Stolen bag from old lady' , NULL , 'alphi111' , 'Eli');

Checking that this line will fail , because it violates the foreign key

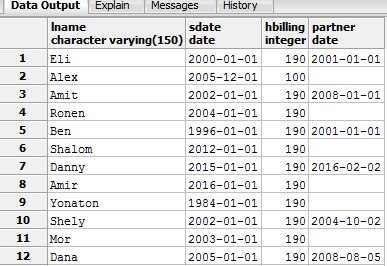
INSERT INTO File VALUES(110323, 'Stolen', 'Stolen bag from old lady' , NULL , 'alphi' , 'Eli');

We can check that data does exist before we continue:

select \* from File

select \* from Client

select \* from Lawyer







Back to the trigger:

Trigger in postgres declare operation that execute in a function

Before , after or instead of

First , we create a function new\_billing, that once it called it just raise a message to sql console:

CREATE OR REPLACE FUNCTION new\_billing() RETURNS TRIGGER AS $$

BEGIN

RAISE NOTICE 'Billing has a new row';

return null;

END;

$$ LANGUAGE plpgsql;

Then we create trigger also called new\_billing, that check that after a billing row is inserted , for each row it show the function message

CREATE TRIGGER new\_billing

AFTER INSERT ON Billing

FOR EACH ROW

EXECUTE PROCEDURE new\_billing();

INSERT INTO Billing VALUES(110323, '6-25-2016', 'Eli', 2, 'initial papers and arranging data');

For example, if I add those billing:

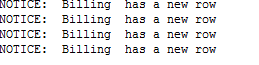
INSERT INTO Billing VALUES(110323, '6-25-2016', 'Eli', 2, 'initial papers and arranging data');

INSERT INTO Billing VALUES(110323, '6-1-2016', 'Eli', 1, 'initial papers and arranging data');

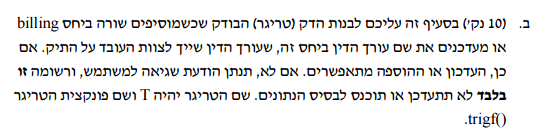
INSERT INTO Billing VALUES(110323, '6-2-2016', 'Eli', 1, 'initial papers and arranging data');

INSERT INTO Billing VALUES(110323, '6-3-2016', 'Eli', 2, 'initial papers and arranging data');

I am getting this notice:



In the next task



CREATE TRIGGER update\_lawyer\_name

BEFORE INSERT OR UPDATE ON Billing

FOR EACH ROW

EXECUTE PROCEDURE trigf ();

CREATE OR REPLACE FUNCTION trigf() () RETURNS TRIGGER AS $$

BEGIN

PERFORM 1 FROM File WHERE File.lname = NEW.lname LIMIT 1 ;

IF FOUND THEN

BEGIN

RAISE NOTICE 'new raw inserted';

END;

ELSE

BEGIN

IF EXISTS (SELECT 1 FROM Lonfile WHERE Lonfile.lname = NEW.lname) THEN

BEGIN

RAISE NOTICE 'update can continue';

END;

ELSE

RAISE EXCEPTION 'Lawyer name does not exists';

END IF;

END;

END IF;

**return NEW**;

END;

$$ LANGUAGE plpgsql;

**UPDATE Billing SET lname = 'Amit' WHERE fid = 110323 AND bdate = '6-25-2016'**

Some notes:

1. The מדריך למידה in page 118 has an example of before insert and update on.
2. The idea beyond this exercise is to be able to check if table File or Table Lonfile has the lawyer name before we can insert or update the Billing table.
3. The New is a postgres keyword for the new row.

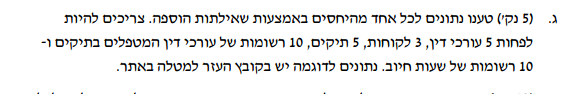
There are several ways to solve this.

Using foreach statement ( like in page 118) the not in is not working , so I drop this option

We can create instead of insert trigger to replace the insert statement in case we decides is ok to insert.

I used here If statement to check OR

The update is a use case . to fully test it we need to remove the lawyer from File and , or ,from Lonfile



* Lawyer

INSERT INTO Lawyer VALUES('Alex', '12-1-2005', 100, NULL);

INSERT INTO Lawyer VALUES('Eli', '3-2-2010', 190, '1-1-2001');

INSERT INTO Lawyer VALUES('Amit', '1-1-2002', 190, '1-1-2008');

INSERT INTO Lawyer VALUES('Ronen', '1-1-2004', 190, NULL);

INSERT INTO Lawyer VALUES('Ben', '1-1-1996', 190, '1-1-2001');

INSERT INTO Lawyer VALUES('Shalom', '1-1-2012', 190, NULL);

INSERT INTO Lawyer VALUES('Danny', '1-1-2015', 190, '2-2-2016');

INSERT INTO Lawyer VALUES('Amir', '1-1-2016', 190, NULL);

INSERT INTO Lawyer VALUES('Yonaton', '1-1-1984', 190, NULL);

INSERT INTO Lawyer VALUES('Shely', '1-1-2002', 190, '10-2-2004');

INSERT INTO Lawyer VALUES('Mor', '1-1-2003', 190, NULL);

INSERT INTO Lawyer VALUES('Dana', '1-1-2005', 190, '8-5-2008');

INSERT INTO Lawyer VALUES('Karin', '1-1-2009', 190, '2-3-2015');

* Client

INSERT INTO Client VALUES('alphi', '0504123133', 'Ramat-gan');

INSERT INTO Client VALUES('noam', '0534123133', 'Haifa');

INSERT INTO Client VALUES('arkady', '0524123133', 'Jerusalem');

INSERT INTO Client VALUES('lital', '0514123133', 'tel-aviv');

* File

INSERT INTO File VALUES(110323, 'Stolen', 'Stolen bag from old lady' , NULL , 'alphi' , 'Eli');

INSERT INTO File VALUES(120343, 'Killing', 'Mardring my panda' , NULL , 'noam' , 'Amir');

INSERT INTO File VALUES(120333, 'Stolen', 'Stolen car' , NULL , 'arkady' , 'Eli');

INSERT INTO File VALUES(120220, 'divorce', 'ugly divorce with 10 arkady' , NULL , 'alphi' , 'Shely');

INSERT INTO File VALUES(113210, 'Stolen', 'Stolen bag from old lady' , NULL , 'alphi' , 'Ben');

INSERT INTO File VALUES(914323, 'Stolen', 'Stolen bag from old lady' , NULL , 'lital' , 'Dana');

INSERT INTO File VALUES(112229, 'Pre-nuptial agreement', 'Stolen bag from old lady' , NULL , 'lital' , 'Karin');

* Lonfile

INSERT INTO LonFile VALUES(120343, 'Mor', 'adviser');

INSERT INTO LonFile VALUES(120333, 'Danny', 'must have second option');

INSERT INTO LonFile VALUES(110323, Amit, 'Bringing coffie');

INSERT INTO Billing VALUES(110323, '2-3-2016', 'Eli', 1 , 'reading the avidance');

INSERT INTO Billing VALUES(120343, '2-3-2016', 'Amir', 1 , 'reading the avidance');

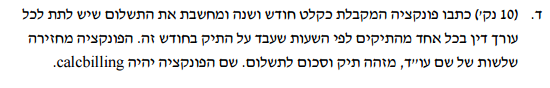
INSERT INTO Billing VALUES(120333, '2-3-2016', 'Eli', 1 , 'reading the avidance');

INSERT INTO Billing VALUES(120220, '2-3-2016', 'Shely', 1 , 'reading the avidance');

INSERT INTO Billing VALUES(113210, '2-3-2016', 'Ben', 1 , 'reading the avidance');

INSERT INTO Billing VALUES(914323, '2-3-2016', 'Dana', 1 , 'reading the avidance');

INSERT INTO Billing VALUES(112229, '2-3-2016', 'Karin', 1 , 'reading the avidance');



For this question, I needed to play with the data on the billing table, to have more rows and especially the some layers working on the same file.

INSERT INTO Billing VALUES(110323, '1-3-2016', 'Eli', 1 , 'reading the avidance');

INSERT INTO Billing VALUES(120343, '2-3-2016', 'Amir', 1 , 'reading the avidance');

INSERT INTO Billing VALUES(120333, '3-4-2016', 'Eli', 1 , 'reading the avidance');

INSERT INTO Billing VALUES(120220, '4-4-2016', 'Shely', 1 , 'reading the avidance');

INSERT INTO Billing VALUES(113210, '5-3-2016', 'Ben', 1 , 'reading the avidance');

INSERT INTO Billing VALUES(113210, '5-4-2016', 'Ben', 2 , 'reading the avidance');

INSERT INTO Billing VALUES(113210, '5-5-2015', 'Ben', 2 , 'reading the avidance');

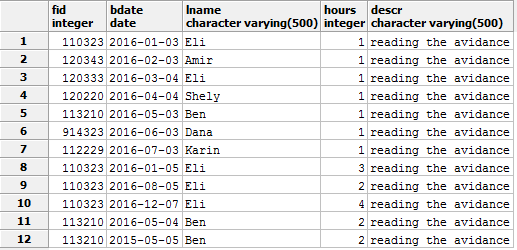
INSERT INTO Billing VALUES(914323, '6-3-2016', 'Dana', 1 , 'reading the avidance');

INSERT INTO Billing VALUES(112229, '7-3-2016', 'Karin', 1 , 'reading the avidance');

INSERT INTO Billing VALUES(110323, '1-5-2016', 'Eli', 3 , 'reading the avidance');

INSERT INTO Billing VALUES(110323, '8-5-2016', 'Eli', 2 , 'reading the avidance');

INSERT INTO Billing VALUES(110323, '12-7-2016', 'Eli', 4 , 'reading the avidance');



This will help us to see that Eli worked on a file 110323 several times abd Ben also worked on a the same fid more then once.

Before jumping into a function , I tested the following query:

select Billing.fid, Billing.lname, sum(Billing.hours) as totalHours , sum(Billing.hours\*lawyer.hbilling) as price from Billing

inner join lawyer

on Billing.lname = lawyer.lname

where Extract(month from Billing.bdate) >= 1 and Extract(year from Billing.bdate) = 2015

group by Billing.fid , Billing.lname

I also added the sum of all hours just to see it.

The result for the above query return only one row ( see the 2015 restriction)



From here we just need to warp it in a function and call the function with a parameters for year and month:

create type some\_type1 as (

fid int,

lname varchar,

totalHours int,

price int

);

DROP FUNCTION calcbilling(integer,integer)

create or replace function calcbilling(month int , year int) returns setof some\_type1 as

$$

declare

r some\_type1;

begin

for r in

select Billing.fid, Billing.lname, sum(Billing.hours) as totalHours , sum(Billing.hours\*lawyer.hbilling) as price from Billing

inner join lawyer

on Billing.lname = lawyer.lname

where Extract(month from Billing.bdate) >= month and Extract(year from Billing.bdate) = year

group by Billing.fid , Billing.lname

loop

return next r;

end loop;

return;

end;

$$

LANGUAGE plpgsql;

SELECT calcbilling(1, 2016)



select \* from lawyer

where Extract(year from partner) <= 2010



Check if we need to add more data like:

INSERT INTO LonFile VALUES(110323, 'Danny', 'student');

INSERT INTO LonFile VALUES(110323, 'Ben', 'student');

To make sure we can see the result well.

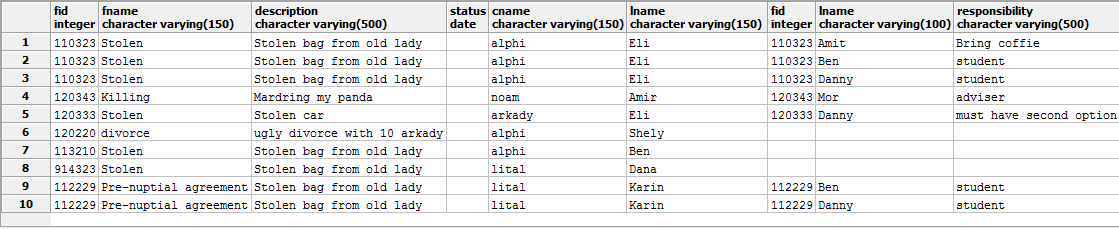
This query helps to see the intermidiate results:

select \* from File

left join lonfile

on lonfile.fid = file.fid

where file.status is null



According to the table defenitions, main responsibility is listed in the file relation and additional lawyer is added in the lonfile relation.

So to count up all layers , we have all the layers attached to the fid + 1 of the layer from File relation

This is the query:

select file.lname, lonfile.fid, (count(lonfile.fid) + 1) as total\_lawyers from File

left join lonfile

on lonfile.fid = file.fid

where file.status is null

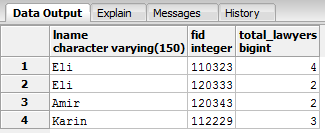
group by lonfile.fid, file.lname

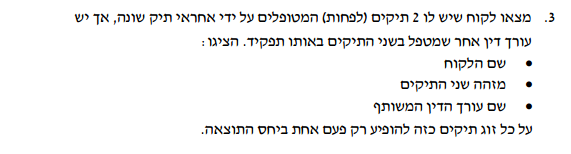
having (count(lonfile.fid) + 1) > 1

We group by fid and lname and adding + 1.

The left join is because we want all the lawyers from the file relation + all the lawyers from the lonfile relation

\*And present having > 1





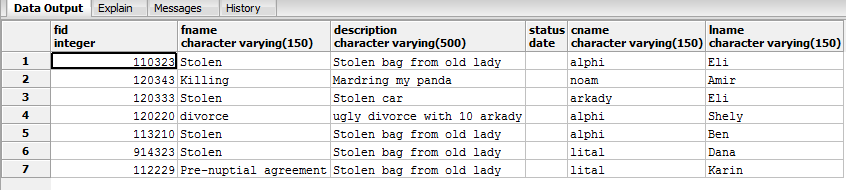
The way I am solving those queries is , first insert data that once the query will run it can be seen , means the query will find the specified injected data directly for this query.

The idea is to search for two different lawyers in the File relation.

This already brings to my mind that we need to use Cartesian product, why?

Because we need to compare the same attributes from the same table.

This is my initial table



From my initial File relation , I can see that customer alphi has two cases( different fid) and two different lawyers ( Eli and Shely , fis 110323 , 120220)

There are two Cartesian product in this query

The first one , compare from the file relation and the second one

Takes this sub query result and cross join it again to put the lonfile.lname in the same row

And again select the fields as requested.

select distinct count(q1.fid), q1.lon\_lname, q1.cname, q1.fid from (select f1.fid as fid, l1.lname as lon\_lname, l1.responsibility, f1.cname , f1.lname as f1lname , f2.lname as f2lname from file as f1 , file as f2, lonfile as l1

where f1.fid != f2.fid and f1.cname = f2.cname and l1.fid = f1.fid ) as q1,

(select f1.fid, l1.lname as lon\_lname, l1.responsibility, f1.cname , f1.lname as f1lname , f2.lname as f2lname from file as f1 , file as f2, lonfile as l1

where f1.fid != f2.fid and f1.cname = f2.cname and l1.fid = f1.fid ) as q2

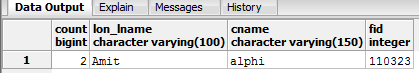
where q1.fid = q2.fid and q1.responsibility = q2.responsibility and

q1.fid = q2.fid and q1.lon\_lname = q2.lon\_lname and q1.f2lname = q2.f2lname and q1.f1lname = q2.f1lname

group by q1.fid, q1.lon\_lname , q1.cname

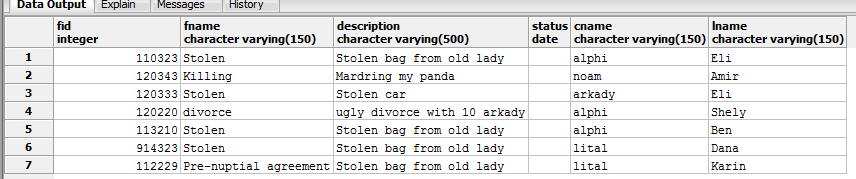
having count(q1.fid) > 1

The result I found

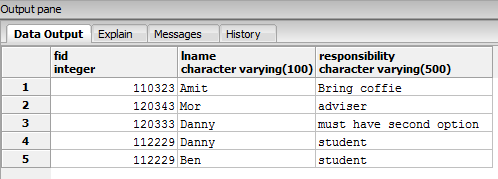


Input was:

File relation:



Lonfile relation



It is possible to do the sub query as CTE to reduce code size.