Functions and Triggers
PL/pgSQL
for Database Server

SAVONIA
AMMATTIKORKEAKOULU

Alexey Tukalo,
EFA12SF,
Information Technology,
Savonia University of Applied Sciences

October 25, 2014

Contents

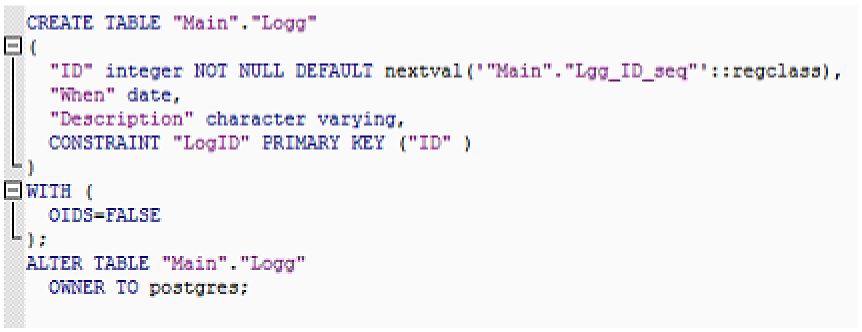
1 Logg table, function and triggers for it

1.1 Logg table

I have create the Logg table with three columns:

1. ID - integer Primery Key
2. When - date
3. Description - character varying

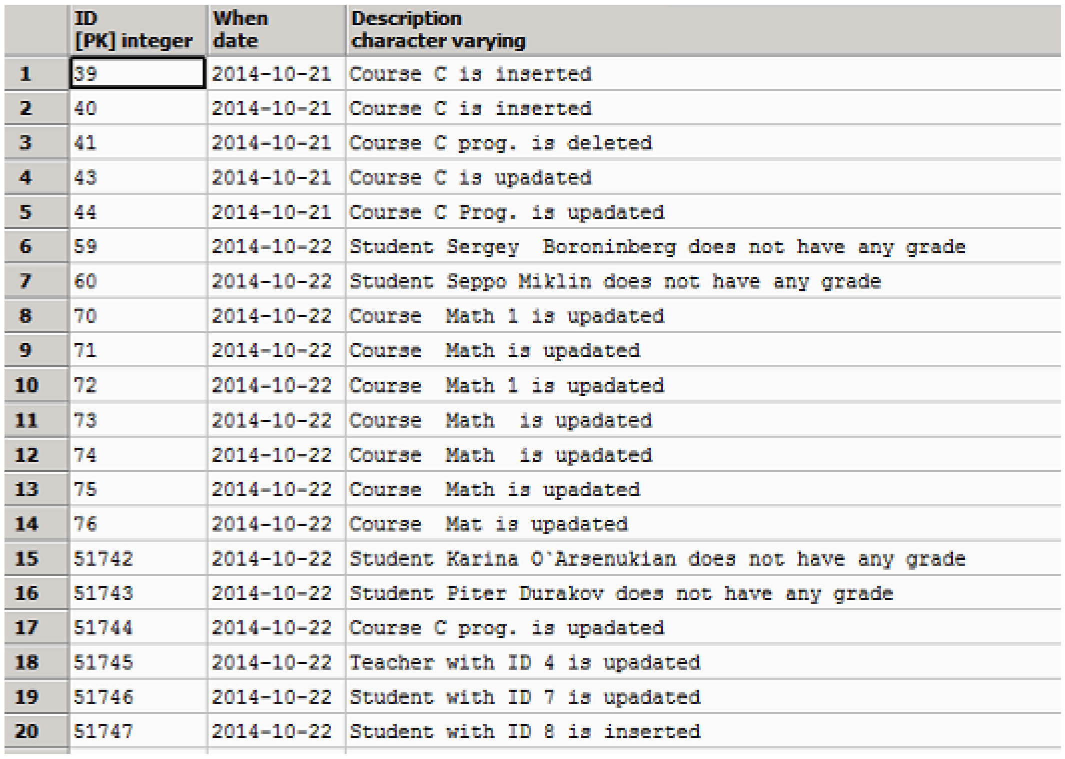
It was created by script on the picture 1.



```
CREATE TABLE "Main"."Logg"
(
  "ID" integer NOT NULL DEFAULT nextval('"Main"."Lgg_ID_seq"'::regclass),
  "When" date,
  "Description" character varying,
  CONSTRAINT "LogID" PRIMARY KEY ("ID" )
)
WITH (
  OIDS=FALSE
);
ALTER TABLE "Main"."Logg"
  OWNER TO postgres;
```

Figure 1: Script for Logg table

And the table itself is displayed on the figure 2.

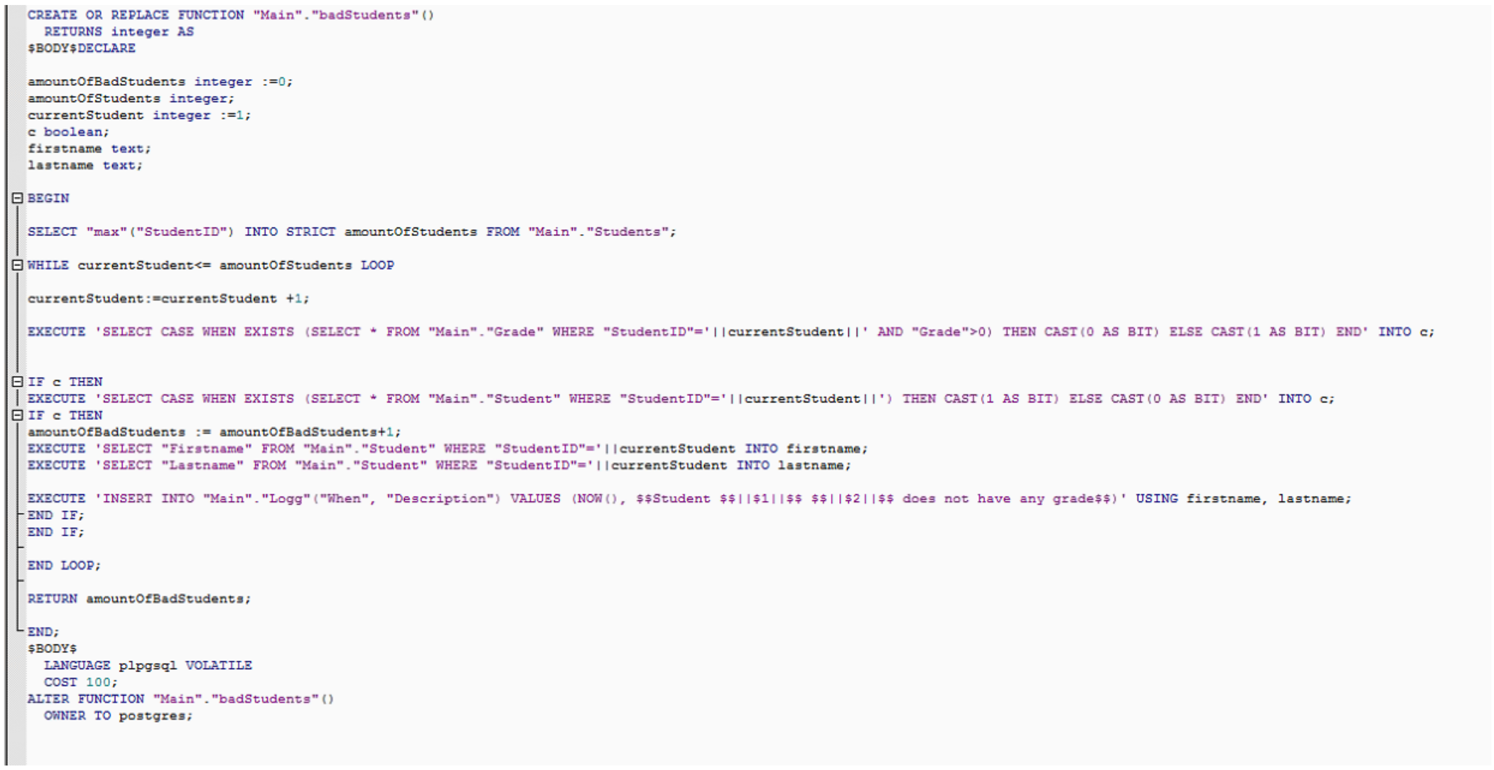


	ID [PK] integer	When date	Description character varying
1	39	2014-10-21	Course C is inserted
2	40	2014-10-21	Course C is inserted
3	41	2014-10-21	Course C prog. is deleted
4	43	2014-10-21	Course C is upadated
5	44	2014-10-21	Course C Prog. is upadated
6	59	2014-10-22	Student Sergey Boroninberg does not have any grade
7	60	2014-10-22	Student Seppo Miklin does not have any grade
8	70	2014-10-22	Course Math 1 is upadated
9	71	2014-10-22	Course Math is upadated
10	72	2014-10-22	Course Math 1 is upadated
11	73	2014-10-22	Course Math is upadated
12	74	2014-10-22	Course Math is upadated
13	75	2014-10-22	Course Math is upadated
14	76	2014-10-22	Course Mat is upadated
15	51742	2014-10-22	Student Karina O`Arsenukian does not have any grade
16	51743	2014-10-22	Student Piter Durakov does not have any grade
17	51744	2014-10-22	Course C prog. is upadated
18	51745	2014-10-22	Teacher with ID 4 is upadated
19	51746	2014-10-22	Student with ID 7 is upadated
20	51747	2014-10-22	Student with ID 8 is inserted

Figure 2: Logg table

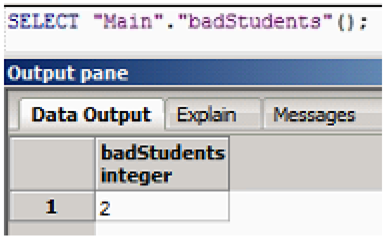
1.2 Function badStudents()

After that I have written the code for the function which return amount of students without any accepted course, also it insets a row per student into the Logg table, the description field may look like this ”Student XX YY does not have any grades”. You can see the code and output of the function on the figures below.



```
CREATE OR REPLACE FUNCTION "Main"."badStudents"()
  RETURNS integer AS
$BODY$DECLARE

amountOfBadStudents integer :=0;
amountOfStudents integer;
currentStudent integer :=1;
c boolean;
firstname text;
lastname text;

BEGIN

SELECT "max"("StudentID") INTO STRICT amountOfStudents FROM "Main"."Students";

WHILE currentStudent<= amountOfStudents LOOP

currentStudent:=currentStudent +1;

EXECUTE 'SELECT CASE WHEN EXISTS (SELECT * FROM "Main"."Grade" WHERE "StudentID"='||currentStudent||' AND "Grade">0) THEN CAST(0 AS BIT) ELSE CAST(1 AS BIT) END' INTO c;


IF c THEN
EXECUTE 'SELECT CASE WHEN EXISTS (SELECT * FROM "Main"."Student" WHERE "StudentID"='||currentStudent||') THEN CAST(1 AS BIT) ELSE CAST(0 AS BIT) END' INTO c;
IF c THEN
amountOfBadStudents := amountOfBadStudents+1;
EXECUTE 'SELECT "Firstname" FROM "Main"."Student" WHERE "StudentID"='||currentStudent INTO firstname;
EXECUTE 'SELECT "Lastname" FROM "Main"."Student" WHERE "StudentID"='||currentStudent INTO lastname;

EXECUTE 'INSERT INTO "Main"."Logg"("When", "Description") VALUES (NOW(), $$Student $$||$1||$$ $$||$2||$$ does not have any grade$$)' USING firstname, lastname;
END IF;
END IF;

END LOOP;

RETURN amountOfBadStudents;

END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION "Main"."badStudents"()
  OWNER TO postgres;
```

Figure 3: Code of badStudents()



```
SELECT "Main"."badStudents"();
```

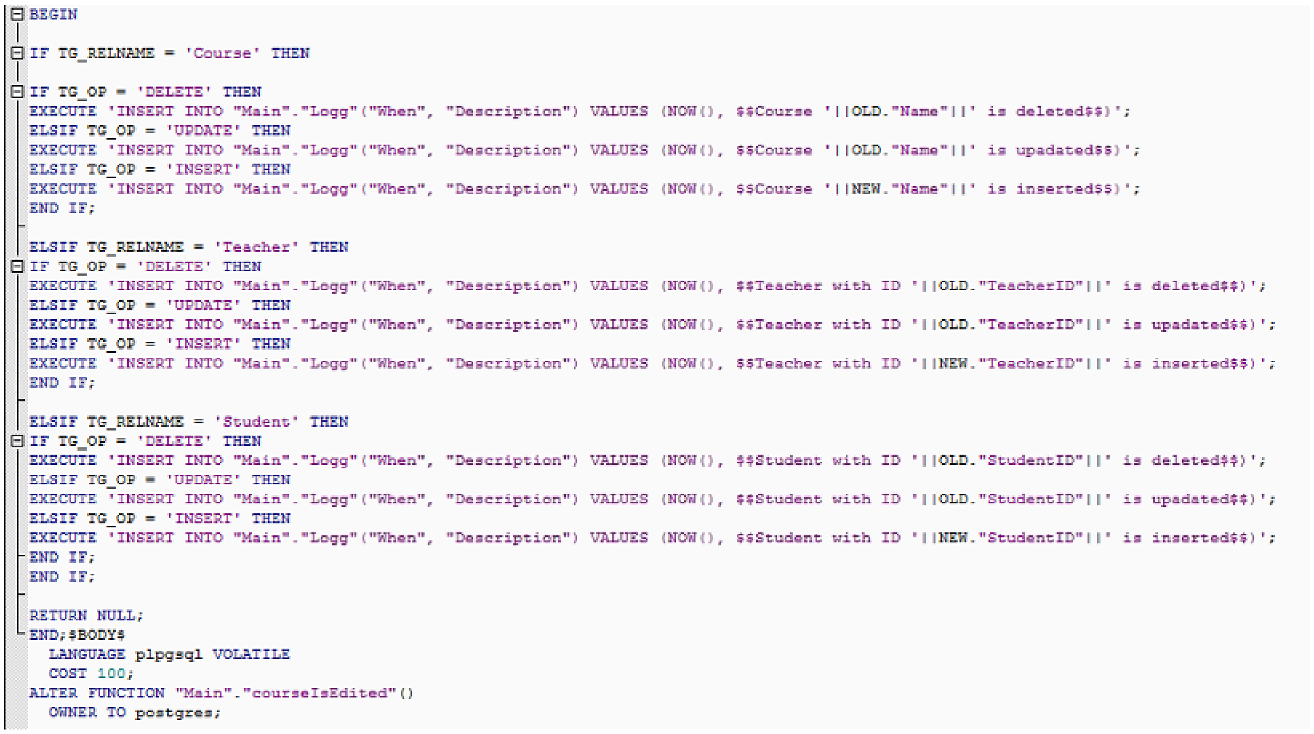
	badStudents integer
1	2

Figure 4: badStudents() output

After the execution of the function it also added rows number 15 and 16 on the Logg table, look at picture 2.

1.3 edited()

I also made a trigger edited() which inserts a row into the logg table if course information is inserted/updated/deleted. The content of the description field may look like this ” Course ZZZ is inserted/deleted/updated”. The output of the trigger you can find on the figure 2.



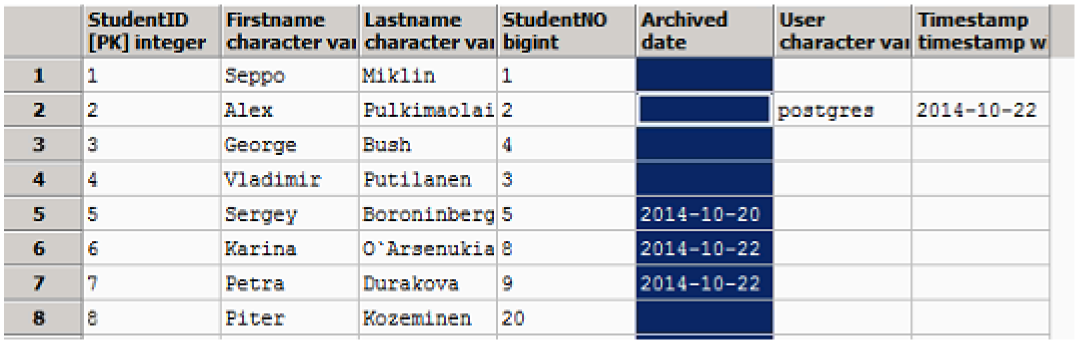
```
BEGIN

IF TG_RELNAME = 'Course' THEN

IF TG_OP = 'DELETE' THEN
EXECUTE 'INSERT INTO "Main"."Logg"("When", "Description") VALUES (NOW(), $$Course '||OLD."Name"||' is deleted$$)';
ELSIF TG_OP = 'UPDATE' THEN
EXECUTE 'INSERT INTO "Main"."Logg"("When", "Description") VALUES (NOW(), $$Course '||OLD."Name"||' is upadated$$)';
ELSIF TG_OP = 'INSERT' THEN
EXECUTE 'INSERT INTO "Main"."Logg"("When", "Description") VALUES (NOW(), $$Course '||NEW."Name"||' is inserted$$)';
END IF;

ELSIF TG_RELNAME = 'Teacher' THEN
IF TG_OP = 'DELETE' THEN
EXECUTE 'INSERT INTO "Main"."Logg"("When", "Description") VALUES (NOW(), $$Teacher with ID '||OLD."TeacherID"||' is deleted$$)';
ELSIF TG_OP = 'UPDATE' THEN
EXECUTE 'INSERT INTO "Main"."Logg"("When", "Description") VALUES (NOW(), $$Teacher with ID '||OLD."TeacherID"||' is upadated$$)';
ELSIF TG_OP = 'INSERT' THEN
EXECUTE 'INSERT INTO "Main"."Logg"("When", "Description") VALUES (NOW(), $$Teacher with ID '||NEW."TeacherID"||' is inserted$$)';
END IF;

ELSIF TG_RELNAME = 'Student' THEN
IF TG_OP = 'DELETE' THEN
EXECUTE 'INSERT INTO "Main"."Logg"("When", "Description") VALUES (NOW(), $$Student with ID '||OLD."StudentID"||' is deleted$$)';
ELSIF TG_OP = 'UPDATE' THEN
EXECUTE 'INSERT INTO "Main"."Logg"("When", "Description") VALUES (NOW(), $$Student with ID '||OLD."StudentID"||' is upadated$$)';
ELSIF TG_OP = 'INSERT' THEN
EXECUTE 'INSERT INTO "Main"."Logg"("When", "Description") VALUES (NOW(), $$Student with ID '||NEW."StudentID"||' is inserted$$)';
END IF;
END IF;

RETURN NULL;
END;$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION "Main"."courseIsEdited"()
  OWNER TO postgres;
```

Figure 5: edited() source code

2 Archived column and function for it

2.1 Archived column

After that I have added new column for Student table, the column is selected on the 6th picture.

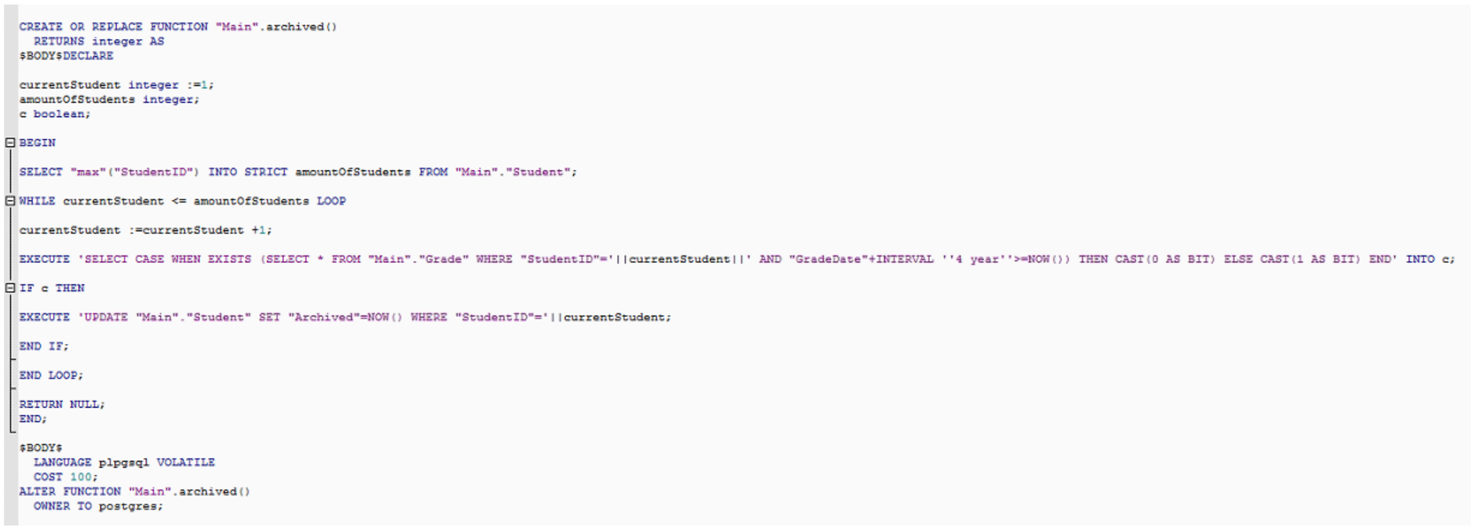


	StudentID [PK] integer	Firstname character vai	Lastname character vai	StudentNO bigint	Archived date	User character vai	Timestamp timestamp w
1	1	Seppo	Miklin	1			
2	2	Alex	Pulkimaolai	2		postgres	2014-10-22
3	3	George	Bush	4			
4	4	Vladimir	Putilanen	3			
5	5	Sergey	Boroninberg	5	2014-10-20		
6	6	Karina	O`Arsenukia	8	2014-10-22		
7	7	Petra	Durakova	9	2014-10-22		
8	8	Piter	Kozeminen	20			

Figure 6: Student Table with Archived column

2.2 Function archived()

At the next step I have written the code below and after the execution the function added dates into rows from 5th to 7th on the column Archived.

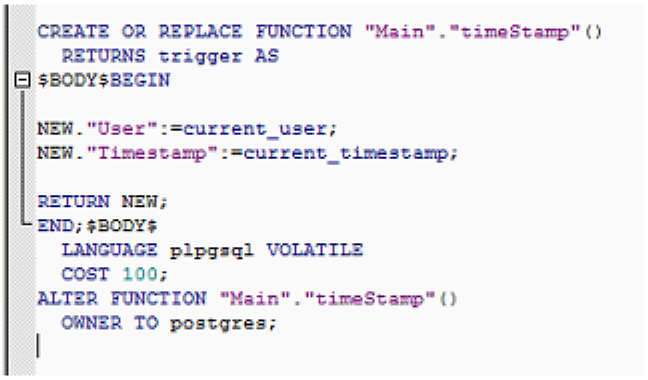


```
CREATE OR REPLACE FUNCTION "Main".archived()
  RETURNS integer AS
$BODY$DECLARE

currentStudent integer :=1;
amountOfStudents integer;
c boolean;

BEGIN

SELECT "max"("StudentID") INTO STRICT amountOfStudents FROM "Main"."Student";

WHILE currentStudent <= amountOfStudents LOOP

currentStudent :=currentStudent +1;

EXECUTE 'SELECT CASE WHEN EXISTS (SELECT * FROM "Main"."Grade" WHERE "StudentID"='||currentStudent||' AND "GradeDate"+INTERVAL ''4 year''>=NOW()) THEN CAST(0 AS BIT) ELSE CAST(1 AS BIT) END' INTO c;

IF c THEN

EXECUTE 'UPDATE "Main"."Student" SET "Archived"=NOW() WHERE "StudentID"='||currentStudent;

END IF;

END LOOP;

RETURN NULL;
END;

$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION "Main".archived()
  OWNER TO postgres;
```

Figure 7: Source code for archived()

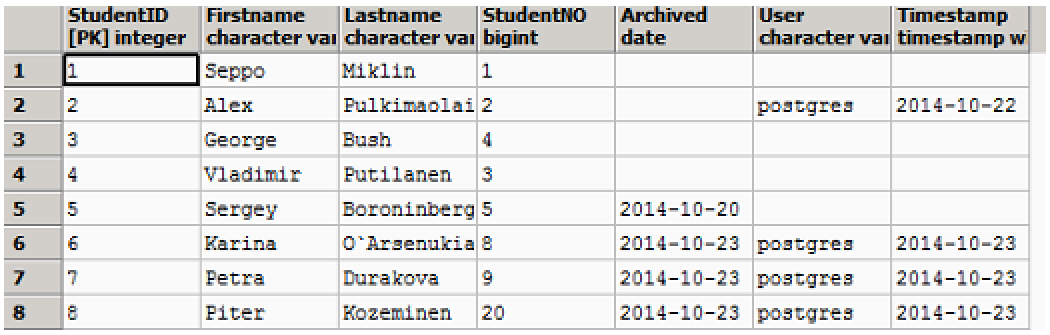
3 Timestamp and User columns

I have added the columns for Timestamp and User for tables Course, Teacher, Student. And the trigger code you can see on the picture 8. The result of the code is represented on the 9th figure.

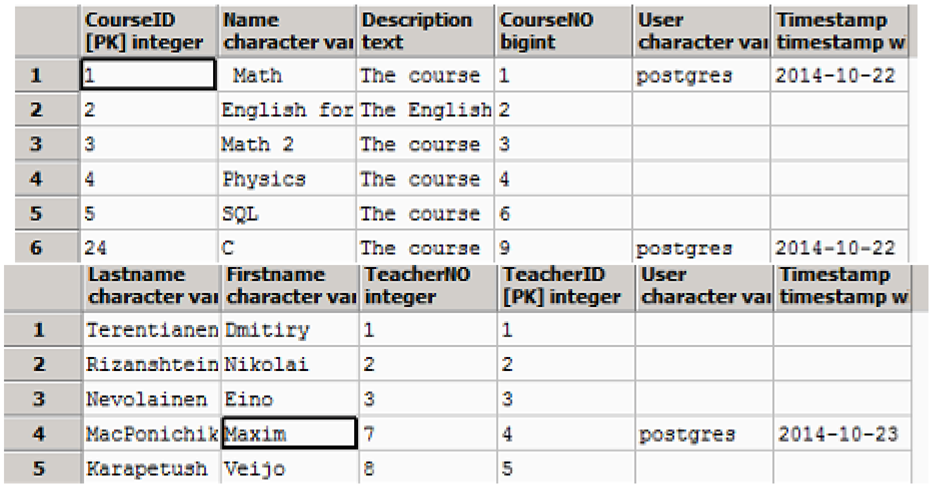


```
CREATE OR REPLACE FUNCTION "Main"."timeStamp"()
  RETURNS trigger AS
$BODY$BEGIN

NEW."User":=current_user;
NEW."Timestamp":=current_timestamp;

RETURN NEW;
END;$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION "Main"."timeStamp"()
  OWNER TO postgres;
```

Figure 8: timeStamp() source code



	StudentID [PK] integer	Firstname character vai	Lastname character vai	StudentNO bigint	Archived date	User character vai	Timestamp timestamp w
1	1	Seppo	Miklin	1			
2	2	Alex	Pulkimaolai	2		postgres	2014-10-22
3	3	George	Bush	4			
4	4	Vladimir	Putilanen	3			
5	5	Sergey	Boroninberg	5	2014-10-20		
6	6	Karina	O`Arsenukia	8	2014-10-23	postgres	2014-10-23
7	7	Petra	Durakova	9	2014-10-23	postgres	2014-10-23
8	8	Piter	Kozeminen	20	2014-10-23	postgres	2014-10-23



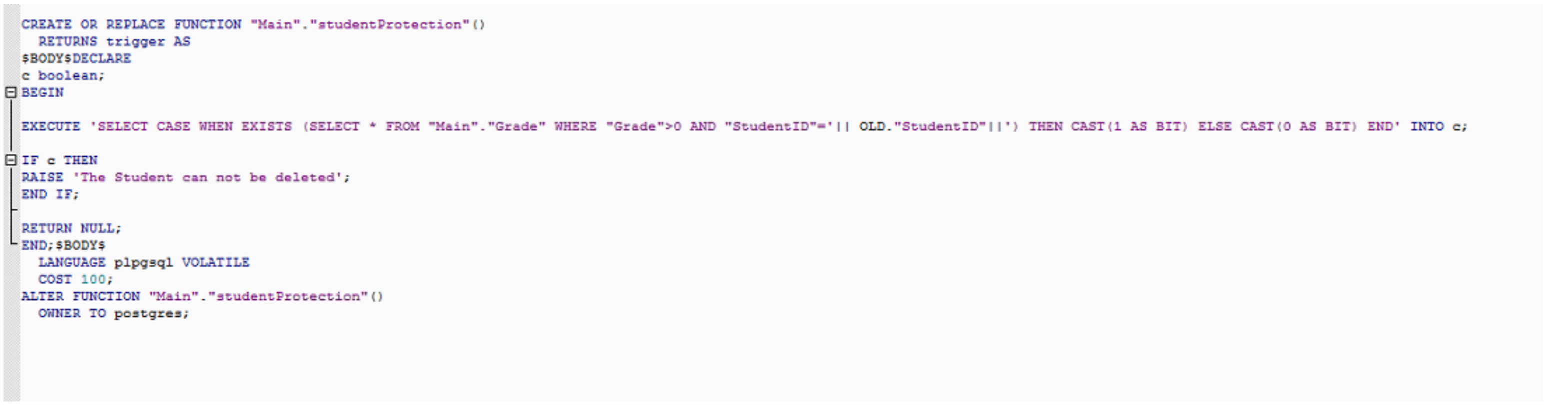
	CourseID [PK] integer	Name character vai	Description text	CourseNO bigint	User character vai	Timestamp timestamp w
1	1	Math	The course	1	postgres	2014-10-22
2	2	English for	The English	2		
3	3	Math 2	The course	3		
4	4	Physics	The course	4		
5	5	SQL	The course	6		
6	24	C	The course	9	postgres	2014-10-22

	Lastname character vai	Firstname character vai	TeacherNO integer	TeacherID [PK] integer	User character vai	Timestamp timestamp w
1	Terentianen	Dmitiry	1	1		
2	Rizanshtein	Nikolai	2	2		
3	Nevolainen	Eino	3	3		
4	MacPonichik	Maxim	7	4	postgres	2014-10-23
5	Karapetush	Veijo	8	5		

Figure 9: Tables with the columns

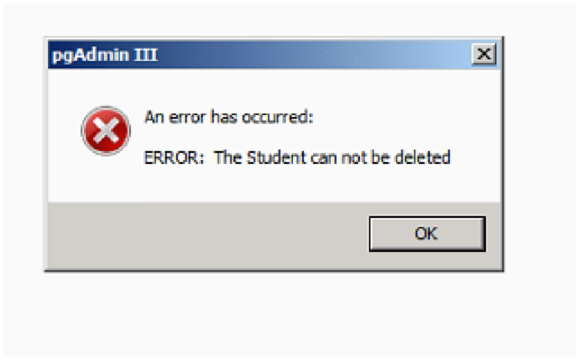
4 Student protection

The last task asked to make the trigger which protects students with accepted courses from deleting. The function has to check it via query and show error message if it is impossible to delete the student, picture 10.



```
CREATE OR REPLACE FUNCTION "Main"."studentProtection"()
  RETURNS trigger AS
$BODY$DECLARE
c boolean;
BEGIN

EXECUTE 'SELECT CASE WHEN EXISTS (SELECT * FROM "Main"."Grade" WHERE "Grade">0 AND "StudentID"='|| OLD."StudentID"||') THEN CAST(1 AS BIT) ELSE CAST(0 AS BIT) END' INTO c;

IF c THEN
RAISE 'The Student can not be deleted';
END IF;

RETURN NULL;
END;$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION "Main"."studentProtection"()
  OWNER TO postgres;
```

Figure 10: studentProtection() source code



```
pgAdmin III

An error has occurred:

ERROR:  The Student can not be deleted
```

Figure 11: studentProtection() Error message