Yunus Emre Ak / 1306150001 / Veri Tabanı 4.Ödev - Trigger

```sql
        -- 1. soru
-- Student tablosundaki GPA değerini gerektiğinde
güncelleyen triggerları yazınız
-- (Take tablosundaki sid değiştiğinde ve take
tablosuna kayıt eklenip silindiğinde çalışması
-- yeterli)

-- GPA Updater Function
CREATE FUNCTION update_gpa() RETURNS
TRIGGER AS $$
DECLARE
id int; -- Id of student
BEGIN
-- If operation is DELETE
IF (TG_OP = 'DELETE') THEN
id = OLD.sid;
-- Otherwise
ELSE
id = NEW.sid;
END IF;

-- Calculate new GPAs what students have
WITH newgpa as (
SELECT SUM(credits * grade) / SUM(credits) FROM
Take, Course
WHERE id = Take.sid AND Take.cid = Course.cid
GROUP BY Take.sid
)

UPDATE Student SET Student.gpa = newgpa
WHERE sid = id;
END;
$$ LANGUAGE plpgsql;

-- Create trigger to update each student
CREATE TRIGGER gpa_update
AFTER INSERT OR UPDATE OR DELETE ON Take
FOR EACH ROW
EXECUTE PROCEDURE update_gpa();


-- 2 Soru (?)
-- Course tablosundaki did için CREATE TABLE
komutunda FOREIGN KEY yazılmadığını kabul edip,
-- “did FOREIGN KEY references Department(did)
ON DELETE CASCADE” yazılmış olsaydı,
-- “(i) Department tablosundan kayıt silindiğinde o
bölümün derslerini course tablosunda da
-- silen ve (ii) Course tablosuna INSERT veya (did
alanı) UPDATE yapıldığında” veritabanı
-- sistemi tarafından otomatik yapılacak işlem ve
kontrolleri yapacak TRIGGERları yazınız.
-- (Eğer bir bolumun course tablosunda öğrencisi
varsa ve o bölüm department tablosundan delete
-- edilmeye çalışılıyorsa buna izin vermeyiniz, yani
hata üretiniz. Hata üretmek için EXCEPTION
-- throw edebilirsiniz. PostgreSQL
dokumantasyonuna bakınız)

CREATE FUNCTION delete_from_course() RETURNS
TRIGGER AS $$
BEGIN
   Delete From Course c
   where c.did = OLD.did;
   RETURN OLD;
END
$$ LANGUAGE plpgsql;

CREATE TRIGGER gpa_update
BEFORE DELETE ON department
FOR EACH ROW
EXECUTE PROCEDURE delete_from_course();

-------------

CREATE OR REPLACE FUNCTION ogrenci_sayisi(
   id integer)
RETURNS integer AS $$
declare toplam_sayi int;
BEGIN
select count(s.sid)
into toplam_sayi
from student s
where s.did in (
select d.did
from department d
where d.did = id
);
return toplam_sayi;
END;
$$ LANGUAGE plpgsql;

-- 3 Soru
-- tid’si verilen bir hocanın verdiği dersi alan
öğrencilerin kayıtlarını döndüren
-- stored functıon’ı yazınız. Bu fonksiyonu
herhangi bir sorguda kullanınız.

CREATE FUNCTION get_students(tid int) RETURNS
SETOF Student AS $$
BEGIN
RETURN QUERY SELECT Student.* FROM Teach,
Take, Student
WHERE Teach.tid = tid AND Teach.cid = Take.cid
AND Take.sid = Student.sid
GROUP BY tid;
END;
$$ LANGUAGE plpgsql;


-- 4. Soru (?)
-- Department tablosuna yapılan INSERT, UPDATE
ve DELETE komutlarının hangi gün ve saatte
```

Yunus Emre Ak / 1306150001 / Veri Tabanı 4.Ödev - Trigger

```
-- apıldığını log(tarihSaat, komut) tablosunda
yedekleyen (yani INSERT, UPDATE ve DELETE
-- komutlarından biri çalıştırılınca log tablosuna
INSERT yapan) statement level TRIGGERlari
-- yazınız (derste çözmüştük)

CREATE TABLE log (tarihSaat TIMESTAMP, komut
VARCHAR(77));

CREATE OR REPLACE FUNCTION add_log()
RETURNS TRIGGER AS $$
Declare
id int;
BEGIN
      IF (TG_OP = 'DELETE') THEN
         Insert Into log values(OLD.did,
current_timestamp ,TG_OP);
         RETURN OLD;
ELSE
       Insert Into log values(NEW.did,
current_timestamp ,TG_OP);
RETURN NEW;
End IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER update_happens
BEFORE UPDATE or DELETE or INSERT
ON department
FOR EACH ROW
EXECUTE PROCEDURE add_log();


-- 5. Soru (Odev4.java)
-- Teacher tablosundaki kayıtları listeleyen, tid'si
verilen bir kaydı silen, yeni kayıt
-- ekleyen, tid'si verilen bir kaydın bilgilerini
güncelleyen Java konsol uygulamasını
-- PostgrSQL JDBC kütüphanesini kullanarak
yazınız.


// Yunus Emre Ak
// 1306150001
// Odev4 - Trigger
// Tüm kodlar ve açıklamalar bana aittir!

import java.util.Scanner;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;
```

```
import java.util.Properties;

class Odev4 {
// Connection variable
private static Connection conn = null;

// Input var for user reaction
public static Scanner input = null;

// For Visuality
private final static String FORMAT_FIRST = "%-
7s";
private final static String FORMAT_ELSE = "%-
30s";

/**
* Main method s
*
* @param args Console arguments
*/
public static void main(String args[]) {
// If can't connect force app to exit
if (!connectPSQL()) {
System.out.println("Connection error! Cannot
connect database.");
System.exit(-1);
}

// Start the interface
ui();

// Close connection
closeConnection();
}

/**
* Try to connect postgreSQL
*
* @return True if can, otherwise false
*/
public static boolean connectPSQL() {
try {
// Loading the driver
Class.forName("org.postgresql.Driver");

// Setting url
String url = "jdbc:postgresql://localhost/odev4";

// Setting properties of database
Properties props = new Properties();
props.setProperty("user", "yemreak");
props.setProperty("password", "123");

// Create a connection to postgreSQL database
conn = DriverManager.getConnection(url, props);

return true;
```

```java
} catch (ClassNotFoundException | SQLException
e) {
System.out.println(e);
return false;
}
}

/**
* Close connection safely
*/
public static void closeConnection() {
try {
if (conn != null) {
conn.close();
newRow();
System.out.println("Connection is closed
succesfuly.");
}
} catch (SQLException e) {
System.out.println(e);
}
}

/**
* User interface
*/
public static void ui() {
// Define the input var
input = new Scanner(System.in);

// Define and initlialise answer var
boolean loop = true;

// UI
while (loop) {
newRow();
System.out.println("Main Menu");
newRow();
System.out.println("1- List");
System.out.println("2- Add");
System.out.println("3- Delete");
System.out.println("4- Update");
System.out.println("0- Exit");
newRow();
System.out.print("-> ");

// Getting the answer from user input
answer = input.nextInt();

switch (answer) {
case 1:
uiList();
break;
case 2:
uiAdd();
break;
case 3:
uiDelete();
break;
case 4:
uiUpdate();
break;
}
}

// Closing the input
input.close();
}

/**
* List all teachers which is one of the table of the
database
*/
public static void uiList() {
try {
// Creating sql statement and result set to store it
and result set meta data to
// get names of columns
Statement st = conn.createStatement();
ResultSet rs = st.executeQuery("SELECT * FROM
Teacher");
ResultSetMetaData rsmd = rs.getMetaData();

// For visuality
for (int i = 1; i <= rsmd.getColumnCount(); i++) {
if (i == 1) {
System.out.printf(FORMAT_FIRST,
rsmd.getColumnName(i));
} else {
System.out.printf(FORMAT_ELSE,
rsmd.getColumnName(i));
}
}

// For new line
System.out.println();

// For visuality
for (int i = 1; i <= rsmd.getColumnCount(); i++) {
if (i == 1) {
System.out.printf(FORMAT_FIRST, "---");
} else {
System.out.printf(FORMAT_ELSE, "-----------");
}
}

// For new line
System.out.println();

// Processing result set
while (rs.next()) {
// Write all column
for (int i = 1; i <= rsmd.getColumnCount(); i++) {
if (i == 1) {
```

```
System.out.printf(FORMAT_FIRST, rs.getString(i));
} else {
System.out.printf(FORMAT_ELSE, rs.getString(i));
}
}
// New line
System.out.print("\n");
}

System.out.print("\n");

rs.close();
st.close();
} catch (SQLException e) {
System.out.println(e);
}
}

/**
* The interface addition teacher to database
*/
public static void uiAdd() {
try {
// User answer
int id;
String name;
String birthPlace;

newRow();
System.out.println("Id of the teacher?");
System.out.print("-> ");
id = input.nextInt();

// Catch the \n error
input.nextLine();

newRow();
System.out.println("Name of the teacher?");
System.out.print("-> ");
name = input.nextLine();

newRow();
System.out.println("BirthPlace of the teacher?");
System.out.print("-> ");
birthPlace = input.nextLine();

// Prepare statement with our inputs
PreparedStatement ps =
conn.prepareStatement("INSERT INTO Teacher
VALUES(?, ?, ?)");
ps.setInt(1, id);
ps.setString(2, name);
ps.setString(3, birthPlace);

// Execute the sql
ps.executeUpdate();
ps.close();

// Shows the response of db
newRow();
System.out.println("Teacher has been created");

} catch (SQLException e) {
// Shows the response of db
newRow();
System.out.println("Teacher cant be created");
System.out.println(e);
}
}

/**
* The interface of deletion teacher from database
via id
*/
public static void uiDelete() {
try {
// User answer
int id;

newRow();
System.out.println("Id of the teacher who you
want to delete?");
System.out.print("-> ");
id = input.nextInt();

// Prepare statement with our inputs
PreparedStatement ps =
conn.prepareStatement("DELETE FROM Teacher
WHERE tid = ?");
ps.setInt(1, id);

// Execute the sql
if (ps.executeUpdate() > 0) {
// Shows the response of db
newRow();
System.out.println("Teacher has been deleted");
} else {
// Shows the response of db
newRow();
System.out.println("No deletion made. May ID
wrong?");
}
ps.close();

} catch (SQLException e) {
// Shows the response of db
newRow();
System.out.println("Teacher cant be deleted.
Database error!");
System.out.println(e);
}
}

/**
```

```
* The interface of update user in the database via
id
*/
public static void uiUpdate() {
try {
// User answer
int id;
String name;
String birthPlace;

newRow();
System.out.println("Id of the teacher who you
want to update?");
System.out.print("-> ");
id = input.nextInt();

// Catch the \n error
input.nextLine();

newRow();
System.out.println("New name of the teacher?");
System.out.print("-> ");
name = input.nextLine();

newRow();
System.out.println("New birthPlace of the
teacher?");
System.out.print("-> ");
birthPlace = input.nextLine();

// Prepare statement with our inputs
PreparedStatement ps =
conn.prepareStatement("UPDATE Teacher SET
name = ?, placeOfBirth = ? WHERE tid = ?");
ps.setString(1, name);
ps.setString(2, birthPlace);
ps.setInt(3, id);

// Execute the sql
if (ps.executeUpdate() > 0) {
// Shows the response of db
newRow();
System.out.println("Teacher has been deleted");
} else {
// Shows the response of db
newRow();
System.out.println("No update made. May ID
wrong?");
}
ps.close();

} catch (SQLException e) {
// Shows the response of db
newRow();
System.out.println("Teacher cant be changed.
Database error!");
System.out.println(e);
}
}

/**
* Writes "-----" row to console
*/
public static void newRow() {
System.out.println("----------------");
}

}
```