SQL

- יצירת יחסים והכנסת רשומות חדשות
- שליפת נתונים מתוך יחס אחד
- שליפת נתונים מתוך מספר יחסים
- פעולת הקבצה ופונקציות הקבצה
- פונקציות שימושיות
- ⇐ טריגרים ופונקציות בפורמט PostgreSQL
- תרגול

טריגרים ופונקציות בפורמט PostgreSQL

כאשר משתמש מנסה לעדכן או לשנות נתונים בבסיס הנתונים, חשוב לוודא שהשינויים שהוא מנסה לבצע עומדים בחוקיות שהוגדרה מראש. בקרות כאלו ניתן לבצע בשני מקומות:

1. **בדיקות בצד הלקוח (Client Side)**

בדיקות אלו מתבצעות בממשק המשתמש (כמו אפליקציה או טופס באתר), ונועדו למנוע מראש הזנה שגויה. לדוגמה:

אימות שכתובת אימייל מכילה את התו '@' , בדיקה שמספר טלפון מורכב אך ורק מספרים, מתחיל ב־'0' ובאורך נכון.

בדיקות אלו אינן קשורות ישירות לבסיס הנתונים, ונעשות לפני שמתקיים ניסיון לשמור את הנתונים בפועל.

2. **בדיקות בצד השרת (Server Side)**

ישנם תנאים שחייבים להיבדק רק בצד השרת, כלומר — בתוך בסיס הנתונים עצמו. לדוגמה:

מניעת הזנה של כתובת דוא"ל שכבר קיימת בטבלה. מניעת הזנה של תעודת זהות כפולה (שמפרה מפתח ראשי). מניעת רישום לתור לרופא שכבר תפוס בשעה מסוימת.

לשם כך נשתמש ב־טריגרים (Triggers). טריגר הוא מנגנון בבסיס הנתונים שמופעל באופן אוטומטי כאשר מתרחשת פעולה מסוימת (INSERT / UPDATE / DELETE) על טבלה. מטרתו לבצע בדיקה או פעולה לפני או אחרי שהתבצעה הפעולה.

טריגר ב־PostgreSQL מורכב משני חלקים:

1. **פונקציה (Function)** - החלק שבו נכתבת הלוגיקה של הבדיקה. אם התנאים מתקיימים – הפעולה מאושרת (RETURN NEW) , ואם לא – הפעולה נבלמת (RETURN NULL).
2. **טריגר (Trigger)** - ההגדרה שמציינת **מתי** תופעל הפונקציה (לפני או אחרי (INSERT/UPDATE/DELETE) ועל **אילו רשומות**).

חשוב: הפונקציה תמיד צריכה להיווצר **לפני** יצירת הטריגר, כדי שהטריגר "יכיר" את הפונקציה שהוא צריך לקרוא אליה.

טריגרים ופונקציות בפורמט PostgreSQL

טריגר ופונקציה בסיסית מורכבים מהחלקים שלהלן:

create or replace function trig_name() returns trigger as$$	פסוק ליצירת הטריגר trig_name
declare	פתיחת אזור להכרזה על משתנים. אם אין שימוש במשתנים אין צורך ב-declare
varName1 integer; varName2 boolean; varName3 numeric(4,2); varName4 varchar(20);	שמות המשתנים והסוג שלהם. • משתנה מסוג numeric משמש למספרים עשרוניים, יש להגדיר את כמות הספרות בסך הכל (בדוגמא-4) וכן את כמות הספרות אחרי הנקודה (בדוגמא-2), לדוגמא עבור המספר 123.58 יש להגדיר: numeric(5,2). • משתנה מסוג varchar משמש לקליטת מילים, יש להגדיר את אורך המילה המקסימלית האפשרית (בדוגמא-20).
begin	התחלת הפונקציה עצמה. גוף הפונקציה יכול לכלול לולאות for, משפטי if, ופעולות נוספות כמו הכנסה של רשומה ליחס כלשהו: insert into table_name values(A,B,C,D…)
If(condition) then	התחלת משפט "אם". בתוך התנאי אפשר לבדוק האם מתקיים משהו בתוך יחס הקיים במערכת (באמצעות שאילתה רגילה), אפשר להשוות את אחד הערכים בשורה החדשה שמנסים להכניס לבסיס הנתונים, לדוגמא: if(new.columnA in (select columnA from tale_name))
raise notice 'error message for the user';	הקפצת הודעת שגיאה למשתמש
return null;	החזרת ערך ריק למשתמש ומניעת הכנסת הרשומה החדשה לבסיס הנתונים
end if;	סיום משפט "אם"
return new;	הכנסת הרשומה החדשה לבסיס הנתונים
end; $$ language plpgsql;	סיום הפונקציה

create trigger T1 before insert or update on table_name	יצירת הטריגר T1 וקביעה מתי הוא יקרה (before/after) ועבור איזה סוגים של אירועים (הכנסה, עדכון או מחיקה של רשומה) ועל איזו טבלה הטריגר יופעל
for each row	האם לבצע את הטריגר לכל שורה בנפרד: for each row, או לכל ההוראה ביחד: for each statement
when (condition)	אם רוצים להפעיל את הטריגר רק במקרים מסוימים אפשר להשתמש ב- when (לא חובה)
execute procedure trig_name();	זימון הפונקציה והפעלתה (אפשר לשלוח לפונקציה פרמטרים לשימושה בתוך הסוגריים)

טריגרים ופונקציות בפורמט PostgreSQL

אם נניח שלכל רופא יש בכל יום רק תור אחד שבו הוא מקבל, ונרצה לוודא שאין שני קליינטים הנרשמים לאותו התאריך:

```
CREATE OR REPLACE FUNCTION trigf1() RETURNS TRIGGER AS $$
BEGIN
        IF (EXISTS (SELECT *
                FROM treatment
                WHERE new.did=did AND new.tdate=tdate))
        THEN
                RAISE NOTICE 'התור המבוקש תפוס, אנא נסה תאריך או רופא אחרים';
                RETURN NULL;
        END IF;
                RETURN NEW;
END; $$ LANGUAGE PLPGSQL;

CREATE TRIGGER T1 BEFORE INSERT ON treatment
FOR EACH ROW
EXECUTE PROCEDURE trigf1();
```

doctor (did, dname, dfamily, dphone, dcity, dstreet, dhouse, dspeciality, ddatestart, dsalary)
patient (pid, pname, pfamily, pbirthdate)
clinic (cid, ccity, caddress, cname)
treatment (tdate, did, pid, cid)

טריגרים ופונקציות בפורמט PostgreSQL

נרצה לוודא שבכל עדכון לשכר של רופא מתקיימים התנאים הבאים:

1. רופא לא יכול לעבור את תקרת השכר של 60,000 ש"ח.
2. אם השכר עולה ביותר מ-10% לעומת השכר הקודם, נרצה לרשום הודעה (RAISE NOTICE) עם פרטי הרופא והשינוי באחוזים.
3. התנאים האלה צריכים להיבדק רק אם באמת עודכן השכר.

```
CREATE OR REPLACE FUNCTION trigf2() RETURNS trigger AS$$
DECLARE
    old_salary INTEGER := OLD.dsalary;
    new_salary INTEGER := NEW.dsalary;
    raise_percent NUMERIC;
BEGIN
    IF new_salary > 60000 THEN
        RAISE EXCEPTION  'שכר חדש (% ש"ח) גבוה מהמותר' , new_salary;
    END IF;
    raise_percent := ((new_salary - old_salary) * 100.0) / old_salary;
    IF raise_percent > 10 THEN
        RAISE NOTICE  'הרופא % % קיבל העלאה של %. (מ- % ל- %)'   ,  NEW.dname, NEW.dfamily, raise_percent, old_salary, new_salary;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER T2 BEFORE UPDATE ON doctor
FOR EACH ROW
WHEN (NEW.dsalary > OLD.dsalary AND NEW.dsalary IS DISTINCT FROM OLD.dsalary)
EXECUTE FUNCTION trigf2() ;
```

טריגרים ופונקציות בפורמט PostgreSQL

נרצה לוודא שבכל הוספה או עדכון של רופא, שהשכר החדש לא גבוה מ-50% מהשכר הממוצע בתחום ההתמחות של הרופא הנ"ל

```
CREATE OR REPLACE FUNCTION trigf3() RETURNS trigger AS$$
DECLARE
        avg_speciality_salary NUMERIC;
BEGIN
   SELECT AVG(dsalary)
   INTO avg_speciality_salary
   FROM doctor
   WHERE dspeciality = NEW.dspeciality;
   IF new_salary > 1.5 * avg_speciality_salary THEN
      RAISE EXCEPTION   'New salary %. is too high for speciality % (avg = %) ' , NEW.dsalary, NEW.dspeciality, avg_speciality_salary;
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql
```

```
CREATE TRIGGER T3 BEFORE INSERT OR UPDATE ON doctor
FOR EACH ROW
EXECUTE FUNCTION trigf3() ;
```

טריגרים ופונקציות בפורמט PostgreSQL

אפשר להשתמש בפונקציה גם ללא טריגר, לדוגמה, אם נרצה לכתוב פונקציה המקבלת כקלט מספר מזהה של רופא ושני תאריכים ומחזירה את כמות הטיפולים שהרופא ביצע בין תאריכים אלו:

```
CREATE OR REPLACE FUNCTION count_treatments (doctor_id INT, start_date DATE, end_date DATE) RETURNS INT AS$$
DECLARE
    treatment_count INT;
BEGIN
    SELECT COUNT(*)
    INTO treatment_count
    FROM treatment
    WHERE did = doctor_id
      AND tdate BETWEEN start_date AND end_date;

    RETURN treatment_count;
END;
 $$LANGUAGE plpgsql;
```

בכדי להשתמש בפונקציה נוכל לעשות זאת כך:

```
SELECT count_treatments (101, '2023-01-01', '2023-12-31')
```