NESTED SUBQUERIES και πιο ΣΥΝΘΕΤΑ ΕΡΩΤΗΜΑΤΑ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

Αντώνιος Δεληγιαννάκης

Υποερωτήματα - SUBQUERIES

Στο WHERE μπορείτε να έχετε άλλα SELECT-FROM-WHERE υποερωτήματα

- **Χρειάζονται παρενθέσεις** στα υποερωτήματα
- Μπορούμε να ελέγξουμε αν για το κάθε ένα tuple που εξετάζουμε ξεχωριστά στο WHERE ισχύει μία πολύπλοκη συνθήκη

Παραδείγματα Υποερωτημάτων

Μπορούμε να ελέγξουμε αν για το κάθε ένα tuple που εξετάζουμε ξεχωριστά στο WHERE ισχύει μία πολύπλοκη συνθήκη

Παραδείγματα συνθηκών – Έλεγξε αν … :

- Έχει ένα attribute του tuple τιμή που επιστρέφεται από ένα άλλο SQL ερώτημα
- Έχει ένα attribute του tuple τιμή που ΔΕΝ επιστρέφεται από ένα άλλο SQL ερώτημα
- Υπάρχει έστω 1 tuple στο αποτέλεσμα ενός άλλου ερωτήματος που χρησιμοποιεί τιμές του τωρινού tuple σε συνθήκες (πχ στο WHERE clause του)
- Δεν υπάρχει κανένα tuple στο αποτέλεσμα ενός άλλου ερωτήματος που χρησιμοποιεί τιμές του τωρινού tuple σε συνθήκες (πχ στο WHERE clause του)
- συν άλλες συνθήκες …

Υποερωτήματα - SUBQUERIES

Τελεστές που θα δούμε:

- IN και NOT IN
- EXISTS και NOT EXISTS
- ANY
- ALL

Τελεστής IN
Σημείωση: ΜΟΝΟ 1 attribute στο IN

Βρες τα sID, ονόματα όλων των μαθητών που έχουν τον ίδιο μέσο όρο με κάποιον Jim

```
SELECT  sID, sName
FROM  Student
WHERE GPA IN   (SELECT GPA
                FROM STUDENT
                WHERE sName = 'Jim')
```

Εκτελείται για κάθε tuple του WHERE

sID	sName
1	Jim
3	Mary
4	Jim

GPA
3.5
3.8

College

cName	state	enr
Mary	MD	1000
MIT	MA	17000
Jim	CH	12000

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

Apply

sID	cName	major	dec
1	Mary	CS	YES
1	MIT	MPD	NO
2	MIT	EE	NO
2	Mary	CS	YES

Τελεστής IN – Πώς Εκτελείται;

Κάθε tuple που ελέγχει το WHERE αντιστοιχεί σε 1 μαθητή

Ελέγχουμε αν το γνώρισμα GPA του μαθητή έχει τιμή που βρίσκεται μέσα στα αποτελέσματα του Nested Subquery

```
SELECT  sID, sName
FROM  Student
WHERE GPA IN  (SELECT GPA
               FROM STUDENT
               WHERE sName = 'Jim')
```

sID	sName
1	Jim

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

Βρίσκεται το GPA=3.5 μέσα σε αυτό το αποτέλεσμα;

GPA
3.5
3.8

Τελεστής IN – Πώς Εκτελείται;

Κάθε tuple που ελέγχει το WHERE αντιστοιχεί σε 1 μαθητή

Ελέγχουμε αν το γνώρισμα GPA του μαθητή έχει τιμή που βρίσκεται μέσα στα αποτελέσματα του Nested Subquery

```
SELECT  sID, sName
FROM  Student
WHERE GPA IN  (SELECT GPA
               FROM STUDENT
               WHERE sName = 'Jim')
```

sID	sName
1	Jim

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

Βρίσκεται το GPA=3.7 μέσα σε αυτό το αποτέλεσμα;

GPA
3.5
3.8

Τελεστής IN – Πώς Εκτελείται;

Κάθε tuple που ελέγχει το WHERE αντιστοιχεί σε 1 μαθητή

Ελέγχουμε αν το γνώρισμα GPA του μαθητή έχει τιμή που βρίσκεται μέσα στα αποτελέσματα του Nested Subquery

```
SELECT  sID, sName
FROM  Student
WHERE GPA IN  (SELECT GPA
               FROM STUDENT
               WHERE sName = 'Jim')
```

sID	sName
1	Jim
3	Mary

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

GPA
3.5
3.8

Βρίσκεται το GPA=3.5 μέσα σε αυτό το αποτέλεσμα;

Τελεστής IN – Πώς Εκτελείται;

Κάθε tuple που ελέγχει το WHERE αντιστοιχεί σε 1 μαθητή

Ελέγχουμε αν το γνώρισμα GPA του μαθητή έχει τιμή που βρίσκεται μέσα στα αποτελέσματα του Nested Subquery

```
SELECT  sID, sName
FROM  Student
WHERE GPA IN  (SELECT GPA
               FROM STUDENT
               WHERE sName = 'Jim')
```

sID	sName
1	Jim
3	Mary
4	Jim

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

GPA
3.5
3.8

Βρίσκεται το GPA=3.8 μέσα σε αυτό το αποτέλεσμα;

Τελεστής IN

Γιατί το παρακάτω ΔΕΝ είναι ισοδύναμο ερώτημα;

```
SELECT  sID, sName
FROM  Student
WHERE GPA =  (SELECT GPA
              FROM STUDENT
              WHERE sName = 'Jim')
```

ΠΟΤΕ δε βάζουμε = αν το subquery επιστρέφει > 1 tuples

sID	sName
1	Jim
3	Mary

Σκεφτείτε ότι πολλοί μαθητές μπορεί να έχουν όνομα Jim

Έλεγχος μόνο με το 1ο GPA που επιστρέφεται στο υποερώτημα

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

Τελεστής NOT IN

Βρες τα sID, ονόματα όλων των μαθητών που ΔΕΝ έχουν τον ίδιο μέσο όρο με κάποιον Jim

```
SELECT  sID, sName
FROM  Student
WHERE GPA NOT IN   (SELECT GPA
                    FROM STUDENT
                    WHERE sName = 'Jim')
```

sID	sName
2	Peter

GPA
3.5
3.8

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

Σημαντικό Σχόλιο

Τα IN και NOT IN επιτρέπουν έλεγχο **σε 1 γνώρισμα μόνο**

Αυτός είναι ο κανόνας, και πρέπει τα ερωτήματά σας να ακολουθούν αυτόν τον κανόνα

Κάποια ΣΔΒΔ παραβιάζουν τον κανόνα και επιτρέπουν να ελέγχεις αν πολλά γνωρίσματα παίρνουν ταυτόχρονα τιμή που υπάρχει ως tuple σε ένα ερώτημα που επιστρέφει πολλά γνωρίσματα

ΔΕΝ πρέπει να γράφετε έτσι ερωτήματα

- Το ερώτημά σας δε θα δουλεύει ούτε καν σε παλαιότερες εκδόσεις του ίδιου ΣΔΒΔ, αν παραβιάσετε τον κανόνα

EXISTS και NOT EXISTS

Πώς κάνω σωστά έλεγχο σε πολλά γνωρίσματα ταυτόχρονα;

- EXISTS
- NOT EXISTS

Τελεστής EXISTS

Τυπικά θα αναφερθείτε μέσα στο υποερώτημα σε attribute του εξωτερικού ερωτήματος. Αντίστοιχα και στο NOT EXISTS

```
SELECT  A1, A2, ..., An
FROM  R1, R2, …, Rm
WHERE EXISTS   (SELECT B1, B2,  …, Br
                FROM V1, V2, …, Vc
                WHERE Vi.B1 = Rj.A3 and …. )
```

Τοπικό Attribute (Vi.B1)

και

Εξωτερικό Attribute (Rj.A3)

Τελεστής EXISTS

Θυμηθείτε ότι στο WHERE ελέγχουμε 1-1 τα tuples που προκύπτουν από το JOIN των πινάκων στο FROM

Κάθε ένα τέτοιο tuple στο WHERE θα έχει πολλά γνωρίσματα, και τιμή σε καθένα από αυτά

Μπορούμε να αναφερθούμε μέσα στο Nested Subquery EXISTS στις τιμές των γνωρισμάτων αυτού του tuple με Ri.ONOMA_ATTRIBUTE

```
SELECT  A1, A2, …, An
FROM  R1, R2, …, Rm
WHERE EXISTS   (SELECT B1, B2,  …, Br
                FROM V1, V2, …, Vc
                WHERE Vi.B1 = Rj.A3 and …. )
```

EXISTS: ΣΗΜΑΝΤΙΚΟ

Αν το ερώτημα μέσα στο EXISTS δεν αναφέρεται σε κάποιο γνώρισμα του tuple που ελέγχουμε στο WHERE, ή δε χρειάζεστε EXISTS, ή το ερώτημά σας είναι λάθος

- Συνήθως θα θέλετε να αναφερθείτε σε >= 2 γνωρίσματα του tuple που ελέγχετε στο WHERE
- Αλλιώς (για 1) θα μπορούσατε να είχατε χρησιμοποιήσει IN

Το EXISTS ικανοποιείται αν το ερώτημα επιστρέφει 1 ή περισσότερα tuples

- Δεν έχει σημασία πόσα γνωρίσματα επιστρέφει το φωλιασμένο ερώτημα

Τελεστής EXISTS

Βρες τα sID όλων των μαθητών που έχουν κάνει αίτηση σε κολλέγια στα οποία δεν είναι μοναδικοί υποψήφιοι (θα πρέπει και κάποιος άλλος να έχει κάνει αίτηση εκεί)

```
SELECT  DISTINCT sID
FROM  Apply A1
WHERE EXISTS  (SELECT *  FROM APPLY A2
        WHERE A2.sID <> A1.sID and
              A2.cName = A1.cName)
```

Γιατί DISTINCT;

Τοπικό Attribute και Εξωτερικό Attribute

sID
1
3
4

Apply

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
3	MIT	EE	NO
4	TUC	CS	YES
5	EMP	CS	NO

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

Τελεστής EXISTS – Πώς εκτελείται;

```
SELECT  DISTINCT sID
FROM  Apply A1
WHERE EXISTS  (SELECT *  FROM APPLY A2
                  WHERE A2.sID <> A1.sID and
                        A2.cName = A1.cName)
```

Apply

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
3	MIT	EE	NO
4	TUC	CS	YES
5	EMP	CS	NO

Βήμα 1: Γίνεται το Join των πινάκων στο FROM

- Εδώ έχουμε μόνο 1 πίνακα

Βήμα 2: Σκεφτείτε ότι γίνεται ένα For Loop στα tuples που προκύπτουν από το JOIN και ελέγχεται το EXISTS σε 1-1 από αυτά

- Για κάθε tuple, χρησιμοποιούμε τις τιμές του στα εξωτερικά γνωρίσματα A1.sID, A1.cName

Τελεστής EXISTS – Πώς εκτελείται;

```
SELECT  DISTINCT sID
FROM  Apply A1
WHERE EXISTS   (SELECT *  FROM APPLY A2
                  WHERE A2.sID <> A1.sID and
                        A2.cName = A1.cName)
```

sID
1

Apply

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
3	MIT	EE	NO
4	TUC	CS	YES
5	EMP	CS	NO

Ελέγχω αν υπάρχει έστω ένα tuple στο

```
SELECT *  FROM APPLY A2
WHERE A2.sID <> 1 and
      A2.cName = 'TUC'
```

ΝΑΙ. Θα επιστραφεί η 4η εγγραφή του Apply

Τελεστής EXISTS – Πώς εκτελείται;

```
SELECT  DISTINCT sID
FROM  Apply A1
WHERE EXISTS  (SELECT *  FROM APPLY A2
                  WHERE A2.sID <> A1.sID and
                        A2.cName = A1.cName)
```

sID
1

Apply

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
3	MIT	EE	NO
4	TUC	CS	YES
5	EMP	CS	NO

Ελέγχω αν υπάρχει έστω ένα tuple στο

```
SELECT *  FROM APPLY A2
WHERE A2.sID <> 1 and
      A2.cName = 'MIT'
```

ΝΑΙ. Θα επιστραφεί η 3η εγγραφή του Apply

Το DISTINCT δε θα προσθέσει το sID = 1

Τελεστής EXISTS – Πώς εκτελείται;

```
SELECT  DISTINCT sID
FROM  Apply A1
WHERE EXISTS   (SELECT *  FROM APPLY A2
                  WHERE A2.sID <> A1.sID and
                        A2.cName = A1.cName)
```

sID
1
3

Apply

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
3	MIT	EE	NO
4	TUC	CS	YES
5	EMP	CS	NO

Ελέγχω αν υπάρχει έστω ένα tuple στο

```
SELECT *  FROM APPLY A2
WHERE A2.sID <> 3 and
      A2.cName = 'MIT'
```

ΝΑΙ. Θα επιστραφεί η 2η εγγραφή του Apply

Τελεστής EXISTS – Πώς εκτελείται;

```
SELECT  DISTINCT sID
FROM  Apply A1
WHERE EXISTS  (SELECT *  FROM APPLY A2
                  WHERE A2.sID <> A1.sID and
                        A2.cName = A1.cName)
```

sID
1
3
4

Apply

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
3	MIT	EE	NO
4	TUC	CS	YES
5	EMP	CS	NO

Ελέγχω αν υπάρχει έστω ένα tuple στο

```
SELECT *  FROM APPLY A2
WHERE A2.sID <> 4 and
      A2.cName = 'TUC'
```

ΝΑΙ. Θα επιστραφεί η 1η εγγραφή του Apply

Τελεστής EXISTS – Πώς εκτελείται;

```
SELECT  DISTINCT sID
FROM  Apply A1
WHERE EXISTS   (SELECT *  FROM APPLY A2
                  WHERE A2.sID <> A1.sID and
                        A2.cName = A1.cName)
```

sID
1
3
4

Apply

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
3	MIT	EE	NO
4	TUC	CS	YES
5	EMP	CS	NO

Ελέγχω αν υπάρχει έστω ένα tuple στο

```
SELECT *  FROM APPLY A2
WHERE A2.sID <> 5 and
      A2.cName = 'EMP'
```

ΟΧΙ. Δεν επιστρέφει καμία εγγραφή

EXISTS και IN μαζί…

Βρες τα sID, ονόματα όλων των μαθητών που έχουν κάνει αίτηση σε κολλέγια στα οποία δεν είναι μοναδικοί υποψήφιοι
(θα πρέπει και κάποιος άλλος να έχει κάνει αίτηση εκεί)

sID	sName
1	Jim
3	Mary
4	Jim

```
SELECT  sID, sName
FROM  Student
WHERE sID IN
(SELECT  DISTINCT sID
    FROM  Apply A1
    WHERE EXISTS   (SELECT *  FROM APPLY A2
                        WHERE A2.sID <> A1.sID and
                              A2.cName = A1.cName))
```

Αυτό το subquery βρίσκει τα sID μόνο

Ίδιο Ερώτημα

```
SELECT  DISTINCT   S.sID , S.sName
FROM  Apply A1, Student S
WHERE S.sID = A1.sID and EXISTS
    (SELECT *  FROM APPLY A2
     WHERE A2.sID <> A1.sID and
     A2.cName = A1.cName)
```

S.sID	S.sName
1	Jim
3	Mary
4	Jim

Apply

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
3	MIT	EE	NO
4	TUC	CS	YES
5	EMP	CS	NO

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

ALL

Βρες τα sID, ονόματα των μαθητών με το μεγαλύτερο μέσο όρο

sID	sName
4	Jim

```
SELECT  sID, sName
FROM  Student
WHERE GPA >= ALL (SELECT  GPA  FROM  Student )
```

ALL: Η σύγκριση πρέπει να πετύχει για ΌΛΑ τα tuples του υποερωτήματος

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

ALL

Είναι ισοδύναμα τα ακόλουθα ερωτήματα;

```
SELECT  sID, sName
FROM  Student
WHERE GPA >= ALL (SELECT  GPA  FROM  Student )
```

```
SELECT  sID, sName
FROM  Student s1
WHERE GPA >    ALL (SELECT  GPA  FROM  Student s2
                    WHERE  s1.sID <> s2.sID)
```

ΟΧΙ! Σκεφτείτε μαθητές με ίδιο GPA

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

ANY

Βρες τα sID, ονόματα των μαθητών που δεν έχουν το μικρότερο μέσο όρο

```
SELECT  sID, sName
FROM  Student
WHERE GPA > ANY (SELECT  GPA  FROM  Student )
```

ANY: Η σύγκριση αρκεί να πετύχει για 1 tuple του υποερωτήματος

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

ALL & ANY

Δεν υποστηρίζουν όλα τα ΣΔΒΔ τα ALL και ANY

Ας δούμε πώς μπορούμε να ζήσουμε χωρίς αυτά …

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

ALL

```
SELECT  sID, sName
FROM  Student
WHERE GPA >= ALL (SELECT  GPA  FROM  Student )
```

Ισοδύναμο με ....

```
SELECT  sID, sName
FROM  Student s1
WHERE GPA = (SELECT  MAX(GPA)  FROM  Student )
```

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

ALL

```
SELECT  sID, sName
FROM  Student
WHERE GPA <= ALL (SELECT  GPA  FROM  Student )
```

Ισοδύναμο με ....

```
SELECT  sID, sName
FROM  Student s1
WHERE GPA = (SELECT  MIN(GPA)  FROM  Student )
```

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

ANY

```
SELECT  sID, sName
FROM  Student
WHERE GPA > ANY (SELECT  GPA  FROM  Student )
```

Ισοδύναμο με ....

```
SELECT  sID, sName
FROM  Student
WHERE GPA > (SELECT  MIN(GPA)  FROM  Student )
```

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

ANY

```
SELECT  sID, sName
FROM  Student
WHERE GPA < ANY (SELECT  GPA  FROM  Student )
```

Ισοδύναμο με ....

```
SELECT  sID, sName
FROM  Student
WHERE GPA < (SELECT  MAX(GPA)  FROM  Student )
```

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

ALL & ANY

Θυμηθείτε ότι συγκρίσεις κάνουμε στο WHERE και στο HAVING

- Στο WHERE μπορούμε να συγκρίνουμε την τιμή ενός attribute
- Στο HAVING μπορούμε να συγκρίνουμε την τιμή κάποιας συναθροιστικής τιμής που υπολογίζεται για κάθε ομάδα
- Πώς θα υπολογίζαμε τα ονόματα των κολλεγίων στα οποία έχουν γίνει οι περισσότερες αιτήσεις;
  - Όλη η πληροφορία υπάρχει στον πίνακα Apply
  - Θέλουμε αριθμό αιτήσεων, άρα κάποιο count
  - Θέλουμε count αιτήσεων ανά κολλέγιο, άρα group by με το όνομα του κολλεγίου
  - Συνθήκη στο count (θέλουμε το μεγαλύτερο), άρα συνθήκη στο HAVING

ALL & ANY

Πώς θα υπολογίζαμε τα ονόματα των κολλεγίων στα οποία έχουν γίνει οι περισσότερες αιτήσεις;

```
SELECT cName
FROM    Apply
GROUP BY cName
HAVING COUNT(*) >= ALL (SELECT COUNT(*)
                        FROM Apply
                        GROUP BY cName)
```

Σημείωση: Πολύ συχνά (όχι πάντα), το υποερώτημα μέσα στο ALL θα μοιάζει με το εξωτερικό ερώτημα.

Υποερωτήματα - SUBQUERIES

Υποερωτήματα μπορούν να υπάρχουν και στο FROM, καθώς και σπανιότερα στο SELECT

Το αποτέλεσμά τους στο FROM είναι μία σχέση, στην οποία μπορούμε να δώσουμε ένα νέο όνομα με μία μεταβλητή

Το αποτέλεσμά τους στο SELECT είναι μία τιμή (αποτέλεσμα υποερωτήματος), στην οποία μπορούμε να δώσουμε ένα νέο όνομα με μία μεταβλητή

Υποερωτήματα στο FROM

Βρες τα sID των μαθητών που έγιναν δεκτοί σε κολλέγιο με περισσότερους από 10000 μαθητές

```
SELECT DISTINCT sID
FROM    Apply A1,
        (SELECT cName FROM College WHERE enr > 10000) as largeCollege
WHERE A1.cName = largeCollege.cName AND dec = 'YES'
```

Μετά τον ορισμό του, το largeCollege το χρησιμοποιούμε στο ερώτημα σαν να ήταν πίνακα. Χρειάζεται **ΠΑΝΤΑ** ψευδώνυμο το υποερώτημα στο FROM.

College

cName	state	enr
Mary	MD	1000
MIT	MA	17000
Jim	CH	12000

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

Apply

sID	cName	major	dec
1	Mary	CS	YES
1	MIT	MPD	NO
2	MIT	EE	NO
2	Mary	CS	YES

Υποερωτήματα στο FROM

```
SELECT DISTINCT sID
FROM     Apply A1,
         (SELECT cName FROM College WHERE enr > 10000) as largeCollege
WHERE A1.cName = largeCollege.cName AND dec = 'YES'
```

largeCollege

cName
MIT
Jim

Αποτέλεσμα μετά το JOIN

sID	A1.cName	major	dec	largeCollege.cName
1	Jim	CS	YES	Jim
2	MIT	EE	YES	MIT

Αποτέλεσμα

sID
1
2

College

cName	state	enr
Mary	MD	1000
MIT	MA	17000
Jim	CH	12000

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

Apply

sID	cName	Major	dec
1	Jim	CS	YES
1	MIT	MPD	NO
2	MIT	EE	YES
2	Mary	CS	YES

Υποερωτήματα στο SELECT

Βρες για κάθε κολλέγιο τον αριθμό των αιτήσεων που έγιναν σε αυτό

Μία τιμή ΜΟΝΟ πρέπει να επιστρέφεται από το υποερώτημα στο SELECT

```
SELECT DISTINCT cName,
        (SELECT COUNT(*)
         FROM APPLY A2
         WHERE A2.cName = A1.cName) as appCount
FROM    Apply A1
```

cName	appCount
Mary	2
MIT	2

Εκτελείται για κάθε tuple που έρχεται στο SELECT

College

cName	state	enr
Mary	MD	1000
MIT	MA	17000
Jim	CH	12000

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

Apply

sID	cName	major	dec
1	Mary	CS	YES
1	MIT	MPD	NO
2	MIT	EE	NO
2	Mary	CS	YES

Υποερωτήματα στο SELECT

Πώς θα άλλαζε το ερώτημα αν θέλατε και τα κολλέγια με 0 αιτήσεις;

```
SELECT DISTINCT cName,
        (SELECT COUNT(*)
         FROM APPLY A2
         WHERE A2.cName = A1.cName) as appCount
FROM    Apply A1    College A1
```

Θυμηθείτε ότι το ερώτημα γίνεται και με left outer join

College

cName	state	enr
Mary	MD	1000
MIT	MA	17000
Jim	CH	12000

Student

sID	sName	GPA	hs
1	Jim	3.5	1200
2	Peter	3.7	800
3	Mary	3.5	1200
4	Jim	3.8	1400

Apply

sID	cName	major	dec
1	Mary	CS	YES
1	MIT	MPD	NO
2	MIT	EE	NO
2	Mary	CS	YES

Common Table Expression: CTE

Σε ένα ερώτημα μπορούμε να δηλώσουμε έναν προσωρινό πίνακα, τον οποίο να χρησιμοποιήσουμε αργότερα σαν να ήταν πίνακας της βάσης μας

Ο πίνακας αυτός ΔΕΝ υπάρχει μετά την εκτέλεση του ερωτήματος

Λέξη κλειδί: WITH

CTE

```
WITH cte_name(col_name1, …, col_name_n) AS (
        SQL QUERY που επιστρέφει τον προσωρινό πίνακα
)
SELECT *
FROM cte_name, other_tables
WHERE συνθήκες και πιθανά joins κτλ ...

WITH MaxHS (maxhs) AS (
   SELECT MAX(hs)
   FROM Student
)
SELECT sID, sName
FROM Student, MaxHS
WHERE hs = MaxHS.maxhs
```

CTE – Πολλαπλοί ορισμοί

```
WITH MaxHS (maxhs) AS (
    SELECT MAX(hs)
    FROM Student
),
StudentsWithMaxHS AS (
    SELECT sID, sName
    FROM Student, MaxHS
    WHERE hs = MaxHS.maxhs
)
SELECT *
FROM StudentsWithMaxHS
```

Αν δε δηλώσουμε ρητά τα ονόματα των attributes, παίρνει τα ονόματα των attributes του ερωτήματος που ορίζει το CTE

Recursion
(Αναδρομικά Ερωτήματα)

Recursion - Αναδρομή

Δυνατότητα δήλωσης αναδρομικού υπολογισμού σχέσης

Σκεφτείτε: ParentOf(parent, child)

Βρείτε όλους τους προγόνους/απογόνους ενός ατόμου X

Αν θέλαμε μόνο τους γονείς του X…

select parent from ParentOf where child='X'

ParentOf

parent	child
George	Antonis
Mary	Antonis
Antonis	Jim
Peter	Mary

Recursion - Αναδρομή

Σκεφτείτε: ParentOf(parent, child)

Αν θέλαμε μόνο τους γονείς του X ή παππούδες/γιαγιάδες του X...

Επέστρεψε τους γονείς του 'X'

union

Επέστρεψε όσους είναι παππούς/γιαγιά του 'X'

(αυτοί που επιστρέφω έχουν ως παιδί κάποιον που είναι γονέας του 'X')

ParentOf

parent	child
George	Antonis
Mary	Antonis
Antonis	Jim
Peter	Mary

Recursion - Αναδρομή

Σκεφτείτε: ParentOf(parent, child)

Αν θέλαμε μόνο τους γονείς του X ή παππούδες/γιαγιάδες του X...

```
select parent from ParentOf where child='X'
union
select a.parent
from ParentOf a, ParentOf b
where b.child = 'X' and a.child = b.parent
```

Αν βάλουμε Jim αντί για X

ParentOf a

a.parent = παππούς/ γιαγιά

parent	child
George	Antonis
Mary	Antonis
Antonis	Jim
Peter	Mary

ParentOf b

b.parent = πατέρας του Jim

parent	child
George	Antonis
Mary	Antonis
Antonis	Jim
Peter	Mary

Recursion - Αναδρομή

Αν θέλαμε συγγενείς μέχρι συγκεκριμένη απόσταση, αρκεί να γίνει join κατάλληλο αριθμό φορών η σχέση ParentOf με τον εαυτό της

Απροσδιόριστη η απόσταση των συγγενών => ανάγκη αναδρομής

Απλή μορφή:

Εκτελείται 1 φορά για αρχικοποίηση της R

With Recursive

R as (μη αναδρομικό ερώτημα

union

αναδρομικό ερώτημα **που περιλαμβάνει την R**)

Εδώ συμπληρώνεται SQL ερώτημα **που περιλαμβάνει την R** και πιθανώς άλλους πίνακες

ΠΡΟΣΟΧΗ: **Η αναδρομή τερματίζεται όταν νέες επαναλήψεις δεν προσθέτουν εγγραφές στην R**

Συνήθως δε θα δείτε union all γιατί συνεχώς προσθέτει tuples στην R

Recursion - Αναδρομή

```
With Recursive
        Anc(a,d)  as (
                select parent as a, child as d from ParentOf
                union
                select Anc.a, ParentOf.child as d
                from Anc, ParentOf
                where Anc.d = ParentOf.parent )
select a
from Anc
where d='Antonis'
```

Αντί για ParentOf a βάλαμε Anc και αντί για a.child βάλαμε Anc.d

Αν το Anc περιέχει σε κάποιο στάδιο της αναδρομής προγόνους σε απόσταση έως 2, τώρα υπολογίζει προγόνους σε απόσταση έως 3

Ερώτημα που χρησιμοποιεί την Anc σαν να είναι πίνακας στη βάση μας

Βρίσκει όλους τους πρόγονους του Antonis

Πως θα βρίσκατε τους απογόνους του Antonis;

Recursion – Παράδειγμα Εκτέλεσης

With Recursive

Anc(a,d) as (

select parent as a, child as d from ParentOf

union

select Anc.a, ParentOf.child as d

from Anc, ParentOf

where Anc.d = ParentOf.parent)

ΥΠΟΛΟΙΠΟ ΕΡΩΤΗΜΑ

Anc – Αρχικοποίηση;
Ίδιο με το ParentOf

a	d
George	Antonis
Mary	Antonis
Antonis	Jim
Peter	Mary

ParentOf

parent	child
George	Antonis
Mary	Antonis
Antonis	Jim
Peter	Mary

Anc – Μετά από 1 union

a	d
George	Antonis
Mary	Antonis
Antonis	Jim
Peter	Mary
George	Jim
Mary	Jim
Peter	Antonis

Recursion – Παράδειγμα Εκτέλεσης

With Recursive

```
Anc(a,d) as (
        select parent as a, child as d from ParentOf
        union
        select Anc.a, ParentOf.child as d
        from Anc, ParentOf
        where Anc.d = ParentOf.parent )
```

ΥΠΟΛΟΙΠΟ ΕΡΩΤΗΜΑ

Anc – Μετά από 1 union

a	d
George	Antonis
Mary	Antonis
Antonis	Jim
Peter	Mary
George	Jim
Mary	Jim
Peter	Antonis

ParentOf

parent	child
George	Antonis
Mary	Antonis
Antonis	Jim
Peter	Mary

Anc – Μετά από 2 union

a	d
George	Antonis
Mary	Antonis
Antonis	Jim
Peter	Mary
George	Jim
Mary	Jim
Peter	Antonis
Peter	Jim

Σειρά Anc, ParentOf και Φίλτρο μέσα στο Ερώτημα

```
With Recursive
        Anc(a,d)  as (
                select parent as a, child as d from ParentOf
                where child='Antonis'
                union
                select ParentOf.parent as a, Anc.d as d
                from Anc, ParentOf
                where Anc.a = ParentOf.child )
select a
from Anc
```

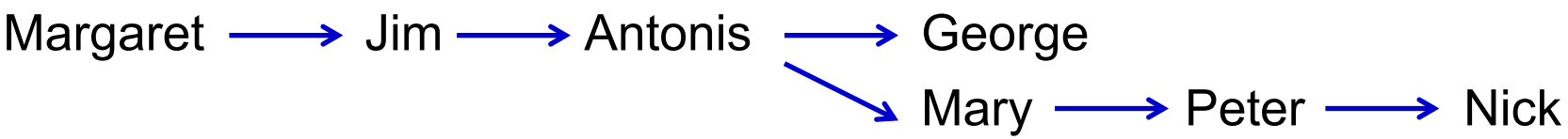
Το παραπάνω ερώτημα έχει περάσει το φίλτρο μέσα στο αρχικό ερώτημα. Βρίσκει μόνο τους προγόνους ενός ατόμου και όχι όλων => πολύ πιο γρήγορο

Και ένα Παράδειγμα …

Θυμηθείτε… ParentOf(parent, child)

```
insert into ParentOf values('George', 'Antonis');
insert into ParentOf values('Mary', 'Antonis');
insert into ParentOf values('Antonis', 'Jim');
insert into ParentOf values('Jim', 'Margaret');
insert into ParentOf values('Peter', 'Mary');
insert into ParentOf values('Nick', 'Peter');
```

parent	child
George	Antonis
Mary	Antonis
Antonis	Jim
Jim	Margaret
Peter	Mary
Nick	Peter



Πρόγονοι του Antonis: George, Mary, Peter, Nick

Recursion – Παράδειγμα Εκτέλεσης

```
With Recursive
        Anc(a,d)  as (
                select parent as a, child as d from ParentOf
                where child='Antonis'
                union
                select ParentOf.parent as a, Anc.d as d
                from Anc, ParentOf
                where Anc.a = ParentOf.child )
select a
from Anc
```

ParentOf

parent	child
George	Antonis
Mary	Antonis
Antonis	Jim
Jim	Margaret
Peter	Mary
Nick	Peter

Anc – Αρχικοποίηση

a	d
George	Antonis
Mary	Antonis

Anc – Μετά από 1 union

a	d
George	Antonis
Mary	Antonis
Peter	Antonis

Anc – Μετά από 2 union

a	d
George	Antonis
Mary	Antonis
Peter	Antonis
Nick	Antonis

Το Παρακάτω είναι Σωστό;

```
With Recursive
   Anc(a,d)  as (
         select parent as a, child as d
         from ParentOf
         where child='Antonis'
         union
         select ParentOf.parent as a,
                  Anc.d as d
         from Anc, ParentOf
         where Anc.a = ParentOf.child )
select a
from Anc
```

ΠΑΛΙΟ ΕΡΩΤΗΜΑ: ΣΩΣΤΟ

```
With Recursive
   Anc(a,d)  as (
         select parent as a, child as d
         from ParentOf
         where child='Antonis'
         union
         select Anc.a,
                  ParentOf.child as d
         from Anc, ParentOf
         where Anc.d = ParentOf.parent)
select a
from Anc
```

ΥΠΟΨΗΦΙΟ ΕΡΩΤΗΜΑ: ΣΩΣΤΟ;;;;;;

Αν το προσέξετε, βρίσκει τους απογόνους των γονέων του Antonis… θέλει προσοχή

Χωρίς το Antonis το ερώτημα είναι σωστό

Γιατί είναι Λάθος …

ParentOf

parent	child
George	Antonis
Mary	Antonis
Antonis	Jim
Jim	Margaret
Peter	Mary
Nick	Peter

```
With Recursive
   Anc(a,d)  as (
         select parent as a, child as d
         from ParentOf     where child='Antonis'
         union
         select Anc.a, ParentOf.child as d
         from Anc, ParentOf
         where Anc.d = ParentOf.parent)
select a
from Anc
```

Anc – Αρχικοποίηση

a	d
George	Antonis
Mary	Antonis

Anc – 1η επανάληψη

a	d
George	Antonis
Mary	Antonis
Mary	Jim
George	Jim

Anc – 2η επανάληψη

a	d
George	Antonis
Mary	Antonis
Mary	Jim
George	Jim
George	Margaret
Mary	Margaret

Recursive Views

Θα δούμε ότι οι Όψεις (Views) μας παρέχουν πολλές δυνατότητες και ευκολίες

Προς το παρόν θα δούμε ότι μπορούμε να ορίσουμε μία όψη που θα παρέχει τον ορισμό ενός αναδρομικού ερωτήματος

Μετά μπορούμε στο FROM ερωτημάτων να χρησιμοποιούμε την όψη σαν τους υπόλοιπους πίνακες

```
CREATE RECURSIVE VIEW Anc(a,d)  as (
                select parent as a, child as d from ParentOf
                where child='Antonis'
                union
                select ParentOf.parent as a, Anc.d as d
                from Anc, ParentOf
                where Anc.a = ParentOf.child );
select a from Anc where d = 'Antonis';
```

Εδώ τελειώνει η δήλωση του view

Triggers
(Σκανδάλη)

Triggers – Τι Είναι

Trigger: κομμάτι κώδικα που εκτελείται πριν ή μετά από κάποια εντολή που μεταβάλει δεδομένα (insert, delete, update, truncate)

Χρήσιμο για μεταφορά λογικής των εφαρμογών εντός ενός ΣΔΒΔ

Συμβάλει στην επιβολή περιορισμών

Μπορούμε να δηλώσουμε κάποιο trigger για 1 ή περισσότερες λειτουργίες σε ένα πίνακα. Πχ:

- Ένα trigger που εκτελείται κατά (πριν/μετά) την εισαγωγή και διαγραφή δεδομένων στον πίνακα Χ
- Ένα trigger που εκτελείται κατά (πριν/μετά) την ενημέρωσης εγγραφών (ή συγκεκριμένων μόνο attributes) δεδομένων στον πίνακα Χ

Triggers – Πού χρησιμεύουν;

Ο κώδικας ενός trigger εκτελείται αυτόματα και δε μπορούμε να τον καλέσουμε εμείς απευθείας

Χρήση Triggers:

1. Επιβολή περιορισμών
2. Καταγραφή μηνυμάτων σε πίνακες (πχ, πότε και από ποιους έγινε διαγραφή tuples από σχέσεις)
3. Επιτρέπουν την πρόσβαση και μεταβολή δεδομένων σε άλλους πίνακες

Triggers - Σύνταξη

Event:
insert
delete
update [of columnName[, …]]
truncate

CREATE TRIGGER name
{BEFORE | AFTER | INSTEAD OF } { event [OR ...] }
ON table_name
[FOR [EACH] { ROW | STATEMENT }]
[WHEN (condition)]
EXECUTE PROCEDURE function_name (arguments)

Πόσες φορές εκτελείται ο trigger; Για κάθε πλειάδα που επηρεάζεται ή 1 φορά για όλες τις πλειάδες;

Πότε εκτελείται ο trigger;

Πριν ή μετά το γεγονός που ορίζεται (σειρά 2).

Το INSTEAD OF έχει νόημα σε όψεις μόνο.

Παραδείγματα

Trigger που εκτελείται πριν την ενημέρωση οποιουδήποτε attribute του πίνακα accounts

```
CREATE TRIGGER check_update
BEFORE UPDATE ON accounts
FOR EACH ROW
EXECUTE PROCEDURE check_account_update();
```

Trigger που εκτελείται μετά την ενημέρωση του attribute balance του πίνακα accounts

```
CREATE TRIGGER check_update
AFTER UPDATE OF balance ON accounts
FOR EACH ROW
EXECUTE PROCEDURE check_account_update();
```

Παραδείγματα

Μπορείτε να ορίσετε έναν Trigger για πολλαπλές μεταβολές

```
CREATE TRIGGER check_update
BEFORE INSERT OR DELETE OR UPDATE ON accounts
FOR EACH ROW
EXECUTE PROCEDURE check_account_update();
```

Μέσα στη συνάρτηση που καλεί ο Trigger υπάρχει μια μεταβλητή που μας λέει ποια ήταν η μεταβολή που οδήγησε στην ενεργοποίηση του Trigger

OLD και NEW

Αν χρησιμοποιούμε FOR EACH ROW, αναφορά σε παλιές και νέες τιμές εγγραφών είναι δυνατή με OLD.columnName και NEW.columnName

Σκεφτείτε ότι το OLD μας δίνει το tuple ΠΡΙΝ την μεταβολή

Σκεφτείτε ότι το NEW μας δίνει το tuple ΜΕΤΑ την προτεινόμενη μεταβολή

OLD και NEW

Σε εισαγωγές, OLD = NULL – αναφερόμαστε μόνο σε NEW

Σε διαγραφές, NEW = NULL – αναφερόμαστε μόνο σε OLD

Σε ενημερώσεις έχουν νόημα και το NEW και το OLD

Τα NEW, OLD είναι προσβάσιμα ΚΑΙ στη συνάρτηση του trigger

INSERT	DELETE	UPDATE
NEW: Το νέο tuple που κάνουμε insert	NEW: NULL Δεν θα υπάρχει το tuple στον πίνακα μετά	NEW: Το tuple που κάνουμε update ΜΕΤΑ την ενημέρωση
OLD: NULL Δεν υπήρχε το tuple στον πίνακα πιο πριν	OLD: Το tuple που κάνουμε delete	OLD: Το tuple που κάνουμε update ΠΡΙΝ την ενημέρωση

OLD και NEW

Σε εισαγωγές, OLD = NULL – αναφερόμαστε μόνο σε NEW

Σε διαγραφές, NEW = NULL – αναφερόμαστε μόνο σε OLD

Σε ενημερώσεις έχουν νόημα και το NEW και το OLD

Τα NEW, OLD είναι προσβάσιμα ΚΑΙ στη συνάρτηση του trigger

```
CREATE TRIGGER check_before_insert
BEFORE UPDATE OF balance ON accounts
FOR EACH ROW
WHEN (NEW.balance  < OLD.balance)
EXECUTE PROCEDURE check_account_update();
```

Παραδείγματα

Θα υλοποιήσουμε τα ακόλουθα triggers:

1) Σε κάθε ενημέρωση του GPA στη σχέση Student, μαθητές που αποκτούν GPA > 3.5 (ενώ πριν είχαν GPA <= 3.5) θα γίνονται αυτόματα δεκτοί στο TUC (attribute dec στη σχέση Apply)
2) Σε περίπτωση διαγραφής ενός κολλεγίου από τη σχέση College, θα διαγράφουμε όλες τις αιτήσεις προς αυτό το κολλέγιο

College

cName	state	enr

Student

sID	sName	GPA	hs
1	Jim	3.5	1100
2	Peter	3.7	800
3	Mary	3.6	1200
4	Jim	3.8	1400
5	Mac	3.4	900

Apply

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
3	MIT	EE	NO
4	TUC	CS	YES
5	EMP	CS	NO

Τιμές Επιστροφής σε Row Level Triggers

BEFORE TRIGGERS:

1. Αν θέλουμε να ΜΗ γίνει η εισαγωγή, διαγραφή, ενημέρωση της εγγραφής, επιστρέφουμε NULL
2. Αν θέλουμε να προχωρήσει η διαγραφή, εισαγωγή, ενημέρωση τότε:
   1. Για DELETE, επιστρέφουμε OLD ή κάτι που δεν είναι NULL
   2. Για INSERT/UPDATE επιστρέφουμε NEW.

      Σημείωση: Αν θέλουμε, στα πεδία του NEW μπορούμε να βάλουμε δικές μας τιμές μέσα στη συνάρτηση του trigger

AFTER TRIGGERS: Η τιμή επιστροφής αγνοείται

Παραδείγματα – Update στη Student

Σε κάθε ενημέρωση του GPA στη σχέση Student, μαθητές που αποκτούν GPA > 3.5 (ενώ πριν είχαν GPA <= 3.5) θα γίνονται αυτόματα δεκτοί στο TUC (attribute dec στη σχέση Apply).

```
CREATE TRIGGER check_Student_update
BEFORE UPDATE OF "GPA" ON "Student"
FOR EACH ROW
WHEN (NEW."GPA" > 3.5 and OLD."GPA" <= 3.5)
EXECUTE PROCEDURE update_apply_tuc();

CREATE OR REPLACE FUNCTION update_apply_tuc()
RETURNS TRIGGER AS $$
BEGIN
        UPDATE "Apply"  SET dec= 'YES'
        WHERE "Apply"."sID" = OLD."sID" and "Apply"."cName"= 'TUC';
        RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Πάντα δηλώνεται χωρίς ορίσματα και επιστρέφει TRIGGER

BEFORE TRIGGER
Επιστρέφουμε NEW για update

Τι Συμβαίνει σε Update;

Αρχίζουμε το παράδειγμα με τους πίνακες στο κάτω μέρος

update "Student" set "GPA"= "GPA" + 0.1;

select * from "Apply";

Περιμένουμε να έχουν επηρεαστεί οι αιτήσεις των sID=1 και 5.

Ο sID=4 που έχει κάνει αίτηση στο TUC είχε ήδη βαθμό > 3.5.

College

cName	state	enr

Student

sID	sName	GPA	hs
1	Jim	3.5	1100
2	Peter	3.7	800
3	Mary	3.6	1200
4	Jim	3.8	1400
5	Mac	3.45	900

Apply

sID	cName	major	dec
1	TUC	CS	NO
1	MIT	MPD	NO
3	MIT	EE	YES
4	TUC	CS	NO
5	TUC	CS	NO

Τι Συμβαίνει σε Update;

Αρχίζουμε το παράδειγμα με τους πίνακες στο κάτω μέρος

update "Student" set "GPA"= "GPA" + 0.1;

select * from "Apply";

Περιμένουμε να έχουν επηρεαστεί οι αιτήσεις των sID=1 και 5.

Ο sID=4 που έχει κάνει αίτηση στο TUC είχε ήδη βαθμό > 3.5.

Νέο Apply

sID	cName	major	dec
1	TUC	CS	YES
1	MIT	MPD	NO
3	MIT	EE	YES
4	TUC	CS	NO
5	TUC	CS	YES

College

cName	state	enr

Student

sID	sName	GPA	hs
1	Jim	3.6	1100
2	Peter	3.8	800
3	Mary	3.7	1200
4	Jim	3.9	1400
5	Mac	3.55	900

Παλιό Apply

sID	cName	major	dec
1	TUC	CS	NO
1	MIT	MPD	NO
3	MIT	EE	YES
4	TUC	CS	NO
5	TUC	CS	NO

Ορίσματα Εντός της Συνάρτησης Ενός TRIGGER

Τα παρακάτω ορίσματα είναι διαθέσιμα ΕΝΤΟΣ της συνάρτησης ενός trigger

ΜΕΤΑΒΛΗΤΗ	ΣΗΜΑΣΙΑ
TG_NAME	Όνομα του TRIGGER που κάλεσε τη συνάρτηση
TG_WHEN	BEFORE ή AFTER ή INSTEAD OF
TG_LEVEL	ROW ή STATEMENT
TG_OP	INSERT ή DELETE ή UPDATE ή TRUNCATE
TG_TABLE_NAME	Το όνομα του Πίνακα στο οποίο έγινε η μεταβολή
TG_NARGS	Αριθμός ορισμάτων στη συνάρτηση
TG_ARGV[]	Ορίσματα στη συνάρτηση (ξεκινώντας από τη θέση 0)

Παραδείγματα – Delete στην College

Σε περίπτωση διαγραφής ενός κολλεγίου από τη σχέση College, θα διαγράφουμε όλες τις αιτήσεις προς αυτό το κολλέγιο

```
CREATE TRIGGER check_College_delete
BEFORE DELETE ON "College"
FOR EACH ROW
EXECUTE PROCEDURE delete_apply();

CREATE OR REPLACE FUNCTION delete_apply()
RETURNS TRIGGER AS $$
BEGIN
        DELETE FROM "Apply" WHERE "Apply"."cName" = OLD."cName";
        RETURN OLD;
END;
$$ LANGUAGE plpgsql;
```

Αν έχουμε ξένο κλειδί, ΠΡΕΠΕΙ πρώτα να διαγράψουμε από την Apply και μετά από την College

BEFORE TRIGGER
Για DELETE επιστρέφουμε OLD, ή κάτι που δεν είναι NULL

Τι Συμβαίνει σε Delete;

Αρχίζουμε το παράδειγμα με τους πίνακες στο κάτω μέρος

delete from "College" where "cName"='MIT'

select * from "Apply";

Περιμένουμε να έχουν διαγραφεί οι αιτήσεις προς το MIT

College

cName	state	enr
~~MIT~~	~~MA~~	~~1500~~
TUC	CH	800
EMP	AT	1100

Student

sID	sName	GPA	hs
1	Jim	3.6	1100
2	Peter	3.8	800
3	Mary	3.7	1200
4	Jim	3.9	1400
5	Mac	3.55	900

Apply

sID	cName	major	dec
1	TUC	CS	YES
~~1~~	~~MIT~~	~~MPD~~	~~NO~~
~~3~~	~~MIT~~	~~EE~~	~~YES~~
4	TUC	CS	NO
5	TUC	CS	YES

Τι Συμβαίνει σε Delete;

Αρχίζουμε το παράδειγμα με τους πίνακες στο κάτω μέρος

delete from "College" where "cName"='MIT'

select * from "Apply";

Περιμένουμε να έχουν διαγραφεί οι αιτήσεις προς το MIT

Νέο Apply

sID	cName	major	dec
1	TUC	CS	YES
4	TUC	CS	NO
5	TUC	CS	YES

Νέο College

cName	state	enr
TUC	CH	800
EMP	AT	1100

Student

sID	sName	GPA	hs
1	Jim	3.6	1100
2	Peter	3.8	800
3	Mary	3.7	1200
4	Jim	3.9	1400
5	Mac	3.55	900

Παλιό Apply

sID	cName	major	dec
1	TUC	CS	YES
~~1~~	~~MIT~~	~~MPD~~	~~NO~~
~~3~~	~~MIT~~	~~EE~~	~~YES~~
4	TUC	CS	NO
5	TUC	CS	YES

Παραδείγματα – Insert στη Student

Σε κάθε εισαγωγή μαθητή με GPA > 3.75 θα εισάγουμε απευθείας αίτησή του στο CS του MIT και του TUC με μη καθορισμένη (NULL) απόφαση

```
CREATE TRIGGER check_Student_insert
AFTER INSERT ON "Student"
FOR EACH ROW
WHEN (NEW."GPA" > 3.75)
EXECUTE PROCEDURE insert_apply_dec();

CREATE OR REPLACE FUNCTION insert_apply_dec()
RETURNS TRIGGER AS $$
BEGIN
        INSERT INTO "Apply" VALUES(NEW."sID", 'MIT', 'CS', null);
        INSERT INTO "Apply" VALUES(NEW."sID", 'TUC', 'CS', null);
        RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Αν έχουμε ξένο κλειδί, ΠΡΕΠΕΙ πρώτα να γίνει η εισαγωγή στη Student και μετά στην Apply

AFTER TRIGGER
Δεν έχει σημασία η τιμή επιστροφής. Για να μη μπερδεύεστε, μπορείτε να επιστρέφετε ότι και σε BEFORE trigger

Τι Συμβαίνει σε Insert;

Αρχίζουμε το παράδειγμα με τους πίνακες στο κάτω μέρος

```
insert into "Student" values(6, 'Ge', 3.8, 900);
insert into "Student" values(7, 'Hi', 3.7, 1900);
select * from "Apply";
```

Περιμένουμε να έχουν προστεθεί αιτήσεις για το μαθητή με sID=6

Έστω ότι υπάρχει το MIT

College

cName	state	enr
MIT	MA	1500
TUC	CH	800
EMP	AT	1100

Student

sID	sName	GPA	hs
1	Jim	3.6	1100
2	Peter	3.8	800
3	Mary	3.7	1200
4	Jim	3.9	1400
5	Mac	3.55	900

Apply

sID	cName	major	dec
1	TUC	CS	YES
4	TUC	CS	NO
5	TUC	CS	YES

Τι Συμβαίνει σε Insert;

Αρχίζουμε το παράδειγμα με τους πίνακες στο κάτω μέρος

```
insert into "Student" values(6, 'Ge', 3.8, 900);
insert into "Student" values(7, 'Hi', 3.7, 1900);
select * from "Apply";
```

Περιμένουμε να έχουν προστεθεί αιτήσεις για το μαθητή με sID=6

Νέο Apply

sID	cName	major	dec
1	TUC	CS	YES
4	TUC	CS	NO
5	TUC	CS	YES
6	MIT	CS	
6	TUC	CS	

Νέο Student

sID	sName	GPA	hs
1	Jim	3.6	1100
2	Peter	3.8	800
3	Mary	3.7	1200
4	Jim	3.9	1400
5	Mac	3.55	900
6	Ge	3.8	900
7	Hi	3.7	1900

College

cName	state	enr
MIT	MA	1500
TUC	CH	800
EMP	AT	1100

Παλιό Apply

sID	cName	major	dec
1	TUC	CS	YES
4	TUC	CS	NO
5	TUC	CS	YES

Παραδείγματα – Απόρριψη Delete

Σε αυτή την περίπτωση πρέπει η συνάρτηση μας να επιστρέψει NULL αν εκτελεστεί πριν το DELETE. Αν εκτελεστεί μετά το DELETE, πρέπει μέσα στη συνάρτηση να διαγράψουμε τη νέα εγγραφή. Η πρώτη επιλογή είναι πιο ασφαλής

```
CREATE TRIGGER check_Student_delete
BEFORE DELETE ON "Student"
FOR EACH ROW EXECUTE PROCEDURE check_delete_student();

CREATE OR REPLACE FUNCTION check_delete_student()
RETURNS TRIGGER AS $$
BEGIN
        IF OLD."sID" > 5 THEN
                RAISE NOTICE 'Will not delete sID greater than 5';
                RETURN NULL;
        END IF;
        RETURN OLD;
 END;
$$ LANGUAGE plpgsql;
```

Επιστροφή NULL σε BEFORE DELETE σημαίνει απόρριψη της διαγραφής

Τι Συμβαίνει σε Delete;

Αρχίζουμε το παράδειγμα με τους πίνακες στο κάτω μέρος

delete from "Student" where "sID" = 5;

delete from "Student" where "sID" = 6;

select * from "Student";

Περιμένουμε να μην έχει διαγραφεί ο μαθητής με sID=6

College

cName	state	enr
MIT	MA	1500
TUC	CH	800
EMP	AT	1100

Student

sID	sName	GPA	hs
1	Jim	3.6	1100
2	Peter	3.8	800
3	Mary	3.7	1200
4	Jim	3.9	1400
5	Mac	3.55	900
6	Ge	3.8	900
7	Hi	3.7	1900

Τι Συμβαίνει σε Delete;

Αρχίζουμε το παράδειγμα με τους πίνακες στο κάτω μέρος

```
delete from "Student" where "sID" = 5;
delete from "Student" where "sID" = 6;
select * from "Student";
```

Περιμένουμε να μην έχει διαγραφεί ο μαθητής με sID=6

College

cName	state	enr
MIT	MA	1500
TUC	CH	800
EMP	AT	1100

Νέο Student

sID	sName	GPA	hs
1	Jim	3.6	1100
2	Peter	3.8	800
3	Mary	3.7	1200
4	Jim	3.9	1400
6	Ge	3.8	900
7	Hi	3.7	1900