

**ΠΕΡΙΕΧΟΜΕΝΑ:**

1. OUTER JOINS
 1. LEFT OUTER JOIN
 2. RIGHT OUTER JOIN
 3. FULL OUTER JOIN
2. SELF JOIN

Σπύρος Π.

Σμαραγδένιος Χορηγός Μαθήματος

Θανάσης Σ.

Χρυσός Χορηγός Μαθήματος

LEFT OUTER JOIN (αριστερή εξωτερική σύνδεση/συνένωση)

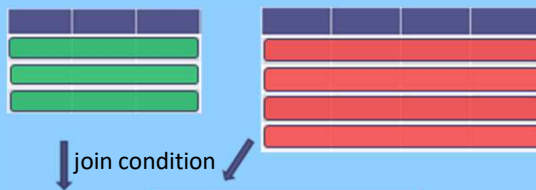
- ΣΥΝΤΑΚΤΙΚΟ:

```
FROM table1 t1 LEFT [OUTER] JOIN table2 t2
ON join_condition
```

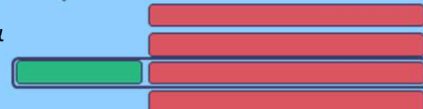
- Σε αυτόν τον τύπο σύνδεσης:
 - Έχουμε το ίδιο αποτέλεσμα όπως στο INNER JOIN
 - Αλλά πρόσθετα επιστρέφονται και εκείνες οι γραμμές του αριστερού πίνακα που δεν ικανοποιήθηκε η συνθήκη join με καμία γραμμή του δεξιού πίνακα
 - Αυτές οι γραμμές, συμπληρώνονται με NULL σε όλα τα πεδία του δεξιού πίνακα.

- Σχηματικά (π.χ.):

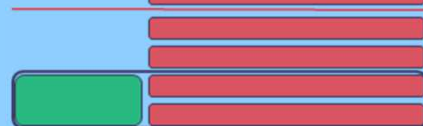
Κάνουμε LEFT JOIN
(προσοχή ότι η διάταξη
των πινάκων έχει σημασία)



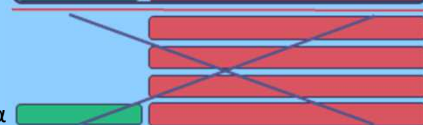
Η 1η γραμμή του 1ου πίνακα
“ταιριάζει” με
μία γραμμή του 2ου πίνακα



Η 2η γραμμή του 1ου πίνακα
“ταιριάζει” με
δύο γραμμές του 2ου πίνακα



Η 3η γραμμή του 1ου πίνακα
δεν “ταιριάζει” με
καμία γραμμή του 2ου πίνακα



Η 3η γραμμή επιστρέφεται,
συμπληρωμένη, σε όλες τις
στήλες του 2ου πίνακα με
NULL (γκρι χρώμα)

Παράδειγμα 1: Οι χώρες και οι πόλεις (DB: world, left.outer.sql)

Παρατηρήστε το ερώτημα

-- example 1.1

```
SELECT cn.name AS country, count(ct.name) AS cities
FROM country cn JOIN city ct
ON cn.Code = ct.CountryCode
```

GROUP BY cn.Name

ORDER BY 2, 1;

το οποίο επιστρέφει 232 χώρες.

Ωστόσο το ερώτημα:

-- example 1.2

```
SELECT COUNT(*) AS countries
```

```
FROM country;
```

επιστρέφει 239 χώρες.

Πράγματι υπάρχουν 7 χώρες για τις οποίες δεν έχει καταχωρηθεί καμία πόλη στη βάση μας. Συνεπώς μπορούμε να εμφανίσουμε και αυτές τις πόλεις, διορθώνοντας το INNER JOIN σε LEFT JOIN:

-- example 1.3

```
SELECT cn.name AS country, count(ct.name) AS cities
```

```
FROM country cn LEFT JOIN city ct
```

```
ON cn.Code = ct.CountryCode
```

GROUP BY cn.Name

ORDER BY 2, 1;

Παρατηρήσεις για τη LEFT OUTER JOIN σε ερωτήματα FK-PK:

- Στο προηγούμενο παράδειγμα είδαμε LEFT JOIN σε σχέση PK-FK όπου το PK ήταν αριστερά του JOIN.
 - και παρατηρήσαμε ότι “έξτρα” γραμμές που φέρνει η LEFT JOIN, είναι αυτές που το PK δεν έχει καμία εγγραφή FK στον αριστερό πίνακα.
- Αν το FK είναι αριστερά του LEFT JOIN, τότε η μορφή του τελικού πίνακα θα είναι:

FK				PK			

- Κάθε γραμμή που έχει FK: NULL, συμπληρώνεται με NULL (γκρι χρώμα) σε όλα τα πεδία

Παράδειγμα 2: -- example 2.1 (script: left.outer2.sql, DB: classic)

```
SELECT c.customerName AS customer,
       e.lastName AS assigned_to
FROM customers c LEFT JOIN employees e
  ON c.salesRepEmployeeNumber = e.employeeNumber
ORDER BY 2;
```

RIGHT OUTER JOIN (δεξιά εξωτερική σύνδεση/συνένωση)

- Συντακτικό:

```
FROM table1 t1 RIGHT [OUTER] JOIN table2 t2
  ON join_condition
```

- Σε αυτόν τον τύπο σύνδεσης:
 - Έχουμε το ίδιο αποτέλεσμα όπως στο INNER JOIN
 - Αλλά πρόσθετα επιστρέφονται και εκείνες οι γραμμές του δεξιού πίνακα που δεν ικανοποιήθηκε η συνθήκη join με καμία γραμμή του αριστερού πίνακα
 - Αυτές οι γραμμές, συμπληρώνονται με NULL σε όλα τα πεδία του αριστερού πίνακα
- (ή πιο λαϊκά: τα ίδια με το LEFT OUTER JOIN, αλλά από την άλλη μεριά, με γνώμονα το δεξιό πίνακα)

Σημειώσεις:

- Ισχύουν οι συμμετρικές παρατηρήσεις για τη μορφή των πινάκων με αυτές που κάναμε για το LEFT OUTER JOIN

- Στην πράξη, αφού ότι μπορούμε να κάνουμε με το RIGHT JOIN μπορούμε να το πετύχουμε με LEFT JOIN (αντιστρέφοντας τη σειρά των πινάκων), γενικά θα αποφεύγουμε το RIGHT JOIN και θα σκεφτόμαστε μόνο στα πλαίσια του LEFT JOIN

Άσκηση 1: DB: Sakila

Ο πίνακας inventory αποθηκεύει την πληροφορία για τις ταινίες που έχουμε συνολικά στα καταστήματα μάς. ως στήλες έχει:

- inventory_id: PK, απλός αύξων αριθμός
- film_id: FK στον πίνακα film (κωδικός ταινίας)
- store_id: FK στον πίνακα store (κωδικός καταστήματος)

Γράψτε ένα ερώτημα που να επιστρέφει μία λίστα των ταινιών που δεν έχουμε απόθεμα σε κανένα κατάστημά μας. Οι πρώτες γραμμές του πίνακα -αποτελέσματος πρέπει να είναι:

MOVIE
ALICE FANTASIA
APOLLO TEEN
ARGONAUTS TOWN
ARK RIDGEMONT
...

(συνολικά είναι: 42 ταινίες)

Άσκηση 2: Πίνακες inventory, rental

Παρατηρήστε ότι ο πίνακας inventory έχει πολλαπλές εμφανίσεις της ίδιας ταινίας στο ίδιο κατάστημα (λογικό, αν σκεφτούμε ότι ένα κλαμπ έχει αντίτυπα της ταινίας προς ενοικίαση). Επίσης ο πίνακας rental περιέχει τις ενοικιάσεις που έχουν γίνει για κάθε αντίτυπο ταινίας.

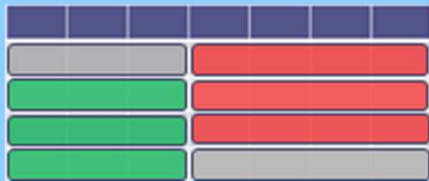
Κατασκευάστε ερώτημα το οποίο θα επιστρέφει για κάθε ταινία, το πλήθος των ενοικιάσεων της σε φθίνουσα σειρά:

MOVIE	TOTAL_RENTALS
BUCKET BROTHERHOOD	34
ROCKETEER MOTHER	33
FORWARD TEMPLE	32
GRIT CLOCKWORK	32
...	

(Συνολικά είναι: 1000 γραμμές, όσες και οι ταινίες του πίνακα)

FULL OUTER JOIN (ολική εξωτερική σύνδεση/συνένωση)

- Σε αυτόν τον τύπο σύνδεσης:
 - Έχουμε το ίδιο αποτέλεσμα όπως στο **INNER JOIN**
 - Αλλά πρόσθετα επιστρέφονται και εκείνες οι γραμμές του δεξιού πίνακα που δεν ικανοποιήθηκε η συνθήκη join με καμία γραμμή του αριστερού πίνακα
 - και επίσης εκείνες οι γραμμές του αριστερού πίνακα που δεν ικανοποιήθηκε η συνθήκη join με καμία γραμμή του αριστερού πίνακα



- Η MySQL δεν υποστηρίζει το FULL OUTER JOIN
 - αλλά μπορούμε να το προσομοιώσουμε με χρήση της UNION (ένωση), την οποία θα μελετήσουμε αναλυτικά σε επόμενο μάθημα.



Παράδειγμα 3:

-- example 3.1 (script: full.outer.sql, DB: classicmodels)

```
SELECT c.customerName AS customer,
       e.lastName AS assigned_to
FROM customers c LEFT JOIN employees e
  ON c.salesRepEmployeeNumber = e.employeeNumber
UNION
SELECT c.customerName AS customer,
       e.lastName AS assigned_to
FROM customers c RIGHT JOIN employees e
  ON c.salesRepEmployeeNumber = e.employeeNumber
```

Άσκηση 3: DB: Sakila

Κατασκευάστε ένα FULL OUTER JOIN πάνω στη σχέση FK-PK των πινάκων payment και rental.

Λίγα λόγια για την UNION:

- Η UNION (συνολοθεωρητική ένωση) παίρνει δύο πίνακες που περιέχουν τα ίδια πεδία:
 - και τις ενώνει σε έναν νέο πίνακα,
 - ο οποίος περιέχει κάθε εγγραφή των πινάκων ακριβώς μία φορά.
 - (αν κάποια εγγραφή υπήρχε και στους δύο πίνακες, τότε θα βρίσκεται μία φορά στο νέο πίνακα)


ΜΑΘΗΜΑ 1.6: OUTER JOINS

Το JOIN ενός πίνακα με τον εαυτό του, λέγεται SELF JOIN

- Σύνταξη (ίδια με το INNER JOIN , ορίζεται και για OUTER JOIN):

```
FROM table t1 [INNER] JOIN table t2  
ON join condition
```
- (και τελείως αντίστοιχα, μπορούν να οριστούν SELF OUTER JOIN (LEFT ή RIGHT ή OUTER)

Σημείωση:



employees
employeeNumber INT
lastName VARCHAR(50)
firstName VARCHAR(50)
extension VARCHAR(10)
email VARCHAR(100)
officeCode VARCHAR(10)
reportsTo INT
jobTitle VARCHAR(50)

2. SELF JOIN

Παράδειγμα 4:

-- example 4.1 (script: full.outer.sql, DB: classicmodels)

```
SELECT CONCAT(e1.firstName, ' ', e1.lastName) AS employee,  
       CONCAT(e2.firstName, ' ', e2.lastName) AS reports_to  
FROM employees e1 JOIN employees e2  
ON e1.reportsTo = e2.employeeNumber
```

Άσκηση 4:

Κατασκευάστε ερώτημα που μετράει πόσους υφιστάμενους έχει κάθε προϊστάμενος (δηλαδή υπάλληλοι που έχουν τουλάχιστον έναν υφιστάμενο)

Άσκηση 5:

Εντοπίστε σε ποιες πόλεις δουλεύουν οι υπάλληλοι (πίνακας offices) που δεν έχουν υφισταμένους.
Επιστρέψτε μόνο τις διακριτές πόλεις (χωρίς τα ονόματα των υπαλλήλων).

Άσκηση 6: DB: Sakila - Πίνακες customer, payment

A/ Βρείτε τους 10 καλύτερους πελάτες μας [αυτούς που μας έχουν δώσει τα περισσότερα χρήματα]. Επιστρέψτε για καθέναν από αυτούς, το πλήρες ονοματεπώνυμό του, αν είναι ακόμη ενεργός (στήλη active), το email του και το συνολικό ποσό που έχει πληρώσει.

B/ Βρείτε πόσα χρήματα μας έχουν δώσει συνολικά οι 100 χειρότεροι πελάτες μας [Προσοχή, να συνεκτιμηθούν οι πελάτες που δεν έχουν κάνει καμία πληρωμή]

Άσκηση 7: DB Sakila: country, city, address, payment, customer

Ταξινομήστε τις χώρες σε φθίνουσα σειρά συνολικού ποσού εισπράξεων που έγιναν από πελάτες που έχουν διεύθυνση στην αντίστοιχη χώρα.

Στο τελικό πίνακα, να εμφανιστούν και οι 109 πόλεις του πίνακα country, ακολουθούμενες από το αντίστοιχο ποσό εισπράξεων.