Διάλεξη 15: Γλώσσα Επεξεργασίας Δεδομένων/ Data Manipulation Language (SQL DML) III

Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:
Εισαγωγή στις έννοιες:

- Προχωρημένες Συνθήκες (LIKE, IS, Συναρτήσεις, κα.)
- Εμφωλευμένες Επερωτήσεις (IN, EXISTS)
- Διαίρεση

Διδάσκων: Παναγιώτης Ανδρέου

Προχωρημένες συνθήκες επιλογής/ελέγχου

- Οι συνθήκες επιλογής/ελέγχου Θ δεν συμπεριλαμβάνουν πάντα ισότητες ή ανισότητες
  - WHERE NAME LIKE ‘Beer%’
  - ADD CONSTRAINT C_TEST CHECK (NAME LIKE ‘Beer%’)
- Μπορούν να είναι αρκετά περίπλοκές και να κάνουν χρήση συναρτήσεων
- Σημαντικοί τελεστές/συναρτήσεις
  - **LIKE**: Σύγκριση **συμβολοσειρών**
  - **Συναρτήσεις: SUBSTRING, SOUNDEX, DATEDIFF**
    Σημαντικό να γνωρίζουμε τι υποστηρίζει η ΒΔ
  - **IS**: Σύγκριση με **NULL, ISNULL**
  - **IN**: Σύγκριση με **σύνολα**

Σύγκριση Συμβολοσειρών με LIKE

- Για την σύγκριση συμβολοσειρών (substring matching) σε SQL γίνεται χρήση του **LIKE**.

WHERE Attribute [NOT] LIKE Pattern

  - **Attribute**: Γνώρισμα ή οποιαδήποτε έγκυρη έκφραση.
  - **Pattern:** Συμβολοσειρά (υπό μορφή «Κανονικής Έκφρασης») η οποία θα αναζητηθεί στο attribute.
    - Το Pattern μπορεί να περιέχει χαρακτήρες wildcard
    - Το pattern μπορεί να είναι μέχρι 8,000 bytes στην TSQL
- Παράδειγμα

```
SELECT FirstName, LastName, Phone
FROM Person.Contact
WHERE phone LIKE '415%'
```

Σύγκριση Συμβολοσειρών με LIKE

Wildcard character	Περιγραφή	Παράδειγμα
%	**Συμβολοσειρά 0 ή περισσοτέρων χαρακτήρων**	**WHERE title LIKE '%computer%'** finds all book titles with the word 'computer' anywhere in the book title.
_ (underscore)	**Οποιοσδήποτε Χαρακτήρας**	**WHERE name LIKE '_ean'** finds all four-letter first names that end with ean (Dean, Sean, and so on).
[ ]	**Οποιοσδήποτε χαρακτήρας σε εύρος ([a-f]) ή σύνολο ([abcdef]).**	**WHERE name LIKE '[C-P]arsen'** finds author last names ending with arsen and starting with any single character between C and P, for example Carsen, Larsen, Karsen, and so on..
[^]	**Οποιοσδήποτε χαρακτήρας που ΔΕΝ είναι σε ευρος ([^a-f]) ή σύνολο ([^abcdef]).**	**WHERE name LIKE 'de[^l]%'** all author last names starting with de and where the following letter is not l.

Σύγκριση Συμβολοσειρών με LIKE

- **Παράδειγμα: Βρες τους υπαλλήλους που γεννήθηκαν κατά την δεκαετία του 1950s.**
  - Θεωρήστε ότι η ημερομηνία έχει την μορφοποίηση: MMDDYYYY
  - Συνεπώς, ψάχνουμε το BDATE με τιμή '_ _ _ _ 1 9 5_', όπου το underscore υποδηλώνει ένα αυθαίρετο χαρακτήρα.
  - SELECT FNAME, LNAME
    FROM EMPLOYEE
    WHERE **BDATE LIKE '_ _ _ _ 1 9 5_'**

- **Παράδειγμα: Βρες τα προϊόντα που το όνομα τους είναι "Πανάκια για " + 1 ή 2 ή 3 " χρόνια".**
  - SELECT *
    FROM PRODUCTS
    WHERE **PNAME LIKE 'Πανάκια για [123] χρόνια'**

Παραδείγματα Συναρτήσεων

- SELECT LastName, **SUBSTRING(**FirstName, 1, 1**)** AS Initial
- SELECT FNAME, LNAME, **1.1*SALARY AS T_SALARY**
- SELECT **SOUNDEX(**Name**)**
  - Επιστρέφει ένα κωδικό 4 χαρακτήρων που μπορεί να αξιοποιηθεί για να αποτιμηθεί εάν 2 strings ακούγονται το ίδιο
    Π.χ., SELECT SOUNDEX ('Smith'), SOUNDEX ('Smythe');
    Επιστρέφει S530 S530
- SELECT **DIFFERENCE(**Name, Surname**)**
  - Βρίσκει την ομοιότητα δυο γνωρισμάτων βάσει του SOUNDEX code.
  - Λαμβάνει υπόψη τα πρώτα 8000 bytes των char, varchar ή text
  - Παίρνει τιμές από 0 (καμία ομοιότητα) .. 4 (ίδια)
- … και πολλές άλλες

Συγκρίσεις με NULL

- Σε τυπικές γλώσσες προγραμματισμού, οι λογικές εκφράσεις αποτιμούνται σε **TRUE** ή **FALSE**.
- Στην SQL ωστόσο, η ύπαρξη NULL τιμών επιβάλει την χρήση της **Λογικής Τριών Τιμών (Three-value logic 3VL)**
- Συγκεκριμένα, μια λογική έκφραση μπορεί να αποτιμηθεί σε **TRUE**, **FALSE** ή **UNKNOWN**
  - Π.χ., NULL AND TRUE αποτιμάται σε UNKNOWN
- Η αποτίμηση λογικών εκφράσεων γίνεται με βάσει των ακόλουθων πινάκων αληθείας:

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

NOT	
TRUE	FALSE
FALSE	TRUE
UNKNOWN	UNKNOWN

Συγκρίσεις με NULL

Παράδειγμα

```
CREATE TABLE test (
    id INTEGER PRIMARY KEY,
    age INTEGER CHECK (age < 0 AND age = 0 AND age> 0)
)
```

- **Τι θα γίνει εάν προσπαθήσουμε να εισάγουμε το age=NULL;**
  - Θα απαγορέψει την εισαγωγή οποιασδήποτε τιμής ωστόσο μπορεί να εισαχθεί το NULL

- **Για να αποφύγω και τα NULL πλήρως:**

```
CREATE TABLE test (id INTEGER PRIMARY KEY,
        age INTEGER NOT NULL CHECK (….) );
```

Συγκρίσεις με NULL

- Μια έκφραση **WHERE Attribute=NULL**, είναι λάθος
- Για σύγκριση ενός γνωρίσματος με NULL στην SQL χρησιμοποιείται η έκφραση **IS NULL ή NOT IS NULL**.
- **Επισημάνσεις**
  - **ANSI**: Δυο NULL τιμές είναι ανεξάρτητες (διαφορετικές)
  - **MSSQL**: ΔΥΟ NULL τιμές είναι οι ίδιες (εξ' ορισμού). Επομένως σε UNIQUE πεδίο δεν μπορούμε να έχουμε δυο εγγραφές με τιμή NULL.
    - Αυτό, επειδή υπάρχει η εξ' ορισμού ρύθμιση **ANSI_NULLS OFF** (στις ρυθμίσεις μιας βάσης)
    - Εάν ενεργοποιηθεί η εν λόγω μεταβλητή, με **SET ANSI_NULLS ON,** τότε δυο διακριτές NULL τιμές θα θεωρούνται διαφορετικές όπως άλλωστε προβλέπει και το ANSI πρότυπο.
- Χρήσιμη συνάρτηση: ISNULL(<attr>, <default>) – αν το <attr> είναι null τότε επιστρέφει μία default τιμή <default>

Συγκρίσεις με Σύνολα (IN)

- Όταν θέλουμε να **συγκρίνουμε με ένα σύνολο διακριτών τιμών** τότε μπορούμε να χρησιμοποιήσουμε τον τελεστή σύγκρισης **IN** (παρόμοια με SQL-DDL)
  - Π.χ., SELECT * FROM PERSON **WHERE DNO IN (1, 2)**
- Το IN έχει αντίστοιχη λογική με τον τελεστή συνόλων ∈ και μας επιτρέπει να ομαδοποιήσουμε πολλές συνθήκες OR μαζί.
  - Π.χ., SELECT * FROM PERSON **WHERE DNO=1 OR DNO=2**
- Αν είχαμε 100 διαφορετικές συνθήκες;
  - 100 διαφορετικά OR ή μία λίστα (σύνολο) με 100 τιμές και σύγκριση με IN
- Τι συμβαίνει αν δεν ξέρουμε εκ των προτέρων το σύνολο με τις διακριτές τιμές;

PERSON (P)

PID	PNAME	DID
1	Andreas	1
2	Kostas	1
3	Maria	1
4	Eleni	2
5	Nikos	2
6	Eleni	3

Εμφωλευμένες Επερωτήσεις (Nested Queries)

- Οι εμφωλευμένες επερωτήσεις μας επιτρέπουν να εκφράσουν κάποιο αποτέλεσμα που είναι σύνολο (πίνακας)
  - Π.χ.,

```
SELECT      *                          Outer
FROM        PERSON                     scope
WHERE       DNO IN (
                SELECT DNO                                  Inner
                FROM DEPARTMENT                             scope
                WHERE DNAME = 'ACC' OR DNAME = 'IT'
            )
```

- Παρόμοια με τις γλώσσες προγραμματισμού, υπάρχει η έννοια της εμβέλειας (scope)
  - Στο εσωτερικό scope μπορούμε να αναφερθούμε σε πίνακες του εξωτερικού scope

Εμφωλευμένες Επερωτήσεις (συν.)

- Με τις εμφωλευμένες επερωτήσεις και τον τελεστή IN μπορούμε να απαντήσουμε και τις επερωτήσεις που παρουσίαζαν πρόβλημα με τη σύγκριση τιμής με σύνολο

- Παράδειγμα: Βρες τον υπάλληλος με το μέγιστο μισθό
  - SELECT * FROM EMPLOYEE
    WHERE **SALARY = MAX(SALARY)** **→ΛΑΘΟΣ**

  - SELECT * FROM EMPLOYEE
    WHERE **SALARY IN (**
    SELECT MAX(SALARY)
    FROM EMPLOYEE) **→ΟΡΘΟ**

Εμφωλευμένες Επερωτήσεις (συν.)

- **ΚΑΘΕ Εμφωλευμένη Επερώτηση** μπορεί να αναπαρασταθεί από Επερώτηση 1-μπλοκ με χρήση πράξεων συνόλων, συνενώσεις ή συνδυασμό.
  - Δεν είναι αποδοτική πράξη γι' αυτό πρέπει να αποφεύγεται όπου είναι δυνατό
- Μία ΒΔ συνήθως περιορίζει τον αριθμό αναδρομικών εμφωλεύσεων (π.χ., στην TSQL μέχρι 32 επίπεδα).
- Συνήθως χρησιμοποιούνται για δημιουργία δυναμικών όψεων (views)
- Μπορούν να εκφράσουν τα ενδιάμεσα αποτελέσματα της ΣΑ

```
SELECT *
FROM (SELECT * FROM PARTS WHERE PID>5) Q1
        INNER JOIN
        (SELECT * FROM CATALOG WHERE SID>5) Q2
        ON Q1.SID=Q2.SID
```

Συσχετιζόμενες Εμφωλευμένες Επερωτήσεις

- Όταν μια συνθήκη στο WHERE του Outer-block αναφέρεται σε κάποιο γνώρισμα του Inner-block τότε οι δυο επερωτήσεις λέγεται ότι είναι **Συσχετιζόμενες (Correlated)**

- **Query:** Βρες το **όνομα** κάθε employee που έχει ένα **dependent** με το **ίδιο όνομα** με τον **employee**.

```
SELECT      E.FNAME, E.LNAME
FROM        EMPLOYEE E
WHERE       E.SSN IN
              (SELECT   D.ESSN
               FROM     DEPENDENT D
               WHERE    D.ESSN=E.SSN AND
                        E.FNAME=D.DEPENDENT_NAME)
```

Εμφωλευμένες Επερωτήσεις (ALL, ANY)

- Οι τελεστές **ANY** και **ALL**, χρησιμοποιούνται σε συνδυασμό με τους αριθμητικούς τελεστές σύγκρισης σε εμφωλευμένες επερωτήσεις
- Επιτρέπουν την **σύγκριση τιμής με όλες τις τιμές κάποιου συνόλου** που είναι το αποτέλεσμα μίας εμφ. επερώτησης
- **Παράδειγμα**

```
SELECT * FROM EMPLOYEE
WHERE SALARY >= ALL (
                SELECT SALARY
                FROM EMPLOYEE )
```

- **ALL:** οι πλειάδες που επιστρέφονται ικανοποιούν την συνθήκη για όλες τις πλειάδες της εμφ. ερώτησης
- **ANY:** οι πλειάδες που επιστρέφονται ικανοποιούν την συνθήκη για τουλάχιστον μία πλειάδα της εμφ. ερώτησης
- Ο τύπος του γνωρίσματος (π.χ., SALARY) πρέπει να συμφωνεί με τον τύπο της εμφωλευμένης ερώτησης

Εμφωλευμένες Επερωτήσεις (EXISTS)

- Η εντολή **EXISTS επιστρέφει TRUE** εάν το αποτέλεσμα μιας εμφωλευμένης επερώτησης **υπάρχει** (**ΔΕΝ είναι κενό/null**)

WHERE [NOT] EXISTS subquery

- Το **EXISTS** έχει αντίστοιχη λογική με τον υπαρξιακό ποσοδείκτη ∃*x*
- Παρόμοια το NOT EXISTS **επιστρέφει TRUE** εάν το αποτέλεσμα μιας εμφωλευμένης επερώτησης **ΔΕΝ υπάρχει** (**είναι κενό/null**)
- Το EXISTS μπορεί να διατυπωθεί και με άλλους τρόπους (π.χ., IN)
  Παράδειγμα: Βρες το όνομα κάθε employee που έχει ένα dependent με το ίδιο όνομα με τον employee.

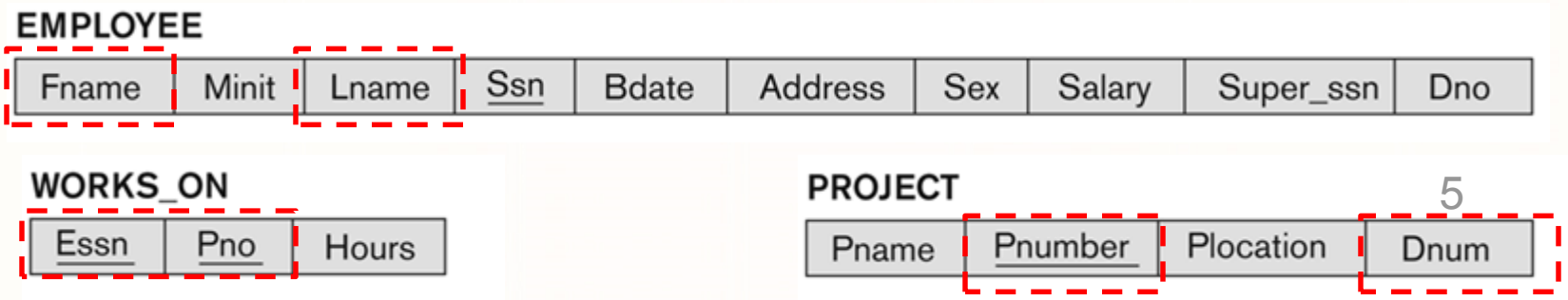
```
SELECT      E.FNAME, E.LNAME
FROM        EMPLOYEE E
WHERE       EXISTS ( SELECT      *
                     FROM DEPENDENT D
                     WHERE       E.SSN=D.ESSN    AND
                                 E.NAME = D.DEPENDENT_NAME)
```

Διαίρεση

- Παρόλο που οι αρχικές εκδόσεις της SQL όριζαν **εξειδικευμένη εντολή διαίρεσης**, την **CONTAINS**, μια τέτοια δυνατότητα **ΔΕΝ υπάρχει** στα **νεότερα πρότυπα** και **υλοποιήσεις**:
  - Ενδεχόμενοι **Λόγοι Εγκατάλειψης** του CONTAINS:
    - Δυσκολία **Αποδοτικής Υλοποίησης**
    - **Μειωμένη Χρήση** της εν λόγω εντολής
  - Στα πλαίσια αυτού του μαθήματος θεωρήστε ότι **ΔΕΝ** υπάρχει η εντολή διαίρεσης CONTAINS
- Για να **διαιρέσουμε δυο σχέσεις** θα χρησιμοποιήσουμε την λογική του **NOT EXISTS …. EXCEPT** που ακολουθεί.
- Σημειώστε ότι σε **TSQL**, η εντολή **CONTAINS** χρησιμοποιείται για κάτι διαφορετικό … την **αναζήτηση σε πεδία κειμένου (όπως η LIKE)**
- **Παράδειγμα Διαίρεσης:** Βρες τα "ονόματα των employees" που δουλεύει σε ΚΑΘΕ "project που ελέγχεται από το department 5"

Παράδειγμα Διαίρεσης 1

- Βρες τα "**ονόματα των employees**" που δουλεύει σε "**ΚΑΘΕ project που ελέγχεται από το department 5**"




```
SELECT   E.FNAME, E.LNAME FROM  EMPLOYEE E
WHERE   NOT EXISTS (
            SELECT          P.Pnumber
            FROM            PROJECT P
            WHERE           P.Dnum=5

            EXCEPT

            SELECT          W.PNO
            FROM            WORKS_ON W
            WHERE           E.SSN=W.ESSN
)
```

Συσχέτιση κάθε υπαλλήλου στο εξωτερικό block, με τα projects τα οποία δουλεύει που θέλουμε να συσχετίσουμε με το πρώτο εσωτερικό block

Παράδειγμα Διαίρεσης 2

Suppliers(sid, sname, address)

sid	sname	Address
s1	Supplier 1	Address 1
s2	Supplier 2	Address 2
s3	Supplier 3	Address 3
s4	Supplier 4	Address 4

Parts(pid, pname, color)

pid	pname	color
p1	Part 1	red
p2	Part 2	red
p3	Part 3	green
p4	Part 4	blue

sid	pid	cost
s1	p1	100
s1	p2	150
s1	p3	100
s1	p4	200
s2	p1	100
s2	p2	150
s3	p2	100
s4	p2	200
s4	p4	250

Query: Βρες τους suppliers που πουλάνε τα parts p2 και p4

Query: Βρες τους suppliers που δεν υπάρχει part (μέσα στη λίστα με τα parts p2 και p4) που να μην τα παρέχουν

Διπλή Άρνηση ➔ Διπλό NOT EXISTS

Παράδειγμα Διαίρεσης 2

Query: Βρες τους suppliers που δεν υπάρχει part (μέσα στη λίστα με τα parts p2 και p4) που να μην τα παρέχουν

A: SELECT **s**.sid FROM suppliers **s**

WHERE **NOT EXISTS** (

SELECT **p**.pid FROM parts **p** WHERE pid IN ('p2', 'p4')

AND **NOT EXISTS** (

SELECT **c**.pid from catalog **c** WHERE **s**.sid = **c**.sid AND **p**.pid=**c**.pid))

B: SELECT **s.**sid FROM suppliers **s**

WHERE **NOT EXISTS** (

SELECT **p.**pid FROM parts **p** WHERE p.pid IN ('p2', 'p4')

EXCEPT

SELECT **c.**pid FROM catalog **c** WHERE **s**.sid=**c.**sid)

Γ: SELECT **s.**sid FROM suppliers **s**

WHERE **NOT EXISTS** (

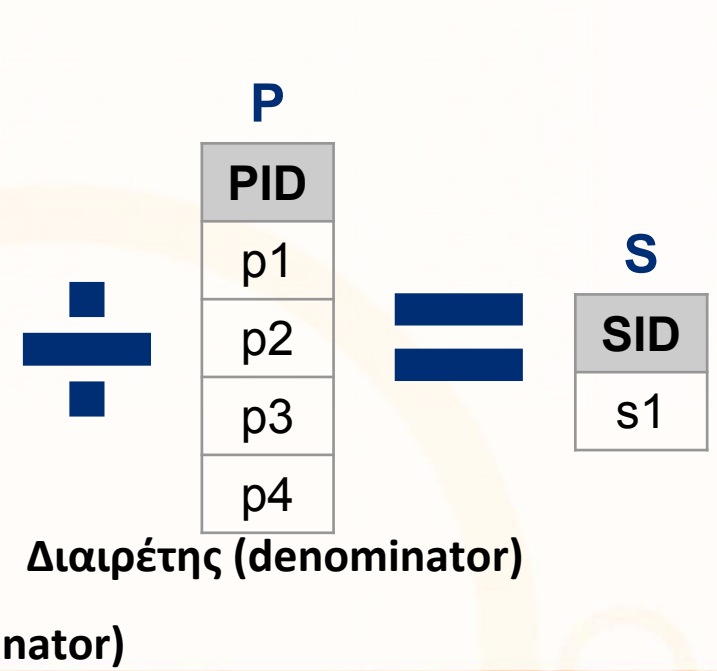
SELECT **p.**pid FROM parts **p** WHERE pid IN ('p2', 'p4')

AND pid **NOT IN** (SELECT pid FROM catalog **c** WHERE **s**.sid=**c**.sid))

Τελεστής Διαίρεσης (Division Operator)

$$R \div S \quad R / S = \{ x \mid \exists (x,y) \in R \ \ \forall y \in S \}$$

- **Είσοδος:** Δύο σχέσεις R και S
- **Έξοδος:** Νέα σχέση η οποία περιλαμβάνει τις πλειάδες της R οι οποίες μπορούν να συνενωθούν με κάθε πλειάδα της S σύμφωνα με τα κοινά τους γνωρίσματα.
- **Παράδειγμα:** *Βρες τους Suppliers (S) που προσφέρουν ΌΛΑ τα Parts (P)* **Suppliers ÷ Parts**



- Δεν είναι βασικός τελεστής: μπορεί να εκφραστεί και σαν
  - $T_1 \leftarrow \pi_A (R)$
  - $T_2 \leftarrow \pi_A ((T_1 x S) - R)$
  - $T \leftarrow T_2 - T_1$

Τελεστής Διαίρεσης (Division Operator)

- R=SUPPLIERS_PARTS, S=PARTS

1. $T_1 \leftarrow \pi_A(R)$: Βρες τους μοναδικούς Suppliers

T_1

SID
s1
s2
s3
s4

2. $T_2 \leftarrow \pi_A((T_1 x S) - R)$
   - Βρες όλους τους συνδυασμούς των μοναδικών Suppliers με τα parts της επερώτησης (όλα τα parts)
   - Αφαίρεσε τους συνδυασμούς που υπάρχουν ώστε να παραμείνουν οι ανεπιθύμητοι συνδυασμοί
   - Επέστρεψε λίστα με SID

T1xS

SID	PID
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s2	p3
s2	p4
s3	p1
s3	p2
s3	p3
s3	p4
s4	p1
s4	p2
s4	p3
s4	p4

−

R

SID	PID
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

=

$(T_1 x S) - R$

SID	PID
s2	p3
s2	p4
s3	p1
s3	p3
s3	p4
s4	p1
s4	p3

T_2

SID
s2
s3
s4

3. $T \leftarrow T_1 - T_2$
   - Αφαίρεσε από την αρχική λίστα τους ανεπιθύμητους

T

SID
s1

Παράδειγμα Διαίρεσης (από Σχεσ. Αλγ.)

Suppliers(sid, sname, address) **Parts(pid, pname, color)** **Catalog(sid, pid, cost)**

Δ: SELECT sid FROM catalog $\pi_A(R)$

EXCEPT —

SELECT sid FROM ($\pi_A((\pi_A(R) \times S) - R)$

$(\pi_A(R) \times S) - R$

SELECT DISTINCT sid, Q1.pid

FROM catalog, $\pi_A(R)$ ×

(SELECT pid FROM parts **S**

WHERE pid IN ('p2', 'p4')) Q1

$((\pi_A(R) \times S)$

EXCEPT —

SELECT sid, pid FROM catalog **R**

) Q2

$$R \div S = \pi_A(R) - \pi_A((\pi_A(R) \times S) - R)$$

Διάφορα Θέματα – Συνάρτηση CASE

- **Συνάρτηση/Τελεστής CASE**: επιτρέπει την δημιουργία καινούριων γνωρισμάτων εφαρμόζοντας συγκρίσεις σε υπάρχων γνωρίσματα ή μεταβλητές
- Παράδειγμα

```
SELECT PID, PNAME,
CASE
        WHEN GENDER IS NULL THEN 'N/A'
        WHEN GENDER='F' THEN 'Female'
        WHEN GENDER='M' THEN 'Male'
        ELSE 'Error'
END
AS GENDER          (or AS <όνομα>)

FROM PERSON
```

PERSON (P)

PID	PNAME	GENDER
1	Andreas	M
2	Kostas	M
3	Maria	F
4	Eleni	F
5	Nikos	NULL

PERSON (P)

PID	PNAME	GENDER
1	Andreas	Male
2	Kostas	Male
3	Maria	Female
4	Eleni	Female
5	Nikos	N/A

Διάφορα Θέματα – COUNT(DISTINCT *)

- **Με τις εμφωλευμένες ερωτήσεις μπορούμε να εκφράσουμε το** COUNT(DISTINCT *) το οποίο δεν υποστηρίζεται
- Παράδειγμα

```
SELECT COUNT(*)
FROM (
        SELECT DISTINCT *  FROM …
) Q
```

- Το DISTINCT * χρειάζεται μόνο εάν ο πίνακας ΔΕΝ έχει Primary Key.
- Εάν έχει PRIMARY key ο πίνακας τότε εξυπακούεται, δηλ., είναι ισοδύναμο με SELECT COUNT(*) FROM table.