Διάλεξη 14: Γλώσσα Επεξεργασίας Δεδομένων/ Data Manipulation Language (SQL DML) II

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

- Συνενώσεις με Καρτεσιανό Γινόμενο και JOIN
- Συναθροιστικές Συναρτήσεις
- Ομαδοποίηση (GROUP BY, HAVING)

Διδάσκων: Παναγιώτης Ανδρέου

Καρτεσιανό Γινόμενο και Συνενώσεις σε SQL

- **Παράδειγμα με δύο ή περισσότερους πίνακες:**
  - Αναφορά με Ταυτότητα, Όνομα Υπαλλήλου και όνομα τμήματος του υπαλλήλου
  - Οι πίνακες (σχέσεις) συνενώνονται βάση των ξένων κλειδιών

D: DEPARTMENT

DNO	DNAME

E: EMPLOYEE

SSN	ENAME	DOB	GENDER	DNUMBER

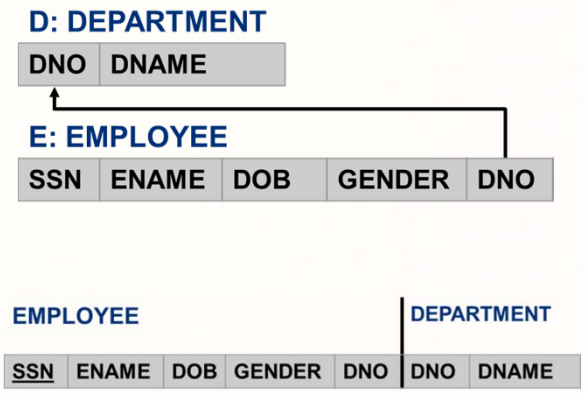
E: EMPLOYEE

SSN	ENAME	DOB	GENDER	DNO

- **Σχεσιακή Άλγεβρα:**
  - $\pi_{SSN,ENAME,DNAME}\ \sigma_{E.DNO=D.DNO}$ (EMPLOYEE x DEPARMENT)
  - $\pi_{SSN,ENAME,DNAME}$ (EMPLOYEE $\bowtie_{E.DNO=D.DNO}$ DEPARMENT)
  - $\pi_{SSN,ENAME,DNAME}$ (EMPLOYEE *DEPARMENT)

Καρτεσιανό Γινόμενο και Συνενώσεις σε SQL

- Οι **συνενώσεις σε SQL** μπορούν να γίνουν με τους τελεστές: **Καρτεσιανό Γινόμενο** και **JOIN**
- **Το αποτέλεσμα είναι** μία καινούρια **όψη** η οποία συμπεριλαμβάνει όλα τα γνωρίσματα των σχέσεων



- **Καρτεσιανό Γινόμενο ,/CROSS JOIN:**
  - ΣΑ: $\pi_{SSN,ENAME,DNAME}\ \sigma_{E.DNO=D.DNO}$ (**EMPLOYEE x DEPARMENT**)
  - SQL:
    ```
    SELECT  SSN,ENAME,DNAME
    FROM    EMPLOYEE, DEPARMENT      ( ή EMPLOYEE CROSS JOIN DEPARMENT)
    WHERE   EMPLOYEE.DNO=DEPARTMENT.DNO
    ```
- **Συνένωση JOIN:**
  - ΣΑ: $\pi_{SSN,ENAME,DNAME}$ (**EMPLOYEE** $\bowtie_{E.DNO=D.DNO}$ **DEPARMENT**)
  - SQL:
    ```
    SELECT  SSN,ENAME,DNAME
    FROM    EMPLOYEE JOIN DEPARMENT  ( ή EMPLOYEE INNER JOIN DEPARMENT)
            ON EMPLOYEE.DNO=DEPARTMENT.DNO
    ```

Αντιστοίχηση Σχεσιακής Άλγεβρας με SQL-DML

R

A	B
a	1
b	2
c	4

S

B	C
1	x
2	y
3	z

$R \bowtie_{\Theta=B\ (R.B=S.B)} S$

A	B	C
a	1	x
b	2	y

Συνένωση Θ/ Θ-Join ($\bowtie_\Theta$)

θ: R.B=S.B

SELECT R.A, R.B, S.C
FROM R JOIN S ON θ

R x S

A	B	B	C
a	1	1	x
a	1	2	y
a	1	3	z
b	2	1	x
b	2	2	y
b	2	3	z
c	4	1	x
c	4	2	y
c	4	3	z

SELECT R.A, R.B, S.C
FROM R , S (CROSS JOIN)

$R * S$

A	B	C
a	1	x
b	2	y

Φυσική Συνένωση/ Natural Join (*)

SELECT A,B,C
FROM R * S

$R \ltimes_B S$

A	B
a	1
b	2

Ημι-Συνένωση / Semi Join ($\ltimes_\Theta$)

SELECT R.A, R.B
FROM R JOIN S ON θ

$R \triangleright_B S$

A	B
c	4

Αντι-Συνένωση / Anti Join ($\triangleright_\Theta$)

?

$R \ \overline{\bowtie}_B \ S$

A	B	C
a	1	x
b	2	y
c	4	

Αριστερή Εξ. Συνένωση/ Left (Outer) Join ($\overline{\bowtie}_\Theta$)

SELECT R.A, R.B , S.C
FROM R
LEFT JOIN S ON θ

$R \ \overline{\bowtie}_B \ S$

A	B	C
a	1	x
b	2	y
c	4	
	3	z

Ολική Εξ. Συνένωση/ Full (Outer) Join ($\overline{\bowtie}_\Theta$)

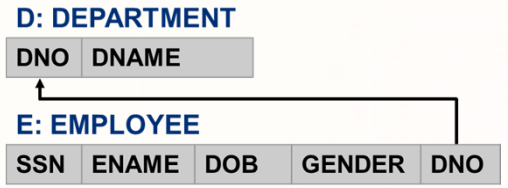
FROM R FULL
OUTER JOIN S ON θ

Διφορούμενα Ονόματα Γνωρισμάτων

- Στις συνενώσεις, υπάρχει μεγάλη πιθανότητα κάποια **ονόματα μεταξύ διαφορετικών σχέσεων να είναι όμοια (Διφορούμενα Ονόματα Γνωρισμάτων)**
  - Π.χ., Employee(ssn, **name**), Department(dno, **name**), Project(pid, **name**)
- Για να αντιμετωπίσουμε αυτό το πρόβλημα, μπορούμε:
  - Να αναφερόμαστε με το **όνομα του πίνακα σαν πρόθεμα**
    - Π.χ., **Employee.name**, **Department.name**, **Project.name**
  - Να χρησιμοποιούμε **μοναδικά ονόματα** για όλα τα γνωρίσματα
    - Πρακτικά πολύ δύσκολο όταν υπάρχουν πολλοί πίνακες
  - Να χρησιμοποιούμε **συγκεκριμένη ονοματολογία**
    - Π.χ., <Όνομα Πίνακα>_<Όνομα Γνωρίσματος>
      Για παράδειγμα: Department(dno, **Department_name**), Project(pid, **Project_name**)
  - Να χρησιμοποιούμε τη **μετονομασία (AS)**

Διφορούμενα Ονόματα Γνωρισμάτων

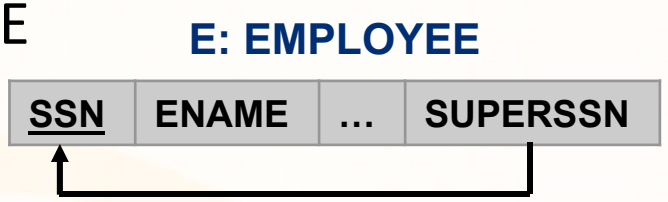
- Παράδειγμα μετονομασίας




```
SELECT    E.SSN, E.ENAME, D.DNAME
FROM      EMPLOYEE [AS] E, DEPARMENT [AS] D
WHERE     E.DNO=D.DNO
```

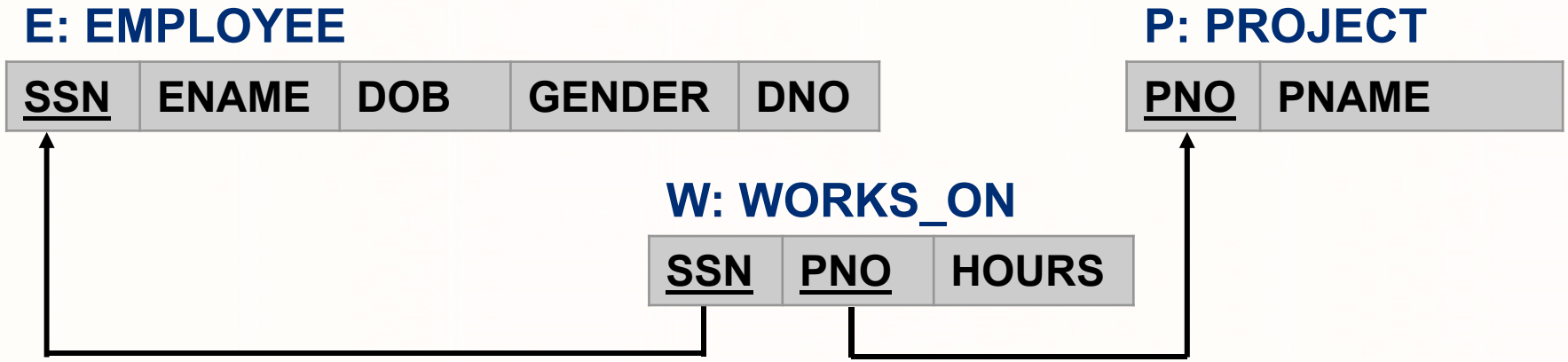
- Όπως αναφέραμε, η μετονομασία είναι ο μόνος τρόπος να αναφερθούμε στον ίδιο πίνακα 2 ή περισσότερες φορές

```
SELECT    E.SSN, E.ENAME, S.SSN, S.ENAME
FROM      EMPLOYEE E, EMPLOYEE S
WHERE     E.SUPERSSN=S.SSN
```



- Ακόμη και εάν **δεν υπάρχουν διφορούμενα** ονόματα, είναι κάλο να χρησιμοποιείται το **AS** για πιο **ευανάγνωστο** κώδικα.
- Για **μετονομασία γνωρισμάτων** (όχι σε TSQL)
  - **FROM EMPLOYEE AS E(EID, ENAME, EDNO)**

Παράδειγμα με 3 Πίνακες



- **Καρτεσιανό Γινόμενο**

```
SELECT    E.SSN, E.ENAME, D.DNAME
FROM      EMPLOYEE E, WORKS_ON W, PROJECT P
WHERE     E.SSN=W.SSN  AND  P.PNO=W.PNO
```

- **Συνένωση JOIN**

```
SELECT    E.SSN, E.ENAME, D.DNAME
FROM      EMPLOYEE E
          INNER JOIN WORKS_ON W ON E.SSN=W.SSN
          INNER JOIN PROJECT P ON P.PNO=W.PNO
```

Καρτεσιανό Γινόμενο vs. JOIN

- Το **Καρτεσιανό Γινόμενο** με επιλογή (WHERE Θ) είναι **ισοδύναμο** με το **JOIN** ON Θ

- Το JOIN έχει το πλεονέκτημα ότι **δεν περιπλέκει** την **συνθήκη συνένωσης** με την **συνθήκη της επερώτησης**
  - **Π.χ.,** Αναφορά με υπαλλήλους που δουλεύουν για το **'Research Department'**

```
SELECT   E.SSN, E.ENAME, D.DNAME
FROM     EMPLOYEE E, DEPARMENT D
WHERE    E.DNO=D.DNO AND D.DNAME='Research'

SELECT   SSN,ENAME,DNAME
FROM     EMPLOYEE E
         JOIN DEPARMENT  D ON E.DNO=D.DNO
WHERE    D.DNAME='Research'
```

Φυσική Συνένωση (NATURAL JOIN)

- Το **NATURAL JOIN** (σε ANSI-SQL, π.χ., υλοποιείται σε PostgreSQL αλλά όχι σε T-SQL) υλοποιεί την συνένωση με τέτοιο τρόπο ώστε η **συνθήκη συνένωσης να είναι τα κοινά γνωρίσματα**.
- Παράδειγμα **Φυσικής Συνένωσης NATURAL JOIN:**
  - ΣΑ: $\pi_{SSN,ENAME,DNAME}$ (EMPLOYEE *DEPARMENT)
  - SQL: SELECT SSN,ENAME,DNAME
    FROM **EMPLOYEE NATURAL JOIN DEPARMENT**
- Δεν συμπεριλαμβάνει τα κοινά γνωρίσματα στην όψη

- Εάν δεν υπάρχει κοινά διατυπωμένο γνώρισμα μπορεί να προηγηθεί μετονομασία:

  - Π.χ., SELECT * FROM EMPLOYEE NATURAL JOIN
    DEPARMENT **AS DEPARMENT(DNUMBER, DNAME)**

Ημι-Συνένωση (SemiJoin)

- Όμοιο με JOIN αλλά **επιστρέφει μόνο τα γνωρίσματα του πρώτου πίνακα**
- **Υλοποιείται με το JOIN**
- Παράδειγμα **Ημι-Συνένωσης (SEMI JOIN):**
  - ΣΑ: $\pi_{SSN,ENAME,DNAME}(P \ltimes_{P.DID=D.DNO} D)$
  - SQL: SELECT PID, NAME, DID
    FROM **P JOIN D ON P.DID=D.DNO**

PERSON (P)

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

DEPARTMENT (D)

DNO	DNAME
1	ACC
2	HR

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

Αριστερή/Δεξιά Συνένωση (Left Join)

- Παρόμοιο με JOIN με τη διαφορά ότι **επιστρέφει και τις πλειάδες που δεν ικανοποιούν την συνθήκη συνένωσης από τον πρώτο πίνακα (αριστερό πίνακα)**
  - Τα γνωρίσματα πλειάδων που δεν ικανοποιούν την συνθήκη έχουν τιμή null
- **Παράδειγμα Αριστερής Συνένωσης:**
  - ΣΑ: $\pi_{SSN,ENAME,DNAME}(P \overset{}{\supset\!\bowtie}_{P.DID=D.DNO} D)$
  - SQL: SELECT PID, NAME, DID
    FROM **P LEFT JOIN D ON P.DID=D.DNO**

PERSON

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

DEPARTMENT

DNO	DNAME
1	ACC
2	HR

PID	PNAME	DID	DNO	DNAME
1	Andreas	1	1	ACC
2	Kostas	1	1	ACC
3	Maria	1	1	ACC
4	Eleni	2	2	HR
5	Nikos	2	2	HR
6	**Eleni**	**3**	**null**	**null**

Αριστερή/Δεξιά Συνένωση (Left Join)

- Η δεξιά συνένωση είναι συμμετρική με την αριστερή
  - **P LEFT JOIN D** ON P.DID=D.DNO ⇔
  - **D RIGHT JOIN P** ON P.DID=D.DNO
- Προσοχή στις περιπτώσεις με περισσότερους από 2 πίνακες
  - Η αριστερή συνένωση πρέπει να διαδοθεί ανάλογα
  - Παράδειγμα: Τι θα επιστρέψει το πιο κάτω;

```
SELECT PID, NAME, DID
FROM    P LEFT JOIN D ON P.DID=D.DNO
        INNER JOIN D ON D.CID=C.CID
        LEFT JOIN D ON D.CID=C.CID
```

PERSON

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

DEPARTMENT

DNO	DNAME	CID
1	ACC	1
2	HR	1

CAMPUS

CID	CNAME
1	Campus A

PID	PNAME	DID	DNO	DNAME	CID	CID	CNAME
1	Andreas	1	1	ACC	1	1	Campus A
2	Kostas	1	1	ACC	1	1	Campus A
3	Maria	1	1	ACC	1	1	Campus A
4	Eleni	2	2	HR	1	1	Campus A
5	Nikos	2	2	HR	1	1	Campus A
6	Eleni	3	null	null	null		

Ολική Εξωτερική Συνένωση (Full Outer Join)

- Παρόμοιο με LEFT JOIN με τη διαφορά ότι **επιστρέφει και τις πλειάδες που δεν ικανοποιούν την συνθήκη συνένωσης και από τους δύο πίνακες**
- **Παράδειγμα Αριστερής Συνένωσης:**
  - ΣΑ: $\pi_{SSN,ENAME,DNAME}(P \bowtie_{P.DID=D.DNO} D)$
  - SQL: SELECT PID, NAME, DID
    FROM **P FULL OUTER JOIN D ON P.DID=D.DNO**

PERSON

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

DEPARTMENT

DNO	DNAME
1	ACC
2	HR
4	**IT**

PID	PNAME	DID	DNO	DNAME
1	Andreas	1	1	ACC
2	Kostas	1	1	ACC
3	Maria	1	1	ACC
4	Eleni	2	2	HR
5	Nikos	2	2	HR
6	**Eleni**	**3**	**null**	**null**
null	**null**	**null**	**4**	**IT**

Αντι-Συνένωση (AntiJoin)

- Πως θα εκφράσουμε τον τελεστή AntiJoin της σχεσιακής άλγεβρας σε SQL;
- **Θυμηθείτε ότι R $\triangleright_{\Theta}$ S:** Νέα σχέση η οποία συνενώνει τις R και S και περιλαμβάνει **μόνο γνωρίσματα που ανήκουν στην R** και όλες τις πλειάδες που **ΔΕΝ ικανοποιούν την συνθήκη Θ**

PERSON

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

$\triangleright_{\Theta}$

DEPARTMENT

DNO	DNAME
1	ACC
2	HR

PERSON_DEPARTMENT

PID	NAME	DID
6	**Eleni**	**3**

Συναθροιστικές Συναρτήσεις

- Παρόμοια με την ΣΑ, η SQL επιτρέπει την χρήση συναθροιστικών συναρτήσεων για την **ανάκτηση χρήσιμων στατιστικών**
- Παραδείγματα συναρτήσεων: **COUNT, SUM, MAX, MIN, και AVG**
- Μπορούν να δηλωθούν με DISTINCT για να επιστρέψουν τις μοναδικές τιμές

E: EMPLOYEE

SSN	ENAME	DOB	GENDER	DNO	SALARY
1	Andreas	1/1/80	M	1	1000
2	Kostas	1/1/85	M	1	1500
3	Maria	4/3/83	F	1	2000
4	Eleni	6/7/85	F	2	2500
5	Victoria	27/2/84	F	3	3000

- **Επερώτηση:** Βρες τον μέγιστο μισθό από τους υπαλλήλους
- SELECT **MAX(SALARY)** FROM EMPLOYEE
- **Επερώτηση:** Βρες πόσοι είναι οι υπάλληλοι
- SELECT **COUNT(*)** FROM EMPLOYEE
- **Επερώτηση:** Βρες τα μοναδικά φύλα από τους υπαλλήλους
- SELECT **COUNT(DISTINCT GENDER)** FROM EMPLOYEE

Συναθροιστικές Συναρτήσεις

- Οι συναθροιστικές συναρτήσεις μπορούν να χρησιμοποιηθούν με τις συνενώσεις ή καρτεσιανό

E: EMPLOYEE

SSN	ENAME	DOB	GENDER	DNO	SALARY
1	Andreas	1/1/80	M	1	1000
2	Kostas	1/1/85	M	1	1500
3	Maria	4/3/83	F	1	2000
4	Eleni	6/7/85	F	2	2500
5	Victoria	27/2/84	F	3	3000

D: DEPARTMENT

DNO	DNAME
1	ACC
2	HR
3	IT

- **Επερώτηση:** Βρες τον μέγιστο μισθό από τους υπαλλήλους του τμήματος ACC
- SELECT **MAX(SALARY)** AS MSAL FROM E JOIN D ON E.DNO=D.DNO WHERE D.DNAME='ACC'
- SELECT **MAX(SALARY)** FROM E, D WHERE E.DNO=D.DNO AND D.DNAME='ACC'

MSAL
2000

Ομαδοποίηση

- Η ομαδοποίηση, συνεργάζεται (συνήθως) με συναθροιστικές συναρτήσεις για την παραγωγή στατιστικών ανά υποομάδες
- Επιτυγχάνεται μέσω της δήλωσης GROUP BY

E: EMPLOYEE

SSN	ENAME	DOB	GENDER	DNO	SALARY
1	Andreas	1/1/80	M	1	1000
2	Kostas	1/1/85	M	1	1500
3	Maria	4/3/83	F	1	2000
4	Eleni	6/7/85	F	2	2500
5	Victoria	27/2/84	F	3	3000

D: DEPARTMENT

DNO	DNAME
1	ACC
2	HR
3	IT

- **Επερώτηση:** Βρες τον μέγιστο μισθό υπαλλήλων για το κάθε τμήμα
- SELECT **D.DNAME, MAX(E.SALARY) AS MSAL**
  FROM E JOIN D ON E.DNO=D.DNO
  GROUP BY D.DNAME

DNAME	MSAL
1	2000
2	2500
3	3000

Επιλογή μετά από Ομαδοποίηση

- Η δήλωση **HAVING**, χρησιμοποιείται όταν θέλουμε να επιλέξουμε ένα **υποσύνολο** ενός ομαδοποιημένου αποτελέσματος βάσει συνθήκης.
- Π.χ.,

```
SELECT      D.DNAME, MAX(E.SALARY) AS MSAL
FROM  E JOIN D ON E.DNO=D.DNO
GROUP BY    D.DNAME
HAVING      MAX(E.SALARY) > 2000
```

DNAME	MSAL
~~1~~	~~2000~~
2	2500
3	3000

- Το **HAVING** έχει αντίστοιχη λειτουργία με το **WHERE**, μόνο που η **συνθήκη επιλογής** είναι πάνω σε **ομάδες** παρά σε επί μέρους **πλειάδες**.
- Τα γνωρίσματα που εμφανίζονται στο **HAVING** είναι είτε συναθροιστικές συναρτήσεις (π.χ., COUNT, SUM) ή γνωρίσματα που εμφανίζονται στη λίστα του **GROUP BY**
  - Π.χ., **HAVING AVG(E.SALARY)> 1500, HAVING D.DNAME<3**

Εννοιολογική Εκτέλεση Ενός SQL Block

- Ας δούμε λίγο καλύτερα με ποια **σειρά εκτελείται** ένα SQL μπλοκ σε ένα **αφαιρετικό (εννοιολογικό)** επίπεδο.
- Η **περιγραφή** αυτή είναι σε **εννοιολογικό επίπεδο,** επομένως το **πλάνο εκτέλεσης (query plan)** της επερώτησης στα παραδείγματα ΔΕΝ θα είναι βέλτιστο.
- Η **πραγματικό πλάνο εκτέλεσης** εναπόκειται αποκλειστικά στην βάση δεδομένων.
  - Αυτό διεκπεραιώνεται από τον **βελτιστοποιητή επερωτήσεων (query optimizer)** ο οποίος έχει μεγάλη πολυπλοκότητα.
    - Μια υλοποίηση του σε εμπορική βάση λέγεται ότι πήρε 50 ανθρωποχρόνια εργασίας.
- Θεωρούμε ότι η βάση μας είναι **ένας** ή **περισσότεροι πίνακες** αποθηκευμένοι στον δίσκο **χωρίς άλλες δομές δεδομένων** (ευρετήρια αναζήτησης, κτλ).
- Η **περιγραφή** αυτή θα μας επιτρέψει να **καταλάβουμε** καλύτερα τι **παράγεται** από μια **επερώτηση**.

Εννοιολογική Εκτέλεση Ενός SQL Block

- Ένα μπλοκ επερώτησης SQL αποτελείται από έξι όρους (clauses) οι οποίοι εκτελούνται (σε λογικό επίπεδο) όπως φαίνεται πιο κάτω:

  6) SELECT <Attribute(s) AS Alias(s)> **3)** Agg1 AS Alias, …AggN as Alias

  1) FROM <table(s)>

  2) [**WHERE** <condition>]

  3) [**GROUP BY** <grouping attribute(s)>]

  4) [**HAVING** <group condition>]

  5) [**ORDER BY** <attribute list>]

- Ένα Query εκτελείται εννοιολογικά με την ακόλουθη σειρά:

  1. **FROM:** Συνένωσε (ή Καρτεσιανό Γινόμενο) τους πίνακες του table-list, π.χ., FROM Employee E, Department D
  2. **WHERE:** Διάσχισε Γραμμικά τον Πίνακα που παράγεται στο βήμα 1 αποτιμώντας την έκφραση <condition> σε κάθε πλειάδα.

     π.χ., WHERE E.Dno=D.Dnumber and D.Dname="Research"

Εννοιολογική Εκτέλεση Ενός SQL Block

Έστω το ενδιάμεσο αποτέλεσμα: TEMP1(DNO, SSN, Salary)

3. **GROUP BY:** Διάσχισε γραμμικά το TEMP1, ομαδοποιώντας τα αποτελέσματα βάσει του grouping attribute(s) και υπολογίζοντας ανά ομάδα τις αιτούμενες συναθρ. Πράξεις που έχουν δηλωθεί στο **SELECT και HAVING**:
   - MAX, MIN, COUNT, SUM: Υλοποιούνται με μια μεταβλητή ανά ομάδα
   - AVG = SUM / COUNT

   π.χ., GROUP BY DNO

Έστω το ενδιάμεσο αποτέλεσμα: TEMP2(DNO, AVG(salary))

4. **HAVING:** Διάσχισε **TEMP2** εφαρμόζοντας το <group condition> πάνω σε κάθε πλειάδα, **π.χ., HAVING AVG(Salary)>3000**
5. **ORDER BY:** Ταξινόμησε τα αποτελέσματα βάσει της συνθήκης

   π.χ., ORDER BY AVG(Salary)
6. **SELECT:** Πρόβαλε τα αιτούμενα γνωρίσματα (**από το TEMP2)** εφαρμόζοντας τα σχετικά aliases στα απλά γνωρίσματα.

   π.χ., SELECT DNO (τα γνωρίσματα του ORDER BY δεν χρειάζεται να εμφανίζονται στο SELECT)

Ομαδοποίηση

- **Επερώτηση: Βρες τον αριθμό υπαλλήλων και μέσο μισθό ανά τμήμα**

```
SELECT      DNO, COUNT (*), AVG (SALARY)
FROM        EMPLOYEE
GROUP BY    DNO
```

- **Εννοιολογική Εκτέλεση Επερώτησης:**
  - Η βάση εκτελεί μια γραμμική διέλευση του πίνακα Employee.
  - Για κάθε πλειάδα, βρίσκει το **DNO** της πλειάδας, βάσει του οποίου αυξάνει τους ακόλουθους μετρητές:
    A. **TUPLE_COUNT[DNO]**,
    B. **SALARY_SUM[DNO],**
    C. **SALARY_COUNT_NONULLS[DNO]**,
  - Στο τέλος τυπώνει για κάθε **DNO** τα ακόλουθα: **DNO, A, B/C**

Επισημάνσεις

- Η λίστα με τα γνωρίσματα που υπάρχουν στο SELECT ΠΡΕΠΕΙ να υπάρχει στο GROUP BY αλλά όχι το αντίθετο.
  - Π.χ., SELECT DNO, **GENDER**, AVG(SALARY)
    FROM E
    GROUP BY DNO ➔ **ΛΑΘΟΣ**
    - Γιατί; Διότι όπως είδαμε στην εκτέλεση ενός SQL Block το SELECT εκτελείται τελευταίο και το GENDER δεν θα υπάρχει στις μεταβλητές ομαδοποίησης
- Οι συναθροιστικές πράξεις μπορεί να χρησιμοποιηθούν για επιλογή (δηλ., HAVING) μόνο και να μην επιστραφούν στη λίστα του SELECT
- Εάν υπάρχει **NULL** στο **γνώρισμα ομαδοποίησης** τότε τα ομαδοποιημένα αποτελέσματα εμφανίζονται σε μια νέα επιπλέον ομάδα.
  - Π.χ., E(ID,CITY): (1,A), (2,B), (3,B), (4,NULL) – SELECT CITY, COUNT(ID) ➔ (NULL, 0), (A, 1), (B, 2)

Επισημάνσεις

- Συναθροιστικές Συναρτήσεις SUM, AVG (SUM/COUNT), MAX, MIN μπορούν να εφαρμοστούν σε σύνολα ή πολυσύνολα (σχέσεις με μοναδικές έγγραφες ή μη) αριθμητικών τιμών.
  - Οι **MAX, MIN** εφαρμόζονται και σε **αλφαριθμητικά δεδομένα** (π.χ., 'a' < 'b')
  - Το COUNT σε TSQL εφαρμόζεται στα πάντα εκτός από text, image, ή ntext.
- Οι συναθροιστικές συναρτήσεις ορίζονται πάντοτε στο όρο SELECT και στο HAVING
  - Π.χ., SELECT * FROM A WHERE salary=MAX(salary) ➔ ΛΑΘΟΣ
  - Θυμηθείτε ότι και στην ΣΑ δεν επιτρεπόταν να συγκρίνουμε τιμή με σύνολο

Επισημάνσεις (συν.)

- Οι συναθροιστικές συναρτήσεις **ΔΕΝ αγνοούν τα διπλότυπα**
  - Π.χ., r(R)=5, 3 μοναδικές πόλεις ➔ COUNT(CITY)=5
  - Για να αγνοηθούν τα διπλότυπα χρησιμοποιείται το DISTINCT
    - COUNT(DISTINCT CITY)=3
- Οι συναθροιστικές συναρτήσεις **αγνοούν τα NULL**
  - Π.χ., r(R)=5, 4 πόλεις, 1 null ➔ COUNT(CITY)=4
  - Πρέπει να είμαστε προσεκτικοί για τα αποτελέσματα που χρειαζόμαστε
- Ο **τελεστής *** δηλώνει πως θα ληφθούν υπόψη όλα τα γνωρίσματα στον υπολογισμό μιας συναθροιστικής συνάρτησης
  - Συνεπώς το **COUNT(*) μετράει γραμμές,** δηλ., συνδυασμό όλων των γνωρισμάτων που συμμετέχουν **(δεν παίζει ρόλο αν κάποια έχουν την τιμή NULL)**
  - **Δεν υποστηρίζεται το COUNT(DISTINCT *)**

Ασκήσεις

- Δώστε τα αποτελέσματα για όλα τα είδη JOIN στα πιο κάτω δεδομένα (P.CID=C.CID)

PRODUCTS (P)		
PID	**PNAME**	**CID**
1	PA	1
2	PB	1
3	PC	1
4	PD	2
5	PE	2
6	PF	3
7	PG	<NULL>

CATEGORIES (C)	
CID	**CNAME**
1	CA
2	CB
3	CC
4	CD

- Δώστε τα προϊόντα τα οποία έχουν το PID τους είναι το ίδιο με κάποιο CID από τις κατηγορίες

Ασκήσεις

E: EMPLOYEE

SSN	ENAME	DOB	GENDER	DNO	SALARY
1	Andreas	1/1/80	M	1	1000
2	Kostas	1/1/85	M	1	1500
3	Maria	4/3/83	F	1	2000
4	Eleni	6/7/85	F	2	2500
5	Victoria	27/2/84	F	3	3000

D: DEPARTMENT

DNO	DNAME
1	ACC
2	HR
3	IT
4	MNG

- Δώστε τον αριθμό των υπαλλήλων, μέγιστο και ελάχιστο μισθό
- Δώστε τον αριθμό των υπαλλήλων ανά τμήμα (για όλα τα τμήματα)
- Δώστε τον μέσο μισθό των υπαλλήλων ανά τμήμα και φύλο (για όλα τα τμήματα)
- Δώστε τον υπάλληλο με τον μέγιστο μισθό
- Δώστε τον υπάλληλο με τον δεύτερο μεγαλύτερο μισθό