



## **EPL342 –Databases**

# **Lecture 16: SQL DML III**

## **SQL Structured Query Language**

(Chapter 6.5.5-6.6, Elmasri-Navathe 7ED)

**Demetris Zeinalipour**

<http://www.cs.ucy.ac.cy/courses/EPL342>



# Περιεχόμενο Διάλεξης

## Ολοκλήρωση Διάλεξης 15.

### Κεφάλαιο 8.5.5-8.6: SQL DML III

- *Εντολή Μετονομασίας **AS** σε SQL*
- *Προχωρημένες Συνενώσεις σε SQL (**JOINS**)*
- *Συναθροιστικές Συναρτήσεις σε SQL (**COUNT, MAX, MIN, AVG, SUM**)*
- *Εντολή Ομαδοποίησης (**GROUP-BY**) και Εντολή Επιλογής-μετά-από-Ομαδοποίηση (**HAVING**) σε SQL*
- *Εντολές Εισαγωγής/Διαγραφής/Ενημέρωσης (**INSERT / DELETE / UPDATE**) σε SQL.*

# Εντολή Μετονομασίας AS σε SQL

- Για την μετονομασία **γνωρισμάτων** ή **σχέσεων** σε SQL χρησιμοποιείται ο όρος **AS**
- **Q8a:** Για τον κάθε υπάλληλο τύπωσε το όνομα του και το όνομα του προϊσταμένου του. **Οι στήλες του αποτελέσματος να φέρουν τα όνοματα Employee\_Name και Supervisor\_Name.**

```
SELECT E.Lname AS Employee_Name, S.Lname AS Supervisor_Name  
FROM EMPLOYEE AS E, EMPLOYEE AS S  
WHERE E.Super_ssn = S.ssn
```

– **Σημείωση:** Το AS εφαρμόζει αυτόματα από τον Query Builder του SQL Server.

## Δημιουργία Ενδιάμεσου Πίνακα

```
SELECT T.Lname, T.Lname  
FROM (SELECT * FROM EMPLOYEE E WHERE E.age=30) AS T  
WHERE T.Lname="Smith"
```

# Προχωρημένες Συνενώσεις σε SQL



- Νωρίτερα, μελετήσαμε την απλή εκδοχή της εντολής συνένωσης στην SQL (θ-join), π.χ.,  

```
SELECT *  
FROM Employee E, Department D  
WHERE E.Dno = D.Dno AND D.Dno=5;
```

(Χωρίς το *WHERE* εκφράζουμε το καρτεσιανό γινόμενο)
- Σήμερα θα δούμε πως μπορούμε να ορίσουμε τους ακόλουθες εξειδικευμένες συνενώσεις.

- **INNER JOIN (Απλή Συνένωση)**

- Εναλλακτικός τρόπος διατύπωσης της Απλής Συνένωσης.

- **NATURAL JOIN (Φυσική Συνένωση), [ Δεν υπάρχει σε TSQL ]**

- Η συνθήκη συνένωσης είναι το κοινά διατυπωμένο γνώρισμα.

- **OUTER JOIN (Εξωτερική Συνένωση: LEFT, RIGHT, FULL)**

- Συμπερίληψη πλειάδων που δεν συνενώνονται.

- **CROSS JOIN (Καρτεσιανό Γινόμενο)**

- Εναλλακτικός τρόπος διατύπωσης του Καρτεσιανού Γινομένου.

# Προχωρημένες Συνενώσεις σε SQL (INNER JOIN)



- Το **INNER JOIN** αποτελεί εναλλακτικό τρόπο διατύπωσης της Απλής Συνένωσης που είδαμε νωρίτερα. Π.χ.,

```
SELECT *
```

```
FROM Employee E INNER JOIN Department D ON E.Dno = D.Dno
```

```
WHERE D.Dno=5;
```

- **Πλεονέκτημα**

- Δεν πλέκεται η **συνθήκη συνένωσης** με την **συνθήκη της επερώτησης** (βολικό σε περιπτώσεις **πολλαπλών joins**)

- **Μειονέκτημα**

- **Διαφορετική υλοποίηση** σε διαφορετικές βάσεις δημιουργεί προβλήματα συμβατότητας ενώ η Απλή Συνένωση είναι ευρέως διαδεδομένη.

- **Άλλες Επισημάνσεις**

- Εάν δεν χρησιμοποιηθεί aliasing (π.χ., D, E) τότε μπορεί να προκύψουν σφάλματα μετάφρασης της επερώτησης σε διαφορετικές καταστάσεις
- Ο όρος **INNER** (στο **INNER JOIN**) σε ANSI-SQL είναι προαιρετικό.

# Προχωρημένες Συνενώσεις σε SQL (INNER JOIN)




- **Παράδειγμα:** Μετασχηματίστε το ακόλουθο σε ισοδύναμο query με χρήση Inner Join.

```
Q8:SELECT *  
      FROM EMPLOYEE E, EMPLOYEE S  
      WHERE E.SUPERSSN=S.SSN
```

- **Απάντηση:**

```
Q8:SELECT *  
      FROM (EMPLOYEE E INNER JOIN EMPLOYEE S  
            ON E.SUPERSSN=S.SSN)
```

# Προχωρημένες Συνενώσεις σε SQL (*NATURAL JOIN* – **ΌΧΙ** σε *TSQL*)



- Το **NATURAL JOIN** (σε ANSI-SQL, π.χ., υλοποιείται σε PostgreSQL) υλοποιεί την συνένωση με τέτοιο τρόπο ώστε η **συνθήκη συνένωσης** είναι το κοινά διατυπωμένο γνώρισμα., π.χ.,  
SELECT \* FROM Employee E **NATURAL JOIN** Department D  
WHERE D.Dno=5;
- **Επισημάνσεις**
  - Δεν απαιτεί τον ορισμό της κοινής στήλης ούτε την παράγει διπλά.
  - Εάν δεν υπάρχει κοινά διατυπωμένο γνώρισμα μπορεί να προηγηθεί μετονομασία:
    - Π.χ., **SELECT \* FROM EMPLOYEE NATURAL JOIN Department AS Dept(Dname, Dno, Mssn, Msdate)** (μετονομ. δεν δουλεύει σε TSQL)
  - Γενικότερα, το NATURAL JOIN δεν υλοποιείται σε TSQL για αυτό δεν θα μελετηθεί περαιτέρω.

# Προχωρημένες Συνενώσεις σε SQL (NATURAL JOIN – **Όχι σε TSQL**)

- **Παράδειγμα:** Μετασχηματίστε το ακόλουθο σε ισοδύναμο query με χρήση Natural Join.

- **Παράδειγμα :**

```
Q1:SELECT      FNAME, LNAME, ADDRESS
      FROM      EMPLOYEE, DEPARTMENT
      WHERE     DNAME='Research' AND DNUMBER=DNO
```

- **Μπορούσε να γραφεί με NATURAL JOIN ως:**

```
Q1:SELECT      FNAME, LNAME, ADDRESS
      FROM      (EMPLOYEE NATURAL JOIN DEPARTMENT
      AS DEPT(DNAME, DNO, MSSN, MSDATE))
      WHERE     DNAME='Research'
```



# Προχωρημένες Συνενώσεις σε SQL (OUTER JOIN)



- Σε μια εσωτερική συνένωση (Απλή, Inner, Natural), το αποτέλεσμα περιλαμβάνει **MONO** πλειάδες που έχουν ίσο γνώρισμα συνένωσης
- Η **εξωτερική συνένωση** (OUTER JOIN) είναι χρήσιμη για παραγωγή αποτελεσμάτων που θέλουν στο αποτέλεσμα όλες τις εγγραφές μιας σχέσης (αριστερής, δεξιάς ή και τις δυο) ανεξάρτητα εάν **συνενώνονται**

**<Table1> {LEFT|RIGHT|FULL} [OUTER] JOIN <Table2> ON <condition>**

EMPLOYEE E

ssn	Fname	Minit	Lname	Dno
1	John	B	Smith	2
2	Franklin	T	Wong	5
3	Alicia	J	Wong	2
4	Jennifer	S	Zelaya	4
5	Ramesh	K	Wallace	2
6	Joyce	A	Narayan	2
7	Ahmad	V	English	2
8	James	E	Jabbar	2

DEPARTMENT D

Dno	Dname
1	Education
4	IT
5	Research

**E LEFT [OUTER] JOIN D**  
ή **WHERE E \*= D (SQL Server 2000)**

ssn	Fname	Minit	Lname	Dno	Expr1	Dname
1	John	B	Smith	2	NULL	NULL
2	Franklin	T	Wong	5	5	Research
3	Alicia	J	Wong	2	NULL	NULL
4	Jennif...	S	Zelaya	4	4	IT
5	Rame...	K	Wallace	2	NULL	NULL
6	Joyce	A	Narayan	2	NULL	NULL
7	Ahma...	V	English	2	NULL	NULL
8	James...	E	Jabbar	2	NULL	NULL

**Σημείωση:** Θεωρήστε ότι δεν υπάρχει ο αναφορικός περιορισμός  $E.DNO \Rightarrow D.DNO$

# Προχωρημένες Συνενώσεις σε SQL (OUTER JOIN)



EMPLOYEE E

ssn	Fname	Minit	Lname	Dno
1	John	B	Smith	2
2	Franklin	T	Wong	5
3	Alicia	J	Wong	2
4	Jennifer	S	Zelaya	4
5	Ramesh	K	Wallace	2
6	Joyce	A	Narayan	2
7	Ahmad	V	English	2
8	James	E	Jabbar	2

DEPARTMENT D

Dno	Dname
1	Education
4	IT
5	Research

## E FULL [OUTER] JOIN D

WHERE E **\*=**\* D (SQL Server 2000)

ssn	Fname	Minit	Lname	Dno	Expr1	Dname
1	John	B	Smith	2	NULL	NULL
2	Franklin	T	Wong	5	5	Research
3	Alicia	J	Wong	2	NULL	NULL
4	Jennif...	S	Zelaya	4	4	IT
5	Rame...	K	Wallace	2	NULL	NULL
6	Joyce	A	Narayan	2	NULL	NULL
7	Ahma...	V	English	2	NULL	NULL
8	James...	E	Jabbar	2	NULL	NULL
NULL	NULL	NULL	NULL	NULL	1	Education

## E RIGHT [OUTER] JOIN D

WHERE E **=**\* D (SQL Server 2000)

ssn	Fname	Minit	Lname	Dno	Expr1	Dname
NULL	NULL	NULL	NULL	NULL	1	Education
4	Jennif...	S	Zelaya	4	4	IT
2	Franklin	T	Wong	5	5	Research

# Προχωρημένες Συνενώσεις σε SQL (OUTER JOIN)



- **SQL Server 2005/2008 (ANSI τρόπος)**

- **Left Outer Join:** Μετονομασία γίνεται για να μην έχουμε διπλά Dno στο αποτέλεσμα.

```
SELECT E.ssn, E.Fname, E.Minit, E.Lname, E.Dno,  
       D.Dno AS Expr1, D.Dname  
FROM   Emp AS E LEFT OUTER JOIN Dep AS D ON E.Dno = D.Dno
```

- **Right Outer Join:**

```
SELECT E.ssn, E.Fname, E.Minit, E.Lname, E.Dno,  
       D.Dno AS Expr1, D.Dname  
FROM   Emp AS E RIGHT OUTER JOIN Dep AS D ON E.Dno = D.Dno
```

- **Full Outer Join:**

```
SELECT E.ssn, E.Fname, E.Minit, E.Lname, E.Dno,  
       D.Dno AS Expr1, D.Dname  
FROM   Emp AS E FULL OUTER JOIN Dep AS D ON E.Dno = D.Dno
```

# Προχωρημένες Συνενώσεις σε SQL (OUTER JOIN)



- **SQL Server 2000 (Non-ANSI τρόπος)**

- **Left Outer Join:**

```
SELECT *  
FROM Emp E, Dep D  
WHERE E.Dno *= D.Dno
```

- **Right Outer Join:**

```
SELECT *  
FROM Emp E, Dep D  
WHERE E.Dno =* D.Dno
```

- **Full Outer Join:**

```
SELECT *  
FROM Emp E, Dep D  
WHERE E.Dno *=* D.Dno
```

# Προχωρημένες Συνενώσεις σε SQL (CROSS JOIN)



- Το **CROSS JOIN** αποτελεί εναλλακτικό τρόπο διατύπωσης του Καρτεσιανού Γινομένου το οποίο έχουμε δει ήδη, δηλ.  
`SELECT * FROM Employee E, Department D`

`<table_source> CROSS JOIN <table_source>`

- **Γενικές Επισημάνσεις**

- Όλοι οι τρόποι συνένωσης που είδαμε μπορούν να υποστηρίξουν περισσότερες από 2 σχέσεις.
  - » π.χ., ((Table JOIN TableB ON <cond>) JOIN TableC ON <cond>)
- Είναι καλό να χρησιμοποιούμε την **ANSI SQL** μέθοδο σε SQL συνενώσεις για λόγους συμβατότητας μεταξύ κατασκευαστών.
- Στις εξωτερικές συνενώσεις (LEFT και RIGHT) **ΔΕΝ ισχύει η αντιμεταθετική ιδιότητα** (δηλ., η σειρά των joins δεν μπορεί να αλλάξει αυθαίρετα), δηλ.,  **$(A \otimes B) \neq (B \otimes A)$** 
  - » Δώστε έμφαση στη σειρά των συνενώσεων διότι αυτό μπορεί να σας οδηγήσει εύκολα σε λάθη.

# Συναθροιστικές Συναρτήσεις (Aggregate Functions)



- Οι **Συναθροιστικές Συναρτήσεις (Aggregate Functions)** προσδιορίζουν μαθηματικές πράξεις πάνω σε συλλογές τιμών της βάσης συμπεριλαμβανομένων των ακόλουθων: **COUNT, SUM, MAX, MIN, και AVG**
- **Query 15:** Βρες το **μέγιστο salary**, το **ελάχιστο salary**, και τον **μέσο όρο** των salaries από όλους τους employees.
- Q15: 

```
SELECT MAX(SALARY), MIN(SALARY),  
        AVG(SALARY)  
FROM EMPLOYEE
```

Σημειώστε ότι δεν λαμβάνονται υπόψη τα NULLS!

# Συναθροιστικές Συναρτήσεις (Aggregate Functions)



- **Query 16:** Βρες το **μέγιστο salary**, το **ελάχιστο salary**, και τον **μέσο όρο** των salaries μεταξύ όλων των employees που δουλεύουν για το 'Research' department.

```
Q16:  SELECT  MAX(SALARY),  
        MIN(SALARY), AVG(SALARY)  
FROM    EMPLOYEE, DEPARTMENT  
WHERE   DNO=DNUMBER AND  
        DNAME='Research'
```

**Σημείωση:** Η εκτέλεσης της συνένωση προηγείται της συναθροιστικής συνάρτησης

# Συναθροιστικές Συναρτήσεις (Aggregate Functions)



- **Επισημάνσεις**

- Συναθροιστικές Συναρτήσεις SUM, AVG (δηλ., 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., «Δίας» < «Φεγγάρι»)
  - Το **COUNT** σε TSQL εφαρμόζεται στα πάντα εκτός από **text, image, ή ntext**.
- Οι Συναθροιστικές συναρτήσεις ορίζονται **πάντοτε** στο όρο **SELECT** (και στο **HAVING** που θα δούμε αργότερα)
  - ~~**ΛΑΘΟΣ: SELECT \* FROM A WHERE salary=MAX(salary)**~~



# Συναθροιστικές Συναρτήσεις (Aggregate Functions)



- Ο **τελεστής \*** (δηλ., **func(\*)**) υποδηλώνει πως πρέπει να ληφθούν υπόψη όλα τα γνωρίσματα στον υπολογισμό μιας συναθροιστικής συνάρτησης
- **Αριθμός των employees** (includes NULLs & duplicates)  
Q17:           SELECT           **COUNT (\*)**  
                  FROM           EMPLOYEE
- **Αριθμός employees του Research Department**  
Q18:           SELECT           **COUNT (\*)**  
                  FROM           EMPLOYEE, DEPARTMENT  
                  WHERE          DNO=DNUMBER AND  
                                  DNAME='Research'

# Συναθροιστικές Συναρτήσεις (Aggregate Functions)



- **Μετρούν Γραμμές (είτε περιέχουν NULL είτε όχι)\***
  - **COUNT(\*)**: Βρίσκει αριθμό γραμμών ενός πίνακα (περιλαμβάνει και NULLs) και διπλότυπα

- **Αγνοούν τα NULL\***

**I. COUNT(expression)**: Μέτρα τις εμφανίσεις του *expression* (μετρώντας ξανά τα διπλότυπα)

- Παράδειγμα:  $R(A,B)=\{(1,1), (2,2), (2, NULL), (5, NULL)\}$
- `SELECT COUNT(A) FROM R` returns 4
- `SELECT COUNT(B) FROM R` returns 2
- `SELECT COUNT(CAST(A as char) + CAST(B as char)) FROM R;`

The above returns 2

**II. COUNT(ALL expression)**: Ίδιο με το **COUNT(expression)**

\* **ANSI\_NULLS** does *not* have any effect on COUNT, as it deals with whether a NULL is **distinct** (TSQL) or not (ANSI)

# Συναθροιστικές Συναρτήσεις (Aggregate Functions)



**III. COUNT(DISTINCT expression):** Μετρά μοναδικές εμφανίσεις του *attribute*.

- Παράδειγμα:  $R(A,B)=\{(1,1), (2,2), (2, \text{NULL}), (5, \text{NULL})\}$
- `SELECT COUNT(DISTINCT A) FROM R -- returns 3`
- `SELECT COUNT(DISTINCT B) FROM R -- returns 2`
- `SELECT COUNT(DISTINCT CAST(A as char) + CAST(B as char)) FROM R; -- The above returns 2 (δηλ., ότι περιέχει έστω ένα NULL αγνοείται)`

**IV. COUNT (DISTINCT \*):** Δεν εφαρμόζεται σε TSQL  
(δες επόμενη διαφάνεια για λύση)

- **Επισήμανση**

- Το **COUNT** παίρνει τιμές μέχρι  $2^{31}-1$ . Όταν υπάρχει ανάγκη για μεγαλύτερες τιμές τότε υπάρχει και το **COUNT\_BIG**

# Συναθροιστικές Συναρτήσεις (Aggregate Functions)



- Αντίστοιχο της «COUNT (DISTINCT \*)»

```
SELECT COUNT(*)
```

```
FROM (SELECT DISTINCT * FROM table)
```

```
AS Temp
```

– Σημείωση: Εδω το *DISTINCT \** χρειάζεται μόνο εάν το *table* ΔΕΝ έχει *Primary Key*.

– Εάν έχει *PRIMARY key* το *table* τότε εξυπακούεται, δηλ., είναι απλά

```
SELECT COUNT(*) FROM table.
```

# Συναθροιστικές Συναρτήσεις

## ~~COUNT(DISTINCT \*)~~



**SELECT \* FROM Emp**

ssn	Fname	Minit	Lname	Dno
1	John	B	Smith	2
2	Franklin	T	Wong	5
3	Alicia	J	Wong	2
4	Jennifer	NULL	Zelaya	4
5	Ramesh	NULL	Wallace	2
6	Joyce	A	Naray...	2
7	Ahmad	T	Wong	2
8	James	E	Jabbar	2
1	John	B	Smith	2

**SELECT DISTINCT \* FROM Emp**

ssn	Fname	Minit	Lname	Dno
1	John	B	Smith	2
2	Franklin	T	Wong	5
3	Alicia	J	Wong	2
4	Jennifer	NULL	Zelaya	4
5	Ramesh	NULL	Wallace	2
6	Joyce	A	Naray...	2
7	Ahmad	T	Wong	2
8	James	E	Jabbar	2

Εναλλακτική διατύπωση: ~~SELECT COUNT(DISTINCT \*)~~

**SELECT COUNT(\*) AS CountVar  
FROM (  
    SELECT DISTINCT \*  
    FROM Emp) AS TEMP**

CountVar
8

# Συναθροιστικές Συναρτήσεις (Aggregate Functions)



- Πέρα από το **COUNT (DISTINCT \*)**, ούτε και το **COUNT (DISTINCT Attr1,Attr2)** **ΔΕΝ** εφαρμόζεται σε αρκετές εκδοχές της SQL (π.χ., TSQL).
- Για αυτό μπορούμε να εφαρμόσουμε την ακόλουθη εναλλακτική προσέγγιση (με εμφωλευμένη επερώτηση):
- Διατύπωση «COUNT (DISTINCT Attr1,Attr2)»
  - Εκδοχή A με CONCAT (όπως πριν):  
SELECT COUNT(DISTINCT CAST(A as char) +  
CAST(B as char)) FROM table;
  - Εκδοχή B με ενδιάμεσο πίνακα:  
SELECT COUNT(\*)  
FROM (SELECT DISTINCT Attr1, Attr2 FROM table)

# Παράδειγμα Counting



```
SELECT COUNT(*),  
       COUNT(Field1),  
       COUNT(Field2),  
       COUNT(DISTINCT Field3)  
FROM Table
```

**Αποτέλεσμα:** {(4, 3, 2, 1)}

## Επεξήγηση:

**COUNT(\*) => 4;** -- count all rows, even null/duplicates  
-- count only rows without null values on that field

**COUNT(Field1) = COUNT(Field2) => 3**

**COUNT(Field3) => 2**

**COUNT(DISTINCT Field3) => 1** -- Ignore duplicates

[ANSI\\_NULLS](#) does *not* have any effect on COUNT as it mainly deals with whether a NULL is distinct (TSQL) or not (ANSI)

Field1	Field2	Field3
1	1	1
NULL	NULL	NULL
2	2	NULL
1	3	1