



**EPL342 –Databases
Lecture 20: External DB
Programming II**

**Internal Programming & External
Programming**

(Chapter 10.2-10.4, Elmasri-Navathe 7ED +
TransactSQL Reference Guide

Demetris Zeinalipour



Περιεχόμενο Διάλεξης

Εσωτερικός Προγραμματισμός DB II

- **Scripts/Batches σε TSQL**
- Stored Procedures (Sprocs) σε TSQL
 - SQL Injection Attacks – How to prevent?
- User Defined Functions (UDFs) σε TSQL
- Cursors σε TSQL
- Transactions in TSQL
- Indexes in TSQL
- Temporary Tables #, ##, CTE, Recursive CTE,
- Passwords in Databases
- Database Catalog
- Unit Testing

Εξωτερικός Προγραμματισμός DB:

- Bridges: ODBC, JDBC, SqlSrv,
- HTML, HTTP, Web Browser (εργαστήριο)

SQL/PSM - Persistent Stored Modules



- Στο πρότυπο ANSI, τα **Sprocs** και **UDFs** ορίζονται ως μέρος της ενότητας **SQL/PSM (SQL/Persistent Stored Modules)**
 - Το **Persistent** (δηλ., μόνιμα) υποδηλώνει ότι οι λειτουργίες αυτές **αποθηκεύονται** και **εκτελούνται** από τον διαθέτη της **βάσης** παρά τον **πελάτη**.
- **Βασικό Πρόβλημα** είναι το γεγονός ότι ο κάθε **κατασκευαστής** ακολουθεί δική του σύνταξη για υλοποίηση του **PSM**
 - Η **TSQL** είναι ένα παράδειγμα μη-συμβατής γλώσσας.
 - Το ίδιο ισχύει για την **PL/SQL** (Procedural Language/SQL) της Oracle

B) Stored Procedures σε TSQL

- **Stored Procedures (Sprocs)**, είναι εντολές (T)SQL οι οποίες αποθηκεύονται στη βάση δεδομένων και οι οποίες μπορούν να πάρουν ορίσματα και να επιστρέψουν τιμές.
- **Παράδειγμα:** Δημιουργία Sproc το οποίο επιστρέφει όλους τους Employee

USE ep1342

GO

CREATE PROC spEmployee

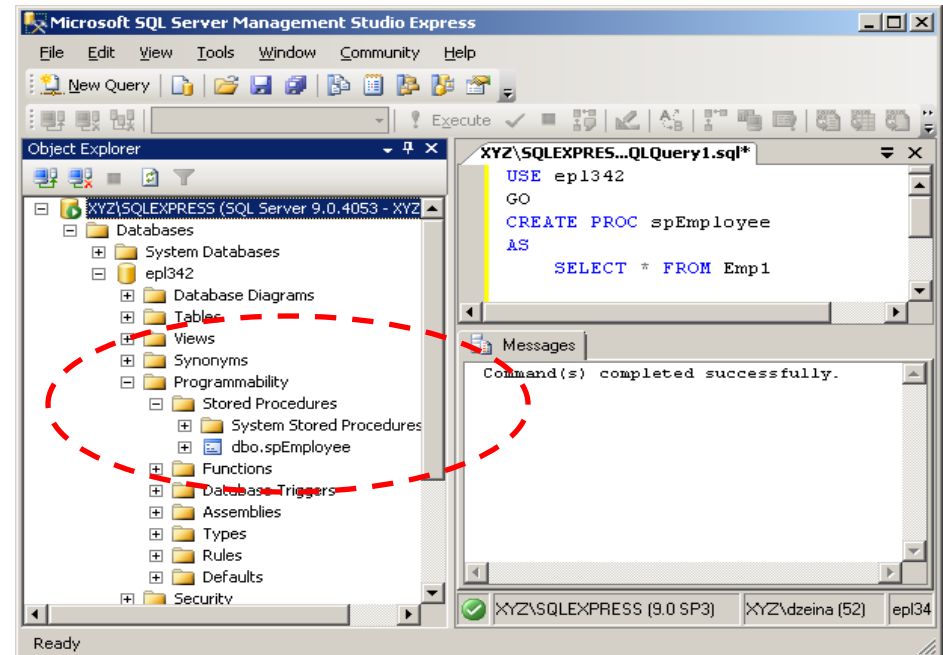
AS

SELECT * FROM Emp1

GO

GO is not a SQL keyword.

It's a batch separator used by client tools (like SSMS) to break the entire script up into batches



B) Stored Procedures σε TSQL (Sproc χωρίς Παραμέτρους)



- Κλήση Sproc (χωρίς Παραμέτρους) μέσω EXEC

USE epl342

GO

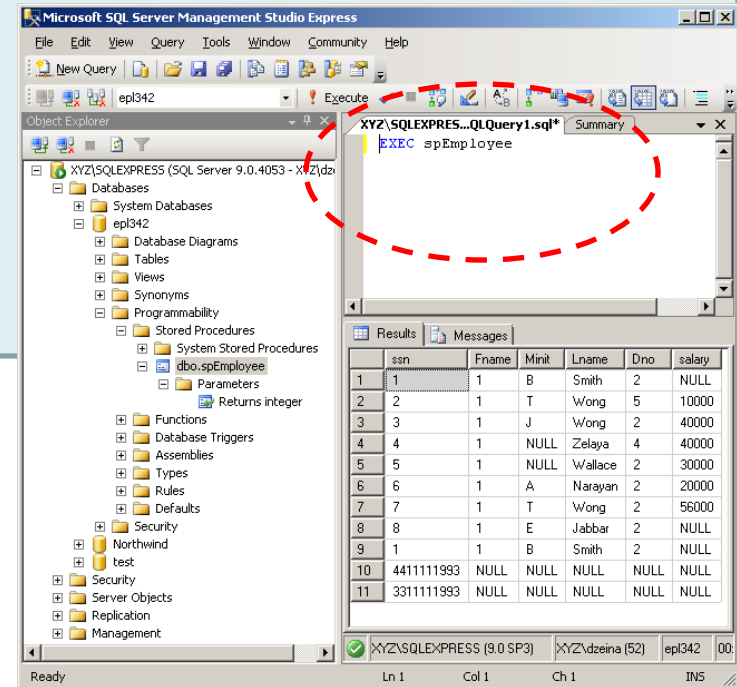
EXEC spEmployee

Άλλες Εντολές

- Ακύρωση SProc
 - DROP PROC spEmployee [;]
- Μεταβολή Sproc
 - ALTER PROC

AS SELECT * FROM Emp1

GO

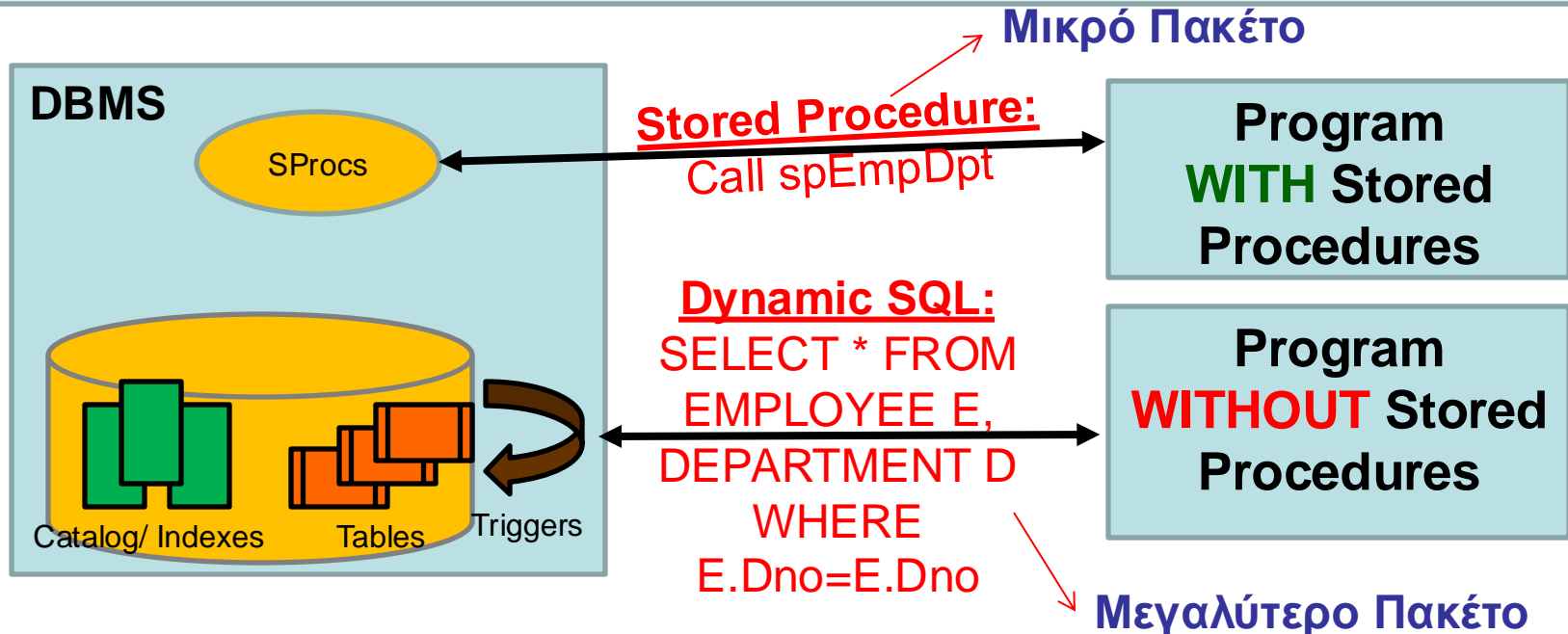


B) Stored Procedures σε TSQL (Χαρακτηριστικά Sprocs)



Βασικό Πλεονέκτημα Sprocs:

- **Επίδοση/Μειωμένη Κίνηση Δικτύου:** Το **Sproc** είναι **Precompiled/Optimized** πριν την κλήση του. Επίσης **μειώνεται η κυκλοφορία δεδομένων** μεταξύ DBMS και προγράμματος εφαρμογής.



B) Stored Procedures σε TSQL (Χαρακτηριστικά Sprocs)



- **Άλλα Πλεονεκτήματα Sprocs**

- **Ασφάλεια:** Τα Sprocs μας επιτρέπουν να δώσουμε **πρόσβαση** σε **λειτουργίες** της βάσης **χωρίς** να δίνουμε πρόσβαση στους **πίνακες**.
- **Ακρίβεια:** Οι διεπαφές (ODBC, JDBC, κτλ) στη προσπάθεια να προσφέρουν ένα κοινό υπόβαθρο λειτουργίας σε διαφορετικούς κατασκευαστές βάσεων κάνουν διάφορους συμβιβασμούς στην ακρίβεια (π.χ., μετατροπή τύπων κτλ)
- **Εύκολη μεταφορά σε άλλη Γλώσσα Προγραμματισμού:** Αυτό εφόσον όλη η λογική της βάσης υλοποιείται εσωτερικά και όχι στην γλώσσα προγραμματισμού.
- Μειώνουμε τον κίνδυνο για SQL Injection επιθέσεις.
- Μειώνουμε και την έκθεση του πηγαίου μας κώδικα προς τα έξω.

- **Μειονεκτήματα Sprocs**

- **Δύσκολη Μεταφορά σε άλλο DBMS:** Αυτό λόγω του ότι οι γλώσσες που χρησιμοποιούνται σε Προγ. Γλώσσες Βάσεων Δεδομένων (π.χ., σύνταξη της TSQL) δεν είναι προτυποποιημένες

B) Stored Procedures σε TSQL (Sproc με Παράμετρο Εισόδου)



- Παράδειγμα Sproc: Βρες όλα τα name, dependent name βάσει κάποιας συμβολοσειράς αναζήτησης

```
USE ep1342
```

```
GO
```

```
CREATE PROC spEmpDep  
    @SearchName nvarchar(50)
```

```
AS
```

```
SELECT E.fname, D.dname  
FROM Dependent D, Emp1 E  
WHERE D.ssn=E.ssn AND  
    D.Dname LIKE '%' + @ SearchName + '%';
```

Παράμετρος Εισόδου
(max αριθμός = 2100)

EMPLOYEE

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

ssn	dname
1	Natali
1	Stefanos
2	Achilleas
2	Prokopis
3	Irini

Κλήση Sproc: **EXEC spEmpDep 'Ach'**

➔ Επιστρέφει Franklin, Achilleas

DEPENDENT

B) Stored Procedures σε TSQL

(Sproc με Παράμετρο Εισόδου/Εξόδου)



- Παράδειγμα Stored Procedure που λαμβάνει ως είσοδο ένα CustomerID και ελέγχει το ιστορικό αγορών του εν λόγω πελάτη.

```
CREATE PROCEDURE sp_CustomerLevel
@CustomerID INTEGER,
@CustomerLevel VARCHAR(20) OUT
AS
DECLARE @PurchaseTotal DECIMAL(8,2)
SET @PurchaseTotal = (SELECT SUM(amount) FROM transactions tr
WHERE tr.CustomerID = @CustomerID)
IF @PurchaseTotal = 0
BEGIN
SET @CustomerLevel = 'Empty'
RETURN
END
IF @PurchaseTotal > 1000
SET @CustomerLevel = 'Standard'
ELSE
SET @CustomerLevel = 'Gold'
```

Μεταβλητή εξόδου

```
Κλήση Sproc sp_CustomerLevel
DECLARE @CustomerLevel VARCHAR(20)
EXEC sp_CustomerLevel 12391, @CustomerLevel OUT
PRINT @CustomerLevel
→ Επιστρέφει Empty
```

SQL Injection Attacks

Τι είναι;



- Είναι η περίπτωση που ένας κακόβουλος χρήστης διεξάγει ανεπίτρεπτες πράξεις στην κατάσταση της βάσης μας
 - Π.χ., εισαγωγή/διαγραφή/ενημέρωση δεδομένων ή πινάκων SELECT/INSERT/UPDATE/CREATE, εκτέλεση ρουτίνων της βάσης EXEC για π.χ., άνοιγμα πορτών, εμφύτευση κακόβουλου κώδικα, κτλ.
 - Το 2018-2019 η Fortite, Tesla και Cisco ήταν θύματα.
- Αφορά περιπτώσεις Δυναμικής SQL

```
SELECT E.fname, D.dname  
FROM Dependent D, Emp1 E  
WHERE D.ssn=E.ssn AND  
D.Dname LIKE '%' + @ SearchName + '%';
```

Παράδειγμα



- Υποθέστε τον ακόλουθο C# κώδικα (μπορούσε να είναι PHP, Java, κτλ.)

```
var Shipcity;
```

```
ShipCity = Request.form ("ShipCity");
```

```
var sql = "select * from OrdersTable where ShipCity = '" +  
ShipCity + "'";
```

- Ο χρήστης δίνει στο input

```
Redmond'; drop table OrdersTable--
```

- Τι εκτελείται;

```
SELECT * FROM OrdersTable WHERE ShipCity =  
'Redmond'; drop table OrdersTable--'
```



SQL Injection Attacks

Τύποι Επιθέσεων



- Union-based SQL Injection
 - Επικόλληση στο result-set επιπλέον αποτελεσμάτων με UNION
- Error-Based SQL Injection
 - Εκτύπωση επιπλέον αποτελεσμάτων με PRINT στο standard error
- SQL injection based on user input
 - Επικόλληση επιπλέον statements π.χ., DROP, DELETE, κτλ ως μέρος του query
-

SQL Injection Attacks

Πως αποτρέπεται;



- Είναι περίπλοκο θέμα σε πολλά επίπεδα το οποίο θα αγγίξουμε μόνο επιδερμικά
 - Φιλτράρισμα Εισόδου & Blacklisting (π.χ., σε χαρακτήρες “<>/?*()&”, SELECT, EXEC, INSERT, file name handlers: AUX, CLOCK\$, COM1 through COM8, CON, CONFIG\$, LPT1 through LPT8, NUL, and PRN.κτλ)
 - Ελάχιστα δικαιώματα στον χρήστη
 - Χρήση Stored Procedures (όχι Dynamic SQL)
 - Χρήση εξειδικευμένων σουιτών ασφάλειας και εργαλείων ελέγχου (penetration testing)
 - Περισσότερα: <https://learn.microsoft.com/en-us/sql/relational-databases/security/sql-injection?view=sql-server-ver16>

SQL Injection Attacks

QUOTENAME, REPLACE



- **SELECT QUOTENAME** ('character_string' [, 'quote_character'])
 - Περιορίζει την είσοδο σε 127 χαρακτήρες (αποτρέπει buffer overflow attacks through SQL injection - see example on prior slide)
 - μεγαλύτερα strings εισόδου επιστρέφουν NULL
 - Προσθέτει διαχωριστικά έτσι ώστε η συμβολοσειρά εισόδου να είναι δόκιμη
 - single quotation mark ('), a left or right bracket ([]), a double quotation mark ("), a left or right parenthesis (()), a greater than or less than sign (><), a left or right brace ({ }) or a backtick (`).
 - Τα left or right bracket ([]) είναι default
 - **SELECT QUOTENAME**('abc def'); Δίνει [abc def] (1 row(s) affected)
- **SELECT REPLACE**('abcdefghicde','cde','xxx');
 - Αντικαθιστά χαρακτήρες σε μια ακολουθία εισόδου

Sanitizing Input in Stored Procedures



```
ALTER PROCEDURE dbo.My_Stored_Procedure
@String_Parameter VARCHAR(289)
AS
BEGIN
SELECT @String_Parameter = REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPL
ACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLAC
E(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLAC
E(REPLACE(REPLACE(REPLACE(REPLACE(@String_Parameter, '@', ''), '#', ''), '%', ''), '!', ''), '$', ''), '^', ''), '
&', ''), '*', ''), '(', ''), ')', ''), '_', ''), '-', ''), '=', ''), '+', ''), '[', ''), ']', ''), '{', ''), '}', ''), '\', ''), '|', ''), ';', ''), ':', ''), '""',
"), '"', ''), ', , '), '<', ''), ':', ''), '>', ''), '/', ''), '?', ''), '^', ''), '~', ''), ' ', ' ');
SELECT @String_Parameter;
END
```

Daisy chaining replace is messy but works (as opposed to regex solution):

<https://www.sqlshack.com/sanitizing-inputs-avoiding-security-usability-disasters/>

Παράδειγμα Αλλαγής Password



```
CREATE PROCEDURE sp_MySetPassword
    @loginname sysname,
    @old sysname,
    @new sysname
AS

-- Declare variables.
DECLARE @login sysname
DECLARE @newpassword sysname
DECLARE @oldpassword sysname
DECLARE @command varchar(2000)

-- In the following statements, the data stored in temp variables
-- will be truncated because the buffer size of @login, @oldpassword,
-- and @newpassword is only 128 characters, but QUOTENAME() can return
-- up to 258 characters.
SET @login = QUOTENAME(@loginname, '''')
SET @oldpassword = QUOTENAME(@old, '''')
SET @newpassword = QUOTENAME(@new, '''')

-- Construct the dynamic Transact-SQL.
-- If @new contains 128 characters, then @newpassword will be '123... n
-- where n is the 127th character.
-- Because the string returned by QUOTENAME() will be truncated,
-- it can be made to look like the following statement:
-- UPDATE Users SET password = '1234. . .[127] WHERE username=' -- other stuff f
SET @command = 'UPDATE Users set password = ' + @newpassword
    + ' where username = ' + @login + ' AND password = ' + @oldpassword;

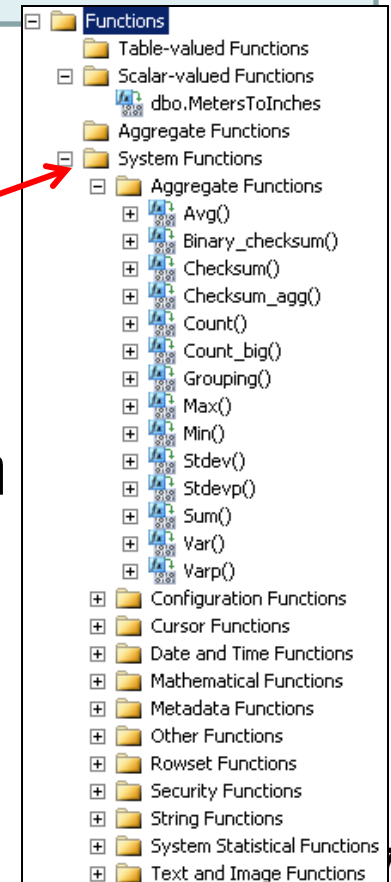
-- Execute the command.
EXEC (@command);
GO
```


C) UDFs σε TSQL



- **UDF (User Defined Functions):** συναρτήσεις του χρήστη οι οποίες αποτελούνται από εντολές (T)SQL και οι οποίες αποθηκεύονται στη βάση δεδομένων.

- Σημειώστε ότι κάθε βάση δεδομένων παρέχει και built-in συναρτήσεις οι οποίες ονομάζονται **System Functions** (π.χ., MAX, MIN, ABS, κτλ)
- Τα UDFs (TSQL) διακρίνονται στις κατηγορίες:
 - **Scalar-valued Functions** → Επιστρέφουν Βαθμωτή Τιμή (Scalar: int, varchar, κτλ.)
 - **Table-Valued Functions** → Επιστρέφουν Πίνακα
 - **Aggregate Functions** → Επιστρέφουν κάποιο συναθροιστικό αποτέλεσμα (π.χ., μια εξειδικευμένη MAX συνάρτηση)



C) UDFs σε TSQL (Παράδειγμα I)



- Παράδειγμα Δημιουργίας UDF (Scalar) που μετατρέπει τα **Μέτρα** σε **Ίντσες**

```
CREATE FUNCTION dbo.MetersToInches (@Meters DECIMAL(10,3))  
-- Παράμετρος Εισόδου (δια τιμής)
```

```
RETURNS DECIMAL(10,3) -- Τιμή Επιστροφής (δεν μπορούμε να  
περάσουμε τιμές δια αναφοράς ☹)
```

```
AS
```

```
BEGIN
```

```
DECLARE @Inches DECIMAL(10,3)
```

```
SET @Inches = (@Meters * 3.281 ) * 12
```

```
RETURN @Inches -- Επιστροφή Αποτελέσματος
```

```
END
```

Γενικά, ισχύουν οι γνωστοί κανόνες συναρτήσεων που υπάρχουν σε άλλες γλώσσες (π.χ., εμβέλεια μεταβλητών, κτλ)

C) UDFs σε TSQL (Παράδειγμα I)



- **Κλήση UDF (Scalar):**

```
SELECT dbo.MetersToInches(123.45) AS 'Inches'
```

Επιστρέφει:

Inches

4860.473

(1 row(s) affected)

Άλλο Παράδειγμα:

```
SELECT Weight, dbo.MetersToInches(Length), Cost, ...  
FROM Products  
WHERE ProductID = 199232
```

C) UDFs σε TSQL (Παράδειγμα II)



- **UDF χωρίς παράμετρο**

```
CREATE FUNCTION dbo.Two ()  
RETURNS INT -- Τιμή Επιστροφής  
AS  
    BEGIN  
        RETURN 2 -- Επιστροφή Αποτελέσματος  
    END
```

Query

```
SELECT *  
FROM EMP1  
WHERE ssn = dbo.Two()
```

*Κλήση UDF
στο WHERE*



Αποτέλεσμα Εκτέλεσης Query

ssn	Fname	Minit	Lname	Dno	salary
2	Franklin	T	Wong	5	10000

(1 row(s) affected)

C) UDFs σε TSQL

(UDF Table-Valued Function)



```
CREATE FUNCTION dbo.GetEmployees()
```

```
RETURNS @records TABLE -- Δήλωση ότι η Συνάρτηση Επιστρέφει Πίνακα
```

```
( -- Πεδία Σχέσης Επιστροφής
```

```
  EmpID nchar(10) NOT NULL,
```

```
  FirstName nchar(10) NULL
```

```
)
```

```
AS
```

```
BEGIN
```

```
-- Εισαγωγή Όλων των αποτελεσμάτων της σχέσης Emp1 στη σχέση @records
```

```
INSERT INTO @records
```

```
  SELECT ssn,fname FROM Emp1
```

```
-- Τερματισμός της συνάρτησης (το records επιστρέφεται έτσι και αλλιώς)
```

```
RETURN;
```

```
END
```

Κλήση UDF (Table-Valued) στο FROM

Select *

FROM dbo.GetEmployees()

Σημείωση: Το αποτέλεσμα της συνάρτησης μπορεί να χρησιμοποιηθεί στο FROM

C) UDFs σε TSQL (Διαφορές με Sprocs)



- Τα **UDFs** και στα **Sprocs** έχουν περισσότερες ομοιότητες παρά διαφορές στην TSQL.
- **Βασικές διαφορές:**
 - **A) Στο πως γίνεται η κλήση:** Στα **UDFs** η κλήση συνήθως στο **SELECT** (και κάποτε στο **WHERE** ή **FROM**) ενώ στα **SPROC** η κλήση γίνεται μέσω **EXEC**.
 - **B) Στο πως επιστρέφονται τα αποτελέσματα:** Στα **UDFs** τα αποτελέσματα μπορούν να χρησιμοποιηθούν άμεσα ενώ στα **SPROC** πρέπει να τοποθετηθούν σε **ενδιάμεσους πίνακες**
Π.χ., Για χρήση αποτελεσμάτων της spEmployee σε ένα query τα εισάγω πρώτα σε ένα ενδιάμεσο πίνακα EmpBack

```
INSERT INTO EmpBack
```

```
EXEC spEmployee
```

C) UDFs σε TSQL



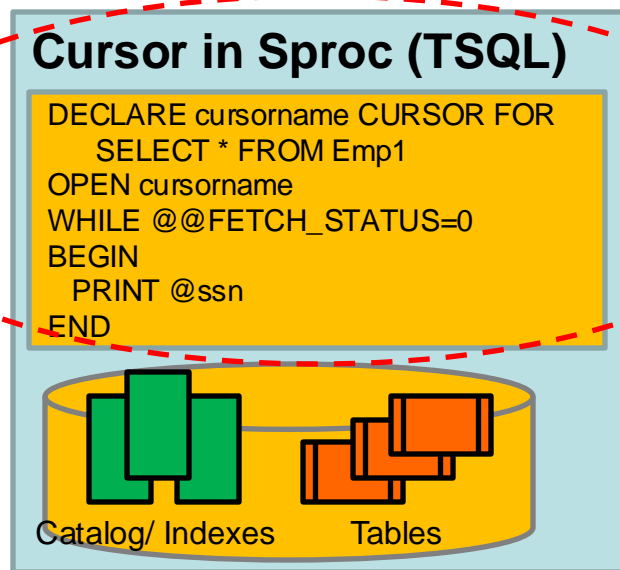
(Άλλες Διαφορές UDF vs. Sprocs)

- Το **UDF** είναι **υποπρόγραμμα** το οποίο γράφεται για να εκτελεί κάποιους **υπολογισμούς** και να επιστρέφει **μια μοναδική τιμή**.
- Το **Sproc** είναι **υποπρόγραμμα** το οποίο γράφεται για να εκτελεί μια **ακολουθία** από **εντολές** και να **επιστρέφει 0** ή **περισσότερες τιμές**.
- Τα **UDFs ΠΡΕΠΕΙ** να επιστρέφουν τιμή με το **RETURN** ενώ τα **SPROCs** μπορούν να χρησιμοποιούν το **RETURN** αλλά **χωρίς να επιστρέφουν τιμή**.
- UDFs μπορεί να χρησιμοποιηθούν στο **SELECT** (δεδομένου του ότι δεν κάνουν επεξεργασία πινάκων)
- Τα UDFs έχουν μόνο **IN** παραμέτρους. Τα SPROCs μπορεί να έχουν **OUT** ή **IN/OUT** παραμέτρους.

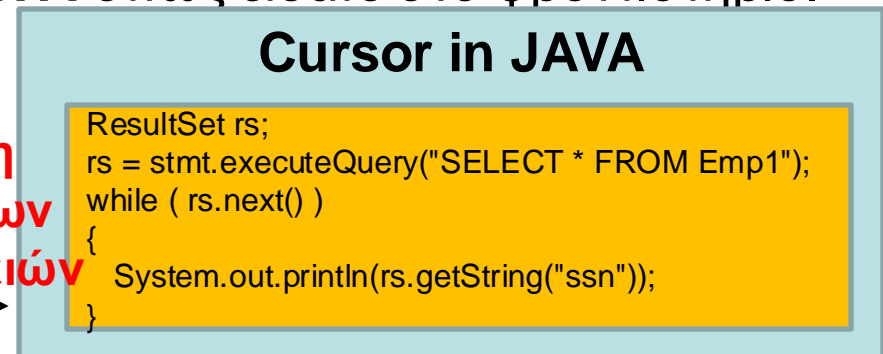
D) Cursors (Δρομείς) σε TSQL



- Γνωρίζουμε ότι τα **αποτελέσματα** επερωτήσεων επιστρέφονται σε μια επερώτηση υπό **μορφή μιας σχέσης**
 - Π.χ., “**SELECT * FROM Emp1**” επιστρέφει ένα ολόκληρο πίνακα.
- Εάν θέλουμε να **επεξεργαστούμε** τα **αποτελέσματα** αυτά **γραμμή-γραμμή** (αντί να απλά να τυπωθούν), τότε χρησιμοποιούμε την έννοια του **Δρομέα (Cursor)**.
 - **Cursors** υπάρχουν και στην **JAVA** όπως είδατε στο φροντιστήριο.



**Εμφαση
επόμενων
διαφανειών**



- **ΔΕΝ** χρησιμοποιούμε CURSORS ως υποκατάστατο σε SQL queries εφόσον δεν βελτιστοποιούνται και αργά.
- **Μόνο για περιπλοκά queries** (π.χ., πολλαπλά βήματα όπως **αναδρομική κλειστότητα**, κτλ.)

D) Cursors (Δρομείς) σε TSQL (Παράδειγμα Χρήσης)



```
USE ep1342
DECLARE @ssn nchar(10), @fname nchar(10)
DECLARE employee_cursor CURSOR FOR -- Δήλωση μεταβλητής τύπου Cursor
    SELECT ssn,fname FROM Emp1 -- Επερώτηση που συνδέεται με τον Cursor
OPEN employee_cursor -- Άνοιγμα Cursor (Εκτέλεση Επερώτησης)
-- Ανάγνωση πρώτης γραμμής στις μεταβλητές id, fname
FETCH NEXT FROM employee_cursor
INTO @ssn, @fname
WHILE @@FETCH_STATUS=0 -- όσο δεν άδειασαν τα αποτελέσματα
BEGIN
    PRINT RTRIM(@ssn) + ', ' + RTRIM(@fname) -- RTRIM: φεύγει right spaces
    FETCH NEXT FROM employee_cursor -- Ανάγνωση επόμενης γραμμής
    INTO @ssn, @fname
END
CLOSE employee_cursor -- Κλείσιμο Cursor
DEALLOCATE employee_cursor -- Αποδέσμευση Πίνακα Ενδιάμεσων
Αποτελεσμάτων που χρησιμοποιεί ο SqlServer για το Result του Cursor .
Εναλλακτικά μένει στην μνήμη μέχρι το Κλείσιμο του Session.
```

0=OK, -1=ERROR,

ΤΥΠΩΝΕΙ

1, 1
2, Franklin
3, Alicia
4, Jennifer
5, Ramesh
6, Joyce
7, Ahmad
8, James
1, 1

D) Cursors (Δρομείς) σε TSQL (Χαρακτηριστικά Cursor)



- **Κατεύθυνση Cursor:** Η default λειτουργία του cursor είναι να κινείται μπροστά **NEXT (default)**, εγγραφή-εγγραφή μέσα σε ένα αποτέλεσμα. Υπάρχουν ορίσματα (**SCROLL**) για να κινείται διαφορετικά **FIRST, LAST, PRIOR, RELATIVE, κτλ.**
- **Ενημέρωση Αποτελεσμάτων:** Είναι δυνατό να ενημερώνεται το αποτέλεσμα (από άλλα transactions) που προσπελάζεται από ένα CURSOR (**READ ONLY (default) | UPDATE**)
- **Εμβέλεια Cursor:** Η εμβέλεια ενός CURSOR μπορεί να περιοριστεί τοπικά μέσα στο ίδιο batch) (**LOCAL (default) | GLOBAL**)
 - local to the batch, stored procedure, or trigger in which the cursor was created.

D) Cursors (Δρομείς) σε TSQL (Παράδειγμα Χρήσης)



FIRST, LAST, PRIOR, RELATIVE

ISO Syntax

```
DECLARE cursor_name [ INSENSITIVE ] [ SCROLL ] CURSOR
  FOR select_statement
  [ FOR { READ ONLY | UPDATE [ OF column_name [ ,...n ] ] } ]
[;]
```

Transact-SQL Extended Syntax

```
DECLARE cursor_name CURSOR [ LOCAL | GLOBAL ]
  [ FORWARD_ONLY | SCROLL ]
  [ STATIC | KEYSET | DYNAMIC | FAST_FORWARD ]
  [ READ_ONLY | SCROLL_LOCKS | OPTIMISTIC ]
  [ TYPE_WARNING ]
  FOR select_statement
  [ FOR UPDATE [ OF column_name [ ,...n ] ] ]
[;]
```

perform cursor work on data without being affected by changes happening in the underlying table while the query is **executing**. Insensitive cursors can't make changes to underlying tables. They do create a **'snapshot' of the existing data in tempdb** when the cursor is declared.

ISO syntax

```
DECLARE MyCursor INSENSITIVE CURSOR
FOR SELECT TOP 1000 * FROM Sales
```

T-SQL extended syntax

```
DECLARE MyCursor CURSOR STATIC
FOR SELECT TOP 1000 * FROM Sales
```

D) Cursors (Δρομείς) σε TSQL

(Παράδειγμα Script για Backup Βάσεων)



```
DECLARE @name VARCHAR(50) -- database name
DECLARE @path VARCHAR(256) -- path for backup files
DECLARE @fileName VARCHAR(256) -- filename for backup
DECLARE @fileDate VARCHAR(20) -- used for file name
```

```
SET @path = 'C:\Backup\' -- κατάλογος στον οποίο θα γίνει το backup
```

```
SELECT @fileDate = CONVERT(VARCHAR(20),GETDATE(),112)
```

```
DECLARE db_cursor CURSOR FOR
  SELECT name
  FROM master.dbo.sysdatabases
  WHERE name NOT IN ('master','model','msdb','tempdb')
```

Μορφοποίηση Ημερ. με
style 112, δηλ. σε:
20091119

```
-- Continued On Next Page
```

Πίνακας του Καταλόγου που περιέχει τα
ονόματα όλων των βάσεων

D) Cursors (Δρομείς) σε TSQL (Παράδειγμα Χρήσης)



-- Continued From Previous Page

```
OPEN db_cursor -- άνοιγμα cursor
```

```
FETCH NEXT FROM db_cursor -- ανάκτηση επόμενης τιμής σε τοπική var  
INTO @name
```

```
WHILE @@FETCH_STATUS = 0 π.χ., 'C:\Backup\ep1342_20091119.BAK  
BEGIN
```

```
-- δημιουργία filename
```

```
SET @fileName = @path + @name + '_' + @fileDate + '.BAK'
```

```
-- εντολή για backup βάσης στο directory/name @fileName
```

```
BACKUP DATABASE @name TO DISK = @fileName
```

```
-- Το RESTORE DATABASE κάνει restore την βάση
```

```
FETCH NEXT FROM db_cursor INTO @name
```

```
END
```

```
CLOSE db_cursor
```

```
DEALLOCATE db_cursor
```

E) Transactions in TSQL



Μια **δοσοληψία (transaction)** είναι μια συλλογή από SQL statements η οποία εκτελείται ως 1 μονάδας εργασίας διασφαλίζοντας έτσι μεταξύ άλλων την ατομικότητα (όλα ή τίποτα!)

```
USE tempdb;
GO
CREATE TABLE ValueTable ([value] int);
GO

DECLARE @TransactionName varchar(20) = 'Transaction1';

BEGIN TRAN @TransactionName
    INSERT INTO ValueTable VALUES(1), (2);
ROLLBACK TRAN @TransactionName;

INSERT INTO ValueTable VALUES(3),(4);

SELECT [value] FROM ValueTable;

DROP TABLE ValueTable;
```

Result

value
3
4

SQL Transaction Control Language (SQL-TCL)



- Με τις προεκτάσεις TCL της T-SQL (και της SQL ευρύτερα) μπορούμε να βεβαιωθούμε ότι διασφαλίζεται η σημασιολογική ακεραιότητα των δεδομένων πέρα από τους μηχανισμούς που είδαμε ήδη.
- TCL commands are:
- **BEGIN TRAN** - begin of transaction
- **COMMIT TRAN** - commit for completed transaction
- **ROLLBACK** - go back to beginning if something was not right in transaction.

SQL-TCL



```
50 BEGIN TRANSACTION
51 UPDATE dbo.authors
52   SET au_fname = 'Almir'
53   WHERE au_id = '172-32-1176'
54
55 UPDATE authors
56   SET au_fname = 'Almir'
57   WHERE city = 'Mostar'
58
59 IF @@ROWCOUNT = 5
60     COMMIT TRANSACTION
61 ELSE
62     ROLLBACK TRANSACTION
63
```

TSQL Script & Stored
Procedure ΔΕΝ είναι
ΑΤΟΜΙΚΑ!

Στην επόμενη διαφάνεια θα
τα κάνουμε!

Atomic Stored Procedure



```
CREATE PROCEDURE UpdateWorkerData
```

```
    @workerId INT,
```

```
    @newHoursWorked INT,
```

```
    @newBonusAmount DECIMAL(10, 2)
```

```
AS
```

```
BEGIN
```

```
    -- Start a transaction
```

```
    BEGIN TRANSACTION;
```

```
    BEGIN TRY
```

```
        -- Update WorkHoursTable
```

```
        UPDATE WorkHoursTable
```

```
        SET HoursWorked = @newHoursWorked
```

```
        WHERE WorkerID = @workerId;
```

```
        -- Check if any rows were updated
```

```
        IF @@ROWCOUNT = 0
```

```
        BEGIN
```

```
            -- Rollback and raise an error if no rows were affected
```

```
            ROLLBACK TRANSACTION;
```

```
            RAISERROR ('No rows updated in WorkHoursTable', 16, 1);
```

```
            RETURN;
```

```
        END
```

```
        -- Update WorkerBonuses
```

```
        UPDATE WorkerBonuses
```

```
        SET BonusAmount = @newBonusAmount
```

```
        WHERE WorkerID = @workerId;
```

```
        -- Check if any rows were updated
```

```
        IF @@ROWCOUNT = 0
```

```
        BEGIN
```

```
            -- Rollback and raise an error if no rows were affected
```

```
            ROLLBACK TRANSACTION;
```

```
            RAISERROR ('No rows updated in WorkerBonuses', 16, 1);
```

```
            RETURN;
```

```
        END
```

```
        -- Commit the transaction if both updates succeed
```

```
        COMMIT TRANSACTION;
```

```
    END TRY
```

```
    BEGIN CATCH
```

```
        -- Rollback transaction if an error occurs
```

```
        ROLLBACK TRANSACTION;
```

```
        -- Rethrow the error
```

```
        THROW;
```

```
    END CATCH
```

```
END;
```

```
-- Step 1: Create the WorkerBonuses table  
DROP TABLE IF EXISTS WorkerBonuses;
```

```
CREATE TABLE WorkerBonuses (  
    WorkerID INT PRIMARY KEY,  
    BonusAmount DECIMAL(10, 2)  
);
```

```
-- Step 2: Insert sample data into WorkerBonuses table  
INSERT INTO WorkerBonuses (WorkerID,  
    BonusAmount)  
VALUES  
(1, 1000.00),  
(2, 1200.00),  
(3, 1500.00),  
(4, 500.00),  
(5, 800.00);
```

F) Indexes in TSQL



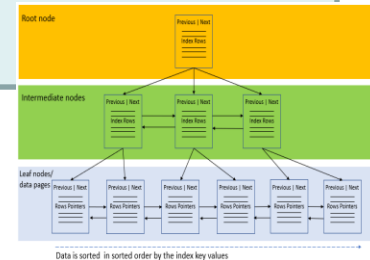
- Ένα **ευρετήριο (index)** είναι μια δομή δεδομένων (μαγνητικού-μέσου) η οποία βελτιώνει την επίδοση ανάκτησης δεδομένων από τους πίνακες.

- **Παράδειγμα**

- **CREATE INDEX** idx **ON** PER (LastName);
- **SELECT *** from PER WHERE Lastname="X"; # now much faster!
- **DROP INDEX** idx; # drops the index.

- **Clustered (Nonclustered) index:** η ταξινόμηση των δεδομένων του ευρετηρίου είναι η ίδια (όχι η ίδια) με αυτή των δεδομένων.

- **CREATE CLUSTERED INDEX** ix_parts_id **ON** production.parts (part_id);

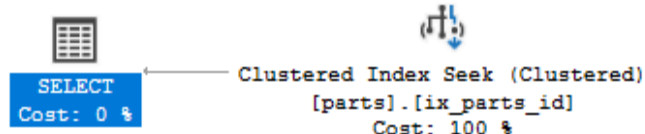


Data is sorted in sorted order by the index key values

w/out index



Query 1: Query cost (relative to the batch): 100%
SELECT part_id, part_name FROM production.parts WHERE part_id = 5



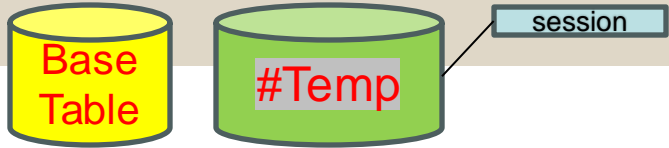
G) Ενδιάμεσοι Πίνακες σε TSQL

(Temporary Tables in TSQL)



- Οι **Τοπικοί Ενδιάμεσοι Πίνακες** δημιουργούνται στα πλαίσια μιας σύνδεσης με την βάση (Session) και καταστρέφονται αμέσως μετά.
 - Είναι χρήσιμοι για προσωρινή αποθήκευση ενδιάμεσων αποτελεσμάτων από προγράμματα TSQL
 - **Δημιουργία:** `CREATE TABLE #tablename (...)`
 - Σημείωση: Ο πίνακας δημιουργείται στην `tempdb`, γίνεται μέρος του transaction log.
 - **Εισαγωγή Δεδ.:** `INSERT INTO #TemporaryTable`
`EXEC sp_SomeStoredProcedure`
 - Καταστροφή: Αυτόματα μόλις κλείσει η σύνδεση
- **Καθολικοί Ενδιάμεσοι Πίνακες:**
 - Έχουν αντίστοιχη λειτουργία με τους Τοπικούς, με την διαφορά ότι σε αυτούς έχουν πρόσβαση όλες οι συνδέσεις με την βάση (sessions).
 - **Δημιουργία:** `CREATE TABLE ##tablename (...)`
 - **Καταστροφή:** Αυτόματα μόλις κλείσει η σύνδεση **ΚΑΙ** δεν υπάρχει άλλη δοσοληψία που εργάζεται στον

Παράδειγμα με (Local) Temp Table



-- Drop the base table and table (optional)

```
DROP TABLE IF EXISTS #TopWorkers;  
DROP TABLE IF EXISTS WorkHoursTable;
```

-- Step 1: Create the WorkHoursTable

```
CREATE TABLE WorkHoursTable (  
    WorkerID INT PRIMARY KEY,  
    WorkerName NVARCHAR(100),  
    HoursWorked INT  
);
```

-- Step 2: Insert sample data into WorkHoursTable

```
INSERT INTO WorkHoursTable (WorkerID, WorkerName,  
HoursWorked)  
VALUES  
(1, 'Alice', 45),  
(2, 'Bob', 50),  
(3, 'Charlie', 60),  
(4, 'Diana', 30),  
(5, 'Eve', 40),  
(6, 'Frank', 55),  
(7, 'Grace', 20),  
(8, 'Hank', 25),  
(9, 'Ivy', 35)
```

-- Step 3: Create a temporary table for the top workers

```
CREATE TABLE #TopWorkers (  
    WorkerID INT PRIMARY KEY,  
    WorkerName NVARCHAR(100),  
    TotalHoursWorked INT  
);
```

-- Step 4: Insert the top 3 workers based on total hours worked

```
INSERT INTO #TopWorkers (WorkerID, WorkerName, TotalHoursWorked)  
SELECT TOP 3
```

```
    WorkerID,  
    WorkerName,  
    SUM(HoursWorked) AS TotalHoursWorked
```

```
FROM WorkHoursTable  
GROUP BY WorkerID, WorkerName  
ORDER BY TotalHoursWorked DESC;
```

-- Step 5: Retrieve the data from the temporary table

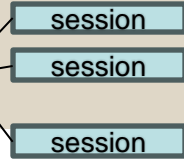
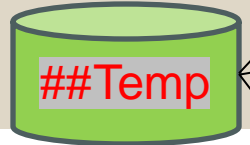
```
SELECT * FROM #TopWorkers;
```

ΠΡΙΝ το
κλείσιμο του
παραθύρου/
session

Μετά το
κλείσιμο του
παραθύρου /
session



Παράδειγμα με (Global) Temp Table



-- Drop the base table and table (optional)

```
DROP TABLE IF EXISTS ##TopWorkers;
DROP TABLE IF EXISTS WorkHoursTable;
```

-- Step 1: Create the WorkHoursTable

```
CREATE TABLE WorkHoursTable (
  WorkerID INT PRIMARY KEY,
  WorkerName NVARCHAR(100),
  HoursWorked INT
);
```

-- Step 2: Insert sample data into WorkHoursTable

```
INSERT INTO WorkHoursTable (WorkerID, WorkerName,
HoursWorked)
VALUES
(1, 'Alice', 45),
(2, 'Bob', 50),
(3, 'Charlie', 60),
(4, 'Diana', 30),
(5, 'Eve', 40),
(6, 'Frank', 55),
(7, 'Grace', 20),
(8, 'Hank', 25),
(9, 'Ivy', 35),
(10, 'Jack', 48),
(11, 'Karen', 52),
(12, 'Liam', 38),
(13, 'Mia', 22)
```

-- Step 3: Create a temporary table for the top workers

```
CREATE TABLE ##TopWorkers (
  WorkerID INT PRIMARY KEY,
  WorkerName NVARCHAR(100),
  TotalHoursWorked INT
);
```

-- Step 4: Insert the top 3 workers based on total hours worked

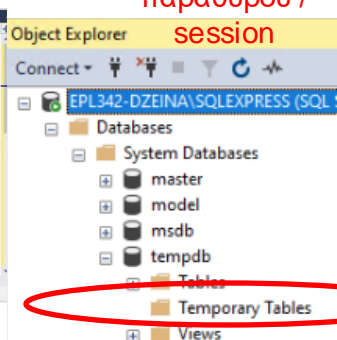
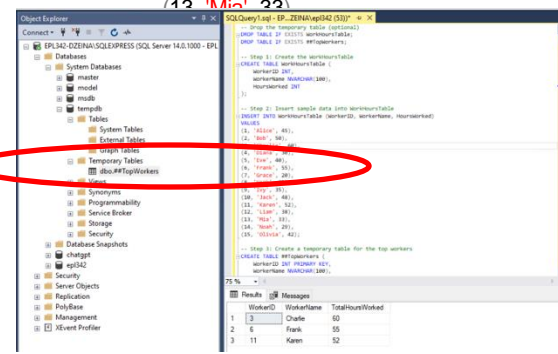
```
INSERT INTO ##TopWorkers (WorkerID, WorkerName, TotalHoursWorked)
SELECT TOP 3
  WorkerID,
  WorkerName,
  SUM(HoursWorked) AS TotalHoursWorked
FROM WorkHoursTable
GROUP BY WorkerID, WorkerName
ORDER BY TotalHoursWorked DESC;
```

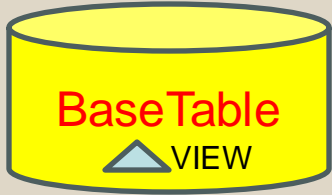
-- Step 5: Retrieve the data from the temporary table

```
SELECT * FROM ##TopWorkers
```

Μετά το κλείσιμο του παραθύρου / session

ΠΡIN το κλείσιμο του παραθύρου / session





Παράδειγμα με VIEW



-- Step 3: Create a temporary table for the top workers

-- Drop the view

```
DROP VIEW IF EXISTS TopWorkers;
```

GO -- Avoid error message "View must be the first statement in batch"

```
CREATE VIEW TopWorkers AS  
  SELECT TOP 3  
    WorkerID,  
    WorkerName,  
    SUM(HoursWorked) AS TotalHoursWorked  
  FROM WorkHoursTable  
  GROUP BY WorkerID, WorkerName  
  ORDER BY TotalHoursWorked DESC;
```

GO -- Avoid error message "View must be the first statement in batch"

-- Step 5: Retrieve the data from the temporary table

```
SELECT * FROM TopWorkers;
```

-- Drop the base table and table (optional)
DROP TABLE IF EXISTS WorkHoursTable;

-- Step 1: Create the WorkHoursTable

```
CREATE TABLE WorkHoursTable (  
  WorkerID INT PRIMARY KEY,  
  WorkerName NVARCHAR(100),  
  HoursWorked INT  
);
```

-- Step 2: Insert sample data into
WorkHoursTable

```
INSERT INTO WorkHoursTable (WorkerID,  
WorkerName, HoursWorked)  
VALUES  
(1, 'Alice', 45),  
(2, 'Bob', 50),  
(3, 'Charlie', 60),  
(4, 'Diana', 30),  
(5, 'Eve', 40),  
(6, 'Frank', 55),  
(7, 'Grace', 20),  
(8, 'Hank', 25),  
(9, 'Ivy', 35),  
(10, 'Jack', 48),  
(11, 'Karen', 52),  
(12, 'Liam', 38),  
(13, 'Mia', 33),  
(14, 'Noah', 29),  
(15, 'Olivia', 42);
```



Παράδειγμα με Table



-- Step 3: Create a temporary table for the top workers

```
DROP TABLE IF EXISTS TopWorkers;
```

```
CREATE TABLE TopWorkers (  
  WorkerID INT PRIMARY KEY,  
  WorkerName NVARCHAR(100),  
  TotalHoursWorked INT  
);
```

-- Step 4: Insert the top 3 workers based on total hours worked

```
INSERT INTO TopWorkers (WorkerID, WorkerName, TotalHoursWorked)  
  SELECT TOP 3  
    WorkerID,  
    WorkerName,  
    SUM(HoursWorked) AS TotalHoursWorked  
  FROM WorkHoursTable  
  GROUP BY WorkerID, WorkerName  
  ORDER BY TotalHoursWorked DESC;
```

-- Step 5: Retrieve the data from the temporary table

```
SELECT * FROM TopWorkers;
```

-- Drop the base table and table (optional)
DROP TABLE IF EXISTS WorkHoursTable;

-- Step 1: Create the WorkHoursTable

```
CREATE TABLE WorkHoursTable (  
  WorkerID INT PRIMARY KEY,  
  WorkerName NVARCHAR(100),  
  HoursWorked INT  
);
```

-- Step 2: Insert sample data into
WorkHoursTable

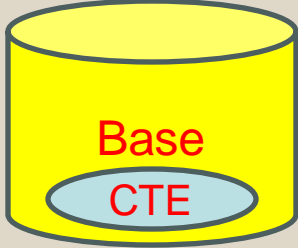
```
INSERT INTO WorkHoursTable (WorkerID,  
  WorkerName, HoursWorked)  
VALUES  
(1, 'Alice', 45),  
(2, 'Bob', 50),  
(3, 'Charlie', 60),  
(4, 'Diana', 30),  
(5, 'Eve', 40),  
(6, 'Frank', 55),  
(7, 'Grace', 20),  
(8, 'Hank', 25),  
(9, 'Ivy', 35),  
(10, 'Jack', 48),  
(11, 'Karen', 52),  
(12, 'Liam', 38),  
(13, 'Mia', 33),  
(14, 'Noah', 29),  
(15, 'Olivia', 42);
```

Ενδιάμεσα **Result-Sets** σε TSQL

(WITH common_table_expression CTE) in TSQL



- Ορίζει εν ενδιάμεσο result set στα πλαίσια ενός **SELECT, INSERT, UPDATE, DELETE**
 - Δηλαδή, **ΔΕΝ** ορίζει κάποιου είδους πίνακα όπως τα #, ##, table, view που είδαμε νωρίτερα.
 - Επομένως ΔΕΝ το βρίσκουμε στο SSMS (management studio)
 - **Εμβέλεια / Διάρκεια:** Στα πλαίσια ενός απλού query ή view (για διευκόλυνση διατύπωσης – αντί εμφωλευμένων επερωτήσεων).
 - Υπάρχει η έννοια του **αυτό-αναφορικού (recursive) CTE** που θα δούμε σύντομα.



Παράδειγμα με CTE



WITH TotalHoursCTE AS (

SELECT

E.Fname **AS** FirstName,

E.Lname **AS** LastName,

SUM(W.Hours) **AS** TotalHours

FROM

EMPLOYEE E, WORKS_ON W

WHERE

E.Ssn = W.Essn

GROUP BY

E.Fname, E.Lname

)

SELECT TOP 3 *

FROM TotalHoursCTE -- **Querying Result-set**

ORDER BY TotalHours **DESC**;

-- Drop the base table and table (optional)
DROP TABLE IF EXISTS WorkHoursTable;

-- Step 1: Create the WorkHoursTable
CREATE TABLE WorkHoursTable (
WorkerID **INT PRIMARY KEY**,
WorkerName **NVARCHAR**(100),
HoursWorked **INT**
);

-- Step 2: Insert sample data into
WorkHoursTable
INSERT INTO WorkHoursTable (WorkerID,
WorkerName, HoursWorked)
VALUES
(1, 'Alice', 45),
(2, 'Bob', 50),
(3, 'Charlie', 60),
(4, 'Diana', 30),
(5, 'Eve', 40),
(6, 'Frank', 55),
(7, 'Grace', 20),
(8, 'Hank', 25),
(9, 'Ivy', 35),
(10, 'Jack', 48),
(11, 'Karen', 52),
(12, 'Liam', 38),
(13, 'Mia', 33),
(14, 'Noah', 29),
(15, 'Olivia', 42);

Παράδειγμα Αναδρομικής CTE (Recursive CTE)



```
WITH EmployeeHierarchy AS (  
  -- Anchor member: Start with the given manager (e.g., Ssn = 333445555)  
  SELECT  
    E.Ssn AS EmployeeSSN,  
    E.Fname AS FirstName,  
    E.Lname AS LastName,  
    E.Super_ssn AS SupervisorSSN,  
    1 AS Level  
  FROM EMPLOYEE E  
  WHERE E.Ssn = 333445555 -- Manager's SSN
```

```
UNION ALL
```

```
-- Recursive member: Find employees reporting to the current level
```

```
SELECT  
  E.Ssn AS EmployeeSSN,  
  E.Fname AS FirstName,  
  E.Lname AS LastName,  
  E.Super_ssn AS SupervisorSSN,  
  EH.Level + 1 AS Level  
FROM EMPLOYEE E, EmployeeHierarchy EH  
WHERE E.Super_ssn = EH.EmployeeSSN
```

https://github.com/tolgahanakgun/Elmasri-Database/blob/master/Employee_Database_Script.sql

	EmployeeSSN	FirstName	LastName	SupervisorSSN	Level
1	333445555	Franklin	Wong	888665555	1
2	123456789	John	Smith	333445555	2
3	453453453	Joyce	English	333445555	2
4	666884444	Ramesh	Narayan	333445555	2

```
SELECT *  
FROM EmployeeHierarchy  
ORDER BY Level, EmployeeSSN;
```

Αναδρομική CTE Συζήτηση



- Η αναδρομή σε αυτό-αναφορικές συνενώσεις διατυπώνεται με τη συσχέτιση R^*
 - Αυτό το ονομάσαμε και Transitive Closure του R

$$R^* = R \bowtie R \bowtie \dots \bowtie R$$

Παραδείγματα

- Βρες όλες τις πτήσεις από το **LCA** προς όλα τα αεροδρόμια του εξωτερικού με **ΟΠΟΙΟΔΗΠΟΤΕ** αριθμό ενδιάμεσων στάσεων.
- Βρες όλα τα άτομα που γνωρίζω σε ένα κοινωνικό δίκτυο μέσω οποιοδήποτε φίλων, φίλων των φίλων, ... κτλ.
- Ο **Mark Zuckerberg (Facebook creator)** αναφέρει το 2005 στο **CS50/Harvard** μάθημα πως με **PHP/MySQL** έφτιαξε το **Facebook** και πως η αρχική του επιτυχία στηρίχθηκε σε **αποδοτική υλοποίηση του R^*** <https://youtu.be/xFFs9UgOAlE?t=265> Λεπτό 4:39

CTEs και Θέματα Επίδοσης



- Τα CTE **ΔΕΝ αποθηκεύουν τα υπολογιζόμενα αποτελέσματα**, άρα χρειάζεται να υπολογιστούν σε κάθε κλήση που δεν είναι αποδοτικό.
- **Πως βελτιώνεται;**
- A) Αποθήκευση ενδιάμεσων αποτελεσμάτων **Εντός της Βάσης**
 - A) Χρήση CTE σε ενδιάμεσους πίνακες #, ## (βλεπε προηγούμενες διαφάνειες)
 - B) Χρήση Stored Procedure WHILE, CURSORS, ETC με ενδιάμεσους πίνακες (θα το δούμε αργότερα)
- B) Αποθήκευση ενδιάμεσων αποτελεσμάτων **Εκτός της Βάσης** (π.χ., Memcache / Unix) => Περισσότερα στο ΕΠΛ421

Παράδειγμα Αναδρομικής CTE (**Caching** R-CTE εντός Βάσης)



```
WITH EmployeeHierarchy AS (  
  -- Anchor member: Start with the given manager (e.g., Ssn = 333445555)  
  SELECT  
    E.Ssn AS EmployeeSSN,  
    E.Fname AS FirstName,  
    E.Lname AS LastName,  
    E.Super_ssn AS SupervisorSSN,  
    1 AS Level  
  FROM EMPLOYEE E  
  WHERE E.Ssn = 333445555 -- Manager's SSN
```

	EmployeeSSN	FirstName	LastName	SupervisorSSN	Level
1	333445555	Franklin	Wong	888665555	1
2	123456789	John	Smith	333445555	2
3	453453453	Joyce	English	333445555	2
4	666884444	Ramesh	Narayan	333445555	2

```
UNION ALL
```

```
-- Recursive member: Find employees reporting to the current level
```

```
SELECT  
  E.Ssn AS EmployeeSSN,  
  E.Fname AS FirstName,  
  E.Lname AS LastName,  
  E.Super_ssn AS SupervisorSSN,  
  EH.Level + 1 AS Level  
FROM EMPLOYEE E, EmployeeHierarchy EH  
WHERE E.Super_ssn = EH.EmployeeSSN
```

```
INSERT INTO EmployeeHierarchy  
SELECT * FROM  
EmployeeHierarchyCTE;
```

https://github.com/tolgahanakgun/Elmasri-Database/blob/master/Employee_Database_Script.sql

Αποθήκευση Password σε Βάσεις (HashBytes)



```
DECLARE @UserName NVARCHAR(50) = 'test_user';
DECLARE @Password NVARCHAR(50) = 'secure_password';
-- Insert the user with the SHA2_512 hash of the password
INSERT INTO USER_TABLE (UserName, PasswordHash)
VALUES (
    @UserName,
    HASHBYTES('SHA2_512', @Password) -- Hash password
);
-- Future Usage
SELECT *
FROM USER_TABLE
WHERE
    UserName = @UserName -- Match username
    AND PasswordHash = HASHBYTES('SHA2_512', @Password);
```

```
DROP TABLE IF EXISTS USER_TABLE;
```

```
CREATE TABLE USER_TABLE (
    UserID INT IDENTITY PRIMARY KEY,
    UserName NVARCHAR(50) NOT NULL,
    PasswordHash VARBINARY(128) NOT
    NULL, -- Store hash as binary
    CreatedAt DATETIME DEFAULT
    GETDATE()
);
```

Συνάρτηση Κατακερματισμού

```
HASHBYTES ( '<algorithm>', {
    @input | 'input' } ) <algorithm>::=
MD2 | MD4 | MD5 | SHA | SHA1 |
SHA2_256 | SHA2_512
```

Πεπαλαιωμένα από TSQL 2016

	UserID	UserName	PasswordHash
1	1	test_user	0x757E9A8FAA3F93708EF872390443B4DC94F507D6F60322EA97E605C7ECE338E1...

I) Κατάλογος (Catalog) Βάσης INFORMATION_SCHEMA (ANSI)



- Ο **κατάλογος (catalog)** είναι ένας κεντρικός αποθηκευτικός χώρος που περιέχει μεταδεδομένα για μια βάση δεδομένων (πίνακες, sprocs, views, triggers, κτλ.)
- Στην SQL (ANSI) ονομάζεται **INFORMATION_SCHEMA** και μπορούμε με SQL να ζητήσουμε πληροφορίες από τον Κατάλογο της Βάσης
 - Στην TSQL υπάρχει το **sys.tables** που επικεντρώνεται αποκλειστικά σε user defined tables

Παράδειγμα Διαγραφής ΟΛΩΝ των Πινάκων (TSQL / sys.*)!



-- Step 1: Drop all foreign key constraints

```
DECLARE @sql NVARCHAR(MAX);

SELECT @sql = STRING_AGG('ALTER TABLE ' +
QUOTENAME(OBJECT_SCHEMA_NAME(parent_object_id)) + '.' +
QUOTENAME(OBJECT_NAME(parent_object_id))
+ ' DROP CONSTRAINT ' + QUOTENAME(name), ';')
FROM sys.foreign_keys;
```

```
IF @sql IS NOT NULL
    EXEC sp_executesql @sql;
ELSE
    PRINT 'No foreign key constraints found.';
```

-- Step 2: Drop all tables

```
SELECT @sql = STRING_AGG('DROP TABLE ' +
QUOTENAME(SCHEMA_NAME(schema_id)) + '.' +
QUOTENAME(name), ';')
FROM sys.tables;
```

```
IF @sql IS NOT NULL
    EXEC sp_executesql @sql;
ELSE
    PRINT 'No tables found to delete.';
```

Παράδειγμα Διαγραφής ΟΛΩΝ ΤΩΝ ΠΙΝΑΚΩΝ (ANSI / sys.*)!

-- Declare a variable to store the dynamic SQL for dropping constraints

```
DECLARE @sql NVARCHAR(MAX);
```

-- Generate the SQL to drop all foreign keys

```
SET @sql = (  
    SELECT STRING_AGG('ALTER TABLE [' + TABLE_SCHEMA + '].[' + TABLE_NAME + '] DROP  
CONSTRAINT [' + CONSTRAINT_NAME + '], ';')  
    FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS  
    WHERE CONSTRAINT_TYPE = 'FOREIGN KEY'  
);
```

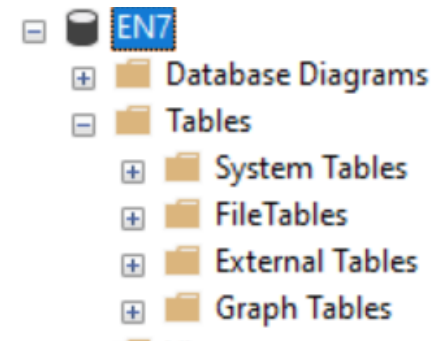
-- Execute the generated SQL to drop foreign keys

```
EXEC sp_executesql @sql;
```

-- Now drop all tables

```
SET @sql = (  
    SELECT STRING_AGG('DROP TABLE [' + TABLE_SCHEMA + '].[' + TABLE_NAME + '], ';')  
    FROM INFORMATION_SCHEMA.TABLES  
    WHERE TABLE_TYPE = 'BASE TABLE'  
);
```

```
EXEC sp_executesql @sql;
```



Remember to refresh tables to
see drop of tables!

J) Unit Testing Λειτουργίες Βάσης



- Δημιουργία Δεδομένων - Βοηθηθείτε μέσω ChatGPT / LLMs

-- Create temporary tables to hold the data -
ChatGPT

```
CREATE TABLE #Department (  
  Dnumber INT PRIMARY KEY,  
  Dname NVARCHAR(50),  
  Mgr_ssn CHAR(9),  
  Mgr_start_date DATE  
);
```

```
CREATE TABLE #DeptLocations (  
  Dnumber INT,  
  Dlocation NVARCHAR(50)  
);
```

```
CREATE TABLE #Employee (  
  Ssn CHAR(9) PRIMARY KEY,  
  Fname NVARCHAR(20),  
  Lname NVARCHAR(20),  
  Dno INT,  
  Super_ssn CHAR(9) NULL,  
  Salary DECIMAL(10, 2)  
);
```

```
CREATE TABLE #Project (  
  Pnumber INT PRIMARY KEY,  
  Pname NVARCHAR(50),  
  Plocation NVARCHAR(50),  
  Dnum INT  
);
```

```
CREATE TABLE #WorksOn (  
  Essn CHAR(9),  
  Pno INT,  
  Hours DECIMAL(4, 1)  
);
```

-- Populate Departments

```
DECLARE @Dnumber INT = 1;  
WHILE @Dnumber <= 4  
BEGIN  
  INSERT INTO #Department (Dnumber, Dname, Mgr_ssn,  
    Mgr_start_date)  
  VALUES (@Dnumber, CONCAT('Department_', @Dnumber),  
    CONCAT('99988', FORMAT(@Dnumber, '000')), DATEADD(YEAR, -  
    @Dnumber, GETDATE()));  
  SET @Dnumber = @Dnumber + 1;  
END
```

-- Populate Employees

```
DECLARE @EmpID INT = 1;  
WHILE @EmpID <= 10  
BEGIN  
  INSERT INTO #Employee (Ssn, Fname, Lname, Dno, Super_ssn, Salary)  
  VALUES (CONCAT('12345', FORMAT(@EmpID, '000')),  
    CONCAT('Emp_FName_', @EmpID),  
    CONCAT('Emp_LName_', @EmpID),  
    @EmpID % 4 + 1,  
    CASE WHEN @EmpID % 2 = 0 THEN NULL ELSE '333445555' END,  
    RAND() * 50000 + 30000);  
  SET @EmpID = @EmpID + 1;  
END
```

-- Populate Department Locations

```
DECLARE @LocIndex INT = 1;  
DECLARE @MaxLocations INT = 2; -- Each department has 2 locations  
WHILE @LocIndex <= @Dnumber * @MaxLocations  
BEGIN  
  INSERT INTO #DeptLocations (Dnumber, Dlocation)  
  VALUES (@Dnumber % 4 + 1, CONCAT('Location_', @LocIndex));  
  SET @LocIndex = @LocIndex + 1;  
END
```

Dnumber	Dname	Mgr_ssn	Mgr_start_date
1	Department_1	99988001	2023-11-01
2	Department_2	99988002	2023-11-01
3	Department_3	99988003	2023-11-01
4	Department_4	99988004	2023-11-01

Dnumber	Dlocation
1	Location_1
2	Location_2
3	Location_3
4	Location_4
5	Location_5
6	Location_6

Ssn	Fname	Lname
12345000	Emp_FName_0	Emp_LName_0
12345001	Emp_FName_1	Emp_LName_1
12345002	Emp_FName_2	Emp_LName_2
12345003	Emp_FName_3	Emp_LName_3
12345004	Emp_FName_4	Emp_LName_4
12345005	Emp_FName_5	Emp_LName_5
12345006	Emp_FName_6	Emp_LName_6
12345007	Emp_FName_7	Emp_LName_7
12345008	Emp_FName_8	Emp_LName_8
12345009	Emp_FName_9	Emp_LName_9
12345010	Emp_FName_10	Emp_LName_10

Pnumber	Pname	Plocation	Dnum
1	Project_1	Obj_1	2
2	Project_2	Obj_2	3
3	Project_3	Obj_3	4
4	Project_4	Obj_4	1
5	Project_5	Obj_5	2
6	Project_6	Obj_6	3

Essn	Pno	Hours
12345000	1	25.1
12345001	2	21.4
12345002	3	40.1
12345003	4	17.7
12345004	5	17.2
12345005	6	34.9
12345006	1	6.4
12345007	2	21.8
12345008	3	21.8
12345009	4	8.8
12345010	5	28.3

Unit Testing Λειτουργίες Βάσης



```
-- UNIT TESTING
SET NOCOUNT ON -- make sure annoying affected rows messages disappear
```

```
-- UNIT TESTING
```

```
-- 1. Test: Department Count
```

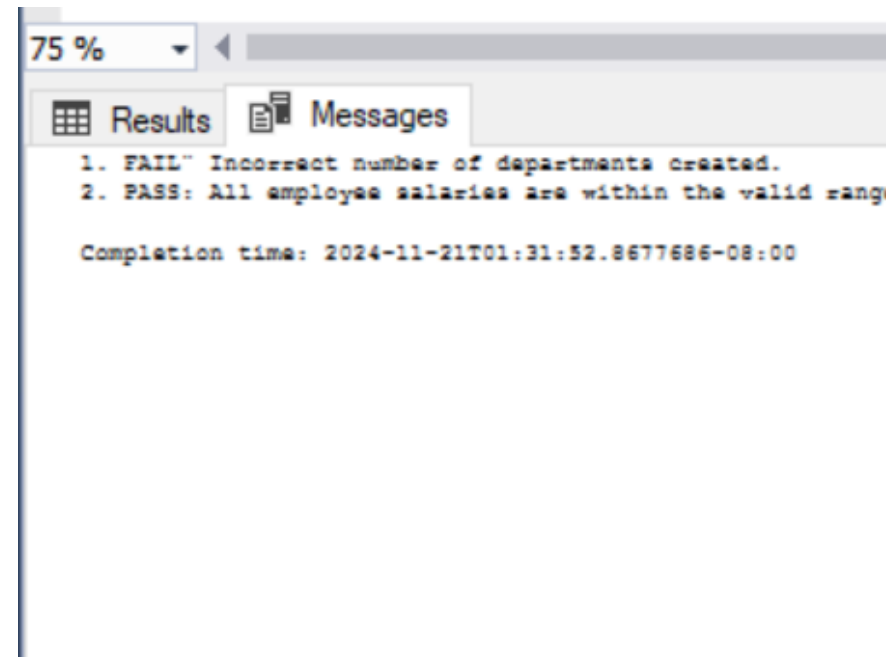
```
IF (SELECT COUNT(*) FROM #Department) = 3
    PRINT '1. PASS: Correct number of departments created.';
ELSE
    PRINT '1. FAIL" Incorrect number of departments created.';
```

```
-- 2. Test: Check Salary Range
```

```
IF NOT EXISTS (
    SELECT 1
    FROM #Employee
    WHERE Salary < 30000 OR Salary > 80000
)
    PRINT '2. PASS: All employee salaries are within the valid range.';
ELSE
    PRINT '2. FAIL: Employee salary is out of the valid range.';
```

```
-- Cleanup temporary tables
```

```
DROP TABLE #Department;
DROP TABLE #DeptLocations;
DROP TABLE #Employee;
DROP TABLE #Project;
DROP TABLE #WorksOn;
```



H) Υλοποίηση Προγρ. Αντικειμένων Βάσης μέσω Γλωσσών Υψηλού Επίπεδου



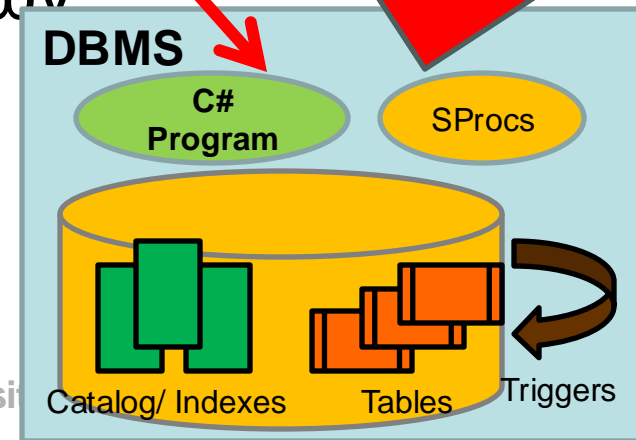
- Τα προγραμματιστικά αντικείμενα μιας βάσης (UDFs, Sprocs, Triggers, User-Defined Types, κτλ) μπορούν σε πολλές βάσεις να γραφούν με σύνταξη **γλωσσών υψηλού επιπέδου** αντί με σύνταξη **SQL/PSM (TSQL, PL/SQL)** π.χ.,

- **SQL Server:** Χρήση .NET και C# (δες SQLCLR: SQL Common Language Runtime). Ακολουθεί παράδειγμα στην επόμενη διαφάνεια.

- **Oracle και DB2:** Υλοποίηση UDFs σε C ή JAVA

- **Λόγοι Χρήσης** αυτών των Τεχνικών:

- Πιο πολλές **δυνατότητες** σε μια **γλώσσα υψηλού επιπέδου**
- **Επίδοση, Ασφάλεια**, κτλ.



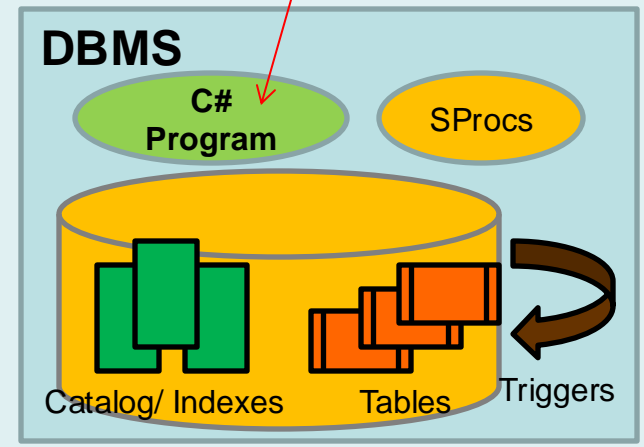
Υλοποίηση Προγρ. Αντικειμένων Βάσης μέσω C#

Προχωρημένο
Εκτός
Μαθήματος

```
using System;
using System.Data;
using System.Data.Sql;
using System.Data.SqlTypes;
using Microsoft.SqlServer.Server;
public partial class UserDefinedFunctions
{
    [Microsoft.SqlServer.Server.SqlFunction]
    public static SqlString RiskProfile()
    {
        // Put your code here
        return new SqlString("Hello");
    }
};
```

Σημείωση: Δεν χρειάζεται να καταλάβετε τι αναπαριστούν οι βιβλιοθήκες στο using

**Υλοποίηση ενός UDF
σε C#
το οποίο
ενσωματώνεται στις
λειτουργίες της βάσης**



Υλοποίηση Προγραμματιστικών Αντικειμένων Βάσης μέσω JAVA



- Μπορούμε να εκτελέσουμε και κώδικα στον SQL Server ο οποίος έχει αναπτυχθεί σε **JAVA**
- Κάνουμε compile (**CLASS**) και package μια βιβλιοθήκη (JAR – “jar cf *jar-file input-file(s)*”) την οποία στη συνέχεια μπορούμε να καλέσουμε σε διάφορα σημεία με **EXEC**.
- **Προϋπόθεση:** Συμπερίληψη του **JAR** στο **CLASSPATH**.

```
DECLARE @param1 int  
SET @param1 = 3
```

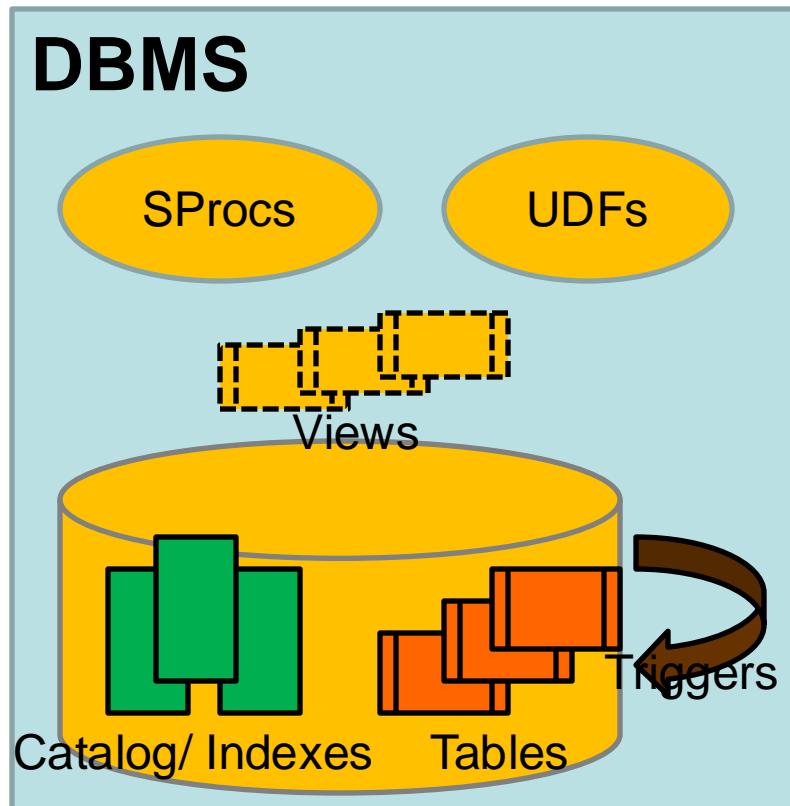
```
EXEC sp_execute_external_script  
@language = N'Java' ,  
@script = N'<packageName>.<ClassName>' ,  
@input_data_1 = N'<Input Query>' ,  
@param1 = @param1
```

<https://docs.microsoft.com/en-us/sql/language-extensions/how-to/call-java-from-sql?view=sql-server-ver15>

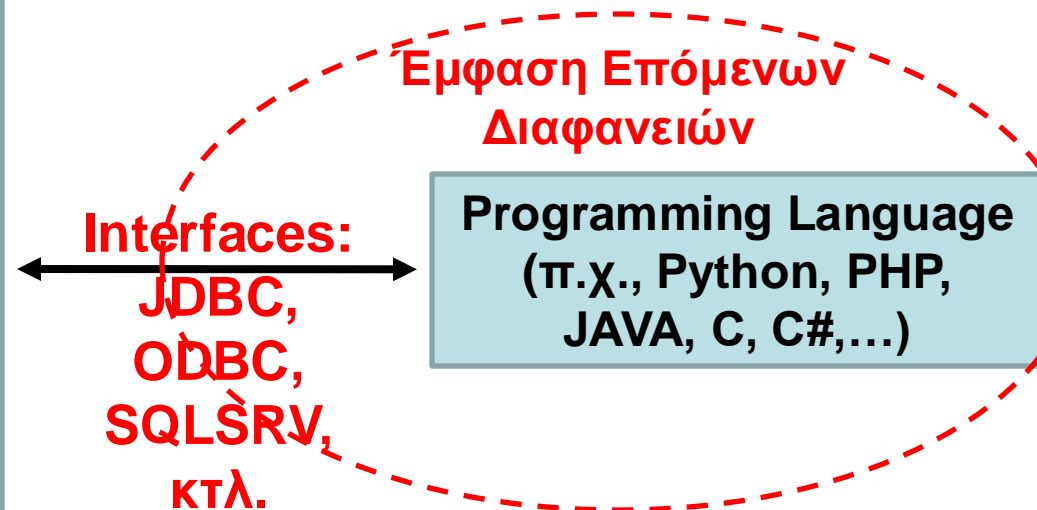


Εξωτερικός Προγραμματισμός DB

Internal DB Programming



External DB Programming



Ενσωματωμένη SQL μέσω Κλήσεων Συναρτήσεων

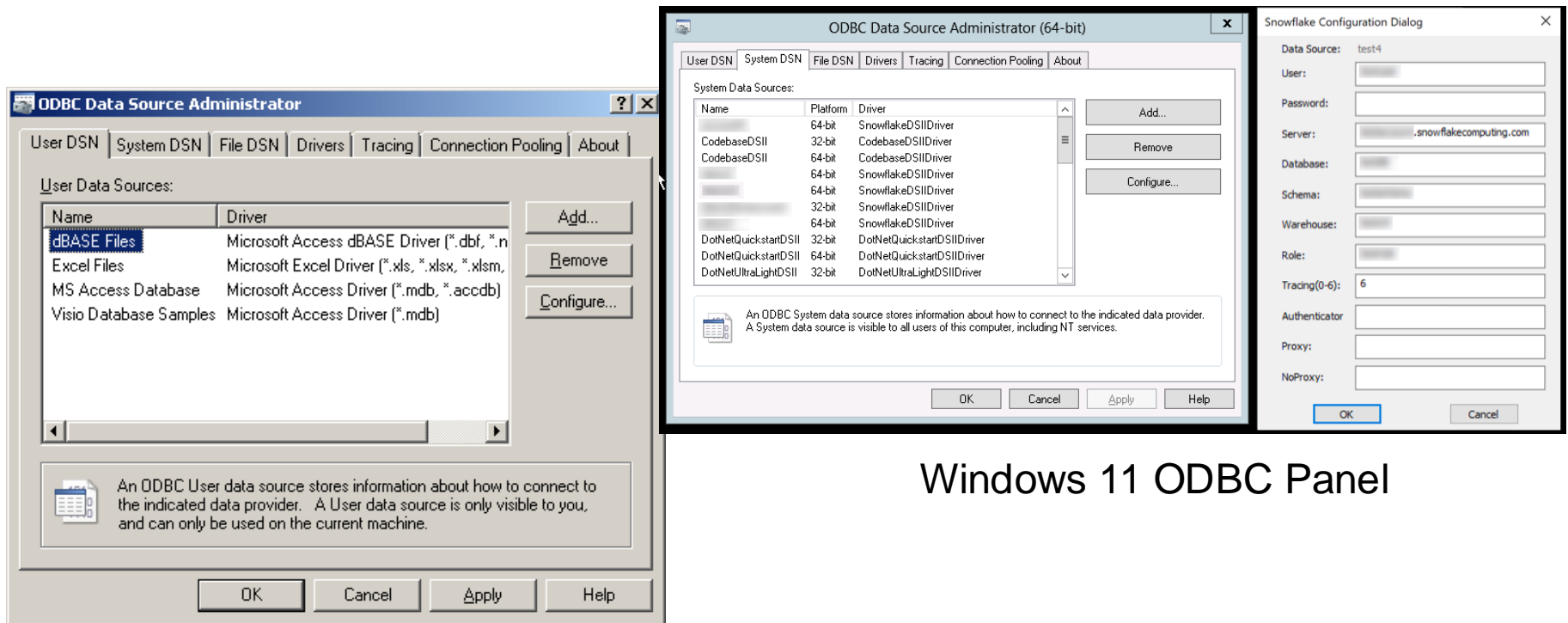


- Ο πιο παραδοσιακός τρόπος σύνδεσης μια εφαρμογής με μια DBMS είναι μέσω **κλήσεων συναρτήσεων**.
 - Εάν η εφαρμογή είναι γραμμένη σε **JAVA** τότε χρησιμοποιείται το **JDBC**, το οποίο είδατε στο φροντιστήριο.
 - Σε Windows χρησιμοποιείται το **ODBC**
 - **Microsoft SQL Server PHP Bridge: SQLSRV (project)**
- Εάν η εφαρμογή είναι γραμμένη σε άλλη γλώσσα τότε χρησιμοποιούνται άλλα API (Application Protocol Interfaces και βιβλιοθήκες).
 - **SQL/CLI** (ο διάδοχος του ODBC) ή **OLEDB** (σε Win)
 - Microsoft's ActiveX Data Objects **ADO** ('96), **ADO.NET**
 - Σύνοψη τεχνολογιών Πρόσβασης σε Δεδομένα από την Microsoft:
<http://support.microsoft.com/kb/190463>

Ενσωματωμένη SQL μέσω Κλήσεων Συναρτήσεων



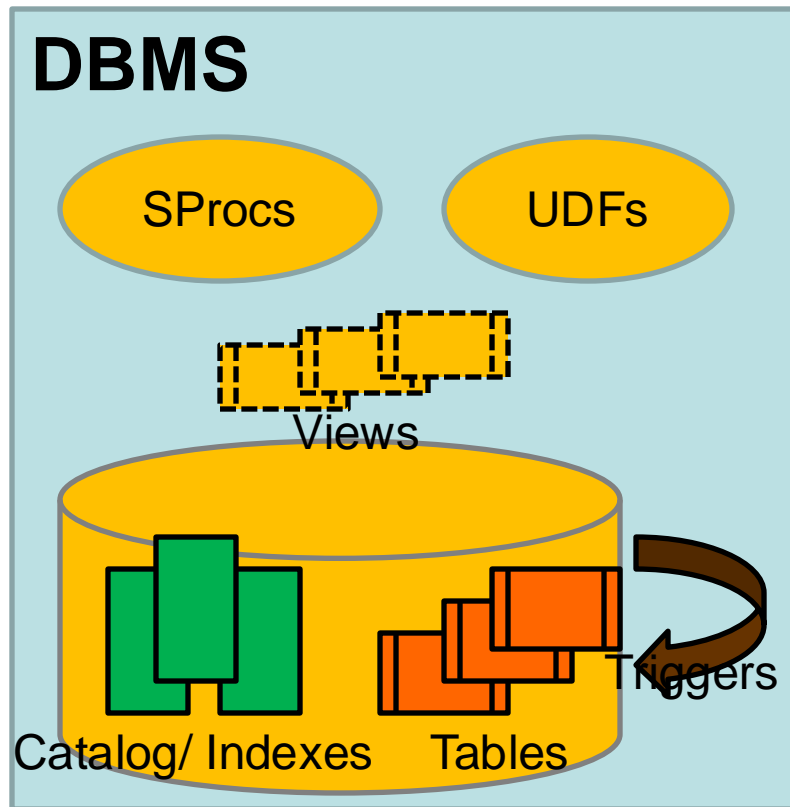
Για σύνδεση μέσω ODBC σε Windows απαιτείται η
δήλωση μιας βάσης στον Data Source
Administrator των Windows



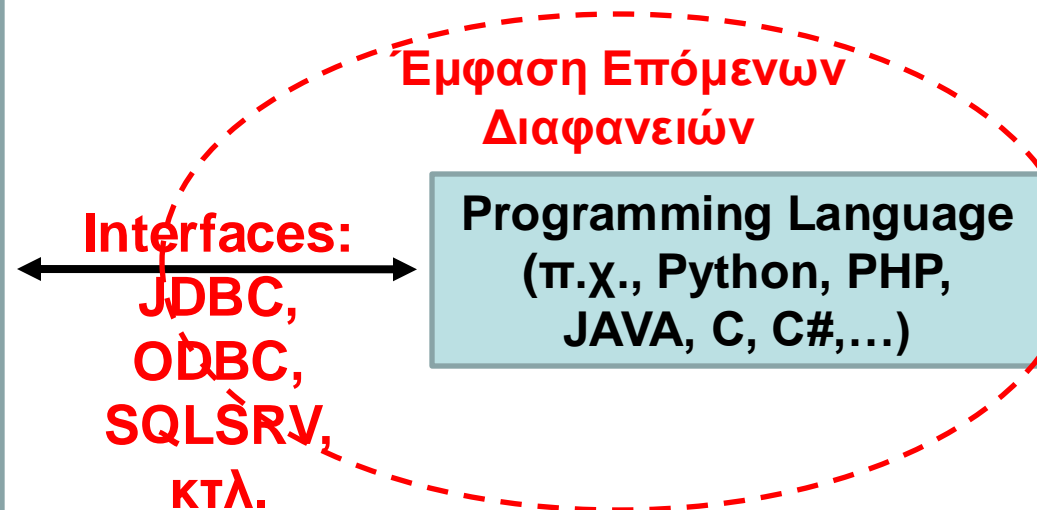
Windows 11 ODBC Panel

Εξωτερικός Προγραμματισμός DB

Internal DB Programming



External DB Programming



Διαδικτυακός Προγραμματισμός

Βάσεων Δεδομένων



- Στο εργαστήριο και την άσκηση είδατε ήδη ένα βασικό σκελετό σύνδεσης στην βάση δεδομένων
 - Περισσότερα στο κεφάλαιο 11 του βιβλίου και μεταγενέστερα μαθήματα (Τεχνολογία Λογισμικού & Προγρ. Συστημάτων)

(a)

```
//Program Segment P1:
0) <?php
1) // Printing a welcome message if the user submitted
   // through the HTML form
2) if ($_POST['user_name']) {
3)   print("Welcome, ");
4)   print($_POST['user_name']);
5) }
6) else {
7)   // Printing the form to enter the user name since no name has
   // been entered yet
8)   print <<<_HTML_
9)   <FORM method="post" action="$_SERVER['PHP_SELF']">
10)  Enter your name: <input type="text" name="user_name">
11)  <BR/>
12)  <INPUT type="submit" value="SUBMIT NAME">
13)  </FORM>
14)  _HTML_;
15) }
16) ?>
```

(b) Enter your name:

(c) Enter your name:

(d) Welcome, John Smith

Stored Procedure in PHP



```
<?php
echo "Connecting to SQL server (" . $serverName . ")<br/>";
echo "Database: " . $connectionOptions[Database] . ", SQL User: " .
    $connectionOptions[Uid] . "<br/>";
//echo "Pass: " . $connectionOptions[PWD] . "<br/>";

$tsql = "{call EmployeesFromCity(?)}";
echo "Executing query: " . $tsql . ") with parameter " . $_GET["city"];

// Getting parameter from the http call and setting it for the SQL call
$params = array(
array($_GET["city"], SQLSRV_PARAM_IN)
);
$getResults= sqlsrv_query($conn, $tsql, $params);
echo ("Results:<br/>");
if ($getResults == FALSE) die(FormatErrors(sqlsrv_errors()));

PrintResultSet($getResults);
sqlsrv_free_stmt($getResults); /* Free query resources. */

sqlsrv_close($conn); /* Free connection resources. */
```

Calling Stored Procedures from JAVA (with input)



```
getEmployeesByLastNameCS(String lastNameStartsWith) {  
    CallableStatement cstmt = null;  
    ResultSet rs = null;  
    try {  
        cstmt = connection.getConnection().prepareCall(  
            "{call HumanResources.uspGetEmployeesByLastName(?)}",  
            ResultSet.TYPE_SCROLL_INSENSITIVE,  
            ResultSet.CONCUR_READ_ONLY);
```

SQL Server Stored Procedure

```
CREATE PROCEDURE [HumanResources].[uspGetEmployeesByLastName]  
    @lastNameStartsWith VARCHAR(20) = 'A'  
AS  
BEGIN  
    SET NOCOUNT ON;  
  
    SELECT Title, FirstName, MiddleName, LastName, Suffix  
    FROM Person.Person  
    WHERE (PersonType = 'EM') AND (LastName LIKE @lastNameStartsWith + '%')  
END  
GO
```

```
        cstmt.setString("lastNameStartsWith", lastNameStartsWith);
```

```
        boolean results = cstmt.execute();           input
```

```
        int rowsAffected = 0;
```

```
        // Protects against lack of SET NOCOUNT in stored procedure
```

```
        while (results || rowsAffected != -1) {  
            if (results) { rs = cstmt.getResultSet(); break; }  
            else { rowsAffected = cstmt.getUpdateCount(); }  
            results = cstmt.getMoreResults();  
        }
```

```
        while (rs.next()) { System.out.println(rs.getString("LastName") + ", "  
            + rs.getString("FirstName") + " " + rs.getString("MiddleName"));  
        }
```

```
    } catch (Exception ex) { e.printStackTrace(); }
```

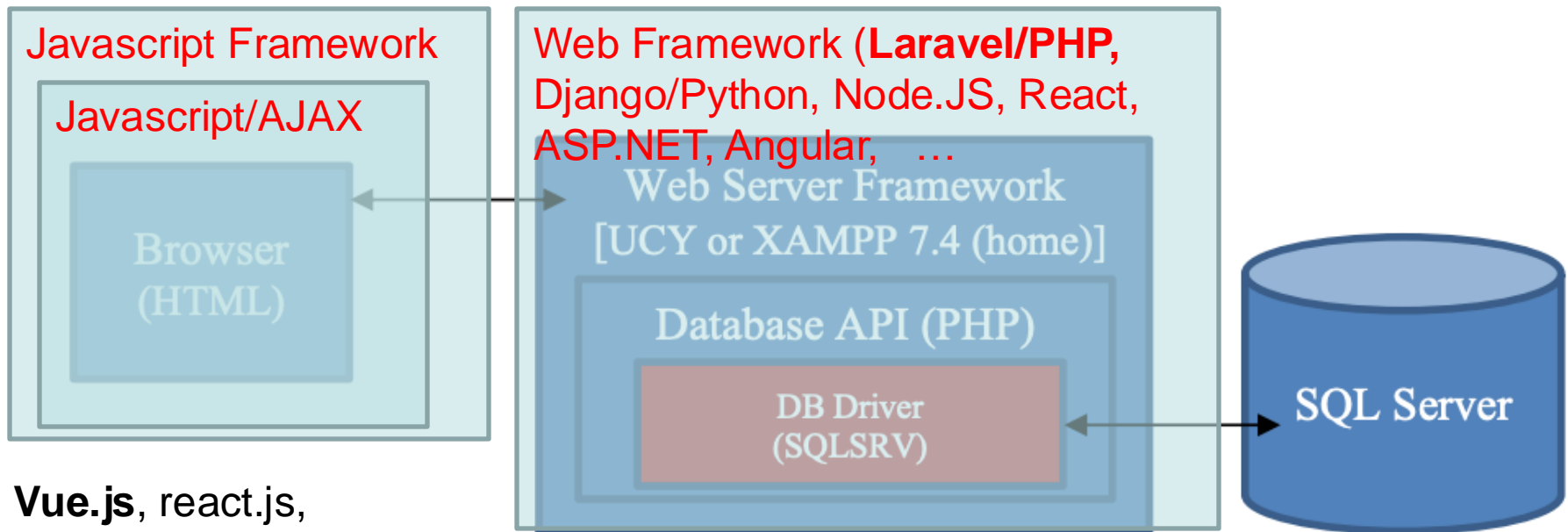
SET NOCOUNT ON

prevents the sending of DONE_IN_PROC messages to the client for each statement in a stored procedure. For stored procedures that contain several statements that do not return much actual data or for procedures that contain Transact-SQL loops, setting SET NOCOUNT to ON can provide a significant performance boost, because network traffic is greatly reduced.

Διαδικτυακός Προγραμματισμός Βάσεων Δεδομένων



- Σήμερα γίνεται εκτεταμένη χρήση πλαισίων για να κλιμακωθεί η ανάπτυξη
- Σημαντικό να γνωρίζουμε διαχείριση δεδομένων στο επίπεδο της βάσης για παρεμβάσεις όπου χρειάζεται.



Vue.js, react.js,
angularjs, Inertia.js,
leaflet.js (maps), ionic
(mobile)

Διαδικτυακός Προγραμματισμός Βάσεων Δεδομένων



The screenshot shows a web browser displaying the EDBT/ICDT 2021 conference page. The page content includes the conference logo, the date and time (Fri, Nov 12, 2021, 12:00 GMT+1), the number of participants (0), a search bar, and a message stating "EDBT/ICDT 2021 Conference finished! We wish you had a wonderful time. Please enjoy the conference material below." The version of VGATE is noted as v1.9.r.

The Chrome DevTools Network tab is open, showing a list of requests. The selected request is for the URL `https://vgate.cs.ncy.ac.cy/edbticdt/clock/`. The request details are as follows:

- Request Method: GET
- Status Code: 200 OK
- Remote Address: 194.42.17.196:443
- Referrer Policy: origin-when-cross-origin

The response headers are:

- Connection: Keep-Alive
- Content-Length: 21
- Content-Type: text/html; charset=UTF-8
- Date: Fri, 12 Nov 2021 10:59:34 GMT
- Keep-Alive: timeout=5

The text "Browser Development Tools (χρήσιμα για προγραμματισμό)" is overlaid on the right side of the screenshot.