



# Ειδικά Θέματα Βάσεων Δεδομένων

## Ενότητα 5: Επεκτάσεις SQL

Δρ. Τιμπίρης Αλκιβιάδης

Τμήμα Μηχανικών Πληροφορικής ΤΕ



# Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



# Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο ΤΕΙ Κεντρικής Μακεδονίας» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.



# Ενότητα 5

---

## Επεκτάσεις SQL

Δρ. Τιμπίρης Αλκιβιάδης

# Περιεχόμενα ενότητας

- Επεκτάσεις της γλώσσας SQL
- Μπλοκ εντολών
- Εντολές IF, WHILE, FOR
- Δηλώσεις μεταβλητών
- Αποθηκευμένες διαδικασίες (Σύνταξη- Εκτέλεση)
- Συναρτήσεις (Σύνταξη- Εκτέλεση)
- Σκανδάλες (Σύνταξη- Εκτέλεση)

# Σκοποί ενότητας

Η ενότητα αυτή αποτελεί ίσως την κυριότερη ενότητα αυτού του συνόλου διαλέξεων. Θα γίνει αναφορά στις επεκτάσεις της γλώσσας SQL και συγκεκριμένα στις εντολές προγραμματισμού που της δίνουν ευελιξία στο αναπτύσσονται σύνθετα προγράμματα. Οι εντολές IF, WHILE, FOR μέσα με μπλοκ εντολών και με συνδυασμό τοπικών και γενικευμένων μεταβλητών μπορούν να χρησιμοποιηθούν σε Αποθηκευμένες διαδικασίες, σε συναρτήσεις οριζόμενες από τον χρήστη αλλά και σε σκανδάλες που ενεργοποιούνται με οποιαδήποτε αλλαγή γίνει σε στοχευόμενους πίνακες.

# Επεκτάσεις SQL

Πρόσθετες προτάσεις Transact-SQL, που αναφέρονται σαν "επεκτάσεις SQL"

- Αποθηκευμένες διαδικασίες ή σκριπτ που αποθηκεύονται στον διακομιστή και μπορούν να ξαναχρησιμοποιηθούν

- Μια δέσμη είναι μια αλληλουχία προτάσεων SQL και επεκτάσεων SQL που στέλνονται στο SQL Server για να εκτελεστούν μαζί.

- Οι προτάσεις ορισμού δεδομένων CREATE VIEW, CREATE PROCEDURE, CREATE RULE, CREATE TRIGGER και CREATE DEFAULT πρέπει να είναι η καθεμία η μόνη πρόταση μέσα σε μια δέσμη.

# Μπλοκ Προτάσεων

---

Ενα μπλοκ επιτρέπει την δημιουργία μονάδων με μια ή περισσότερες προτάσεις SQL. Κάθε μπλοκ αρχίζει με την πρόταση BEGIN και τελειώνει με την πρόταση END:

BEGIN

    Πρόταση 1  
    πρόταση\_2

END



# Πρόταση IF

Ενα μπλοκ μπορεί να χρησιμοποιηθεί μέσα στην πρόταση IF για να επιτρέψει την εκτέλεση περισσότερων της μιας προτάσεων, ανάλογα με μια ορισμένη συνθήκη

```
IF      Συνθήκη Αληθής
        BEGIN
            πρόταση 1
            πρόταση_2
        END
ELSE BEGIN
            πρόταση 3
            πρόταση_4
        END
```

# Πρόταση IF

```
IF (SELECT COUNT(*)
      FROM ΒΑΘΜΟΛΟΓΙΑ
      WHERE KM = '403' AND ΒΑΘΜΟΣ >= 5
      GROUP BY AM ) > 100
    PRINT 'Ο αριθμός των σπουδαστών που πέρασαν το
μάθημα          403 είναι πάνω από 100'
ELSE BEGIN
    PRINT 'Οι σπουδαστές που πέρασαν στο μάθημα 403
είναι:'

    SELECT ΣΠΟΥΔΑΣΤΕΣ.AM, ΕΠΙΘΕΤΟ, ΟΝΟΜΑ
    FROM ΣΠΟΥΔΑΣΤΕΣ, ΒΑΘΜΟΛΟΓΙΑ
    WHERE ΣΠΟΥΔΑΣΤΕΣ.AM = ΒΑΘΜΟΛΟΓΙΑ.AM AND
KM='403'
    END
```

Η πρόταση PRINT είναι μια άλλη επέκταση SQL. Επιστρέφει ένα μήνυμα που ορίζεται από τον χρήστη

# Πρόταση WHILE

Η πρόταση WHILE εκτελεί επαναληπτικά προτάσεις SQL που περικλείονται μέσα σε ένα μπλοκ, όσο η συνθήκη της while επιστρέφει τιμή true.

Ενα μπλοκ μέσα στην πρόταση WHILE μπορεί να περιέχει

BREAK για να σταματά την εκτέλεση των προτάσεων μέσα στο μπλοκ και αρχίζει την εκτέλεση της πρότασης που ακολουθεί αυτό το μπλοκ.

Ή CONTINUE για να σταματά μόνο την τρέχουσα εκτέλεση των προτάσεων μέσα στο μπλοκ και αρχίζει την εκτέλεση του μπλοκ από την αρχή.

# Πρόταση WHILE

```
WHILE (SELECT SUM(ΠΡΟΫΠΟΛΟΓΙΣΜΟΣ)
        FROM ΕΡΓΟ) < 100.000
BEGIN
    UPDATE ΕΡΓΟ SET ΠΡΟΫΠΟΛΟΓΙΣΜΟΣ = ΠΡΟΫΠΟΛΟΓΙΣΜΟΣ
*1.3
    IF (SELECT ΠΡΟΫΠΟΛΟΓΙΣΜΟΣ)
        FROM ΕΡΓΟ) > 50.000 BREAK
    ELSE CONTINUE
END
```

# Τοπικές Μεταβλητές

Επέκταση στην γλώσσα Transact-SQL.

- Αποθηκεύουν τιμές
- Είναι "τοπικές" επειδή χρησιμοποιούνται μόνο μέσα στην ίδια δέσμη, όπου δηλώνονται.
- Το SQL Server υποστηρίζει επίσης καθολικές μεταβλητές(@@)
- Τοπική μεταβλητή ορίζεται με **DECLARE**.
- Ο ορισμός περιέχει το όνομα και τον αντίστοιχο τύπο δεδομένων
- Οι μεταβλητές χρησιμοποιούν το πρόθεμα @
- Η εκχώρηση μιας τιμής σε μια τοπική μεταβλητή γίνεται χρησιμοποιώντας τα παρακάτω:
  1. Με τη **SELECT**
  2. Με την πρόταση **SET**

# Τοπικές Μεταβλητές- ΠΑΡΑΔΕΙΓΜΑ

```
DECLARE @ΜΕΣΗ_ΤΙΜΗ_ΑΓ float, @ΑΥΞ_ΤΙΜΗΣ float
SET @ΑΥΞ_ΤΙΜΗΣ = 10
SELECT @ΜΕΣΗ_ΤΙΜΗ_ΑΓ = AVG(ΤΙΜΗ_ΑΓ) FROM ΑΠΟΘΗΚΗ
IF (SELECT ΤΙΜΗ_ΑΓ FROM ΑΠΟΘΗΚΗ
     WHERE ΚΩΔ_ΕΙΔΟΥΣ='ΑΑ05') < @ΜΕΣΗ_ΤΙΜΗ_ΑΓ
BEGIN
    UPDATE ΑΠΟΘΗΚΗ
    SET ΤΙΜΗ_ΑΓ = ΤΙΜΗ_ΑΓ + @ΑΥΞ_ΤΙΜΗΣ
    WHERE ΚΩΔ_ΕΙΔΟΥΣ = 'ΑΑ05'
    PRINT 'Η ΤΙΜΗ ΑΓΟΡΑΣ ΑΥΞΗΘΗΚΕ ΚΑΤΑ
    @ΑΥΞ_ΤΙΜΗΣ'
END
ELSE PRINT 'Η ΤΙΜΗ ΑΓΟΡΑΣ ΔΕΝ ΑΛΑΞΕ'
```

# Προτάσεις Διαδικασιών

---

- RETURN
- GOTO
- RAISEERROR
- WAITFOR

# RETURN , GOTO

**RETURN** Έχει την ίδια λειτουργικότητα μέσα σε μια δέσμη εντολών με την πρόταση BREAK μέσα στην WHILE.

Αυτό σημαίνει ότι η RETURN κάνει την εκτέλεση της δέσμης να τερματίσει και να αρχίσει να εκτελείται η πρώτη πρόταση που ακολουθεί την δέσμη.

**GOTO:** Διακλαδώνει σε μια ετικέτα, που βρίσκεται μπροστά από μια πρόταση Transact-SQL μέσα σε μια δέσμη



# RAISEERROR

Παράγει ένα σφάλμα που ορίζεται από τον χρήστη και θέτει μια ένδειξη σφάλματος συστήματος.

Ενα μήνυμα σφάλματος που ορίζεται από τον χρήστη πρέπει να είναι μεγαλύτερο από 50000.

Όλοι οι αριθμοί σφάλματος που είναι  $\leq 50000$  είναι αριθμοί που ορίζονται από το σύστημα και δεσμεύονται από το SQL Server.

Οι τιμές σφάλματος αποθηκεύονται στην καθολική μεταβλητή @@ERROR

# WAITFOR

Ορίζει το χρονικό διάστημα που πρέπει να περιμένει το σύστημα, πριν να εκτελέσει την επόμενη πρόταση στην δέσμη.

Η σύνταξη αυτής της πρότασης είναι

**WAITFOR {DELAY 'χρόνος' | TIME 'ώρα'}**

Η **DELAY** λέει στο SQL Server να περιμένει μέχρι να περάσει ο καθορισμένος χρόνος.

Η **TIME** καθορίζει μια ώρα σε μια αποδεκτή μορφή δεδομένων ημερομηνίας/ώρας

---

# Αποθηκευμένες Διαδικασίες

# Αποθηκευμένες Διαδικασίες

**Τι είναι οι Αποθηκευμένες Διαδικασίες?**

**Είναι ένα ειδικό είδος δέσμης γραμμένη σε Transact-SQL, χρησιμοποιώντας την γλώσσα SQL και επεκτάσεις SQL.**

**Αποθηκεύεται στον διακομιστή της βάσης δεδομένων για να βελτιώσει την απόδοση και την συνέπεια επαναληπτικών εργασιών**

**Το SQL Server υποστηρίζει αποθηκευμένες διαδικασίες**

**Δημιουργούνται χρησιμοποιώντας την γλώσσα ορισμού δεδομένων**

# Αποθηκευμένες Διαδικασίες

---

**Η διαδικασία δέχεται τα αντίστοιχα ορίσματα κάθε φορά που καλείται.**

**Οι αποθηκευμένες διαδικασίες μπορούν προαιρετικά να επιστρέψουν μια τιμή, που εμφανίζει τις πληροφορίες που ορίζονται από τον χρήστη**

**ή, στην περίπτωση ενός σφάλματος, το αντίστοιχο μήνυμα σφάλματος**

# Αποθηκευμένες Διαδικασίες

**Μια αποθηκευμένη διαδικασία μεταγλωττίζεται εκ των προτέρων, πριν να αποθηκευτεί σαν αντικείμενο μέσα στην βάση δεδομένων**

**Πλεονέκτημα: η επαναλαμβανόμενη μεταγλώττιση μιας διαδικασίας σχεδόν πάντα εξαλείφεται και η απόδοση της εκτέλεσης αυξάνεται**

**Πλεονέκτημα που αφορά τον όγκο των δεδομένων :**  
χρειάζονται λιγότερα από 50 bytes για να κληθεί μια αποθηκευμένη διαδικασία που περιέχει αρκετές χιλιάδες bytes προτάσεων

# Αποθηκευμένες Διαδικασίες

---

**Γιατί αποθηκευμένες διαδικασίες?**

**Για έλεγχο της εξουσιοδότησης πρόσβασης**

**Για δημιουργία μιας διαδρομής ελέγχου δραστηριοτήτων σε πίνακες βάσης δεδομένων**

**Για διαχωρισμό προτάσεων ορισμού δεδομένων και χειρισμού δεδομένων, που αφορούν μια βάση δεδομένων και όλες τις αντίστοιχες εφαρμογές**

# Αποθηκευμένες Διαδικασίες

## Δημιουργία και Εκτέλεση Αποθηκευμένων Διαδικασιών

### Σύνταξη

```
CREATE PROC [EDURE] [ιδιοκτήτης.]όνομα_διαδικασίας  
    [;αριθμός]  
    [ ( (@παράμετρος1 } τύπος1 [VARYING] [=προεπιλογή1]  
        [OUTPUT]) ) ]  
    { [ ( (@παράμετρος2 } τύπος2 [VARYING] [=προεπιλογή2]  
        [OUTPUT]) ) ] } ...  
    [WITH {RECOMPILE | ENCRYPTION | RECOMPILE,  
        ENCRYPTION}] [FOR REPLICATION]  
AS Δέσμη ΕΝΤΟΛΩΝ
```



# Αποθηκευμένες Διαδικασίες

## Δημιουργία Αποθηκευμένων Διαδικασιών

### Παράδειγμα1

```
CREATE PROCEDURE ΑΥΞΗΣΗ_ΛΙΑΝ_ΤΙΜΗΣ  
    (@ΠΟΣΟΣΤΟ FLOAT=5)  
AS UPDATE ΑΠΟΘΗΚΗ  
    SET ΤΙΜΗ_ΛΙΑΝ=ΤΙΜΗ_ΛΙΑΝ*(1+@ΠΟΣΟΣΤΟ/100)
```

### Παράδειγμα2

```
CREATE PROCEDURE ΑΥΞΗΣΗ_bathmon ( @ΠΟΣΟ FLOAT=1)  
AS UPDATE [ΒΑΘΜΟΙ ΣΠΟΥΔΑΣΤΩΝ]  
    SET ΒΑΘΜΟΣ=ΒΑΘΜΟΣ+@ΠΟΣΟ
```

# Αποθηκευμένες Διαδικασίες

## Εκτέλεση Αποθηκευμένων Διαδικασιών

### ΣΥΝΤΑΞΗ

```
EXEC [UTE] [ @επιστρ_κατάσταση = ] όνομα διαδικασίας  
[ ; αριθμός ]  
{ [ [ @παράμετρος1 = ] τιμή |  
[ @παράμετρος1 = ] @μεταβλητή [ ή [ OUTPUT ] ] } ..  
[ WITH RECOMPILE ]
```

# Αποθηκευμένες Διαδικασίες

## Εκτέλεση Αποθηκευμένων Διαδικασιών

### Παράδειγμα 1

```
EXECUTE ΑΥΞΗΣΗ_ΛΙΑΝ_ΤΙΜΗΣ 15
```

```
EXECUTE ΑΥΞΗΣΗ_bathmon 3
```

# Αποθηκευμένες Διαδικασίες

## Εκτέλεση Αποθηκευμένων Διαδικασιών

**EXECUTE** Αποθ\_Διαδικασία Παράμετρος\_Εισόδου

Η εντολή αυτή εκτελείται είτε μέσα από μια γλώσσα προγραμματισμού προγραμματίζοντας ο TEXT ενός dataSet

ή

από την επιλογή Query του SQL Server

---

# **Συναρτήσεις που Ορίζονται από τον Χρήστη**

## **User Defined Functions**

# Συναρτήσεις που Ορίζονται από τον Χρήστη

Στις γλώσσες προγραμματισμού υπάρχουν γενικά δύο τύποι ρουτινών: **Διαδικασίες & Συναρτήσεις**

Οι διαδικασίες αποτελούνται από αρκετές προτάσεις που έχουν μηδέν ή περισσότερες παραμέτρους εισόδου, αλλά δεν επιστρέφουν παραμέτρους εξόδου. Σε αντίθεση με αυτό, οι συναρτήσεις γενικά επιστρέφουν μια ή περισσότερες παραμέτρους.

Σημείωση Όπως θα δείτε παρακάτω, οι συναρτήσεις SQL Server δεν υποστηρίζουν παραμέτρους εξόδου, αλλά επιστρέφουν μια μόνο τιμή δεδομένων.

# Συναρτήσεις που Ορίζονται από τον Χρήστη

## Δημιουργία και Εκτέλεση των Συναρτήσεων

```
CREATE FUNCTION [ owner_name. ] function_name
    ( [ { @parameter_name [AS] scalar_parameter_data_type [ = default ] } [ ,...n ] ]
    )
    RETURNS @return_variable TABLE < table_type_definition >
    [ WITH < function_option > [ [,] ...n ] ]
    [ AS ]
    BEGIN
        function_body
        RETURN
    END
< function_option > ::=
    { ENCRYPTION | SCHEMABINDING }
< table_type_definition > ::=
    ( { column_definition | table_constraint } [ ,...n ] )
```

# Συναρτήσεις που Ορίζονται από τον Χρήστη

**CREATE FUNCTION** [*ιδιοκτήτης.*] *όνομα συνάρτησης*  
[(*@παράμετρος1* } *τύπος1* [= προεπιλογή1])]  
    ((*@παράμετρος2* } *τύπος2* [= προεπιλογή2] ))}...

**RETURNS** [*βαθμωτός τύπος* | ©*μεταβλητή*] **TABLE**}

[**WITH** {**ENCRYPTION** | **SCHEMABINDING**} [**AS**] {*μπλοκ* |  
**RETURN** (*πρόταση*)}



# Συναρτήσεις που Ορίζονται από τον Χρήστη

**Ιδιοκτήτης** είναι το όνομα του χρήστη στον οποίο έχει εκχωρηθεί η ιδιοκτησία της συνάρτησης που έχει οριστεί από τον χρήστη.

**Όνομα\_συνάρτησης** είναι το όνομα της νέας συνάρτησης.

**@παράμετρος1, @παράμετρος2,...** παράμετροι εισόδου

**τύπος1, τυπος2, ...** καθορίζουν τους τύπους δεδομένων.

Οι παράμετροι είναι τιμές που περνούν από τον καλούντα την συνάρτηση που δημιουργείται από τον χρήστη και χρησιμοποιούνται μέσα στην συνάρτηση.

# Συναρτήσεις που Ορίζονται από τον Χρήστη

**προεπιλογή1, προεπιλογή2** καθορίζουν την προαιρετική προεπιλεγμένη τιμή της αντίστοιχης παραμέτρου, μπορεί επίσης να είναι NULL

**RETURNS** ορίζει ένα τύπο δεδομένων της τιμής που επιστρέφεται από την συνάρτηση που ορίζεται από τον χρήστη.

Αυτός ο τύπος δεδομένων μπορεί να είναι οποιοσδήποτε από τους πρότυπους τύπους δεδομένων που υποστηρίζονται από το SQL Server, περιλαμβανομένου και του τύπου δεδομένων TABLE.

(Οι μόνοι πρότυποι τύποι δεδομένων που δεν μπορείτε να χρησιμοποιήσετε είναι ο τύπος δεδομένων TIMESTAMP και τα δεδομένα τύπου text/image.)

**WITH ENCRYPTION** κρυπτογραφεί τις στήλες του πίνακα συστήματος που περιέχουν το κείμενο της πρότασης CREATE FUNCTION

**WITH SCHEMABINDING**, δεσμεύει την συνάρτηση που ορίζεται από τον χρήστη στα αντικείμενα βάσης δεδομένων στα οποία αναφέρεται.

Κάθε προσπάθεια τροποποίησης της δομής του αντικειμένου βάσης δεδομένων στο οποίο αναφέρεται η συνάρτηση αποτυγχάνει.

# Συναρτήσεις που Ορίζονται από τον Χρήστη

---

**μπλοκ** είναι το μπλοκ **BEGIN-END** που περιέχει την υλοποίηση της συνάρτησης.

Η τελική πρόταση του μπλοκ πρέπει να είναι μια πρόταση **RETURN** με ένα όρισμα.

Η τιμή του ορίσματος είναι η τιμή που επιστρέφεται από την συνάρτηση.

# Συναρτήσεις που Ορίζονται από τον Χρήστη

Στο σώμα ενός μπλοκ **BEGIN-END**, μόνο οι παρακάτω προτάσεις επιτρέπονται:

- Προτάσεις εκχώρησης σαν την **SET**
- Προτάσεις ελέγχου ροής σαν τις **WHILE** και **IF**
- Προτάσεις **DECLARE** που ορίζουν τοπικές μεταβλητές δεδομένων
- Προτάσεις **SELECT** που περιέχουν λίστες **SELECT** με εκφράσεις που εκχωρούν σε μεταβλητές που είναι τοπικές στην συνάρτηση
- Προτάσεις **INSERT**, **UPDATE** και **DELETE**, που τροποποιούν μεταβλητές τύπου δεδομένων **TABLE**, που είναι τοπικές στην συνάρτηση

# Συναρτήσεις που Ορίζονται από τον Χρήστη

**Αυτή η συνάρτηση υπολογίζει πρόσθετα συνολικά κόστη που προκύπτουν αν αυξηθούν οι προϋπολογισμοί των έργων**

```
CREATE FUNCTION computecosts (@percent REAL =10)
RETURNS DECIMAL(14,2)
AS
BEGIN
    DECLARE @additional_costs DEC(16,2),
            @sum_budget DEC(6,2)
    SELECT @sum_budget = (select SUM (budget) FROM
                                                                    project)
    SET @additional_costs = @sum_budget *
                            @percent/100
    RETURN @additional_costs
END
```

# Συναρτήσεις που Ορίζονται από τον Χρήστη

**ΠΑΡΑΔΕΙΓΜΑ** συνάρτησης που επιστρέφει μια μεταβλητή τύπου **TABLE**.

```
CREATE FUNCTION employees_in_project (@pr_number CHAR(4))  
RETURNS TABLE AS
```

```
Begin
```

```
RETURN
```

```
    (SELECT empfname, empLname  
       FROM workson, employee  
       WHERE employee.empno = workson.empno  
             AND projectno = @pr_number)
```

```
End
```

- Η συνάρτηση **employees\_in\_project** χρησιμοποιείται για να εμφανίσει ονόματα όλων των υπαλλήλων που ανήκουν σε ένα συγκεκριμένο έργο.
- Η παράμετρος εισόδου **@pr\_number** καθορίζει ένα αριθμό έργου.
- Ενώ η συνάρτηση γενικά επιστρέφει πολλές γραμμές,
- Η φράση **RETURNS** περιέχει τον τύπο δεδομένων **TABLE**.

# Συναρτήσεις που Ορίζονται από τον Χρήστη

**ΠΑΡΑΔΕΙΓΜΑ** συνάρτησης που επιστρέφει μια μεταβλητή τύπου **TABLE**  
όπου

**Φαίνεται η χρήση της συνάρτησης employees\_in\_project.**

```
SELECT * FROM employees_in_project('p3')
```

**Το αποτέλεσμα είναι**

<b><u>fname</u></b>	<b><u>Lname</u></b>
<b>aaa</b>	<b>aaaa</b>
<b>bbbb</b>	<b>bbbb</b>
<b>cccc</b>	<b>cccc</b>



# Συναρτήσεις που Ορίζονται από τον Χρήστη

**ALTER FUNCTION**, τροποποιεί την δομή μιας συνάρτησης που ορίζεται από τον χρήστη. Η πρόταση ALTER FUNCTION χρησιμοποιείται συνήθως για να καταργήσει μια δέσμευση σχήματος. Όλες οι επιλογές της πρότασης ALTER FUNCTION αντιστοιχούν στις επιλογές με το ίδιο όνομα στην πρόταση CREATE FUNCTION.

**DROP FUNCTION** καταργείται μια συνάρτηση που ορίζεται από τον χρήστη. Μόνο ο ιδιοκτήτης της συνάρτησης (ή τα μέλη των σταθερών ρόλων βάσης δεδομένων db\_owner και sysadmin) μπορούν να καταργήσουν την συνάρτηση.

Ο πίνακας συστήματος **sysobjects** εμπλουτίζεται στο SQL Server έτσι ώστε να μπορεί να εμφανίσει τις πληροφορίες για υπάρχουσες συναρτήσεις που ορίζονται από τον χρήστη. Η διαδικασία συστήματος **sp\_helptext** παρέχει επίσης σχετικές πληροφορίες για συναρτήσεις που ορίζονται από τον χρήστη.

# ΣΚΑΝΔΑΛΕΣ (Ερεθισμοί)- Triggers

Μία σκανδάλη ή αλλιώς ένας ερεθισμός είναι ένας μηχανισμός που καλείται όταν συμβαίνει μια συγκεκριμένη ενέργεια σε ένα συγκεκριμένο πίνακα και εκτελεί δέσμες εντολών που αφορούν το συγκεκριμένο πίνακα ή άλλους πίνακες. Κάθε ερεθισμός έχει τρία γενικά μέρη:

## **Ένα όνομα - Μια ενέργεια - Την εκτέλεση**

Η ενέργεια ενός ερεθισμού μπορεί να είναι μια πρόταση INSERT, UPDATE ή DELETE. Το τμήμα εκτέλεσης μιας σκανδάλης περιέχει συνήθως μια αποθηκευμένη διαδικασία ή μια δέσμη εντολών.

Μία σκανδάλη δημιουργείται χρησιμοποιώντας την πρόταση CREATE TRIGGER, που έχει την παρακάτω μορφή:

# ΣΚΑΝΔΑΛΕΣ (Ερεθισμοί)- Triggers

```
CREATE TRIGGER trigger name ON tablename | viewname
{FOR | AFTER | INSTEAD OF} { [INSERT] [,] [UPDATE] [,]
[DELETE] }
[WITH ENCRYPTION] AS {batch | IF UPDATE(column)
[ {AND|OR} UPDATE(column) ] batch}
```

**trigger\_name** είναι το όνομα του ερεθισμού. **table\_name** είναι το όνομα του πίνακα για τον οποίο καθορίζεται ο ερεθισμός. Στον SQL Server 2000, μπορείτε επίσης να ορίσετε ερεθισμούς για όψεις/views.

**ON tablename | viewname** μια σκανδάλη μπορεί να εφαρμόζεται μόνο σε ένα πίνακα ή όψη.

**AFTER** και **INSTEAD OF** είναι δύο πρόσθετες επιλογές, που μπορείτε να ορίσετε σε μια σκανδάλη.

**FOR** είναι συνώνυμη της AFTER. Η AFTER ξεκινά την εκτέλεση της δέσμης εντολών μετά την εκκίνηση της σκανδάλης.

**INSTEAD OF** εκτελούνται αντί του αντίστοιχου συμβάντος INSERT, UPDATE ή DELETE της σκανδάλης.

# ΣΚΑΝΔΑΛΕΣ (Ερεθισμοί)- Triggers

- Σε κάθε εκτέλεση μιας σκανδάλης δημιουργούνται δύο εικονικοί πίνακες με ειδικά ονόματα. Ο πίνακας **deleted** και ο πίνακας **inserted**.
- Η δομή αυτών των πινάκων είναι ισοδύναμη με την δομή του πίνακα στον οποίο αναφέρεται η σκανδάλη. Ο πίνακας **deleted** περιέχει αντίγραφα από τις γραμμές που διαγράφονται από τον πίνακα που έχει εφαρμοσθεί η σκανδάλη.
- Το ίδιο ισχύει και για τον πίνακα **inserted** όπου δημιουργούνται αντίγραφα των γραμμών που εισάγονται στον πίνακα που έχει εφαρμοσθεί η σκανδάλη.

# ΣΚΑΝΔΑΛΕΣ (Ερεθισμοί)- Triggers

- Αν στην σκανδάλη υπάρχει μια εντολή UPDATE, τότε στον πίνακα **deleted** μπαίνουν τα δεδομένα πριν από την τροποποίηση και στον πίνακα **inserted** μπαίνουν δεδομένα μετά την τροποποίηση.
- Ο πίνακας **deleted** χρησιμοποιείται αν υπάρχει η εντολή **DELETE** ή **UPDATE** μέσα στην CREATE TRIGGER. Ο πίνακας **inserted** χρησιμοποιείται υπάρχει η εντολή **INSERT** ή **UPDATE** μέσα στην CREATE TRIGGER.

# ΠΑΡΑΔΕΙΓΜΑ Create trigger

- 8. Δημιουργήστε έναν πίνακα με όνομα LOG\_AUTO στον οποίο με την ενεργοποίηση μιας σκανδάλης (TRIGGER) που θα εφαρμόζεται πάνω στον βασικό πίνακα ΑΥΤΟΚΙΝΗΤΑ, θα αποθηκεύονται οι τροποποιήσεις που θα γίνονται στο πεδίο ΤΙΜΗ\_ΕΝΟΙΚΙΑΣΙΣ του πίνακα ΑΥΤΟΚΙΝΗΤΑ.

```
CREATE TABLE LOG_AUTO( AUTO_NUM VARCHAR(10),
                        USER_ID VARCHAR(20),
                        ΗΜΕΡΟΜΗΝΙΑ DATETIME,
                        OLD FLOAT,
                        NEW FLOAT)
```

```
CREATE TRIGGER TRIG_LOG ON ΑΥΤΟΚΙΝΗΤΑ
AFTER UPDATE AS
IF UPDATE(TΙΜΗ_ΕΝΟΙΚΙΑΣΙΣ)
BEGIN
DECLARE @ΤΙΜΗ_old FLOAT
DECLARE @ΤΙΜΗ_new FLOAT
DECLARE @auto_number VARCHAR(10)
SELECT @ΤΙΜΗ_old = (SELECT ΤΙΜΗ_ΕΝΟΙΚΙΑΣΙΣ FROM deleted)
SELECT @ΤΙΜΗ_new = (SELECT ΤΙΜΗ_ΕΝΟΙΚΙΑΣΙΣ FROM inserted)
SELECT @auto_number = (SELECT ΚΑ FROM deleted)
INSERT INTO LOG_AUTO VALUES
(@auto_number,USER_NAME(),GETDATE(),@ΤΙΜΗ_old, @ΤΙΜΗ_new)
END
```

```
UPDATE ΑΥΤΟΚΙΝΗΤΑ
SET ΤΙΜΗ_ΕΝΟΙΚΙΑΣΙΣ=200 WHERE ΚΑ=1'
```

- Τα περιεχόμενα του πίνακα LOG\_AUTO θα είναι ως εξής:

<b>AUTO_NUM</b>	<b>USER_ID</b>	<b>ΗΜΕΡΟΜΗΝΙΑ</b>	<b>OLD</b>	<b>NEW</b>
1	200,00	dbo	2008-06-06 11:51	500,00