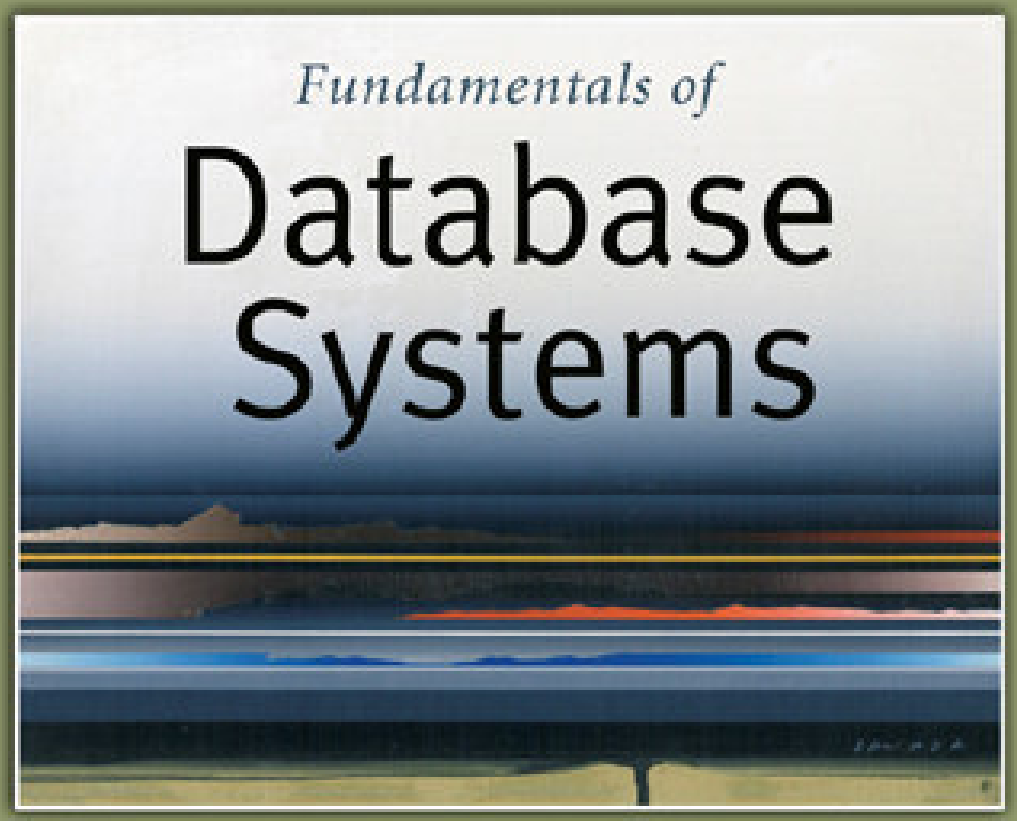
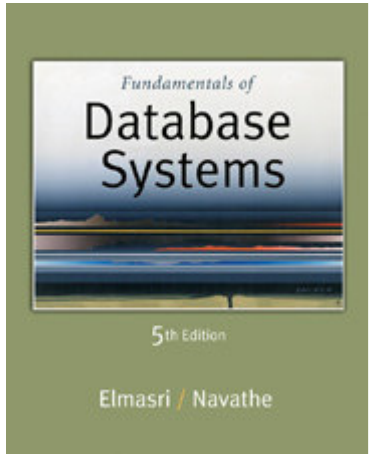


5th Edition

Elmasri / Navathe

Κεφάλαιο 9

Εισαγωγή στις Τεχνικές Προγραμματισμού SQL



Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Διαυλος

Περιεχόμενα

- 9.1 Γενικοί Περιορισμοί σαν Βεβαιώσεις
- 9.2 Όψεις στην SQL
- 9.3 Προγραμματισμός Βάσεων Δεδομένων
- 9.4 Ενσωματωμένη SQL
- 9.5 Κλήσεις Συναρτήσεων, SQL/CLI
- 9.6 Αποθηκευμένες Διαδικασίες, SQL/PSM
- 9.7 Σύνοψη

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Στόχοι

- Ορισμός πιο γενικευμένων **περιορισμών** μέσω βεβαιώσεων
- Δυνατότητες ορισμού όψεων (εικονικοί πίνακες) στην SQL
- Τεχνικές για προσπέλαση και διαχείριση μιας βάσης δεδομένων μέσω προγραμμάτων σε γλώσσες προγραμματισμού γενικού σκοπού
  - Π.χ., Java, C++, κλπ.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Περιορισμοί σαν βεβαιώσεις

- Γενικοί περιορισμοί: περιορισμοί που δεν ανήκουν στις βασικές κατηγορίες της SQL
- Μηχανισμός: **`CREAT ASSERTION`**
  - Περιλαμβάνει:
    - ένα όνομα περιορισμού,
    - ακολουθούμενο από την `CHECK`,
    - ακολουθούμενο από μια συνθήκη

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

βεβαιώσεις: Παράδειγμα

- "Ο μισθός ενός εργαζομένου δεν πρέπει να ξεπερνά το μισθό του διευθυντή του τμήματος για το οποίο εργάζεται"

Όνομα περιορισμού, συνθήκη CHECK,

```
CREAT ASSERTION SALARY_CONSTRAINT
CHECK (NOT EXISTS (SELECT *
        FROM ΕΡΓΑΖΟΜΕΝΟΣ E M,
        ΤΜΗΜΑ D
        WHERE E.ΜΙΣΘΟΣ > M.ΜΙΣΘΟΣ AND
              E.ΑΡΙΘ_Τ=D.ΚΩΔ_ΤΜΗΜ AND
                    D.ΔΙΕΥΘΥΝΤΗΣ=M.ΑΡ_ΤΑΥΤ))
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Χρήση Γενικών Βεβαιώσεων

- Προσδιορίζουν μια ερώτηση που παραβιάζει την συνθήκη· περιλαμβάνει μια `NOT EXISTS` πρόταση
- Το αποτέλεσμα της ερώτησης πρέπει να είναι κενό
  - αν το αποτέλεσμα της ερώτησης δεν είναι κενό, έχει παραβιασθεί η βεβαίωση

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Σκανδάλες στην SQL

- Στόχος: παρακολούθηση μιας βάσης δεδομένων και εκκίνηση ενεργειών όταν συμβεί μια συνθήκη
- Οι σκανδάλες εκφράζονται με μια σύνταξη παρόμοια με τις βεβαιώσεις και περιλαμβάνουν τα:
  - Γεγονός
    - Όπως μια πράξη insert, delete, ή update
  - Συνθήκη
  - Δράση
    - Ενεργοποιείται όταν ικανοποιηθεί η συνθήκη

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Σκανδάλες στην SQL : Ένα Παράδειγμα

- Μια σκανδάλη που συγκρίνει το μισθό ενός εργαζομένου (κατά την εισαγωγή ή την τροποποίηση) με αυτόν τιου προϊσταμένου του :

```
CREATE TRIGGER INFORM_SUPERVISOR
BEFORE INSERT OR UPDATE OF
   ΜΙΣΘΟΣ, SUPERVISOR_SSN ON ΕΡΓΑΖΟΜΕΝΟΣ
   FOR EACH ROW
       WHEN
       (NEW.ΜΙΣΘΟΣ> (SELECT ΜΙΣΘΟΣ FROM ΕΡΓΑΖΟΜΕΝΟΣ
                     WHERE AP_TAYT=NEW.SUPERVISOR_SSN))
       INFORM_SUPERVISOR (NEW.SUPERVISOR_SSN,NEW.SSN);
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Όψεις στην SQL

- Μια όψη είναι ένας “εικονικός” πίνακας που παράγεται από άλλους πίνακες
- Επιτρέπονται περιορισμένες ενημερώσεις
  - Εφόσον ο πίνακας δεν αποθηκεύεται φυσικά
- Υποστηρίζει πλήρεις πράξεις ερωτήσεων
- Αποτελεί διευκόλυνση στις εκφράσεις κάποιων πράξεων

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Ορισμός Όψεων

- Εντολή SQL : **`CREATE VIEW`**
  - όνομα ενός πίνακα (όψης)
  - μια πιθανή λίστα από ονόματα γνωρισμάτων (για παράδειγμα, όταν ορίζονται αριθμητικές πράξεις ή όταν θέλουμε να είναι διαφορετικά τα ονόματα των γνωρισμάτων από αυτά των βασικών σχέσεων)
  - μια ερώτηση που προσδιορίζει τα περιεχόμενα των πινάκων

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Ορισμός Όψεων : Ένα Παράδειγμα

- Ορισμός ενός διαφορετικού πίνακα ΑΠΑΣΧΟΛΗΣΗ

```
CREATE VIEW ΝΕΑ_ΑΠΑΣΧΟΛΗΣΗ AS
SELECT ΟΝΟΜΑ, ΕΠΙΘΕΤΟ, Ε_ΟΝΟΜΑ, ΩΡΕΣ
    FROM ΕΡΓΑΖΟΜΕΝΟΣ, ΕΡΓΟ, ΑΠΑΣΧΟΛΗΣΗ
    WHERE ΑΡ_ΤΑΥΤ=Ε_ΑΡΤΑΥΤ AND
          Κ_ΕΡΓΟ=ΚΩΔ_ΕΡΓΟΥ
    GROUP BY Ε_ΟΝΟΜΑ;
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Χρήση ενός Εικονικού Πίνακα

- Μπορούμε να ορίσουμε SQL ερωτήσεις σε μια όψη:

```
SELECT ΟΝΟΜΑ, ΕΠΙΘΕΤΟ
    FROM ΝΕΑ_ΑΠΑΣΧΟΛΗΣΗ
    WHERE Ε_ΟΝΟΜΑ='Seena';
```

- Όταν δεν χρειαζόμαστε μια όψη μπορούμε να την διαγράψουμε:

```
DROP ΝΕΑ_ΑΠΑΣΧΟΛΗΣΗ;
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Αποτελεσματική Υλοποίηση Όψεων

- Τροποποίηση ερώτησης:
  - Παρουσίαση της ερώτησης της όψης σε σχέση με τους βασικούς πίνακες
- Μειονέκτημα:
  - Δεν είναι αποτελεσματική για όψεις που έχουν ορισθεί μέσω πολύπλοκων ερωτήσεων
    - Ειδικότερα, αν (σε σύντομο χρονικό διάστημα) θα εφαρμοσθούν επιπλέον ερωτήσεις στην όψη

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Αποτελεσματική Υλοποίηση Όψεων

- Υλοποίηση όψεων:
  - Περιλαμβάνει φυσική δημιουργία και προσωρινή αποθήκευση πινάκων
- Υπόθεση:
  - Θα ακολουθήσουν άλλες ερωτήσεις σε όψεις
- Προβληματισμοί:
  - Διατήρηση της αντιστοιχίας μεταξύ βασικού πίνακα και όψης όταν ενημερώνεται ο βασικός πίνακας
- Στρατηγική:
  - Αυξητική ενημέρωση

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Ενημέρωση Όψεων

- Ενημέρωση μιας απλής όψης χωρίς συναθροιστικές πράξεις:
  - Η ενημέρωση μπορεί να απεικονισθεί σε μια ενημέρωση των βασικών πινάκων
- Όψεις που περιλαμβάνουν συνενώσεις:
  - Μια ενημέρωση *μπορεί* να απεικονισθεί σε ενημέρωση των βασικών σχέσεων
    - Δεν είναι πάντοτε εφικτή

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Όψεις που δεν μπορούν να ενημερωθούν

- Όψεις που ορίζονται σε ομάδες και συναθροιστικές συναρτήσεις δεν δύνανται να ενημερωθούν
- Όψεις που ορίζονται σε πολλαπλούς πίνακες με χρήση συνενώσεων γενικά δεν μπορούν να ενημερωθούν
- **`WITH CHECK OPTION`**: πρέπει να προστεθεί στον ορισμό μιας όψης αν η όψη πρόκειται να ενημερωθεί
  - Για να επιτραπεί έλεγχος δυνατότητας ενημέρωσης και να σχεδιασθεί ένα πλάνο για μια στρατηγική εκτέλεσης

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Προγραμματισμός Βάσεων Δεδομένων

- Στόχος:
  - Προσπέλαση μιας βάσης δεδομένων από ένα πρόγραμμα εφαρμογής (αντί μιας διαδραστικής διεπαφής)
- Γιατί?
  - Μια διαδραστική διεπαφή αν και εξυπηρετεί δεν είναι αρκετή
    - Η πλειοψηφία των πράξεων σε βάσεις δεδομένων γίνεται μέσω προγραμμάτων εφαρμογών (με αυξανόμενες τις web εφαρμογές)

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Προγραμματισμός Βάσεων Δεδομένων

- Εμφυτευμένες εντολές:
  - Οι εντολές της βάσης δεδομένων εμφυτεύονται σε μια προγραμματιστική γλώσσα γενικού σκοπού
- Βιβλιοθήκη λειτουργιών βάσεων δεδομένων :
  - Διαθέσιμη στη φιλόξενη γλώσσα για κλήσεις από τη βάση δεδομένων· γνωστό σαν *API*
    - *API* Application Program Interface
- Μια εντελώς νέα, πλήρης γλώσσα
  - Ελαχιστοποιεί το πρόβλημα της ασυμφωνίας δομών

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Ασυμφωνία δομών

- Μη ύπαρξη συμβατότητας μεταξύ της γλώσσας προγραμματισμού και του μοντέλου της βάσης δεδομένων, π.χ.,
  - Ασυμφωνία τύπου και ασυμβατότητες· απαιτεί νέα σύνδεση για κάθε γλώσσα
  - επεξεργασία σύνολο έναντι μιας-μιας εγγραφής
    - απαιτούνται ειδικές μεταβλητές επανάληψης που να διατρέχουν τα αποτελέσματα της ερώτησης και να διαχειρίζονται ατομικές τιμές

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Βήματα στον Προγραμματισμό Βάσεων Δεδομένων

1. Το πρόγραμμα του πελάτη *ανοίγει μια σύνδεση* στον εξυπηρέτη της βάσης δεδομένων
2. Το πρόγραμμα του πελάτη *υποβάλλει ερωτήσεις ή/και ενημερώσεις* στη βάση δεδομένων
3. Όταν δεν χρειάζεται άλλο προσπέλαση στη βάση δεδομένων, το πρόγραμμα του πελάτη *κλείνει (τερματίζει) τη σύνδεση*

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Εμφυτευμένη SQL

- Οι περισσότερες εντολές της SQL μπορούν να εμφυτευθούν σε γενικού σκοπού *φιλόξενη* γλώσσα προγραμματισμού όπως COBOL, C, Java
- Μια εμφυτευμένη SQL εντολή διακρίνεται από τις εντολές της φιλόξενης γλώσσας με εγκλεισμό μεταξύ **EXEC SQL** ή **EXEC SQL BEGIN** και κλείσιμο με **END-EXEC** ή **EXEC SQL END** (ή ερωτηματικό)
  - Η σύνταξη μπορεί να διαφέρει ανάλογα με τη γλώσσα
  - *Οι κοινές μεταβλητές* (αυτές που χρησιμοποιούνται και στις δύο γλώσσες) συνήθως έχουν πρόθεμα τις δύο τελείες (:) στην SQL

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Παράδειγμα: Ορισμός Μεταβλητής στη Γλώσσα C

- Οι μεταβλητές μέσα σε μια **DECLARE** είναι κοινές και μπορούν να εμφανισθούν (ενώ έχουν πρόθεμα το :) σε SQL εντολές
- **Η SQLCODE** χρησιμοποιείται για επικοινωνία λαθών/εξαιρέσεων μεταξύ της βάσης δεδομένων και του προγράμματος

```
int loop;
EXEC SQL BEGIN DECLARE SECTION;
    varchar τ_ονομα[16], ονομα[16], …;
    char αρ_ταυτ[10], ημ_γεν[11], …;
    int αριθ_τ, κωδ_τμημ, SQLCODE, …;
EXEC SQL END DECLARE SECTION;
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Η επικοινωνία μεταξύ του προγράμματος και του ΣΔΒΔ γίνεται με χρήση των SQLCODE και SQLSTATE

Η μεταβλητή SQLCODE είναι μια ακέραια μεταβλητή. Μετά την εκτέλεση κάθε εντολής στη βάση δεδομένων το ΣΔΒΔ επιστρέφει μια τιμή στην SQLCODE. Η τιμή 0 σημαίνει ότι η εντολή εκτελέσθηκε με επιτυχία από το ΣΔΒΔ. Αν SQLCODE>0 (ή συγκεκριμένα αν SQLCODE=100), αυτό δείχνει δεν υπάρχουν επιπλέον διαθέσιμα δεδομένα (εγγραφές) στο αποτέλεσμα της επερώτησης. Αν SQLCODE<0, αυτό δείχνει ότι έχει συμβεί κάποιο λάθος.

Σε μερικά συστήματα -για παράδειγμα στο ΣΣΔΒΔ ORACLE- η SQLCODE είναι ένα πεδίο μιας εγγραφής που ονομάζεται SQLCA (SQL communication area), και η αναφορά σ' αυτήν γίνεται με την SQLCA.SQLCODE. Στην περίπτωση αυτή θα πρέπει στο πρόγραμμα C να συμπεριληφθεί ο ορισμός της SQLCA, με την παρακάτω εντολή:

EXEC SQL include SQLCA;

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Σε μεταγενέστερες εκδόσεις του πρότυπου της SQL, προστέθηκε μια μεταβλητή επικοινωνίας που ονομάζεται SQLSTATE, και είναι μια συμβολοσειρά. Μια τιμή “00000” της SQLSTATE δείχνει ότι δεν συνέβησαν λάθη ή εξαιρέσεις. Άλλες τιμές δείχνουν διάφορα λάθη ή εξαιρέσεις. Για παράδειγμα όταν χρησιμοποιείται η SQLSTATE, τιμή “02000” δείχνει “δεν υπάρχουν άλλα δεδομένα”. Προς το παρόν το πρότυπο της SQL υποστηρίζει και την SQLSTATE και την SQLCODE. Υποτίθεται ότι πολλοί από τους κώδικες λαθών και εξαίρεσης που επιστρέφει η SQLSTATE έχουν τυποποιηθεί για όλους τους κατασκευαστές SQL και όλες τις πλατφόρμες.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

SQL Εντολές για Σύνδεση με μια Βάση Δεδομένων

- Σύνδεση (είναι δυνατές πολλαπλές συνδέσεις αλλά μόνο μια είναι ενεργή)

```
CONNECT TO server-name AS connection-name
AUTHORIZATION user-account-info;
```

- Αλλαγή από μια ενεργή σύνδεση σε μια άλλη

```
SET CONNECTION connection-name;
```

- Αποσύνδεση

```
DISCONNECT connection-name;
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Εμφυτευμένη SQL σε C Παραδείγματα

```
loop = 1;
while (loop) {
  prompt ("Enter αρ_ταυτ: ", αρ_ταυτ);
  EXEC SQL
      select ΟΝΟΜΑ, ΕΠΙΘΕΤΟ, ΔΙΕΥΘΥΝΣΗ, ΜΙΣΘΟΣ
      into :ονομα, :επιθετο, :διευθυνση, :μισθος
      from ΕΡΓΑΖΟΜΕΝΟΣ where ΑΡ_ΤΑΥΤ == :αρ_ταυτ;
      if (SQLCODE == 0) printf(ονομα, …);
      else printf("Δεν υπάρχει ο αρ ταυτ:", αρ_ταυτ);
      prompt("άλλος ΑΡ_ΤΑΥΤ? (1=yes, 0=no): ", loop);

}
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Εμφυτευμένη SQL σε C Παραδείγματα

- Για πολλές πλειάδες χρειάζεται ένας **δρομέας** (**cursor)** (ή μεταβλητή επανάληψης-iterator)
- Οι εντολές **`FETCH`** μετακινούν τον δρομέα στην move *επόμενη* πλειάδα
- Η **`CLOSE CURSOR`** δείχνει ότι έχει ολοκληρωθεί η επεξεργασία των αποτελεσμάτων της ερώτησης

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Ανάκτηση πολλαπλών Πλειάδων-Χρήση Δρομέα (CURSOR)

Μπορούμε να θεωρήσουμε ένα δρομέα σαν ένα δείκτη που δείχνει μια πλειάδα (γραμμή) από το αποτέλεσμα μιας επερώτησης με πολλές πλειάδες. Ο δρομέας δηλώνεται όταν δηλώνεται στο πρόγραμμα η εντολή της SQL επερώτησης. Αργότερα στο πρόγραμμα, μια εντολή OPEN CURSOR φέρνει το αποτέλεσμα της επερώτησης από τη βάση δεδομένων και τοποθετεί τον δρομέα μια θέση πριν από την πρώτη γραμμή στο αποτέλεσμα της επερώτησης. Αυτή είναι και η τρέχουσα τιμή του δρομέα. Στη συνέχεια, το πρόγραμμα εκτελεί εντολές FETCH. Κάθε εντολή FETCH μετακινεί τον δρομέα στην επόμενη γραμμή του αποτελέσματος της επερώτησης, την οποία και καθιστά τρέχουσα γραμμή και αντιγράφει τις τιμές των γνωρισμάτων στις μεταβλητές του προγράμματος C (φιλόξενη γλώσσα) που προσδιορίζονται στην εντολή FETCH με την πρόταση INTO. Η μεταβλητή του δρομέα είναι βασικά μια μεταβλητή επανάληψης που διατρέχει τις πλειάδες του αποτελέσματος της επερώτησης –μια πλειάδα κάθε φορά. Αυτό μοιάζει με την επεξεργασία μιας-μιας εγγραφής από ένα αρχείο

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

```
0)        prompt("Δώστε το όνομα του τμήματος: ", τ_ονομα);
1)        EXEC SQL
2)        select Κωδ_Τμημ into :κωδ_τμημ
3)         from ΤΜΗΜΑ where Τ_ΟΝΟΜΑ = :τ_ονομα;
4)        EXEC SQL DECLARE ΕΡΓ CURSOR FOR
5)        select ΑΡ_ΤΑΥΤ, ΟΝΟΜΑ, ΑΡΧ_ΠΑΤ, ΕΠΙΘΕΤΟ, ΜΙΣΘΟΣ
6)        from ΕΡΓΑΖΟΜΕΝΟΣ where ΑΡΙΘ_Τ=:κωδ_τμημ
7)        FOR UPDATE OF ΜΙΣΘΟΣ;
8)        EXEC SQL OPEN ΕΡΓ ;
9)        EXEC SQL FETCH from ΕΡΓ into :αρ_ταυτ, :ονομα, :αρχ_πατ, :επιθετο,
          :μισθος;
10)       while (SQLCODE ==0) {
11)        printf("Το όνομα του εργαζομένου είναι: ", ονομα, αρχ_πατ, επιθετο)
12)       prompt("Δώστε το ποσό αύξησης: ", αυξηση);
13)       EXEC SQL
14)          update ΕΡΓΑΖΟΜΕΝΟΣ
15)         set ΜΙΣΘΟΣ = ΜΙΣΘΟΣ + :αυξηση
16)         where CURRENT OF ΕΡΓ ;
17)       EXEC SQL FETCH from ΕΡΓ into :αρ_ταυτ,  :ονομα, :αρχ_πατ, επιθετο,
          :μισθος
18)       }
19) EXEC SQL CLOSE ΕΡΓ;
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Ενημερώσεις

Όταν ορίζεται ένας δρομέας για γραμμές που πρόκειται να τροποποιηθούν (ενημερωθούν), πρέπει στον ορισμό του δρομέα να προσθέσουμε την πρόταση FOR UPDATE OF και την λίστα των ονομάτων των γνωρισμάτων που θα ενημερωθούν από το πρόγραμμα.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Ορισμός Δρομέα

DECLARE <όνομα δρομέα> [INSENSITIVE] [SCROLL] CURSOR

[WITH HOLD] FOR <ορισμός επερώτησης>

[ORDER BY <προσδιορισμός διάταξης>

[FOR READ ONLY | FOR UPDATE | OF <λίστα γνωρισμάτων>]];

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Χρήση της SCROLL

Όταν ορίζεται η προαιρετική λέξη κλειδί SCROLL στην δήλωση ενός δρομέα, είναι πιθανόν να τοποθετηθεί ο δρομέας κατά άλλους τρόπους από την καθαρά σειριακή προσπέλαση. Στην εντολή FETCH μπορεί να προστεθεί μια κατεύθυνση που η τιμή της μπορεί να είναι μια από τις NEXT, PRIOR, FIRST, LAST, ABSOLUTE i, και RELATIVE i. Στις δύο τελευταίες εντολές, η τιμή του i πρέπει να είναι ακέραια και προσδιορίζει μια απόλυτη θέση πλειάδας (ABSOLUTE) ή μια σχετική θέση πλειάδας (RELATIVE) σε σχέση με την τρέχουσα θέση του δρομέα.

FETCH [[<κατεύθυνση>] FROM] <όνομα δρομέα>
INTO <στη λίστα ανάκτησης>;

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Δυναμική SQL

- Στόχος:
  - Ο σχηματισμός και η εκτέλεση νέων (που δεν έχουν μεταφρασθεί από πριν) SQL εντολών κατά την εκτέλεση
    - ένα πρόγραμμα κατά την εκτέλεση του δέχεται SQL εντολές από το πληκτρολόγιο
    - με μια πράξη επιλογής και κλικ μεταφράζεται σε μια SQL ερώτηση
- Η δυναμική ενημέρωση είναι σχετικά απλή· μια δυναμική ερώτηση μπορεί να είναι πολύπλοκη
  - επειδή,κατά την μετάφραση, δεν είναι γνωστός ο τύπος και το πλήθος των γνωρισμάτων που θα ανακτηθούν

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Δυναμική SQL : Ένα παράδειγμα

```
EXEC SQL BEGIN DECLARE SECTION;
varchar sqlupdatestring[256];
EXEC SQL END DECLARE SECTION;
…
prompt ("Δώστε την εντολή ενημέρωσης:",
   sqlupdatestring);
EXEC SQL PREPARE sqlcommand FROM :sqlupdatestring;
EXEC SQL EXECUTE sqlcommand;
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Εμφυτευμένη SQL σε Java

- SQLJ: ένα πρότυπο για εμφύτευση της SQL σε Java
- Ένας SQLJ μεταφραστής μετατρέπει εντολές SQL σε Java
  - Αυτές εκτελούνται μέσω της διεπαφής *JDBC*
- Πρέπει να εισαχθεί ένα πλήθος κλάσεων
  - Π.χ., **`java.sql`**

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Συνδεσιμότητα Java Βάσεων Δεδομένων

- JDBC:
  - Κλήσεις από SQL της συνάρτησης σύνδεσης για Java προγραμματισμό
- Ένα πρόγραμμα Java με JDBC συναρτήσεις μπορεί να προσπελάσει κάθε σχεσιακό ΣΔΒΔ που διαθέτει JDBC
- Η JDBC επιτρέπει σε ένα πρόγραμμα να συνδέεται με πολλές βάσεις δεδομένων (γνωστών σαν *πηγών δεδομένων*)

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Βήματα για JDBC Προσπέλαση Βάσεων

1. Import την JDBC βιβλιοθήκη **`(java.sql.*)`**
2. Φόρτωση της JDBC: **`Class.forname("oracle.jdbc.driver.OracleDriver")`**
3. Ορισμός κατάλληλων μεταβλητών
4. Δημιουργία ενός αντικειμένου σύνδεσης (μέσω της **`getConnection`**)
5. Δημιουργία ενός αντικειμένου εντολής από την κλάση **`Statement`** :
   - 1. PreparedStatment 2. CallableStatement
6. Προσδιορισμός των παραμέτρων της εντολής (εμφανίζονται με ερωτηματικά)
7. Σύνδεση των παραμέτρων με μεταβλητές του προγράμματος
8. Εκτέλεση της SQLεντολής (αναφέρεται από ένα αντικείμενο) μέσω της JDBC **`executeQuery`**
9. Επεξεργασία των αποτελεσμάτων της ερώτησης (επιστρέφονται σε ένα αντικείμενο τύπου ResultSet)
   - **`ResultSet`** είναι ένας πίνακας 2 διαστάσεων

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Εμφύτευση SQL σε Java: Παράδειγμα

```
ssn = readEntry("Δώσε ένα ΑΡ_ΤΑΥΤ: ");
try {
   #sql{select ΟΝΟΜΑ,ΕΠΙΘΕΤΟ, ΔΙΕΥΘΥΝΣΗ, ΜΙΣΘΟΣ
   into :ονομα, :επιθετο, :διευθυνση, :μισθος
   from ΕΡΓΑΖΟΜΝΕΟΣ where ΑΡ_ΤΑΥΤ = :αρ_ταυτ};
}
catch (SQLException se) {
   System.out.println("ΑΡ_ΤΑΥΤ δεν υπάρχει:
   ",+αρ_ταυτ);
   return;
}
System.out.println(onoma + " " + epiueto + … );
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Πολλές πλειάδες στην SQLJ

- Η SQLJ υποστηρίζει δύο τύπους μεταβλητών επανάληψης:
  - *Μεταβλητή επανάληψης με όνομα*: σχετίζεται με το αποτέλεσμα μιας ερώτησης
  - *Μεταβλητή επανάληψης θέσης*: παραθέτει μόνο τους τύπους γνωρισμάτων του αποτελέσματος μιας ερώτησης
- Μια πράξη **FETCH** ανακτά την επόμενη πλειάδα του αποτελέσματος:

```
fetch iterator-variable into program-variable
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Προγραμματισμός Βάσεων Δεδομένων με Συναρτησιακές Κλήσεις

- Η εμφυτευμένη SQL υποστηρίζει στατικό προγραμματισμό βάσεων δεδομένων
- API: Δυναμικός προγραμματισμός βάσεων δεδομένων με μια βιβλιοθήκη συναρτήσεων
  - Πλεονέκτημα:
    - Δεν απαιτείται προ-επεξεργασία (επομένως μεγαλύτερη ευελιξία)
  - Μειονέκτημα:
    - Οι SQL έλεγχοι σύνταξης θα γίνουν κατά την εκτέλεση

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

SQL Call Level Interface

- Τμήμα του SQL πρότυπου
- Υποστηρίζει εύκολη προσπέλαση σε πολλές βάσεις δεδομένων από το ίδιο πρόγραμμα
- Πρέπει να είναι διαθέσιμες και να εγκατασταθούν διάφορες βιβλιοθήκες (π.χ., η **`sqlcli.h`** για τη C)
- Οι SQL εντολές δημιουργούνται δυναμικά και μεταβιβάζονται σαν παράμετροι συμβολοσειρές στις κλήσεις

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Στοιχεία του SQL/CLI

- *Εγγραφή περιβάλλοντος*:
  - Κρατάει πληροφορίες για τις συνδέσεις της βάσης δεδομένων
- *Εγγραφή σύνδεσης*:
  - Κρατάει πληροφορίες που απαιτούνται για μια σύνδεση
- *Εγγραφή εντολής*:
  - Κρατάει πληροφορίες που απαιτούνται για μια SQL εντολή
- *Εγγραφή Περιγραφής*:
  - Κρατάει πληροφορίες πλειάδων

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Βήματα για C και SQL/CLI Προγραμματισμό

1. Φόρτωση των βιβλιοθηκών SQL/CLI
2. Δήλωση μεταβλητών διαχείρισης εγγραφών για τα προηγούμενα στοιχεία (ονομάζονται: **SQLHSTMT, SQLHDBC, SQLHENV, SQLHDEC**)
3. Δημιουργία μιας εγγραφής περιβάλλοντος με χρήση της **SQLAllocHandle**
4. Δημιουργία μιας εγγραφής σύνδεσης με χρήση της **SQLAllocHandle**
5. Δημιουργία μιας εγγραφής εντολής με χρήση της **SQLAllocHandle**
6. Προετοιμασία μιας εντολής με χρήση της SQL/CLI συνάρτησης **SQLPrepare**
7. Σύνδεση των παραμέτρων με μεταβλητές του προγράμματος
8. Εκτέλεση της SQL εντολής μέσω της **SQLExecute**
9. Σύνδεση των στηλών της ερώτησης με μια C μεταβλητή μέσω της **SQLBindCol**
10. Χρήση της **SQLFetch** για ανάκτηση των τιμών των στηλών σε μεταβλητές C

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Αποθηκευμένες Διαδικασίες

- Μόνιμες διαδικασίες/συναρτήσεις αποθηκεύονται τοπικά και εκτελούνται από τον διακομιστή βάσης δεδομένων
  - Σε αντίθεση εκτέλεσης από τους πελάτες
- Πλεονεκτήματα:
  - Αν πολλές εφαρμογές χρειάζονται την διαδικασία, μπορεί να ενεργοποιηθεί από οποιαδήποτε από αυτές (μειώνοντας έτσι τις επαναλήψεις)
  - Εκτέλεση από ένα διακομιστή μειώνει το επικοινωνιακό κόστος
  - Επεκτείνει τις δυνατότητες μοντελοποίησης των όψεων
- Μειονεκτήματα:
  - Κάθε ΣΔΒΔ έχει την δική του σύνταξη και αυτό μειώνει την μεταφερσιμότητα

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Κατασκευαστές Αποθηκευμένων Διαδικασιών

- Μια αποθηκευμένη διαδικασία

```
CREATE PROCEDURE procedure-ονομα (παραμετροι)
τοπικές δηλώσεις
procedure-body;
```

- Μια αποθηκευμένη συνάρτηση

```
CREATE FUNCTION fun-ονομα (παράμετροι) RETRUNS
return-type
τοπικές δηλώσεις
function-body;
```

- Κλήση μιας διαδικασίας ή συνάρτησης

```
CALL procedure-name/fun-name (arguments);
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

SQL Persistent Stored Modules

- SQL/PSM:
  - Μέρος του SQL πρότυπου για σύνταξη μόνιμα αποθηκευμένων τμημάτων
- SQL + αποθηκευμένες διαδικασίες/συναρτήσεις + επιπλέον προγραμματιστικές δομές
  - Π.χ., εντολές διακλάδωσης και επανάληψης
  - Επέκταση των δυνατοτήτων της SQL

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

SQL/PSM: Παράδειγμα

```
CREATE FUNCTION DEPT_SIZE (IN τμημ_αρ INTEGER)
RETURNS VARCHAR[7]
DECLARE TOT_EMPS INTEGER;

SELECT COUNT (*) INTO TOT_EMPS
  FROM SELECT ΕΡΓΑΖΟΜΕΝΟΣ WHERE Τ_ΑΡΙΘ =
  τμημ_αρ;
IF TOT_EMPS > 100 THEN RETURN "HUGE"
  ELSEIF TOT_EMPS > 50 THEN RETURN "LARGE"
  ELSEIF TOT_EMPS > 30 THEN RETURN "MEDIUM"
  ELSE RETURN "SMALL"
ENDIF;
```

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος

Σύνοψη

- Οι βεβαιώσεις υποστηρίζουν τον προσδιορισμό επιπλέον περιορισμών
- Οι σκανδάλες είναι βεβαιώσεις που ορίζουν ενέργειες που πρέπει να γίνουν αυτόματα όταν συμβούν κάποιες συνθήκες
- Οι όψεις δημιουργούν προσωρινούς (εικονικούς) πίνακες
- Μια βάση δεδομένων μπορεί να προσπελάζεται διαδραστικά
- Ωστόσο, πιο συχνά, η επεξεργασία των δεδομένων μας βάσης γίνεται με προγράμματα εφαρμογών
- Διάφορες μέθοδοι προγραμματισμού βάσεων δεδομένων:
  - Εμφυτευμένη SQL
  - Δυναμική SQL
  - Αποθηκευμένες διαδικασίες και συναρτήσεις

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe Ελληνική Έκδοση Δίαυλος