Προηγμένα Θέματα Βάσεων Δεδομένων

Διδάσκων: Άγγελος Μιχάλας

Περιεχόμενα

- Stored Procedures

Εισαγωγή

- Μία stored procedure είναι ένα τμήμα από SQL εντολές αποθηκευμένες στον κατάλογο της ΒΔ.
- Μία stored procedure μπορεί να καλεστεί από **triggers**, άλλες **stored procedures**, και **εφαρμογές** όπως Java, Python, PHP.
- Τα περισσότερα ΣΔΒΔ υποστηρίζουν αναδρομικά stored procedures.
  - Η MySQL δεν υποστηρίζει.
- Προστέθηκαν από την MySQL έκδοση 5.0

Πλεονεκτήματα

- Οι stored procedures βοηθούν στην αύξηση της απόδοσης των εφαρμογών.
  - Αφού δημιουργηθούν, οι stored procedures μεταφράζονται (compiled) και απόθηκεύονται στη ΒΔ.
- Οι stored procedures βοηθούν στη μείωση της κίνησης ανάμεσα στην εφαρμογή και στον εξυπηρετητή της ΒΔ (database server)
  - Η εφαρμογή αποστέλλει μόνο το όνομα και τις παραμέτρους της stored procedure.

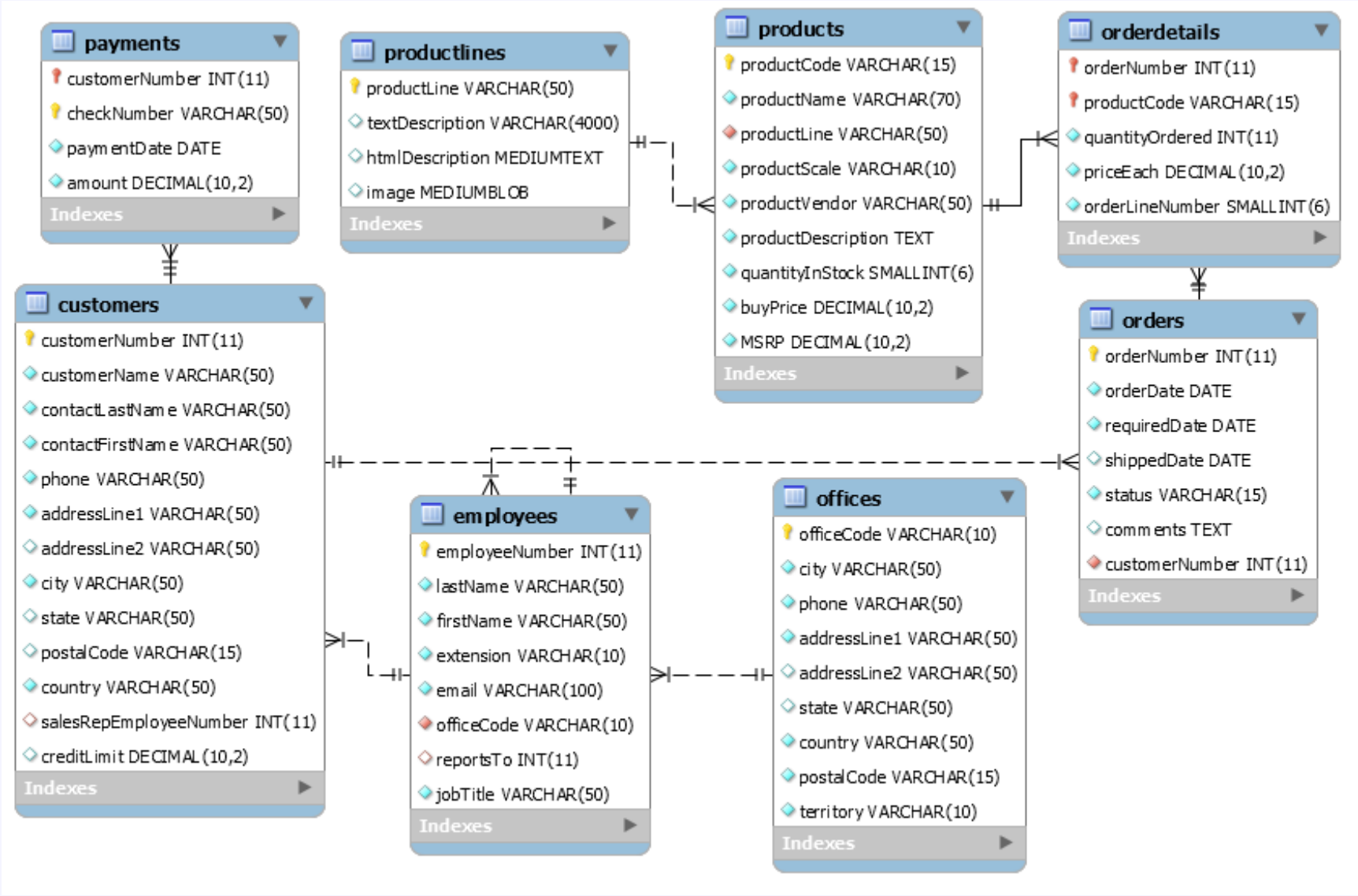
Πλεονεκτήματα

- Οι stored procedures είναι επαναχρησιμοποιήσιμες και διαφανείς στις εφαρμογές.
  - Οι stored procedures εξάγουν τη διασύνδεση της ΒΔ σε όλες τις εφαρμογές. Οι προγραμματιστές δεν χρειάζεται να αναπτύξουν συναρτήσεις που ήδη υποστηρίζονται στις stored procedures
- Οι stored procedures είναι ασφαλείς.
  - Ο διαχειριστής της ΒΔ μπορεί να χορηγήσει άδειες στις εφαρμογές που προσπελαύνουν τις stored procedures στη ΒΔ χωρίς να χορηγήσει άδειες στους πίνακες της ΒΔ.

Μειονεκτήματα

- Οι stored procedure δεν σχεδιάστηκαν για ανάπτυξη πολύπλοκης επιχειρησιακής λογικής.
- Είναι δύσκολος ο εντοπισμός σφαλμάτων (debug) στις stored procedures.
  - Μόνο λίγα ΣΔΒΔ επιτρέπουν τον εντοπισμό σφαλμάτων σε stored procedures. Η MySQL δεν υποστηρίζει εντοπισμό σφαλμάτων σε stored procedures.
- Εάν υπάρχουν πολλές stored procedures, η χρήση της μνήμης για κάθε σύνδεση στη ΒΔ που χρησιμοποιεί τις stored procedures θα αυξηθεί σημαντικά.

Σχήμα ΒΔ – Παράδειγμα



Δημιουργία Procedure

- Σύνταξη

```
delimiter $$
CREATE PROCEDURE name (param1, param2, …)
BEGIN
    …
END $$
delimiter ;
```

Δημιουργία Procedure – Παράδειγμα

```
CREATE PROCEDURE `GetAllProducts`()
BEGIN
   select *
   from products;
END
```

Κλήση Stored Procedure

- Σύνταξη

CALL STORED_PROCEDURE_NAME (param1, param2, …)

- Παράδειγμα

```
CALL GetAllProducts();
```

Δήλωση Μεταβλητών

- Σύνταξη

DECLARE variable_name **datatype(size) DEFAULT** default_value**;**

- Παράδειγμα

```
DECLARE total_sale INT DEFAULT 0;
DECLARE x, y INT DEFAULT 0;
```

Ανάθεση τιμών σε μεταβλητές

- Σύνταξη

1. **SET** variable_name = value**;**
2. **SELECT** … **INTO** variable_name

   FROM table_name

- Παράδειγμα 1

```
DECLARE total_count INT DEFAULT 0;
SET total_count = 10;
```

- Παράδειγμα 2

```
DECLARE total_products INT DEFAULT 0;
SELECT COUNT(*) INTO total_products
FROM products
```

Πεδίο εφαρμογής της μεταβλητής

- Μία μεταβλητή έχει το δικό τη πεδίο εφαρμογής που ορίζει τη διάρκεια ζωής της μεταβλητής.
  - Εάν δηλώσετε μία μεταβλητή μέσα σε μια stored procedure, θα έχει πεδίο εφαρμογής μόνο στη stored procedure (μέχρι το END).
- Μπορείτε να δηλώσετε δύο ή περισσότερες μεταβλητές με το ίδιο όνομα σε διαφορετικά πεδία εφαρμογής
- Μία μεταβλητή που αρχίζει με το σύμβολο@ είναι μία session μεταβλητή.
  - Είναι διαθέσιμη μέχρι να τελειώσει το session.

Παράμετροι των Stored Procedures

- **IN** (default). Το καλών πρόγραμμα πρέπει να περάσει ένα όρισμα στην stored procedure.
  - **Η τιμή μιας IN παραμέτρου είναι προστατευμένο**. Ακόμα και αν η τιμή της IN παραμέτρου μεταβληθεί μέσα σε μια stored procedure, η αρχική τιμή παραμένει αφού τερματιστεί η stored procedure.
- **OUT**. Η τιμή της παραμέτρου OUT μπορεί να αλλαχθεί μέσα σε μια stored procedure. Η νέα τιμή επιστρέφεται στο καλών πρόγραμμα.
  - Η stored procedure δεν μπορεί να προσπελάσει την αρχική τιμή μιας OUT παραμέτρου όταν εκκινεί.

Παράμετροι των Stored Procedures

- **INOUT**. Μία INOUT παράμετρος είναι ο συνδυασμός IN και OUT παραμέτρων.
  - Το καλών πρόγραμμα μπορεί να περάσει ένα όρισμα και η stored procedure μπορεί να μεταβάλλει την τιμή της INOUT παραμέτρου και να επιστρέψει τη νέα τιμή στο καλών πρόγραμμα.

Παράμετροι των Stored Procedures

- Σύνταξη

  MODE param_name param_type(param_size)

Παράμετροι των Stored Procedures

- **IN** (Παράδειγμα: επιλογή γραφείων με βάση τη χώρα)

```
DELIMITER //
CREATE PROCEDURE GetOfficeByCountry(IN countryName
    VARCHAR(255))
BEGIN
    SELECT *
    FROM offices
    WHERE country = countryName;
END //
DELIMITER ;

CALL GetOfficeByCountry('USA');
```

Παράμετροι των Stored Procedures

- **OUT** (Παράδειγμα: επιλογή του αριθμού των παραγγελιών με βάση τη κατάσταση της παραγγελίας)

```
DELIMITER //
CREATE PROCEDURE CountOrderByStatus( IN orderStatus
    VARCHAR(25), OUT total INT)
BEGIN
    SELECT count(orderNumber) INTO total
    FROM orders
    WHERE status = orderStatus;
END //
DELIMITER ;

CALL customerproduct.CountOrderByStatus('Shipped',@total);
SELECT @total;
```

Παράμετροι των Stored Procedures

- **INOUT** (Παράδειγμα)

```
DELIMITER //
CREATE PROCEDURE SetCounter(INOUT count INT(4), IN inc
    INT(4))
BEGIN
    SET count = count + inc;
END//
DELIMITER ;

SET @counter = 1;
CALL customerproduct.SetCounter(@counter,1); -- 2
CALL customerproduct.SetCounter(@counter,5); -- 7
SELECT @counter;
```

Επιστροφή Πολλαπλών Τιμών

- **Επιστροφή Πολλαπλών Τιμών** (Παράδειγμα)

```
CREATE PROCEDURE GetOrderByCust( IN cust_no INT, OUT
    shipped INT, OUT canceled INT, OUT resolved INT)
BEGIN
    -- shipped
    SELECT count(*) INTO shipped
    FROM orders
    WHERE customerNumber = cust_no AND status = 'Shipped';
    -- canceled
    SELECT count(*) INTO canceled
    FROM orders
     WHERE customerNumber = cust_no AND status = 'Canceled';
```

Επιστροφή Πολλαπλών Τιμών

```
    -- resolved
    SELECT count(*) INTO resolved
    FROM orders
    WHERE customerNumber = cust_no AND status = 'Resolved';
    -- disputed
    SELECT count(*) INTO disputed
    FROM orders
    WHERE customerNumber = cust_no AND status = 'Disputed';
END

CALL
    customerproduct.GetOrderByCust(141,@shipped,@canceled,@resolve
    d,@disputed);
SELECT @shipped,@canceled,@resolved,@disputed;
```

Δήλωση IF

- Σύνταξη

IF condition THEN statement(s)

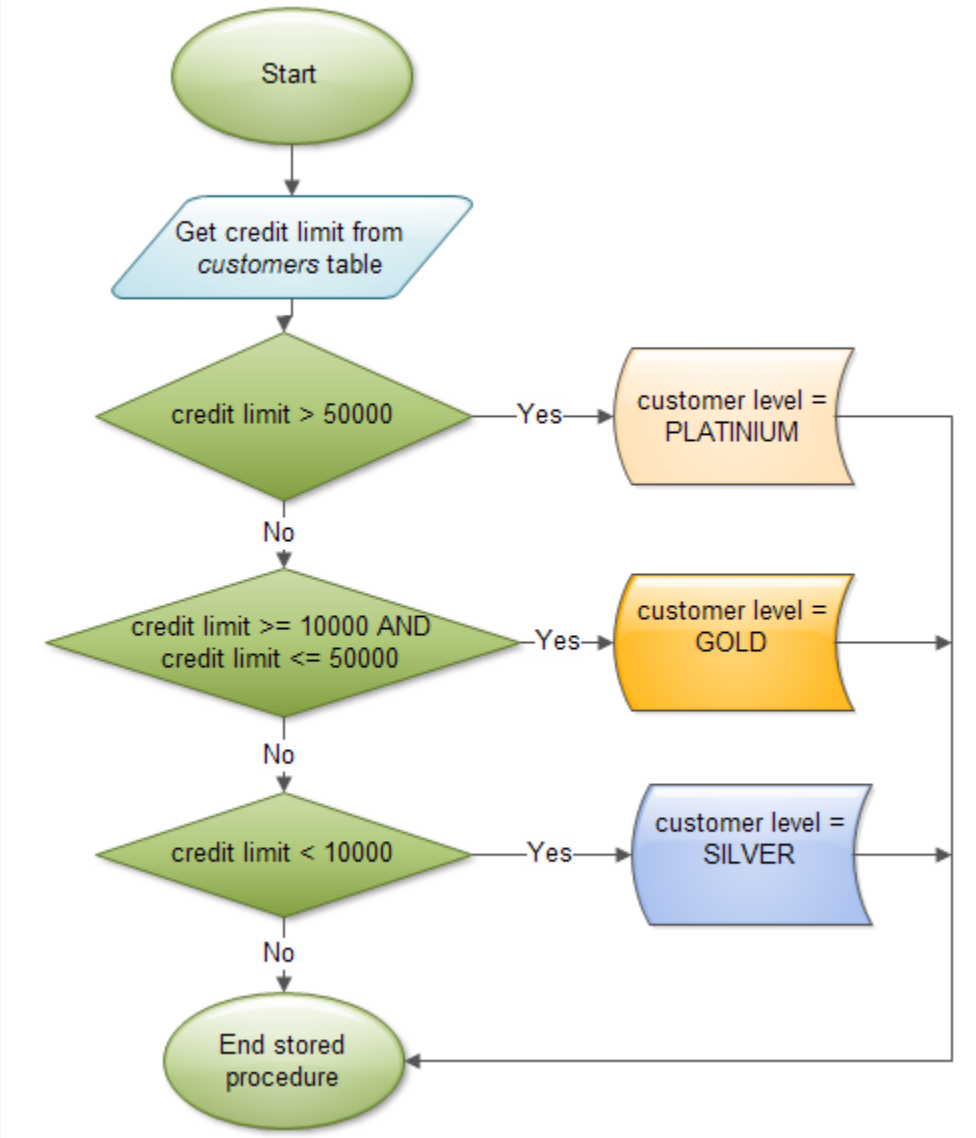
[ELSEIF condition THEN statement(s)] ...

[ELSE statement(s)]

END IF

- Η συνάρτηση IF είναι διαφορετική από τη δήλωση IF

Δήλωση IF – Παράδειγμα



Δήλωση IF – Παράδειγμα

```
CREATE PROCEDURE `GetCustomerLevel`(in  p_customerNumber
    int(11), out p_customerLevel  varchar(10))
BEGIN
    DECLARE creditlim double;
    SELECT creditlimit INTO creditlim FROM customers
    WHERE customerNumber = p_customerNumber;
    IF creditlim > 50000 THEN
          SET p_customerLevel = 'PLATINUM';
    ELSEIF (creditlim <= 50000 AND creditlim >= 10000) THEN
       SET p_customerLevel = 'GOLD';
    ELSEIF creditlim < 10000 THEN
       SET p_customerLevel = 'SILVER';
    END IF;
END
```

Δήλωση CASE

- Σύνταξη

```
CASE  case_expression
  WHEN when_expression_1 THEN commands
  WHEN when_expression_2 THEN commands
  …
  ELSE commands
END CASE;
```

Δήλωση CASE – Παράδειγμα

```
CREATE PROCEDURE `GetCustomerShipping`(in  p_customerNumber
    int(11), out p_shiping varchar(50))
BEGIN
    DECLARE customerCountry varchar(50);
    SELECT country INTO customerCountry
    FROM customers
    WHERE customerNumber = p_customerNumber;

    CASE customerCountry
          WHEN  'USA' THEN SET p_shiping = '2-day Shipping';
          WHEN 'Canada' THEN SET p_shiping = '3-day Shipping';
          ELSE SET p_shiping = '5-day Shipping';
    END CASE;
END
```

Δήλωση Searched CASE

- Η απλή δήλωση CASE επιτρέπει μόνο τον έλεγχο μιας τιμής μιας έκφρασης έναντι ενός συνόλου διακριτών τιμών.
- Η δήλωση ενός searched CASE είναι ισοδύναμη με μια δήλωση IF
- Σύνταξη

```
CASE
  WHEN condition_1 THEN commands
  WHEN condition_2 THEN commands
  ...
  ELSE commands
END CASE;
```

Δήλωση Searched CASE

```
BEGIN
    DECLARE creditlim double;
    SELECT creditlimit INTO creditlim
    FROM customers
    WHERE customerNumber = p_customerNumber;
    CASE
        WHEN creditlim > 50000 THEN
                SET p_customerLevel = 'PLATINUM';
        WHEN (creditlim <= 50000 AND creditlim >= 10000) THEN
                SET p_customerLevel = 'GOLD';
        WHEN creditlim < 10000 THEN
                SET p_customerLevel = 'SILVER';
    END CASE;
END
```

Δήλωση WHILE

- Σύνταξη

WHILE expression DO

statements

END WHILE

Δήλωση WHILE – Παράδειγμα

```
CREATE PROCEDURE TestMysqlWhileLoop ()
BEGIN
    DECLARE x  INT;
    DECLARE str  VARCHAR(255);
    SET x = 1;
    SET str =  '';

    WHILE x  <= 5 DO
          SET  str = CONCAT(str,x,',');
          SET  x = x + 1;
    END WHILE;

    SELECT str;
END
```

Δήλωση REPEAT

- Σύνταξη

```
REPEAT
    statements;
    UNTIL expression
END REPEAT
```

Δηλώσεις LEAVE, ITERATE

- Η δήλωση **LEAVE** επιτρέπει την έξοδο από ένα loop χωρίς την ανάμονη για τον έλεγχο μιας συνθήκης.
  - *Η δήλωση LEAVE λειτουργεί αρκιβώς όπως η δήλωση **break***
- Η δήλωση **ITERATE** επιτρέπει την παράλειψη ενός αριθμού εντολών κώδικα και την εκκίνηση νέας επανάληψης.
  - *Η δήλωση ITERATE είναι παρόμοια με τη δήλωση **continue***

Δήλωση LOOP

- Η MySQL παρέχει επίσης τη δήλωση **LOOP** η οποία εκτελεί ένα μπλοκ κώδικα επαναληπτικά με τη χρήση μιας ετικέτας loop **(loop label)**.
- Σύνταξη

```
loop_label: LOOP
    commands
END LOOP;
```

Δήλωση LOOP – Παράδειγμα

```
CREATE PROCEDURE TestMysqlLoop()
BEGIN
    DECLARE x  INT;
    DECLARE str  VARCHAR(255);
    SET x = 1;
    SET str =  '';

    loop_label:  LOOP
          IF  x > 10 THEN
                LEAVE  loop_label;
          END  IF;
```

Δήλωση LOOP – Παράδειγμα

```
        SET  x = x + 1;
        IF  (x mod 2) THEN
                ITERATE  loop_label;
        ELSE
                SET  str = CONCAT(str,x,',');
        END  IF;
    END LOOP;

    SELECT str;
END;
```

CURSOR

- Ο cursor χρησιμοποιείται για τη διαχείριση ενός συνόλου αποτελεσμάτων μέσα σε μια stored procedure. Ένας cursor επιτρέπει τον έλεγχο ενός συνόλου από εγγραφές που επιστρέφονται από ένα query.
- Οι cursors μπορούν να χρησιμοποιηθούν σε stored procedures, stored functions, και triggers.
- **Read only**: Δεν μπορείτε να ενημερώσετε δεδομένα των πινάκων μέσω του cursor.
- **Non-scrollable**: Μπορείτε μόνο να φέρετε εγγραφές με τη σειρά που καθορίζεται στη δήλωση SELECT.
  - Δεν μπορείτε να παραλείψετε εγγραφές ή να κατευθυνθείτε σε συγκεκριμένη εγγραφή.

CURSOR

- **Asensitive**: υπάρχουν δύο ειδών από cursors, asensitive cursor και insensitive cursor.
- Ένας asensitive cursor δείχνει στα πραγματικά δεδομένα, ενώ ένας insensitive cursor χρησιμοποιεί ένα προσωρινό αντίγραφο των δεδομένων.
- Ένας asensitive cursor έχει καλύτερη απόδοση από ένα insensitive cursor επειδή δεν χρειάζεται να δημιουργήσει ένα προσωρινό αντίγραφο.

CURSOR

- **Βήμα 1**: Δήλωση του cursor.

  DECLARE cursor_name CURSOR FOR SELECT_statement;

- Η δήλωση του cursor γίνεται μετά από κάθε δήλωση μεταβλητής
- Ένας cursor πρέπει πάντα να συσχετίζεται μέ μια δήλωση SELECT

CURSOR

- **Βήμα 2**: Άνοιγμα του cursor.

  OPEN cursor_name;

- Αρχικοποιεί το σύνολο των αποτελεσμάτων για τον cursor.
- Κλήση της δήλωσης OPEN πριν ανακτήσετε τις εγγραφές από το σύνολο αποτελεσμάτων (result set).

CURSOR

- **Βήμα 3**: Ανάκτηση των εγγραφών.

  FETCH cursor_name INTO variables list;

- Χρήση της δήλωσης FETCH για να ανακτήσετε την επόμενη εγγραφή που δείχνει ο cursor και μετακίνηση του cursor στην επόμενη εγγραφή του συνόλου αποτελεσμάτων.

Working with CURSOR

- **Βήμα 4**: Απενεργοποίηση του cursor και απελευθέρωση της μνήμης.

  CLOSE cursor_name;

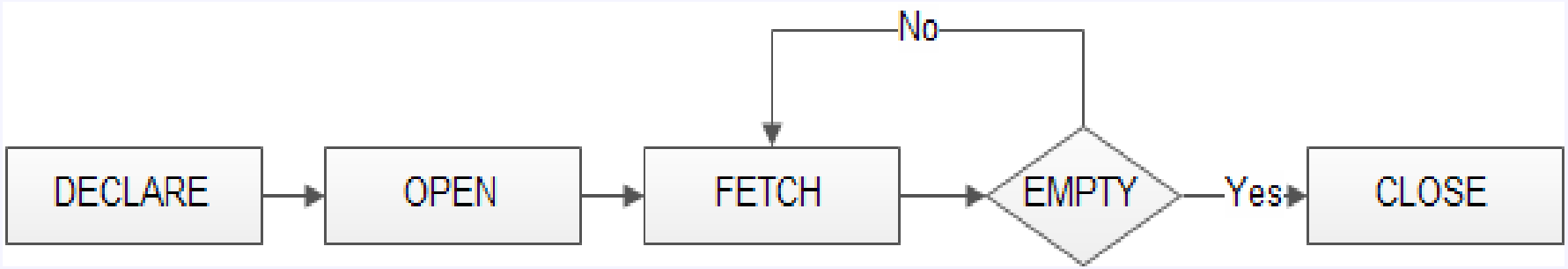
CURSOR

- **Σημαντικό**: Πρέπει να δηλώσετε το διαχειριστή **NOT FOUND** για να χειριστεί την περίπτωση που ο cursor δεν μπορεί να βρεί εγγραφή (είτε στην αρχή είτε στο τέλος)

```
DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;
```

- Όταν ο cursor φθάσει στο τέλος δεν μπορεί να φέρει δεδομένα. Ο διαχειριστής χρησιμοποιείται για να χειριστή την κατάσταση.

CURSOR



CURSOR – Παράδειγμα

```
CREATE PROCEDURE BuildEmailList (INOUT email_list
    varchar(4000))
BEGIN
    DECLARE v_finished INTEGER DEFAULT 0;
    DECLARE v_email varchar(100) DEFAULT "";
    -- declare cursor for employee email
    DECIARE email_cursor CURSOR FOR SELECT email FROM
    employees;
    -- declare NOT FOUND handler
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET
    v_finished = 1;
    -- open cursor
    OPEN email_cursor;
```

CURSOR – Παράδειγμα

```
    get_email: LOOP
        FETCH email_cursor INTO v_email;
        IF v_finished = 1 THEN
            LEAVE get_email;
        END IF;
        -- build email list
        SET email_list = CONCAT(v_email,";",email_list);
    END LOOP get_email;
    CLOSE email_cursor;
END

set @emailList = '';
call customerproduct.BuildEmailList(@emailList);
select @emailList;
```

Εμφάνιση Stored Procedures

- Σύνταξη

1. **SHOW PROCEDURE STATUS [LIKE 'pattern' | WHERE expr];**
2. **SHOW CREATE PROCEDURE stored_procedure_name;**

- Παράδειγμα

```
SHOW PROCEDURE STATUS WHERE name LIKE '%product%'
SHOW CREATE PROCEDURE GetAllProducts
```

Διαχείριση Λαθών

- Μπορείτε να δηλώσετε διαχειριστές που διαχειρίζονται από γενικές συνθήκες όπως προειδοποιήσεις (warnings) ή εξαιρέσει (exceptions) μέχρι συγκεκριμένους κωδικούς λάθους
- Σύνταξη

  DECLARE action HANDLER FOR condition_value statement;
- Το πεδίο action λαμβάνει μία από τις δύο τιμές:
  - **CONTINUE**: Η εκτέλεση το μπλοκ κώδικα (BEGIN … END) συνεχίζεται
  - **EXIT**: Η εκτέλεση του μπλοκ κώδικα τερματίζεται

Διαχείριση Λαθών

- Το πεδίο condition_value λαμβάνει μία από τις παρακάτω τιμές:
  - Ένα κωδικό λάθους της MySQL
  - Μία τιμή SQLSTATE, ή μπορεί να είναι ένα SQLWARNING , NOTFOUND ή SQLEXCEPTION συνθήκη
  - Μία συνθήκη που σχετίζεται με ένα κωδικό λάθους της MySQL ή μια τιμή SQLSTATE
- Η δήλωση μπορεί να είναι απλή ή σύνθετη δήλωση που περικλύεται μεταξύ των λέξεων κλειδιών BEGIN και END

Διαχείριση Λαθών – Παράδειγμα

```
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET
    has_error = 1;

DECLARE EXIT HANDLER FOR SQLEXCEPTION
BEGIN
ROLLBACK;
SELECT 'An error has occurred, operation rollbacked and the stored
    procedure was terminated';
END;

DECLARE CONTINUE HANDLER FOR 1062
SELECT 'Error, duplicate key occurred';
```

Διαχείριση Λαθών – Παράδειγμα

```
CREATE PROCEDURE `InsertOrderDetails` (IN orderNumber INT,
IN productCode VARCHAR(15), IN quantityOrdered INT,
IN priceEach DECIMAL(10,2), IN orderLineNumber SMALLINT(6))
BEGIN
    DECLARE CONTINUE HANDLER FOR 1062
    SELECT CONCAT('duplicate keys
    (',orderNumber,',',productCode,') found') AS msg;
    INSERT INTO orderdetails(orderNumber, productCode,
    quantityOrdered, priceEach, orderLineNumber)
    VALUES (orderNumber, productCode, quantityOrdered,
    priceEach, orderLineNumber);
    -- SELECT COUNT(*) FROM orderdetails;
END
```