|  |
| --- |
| Ράικος Ιάσων  ΑΜ: 1084552  Έτος: 3o  Μακρής Αντώνιος Ορέστης  ΑΜ: 1084516  Έτος: 3o  Δελημπαλταδάκης Γρηγόρης  ΑΜ: 1084647  Έτος: 3o |

|  |
| --- |
|  |
| Project Εργαστηρίου Βάσεων Δεδομένων 2022-2023 |
|  |

Περιεχόμενα

[Μέρος Α: Σχεδιασμός ΒΔ και SQL 2](#_Toc126237070)

[Κεφάλαιο 1 2](#_Toc126237071)

[Κεφάλαιο 2 3](#_Toc126237072)

[**3.1.3.1** 3](#_Toc126237073)

[**3.1.3.2** 5](#_Toc126237074)

[**3.1.3.3** 7](#_Toc126237075)

[**3.1.3.4** 12](#_Toc126237076)

[Κεφάλαιο 3 14](#_Toc126237077)

[**3.1.4.1** 14](#_Toc126237078)

[**3.1.4.2** 14](#_Toc126237079)

[**3.1.4.3** 16](#_Toc126237080)

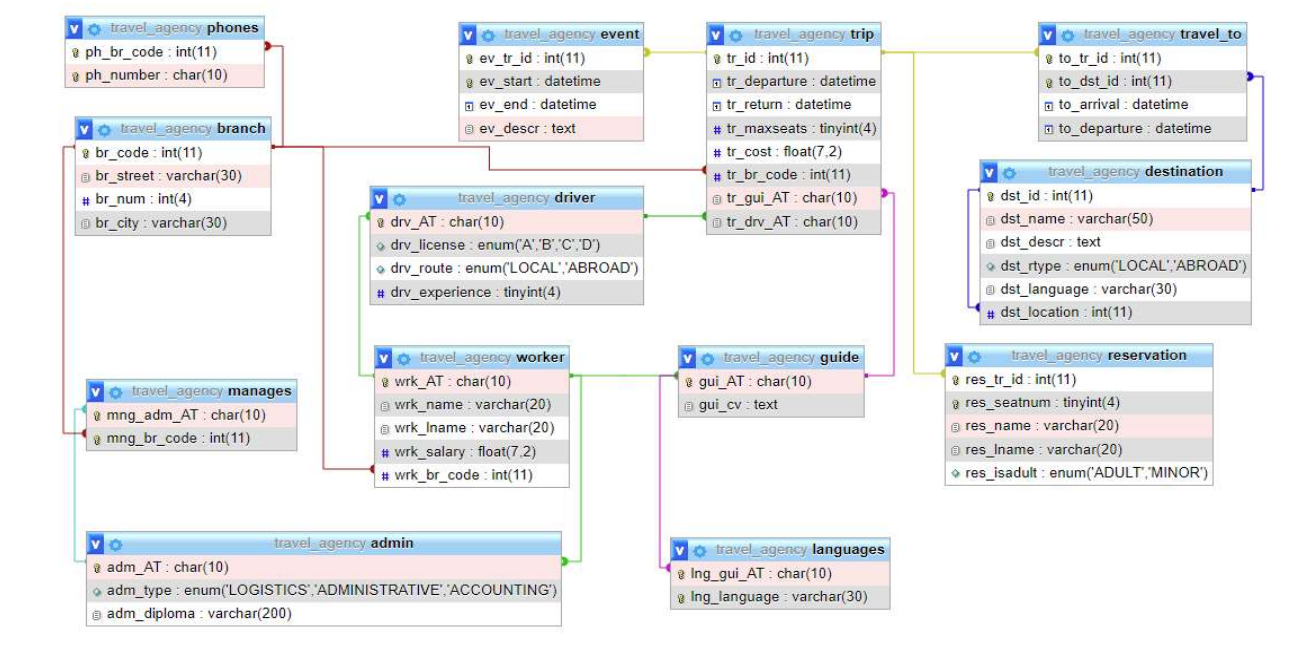
[Μέρος Β: GUIs 19](#_Toc126237081)

[Κεφάλαιο 4 19](#_Toc126237082)

[Κεφάλαιο 5 20](#_Toc126237083)

# Μέρος Α: Σχεδιασμός ΒΔ και SQL

# Κεφάλαιο 1



# Κεφάλαιο 2

## **3.1.3.1**

DROP PROCEDURE IF EXISTS insert\_driver;

DELIMITER $

CREATE PROCEDURE insert\_driver(IN AT char(10),IN name varchar(20),IN lname varchar(20),IN salary float(7,2),IN license ENUM('A','B','C','D'),IN route ENUM('LOCAL','ABROAD'),IN experience tinyint(4))

BEGIN

DECLARE code INT;

SELECT br\_code INTO code FROM (SELECT branch.br\_code,IFNULL(count(wrk\_AT),0) AS 'Number of Drivers' FROM driver INNER JOIN worker ON drv\_AT=wrk\_AT RIGHT JOIN branch ON br\_code=wrk\_br\_code GROUP BY br\_code ORDER BY IFNULL(count(wrk\_AT),0) ASC LIMIT 1) AS minimum\_drivers;

INSERT INTO worker VALUES (AT,name,lname,salary,code);

INSERT INTO driver VALUES (AT,license,route,experience);

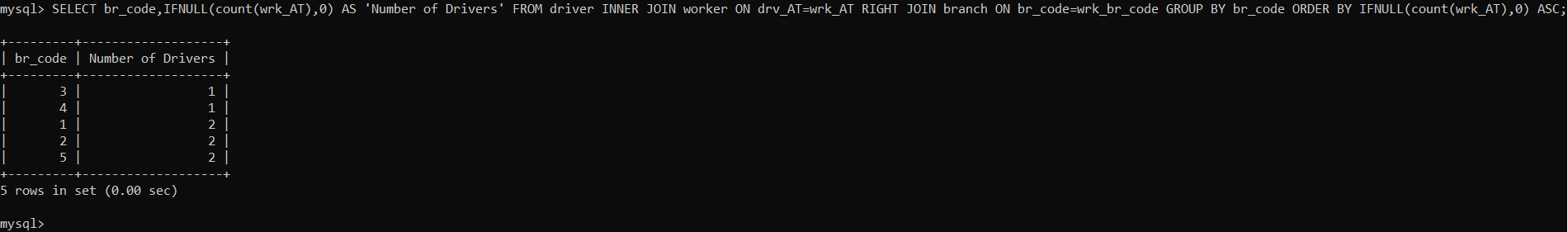
END$

DELIMITER ;

Στο **procedure** αυτό δίνονται ως είσοδοι τα στοιχεία ενός νέου οδηγού (αριθμός ταυτότητας, όνομα, επώνυμο, μισθός τύπος άδειας, τύπος δρομολογίου και μήνες εμπειρίας). Η **procedure** εντοπίζει το υποκατάστημα με τους λιγότερους οδηγούς και εισάγει τον οδηγό ως υπάλληλο του συγκεκριμένου υποκαταστήματος(τον εισάγει στους πίνακες worker και driver).Ο κώδικας στην ουσία αναθέτει την τιμή του br\_code του υποκαταστήματος με τους λιγότερους οδηγούς στην τοπική μεταβλητή code και μετά κάνει τα κατάλληλα **insert**.

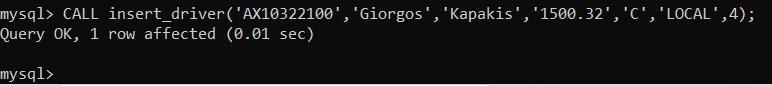
SELECT br\_code,IFNULL(count(wrk\_AT),0) AS 'Number of Drivers' FROM driver INNER JOIN worker ON drv\_AT=wrk\_AT RIGHT JOIN branch ON br\_code=wrk\_br\_code GROUP BY br\_code ORDER BY IFNULL(count(wrk\_AT),0) ASC;

Πρώτα εκτελούμε αυτό το **select** για να δούμε πόσους οδηγούς έχει κάθε branch.

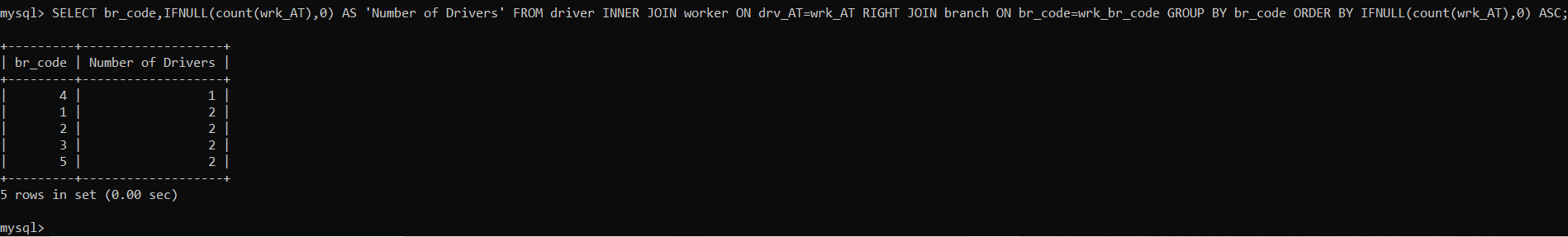


CALL insert\_driver('AX10322100','Giorgos','Kapakis','1500.32','C','LOCAL',4);

‘Επειτα καλούμε την **insert\_driver**.



Και ξανακαλούμε το **select** που είχαμε αναφέρει.



Βλέπουμε ότι ο οδηγός προστέθηκε στο branch με br\_code=3, γιατί ήταν ένα από τα branch με τους λιγότερους οδηγούς. Αν κάποια branch έχουν ίσο σε αριθμό οδηγούς και είναι αυτά με τους λιγότερους οδηγούς σε σχέση με τα άλλα τότε ο νέος οδηγός προστίθετε στο branch με το μικρότερο id.

## **3.1.3.2**

DROP PROCEDURE IF EXISTS search\_trip;

DELIMITER $

CREATE PROCEDURE search\_trip(IN code INT,IN start\_date datetime,IN end\_date datetime)

BEGIN

DECLARE trip\_id INT;

DECLARE dep DATETIME;

DECLARE ret DATETIME;

DECLARE maxseats TINYINT;

DECLARE cost FLOAT(7,2);

DECLARE br\_code INT;

DECLARE guide\_AT CHAR(10);

DECLARE driver\_AT CHAR(10);

DECLARE finishedFlag INT;

DECLARE no\_of\_reservations INT;

DECLARE driver\_name VARCHAR(20);

DECLARE driver\_lname VARCHAR(20);

DECLARE guide\_name VARCHAR(20);

DECLARE guide\_lname VARCHAR(20);

DECLARE empty\_seats INT;

DECLARE tripCurs CURSOR FOR SELECT \* FROM trip WHERE tr\_departure>=start\_date AND tr\_departure<=end\_date AND tr\_br\_code = code;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedFlag=1;

OPEN tripCurs;

SET finishedFlag=0;

FETCH tripCurs INTO trip\_id,dep,ret,maxseats,cost,br\_code,guide\_AT,driver\_AT;

WHILE(finishedFlag=0)

DO

SELECT COUNT(\*) INTO no\_of\_reservations FROM reservation INNER JOIN trip ON res\_tr\_id=trip\_id WHERE tr\_departure>=start\_date AND tr\_departure<=end\_date AND tr\_br\_code=code AND res\_tr\_id=tr\_id;

SELECT wrk\_name,wrk\_lname INTO driver\_name,driver\_lname FROM trip INNER JOIN worker ON tr\_drv\_AT=wrk\_AT WHERE tr\_id=trip\_id;

SELECT wrk\_name,wrk\_lname INTO guide\_name,guide\_lname FROM trip INNER JOIN worker ON tr\_gui\_AT=wrk\_AT WHERE tr\_id=trip\_id;

SET empty\_seats=maxseats-no\_of\_reservations;

SELECT trip\_id AS 'Trip Id',cost AS 'Cost',maxseats AS 'Maxseats',no\_of\_reservations AS 'Number of Reservations',empty\_seats AS 'Empty Seats',driver\_lname AS 'Driver Lastname',driver\_name AS 'Driver Name',guide\_lname AS 'Guide Lastname',guide\_name AS 'Guide Name',dep AS 'Departure',ret AS 'Return';

FETCH tripCurs INTO trip\_id,dep,ret,maxseats,cost,br\_code,guide\_AT,driver\_AT;

END WHILE;

CLOSE branchCurs;

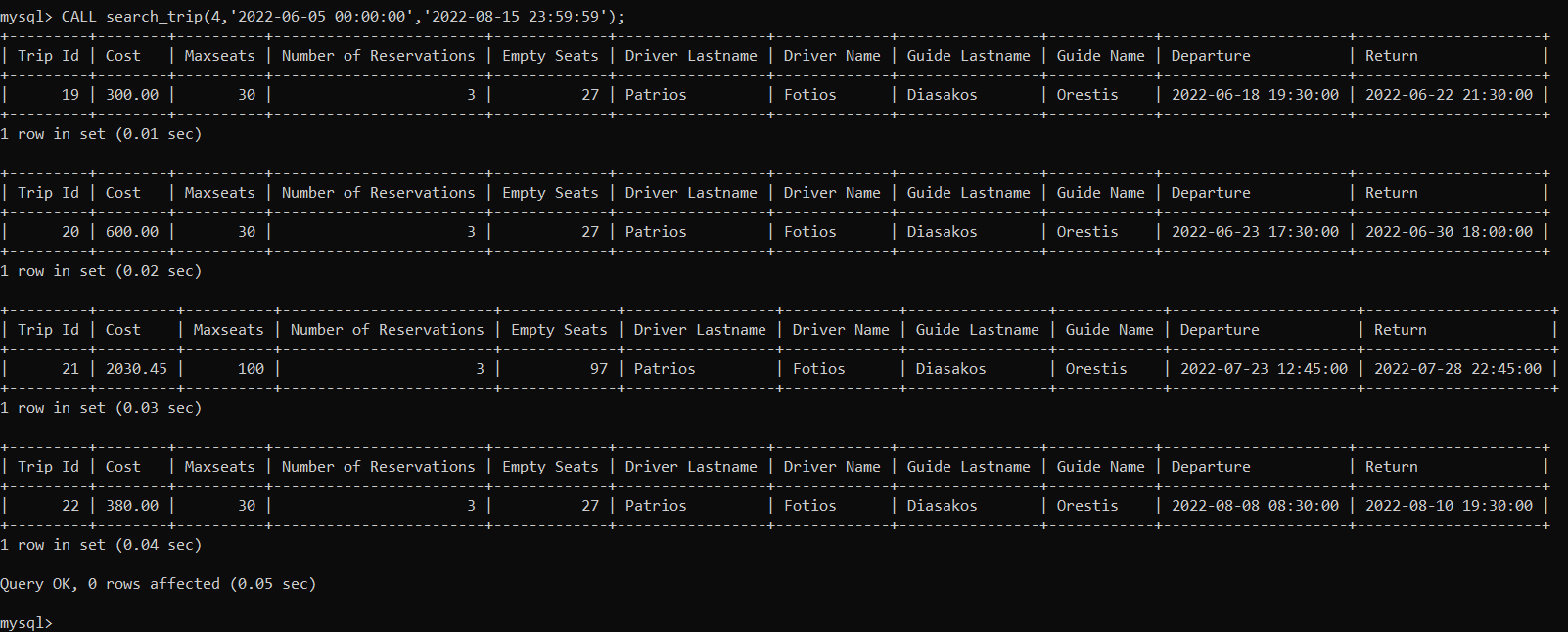
END$

DELIMITER ;

Στο **procedure** αυτό δίνονται ως ορίσματα ο κωδικός ενός υποκαταστήματος και δύο ημερομηνίες. Για τα ταξίδια (trip) που διοργανώνονται από το υποκατάστημα του οποίου δόθηκε ο κωδικός, και των οποίων η ημερομηνία αναχώρησης είναι μέσα στο διάστημα που δόθηκε, επιστρέφονται το κόστος ταξιδιού (tr\_cost ), οι μέγιστες θέσεις(maxseat), το σύνολο κρατήσεων (reservations), οι κενές θέσεις (maxseat – σύνολο κρατήσεων), το επώνυμο και το όνομα του οδηγού και του ξεναγού και τέλος η ημερομηνία αναχώρησης και επιστροφής.Δημιουργούμε έναν **cursor** τον οποίο τρέχουμε πάνω στον πίνακα **trip** και αναθέτουμε τις κατάλληλες πληροφορίες στις τοπικές μεταβλητές που δηλώσαμε και στο τέλος τις εκτυπόνουμε.

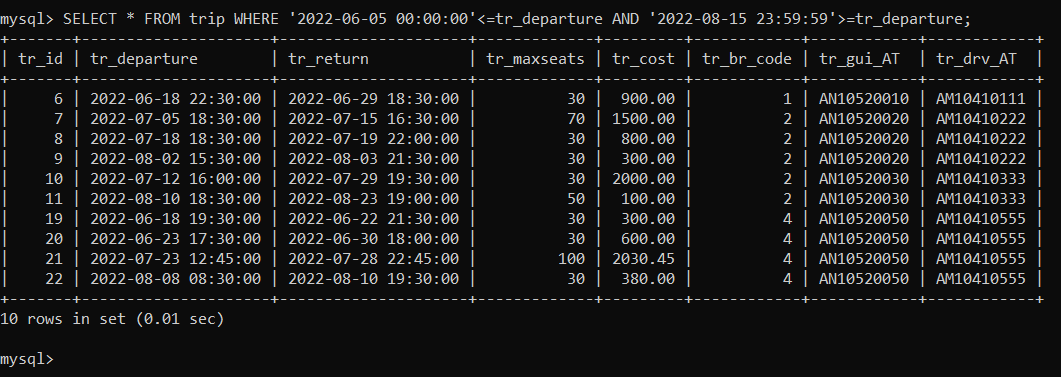
CALL search\_trip(4,'2022-06-05 00:00:00','2022-08-15 23:59:59');

Καλούμε την συνάρτηση **search\_trip**.



SELECT \* FROM trip WHERE '2022-06-05 00:00:00'<=tr\_departure AND '2022-08-15 23:59:59'>=tr\_departure;

Μετά εκτελούμε το παραπάνω **select** για να ελέξουμε αν λειτουργεί σωστά το **procedure**.



Στην περίπτωσή μας επιλέγει τα σωστά **trips**.

## **3.1.3.3**

DROP PROCEDURE IF EXISTS delete\_worker;

DELIMITER $

CREATE PROCEDURE delete\_worker(IN name varchar(20),IN lname varchar(20))

BEGIN

IF exists(SELECT \* FROM worker INNER JOIN admin ON wrk\_AT=adm\_AT WHERE wrk\_lname=lname AND wrk\_name=name AND adm\_type="ADMINISTRATIVE")

THEN

SELECT "This person is an admin in a branch therefore the deletion is restricted!" AS Error;

ELSEIF exists(SELECT \* FROM worker WHERE wrk\_lname=lname AND wrk\_name=name)

THEN

DELETE FROM worker WHERE wrk\_lname=lname AND wrk\_name=name;

ELSE

SELECT "This person isn't a worker in any branch!" AS Error;

END IF;

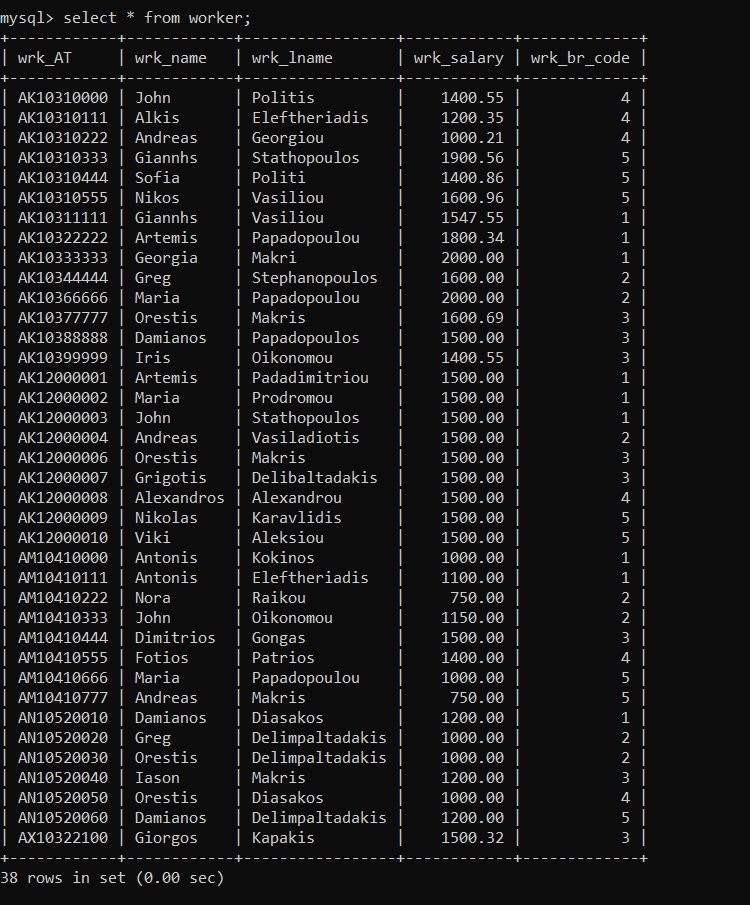
END$

DELIMITER ;

Το **procedure** αυτό παίρνει ως όρισμα το όνομα και το επώνυμο ενός υπαλλήλου. Αν είναι διοικητικός τον διαγράφει. Αν ο υπάλληλος είναι διευθυντής υποκαταστήματος τότε εμφανίζεται μήνυμα ότι είναι διευθυντής του υποκαταστήματος και δεν επιτρέπει τη διαγραφή.Με ένα if ελέγχουμε αν ένας υπάλληλος είναι διευθυντής και αν ισχύει τότε εκτυπώνουμε ένα error.Αν δεν είναι τότε μεταβαίνουμε στο επόμενο if και ελέγχουμε αν υπάρχει κάποιος υπάλληλος με το ονοματεπώνυμο που δώθηκε.Τέλος,αν δεν υπάρχει υπάλληλος τότε εκτυπώνουμε αντίστοιχο error.

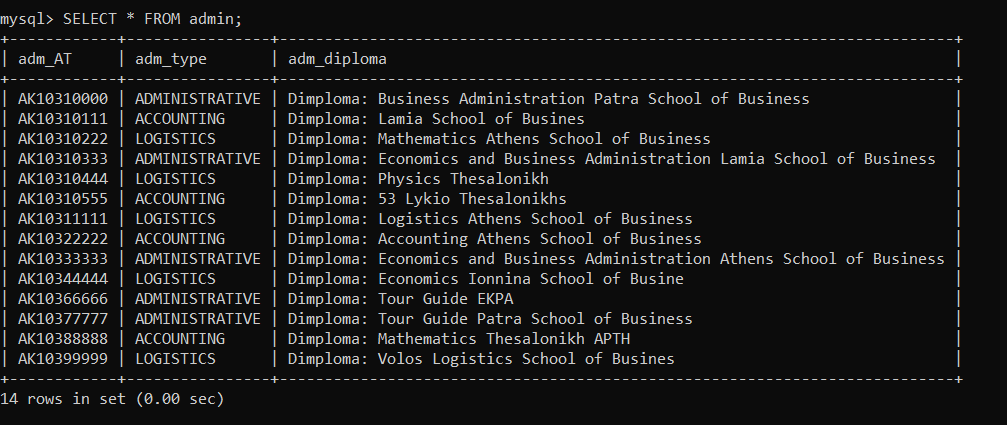
SELECT \* FROM worker;

Κάνουμε το παραπάνω **select**.



SELECT \* FROM admin;

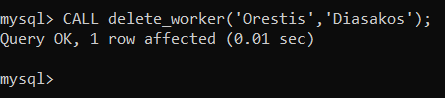
Ξανακάνουμε **select** αλλά το παραπάνω.



Με αυτά τα δύο select βλέπουμε τους workers από τους οποίους θα επιλέξουμε να κάνουμε **delete\_worker**.

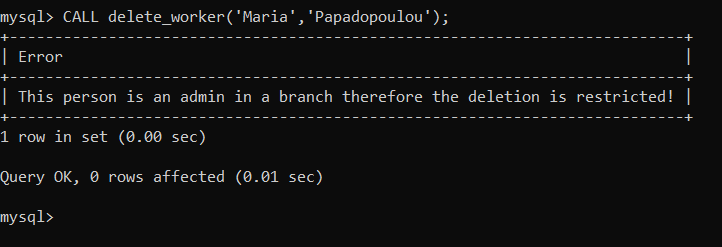
CALL delete\_worker('Orestis','Diasakos');

Κάνουμε το παραπάνω **call**.

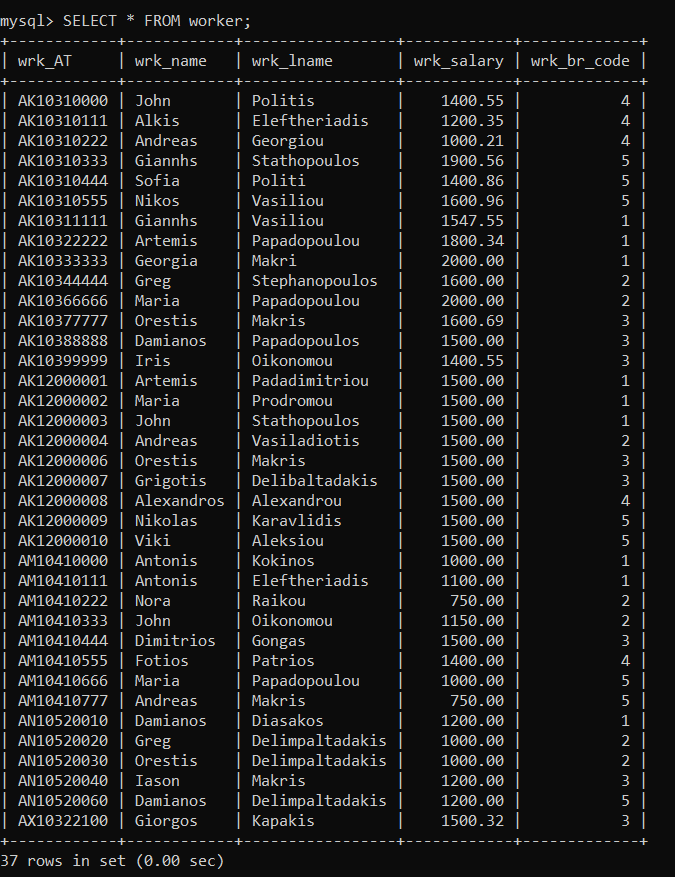


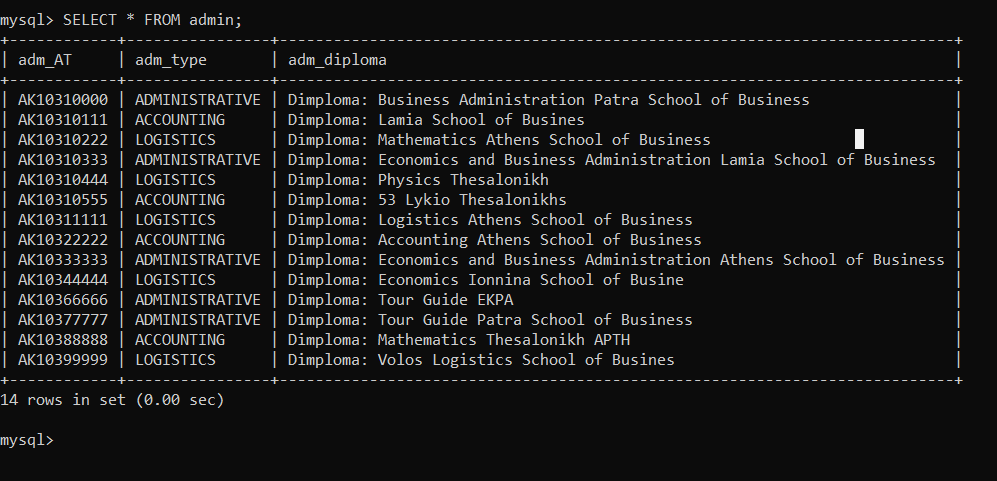
CALL delete\_worker('Maria','Papadopoulou');

Κάνουμε πάλη κλήση της **delete\_worker** αλλά με το παραπάνω **call**.



Κάνουμε ξανά τα δύο προηγούμενα **select** με την ίδια σειρά.





Με αυτά τα δύο **select** ελέγχουμε αν λειτουργεί το **procedure**.

## **3.1.3.4**

DROP INDEX res\_ind ON reservation\_offers;

CREATE INDEX res\_ind ON reservation\_offers(deposit\_amount, rsv\_lastname, rsv\_offer\_id);

Πρώτα από όλα λόγω του μεγάλου πλήθους εγγραφών, δημιουργούμε ένα ευρετήριο στον πίνακα reservation\_offers ώστε το πρόγραμμα να αναζητεί πιο αποδοτικά τις εγγραφές του πίνακα. Κάνουμε πρώτα **drop** σε περίπτωση που ήδη υπάρχει και το δημιουργούμε για τα πεδία **rsv\_lastname** ,**deposit\_amount** και **rsv\_offer\_id**. Δημιουργούμε το index για την στήλη **deposit\_amount,** επειδή χρησιμοποιούμε αυτό το πεδίο μέσα σε **WHERE** **condition** για το **procedure** του υποερωτήματος **α)** και για τις **rsv\_lastname, rsv\_offer\_id,** αντίστοιχα επειδή τα χρησιμοποιούμε σε πολλά **conditions** του **procedure** του υποερωτήματος **β)**.

**α)**

DROP PROCEDURE IF EXISTS res\_range;

DELIMITER $

CREATE PROCEDURE res\_range(IN r1 INT, IN r2 INT)

BEGIN

    SELECT rsv\_name AS 'Firstname', rsv\_lastname AS 'Lastname'

    FROM reservation\_offers

    WHERE deposit\_amount >= r1 AND deposit\_amount<=r2;

END$

DELIMITER ;

Το **procedure** αυτό παίρνει ως παραμέτρους εισόδου δύο τιμές (π.χ. 50, 200) και τυπώνει τους πελάτες (επώνυμο, όνομα) που έκαναν κρατήσεις σε προσφορές ταξιδιών και πλήρωσαν για προκαταβολή κάποιο ποσό ανάμεσα στις δύο τιμές.

*Έλεγχος*

Έχουμε τον πίνακα reservation\_offers, ο οποίος έχει 60000 εγγραφές.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Έστω ότι καλούμε την procedure res\_range για το εύρος προκαταβολών [80,95].

**Χωρίς το ευρετήριο**:



Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Βλέπουμε ότι το procedure εκτελέστηκε σε 0.02 s.

**Με το ευρετήριο:**

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Το procedure εκτελείται σε 0.01 s.

**β)**

DROP PROCEDURE IF EXISTS res\_names;

DELIMITER $

CREATE PROCEDURE res\_names(IN lname VARCHAR(20))

BEGIN

    DECLARE flag INT;

    DECLARE offer\_id INT;

    DECLARE no\_of\_res INT;

    DECLARE finishedFlag INT;

    DECLARE multResCurs CURSOR FOR

    SELECT rsv\_offer\_id, COUNT(rsv\_offer\_id)

    FROM reservation\_offers

    WHERE rsv\_lastname=lname

    GROUP BY rsv\_offer\_id

    HAVING COUNT(rsv\_offer\_id)>1

    ORDER BY rsv\_offer\_id;

    DECLARE singleResCurs CURSOR FOR

    SELECT rsv\_offer\_id, COUNT(rsv\_offer\_id)

    FROM reservation\_offers

    WHERE rsv\_lastname=lname

    GROUP BY rsv\_offer\_id

    HAVING COUNT(rsv\_offer\_id)=1

    ORDER BY rsv\_offer\_id;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedFlag=1;

    OPEN multResCurs;

    SET finishedFlag=0;

    FETCH multResCurs INTO offer\_id, no\_of\_res;

    SET flag = 0;

    WHILE(finishedFlag=0) DO

        IF(flag = 0) THEN

            SELECT 'Reserved offers for last name' AS '', lname AS '';

        END IF;

        SELECT offer\_id AS 'Offer Id', no\_of\_res AS 'Number of reserved offers';

        FETCH multResCurs INTO offer\_id,no\_of\_res;

        SET flag =  1;

    END WHILE;

    CLOSE multResCurs;

    OPEN singleResCurs;

    SET finishedFlag=0;

    SET flag = 0;

    FETCH singleResCurs INTO offer\_id, no\_of\_res;

    WHILE(finishedFlag=0) DO

        IF(flag = 0) THEN

            SELECT 'Reserved offers with one reservation per offer' AS '';

        END IF;

        SELECT rsv\_name, rsv\_lastname, rsv\_offer\_id

        FROM reservation\_offers

        WHERE rsv\_offer\_id = offer\_id AND rsv\_lastname = lname;

        SET flag = 1;

        FETCH singleResCurs INTO offer\_id,no\_of\_res;

    END WHILE;

END$

DELIMITER ;

Στο **procedure** αυτό δίνεται ως παράμετρος εισόδου το επώνυμο ενός πελάτη και εκτυπώνει τα ονοματεπώνυμα των πελατών με το επώνυμο αυτό και την προσφορά ταξιδιού στην οποία έχουν κάνει κράτηση. Σε περίπτωση που υπάρχουν πάνω από ένας επιστρέφει το πλήθος των πελατών με το επώνυμο αυτό, ανά προσφορά ταξιδιού.

Η υλοποίηση έγινε με χρήση δύο cursors, **DECLARE multResCurs CURSOR** και **DECLARE singleResCurs CURSOR** τα οποία διατρέχουν το πίνακα reservation\_offers και εντοπίζουν και εκτυπώνουν πολλαπλές και μονές κρατήσεις αντίστοιχα.

*Έλεγχος*

**Χωρίς ευρετήριο:**

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

**Με ευρετήριο:**

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Με την χρήση ευρετηρίου η εκτέλεση είναι κατά 0.01 s πιο γρήγορη.

Ο πίνακας reservation\_offers δημιουργήθηκε με την χρήση του δοθέντος αρχείου names.txt που περιέχει 10000 ονόματα, με τα οποία εκτελέσαμε τα INSERT για την εισαγωγή 60000 εγγραφών. Έτσι κάθε επώνυμο έχει κάνει πολλαπλές κρατήσεις σε κάθε προσφορά και για αυτό παραπάνω εκτυπώνεται ένας πίνακας με πολλαπλές κρατήσεις ανά offer.

Για να γίνει σωστός έλεγχος του procedure λοιπόν, εισάγουμε μερικές επιπλέον κρατήσεις:

INSERT INTO reservation\_offers VALUES(null,'Greg','Deli',50,1);

INSERT INTO reservation\_offers VALUES(null,'Maria','Deli',50,1);

INSERT INTO reservation\_offers VALUES(null,'Greg','Deli',50,2);

INSERT INTO reservation\_offers VALUES(null,'Greg','Deli',50,3);

Δηλαδή εισάγουμε 3 κρατήσεις για τον πελάτη Greg Deli, μία σε κάθε offer και μία κράτηση για την Maria Deli στο offer me id=1.

Αποτελέσματα:

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Βλέπουμε λοιπόν ότι για το επώνυμο Deli έχουμε δύο κρατήσεις για την προσφορά 1 (μία από τον Greg και μία από την Maria) και από μία κράτηση στις άλλες δυο προσφορές από τον Greg.

# Κεφάλαιο 3

## **3.1.4.1**

## **3.1.4.2**

DROP TRIGGER IF EXISTS check\_reservations;

DELIMITER $

CREATE TRIGGER check\_reservations BEFORE UPDATE ON trip

FOR EACH ROW

BEGIN

IF exists(SELECT \* FROM reservation WHERE res\_tr\_id=OLD.tr\_id)

THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT='This trip already has reservations therefore updates are restricted!';

END IF;

END$

DELIMITER ;

Το **trigger** αυτό αποτρέπει την αλλαγή της ημερομηνίας αναχώρησης,της ημερομηνίας επιστροφής και του κόστους του ταξιδιού, αν έχουν ήδη γίνει κρατήσεις γι’ αυτό.Ο έλεχγος γίνεται με ένα **if** που αν ισχύει τότε εκτυπώνεται κατάλληλο error.

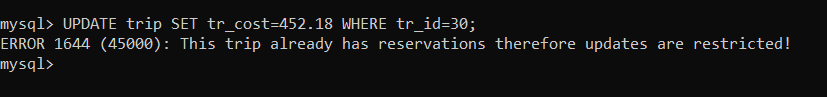
SELECT \* FROM reservation;

SELECT \* FROM trip;

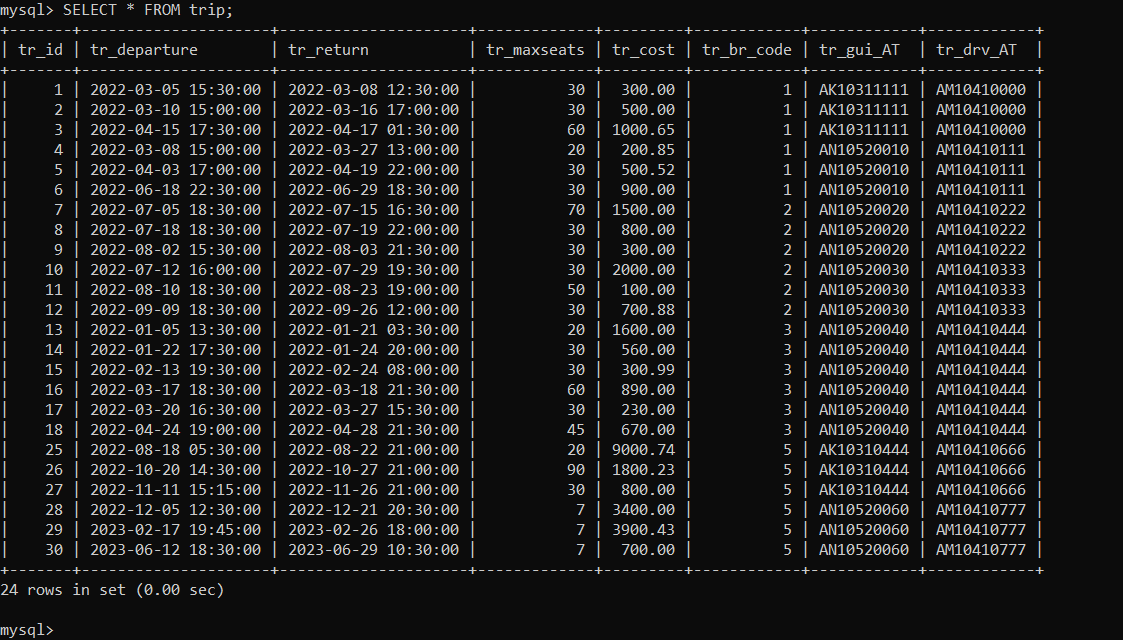
Κάνουμε τα δύο παραπάνω **select** για να δούμε ποια **trips** έχουν κρατήσεις και ποια όχι.

UPDATE trip SET tr\_cost=452.18 WHERE tr\_id=30;

Στη συνέχεια κάνουμε το παραπάνω **update**.



Κάνουμε το **select** για το **trip** και βλέπουμε αν έχει γίνει αλλαγή.



Όπως το περιμέναμε δεν έγινε κάποια αλλαγή.

## **3.1.4.3**

DROP TRIGGER IF EXISTS check\_salary;

DELIMITER $

CREATE TRIGGER check\_salary BEFORE UPDATE ON worker

FOR EACH ROW

BEGIN

IF OLD.wrk\_salary>=NEW.wrk\_salary

THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT='The new salary must be greater than the older!';

END IF;

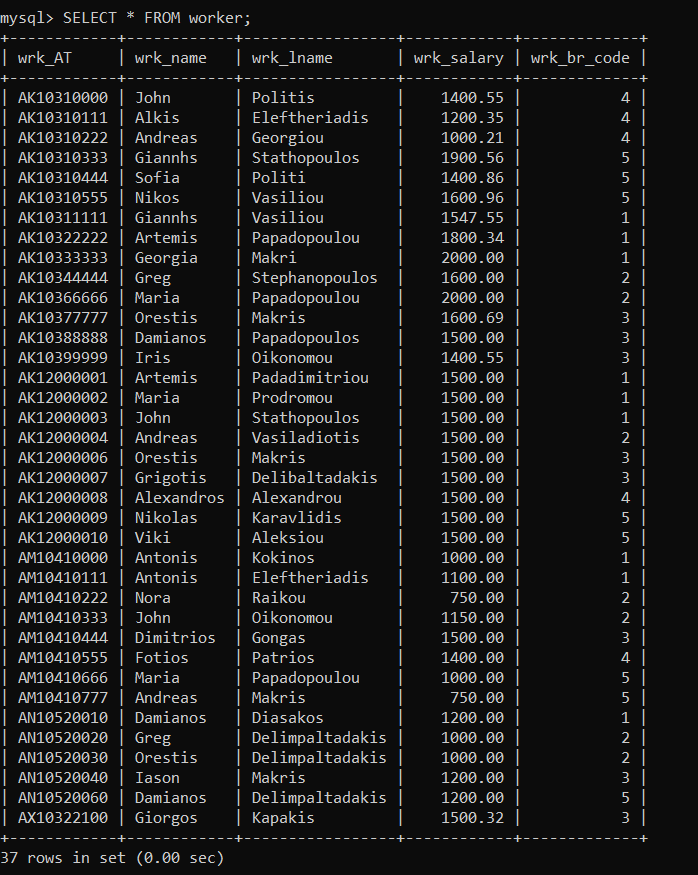
END$

DELIMITER ;

Το **trigger** αυτό δεν επιτρέπει τη μείωση του μισθού ενός υπαλλήλου.Με ένα if ελέχγουμε αν ο παλιός μισθός είναι μεγαλύτερος ή ίσος από τον καινούριο και είναι τότε εκτυπώνεται ένα error.

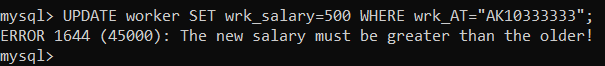
SELECT \* FROM worker;

Κάνουμε πρώτα αυτό το **select** για να διαλέξουμε τον worker στον οποίο θα προσπαθήσουμε να αλλάξουμε τον μισθό.

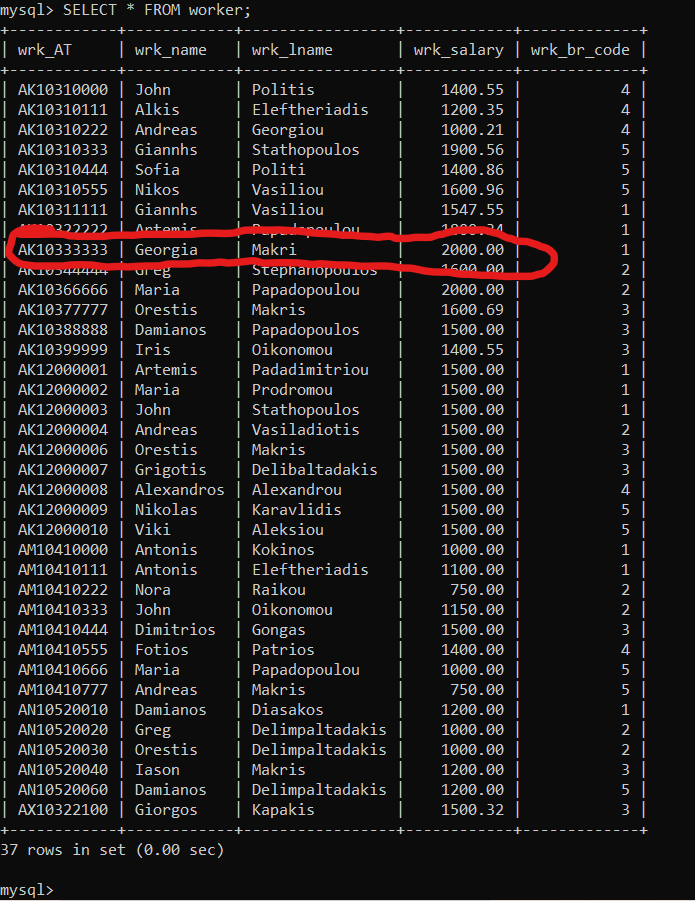


UPDATE worker SET wrk\_salary=500 WHERE wrk\_AT="AK10333333";

Εκτελούμε το παραπάνω **update**.



Ξανακάνουμε το **select** με το worker για να δούμε αν έχει γίνει αλλαγή.



Απ’ ότι φαίνεται δεν έχει γίνει αλλαγή.

# Μέρος Β: GUIs

# Κεφάλαιο 4

# Κεφάλαιο 5