**Βάσεις Δεδομένων Project 2022-2023**

Μέλη(με αλφαβητική σειρά επωνύμων):

1. Όνομα: Αγγουρά Ρουμπίνη – Μαρία

ΑΜ: 1084634

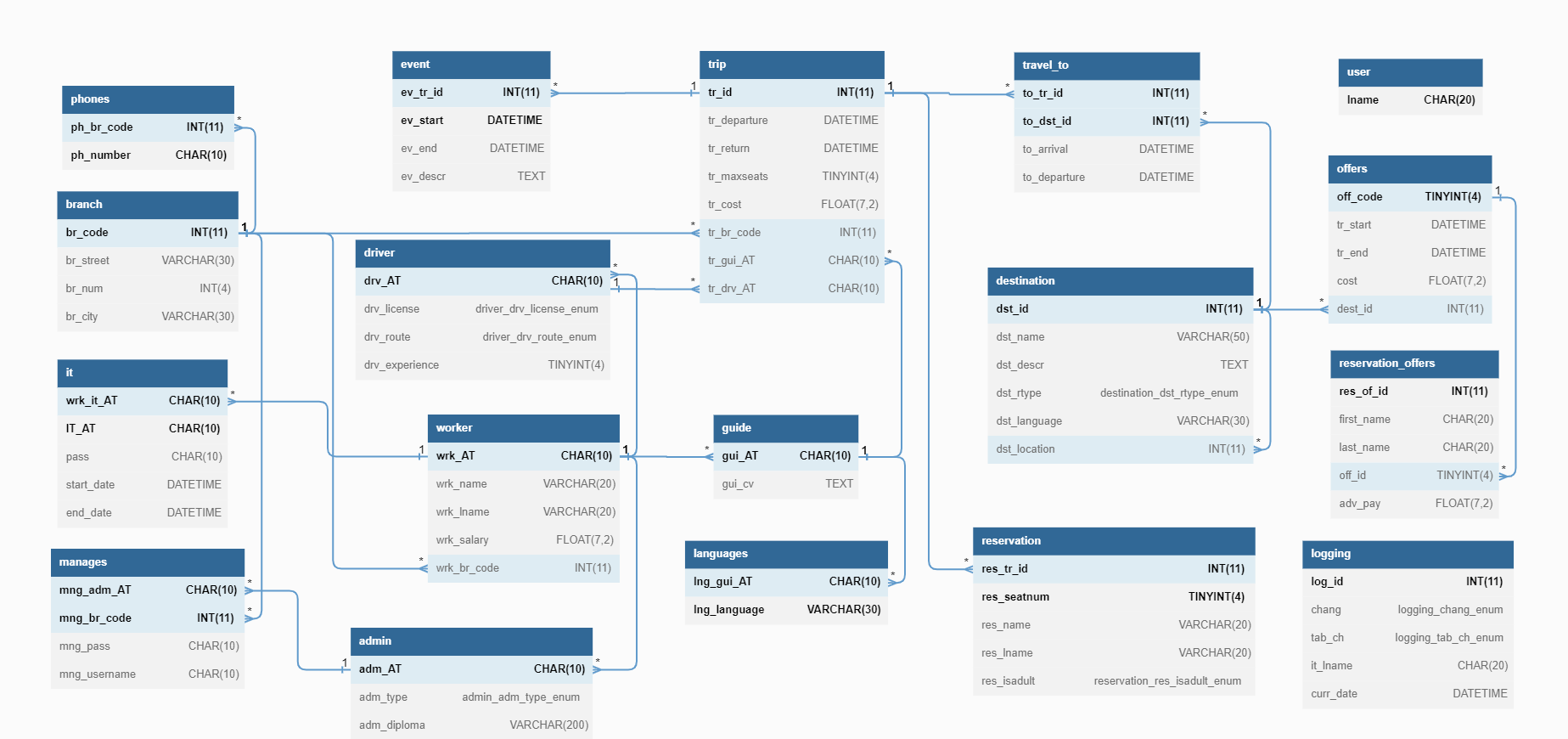
Έτος Φοίτησης: 3ο

1. Όνομα: Παυλόπουλος Ιάσονας

ΑΜ: 1084565

Έτος Φοίτησης: 3ο

**Κεφάλαιο 1**

Σχεσιακό Διάγραμμα

**Νέοι Πίνακες:**

* It: Αντιπροσωπεύει τους workers που ανήκουν στην κατηγορία It Managers. Όπως φαίνεται στο παραπάνω διάγραμμα, ο πίνακας It, αποτελείται από 5 πεδία, εκ των οποίων τα δύο αποτελούν και πρωτεύον κλειδιά του. Τα πεδία του είναι τα εξής:
* wrk\_it\_AT: Είναι πρωτεύον και ξένο κλειδί που δείχνει στον πίνακα worker, και αντιπροσωπεύει τον αριθμό ταυτότητας των εργαζομένων, στην συγκεκριμένη περίπτωση του It Manager.
* IT\_AT: Είναι πρωτεύον κλειδί του πίνακα μαζί με το wrk\_it\_AT. Αποτελεί το username του IT Manager με το οποίο μπορεί να συνδεθεί στην βάση δεδομένων.
* pass: Είναι ο κωδικός του It Manager με τον οποίο μπορεί να συνδεθεί στην βάση.
* start\_date: Αποτελεί την ημερομηνία που ο IT Manager ξεκίνησε να έχει αυτήν την θέση.
* end\_date: Αποτελεί την ημερομηνία που ο IT Manager σταμάτησε να έχει αυτήν την θέση. Αν ο IT Manager δεν έχει φύγει ακόμα από την εταιρεία, το πεδίο αυτό παίρνει την τιμή null.
* Offers: Έπειτα, δημιουργήσαμε τον πίνακα offers, στόχος του οποίου είναι να αποθηκεύει offers για καινούρια ταξίδια που δεν έχει δοθεί ακριβής ημερομηνία ακόμα. Ο πίνακας Offers αποτελείται από 5 πεδία:
* off\_code: Είναι το Πρωτεύον Κλειδί και αντιπροσωπεύει τον αριθμό του offer. Αυξάνεται με auto\_increment.
* tr\_start: Αποτελεί την ημερομηνία και ώρα έναρξης που μπορεί να πραγματοποιηθεί ένα ταξίδι
* tr\_end: Αποτελεί την ημερομηνία και ώρα λήξης που μπορεί να πραγματοποιηθεί ένα ταξίδι
* cost: Αποτελεί το κόστος ανά άτομο για αυτό το offer
* dest\_id: Αποτελεί τον κωδικό προορισμού. Είναι ξένο κλειδί το οποίο συνδέεται με τον πίνακα destination.
* Reservation\_offers: Αποτελεί έναν πίνακα στον οποίο ανήκουν όλες οι προσφορές για τα ταξίδια που ανήκουν στον πίνακα offers. Έχει 5 πεδία τα οποία είναι τα εξής:
* res\_of\_id: Είναι το πρωτεύον κλειδί του πίνακα. Αποτελεί τον μοναδικό κωδικό της προσφοράς, και αυξάνεται με auto\_increment.
* last\_name: Αποτελεί το επώνυμο του ατόμου που κάνει την προσφορά για κράτηση.
* first\_name: Αποτελεί το όνομα του ατόμου που κάνει την προσφορά για κράτηση.
* off\_id: Αποτελεί το id της προσφοράς ταξιδιού στην οποία απευθύνεται η κράτηση. Είναι ξένο κλειδί που συνδέεται με τον πίνακα offers.
* adv\_pay: Αποτελεί τον προκαταβολή που προσφέρει το άτομο που έκανε την κράτηση.
* Logging: Ο πίνακας logging, είναι απαραίτητος για την αναγνώριση των αλλαγών που γίνονται στην βάση, το είδος τους, και τον χρήστη που τις πραγματοποίησε. Αποτελείται από τα εξής 5 πεδία:
* log\_id: Είναι το Πρωτεύον Κλειδί και αντιπροσωπεύει τον αριθμό της αλλαγής ως αναγνωριστικό κωδικό. Αυξάνεται με auto\_increment.
* chang: Αποτελεί το είδος της αλλαγής η οποία εκτελείται.
* tab\_ch: Αποτελεί το όνομα του πίνακα στον οποίο γίνεται η αλλαγή.
* it\_lname: Αποτελεί το επώνυμο του χρήστη ο οποίος κάνει την αλλαγή.
* curr\_date: Αποτελεί την ημερομηνία και ώρα κατά την οποία γίνεται η αλλαγή.
* User: Τέλος δημιουργήσαμε τον πίνακα user, ο οποίος χρειάζεται ωστέ να αποθηκεύουμε σε κάθε σύνδεση χρήστη, το επώνυμό του. Αποτελείται από το εξής πεδίο:
* lname: Είναι το Πρωτεύον Κλειδί και αντιπροσωπεύει το επώνυμο του χρήστη που κάνει login στην βάση.

**Τροποποιήσεις σε ήδη υπάρχοντες πίνακες:**

* Manages: Στον πίνακα manages προσθέσαμε δύο ακόμα πεδία. Αυτό έγινε για να μπορέσουμε να έχουμε σύνδεση στην βάση και των διευθυντών του κάθε branch.

Τα καινούρια κλειδιά είναι τα εξής:

* mng\_pass: Αποτελεί τον κωδικό του admin για την σύνδεση του στην βάση.
* mng\_username: Αποτελεί το username του admin για την σύνδεσή του στην βάση.

**Κώδικας για την δημιουργία της βάσης:**

Προσοχή!! Στον παρακάτω κομμάτι κώδικα δεν συμπεριλαμβάνουμε τα inserts για τον πίνακα reservation\_offers καθώς είναι 60000 γραμμές κώδικα και δεν γίνεται να είναι σε αυτό το αρχείο. Αυτές οι εγγραφές θα βρίσκονται σε έναν έξτρα αρχείο txt, με το όνομα inserts.txt .

DROP database travel\_agency;

CREATE DATABASE travel\_agency;

USE travel\_agency;

CREATE TABLE branch(

br\_code INT(11) NOT NULL AUTO\_INCREMENT,

br\_street VARCHAR(30) DEFAULT 'UNKNOWN' NOT NULL,

br\_num INT(4) NOT NULL,

br\_city VARCHAR(30) DEFAULT 'UNKNOWN' NOT NULL,

PRIMARY KEY(br\_code)

);

INSERT INTO branch VALUES

(null, 'Karaiskaki', 13, 'Patra'),

(null, 'Kanakari', 10, 'Patra'),

(null, 'Korinthou', 27, 'Patra'),

(null, 'Zaimi', 30, 'Patra'),

(null, 'Maizonos', 115, 'Patra'),

(null, 'Gounari', 86, 'Patra'),

(null, 'Panourgia', 54, 'Patra'),

(null, 'Aretha', 57, 'Patra'),

(null, 'Ag. Nikolaou', 73, 'Patra'),

(null, 'Kolokotroni', 163, 'Patra');

CREATE TABLE phones(

ph\_br\_code INT(11) NOT NULL,

ph\_number CHAR(10) NOT NULL,

PRIMARY KEY(ph\_br\_code, ph\_number),

CONSTRAINT phones1 FOREIGN KEY (ph\_br\_code) REFERENCES branch(br\_code)

ON DELETE CASCADE ON UPDATE CASCADE

);

INSERT INTO phones VALUES

(1, '2104374832'),

(1, '6972345211'),

(2, '2114659823'),

(2, '6984635247'),

(3, '2432233445'),

(3, '6908989786'),

(4, '2103774332'),

(4, '6984756344'),

(5, '2124621442'),

(5, '6927651853'),

(6, '2152663927'),

(6, '6937462443'),

(7, '2108694523'),

(7, '6932618542'),

(8, '2721074556'),

(8, '6936172534'),

(9, '2710456217'),

(9, '6947527142'),

(10, '2335473923'),

(10, '6936251625');

CREATE TABLE worker(

wrk\_AT CHAR(10) NOT NULL,

wrk\_name VARCHAR(20) DEFAULT 'UNKNOWN' NOT NULL,

wrk\_lname VARCHAR(20) DEFAULT 'UNKNOWN' NOT NULL,

wrk\_salary FLOAT(7,2) NOT NULL,

wrk\_br\_code INT(11) NOT NULL,

PRIMARY KEY(wrk\_AT),

CONSTRAINT worker1 FOREIGN KEY(wrk\_br\_code) REFERENCES branch(br\_code)

ON DELETE CASCADE ON UPDATE CASCADE

);

INSERT INTO worker VALUES

('AM71514316', 'Petros', 'Giorgos', 568.1, 1),

('AW79051091', 'Mario', 'Luigi', 874.7, 2),

('CX68594948', 'Andreas', 'Oikonomou', 1233.2, 3),

('AT53041686', 'Ektoras', 'Samouil', 820.6, 4),

('YS17377807', 'Konstantinos', 'Kostantinou', 889.1, 5),

('CX21653217', 'Iasonas', 'Pavlopoulos', 2776.2, 6),

('XR13453928', 'Andreas', 'Andreadis', 1500.32, 7),

('SA33423559', 'Meletios', 'Polidouris', 1222.44, 8),

('TS73756168', 'Loukas', 'Ramakis', 1000.1, 9),

('YR41149523', 'Axilleas', 'Patroklos', 1022.3, 10),

('RW93324684', 'Roumpini', 'Aggoura', 2620.7, 1),

('WA59455740', 'Maria', 'Louloudi', 433.5, 2),

('SC32758869', 'Prigkipissa', 'Vasilissa', 655.2, 3),

('FG91992776', 'Katerina', 'Papaflessa', 836.44, 4),

('WE17292308', 'Eleni', 'Oikonomakou', 988.2, 5),

('AC18362095', 'Maritini', 'Petroula', 2500.21, 5),

('AY95427438', 'Sevi', 'Papadopoulou', 547.2, 6),

('AT13240957', 'Vasiliki', 'Malama', 766.5, 8),

('AT26598124', 'Danae', 'Kathariou', 1923.9, 7),

('WE30613700', 'Anastasia', 'Petropoulou', 957.2, 1),

('AQ88165869', 'Paris', 'Pettas', 1120.1, 9),

('AT23079419', 'Dimitris', 'Petreas', 1200.3, 10),

('TA54439709', 'Efi', 'Persikou', 942.3, 3),

('AY57791065', 'Davon', 'Sampson', 977.53, 4),

('AS33050797', 'Declan', 'Kelly', 1024.82, 5),

('YT13536905', 'Donavan', 'Osborn', 845.9, 6),

('XY88410423', 'Rashad', 'Noble', 1230.23, 7),

('SA88164961', 'Eleni', 'Papadaki', 1422.61, 1),

('AS83636600', 'Dionissis', 'Trivizas', 873.21, 2),

('XH62833740', 'Xristos', 'Petropoulos', 965.22, 3),

('FX23226072', 'Xaralampos', 'Xristidis', 743.1, 9),

('SY94496257', 'Andreas', 'Protopsaltis', 1247.91, 8),

('AW92790994', 'Aggeliki', 'Reka', 998.91, 10),

('SF16806081', 'Zaxarias', 'Michalainas', 1123.9, 7),

('SA47300877', 'Maria', 'Neokosmidi', 1273.85, 10),

('SR77541159', 'Tristin', 'Riley', 1273.85, 2),

('QW82328598', 'Manuel', 'Rowland', 957.21, 4),

('WQ64668093', 'Deanna', 'York', 853.23, 10),

('SS46333586', 'Tommy', 'Carney', 945.12, 6),

('AS85953443', 'Matt', 'Boss', 2931.40, 1),

('AX57737772', 'Gilberto', 'Archer', 2341.23, 1),

('SA98242556', 'Samantha', 'Wilkins', 2422.03, 1),

('SA9824TEST', 'Test', 'User', 2131.03, 1);

CREATE TABLE it(

wrk\_it\_AT CHAR(10) NOT NULL,

IT\_AT CHAR(10) NOT NULL,

pass CHAR(10) DEFAULT 'password' NOT NULL,

start\_date DATETIME NOT NULL,

end\_date DATETIME,

PRIMARY KEY(IT\_AT, wrk\_it\_AT),

CONSTRAINT it1 FOREIGN KEY(wrk\_it\_AT) REFERENCES worker(wrk\_AT)

ON DELETE CASCADE ON UPDATE CASCADE

);

INSERT INTO it VALUES

('AS85953443', 'IT01', 'pass0', '2022-01-01 09:00:00', '2022-12-30 09:00:00'),

('AX57737772', 'IT02', 'pass1', '2023-01-01 09:00:00', null),

('SA98242556', 'IT03', 'pass2', '2022-01-02 09:00:00', null);

CREATE TABLE driver(

drv\_AT CHAR(10) NOT NULL,

drv\_license ENUM('A', 'B', 'C', 'D') NOT NULL,

drv\_route ENUM('LOCAL', 'ABROAD') NOT NULL,

drv\_experience TINYINT(4) NOT NULL,

PRIMARY KEY(drv\_AT),

CONSTRAINT driver1 FOREIGN KEY(drv\_AT) REFERENCES worker(wrk\_AT)

ON DELETE CASCADE ON UPDATE CASCADE

);

INSERT INTO driver VALUES

('SA88164961', 'C', 'ABROAD', 12),

('WA59455740', 'B', 'LOCAL', 11),

('AS33050797', 'D', 'LOCAL', 28),

('XR13453928', 'D', 'LOCAL', 43),

('SC32758869', 'D', 'LOCAL', 1),

('YT13536905', 'D', 'LOCAL', 2),

('AT53041686', 'B', 'ABROAD', 7),

('SY94496257', 'C', 'LOCAL', 32),

('TS73756168', 'D', 'ABROAD', 15),

('YR41149523', 'C', 'ABROAD', 23),

('SR77541159', 'B', 'ABROAD', 23),

('QW82328598', 'C', 'ABROAD', 23),

('WQ64668093', 'B', 'ABROAD', 23),

('SS46333586', 'B', 'ABROAD', 23);

CREATE TABLE admin(

adm\_AT CHAR(10) NOT NULL,

adm\_type ENUM('LOGISTICS', 'ADMINISTRATIVE', 'ACCOUNTING') NOT NULL,

adm\_diploma VARCHAR(200) DEFAULT 'UNKNOWN' NOT NULL,

PRIMARY KEY(adm\_AT),

CONSTRAINT admin1 FOREIGN KEY(adm\_AT) REFERENCES worker(wrk\_AT)

ON DELETE CASCADE ON UPDATE CASCADE

);

INSERT INTO admin VALUES

('WE30613700', 'LOGISTICS', 'Logistics Diploma'),

('YS17377807', 'LOGISTICS', 'Economics Diploma'),

('TA54439709', 'ACCOUNTING', 'Mathematical Engineer Diploma Diploma'),

('SF16806081', 'LOGISTICS', 'Logistics Diploma'),

('SA47300877', 'ACCOUNTING', 'Economics Diploma'),

('RW93324684', 'ADMINISTRATIVE', 'Computer Engineer Diploma'),

('AW79051091', 'ADMINISTRATIVE', 'Mechanical Engineer Diploma'),

('CX68594948', 'ADMINISTRATIVE', 'Chemical Engineer Diploma'),

('AY57791065', 'ADMINISTRATIVE', 'Chemical Engineer Diploma'),

('AC18362095', 'ADMINISTRATIVE', 'Buisness Managment Diploma'),

('AT26598124', 'ADMINISTRATIVE', 'Mathematical Engineer Diploma'),

('SA33423559', 'ADMINISTRATIVE', 'Chemical Engineer Diploma'),

('CX21653217', 'ADMINISTRATIVE', 'Computer Engineer Diploma'),

('AQ88165869', 'ADMINISTRATIVE', 'Buisness Management Diploma'),

('AT23079419', 'ADMINISTRATIVE', 'Computer Engineer Diploma');

CREATE TABLE manages(

mng\_adm\_AT CHAR(10) NOT NULL,

mng\_br\_code INT(11) NOT NULL,

mng\_pass CHAR(10) NOT NULL,

mng\_username CHAR(10) NOT NULL,

PRIMARY KEY(mng\_adm\_AT,mng\_br\_code),

CONSTRAINT manages1 FOREIGN KEY (mng\_adm\_AT) REFERENCES admin(adm\_AT)

ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT manages2 FOREIGN KEY (mng\_br\_code) REFERENCES branch(br\_code)

ON DELETE CASCADE ON UPDATE CASCADE

);

INSERT INTO manages VALUES

('RW93324684', 1, "mng1", "MNG01"),

('AW79051091', 2, "mng2", "MNG02"),

('CX68594948', 3, "mng3", "MNG03"),

('AY57791065', 4, "mng4", "MNG04"),

('AC18362095', 5, "mng5", "MNG05"),

('AT26598124', 7, "mng6", "MNG06"),

('SA33423559', 8, "mng7", "MNG07"),

('CX21653217', 6, "mng8", "MNG08"),

('AQ88165869', 9, "mng9", "MNG09"),

('AT23079419', 10, "mng10", "MNG10");

CREATE TABLE guide(

gui\_AT CHAR(10) NOT NULL,

gui\_cv TEXT,

PRIMARY KEY(gui\_AT),

CONSTRAINT guide1 FOREIGN KEY (gui\_AT) REFERENCES worker(wrk\_AT)

ON DELETE CASCADE ON UPDATE CASCADE

);

INSERT INTO guide VALUES

('AM71514316', 'Worked as a guide at the Acropolis site for 10 years'),

('FG91992776', 'Worked as a guide at the Santorini island for 5 years'),

('WE17292308', 'Worked as a guide at the Mykonos island for 7 years'),

('AY95427438', 'Worked as a guide at the Acropolis site for 5 years'),

('AT13240957', 'Worked as a guide at the Monemvasia castle for 20 years'),

('XY88410423', 'Worked as a guide at the Acropolis site for 15 years'),

('AS83636600', 'Worked as a guide at the Acropolis site for 1 year'),

('XH62833740', 'Worked as a guide at the Acropolis site for 1 year'),

('FX23226072', 'Worked as a guide at the Acropolis site for 1 year'),

('AW92790994', 'Worked as a guide at the Acropolis site for 1 year');

CREATE TABLE languages(

lng\_gui\_AT CHAR(10) NOT NULL,

lng\_language VARCHAR(30) DEFAULT 'Uknown' NOT NULL,

PRIMARY KEY(lng\_language, lng\_gui\_AT),

CONSTRAINT languages1 FOREIGN KEY (lng\_gui\_AT) REFERENCES guide(gui\_AT)

ON DELETE CASCADE ON UPDATE CASCADE

);

INSERT INTO languages VALUES

('AM71514316', 'English'),

('AM71514316', 'Greek'),

('FG91992776', 'English'),

('FG91992776', 'Greek'),

('WE17292308', 'English'),

('WE17292308', 'Greek'),

('AY95427438', 'English'),

('AY95427438', 'Greek'),

('AT13240957', 'English'),

('AT13240957', 'Greek'),

('XY88410423', 'English'),

('XY88410423', 'Spanish'),

('XY88410423', 'French'),

('AS83636600', 'English'),

('AS83636600', 'Greek'),

('XH62833740', 'English'),

('XH62833740', 'Greek'),

('XH62833740', 'Italian'),

('XH62833740', 'German'),

('FX23226072', 'English'),

('FX23226072', 'Greek'),

('AW92790994', 'English'),

('AW92790994', 'Greek'),

('AW92790994', 'Spanish'),

('AW92790994', 'French');

CREATE TABLE trip(

tr\_id INT(11) NOT NULL AUTO\_INCREMENT,

tr\_departure DATETIME,

tr\_return DATETIME,

tr\_maxseats TINYINT(4),

tr\_cost FLOAT(7,2),

tr\_br\_code INT(11) NOT NULL,

tr\_gui\_AT CHAR (10),

tr\_drv\_AT CHAR (10),

PRIMARY KEY(tr\_id),

CONSTRAINT trip1 FOREIGN KEY (tr\_br\_code) REFERENCES branch(br\_code)

ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT trip2 FOREIGN KEY (tr\_drv\_AT) REFERENCES driver(drv\_AT)

ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT trip3 FOREIGN KEY (tr\_gui\_AT) REFERENCES guide(gui\_AT)

ON DELETE CASCADE ON UPDATE CASCADE

);

INSERT INTO trip VALUES

(null, '2023-01-29 11:00:00', '2023-02-08 18:00:00', 20, 1023.11, 1, 'AM71514316', 'SA88164961'),

(null, '2023-11-25 10:30:00', '2023-11-30 17:30:00', 5, 361.21, 2, 'AS83636600', 'WA59455740'),

(null, '2023-10-12 11:00:00', '2023-10-16 18:30:00', 15, 451.76, 7, 'XY88410423', 'XR13453928'),

(null, '2023-06-03 12:30:00', '2023-06-15 16:00:00', 6, 842.54, 4, 'FG91992776', 'AT53041686'),

(null, '2023-05-09 11:30:00', '2023-05-20 17:30:00', 9, 822.97, 10, 'AW92790994', 'YR41149523'),

(null, '2023-08-13 11:30:00', '2023-08-23 18:00:00', 50, 677.91, 6, 'AY95427438', 'YT13536905'),

(null, '2023-09-12 12:30:00', '2023-09-17 19:30:00', 30, 532.12, 3, 'XH62833740', 'SC32758869'),

(null, '2023-12-12 10:30:00', '2023-12-20 17:30:00', 15, 742.23, 8, 'AT13240957', 'SY94496257'),

(null, '2023-01-27 12:30:00', '2023-01-30 20:30:00', 17, 352.11, 5, 'WE17292308', 'AS33050797'),

(null, '2023-02-18 11:30:00', '2023-02-28 21:00:00', 20, 874.44, 9, 'FX23226072', 'TS73756168'),

(null, '2023-05-23 11:00:00', '2023-05-30 17:30:00', 10, 858.31, 1, 'AM71514316', 'SA88164961'),

(null, '2023-07-04 12:30:00', '2023-07-17 18:30:00', 4, 923.55, 2, 'AS83636600', 'SR77541159'),

(null, '2023-03-02 12:00:00', '2023-03-08 19:30:00', 22, 690.69, 7, 'XY88410423', 'XR13453928'),

(null, '2023-04-01 11:30:00', '2023-04-06 21:00:00', 14, 570.09, 4, 'FG91992776', 'QW82328598'),

(null, '2023-07-22 12:30:00', '2023-07-28 19:30:00', 5, 700.99, 10, 'AW92790994', 'WQ64668093'),

(null, '2023-09-09 11:30:00', '2023-09-15 18:30:00', 4, 790.89, 6, 'AY95427438', 'SS46333586'),

(null, '2023-09-15 10:30:00', '2023-09-19 19:30:00', 7, 350.99, 3, 'XH62833740', 'SC32758869'),

(null, '2023-12-12 11:30:00', '2023-12-17 17:00:00', 10, 500.34, 8, 'AT13240957', 'SY94496257'),

(null, '2023-10-07 12:30:00', '2023-10-10 17:00:00', 30, 357.10, 5, 'WE17292308', 'AS33050797'),

(null, '2023-12-04 10:30:00', '2023-12-15 18:00:00', 45, 1289.99, 9, 'FX23226072', 'TS73756168'),

(null, '2023-12-04 10:30:00', '2023-12-15 18:00:00', 32, 1289.99, 9, 'FX23226072', 'TS73756168');

CREATE TABLE event(

ev\_tr\_id INT(11) NOT NULL,

ev\_start DATETIME NOT NULL,

ev\_end DATETIME NOT NULL,

ev\_descr TEXT,

PRIMARY KEY(ev\_tr\_id, ev\_start),

CONSTRAINT event1 FOREIGN KEY(ev\_tr\_id) REFERENCES trip(tr\_id)

ON DELETE CASCADE ON UPDATE CASCADE

);

INSERT INTO event VALUES

(1, '2023-01-31 12:00:00', '2023-01-31 14:00:00', 'Visiting The Louvre Museum.'),

(2, '2023-11-27 11:30:00', '2023-11-27 14:30:00', 'Learning about the prosperous history of Kalambaka.'),

(2, '2023-11-28 15:00:00', '2023-11-28 17:30:00', 'Visiting the old part of Kalamata and watching a play at the Castle of Isabeau.'),

(3, '2023-10-13 13:00:00', '2023-10-13 15:30:00', 'Doing a tour of the villages of the island of Corfu!'),

(3, '2023-10-12 19:30:00', '2023-10-12 22:00:00', 'Eating at a local Restaurant.'),

(4, '2023-06-07 13:30:00', '2023-06-07 16:00:00', 'Visiting the Colloseum and the Ancient Market'),

(4, '2023-06-10 15:30:00', '2023-06-10 17:30:00', 'Exploring the canals of the city.'),

(5, '2023-05-13 15:30:00', '2023-05-13 17:30:00', 'Exploring the canals of the city.'),

(6, '2023-08-15 12:30:00', '2023-08-15 13:30:00', 'Visiting the Acropolis and The Acropolis Museum'),

(6, '2023-08-20 12:30:00', '2023-08-20 15:30:00', 'Visiting the White Tower Of Thessaloniki.'),

(7, '2023-09-13 17:30:00', '2023-09-13 19:30:00', 'Visiting the old part of Kalamata and watching a play at the Castle of Isabeau.'),

(8, '2023-12-16 12:30:00', '2023-12-16 15:30:00', 'Visiting the White Tower Of Thessaloniki.'),

(9, '2023-01-28 12:30:00', '2023-01-28 14:30:00', 'Eating at a local Restaurant.'),

(10, '2023-02-23 16:30:00', '2023-02-23 18:00:00', 'Visiting the Barcelona FC Stadium.'),

(10, '2023-02-25 15:00:00', '2023-02-25 17:30:00', 'Visiting a Spectacular small town close to Amsterdam, Delft.'),

(11, '2023-05-24 15:00:00', '2023-05-24 17:30:00', 'Visiting a Spectacular small town close to Amsterdam, Delft.'),

(12, '2023-07-08 16:30:00', '2023-07-08 18:30:00', 'Visiting the Big Ben.'),

(12, '2023-07-11 13:30:00', '2023-07-11 15:30:00', 'Visiting the Berlin Television Tower which is the tallest building in Berlin.'),

(13, '2023-03-03 12:00:00', '2023-03-03 14:30:00', 'Visiting the old part of Kalamata and watching a play at the Castle of Isabeau.'),

(14, '2023-04-02 11:30:00', '2023-04-02 13:00:00', 'Doing a tour of the villages of the island of Corfu!'),

(15, '2023-07-24 15:30:00', '2023-07-24 19:30:00', 'Swimming in the Local Beaches of the Island.'),

(16, '2023-09-11 13:30:00', '2023-09-11 15:30:00', 'Visiting the Berlin Television Tower which is the tallest building in Berlin.'),

(17, '2023-09-16 15:30:00', '2023-09-16 18:30:00', 'Taking a tour around this amazing island.'),

(18, '2023-12-13 16:30:00', '2023-12-13 18:00:00', 'Learning about the prosperous history of Kalambaka.'),

(18, '2023-12-15 12:30:00', '2023-12-15 13:30:00', 'Visiting the Acropolis and The Acropolis Museum.'),

(19, '2023-10-08 12:30:00', '2023-10-08 15:00:00', 'Eating at a local Restaurant.'),

(20, '2023-12-08 12:30:00', '2023-12-15 15:00:00', 'Visiting The Louvre Museum.'),

(20, '2023-12-12 13:30:00', '2023-12-12 16:00:00', 'Visiting the Colloseum and the Ancient Market.'),

(21, '2023-11-04 10:30:00', '2023-11-15 18:00:00', 'Visiting The Louvre Museum.');

CREATE TABLE destination(

dst\_id INT(11) NOT NULL AUTO\_INCREMENT,

dst\_name VARCHAR(50) DEFAULT 'Uknown' NOT NULL,

dst\_descr TEXT,

dst\_rtype ENUM('LOCAL', 'ABROAD'),

dst\_language VARCHAR(30) DEFAULT 'Uknown' NOT NULL,

dst\_location INT(11) NOT NULL,

PRIMARY KEY(dst\_id),

CONSTRAINT destination1 FOREIGN KEY (dst\_location) REFERENCES destination(dst\_id)

ON DELETE CASCADE ON UPDATE CASCADE

);

INSERT INTO destination VALUES

(null, 'Kalamata', 'Great place for fans of theatre and the arts to visit. The main draw here is the Castle of Isabeau.', 'LOCAL', 'Greek', 1),

(null, 'Nafpaktos', 'A stunning little port town across the Rio-Antirrio Bridge from Patras. Great place for a weekend trip away from Athens.', 'LOCAL', 'Greek', 2),

(null, 'Corfu', 'It boasts some of the most beautiful beaches in all of Europe.', 'LOCAL', 'Greek', 3),

(null, 'Kalambaka', 'This town is located in Meteora, giving it a stunning landscape as a backdrop to this picturesque city.', 'LOCAL', 'Greek', 4),

(null, 'Heraklion', 'Heraklion, capital of the largest Greek islands, is a great place for those interested in Ancient Greek history.', 'LOCAL', 'Greek', 5),

(null, 'Thessaloniki', 'Thessaloniki, known as the cultural capital of Greece, is also the second-largest city in Greece.', 'LOCAL', 'Greek', 6),

(null, 'Rhodes', 'For those interested in Medieval history. This city is also a UNESCO World Heritage Site.', 'LOCAL', 'Greek', 7),

(null, 'Athens', 'Lots of sightseeing here, especially for those interested in Ancient Greek history', 'LOCAL', 'Greek', 8),

(null, 'Venice', 'Save this one for your one true love, it’s a magic spot to share with that special someone.', 'ABROAD', 'Italian', 9),

(null, 'Amsterdam', 'This city is not only one of the most charming cities due to its character and being positioned on the canals, but it is also incredibly fun.', 'ABROAD', 'Dutch', 10),

(null, 'Barcelona', 'It is a hub of new trends in the world of culture, fashion and cuisine.', 'ABROAD', 'Spanish', 11),

(null, 'Rome', 'There’s no place like Rome and of course when in Rome, do as the Romans do.', 'ABROAD', 'Italian', 12),

(null, 'Berlin', 'Germany’s capital has more history than you can comprehend even after spending weeks in the city.', 'ABROAD', 'German', 13),

(null, 'London', 'There’s no other city quite like London and it should go without saying that you simply must visit once in your lifetime.', 'ABROAD', 'English', 14),

(null, 'Paris', 'Paris... a girl’s dream. The croissants, crepes, baguettes... yes, this is Paris. You will be in love.', 'ABROAD', 'French', 15),

(null, 'Rome', 'There’s no place like Rome and of course when in Rome, do as the Romans do.', 'ABROAD', 'Italian', 16);

CREATE TABLE travel\_to(

to\_tr\_id INT(11) NOT NULL,

to\_dst\_id INT(11) NOT NULL,

to\_arrival DATETIME,

to\_departure DATETIME,

PRIMARY KEY(to\_tr\_id, to\_dst\_id),

CONSTRAINT travel\_to1 FOREIGN KEY (to\_tr\_id) REFERENCES trip(tr\_id)

ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT travel\_to2 FOREIGN KEY (to\_dst\_id) REFERENCES destination(dst\_id)

ON DELETE CASCADE ON UPDATE CASCADE

);

INSERT INTO travel\_to VALUES

(1, 15, '2023-01-29 15:00:00', '2023-02-08 11:00:00'),

(2, 4, '2023-11-25 14:30:00', '2023-11-27 12:30:00'),

(2, 1, '2023-11-27 17:30:00', '2023-11-30 13:30:00'),

(3, 3, '2023-10-12 17:00:00', '2023-10-14 12:30:00'),

(3, 2, '2023-10-14 18:00:00', '2023-10-16 11:30:00'),

(4, 12, '2023-06-03 16:30:00', '2023-06-09 10:00:00'),

(4, 9, '2023-06-09 16:30:00', '2023-06-15 11:00:00'),

(5, 9, '2023-05-09 15:30:00', '2023-05-20 11:00:00'),

(6, 8, '2023-08-13 15:30:00', '2023-08-19 09:00:00'),

(6, 6, '2023-08-19 15:30:00', '2023-08-23 10:00:00'),

(7, 1, '2023-09-12 16:30:00', '2023-09-17 12:00:00'),

(8, 6, '2023-12-12 14:30:00', '2023-12-20 10:00:00'),

(9, 2, '2023-01-27 16:30:00', '2023-01-30 12:00:00'),

(10, 11, '2023-02-18 15:30:00', '2023-02-24 09:00:00'),

(10, 10, '2023-02-24 14:30:00', '2023-02-28 11:00:00'),

(11, 10, '2023-05-23 16:00:00', '2023-05-30 10:00:00'),

(12, 14, '2023-07-04 16:30:00', '2023-07-10 11:00:00'),

(12, 13, '2023-07-10 14:30:00', '2023-07-17 10:00:00'),

(13, 1, '2023-03-02 15:00:00', '2023-03-08 12:00:00'),

(14, 3, '2023-04-01 14:30:00', '2023-04-06 10:00:00'),

(15, 7, '2023-07-22 15:30:00', '2023-07-28 10:00:00'),

(16, 13, '2023-09-09 13:30:00', '2023-09-15 11:00:00'),

(17, 5, '2023-09-15 14:30:00', '2023-09-19 12:00:00'),

(18, 4, '2023-12-12 16:30:00', '2023-02-14 10:00:00'),

(18, 8, '2023-12-14 12:30:00', '2023-02-17 12:00:00'),

(19, 2, '2023-10-07 15:30:00', '2023-10-10 13:00:00'),

(20, 15, '2023-12-04 13:30:00', '2023-12-11 12:00:00'),

(20, 12, '2023-12-11 14:30:00', '2023-12-15 12:00:00'),

(21, 16, '2023-11-05 10:30:00', '2023-11-05 18:00:00');

CREATE TABLE reservation(

res\_tr\_id INT(11) NOT NULL,

res\_seatnum TINYINT(4) NOT NULL,

res\_name VARCHAR(20) DEFAULT 'Uknown' NOT NULL,

res\_lname VARCHAR(20) DEFAULT 'Uknown' NOT NULL,

res\_isadult ENUM('ADULT', 'MINOR'),

PRIMARY KEY(res\_tr\_id, res\_seatnum),

CONSTRAINT reservation1 FOREIGN KEY (res\_tr\_id) REFERENCES trip(tr\_id)

ON DELETE CASCADE ON UPDATE CASCADE

);

INSERT INTO reservation VALUES

(1, 15, 'Giorgos', 'Papadopoulos', 'ADULT'),

(1, 14, 'Maria', 'Kokkinou', 'ADULT'),

(1, 13, 'Takis', 'Papadopoulos', 'MINOR'),

(2, 1, 'Giannis', 'Dimitriadis', 'ADULT'),

(2, 2, 'Leonidas', 'Dimitriadis', 'MINOR'),

(2, 3, 'Tasos', 'Dimitriadis', 'MINOR'),

(2, 4, 'Katerina', 'Dimitriadi', 'MINOR'),

(2, 5, 'Petroula', 'Asimakopoulou', 'ADULT'),

(3, 1, 'Leonidas', 'Asimakopoulos', 'ADULT'),

(3, 2, 'Konstantinos', 'Asimakopoulos', 'ADULT'),

(4, 5, 'Babis', 'Papageorgiou', 'ADULT'),

(4, 6, 'Danae', 'Papageorgiou', 'MINOR'),

(5, 8, 'Filia', 'Papakonstantinou', 'ADULT'),

(5, 9, 'Maria', 'Papakonstantinou', 'MINOR'),

(6, 2, 'John', 'Smith', 'ADULT'),

(6, 3, 'Maria', 'Smith', 'MINOR'),

(6, 4, 'Natasha', 'Smith', 'MINOR'),

(6, 12, 'Clint', 'Smith', 'ADULT'),

(6, 13, 'Peter', 'Jones', 'ADULT'),

(6, 50, 'Giorgia', 'Dourou', 'ADULT'),

(7, 4, 'Roula', 'Andreadi', 'ADULT'),

(7, 5, 'Konstantina', 'Andreadi', 'MINOR'),

(7, 6, 'Eirini', 'Andreadi', 'MINOR'),

(8, 11, 'May', 'Jones', 'ADULT'),

(8, 15, 'Sarah', 'Hans', 'ADULT'),

(8, 13, 'Eleni', 'Dimoula', 'ADULT'),

(8, 14, 'Anastasia', 'Dimoula', 'MINOR'),

(9, 15, 'Iasonas', 'Pavlopoulos', 'ADULT'),

(9, 14, 'Nefeli', 'Pavlopoulou', 'ADULT'),

(9, 8, 'Iasonas', 'Stamoulis', 'ADULT'),

(10, 3, 'Roubini', 'Louloudi', 'ADULT'),

(10, 4, 'Maria', 'Louloudi', 'MINOR'),

(10, 10, 'Princess', 'Violet', 'Adult'),

(11, 24, 'Giorgos', 'Papadopoulos', 'ADULT'),

(11, 25, 'Petros', 'Papadopoulos', 'MINOR'),

(13, 15, 'Giorgos', 'Papandreou', 'ADULT'),

(13, 12, 'Tasos', 'Helas', 'ADULT'),

(13, 20, 'Anaximandos', 'Asimakopoulos', 'ADULT'),

(13, 21, 'Andromaxi', 'Asimakopoulou', 'MINOR'),

(14, 3, 'Sam', 'Parker', 'ADULT'),

(14, 4, 'Happy', 'Smith', 'ADULT'),

(15, 6, 'Maria', 'Papadopoulou', 'ADULT'),

(15, 7, 'Nikolas', 'Doukas', 'ADULT'),

(15, 1, 'Marios', 'Panagiotou', 'ADULT'),

(16, 1, 'Patricia', 'Pilster', 'ADULT'),

(16, 2, 'Alice', 'Drake', 'ADULT'),

(16, 3, 'Giorgia', 'Dourou', 'MINOR'),

(16, 4, 'Jordan', 'Wall', 'MINOR'),

(17, 4, 'Jan', 'Greene', 'ADULT'),

(17, 5, 'Dylan', 'Mendez', 'MINOR'),

(19, 15, 'Kennedy', 'Summers', 'ADULT'),

(19, 16, 'Monserrat', 'Carter', 'ADULT'),

(19, 17, 'Victoria', 'Holden', 'ADULT'),

(20, 3, 'Mitchell', 'Scott', 'ADULT'),

(20, 7, 'Trevor', 'Braun', 'ADULT'),

(20, 23, 'Jaycee', 'Douglas', 'ADULT'),

(20, 14, 'Amir', 'Durham', 'ADULT'),

(20, 32, 'Alisha', 'Hunter', 'ADULT'),

(20, 28, 'Mateo', 'Garrett', 'ADULT');

CREATE TABLE offers(

off\_code TINYINT(4) NOT NULL AUTO\_INCREMENT,

tr\_start DATETIME NOT NULL,

tr\_end DATETIME NOT NULL,

cost FLOAT(7,2) NOT NULL,

dest\_id INT(11) NOT NULL,

PRIMARY KEY(off\_code),

CONSTRAINT off1 FOREIGN KEY(dest\_id) REFERENCES destination(dst\_id)

ON DELETE CASCADE ON UPDATE CASCADE

);

INSERT INTO offers VALUES

(null, '2023-07-05 09:00:00', '2023-09-27 09:00:00', 456.53, 7),

(null, '2023-11-10 11:00:00', '2023-12-30 11:00:00', 654.53, 10),

(null, '2023-03-08 12:00:00', '2023-05-15 12:00:00', 923.53, 15);

CREATE TABLE reservation\_offers(

res\_of\_id INT(11) NOT NULL AUTO\_INCREMENT,

first\_name CHAR(20) NOT NULL,

last\_name CHAR(20) NOT NULL,

off\_id TINYINT(4) NOT NULL,

adv\_pay FLOAT(7,2) NOT NULL,

PRIMARY KEY(res\_of\_id),

CONSTRAINT res\_off1 FOREIGN KEY(off\_id) REFERENCES offers(off\_code)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE logging(

log\_id INT(11) NOT NULL AUTO\_INCREMENT,

chang ENUM('INSERT', 'UPDATE', 'DELETE') NOT NULL,

tab\_ch ENUM('trip', 'reservation', 'event', 'travel\_to', 'destination') NOT NULL,

it\_lname CHAR(20) NOT NULL,

curr\_date DATETIME NOT NULL,

PRIMARY KEY(log\_id)

);

CREATE TABLE user(

lname CHAR(20) NOT NULL,

PRIMARY KEY(lname)

);

**Κεφάλαιο 2**

Stored Procedures

3.1.3.1

**Κώδικας:**

DROP PROCEDURE IF EXISTS new\_driver;

DELIMITER $

CREATE PROCEDURE new\_driver(drvAT CHAR(10), first\_name VARCHAR(20), last\_name VARCHAR(20), salary FLOAT(7,2), license ENUM('A', 'B', 'C', 'D'), dr\_route ENUM('LOCAL', 'ABROAD'), experience TINYINT(4))

BEGIN

    DECLARE not\_found INT;

    DECLARE min\_br TINYINT;

    DECLARE temp TINYINT;

    DECLARE branches TINYINT;

    DECLARE first\_br TINYINT;

    DECLARE drvcursor CURSOR FOR

    SELECT br\_code FROM branch;

    DECLARE CONTINUE HANDLER FOR NOT FOUND

    SET not\_found=1;

    SET min\_br=1;

    SELECT COUNT(\*) INTO first\_br FROM driver

    INNER JOIN worker ON drv\_AT=wrk\_AT

    WHERE wrk\_br\_code=1;

    SET not\_found=0;

    OPEN drvcursor;

    REPEAT

        FETCH drvcursor INTO branches;

        IF(not\_found=0)

        THEN

            SELECT COUNT(\*) INTO temp FROM driver

            INNER JOIN worker ON drv\_AT=wrk\_AT

            WHERE wrk\_br\_code=branches;

            IF(temp<first\_br)

            THEN

                SELECT branches INTO min\_br;

                SET first\_br=temp;

            END IF;

        END IF;

    UNTIL(not\_found=1)

    END REPEAT;

    INSERT INTO worker VALUES

    (drvAT, first\_name, last\_name, salary, min\_br);

    INSERT INTO driver VALUES

    (drvAT, license, dr\_route, experience);

END$

DELIMITER ;

**Η παραδοχή μας:** Σε αυτό το stored procedure, ο στόχος είναι η προσθήκη ενός νέου οδηγού(driver) στο υποκατάστημα(branch) που έχει τους λιγότερους.

Παραδοχή που ακολουθήσαμε: Αν υπάρχει παραπάνω από ένα υποκατάστημα με τον μικρότερο αριθμό οδηγών, ο καινούριος οδηγός προστίθεται αυτόματα σε αυτό με το μικρότερο br\_code.

Επίσης θεωρούμε πως από την στιγμή που ζητείται να δοθούν ως ορίσματα, χαρακτηριστικά που ανήκουν στον πίνακα worker, ότι αυτός ο driver δεν έχει εισαχθεί ακόμα ούτε ως worker. Για αυτόν τον λόγο στο τέλος τον εισάγουμε και στον πίνακα worker και στον πίνακα driver.

**Επεξήγηση Κώδικα:** Για την επίτευξη του παραπάνω ζητήματος, δημιουργήσαμε έναν cursor ο οποίος κρατάει τα br\_code από όλα τα branches.

Έπειτα με ένα Select Into Statement, βάλαμε στην τοπική μεταβλητή first\_br, τον αριθμό των drivers που υπάρχουν στο υποκατάστημα με br\_code=1. Αυτό θα το χρησιμοποιήσουμε για να συγκρίνουμε αργότερα το πλήθος των οδηγών των υπόλοιπων υποκαταστημάτων και να κρατήσουμε αυτό με το μικρότερο.

Με την χρήση του cursor λοιπόν περνάμε από όλα τα υποκαταστήματα και μετράμε το πλήθος των οδηγών που ανήκουν σε καθένα από αυτά (count(\*)). Έπειτα, αν το πλήθος αυτό είναι μικρότερο από το “default” που ορίσαμε, δηλαδή αυτό του υποκαταστήματος με br\_code=1, τότε αποθηκεύουμε στην μεταβλητή min\_br το br\_code αυτού του υποκαταστήματος. (Η min\_br έχει αρχικοποιηθεί νωρίτερα με την τιμή 1, ώστε αν δεν βρεθεί branch με μικρότερο αριθμό οδηγών από το “default” να κρατηθεί το 1)

Όταν συγκρίνουμε όλα τα branches, βγαίνουμε από το loop του cursor, και κάνουμε add και στον worker και στον driver τα στοιχεία του driver που δόθηκαν σαν ορίσματα στο stored procedure.

Table

Description automatically generated**Παράδειγμα Υλοποίησης:**

Text

Description automatically generatedΠρώτα ας παρατηρήσουμε το πλήθος των οδηγών σε κάθε υποκατάστημα πριν καλέσουμε την Stored Procedure, με την χρήση ενός Select Statement:

Έπειτα ας δούμε τους πίνακες worker και driver:

Text

Description automatically generated(O πίνακας worker είναι πολύ μεγάλος για να χωρέσει στην αναφορά όποτε θα κάνουμε ένα Select με count για να δούμε πόσοι workers υπάρχουν πριν καλέσουμε την συνάρτηση):

Table

Description automatically generated

Με βάση τα παραπάνω στοιχεία και τις παραδοχές που έχουμε κάνει, παρατηρούμε πως ο καινούριος driver που θα προσθέσουμε θα πρέπει να εισαχθεί στο branch με br\_code = 1.

Καλούμε την Stored Procedure:

Graphical user interface, text, application

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedΑς συγκρίνουμε τους πίνακες τώρα με τους αντίστοιχους πίνακες πριν:



Παρατηρούμε λοιπόν απευθείας ότι το branch με br\_code=1, έχει έναν καινούριο driver.

Από τον πίνακα driver και worker παρατηρούμε πως έχει προστεθεί όντως ο σωστός driver με τα σωστά στοιχεία. Επίσης παρατηρούμε πως όντως αυτός ο worker δεν υπήρχε πριν καλέσουμε την Stored Procedure καθώς βλέπουμε ότι ο αριθμός των workers αυξήθηκε από 43 σε 44.

3.1.3.2

**Κώδικας:**

DROP PROCEDURE IF EXISTS date\_check;

DELIMITER $

CREATE PROCEDURE date\_check(br\_code INT, date1 DATE, date2 DATE)

BEGIN

DECLARE dates DATETIME;

DECLARE tripid INT;

DECLARE reservations INT;

DECLARE max\_seats tinyINT;

DECLARE seatdiff tinyINT;

DECLARE not\_found INT;

DECLARE tridcursor CURSOR FOR

SELECT tr\_id FROM trip WHERE tr\_br\_code=br\_code;

DECLARE CONTINUE HANDLER FOR NOT FOUND

SET not\_found=1;

SET not\_found=0;

DROP TABLE IF EXISTS new;

CREATE TABLE new(

id INT AUTO\_INCREMENT NOT NULL,

trcost FLOAT(7,2),

maxseats\_ INT,

reservations\_ INT,

seatdiff\_ TINYINT,

drv\_name\_ CHAR(20),

drv\_lname\_ CHAR(20),

gui\_name\_ CHAR(20),

gui\_lname\_ CHAR(20),

tr\_dep\_ DATETIME,

tr\_ret\_ DATETIME,

PRIMARY KEY(id)

);

OPEN tridcursor;

REPEAT

FETCH tridcursor INTO tripid;

IF(not\_found=0)

THEN

SELECT tr\_departure INTO dates FROM trip

WHERE tr\_id=tripid;

IF (DATEDIFF(dates, date1)>0 && DATEDIFF(dates, date2)<0)

THEN

SELECT COUNT(\*) INTO reservations FROM reservation

WHERE tripid=res\_tr\_id;

SELECT tr\_maxseats INTO max\_seats FROM trip

WHERE tr\_id=tripid;

SET seatdiff=max\_seats-reservations;

INSERT INTO new

SELECT null, tr\_cost, tr\_maxseats, reservations,

seatdiff, a.wrk\_name, a.wrk\_lname, b.wrk\_name,

b.wrk\_lname, tr\_departure, tr\_return

FROM trip

INNER JOIN worker AS a ON tr\_drv\_AT=a.wrk\_AT

INNER JOIN worker AS b ON tr\_gui\_AT=b.wrk\_AT

WHERE tr\_id=tripid;

END IF;

END IF;

UNTIL(not\_found=1)

END REPEAT;

SELECT \* FROM new;

END$

DELIMITER ;

**Επεξήγηση Κώδικα:** Αρχικά δημιουργούμε έναν cursor, ο οποίος κρατάει τα tr\_ids των trips τα οποία διοργανώνονται από το υποκατάστημα που έχει δοθεί ως όρισμα. Δημιουργούμε έναν πίνακα στον οποίο θα αποθηκεύσουμε όλες τις τιμές που θα πάρουμε μέσα από το repeat του cursor ώστε να μπορέσουμε να τις τυπώσουμε όλες στο τέλος σε έναν πίνακα. Έπειτα με repeat του cursor, περνάμε για καθένα από αυτά τα ταξίδια την τιμή tr\_departure τους, στην μεταβλητή dates. Στην συνέχεια με την χρήση ενός If, ελέγχουμε αν αυτή η ημερομηνία βρίσκεται ανάμεσα στις δύο που δόθηκαν στην αρχή. Αν βρίσκεται, μετράμε πόσα reservations έχουν γίνει σε αυτό το ταξίδι και αποθηκεύουμε την τιμή αυτή στην μεταβλητή reservations. Επιπρόσθετα, αποθηκεύουμε στην μεταβλητή max\_seats, τον μέγιστο αριθμό θέσεων που υποστηρίζει το ταξίδι και πραγματοποιούμε και μία αφαίρεση για να υπολογίσουμε πόσες από αυτές τις θέσεις είναι ακόμα κενές. Τέλος περνάμε όλες τις πληροφορίες στον πίνακα new. Βγαίνουμε από το repeat του cursor και κάνουμε έναν SELECT \* στον πίνακα new για να τυπωθούν τα επιθυμητά αποτελέσματα.

**Παράδειγμα Υλοποίησης:**

Αρχικά ας δούμε τον πίνακα trip για να δούμε αναλυτικά όλα τα ταξίδια και τις ημερομηνίες αναχώρησης του καθενός από αυτά:

A picture containing table

Description automatically generated

Graphical user interface, application, Teams

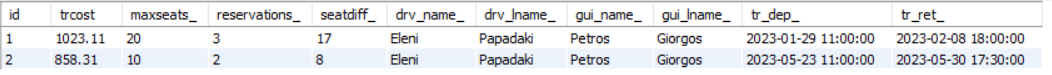
Description automatically generated

Καλούμε την συνάρτηση:



Από τον παραπάνω πίνακα παρατηρούμε πως μόνο ένα ταξίδι καλύπτει τις προϋποθέσεις των ορισμάτων της Stored Procedure, και αυτό είναι και αυτό που εμφανίζει και η Stored Procedure.

Καλούμε ξανά την Stored Procedure με άλλα ορίσματα για να σιγουρευτούμε ότι δουλεύει για για περισσότερα από 1 ταξίδια:



Παρατηρούμε πως ακόμα και για αυτήν την περίπτωση τα αποτελέσματα είναι τα σωστά.

3.1.3.3

**Κώδικας:**

DROP PROCEDURE IF EXISTS admin\_check;

DELIMITER $

CREATE PROCEDURE admin\_check(work\_name CHAR(20), work\_lname CHAR(20))

BEGIN

    DECLARE detail ENUM('LOGISTICS', 'ADMINISTRATIVE', 'ACCOUNTING');

    DECLARE admid CHAR(10);

    SELECT adm\_type, adm\_AT INTO detail, admid FROM admin

    INNER JOIN worker ON adm\_AT=wrk\_AT

    WHERE work\_name=wrk\_name AND work\_lname=wrk\_lname;

    IF (detail!='ADMINISTRATIVE' AND detail IS NOT NULL)

    THEN

        DELETE FROM admin WHERE adm\_AT=admid;

        DELETE FROM worker WHERE wrk\_AT=admid;

    ELSEIF (detail IS NULL)

    THEN

        SELECT('This Worker is not an Admin!');

    ELSE

        SELECT('You cannot Delete this User. This User is an Administrative.');

    END IF;

END$

DELIMITER ;

**Επεξήγηση Κώδικα:** Αρχικά δημιουργούμε δύο μεταβλητές, την detail και την admid και περνάμε σε αυτές τις τιμές adm\_type και adm\_AT αντίστοιχα (του πίνακα admin) για τις περιπτώσεις που το όνομα και το επώνυμο στον πίνακα admin είναι ίδια με αυτά που δόθηκαν σαν όρισμα στην Stored Procedure.

Έπειτα δημιουργούμε ένα If, και ελέγχουμε αν το type του worker που δόθηκε δεν είναι null και ταυτόχρονα δεν είναι και ‘ADMINISTRATIVE’. Αυτό το κάνουμε για να ελέγξουμε αν ο εργάτης που δόθηκε είναι admin αλλά και ταυτόχρονα αν ανήκει στις κατηγορίες των Admin που μπορούμε να διαγράψουμε ή όχι. Αν λοιπόν δεν είναι null και δεν ανήκει στην κατηγορία ‘ADMINISTRATIVE’ τότε κάνουμε αυτόν τον εργάτης delete και από τον πίνακα worker αλλά και από τον πίνακα admin. Επιπρόσθετα, έχουμε και ένα IF ELSE, το οποίο ελέγχει αν ο εργάτης που δόθηκε σαν όρισμα είναι admin, αν δεν είναι τυπώνει κατάλληλο μήνυμα. Τέλος έχουμε την περίπτωση που ο worker είναι admin και ταυτόχρονα ανήκει στην κατηγορία ‘ADMINISTRATIVE’, τότε τυπώνουμε μήνυμα που λέει ότι δεν μπορεί να διαγραφεί αυτός ο Worker καθώς ανήκει στην παραπάνω κατηγορία.

**Παράδειγμα Υλοποίησης:**

Graphical user interface, text

Description automatically generatedΠρώτα ας κάνουμε Print όλους τους Admin, πριν καλέσουμε την stored procedure:

Έπειτα θα κάνουμε 3 διαφορετικές δοκιμές.

Η πρώτη θα καλεί την Stored Procedure με ορίσματα που ανήκουν σε Admin τύπου ADMINISTRATIVE. Η δεύτερη θα καλεί την Stored Procedure με ορίσματα Admin που δεν είναι τύπου ADMINISTRATIVE. Και η τρίτη θα καλεί την Stored Procedure με ορίσματα που δεν ανήκουν σε κάποιον Admin.

1ο Call:



2ο Call:



Graphical user interface, text, application

Description automatically generated

Παρατηρούμε ότι ο Admin με το όνομα Anastasia Petropoulou, δεν υπάρχει πια στην λίστα με τους Admin.

3ο Call:



Παρατηρούμε πως ο εργάτης Petros Giorgos, δεν έχει εμφανιστεί στον πιο πάνω πίνακα, δηλαδή δεν είναι Admin, αρά τυπώνεται σωστά το παραπάνω μήνυμα και δεν επιτρέπει την διαγραφή του.

3.1.3.4 α)

**Κώδικας:**

DROP PROCEDURE IF EXISTS prepaid;

DELIMITER $

CREATE PROCEDURE prepaid(cost1 FLOAT(7,2), cost2 FLOAT(7,2))

BEGIN

    DECLARE costs FLOAT(7,2);

    DECLARE res\_id INT(11);

    DECLARE not\_found INT;

    DECLARE precursor CURSOR FOR

    SELECT res\_of\_id FROM reservation\_offers;

    DECLARE CONTINUE HANDLER FOR NOT FOUND

    SET not\_found=1;

    SET not\_found=0;

    DROP TABLE IF EXISTS final\_paid;

    CREATE TABLE final\_paid(

        last\_name CHAR(20),

        first\_name CHAR(20),

        id INT(11),

        PRIMARY KEY(id)

    );

    OPEN precursor;

    REPEAT

        FETCH precursor INTO res\_id;

        IF(not\_found=0)

        THEN

            SELECT adv\_pay INTO costs FROM reservation\_offers

            WHERE res\_of\_id=res\_id;

                INSERT INTO final\_paid

                SELECT last\_name, first\_name, res\_of\_id FROM reservation\_offers

                USE INDEX (reservation\_offers\_adv\_pay\_idx)

                WHERE res\_of\_id=res\_id && adv\_pay >=cost1 && adv\_pay <= cost2;

        END IF;

    UNTIL(not\_found=1)

    END REPEAT;

    SELECT last\_name AS Last\_Name, first\_name AS First\_Name FROM final\_paid;

END$

DELIMITER ;

**Επεξήγηση Κώδικα:** Αρχικά δημιουργούμε έναν cursor στον οποίο περνάμε όλα τα res\_od\_id από τον πίνακα reservation\_offers. Έπειτα δημιουργούμε ένα καινούριο table, το final\_paid στο οποίο θα περάσουμε στην συνέχεια όλα τα δεδομένα που θέλουμε να τυπώσουμε, για να τυπωθούν όλα μαζί σε έναν πίνακα. Έπειτα ξεκινάει το repeat του cursor. Σε αυτό περνάμε για κάθε τιμή του cursor, το ποσό που έχουν δώσει ως προκαταβολή, στην μεταβλητή costs. Έπειτα εισάγουμε στον πίνακα final\_paid τα δεδομένα που θα τυπώσουμε. Σε αυτό το κομμάτι χρησιμοποιούμε και τον index, παρόλο που δεν κάνει καμία διαφορά στον χρόνο που χρειάζεται για να εμφανιστούν τα αποτελέσματα. Τέλος βγαίνουμε από το repeat και τυπώνουμε όλο τον πίνακα final\_paid

A picture containing table

Description automatically generated**Παράδειγμα Υλοποίησης:**

Εδώ καθώς το πλήθος των εγγραφών είναι απίστευτα μεγάλο, δεν γίνεται να έχουμε παράδειγμα πριν την υλοποίηση του stored procedure. Οπότε θα παραθέσουμε μόνο τον πίνακα που τυπώνεται αφού καλέσουμε την συνάρτηση (και αυτόν όχι ολόκληρο)



3.1.3.4 β)

**Κώδικας:**

DROP PROCEDURE IF EXISTS check\_offers;

DELIMITER $

CREATE PROCEDURE check\_offers(lname CHAR(20))

BEGIN

    DECLARE res\_id INT(11);

    DECLARE not\_found INT;

    DECLARE ofcursor CURSOR FOR

    SELECT res\_of\_id FROM reservation\_offers WHERE last\_name=lname;

    DECLARE CONTINUE HANDLER FOR NOT FOUND

    SET not\_found=1;

    SET not\_found=0;

    DROP TABLE IF EXISTS new\_res\_off;

    CREATE TABLE new\_res\_off(

        offid TINYINT(4),

        resid INT(11),

        first\_name CHAR(20),

        last\_name CHAR(20),

        PRIMARY KEY(resid)

    );

    OPEN ofcursor;

    REPEAT

        FETCH ofcursor INTO res\_id;

        IF(not\_found=0)

        THEN

            INSERT INTO new\_res\_off

            SELECT off\_id, res\_of\_id, first\_name, last\_name FROM reservation\_offers

            WHERE last\_name=lname AND res\_of\_id=res\_id;

        END IF;

    UNTIL(not\_found=1)

    END REPEAT;

    SELECT offid AS Offer\_ID, first\_name AS First\_Name, last\_name AS Last\_Name FROM new\_res\_off

    ORDER BY offid;

END$

DELIMITER ;

**Επεξήγηση Κώδικα:** Αρχικά ξεκινάμε δημιουργώντας έναν cursor ο οποίος κρατάει όλα τα res\_of\_it από τον πίνακα reservation\_offers, των οποίων τα lname είναι ίδιο με το lastname που δόθηκε σαν όρισμα στο Stored Procedure. Έπειτα όπως και στο προηγούμενο παράδειγμα δημιουργούμε έναν πίνακα στον οποίο θα αποθηκεύσουμε αργότερα όλες τις τιμές που θα θέλουμε να εκτυπώσουμε, για να μπορέσουμε να τις εμφανίσουμε όλες στον ίδιο πίνακα. Ξεκινάμε το repeat του cursor, και για κάθε διαφορετική τιμή του περνάμε μέσα στον νέο μας πίνακα, τα στοιχεία που ζητούνται από την εκφώνηση. Όταν περάσουμε από όλες τις τιμές του cursor, βγαίνουμε από το repeat και κάνουμε έναν Select για τον πίνακα που δημιουργήσαμε. Επίσης ορίζουμε να εμφανιστούν με βάση το offid τους.

Table

Description automatically generated**Παράδειγμα Υλοποίησης:**

Όπως και στο προηγούμενο παράδειγμα, καθώς το πλήθος των εγγραφών είναι απίστευτα μεγάλο, δεν γίνεται να έχουμε παράδειγμα πριν την υλοποίηση του stored procedure. Οπότε θα παραθέσουμε μόνο τον πίνακα που τυπώνεται αφού καλέσουμε την συνάρτηση (και αυτόν όχι ολόκληρο):



Όπως βλέπουμε τυπώνονται με βάση το offid όπως ακριβώς ζητείται στην εκφώνηση. (Αυτή δεν είναι η αρχή της εκτύπωσης του πίνακα αλλά το κομμάτι στο οποίο βλέπουμε ότι περνάει από τις εγγραφές με offid=1 σε αυτές με offid=2.

Έξτρα Stored Procedures

* **Κώδικας:**

DROP PROCEDURE IF EXISTS seats\_trip;

DELIMITER $

CREATE PROCEDURE seats\_trip(IN res\_id INT, OUT res TINYINT)

BEGIN

DECLARE currentSeats TINYINT;

DECLARE rsvid INT;

DECLARE not\_found INT;

DECLARE max TINYINT;

DECLARE tripcursor CURSOR FOR

SELECT res\_tr\_id FROM reservation WHERE res\_tr\_id=res\_id;

DECLARE CONTINUE HANDLER FOR NOT FOUND

SET not\_found=1;

SELECT tr\_maxseats INTO max FROM trip WHERE tr\_id=res\_id;

SET not\_found=0;

OPEN tripcursor;

REPEAT

FETCH tripcursor INTO rsvid;

IF(not\_found=0)

THEN

SELECT COUNT(\*) INTO currentSeats FROM reservation WHERE res\_tr\_id=rsvid;

END IF;

UNTIL(not\_found=1)

END REPEAT;

IF (currentSeats < max OR currentSeats IS NULL)

THEN SET res=1;

ELSE SET res=0;

END IF;

END$

DELIMITER ;

**Επεξήγηση Κώδικα:** Για την καλύτερη λειτουργία του προγράμματος κρατήσαμε στον κώδικας μας το stored procedure που είχε ζητηθεί να δημιουργήσουμε ως 5η εργαστηριακή άσκηση. Αυτό το stored procedure έχει ως στόχο, πριν από κάθε insert στον πίνακα reservation, να ελέγχει αν υπάρχουν διαθέσιμες θέσεις για το ταξίδι στο οποίο αναφέρεται το insert. Έπειτα με ένα trigger που θα αναλύσουμε στην συνέχεια, απαγορεύει το insert σε περίπτωση που δεν υπάρχουν άλλες διαθέσιμες θέσεις.

Για να υλοποιήσουμε αυτό το Stored Procedure, ξεκινάμε δημιουργώντας έναν cursor, που ονομάζουμε tripcursor. Σε αυτόν περνάμε τα res\_tr\_id των reservation που έχουν την ίδια τιμή με το δοσμένο res\_id.

Στην συνέχεια, περνάμε στην μεταβλητή max, το tr\_maxseats από τον πίνακα trip.

Έπειτα ξεκινάμε το repeat του tripcursor και με count(\*) μετράμε όλες τις κρατήσεις που έχουν γίνει για το συγκεκριμένο trip. Τέλος βγαίνουμε από το repeat και με ένα If Statement, ελέγχουμε αν υπάρχουν διαθέσιμες θέσεις ή όχι. Αν υπάρχουν, θέτουμε την μεταβλητή res=1, ενώ αν δεν υπάρχουν, θέτουμε res=0. Αυτά θα μας χρειαστούν στην συνέχεια στον trigger.

**Παράδειγμα Υλοποίησης:**

Καθώς αυτό το Stored Procedure συνδέεται άμεσα με ένα trigger δεν μπορούμε να δούμε την πραγματική του λειτουργία χωρίς αυτό. Παρόλαυτα, μπορούμε να δούμε αν η μεταβλητή res, παίρνει στο τέλος την τιμή που πρέπει να πάρει.

Θα καλέσουμε λοιπόν αυτό το Stored Procedure δύο φορές, μία για ένα ταξίδι που δεν έχει άλλες διαθέσιμες θέσεις, και μία για ένα που έχει.

Για τις ανάγκες του παραδείγματος θα πρέπει να προσθέσουμε στον κώδικα μια έξτρα εντολή select για να εκτυπώσουμε το res στην οθόνη μας.

Ας ξεκινήσουμε με αυτό που δεν έχει άλλες διαθέσιμες θέσεις, το res του θα πρέπει να είναι 0.



Και τώρα ας δούμε για ένα trip που έχει ακόμα διαθέσιμες θέσεις.



1. Triggers

3.1.4.1.

**Κώδικας:**

DROP TRIGGER IF EXISTS logUTrip;

DELIMITER $

CREATE TRIGGER logUTrip AFTER UPDATE ON trip

FOR EACH ROW

BEGIN

    DECLARE curr\_date DATETIME;

    DECLARE last\_name CHAR(20);

    SELECT lname INTO last\_name FROM user;

    SELECT now() INTO curr\_date;

    INSERT INTO logging VALUES

    (null, 'Update', 'trip', last\_name, curr\_date);

END$

DELIMITER ;

DROP TRIGGER IF EXISTS logDTrip;

DELIMITER $

CREATE TRIGGER logDTrip AFTER DELETE ON trip

FOR EACH ROW

BEGIN

    DECLARE curr\_date DATETIME;

    DECLARE last\_name CHAR(20);

    SELECT lname INTO last\_name FROM user;

    SELECT now() INTO curr\_date;

    INSERT INTO logging VALUES

    (null, 'Delete', 'trip', last\_name, curr\_date);

END$

DELIMITER ;

DROP TRIGGER IF EXISTS logITrip;

DELIMITER $

CREATE TRIGGER logITrip AFTER INSERT ON trip

FOR EACH ROW

BEGIN

    DECLARE curr\_date DATETIME;

    DECLARE last\_name CHAR(20);

    SELECT lname INTO last\_name FROM user;

    SELECT now() INTO curr\_date;

    INSERT INTO logging VALUES

    (null, 'Insert', 'trip', last\_name, curr\_date);

END$

DELIMITER ;

DROP TRIGGER IF EXISTS logURes;

DELIMITER $

CREATE TRIGGER logURes AFTER UPDATE ON reservation

FOR EACH ROW

BEGIN

    DECLARE curr\_date DATETIME;

    DECLARE last\_name CHAR(20);

    SELECT lname INTO last\_name FROM user;

    SELECT now() INTO curr\_date;

    INSERT INTO logging VALUES

    (null, 'Update', 'reservation', last\_name, curr\_date);

END$

DELIMITER ;

DROP TRIGGER IF EXISTS logDRes;

DELIMITER $

CREATE TRIGGER logDRes AFTER DELETE ON reservation

FOR EACH ROW

BEGIN

    DECLARE curr\_date DATETIME;

    DECLARE last\_name CHAR(20);

    SELECT lname INTO last\_name FROM user;

    SELECT now() INTO curr\_date;

    INSERT INTO logging VALUES

    (null, 'Delete', 'reservation', last\_name, curr\_date);

END$

DELIMITER ;

DROP TRIGGER IF EXISTS logIRes;

DELIMITER $

CREATE TRIGGER logIRes AFTER INSERT ON reservation

FOR EACH ROW

BEGIN

    DECLARE curr\_date DATETIME;

    DECLARE last\_name CHAR(20);

    SELECT lname INTO last\_name FROM user;

    SELECT now() INTO curr\_date;

    INSERT INTO logging VALUES

    (null, 'Insert', 'reservation', last\_name, curr\_date);

END$

DELIMITER ;

DROP TRIGGER IF EXISTS logUEv;

DELIMITER $

CREATE TRIGGER logUEv AFTER UPDATE ON event

FOR EACH ROW

BEGIN

    DECLARE curr\_date DATETIME;

    DECLARE last\_name CHAR(20);

    SELECT lname INTO last\_name FROM user;

    SELECT now() INTO curr\_date;

    INSERT INTO logging VALUES

    (null, 'Update', 'event', last\_name, curr\_date);

END$

DELIMITER ;

DROP TRIGGER IF EXISTS logDEv;

DELIMITER $

CREATE TRIGGER logDEv AFTER DELETE ON event

FOR EACH ROW

BEGIN

    DECLARE curr\_date DATETIME;

    DECLARE last\_name CHAR(20);

    SELECT lname INTO last\_name FROM user;

    SELECT now() INTO curr\_date;

    INSERT INTO logging VALUES

    (null, 'Delete', 'event', last\_name, curr\_date);

END$

DELIMITER ;

DROP TRIGGER IF EXISTS logIEv;

DELIMITER $

CREATE TRIGGER logIEv AFTER INSERT ON event

FOR EACH ROW

BEGIN

    DECLARE curr\_date DATETIME;

    DECLARE last\_name CHAR(20);

    SELECT lname INTO last\_name FROM user;

    SELECT now() INTO curr\_date;

    INSERT INTO logging VALUES

    (null, 'Insert', 'event', last\_name, curr\_date);

END$

DELIMITER ;

DROP TRIGGER IF EXISTS logUTravel;

DELIMITER $

CREATE TRIGGER logUTravel AFTER UPDATE ON travel\_to

FOR EACH ROW

BEGIN

    DECLARE curr\_date DATETIME;

    DECLARE last\_name CHAR(20);

    SELECT lname INTO last\_name FROM user;

    SELECT now() INTO curr\_date;

    INSERT INTO logging VALUES

    (null, 'Update', 'travel\_to', last\_name, curr\_date);

END$

DELIMITER ;

DROP TRIGGER IF EXISTS logDTravel;

DELIMITER $

CREATE TRIGGER logDTravel AFTER DELETE ON travel\_to

FOR EACH ROW

BEGIN

    DECLARE curr\_date DATETIME;

    DECLARE last\_name CHAR(20);

    SELECT lname INTO last\_name FROM user;

    SELECT now() INTO curr\_date;

    INSERT INTO logging VALUES

    (null, 'Delete', 'travel\_to', last\_name, curr\_date);

END$

DELIMITER ;

DROP TRIGGER IF EXISTS logITravel;

DELIMITER $

CREATE TRIGGER logITravel AFTER INSERT ON travel\_to

FOR EACH ROW

BEGIN

    DECLARE curr\_date DATETIME;

    DECLARE last\_name CHAR(20);

    SELECT lname INTO last\_name FROM user;

    SELECT now() INTO curr\_date;

    INSERT INTO logging VALUES

    (null, 'Insert', 'travel\_to', last\_name, curr\_date);

END$

DELIMITER ;

DROP TRIGGER IF EXISTS logUDest;

DELIMITER $

CREATE TRIGGER logUDest AFTER UPDATE ON destination

FOR EACH ROW

BEGIN

    DECLARE curr\_date DATETIME;

    DECLARE last\_name CHAR(20);

    SELECT lname INTO last\_name FROM user;

    SELECT now() INTO curr\_date;

    INSERT INTO logging VALUES

    (null, 'Update', 'destination', last\_name, curr\_date);

END$

DELIMITER ;

DROP TRIGGER IF EXISTS logDDest;

DELIMITER $

CREATE TRIGGER logDDest AFTER DELETE ON destination

FOR EACH ROW

BEGIN

    DECLARE curr\_date DATETIME;

    DECLARE last\_name CHAR(20);

    SELECT lname INTO last\_name FROM user;

    SELECT now() INTO curr\_date;

    INSERT INTO logging VALUES

    (null, 'Delete', 'destination', last\_name, curr\_date);

END$

DELIMITER ;

DROP TRIGGER IF EXISTS logIDest;

DELIMITER $

CREATE TRIGGER logIDest AFTER INSERT ON destination

FOR EACH ROW

BEGIN

    DECLARE curr\_date DATETIME;

    DECLARE last\_name CHAR(20);

    SELECT lname INTO last\_name FROM user;

    SELECT now() INTO curr\_date;

    INSERT INTO logging VALUES

    (null, 'Insert', 'destination', last\_name, curr\_date);

END$

DELIMITER ;

**Επεξήγηση Κώδικα:** Για να υλοποιήσουμε το ζητούμενο του ερωτήματος, δημιουργήσαμε 15 triggers. Δηλαδή 3 triggers για κάθε έναν από τους πίνακες που αναφέρονται στο ερώτημα. Πιο συγκεκριμένα, για κάθε πίνακα έχουμε ένα After Insert, ένα After Delete και ένα After Update trigger.

Στόχος των trigger είναι μετά από κάθε κίνηση σε οποιονδήποτε από τους ζητούμενους πίνακες, να ενημερώνεται ο πίνακας logging για την κίνηση αυτή.

Σε κάθε trigger από τους παραπάνω, παίρνουμε με δύο μεταβλητές τις τιμές curr\_date και last\_name, και έπειτα χρησιμοποιώντας τες, κάνουμε INSERT στον πίνακα logging με τις απαραίτητες πληροφορίες. Το last\_name που κρατάμε είναι το επώνυμο του it manager που έκανε κάθε φορά την αλλαγή και το παίρνουμε από έναν πίνακα user που έχουμε δημιουργήσει και κρατάμε κάθε φορά το επώνυμο του user που είναι συνδεδεμένος κάθε φορά στην εφαρμογή.

**Παράδειγμα Υλοποίησης:**

Παράδειγμα υλοποίησης τους θα γίνει πιο κάτω στην ενότητα για το GUI καθώς η χρήση τους δεν γίνεται να φανεί μέσω της sql.

3.1.4.2

**Κώδικας:**

DROP TRIGGER IF EXISTS no\_change;

DELIMITER $

CREATE TRIGGER no\_change BEFORE UPDATE ON trip

FOR EACH ROW

BEGIN

    DECLARE temp INT(11);

    IF (NEW.tr\_departure<>OLD.tr\_departure OR NEW.tr\_return<>OLD.tr\_return OR NEW.tr\_cost<>OLD.tr\_cost)

    THEN

        SELECT res\_tr\_id INTO temp FROM reservation

        WHERE res\_tr\_id = OLD.tr\_id LIMIT 0,1;

        IF (temp IS NOT NULL)

        THEN

            SIGNAL SQLSTATE '45000'

            SET MESSAGE\_TEXT = 'You cannot update this trip because there is alredy a registration.';

        END IF;

    END IF;

END$

DELIMITER ;

**Επεξήγηση Κώδικα:**

Το συγκεκριμένο trigger ενεργοποιείται πριν κάθε Update στον πίνακα trip. Δουλειά του είναι να ελέγχει αν έχουν γίνει κρατήσεις σε αυτό το trip, αν έχουν γίνει και το Update που πάνε να κάνουμε είναι πάνω σε ημερομηνία ή στο κόστος, τότε δεν μας αφήνει να το κάνουμε.

Για να υλοποιήσουμε το ζητούμενο, ελέγχουμε ως παράμετρο στο if, αν κάποια από τις μεταβλητές που δεν θέλουμε να αλλάξει, πάει να αλλάξει με το Update. Αν γίνεται κάτι τέτοιο, τότε περνάμε μέσα στο If, και με ένα Select κοιτάμε αν υπάρχουν reservations για το συγκεκριμένο ταξίδι. Αν υπάρχουν τότε ακυρώνουμε την ενημέρωση και βγάζουμε μήνυμα λάθους.

**Παράδειγμα Υλοποίησης:**

1. Πηγαίνοντας να αλλάξουμε την ημερομηνία αναχώρησης σε ένα ταξίδι που έχει ήδη reservations, παίρνουμε το παρακάτω μήνυμα:



1. Πηγαίνοντας να αλλάξουμε την ημερομηνία επιστροφής σε ένα ταξίδι που έχει ήδη reservations, παίρνουμε το παρακάτω μήνυμα:





1. Πηγαίνοντας να αλλάξουμε το κόστος σε ένα ταξίδι που έχει ήδη reservations, παίρνουμε το παρακάτω μήνυμα:



1. Πηγαίνοντας να αλλάξουμε όμως την ημερομηνία αναχώρησης σε ένα ταξίδι που δεν έχει reservations:



Παρατηρούμε πως σε αυτήν την περίπτωση μας αφήνει να αλλάξουμε την ημερομηνία αναχώρησης καθώς σε αυτό το trip δεν έχει γίνει κανένα reservation ακόμα.

3.1.4.3

**Κώδικας:**

DROP TRIGGER IF EXISTS no\_decrease;

DELIMITER $

CREATE TRIGGER no\_decrease BEFORE UPDATE ON worker

FOR EACH ROW

BEGIN

IF (NEW.wrk\_salary<OLD.wrk\_salary)

THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'You cannot decrease the salary of a worker';

END IF;

END$

DELIMITER ;

**Επεξήγηση Κώδικα:** Αυτό το Trigger ενεργοποιείται κάθε φορά πριν γίνει κάποια ενημέρωση στον πίνακα worker. Ως παράμετρο σε ένα If Statement ελέγχουμε αν το update γίνεται πάνω στον μισθό του υπαλλήλου. Αν γίνεται εκεί, τότε μέσα στο If Statement ελέγχουμε αν το update έχει στόχο να μειώσει ή να αυξήσει τον μισθό. Αν προσπαθεί να τον μειώσει, τότε βγάζουμε μήνυμα λάθους και απαγορεύουμε την αλλαγή, αν προσπαθεί να τον αυξήσει τότε προχωράμε με το Update.

**Παράδειγμα Υλοποίησης:**

Ας δούμε πρώτα την περίπτωση που αυξάνουμε τον μισθό ενός worker. Πριν κάνουμε το update ίσχυε το παρακάτω:



Καλούμε την συνάρτηση και έπειτα:



Παρατηρούμε πως σε αυτήν την περίπτωση όντως άλλαξε ο μισθός του worker.

Ας δούμε τώρα τι θα γίνει άμα προσπαθήσουμε να μειώσουμε τον μισθό του ίδιου worker.



Έξτρα Triggers:

* **Κώδικας:**

DROP TRIGGER IF EXISTS capacity;

DELIMITER $

CREATE TRIGGER capacity BEFORE INSERT ON reservation

FOR EACH ROW

BEGIN

DECLARE max\_s TINYINT;

CALL seats\_trip(NEW.res\_tr\_id, @res);

IF(@res=0)

THEN SIGNAL SQLSTATE VALUE '45000'

SET MESSAGE\_TEXT = 'The maximum amount of seats have been Reached.';

END IF;

SELECT tr\_maxseats INTO max\_s FROM trip

WHERE tr\_id=NEW.res\_tr\_id;

IF(NEW.res\_seatnum>max\_s)

THEN SIGNAL SQLSTATE VALUE '45000'

SET MESSAGE\_TEXT = 'The seat you are trying to book exceeds the seat number limit of the bus.';

END IF;

END$

DELIMITER ;

**Επεξήγηση Κώδικα:** Δημιουργήσαμε έναν trigger ο οποίος να αξιοποιεί τα δεδομένα που παίρνουμε από την Stored Procedure ‘seats\_trip’. Πιο συγκεκριμένα, στόχος μας ήταν το trigger να μην επιτρέπει το reservation σε κάποιο trip το οποίο έχει ήδη φτάσει τον μέγιστο αριθμό κρατήσεων, και ταυτόχρονα να μην επιτρέπει σε κάποιον να κάνει κράτηση θέσης με αριθμό μεγαλύτερο από τον μέγιστο. Δηλαδή αν το ταξίδι έχει 50 θέσεις, να μην μπορέσει κάποιος να κάνει κράτηση στην θέση 51, καθώς αυτή δεν υπάρχει. Έτσι δημιουργούμε έναν trigger για BEFORE INSERT στον πίνακα reservation.

Για να πετύχουμε τα παραπάνω θα δημιουργήσουμε δύο If Statements.

Στο πρώτο, παίρνοντας την out τιμή από το Stored Procedure seats\_trip, ελέγχουμε ποια είναι η τιμή του. Αν η τιμή του είναι 0, σημαίνει πως δεν υπάρχουν άλλες διαθέσιμες θέσεις στο ταξίδι και έτσι δεν μπορεί να γίνει άλλο reservation σε αυτό. Έτσι βγάζει μήνυμα λάθους και δεν επιτρέπει την πραγματοποίηση του Insert.

Στο δεύτερο If Statement, ελέγχουμε αν η θέση που αντιστοιχεί στην κράτηση είναι μεγαλύτερη από την μεγαλύτερη θέση του ταξιδιού ή αν είναι η θέση 0. Αν ισχύει κάτι από τα δύο, βγαίνει πάλι μήνυμα λάθους και δεν επιτρέπεται να γίνει το συγκεκριμένο reservation.

**Παράδειγμα Υλοποίησης:**

Για το πρώτο If Statement:





Για να ελέγξουμε αν δουλεύει το δεύτερο If Statement, θα πάμε σε ένα ταξίδι που έχει ακόμα διαθέσιμες θέσεις. Σε αυτό το παράδειγμα θα προσπαθήσουμε να κάνουμε Insert στο ταξίδι με id=1 και να κρατήσουμε την θέση 21, ενώ το ταξίδι αυτό προσφέρει 20 θέσεις.





* **Κώδικας:**

DROP TRIGGER IF EXISTS check\_driver;

DELIMITER $

CREATE TRIGGER check\_driver BEFORE INSERT ON driver

FOR EACH ROW

BEGIN

DECLARE temp1 char(10);

DECLARE temp2 char(10);

DECLARE temp3 char(10);

SELECT adm\_AT INTO temp1 FROM admin

WHERE adm\_AT = NEW.drv\_AT

limit 0,1;

SELECT gui\_AT INTO temp2 FROM guide

WHERE gui\_AT = NEW.drv\_AT

LIMIT 0,1;

SELECT wrk\_it\_AT INTO temp3 FROM it

WHERE wrk\_it\_AT = NEW.drv\_AT

LIMIT 0,1;

IF(temp1 = NEW.drv\_AT)

THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'You cannot add t his worker as a driver because he is an Admin.';

END IF;

IF(temp2 = NEW.drv\_AT)

THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'You cannot add this worker as a driver because he is a Guide.';

END IF;

IF(temp3 = NEW.drv\_AT)

THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'You cannot add this worker as a driver because he is an IT Manager.';

END IF;

END$

DELIMITER ;

DROP TRIGGER IF EXISTS check\_admin;

DELIMITER $

CREATE TRIGGER check\_admin BEFORE INSERT ON admin

FOR EACH ROW

BEGIN

DECLARE temp1 char(10);

DECLARE temp2 char(10);

DECLARE temp3 char(10);

SELECT drv\_AT INTO temp1 FROM driver

WHERE drv\_AT = NEW.adm\_AT

limit 0,1;

SELECT gui\_AT INTO temp2 FROM guide

WHERE gui\_AT = NEW.adm\_AT

LIMIT 0,1;

SELECT wrk\_it\_AT INTO temp3 FROM it

WHERE wrk\_it\_AT = NEW.adm\_AT

LIMIT 0,1;

IF(temp1 = NEW.adm\_AT)

THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'You cannot add this worker as an admin because he is a Driver.';

END IF;

IF(temp2 = NEW.adm\_AT)

THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'You cannot add this worker as an admin because he is a Guide.';

END IF;

IF(temp3 = NEW.adm\_AT)

THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'You cannot add this worker as an admin because he is an IT Manager.';

END IF;

END$

DELIMITER ;

DROP TRIGGER IF EXISTS check\_guide;

DELIMITER $

CREATE TRIGGER check\_guide BEFORE INSERT ON guide

FOR EACH ROW

BEGIN

DECLARE temp1 char(10);

DECLARE temp2 char(10);

DECLARE temp3 char(10);

SELECT drv\_AT INTO temp1 FROM driver

WHERE drv\_AT = NEW.gui\_AT

limit 0,1;

SELECT adm\_AT INTO temp2 FROM admin

WHERE adm\_AT = NEW.gui\_AT

LIMIT 0,1;

SELECT wrk\_it\_AT INTO temp3 FROM it

WHERE wrk\_it\_AT = NEW.gui\_AT

LIMIT 0,1;

IF(temp1 = NEW.gui\_AT)

THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'You cannot add this worker as a Guide because he is a Driver.';

END IF;

IF(temp2 = NEW.gui\_AT)

THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'You cannot add this worker as a Guide because he is an Admin.';

END IF;

IF(temp3 = NEW.gui\_AT)

THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'You cannot add this worker asa Guide because he is an IT Manager.';

END IF;

END$

DELIMITER ;

DROP TRIGGER IF EXISTS check\_it;

DELIMITER $

CREATE TRIGGER check\_it BEFORE INSERT ON it

FOR EACH ROW

BEGIN

DECLARE temp1 char(10);

DECLARE temp2 char(10);

DECLARE temp3 char(10);

SELECT drv\_AT INTO temp1 FROM driver

WHERE drv\_AT = NEW.wrk\_it\_AT

limit 0,1;

SELECT adm\_AT INTO temp2 FROM admin

WHERE adm\_AT = NEW.wrk\_it\_AT

LIMIT 0,1;

SELECT gui\_AT INTO temp3 FROM guide

WHERE gui\_AT = NEW.wrk\_it\_AT

LIMIT 0,1;

IF(temp1 = NEW.wrk\_it\_AT)

THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'You cannot add this worker as an IT Manager because he is a Driver.';

END IF;

IF(temp2 = NEW.wrk\_it\_AT)

THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'You cannot add this worker as an IT Manager because he is an Admmin.';

END IF;

IF(temp3 = NEW.wrk\_it\_AT)

THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'You cannot add this worker as an IT Manager because he is a Guide.';

END IF;

END$

DELIMITER ;

**Επεξήγηση Κώδικα:** Κάνοντας κάποια δοκιμαστικά Inserts για να δούμε την λειτουργία της βάσης μας, παρατηρήσαμε ότι ένας Worker μπορεί να εισαχθεί σε πάνω από μία υποκατηγορία. Δηλαδή για παράδειγμα να είναι και admin και driver ταυτόχρονα. Αυτό είναι κάτι που δεν θέλουμε να συμβαίνει. Για αυτόν τον λόγο αποφασίσαμε να δημιουργήσουμε ένα trigger για κάθε table που ανήκει στην κατηγορία worker. Δηλαδή για τους admin, guide, driver και it. Κάθε φορά πριν από insert σε έναν από αυτούς τους πίνακες θα ελέγχεται αν ο worker που πάμε να προσθέσουμε ανήκει ήδη κάπου αλλού ή όχι. Αν ανήκει κάπου αλλού θα απαγορεύουμε την εισαγωγή του στον πίνακα.

Πιο συγκεκριμένα, αυτό υλοποιείται δημιουργώντας για κάθε trigger, τρεις temp μεταβλητές. Για κάθε μία από αυτές κάνουμε ένα Select Into. Σε αυτό τσεκάρουμε αν υπάρχει το ίδιο ΑΤ με αυτό που πάμε να προσθέσουμε σε κάποιον από τους υπόλοιπους 3 πίνακες, αν δεν υπάρχει τότε η αντίστοιχη μεταβλητή παίρνει την τιμή null. Αν είναι και οι 3 μεταβλητές null, τότε επιτρέπεται η εισαγωγή στον πίνακα, αν έστω και μία από αυτές έχει τιμή, τότε απαγορεύουμε την εισαγωγή στον πίνακα και βγάζουμε κατάλληλο μήνυμα.

**Παράδειγμα Υλοποίησης:**

Και τα 4 triggers στα οποία αναφερθήκαμε παραπάνω ακολουθούν το ίδιο ακριβώς μοτίβο, οπότε φτάνει να τεστάρουμε ένα για να δούμε ότι δουλεύει.

Ας δοκιμάσουμε λοιπόν να προσθέσουμε στον πίνακα driver, έναν worker που ανήκει ήδη στον πίνακα it.

Text

Description automatically generated



Όπως βλέπουμε, βγαίνει μήνυμα λάθους και δεν μας επιτρέπεται η εισαγωγή αυτού του worker ως driver.

Ας σιγουρευτούμε τώρα ότι ο trigger δεν μας δημιουργεί πρόβλημα όταν πάμε να προσθέσουμε έναν driver χωρίς να υπάρχει ήδη αυτός ο worker σε κάποιους από τους πίνακες admin,it και guide.

Table

Description automatically generatedTable

Description automatically generatedText

Description automatically generated with medium confidence

Όπως βλέπουμε ο driver προστέθηκε με επιτυχία.

**Κεφάλαιο 3**