Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Εξαμηνιαία Εργασία
Βάσεις Δεδομένων

Αριθμός Ομάδας: 107
Μέλη: Γιώργος Σωτηρόπουλος (el19848)
Παναγιώτης Στεφανής (el19096)
Μιχάλης Τσιλιμιγκουνάκης (el19001)
Εξάμηνο: 6ο

Περιγραφή

Στην παρακάτω εργασία αναπτύξαμε μία βάση δεδομένων που προσομοιάζει τον οργανισμό ΕΛΙΔΕΚ με όλα τα χαρακτηριστικά και τις σχέσεις του. Επίσης αναπτύξαμε έναν web-server με UI για την εύκολη διαχείριση από τον χρήστη. Για τον web-server χρησιμοποιήσαμε Python και το framework Flask, για το development του site χρησιμοποιήσαμε HTML ενώ για την δημιουργία της βάσης χρησιμοποιήσαμε MySQL.

Λειτουργίες που υποστηρίζονται

Μέσω του User Interface υποστηρίζονται οι παρακάτω λειτουργίες:

- Υποστηρίζεται η λειτουργία **Read** δηλαδή η ανάγνωση των tables της βάσης μέσω ενός drop down menu.

- Υποστηρίζεται η λειτουργία **Delete** κατά την οποία ο χρήστης διαγράφει κάποιο row από κάποιο table της επιλογής του, αφού βεβαιωθούμε (μέσω triggers) ότι η διαγραφή του row δεν προκαλεί κάποια παραβίαση στους περιορισμούς της εκφώνησης.

- Υποστηρίζεται η λειτουργία **Insert** κατά την οποία ο χρήστης μπορεί να προσθέσει κάποιο στοιχείο στην βάση, αφού βεβαιωθούμε (μέσω triggers) ότι η εισαγωγή του row δεν προκαλεί κάποια παραβίαση στους περιορισμούς της εκφώνησης.

- Υποστηρίζεται η λειτουργία **Update** κατά την οποία ο χρήστης μπορεί να ενημερώσει κάποιο στοιχείο της βάσης, αφού βεβαιωθούμε (μέσω triggers) ότι η ενημέρωση του row δεν προκαλεί κάποια παραβίαση στους περιορισμούς της εκφώνησης.

- Παράλληλα υποστηρίζονται οι λειτουργίες για τα 8 **Queries** της εκφώνησης.

Οδηγός εγκατάστασης της εφαρμογής

Παρακάτω παραθέτουμε τον οδηγό εγκατάστασης της εφαρμογής μας για το λειτουργικό σύστημα Linux (Debian-based). Το link για το github repo: https://github.com/GeorgeSot71/NTUA_Databases_ELIDEK όπου αρχικά πρέπει να κατεβάσουμε και να κάνουμε extract.

- Αρχικά πρέπει να έχουμε εγκατεστημένο τον mysql-server (παρακάτω φαίνεται ένας σύντομος οδηγός εγκατάστασης). Εγκαθιστούμε τον mysql server με την παρακάτω εντολή:
  sudo apt-get install mysql-server

- Για να ρυθμίσω το password για τον mysql root user τρέχω την παρακάτω εντολή:
  sudo mysql_secure_installation

- *Για το κομμάτι του web server deamon χρειαζόμαστε εγκατεστημένη την python3 και τον package manager που θα μας επιτρέψει να εγκαταστήσουμε το framework flask. Τα παραπάνω γίνονται με τις παρακάτω εντολές:*
  sudo apt install python3
  sudo apt install pip
  pip install flask
  pip install mysql-connector-python

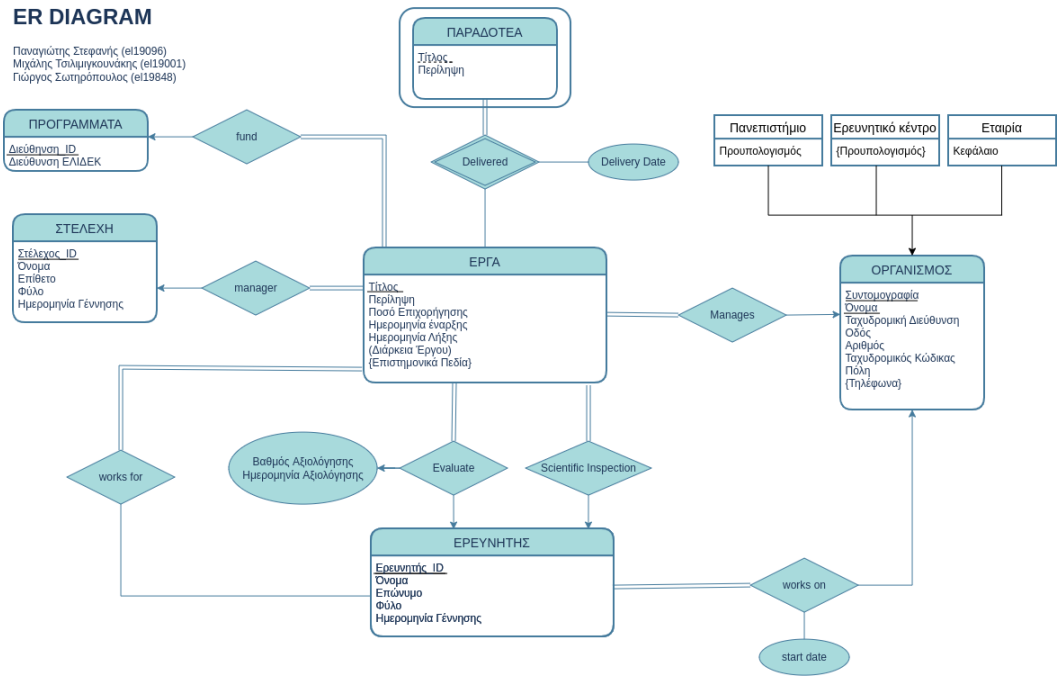
- Στην συνέχεια τρέχουμε τα αρχεία elidek_tables_script.sql και all_insert.sql (είτε από terminal κάνοντας πρώτα login με την εντολή ***mysql -u root -p*** είτε από DBMS client) ώστε να δημιουργηθεί η βάση και να εισάγουμε τα απαραίτητα δοκιμαστικά δεδομένα.

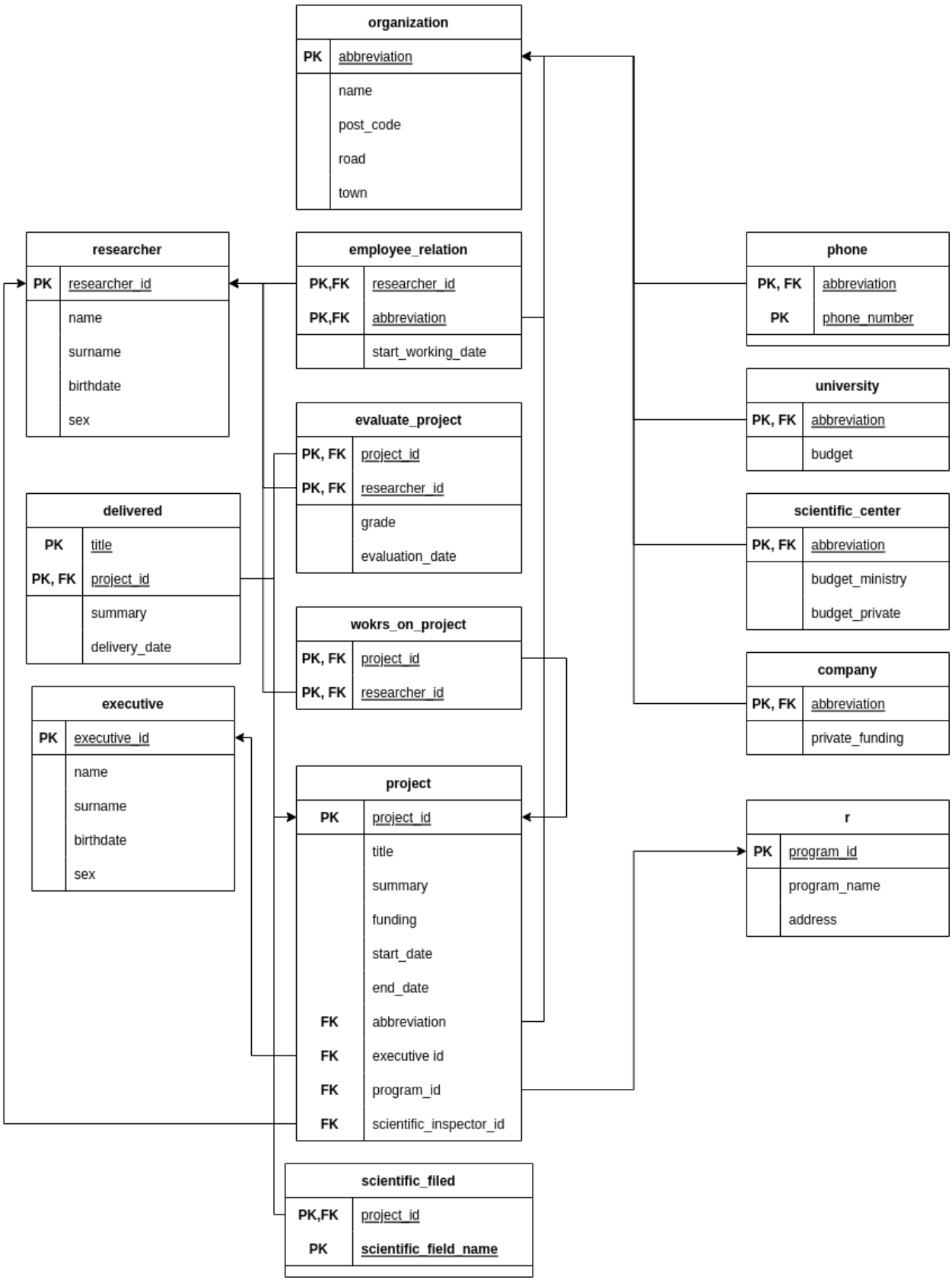
- Στο αρχείο app.py αλλάζουμε το password στην αρχή (για την πρόσβαση στην βάση) ώστε να είναι το ίδιο με αυτό που έχουμε για την MySQL βάση.

- Για να ανοίξει ο server και να έχουμε πρόσβαση στο ui ανοίγουμε ένα terminal στο directory του αρχείου app.py και τρέχουμε την παρακάτω εντολή.
  python3 -m flask run

- Στην συνέχεια ανοίγουμε έναν browser και πληκτρολογούμε για την πρόσβαση στο UI:
  http://localhost:5000/

Διάγραμμα ER και σχεσιακό διάγραμμα βάσης RM




organization
PK abbreviation
name
post_code
road
town
researcher
PK researcher_id
name
surname
birthdate
sex
employee_relation
PK,FK researcher_id
PK,FK abbreviation
start_working_date
phone
PK, FK abbreviation
PK phone_number
university
PK, FK abbreviation
budget
evaluate_project
PK, FK project_id
PK, FK researcher_id
grade
evaluation_date
delivered
PK title
PK, FK project_id
summary
delivery_date
scientific_center
PK, FK abbreviation
budget_ministry
budget_private
wokrs_on_project
PK, FK project_id
PK, FK researcher_id
company
PK, FK abbreviation
private_funding
executive
PK executive_id
name
surname
birthdate
sex
project
PK project_id
title
summary
funding
start_date
end_date
FK abbreviation
FK executive id
FK program_id
FK scientific_inspector_id
r
PK program_id
program_name
address
scientific_filed
PK,FK project_id
PK scientific_field_name

Το σχεσιακό διάγραμμα της βάσης RM υλοποιήθηκε με βάση των ER diagram της βάσης. Για την πραγματοποίηση του RM λάβαμε υπόψιν του κανόνες σχεδίασης ενός σχεσιακού διαγράμματος. Πιο συγκεκριμένα:

- Η αναπαράσταση ενός multivalued attribute γίνεται μέσω ενός ξεχωριστού table, όπου θα έχει ως primary key το primary key του αρχικού table συν το multivalued attribute. Πχ για να υλοποιήσουμε το attribute “phone” του organization δημιουργήσαμε ένα καινούριο table phone , όπου έχει primary key το abbreviation και το κάθε phone number.

- Οι σχέσεις , οι οποίες έχουν total participation και το πολύ μία συμμετοχή (βέλος από τη μια πλευρά της σχέσης στο ER diagram) δεν υλοποιούνται με ξεχωριστό table αλλά η μία πλευρά της σχέσης (αυτή στην οποία “δείχνει” το βέλος) προστιθεται ως ένα νέο attribute στο table ”της πλευράς” με την total participation. Πχ το table του project διαθέτει ένα νέο attribute “executive_id”.

- Στο σχεσιακό διάγραμμα υποδηλώνονται με βέλη το reference integrity, δηλαδή τα attributes που ειναι foreign keys σε άλλα tables δείχνουν με ένα βέλος στα αντίστοιχα table .

- Οι αδύναμες οντότητες του ER diagram υλοποιούνται στο RM ως ένα νέο table , το οποίο περιλαμβάνει μια νέα στήλη για το primary key του “identifying strong entity set”. Βλέπε υλοποίηση delivered table.

- Τα derived χαρακτηριστικά στο ER diagram δεν περιλαμβάνονται στο relational model. Γι αυτό και απουσιάζει το age του researcher και το duration του project.

- Τέλος τα σύνθετα χαρακτηριστικά, όπως η διεύθυνση του οργανισμού υλοποίουνται απλά με τόσα attributes στο συγκεκριμένο table όσα αυτα που περιέχουν.

Επίσης κατά την υλοποίηση της βάσης δημιουργήσαμε 5 διαφορετικά indexes με στόχο την γρηγορότερη αναζήτηση μέσα στα δεδομένα της βάσης σε πιθανά queries μέσω των B+ Trees. Τα indexes αυτά αφορούν τα πεδία:

- scientific field name
- project title
- researcher full name
- program name
- project start date

Τα οποία και παρατηρήσαμε πως έχουν την μεγαλύτερη συχνότητα στα query της δικής μας υλοποίησης.

SQL scripts

Παρακάτω φαίνονται τα create tables:

```sql
CREATE TABLE program(
    program_id INT NOT NULL AUTO_INCREMENT,
    program_name VARCHAR(300) NOT NULL,
    address VARCHAR(30) NOT NULL,
    PRIMARY KEY (program_id)
)ENGINE=INNODB;

CREATE TABLE executive (
    executive_id INT NOT NULL AUTO_INCREMENT,
    name VARCHAR(30) NOT NULL,
    surname VARCHAR(30) NOT NULL,
    birthday DATE NOT NULL,
    sex VARCHAR(10) NOT NULL,
    PRIMARY KEY (executive_id)
)ENGINE=INNODB;

CREATE TABLE researcher (
    researcher_id INT NOT NULL AUTO_INCREMENT,
    name VARCHAR(30) NOT NULL,
    surname VARCHAR(30) NOT NULL,
    birthday DATE NOT NULL,
    sex VARCHAR(30) NOT NULL,
    PRIMARY KEY (researcher_id)
)ENGINE=INNODB;

CREATE TABLE organization (
    abbreviation VARCHAR(30) NOT NULL,
    name VARCHAR(30) NOT NULL,
    post_code INT NOT NULL,
    road VARCHAR(30) NOT NULL,
    town VARCHAR(30) NOT NULL,
    PRIMARY KEY (abbreviation)
)ENGINE=INNODB;

CREATE TABLE phone (
    abbreviation VARCHAR(30) NOT NULL,
    phone_number BIGINT NOT NULL,
    PRIMARY KEY (abbreviation, phone_number),
    FOREIGN KEY (abbreviation)
        REFERENCES organization(abbreviation) ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=INNODB;
```

```sql
CREATE TABLE university (
    abbreviation VARCHAR(50) NOT NULL,
    budget DECIMAL(13,2) NOT NULL CHECK (budget>0),
    PRIMARY KEY (abbreviation),
    FOREIGN KEY (abbreviation)
        REFERENCES organization(abbreviation) ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=INNODB;

CREATE TABLE scientific_center (
    abbreviation VARCHAR(50) NOT NULL,
    budget_ministry DECIMAL(13,2) NOT NULL CHECK (budget_ministry>0),
    budget_private DECIMAL(13,2) NOT NULL CHECK (budget_private>0),
    PRIMARY KEY (abbreviation),
    FOREIGN KEY (abbreviation)
        REFERENCES organization(abbreviation) ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=INNODB;

CREATE TABLE company (
    abbreviation VARCHAR(30) NOT NULL,
    private_funding DECIMAL(13,2) NOT NULL CHECK (private_funding>0),
    PRIMARY KEY (abbreviation),
    FOREIGN KEY (abbreviation)
        REFERENCES organization(abbreviation) ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=INNODB;

CREATE TABLE employee_relation (
    researcher_id INT NOT NULL,
    abbreviation VARCHAR(30) NOT NULL,
    start_working_date DATE NOT NULL,
    PRIMARY KEY (researcher_id, abbreviation),
    FOREIGN KEY (researcher_id)
        REFERENCES researcher(researcher_id) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (abbreviation)
        REFERENCES organization(abbreviation) ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=INNODB;
```

```sql
CREATE TABLE project (
    project_id INT NOT NULL AUTO_INCREMENT,
    title VARCHAR(100) NOT NULL,
    summary TEXT NOT NULL,
    funding DECIMAL(10,2) NOT NULL CHECK (funding >= 100000 AND funding <= 1000000) ,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    abbreviation VARCHAR(30) NOT NULL,
    executive_id INT NOT NULL,
    program_id INT NOT NULL,
    scientific_inspector_id INT NOT NULL,
    PRIMARY KEY (project_id),
    FOREIGN KEY (abbreviation)
        REFERENCES organization(abbreviation) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (executive_id)
        REFERENCES executive(executive_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (program_id)
        REFERENCES program(program_id) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (scientific_inspector_id)
        REFERENCES researcher(researcher_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT project_duration_check CHECK ((DATEDIFF(end_date,start_date)>=365) AND DATEDIFF(end_date,start_date)<=1460)
)ENGINE=INNODB;

CREATE TABLE delivered (
    title VARCHAR(100) NOT NULL,
    project_id INT NOT NULL,
    summary TEXT NOT NULL,
    delivery_date DATE NOT NULL,
    PRIMARY KEY (title,project_id),
    FOREIGN KEY (project_id)
        REFERENCES project(project_id) ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=INNODB;
```

```sql
CREATE TABLE works_on_project(
    project_id INT NOT NULL,
    researcher_id INT NOT NULL,
    PRIMARY KEY (project_id, researcher_id),
    FOREIGN KEY (project_id)
    REFERENCES project(project_id) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (researcher_id)
    REFERENCES researcher(researcher_id) ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=INNODB;

CREATE TABLE evaluate_project(
    project_id INT NOT NULL,
    researcher_id INT NOT NULL,
    grade DECIMAL(4,2) NOT NULL CHECK (grade>=0 AND grade<=10),
    evaluation_date DATE NOT NULL,
    PRIMARY KEY (project_id,researcher_id),
    FOREIGN KEY (project_id)
        REFERENCES project(project_id) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (researcher_id)
        REFERENCES researcher(researcher_id) ON DELETE RESTRICT ON UPDATE CASCADE
)ENGINE=INNODB;

CREATE TABLE scientific_field (
    project_id INT NOT NULL,
    scientific_field_name VARCHAR(50) NOT NULL,
    PRIMARY KEY (project_id,scientific_field_name),
    FOREIGN KEY (project_id)
        REFERENCES project(project_id) ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=INNODB;
```

Παρακάτω φαίνονται οι υλοποιήσεις των triggers:

```
DELIMITER $$
/* 1 */
CREATE TRIGGER same_evaluator_researcher_insert BEFORE INSERT ON evaluate_project
    FOR EACH ROW
    BEGIN
    IF NEW.researcher_id IN (SELECT researcher_id
                        FROM works_on_project
                                            WHERE works_on_project.project_id = NEW.project_id)
                                            THEN SIGNAL SQLSTATE '45000'  SET MESSAGE_TEXT = 'You cannot add an evaluator that already works on project' ;
    END IF;
  END;$$

/* 2 */
CREATE TRIGGER same_researcher_evaluator_insert BEFORE INSERT ON works_on_project
    FOR EACH ROW
    BEGIN
    IF NEW.researcher_id IN (SELECT researcher_id
                        FROM evaluate_project
                                            WHERE evaluate_project.project_id = NEW.project_id)
                                            THEN SIGNAL SQLSTATE '45000'  SET MESSAGE_TEXT = 'You cannot add a researcher that is already evaluator in the same project' ;
        END IF;
  END;$$

    /* 3 */
    CREATE TRIGGER researcher_in_project_and_not_in_organisation_insert BEFORE INSERT ON works_on_project
    FOR EACH ROW
        BEGIN
      IF NEW.researcher_id  NOT IN (SELECT researcher_id
                                FROM employee_relation INNER JOIN project p ON p.abbreviation = employee_relation.abbreviation
                                                        WHERE p.project_id = NEW.project_id )
                                THEN SIGNAL SQLSTATE '45000'  SET MESSAGE_TEXT = 'You cannot assign a researcher as working on a project if he/she is not working for the organization that manages the project' ;
      END IF;
    END;$$
```

```
/* 4 */
  CREATE TRIGGER scientific_inspector_in_project_and_not_in_organisation_insert BEFORE INSERT ON project
  FOR EACH ROW
      BEGIN
    IF NEW.scientific_inspector_id  NOT IN (SELECT researcher_id
                              FROM employee_relation
                                                    WHERE abbreviation = NEW.abbreviation)
                              THEN SIGNAL SQLSTATE '45000'  SET MESSAGE_TEXT = 'You cannot assign a scientific inspector on a project if he/she is not working for the organization that manages the project' ;
    END IF;
END;$$

  /* 5 */
CREATE TRIGGER researcher_in_two_different_organisation_insert BEFORE INSERT ON employee_relation
  FOR EACH ROW
      BEGIN
    IF NEW.researcher_id IN (SELECT researcher_id
                FROM employee_relation )
                THEN SIGNAL SQLSTATE '45000'  SET MESSAGE_TEXT = 'You cannot assign a researcher to work on two different organizations' ;
    END IF;
END;$$

/* 6 */
  CREATE TRIGGER same_evaluator_researcher_update BEFORE UPDATE ON evaluate_project
  FOR EACH ROW
  BEGIN
  IF NEW.researcher_id IN (SELECT researcher_id
                FROM works_on_project
                              WHERE works_on_project.project_id = NEW.project_id)
                              THEN SIGNAL SQLSTATE '45000'  SET MESSAGE_TEXT = 'You cannot make a researcher evaluate a project if he/she already works on it' ;
  END IF;
END;$$
```

```
/* 7 */
CREATE TRIGGER same_researcher_evaluator_update BEFORE UPDATE ON works_on_project
    FOR EACH ROW
    BEGIN
    IF NEW.researcher_id IN (SELECT researcher_id
                  FROM evaluate_project
                                WHERE evaluate_project.project_id = NEW.project_id)
                                THEN SIGNAL SQLSTATE '45000'  SET MESSAGE_TEXT = 'You cannot make a researcher work on a project in which he/she is already evaluator';
    END IF;
  END;$$

 /* 8*/
  CREATE TRIGGER researcher_in_project_and_not_in_organisation_update BEFORE UPDATE ON works_on_project
  FOR EACH ROW
      BEGIN
    IF NEW.researcher_id  NOT IN (SELECT researcher_id
                  FROM employee_relation INNER JOIN project p ON p.abbreviation = employee_relation.abbreviation
                                  WHERE p.project_id = NEW.project_id )
                  THEN SIGNAL SQLSTATE '45000'  SET MESSAGE_TEXT = 'You cannot make a researcher work on a project if he/she is not working for the organization that manages the project' ;
    END IF;
  END;$$

/* 9 */
  CREATE TRIGGER scientific_inspector_in_project_and_not_in_organisation_update BEFORE UPDATE ON project
  FOR EACH ROW
      BEGIN
    IF NEW.scientific_inspector_id  NOT IN (SELECT er.researcher_id
                          FROM employee_relation er
                                                WHERE NEW.abbreviation = er.abbreviation)
                          THEN SIGNAL SQLSTATE '45000'  SET MESSAGE_TEXT = 'You cannot make a scientific inspector on a project if he/she is not working for the organization that manages the project';
    END IF;
END;$$
```

```
CREATE TRIGGER researcher_in_two_different_organisation_update BEFORE UPDATE ON employee_relation
        FOR EACH ROW
        BEGIN
            IF NEW.researcher_id IN (SELECT researcher_id FROM employee_relation) AND NEW.abbreviation != OLD.abbreviation
                                THEN SIGNAL SQLSTATE '45000'  SET MESSAGE_TEXT = 'You cannot make researcher to work on two different organizations' ;
            END IF;
    END;$$
DELIMITER ;
```

Παρακάτω φαίνονται τα views:

```
CREATE VIEW projects_per_researcher
AS
SELECT r.name, r.surname, p.title
FROM researcher r, project p
WHERE r.researcher_id IN (SELECT researcher_id FROM works_on_project WHERE project_id = p.project_id)
ORDER BY r.researcher_id;

CREATE VIEW organization_info
AS
SELECT org.abbreviation ,org.name ,org.post_code, org.road, org.town, ph.phone_number
FROM organization org
LEFT JOIN
phone ph
ON ph.abbreviation = org.abbreviation;
```

Παρακάτω φαίνονται τα index:

```
CREATE INDEX  idx_scientific_field_name
ON scientific_field (scientific_field_name);

CREATE INDEX  idx_project_title
ON project (title);

CREATE INDEX  idx_researcher_fullname
ON researcher (name,surname);

CREATE INDEX  idx_program_name
ON program (program_name);

CREATE INDEX  idx_project_start_date
ON project (start_date);
```