



ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Εξαμηνιαία Εργασία
στο μάθημα «**Βάσεις Δεδομένων**»

Ο διευθυντής του Ε.Λ.Ι.Δ.Ε.Κ. σας κάλεσε για να σχεδιάσετε μία βάση δεδομένων...

της ομάδας 129

Ηλιόπουλος Γεώργιος, Α.Μ.: 03118815

Ηλιόπουλος Ανδρέας, Α.Μ.: 03120815

GitHub repo: <https://github.com/jugger96/DB-NTUA-Database-for-ELIDEK>

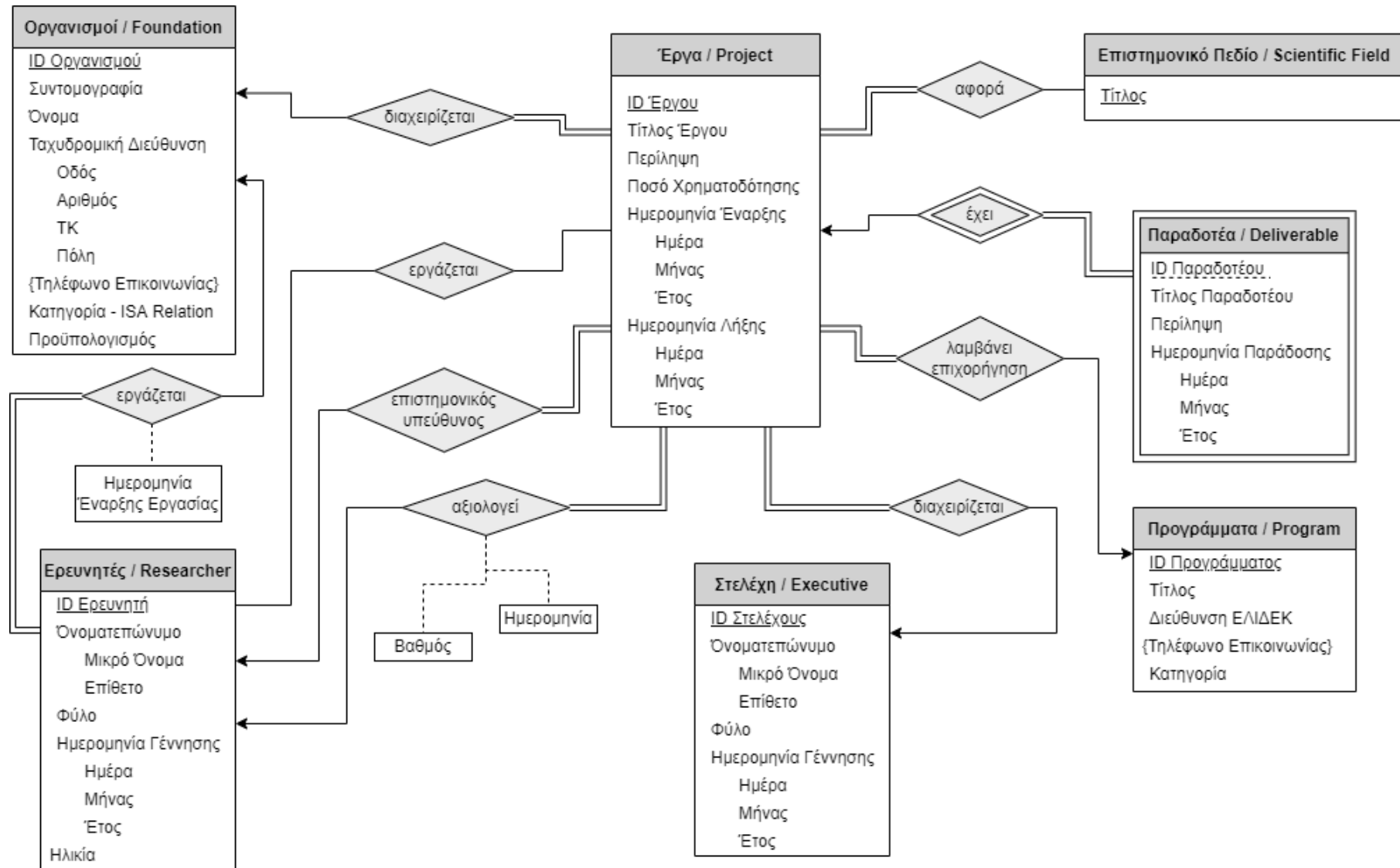
Αθήνα, Ιούνιος 2022

Περιεχόμενα

1 ER Diagram	2
2.1 Σχεσιακό διάγραμμα.....	3
2.2 DDL και DML κώδικας και επεξήγηση.....	4
2.2.1 Tables	4
2.2.2. Triggers	11
2.2.3. Ευρετήρια – Indexes.....	15
2.2.4 Views	15
2.2.5 Δημιουργία fake data	25
2.3 Βήματα Εγκατάστασης	26
3 Web App Presentation.....	27
3.1 Γενική Περιγραφή Περιβάλλοντος.....	27
3.2 Extra Features	30

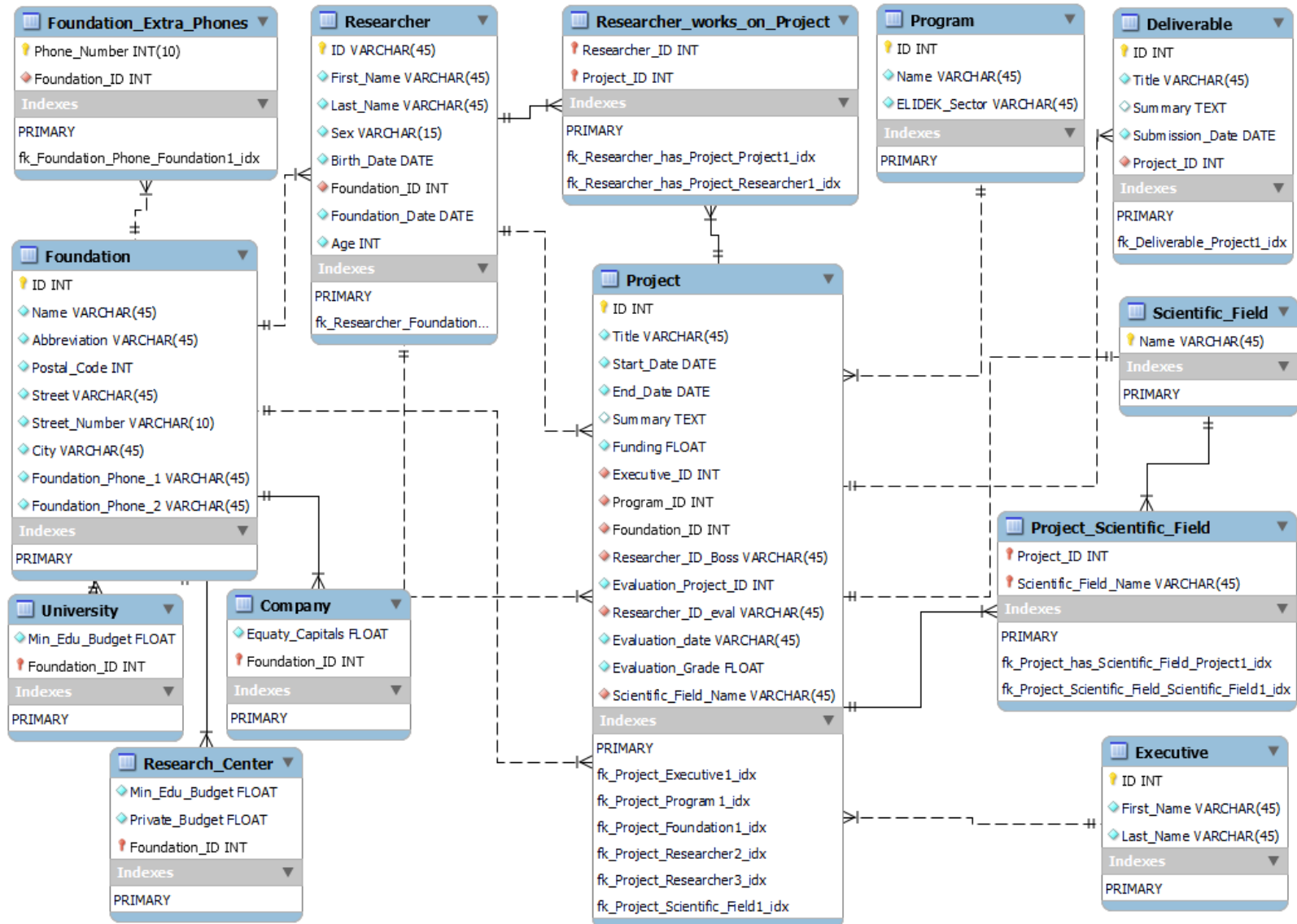
1 ER Diagram

Παρακάτω φαίνεται το ER διάγραμμα όπου βασίστηκε η υλοποίηση της βάσης. Δεν υπάρχουν ιδιαίτερες διαφορές με το ενδεικτικό.



2.1 Σχεσιακό διάγραμμα

Ακολουθεί το σχεσιακό διάγραμμα που δημιουργήσαμε στο MySQL Workbench στο οποίο απεικονίζονται όλοι οι πίνακες καθώς και τα indexes. Αναλυτική εξήγηση για τον κάθε πίνακα υπάρχει στην ενότητα 2.2.1.



2.2 DDL και DML κώδικας και επεξήγηση

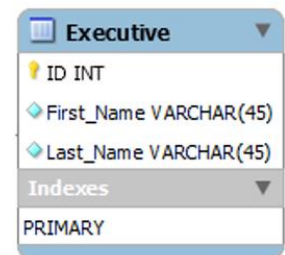
2.2.1 Tables

Στην συνέχεια ακολουθεί ο κώδικας για τη δημιουργία των tables καθώς και των views που θεωρήσαμε χρήσιμο να δημιουργήσουμε.

Ενδεικτικά θα αναλύσουμε τον sql κώδικα για τη δημιουργία ορισμένων πινάκων και έπειτα θα αναφέρουμε ό,τι καινούργιο.

1 – Στελέχη / Executives

```
-- -----  
-- Table `Executive`  
-- -----  
  
DROP TABLE IF EXISTS `Executive`;  
CREATE TABLE IF NOT EXISTS `Executive` (  
  `ID` VARCHAR(45) NOT NULL,  
  `First_Name` VARCHAR(45) NOT NULL,  
  `Last_Name` VARCHAR(45) NOT NULL,  
  PRIMARY KEY (`ID`)  
) ENGINE=INNODB;
```

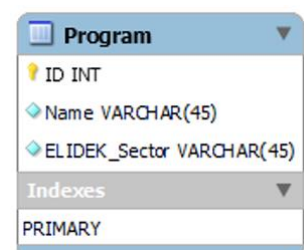


Executive	
ID	INT
First_Name	VARCHAR(45)
Last_Name	VARCHAR(45)
Indexes	
PRIMARY	

Ένα στέλεχος (executive) έχει υποχρεωτικά όνομα οπότε στον πίνακα των στελεχών βάλαμε δύο attributes `First_Name` και `Last_Name` τα οποία θέσαμε ως VARCHAR(45) αφού πρόκειται για αλφαριθμητικό δεδομένο και NOT NULL ώστε να μην επιτρέπεται να μείνει κενό αυτό το πεδίο. Επίσης ως primary key ορίσαμε τον αριθμό κοινωνικής ασφάλισης του στελέχους (SSN - `ID`) το οποίο από τον ορισμό είναι μοναδικό για κάθε άνθρωπο.

2 – Προγράμματα / Programs

```
-- -----  
-- Table `Program`  
-- -----  
  
DROP TABLE IF EXISTS `Program`;  
CREATE TABLE IF NOT EXISTS `Program` (  
  `ID` INT NOT NULL AUTO_INCREMENT,  
  `Name` VARCHAR(45) NOT NULL,  
  `ELIDEK_Sector` VARCHAR(45) NOT NULL,  
  PRIMARY KEY (`ID`)  
) ENGINE=INNODB;
```

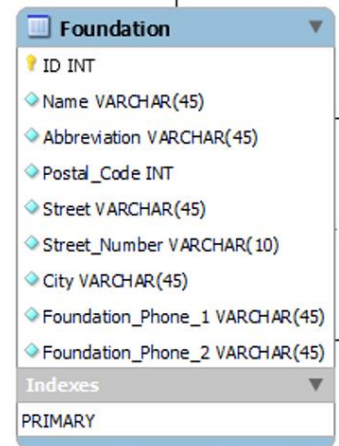


Program	
ID	INT
Name	VARCHAR(45)
ELIDEK_Sector	VARCHAR(45)
Indexes	
PRIMARY	

Ένα πρόγραμμα (program) έχει όνομα και ανήκει σε έναν τομέα του ΕΛΙΔΕΚ οπότε στον αντίστοιχο πίνακα βάλαμε χαρακτηριστικά `Name` και `ELIDEK_Sector` ως VARCHAR(45) αφού πρόκειται για αλφαριθμητικό δεδομένο και NOT NULL ώστε να μην επιτρέπεται να μείνει κενό αυτό το πεδίο. Ως αναγνωριστικό του κάθε προγράμματος ορίσαμε ένα AUTO_INCREMENT `ID` και το θέσαμε ως το κλειδί αυτού του πίνακα.

3 - Οργανισμοί / Foundations

```
-- Table `Foundation`  
--  
DROP TABLE IF EXISTS `Foundation`;  
CREATE TABLE IF NOT EXISTS `Foundation` (  
  `ID` INT NOT NULL AUTO_INCREMENT,  
  `Name` VARCHAR(45) NOT NULL,  
  `Abbreviation` VARCHAR(45) NOT NULL,  
  `Postal_Code` INT NOT NULL,  
  `Street` VARCHAR(45) NOT NULL,  
  `Street_Number` VARCHAR(10) NOT NULL,  
  `City` VARCHAR(45) NOT NULL,  
  `Foundation_Phone_1` VARCHAR(45) NOT NULL,  
  `Foundation_Phone_2` VARCHAR(45) NOT NULL,  
  PRIMARY KEY (`ID`)  
) ENGINE=INNODB ROW_FORMAT=DEFAULT;
```

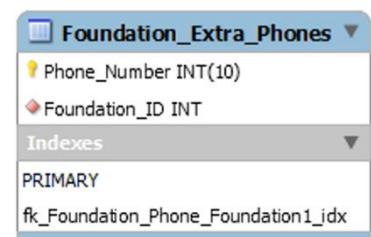


Foundation	
ID	INT
Name	VARCHAR(45)
Abbreviation	VARCHAR(45)
Postal_Code	INT
Street	VARCHAR(45)
Street_Number	VARCHAR(10)
City	VARCHAR(45)
Foundation_Phone_1	VARCHAR(45)
Foundation_Phone_2	VARCHAR(45)
Indexes	
PRIMARY	

Στον πίνακα για τους οργανισμούς έχουμε βάλει δύο τηλέφωνα τα οποία πρέπει να εισάγονται όταν κανείς δημιουργεί έναν νέο οργανισμό αφού κατά την εκφώνηση πρέπει ένας οργανισμός να έχει πάνω από 1 τηλέφωνο (δηλαδή τουλάχιστον 2). Τα επιπλέον τηλέφωνα που μπορεί να έχει ένας οργανισμός εισάγονται σε άλλον πίνακα που περιγράφεται παρακάτω. Αν ένας οργανισμός είναι εταιρεία, ερευνητικό κέντρο ή πανεπιστήμιο αναλύεται στους πίνακες 12,13 και 14 πως επιτυγχάνεται.

4 - Επιπλέον τηλέφωνα οργανισμών / Foundation Extra Phones

```
-- Table `Foundation_Phone`  
--  
DROP TABLE IF EXISTS `Foundation_Extra_Phones`;  
CREATE TABLE IF NOT EXISTS `Foundation_Extra_Phones` (  
  `Phone_Number` VARCHAR(45) NOT NULL,  
  `Foundation_ID` INT NOT NULL,  
  PRIMARY KEY (`Phone_Number`),  
  INDEX `fk_Foundation_Extra_Phones_Foundation1_idx`  
  (`Foundation_ID` ASC),  
  CONSTRAINT `fk_Foundation_Extra_Phones_Foundation1`  
  FOREIGN KEY (`Foundation_ID`)  
    REFERENCES `Foundation` (`ID`)  
    ON DELETE CASCADE ON UPDATE CASCADE  
) ENGINE=INNODB;
```




Foundation_Extra_Phones	
Phone_Number	INT(10)
Foundation_ID	INT
Indexes	
PRIMARY	
fk_Foundation_Phone_Foundation1_idx	

Στον συγκεκριμένο πίνακα αποθηκεύουμε τα επιπλέον τηλέφωνα που μπορεί να έχει ένας οργανισμός. Περιέχει δύο στοιχεία το τηλέφωνο (`Phone_Number`) και το ID του οργανισμού (`Foundation_ID`) όπου ανήκει το τηλέφωνο. Το `Foundation_ID` είναι foreign key που αναφέρεται στο πίνακα Foundation και έχουμε εισάγει και το αντίστοιχο **FOREIGN KEY CONSTRAINT**. Ως κλειδί αυτού του πίνακα ορίζουμε το τηλέφωνο αφού από τη φύση του είναι μοναδικό. Στον παραπάνω κώδικα παρατηρούμε πως δημιουργείται και index στα foreign keys

που συχνά χρησιμοποιούνται για εύρεση δεδομένων. Τα indexes αναλύονται περισσότερο σε επόμενη ενότητα.

5 – Ερευνητές / Researcher

```
-- Table `Researcher`  
-----  
  
DROP TABLE IF EXISTS `Researcher`;  
CREATE TABLE IF NOT EXISTS `Researcher` (  
  `ID` VARCHAR(45) NOT NULL,  
  `First_Name` VARCHAR(45) NOT NULL,  
  `Last_Name` VARCHAR(45) NOT NULL,  
  `Sex` VARCHAR(15) NOT NULL,  
  `Birth_Date` DATE NOT NULL,  
  `Foundation_ID` INT NOT NULL,  
  `Foundation_Date` DATE NOT NULL,  
  `Age` int AS (round(datediff(curdate(),  
birth_date)/365, 0)),  
  PRIMARY KEY (`ID`),  
  INDEX `fk_Researcher_Foundation1_idx` (`Foundation_ID`  
ASC),  
  CONSTRAINT `fk_Researcher_Foundation1`  
    FOREIGN KEY (`Foundation_ID`)  
    REFERENCES `Foundation` (`ID`)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE)  
ENGINE = InnoDB;
```

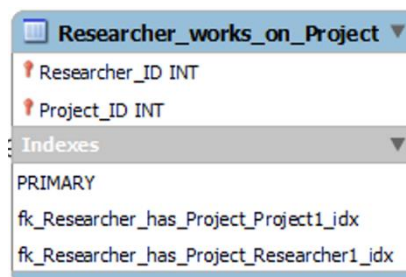


Researcher	
ID	VARCHAR(45)
First_Name	VARCHAR(45)
Last_Name	VARCHAR(45)
Sex	VARCHAR(15)
Birth_Date	DATE
Foundation_ID	INT
Foundation_Date	DATE
Age	INT
Indexes	
PRIMARY	
fk_Researcher_Foundation...	

Εκτός των άλλων εδώ έχουμε ορίσει την ημερομηνία γέννησης ως τύπο δεδομένων `DATE`. Ιδιαίτερο στον παραπάνω κώδικα είναι η εισαγωγή ενός παραγόμενου attribute, αυτού της ηλικίας (``Age` int AS ...`). Για να πάρουμε την ηλικία αφαιρούμε από την σημερινή ημερομηνία (`curdate()`) την ημερομηνία γέννησης χρησιμοποιώντας την συνάρτηση `datediff` που δίνει ως αποτέλεσμα διαφορά ημερών. Έτσι διαιρώντας με 365 και στρογγυλοποιώντας με την `round(x, 0)` παίρνουμε έναν ακέραιο ως ηλικία.

6 – Σχέση Ερευνητών με Έργα / Researcher Works On Project

```
-- Table `Researcher_works_on_Project`  
-----  
  
DROP TABLE IF EXISTS `Researcher_works_on_Project`;  
CREATE TABLE IF NOT EXISTS  
`Researcher_works_on_Project` (  
  `Researcher_ID` VARCHAR(45) NOT NULL,  
  `Project_ID` INT NOT NULL,  
  PRIMARY KEY (`Researcher_ID`, `Project_ID`),  
  INDEX `fk_Researcher_has_Project_Project1_idx`  
(`Project_ID` ASC),  
  INDEX  
`fk_Researcher_has_Project_Researcher1_idx`  
(`Researcher_ID` ASC),
```



Researcher_works_on_Project	
Researcher_ID	INT
Project_ID	INT
Indexes	
PRIMARY	
fk_Researcher_has_Project_Project1_idx	
fk_Researcher_has_Project_Researcher1_idx	

```

    CONSTRAINT
`fk_Researcher_works_on_Project_Researcher1` FOREIGN
KEY (`Researcher_ID`)
    REFERENCES `Researcher` (`ID`)
    ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT
`fk_Researcher_works_on_Project_Project1` FOREIGN
KEY (`Project_ID`)
    REFERENCES `Project` (`ID`)
    ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=INNODB;

```

Ο πίνακας αυτό περιέχει δύο χαρακτηριστικά που είναι και τα δύο **FOREIGN KEYS**, και δείχνει την σχέση μεταξύ των ερευνητών και των έργων. Με αυτόν τον πίνακα στην ουσία υλοποιείται μία many-to-many σχέση.

7 – Επιστημονικά Πεδία / Scientific Fields

```

-- -----
-- Table `Scientific_Field`
-- -----
DROP TABLE IF EXISTS `Scientific_Field`;
CREATE TABLE IF NOT EXISTS `Scientific_Field` (
  `Name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Name`)
) ENGINE=INNODB;

```

Scientific_Field
Name VARCHAR(45)
Indexes
PRIMARY

Εδώ ορίζουμε ως **PRIMARY KEY** το όνομα του επιστημονικού πεδίου (`Name`) γιατί θεωρούμε πως δύο επιστημονικά δεν μπορούν να έχουν το ίδιο όνομα.

8 – Σχέση Έργων με Επιστημονικά Πεδία

```

-- -----
-- Table `Project_Scientific_Field`
-- -----
DROP TABLE IF EXISTS `mydb`.`Project_Scientific_Field`;
CREATE TABLE IF NOT EXISTS
`mydb`.`Project_Scientific_Field` (
  `Project_ID` INT NOT NULL,
  `Scientific_Field_Name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Project_ID`, `Scientific_Field_Name`),
  INDEX `fk_Project_has_Scientific_Field_Project1_idx`
  (`Project_ID` ASC),
  INDEX
`fk_Project_Scientific_Field_Scientific_Field1_idx`
  (`Scientific_Field_Name` ASC),
  CONSTRAINT `fk_Project_has_Scientific_Field_Project1`
FOREIGN KEY (`Project_ID`)
  REFERENCES `mydb`.`Project` (`ID`)
  ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT
`fk_Project_Scientific_Field_Scientific_Field1` FOREIGN
KEY (`Scientific_Field_Name`)
  REFERENCES `mydb`.`Scientific_Field` (`Name`)
)

```

Project_Scientific_Field
Project_ID INT
Scientific_Field_Name VARCHAR(45)
Indexes
PRIMARY
fk_Project_has_Scientific_Field_Project1_idx
fk_Project_Scientific_Field_Scientific_Field1_idx


```

        ON DELETE NO ACTION ON UPDATE CASCADE
    ) ENGINE=INNODB;

```

Ο παραπάνω πίνακας αναπαριστά όπως και ο πίνακας 6 (Σχέση Ερευνητών με Έργα) μία many-to-many σχέση.

9 - Έργα/Επιχορηγήσεις / Project

```

-- Table `Project`
--
DROP TABLE IF EXISTS `mydb`.`Project`;
CREATE TABLE IF NOT EXISTS `mydb`.`Project` (
  `ID` INT NOT NULL AUTO_INCREMENT,
  `Title` VARCHAR(79) NOT NULL,
  `Start_Date` DATE NOT NULL,
  `End_Date` DATE NOT NULL,
  `Summary` TEXT NULL,
  `Funding` FLOAT NOT NULL,
  `Executive_ID` VARCHAR(45) NOT NULL,
  `Program_ID` INT NOT NULL,
  `Foundation_ID` INT NOT NULL,
  `Researcher_ID_Boss` VARCHAR(45) NOT NULL,
  `Researcher_ID_eval` VARCHAR(45) NOT NULL,
  `Evaluation_date` DATE NOT NULL,
  `Evaluation_Grade` FLOAT NOT NULL,
  `Scientific_Field_Name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`ID`),
  INDEX `fk_Project_Executive1_idx` (`Executive_ID`
    ASC),
  INDEX `fk_Project_Program1_idx` (`Program_ID` ASC),
  INDEX `fk_Project_Foundation1_idx` (`Foundation_ID`
    ASC),
  INDEX `fk_Project_Researcher2_idx`
    (`Researcher_ID_Boss` ASC),
  INDEX `fk_Project_Researcher3_idx`
    (`Researcher_ID_eval` ASC),
  INDEX `fk_Project_Scientific_Field1_idx`
    (`Scientific_Field_Name` ASC),
  CONSTRAINT `fk_Project_Executive1` FOREIGN KEY
    (`Executive_ID`)
    REFERENCES `mydb`.`Executive` (`ID`)
    ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `fk_Project_Program1` FOREIGN KEY
    (`Program_ID`)
    REFERENCES `mydb`.`Program` (`ID`)
    ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `fk_Project_Foundation1` FOREIGN KEY
    (`Foundation_ID`)
    REFERENCES `mydb`.`Foundation` (`ID`)
    ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `fk_Project_Researcher2` FOREIGN KEY
    (`Researcher_ID_Boss`)
    REFERENCES `mydb`.`Researcher` (`ID`)
    ON DELETE NO ACTION ON UPDATE NO ACTION,

```

Project	
ID	INT
Title	VARCHAR(45)
Start_Date	DATE
End_Date	DATE
Summary	TEXT
Funding	FLOAT
Executive_ID	INT
Program_ID	INT
Foundation_ID	INT
Researcher_ID_Boss	VARCHAR(45)
Researcher_ID_eval	VARCHAR(45)
Evaluation_date	VARCHAR(45)
Evaluation_Grade	FLOAT
Scientific_Field_Name	VARCHAR(45)
Indexes	
PRIMARY	
fk_Project_Executive1_idx	
fk_Project_Program1_idx	
fk_Project_Foundation1_idx	
fk_Project_Researcher2_idx	
fk_Project_Researcher3_idx	
fk_Project_Scientific_Field1_idx	

```

CONSTRAINT `fk_Project_Researcher3` FOREIGN KEY
(`Researcher_ID_eval`)
REFERENCES `mydb`.`Researcher` (`ID`)
ON DELETE NO ACTION ON UPDATE NO ACTION,
CONSTRAINT `fk_Project_Scientific_Field1` FOREIGN
KEY (`Scientific_Field_Name`)
REFERENCES `mydb`.`Scientific_Field` (`Name`)
ON DELETE NO ACTION ON UPDATE CASCADE
) ENGINE=INNODB;

```

Ο παραπάνω πίνακας περιέχει όλα τα στοιχεία για ένα έργο. Εντός αυτού του πίνακα αποθηκεύεται και ο ερευνητής που έχει αξιολογήσει ένα έργο (`Researcher_ID_eval`) μαζί με την ημερομηνία αξιολόγησης και τον βαθμό (τα χαρακτηριστικά μίας σχέσης στο ER Diagram) και ο ερευνητής που είναι επιστημονικός υπεύθυνος (`Researcher_ID_Boss`) αφού και οι δύο πρόκειται για one-to-many σχέσεις και δεν απαιτούν έξτρα πίνακα για να αναπαρασταθούν. Δεδομένου πως στην εκφώνηση απαιτείται κάθε έργο να έχει τουλάχιστον ένα επιστημονικό πεδίο δημιουργώντας ένα έργο αναγκάζουμε τον χρήστη να εισάγει ένα.

10 - Παραδοτέα / Deliverable

```

-- Table `Deliverable`
--
DROP TABLE IF EXISTS `Deliverable`;
CREATE TABLE IF NOT EXISTS `Deliverable` (
  `ID` INT NOT NULL AUTO_INCREMENT,
  `Title` VARCHAR(45) NOT NULL,
  `Summary` TEXT NULL,
  `Submission_Date` DATE NOT NULL,
  `Project_ID` INT NOT NULL,
  PRIMARY KEY (`ID`),
  INDEX `fk_Deliverable_Project1_idx` (`Project_ID`
    ASC),
  CONSTRAINT `fk_Deliverable_Project1` FOREIGN KEY
    (`Project_ID`)
    REFERENCES `Project` (`ID`)
    ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=INNODB;

```

Deliverable	
ID	INT
Title	VARCHAR(45)
Summary	TEXT
Submission_Date	DATE
Project_ID	INT
Indexes	
PRIMARY	
fk_Deliverable_Project1_idx	

Αφού κάθε παραδοτέο μπορεί να αντιστοιχιστεί μόνο σε ένα έργο κατευθείαν με τη δημιουργία του παραδοτέου εισάγεται και σε έργο αυτό αντιστοιχεί.

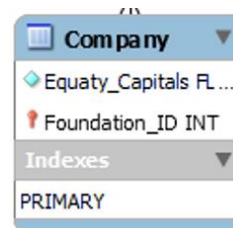
ISA Σχέση

Για την υλοποίηση της ISA σχέσης μεταξύ οργανισμού και εταιρείας, πανεπιστημίου και ερευνητικού κέντρου δημιουργήσαμε έναν πίνακα που περιέχει όλα τα στοιχεία για όλους τους οργανισμούς αλλά όχι την πληροφορία σε ποια υποκατηγορία ανήκει. Στη συνέχεια φτιάξαμε έναν πίνακα για κάθε υποκατηγορία που έχει ως κλειδί το foreign key από το Foundation και τον προϋπολογισμό της κάθε μίας. Από το front-end κομμάτι της εφαρμογής δεν επιτρέπουμε στον χρήστη να εισάγει π.χ. στον πίνακα εταιρεία έναν οργανισμό που είναι ήδη σε άλλη

κατηγορία γιατί δίνουμε τη δυνατότητα δημιουργίας οργανισμού μόνο από κάτω προς τα πάνω επιλέγοντας πρώτα τι είδους οργανισμός είναι.

11 - Εταιρεία / Company

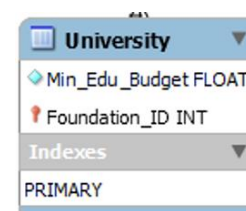
```
-- Table `Company`  
  
DROP TABLE IF EXISTS `Company`;  
CREATE TABLE IF NOT EXISTS `Company` (  
  `Equaty_Capitals` VARCHAR(45) NOT NULL,  
  `Foundation_ID` INT NOT NULL,  
  PRIMARY KEY (`Foundation_ID`),  
  CONSTRAINT `fk_Company_Foundation1` FOREIGN KEY  
    (`Foundation_ID`)  
    REFERENCES `Foundation` (`ID`)  
    ON DELETE CASCADE ON UPDATE CASCADE  
) ENGINE=INNODB;
```



Company	
Equaty_Capitals	FL...
Foundation_ID	INT
Indexes	
PRIMARY	

12 - Πανεπιστήμιο / University

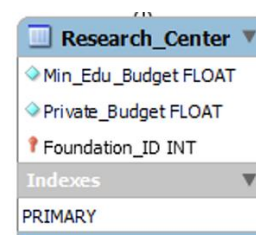
```
-- Table `University`  
  
DROP TABLE IF EXISTS `University`;  
CREATE TABLE IF NOT EXISTS `University` (  
  `Min_Edu_Budget` VARCHAR(45) NOT NULL,  
  `Foundation_ID` INT NOT NULL,  
  PRIMARY KEY (`Foundation_ID`),  
  CONSTRAINT `fk_University_Foundation1` FOREIGN KEY  
    (`Foundation_ID`)  
    REFERENCES `Foundation` (`ID`)  
    ON DELETE CASCADE ON UPDATE CASCADE  
) ENGINE=INNODB;
```



University	
Min_Edu_Budget	FLOAT
Foundation_ID	INT
Indexes	
PRIMARY	

13 - Ερευνητικό Κέντρο

```
-- Table `Research_Center`  
  
DROP TABLE IF EXISTS `Research_Center`;  
CREATE TABLE IF NOT EXISTS `Research_Center` (  
  `Min_Edu_Budget` VARCHAR(45) NOT NULL,  
  `Private_Budget` VARCHAR(45) NOT NULL,  
  `Foundation_ID` INT NOT NULL,  
  PRIMARY KEY (`Foundation_ID`),  
  CONSTRAINT `fk_Research_Center_Foundation1`  
    FOREIGN KEY (`Foundation_ID`)  
    REFERENCES `Foundation` (`ID`)  
    ON DELETE CASCADE ON UPDATE CASCADE  
) ENGINE=INNODB;
```



Research_Center	
Min_Edu_Budget	FLOAT
Private_Budget	FLOAT
Foundation_ID	INT
Indexes	
PRIMARY	

2.2.2. Triggers

Για να εξασφαλίσουμε την λογικά ορθή λειτουργία της βάσης καθώς και κάποιους περιορισμούς της εκφώνησης δημιουργήσαμε τα ακόλουθα triggers.

1 – researchers_dates: με αυτό το before insert trigger ελέγχουμε πριν πάει να γίνει εισαγωγή ενός ερευνητή αν αυτός είναι πολύ μεγάλος ή πολύ νέος και αν η ημερομηνία που πάει να δουλέψει σε έναν οργανισμό βγάζει νόημα σε σχέση με την ηλικία του.

```
delimiter $$
create trigger researchers_dates before insert on researcher
for each row
begin
    if (new.birth_date < '1920-1-1')
    then SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'The researcher seems to be
        too old!';
    end if;
    if (new.birth_date > '2020-1-1')
    then SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'The researcher seems to be
        an infant!';
    end if;
    if ((timestampdiff(year, new.birth_date, new.foundation_date)) < 12)
    then SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'The researcher seems to be
        too young for this foundation!';
    END IF;
end;
$$
```

2 – legitimate_worker: πριν γίνει εισαγωγή μίας σχέσης στον πίνακα researcher_works_on_project ελέγχουμε αν αυτός ο ερευνητής είναι και ο αξιολογητής του έργου και αν είναι δεν του επιτρέπουμε να εισαχθεί. Επίσης ελέγχουμε αν αυτός ο ερευνητής δούλεψε σε κάποιο οργανισμό κατά τη διάρκεια του έργου.

```
delimiter $$
create trigger legitimate_worker before insert on researcher_works_on_project
for each row
begin
    if exists (select * from project p where ((p.id = new.project_id) and
        (p.Researcher_ID_eval = new.researcher_id)))
    then
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'This researcher can't work for this project because
            he is the evaluator';
    END IF;

    if exists (select * from researcher r, project p where ((r.ID =
        new.researcher_id) and (p.id = new.project_id)
            and (r.Foundation_Date > p.end_date)))
    then
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'The researcher isn't working for any foundation
            during the the project';
    END IF;
end;
```

\$\$

5 – project_parameters: πριν γίνει εισαγωγή δεδομένων στον πίνακα project γίνεται έλεγχος για τα εξής:

- ο βαθμός της αξιολόγησης να ανήκει στο διάστημα [60, 100]
- ο αξιολογητής να μην είναι και επιστημονικός υπεύθυνος
- η ημερομηνία έναρξης να είναι πριν την ημερομηνία λήξης
- η διάρκεια να είναι από 1 μέχρι 4 έτη
- η χρηματοδότηση να είναι μεταξύ 100k και 1M
- η ημερομηνία αξιολόγησης να είναι πριν την ημερομηνία έναρξης
- ο αξιολογητής να δουλεύει σε κάποιο οργανισμό την ημερομηνία αξιολόγησης
- ο επιστημονικός υπεύθυνος να εργάζεται σε κάποιον οργανισμό κατά τη διάρκεια του έργου

```
delimiter $$
create trigger project_parameters before insert on project
for each row
begin

    if new.evaluation_grade > 100 or new.evaluation_grade < 60
    then
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Grade must be between 60 and 100 points';
    END IF;

    if new.Researcher_ID_eval = new.researcher_id_boss
    then
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'This researcher can't manage this project because he
        is the evaluator';
    END IF;

    if new.start_date > new.end_date
    then
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Start Date can't be after end date';
    END IF;

    if ((timestampdiff(year, new.start_date, new.end_date)) < 1)
    then
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Minimum duration is 1 year';
    END IF;

    if ((timestampdiff(year, new.start_date, new.end_date)) > 4)
    then
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Maximum duration is 4 years';
    END IF;

    if (new.funding > 1000000 or new.funding < 100000)
```

```

        then
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Funding must be between 100k and 1M Euros';
    END IF;

    if new.evaluation_date > new.start_date
    then
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Evaluation date must be before the start of the
        project';
    END IF;

    if exists (select * from researcher r where ((r.ID =
        new.researcher_id_boss) and r.Foundation_Date > new.end_date))
    then
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'The manager isn't working for any foundation during
        the project';
    END IF;

    if exists (select * from researcher r where ((r.ID =
        new.researcher_id_eval) and r.Foundation_Date > new.evaluation_date))
    then
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'The evaluator isn't working for any foundation at
        the moment of the evaluation of the the project';
    END IF;
end;
$$

```

*Όλα τα παραπάνω triggers υπάρχουν και για before update.

6 – add_1st_field: Μετά την εισαγωγή ενός έργου εισάγουμε το αρχικό επιστημονικό πεδίο στην σχέση project_scientific_field.

```

delimiter $$
create trigger add_1st_field after insert on project
for each row
begin
    INSERT into project_scientific_field (Project_ID,Scientific_field_name)
VALUES (new.id, new.scientific_field_name);
end;
$$

```

7 – add_boss: Μετά την εισαγωγή ενός έργου εισάγουμε τον επιστημονικό υπεύθυνο στη σχέση researcher_works_on_project.

```

delimiter $$
create trigger add_boss after insert on project
for each row
begin
    INSERT into researcher_works_on_project (Researcher_ID,Project_ID) VALUES
(new.researcher_id_boss, new.id);
end;

```

```
$$
```

8 - add_phone: Μετά την εισαγωγή ενός έργου εισάγουμε τον επιστημονικό υπεύθυνο στη σχέση foundation_extra_phones.

```
delimiter $$
create trigger add_phone after insert on foundation
for each row
begin
    INSERT into foundation_extra_phones (Phone_number,Foundation_ID) VALUES
(new.foundation_phone_1, new.id);
    INSERT into foundation_extra_phones (Phone_number,Foundation_ID) VALUES
(new.foundation_phone_2, new.id);
end;
$$
```

9 - boss_dont_leave_us: Δεν επιτρέπουμε να σβηστεί από τον πίνακα researcher_works_on_project ο υπεύθυνος του έργου.

```
delimiter $$
create trigger boss_dont_leave_us before delete on researcher_works_on_project
for each row
begin
    if exists (select * from project p where (p.researcher_id_boss =
old.researcher_id) and (p.id = old.project_id))
    then
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'This Relation can't be deleted because this Researcher
is the Manager of this Project';
    END IF;
end;
$$
```

10 - field_dont_leave_us: Δεν επιτρέπουμε να σβηστεί από τον πίνακα project_scientific_field μία σχέση αν υπάρχει μόνο ένα επιστημονικό πεδίο σε ένα έργο γιατί αναγκαστικά (βάσει της εκφώνησης) ένα έργο έχει τουλάχιστον ένα επιστημονικό πεδίο.

```
delimiter $$
create trigger field_dont_leave_us before delete on project_scientific_field
for each row
begin
    if not exists (select * from project_scientific_field psf where
(psf.scientific_field_name <> old.scientific_field_name)
and (psf.project_id = old.project_id))
    then
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'A project can't be left without a scientific field!';
    END IF;
end;
$$
```

2.2.3. Ευρετήρια – Indexes

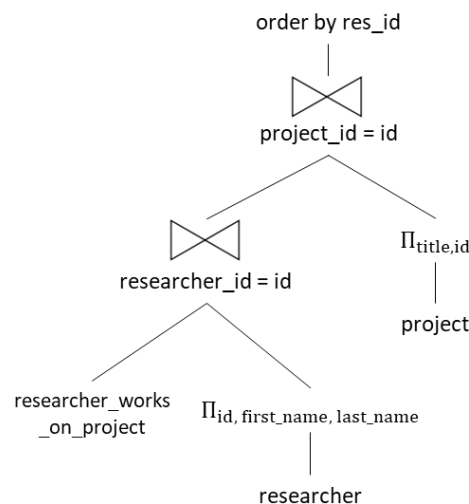
Ως ευρετήρια έχουμε διατηρήσαμε τα default indexes που δημιουργεί η MySQL δηλαδή indexes στα primary keys και επίσης σε κάθε πίνακα με foreign keys προσθέσαμε indexes στα foreign keys. Θεωρήσαμε πως δεν χρειάζονται άλλα indexes αφού οι αναζητήσεις γίνονται κυρίως βάσει των primary keys.

2.2.4 Views

Αποφασίσαμε να δημιουργήσουμε τις παρακάτω όψεις που κυρίως είναι εικονικοί πίνακες που απαντούν τα ερωτήματα 3.2-3.8 ή αντιστοιχούν σε άλλα στοιχεία (συνδυασμού των αρχικών πινάκων) που κρίναμε πως θα είναι συχνά προσπελάσιμα.

Για κάθε view παραθέτουμε query plan trees που οπτικοποιούν την λογική υλοποίησης, τον sql κώδικα και σύντομες επεξηγήσεις. Προσπαθούμε κάθε φορά τα selects να είναι όσο πιο χαμηλά στο δέντρο παρόλο που ο βελτιστοποιητής φροντίζει για αυτό όπως και αν γράψουμε τον κώδικα

1 – View για το ερώτημα 3.2: Έργα ανά ερευνητή



Για αυτό το view έχει γίνει **INNER JOIN** μεταξύ τριών πινάκων αφού θέλουμε τα στοιχεία του πίνακα `researcher_works_on_project`, που περιέχει όμως μόνο τα SSN και το ID των έργων, αλλά και τον τίτλο του έργου (από τον πίνακα `project`) και το όνομα του ερευνητή (από τον πίνακα `researcher`).

```
CREATE VIEW view32a AS
(SELECT
  res_id, first_name, last_name, Project_ID, title
FROM
  (SELECT
    *
  FROM
    researcher_works_on_project) a
  INNER JOIN
  (SELECT
```

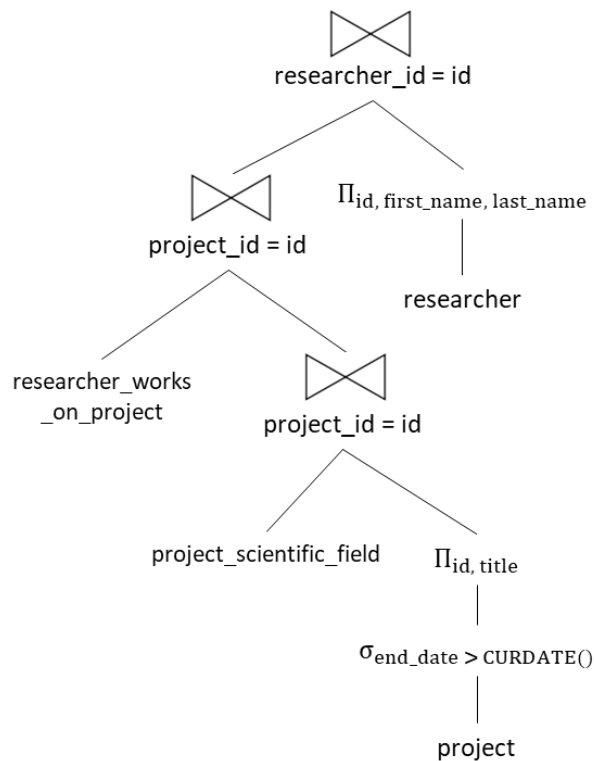


```

        r.id AS res_id, r.first_name, r.last_name
FROM
    researcher r) b ON a.researcher_id = b.res_id
INNER JOIN
(SELECT
    p.title, p.id
FROM
    project p) d ON d.id = project_id
ORDER BY res_id);

```

2 - View με ερευνητές που εργάζονται σε έργα σχετιζόμενα με ένα επιστημονικό πεδίο τον τελευταίο χρόνο (έστω και για λίγο) και τα έργα αυτά (ερώτημα 3.3)

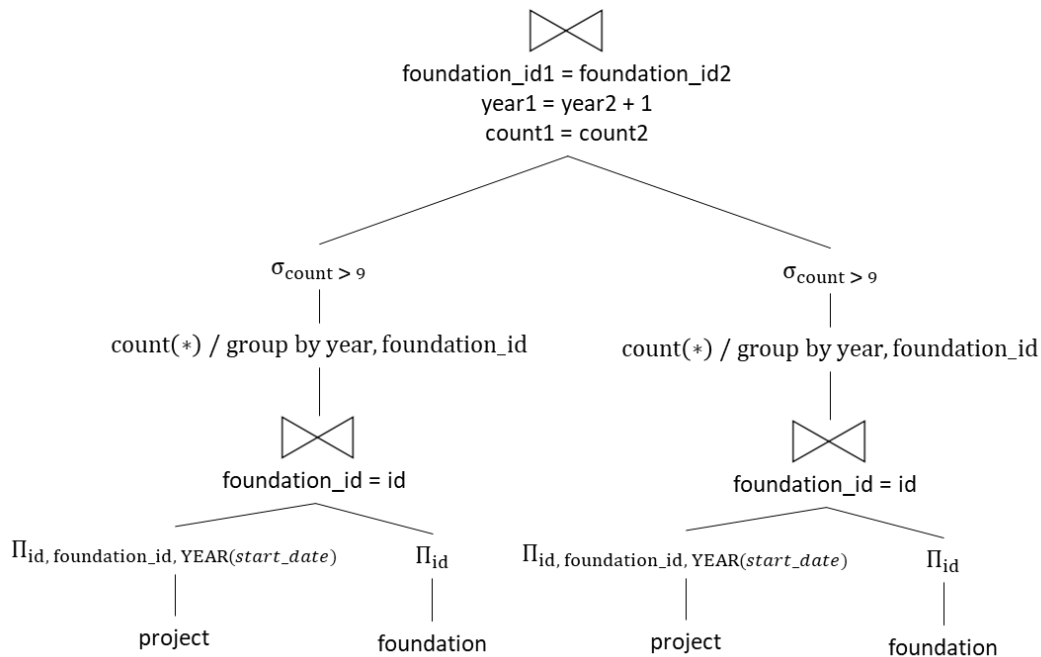


```

CREATE VIEW view33 AS
(SELECT
    ps.scientific_field_name,
    ps.project_id,
    p.title,
    rw.researcher_id,
    r.first_name,
    r.last_name,
    p.end_date
FROM
    (project_scientific_field ps
    INNER JOIN project p ON p.id = ps.project_id
    INNER JOIN researcher_works_on_project rw ON rw.project_id =
        ps.project_id
    INNER JOIN researcher r ON r.id = rw.researcher_id)
WHERE
    p.end_date > CURDATE());

```

3 – Οργανισμοί που έχουν λάβει τον ίδιο αριθμό έργων σε διάστημα δύο συνεχόμενων ετών, με τουλάχιστον 10 έργα ετησίως



Για να εξάγουμε τα επιθυμητά δεδομένα για αυτό το ερώτημα αρχικά δημιουργήσαμε έναν πίνακα με τα id των οργανισμών και όλα τα έργα που έχουν αναλάβει. Στη συνέχεια κάνοντας **GROUP BY** year , f_id και **COUNT**(*) μετρήσαμε πόσα έργα έχει κάθε οργανισμό κάθε χρονιά και μετά επιλέγουμε μόνο αυτού που έχουν count > 9 (τουλάχιστον 10). Εν τέλει κάνοντας **INNER JOIN** με τον εαυτό του **ON** k.f_id1 = b.f_id2 **AND** k.year1 = b.year2 + 1 **AND** k.number1 = b.number2 επιλέγουμε τους επιθυμητούς οργανισμούς.

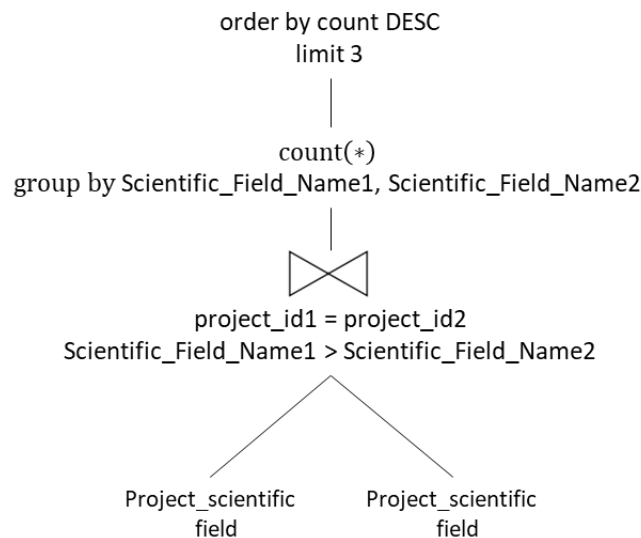
```
CREATE VIEW view34 AS
(SELECT
  *
FROM
  (SELECT
    *
  FROM
    (SELECT
      COUNT(*) AS number1,
      YEAR(p.start_date) AS year1,
      f.id AS f_id1,
      f.name AS name1
    FROM
      project p
    INNER JOIN foundation f ON p.foundation_id = f.id
    GROUP BY year1 , f_id1) g
  WHERE
    g.number1 > 9) k
  INNER JOIN
    (SELECT
      *
    FROM
      (SELECT
```

```

COUNT(*) AS number2,
YEAR(p.start_date) AS year2,
f.id AS f_id2,
f.name AS name2
FROM
project p
INNER JOIN foundation f ON p.foundation_id = f.id
GROUP BY year2 , f_id2) l
WHERE
l.number2 > 9) b ON k.f_id1 = b.f_id2
AND k.year1 = b.year2 + 1
AND k.number1 = b.number2);

```

4 – Πολλά έργα είναι διεπιστημονικά (δηλαδή καλύπτουν περισσότερα από ένα πεδία/ τομείς). Ανάμεσα σε ζεύγη πεδίων (π.χ. επιστήμη των υπολογιστών και μαθηματικά) που είναι κοινά στα έργα, βρείτε τα 3 κορυφαία (top-3) ζεύγη που εμφανίστηκαν σε έργα (ενεργά και μη ενεργά) – ερώτημα 3.5.



Εδώ κάνουμε τον πίνακα `project_scientific_field` με τον εαυτό του **ON** `ps1.Project_ID = ps2.Project_ID` **AND** `ps1.Scientific_Field_Name > ps2.Scientific_Field_Name` ώστε να μην πάρουμε τον ίδιο συνδυασμό 2 φορές και στη συνέχεια κάνοντας **COUNT (*)** **GROUP BY** και με τα δύο ονόματα των πεδίων φτιάχνουμε μία λίστα που μας λέει πόσες φορές εντοπίζεται κάθε συνδυασμός πεδίων. Από αυτά επιλέγουμε τα top 3 με **LIMIT 3**.

```

CREATE VIEW view35 AS
(SELECT
ps1.Scientific_Field_Name AS sf1,
ps2.Scientific_Field_Name AS sf2,
COUNT(*) AS count
FROM
project_scientific_field ps1
INNER JOIN
project_scientific_field ps2 ON ps1.Project_ID = ps2.Project_ID
AND ps1.Scientific_Field_Name > ps2.Scientific_Field_Name

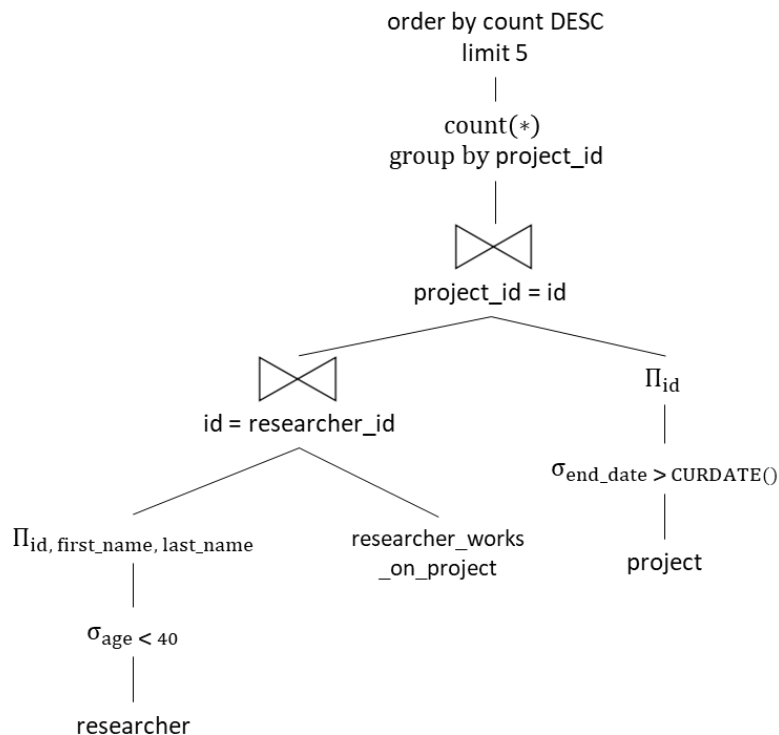
```

```

GROUP BY ps1.Scientific_Field_Name , ps2.Scientific_Field_Name
ORDER BY count DESC
LIMIT 3);

```

5 – Νέοι ερευνητές (ηλικία < 40 ετών) που εργάζονται στα περισσότερα ενεργά έργα και τον αριθμό των έργων που εργάζονται (ερώτημα 3.6).



Αρχικά επιλέγουμε τους νέους (<40 ετών) ερευνητές από τον πίνακα `researcher` και κάνοντας **INNER JOIN** με τον `researcher_works_on_project` βρίσκουμε σε ποια έργα δουλεύουν αυτοί. Έπειτα βρίσκουμε ποια έργα είναι ενεργά με `end_date > CURDATE()` από τον πίνακα `project`. Κάνουμε **INNER JOIN** στους δύο παραπάνω πίνακες που δημιουργήσαμε ώστε να αποκλείσουμε τα μη ενεργά έργα και **COUNT(*) GROUP BY id** βρίσκουμε σε πόσα ενεργά έργα εργάζεται ο κάθε νέος ερευνητής. Τους ταξινομούμε με **ORDER BY projects DESC** ώστε να εμφανίζονται με τη σωστή σειρά.

```

CREATE VIEW view36 AS
(SELECT
  id, first_name, last_name, COUNT(*) AS projects
FROM
  (SELECT
    *
  FROM
    (SELECT
      r.id, r.first_name, r.last_name, r.birth_date
    FROM
      researcher r
    WHERE
      DATEDIFF(CURDATE(), birth_date) < 14600) te

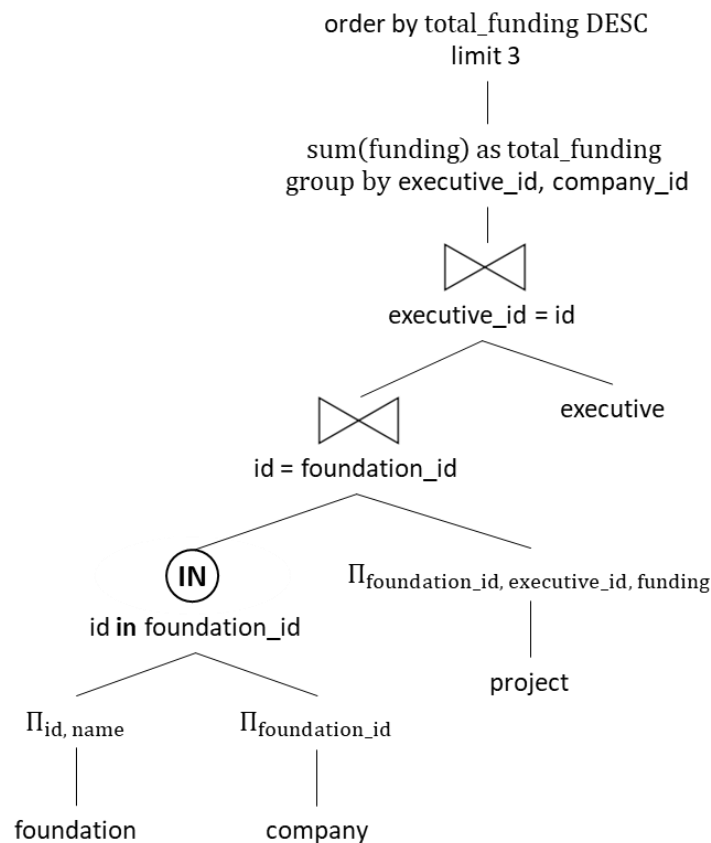
```

```

INNER JOIN researcher_works_on_project rp ON te.id = rp.researcher_id) ta
INNER JOIN
(SELECT
  p.id AS project_id, p.end_date
FROM
  project p
WHERE
  p.end_date > CURDATE()) tb ON tb.project_id = ta.project_id
GROUP BY id
ORDER BY projects DESC , last_name
LIMIT 5);

```

6 - top-5 στελέχη που δουλεύουν για το ΕΛ.ΙΔ.Ε.Κ. και έχουν δώσει το μεγαλύτερο ποσό χρηματοδοτήσεων σε μια εταιρεία. (όνομα στελέχους, όνομα εταιρείας και συνολικό ποσό χρηματοδότησης) – (ερώτημα 3.7)



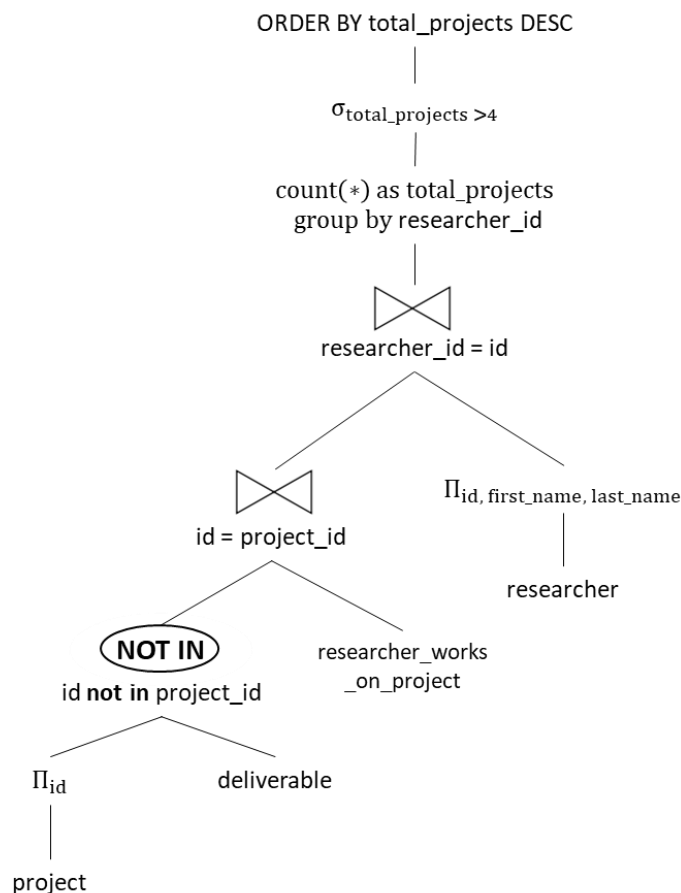
Εδώ αρχικά απομονώσαμε από τους οργανισμούς τις εταιρείες χρησιμοποιώντας τον τελεστή **IN** μεταξύ του `id` του πίνακα `foundation` και του `foundation_id` του πίνακα `company`. Έπειτα κάναμε **INNER JOIN ON** `foundation_id = id` μεταξύ με τον πίνακα `project` ώστε να δούμε ποια έργα σχετίζονται οργανισμό-εταιρεία. Στη συνέχεια κάναμε **INNER JOIN ON** `foundation_id = id` με τον πίνακα `project` ώστε να έχουμε και το όνομα του στελέχους. Τέλος αθροίζουμε τα `funding` που έχει δώσει κάθε στέλεχος σε μία εταιρεία κάνοντας **SUM(funding) GROUP BY executive_id , company_id** και ταξινομούμε με βάση αυτό το άθροισμα και παίρνουμε τα 3 πρώτα στελέχη.

```

CREATE VIEW view37 AS
  (SELECT
    executive_ID,
    first_name,
    last_name,
    name AS Company_Name,
    ta.id AS company_id,
    SUM(funding) AS total_funding
  FROM
    (SELECT
      p.funding, p.executive_id, p.foundation_id
    FROM
      project p) te
    INNER JOIN
    (SELECT
      f.name, f.id
    FROM
      foundation f
    WHERE
      f.id IN (SELECT
        c.foundation_id
      FROM
        company c)) ta ON te.foundation_id = ta.id
    INNER JOIN
    executive e ON e.id = te.executive_id
  GROUP BY executive_id , company_id
  ORDER BY total_funding DESC
  LIMIT 5);

```

7 - Ερευνητές που εργάζονται σε 5 ή περισσότερα έργα που δεν έχουν παραδοτέα (όνομα ερευνητή και αριθμός έργων)



Αρχικά βρήκαμε τα έργα που δεν έχουν παραδοτέα με τον τελεστή **NOT IN** μεταξύ του id του πίνακα $project$ και του $project_id$ του πίνακα $deliverable$. Έπειτα κάνουμε **INNER JOIN** με τον πίνακα $researcher_works_on_project$ για να δούμε ποιοι ερευνητές δουλεύουν σε έργα χωρίς παραδοτέα. Στη συνέχεια κάνουμε **INNER JOIN** με τον πίνακα $researcher$ για να λάβουμε και το όνομα του ερευνητή. Κάνουμε **COUNT(*) GROUP BY researcher_id** και βρίσκουμε σε πόσα έργα χωρίς παραδοτέα εργάζεται ο κάθε ερευνητής. Τέλος από αυτός επιλέγουμε όσους ισχύει η συνθήκη $total_projects > 4$ και τους ταξινομούμε με **ORDER BY total_projects DESC** ώστε να εμφανίζονται με τη σωστή σειρά.

```

CREATE VIEW view38 AS
(
  SELECT
    *
  FROM
    (
      SELECT
        project_id,
        researcher_id,
        first_name,
        last_name,
        COUNT(*) AS total_projects
      FROM
        (
          SELECT

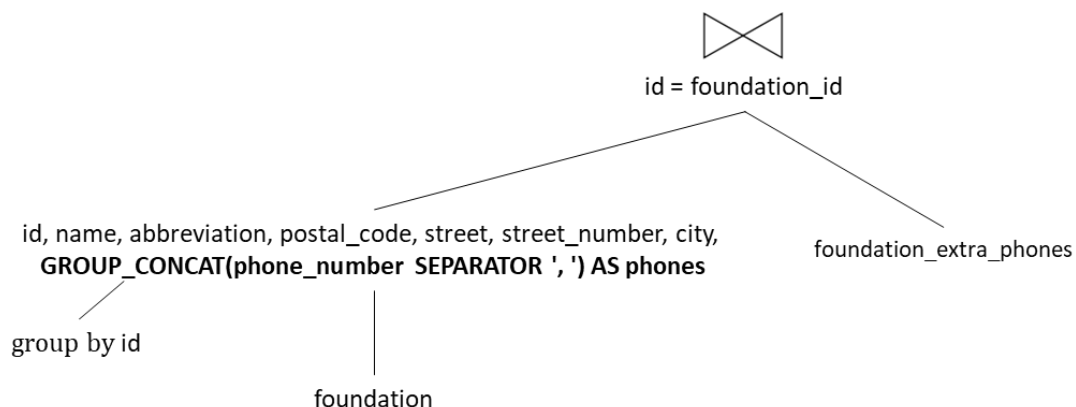
```

```

*
FROM
  (SELECT
    p.id
  FROM
    project p
  WHERE
    p.id NOT IN (SELECT
      d.project_id
    FROM
      deliverable d)) te
INNER JOIN researcher_works_on_project rp ON rp.project_id = te.id) ta
INNER JOIN (SELECT
  first_name, last_name, id AS res_id
FROM
  researcher) r ON r.res_id = ta.researcher_id
GROUP BY ta.researcher_id) tt
WHERE
  total_projects > 4
ORDER BY total_projects DESC , last_name);

```

8 – Οργανισμοί μαζί με τα τηλέφωνά τους (σε ένα attribute όλα τα τηλέφωνα)



Αυτό το view το χρησιμοποιήσαμε για να έχουμε έναν πίνακα όπου βρίσκονται όλα τα στοιχεία ενός οργανισμού και όλα τα τηλέφωνα του. Για να βάλουμε όλα τα τηλέφωνα σε ένα attribute χρησιμοποιήσαμε την εντολή `GROUP_CONCAT(phone_number SEPARATOR ',') AS phones` με `GROUP BY id`.

```

CREATE VIEW foundations_with_phones AS
SELECT
  id,
  name,
  abbreviation,
  postal_code,
  street,
  street_number,
  city,
  GROUP_CONCAT(phone_number
    SEPARATOR ',') AS phones
FROM

```



```

        foundation f
        INNER JOIN
        foundation_extra_phones p ON f.id = p.foundation_id
    GROUP BY id;

```

9 – Εταιρεία – Περιέχει όλα τα χαρακτηριστικά μίας εταιρείας

Για να δημιουργήσουμε αυτό το view κάναμε **INNER JOIN** μεταξύ του πίνακα `company` και του view `foundations_with_phones` που δημιουργήσαμε παραπάνω. Επίσης έχουμε χρησιμοποιήσει την συνάρτηση `FORMAT(equaty_capitals, 'C')` έτσι ώστε ένα νούμερο να διαμορφώνεται σε μορφή ως `currency`.

```

CREATE VIEW company_v AS
SELECT
    id,
    name,
    abbreviation,
    postal_code,
    street,
    street_number,
    city,
    phones,
    FORMAT(equaty_capitals, 'C') AS equaty_capitals
FROM
    foundations_with_phones
    INNER JOIN
    company ON id = foundation_id;

```

10 – Πανεπιστήμιο

Όπως ακριβώς και στο view `company`.

```

CREATE VIEW university_v AS
SELECT
    id,
    name,
    abbreviation,
    postal_code,
    street,
    street_number,
    city,
    phones,
    FORMAT(min_edu_budget, 'C') AS min_edu_budget
FROM
    foundations_with_phones
    INNER JOIN
    university ON id = foundation_id;

```

11 – Ερευνητικό Κέντρο

Όπως ακριβώς και στο view `company`.

```
CREATE VIEW research_center_v AS
SELECT
    id,
    name,
    abbreviation,
    postal_code,
    street,
    street_number,
    city,
    phones,
    FORMAT(min_edu_budget, 'C') AS min_edu_budget,
    FORMAT(private_budget, 'C') AS private_budget
FROM
    foundations_with_phones
    INNER JOIN
        research_center ON id = foundation_id;
```

2.2.5 Δημιουργία fake data

Για την δημιουργία των δεδομένων που εισάγουμε στη βάση χρησιμοποιήσαμε συναρτήσεις της βιβλιοθήκης faker της python και κατάλληλη λογική ώστε τα δεδομένα μας να πληρούν τις απαιτήσεις της βάσης και να μην ενεργοποιούνται triggers κατά την εισαγωγή τους. Ο κώδικας περιλαμβάνεται στο [git repo](#).

2.3 Βήματα Εγκατάστασης

1. Αρχικά κατεβάζουμε το anaconda από το [εδώ](#) και το εγκαθιστούμε.
2. Από το command prompt του ανακόντα με χρήση του cd μπαίνουμε στο directory του folder που κατεβάσαμε από το github.
3. Έπειτα πληκτρολογούμε και τρέχουμε την εντολή `pip install -r requirements.txt` ώστε να εγκαταστήσουμε όλα τα απαραίτητα πακέτα για το framework του webapp και για τη σύνδεση της βάσης με αυτό.
4. Στη συνέχεια, και αφού έχουμε κατοχυρώσει και επιβεβαιώσουμε σύνδεση κάνοντας χρήση του xampp (run as administrator), εκτελούμε τα αρχεία `schema.sql`, `triggers.sql`, `views.sql` και `data.sql` με την σειρά που αναγράφονται, ενδεικτικά μέσω του MySQL wokbench, έτσι ώστε να φτιάξουμε τη βάση μας, με όλα τα tables, triggers και views που είναι απαραίτητα για την ορθή λειτουργία της, και να τη γεμίσουμε με δεδομένα.
5. Ανοίγουμε ολόκληρο το folder που κατεβάσαμε από το github με κάποιο ide, επιβεβαιώνουμε ότι έχουμε ορίσει ως python interpreter τον anaconda3. Για το κάνουμε αυτό ευκολά θα μπορούσαμε να χρησιμοποιήσουμε ως ide το "Visual Studio Code", ανοίγοντάς το από τον "Anaconda Navigator".
6. Μέσω του ide, βάζουμε στο αρχείο `__init__.py` τα στοιχεία της βάσης και του user, προσοχή αν έχετε κωδικό για τον root user βγάλτε το πεδίο `app.config[MYSQL_PASSWORD]` από σχόλιο και πληκτρολογήστε σε αυτό τον κωδικό που έχετε ορίσει.
7. Τέλος εκτελούμε το αρχείο `run.py` και μπορούμε να ανοίξουμε τη βάση στο browser στη διεύθυνση που ορίζει ο host (by default: <http://localhost:3000>).

3 Web App Presentation

Την υλοποίηση του web app και την επικοινωνία του με την βάση πετύχαμε με χρήση Python. Συγκεκριμένα χρησιμοποιήσαμε τις βιβλιοθήκες Flask, Flask-MySQLdb και WTForms. Για να σταλούν queries από τη Python στη βάση και αντίστοιχα δεδομένα από τη βάση στη Python χρησιμοποιήσαμε ένα cursor object από τη Flask-MySQLdb σε συνδυασμό με τις κατάλληλες μεθόδους (execute, commit). Για το UI του web app χρησιμοποιήθηκε html.

3.1 Γενική Περιγραφή Περιβάλλοντος

Στην αρχική σελίδα φαίνονται τα αποτελέσματα από τα ερωτήματα που ζητούν συγκεκριμένο αριθμό αποτελεσμάτων ενώ για τα υπόλοιπα που ενδέχεται τα αποτελέσματα να είναι ολόκληρη λίστα υπάρχει κουμπί για μετάβαση σε άλλη σελίδα. Επίσης μπορεί κανείς να δει τα περιεχόμενα κάθε πίνακα που έχουμε στη βάση μας.

ΕΛΙΑΔΕΚ Data Base - Landing Page

Landing

View Projects

Query 3.1

View and Search Projects and CRUD functions

Show

View Researchers

View Researchers and CRUD functions

Show

Some views

Query 3.2

View 3.2a: Projects per Researcher + CRUD

View 3.2b: Companies

View Scientific Fields

Query 3.3

View Scientific Fields, CRUD functions and info about a popular field

Show

Magic Foundations

Query 3.4

'Murray-Glover' (ID: 5),
'Molina PLC' (ID: 17).

Top 3 pairs of scientific fields

Query 3.5

Pair 1: 'Mathematics' and 'Computer and information sciences'
Pair 2: 'Physical science' and 'Computer and information sciences'
Pair 3: 'Electrical engineering' and 'Computer and information sciences'

Top 5 young Researchers

Query 3.6

1. Karina Bennett works on 5 active projects
2. Charles Marsh works on 4 active projects
3. Jessica Bradley works on 3 active projects
4. Erica Brown works on 3 active projects
5. Jack Brown works on 3 active projects

Top 5 Executives

Query 3.7

1. Lee Jones gave to Company 'Vance, Tucker and Fisher' \$3,159,494.25 total fundings
2. Lisa Phillips gave to Company 'Hardy Inc' \$3,060,689.69 total fundings
3. Colleen Allen gave to Company 'Lopez-Marshall' \$2,698,541.02 total fundings
4. Lisa Phillips gave to Company 'Duffy Inc' \$2,339,961.53 total fundings
5. Thomas Novak gave to Company 'Lopez-Marshall' \$2,237,012.50 total fundings

Researchers working on 5 or more projects with no deliverables

Query 3.8

Show

View Foundations

View and Search Foundations and CRUD functions

Companies Universities

Research Centers

Programs

View Programs and CRUD functions

Show

Deliverables

View Deliverables and CRUD functions

Show

Executives

View Executives and CRUD functions

Show

Scientific Fileds per Project

View Scientific Fileds - Project and CRUD functions

Show

- 27 -

Όταν κανείς πατήσει να δει έναν συγκεκριμένο πίνακα μεταφέρεται στην ακόλουθη σελίδα όπου μπορεί να δημιουργήσει, διαβάσει, επεξεργαστεί και διαγράψει (CRUD) δεδομένα (εκτός από περιπτώσεις που κρίναμε πως αυτό δεν θα είχε νόημα).

[Add new Researcher](#)

SSN	First Name	Last Name	Sex	Birth Date	Age	Foundation Id	Foundation Date		
002-09-6083	Randy	Ray	male	1975-06-25	47	20	2007-10-24		
005-52-2165	Patrick	Lang	male	1963-09-21	59	19	1985-12-03		
007-87-7405	Cynthia	Garcia	female	1986-03-06	36	17	2006-01-06		
008-75-	Denise	Smith	female	1957-01-	65	19	1980-11-12		

Πατώντας το κουμπί δημιουργίας νέου ερευνητή μεταφέρεται στην ακόλουθη σελίδα.

SSN

First Name

Last Name

Gender :
☐ Male ☐ Female ☐ Other

Birth Date :

Choose Foundation

Date researcher entered the foundation:

[Create](#) [Back](#)

Χρησιμοποιώντας drop down lists (όταν ένας πίνακας έχει foreign keys οπότε ο χρήστης θα έπρεπε να μπει και να ψάξει σε άλλο πίνακα ποιο στοιχείο θα επιλέξει), radio buttons, πεδία επιλογής ημερομηνίας με ημερολόγιο και place holders δημιουργήσαμε ένα φιλικό και εύκολο προς το χρήστη περιβάλλον. Ακόμα στο κάτω μέρος υπάρχει κουπί back που τον επιστρέφει στην σελίδα των ερευνητών αν ο χρήστης αλλάξει γνώμη. Μετά την επιτυχή δημιουργία ενός ερευνητή εμφανίζεται μήνυμα επιτυχίας ενώ σε περίπτωση εισαγωγής λανθασμένων δεδομένων που παραβιάζουν τους περιορισμούς της βάσης εμφανίζεται το αντίστοιχο μήνυμα σφάλματος.

Αν ο χρήστης επιθυμεί να επεξεργαστεί τα δεδομένα πατώντας το αντίστοιχο κουμπί εμφανίζεται ένα pop up window με προσυμπληρωμένα τα δεδομένα ενός ερευνητή ώστε να γίνουν εύκολα όποιες αλλαγές. Και πάλι εμφανίζονται μηνύματα επιτυχίας ή αποτυχίας.

SSN	First Name	Last Name	Sex	Birth Date	Age	Foundation Date
000-00-0000	John					22-06-08
002-09-6083	Randy	Ray	male	1975-06-25	47	20
005-52-2165	Patrick	Lang	male	1963-09-21	59	19

Η διαγραφή γίνεται εύκολα πατώντας το κουμπί του κάδου. Πριν τη διαγραφή εμφανίζεται μήνυμα προειδοποίησης.

SSN	First Name	Last Name	Sex	Birth Date	Age	Foundation Date
000-00-	John	Briggs	male	2002-12-	20	1

Κατά αντίστοιχο τρόπο λειτουργούν και οι σελίδες για τους υπόλοιπους πίνακες της βάσης μας.

3.2 Extra Features

Στη σελίδα με τα έργα εκτός των άλλων υπάρχει μπάρα με αναζήτηση υπό κριτήρια ελάχιστης ημερομηνίας έναρξης ενός έργου, μέγιστης διάρκειας και SSN στελέχους.

ase - Projects Page

Min Start Date: Max Duration: Executive ID:

ID	Title	Start Date	End Date	Summary	Funding	Exec. ID	Pr. ID	Found. ID	Manager ID	Evaluator ID	Ev. Date	Ev. Grade			
1	integrate open-source niches	2016-01-03	2017-12-08	Surface wind necessary who people. Risk raise focus will skin whatever.	127061.0	127-38-1269	9	11	043-11-1851	167-13-1668	2015-08-30	86.0			<input type="button" value="workers"/>
2	deliver frictionless interfaces	2016-11-05	2020-04-11	Serious item better realize play.	569750.0	596-02-1281	4	24	352-96-5897	449-97-9290	2016-01-11	80.0			<input type="button" value="workers"/>

Επίσης δίνεται η δυνατότητα να δει κανείς ποιοι ερευνητές εργάζονται σε ένα έργο πατώντας το κουμπί workers.

Here is everyone that works on project 1

First Name	Last Name	Researcher ID
Jennifer	Parker	043-11-1851
Veronica	Mason	289-93-7032

Επίσης μετά τη δημιουργία ενός έργου ο χρήστης ανακατευθύνεται σε μία σελίδα όπου μπορεί να ορίσει όσους ερευνητές θέλει για αυτό το έργο.



ta Base - Assign Researchers to new Project

Project inserted successfully

Assign a Researcher as a worker:

Τέλος στη σελίδα των επιστημονικών πεδίων υπάρχει κουμπί info που μεταφέρει στον χρήστη σε μία σελίδα που παρουσιάζονται όλοι οι ερευνητές που εργάζονται σε έργα σχετιζόμενα με αυτό το πεδίο τον τελευταίο χρόνο καθώς και τα έργα αυτά (ερώτημα 3.3).

ic Fields Page

Add new Scientific Field		
Name		Query 3.3
Biological science		info
Chemical sciences		info


ic Fields Page

Interested in this Field? ×

Now you will see the researchers and the projects associated with this field at the last year

[Close](#) [Show me](#)

Chemical sciences



Query 3.3

[info](#)

[info](#)

Base - Magic Scientific Fields Page

You want it? I got it! Here is everything about Biological science ×

Back				
Project ID	Title	Researchers (ID)	First Name	Last Name
17	aggregate frictionless applications	178-49-6564	Alexis	Rodriguez
17	aggregate frictionless applications	305-94-0169	Tina	Smith
17	aqreqate frictionless applications	473-84-5558	James	Graham

GitHub repo: <https://github.com/jugger96/DB-NTUA-Database-for-ELIDEK>



The End...