



ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
Ακ. έτος 2021-2022, 6ο εξάμηνο, ΣΗΜΜΥ

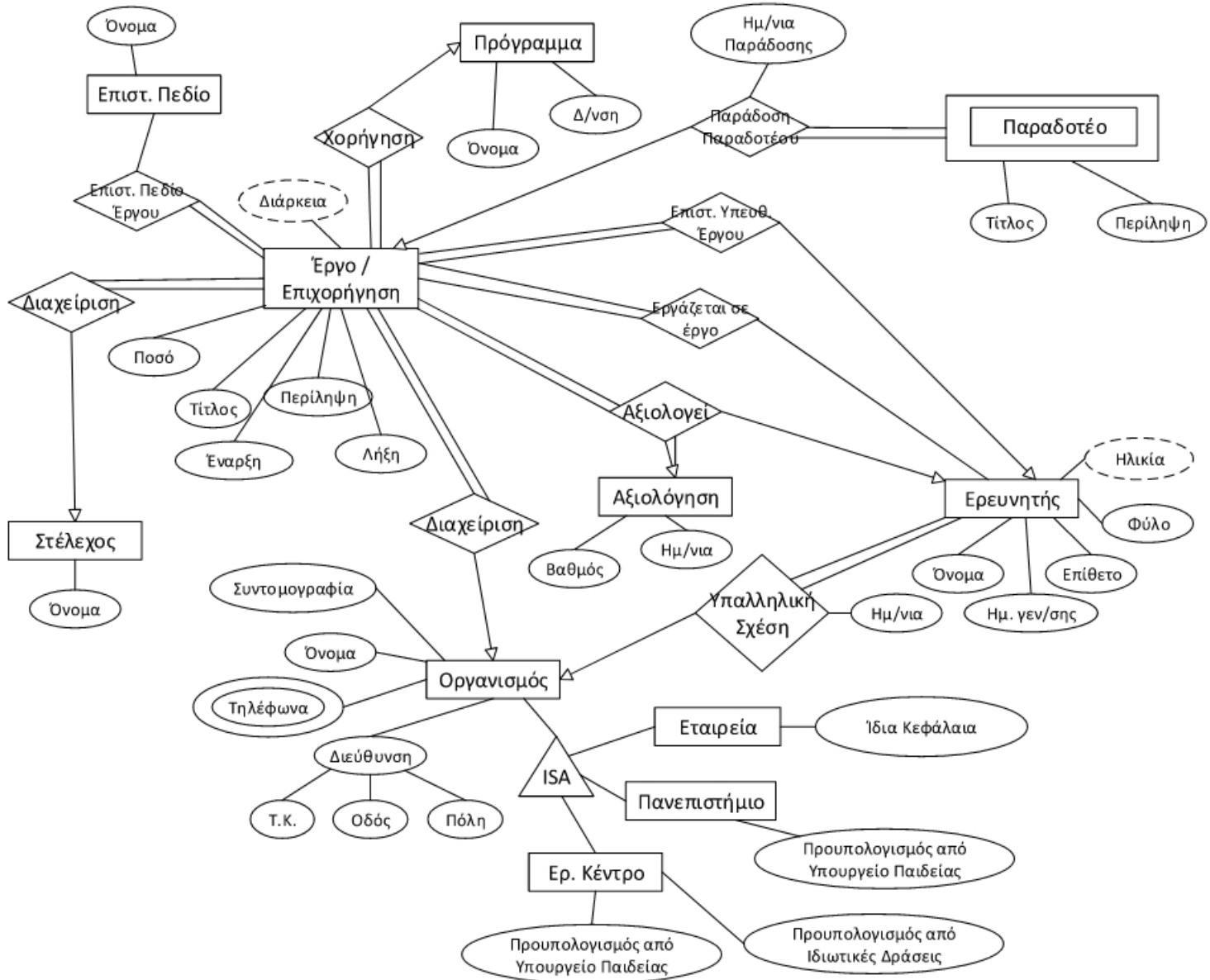
**Βάσεις Δεδομένων
Εξαμηνιαία Εργασία**

Αναφορά Εξαμηνιαίας Εργασίας

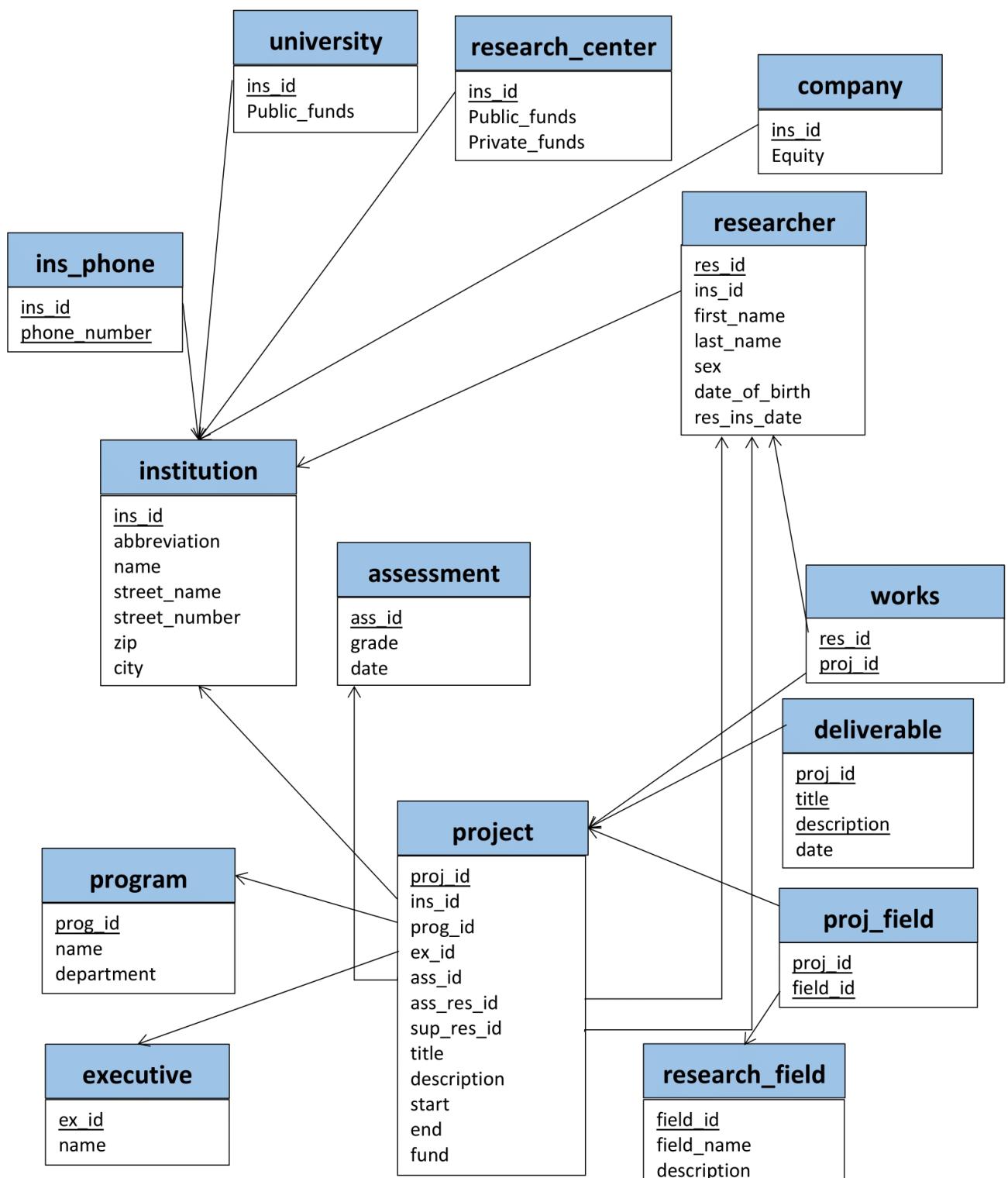
**Ραπτόπουλος Πέτρος (el19145)
Κόγιος Δημήτριος (el19220)
Αραβανής Τηλέμαχος (el19024)**

2.1

Για το σχεσιακό μοντέλο χρησιμοποιήσαμε το πρότυπο ER που μας δόθηκε:



Από αυτό δημιουργήσαμε το παρακάτω σχεσιακό μοντέλο:



Αποφασίσαμε να βάλουμε IDs σχεδόν σε κάθε νέα σχέση ώστε να έχουμε έναν εύκολο τρόπο να αναφερόμαστε στα primary keys.

Όπως γνωρίζουμε από τις διαλέξεις, τα composite attributes αναλύονται σε νέα attributes στο σχεσιακό μοντέλο. Για παράδειγμα, στο ER βλέπουμε ότι η διεύθυνση ενός οργανισμού αποτελείται από T.K, οδό και πόλη. Συνεπώς στο σχεσιακό μοντέλο φτιάχνουμε ξεχωριστά attributes για τα attributes street_name (όνομα οδού), street_number (αριθμός), zip (T.K) και city (πόλη).

Tα derived attributes δεν μπαίνουν στο σχεσιακό μοντέλο αφού μπορούμε να τα παράγουμε εύκολα από τα υπόλοιπα attributes μιας σχέσης. Για παράδειγμα, η διάρκεια ενός έργου και η ηλικία ενός ερευνητή ενώ υπάρχουν ως attributes των οντοτήτων “Έργο/επιχορήγηση” και “Ερευνητής” αντίστοιχα, δεν υπάρχουν ως attributes στις αντίστοιχες σχέσεις project και researcher του σχεσιακού.

Tα multivalued attributes γίνονται νέες σχέσεις στο σχεσιακό μοντέλο. Αυτό παρατηρείται στο attribute “Τηλέφωνα” της οντότητας “Οργανισμός” που στο σχεσιακό έγινε νέα σχέση ins_phone με νέα attributes ins_id (το ID του οργανισμού(institution) στο οποίο αναφέρεται) και phone_number. Το ζευγάρι ins_id και phone_number είναι primary key της σχέσης ins_phone και φυσικά το ins_id είναι foreign key από τη σχέση institution όπως φαίνεται και από το βελάκι που συνδέει τις δύο σχέσεις στο σχεσιακό μοντέλο.

Για να δημιουργήσουμε το σχεσιακό από το ER αρχικά δημιουργήσαμε μία σχέση για κάθε οντότητα του ER.
-Για κάθε one-to-many συσχέτιση δεν απαιτείται ξεχωριστό relation καθώς η συσχέτιση μπορεί να ενσωματωθεί κατάλληλα στα relation των εν λόγω entities.
-Για κάθε many-to-many συσχέτιση απαιτείται ξεχωριστό relation.

Πιο συγκεκριμένα:

-Όπως βλέπουμε στο ER, κάθε έργο έχει ένα ή και περισσότερα επιστημονικά πεδία. Επειδή η συσχέτισή μας είναι πολλά-προς-πολλά, εκτός από τις σχέσεις για τις οντότητες έργο(project) και επιστημονικό πεδίο(research_field), πρέπει να δημιουργήσουμε και μία σχέση proj_field η οποία θα έχει ως attributes τα proj_id(δηλ ID του έργου) και field_id(δηλ ID του επιστημονικού πεδίου) και θα συνδέει τα έργα με τα επιστημονικά πεδία στα οποία υπάγονται. Και τα δύο attributes της σχέσης proj_field είναι foreign keys από τις σχέσεις project και research_field αντίστοιχα. Ταυτόχρονα, το ζευγάρι proj_id,field_id είναι primary key της σχέσης proj_field.

-Μόνο ένα στέλεχος του ΕΛ.ΙΔ.ΕΚ διαχειρίζεται κάθε έργο. Η συσχέτιση είναι one-to-many (και με total participation στην πλευρά του many) αφού ένα στέλεχος μπορεί και να διαχειρίζεται πάνω από ένα έργο. Άρα, όπως γνωρίζουμε, δεν χρειάζεται να δημιουργήσουμε ξεχωριστό relation για τη συσχέτιση “Διαχείριση” αφού λόγω της one-to-many συσχέτισης, αρκεί να βάλουμε ως attribute στη σχέση project το primary key του στελέχους που το διαχειρίζεται(ex_id). Αυτό είναι και foreign key από τη σχέση έργο(project) προς τη σχέση executive(στέλεχος).

Η ίδια απλοποίηση μπορεί να γίνει στη συσχέτιση “Χορήγηση” όπου απλά βάζουμε ως attribute του έργου το primary key του προγράμματος που του προσφέρει τη χορήγηση(prog_id). Όπως προαναφέραμε, αυτό είναι και foreign key από τη σχέση project(έργο) προς τη σχέση program(Πρόγραμμα).

Ομοίως, στη συσχέτιση “Επιστ.Υπεύθ. Έργου” που είναι ξανά one-to-many (και με total participation στην πλευρά του many), περνάμε ως attribute του έργου το primary key του επιστημονικού υπεύθυνού του (to attribute sup_res_id προκύπτει από το ID του supervisor researcher). Με αυτόν τον τρόπο απλοποιούμε ξανά τη συσχέτιση με το foreign key sup_res_id από τη σχέση project(έργο) προς τη σχέση researcher(ερευνητής).

Ακόμα μία συσχέτιση του ER που μπορεί να απλοποιηθεί στο σχεσιακό μοντέλο είναι η one-to-many (με total participation στην πλευρά του many) συσχέτιση “Διαχείριση” όπου περνάμε το primary key του institution(οργανισμός) ως foreign key στη σχέση project(έργο). Με αυτόν τον τρόπο έχουμε ως attribute του έργου τον(έναν και μόνο) οργανισμό που το διαχειρίζεται.

Η one-to-many (με total participation στην πλευρά του many) συσχέτιση “Υπαλληλική σχέση” μπορεί επίσης να απλοποιηθεί από το σχεσιακό μοντέλο αρκεί να περάσουμε ως attribute στον researcher(ερευνητή) ένα foreign key προς το institution(οργανισμό) στον οποίο δουλέψει. Παρατηρούμε επίσης ότι η συσχέτιση “Υπαλληλική σχέση” έχει και attribute την ημερομηνία που ο ερευνητής ξεκίνησε να εργάζεται για τον οργανισμό. Αυτό μπορούμε να το περάσουμε ως attribute του ερευνητή(res_ins_date από το researcher institution date).

Ακόμη μία συσχέτιση one-to-many (με total participation στην πλευρά του many) είναι η συσχέτιση “Παράδοση Παραδοτέου” όπου συνδέει την οντότητα “Έργο/Επιχορήγηση” με την αδύναμη(weak) οντότητα “Παραδοτέο”. Αρχικά, δημιουργούμε νέα σχέση για την οντότητα “Παραδοτέο” η οποία θα έχει ως attributes το ID του project για το οποίο παραδίδεται, τον τίτλο του παραδοτέου, την περίληψή του και την ημερομηνία παράδοσης του η οποία στο ER είναι attribute της συσχέτισης “Παράδοση Παραδοτέου”. Η τριάδα attributes proj_id, title,description χαρακτηρίζει μοναδικά την αδύναμη οντότητα Παραδοτέο (deliverable) και για αυτό μπαίνει ως primary key της σχέσης deliverable στο σχεσιακό. Φυσικά το proj_id είναι foreign key από τη σχέση deliverable(παραδοτέο) στη σχέση project(έργο).

Η συσχέτιση “Εργάζεται σε Έργο” είναι many-to-many συνεπώς δεν γίνεται να απλοποιηθεί και πρέπει να δημιουργήσουμε νέα σχέση για αυτήν τη συσχέτιση στο σχεσιακό μοντέλο. Αυτό το κάναμε στη σχέση works που έχει ως primary key το ζευγάρι attributes proj_id,res_id. Το πρώτο ID είναι foreign key προς τη σχέση project(έργο) ενώ το δεύτερο foreign key προς τη σχέση researcher(ερευνητής). Μέσα από τη σχέση works μπορούμε να βρούμε ποιος ερευνητής δουλεύει στο κάθε έργο.

Για να εξηγήσουμε ότι ο οργανισμός(institution) μπορεί να είναι είτε Πανεπιστήμιο(university), είτε Εταιρεία(company) , είτε Ερευνητικό Κέντρο(research_center) δημιουργήσαμε τις τρεις ομώνυμες σχέσεις που παίρνουν ως foreign key (και primary) το ID του οργανισμού(ins_id) τον οποίο χαρακτηρίζουν και ως attributes τα αντίστοιχα attributes που μας δίνει το ER.

Τέλος, από την τριαδική συσχέτιση “Αξιολογεί” παίρνουμε την εξής πληροφορία:

Κάθε έργο έχει μία και μόνο αξιολόγηση η οποία έγινε από έναν συγκεκριμένο ερευνητή (τον αξιολογητή) μια συγκεκριμένη ημερομηνία και πήρε έναν βαθμό. Για να αποτυπώσουμε αυτήν την πληροφορία στο σχεσιακό μοντέλο δημιουργήσαμε μία ξεχωριστή σχέση assessment(αξιολόγηση) για την οντότητα “Αξιολόγηση”. Αφού το κάθε έργο έχει μία μόνο αξιολόγηση περάσαμε το primary key(ass_id από το assessment ID) της σχέσης assessment(αξιολόγηση) ως foreign key στη σχέση project(έργο). Επίσης, αφού το κάθε έργο έχει έναν και μόνο αξιολογητή περνάμε ως foreign key προς τη σχέση researcher(αφού ο αξιολογητής είναι κάποιος ερευνητής) από τη σχέση project(έργο) το ID του αντίστοιχου αξιολογητή που αξιολόγησε το συγκεκριμένο project(ass_res_id από το assessor researcher ID).

Όλα τα foreign keys αποτυπώνονται στο σχεσιακό μοντέλο μέσα από τα βελάκια που έχουμε βάλει.

2.2

Για τη δημιουργία/επεξεργασία της βάσης μας χρησιμοποιήσαμε τη mySQL και αυτά που αναφέρουμε ισχύουν για τη mySQL, δεν γνωρίζουμε τι ισχύει για τα υπόλοιπα DBMS.

Επισυνάπτονται τα δύο αρχεία create.sql και insert.sql τα οποία περιέχουν τα DDL και DML scripts αντίστοιχα.

Ξεκινώντας με το DDL script, το αρχείο create.sql περιέχει :

- Τα tables όλων των σχέσεων που φαίνονται στο σχεσιακό μοντέλο.
- Τα indexes.
- Τα views.
- Τους users και τα δικαιώματα που τους δίνονται.
- Τα triggers για να ελέγχονται διάφορα constraints (π.χ ότι ο αξιολογητής του έργου δεν μπορεί να ανήκει στο δυναμικό του οργανισμού που συμμετέχει στην πρόταση).

Ας δούμε πιο αναλυτικά πως δημιουργήσαμε κάποια tables :

Για κάθε table βάλαμε την εντολή DROP TABLE IF EXISTS ώστε να μπορούμε να ξαναδημιουργούμε τη βάση όταν κάναμε κάποιον έλεγχο.

DROP TABLE IF EXISTS institution;

CREATE TABLE institution (

```
ins_id INT UNSIGNED NOT NULL AUTO_INCREMENT,  
abbreviation VARCHAR(10) NOT NULL,  
name VARCHAR(100) NOT NULL,  
street_name VARCHAR(45) NOT NULL,  
street_number SMALLINT UNSIGNED NOT NULL,  
zip INT UNSIGNED NOT NULL,  
city VARCHAR(45) NOT NULL,  
PRIMARY KEY (ins_id)  
)ENGINE=InnoDB AUTO_INCREMENT=78001 DEFAULT CHARSET=utf8mb4  
COLLATE=utf8mb4_0900_ai_ci;
```

Εδώ βλέπουμε το table του οργανισμού(institution).

Τα attributes του είναι τα ίδια με τα attributes της σχέσης institution στο σχεσιακό μοντέλο. Για δική μας ευκολία ορίσαμε ότι τα IDs του κάθε table θα ξεκινούν με διαφορετικό αριθμό. Για τους οργανισμούς ορίσαμε ότι τα IDs τους θα ξεκινάνε με το 78 (όπως οι φοιτητές της σχολής HMMY έχουν AM που ξεκινάει από 031). Αυτό το

```

DROP TABLE IF EXISTS ins_phone;
CREATE TABLE ins_phone (
    ins_id INT UNSIGNED NOT NULL,
    phone_number VARCHAR(20) NOT NULL,          /* not null cause its
primary key */
    PRIMARY KEY (ins_id, phone_number),
    CONSTRAINT fk_phone_ins FOREIGN KEY (ins_id)
        REFERENCES institution (ins_id) ON DELETE CASCADE ON UPDATE
CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

Περισσότερο ενδιαφέρον παρουσιάζει το παραπάνω table που ταυτίζεται με τη σχέση ins_phone του σχεσιακού (πρόκεται για το τηλέφωνο του οργανισμού που όπως προαναφέραμε, έγινε νέα σχέση στο σχεσιακό καθώς ήταν multivalued attribute στο ER).

Ως primary key έχει το ζευγάρι ins_id, phone_number και το ins_id είναι foreign key από το table της προηγούμενης σελίδας. Μετά από συνομιλία μας με βοηθό του μαθήματος στο discord, αποφασίσαμε να κάνουμε αυτό το foreign key ON DELETE CASCADE έτσι ώστε κάθε φορά που διαγράφουμε έναν οργανισμό να διαγράφονται αυτόματα και τα τηλέφωνα του. Αυτό το κάναμε καθώς ο user ο οποίος θα έχει δικαιώματα να κάνει manipulate τα δεδομένα της βάσης είναι έμπειρο μέλος του τμήματος IT και γνωρίζει καλά τι αλλαγές θα συμβούν άμα διαγράψει κάτι από τη βάση (επίσης, η εφαρμογή μας, βγάζει ενημερωτικό μήνυμα όταν πάμε να διαγράψουμε κάτι από τη βάση γνωστοποιώντας μας ποιες άλλες αλλαγές θα συμβούν). Το ON UPDATE CASCADE, θα μεταφέρει αυτόματα οποιεσδήποτε αλλαγές γίνουν στο parent table (institution) στο table “παιδί” (ins_phone). Δηλαδή άμα κάνουμε update το ins_id στη σχέση institution, θα αλλάξει αυτόματα το ins_id της σχέσης ins_phone για τα τηλέφωνα του αντίστοιχου οργανισμού.

```

DROP TABLE IF EXISTS university;
CREATE TABLE university (
    ins_id INT UNSIGNED NOT NULL,
    Public_funds INT UNSIGNED NULL,
    PRIMARY KEY (ins_id),
    CONSTRAINT fk_uni_ins FOREIGN KEY (ins_id)
        REFERENCES institution (ins_id) ON DELETE CASCADE ON UPDATE
        CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

Εδώ βλέπουμε τη σχέση/table university(πανεπιστήμιο) το οποίο είναι οργανισμός και για αυτό έχει ως primary και foreign key το ins_id που είναι και το primary key του αντίστοιχου οργανισμού. Το ON DELETE CASCADE θα διαγράψει αυτόματα το tuple του table university όταν σβήσουμε την αντίστοιχη tuple από το table institution(όπως είπαμε ο χρήστης που έχει δικαίωμα διαγραφής, γνωρίζει καλά τι αλλαγές θα επιφέρει μια επικείμενη διαγραφή).
Τα tables research_center και company είναι εντελώς αντίστοιχα με το university και μπορείτε να τα δείτε στο αρχείο create.sql .

```

DROP TABLE IF EXISTS program;
CREATE TABLE program (
    prog_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    name VARCHAR(45) NOT NULL,
    department VARCHAR(100) NOT NULL,
    PRIMARY KEY (prog_id)
)ENGINE=InnoDB AUTO_INCREMENT=23001 DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

```

To table program(πρόγραμμα) έχει primary key ID που ξεκινάει με 23 σύμφωνα με δική μας απόφαση για την ευκολότερη κωδικοποίηση των δεδομένων.
Παρατηρούμε ότι το παραπάνω table δεν έχει κανένα foreign key.

```

DROP TABLE IF EXISTS executive;

CREATE TABLE executive (
    ex_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    name VARCHAR(45) NOT NULL,
    PRIMARY KEY (ex_id)
)ENGINE=InnoDB AUTO_INCREMENT=11001 DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

```

To table executive(στέλεχος ΕΛΙΔΕΚ) μοιάζει αρκετά με το table program που δείχαμε πιο πάνω, δεν έχει foreign key αλλά μόνο primary key που ξεκινάει από 11 και τα κατάλληλα attributes από το ER και το σχεσιακό μοντέλο.

```

DROP TABLE IF EXISTS assessment;

CREATE TABLE assessment (
    ass_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    grade TINYINT NOT NULL,
    date DATE NOT NULL,
    CONSTRAINT con_grade CHECK (grade between 40 AND 100),
    PRIMARY KEY (ass_id)
)ENGINE=InnoDB AUTO_INCREMENT=13001 DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

```

To table assessment(αξιολόγηση) έχει primary key που ξεκινάει από 13 και κανένα foreign key. Έχει όμως ένα check constraint που ελέγχει ότι ο βαθμός της αξιολόγηση είναι ≥ 40 και ≤ 100 . Αυτό καθώς αποφασίσαμε ότι στη βάση μας θα μπούνε μόνο έργα που έχουν περάσει την αξιολόγηση και θα υλοποιηθούν. Κοιτώντας το επίσημο σάιτι του ΕΛΙΔΕΚ, ως κατώτατο βαθμό επιτυχημένης αξιολόγησης βάλαμε το 40 και ως ανώτατο το 100.

```

DROP TABLE IF EXISTS researcher;

CREATE TABLE researcher (
    res_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    ins_id INT UNSIGNED NOT NULL,
    first_name VARCHAR(45) NOT NULL,
    last_name VARCHAR(45) NOT NULL,
    sex ENUM('M','F','O') NULL, /* 'O' stands for other */
    date_of_birth DATE NULL,
    res_ins_date DATE NOT NULL,
    PRIMARY KEY (res_id),
    CONSTRAINT fk_res_ins FOREIGN KEY (ins_id)
        REFERENCES institution (ins_id) ON DELETE CASCADE ON UPDATE
        CASCADE
)ENGINE=InnoDB AUTO_INCREMENT=310001 DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

```

Όπως βλέπουμε, το primary key res_id των ερευνητών ξεκινάει από 31. Ο κάθε ερευνητής δουλεύει για έναν και μόνο institution(οργανισμό) και για αυτό έχει foreign key το ID του οργανισμού για τον οποίο δουλεύει. Αυτό το foreign key είναι ON DELETE CASCADE και ON UPDATE CASCADE καθώς όπως εξηγήσαμε πιο πάνω, γνωρίζουμε τι αλλαγές θα συμβούν εάν διαγράψουμε κάτι από τη βάση. Πράγματι, άμα το σκεφτούμε με βάση τον πραγματικό κόσμο, άμα κλείσει μια εταιρεία ή ένα ερευνητικό κέντρο, θα απολυθούν και όλοι οι εργαζόμενοι. Συνεπώς, βγάζει νόημα όταν σβήνουμε έναν οργανισμό να σβήσουμε και τους ερευνητές που δούλευαν εκεί.

```

DROP TABLE IF EXISTS project;

CREATE TABLE project (
    proj_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    ins_id INT UNSIGNED NOT NULL,
    prog_id INT UNSIGNED NOT NULL,
    ex_id INT UNSIGNED NOT NULL,
    ass_id INT UNSIGNED NOT NULL,
    ass_res_id INT UNSIGNED NOT NULL,
    sup_res_id INT UNSIGNED NOT NULL,
    title VARCHAR(90) NOT NULL,
    description VARCHAR(450) NULL,
    start DATE NULL,
    end DATE NULL,
    fund numeric(9,2) NULL
    CONSTRAINT con_funding CHECK (fund between 100000 AND 1000000),
    CONSTRAINT con_duration CHECK ((datediff(end, start)/365.25) between 0.99 AND
4.0),
    PRIMARY KEY (proj_id),
    CONSTRAINT fk_proj_ins FOREIGN KEY (ins_id)
        REFERENCES institution (ins_id) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_proj_prog FOREIGN KEY (prog_id)
        REFERENCES program (prog_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT fk_proj_ex FOREIGN KEY (ex_id)
        REFERENCES executive (ex_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT fk_proj_ass FOREIGN KEY (ass_id)
        REFERENCES assessment (ass_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT fk_proj_ass_res FOREIGN KEY (ass_res_id)
        REFERENCES researcher (res_id) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_proj_sup_res FOREIGN KEY (sup_res_id)
        REFERENCES researcher (res_id) ON DELETE RESTRICT ON UPDATE CASCADE
)ENGINE=InnoDB AUTO_INCREMENT=45001 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

Το πιο πολύπλοκο και μεγάλο table είναι το table του project(έργο).

Αυτό έχει ως primary key το proj_id που ξεκινάει από 45.

Κάθε project έχει έναν οργανισμό που το διαχειρίζεται συνεπώς κάθε έργο έχει foreign key το ins_id του οργανισμού που το διαχειρίζεται.

Επίσης, έχει ένα πρόγραμμα που το χρηματοδοτεί οπότε θα έχει foreign key το prog_id, δηλαδή το ID του προγράμματος που το χορηγεί.

Ακόμα, κάθε πρόγραμμα έχει ένα στέλεχος του ΕΛΙΔΕΚ που το διαχειρίζεται. Επιπλέον, έχει έναν αξιολογητή researcher, έναν υπεύθυνο researcher, μια αξιολόγηση και τα υπόλοιπα attributes. Για τα πρώτα τρία έχει foreign keys από τα tables researcher(για τα πρώτα δύο) και assessment(για το τρίτο).

Για τα check constraints του συγκεκριμένου table, όπως αναφέρει η εκφώνηση, ένα έργο μπορεί να έχει χρηματοδότηση από εκατό χιλιάδες έως ένα εκατομμύριο ευρώ, ενώ η διάρκειά του μπορεί να είναι από ένα εως και τέσσερα έτη. Για να υπολογίσουμε τη διάρκεια ενός έργου, χρησιμοποιήσαμε τη συνάρτηση datediff της SQL που αφαιρεί δύο ημερομηνίες και μας δίνει τη διαφορά σε ημέρες. Υστερα διαιρούμε το αποτέλεσμα με το 365.25 για να μην έχουμε πρόβλημα με τα δίσεκτα έτη. Θέλουμε το αποτέλεσμα αυτό να είναι μεταξύ 0.99 και 4 (έτη). Το κάτω όριο είναι 0.99 και όχι 1 καθώς εάν ένα έργο έχει διάρκεια ακριβώς ένα έτος που δεν είναι δίσεκτο(για παράδειγμα έστω ότι ξεκινάει στις 14/8/2021 και τελειώνει στις 14/8/2022 , παραρηρείστε ότι το 2022 δεν είναι δίσεκτο έτος) τότε η συνάρτηση datediff θα επιστρέψει ακριβώς 365 και η διαίρεση με το 365.25 θα βγει μικρότερη του 1. Για να παρακάμψουμε αυτό το πρόβλημα βάλαμε το 0.99.

Παρατηρείστε ότι σε κάποια foreign keys έχουμε βάλει ON DELETE CASCADE ενώ σε άλλα ON DELETE RESTRICT για λόγους που θα εξηγηθούν πιο κάτω όταν θα εξηγήσουμε τα triggers. Για τώρα ας πούμε ότι είναι λογικό όταν διαγράφεται από τη βάση ένας οργανισμός(πχ κλείνει μια εταιρεία) να σταματάνε και τα projects της. Για να μην έχουμε errors κατά τη διαγραφή αυτή, βάλαμε και ON DELETE CASCADE στο ass_res_id (ID του αξιολογητή). Για περισσότερες λεπτομέρειες δείτε τα τελευταία triggers, στο τέλος αυτού του κειμένου.

DROP TABLE IF EXISTS works;

CREATE TABLE works (

```
res_id INT UNSIGNED NOT NULL,
proj_id INT UNSIGNED NOT NULL,
PRIMARY KEY (res_id, proj_id),
CONSTRAINT fk_works_res FOREIGN KEY (res_id)
    REFERENCES researcher (res_id) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT fk_works_proj FOREIGN KEY (proj_id)
    REFERENCES project (proj_id) ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

To table works περιγράφει ποιος researcher δουλένει σε κάθε project. Τα δύο foreign keys , είναι ON DELETE CASCADE καθώς άμα διαγραφεί(πχ απολυθεί) κάποιος ερευνητής τότε αυτόματα σταματάει να δουλεύει στα έργα και αν διαγραφεί ένα έργο(πχ ακυρώνεται λόγω ελλιπής χρηματοδότησης) δεν έχει νόημα να πούμε ότι υπάρχουν ερευνητές που δουλεύουν σε μη υπάρχοντα έργα.

```

DROP TABLE IF EXISTS deliverable;
CREATE TABLE deliverable (
    proj_id INT UNSIGNED NOT NULL,
    title VARCHAR(45) NOT NULL,
    description VARCHAR(450) NOT NULL,
    date DATE NOT NULL,
    PRIMARY KEY (proj_id, title, description),
    CONSTRAINT fk_del_proj FOREIGN KEY (proj_id)
        REFERENCES project (proj_id) ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

```

To table deliverable(παραδοτέο) έχει ως foreign key το project για το οποίο παραδίδεται και ως primary key την τριάδα proj_id,title,description. Στο foreign key proj_id μπαίνει ON DELETE CASCADE καθώς στο ER η οντότητα παραδοτέο είναι αδύναμη οπότε είναι λογικό να σβήσουμε τα παραδοτέο έργων που δεν υπάρχουν πλέον.

```

DROP TABLE IF EXISTS research_field;
CREATE TABLE research_field (
    field_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    field_name VARCHAR(45) NOT NULL,
    description VARCHAR(450) NULL,
    PRIMARY KEY (field_id)
)ENGINE=InnoDB AUTO_INCREMENT=99001 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

To table research_field(επιστημονικό πεδίο) περιγράφεται από πάνω και έχει το primary key field_id να ξεκινάει από 99.

Γενικό σχόλιο για τα tables: Παρατηρείστε ότι πολλά foreign keys έχουν τον περιορισμό NOT NULL. Έτσι επιτυγχάνεται το total participation που προβλέπεται από το ER. Για παράδειγμα, το foreign key ex_id στη σχέση project είναι NOT NULL αφού κάθε έργο πρέπει να έχει ένα στέλεχος του ΕΛΙΔΕΚ που το διαχειρίζεται.

```
DROP TABLE IF EXISTS proj_field;
CREATE TABLE proj_field (
    proj_id INT UNSIGNED NOT NULL,
    field_id INT UNSIGNED NOT NULL,
    PRIMARY KEY (proj_id, field_id),
    CONSTRAINT fk_field_proj FOREIGN KEY (proj_id)
        REFERENCES project (proj_id) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_proj_field FOREIGN KEY (field_id)
        REFERENCES research_field (field_id) ON DELETE RESTRICT ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Βλέπουμε το table του proj_field το οποίο περιγράφει με ποιο επιστημονικό πεδίο ασχολείται το κάθε έργο. Για αυτό έχει δύο foreign key από τα αντίστοιχα tables ενώ για primary key έχει το ζευγάρι των δύο αυτών primary keys.

To foreign key proj_id είναι ON DELETE CASCADE ώστε όταν διαγράφουμε ένα project, να διαγράφονται και οι tuples της σχέσης proj_field ενώ το foreign key field_id είναι ON DELETE RESTRICT αφού δεν είναι λογικό να διαγράψουμε ένα επιστημονικό πεδίο από τη βάση, δεν γίνεται δηλαδή να πούμε ότι σταματάει να υπάρχει Φυσική ή Χημεία.

Αυτά ήταν τα tables. Ας ρίξουμε μια σύντομη ματιά στα indexes:

```
CREATE INDEX idx_res_ins_id ON researcher(ins_id);
CREATE INDEX idx_proj_ins_id ON project(ins_id);
CREATE INDEX idx_prog_id ON project(prog_id);
CREATE INDEX idx_ex_id ON project(ex_id);
CREATE UNIQUE INDEX idx_ass_id ON project(ass_id);
CREATE INDEX idx_ass_res_id ON project(ass_res_id);
CREATE INDEX idx_sup_res_id ON project(sup_res_id);
```

```
CREATE INDEX idx_start_date ON project(start);
CREATE INDEX idx_end_date ON project(end);
CREATE INDEX idx_date_of_birth ON researcher(date_of_birth);
```

Όσο αφορά τα indexes διαχρίνουμε τις επιλογές μας στις δύο παρακάτω κατηγορίες. Αρχικά ορίσαμε indexes σε όλα τα Foreign key της βάσης μας. Με αυτό τον τρόπο βελτιστοποιούμε την απόδοση των διαφόρων join των Query μας που χρησιμοποιούν τα Foreign key για να δημιουργήσουν υποσύνολα του καρτεσιανού γινομένου δύο σχέσεων. Επιπλέον αποτελεί σχεδιαστική επιλογή σε ορισμένες σχέσεις να έχουμε ON DELETE CASCADE ON UPDATE CASCADE στα FOREIGN KEY REFERENCES και ακόμη και όταν δεν συμβαίνει αυτό να υπάρχουν κατάλληλα trigger που πριν διαγραφεί μια εγγραφή στη σχέση πατέρα να διαγράφονται οι εγγραφές που κληρονομούν FOREIGN KEY στη σχέση παιδί.

Αυτές οι λειτουργίες προφανώς βασίζονται σε αναζητήσεις πάνω στα FOREIGN KEY μεταξύ σχέσεων και άρα επωφελούνται από το παραπάνω indexing. Επειτα ορίσαμε indexes σε attribute σχέσεων που χρησιμοποιούνται συχνά για την επιλογή εγγραφών σε συγκεκριμένα Query(WHERE clause). Συγκεκριμένα στις ημερομηνίες έναρξης και λήξης των έργων που χρησιμοποιούνται στα Query 3,6,8 πχ για να προσδιορίσουμε τα ενεργά έργα, στην ημερομηνία γέννησης του ερευνητή που χρησιμοποιείται στο Query 6 για να προσδιοριστεί η ηλικία του ερευνητή. Επιπλέον στο όνομα του ερευνητικού πεδίου που χρησιμοποιείται για την ομαδοποίηση εγγραφών(GROUP BY clause) στο Query 5 ώστε να προκύψουν τα project ανά τούπλα ερευνητικών πεδίων. Παρατηρείστε παρακάτω ότι η επίδοση του Query 3 για την εύρεση των ερευνητών που εργάστηκαν σε ένα συγκεκριμένο ερευνητικό πεδίο τον τελευταίο χρόνο αλλάζει σημαντικά με την προσθήκη των παραπάνω indexes. Ανάλογη βελτιστοποίηση παρατηρείται και στα υπόλοιπα Queries που αναφέρθηκαν

Σημείωση: Παρατηρείστε ότι το index idx_ass_id ON project(ass_id) είναι UNIQUE.

Αυτό έγινε ώστε να μην γίνεται δύο projects να έχουν την ίδια αξιολόγηση. Σημειώνεται ότι αυτό θα μπορούσε και να ελεγχθεί μέσω UNIQUE CONSTRAINT στο foreign key ass_id της σχέσης project.

Τέλος, στα indexes δεν έχουμε βάλει DROP INDEX IF EXISTS, καθώς πρώτον η mySQL δεν έχει τέτοιο statement και δύετερον καθώς δεν γίνεται να υπάρξει index χωρίς το table στο οποίο αναφέρεται οπότε όταν κάνουμε drop τα tables διαγράφονται και τα indexes.

Χωρίς indexes:

```
MariaDB [HFRI]> SELECT DISTINCT r.res_id, r.first_name, r.last_name, r.sex, r.date_of_birth, r.ins_id, r.res_ins_date FROM research_field rf
-> INNER JOIN proj_field pf
-> ON rf.field_id = pf.field_id
-> INNER JOIN project p
-> ON pf.proj_id = p.proj_id
-> INNER JOIN works w
-> ON p.proj_id = w.proj_id
-> INNER JOIN researcher r
-> ON w.res_id = r.res_id
-> WHERE rf.field_id = 99012 AND (p.start <= current_date() AND p.end >= DATE_SUB(current_date(), INTERVAL 1 YEAR));
+-----+-----+-----+-----+-----+-----+
| res_id | first_name | last_name | sex | date_of_birth | ins_id | res_ins_date |
+-----+-----+-----+-----+-----+-----+
| 310007 | Maisy | Andrews | F | 1959-07-25 | 78007 | 1992-12-09 |
| 310037 | Yannis | Walker | M | 1972-01-20 | 78007 | 2000-05-16 |
| 310067 | Kole | Holman | M | 1979-06-24 | 78007 | 2004-02-28 |
| 310088 | Ida | Lister | F | 1966-02-14 | 78008 | 1985-06-18 |
| 310038 | Cosmo | Edge | M | 1972-09-11 | 78008 | 2000-09-21 |
| 310068 | Leonard | O'Doherty | M | 1979-10-27 | 78008 | 2011-03-26 |
| 310098 | Jeanne | Dennis | F | 1983-08-04 | 78008 | 2007-12-14 |
| 310009 | Ingrid | Holding | F | 1970-08-19 | 78009 | 2003-01-29 |
| 310039 | Karishma | Berger | F | 1979-07-31 | 78009 | 2009-09-16 |
| 310015 | Spike | Ellison | M | 1980-02-11 | 78015 | 2007-01-28 |
| 310045 | Kay | Andrew | M | 1979-01-07 | 78015 | 2007-06-01 |
| 310004 | Sheena | Hamer | F | 1983-08-17 | 78004 | 2008-03-15 |
+-----+-----+-----+-----+-----+-----+
12 rows in set (0.004 sec)
```

Με indexes:

```
MariaDB [HFRI]> SELECT DISTINCT r.res_id, r.first_name, r.last_name, r.sex, r.date_of_birth, r.ins_id, r.res_ins_date FROM research_field rf
-> INNER JOIN proj_field pf
-> ON rf.field_id = pf.field_id
-> INNER JOIN project p
-> ON pf.proj_id = p.proj_id
-> INNER JOIN works w
-> ON p.proj_id = w.proj_id
-> INNER JOIN researcher r
-> ON w.res_id = r.res_id
-> WHERE rf.field_id = 99012 AND (p.start <= current_date() AND p.end >= DATE_SUB(current_date(), INTERVAL 1 YEAR));
+-----+-----+-----+-----+-----+-----+
| res_id | first_name | last_name | sex | date_of_birth | ins_id | res_ins_date |
+-----+-----+-----+-----+-----+-----+
| 310007 | Maisy | Andrews | F | 1959-07-25 | 78007 | 1992-12-09 |
| 310037 | Yannis | Walker | M | 1972-01-20 | 78007 | 2000-05-16 |
| 310067 | Kole | Holman | M | 1979-06-24 | 78007 | 2004-02-28 |
| 310088 | Ida | Lister | F | 1966-02-14 | 78008 | 1985-06-18 |
| 310038 | Cosmo | Edge | M | 1972-09-11 | 78008 | 2000-09-21 |
| 310068 | Leonard | O'Doherty | M | 1979-10-27 | 78008 | 2011-03-26 |
| 310098 | Jeanne | Dennis | F | 1983-08-04 | 78008 | 2007-12-14 |
| 310009 | Ingrid | Holding | F | 1970-08-19 | 78009 | 2003-01-29 |
| 310039 | Karishma | Berger | F | 1979-07-31 | 78009 | 2009-09-16 |
| 310015 | Spike | Ellison | M | 1980-02-11 | 78015 | 2007-01-28 |
| 310045 | Kay | Andrew | M | 1979-01-07 | 78015 | 2007-06-01 |
| 310004 | Sheena | Hamer | F | 1983-08-17 | 78004 | 2008-03-15 |
+-----+-----+-----+-----+-----+-----+
12 rows in set (0.001 sec)
```

Μετά τα indexes έχουμε δημιουργήσει και τα παρακάτω views:

```
DROP VIEW IF EXISTS projects_per_researcher;  
CREATE VIEW projects_per_researcher  
(res_id, first_name, last_name, num_of_projects)  
AS  
SELECT r.res_id, r.first_name, r.last_name, count(*) FROM researcher r  
INNER JOIN works w  
ON r.res_id = w.res_id  
GROUP BY r.res_id  
UNION  
SELECT r.res_id, r.first_name, r.last_name, 0 FROM researcher r  
LEFT OUTER JOIN works w  
ON r.res_id = w.res_id  
WHERE w.proj_id IS NULL;
```

```
DROP VIEW IF EXISTS projects_per_institution_per_year;  
CREATE VIEW projects_per_institution_per_year  
(ins_id, abbreviation, name, projects, year)  
AS  
SELECT i.ins_id, i.abbreviation, i.name, count(*), YEAR(p.start) as year FROM  
institution i  
INNER JOIN project p  
ON i.ins_id = p.ins_id  
GROUP BY i.ins_id, year;
```

Το πρώτο είναι έργα ανά ερευνητή. Παρατηρείστε ότι γίνεται union και outer join ώστε να δούμε και τους ερευνητές που δεν δουλεύουν σε κάποιο έργο αυτή τη χρονική στιγμή.

Το δεύτερο view είναι έργα ανά οργανισμό κάθε χρόνο και μας βοήθησε στην επίλυση του ερωτήματος 3.4.

Στη συνέχεια έχουμε δημιουργήσει δύο users:

```
DROP USER IF EXISTS 'User_1'@'localhost';
```

```
DROP USER IF EXISTS 'User_2'@'localhost';
```

```
CREATE USER 'User_1'@'localhost' IDENTIFIED BY 'DBdb11@@@';
```

```
CREATE USER 'User_2'@'localhost' IDENTIFIED BY 'DBdb22@@@';
```

```
GRANT ALL ON HFRI.* TO 'User_1'@'localhost';
```

```
GRANT SELECT ON HFRI.* TO 'User_2'@'localhost';
```

```
FLUSH PRIVILEGES;
```

Ο user 1 έχει δικαιώματα να κάνει όποια αλλαγή θέλει στη βάση αφού πρόκειται για κάποιο μέλος του τμήματος IT που θα πρέπει να έχει δικαιώματα να κάνει διάφορα CRUD στη βάση, ενώ ο user 2 που είναι ένας απλός χρήστης, έχει δικαιώματα να κάνει μόνο select. Με τους αντίστοιχους κωδικούς μπορούμε να συνδεθούμε στη βάση είτε ως user 1 είτε ως user 2.

Τέλος, έχουμε βάλει και διάφορα triggers για να γίνονται διάφορα checks κατά τη διάρκεια των CRUD στα δεδομένα.

DELIMITER \$\$

DROP TRIGGER IF EXISTS delete_proj_assessment\$\$

/ delete assessment when you delete a project */*

CREATE TRIGGER delete_proj_assessment

after delete

ON project FOR EACH ROW

BEGIN

DELETE FROM assessment where ass_id = old.ass_id;

END \$\$

Αρχικά αρχικοποιούμε το delimiter σε κάποιο νέο χαρακτήρα ώστε η SQL να καταλαβαίνει που τελειώνει μια γραμμή και που τελειώνει μια εντολή.

Αυτό το trigger διαγράφει αυτόματα την αξιολόγηση ενός έργου από τη βάση όταν σβήνουμε το έργο ώστε να μην υπάρχουν αξιολογήσεις που δεν αξιολογούν κάποιο έργο.

DROP TRIGGER IF EXISTS assessor_trigger_on_project_insert\$\$

/ check if project assessor works at the same institution as the project */*

CREATE TRIGGER assessor_trigger_on_project_insert

before insert

ON project FOR EACH ROW

BEGIN

IF new.ass_res_id in (select r.res_id from researcher r where r.ins_id = new.ins_id) THEN

SIGNAL SQLSTATE '50001' SET MESSAGE_TEXT = 'Assessor cannot work for institution';

END IF;

END \$\$

Αυτό το trigger ελέχγει εάν ο αξιολογητής του έργου εργάζεται στον ίδιο οργανισμό που διαχειρίζεται το έργο και αν ναι, τότε δεν αφήνει την εισαγωγή του νέου αυτού project στη βάση και τυπώνει το ατνίστοιχο μήνυμα σφάλματος.

```

DROP TRIGGER IF EXISTS sup_trigger_on_project_insert $\$$ 
/* check if supervisor works at the same institution as the project*/
CREATE TRIGGER sup_trigger_on_project_insert
before insert
ON project FOR EACH ROW
BEGIN
IF new.sup_res_id not in (select r.res_id from researcher r where r.ins_id = new.ins_id)
THEN
SIGNAL SQLSTATE '50002' SET MESSAGE_TEXT = 'Supervisor must work for institution';
END IF;
END  $\$$ 

```

Το παραπάνω trigger ελέγχει εάν ο υπένθυνος ερευνητής ενός έργου εργάζεται στον οργανισμό ο οποίος διαχειρίζεται το έργο και αν όχι, τυπώνει σφάλμα.

```

DROP TRIGGER IF EXISTS proj_works_sup $\$$ 
/* after creating the project, automatically add the supervisor as a worker on that project*/
CREATE TRIGGER proj_works_sup
after insert
ON project FOR EACH ROW
BEGIN
INSERT INTO works (res_id, proj_id) VALUES (new.sup_res_id, new.proj_id);
END  $\$$ 

```

Αυτό το trigger δημιουργεί μια tuple στη σχέση works με το νέο έργο και τον νέο υπεύθυνο ερευνητή του αφού βάλουμε ένα νέο έργο στη βάση.

```

DROP TRIGGER IF EXISTS does_the_researcher_work_for_the_institution$$
/* check if the researcher you are adding as a worker to a project works at the same
institution as the project*/
CREATE TRIGGER does_the_researcher_work_for_the_institution
before insert
ON works FOR EACH ROW
BEGIN
IF ((select p.ins_id from project p where p.proj_id = new.proj_id) <> (select r.ins_id from
researcher r where r.res_id = new.res_id)) THEN
SIGNAL SQLSTATE '50003' SET MESSAGE_TEXT = 'This researcher cannot work in this
project as he is not at the same institution as the project';
END IF;
END $$

```

'Οπως λέει και το σχόλιο, αυτό το trigger ελέγχει εάν ο ερευνητής που πάμε να προσθέσουμε ως εργαζόμενο ενός έργου εργάζεται στον οργανισμό που έχει υποβάλλει την πρόταση για το έργο και αν όχι τυπώνει σφάλμα.

```

DROP TRIGGER IF EXISTS assessor_trigger_on_project_update$$
/* check if the new assessor works at the same institution as the project */
CREATE TRIGGER assessor_trigger_on_project_update
before update
ON project FOR EACH ROW
BEGIN
IF new.ass_res_id in (select r.res_id from researcher r where r.ins_id = new.ins_id) THEN
SIGNAL SQLSTATE '50004' SET MESSAGE_TEXT = 'New assessor cannot work for
institution';
END IF;
END $$

```

Αυτό το trigger είναι το ίδιο με το δεύτερο αλλά όταν γίνεται update και όχι insert.

```

DROP TRIGGER IF EXISTS sup_trigger_on_project_update$$
/* check if the new supervisor works at the same institution as the project*/
CREATE TRIGGER sup_trigger_on_project_updatebefore update
ON project FOR EACH ROW
BEGIN
IF new.sup_res_id not in (select r.res_id from researcher r where r.ins_id = new.ins_id)
THEN
SIGNAL SQLSTATE '50005' SET MESSAGE_TEXT = 'New supervisor must work for
institution';
END IF;
END $$
```

To ίδιο trigger με τον υπεύθυνο πριν αλλάξει update project.

```

DROP TRIGGER IF EXISTS does_the_researcher_work_for_the_institution_update$$
/* check if the researcher you are adding as a worker to a project works at the same
institution as the project*/
CREATE TRIGGER does_the_researcher_work_for_the_institution_update
before update
ON works FOR EACH ROW
BEGIN
IF ((select p.ins_id from project p where p.proj_id = new.proj_id) <> (select r.ins_id from
researcher r where r.res_id = new.res_id)) THEN
SIGNAL SQLSTATE '50006' SET MESSAGE_TEXT = 'This researcher cannot work in this
project as he is not at the same institution as the project';
END IF;
END $$
```

To ίδιο trigger με το τέταρτο αλλάξει update στο works.

```

DROP TRIGGER IF EXISTS proj_works_sup_update $\$$ 
/* after creating the project, automatically add the new supervisor as a worker on that
project and delete the old one*/
CREATE TRIGGER proj_works_sup_updateafter update
ON project FOR EACH ROW
BEGIN
SET @TRIGGER_DISABLED = 1;
delete from works where res_id = old.sup_res_id and proj_id = old.proj_id;
delete from works where res_id = new.sup_res_id and proj_id = old.proj_id; -- do not
create duplicates. Duplicates will mess up the process and not let the update go through.
insert into works (res_id,proj_id) values (new.sup_res_id,new.proj_id);
SET @TRIGGER_DISABLED = 0;
END  $\$$ 

```

Όταν κάνεις update τον υπεύθυνο ερευνητή του project, διέγραψε τον παλιό από τη σχέση works και βάλε τον καινούργιο. Με τη μεταβλητή TRIGGER_DISABLED απενεργοποιεί το trigger cannot_delete_supervisor(το οποίο εξηγείται παρακάτω) μέχρι να κάνει τις αλλαγές που πρέπει.

```

DROP TRIGGER IF EXISTS supervisor_cannot_be_updated_this_way $\$$ 
/* If the user tries to update the supervisor of a project from the works table then do not
allow it. Instead the user must update it by updating the project table.*/
CREATE TRIGGER supervisor_cannot_be_updated_this_way
before update
ON works FOR EACH ROW
BEGIN
IF old.res_id = (select sup_res_id from project where proj_id = old.proj_id) THEN
SIGNAL SQLSTATE '50007' SET MESSAGE_TEXT = 'Supervisor cannot be updated this
way';
END IF;
END  $\$$ 

```

Όταν ο χρήστης πάει να κάνει update στο works την tuple όπου είναι ο υπεύθυνος του project τότε δεν πρέπει να αφήσουμε αλλά εάν θέλουμε να αλλάξουμε τον υπεύθυνο ενός project να πρέπει να το κάνουμε μέσω update του project και ο νέος υπεύθυνος θα μπει αυτόματα στη σχέση works λόγω των προηγούμενων triggers.

```

DROP TRIGGER IF EXISTS cannot_delete_supervisor$$
/*
Do not allow deletion of supervisor from works table This trigger works ONLY if you
try to delete tuples from works one by one Otherwise no changes will go through*/
CREATE TRIGGER cannot_delete_supervisor
before delete
ON works FOR EACH ROW
BEGIN
IF @TRIGGER_DISABLED = 0 THEN
IF old.res_id = (select sup_res_id from project where proj_id = old.proj_id) THEN
SIGNAL SQLSTATE '50008' SET MESSAGE_TEXT = 'Supervisor cannot be deleted.';
END IF;
END IF;
END $$
```

Αυτό το trigger δεν επιτρέπει τη διαγραφή της tuple του υπεύθυνου του έργου από τη σχέση works και ελέγχεται από τη μεταβλητή TRIGGER_DISABLED(που αρχικοποιείται στο ο σημ αρχή του script). Όταν αυτή είναι ο, τότε το trigger λειτουργεί. Αυτό συμβαίνει ώστε να μην έχουμε πρόβλημα με το trigger proj_works_sup_update. Όταν ενεργοποιηθεί το proj_works_sup_update, δεν θέλουμε αυτό το trigger να μην το αφήνει να κάνει τις κατάλληλες αλλαγές στη σχέση works. Το trigger αυτό είναι απαραίτητο γιατί κάθε project πρέπει να έχει τουλάχιστον έναν ερευνητή που να εργάζεται σε αυτό, τον υπεύθυνο. Παρατηρείστε ότι η σχέση works είναι on delete cascade στο foreign key proj_id. Γνωρίζουμε ότι τα triggers δεν ενεργοποιούνται όταν γίνεται delete cascade (αυτό ισχύει στην MySQL, δεν είμαστε σίγουροι για τι ισχύει για τα υπόλοιπα DBMS), συνεπώς ο μόνος τρόπος για να διαγράψουμε την tuple της σχέσης works που έχει τον υπεύθυνο του έργου είναι να κάνουμε delete το project.

```

DROP TRIGGER IF EXISTS cannot_delete_all_project_fields$$
/*
Do not allow deletion of supervisor from works table
This trigger works ONLY if you try to delete tuples from proj_field one by one*/
CREATE TRIGGER cannot_delete_all_project_fieldsbefore delete
ON proj_field FOR EACH ROW
BEGIN
IF ((select count(*) from proj_field where proj_id = old.proj_id) = 1) THEN
SIGNAL SQLSTATE '50009' SET MESSAGE_TEXT = 'Cannot have a project with no research
fields';
END IF;
END $$
```

Αυτό το trigger ελέγχει ότι δεν υπάρχει project που να μην έχει κάποιο επιστημονικό πεδίο.

```
DROP TRIGGER IF EXISTS delete_projects_when_you_delete_programs$$
```

```
/* this trigger procs the first trigger which also deletes the assessment of a project after  
deleting that particular project */  
CREATE TRIGGER delete_projects_when_you_delete_programs  
before delete  
ON program FOR EACH ROW  
BEGIN  
delete from project where prog_id = old.prog_id;  
END $$
```

Όταν διαγράφουμε ένα program θέλουμε να διαγράφονται και όλα τα projects αυτού του προγράμματος. Ο λόγος που αυτό υλοποιήθηκε με trigger και όχι με on delete cascade στο foreign key prog_id της σχέσης project είναι γιατί με το on delete cascade δεν θα ενεργοποιούτων το πρώτο trigger, το οποίο διαγράφει την αξιολόγηση του έργου από τη βάση όταν διαγράφεται ένα έργο. Ενώ με το trigger, το ένα trigger ενεργοποιεί το άλλο ώστε να μην έχουμε junk πληροφορίες στη βάση μας. Άμα δεν θέλουμε να διαγράφουμε τα projects αυτού του έργου, θα πρέπει μέσω update των έργων στο foreign key prog_id , να νιοθετηθούν από κάποιο άλλο πρόγραμμα.

```
DROP TRIGGER IF EXISTS delete_projects_when_you_delete_executive$$
```

```
/* this trigger procs the first trigger which also deletes the assessment of a project after  
deleting that particular project */  
CREATE TRIGGER delete_projects_when_you_delete_executive  
before delete  
ON executive FOR EACH ROW  
BEGIN  
delete from project where ex_id = old.ex_id;  
END $$
```

Θέλουμε να διαγράφουμε τα projects ενός executive όταν αυτός σβήνεται από τη βάση. Άμα δεν θέλουμε να κάνουμε αυτή τη διάγραφη γιατί για παράδειγμα θέλουμε όταν συνταξιοδοτείται ένας executive, τα projects του να τα αναλαμβάνει ένας νέος executive, αρκεί να κάνουμε update το ex_id των projects του παλιού executive. Υλοποιήσαμε trigger και όχι on delete cascade στο foreing key ex_id της σχέσης project για τους ίδιους ακριβώς λόγους που εξηγούμε στο προηγούμενο trigger.

Στο τέλος του αρχείου create.sql, επαναφέρουμε το delimiter :

```
DELIMITER ;
```

Για το DML script:

Το αρχείο insert.sql, κάνει insert στη βάση dummy data για όλα τα tables της βάσης.

Ας δούμε αναλυτικότερα τα queries που μας ζητούνται από την εκφώνηση:

Για το 3.1:

```
/* Query 1 */

/* Όλα τα προγράμματα που είναι διαθέσιμα */

SELECT * from program;

/* όλα τα έργα/επιχορηγήσεις που έχουν καταχωριστεί με βάση πολλαπλά κριτήρια */

SELECT * from project;

SELECT p.title from project p WHERE start = '2020-09-22';

/* date */

SELECT p.title from project p WHERE end = '2024-08-27';

/* date */

SELECT p.title, p.description from project p INNER JOIN executive e ON e.ex_id = p.ex_id
WHERE e.name = 'Brookes';

/* show all researchers working on a specific project */

SELECT r.first_name, r.last_name FROM researcher r
INNER JOIN works w
ON w.res_id = r.res_id
INNER JOIN project p ON w.proj_id = p.proj_id
WHERE p.title = 'Proj14';
```

Για το 3.2:

```
SELECT * FROM projects_per_institution_per_year ORDER BY ins_id, year;
SELECT * FROM projects_per_researcher ORDER BY res_id;
```

Η υλοποίηση αυτών των views υπάρχει πιο πάνω.

Για το 3.3:

```
SELECT p.proj_id, p.title, p.description, p.start, p.end, p.fund, p.ins_id, p.prog_id,
p.ex_id, p.ass_id, p.ass_res_id, p.sup_res_id
FROM research_field rf
INNER JOIN proj_field pf
ON rf.field_id = pf.field_id
INNER JOIN project p
ON pf.proj_id = p.proj_id
WHERE rf.field_id = 99012 AND p.start <= current_date() AND p.end >= current_date();
/* User picked field_id on app*/
```

Αυτό το select μας επιστρέφει τα IDs, τους τίτλους, τις περιλήψεις, την ημερομηνία έναρξης, την ημερομηνία ολοκλήρωσης και το ποσό χρηματοδότησης των ενεργών έργων που ασχολούνται με το research field που έχει ID 99012 (η επιλογή του ID έγινε τυχαία, κάλλιστα θα μπορούσαμε να βάλουμε οποιοδήποτε επιστημονικό πεδίο υπάρχει στη βάση. Στην εφαρμογή μας, ο χρήστης μπορεί να επιλέξει όποιο επιστημονικό πεδίο επιθυμεί).

Για να το βρούμε αυτό κάνουμε δύο inner joins που μας πάνε από το table research_field στο table proj_field και από εκεί στο project.

```

SELECT DISTINCT r.res_id, r.first_name, r.last_name, r.sex, r.date_of_birth, r.ins_id,
r.res_ins_date
FROM research_field rf
INNER JOIN proj_field pf
ON rf.field_id = pf.field_id
INNER JOIN project p
ON pf.proj_id = p.proj_id
INNER JOIN works w
ON p.proj_id = w.proj_id
INNER JOIN researcher r
ON w.res_id = r.res_id
WHERE rf.field_id = 99012 AND (p.start <= current_date() AND p.end >=
DATE_SUB(current_date(), INTERVAL 1 YEAR)); /* User picked field_id on app*/

```

Αυτό το query μας επιστρέφει τους ερευνητές που ασχολήθηκαν με το αντίστοιχο επιστημονικό πεδίο τον τελευταίο χρόνο. Επειδή δεν ήμασταν 100% σίγουροι τι ακριβώς ζητούσε η εκφώνηση συμπεριλάβαμε όχι μόνο τους ερευνητές που ασχολούνται τώρα(δηλ ερευνητές σε ενεργά έργα μόνο) αλλά και εκείνους που ασχολήθηκαν κάποια στιγμή τις προηγούμενες 365 μέρες με αυτό το επιστημονικό πεδίο(δηλ το έργο στο οποίο εργαζόντουσαν έληξε μέσα στον προηγούμενο χρόνο). Άμα θέλουμε μόνο ενεργά έργα, αρκεί να γράψουμε p.end >= current_date().

Για το 3.4:

```

SELECT i.ins_id, i.abbreviation, i.name, i.year AS first_year, j.year AS second_year,
i.projects AS projects_each_year
FROM projects_per_institution_per_year i
INNER JOIN projects_per_institution_per_year j
ON i.ins_id = j.ins_id
WHERE i.year = j.year - 1 AND i.projects = j.projects AND i.projects >= 10;

```

Όπως αναφέραμε και πιο πάνω, για την επίλυση αυτού του query χρησιμοποιήσαμε ένα view που εμπεριέχει πληροφορία για τα projects κάθε οργανισμού ανά χρόνο. Με ένα join αυτού του view με τον εαυτό του, καταφέραμε να παράξουμε την πληροφορία που μας ζητείται από την εκφώνηση και αφού ψάχνουμε δύο συνεχόμενα έτη θα πρέπει το έτος του ενός view να είναι το έτος του άλλου view -1. Και αφού θέλουμε να ψάξουμε για ποιον οργανισμό ισχύει, πρέπει τα IDs των οργανισμών στα οποία αναφέρονται τα views να είναι ίδια.

Για το 3.5 :

```
WITH pf (field_name, field_id, proj_id) AS
(SELECT rf.field_name, rf.field_id, pf.proj_id
FROM research_field rf
INNER JOIN proj_field pf
ON rf.field_id = pf.field_id)
SELECT pf1.field_name AS field1, pf2.field_name AS field2, count(*) AS popularity
FROM pf pf1
INNER JOIN pf pf2
ON pf1.proj_id = pf2.proj_id
WHERE pf1.field_id < pf2.field_id
GROUP BY pf1.field_name, pf2.field_name
ORDER BY popularity DESC
LIMIT 3;
```

Ορίζουμε μια προσωρινή σχέση με την έκφραση WITH που είναι το αποτέλεσμα του INNER JOIN του επιστημονικού πεδίου research_field με τις τούπλες της σχέσης proj_field που παριστάνουν τα επιστημονικά πεδία κάθε έργου με δεδομένο proj_id. Με INNER JOIN της συγκεκριμένης σχέσης με τον εαυτό της και κατάλληλη ομαδοποίηση καταλήγουμε στις πιο "δημοφιλείς" τούπλες επιστημονικών πεδίων μεταξύ των έργων στη βάση μας.

Για το 3.6:

```
SELECT r.res_id, r.first_name, r.last_name, floor(DATEDIFF(current_date(),  
r.date_of_birth) / 365.25) AS age, count(*) AS total_projects  
FROM researcher r  
INNER JOIN works w  
ON r.res_id = w.res_id  
INNER JOIN project p  
ON w.proj_id = p.proj_id  
WHERE floor(DATEDIFF(current_date(), r.date_of_birth) / 365.25) <= 40 AND (p.start <=  
current_date() AND p.end >= current_date())  
GROUP BY r.res_id  
ORDER BY count(*) DESC  
LIMIT 10;
```

Για να βρούμε τους ερευνητές με ηλικία κάτω των 40 που εργάζονται στα περισσότερα έργα και τον αριθμό των έργων αρκεί από τη σχέση researcher να πάμε μέσω ενός inner join στη σχέση works, από εκεί με ένα ακόμα inner join πάμε στη σχέση project. Στο where ελέγχουμε την ηλικία του ερευνητή και βεβαιωνόμαστε ότι το έργο είναι ενεργό.

Πρέπει να κάνουμε group by res_id ώστε να ομαδοποιήσουμε τις tuples για τον κάθε ερευνητή και τέλος κάνουμε φθίνουσα εκτύπωση με βάση το count(*) που είναι ο αριθμός των projects κάθε ερευνητή πριν γίνει group by.

Η εκφώνηση δεν διευκρινίζει πόσες tuples να τυπώσουμε οπότε τυπώσαμε τις πρώτες δέκα.

Για το 3.7:

```
SELECT e.name AS executive_name, i.name AS company_name, sum(p.fund) AS  
total_fund  
FROM executive e  
INNER JOIN project p  
ON e.ex_id = p.ex_id  
INNER JOIN institution i  
ON p.ins_id = i.ins_id  
INNER JOIN company c  
ON i.ins_id = c.ins_id  
GROUP BY e.ex_id, c.ins_id  
ORDER BY sum(p.fund) DESC  
LIMIT 5;
```

Η εκφώνηση μας λέει συγκεκριμένα εταιρείες οπότε ξεκινώντας από το executive θα πάμε στο project που διαχειρίζεται το στέλεχος, από εκεί στον οργανισμό που διαχειρίζεται τα projects και από εκεί στις εταιρείες. Κάνουμε group by με ex_id αλλά και c.ins_id καθώς θέλουμε να βρούμε το ποσό που έχει δώσει το στέλεχος σε κάθε εταιρεία, θα αθροίσουμε με τη βοηθητική συνάρτηση sum όλα τα ποσά χρηματοδότησης και θα τυπώσουμε τα top 5 χρησιμοποιώντας DESC και LIMIT 5.

Τέλος, για το 3.8:

```
SELECT r.res_id, r.first_name, r.last_name, count(*) AS num_of_projects
FROM researcher r
INNER JOIN works w
ON r.res_id = w.res_id
INNER JOIN
(SELECT * FROM project
 WHERE proj_id not in (SELECT proj_id FROM deliverable)) p
ON w.proj_id = p.proj_id
WHERE p.start <= current_date() AND p.end >= current_date() GROUP BY
r.res_id HAVING count(*) >= 5 ORDER BY num_of_projects DESC;
```

Για να βρούμε τα ενεργά έργα χωρίς παραδοτέα χρησιμοποιήσαμε ένα εσωτερικό select στο query μας ώστε να βρούμε τα proj_id που δεν υπάρχουν στη σχέση deliverable.

Βρίσκουμε ποιοι ερευνητές δουλεύουν σε αυτά τα έργα κάνουμε group by με το ID αυτών των ερευνητών και κρατάμε, μέσω του HAVING, τους ερευνητές που δουλεύουν σε τουλάχιστον 5 τέτοια έργα. Υστερα τυπώνουμε το αποτέλεσμα κατά φθίνουσα σειρά.

Οδηγίες Εγκατάστασης της Εφαρμογής (για Ubuntu):

Έχουμε θεωρήσει ότι το πακέτο LAMP είναι επιτυχώς εγκατεστημένο:

(<https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-20-04>)

Βήμα 1: Άνοιγμα του terminal και εκτέλεση της εντολής git init σε ένα άδειο directory:

```
petrosrapto@petrosraptoAssistant:~/Desktop/Installation$ git init
Initialized empty Git repository in /home/petrosrapto/Desktop/Installation/.git/
```

Βήμα 2: Εκτέλεση της εντολής git clone <https://github.com/petrosrapto/DataBase.git>

```
petrosrapto@petrosraptoAssistant:~/Desktop/Installation$ git clone https://github.com/petrosrapto/DataBase.git
Cloning into 'DataBase'...
remote: Enumerating objects: 396, done.
remote: Counting objects: 100% (102/102), done.
remote: Compressing objects: 100% (66/66), done.
remote: Total 396 (delta 49), reused 59 (delta 35), pack-reused 294
Receiving objects: 100% (396/396), 1.57 MiB | 662.00 KiB/s, done.
Resolving deltas: 100% (197/197), done.
```

Βήμα 3: Μετάβαση στο directory DataBase/App και εκτέλεση της εντολής sudo pip install -r requirements.txt

(Θα χρειαστεί να συμπληρωθούν οι κωδικοί του root χρήστη)

```
petrosrapto@petrosraptoAssistant:~/Desktop/Installation$ cd DataBase/App/
petrosrapto@petrosraptoAssistant:~/Desktop/Installation/DataBase/App$ sudo pip install
-r requirements.txt
[sudo] password for petrosrapto:
Requirement already satisfied: click==8.1.2 in /usr/local/lib/python3.8/dist-packages
(from -r requirements.txt (line 1)) (8.1.2)
Requirement already satisfied: dnspython==2.2.1 in /usr/local/lib/python3.8/dist-packages
(from -r requirements.txt (line 2)) (2.2.1)
Requirement already satisfied: email-validator==1.1.3 in /usr/local/lib/python3.8/dist-
packages (from -r requirements.txt (line 3)) (1.1.3)
Requirement already satisfied: Faker==13.3.4 in /usr/local/lib/python3.8/dist-packages
(from -r requirements.txt (line 4)) (13.3.4)
Requirement already satisfied: Flask==2.1.1 in /usr/local/lib/python3.8/dist-packages
(from -r requirements.txt (line 5)) (2.1.1)
Requirement already satisfied: Flask-MQLdb==1.0.1 in /usr/local/lib/python3.8/dist-p
ackages (from -r requirements.txt (line 6)) (1.0.1)
Requirement already satisfied: Flask-WTF==1.0.1 in /usr/local/lib/python3.8/dist-p
ackages (from -r requirements.txt (line 7)) (1.0.1)
Requirement already satisfied: idna==3.3 in /usr/local/lib/python3.8/dist-packages (fr
om -r requirements.txt (line 8)) (3.3)
Requirement already satisfied: importlib-metadata==4.11.3 in /usr/local/lib/python3.8/
dist-packages (from -r requirements.txt (line 9)) (4.11.3)
Requirement already satisfied: itsdangerous==2.1.2 in /usr/local/lib/python3.8/dist-p
ackages (from -r requirements.txt (line 10)) (2.1.2)
Requirement already satisfied: Jinja2==3.1.1 in /usr/local/lib/python3.8/dist-packages
(from -r requirements.txt (line 11)) (3.1.1)
Requirement already satisfied: MarkupSafe==2.1.1 in /usr/local/lib/python3.8/dist-p
ackages (from -r requirements.txt (line 12)) (2.1.1)
Requirement already satisfied: mysqlclient==2.1.0 in /usr/local/lib/python3.8/dist-p
ackages (from -r requirements.txt (line 13)) (2.1.0)
Requirement already satisfied: python-dateutil==2.8.2 in /usr/local/lib/python3.8/dist-
packages (from -r requirements.txt (line 14)) (2.8.2)
Requirement already satisfied: six==1.16.0 in /usr/local/lib/python3.8/dist-packages (
from -r requirements.txt (line 15)) (1.16.0)
Requirement already satisfied: Werkzeug==2.1.1 in /usr/local/lib/python3.8/dist-p
ackages (from -r requirements.txt (line 16)) (2.1.1)
Requirement already satisfied: WTForms==3.0.1 in /usr/local/lib/python3.8/dist-package
s (from -r requirements.txt (line 17)) (3.0.1)
Requirement already satisfied: zipp==3.8.0 in /usr/local/lib/python3.8/dist-packages (
from -r requirements.txt (line 18)) (3.8.0)
```

Σημείωση: Η παραπάνω εικόνα απεικονίζει το output της εντολής sudo pip install -r requirements.txt σε περίπτωση που έχουν εγκατασταθεί επιτυχώς οι απαραίτητες από την εφαρμογή βιβλιοθήκες.

Βήμα 4: Εκτέλεση των παρακάτω εντολών και αντιγραφή του αποτελέσματος (absolute path του αρχείου):

```
petrosrapto@petrosraptoAssistant:~/Desktop/Installation/DataBase/App$ cd ..
petrosrapto@petrosraptoAssistant:~/Desktop/Installation/DataBase$ pwd
/home/petrosrapto/Desktop/Installation/DataBase
```

Βήμα 5: Άνοιγμα της MySQL μέσω των εντολών sudo mysql -u root -p

(Θα χρειαστεί να συμπληρωθούν οι κωδικοί του root χρήστη)

```
petrosrapto@petrosraptoAssistant:~/Desktop/Installation/DataBase$ sudo mysql -u root -p
[sudo] password for petrosrapto:
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1876
Server version: 8.0.29-Ubuntu0.20.04.3 (Ubuntu)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> □
```

Βήμα 6: Γράφουμε source και επικολλούμε το absolute path που αντιγράψαμε στο βήμα 4.

Προσθέτουμε στο τέλος /create.sql και πατάμε Enter.

```
mysql> source /home/petrosrapto/Desktop/Installation/DataBase/create.sql
Query OK, 0 rows affected (0.00 sec)

Query OK, 16 rows affected (0.11 sec)

Query OK, 1 row affected (0.00 sec)

Database changed
Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected, 1 warning (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected, 1 warning (0.00 sec)
```

Σημείωση: Αγνοούμε τα warnings που εμφανίζονται (έχουν ληφθεί υπόψιν).

Βήμα 7: Γράφουμε source και επικολλούμε το absolute path που αντιγράψαμε στο βήμα 4.

Προσθέτουμε στο τέλος /insert.sql και πατάμε Enter.

```
mysql> source /home/petrosrapto/Desktop/Installation/DataBase/insert.sql
Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.01 sec)

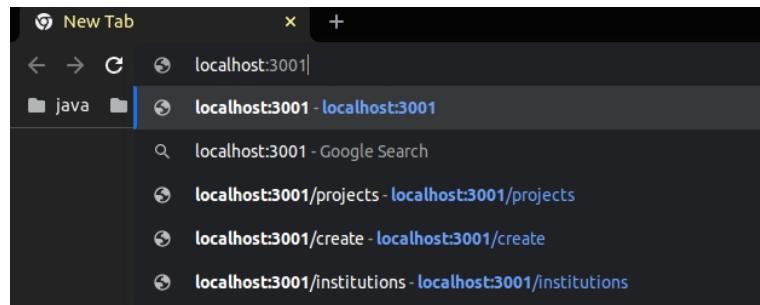
Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.01 sec)
```

Βήμα 8: Γράφουμε exit για να κλείσουμε την MySQL και εκτελούμε διαδοχικά τις παρακάτω εντολές ώστε να βάλουμε σε λειτουργία των σέρβερ (τοπικά):

```
mysql> exit
Bye
petrosrapto@petrosraptoAssistant:~/Desktop/Installation/DataBase$ cd App/
petrosrapto@petrosraptoAssistant:~/Desktop/Installation/DataBase/App$ python3 run.py
 * Serving Flask app 'module' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://localhost:3001 (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 144-605-012
```

Βήμα 9: Ανοίγουμε τον browser και γράφουμε localhost:3001



Βήμα 10: Πατάμε Enter και είμαστε έτοιμοι! Μπορούμε τώρα να εξερευνήσουμε την εφαρμογή:

| No | Field Name | Field Name | Projects |
|----|--------------|-------------|----------|
| 1 | Archaeology | Chemistry | 6 |
| 2 | Architecture | Engineering | 4 |
| 3 | Archaeology | Economics | 3 |

| No | Executive's Name | Funded Company | Total Fund Amount |
|----|------------------|----------------|-------------------|
| 1 | Duffy | LabMouse | 750000.00 |
| 2 | Horner | Faculty Labs | 540000.00 |
| 3 | Vang | Scipark | 500000.00 |
| 4 | Riddle | Config Lab | 450000.00 |
| 5 | Coates | Linkage Labs | 410000.00 |

| No | First Name | Last Name | Age | Projects |
|----|------------|-----------|-----|----------|
| 1 | Sheena | Hamer | 38 | 20 |

localhost:3001/create +

Hellenic Foundation for Research & Innovation - Create Page

[Home](#)

Create Program

| | |
|--|--|
| Name | <input style="width: 85%;" type="text"/> |
| Department | <input style="width: 85%;" type="text"/> |
| <input style="background-color: #2e6b2e; color: white; border: none; padding: 5px 10px; border-radius: 5px; font-weight: bold;" type="button" value="Create"/> | |

Create Research Field

| | |
|--|--|
| Name | <input style="width: 85%;" type="text"/> |
| Description | <input style="width: 85%;" type="text"/> |
| <input style="background-color: #2e6b2e; color: white; border: none; padding: 5px 10px; border-radius: 5px; font-weight: bold;" type="button" value="Create"/> | |

Create Institution

localhost:3001/fields +

Hellenic Foundation for Research & Innovation - Research Fields Page

[Home](#)

| ID | Name | Description | |
|-------|---------------------|--|--|
| 99001 | Archaeology | Archaeology is the scientific study of human activity through the recovery and analysis of material culture. | |
| 99002 | Economics | Economics is the social science that studies the production, distribution, and consumption of goods and services. | |
| 99003 | Biology | Biology is the scientific study of life. | |
| 99004 | Chemistry | Chemistry is the scientific study of the properties and behavior of matter. | |
| 99005 | Earth sciences | Earth science or geoscience includes all fields of natural science related to the planet Earth | |
| 99006 | Physics | Physics is the natural science that studies matter, its fundamental constituents, its motion and behavior through space and time, and the related entities of energy and force. | |
| 99007 | Astronomy | Astronomy is a natural science that studies celestial objects and phenomena. | |
| 99008 | Pure mathematics | Pure mathematics is the study of mathematical concepts independently of any application outside mathematics. | |
| 99009 | Applied mathematics | Applied mathematics is the application of mathematical methods by different fields such as physics, engineering, medicine, biology, finance, business, computer science, and industry. | |
| 99010 | Business | Business is the activity of making one's living or making money by producing or buying and selling products (such as goods and services). | |
| 99011 | Architecture | Architecture is both the process and the product of planning, designing, and constructing buildings or other structures. | |
| 99012 | Engineering | Engineering is the use of scientific principles to design and build machines, structures, and other items, including bridges, tunnels, roads, vehicles, and buildings. | |

Σημείωση: Κατά την εκτέλεση του βήματος 2 θα χρειαστεί ταυτοποίηση του λογαριασμού github. Σε περίπτωση δυσκολίας ακολουθείστε τα βήματα του συνδέσμου: <https://docs.github.com/en/authentication>

2.4

Σύνδεσμος για το git repo της εφαρμογής:

<https://github.com/petrosrapto/DataBase.git>