Βάσεις Δεδομένων
Αναφορά Εξαμηνιαίας Εργασίας

Ομάδα 44
Μαλεκκίδης Νικόλαος, ge18708
Ματθαίου Πέτρος, ge18710

Εισαγωγή.

Η παρούσα αναφορά αποτελεί έναν οδηγό για καλύτερη κατανόηση των όσων συμπεριλήφθηκαν στα υπόλοιπα αρχεία, ώστε η εργασία στο σύνολο της να γίνει ευανάγνωστη και κατανοητή. Μέσω αυτού θα επιχειρήσουμε να σας εντάξουμε στον δικό μας τρόπο προσέγγισης του προβλήματος, θα διευκρινίσουμε όλες τις παραδοχές που έχουμε κάνει, όλους τους περιορισμούς τους οποίους έχουμε θέσει, οι οποίοι άλλοτε πήγαζαν από την εκφώνηση του προβλήματος και άλλοτε μας φάνηκε απαραίτητο να επέμβουμε χειροκίνητα και να επιβάλουμε κάποιους περιορισμούς στα δεδομένα ώστε να γίνει πιο ξεκάθαρος ο τρόπος λειτουργίας της βάσης.

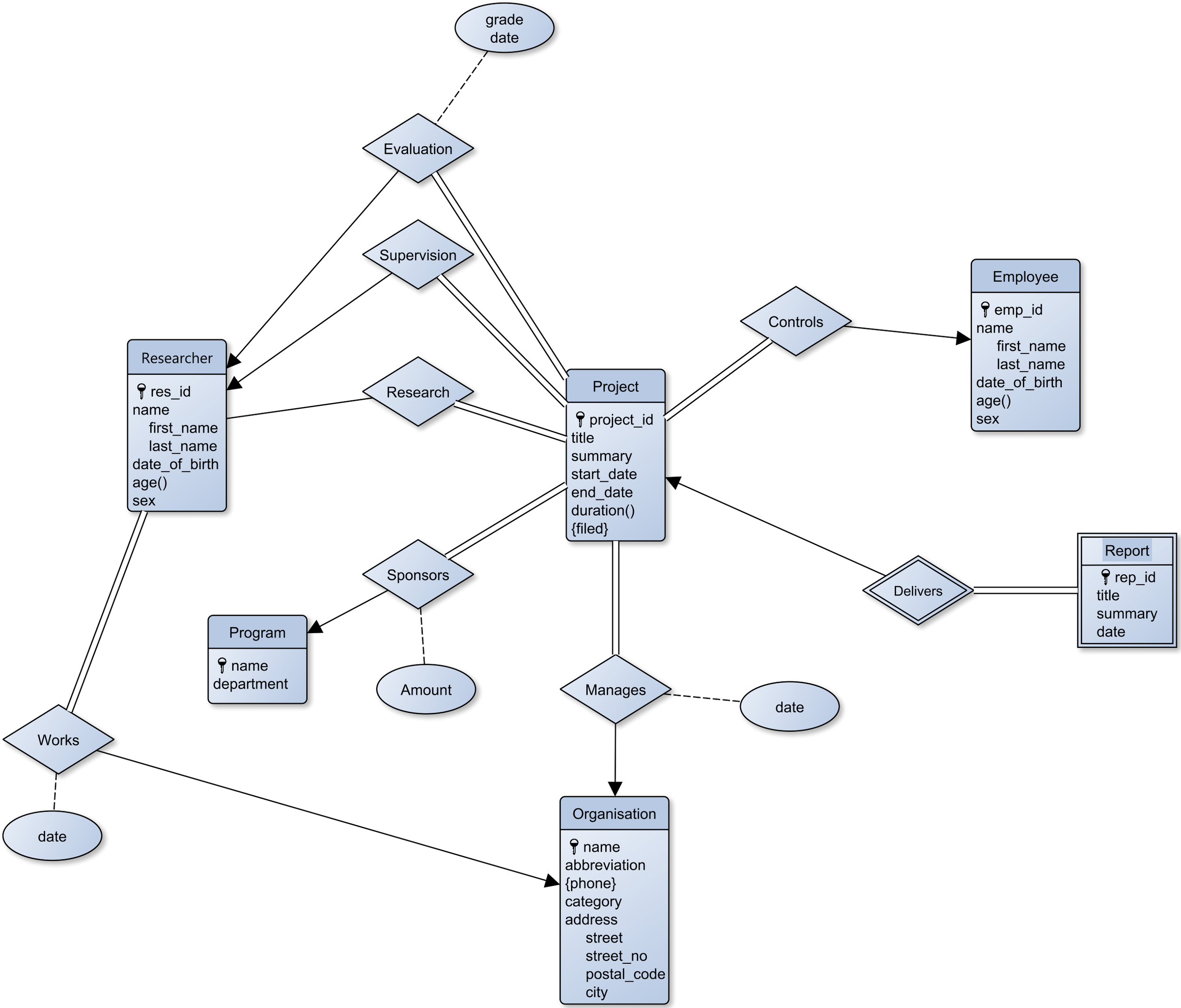
Η αναφορά χωρίζεται σε δύο μέρη. Στο πρώτο μέρος θα αναφερθούμε στο κομμάτι της προσέγγισης, σχεδίασης και υλοποίησης. Συγκεκριμένα θα αναφερθούμε στα σημεία που διαφοροποιηθήκαμε σε σχέση με την προτεινόμενη λύση για το διάγραμμα οντοτήτων συσχετίσεων, την μετατροπή αυτού σε σχεσιακό μοντέλο καθώς στον σχεδιασμό της βάσης και τις παραδοχές και τους περιορισμούς που θεωρήσαμε απαραίτητο να συμπεριληφθούν ώστε να υπάρχει λογική αλληλουχία των δεδομένων που εισάγονται σε αυτήν. Στο δεύτερο μέρος της εργασίας θα αναφερθούμε στον τρόπο με τον οποίο παρήχθησαν τα δεδόμενα τα οποία εισηγάγαμε στην βάση καθώς επίσης το πώς διαμορφώσαμε τα ερωτήματα (*queries*) που συντάξαμε ώστε να ανταποκρίνονται στα ζητούμενα της άσκησης.

Εργαλεία.

Αρχικά να αναφερθούμε στα εργαλεία που έχουμε επιλέξει να χρησιμοποιήσουμε για την συγκεκριμένη εργασία. Να σημειώσουμε ότι όλα τα παρακάτω έγιναν σε λειτουργικό σύστημα *Windows 10/11*. Η σχεδίαση του διαγράμματος οντοτήτων-συσχετίσεων (*ER diagram*) έχει γίνει στην εφαρμογή *yEd Graph Editor*, ενώ η σχεδίαση του σχεσιακού μοντέλου (*Relational Model*) στον ιστότοπο *draw.io*. Όσον αφορά τη σχεδίαση της βάσης στην *SQL* επιλέξαμε να χρησιμοποιήσουμε το *XAMPP* και να εργαστούμε στο περιβάλλον του *MySQL Workbench* και του *DBeaver*. Χρησιμοποίησαμε *PHP* για το *server side* της εφαρμογής και *HTML* για το *client side* της εφαρμογής. Τέλος, η παραγωγή των ψευδο-δεδομένων έγινε με κατάλληλο κώδικα στη *Python*.

Διάγραμμα *ER*

Το πρώτο μέρος αναφέρεται αρχικά στο διάγραμμα οντοτήτων-συσχετίσεων, η δημιουργία του οποίου ήταν και το πρώτο μας μέλημα. Στο διάγραμμα οντοτήτων συσχετίσεων φαίνονται όλες οι διαφορετικές οντότητες πχ. Έργο, Οργανισμός με τα δικά τους χαρακτηριστικά η κάθε μία, και οι συσχετίσεις μέσω των οποίων συνδέονται. Θα παρατηρήσετε ότι το δικό μας επισυναπτόμενο διάγραμμα διαφέρει με την προτεινόμενη λύση σε σημεία. Δίνουμε το διάγραμμα οντοτήτων-συσχετίσεων μας στην ερχόμενη σελίδα.


grade
date
Evaluation
Supervision
Research
Sponsors
Controls
Delivers
Manages
Works
Researcher
res_id
name
first_name
last_name
date_of_birth
age()
sex
Project
project_id
title
summary
start_date
end_date
duration()
{filed}
Employee
emp_id
name
first_name
last_name
date_of_birth
age()
sex
Report
rep_id
title
summary
date
Program
name
department
Amount
date
date
Organisation
name
abbreviation
{phone}
category
address
street
street_no
postal_code
city

Σημεία διαφοροποίησης με την προτεινόμενη λύση:

- Κατ' αρχάς θεωρήσαμε το επιστημονικό πεδίο (*field*) του έργου (*project*) ως ένα *attribute*, πολλαπλών τιμών, του ίδιου του έργου και όχι ως ξεχωριστή οντότητα. Εφόσον το επιστημονικό πεδίο ενός έργου συνδέεται αποκλειστικά με το έργο κρίναμε πως δεν υπήρχε λόγος να θεωρηθεί ως ξεχωριστή οντότητα.
- Για τον ίδιο ακριβώς λόγο θεωρήσαμε και την κατηγορία (*category*) του οργανισμού (*organisation*) ως ένα χαρακτηριστικό του ίδιου του οργανισμού, και όχι ως ξεχωριστή οντότητα με την οποία να συνδέεται μέσω μιας άλλης συσχέτισης.
- Να αναφέρουμε ότι ο προϋπολογισμός της κάθε κατηγορίας δεν συνδεόταν με κανέναν τρόπο στους σκοπούς λειτουργίας της βάσης μας και επομένως θεωρήσαμε περιττό να τον συμπεριλάβουμε στην λύση μας με οποιονδήποτε τρόπο.

Σχεσιακό Μοντέλο

Ακολούθως, μετατρέψαμε το διάγραμμα οντοτήτων συσχετίσεων σε σχεσιακό μοντέλο. Στο σχεσιακό μοντέλο φαίνονται αναλυτικά οι πίνακες που τελικά θα δημιουργηθούν στη βάση δεδομένων, και τα χαρακτηριστικά (στήλες) της κάθε εγγραφής στον αντίστοιχο πίνακα. Δίνουμε στην ερχόμενη σελίδα το σχεσιακό μας μοντέλο.

Organisation	
PK	id
	org_name
	abbreviation
	category
	street
	street_no
	postal_code
	city

Field	
PK, FK	project_id
PK	field_name

Phone	
PK, FK	org_id
PK	phone

Researcher	
PK	id
FK	org_id
	first_name
	last_name
	date_of_birth
	sex
	works_date

Project	
PK	id
FK	evaluator_id
FK	supervisor_id
FK	controller_id
FK	sponsor_id
FK	managing_org_id
	title
	summary
	start_date
	end_date
	evaluation_grade
	evaluation_date
	sponsor_amount
	managing_date

Employee	
PK	id
	first name
	last name
	date of birth
	sex

Report	
PK	id
PK,FK	project_id
	title
	summary
	delivery_date

Program	
PK	id
	name department

Works_on	
PK, FK	project_id
PK,FK	researcher_id

Τα ονόματα των πινάκων και των στηλών στο σχεσιακό μοντέλο είναι κοινά ανάμεσα στο σχεσιακό μοντέλο και στη βάση δεδομένων αργότερα. Παρατηρούμε ότι τα χαρακτηριστικά (*attributes*) που δύναται να λάβουν περίσσοτερες από μία τιμές (όπως πχ το *field* στην οντότητα *project* και το *phone* στο *organisation*) μετατρέπονται πλέον σε ξεχωριστούς πίνακες στο σχεσιακό μοντέλο.

Πολύ σημαντική πληροφορία που λαμβάνεται από το σχεσιακό μοντέλο είναι τα πρωτεύοντα (*primary*) κλειδιά του κάθε πίνακα και τα εξωτερικά (*foreign*) κλειδιά, όπου υπάρχουν. Η σχηματική αναπαράσταση βοηθά ακριβώς στο να ξεκαθαρίσουμε ποια στοιχεία των πινάκων συνδέονται με την σχέση πρωτεύοντος και εξωτερικού κλειδιού.

Ακόμη, τα χαρακτηριστικά της συσχέτισης που συνδέει τις δύο οντότητες (π.χ. *one to many*, ολική συμμετοχή κλπ) καθορίζουν πλήρως τον τρόπο με τον οποίο θα συνδεθούν οι δύο αντίστοιχοι πίνακες στο σχεσιακό μοντέλο. Για παράδειγμα, η συσχέτιση *Research* (στο *ER*) ανάμεσα στο *project* και τον *researcher* είναι *many to many* με ολική συμμετοχή στα *projects*, το οποίο μεταφράζεται σε ανάγκη δημιουργίας νέου πίνακα (ο πίνακας *Works_on* στο σχεσιακό) για την συγκεκριμένη συσχέτιση, ο οποίος θα περιλαμβάνει, ως στοιχεία, τα κλειδιά των δύο ξεχωριστών πινάκων.

Υλοποίηση της Βάσης

Όταν πλέον ολοκληρώσαμε το σχεσιακό μοντέλο, προχωρήσαμε πλέον στην υλοποίηση της βάσης. Η βάση που δημιουργήσαμε αποτελείται ουσιαστικά από τους πίνακες και τα χαρακτηριστικά τους όπως ακριβώς υποδεικνύει το σχεσιακό μοντέλο.

Δημιουργήσαμε αρχικά τον πίνακα *organisation*, ο οποίος προφανώς αντιστοιχεί στην οντότητα του οργανισμού. Δίνουμε πιο κάτω το συντακτικό δημιουργίας του πίνακα στην *sql*:

```sql
CREATE OR REPLACE TABLE organisation(
  id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  org_name VARCHAR(100) NOT NULL,
  abbreviation VARCHAR(15) UNIQUE NOT NULL,
  category ENUM('University', 'Research Center', 'Company') NOT NULL,
  street VARCHAR(50) NOT NULL,
  street_no BIGINT UNSIGNED NOT NULL,
  postal_code BIGINT UNSIGNED NOT NULL,
  city VARCHAR(50) NOT NULL
);
```

Σημειώνουμε ότι:

- Ορίζουμε αυτόματη δεικτοδότηση ως το πρωτεύον κλειδί των οργανισμών, βασιζόμενοι στο οτι το όνομα ενός οργανισμού ενδέχεται να αλλάξει, το οποίο δεν είναι επιθυμητό (για πρωτεύων κλειδιά).
- Θεωρούμε πως η συντομογραφία του ονόματος ενός οργανισμού πρέπει να είναι μοναδική.
- Προσθέτουμε την κατηγορία του οργανισμού ως χαρακτηριστικό στον πίνακα, του οποίου το πεδίο τιμών είναι τα τρία είδη οργανισμών: *University*, *Research Center* και *Company* που αναφέρονται στην εκφώνηση.
- Το σύνθετο *attribute* της διεύθυνσης του οργανισμού διασπάται στα επιμέρους κομμάτια της: *street* (οδός), *street_no* (αριθμός), *postal_code* (ταχυδρομικός κώδικας) και *city* (πόλη).

Τα τηλέφωνα επικοινωνίας του οργανισμού, ως *attribute* πολλαπλών τιμών, ορίζει απο μόνο του ενα καινούργιο πίνακα, *phone*. Δίνουμε πιο κάτω το αντίστοιχο μέρος του κώδικα:

```sql
CREATE OR REPLACE TABLE phone(
  phone BIGINT UNSIGNED NOT NULL,
  org_id INT UNSIGNED NOT NULL,
    FOREIGN KEY(org_id) REFERENCES organisation(id) ON UPDATE CASCADE,
  PRIMARY KEY(phone, org_id)
);
```

Σημειώνουμε ότι:

- Θεωρούμε πως ο αριθμός τηλεφώνου, *phone*, δίνεται ως "καθαρός" αριθμός, δηλαδή σύμβολα όπως: *+,),(* κτλ. Εναλλακτικά θα μπορούσαμε να τον ορίσουμε ως *string* μεταβλητού μεγέθους, *varchar*, για να καλύψουμε και αυτό το ενδεχόμενο.
- Προφανώς ο δείκτης του αντίστοιχου οργανισμού υπάρχει ως εξωτερικό κλειδί της εγγραφής. Η γενική πολιτική που ακολουθήσαμε στο παρόν *project* είναι να επιτρέπουμε μόνο ανανέωση του εξωτερικού κλειδιού σε περίπτωση αλλαγής του (*on update cascade*) και οχι διαγραφή (αφήσαμε το *default*, *on delete restrict*).

Έπειτα δημιουργούμε τον πίνακα για τα στελέχοι του ΕΛΙΔΕΚ, *employee*. Δίνουμε πιο κάτω το αντίστοιχο μέρος του κώδικα:

```sql
CREATE OR REPLACE TABLE employee(
  id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  first_name VARCHAR(30) NOT NULL,
  last_name VARCHAR(30) NOT NULL,
  date_of_birth DATE NOT NULL,
  sex ENUM('M', 'F')
);
```

Σημειώνουμε ότι:

- Συμβολίζουμε το φύλο του στελέχους με M για αρσενικό και F για θηλυκό.
- Για όμοιους λόγους με πιο πάνω, χρησιμοποιήσαμε αυτόματη δεικτοδότηση ως πρωτεύον κλειδί των εγγραφών.

Δίνουμε τώρα το μέρος του κώδικα που αντιστοιχεί στον πίνακα των διαθέσιμων προγραμμάτων του ΕΛΙΔΕΚ, *program*:

```sql
CREATE OR REPLACE TABLE program(
  id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50) NOT NULL,
  department ENUM('ΚΑΙΝΟΤΟΜΙΑ', 'ΕΠΕΝΔΥΣΕΙΣ', 'ΑΓΟΡΑ', 'ΑΝΑΚΑΜΨΗ',
                  'ΑΝΘΡΩΠΟΣ', 'ΠΕΡΙΒΑΛΛΟΝ') NOT NULL
);
```

Σημειώνουμε ότι:

- Χρησιμοποιήσαμε αυτόματη δεικτοδότηση ως πρωτεύον κλειδί των εγγραφών.
- Έστω οτι οι διευθύνσεις του ΕΛΙΔΕΚ είναι οι εξής: *Καινοτομία*, *Επενδύσεις*, *Αγορά*, *Ανάκαμψη*, *Άνθρωπος* και *Περιβάλλον*.

Δίνουμε πιο κάτω τον κώδικα σχετικά με τη δημιουργία του πίκανα των ερευνητών, *researcher*:

```
CREATE OR REPLACE TABLE researcher(
  id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  org_id INT UNSIGNED NOT NULL,
    FOREIGN KEY(org_id) REFERENCES organisation(id) ON UPDATE CASCADE,
  first_name VARCHAR(30) NOT NULL,
  last_name VARCHAR(30) NOT NULL,
  date_of_birth DATE NOT NULL,
  sex ENUM('M', 'F'),
  works_date DATE NOT NULL CHECK( date_of_birth < works_date )
);
```

Σημειώνουμε ότι:

- Χρησιμοποιήσαμε αυτόματη δεικτοδότηση ως πρωτεύον κλειδί των εγγραφών.
- Η ολική συμμετοχή στη συσχετίση *Works* (του *ER*) εξασφαλίζεται προσθέτοντας το αντίστοιχο *id* του οργανισμού (*org_id*) σε κάθε εγγραφή.
- Θεωρούμε επίσης την προφανής συνθήκη οτι η ημερομηνία πρόσληψη του ερευνητή στον οργανισμό, *works_date*, πρέπει να είναι μετά την ημερομηνία γέννησής του.

Δίνουμε πιο κάτω τον κώδικα σχετικά με τη δημιουργία του πίκανα των έργων, *project*:

```
CREATE OR REPLACE TABLE project(
  id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  title VARCHAR(100) NOT NULL,
  summary MEDIUMBLOB NOT NULL,
  start_date DATE NOT NULL,
  end_date DATE NOT NULL CHECK( DATEDIFF(end_date,start_date)<=4*365
                          AND DATEDIFF(end_date,start_date)>=365 ),
  evaluator_id INT UNSIGNED NOT NULL,
    FOREIGN KEY(evaluator_id) REFERENCES researcher(id) ON UPDATE CASCADE,
  evaluator_grade INT NOT NULL CHECK( evaluator_grade BETWEEN 0 AND 100 ),
  evaluation_date DATE NOT NULL CHECK( evaluation_date<start_date ),
  supervisor_id INT UNSIGNED NOT NULL,
    FOREIGN KEY(supervisor_id) REFERENCES researcher(id) ON UPDATE CASCADE,
  controller_id INT UNSIGNED NOT NULL,
    FOREIGN KEY(controller_id) REFERENCES employee(id) ON UPDATE CASCADE,
  sponsor_id INT UNSIGNED NOT NULL,
    FOREIGN KEY(sponsor_id) REFERENCES program(id) ON UPDATE CASCADE,
  sponsor_amount BIGINT UNSIGNED NOT NULL,
  managing_org_id INT UNSIGNED NOT NULL,
    FOREIGN KEY(managing_org_id) REFERENCES organisation(id) ON UPDATE CASCADE,
  managing_date DATE NOT NULL CHECK( managing_date<start_date
                                AND managing_date>evaluation_date )
);
```

Σημειώνουμε ότι:

- Χρησιμοποιήσαμε αυτόματη δεικτοδότηση ως πρωτεύον κλειδί των εγγραφών.

- Προσθέτουμε ενα περιορισμό (*check*) στην ημερομηνία λήξης, *end_date*, η οποία ελέγχει οτι η διάρκεια του έργου είναι μεταξύ 1 και 4 ετών (μέσω αυτού ελέγχεται αυτόματα και η προφανής συνθήκη οτι η ημερομηνία λήξης πρέπει να είναι μετά της ημερομηνίας έναρξης, *start_date*).
- Θεωρούμε οτι ο βαθμός αξιολόγησης του έργου, *evaluator_grade*), είναι ακέραιος αριθμός μεταξύ του 0 και 100.
- Θεωρούμε οτι η ημερομηνία αξιολόγησης του έργου, *evaluation_date*, πρέπει να είναι πριν την ημερομηνία έναρξης του, δηλαδή το έργο πρέπει να έχει αξιολογηθεί πριν την έναρξή του.
- Με *supervisor_id*, εννοούμε τον δείκτη του ερευνητή που είναι ο επιστημονικός υπεύθυνος του έργου.
- Με *controller_id*, εννοούμε τον δείκτη του στελέχους του ΕΛΙΔΕΚ που διαχειρίζεται το έργο.
- Με *sponsor_id*, εννοούμε τον δείκτη του οργανισμού που χρηματοδοτεί το έργο.
- Θεωρούμε οτι το ποσό χρηματοδότησης είναι θετικός ακέραιος αριθμός.
- Με *managing_org_id*, εννοούμε τον δείκτη του οργανισμού διαχειρίζεται το έργο.
- Θεωρούμε οτι η ημερομηνία που ανέλαβε ο οργανισμός το έργο, *managing_date*, είναι μετά της ημερομηνίας αξιολόγησής του αλλά πριν την ημερομηνία έναρξής του. Γίνεται ο σχετικός έλεγχος μέσω του περιορισμού *check*.

Δημιουργούμε δυο ευρετήρια στον πίνακα *project*, επάνω στα *attributes*: *start_date* και *end_date* καθώς τα χρησιμοποιούμε σε διάφορα ερωτήματα του τρίτου μέρους της εργασίας (πχ για αναζήτηση έργων με βάση τη διάρκεια τους ή εύρεση των ενεργών έργων κτλ). Δημιουργούμε ακόμη ένα ευρετήριο στο *attribute managing_date*, διότι στο σχετικό ερώτημα του τρίτου μέρους, γίνονται διάφορα φιλτραρίσματα με βάση την ημερομηνία αυτή. Δίνουμε πιο κάτω το αντίστοιχο μέρος του κώδικα:

```
CREATE OR REPLACE INDEX idx_project_sd ON project(start_date);
CREATE OR REPLACE INDEX idx_project_ed ON project(end_date);
CREATE OR REPLACE INDEX idx_project_md ON project(managing_date);
```

Προφανώς η ίδια η *SQL* δημιουργεί αυτόματα ευρετήρια για τα πρωτεύων και εξωτερικά κλειδία των πινάκων, τα οποία αποτελούν το βασικό μέσο αναζήτησης στα *queries* μας.

Θεωρούμε επίσης τον εξής περιορισμό σχετικά με τα έργα: ο ερευνητής που αξιολογεί το έργο δεν πρέπει να εργάζεται στον οργανισμό που το διαχειρίζεται. Για τον έλεγχο αυτού δημιουργούμε ένα *trigger*, *check_evaluator*, το οποίο εξετάζει τον περιορισμό αυτό πριν απο κάθε εισαγωγή στον πίνακα *project*. Σε περίπτωση παραβίασης του σταματάει την εισαγωγή και επιστρέφει σχετικό μήνυμα. Δίνουμε τον αντίστοιχο κώδικα πιο κάτω:

```
DELIMITER $$
CREATE OR REPLACE TRIGGER check_evaluator
BEFORE INSERT ON project FOR EACH ROW
BEGIN
  IF NEW.managing_org_id = (
                SELECT org_id FROM researcher
                WHERE researcher.id = NEW.evaluator_id
                                ) THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = "Evaluator shouldn't be working
                        for managing organisation";
  END IF;
END; $$
DELIMITER ;
```

Δίνουμε τώρα τον κώδικα που δημιουργεί τον πίνακα *works_on* που αντιστοιχεί στην *many to many* συσχέτιση *Research* του *ER* (δηλαδή ποιοί ερευνητές εργάζονται στο έργο):

```
CREATE OR REPLACE TABLE works_on(
  project_id INT UNSIGNED NOT NULL,
    FOREIGN KEY(project_id) REFERENCES project(id) ON UPDATE CASCADE,
  researcher_id INT UNSIGNED NOT NULL,
    FOREIGN KEY(researcher_id) REFERENCES researcher(id) ON UPDATE CASCADE,
  PRIMARY KEY(project_id, researcher_id)
);
```

Προφανώς κάθε εγγραφή του πίνακα αποτελείται απο ζεύγη με τους δείκτες του ερευνητή με τον αντίστοιχο δείκτη του έργου.

Θεωρούμε τον εξής περιορισμό: ο ερευνητής που αξιολογεί το έργο δεν πρέπει να εργάζεται στο έργο. Για τον έλεγχο αυτού δημιουργούμε ένα *trigger*, *check_evaluator_2*, το οποίο εξετάζει τον περιορισμό αυτό πριν απο κάθε εισαγωγή στον πίνακα *works_on*. Σε περίπτωση παραβίασης του σταματάει την εισαγωγή και επιστρέφει σχετικό μήνυμα. Δίνουμε τον αντίστοιχο κώδικα πιο κάτω:

```
DELIMITER $$
CREATE OR REPLACE TRIGGER check_evaluator_2
BEFORE INSERT ON works_on FOR EACH ROW
BEGIN
  IF ( SELECT managing_org_id FROM project
    WHERE project.id = NEW.project_id )
   = ( SELECT org_id FROM researcher
    WHERE researcher.id = NEW.researcher_id ) THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = "Evaluator shouldn't be working in the project";
  END IF;
END; $$
DELIMITER ;
```

Όπως προαναφέραμε, τα επιστημονικά πεδία του κάθε *project* θεωρήσαμε πιο χρηστικό να θεωρηθούν ως χαρακτηριστικό του *project*. Εφόσον όμως το κάθε *project* μπορεί να εντάσσεται σε περισσότερα από ένα επιστημονικά πεδία, οδηγηθήκαμε στην δημιουργία ξεχωριστού πίνακα, *field*, ο οποίος περιέχει ως εγγραφές τα ζεύγη κωδικού του έργου (*project_id*) και επιστημονικού πεδίου (*field_name*). Δίνουμε πιο κάτω τον κώδικα:

```
CREATE OR REPLACE TABLE field(
  project_id INT UNSIGNED NOT NULL,
    FOREIGN KEY(project_id) REFERENCES project(id) ON UPDATE CASCADE,
  field_name ENUM('ΦΥΣΙΚΕΣ ΕΠΙΣΤΗΜΕΣ', 'ΤΕΧΝΟΛΟΓΙΑ', 'ΙΑΤΡΙΚΗ', 'ΓΕΩΠΟΝΙΚΗ',
    'ΠΛΗΡΟΦΟΡΙΚΗ', 'ΚΟΙΝΩΝΙΚΕΣ ΕΠΙΣΤΗΜΕΣ', 'ΤΕΧΝΕΣ', 'ΟΙΚΟΝΟΜΙΚΑ') NOT NULL,
  PRIMARY KEY(project_id, field_name)
);
```

Σημειώνουμε ότι:

- Για την αποφυγή μπερδέματος ανάμεσα σε εγγραφές οι οποίες αναφέρονται στο ίδιο επιστημονικό πεδίο με διαφορετικό τρόπο (πχ *economics* και οικονομικά) περιορίσαμε το πεδίο τιμών του *field_name* σε 8

προκαθορισμένα επιστημονικά πεδία: *Φυσικές Επιστήμες*, *Τεχνολογία*, *Ιατρική*, *Γεωπονική*, *Πληροφορική* και *Κοινωνικές Επιστήμες*, *Τέχνες* και *Οικονομικά* (την ίδια λογική ακολουθήσαμε και για το πεδίο *department* στον πίνακα *program* και για το πεδίο φύλο όπου αυτό εμφανίζεται).

Δίνουμε τώρα κώδικα σχετικά με τη δημιουργία του πίκανα των παραδοτέων, *report*:

```
CREATE OR REPLACE TABLE report(
  id INT UNSIGNED NOT NULL AUTO_INCREMENT,
  project_id INT UNSIGNED NOT NULL,
    FOREIGN KEY(project_id) REFERENCES project(id) ON UPDATE CASCADE,
  title VARCHAR(100) NOT NULL,
  summary MEDIUMBLOB NOT NULL,
  delivery_date DATE NOT NULL,
  PRIMARY KEY(id, project_id)
);
```

Σημειώνουμε ότι:

- Χρησιμοποιήσαμε αυτόματη δεικτοδότηση ως πρωτεύον κλειδί των εγγραφών.
- Σε κάθε εγγραφή έχουμε και τον δείκτη του έργου στο οποίο αντιστοιχεί το παραδοτέο.

Θεωρούμε τον εξής προφανή περιορισμό: το παραδοτέο πρέπει να έχει ημερομηνία παράδοσης στο διάστημα κατα το οποίο το αντίστοιχο έργο είναι ακόμη ενεργό. Δημιουργούμε ένα *trigger*, *report_date_check*, το οποίο πριν απο κάθε εισαγωγή στον πίνακα *report* εξετάζει τον περιορισμό αυτό και σε περίπτωση παραβίασης του, σταματάει την εισαγωγή και επιστρέφει σχετικό μήνυμα. Δίνουμε πιο κάτω τον κώδικα:

```
DELIMITER $$
CREATE OR REPLACE TRIGGER report_date_check
BEFORE INSERT ON report FOR EACH ROW
BEGIN
  IF NEW.delivery_date NOT BETWEEN
  ( SELECT start_date FROM project WHERE project.id = NEW.project_id ) AND
  ( SELECT end_date FROM project WHERE project.id = NEW.project_id ) THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = "Report should be delivered while the
                          project is still active";
  END IF;
END; $$
DELIMITER ;
```

Δημιουργούμε επίσης και τρείς όψεις η οποίες θα μας βοηθήσουν σε απλοποίηση των *queries* στα επόμενα στάδια της εργασίας.

Δημιουργούμε αρχικά ένα *view*, *active_projects*, οπου διατηρούμε τα ενεργά έργα στη βάση (ενεργά με βάση την ημερομηνία που δημιουργείται το *view*) καθώς τα επικαλούμαστε συχνά στα ερωτήματα του τρίτου μέρους. Δίνουμε τον αντίστοχο κώδικα:

```
CREATE OR REPLACE VIEW active_projects AS
  SELECT * FROM
  project WHERE( start_date<=CURRENT_DATE() AND end_date>CURRENT_DATE() );
```

Δημιουργούμε επίσης μια όψη, *researcher_project*, όπου αποθηκεύουμε την πλήρης πληροφορία της συσχέτισης *works_on* (μεταξύ έργου και ερευνητή), δηλαδή ως εγγραφές του έχει τα στοιχεία του έργου μαζί με τα στοιχεία του ερευνητή που εργάζεται σ' αυτό, όπου αυτό συμβάινει για κάθε έργο και για κάθε ερευνητή που εργάζεται σ' αυτό. Η ανάγκη γι'αυτό δικαιολογείται στο ότι χρειαζόμαστε συχνά το τριπλό *join* μεταξύ *project*, *researcher* και *works_on* (επίσης ζητείται ρητά στο ερώτημα 3.2). Δίνουμε κάτω τον σχετικό κώδικα:

```
CREATE OR REPLACE VIEW researcher_project AS
  SELECT * FROM
  (SELECT id AS res_id, org_id, first_name, last_name,
      date_of_birth, sex, works_date, project_id FROM
  researcher INNER JOIN works_on
  ON researcher.id = works_on.researcher_id) AS temp
  INNER JOIN project
  ON temp.project_id = project.id;
```

Τέλος, δημιουργούμε ακόμη ένα *view*, *project_organisation*, όπου αποθηκεύουμε τα χαρακτηριστικά των *project* μαζί με τα αντίστοιχα χαρακτηριστικά των οργανισμών που τα διαχειρίζονται. Το δημιουργούμε καθώς χρειαζόμαστε συχνά την πληροφορία του συγκεκριμένου *join*. Δίνουμε κάτω τον σχετικό κώδικα:

```
CREATE OR REPLACE VIEW project_organisation AS
  SELECT * FROM
  project INNER JOIN (
          SELECT id org_id, org_name, abbreviation, category, street,
                  street_no, postal_code, city FROM organisation
                  ) AS temp
  ON project.managing_org_id = temp.org_id;
```

Όλος ο πιο πάνω κώδικας είναι εντός του αρχείου *project_db_tables.sql*.

Ερωτήματα Τρίτου Μέρους.

Σε αυτό το σημείο θα αναφερθούμε στα ερωτήματα (*queries*) του τρίτου μέρους της εργασίας.

Ερώτημα 3.1

Στο ερώτημα 3.1 θεωρούμε οτι μας ζητείται να παρέχουμε στον χρήστη τις εξής δυνατότητες:

(a) Να μπορεί να δει όλα τα προγράμματα που είναι διαθέσιμα.

```
---------- ΕΡΩΤΗΜΑ 3.1 (Α)
-- Όλα τα προγράμματα που είναι διαθέσιμα.
SELECT * FROM program;
```

(b) Να μπορεί να δει τους ερευνητές που εργάζονται σε ένα συγκεκριμένο έργο, δίνοντας το *ID* αυτού.

```
---------- ΕΡΩΤΗΜΑ 3.1 (Β)
-- Ερευνητές που εργάζονται σε δοσμένο έργο (δώσε ID έργου). (πχ id = '23')
SELECT res_id, first_name, last_name, org_id
FROM researcher_project WHERE project_id = '23';
```

(c) Να μπορεί να δεί τα έργα τα οποία θα είναι ενεργά σε μια συγκεκριμένη ημερομηνία, δίνοντας την ημερομηνία αυτή.

```
---------- ΕΡΩΤΗΜΑ 3.1 (C)
-- Έργα τα οποία θα είναι ενεργά σε δοσμένη ημερομηνία (είσοδος απο χρήστη)
-- (πχ date = '2022-09-15')
SELECT * FROM project
WHERE start_date<='2022-09-15' AND end_date>'2022-09-15';
```

(d) Να μπορεί να δει όλα τα έργα που διαχειρίζεται ένα συγκεκριμένο στέλεχος του ΕΛΙΔΕΚ, δίνοντας το *ID* αυτού.

```
---------- ΕΡΩΤΗΜΑ 3.1 (D)
-- Έργα τα οποία χειρίζεται δοσμένο στέλεχος (δώσε ID στελέχους). (πχ id = '13')
SELECT * FROM project
WHERE controller_id = '13';
```

(e) Να μπορεί να δει όλα τα έργα που έχουν μικρότερη διάρκεια από την δοσμένη (σε χρόνια).

```
---------- ΕΡΩΤΗΜΑ 3.1 (E)
-- Έργα με διάρκεια μικρότερη απο δοσμένο όριο. (δώσε διάρκεια σε έτη).
-- (πχ duration = 3.2 έτη)
SELECT *, TRUNCATE(DATEDIFF(end_date,start_date)/365,2) duration FROM project
WHERE DATEDIFF(end_date,start_date)<=365*3.2;
```

Ερώτημα 3.2

Στο ερώτημα 3.2 μας ζητείται να δημιούργησουμε δύο όψεις. Δημιουργήσαμε πιο πάνω τις δύο όψεις *researcher_project* και *active_projects*. Στην πρώτη φαίνεται η σύνδεση μεταξύ ερευνητή και έργου, δηλαδή στην κάθε εγγραφή βρίσκονται τα στοιχεία ενός ερευνητή και τα στοιχεία του έργου στο οποίο εργάζεται. Ενώ στη δεύτερη όψη εμφανίζονται όλα τα έργα τα οποία είναι ενεργά την συγκεκριμένη ημερομηνία (ζητείται στην εκφώνηση). Δώσαμε πιο πάνω στο κομμάτι της υλοποίησης της βάσης τους κώδικες που αντιστοιχούν στην δημιουργία των όψεων αυτών. Δίνουμε πιο κάτω τις εντολές με τις οποίες επιστρέφουμε τα δεδομένα αυτά:

```
---------- ΕΡΩΤΗΜΑ 3.2 (A)
-- Όψη με ενεργά έργα.
SELECT * FROM active_projects;
```

```
---------- ΕΡΩΤΗΜΑ 3.2 (B)
-- Όψη με έργα ανα ερευνητή.
SELECT res_id, first_name, last_name, project_id, title, start_date, end_date
FROM researcher_project;
```

Ερώτημα 3.3

Στο ερώτημα 3.3 δίνουμε στον χρήστη την επιλογή:

1. Να δει τα ενεργά εργα σε ένα επιστημονικό πεδίο που θα επιλέξει.

```
---------- ΕΡΩΤΗΜΑ 3.3 (Β)
-- Ερευνητές που εργάζονται σε δοσμένο επιστημονικό πεδίο (δώσε επιστημονικό πεδίο).
-- (πχ field = ΙΑΤΡΙΚΗ)
SELECT res_id, first_name, last_name, date_of_birth, sex, works_date FROM
researcher_project rp INNER JOIN field ON rp.id = field.project_id
WHERE( field_name = 'ΙΑΤΡΙΚΗ' AND start_date<=CURRENT_DATE()
       AND end_date>DATE_ADD(CURRENT_DATE(), INTERVAL -1 YEAR) );
```

2. Να δει τους ερευνητές που ασχολήθηκαν/εργάστηκαν σε κάποιο έργο του επιστημονικού πεδίου που θα επιλέξει, εντός του διαστήματος ενός έτους.

```
---------- ΕΡΩΤΗΜΑ 3.3 (Β)
-- Ερευνητές που εργάστηκαν στο δοσμένο επιστημονικό πεδίο
-- το τελευταίο έτος (δώσε επιστημονικό πεδίο). (πχ field = ΙΑΤΡΙΚΗ)
SELECT res_id, first_name, last_name, date_of_birth, sex, works_date FROM
researcher_project rp INNER JOIN field ON rp.id = field.project_id
WHERE( field_name = 'ΙΑΤΡΙΚΗ' AND start_date<=CURRENT_DATE()
       AND end_date>DATE_ADD(CURRENT_DATE(), INTERVAL -1 YEAR) );
```

Ερώτημα 3.4

Στο ερώτημα 3.4 αυτό που υλοποιήσαμε είναι να επιστρέφει τα ζεύγη οργανισμών (δηλαδή δυάδες οργανισμών) τα οποία έλαβαν σε διάστημα 2 συνεχόμενων ετών τον ίδιο συνολικό αριθμό έργων, με την επιπλέον συνθήκη και τις δύο χρονιές ο κάθε οργανισμός να έχει λάβει περισσότερα από 10 έργα. Θεωρούμε ότι ένας οργανισμός λαμβάνει ένα έργο την ημερομηνία στην οποία ξεκίνησε να το διαχειρίζεται (*managing_date*). Δίνουμε πιο κάτω τον σχετικό κώδικα:

```
---------- ΕΡΩΤΗΜΑ 3.4
-- Οργανισμοί με ίδιο αριθμό έργων σε δυο συνεχόμενα έτη.
CREATE OR REPLACE TEMPORARY TABLE temp
  SELECT org_id , YEAR(managing_date) m_year, COUNT(*) no_projects
  FROM project_organisation
  GROUP BY org_id, m_year
  HAVING no_projects>=10;
-----
CREATE OR REPLACE TEMPORARY TABLE temp2
  SELECT t1.org_id, t1.m_year year1, t1.no_projects year1_count,
         t2.m_year year2, t2.no_projects year2_count
  FROM temp t1 INNER JOIN temp t2
  ON t1.org_id = t2.org_id AND t1.m_year = t2.m_year-1;
-----
CREATE OR REPLACE TEMPORARY TABLE results
  SELECT t1.org_id id1, t2.org_id id2, t1.year1, t1.year2,
         t1.year1_count + t1.year2_count total
  FROM temp2 t1 INNER JOIN temp2 t2
  ON t1.year1 = t2.year1
  WHERE t1.org_id<t2.org_id AND t1.year1_count+t1.year2_count = t2.year1_count+t2.year2_count;
```

Λόγω του οτι η υλοποίησή μας χρησιμοποιεί πιο σύνθετα *queries*, κάνουμε χρήση προσωρινών πινάκων (*temporary table*) για απλούστερη παρουσίασή τους.

Ερώτημα 3.5

Στο ερώτημα 3.5 παρουσιάζουμε τα 3 πιο δημοφιλή ζεύγη επιστημονικών πεδίων (δυάδες πεδίων), με βάση δυο κριτήρια:

1. Είτε με βάση το συνολικό ποσό επιχορήγησης σε έργα των συγκεκριμένων πεδίων (επιχορηγήσεις σε έργα του συγκεκριμένου ζεύγους).

```
---------- ΕΡΩΤΗΜΑ 3.5 (Α)
-- Top 3 ζεύγη επιστημονικών πεδίων (με κριτήριο το συνολικό ποσό χρηματοδότησης).
CREATE TEMPORARY TABLE field_pairs
  SELECT f1.field_name field1, f2.field_name field2, f1.project_id id FROM
  field f1 INNER JOIN field f2 ON f1.project_id = f2.project_id
  WHERE f1.field_name < f2.field_name;
-----
SELECT field1, field2, SUM(project.sponsor_amount) total FROM
field_pairs INNER JOIN project on field_pairs.id = project.id
GROUP BY field1, field2 ORDER BY total DESC LIMIT 3;
```

2. Είτε με βάση τον αριθμό των έργων με τα οποία συνδέονται.

```
---------- ΕΡΩΤΗΜΑ 3.5 (Β)
-- Top 3 ζεύγη επιστημονικών πεδίων (με κριτήριο το συνολικό πλήθος έργων).
CREATE TEMPORARY TABLE field_pairs
  SELECT f1.field_name field1, f2.field_name field2, f1.project_id id FROM
  field f1 INNER JOIN field f2 ON f1.project_id = f2.project_id
  WHERE f1.field_name < f2.field_name;
-----
SELECT field1, field2, COUNT(*) freq FROM field_pairs
GROUP BY field1, field2 ORDER BY freq DESC LIMIT 3;
```

Ερώτημα 3.6

Στο ερώτημα 3.6 επιστρέφουμε τους/τον ερευνητές/ερευνητή ηλικίας κάτω των 40 ετών οι οποίοι εργάζονται αυτή τη στιγμή στον μεγαλύτερο αριθμό ενεργών έργων. Δίνουμε τον σχετικό κώδικα:

```
---------- ΕΡΩΤΗΜΑ 3.6
-- Νέοι ερευνητές με τα περισσότερα ενεργά έργα.
CREATE OR REPLACE TEMPORARY TABLE temp
  SELECT first_name, last_name, COUNT(*) no_projects  FROM
  active_projects ac INNER JOIN researcher_project rp
  ON ac.id = rp.id
  WHERE rp.date_of_birth>DATE_ADD(CURRENT_DATE(), INTERVAL -40 YEAR)
  GROUP BY rp.id ORDER BY no_projects;
-----
SELECT * FROM temp
WHERE no_projects = (SELECT MAX(no_projects) FROM temp);
```

Ερώτημα 3.7

Για το ερώτημα 3.7, ο χρήστης μπορεί να δει τα top-5 στελέχη του ΕΛΙΔΕΚ με κριτήριο το συνολικό ποσό

χρηματοδότησης που έχουν δώσει σε μία μόνο εταιρεία (δηλαδή να διαχειρίζονται έργα αυτής της εταιρείας). Δίνουμε πιο κάτω τον σχετικό κώδικα:

```
---------- ΕΡΩΤΗΜΑ 3.7
-- Top 5 στελέχοι που έχουν δώσει τα περισσότερα χρήματα σε εταιρείες.
SELECT first_name, last_name, org_name, SUM(sponsor_amount) total FROM
employee INNER JOIN (
        SELECT * FROM project_organisation
        WHERE category = 'Company'
                ) AS temp
ON employee.id = temp.controller_id
GROUP BY employee.id, temp.org_id
ORDER BY total DESC LIMIT 5;
```

Ερώτημα 3.8

Τέλος, στο ερώτημα 3.8 παρουσιάζουμε του ερευνητές που εργάζονται σε 5 ή περισσότερα από τα ενεργά έργα τα οποία όμως να μην έχουν παραδοτέα. Δίνουμε πιο κάτω τον σχετικό κώδικα:

```
---------- ΕΡΩΤΗΜΑ 3.8
-- Ερευνητές που εργάζονται σε τουλάχιστον 5 έργα χωρίς παραδοτέα.
CREATE OR REPLACE TEMPORARY TABLE temp
  SELECT id FROM active_projects
  WHERE id NOT IN (
            SELECT ap.id FROM
            active_projects ap INNER JOIN report
            ON ap.id = report.project_id
            GROUP BY ap.id
                );
-----
SELECT first_name, last_name, COUNT(*) no_projects FROM
researcher_project rp
WHERE rp.id IN (SELECT * FROM temp)
GROUP BY rp.res_id HAVING no_projects>=5 ORDER BY no_projects;
```

Όλα τα πιο πάνω βρίσκονται στο αρχείο *project_db_queries.sql*.

Dummy Data

Πριν προχωρήσουμε στις οδηγίες εγκατάστασης, θα κάνουμε ενα μικρό σχόλιο για τον κώδικά μας στην *python* μέσου του οποίου παράξαμε τα *dummy data* της βάσης. Προσπαθήσαμε να προσαρμόσουμε τον κώδικά μας ούτως ώστε τα παραγόμενα δεδομένα να συμφωνούν με τους περιορισμούς που έχουμε εφαρμόσει στην βάση δεδομένων. Παρ' ολ' αυτά κάποιοι περιορισμοί ήταν περίπλοκο να προγραμματιστούν στην *python* και επομένως ο κώδικας είναι πιθανόν να παράγει κάποια δεδομένα τα οποία δεν ανταποκρίνονται στους περιορισμούς της βάσης. Άρα υπάρχει το ενδεχόμενο κάποιες εγγραφές να μην περάσουν απ' τα *triggers* που προσθέσαμε και κατα συνέπεια και κάποιες άλλες εγγραφές, οι οποίες έχουν εξωτερικό κλειδί σ' αυτές, να έχουν και εκείνες πρόβλημα. Εν γένει όμως ο όγκος δεδομένων που παράγει είναι αρκετά μεγάλος έτσι ώστε και να αγνοηθούν οι προβληματικές αυτές εγγραφές, υπάρχουν αρκετά δεδομένα ώστε τα *queries* να παρουσιάζουν ουσιαστικές απαντήσεις.

Τα δεδομένα όμως στο αρχείο εισαγωγής, *project_db_insert.sql*, είναι φιλτραρισμένα ώστε οι εντολές να μην παρουσιάζουν σφάλματα. Το σχετικό αρχείο με τον κώδικα της *python* είναι το *dummy_data_generator.py*.

Οδηγίες Εγκατάστασης

Το *repository* μας ονομάζεται *Team_44-Databases* και βρίσκεται στον σύνδεσμο:

`https://github.com/NikMalekkides/Team_44-Databases.git`

Σημειώνουμε αρχικά οτι ο κώδικας της *SQL* εχει γραφτεί σε περιβάλλον *MySQL Workbench* και *DBeaver*.

Αρχικά τρέξτε το αρχείο *project_db_tables.sql* το οποίο βρίσκεται εντός του φακέλου *SQL*. Το αρχείο αυτό θα κατασκευάσει το σχήμα, τους πίνακες, τις όψεις, τα ευρετήρια και τα *triggers*.

Έπειτα τρέξτε το αρχείο *project_db_insert.sql* το οποίο βρίσκεται εντός του φακέλου *SQL*. Το αρχείο αυτό εισάγει τα ψευδο-δεδομένα που παράξαμε στη βάση. Σε περίπτωση που το κείμενο του αρχείου έχει μεταφράσει τα ελληνικά σε σύμβολα, έχουμε προσθέσει εντός του φακέλου *SQL* ένα αρχείο *word* μέσα στο οποίο βρίσκονται όλες οι εντολές εισαγωγής που βρίσκονται και εντός του *script*. Ομοίως για τον ίδιο λόγο υπάρχει και το *dummy_data.txt* εντός του φακέλου.

Απομένει η εγκατάσταση του *user interface*. Θα εξηγήσουμε τα διαδικαστικά σε περίπτωση που χρησιμοποείτε *XAMPP* όπως εμείς. Αντιγράφεται τον φάκελο *project_db* και τον τοποθετείτε εντός του φακέλου *htdocs* που βρίσκεται εντός του φακέλου *xampp* που βρίσκεται στο δίσκο σας.

Σε περίπτωση που χρησιμοποιείτε κάποιο άλλο *web development stack*, ακολουθείστε τα ανάλογα βήματα για την αντιγραφή του φακέλου *project_db*. Σ' αυτή την περίπτωση όμως, πηγαίνετε στο αρχείο *db_connect.php* το οποίο βρίσκεται εντός του φακέλου *project_db* και τροποιήστε τις μεταβλητές $servername, $username και $password ώστε να ανταποκρίνονται στα δεδομένα της δικής σας σύνδεσης.

Έχοντας κάνει τα πιο πάνω, εαν πάτε στο σύνδεσμο: `http://localhost/project_db/index.php` θα μπορείτε να δείτε το περιβάλλον της εφαρμογής μας, καθώς και να εκτελέσετε τα *queries* της εκφώνησης.

User Interface.

Και τέλος, ενα μικρό σχόλιο σχετικά με το *user interface* της εφαρμογής μας. Επιλέξαμε στην εφαρμογή να εμφανίζεται και το *id* μιας εγγραφής στον χρήστη καθώς και να ζητάμε το *id* της όποτε αυτή γίνεται *reference* απο κάποιο *foreign key*. Αναγνωρίζουμε οτι σ' ενα πραγματικό σενάριο, η πληροφορία αυτή δεν επιστρέφεται, ούτε ζητείται απο τον χρήστη. Όμως εν τέλει, αποφασίσαμε να επιτρέψουμε την χρήση των *id* των εγγραφών, για απλούστευση της παρουσίασης των λειτουργικών δυνατοτήτων της εφαρμογής μας.