Βάσεις Δεδομένων – Εξαμηνιαία Εργασία

Ομάδα 21

6ο Εξάμηνο 2021 – 2022

Βασιλείου Δημήτριος – el19830
Ζαρίφης Στυλιανός – el20435
Ρόκομος Ιωάννης – el19061

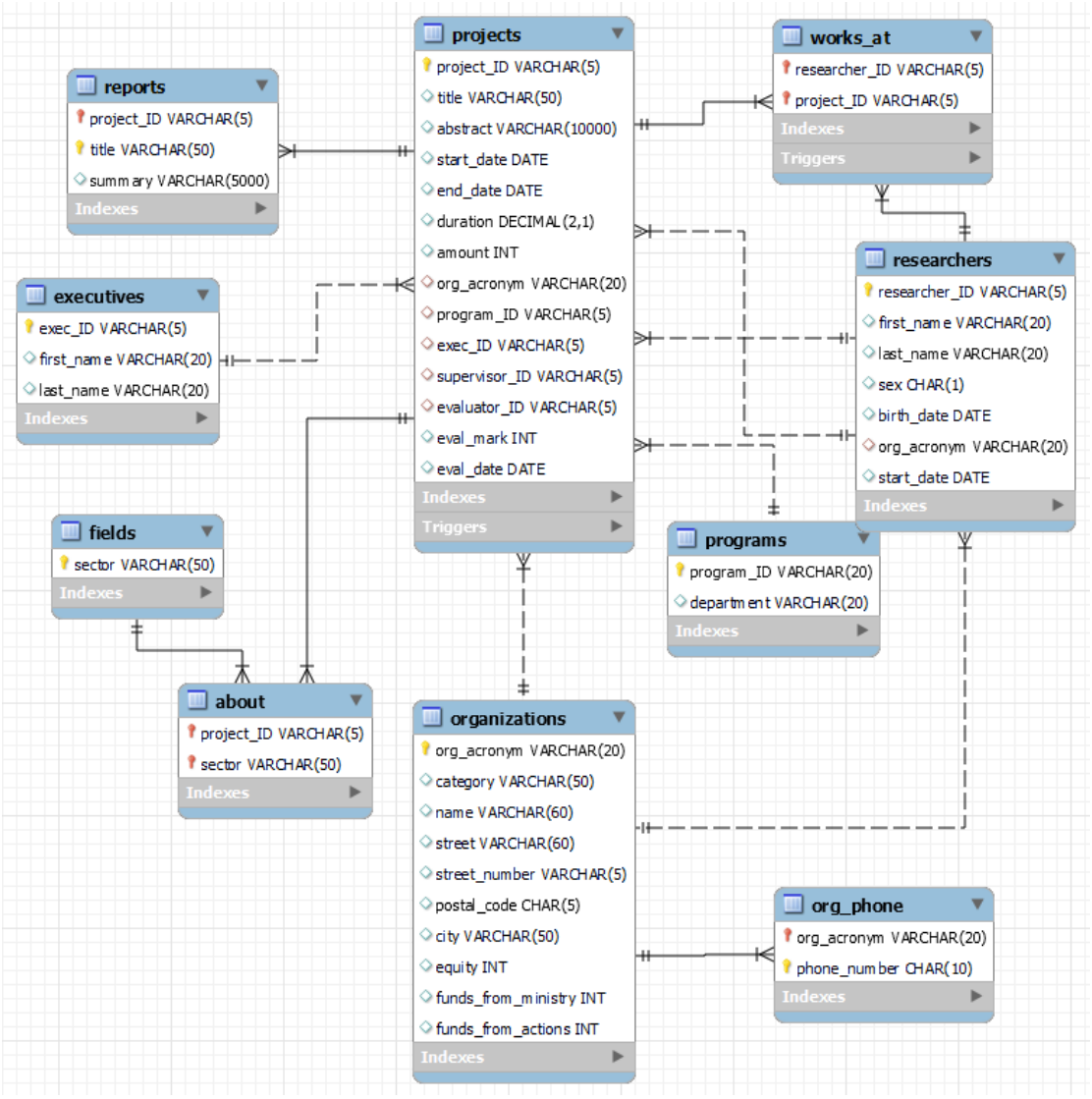
Περιεχόμενα

Ζητούμενο 2.1 2
Σχεσιακό Διάγραμμα 2
Σχέση organizations 2
Σχέση org_phone 3
Σχέση programs 3
Σχέση executives 3
Σχέση fields 3
Σχέση researchers 3
Σχέση projects 3
Σχέση reports 4
Σχέση about 4
Σχέση works_at 4
Ευρετήρια 4
Ζητούμενο 2.2 5
DDL Scripts 5
Relational Schema 5
Indices 7
Triggers 8
DML Scripts 10
Queries 10
Views 12
Ζητούμενο 2.3 13
Βήματα εγκατάστασης της εφαρμογής και βιβλιοθηκών που απαιτούνται 13
Ζητούμενο 2.4 15
Git – Repo 15

Ζητούμενο 2.1

Σχεσιακό Διάγραμμα

Παραθέτουμε σχηματικά το σχεσιακό διάγραμμα της Βάσης Δεδομένων:



Σχέση organizations

Ως Primary Key ορίσαμε το ακρωνύμιο κάθε οργανισμού (org_acronym), διότι μπορεί να τον προσδιορίσει μοναδικά. Σε αυτήν τη σχέση δεν υπάρχουν Foreign Keys.

Το Attribute category αποθηκεύεται η πληροφορία αν ο οργανισμός είναι πανεπιστήμιο, ερευνητικό κέντρο ή εταιρία. Ανάλογα με την κατηγορία αυτή, τα Attributes equity (Ίδια κεφάλαια), funds_from_ministry (προϋπολογισμός από το Υπουργείο Παιδείας) και funds_from_actions (προϋπολογισμός από ιδιωτικές δράσεις) λαμβάνουν κατάλληλες τιμές. Οι τιμές είναι NULL όταν ένας οργανισμός δε διαθέτει τέτοιου είδους χρηματοδοτήσεις. Για παράδειγμα αν ένας οργανισμός είναι εταιρεία, έχει μόνο ίδια κεφάλαια, άρα λαμβάνει τιμές NULL στα Attributes funds_from_ministry και funds_from_actions.

Σχέση org_phone

Στο ER Model, κάθε οργανισμός διαθέτει πολλούς αριθμούς τηλεφώνου, συνεπώς στο Relational Model απαιτείται η δημιουργία μιας νέας σχέσης (org_phone) με Attributes το ακρωνύμιο του εκάστοτε οργανισμού (org_acronym) και έναν αριθμό τηλεφώνου (phone_number). Με αυτόν τον τρόπο εξασφαλίζουμε ότι ένας οργανισμός μπορεί να έχει πολλούς αριθμούς τηλεφώνου.

Το Primary Key της νέας σχέσης είναι το ζεύγος (org_acronym, phone_number).

Ορίσαμε το Attribute org_acronym ως Foreign Key της σχέσης διότι δηλώνει ότι το τηλέφωνο της σχέσης (phone_number) ανήκει στον αντίστοιχο οργανισμό.

Σχέση programs

Ως Primary Key ορίσαμε τον αναγνωριστικό κωδικό του κάθε προγράμματος (program_ID), διότι μπορεί να το προσδιορίσει μοναδικά. Σε αυτήν τη σχέση δεν υπάρχουν Foreign Keys.

Σχέση executives

Επιλέγουμε Primary Key τον αναγνωριστικό κωδικό του κάθε στελέχους (executive_ID), διότι μπορεί να τον προσδιορίσει μοναδικά. Σε αυτήν τη σχέση δεν υπάρχουν Foreign Keys.

Σχέση fields

Ως Primary Key χρησιμοποιήσαμε το μοναδικό Attribute του κάθε πεδίου (sector), διότι μπορεί να το προσδιορίσει μοναδικά. Σε αυτήν τη σχέση δεν υπάρχουν Foreign Keys.

Σχέση researchers

Το Primary Key της σχέσης είναι ο αναγνωριστικός κωδικός του κάθε ερευνητή (researcher_ID).

Ορίσαμε το Attribute org_acronym ως Foreign Key της σχέσης διότι δηλώνει τον οργανισμό στον οποίο εργάζεται ο αντίστοιχος ερευνητής.

Σχέση projects

Επιλέγουμε Primary Key τον αναγνωριστικό κωδικό του κάθε έργου (project_ID), διότι μπορεί να το προσδιορίσει μοναδικά.

Ορίσαμε το Attribute program_ID ως Foreign Key της σχέσης μεταξύ έργων και προγραμμάτων διότι δηλώνει ότι το έργο αυτό λαμβάνει χρηματοδότηση από το αντίστοιχο πρόγραμμα.

Το Attribute exec_ID είναι Foreign Key της σχέσης μεταξύ έργων και στελεχών καθώς δηλώνει ότι το έργο αυτό διαχειρίζεται από το αντίστοιχο στέλεχος.

Το Attribute org_acronym χρησιμοποιείται ως Foreign Key της σχέσης μεταξύ έργων και οργανισμών καθώς δηλώνει ότι το έργο αυτό ανήκει στον αντίστοιχο.

Χρησιμοποιούμε το Attribute supervisor_ID ως Foreign Key της σχέσης μεταξύ έργων και επιβλεπόντων καθώς δηλώνει ότι το έργο αυτό επιβλέπεται από τον αντίστοιχο επιβλέποντα.

Το Attribute evaluator_ID χρησιμοποιείται ως Foreign Key της σχέσης μεταξύ έργων και αξιολογητών καθώς δηλώνει ότι το έργο αυτό αξιολογείται από τον αντίστοιχο αξιολογητή.

Σχέση reports

Χρησιμοποιούμε ως Primary Key το συνδυασμό (project_ID, title) αφού μπορεί να προσδιορίζει μοναδικά τα παραδοτέα.

Σχέση about

Επιλέγουμε Primary Key το ζεύγος (project_ID, sector), διότι μπορεί να προσδιορίσει μοναδικά τα records.

Τα Attributes project_ID και sector χρησιμοποιούνται ως Foreign Keys της σχέσης μεταξύ έργων και πεδίων καθώς δηλώνει ότι το έργο αυτό πραγματεύεται το αντίστοιχο πεδίο.

Σχέση works_at

Χρησιμοποιούμε ως Primary Key το συνδυασμό (researcher_ID, project_ID) αφού μπορεί να προσδιορίσει μοναδικά τα records.

Τα Attributes researcher_ID και project_ID χρησιμοποιούνται ως Foreign Keys της σχέσης μεταξύ έργων και ερευνητών καθώς δηλώνει ότι στο έργο αυτό εργάζεται ο αντίστοιχος ερευνητής.

Ευρετήρια

Η επιλογή των ευρετηρίων έγινε βάσει των απαιτήσεων των ερωτημάτων (Queries).

Ορίσαμε indices για τα ακόλουθα Attributes των σχέσεων:

Relation	Attribute	Query
project	title	3.1
	start_date	3.1
	end_date	3.1
	duration	3.1
organization	category	3.7

Ζητούμενο 2.2

Όλα τα παρακάτω αρχεία μπορούν να βρεθούν και στο Git – Repo.

DDL Scripts

Relational Schema

```
CREATE SCHEMA elidek;
USE elidek;

drop table if exists organizations;
create table organizations (
      org_acronym         varchar(20),
      category            varchar(50), check (category in ('Centr',
'Univ', 'Comp')),
      name                varchar(60),
      street              varchar(60),
      street_number       varchar(5),
      postal_code         char(5),
      city                varchar(50),
      equity              int,      check (equity > 0),
      funds_from_ministry int,      check (funds_from_ministry > 0),
      funds_from_actions  int,      check (funds_from_actions > 0),
      primary key (org_acronym));


drop table if exists org_phone;
create table org_phone (
    org_acronym         varchar(20),
    phone_number        char(10),

    primary key (org_acronym, phone_number),
    foreign key (org_acronym) references organizations(org_acronym)
                                        on delete cascade
                                        on update cascade);


drop table if exists programs;
create table programs (
    program_ID          varchar(20),
    department          varchar(20),
    primary key (program_ID));


drop table if exists executives;
create table executives (
    exec_ID             varchar(5),
    first_name          varchar(20),
    last_name           varchar(20),
    primary key (exec_ID));


drop table if exists fields;
create table fields (
    sector varchar(50),

    primary key(sector));


drop table if exists researchers;
create table researchers (
```

```
    researcher_ID       varchar(5),
    first_name          varchar(20),
    last_name           varchar(20),
    sex                 char    (1), check (sex in ('M', 'F')),
    birth_date          date,
    org_acronym         varchar(20),
    start_date          date,

    primary key (researcher_ID),
    foreign key (org_acronym) references organizations(org_acronym)
                                        on delete set null
                                        on update cascade);


drop table if exists projects;
create table projects (
    project_ID    varchar(5),
    title         varchar(50),
    abstract      varchar(10000),
    start_date    date,
    end_date      date,
    duration      numeric(2, 1)  check (duration > 0.9 and duration <
4.1),
    amount        int            check (amount >= 100000 and amount
<= 1000000),
    org_acronym   varchar(20),
    program_ID    varchar(5),
    exec_ID       varchar(5),
    supervisor_ID varchar(5),
    evaluator_ID  varchar(5),
    eval_mark     int        check (eval_mark >= 0 and eval_mark <=
10),
    eval_date     date,

    primary key (project_ID),
    foreign key (program_ID)    references programs(program_ID)
                                        on delete set null
                                        on update cascade,

    foreign key (exec_ID)       references executives(exec_ID)
                                        on delete set null
                                        on update cascade,

    foreign key (org_acronym)   references organizations(org_acronym)
                                        on delete set null
                                        on update cascade,

    foreign key (supervisor_ID) references researchers(researcher_ID)
                                        on delete set null
                                        on update cascade,

    foreign key (evaluator_ID)  references researchers(researcher_ID)
                                        on delete set null
                                        on update cascade);


drop table if exists reports;
create table reports (
    project_ID    varchar(5)
    title         varchar(50),
    summary       varchar(5000),
```

```
    primary key(project_ID, title),
    foreign key (project_ID) references projects(project_ID)
                                        on delete cascade
                                        on update cascade);


drop table if exists about;
create table about (
    project_ID    varchar(5),
    sector        varchar(50),

    primary key (project_ID, sector),
    foreign key (project_ID)  references projects(project_ID)
                                        on delete cascade
                                        on update cascade,
    foreign key (sector)      references fields(sector)
                                        on delete cascade
                                        on update cascade);


drop table if exists works_at;
create table works_at (
    researcher_ID       varchar(5),
    project_ID          varchar(5),
    primary key (researcher_ID, project_ID),
    foreign key (researcher_ID) references researchers(researcher_ID)
                                        on delete cascade
                                        on update cascade,
    foreign key (project_ID)  references projects(project_ID)
                                        on delete cascade
                                        on update cascade);
```

Indices

```
create index project_title_index on projects(title);
create index project_sdate_index on projects(start_date);
create index project_edate_index on projects(end_date);
create index project_duration_index on projects(duration);
create index org_categ_index on organizations(category);
```

Triggers

```
/*check if a researcher works at a project that he evaluates*/
drop trigger if exists integrity;
delimiter //
create trigger integrity after insert on works_at
for each row
begin
	declare message_text varchar(50);

	if new.researcher_ID = (select evaluator_ID
				from projects
				where new.project_ID
				=projects.project_ID)

	then
		signal sqlstate '45000' set message_text = 'Error!
		Researcher = Evaluator!';
	end if;
end;//

/*check if a researcher works at a project that he evaluates*/
/*drop trigger if exists integrity_update;*/
delimiter //
create trigger integrity_update after update on works_at
for each row
begin
	declare message_text varchar(50);

	if new.researcher_ID = (select evaluator_ID
				from projects
				where new.project_ID =
				projects.project_ID)

	then
		signal sqlstate '45000' set message_text = 'Error!
		Researcher = Evaluator';
	end if;
end;//


/*check if the evaluator of project is the same with supervisor*/
/*drop trigger if exists integrity_projects;*/
delimiter //
create trigger integrity_projects after insert on projects
for each row
begin
	declare message_text varchar(50);

	if new.supervisor_ID = new.evaluator_ID then

		signal sqlstate '45000' set message_text = 'Error!
		Evaluator = Supervisor';

	end if;
end;//
```

```
/*check if the evaluator of project is the same with supervisor*/
/*drop trigger if exists integrity_projects_update;*/
delimiter //
create trigger integrity_projects_update after update on projects
for each row
begin
      declare message_text varchar(50);

      if new.supervisor_ID = new.evaluator_ID then

            signal sqlstate '45000' set message_text = 'Error!
            Evaluator = Supervisor';

      end if;
end;//

/*check if a researcher works only in projects of his organization*/
/*drop trigger if exists researcher_org;*/
delimiter //
create trigger researcher_org after insert on works_at
for each row
begin
      declare message_text varchar(50);

      if new.project_ID not in (select p.project_ID
                              from projects as p, organizations as o,
                              researchers as r
                              where new.researcher_ID =
                              r.researcher_ID
                              and r.org_acronym = o.org_acronym
                              and r.org_acronym = p.org_acronym
                              and o.org_acronym = p.org_acronym)
    then
        signal sqlstate '45000' set message_text = 'Researcher works
only in projects of his org!';
    end if;
end;//

/*check if a researcher works only in projects of his organization*/
/*drop trigger if exists researcher_org_update;*/
delimiter //
create trigger researcher_org_update after update on works_at
for each row
begin
    declare message_text varchar(50);

    if new.project_ID not in (
            select p.project_ID,
            from projects as p, organizations as o, researchers as r
            where new.researcher_ID = r.researcher_ID
                and r.org_acronym = o.org_acronym
                and r.org_acronym = p.org_acronym
                and o.org_acronym = p.org_acronym)
    then
        signal sqlstate '45000' set message_text = 'Researcher works
only in projects of his org!';
    end if;
end;//
```

Οι triggers χρησιμοποιήθηκαν για να ελέγξουν διάφορους περιορισμούς που πρέπει να ικανοποιούνται στην βάση.

- Οι triggers integrity και integrity_update εξασφαλίζουν ότι ένας ερευνητής που αξιολογεί κάποιο project δεν μπορεί να εργάζεται στο ίδιο project.
- Οι triggers integrity_projects και integrity_projects_update εξασφαλίζουν ότι σε ένα project δεν μπορεί να ταυτίζεται ο επιστημονικός υπεύθυνος με τον ερευνητή που αξιολογεί το project.
- Οι triggers researcher_org και researcher_org_update εξασφαλίζουν ότι ένας ερευνητής μπορεί να εργάζεται μόνο σε έργα τα οποία διαχειρίζονται από τον οργανισμό στον οποίο ανήκει.

DML Scripts

Queries

Το query 3.1 δέχεται μεγάλη παραμετρικοποίηση οπότε εκτελείται απευθείας στο αρχείο routes.py. Δίνονται ενδεικτικά τα strings που εκτελούνται:

```
"select r.first_name, r.last_name from works_at as w, researchers as
r, projects as p where  w.researcher_ID = r.researcher_ID and
p.project_ID = w.project_ID and p.title = \""+pr+"\";"
```

```
"select title, program_ID from projects
"+where+sdate_cond+edate_cond+duration_cond+exec_id_cond+";"
```

Το query 3.2 εκτελείται με views τα οποία παρατίθενται στην παρακάτω ενότητα.

```
/*3.3*/
/* this query requires user input for the attribute sector. This is
an example with sector = 'C.Science'*/
select p.title
from projects as p, about as a
where p.project_ID = a.project_ID and (datediff(p.end_date,
current_date) > 0) and (datediff(current_date, p.start_date) > 0) and
a.sector = 'C.Science';
```

```
select distinct r.first_name, r.last_name
from researchers as r, works_at as w, about as a, projects as p
where r.researcher_ID = w.researcher_ID and
      p.project_ID = w.project_ID and
      p.project_ID = a.project_ID and
      a.project_ID = w.project_ID and (((datediff(p.end_date,
current_date) > 0) and (datediff(current_date, p.start_date) > 0))
or((datediff(current_date, p.end_date) < 365) and
(datediff(current_date, p.end_date) > 0))) and
      a.sector = 'C.Science';
```

```
/*3.4*/
with help(project_ID, syear, org_acronym) as
(select project_ID,  extract(year from start_date), org_acronym
from projects),

help2(org_acronym, syear, amount) as
(select org_acronym, syear, count(project_ID)  from help
group by org_acronym, syear
order by org_acronym)

select distinct h1.org_acronym from help2 as h1, help2 as h2
where h1.org_acronym = h2.org_acronym and h1.syear = h2.syear + 1 and
h1.amount = h2.amount and h1.amount>9;

/*3.5*/
with help(sector1, sector2, value) as
(select a1.sector, a2.sector, count(*) as "count"
from about as a1, about as a2
where a1.project_ID = a2.project_ID and a1.sector <> a2.sector
        and a1.sector < a2.sector
group by a1.sector, a2.sector
order by count*(-1) limit 3)
select sector1, sector2 from help;



/*3.6*/
with help(researcher_ID, last_name, first_name, value) as
(select r.researcher_ID, r.last_name, r.first_name,
count(w.project_ID)
from researchers as r, works_at as w, projects as p
where r.researcher_ID = w.researcher_ID and p.project_ID =
w.project_ID
            and (datediff(p.end_date, current_date) > 0) and
(datediff(current_date, p.start_date) > 0) and
(datediff(current_date, r.birth_date) < 14600)
group by r.last_name, r.first_name, r.researcher_ID),
help2(value) as
(select MAX(value) from help)
select h1.first_name, h1.last_name, h1.value
from help as h1, help2 as h2
where h1.value = h2.value;



/*3.7*/
with help(v1, v2, first_name, last_name, company_name, amount) as
(select p.project_ID, e.exec_ID, e.first_name, e.last_name, o.name,
p.amount
from executives as e, organizations as o, projects as p
where p.exec_ID = e.exec_ID and p.org_acronym = o.org_acronym
        and o.category = 'Comp'
order by amount*(-1))
select first_name, last_name, company_name, amount
from help
where amount in (select max(amount)
                 from help
                 group by v2) limit 5;
```

```
/*3.8*/
select r.first_name, r.last_name, count(w.project_ID) as "proj_num"
from researchers as r, works_at as w
where r.researcher_ID = w.researcher_ID and
      w.project_ID not in (select p.project_ID
                           from projects as p, reports as rep
                           where p.project_ID = rep.project_ID)
group by r.first_name, r.last_name, r.researcher_ID
having proj_num >= 5
order by r.researcher_ID;
```

Views

```
/*3.2*/
/*find the projects(title) that each researcher works at*/
create view v1 as
select r.first_name, r.last_name, p.title
from researchers as r, projects as p, works_at as w
       where w.researcher_ID = r.researcher_ID and
       w.project_ID = p.project_ID
       order by r.last_name;

/*find all the projects(title) for each department of ELIDEK*/
create view v2 as
select prg.department, p.title
from projects as p, programs as prg
where p.program_ID = prg.program_ID
order by prg.department;
```

Ζητούμενο 2.3

Βήματα εγκατάστασης της εφαρμογής και βιβλιοθηκών που απαιτούνται

1. Εγκατάσταση του mysql-server εισάγοντας στην γραμμή εντολών τις εντολές
   sudo apt-get update
   sudo apt-get install mysql-server

2. Αφού ολοκληρωθεί η εγκατάσταση, αρχικά εκκινούμε την mysql με την εντολή
   sudo systemctl start mysql

   Έχοντας εκτελέσει την παραπάνω εντολή εισάγουμε στην γραμμή εντολών της mysql την εντολή

   sudo mysql

   Δίνουμε στην χρήστη root δικαίωμα πρόσβασης με κωδικό, με την εντολή

   ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'password';

   όπου αντικαθιστούμε το πεδίο **password** με τον κωδικό που επιθυμούμε. Στη συνέχεια πατώντας Ctrl-D βγαίνουμε από την mysql και για να επιβεβαιώσουμε την δυνατότητα του χρήστη root να εισέρχεται στη βάση εκτελούμε την εντολή

   mysql -u root -p

   εισάγοντας τον κωδικό μας.

3. Αφού έχουμε εγκαταστήσει επιτυχώς τον mysql-server κατεβάζουμε τον DBMS client MySQL Workbench ακολουθώντας το ακόλουθο link
   https://dev.mysql.com/downloads/workbench/
   και επιλέγουμε διανομή Ubuntu Linux. Στην συνέχεια ανάλογα με την έκδοση Ubuntu που έχουμε, κατεβάζουμε το ανάλογο πακέτο. Η εργασία μας έγινε σε Ubuntu 20.04 LTS, συνεπώς κατεβάζουμε το πακέτο **Ubuntu Linux 20.04 (x86, 64-bit), DEB Package.** Αφού γίνει η λήψη του πακέτου μεταβαίνουμε στο directory Downloads, εντοπίζουμε το αρχείο που έχει κατέβει και εισάγουμε την εντολή
   sudo dpkg -i mysql-workbench-community_8.0.29-1ubuntu20.04_amd64.deb
   Σε περίπτωση που ο χρήστης χρησιμοποιεί Ubuntu 22.04 LTS η εγκατάσταση του workbench γίνεται με την εντολή
   sudo snap install mysql-workbench-community

4. Αφού εγκατασταθεί το workbench, πρέπει να δημιουργήσουμε μία σύνδεση με την mysql. Οπότε ανοίγουμε το workbench και στο πεδίο MySQL Connections επιλέγουμε το + ώστε να φτιάξουμε την σύνδεση. Δίνουμε το όνομα elidek σε αυτή και βάζουμε σαν κωδικό τον ίδιο με αυτό του χρήστη root της βάσης δεδομένων μας.

5. Μεταβαίνουμε στο link που δίνεται στην επόμενη ενότητα, το οποίο περιέχει το git repo μας. Κατεβάζουμε ως έναν zip φάκελο τα αρχεία που περιέχονται στο repo.

6. Αρχικά εκτελούμε το script **Elidek_schema_v2.sql.**

Στη συνέχεια, εκτελούμε τα αρχεία **triggers.sql** και **indices.sql**, ώστε να υλοποιηθούν αντίστοιχα οι σκανδαλιστές και οι δείκτες στην βάση. Και τα τρία αυτά αρχεία βρίσκονται στον φάκελο **DDLScripts** του repo μας.
Στη συνέχεια, από τον φάκελο **DMLScripts** εκτελούμε το αρχείο **View.sql** για την υλοποίηση των όψεων.

7. Τώρα πρέπει να γίνει η εισαγωγή των δεδομένων στη βάση μας. Χρησιμοποιώντας το feature Table Import Data Wizard του MySQL Workbench εισάγουμε από το αρχείο INSERTS_CSV τα αντίστοιχα δεδομένα στους αντίστοιχους πίνακες. Λόγω της ύπαρξης foreign keys, τα αρχεία CSV πρέπει να εισαχθούν με την ακόλουθη σειρά:
   20organizations
   40phones
   30programs
   30executives
   fields
   100researchers
   50projects
   10reports
   100about
   208works_at

8. Για την εκτέλεση της εφαρμογής απαιτείται η χρήση της γλώσσας Python και ορισμένων βιβλιοθηκών της. Συνεπώς πρέπει να εκτελεστούν οι ακόλουθες εντολές

```
sudo apt-get install python3
sudo apt-get install python3-pip
pip install Flask
sudo apt-get install libmysqlclient-dev
pip install Flask-MySQLdb
pip install Flask-WTF
sudo apt-get install python3-flask
```

9. Μετά την εγκατάσταση των ανωτέρω ανοίγουμε σε κάποιον editor το αρχείο **__init__.py** το οποίο βρίσκεται στο directory **Project/dbdemo** του repo μας.
   Συμπληρώνουμε το αρχείο αυτό με τον εξής τρόπο:

```
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = '****'
app.config['MYSQL_DB'] = 'elidek'
```

   όπου στο πεδίο MYSQL_PASWWORD συμπληρώνουμε τον κωδικό που έχουμε δώσει στον χρήστη root της MySQL.

10. Για να τρέξει η εφαρμογή, μεταβαίνουμε από την γραμμή εντολών στο directory Project του repo και εκτελούμε τις ακόλουθες εντολές:
    export FLASK_APP=run.py

```
flask run
```

Μεταβαίνουμε στην σελίδα http://127.0.0.1:5000 ώστε να χειριστούμε την εφαρμογή.

Ζητούμενο 2.4

Git – Repo

https://github.com/JimV4/Elidek