Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Εξάμηνο 6ο

Μάθημα: Βάσεις Δεδομένων

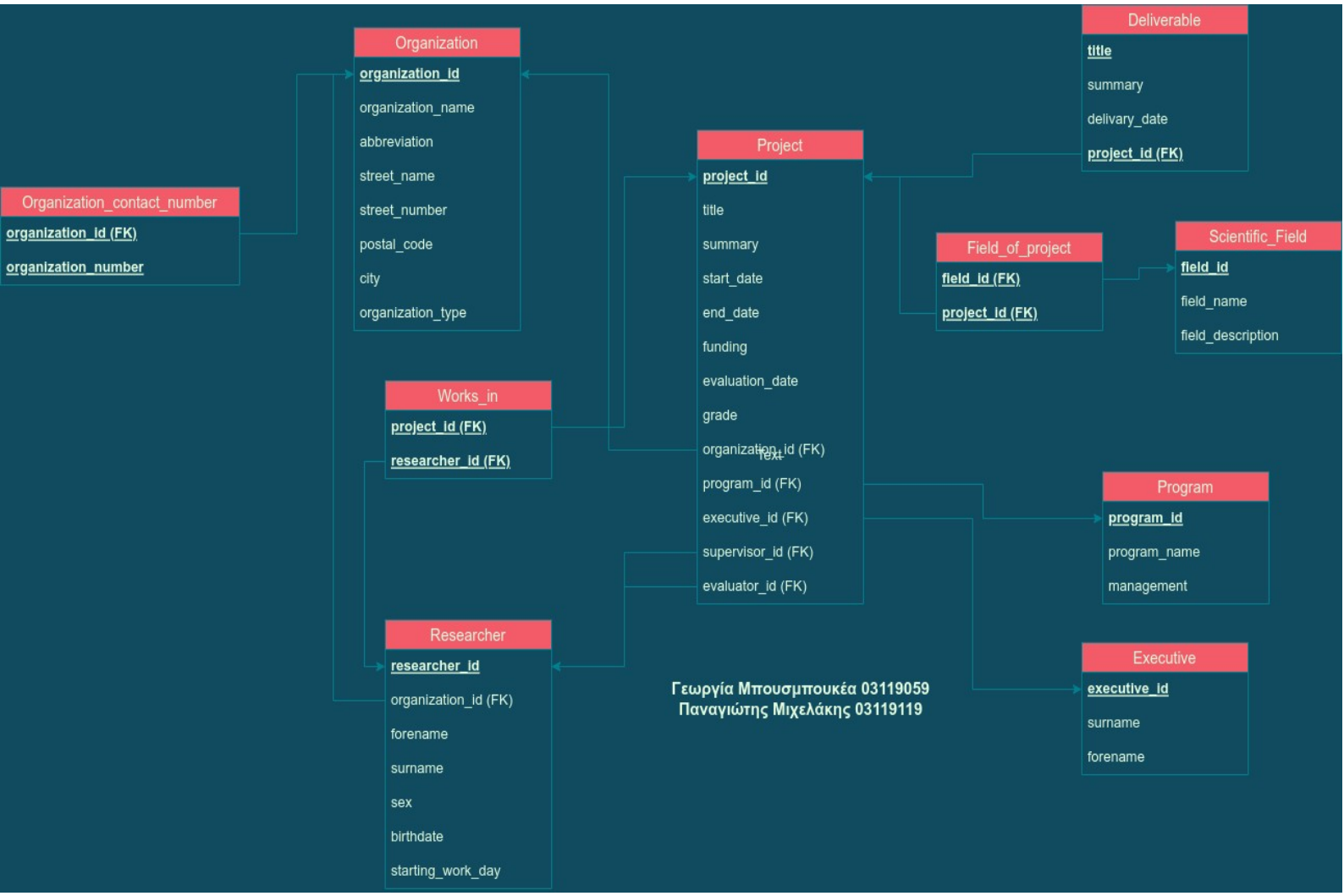
Μέλη Ομάδας: Γεωργία Μπουσμπουκέα 03119059
Παναγιώτης Μιχελάκης 03119119

Έτος: 2021 - 2022

Εξαμηνιαία Εργασία ΒΔ

2.1)

Το σχεσιακό διάγραμμα φαίνεται παρακάτω:



Οι κανόνες που χρησιμοποιήσαμε για την κατασκευή του σχεσιακού διαγράμματος από το ER-διάγραμμα είναι οι εξής:

- Για κάθε οντότητα (entity) δημιουργούμε ένα table το οποίο περιέχει όλα τα simple attributes αυτής.
- Για κάθε αδύναμη οντότητα (weak entity) δημιουργούμε ένα table το οποίο περιέχει όλα τα simple attributes αυτής.
- Όταν έχουμε μία δυαδική 1:1 σχέση στο ER-διάγραμμα, τότε στο σχεσιακό διάγραμμα βάζουμε το primary key της μίας οντότητας ως foreign key στο table της άλλης. Ευνοείται η ολική συμμετοχή, ήτοι, το foreign key μπαίνει στο table της οντότητας με ολική συμμετοχή (total participation) στη σχέση.
- Όταν έχουμε μία δυαδική 1:N σχέση στο ER-διάγραμμα, τότε στο σχεσιακό διάγραμμα βάζουμε το primary key της "1" οντότητας ως foreign key της "N" οντότητας.
- Όταν έχουμε μία δυαδική M:N σχέση στο ER-διάγραμμα, τότε στο σχεσιακό διάγραμμα δημιουργούμε ένα ξεχωριστό table του οποίου το primary key είναι ο συνδυασμός των primary keys των δύο οντοτήτων που συμμετέχουν στη σχέση. Το table αυτό περιέχει, επίσης, οποιαδήποτε attribute της ίδιας της σχέσης που εμφανίζονται στο ER-διάγραμμα.
- Οποιαδήποτε attributes δυαδικών σχέσεων της μορφής 1:1 ή 1:N μπήκαν ως attributes στα tables των οντοτήτων με ολική συμμετοχή (total participation) στη σχέση αυτή.

Σχετικά με τα ευρετήρια (indexes) η MySQL δημιουργεί από μόνη της ευρετήρια για όλα τα Primary Keys. Επίσης, για εκδόσεις της MySQL μεγαλύτερες της 5.5, όπου η InnoDB είναι η default storage engine, δημιουργούνται αυτόματα ευρετήρια και για όλα τα foreign keys. Αυτό ισχύει και στη δική μας περίπτωση.

```
panos@panos-Inspiron-3580:~$ mysql -V
mysql  Ver 14.14 Distrib 5.7.38, for Linux (x86_64) using  EditLine wrapper
panos@panos-Inspiron-3580:~$
```

Εγκατεστημένη έκδοση MySQL

Ελέγχουμε αν είναι default storage engine η InnoDB:

```
SELECT SUPPORT FROM INFORMATION_SCHEMA.ENGINES WHERE ENGINE = 'InnoDB';
```

Output:

#	SUPPORT
1	DEFAULT

Έτσι, λοιπόν, δεν χρειάζεται να δημιουργήσουμε ευρετήρια για τα primary και foreign keys της βάσης μας (κάτι το οποίο θα ήταν απαραίτητο έτσι ώστε να επιταχυνθούν οι εντολές JOIN).

Εξετάζοντας τα ζητούμενα Queries καταλήγουμε στη δημιουργία ευρετηρίων για τα εξής columns:

- Project.start_date
- Project.end_date
- Researcher.birthdate

Τα συγκεκριμένα attributes εμφανίζονται στα WHERE clauses διαφόρων από των Queries που γράψαμε.

Σημειώνεται πως η μόνη αλλαγή που έγινε στο υποβληθέν ER-διάγραμμα είναι το γεγονός ότι προσθέσαμε το attribute program_name στην οντότητα Program.

2.2)

Ας δούμε τώρα αναλυτικότερα πως ακριβώς υλοποιήθηκαν στην SQL όλα τα παραπάνω.

Query για τη δημιουργία της βάσης:

```
CREATE DATABASE elidek;
USE elidek;
```

Δημιουργούμε τη βάση μας υπό το όνομα “elidek” και την επιλέγουμε προς επεξεργασία σε περίπτωση που υπάρχουν και άλλες βάσεις στον server μας.

Query για τη δημιουργία του πίνακα Executive:

```
CREATE TABLE Executive(
    executive_id INT UNSIGNED AUTO_INCREMENT,
    surname VARCHAR(20) NOT NULL,
    forename VARCHAR(15) NOT NULL,
    PRIMARY KEY(executive_id)

);
```

Εδώ το executive_id είναι Primary Key, και αυτό συμβαίνει, διότι είναι το attribute εκείνο το οποίο είναι μοναδικό και ξεχωριστό για κάθε στέλεχος.

Query για τη δημιουργία του πίνακα Scientific_field:

```
CREATE TABLE Scientific_field(
    field_id INT UNSIGNED AUTO_INCREMENT,
    field_name VARCHAR(30) NOT NULL,
    field_description VARCHAR(100) NOT NULL,
    PRIMARY KEY(field_id)
);
```

Εδώ το field_id είναι Primary Key, και αυτό συμβαίνει, διότι είναι το attribute εκείνο το οποίο είναι μοναδικό και ξεχωριστό για κάθε επιστημονικό πεδίο.

Query για τη δημιουργία του πίνακα Program:

```
CREATE TABLE Program(
    program_id INT UNSIGNED AUTO_INCREMENT,
    program_name VARCHAR(30) NOT NULL,
    management  VARCHAR(25) NOT NULL,
    PRIMARY KEY(program_id)

);
```

Εδώ το program_id είναι Primary Key, και αυτό συμβαίνει, διότι είναι το attribute εκείνο το οποίο είναι μοναδικό και ξεχωριστό για κάθε πρόγραμμα.

Query για τη δημιουργία του πίνακα Organization:

```
CREATE TABLE Organization(
    organization_id INT UNSIGNED AUTO_INCREMENT,
    organization_name VARCHAR(25) NOT NULL,
    abbreviation VARCHAR(7) NOT NULL,
    street_name VARCHAR(15) NOT NULL,
    street_number INT NOT NULL,
    postal_code INT NOT NULL,
    city VARCHAR(20) NOT NULL,
    organization_type ENUM ('Company','University', 'Research Facility') NOT NULL,
    PRIMARY KEY(organization_id)

);
```

Εδώ το organization_id είναι Primary Key, και αυτό συμβαίνει, διότι είναι το attribute εκείνο το οποίο είναι μοναδικό και ξεχωριστό για κάθε οργανισμό.

Query για τη δημιουργία του πίνακα Organization_contact_number:

```
CREATE TABLE Organization_contact_number(
    organization_id INT UNSIGNED ,
    organization_number BIGINT UNSIGNED,
    PRIMARY KEY(organization_id, organization_number),
    CONSTRAINT fk_contact_organization FOREIGN KEY(organization_id) REFERENCES Organization(organization_id) ON DELETE CASCADE
);
```

Εδώ το organization_id και το organization_number αποτελούν αμφότερα το Primary Key, και αυτό συμβαίνει, διότι είναι ο συνδυασμός των attributes εκείνος ο οποίος είναι μοναδικός και ξεχωριστός για κάθε αριθμό οργανισμού. Επίσης, στον πίνακα Organization_contact_number το organization_id είναι foreign key για τον πίνακα Organization και αντιστοιχίζει τους δύο πίνακες σε σχέση πολλά(Organization_contact_number) προς ένα(Organization).

Query για τη δημιουργία του πίνακα Researcher:

```
CREATE TABLE Researcher(
    researcher_id INT UNSIGNED AUTO_INCREMENT,
    organization_id INT UNSIGNED,# NOT NULL,
    forename VARCHAR(20) NOT NULL,
    surname VARCHAR(20) NOT NULL,
    sex ENUM('Male', 'Female', 'Non-Binary') NOT NULL,
    birthdate DATE NOT NULL,
    starting_work_day DATE NOT NULL,
    PRIMARY KEY(researcher_id),
    CONSTRAINT fk_researcher_organization FOREIGN KEY(organization_id) REFERENCES Organization(organization_id) ON DELETE SET NULL #CASCADE
);
```

Εδώ το researcher_id είναι Primary Key, και αυτό συμβαίνει, διότι είναι το attribute εκείνο το οποίο είναι μοναδικό και ξεχωριστό για κάθε ερευνητή. Επίσης, στον πίνακα Researcher το organization_id είναι foreign key για τον πίνακα Organization και αντιστοιχίζει τους δύο πίνακες σε σχέση πολλά(Researcher) προς ένα(Organization).

Query για τη δημιουργία του πίνακα Project:

```
CREATE TABLE Project(
    project_id INT UNSIGNED AUTO_INCREMENT,
    title VARCHAR(30) NOT NULL,
    summary VARCHAR(200),
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    funding DOUBLE NOT NULL,
    evaluation_date DATE NOT NULL,
    grade FLOAT,
    organization_id INT UNSIGNED,# NOT NULL,
    program_id INT UNSIGNED,# NOT NULL,
    executive_id INT UNSIGNED,# NOT NULL,
    supervisor_id INT UNSIGNED ,#NOT NULL,
    evaluator_id INT UNSIGNED,# NOT NULL,
    PRIMARY KEY(project_id),
    CONSTRAINT fk_project_program FOREIGN KEY(program_id) REFERENCES Program(program_id) ON DELETE SET NULL, # CASCADE,
    CONSTRAINT fk_project_executive FOREIGN KEY(executive_id) REFERENCES Executive(executive_id) ON DELETE SET NULL, # CASCADE,
    CONSTRAINT fk_project_evaluator FOREIGN KEY(evaluator_id) REFERENCES Researcher(researcher_id) ON DELETE SET NULL, # CASCADE,
    CONSTRAINT fk_project_supervisor FOREIGN KEY(supervisor_id) REFERENCES Researcher(researcher_id) ON DELETE SET NULL, #CASCADE,
    CONSTRAINT fk_project_organization FOREIGN KEY(organization_id) REFERENCES Organization(organization_id) ON DELETE SET NULL, #CASCADE,
    CONSTRAINT check_project_date CHECK(DATEDIFF(end_date, start_date) >= 365 AND DATEDIFF(end_date, start_date) <= 1460)
);
```

Εδώ το project_id είναι Primary Key, και αυτό συμβαίνει, διότι είναι το attribute εκείνο το οποίο είναι μοναδικό και ξεχωριστό για κάθε έργο. Επίσης, στον πίνακα Project τα organization_id, program_id, executive_id, supervisor_id, evaluator_id είναι foreign keys για τους πίνακες Organization, Program, Executive, Supervisor, Evaluator αντίστοιχα και αντιστοιχίζουν τους πίνακες σε σχέση πολλά(Project) προς ένα(Λοιποί Πίνακες).

Query για τη δημιουργία του πίνακα Field_of_Project:

```
CREATE TABLE Field_of_Project(
    project_id INT UNSIGNED,
    field_id INT UNSIGNED,
    PRIMARY KEY(project_id, field_id),
    CONSTRAINT fk_field_project FOREIGN KEY(project_id) REFERENCES Project(project_id) ON DELETE CASCADE,
    CONSTRAINT fk_field_scientific FOREIGN KEY(field_id) REFERENCES Scientific_field(field_id)  ON DELETE CASCADE

);
```

Εδώ το project_id και το field_id αποτελούν αμφότερα το Primary Key, και αυτό συμβαίνει, διότι είναι ο συνδυασμός των attributes εκείνος ο οποίος είναι μοναδικός και ξεχωριστός για κάθε πεδίο έργου. Επίσης, στον πίνακα Field_of_Project το project_id είναι foreign key για τον πίνακα Project και αντιστοιχίζει τους δύο πίνακες σε σχέση πολλά(Scientific_field) προς πολλά(Project).
Παρόμοια, στον πίνακα Field_of_Project το field_id είναι foreign key για τον πίνακα Scientific_field και αντιστοιχίζει τους δύο πίνακες σε σχέση πολλά(Project) προς πολλά(Scientific_field).

Query για τη δημιουργία του πίνακα Deliverable:

```
CREATE TABLE Deliverable(
    title VARCHAR(35),
    summary VARCHAR(100) NOT NULL,
    delivery_date DATE NOT NULL,
    project_id INT UNSIGNED,
    PRIMARY KEY(project_id, title),
    CONSTRAINT fk_deliverable_project FOREIGN KEY(project_id) REFERENCES Project(project_id) ON DELETE CASCADE
);
```

Εδώ το title και το project_id αποτελούν αμφότερα το Primary Key, και αυτό συμβαίνει, διότι είναι ο συνδυασμός των attributes εκείνος ο οποίος είναι μοναδικός και ξεχωριστός για κάθε παραδοτέο. Επίσης, στον πίνακα Deliverable το project_id είναι foreign key για τον πίνακα Project και αντιστοιχίζει τους δύο πίνακες σε σχέση πολλά(Deliverable) προς ένα(Project).

Query για τη δημιουργία του πίνακα Works_in:

```
CREATE TABLE Works_in(
    project_id INT UNSIGNED,
    researcher_id INT UNSIGNED ,
    PRIMARY KEY(project_id, researcher_id),
    CONSTRAINT fk_works_in_project FOREIGN KEY(project_id) REFERENCES Project(project_id) ON DELETE CASCADE,
    CONSTRAINT fk_works_in_researcher FOREIGN KEY(researcher_id) REFERENCES Researcher(researcher_id) ON DELETE CASCADE

);
```

Εδώ το project_id και το researcher_id αποτελούν αμφότερα το Primary Key, και αυτό συμβαίνει, διότι είναι ο συνδυασμός των attributes εκείνος ο οποίος είναι μοναδικός και ξεχωριστός για κάθε “Works_in”. Επίσης, στον πίνακα Works_in το project_id είναι foreign key για τον πίνακα Project και αντιστοιχίζει τους δύο πίνακες σε σχέση πολλά(Researcher) προς πολλά(Project).
Παρόμοια, στον πίνακα Works_in είναι foreign key για τον πίνακα Researcher και αντιστοιχίζει τους δύο πίνακες σε σχέση πολλά(Project) προς πολλά(Researcher).

Query για τη δημιουργία των ευρετηρίων:

```
CREATE INDEX project_start_date_idx
ON Project (start_date);

CREATE INDEX project_end_date_idx
ON Project (end_date);

CREATE INDEX researcher_birthdate_idx
ON Researcher (birthdate);
```

Δεν απαιτείται κάποιος σχολιασμός.

Query για τη δημιουργία του Trigger:

```
DELIMITER $$
CREATE TRIGGER res_trigg
BEFORE INSERT ON Works_in
FOR EACH ROW
BEGIN
IF EXISTS(SELECT
Project.project_id, Works_in.researcher_id FROM Project JOIN Works_in ON NEW.project_id = Project.project_id AND NEW.researcher_id = Project.evaluator_id
)
THEN SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'An evaluator cannot be working in the project he is evaluating';
END IF;
END
DELIMITER ;
```

Το trigger αυτό ενεργοποιείται όταν γίνεται εισαγωγή στον πίνακα Works_in ενός ερευνητή ο οποίος είναι και αξιολογητής του έργου με το οποίο πάει να συνδεθεί. Το trigger απορρίπτει την εισαγωγή αυτή και εμφανίζει αναγνωριστικό πληροφοριακό μήνυμα.

Παρακάτω παραθέτουμε τις απαντήσεις μας για τα ζητούμενα Queries του ερωτήματος 3 της εργασίας.

3.1)

```
/* All the programs*/
SELECT * FROM Program;

/* All projects based on multiple criteria */

/*Date if it only wants to retrieve project  from date*/

SELECT Project.project_id, Project.title FROM Project WHERE Project.start_date = '2022-01-31';

/*Duration if it only wants to retrieve project from duration */

SELECT Project.project_id, Project.title FROM Project WHERE DATEDIFF(Project.end_date, Project.start_date) <= 730;


/*Executive*/

SELECT Project.project_id, Project.title FROM Project WHERE Project.executive_id IN (SELECT Executive.executive_id FROM Executive WHERE Executive.executive_id = 2);
```

Θεωρήσαμε πως το ερώτημα 3.1) αναφέρεται σε πολλά ανεξάρτητα Queries τα οποία υλοποιήσαμε παραπάνω.

Στο πρώτο εξ'αυτών επιλέγουμε απλά όλα τα προγράμματα που υπάρχουν στη βάση μας.

Στο δεύτερο επιλέγουμε όλα τα υπάρχοντα έργα βάσει της ημερομηνίας έναρξής του.

Στο τρίτο επιλέγουμε όλα τα υπάρχοντα έργα βάσει της διάρκειάς τους.

Στο τέταρτο επιλέγουμε όλα τα υπάρχοντας έργα βάσει το στέλεχος που τα επιβλέπει. Σημειώνουμε ότι στην υλοποίηση στην python παραμετρικοποιούμε το “Executive.executive_id” του παραπάνω query όπου εδώ το έχουμε βάλει ίσο με 2.

3.2)

```
#first
create or replace view projects_per_researcher as
select r.researcher_id, r.surname, r.forename, p.project_id, p.title
from Researcher as r
inner join Works_in as w on r.researcher_id=w.researcher_id
inner join Project as p on p.project_id=w.project_id
order by r.researcher_id;

select * from projects_per_researcher;
```

Η πρώτη όψη που υλοποιούμε είναι αυτή των έργων ανά ερευνητή.

```
#second
create or replace view projects_per_organization as
select o.organization_id, o.organization_name, p.project_id, p.title
from Organization as o natural join Project as p
order by organization_id;

select  * from projects_per_organization;
```

Η δεύτερη όψη είναι αυτή των έργων ανά οργανισμό.

3.3)

```
create or replace view famous_field as
select p.project_id, p.title, f.field_id, f.field_name
from Project as p
inner join Field_of_Project as fp on fp.project_id=p.project_id
inner join Scientific_field as f on f.field_id=fp.field_id
where datediff(curdate(), p.start_date)>0 and datediff(p.end_date, curdate())>0 AND f.field_name = "Mathematics";


create or replace view researcher_on_project as
select r.researcher_id, r.surname, r.forename, p.project_id, p.title
from Researcher as r
inner join Works_in as w on w.researcher_id= r.researcher_id
inner join Project as p on p.project_id=w.project_id;



select ff.field_id, ff.field_name, ff.project_id, ff.title, rp.researcher_id, rp.surname, rp.forename
from famous_field as ff
inner join researcher_on_project as rp on ff.project_id=rp.project_id
order by ff.field_id, rp.project_id;
```

Δημιουργούμε δύο views, τις famous_field και researcher_on_project τις οποίες και χρησιμοποιούμε στο τελικό select.

3.4)

```
CREATE OR REPLACE VIEW projects_per_organization_per_year (organization_id, organization_name, projects, yearr)
AS
SELECT org.organization_id, org.organization_name, count(*), YEAR(p.start_date) as yearr FROM Organization org
INNER JOIN Project p
ON org.organization_id = p.organization_id
GROUP BY org.organization_id, yearr;

SELECT i.organization_id, i.organization_name, i.yearr AS first_year, dupl.yearr AS second_year, i.projects AS projects_each_year
FROM projects_per_organization_per_year i, projects_per_organization_per_year dupl
WHERE i.organization_id = dupl.organization_id AND i.yearr = dupl.yearr -1 AND i.projects = dupl.projects AND i.projects >= 10;
```

Στο συγκεκριμένο Query η λογική πίσω απτην υλοποίησή μας είναι ότι μπορούμε να "λουπάρουμε" μία σχέση με ένα Where clause και ένα καρτεσιανό γινόμενο με τον εαυτό της. Με τον τρόπο αυτόν, υλοποιήσαμε τη συνθήκη "δύο συνεχόμενη έτη".

3.5)

```
create or replace view project_with_field as
select p.project_id, p.title, f.field_id, f.field_name
from Project as p
inner join Field_of_Project as fp on fp.project_id=p.project_id
inner join Scientific_field as f on f.field_id=fp.field_id;


create or replace view project_with_pair as
select pf1.project_id, pf1.title, pf1.field_name as f1_name, pf2.field_name as f2_name
from project_with_field as pf1
cross join project_with_field as pf2
where pf1.project_id =pf2.project_id and pf1.field_id >pf2.field_id
order by pf1.project_id;


select count(project_id) as val, f1_name, f2_name
from project_with_pair
group by f1_name, f2_name
order by val desc
limit 3;
```

Όπως και πριν δημιουργούμε δύο επικουρικά views και τα χρησιμοποιούμε μετά στο SELECT clause.

3.6)

```
create or replace view young_Researcher as
select researcher_id, surname, forename
from Researcher
where datediff(curdate(), birthdate)<14640;

create or replace view active_Project as
select project_id, title
from Project
where datediff(curdate(), start_date)>0 and datediff(end_date, curdate())>0;

create or replace view young_Researcher_on_active_Project as
select yr.researcher_id, yr.surname, yr.forename, ap.project_id
from young_Researcher as yr
inner join Works_in as w on yr.researcher_id=w.researcher_id
inner join active_Project as ap on w.project_id=ap.project_id;


select researcher_id, surname, forename, active_projects
from active_projects_of_researcher
where active_projects=(select maxi from (select max(active_projects) as maxi from (select count(project_id) as active_projects  from young_Researcher_on_active_Project
group by researcher_id)
as active_projects_of_researcher )as maximum);
```

Στο ερώτημα αυτό δημιουργούμε τρία views. Στο πρώτο επιλέγουμε τους ερευνητές με ηλικία κάτων των 40 ετών. Στο δεύτερο επιλέγουμε τα ενεργά έργα. Στο τρίτο επιλέγουμε τους νέους ερευνητές που δουλέυουν στα προαναδερθέντα ενεργά έργα. Με το SELECT clause στο τέλος απαντάμε στο ερώτημα.

3.7)

```
SELECT Executive.forename, Executive.surname, Project.funding, Organization.organization_name FROM Project
JOIN Executive ON Project.executive_id = Executive.executive_id
JOIN Organization ON Organization.organization_id = Project.project_id
WHERE Organization.organization_type = 'Company' AND Executive.executive_id IN( SELECT Project.executive_id FROM Project WHERE Project.organization_id IN
(SELECT Organization.organization_id FROM Organization WHERE Organization.organization_type = 'Company')  ORDER BY funding DESC) AND Project.organization_id IN
(SELECT Organization.organization_id FROM Organization WHERE Organization.organization_type = 'Company');
```

Στο ερώτημα αυτό αντί για views κάναμε χρήση των Nested Select για να πάρουμε το ζητούμενο.

3.8)

```
create view Projects_without_deliverables as
select project_id, title
from Project
where project_id not in (select Project.project_id from Project join Deliverable on Project.project_id=Deliverable.project_id);

select * from Projects_without_deliverables;

select researcher_id, surname, forename, count(project_id) as proj_wtht_deliverables
from (select r.researcher_id, r.surname, r.forename, pwtd.project_id, pwtd.title
        from Researcher as r
        inner join Works_in as w on w.researcher_id=r.researcher_id
        inner join Projects_without_deliverables as pwtd on pwtd.project_id=w.project_id)as researcher_on_proj_wtht_deliv
group by researcher_id
having proj_wtht_deliverables>=5;

DROP VIEW Projects_without_deliverables;
```

Στο ερώτημα αυτό επιστρέψαμε στην χρήση των views έτσι ώστε να απαντήσουμε το ζητούμενο.

2.3)

Για την διαχείριση και ανάπτυξη της βάσης χρησιμοποιήθηκε ως RDBMS η MySQL, ενώ ως γραφικό εργαλείο επιλέχθηκε το MySQL Workbench . Για το στήσιμο του web server χρησιμοποιείται flask (Python) και για την σύνδεση μεταξύ βάσης και του server χρησιμοποιείται η βιβλιοθήκη flask-mysqldb. Για το UI χρησιμοποιήθηκε HTML,CSS και Java Script. Για την επικοινωνία μεταξύ backend και frontend χρησιμοποιείται μέθοδος Post.

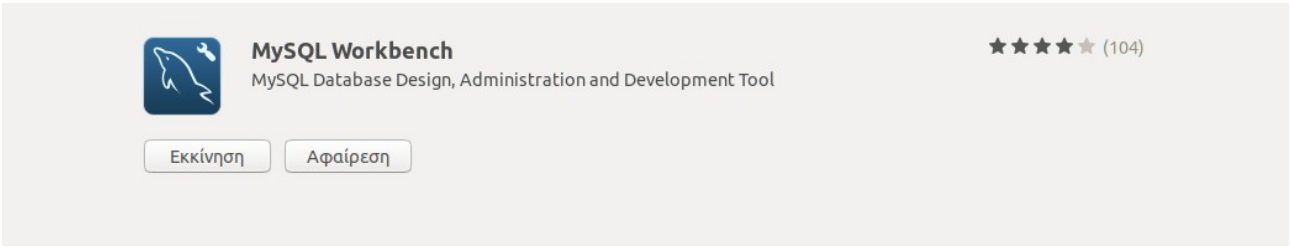
Version:

- *MySQL 5.7.38*
- *MySQL Workbench 6.3.8*
- *Python 3.6.9*
- *Flask 2.0.3*
- *Werkzeug 2.0.3*

Βήματα εγκατάστασης σε λογισμικό Linux – Ubuntu.

1. Απαιτείται η εγκατάσταση της MySQL. Εάν κρίνεται απαραίτητο μπορεί κανείς να εγκαταστήσει ολόκληρο το LAMP software stack το οποίο έρχεται εφοδιασμένο με την MySQL. Οδηγίες για το πώς είθισται να εκκινεί το Apache του πακέτου LAMP μπορούν να βρεθούν εδώ.

2. Εγκαθιστούμε το MySQL Workbench το οποίο μπορεί να βρεθεί στο bundle “Λογισμικό Ubuntu” .



3. Ανοίγουμε το MySQL Workbench, πατάμε το κουμπί “Local Instance” για να ξεκινήσουμε τη διαδικασία τεχνοδιαμόρφωσης. Στο αναδυόμενο παράθυρο πληκτρολογούμε τον κωδικό του root του MySQL server τον οποίο δημιουργήσαμε κατά τη διαδικασία εγκατάστασης της MySQL. Εάν επιθυμούμε να αλλάξουμε τον κωδικό αυτόν, ακολουθούμε τις οδηγίες που παρέχονται εδώ.

4. Αφού συνδεθούμε επιτυχώς, μπορούμε να πάμε να δημιουργήσουμε όλους τους πίνακες και τις απαραίτητες συνδέσεις μεταξύ τους. Στην εργασία έχουμε επισυνάψει τα SQL Scripts που περιέχουν μέσα τα Scripts για την δημιουργία των παραπάνω.

5. Στην συνέχεια, βάζουμε τα δεδομένα στην βάση. Για τη δημιουργία των dummy data που εισήχθησαν στη βάση χρησιμοποιήσαμε free online random data generator, Mockaroo. Τα SQL scripts που παρήχθησαν με τη βοήθεια αυτού του εργαλείου μεταφέρθηκαν και έτρεξαν στο MySQL Workbench. Σημειώνουμε ότι για να αποφεχθούν συγκρούσεις στη χρήση των foreign keys, τόσο τα tables κατά τη δημιουργία τους, όσο και τα δεδομένα κατά την εισαγωγή τους έπρεπε να εισαχθούν με συγκεκριμένη σειρά. Η σειρά αυτή είναι η ακόλουθη:

Executive
Scientific Field
Program
Organization
Organization_contact_number
Researcher
Project
Field_of_Project
Deliverable
Works_in

6. Αφού έχουν οριστεί οι πίνακες, έχει γίνει εισαγωγή των δεδομένων, έχουν δημιουργηθεί τα απαραίτητα ευρετήρια και τα views, προχωράμε στο στήσιμο του περιβάλλοντος πάνω στο οποίο τρέχει η εφαρμογή μας.

Εγκαθιστούμε το web framework Flask. Έπειτα, εγκαθιστούμε την παρακάτω βιβλιοθήκη μέσω κάποιου τερματικού.

$ pip3 install flask-mysqldb

Ύστερα ανοίγουμε τα αρχεία που επισυνάπτονται σε κάποιον editor και αντικαθιστούμε στο αρχείο app.py, τα credentials που ορίσαμε στο βήμα 2. Πιο συγκεκριμένα:

```
db = yaml.safe_load(open('db.yaml'))

app.config['MYSQL_HOST']=db['mysql_host']
app.config['MYSQL_USER']=db['mysql_user']
app.config['MYSQL_PASSWORD']= db['mysql_password']
app.config['MYSQL_DB']=db['mysql_db']
```

Για την ασφάλεια των credentials, αποφύγαμε να τα εισάγουμε με hardcode μέθοδο. Αντ'αυτού δημιουργήσαμε ένα .yaml αρχείο στο εσωτερικό του οποίου είσαγαμε τα προαναφερθέντα credentials υπό τη μορφή ενός dictionary. Για την εγκατάσταση του yaml module μπορεί κανείς να ανατρέξει εδώ.

7. Αφού τοποθετήσουμε, όλα τα αρχεία και σιγουρευτούμε ότι είναι στους σωστούς φακέλους, μπορούμε να τρέξουμε την εφαρμογή μας σε κάποιον localhost. Για να μπορούμε επιτυχώς να το κάνουμε αυτό, πρέπει να βρισκόμαστε στον ίδιο φάκελο που είναι και το αρχείο app.py (Projects) και εκτελούμε τις εντολές:

cd path_to_folder_of_project
source venv/bin/activate
export FLASK_APP=name_of_program.py
flask run

Το project θα τρέχει στον localhost που θα υποδειχθεί στο terminal, όταν εκτελεστούν οι παραπάνω εντολές.

2.4)

Ο σύνδεσμός για το git repo της εφαρμογής μας παρατίθεται παρακάτω:

https://github.com/georgia-bous/ELIDEK--DB2022