ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

ΕΞΑΜΗΝΙΑΙΑ ΕΡΓΑΣΙΑ 2022

ΟΜΑΔΑ 54

Ονοματεπώνυμο: Αντωνίου Ηλίας,el19705

Γιαννάκη Βίκτωρας,el19707

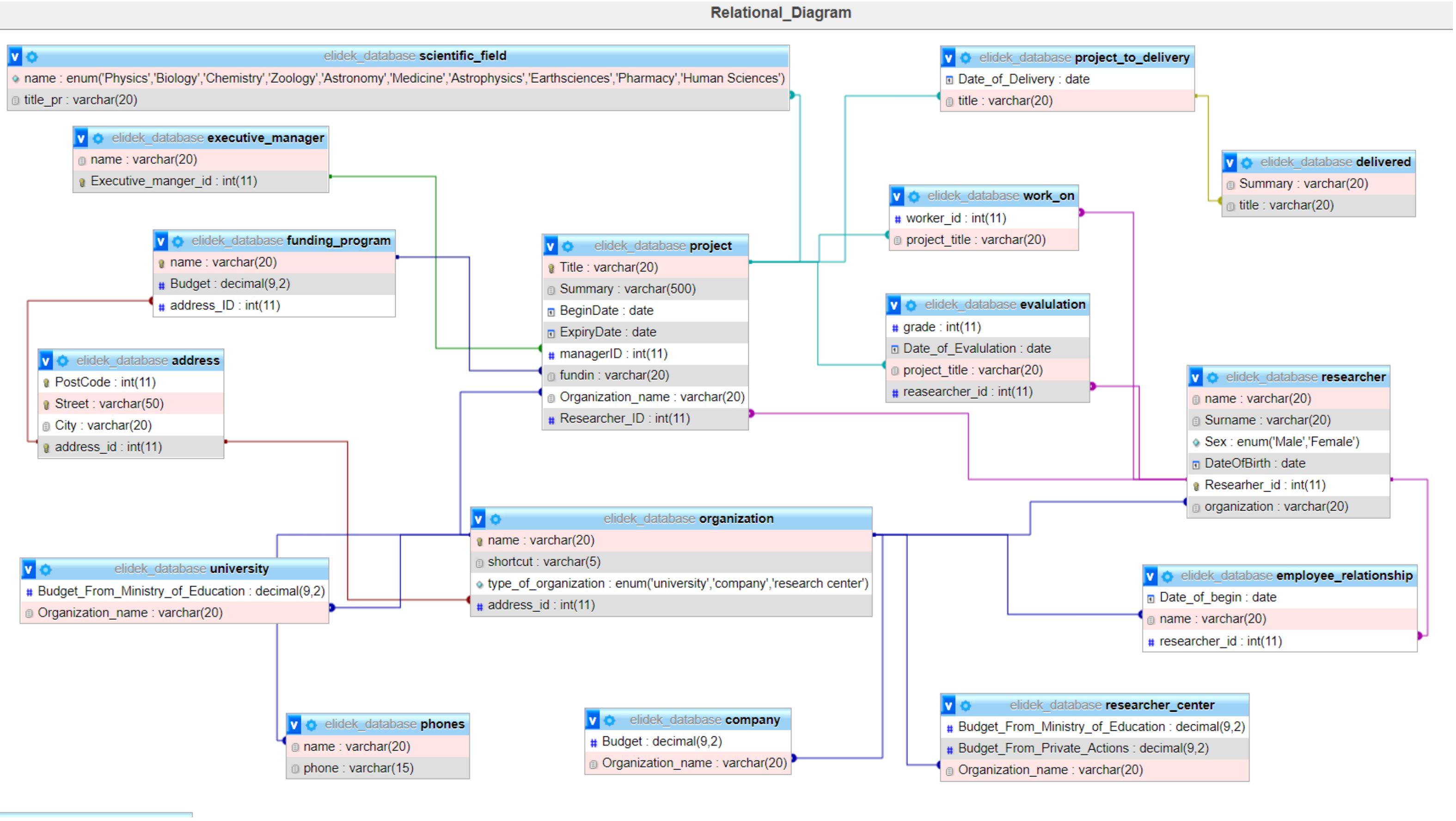
Παπαδόπουλος Βασίλης,el19701

LINK για το git repo της εφαρμογής μας :
https://github.com/vgiann06/DatabaseElidekntua2022

Η βάση δεδομένων που δημιουργήσαμε αποτελείται από τους παρακάτω πίνακες:

- address (address_id, PostCode, Street,City)
- company (Budget, Organization_name)
- delivered (Summay, title)
- employee_relationship (Date_of_begin ,name, researcher_id)
- evaluation (grade, Date_of_Evaluation, project_title ,researcher_id)
- executive_manager (name ,Executive_manger_id)
- funding_program (name, Budget, address_id)
- organization (name, shortcut, type_of_organization, address_id)
- phones (name, phone)
- project (Title, Summary, BeginDate, ExpiryDate, managerID, fundin, Organization_name, Researcher_ID)
- project_to_delivery (Date_of_Delivery, title)
- researcher (name, Surname, Sex, Date0fBirth, Researher_id, organization)
- researcher_center(Budget_From_Ministry_of_Education, Budget_From_Private_Actions, Organization_name)
- scientific_field (name, title_pr)
- university (Budget_From_Ministry_of_Education, , Organization_name)
- work_on (worker_id, project_title)

ΣΧΕΣΙΑΚΟ ΔΙΑΓΡΑΜΜΑ ΒΑΣΗΣ:



Relational_Diagram

elidek_database scientific_field
name : enum('Physics','Biology','Chemistry','Zoology','Astronomy','Medicine','Astrophysics','Earthsciences','Pharmacy','Human Sciences')
title_pr : varchar(20)

elidek_database project_to_delivery
Date_of_Delivery : date
title : varchar(20)

elidek_database executive_manager
name : varchar(20)
Executive_manger_id : int(11)

elidek_database delivered
Summary : varchar(20)
title : varchar(20)

elidek_database funding_program
name : varchar(20)
Budget : decimal(9,2)
address_ID : int(11)

elidek_database project
Title : varchar(20)
Summary : varchar(500)
BeginDate : date
ExpiryDate : date
managerID : int(11)
fundin : varchar(20)
Organization_name : varchar(20)
Researcher_ID : int(11)

elidek_database work_on
worker_id : int(11)
project_title : varchar(20)

elidek_database evalulation
grade : int(11)
Date_of_Evalulation : date
project_title : varchar(20)
reasearcher_id : int(11)

elidek_database address
PostCode : int(11)
Street : varchar(50)
City : varchar(20)
address_id : int(11)

elidek_database researcher
name : varchar(20)
Surname : varchar(20)
Sex : enum('Male','Female')
DateOfBirth : date
Researher_id : int(11)
organization : varchar(20)

elidek_database organization
name : varchar(20)
shortcut : varchar(5)
type_of_organization : enum('university','company','research center')
address_id : int(11)

elidek_database university
Budget_From_Ministry_of_Education : decimal(9,2)
Organization_name : varchar(20)

elidek_database employee_relationship
Date_of_begin : date
name : varchar(20)
researcher_id : int(11)

elidek_database phones
name : varchar(20)
phone : varchar(15)

elidek_database company
Budget : decimal(9,2)
Organization_name : varchar(20)

elidek_database researcher_center
Budget_From_Ministry_of_Education : decimal(9,2)
Budget_From_Private_Actions : decimal(9,2)
Organization_name : varchar(20)

- Το Σχεσιακό Διάγραμμα έγινε βάση του ER Diagram που μας δόθηκε.
- Στις σχέσεις 1-1/1-N χρησιμοποιήθηκε εξωτερικό κλειδί το οποίο προστέθηκε σε ένα από τους δύο πίνακες που ενώνονται.
- Στις σχέσεις N-N και προστέθηκε επιπλέον πίνακας ο οποίος περιέχει τα δύο κύρια κλειδιά των δύο πινάκων που ενώνονται.

- Στις σχέσεις που έχουν επιπλέον γνωρίσματα (π.χ: evaluation) προστέθηκε επιπλέον πίνακας.

Περιορισμοί Ακεραιότητας:

Ακεραιότητα Οντότητας:

Για να εξασφαλίσουμε αυτό τον περιορισμό, θέσαμε την τιμή NOTNULL σε όλα τα κύρια κλειδιά της κάθε οντότητας. Έτσι πετυχαίνουμε την ορθή λειτουργία της βάσης, αφού τα στοιχεία της κάθε οντότητας αναγνωρίζονται μοναδικά μέσω των κύριων κλειδιών.

Αναφορική Ακεραιότητα:

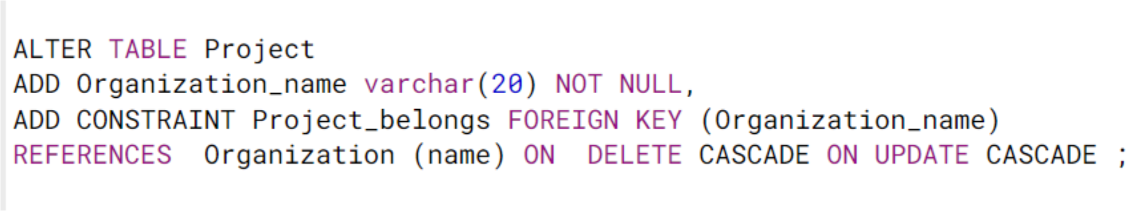
Χρησιμοποιώντας εξωτερικά κλειδιά καταφέρνουμε να σχετίσουμε τις οντότητες μεταξύ τους. Για παράδειγμα η οντότητα scientific_field έχει ως εξωτερικό κλειδί το title_pr που είναι το κύριο κλειδί του πίνακα project.

Πεδίο Τιμών:

Σε όλα τα γνωρίσματα των οντοτήτων έχουμε βάλει ρητούς περιορισμούς ανάλογα με την πραγματική ερμηνεία τους. Για παράδειγμα: στον πίνακα project το Title είναι VARCHAR(20), ενώ το BeginDate είναι DATE . Ακόμη, έχουν δημιουργηθεί και κάποια triggers τα οποία ελέγχουν τη μορφή των δεδομένων που εισάγουμε.

Constraints:

Χρησιμοποιούνται για να ενώσουν τους πίνακες μέσω των εξωτερικών κλειδιών. Για όλα τα coinstraints έγινε η επιλογή ON DELETE CASCADE ON UPDATE CASCADE, ούτως ώστε εάν γίνει κάποια τροποποίηση σε ένα πεδίο κάποιου "πατέρα", θα γίνει και στο αντίστοιχο του "πεδιού". Για παράδειγμα δίνετε η σχέση που συνδέει το Project με το Organization :



```
ALTER TABLE Project
ADD Organization_name varchar(20) NOT NULL,
ADD CONSTRAINT Project_belongs FOREIGN KEY (Organization_name)
REFERENCES  Organization (name) ON  DELETE CASCADE ON UPDATE CASCADE ;
```

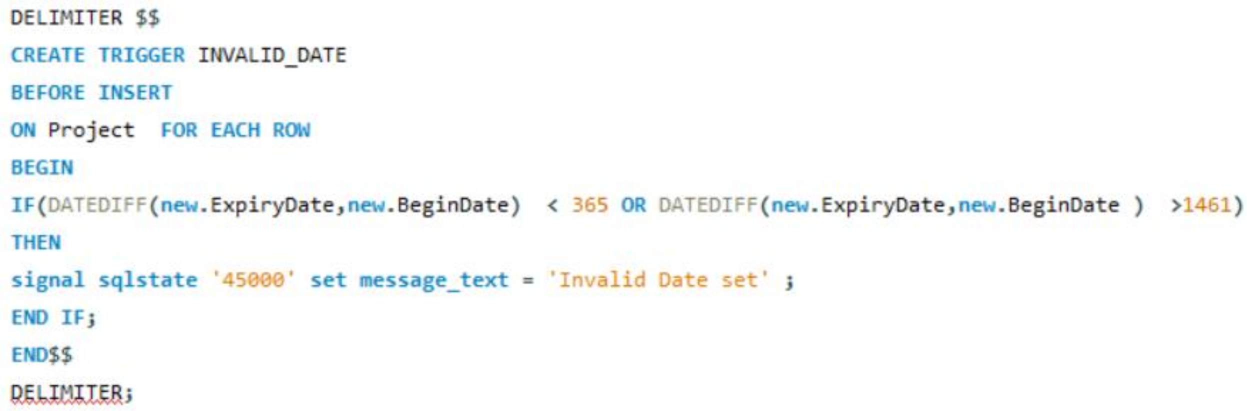
Παρατηρούμε πως εισάγεται στον πίνακα Project το Organization_name ως εξωτερικό κλειδί.

Triggers:

Έγινε χρήση των trigger για την εξασφάλιση της σωστής εισαγωγής τιμών.

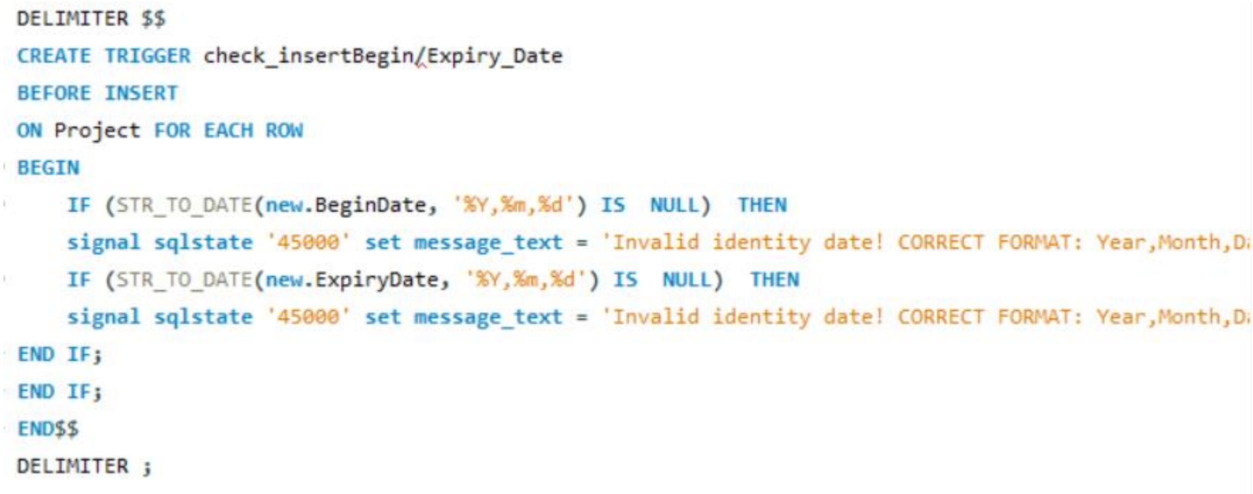
<u>Trigger1-2</u>: Έλεγχος διάρκειας έργου και εξασφάλιση ότι οι τιμές ExpiryDate και BeginDate είναι της μορφής: Year-Month-Day.

Έλεγχος διάρκειας(1-4 χρόνια):



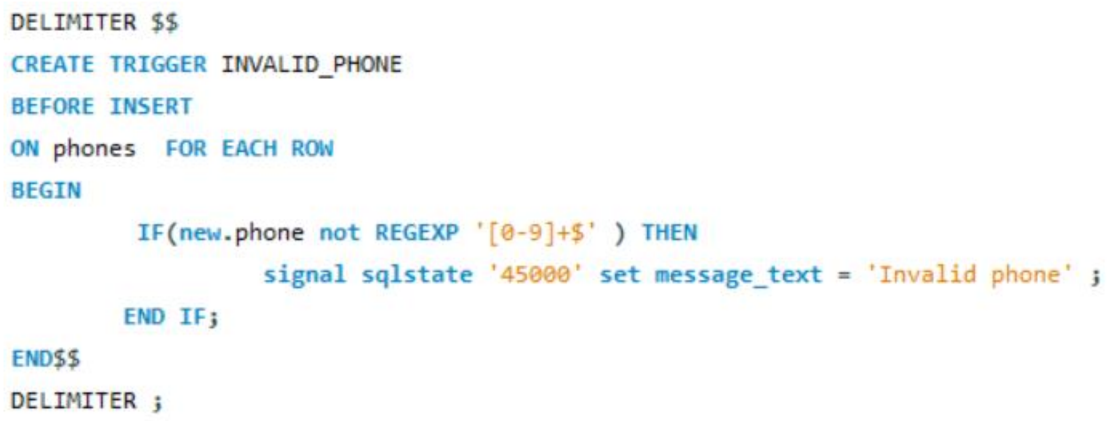
```
DELIMITER $$
CREATE TRIGGER INVALID_DATE
BEFORE INSERT
ON Project  FOR EACH ROW
BEGIN
IF(DATEDIFF(new.ExpiryDate,new.BeginDate)  < 365 OR DATEDIFF(new.ExpiryDate,new.BeginDate )  >1461)
THEN
signal sqlstate '45000' set message_text = 'Invalid Date set' ;
END IF;
END$$
DELIMITER;
```

Έλεγχος τιμών DATE:



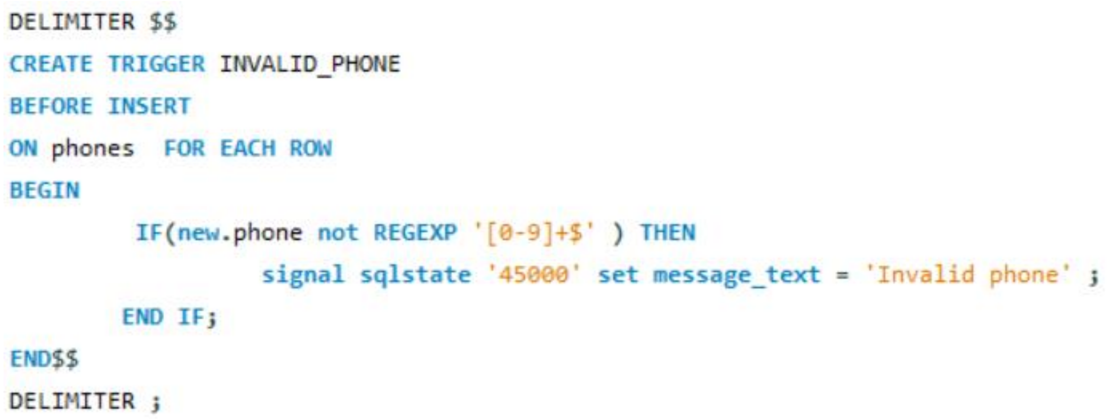
```
DELIMITER $$
CREATE TRIGGER check_insertBegin/Expiry_Date
BEFORE INSERT
ON Project FOR EACH ROW
BEGIN
    IF (STR_TO_DATE(new.BeginDate, '%Y,%m,%d') IS  NULL)  THEN
    signal sqlstate '45000' set message_text = 'Invalid identity date! CORRECT FORMAT: Year,Month,D
    IF (STR_TO_DATE(new.ExpiryDate, '%Y,%m,%d') IS  NULL)  THEN
    signal sqlstate '45000' set message_text = 'Invalid identity date! CORRECT FORMAT: Year,Month,D
END IF;
END IF;
END$$
DELIMITER ;
```

Trigger 3: Έλεγχος τιμής phone(INT)



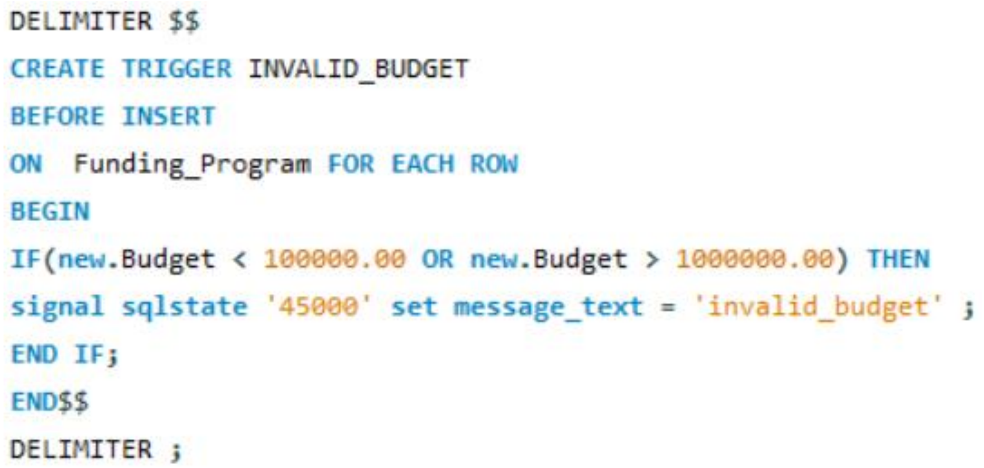
```
DELIMITER $$
CREATE TRIGGER INVALID_PHONE
BEFORE INSERT
ON phones  FOR EACH ROW
BEGIN
         IF(new.phone not REGEXP '[0-9]+$' ) THEN
                  signal sqlstate '45000' set message_text = 'Invalid phone' ;
        END IF;
END$$
DELIMITER ;
```

Trigger 4:Έλεγχος τιμής researcher_id(INT)



```
DELIMITER $$
CREATE TRIGGER INVALID_PHONE
BEFORE INSERT
ON phones  FOR EACH ROW
BEGIN
         IF(new.phone not REGEXP '[0-9]+$' ) THEN
                  signal sqlstate '45000' set message_text = 'Invalid phone' ;
        END IF;
END$$
DELIMITER ;
```

Trigger 5: Έλεγχος όριων τιμής Budget(100000-1000000)

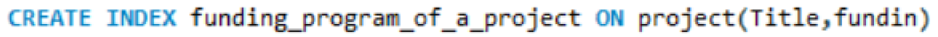


```
DELIMITER $$
CREATE TRIGGER INVALID_BUDGET
BEFORE INSERT
ON  Funding_Program FOR EACH ROW
BEGIN
IF(new.Budget < 100000.00 OR new.Budget > 1000000.00) THEN
signal sqlstate '45000' set message_text = 'invalid_budget' ;
END IF;
END$$
DELIMITER ;
```

Indexes:

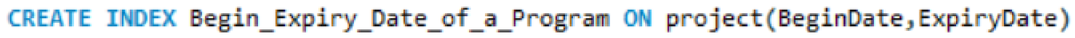
Δημιουργία κατάλληλων ευρετηρίων για ευκολότερη πρόσβαση σε κύρια δεδομένα

<u>Index για τα attributes Title, fundin του πίνακα Project:</u> Ευκολότερος τρόπος να βρίσκουμε τα προγράμματα τα οποία επιχορηγούν τα έργα ο οποίος είναι και ένας από τους κυριότερους Πίνακες της ΒΔ.



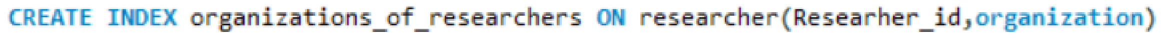
```
CREATE INDEX funding_program_of_a_project ON project(Title,fundin)
```

Index για τα attributes BeginDate ,ExpiryDate του πίνακα Project: Το index αυτό δημιουργήθηκε ως διευκόλυνση στον έλεγχο κατά πόσο ένα έργο είναι ενεργό.



```
CREATE INDEX Begin_Expiry_Date_of_a_Program ON project(BeginDate,ExpiryDate)
```

Index για το attribute Researcher_id του πίνακα Researcher: Δημιουργεία του index με όλα τα researcher_id που επίσης αποτελέι έναν από τους κυριότερους πίνακες της ΒΔ.



```
CREATE INDEX organizations_of_researchers ON researcher(Researher_id,organization)
```

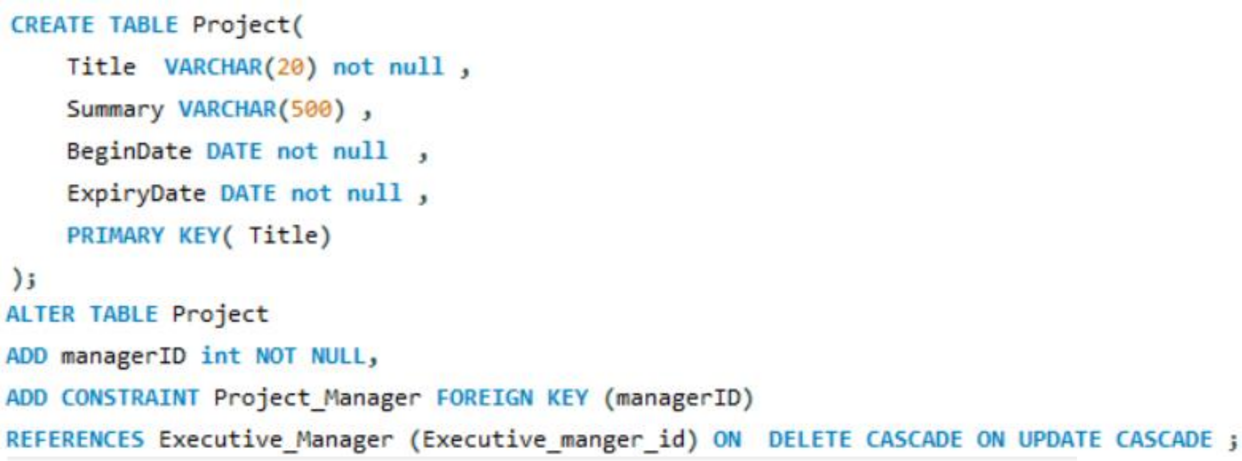
Σύστημα και Γλώσσες Προγραμματισμού

Για την υλοποίηση της ΒΔ χρησιμοποιήθηκε η MySQL. Συγκεκριμένα, χρησιμοποιήσαμε την εφαρμογή phpMyAdmin για την δημιουργία της βάσης.
Έπειτα, χρησιμοποιήσαμε HTML για την δημιουργία του interface και τη γλώσσα JavaScript για την ένωση του User Interface με τη βάση μας.
Σημειώσεις για το στήσιμο της σελίδας υπάρχουν στο τέλος της αναφοράς μας, μετά τον κώδικα sql.

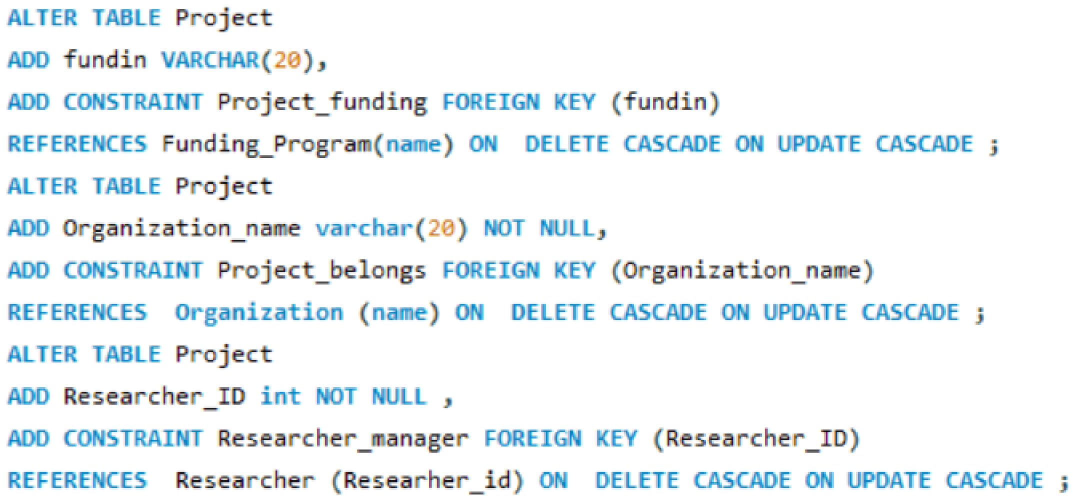
Εγκατάσταση Εφαρμογής

Παραδείγματα κώδικα δημιουργίας πινάκων σε SQL:

Δημιουργία πίνακα Project και προσθήκη σχέσεων(Foreign keys).

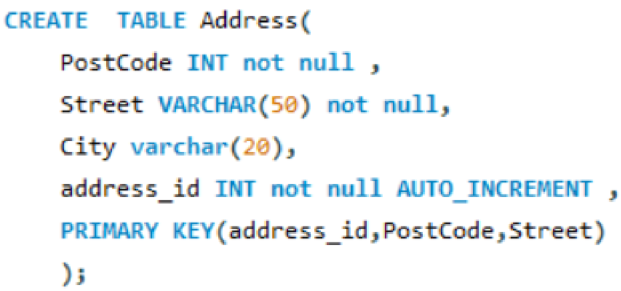


```sql
CREATE TABLE Project(
    Title  VARCHAR(20) not null ,
    Summary VARCHAR(500) ,
    BeginDate DATE not null  ,
    ExpiryDate DATE not null ,
    PRIMARY KEY( Title)
);
ALTER TABLE Project
ADD managerID int NOT NULL,
ADD CONSTRAINT Project_Manager FOREIGN KEY (managerID)
REFERENCES Executive_Manager (Executive_manger_id) ON  DELETE CASCADE ON UPDATE CASCADE ;
```



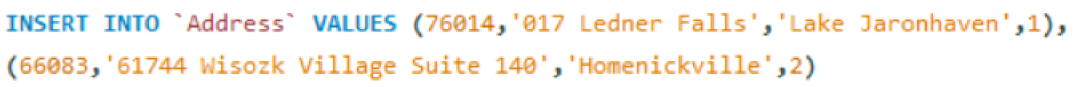
```sql
ALTER TABLE Project
ADD fundin VARCHAR(20),
ADD CONSTRAINT Project_funding FOREIGN KEY (fundin)
REFERENCES Funding_Program(name) ON  DELETE CASCADE ON UPDATE CASCADE ;
ALTER TABLE Project
ADD Organization_name varchar(20) NOT NULL,
ADD CONSTRAINT Project_belongs FOREIGN KEY (Organization_name)
REFERENCES  Organization (name) ON  DELETE CASCADE ON UPDATE CASCADE ;
ALTER TABLE Project
ADD Researcher_ID int NOT NULL ,
ADD CONSTRAINT Researcher_manager FOREIGN KEY (Researcher_ID)
REFERENCES  Researcher (Researher_id) ON  DELETE CASCADE ON UPDATE CASCADE ;
```

Δημιουργία πίνακα Address.

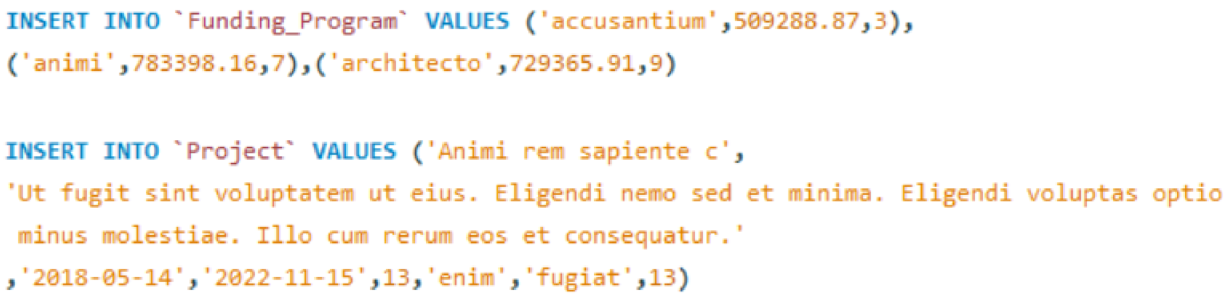


```sql
CREATE  TABLE Address(
    PostCode INT not null ,
    Street VARCHAR(50) not null,
    City varchar(20),
    address_id INT not null AUTO_INCREMENT ,
    PRIMARY KEY(address_id,PostCode,Street)
    );
```

Παραδείγματα εισαγωγής δεδομένων:



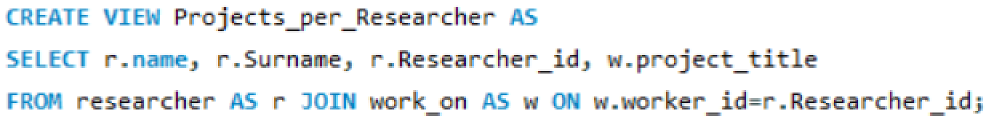
```sql
INSERT INTO `Address` VALUES (76014,'017 Ledner Falls','Lake Jaronhaven',1),
(66083,'61744 Wisozk Village Suite 140','Homenickville',2)
```



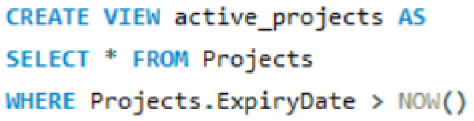
```
INSERT INTO `Funding_Program` VALUES ('accusantium',509288.87,3),
('animi',783398.16,7),('architecto',729365.91,9)

INSERT INTO `Project` VALUES ('Animi rem sapiente c',
'Ut fugit sint voluptatem ut eius. Eligendi nemo sed et minima. Eligendi voluptas optio
 minus molestiae. Illo cum rerum eos et consequatur.'
,'2018-05-14','2022-11-15',13,'enim','fugiat',13)
```

Κώδικας SQL για τα διάφορα ερωτήματα και απαραίτητα views:

Query 3.2: Projects per Researcher + All active projects.

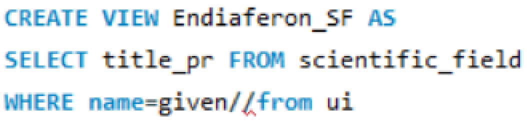


```
CREATE VIEW Projects_per_Researcher AS
SELECT r.name, r.Surname, r.Researcher_id, w.project_title
FROM researcher AS r JOIN work_on AS w ON w.worker_id=r.Researcher_id;
```

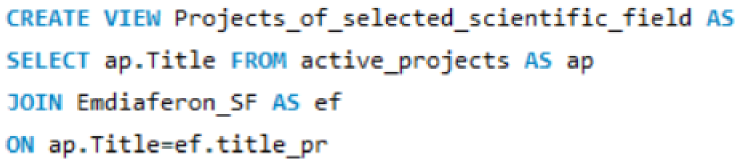


```
CREATE VIEW active_projects AS
SELECT * FROM Projects
WHERE Projects.ExpiryDate > NOW()
```

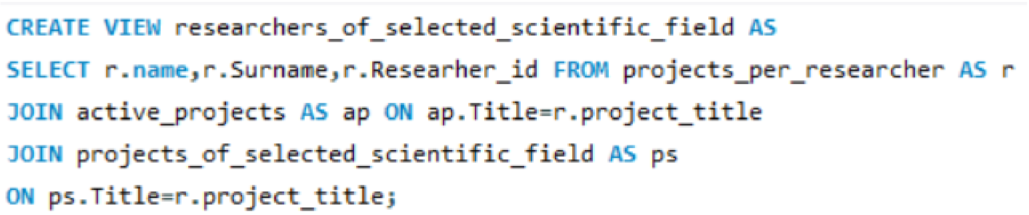
Query 3.3: Active projects with a funding program of a certain scientific field and Researchers working on this scientific field.



```
CREATE VIEW Endiaferon_SF AS
SELECT title_pr FROM scientific_field
WHERE name=given//from ui
```

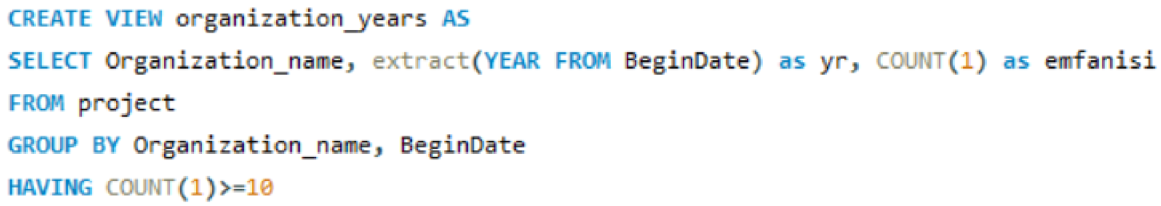


```
CREATE VIEW Projects_of_selected_scientific_field AS
SELECT ap.Title FROM active_projects AS ap
JOIN Emdiaferon_SF AS ef
ON ap.Title=ef.title_pr
```

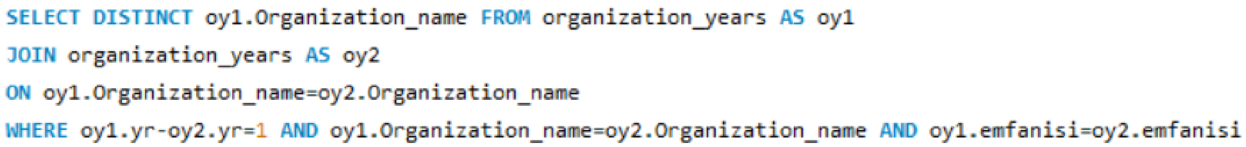


```
CREATE VIEW researchers_of_selected_scientific_field AS
SELECT r.name,r.Surname,r.Researher_id FROM projects_per_researcher AS r
JOIN active_projects AS ap ON ap.Title=r.project_title
JOIN projects_of_selected_scientific_field AS ps
ON ps.Title=r.project_title;
```

Query 3.4: Organizations that received same number of projects in 2 consecutive years with more than 10 projects per year.

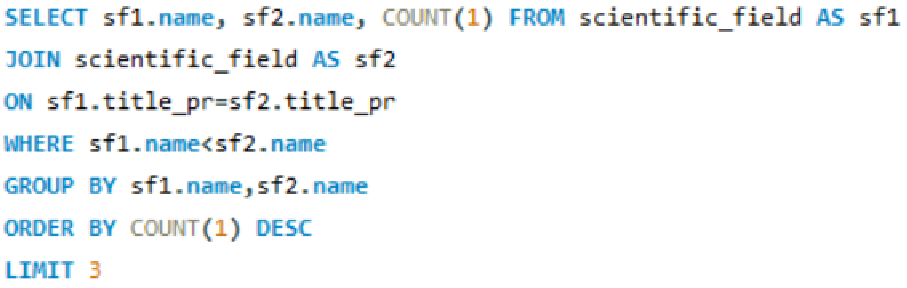


```
CREATE VIEW organization_years AS
SELECT Organization_name, extract(YEAR FROM BeginDate) as yr, COUNT(1) as emfanisi
FROM project
GROUP BY Organization_name, BeginDate
HAVING COUNT(1)>=10
```



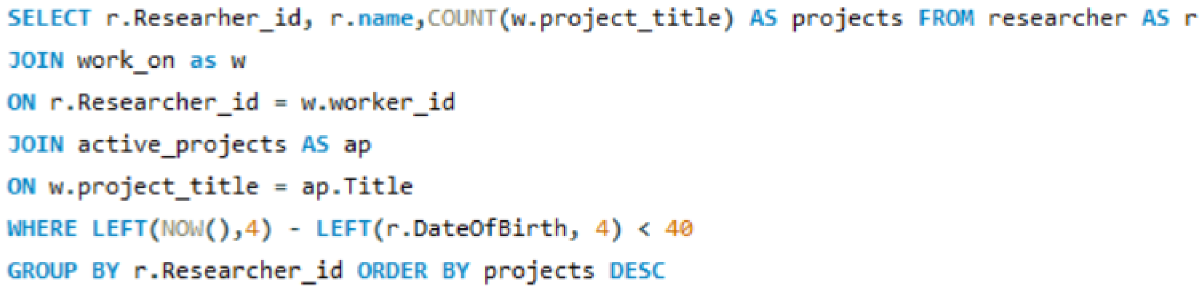
```
SELECT DISTINCT oy1.Organization_name FROM organization_years AS oy1
JOIN organization_years AS oy2
ON oy1.Organization_name=oy2.Organization_name
WHERE oy1.yr-oy2.yr=1 AND oy1.Organization_name=oy2.Organization_name AND oy1.emfanisi=oy2.emfanisi
```

Query 3.5: Top 3 most common combinations of 2 different scientific fields.



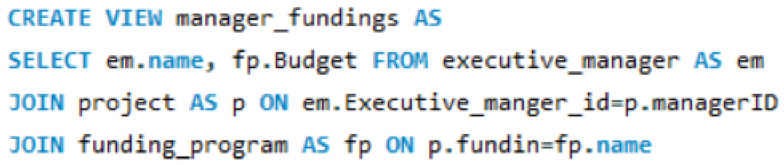
```
SELECT sf1.name, sf2.name, COUNT(1) FROM scientific_field AS sf1
JOIN scientific_field AS sf2
ON sf1.title_pr=sf2.title_pr
WHERE sf1.name<sf2.name
GROUP BY sf1.name,sf2.name
ORDER BY COUNT(1) DESC
LIMIT 3
```

Query 3.6: Top researchers under the age of 40 with active projects.

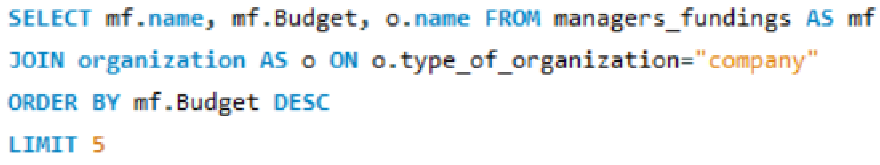


```
SELECT r.Researher_id, r.name,COUNT(w.project_title) AS projects FROM researcher AS r
JOIN work_on as w
ON r.Researcher_id = w.worker_id
JOIN active_projects AS ap
ON w.project_title = ap.Title
WHERE LEFT(NOW(),4) - LEFT(r.DateOfBirth, 4) < 40
GROUP BY r.Researcher_id ORDER BY projects DESC
```

Query 3.7: Top 5 Executive Managers with the highest amount of funding.

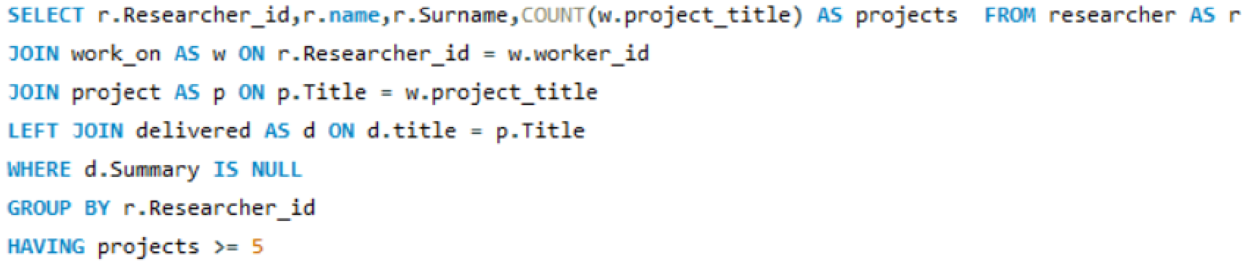


```
CREATE VIEW manager_fundings AS
SELECT em.name, fp.Budget FROM executive_manager AS em
JOIN project AS p ON em.Executive_manger_id=p.managerID
JOIN funding_program AS fp ON p.fundin=fp.name
```



```
SELECT mf.name, mf.Budget, o.name FROM managers_fundings AS mf
JOIN organization AS o ON o.type_of_organization="company"
ORDER BY mf.Budget DESC
LIMIT 5
```

Query 3.8: Top Researchers that work on 5 or more projects without deliveries.



```
SELECT r.Researcher_id,r.name,r.Surname,COUNT(w.project_title) AS projects  FROM researcher AS r
JOIN work_on AS w ON r.Researcher_id = w.worker_id
JOIN project AS p ON p.Title = w.project_title
LEFT JOIN delivered AS d ON d.title = p.Title
WHERE d.Summary IS NULL
GROUP BY r.Researcher_id
HAVING projects >= 5
```

ΚΩΔΙΚΑΣ SQL

```
CREATE TABLE Project(
            Title  VARCHAR(20) not null ,
            Summary VARCHAR(500) ,
            BeginDate DATE not null  ,
            ExpiryDate DATE not null ,
            PRIMARY KEY( Title)
);
CREATE  TABLE Address(
            PostCode INT not null ,
            Street VARCHAR(50) not null,
            City varchar(20),
   address_id INT not null AUTO_INCREMENT ,
   PRIMARY KEY(address_id,PostCode,Street)
);
CREATE TABLE Organization(
            name varchar(20) not null,
            shortcut varchar(5),
    type_of_organization ENUM("university","company","research center"),
            PRIMARY KEY(name)
);
CREATE TABLE  Researcher(
            name varchar(20) not null ,
            Surname varchar(20) ,
	Sex ENUM('Male', 'Female'),
	DateOfBirth DATE ,
	Researher_id int not null AUTO_INCREMENT,
	organization varchar(20) not null,
	PRIMARY KEY (Researher_id));
CREATE TABLE Executive_Manager(
	name varchar (20) not null,
	Executive_manger_id int not null AUTO_INCREMENT,
	PRIMARY KEY(Executive_manger_id)
);
CREATE TABLE Evalulation(
	grade INT ,
	Date_of_Evalulation  Date
) ;

CREATE TABLE Delivered(
	Summary varchar (20)
); /* Deliverd is a weak  entity */
CREATE TABLE Funding_Program(
	name varchar(20) not null,
	Budget DECIMAL(9,2) ,
	PRIMARY KEY(name)
	);
CREATE TABLE Company(
	Budget DECIMAL(9,2));
CREATE TABLE University(
	Budget_From_Ministry_of_Education DECIMAL(9,2));
CREATE TABLE Researcher_Center(
	Budget_From_Ministry_of_Education DECIMAL(9,2),
	Budget_From_Private_Actions DECIMAL(9,2));
/*Project Relation*/
ALTER TABLE Project
ADD managerID int NOT NULL,
ADD CONSTRAINT Project_Manager FOREIGN KEY (managerID)
REFERENCES Executive_Manager (Executive_manger_id) ON  DELETE CASCADE ON UPDATE
CASCADE ;

ALTER TABLE Project
ADD fundin VARCHAR(20),
ADD CONSTRAINT Project_funding FOREIGN KEY (fundin)
REFERENCES Funding_Program(name) ON  DELETE CASCADE ON UPDATE CASCADE ;


ALTER TABLE Project
```

```
ADD Organization_name varchar(20) NOT NULL,
ADD CONSTRAINT Project_belongs FOREIGN KEY (Organization_name)
REFERENCES  Organization (name) ON  DELETE CASCADE ON UPDATE CASCADE ;

ALTER TABLE Project
ADD Researcher_ID int NOT NULL ,
ADD CONSTRAINT Researcher_manager FOREIGN KEY (Researcher_ID)
REFERENCES  Researcher (Researher_id) ON  DELETE CASCADE ON UPDATE CASCADE ;

/*Funding program Relation*/
ALTER TABLE Funding_Program
ADD address_ID int ,
ADD CONSTRAINT Funding_program_organization_address FOREIGN KEY (address_ID)
REFERENCES  Address (address_id) ON  DELETE CASCADE ON UPDATE CASCADE ;




/*ISA*/
ALTER TABLE University
ADD Organization_name varchar(20),
ADD CONSTRAINT Organization_is_University FOREIGN KEY (Organization_name)
REFERENCES  Organization (name) ON  DELETE CASCADE ON UPDATE CASCADE ;

ALTER TABLE Company
ADD Organization_name   varchar(20),
ADD CONSTRAINT Organization_is_Company FOREIGN KEY (Organization_name)
REFERENCES  Organization (name) ON  DELETE CASCADE ON UPDATE CASCADE ;

ALTER TABLE Researcher_Center
ADD Organization_name   varchar(20),
ADD CONSTRAINT Organization_is_reaserch_center FOREIGN KEY (Organization_name)
REFERENCES  Organization (name) ON  DELETE CASCADE ON UPDATE CASCADE ;
/*Organization Relation*/
ALTER TABLE Organization
ADD address_id int,
ADD CONSTRAINT Organization_has_address FOREIGN KEY (address_id)
REFERENCES  Address (address_id) ON  DELETE CASCADE ON UPDATE CASCADE ;

/*Evaluation*/
ALTER TABLE Evalulation
ADD project_title varchar(20),
ADD CONSTRAINT Evaluation_of_project FOREIGN KEY (project_title)
REFERENCES  Project (Title ) ON  DELETE CASCADE ON UPDATE CASCADE,
ADD reasearcher_id int,
ADD CONSTRAINT reasercher_who_eval FOREIGN KEY (reasearcher_id)
REFERENCES  Researcher (Researher_id) ON  DELETE CASCADE ON UPDATE CASCADE;
/*Researcher*/
CREATE TABLE WORK_ON(
            worker_id int,
            project_title varchar(20) ,
            CONSTRAINT Researcher_that_works_on_project FOREIGN KEY (worker_id)
            REFERENCES  Researcher (Researher_id) ON  DELETE CASCADE ON UPDATE CASCADE,
            CONSTRAINT Project_that_Researcher_works FOREIGN KEY (project_title)
            REFERENCES  Project (Title ) ON  DELETE CASCADE ON UPDATE CASCADE);
/*Delivery Sxesi*/
            CREATE TABLE project_to_delivery(
            Date_of_Delivery Date,
            title varchar(20),
            CONSTRAINT Project_to_Delivered FOREIGN KEY (title)
            REFERENCES   Project(title) ON  DELETE CASCADE ON UPDATE CASCADE );

ALTER TABLE Delivered
ADD title varchar(20),
ADD CONSTRAINT Project_has_been_Delivered FOREIGN KEY (title)
REFERENCES   project_to_delivery(title) ON  DELETE CASCADE ON UPDATE CASCADE ;

CREATE TABLE scientific_field(
            name
ENUM('Physics','Biology','Chemistry','Zoology','Astronomy','Medicine','Astrophysics','Earthscien
ces','Pharmacy','Human Sciences') not null,
            title_pr varchar(20) not null,
            CONSTRAINT Project_has_field FOREIGN KEY (title_pr)
REFERENCES   Project(Title) ON  DELETE CASCADE ON UPDATE CASCADE );
CREATE TABLE Employee_Relationship(
            Date_of_begin DATE not null,
            name varchar(20),
   researcher_id int,
            CONSTRAINT Organization_employee_works_for FOREIGN KEY (name)
            REFERENCES   Organization (name) ON  DELETE CASCADE ON UPDATE CASCADE,
            CONSTRAINT employee_has_id FOREIGN KEY (researcher_id)
            REFERENCES   Researcher(Researher_id) ON  DELETE CASCADE ON UPDATE CASCADE);
CREATE TABLE phones(
   name varchar(20) not null,
            phone varchar(15) not null,
   CONSTRAINT Organization_ph   FOREIGN KEY (name)
   REFERENCES   Organization (name) ON  DELETE CASCADE ON UPDATE CASCADE) ;
ALTER TABLE Researcher
```

```
        ADD CONSTRAINT organization_of_researcher FOREIGN KEY(organization)
        REFERENCES Organization (name) ON DELETE CASCADE ON UPDATE CASCADE ;


/*triggers
elegxos oti i diarkia tou ergou ine 4 xronia*/
DELIMITER $$
CREATE TRIGGER INVALID_DATE
BEFORE INSERT
ON Project  FOR EACH ROW
BEGIN
    IF(DATEDIFF(new.ExpiryDate,new.BeginDate)  < 365 OR
DATEDIFF(new.ExpiryDate,new.BeginDate )  >1461 ) THEN
                    signal sqlstate '45000' set message_text = 'Invalid Date set' ;
                END IF;
END$$
DELIMITER;

/*elegxos oti to researcher id ine integer*/
DELIMITER $$
CREATE TRIGGER check_researcher_id
BEFORE INSERT
ON researcher FOR EACH ROW
BEGIN
IF (new.researcher_id not REGEXP '^[0-9]+$') THEN
signal sqlstate '45000' set message_text = 'Invalid id number';
END IF;
END$$
DELIMITER ;

/*CHECK INPUT DATE ON PROJECT*/
DELIMITER $$
CREATE TRIGGER check_insertBegin/Expiry_Date
BEFORE INSERT
ON Project FOR EACH ROW
BEGIN
  IF (STR_TO_DATE(new.BeginDate, '%Y,%m,%d') IS  NULL)  THEN
    signal sqlstate '45000' set message_text = 'Invalid identity date! CORRECT FORMAT:
Year,Month,Date';
        IF (STR_TO_DATE(new.ExpiryDate, '%Y,%m,%d') IS  NULL)  THEN
    signal sqlstate '45000' set message_text = 'Invalid identity date! CORRECT FORMAT:
Year,Month,Date';
END IF;
END IF;
```

```
END$$
DELIMITER ;

/*CHECK INPUT ON PHONES*/
DELIMITER $$
CREATE TRIGGER INVALID_PHONE
BEFORE INSERT
ON phones  FOR EACH ROW
BEGIN
	IF(new.phone not REGEXP '[0-9]+$' ) THEN
				signal sqlstate '45000' set message_text = 'Invalid phone' ;
			END IF;
END$$
DELIMITER ;

/*LIMIT BUDGET OF A FUNDING PROGRAM*/
DELIMITER $$
CREATE TRIGGER INVALID_BUDGET
BEFORE INSERT
ON   Funding_Program FOR EACH ROW
BEGIN
	IF(new.Budget < 100000.00 OR new.Budget > 1000000.00) THEN
				signal sqlstate '45000' set message_text = 'invalid_budget' ;
			END IF;
END$$
DELIMITER ;

/*INDEXES
gia na vrisko efkola apo poio programma xorigounte ta erga*/
CREATE INDEX funding_program_of_a_project ON project(Title,fundin)

/*gia na vrisko efkola tin imerominia enarxis kai lixis ton ergon*/
CREATE INDEX Begin_Expiry_Date_of_a_Program ON project(BeginDate,ExpiryDate)

/*gia na vrisko efkola tous organismous ton erevniton*/
CREATE INDEX organizations_of_researchers ON researcher(Researher_id,organization)

/*VIEWS*/
/*WORKERS OF A PROJECT*/
CREATE VIEW WORKS_project AS
SELECT  project_title,Researcher.Researher_id
,Researcher.name,Researcher.Surname,Researcher.Sex FROM WORK_ON
FULL JOIN  Researcher
ON  worker_id = Researcher.Researher_id
```

```
WHERE project_title = 'PR2';
WHERE Project.fundin != NULL ;

/*VIEWS FOR ORGANIZATIONS*/
CREATE VIEW ORG_UNI AS
SELECT * FROM organization
WHERE type_of_organization="University"

CREATE VIEW ORG_COMP AS
SELECT * FROM organization
WHERE type_of_organization="company"

CREATE VIEW ORG_RC AS
SELECT * FROM organization
WHERE type_of_organization="research center"



/*query 3.2*/

CREATE VIEW Projects_per_Researcher AS
SELECT r.name, r.Surname, r.Researcher_id, w.project_title
FROM researcher AS r JOIN work_on AS w ON w.worker_id=r.Researcher_id;

CREATE VIEW active_projects
SELECT * FROM Projects
WHERE Projects.ExpiryDate > NOW()


/*query 3.3*/
CREATE VIEW Endiaferon_SF AS
SELECT title_pr FROM scientific_field
WHERE name=given//from ui


/*active projects pou exoun to scientific field pou mas endiaferei*/
CREATE VIEW Projects_of_selected_scientific_field AS
SELECT ap.Title FROM active_projects AS ap
JOIN Emdiaferon_SF AS ef
ON ap.Title=ef.title_pr

/*researchers gia to field*/
CREATE VIEW researchers_of_selected_scientific_field AS
SELECT r.name,r.Surname,r.Researher_id FROM projects_per_researcher AS r
```

```
JOIN active_projects AS ap ON ap.Title=r.project_title
JOIN projects_of_selected_scientific_field AS ps
ON ps.Title=r.project_title;

/*query 3.4*/
/*VIEW GIA 3.4*/
CREATE VIEW organization_years AS
SELECT Organization_name, extract(YEAR FROM BeginDate) as yr, COUNT(1) as emfanisi
FROM project
GROUP BY Organization_name, BeginDate
HAVING COUNT(1)>=10


SELECT DISTINCT oy1.Organization_name FROM organization_years AS oy1
JOIN organization_years AS oy2
ON oy1.Organization_name=oy2.Organization_name
WHERE oy1.yr-oy2.yr=0 AND oy1.Organization_name=oy2.Organization_name AND
oy1.emfanisi=oy2.emfanisi


/*query 3.5*/

SELECT sf1.name, sf2.name, COUNT(1) FROM scientific_field AS sf1
JOIN scientific_field AS sf2
ON sf1.title_pr=sf2.title_pr
WHERE sf1.name<sf2.name
GROUP BY sf1.name,sf2.name
ORDER BY COUNT(1) DESC
LIMIT 3


/*query 3.6*/

SELECT r.Researher_id, r.name,COUNT(w.project_title) AS projects FROM researcher AS r
JOIN work_on as w
ON r.Researcher_id = w.worker_id
JOIN active_projects AS ap
ON w.project_title = ap.Title
WHERE LEFT(NOW(),4) - LEFT(r.DateOfBirth, 4) < 40
GROUP BY r.Researcher_id ORDER BY projects DESC


/*query 3.7*/
CREATE VIEW manager_fundings AS
```

```
SELECT em.name, fp.Budget FROM executive_manager AS em
JOIN project AS p ON em.Executive_manger_id=p.managerID
JOIN funding_program AS fp ON p.fundin=fp.name
```

```
SELECT mf.name, mf.Budget, o.name FROM managers_fundings AS mf
JOIN organization AS o ON o.type_of_organization="company"
ORDER BY mf.Budget DESC
LIMIT 5
```

```
/*query 3.8*/
SELECT r.Researcher_id,r.name,r.Surname,COUNT(w.project_title) AS projects  FROM
researcher AS r
JOIN work_on AS w ON r.Researcher_id = w.worker_id
JOIN project AS p ON p.Title = w.project_title
LEFT JOIN delivered AS d ON d.title = p.Title
WHERE d.Summary IS NULL
GROUP BY r.Researcher_id
HAVING projects >= 5
```

Στήσιμο Ιστοσελίδας:

1. Server.js creates the servers using express framework and app.js javasrip to create the wep app

```
    const app = require('./app') here we require the code of app.js  the port of the
server is by deafault 3000 unless the there is an existing port of the server
const port = process.env.SERVER_PORT || 3000
App.js here are the route paths for example localhost:3000/Project is a menu associated
with Project relatable entity’s.  we are using librarys path to and funvtion of the
express Framework to for making the routing better for instance
app.use(express.static(path.join(__dirname, 'public'))); bassicly it it takes the path of
app.js and it make it static and then when  we try to launch app when ever app is lauched
it will use app.js path so the web app will not break for instance if  we are gone use
instead app.use(express.static('public')) the app would break if we try to launch it from
another library. The routing is: app.use('/', layout); is the main page
app.use('/Query',Query); this is menu for all the querys
app.use('/Query/query1',query1)all page that are going to use the Path /Query must be
writteb down here
app.use('/Query/View1',View1) or else the they will not work
app.use('/Query/View2',View2)
//app.use('/Query/query3', query3)
app.use('/Query/query4', query4)
app.use('/Query/query5',query5)
app.use('/Query/query6',query6)
app.use('/Query/query7',query7)
app.use('/Query/query8',query8)
app.use('/Organization',Orglayout)//if  i want to add apge /organization i have to it
below or else it will not waotk and a after continue ather paths
app.use('/Organization/addAddress',addAddress)
app.use('/Organization/deleteaddress',deleteAddress)
app.use('/Organization/editaddress',editAddress)
app.use('/Organization/addOrganization',addOrganization)
app.use('/Organization/delete', deleteOrganization)
app.use('/Organization/edit', editOrganization)
app.use('/Organization/addfundingpr',addfundingpr)
app.use('/Organization/addphones',addphones)
app.use('/Organization/deletephone',deletephone)
app.use('/Organization/editphone',editphone)
app.use('/Organization/addCompany',addCompany)
app.use('/Organization/addUniversity',addUniversity)
app.use('/Organization/addResearchCent',addResearchCent)
app.use('/Project',Prlayout)
app.use('/Project/addManager',addManager)
app.use('/Project/editManager',editManager)
app.use('/Project/deleteManager',deleteManager)
app.use('/Project/addScientific',addScientific)
app.use('/Project/addEval',addEval)
app.use('/Project/addPrtoDeliver',addPrtoDeliver)
app.use('/Project/addDelivered',addDelivered)
app.use('/Project/addProject',addProject)
app.use('/Researcher',Rslayout)
app.use('/Researcher/addResearchers',addResearchers)
app.use('/Researcher/addresEmp',addresEmp)
app.use('/Researcher/addworkson',addworkson)
```

The routes directory are js code the are calling the controllers of a root
There's is an 404 page for when the user enters a path that does not exist
For instance localhost:3000/apples => 404 erorr => finding a page (page does not exist).and the command is the last routing

```
//end of routes using this project

// In case of an endpoint does not exist must return 404.html
app.use((req, res, next) => { res.status(404).render('404.ejs', { pageTitle: '404' }) })
```

And then we created different pages so each page will hold an entity and the user can view edit, add and delete data so each entity should have 4 controllers one to view which the most have as getMain , one cotroller for adding data such as example:

```
exports.getaddress= (req,res,next) => {//req (request ,res for respond

    const Street= req.body.Street;
    const PostCode= req.body.PostCode;
    const City= req.body.City;//from the.ejs file it get the data from the elements with
that id

    /*this parmaters come fromejs in script there difined bythere id*/
    pool.getConnection((err, conn) => { // here we make coonection using a POOL object
        var sqlQuery = `INSERT INTO address (PostCode,Street,City) VALUES(?,?,?)`;
        /*make the query for the add*/
        conn.promise().query(sqlQuery, [  PostCode,Street ,City]) //we use promise
//because the queries is an a asynchronous process with promise is like that we are
//saying when we done processing continue  independent of success or failure ex:
//.then(successs,failure=>
        .then(() => {
            pool.releaseConnection(conn);//we done it release the connection
            res.redirect('/Organization/addAddress');//and stay in the page independent
// the success or failure of the sql code
        })
        .catch(err => {
           res.redirect('/Organization/addAddress');
        })
        /*what ever happens stay in current page*/
    })
}
We using POOLS  and connecting the db :
//To do coonect database
const mysql = require('mysql2'); //using my sql2 as a library //all the extra library
//should be install the system *they  are more info in the code as long with web page tha
//have useful function and how to use them and install code for each library
/* now we have to establish a conection*/
const pool = mysql.createPool({
         host: "localhost",
         port: "3306",
         user: "root",
      password: "",
      database: "ELDIEK" //this is the db name  “” there is no password
 });
//note you can use env.localhost because the db has no password and  it can not parse the empty password
 // object pooling is the procces of retaining a set of unused objects which
 //sharefd a type
 //works: When you need a new object for your code, rather than allocating a new one
// from the system Memory Heap, you instead recycle one of the unused objects from the pool .  So in a few words pool retains  object from destruction
module.exports = {pool} ; //here we export as a library for the js code using it

//We use Bootstrap studio to design a page a single page for an entity ant then with coding we alter the columns(and alos we use some code from the internet to design our page  inputs the button for all pages
//there is a special menu for the queries so the user could Choose the query and see the answer
The web page have 4 categoriesL:
```

- Projects which have pages for(Project ,Manager of Project, Scientific field entity, project to deliver ,evaluation ,delivered projects)
- Researcher for (Researcher info ,Work on relation ship(researchers works on project),Employee relationship(Researcher works for Organization)
- And Organization Which have (Organization entity, Address entity because both Organization and Funding Programs have address field so the are connected with an address numeric id automatically ,Funding Programs, University, Company, Researcher Center ,phones(hold phones of an organization)
- Query’s starting with query1(3.1) to query8(3.8)
  The Css,js formatting code from bootstrap locate in public/ assets