

Βάσεις Δεδομένων

Εξαμηνιαία Εργασία

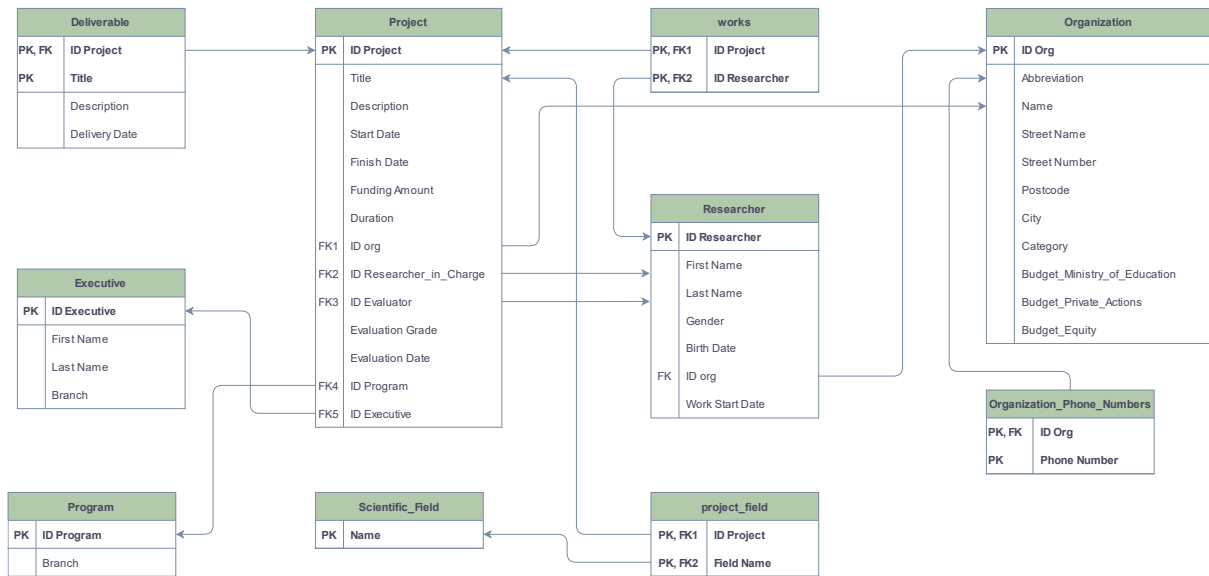
Ομάδα Project 39

Κεφαλληνός Διονύσιος, 03119030
Χατζηαναγνώστου Απόστολος, 03119021
Φωτόπουλος Κωνσταντίνος, 03119196

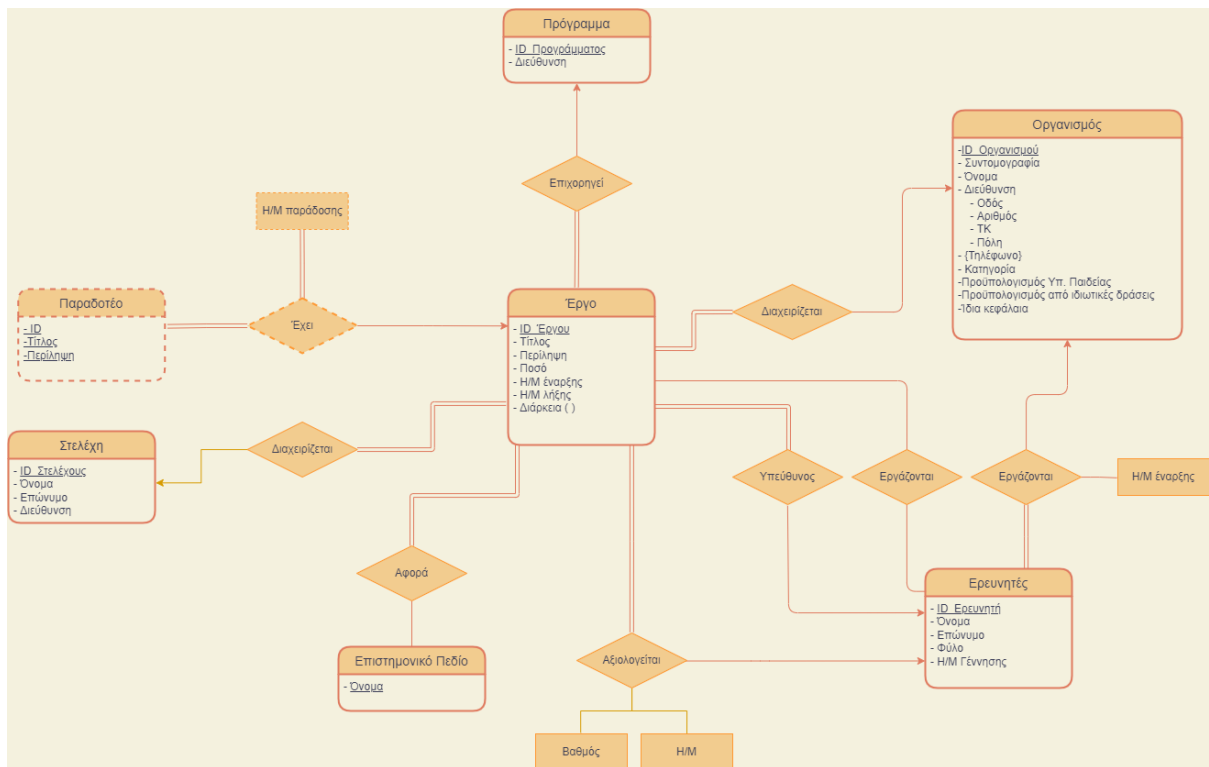
Περιεχόμενα

2.1.	Σχεσιακό διάγραμμα της ΒΔ με αιτιολόγηση
2.2.	Integrity Constraints
2.3.	Indexing
2.4.	Dummy Data
2.5.	DDL και DML script
2.6.	Αναλυτικά βήματα εγκατάστασης της εφαρμογής καθώς και βιβλιοθηκών που απαιτούνται.
2.7.	Σύνδεσμος για το git repo της εφαρμογής μας.

2.1 Σχεσιακό Διάγραμμα της Βάσης Δεδομένων μας.



Το αρχικό μας ER-Diagram ήταν το εξής (με την προσθήκη των Attributes **Προϋπολογισμός Υπ.Παιδείας, Προϋπολογισμός από Ιδιωτικές Δράσεις και Ίδια Κεφάλαια** στον πίνακα **Οργανισμός** ως μοναδική διαφοροποίηση):



Κατά την μετατροπή του ER-Diagram σε σχεσιακό, ακολουθήσαμε το σκεπτικό που θα επεξηγήσουμε παρακάτω:

-Οι πίνακες που αντιστοιχούσαν σε Entity Sets, δηλαδή οι πίνακες **Έργο, Οργανισμός, Ερευνητές, Επιστημονικό Πεδίο, Στελέχη** και **Πρόγραμμα**, παρέμειναν πίνακες και στο σχεσιακό. Ομοίως και για τον πίνακα **Παραδοτέο** που αντιστοιχεί σε Weak Entity Set.

-Οι Σχέσεις του Entity-Relationship model μετατράπηκαν σε foreign keys ή σε ξεχωριστούς πίνακες στο Relational Model.

Συγκεκριμένα, όσες σχέσεις εξέφραζαν στο ER many-to-one ή one-to-many αντιστοιχία, δηλώνονται με την ύπαρξη ενός Foreign Key στο Relational. Για παράδειγμα, η σχέση **Έργο-Διαχειρίζεται-Οργανισμός** δηλώνεται με την παρουσία του **ID_org** (το οποίο αναφέρεται στο Primary Key του πίνακα **Organization**) ως foreign key στον πίνακα **Project**. Με αυτόν τον τρόπο μετατράπηκαν όλες οι σχέσεις αυτού του τύπου και καταλήξαμε να έχουμε στο Relational diagram τα εξής foreign keys:

- **ID_Project** στο **Deliverable**
- **ID_org, ID_Researcher_in_Charge, ID_Evaluator, ID_Program, ID_Executive** στο **Project**
- **ID_org** στο **Researcher**

Αντιθέτως, οι σχέσεις που δήλωναν many-to-many αντιστοιχία, μετατράπηκαν σε ξεχωριστούς πίνακες με Primary Key τις δύο στήλες που περιέχουν τα Primary Keys των πινάκων που συνδέουν. Για παράδειγμα, η σχέση **Ερευνητές-Εργάζονται-Έργο**, μετατράπηκε στον πίνακα **works** ο οποίος περιλαμβάνει σε δύο στήλες τα κύρια κλειδιά των πινάκων **Project** και **Researcher** ως foreign keys. Αυτό προφανώς συμβαίνει για να μπορέσουμε να κρατήσουμε και στο σχεσιακό μοντέλο και στη συνέχεια στην υλοποίηση της βάσης δεδομένων μας την many-to-many αντιστοιχία. Με αυτόν τον τρόπο δημιουργήθηκαν οι πίνακες **works** και **project_field**.

-Τα multivalued attributes, και συγκεκριμένα τα Τηλέφωνα των Οργανισμών μετατρέπονται επίσης σε ξεχωριστό πίνακα **Organization_Phone_Numbers**, με foreign key το **ID_org** του οργανισμού στον οποίο ανήκει το τηλέφωνο.

-Τυχών attribute σε σχέσεις ενσωματώθηκαν στον πίνακα του entity set το οποίο αντιστοιχεί στην μεριά many της many-to-one σχέσης, όπως για παράδειγμα έγινε στην περίπτωση της ημερομηνίας στην σχέση **Έργο-Έχει-Παραδοτέο**.

Έτσι καταλήξαμε στο σχεσιακό διάγραμμα το οποίο θα υλοποιήσουμε παρακάτω ως βάση δεδομένων.

2.2 Integrity Constraints

Σε κάθε πίνακα λάβαμε τους εξής περιορισμούς:

Organization: Έγκυρος Αριθμός Οδού (θετικός)
Έγκυρος Ταχυδρομικός Κώδικας (θετικός)
Απόρριψη αρνητικών προϋπολογισμών και κεφαλαίων
Όλα τα attributes με εξαίρεση τους προϋπολογισμούς έχουν τον περιορισμό NOT NULL
Δημιουργία trigger σχετικά με τα είδη των προϋπολογισμών έτσι ώστε ανάλογα με την κατηγορία του οργανισμού να είναι NOT NULL μόνο τα κατάλληλα πεδία από αυτά. Επίσης θεωρούμε ότι ένας έγκυρος προϋπολογισμός για κάθε κατηγορία οργανισμού θα πρέπει να είναι μεγαλύτερος του 0 (αυτή η συνθήκη υλοποιείται στην Python).

- Executive:**
- Δημιουργία trigger που απαγορεύει τη διαγραφή ενός στελέχους υπεύθυνου σε κάποιο έργο.
 - Όλα τα attributes έχουν τον περιορισμό NOT NULL.
- Program:**
- Όλα τα attributes έχουν τον περιορισμό NOT NULL.
- Researcher:**
- Έλεγχος ότι η ημερομηνία πρόσληψης είναι μετά την ημερομηνία γέννησης
 - Foreign Key που δηλώνει τον οργανισμό στον οποίο εργάζεται ο ερευνητής. Σε περίπτωση διαγραφής ή ενημέρωσης ενός οργανισμού διαγράφονται ή ενημερώνονται όλοι οι ερευνητές του οργανισμού αυτού (on update cascade/on delete cascade).
 - Δημιουργία trigger το οποίο εμποδίζει τη διαγραφή ενός researcher ο οποίος είναι αξιολογητής ενός project.
 - Δημιουργία trigger το οποίο εμποδίζει τη διαγραφή του επιστημονικού υπεύθυνου ενός project.
 - Δημιουργία trigger το οποίο εμποδίζει την αλλαγή του οργανισμού στον οποίο ανήκει ο ερευνητής, όταν αυτός εργάζεται σε κάποιο ενεργό έργο του τωρινού οργανισμού του.
 - Δημιουργία trigger το οποίο σε περίπτωση αλλαγής του οργανισμού στον οποίο ανήκει ένας ερευνητής, να διαγράφει όλες τις καταχωρίσεις από τον πίνακα works του ερευνητή αυτού, δηλαδή να διαγράφει όλες τις σχέσεις που δείχνουν την εργασία του σε project του προηγούμενου οργανισμού του.
 - Όλα τα attributes έχουν τον περιορισμό NOT NULL.
- Organization_Phone_Numbers:**
- Foreign key το ID_Org που δηλώνει τον οργανισμό στον οποίο ανήκει ο εκάστοτε αριθμός τηλεφώνου. Σε περίπτωση διαγραφής ενός οργανισμού διαγράφονται και όλα τα τηλέφωνα του, ενώ σε περίπτωση ανανέωσης του ID ενός οργανισμού, ανανεώνεται και η σχετική στήλη στον πίνακα των τηλεφώνων.
 - Όλα τα attributes έχουν τον περιορισμό NOT NULL.
 - Τα τηλέφωνα αποθηκεύονται με τη μορφή varchar και όχι ως integers λόγω περιορισμού μεγέθους ακεραίων της mysql.
- Project:**
- Ελέγχουμε ότι η ημερομηνία που τελειώνει ένα έργο είναι μεταγενέστερη της ημερομηνίας έναρξής του.
 - Ελέγχουμε ότι το ποσό χρηματοδότησης είναι μεταξύ των 100.000 και 1.000.000 ευρώ.
 - Ελέγχουμε ότι η διάρκεια είναι έως 4 χρόνια.
 - Ελέγχουμε ότι ο βαθμός αξιολόγησης είναι μεγαλύτερος ή ίσος του 0 και μικρότερος ή ίσος του 10.
 - Foreign Keys που αναφέρονται στο ID οργανισμού, στο ID του υπεύθυνου ερευνητή, στο ID του αξιολογητή, στο ID του προγράμματος που ανήκει το έργο και στο ID του στελέχους που το διαχειρίζεται. Κατά την διαγραφή ενός οργανισμού διαγράφονται και όλα του τα project, ενώ δεν επιτρέπεται η διαγραφή προγράμματος ή στελέχους με το οποίο σχετίζονται ένα ή παραπάνω έργα. Για τα delete και update των foreign keys ID_Researcher_in_Charge και ID_Evaluator έχουν δημιουργηθεί όπως αναφέραμε στον πίνακα Researcher τα απαραίτητα triggers.
 - Δημιουργία trigger για την περίπτωση αλλαγής του αξιολογητή σε κάποιον που δουλεύει ή δούλεψε στο project.
 - Δημιουργία trigger που προσθέτει αυτόματα έναν ερευνητή στον πίνακα works ενός έργου αν αυτός ορισθεί ως υπεύθυνος του έργου αυτού.

-Δημιουργία ανάλογου trigger και για την περίπτωση που γίνει αυτό με update. Στην περίπτωση αυτή όμως θέλουμε το συγκεκριμένο έργο να είναι ενεργό. Διαφορετικά δεν προσθέτουμε στον πίνακα works τον νέο υπεύθυνο του έργου.
-Όλα τα attributes είναι NOT NULL εκτός από το composite attribute Duration.

Works: -Όλα τα attributes είναι not NULL.
-Foreign keys τα ID_Project και ID_Researcher τα οποία δείχνουν στα αντίστοιχα attributes των πινάκων project και researcher αντίστοιχα. Σε περίπτωση διαγραφής κάποιου project διαγράφουμε και τις αντίστοιχες σχέσεις του πίνακα works. Σε περίπτωση διαγραφής κάποιου ερευνητή διαγράφουμε τις αντίστοιχες σχέσεις (στις οποίες συμμετείχε ο ερευνητής αυτός) από τον πίνακα works.
-Δημιουργία trigger το οποίο εμποδίζει έναν ερευνητή να εργάζεται στο έργο που αξιολογεί.
-Δημιουργία trigger το οποίο εμποδίζει έναν ερευνητή να εργαστεί σε project οργανισμού διαφορετικού από αυτόν στον οποίο ανήκει.
-Δημιουργία trigger το οποίο εμποδίζει τη διαγραφή από τον πίνακα ενός ερευνητή ο οποίος είναι επιστημονικός υπεύθυνος ενός έργου, όταν το έργο αυτό είναι ενεργό.

Deliverable: -Όλα τα attributes είναι NOT NULL.
-Foreign key το ID_Project το οποίο δείχνει στα ID του πίνακα project. Σε περίπτωση διαγραφής ή ενημέρωσης κάποιου project διαγράφουμε και ενημερώνουμε αντίστοιχα τα κατάλληλα παραδοτέα (που έχουν ID_Project ίδιο με το ID του project που διαγράψαμε ή ενημερώσαμε).

Scientific_Field: -Δημιουργία trigger που απαγορεύει τη διαγραφή επιστημονικού πεδίου στο οποίο υπάγεται έστω και ένα έργο.
-Όλα τα attributes έχουν τον περιορισμό NOT NULL.

Project_field: -Όλα τα attributes είναι not NULL.
-Foreign keys τα ID_Project και Field_Title τα οποία δείχνουν στα attributes ID_Project και Name των πινάκων project και scientific_field αντίστοιχα. Σε περίπτωση διαγραφής κάποιου project διαγράφουμε τις αντίστοιχες καταχωρήσεις στον πίνακα project_field.
-Δημιουργία trigger το οποίο εμποδίζει τη διαγραφή καταχώρισης όταν το αντίστοιχο project της σχέσης εντάσσεται σε ένα ακριβώς επιστημονικό πεδίο (όλα τα project πρέπει να εντάσσονται σε τουλάχιστον ένα επιστημονικό πεδίο). Για να έχει κάθε έργο τουλάχιστον ένα επιστημονικό πεδίο, υλοποιήθηκε στην εφαρμογή της βάσης δεδομένων μας κατάλληλη διεπαφή στην οποία ο χρήστης δηλώνει ένα επιθυμητό επιστημονικό πεδίο κατά την εισαγωγή ενός έργου. Στη συνέχεια μπορεί αν θέλει να προσθέσει και άλλα.

Στα ID των πινάκων Executive, Researcher, Organization, Project και Program χρησιμοποιούμε auto_increment. Στους πίνακες αυτούς δε θεωρήσαμε απαραίτητο να έχει ο χρήστης τη δυνατότητα να κάνει update το ID. Συνεπώς σε αρκετές περιπτώσεις το ON UPDATE CASCADE είναι περιττό. Επιπλέον έχουμε δημιουργήσει triggers που ελέγχουν αν το format των εισαγόμενων ημερομηνιών είναι έγκυρο σε όλα τα attributes των πινάκων που είναι τύπου date.

2.3 Indexing

Κατά την διαδικασία επιλογής των στηλών στις οποίες θα δημιουργήσουμε index, λάβαμε υπόψιν μας ότι είναι περιττό στην MySQL (με βάση το InnoDB Storage Engine) να δημιουργήσουμε δείκτες σε foreign keys καθώς αυτοί δημιουργούνται αυτόματα για όλα τα foreign keys που ορίζουμε. Συνεπώς, έπειτα από προσεκτική επισκόπηση των queries και non-materialized views που υποστηρίζει η βάση δεδομένων μας, επιλέξαμε να δεικτοδοτήσουμε τις στήλες που συναντώνται συχνότερα σε where, group by και on statements. Αυτές είναι οι στήλες **Start_Date** και **Finish_Date** του πίνακα **Project** και η στήλη **Birth_Date** του πίνακα **Researcher**. Καθώς οι πίνακες αυτοί θα έχουν κατά κανόνα τα περισσότερα δεδομένα (λαμβάνοντας υπόψιν μας και τα dummy data τα οποία εισάγαμε στη βάση δεδομένων) θεωρήσαμε ότι οι δείκτες αυτοί είναι οι καταλληλότεροι. Παρόλα αυτά, καθώς τα δεδομένα που έχουμε προς το παρόν είναι λίγα, η διαφορά στην καθυστέρηση των Queries με ή χωρίς ευρετήρια δεν είναι αισθητή.

2.4 Dummy Data

Για να εισάγουμε στη βάση δεδομένων μας κατάλληλα δεδομένα ώστε να ελέγχουμε την σωστή λειτουργία της καθώς και των ζητούμενων επερωτημάτων χρησιμοποιήσαμε μια πληθώρα ιστοσελίδων δημιουργίας δεδομένων καθώς και την βιβλιοθήκη Faker της rython. Κατά την δημιουργία των δεδομένων προσέξαμε αυτά να πληρούν τις συνθήκες που θέσαμε προηγουμένως (για παράδειγμα οι ερευνητές να δουλεύουν μόνο σε έργα του οργανισμού τους) και να είναι αρκετά στο πλήθος ώστε τα επερωτήματα να δίνουν αρκετά και ενδιαφέροντα αποτελέσματα. Καταλήξαμε να δημιουργήσουμε 30 οργανισμούς, 50 προγράμματα, 300 περίπου έργα, 1200 περίπου ερευνητές και 6000 περίπου εργασιακές σχέσεις μεταξύ ερευνητών και έργων!

```
INSERT INTO researcher ('First_Name','Last_Name','Gender','Birth_Date','ID_org','Work_Start_Date') VALUES ('Leroy', 'Garcia', 'Male', '1987-06-11', '30', '2014-01-01');
INSERT INTO researcher ('First_Name','Last_Name','Gender','Birth_Date','ID_org','Work_Start_Date') VALUES ('Lorraine', 'Oconnor', 'Female', '1999-01-31', '30', '2000-09-26');
INSERT INTO researcher ('First_Name','Last_Name','Gender','Birth_Date','ID_org','Work_Start_Date') VALUES ('Cassandra', 'Garcia', 'Female', '1994-04-20', '30', '2011-08-18');
INSERT INTO researcher ('First_Name','Last_Name','Gender','Birth_Date','ID_org','Work_Start_Date') VALUES ('Zachary', 'Juarez', 'Female', '1973-09-02', '30', '2006-06-16');
INSERT INTO researcher ('First_Name','Last_Name','Gender','Birth_Date','ID_org','Work_Start_Date') VALUES ('Jennifer', 'Lewis', 'Male', '1971-07-03', '30', '2010-04-26');
INSERT INTO researcher ('First_Name','Last_Name','Gender','Birth_Date','ID_org','Work_Start_Date') VALUES ('Kristen', 'Nguyen', 'Female', '1982-01-21', '30', '2004-07-16');
INSERT INTO researcher ('First_Name','Last_Name','Gender','Birth_Date','ID_org','Work_Start_Date') VALUES ('Sean', 'Lee', 'Male', '1991-10-02', '30', '2015-12-12');
INSERT INTO researcher ('First_Name','Last_Name','Gender','Birth_Date','ID_org','Work_Start_Date') VALUES ('Susan', 'Grant', 'Male', '1994-02-07', '30', '2015-03-28');
INSERT INTO researcher ('First_Name','Last_Name','Gender','Birth_Date','ID_org','Work_Start_Date') VALUES ('Craig', 'Olson', 'Male', '1985-08-26', '30', '2016-04-07');
INSERT INTO researcher ('First_Name','Last_Name','Gender','Birth_Date','ID_org','Work_Start_Date') VALUES ('Wendy', 'Smith', 'Male', '1990-07-08', '30', '2011-12-14');
INSERT INTO researcher ('First_Name','Last_Name','Gender','Birth_Date','ID_org','Work_Start_Date') VALUES ('Chelsea', 'Norman', 'Female', '1995-05-29', '30', '2014-01-19');
INSERT INTO researcher ('First_Name','Last_Name','Gender','Birth_Date','ID_org','Work_Start_Date') VALUES ('Sonia', 'Rodriguez', 'Male', '1990-12-08', '30', '2002-01-07');
INSERT INTO researcher ('First_Name','Last_Name','Gender','Birth_Date','ID_org','Work_Start_Date') VALUES ('Dana', 'Gentry', 'Female', '1982-06-17', '30', '2012-04-04');

alter table executive auto_increment =1 ;
INSERT INTO `executive` ('ID_Executive', 'First_Name', 'Last_Name', 'Branch') VALUES (1, 'Grant', 'Satterfield', 'Education and Human Society');
INSERT INTO `executive` ('ID_Executive', 'First_Name', 'Last_Name', 'Branch') VALUES (2, 'Lacey', 'Abbott', 'Humanities and Creative Arts');
INSERT INTO `executive` ('ID_Executive', 'First_Name', 'Last_Name', 'Branch') VALUES (3, 'Nellie', 'Walker', 'Mathematical Information and Computing Sciences');
INSERT INTO `executive` ('ID_Executive', 'First_Name', 'Last_Name', 'Branch') VALUES (4, 'Marcelino', 'Kulas', 'Mathematical Information and Computing Sciences');
INSERT INTO `executive` ('ID_Executive', 'First_Name', 'Last_Name', 'Branch') VALUES (5, 'Emilia', 'Krajcik', 'Economics and Commerce');
INSERT INTO `executive` ('ID_Executive', 'First_Name', 'Last_Name', 'Branch') VALUES (6, 'Sibyl', 'Jacobs', 'Humanities and Creative Arts');
```

Μια μικρή περιοχή από το αρχείο δημιουργίας των δεδομένων.

2.5 DDL και DML Script

Ακολουθούν τα ζητούμενα Queries του ερωτήματος 3:

```
1. /* ----- QUERY 3.1 ----- */
2. select ID Project, Title, Description, Start Date, Finish Date,
   Duration, Funding Amount, ID org, ID Program, ID Executive
3. from project
4. where Duration = 2 /* and Start Date = and Finish Date = */ and
   ID Executive = 15; /*edit parameters to customize query*/
5.
6. /* ----- QUERY 3.1 ----- */
7. select p.ID Project, p.Title, r.First Name, r.Last Name, r.Gender,
   r.Birth_Date, r.ID_org, r.Work_Start_Date
8. from project p
9. inner join works w on p.ID Project = w.ID Project
10. inner join researcher r on w.ID_Researcher = r.ID_Researcher
11. where p.ID Project = 15; /*edit parameters to customize query*/
12.
13.
14. /* ----- 3.2 VIEWS ----- */
15. create view researcher_projects as
16. SELECT r.ID_Researcher, r.First_Name, r.Last_Name, p.ID_Project,
   p.Title as Project Title, p.Funding Amount
17. from researcher r
18. inner join works w on r.ID_Researcher = w.ID_Researcher
19. inner join project p on w.ID Project = p.ID Project;
20.
21. create view submissions as
22. select p.ID Project, p.Title as Name, d.Title as Submission Title,
   d.Description
23. from project p inner join deliverable d on p.ID Project =
   d.ID Project ;
24.
25.
26. /* ----- QUERY 3.3 ----- */
27. select pf.Field Title, p.ID Project, p.Title, r.ID Researcher,
   r.First Name, r.Last Name
28. from project field pf
29. inner join project p on pf.ID Project = p.ID Project
30. inner join works w on p.ID_Project = w.ID_Project
31. inner join researcher r on w.ID_Researcher = r.ID_Researcher
32. where pf.Field_Title = 'Education and Human Society' /*edit
   parameters to customize query*/
33. and DATEDIFF(CURDATE(), r.Work Start Date) > 365 /* researcher
   started work on the org at least a year ago */
```

```

34.         and DATEDIFF(CURDATE(), p.Start Date) > 365      /* project
started at least a year ago */
35.         and DATEDIFF(CURDATE(), p.Finish_Date) < 0;      /* project is
active */
36.
37.
38.     /* ----- QUERY 3.4 ----- */
39.     select k.ID, o.Name, k.year, k.Number from (select n.ID as ID, n.year
as year, n.Number as Number from
40.     (select ID_org as ID, year(Start_Date) as year, count(ID_Project) as
Number from project group by ID_org, year(Start_Date)) n
41.     where (n.Number >= 10 and (n.ID, n.year+1, n.Number) in (select p.ID,
p.year, p.Number from
42.     (select ID org as ID, year(Start Date) as year, count(ID Project) as
Number from project group by ID org, year(Start Date)) p))) k
43.     inner join organization o on o.ID Org = k.ID;
44.
45.
46.     /* ----- QUERY 3.5 ----- */
47.     select sf1.Name, sf2.Name, count(pf.ID Project) as number of projects
48.     from scientific field sf1
49.     inner join scientific field sf2 on sf1.Name <> sf2.Name and sf1.Name
< sf2.Name
50.     join project_field pf on pf.Field Title = sf1.Name
51.     where (pf.ID Project, sf2.Name) in (select ID Project, Field Title
from project field)
52.     group by sf1.Name, sf2.Name
53.     order by number_of_projects DESC
54.     limit 3;
55.
56.
57.     /* ----- QUERY 3.6 ----- */
58.     select r.ID Researcher, r.First Name, r.Last Name,
TIMESTAMPDIFF(year, r.Birth Date, CURDATE()) as Age, count(p.ID Project)
as Number of Active Projects
59.     from researcher r
60.     inner join works w on r.ID_Researcher = w.ID_Researcher
61.     inner join project p on w.ID Project = p.ID Project
62.     where DATEDIFF(CURDATE(), p.Finish Date) < 0 /* check if project is
active */
63.     group by r.ID Researcher having Age < 40
64.     order by Number of Active Projects DESC
65.     limit 10;
66.
67.
68.     /* ----- QUERY 3.7 ----- */
69.     select e.First Name, e.Last Name, o.Name, p.Funding Amount
70.     from executive e

```



```

71. inner join project p on e.ID Executive = p.ID Executive
72. inner join organization o on p.ID org = o.ID Org
73. where o.Category = 'Company'
74. order by p.Funding Amount DESC
75. limit 5;
76.
77.
78. /* ----- QUERY 3.8 ----- */
79. select r.ID Researcher, r.First Name, r.Last Name,
    count(p.ID Project) as Number of Projects without Submissions
80. from researcher r
81. inner join works w on r.ID Researcher = w.ID Researcher
82. inner join project p on w.ID Project = p.ID Project
83. where p.ID Project not in (select ID Project from deliverable)
84. group by r.ID Researcher
85. having Number_of_Projects_without_Submissions > 4;

```

Για τα επρωτήματα 3.1 και 3.3 έχουμε θέσει κάποιες υποθετικές τιμές στα πεδία που επιλέγει παραμέτρους ο χρήστης, οι οποίες μπορούν να αλλάξουν για να γίνει αναζήτηση με διαφορετικά κριτήρια. Στην εφαρμογή, κατά την επιλογή των αντίστοιχων ερωτημάτων υπάρχουν κενά πεδία στα οποία συμπληρώνει ο χρήστης όσες από τις παραμέτρους αυτές επιθυμεί να περιορίσει. Επιπλέον, για το δεύτερο view επιλέξαμε να υλοποιήσουμε έναν πίνακα που δείχνει τα παραδοτέα (τίτλο και περιγραφή) για το κάθε έργο.

Παρακάτω παρατίθεται ολόκληρος ο κώδικας του αρχείου που δημιουργεί το Schema μαζί με τους δείκτες και τα views, καθώς και όλα τα triggers και τα constraints. Τον κώδικα αυτόν καθώς και ένα ξεχωριστό αρχείο με τα queries μπορείτε να τον βρείτε και στο repository, το link του οποίου υπάρχει στο τέλος της εργασίας.

```

1. create schema based;
2. use based;
3. CREATE TABLE `organization` (
4.   `ID_Org` int(11) NOT NULL AUTO_INCREMENT,
5.   `Abbreviation` varchar(5) NOT NULL,
6.   `Name` varchar(45) NOT NULL,
7.   `Street_Name` varchar(45) NOT NULL,
8.   `Street_Number` int(4) NOT NULL CHECK (`Street_Number` > 0),
9.   `Postcode` int(7) NOT NULL CHECK (`Postcode` > 0),
10.  `City` varchar(45) NOT NULL,
11.  `Category` varchar(45) NOT NULL,
12.  `Budget_Ministry_of_Education` int(9) DEFAULT NULL CHECK
    (`Budget_Ministry_of_Education` >= 0),
13.  `Budget_Private_Actions` int(9) DEFAULT NULL CHECK
    (`Budget_Private_Actions` >= 0),
14.  `Budget_Equity` int(9) DEFAULT NULL CHECK (`Budget_Equity` >= 0),
15.  PRIMARY KEY (`ID_Org`)

```

```

16.) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
17.
18.
19.CREATE TABLE `executive` (
20.  `ID_Executive` int(11) NOT NULL AUTO_INCREMENT,
21.  `First_Name` varchar(45) NOT NULL,
22.  `Last_Name` varchar(45) NOT NULL,
23.  `Branch` varchar(100) NOT NULL,
24.  PRIMARY KEY (`ID_Executive`)
25.) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
26.
27.
28.CREATE TABLE `program` (
29.  `ID_Program` int NOT NULL auto_increment,
30.  `Branch` varchar(100) NOT NULL,
31.  PRIMARY KEY (`ID_Program`)
32.) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
33.
34.CREATE TABLE `scientific_field` (
35.  `Name` varchar(50) NOT NULL,
36.  PRIMARY KEY (`Name`)
37.) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
38.
39.
40.CREATE TABLE `researcher` (
41.  `ID_Researcher` int(11) NOT NULL AUTO_INCREMENT,
42.  `First_Name` varchar(45) NOT NULL,
43.  `Last_Name` varchar(45) NOT NULL,
44.  `Gender` varchar(6) NOT NULL,
45.  `Birth_Date` date NOT NULL,
46.  `ID_org` int(11) NOT NULL,
47.  `Work_Start_Date` date NOT NULL CHECK (year(`Work_Start_Date`) -
    year(`Birth_Date`) > 0),
48.  PRIMARY KEY (`ID_Researcher`),
49.  KEY `ID_org` (`ID_org`),
50.  CONSTRAINT `ID_org_Researcher` FOREIGN KEY (`ID_org`) REFERENCES
    `organization` (`ID_Org`) ON UPDATE CASCADE ON DELETE CASCADE
51.) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
52.
53.
54.CREATE TABLE `organization_phone_numbers` (
55.  `ID_Org` int(11) NOT NULL,
56.  `Phone_Number` varchar(10) NOT NULL,
57.  PRIMARY KEY (`ID_Org`, `Phone_Number`),
58.  CONSTRAINT `ID_Org_P` foreign key (`ID_Org`) REFERENCES `organization`
    (`ID_Org`) ON DELETE CASCADE ON UPDATE CASCADE
59.) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
60.

```

```

61. CREATE TABLE `project` (
62.   `ID_Project` int(11) NOT NULL AUTO_INCREMENT,
63.   `Title` varchar(45) NOT NULL,
64.   `Description` varchar(45) NOT NULL,
65.   `Start_Date` date NOT NULL,
66.   `Finish_Date` date NOT NULL CHECK (to_days(`Finish_Date`) -
    to_days(`Start_Date`) > 0),
67.   `Funding_Amount` int(11) NOT NULL CHECK (`Funding_Amount` >= 100000 and
    `Funding_Amount` <= 1000000),
68.   `Duration` int(11) GENERATED ALWAYS AS (year(`Finish_Date`) -
    year(`Start_Date`)) VIRTUAL CHECK (`Duration` <= 4),
69.   `ID_org` int(11) NOT NULL,
70.   `ID_Researcher_in_Charge` int(11) NOT NULL,
71.   `ID_Evaluator` int(11) NOT NULL,
72.   `Evaluation_Grade` int(2) NOT NULL CHECK (`Evaluation_Grade` >= 0 and
    `Evaluation_Grade` <= 10),
73.   `Evaluation_Date` date NOT NULL CHECK (to_days(`Evaluation_Date`) -
    to_days(`Start_Date`) < 0),
74.   `ID_Program` int(11) NOT NULL,
75.   `ID_Executive` int(11) NOT NULL,
76.   PRIMARY KEY (`ID_Project`),
77.   KEY `ID_org` (`ID_org`),
78.   KEY `ID_Researcher_in_Charge` (`ID_Researcher_in_Charge`),
79.   KEY `ID_Evaluator` (`ID_Evaluator`),
80.   KEY `ID_Program` (`ID_Program`),
81.   KEY `ID_Executive` (`ID_Executive`),
82.   CONSTRAINT `ID_Evaluator_Project` FOREIGN KEY (`ID_Evaluator`) REFERENCES
    `researcher` (`ID_Researcher`) ON DELETE CASCADE ON UPDATE CASCADE,
83.   CONSTRAINT `ID_Executive_Project` FOREIGN KEY (`ID_Executive`) REFERENCES
    `executive` (`ID_Executive`) ON UPDATE CASCADE,
84.   CONSTRAINT `ID_Program_Project` FOREIGN KEY (`ID_Program`) REFERENCES
    `program` (`ID_Program`) ON UPDATE CASCADE,
85.   CONSTRAINT `ID_Researcher_in_Charge_Project` FOREIGN KEY
    (`ID_Researcher_in_Charge`) REFERENCES `researcher` (`ID_Researcher`) ON
    DELETE CASCADE ON UPDATE CASCADE,
86.   CONSTRAINT `ID_org_Project` FOREIGN KEY (`ID_org`) REFERENCES
    `organization` (`ID_Org`) ON DELETE CASCADE ON UPDATE CASCADE
87.) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
88.
89.
90.
91. CREATE TABLE `deliverable` (
92.   `ID_Project` int(11) NOT NULL,
93.   `Title` varchar(45) NOT NULL,
94.   `Description` varchar(45) NOT NULL,
95.   `Delivery_Date` date NOT NULL,
96.   PRIMARY KEY (`ID_Project`, `Title`),

```

```

97.  CONSTRAINT `ID_Project_del` FOREIGN KEY (`ID_Project`) references
    `project` (`ID_Project`) on delete cascade on update cascade
98.) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
99.
100.  CREATE TABLE `works` (
101.      `ID_Project` int(11) NOT NULL,
102.      `ID_Researcher` int(11) NOT NULL,
103.      PRIMARY KEY (`ID_Project`, `ID_Researcher`),
104.      KEY `ID_Project` (`ID_Project`),
105.      KEY `ID_Researcher` (`ID_Researcher`),
106.      CONSTRAINT `ID_Researcher_Works` FOREIGN KEY (`ID_Researcher`)
    REFERENCES `researcher` (`ID_Researcher`) ON DELETE cascade ON UPDATE
    CASCADE,
107.      CONSTRAINT `ID_Project_Works` FOREIGN KEY (`ID_Project`) REFERENCES
    `project` (`ID_Project`) ON DELETE cascade ON UPDATE CASCADE
108.  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
109.
110.  CREATE TABLE `project_field` (
111.      `ID_Project` int(11) NOT NULL,
112.      `Field_Title` varchar(50) NOT NULL,
113.      PRIMARY KEY (`ID_Project`, `Field_Title`),
114.      KEY `ID_Project` (`ID_Project`),
115.      KEY `Field_Title` (`Field_Title`),
116.      CONSTRAINT `ID_Project_Field` FOREIGN KEY (`ID_Project`) REFERENCES
    `project` (`ID_Project`) ON DELETE CASCADE ON UPDATE CASCADE,
117.      CONSTRAINT `Title_Field` FOREIGN KEY (`Field_Title`) REFERENCES
    `scientific_field` (`Name`) ON DELETE CASCADE ON UPDATE CASCADE
118.  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
119.
120.  DELIMITER $$
121.  create trigger hmmm before insert on works
122.  FOR each row
123.  begin
124.      if (new.ID_Researcher = (select ID_Evaluator as m from project
    where ID_Project = new.ID_Project)) then
125.
126.      signal sqlstate '45000' SET MESSAGE_TEXT = 'A researcher
    cannot work in a project which he evaluated.';
127.      END IF;
128.  end;
129.  $$
130.  DELIMITER ;
131.
132.  DELIMITER $$
133.  create trigger mhh before update on project
134.  for each row
135.  begin

```

```

136.             if (new.ID_Evaluator = (select ID_Researcher from works
where (ID_Project = new.ID_Project and ID_Researcher = new.ID_Evaluator)))
then
137.             signal sqlstate '45000' SET MESSAGE_TEXT = 'This researcher
works or worked in the selected project. Therefore he cannot be its
evaluator';
138.             end if;
139.         end;
140.     $$
141. DELIMITER ;
142.
143. DELIMITER $$
144. create trigger insp after insert on project
145. for each row
146. begin
147.         INSERT INTO works (`ID_Project`, `ID_Researcher`) values
(new.ID_Project, new.ID_Researcher_in_Charge);
148.     end;
149.     $$
150. DELIMITER ;
151.
152. DELIMITER $$
153. create trigger updp after update on project
154. for each row
155. begin
156.         if ((datediff(curdate(),new.Finish_Date) < 0) and
new.ID_Researcher_in_charge not in (select ID_Researcher from works where
ID_Project = new.ID_Project)) then
157.         INSERT INTO works (`ID_Project`, `ID_Researcher`) values
(new.ID_Project, new.ID_Researcher_in_Charge);
158.         end if;
159.     end;
160.     $$
161. DELIMITER ;
162.
163. DELIMITER $$
164. create trigger delr before delete on researcher
165. for each row
166. begin
167.         if (old.ID_Researcher in (select ID_Evaluator from
project)) then
168.         signal sqlstate '45000' SET MESSAGE_TEXT = 'You cannot delete
an evaluator. To do so set a new evaluator to the corresponding project.';
169.         end if;
170.     end;
171.     $$
172. DELIMITER ;
173.

```

```

174.      DELIMITER $$
175.      create trigger delr2 before delete on researcher
176.      for each row
177.      begin
178.          if (old.ID_Researcher in (select ID_Researcher_in_Charge
from project)) then
179.              signal sqlstate '45000' SET MESSAGE_TEXT = 'You cannot delete
a chief. To do so set a new chief to the corresponding project.';
180.          end if;
181.          end;
182.      $$
183.  DELIMITER ;
184.
185.  DELIMITER $$
186.  create trigger delex before delete on executive
187.  for each row
188.  begin
189.      if (old.ID_Executive in (select ID_Executive from
project)) then
190.          signal sqlstate '45000' SET MESSAGE_TEXT = 'You cannot delete
this executive. To do so set a new executive to the corresponding project.';
191.      end if;
192.      end;
193.      $$
194.  DELIMITER ;
195.
196.  DELIMITER $$
197.  create trigger delprf before delete on project_field
198.  for each row
199.  begin
200.      if ((select count(old.ID_Project) from project_field where
ID_Project = old.ID_Project) = 1) then
201.          signal sqlstate '45000' SET MESSAGE_TEXT = 'Every project must
have at least one Field Title.';
202.      end if;
203.      end;
204.      $$
205.  DELIMITER ;
206.
207.  DELIMITER $$
208.  create trigger rescheck before insert on works
209.  for each row
210.  begin
211.      if ((select ID_org from project where ID_Project =
new.ID_Project) <> (select ID_org from researcher where ID_Researcher =
new.ID_Researcher)) then
212.          signal sqlstate '45000' SET MESSAGE_TEXT = 'A researcher can only
work in projects which belong to their organization.';

```

```

213.         end if;
214.     end;
215.     $$
216. DELIMITER ;
217.
218. DELIMITER $$
219. create trigger resc before delete on works
220. for each row
221. begin
222.     if (old.ID_Researcher = (select ID_Researcher_in_Charge from
project where ((ID_Project = old.ID_Project) and (datediff(curdate(),
Finish_Date) < 0)))) then
223.         signal sqlstate '45000' SET MESSAGE_TEXT = 'A chief researcher
cannot be deleted when the project is active. To do so you must first set a
new chief.';
224.     end if;
225. end;
226. $$
227. DELIMITER ;
228.
229. DELIMITER $$
230. create trigger budget before insert on organization
231. for each row
232. begin
233.     if ((new.Category = 'Company') and (new.Budget_Equity = null))
then
234.         signal sqlstate '45000' SET MESSAGE_TEXT = 'A company must have a
not null Equity Budget';
235.     elseif (new.Category = 'Research Center' and
(new.Budget_Ministry_of_Education = null or new.Budget_Private_Actions =
null)) then
236.         signal sqlstate '45000' SET MESSAGE_TEXT = 'A research center must
have a not null Budget from Ministry of Education and from Private Actions';
237.     elseif (new.Category = 'University' and
new.Budget_Ministry_of_Education = null) then
238.         signal sqlstate '45000' SET MESSAGE_TEXT = 'A university must have
a not null Budget from Ministry of Education';
239.     end if;
240. end;
241. $$
242. DELIMITER ;
243.
244. DELIMITER $$
245. create trigger budgetn before insert on organization
246. for each row
247. begin

```

```

248.         if ((new.Category = 'Company') and
            (new.Budget_Ministry_of_Education >=0 or new.Budget_Private_Actions >=0))
            then
249.             signal sqlstate '45000' SET MESSAGE_TEXT = 'A company cannot have
                a not null Budget from Ministry of Education or from Private Actions';
250.             elseif (new.Category = 'Research Center' and (new.Budget_Equity >=
                0)) then
251.                 signal sqlstate '45000' SET MESSAGE_TEXT = 'A research center
                    cannot have a not null Equity Budget';
252.                 elseif (new.Category = 'University' and (new.Budget_Equity >=0 or
                    new.Budget_Private_Actions >=0)) then
253.                     signal sqlstate '45000' SET MESSAGE_TEXT = 'A university cannot
                        have a not null budget from Equity or from Private Actions';
254.                     end if;
255.                 end;
256.             $$
257.         DELIMITER ;
258.
259.
260.     DELIMITER $$
261.     create trigger budgetu before update on organization
262.     for each row
263.     begin
264.         if ((new.Category = 'Company') and
            (new.Budget_Ministry_of_Education >=0 or new.Budget_Private_Actions >=0))
            then
265.             signal sqlstate '45000' SET MESSAGE_TEXT = 'A company cannot have
                a not null Budget from Ministry of Education or from Private Actions';
266.             elseif (new.Category = 'Research Center' and (new.Budget_Equity >=
                0)) then
267.                 signal sqlstate '45000' SET MESSAGE_TEXT = 'A research center
                    cannot have a not null Equity Budget';
268.                 elseif (new.Category = 'University' and (new.Budget_Equity >=0 or
                    new.Budget_Private_Actions >=0)) then
269.                     signal sqlstate '45000' SET MESSAGE_TEXT = 'A university cannot
                        have a not null budget from Equity or from Private Actions';
270.                     end if;
271.                 end;
272.             $$
273.     DELIMITER ;
274.
275.     DELIMITER $$
276.     create trigger inr before insert on researcher
277.     for each row
278.     begin
279.         if ((dayname(new.Work_Start_Date) is null) or
            (dayname(new.Birth_Date) is null)) then

```



```

280.         signal sqlstate '45000' set message_text = 'You inserted a non
        valid date type. Insert a (YY-MM-DD) date.';
281.     end if;
282. end;
283. $$
284. DELIMITER ;
285.
286. DELIMITER $$
287. create trigger upr before update on researcher
288. for each row
289. begin
290.     if ((dayname(new.Work_Start_Date) is null) or
        (dayname(new.Birth_Date) is null)) then
291.         signal sqlstate '45000' set message_text = 'You inserted a non
        valid date type. Insert a (YY-MM-DD) date.';
292.     end if;
293. end;
294. $$
295. DELIMITER ;
296.
297. DELIMITER $$
298. create trigger inp before insert on project
299. for each row
300. begin
301.     if ((dayname(new.Start_Date) is null) or
        (dayname(new.Finish_Date) is null) or (dayname(new.Evaluation_Date))) then
302.         signal sqlstate '45000' set message_text = 'You inserted a non
        valid date type. Insert a (YY-MM-DD) date.';
303.     end if;
304. end;
305. $$
306. DELIMITER ;
307.
308. DELIMITER $$
309. create trigger upp before update on project
310. for each row
311. begin
312.     if ((dayname(new.Start_Date) is null) or
        (dayname(new.Finish_Date) is null) or (dayname(new.Evaluation_Date))) then
313.         signal sqlstate '45000' set message_text = 'You inserted a non
        valid date type. Insert a (YY-MM-DD) date.';
314.     end if;
315. end;
316. $$
317. DELIMITER ;
318.
319. DELIMITER $$
320. create trigger ind before insert on deliverable

```

```

321.         for each row
322.         begin
323.             if (dayname(new.Delivery_Date) is null) then
324.                 signal sqlstate '45000' set message_text = 'You inserted a non
valid date type. Insert a (YY-MM-DD) date.';
325.             end if;
326.             end;
327.         $$
328.     DELIMITER ;
329.
330.     DELIMITER $$
331.     create trigger upd before insert on deliverable
332.     for each row
333.     begin
334.         if (dayname(new.Delivery_Date) is null) then
335.             signal sqlstate '45000' set message_text = 'You inserted a non
valid date type. Insert a (YY-MM-DD) date.';
336.         end if;
337.         end;
338.     $$
339.     DELIMITER ;
340.
341.     DELIMITER $$
342.     create trigger resup before update on researcher
343.     for each row
344.     begin
345.         if ((new.ID_org <> old.ID_org) and (old.ID_Researcher in
(select w.ID_Researcher as ID from works w inner join project p on
w.ID_Project=p.ID_Project where datediff(curdate(), p.Finish_Date) < 0 group
by ID))) then
346.             signal sqlstate '45000' set message_text = 'You cannot change the
organization when the researcher is currently working in a project. To do so
first delete from works.';
347.             end if;
348.         end;
349.     $$
350.     DELIMITER ;
351.
352.     DELIMITER $$
353.     create trigger resupc after update on researcher
354.     for each row
355.     begin
356.         if ((new.ID_org <> old.ID_org)) then
357.             delete from works where ID_Researcher = old.ID_Researcher;
358.         end if;
359.         end;
360.     $$
361.     DELIMITER ;

```

```

362.
363.     DELIMITER $$
364.     create trigger scifii before delete on scientific_field
365.     for each row
366.     begin
367.         if (old.Name in (select Field_Title from project_field group by
            Field_Title)) then
368.             signal sqlstate '45000' set message_text = 'You cannot delete a
                scientific field which characterizes a project';
369.         end if;
370.     end;
371.     $$
372.     DELIMITER ;
373.
374.     create index startdate on project (Start_Date);
375.     create index finishdate on project (Finish_Date);
376.     create index birthdate on researcher (Birth_Date);
377.
378.     create view submissions as
379.     select p.ID_Project, p.Title as Name, d.Title as Submission_Title,
            d.Description
380.     from project p inner join deliverable d on p.ID_Project =
            d.ID_Project;
381.
382.     create view researcher_projects as
383.     SELECT r.ID_Researcher, r.First_Name, r.Last_Name, p.ID_Project,
            p.Title as Project_Title, p.Funding_Amount
384.     from researcher r
385.     inner join works w on r.ID_Researcher = w.ID_Researcher
386.     inner join project p on w.ID_Project = p.ID_Project;
387.

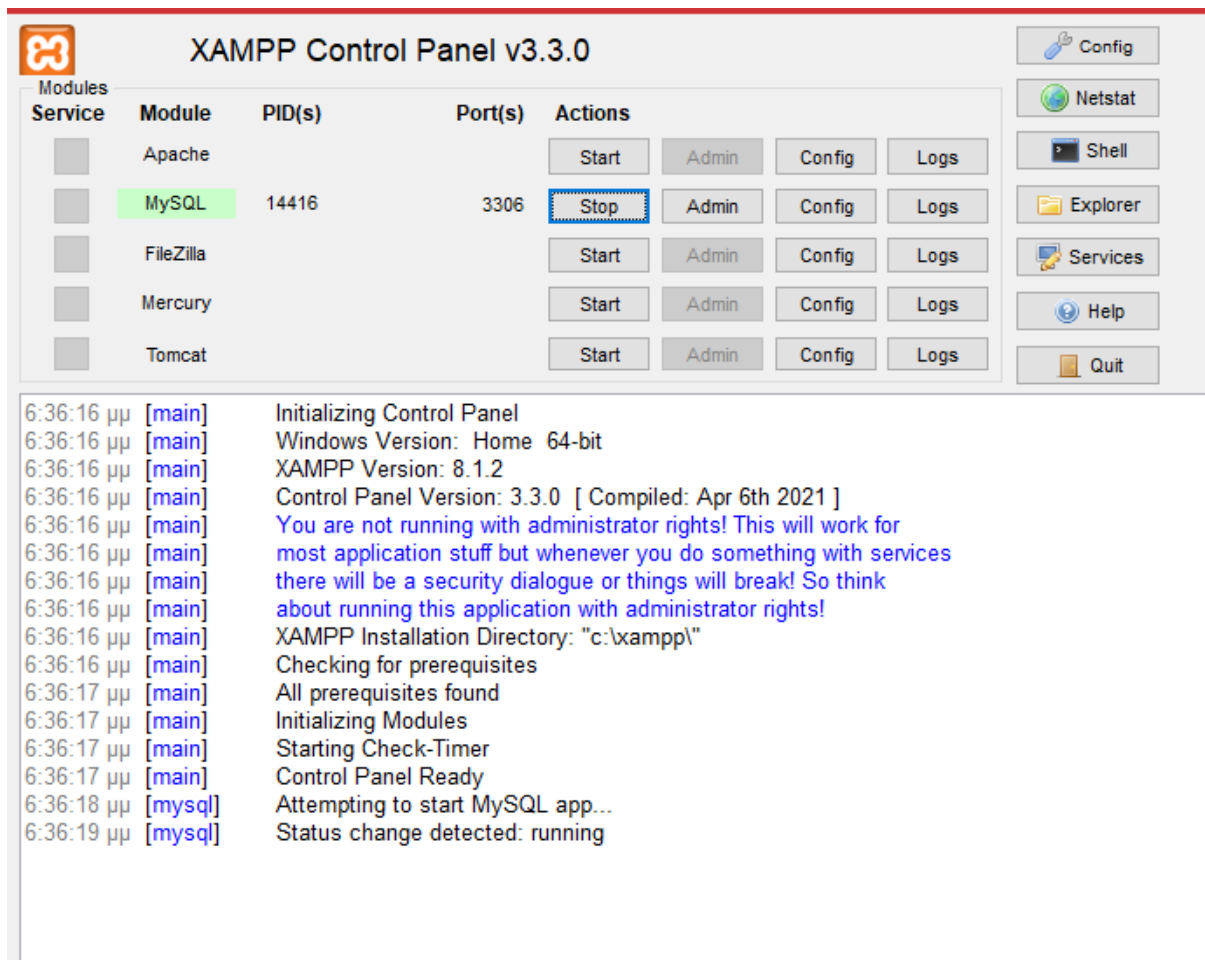
```

2.6 Εγκατάσταση της Εφαρμογής

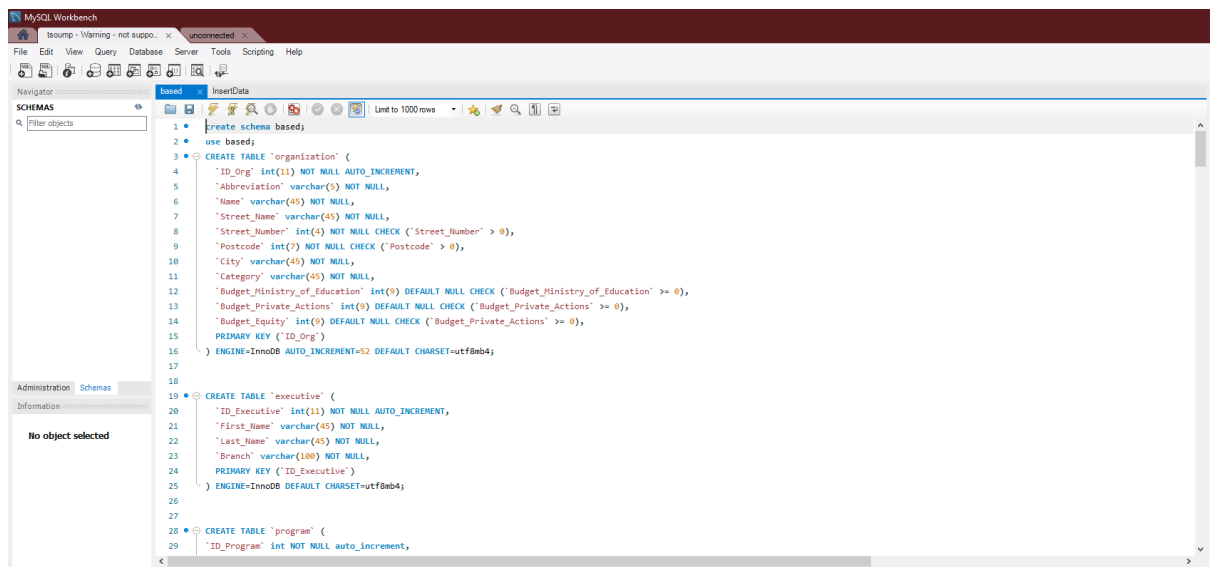
Η δημιουργία του Schema και η εισαγωγή των dummy data γίνεται μέσω του MySQL Workbench αλλά μπορεί να χρησιμοποιηθεί και οποιοδήποτε άλλο πρόγραμμα με βάση την MySQL. Η εφαρμογή της βάσης δεδομένων μας τρέχει σε τοπικό σέρβερ (localhost) με τη χρήση του Flask framework της Python. Η σελίδα της εφαρμογής αποτελείται από απλά αρχεία HTML. Ακολουθούν αναλυτικά βήματα για την εγκατάσταση όλων των απαιτούμενων προγραμμάτων και βιβλιοθηκών.

Εγκατάσταση MySQL Workbench

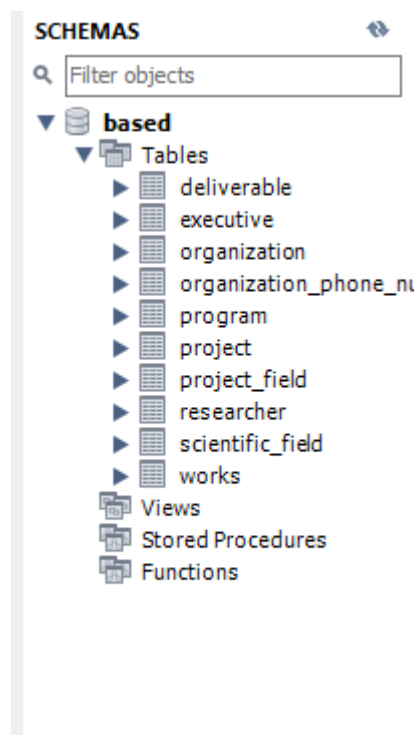
Εγκαθιστούμε αρχικά το XAMPP για να μπορέσουμε να συνδεθούμε με την MySQL Database.



Αφού ενεργοποιήσουμε το πεδίο MySQL στην αντίστοιχη εφαρμογή, το επόμενο βήμα είναι να ανοίξουμε το Workbench της επιλογής μας. Χρησιμοποιούμε MySQL Workbench και ανοίγουμε τα sql scripts με όνομα based.sql και InsertData.sql, που βρίσκονται στο github repository.



Εκτελούμε πρώτα το based.sql για τη δημιουργία του Schema με τα tables, τα views, τα triggers και τα indexes και κατόπιν εκτελούμε το InsertData.sql για την εισαγωγή των δεδομένων. Θα πρέπει αφού πατήσουμε refresh στο πάνελ Schemas να εμφανιστούν όλα τα tables του database based.



Εγκατάσταση Python και Flask Framework

Μπορείτε να χρησιμοποιήσετε οποιονδήποτε Python IDE θέλετε για να εκτελέσετε το αρχείο app.py που θα βρείτε στο repository. Συνιστάται να χρησιμοποιήσετε την τελευταία έκδοση του Anaconda Navigator που περιλαμβάνει εγκατάσταση της Python και του IDE Spyder καθώς αυτό χρησιμοποιήσαμε και εμείς.

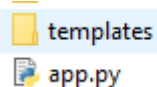
Έχοντας εγκατεστημένη την Python, κατεβάζουμε το Flask framework με την εντολή `pip install Flask` από το directory που είναι εγκατεστημένη η Python ή από το Anaconda Powershell Prompt. Ακριβώς με τον ίδιο τρόπο εκτελούμε στη συνέχεια την εντολή `pip install flask-mysqldb`. Αυτές είναι και οι βιβλιοθήκες που θα χρειαστούμε.

Στο σημείο αυτό ωστόσο αξίζει να επισημανθεί ότι σε περίπτωση που κάποιος προσπαθήσει να εκτελέσει την εφαρμογή σε κάποιον άλλον editor, τότε είναι πιθανό να χρειαστεί να εγκαταστήσει και άλλα packages. Το Anaconda ως γνωστόν παρέχει ήδη από την εγκατάστασή του μια αρκετά μεγάλη συλλογή βιβλιοθηκών της Python οπότε υπάρχει περίπτωση σε άλλους editors ορισμένες βιβλιοθήκες να μην είναι preinstalled. Σε κάθε περίπτωση πάντως εκτός από το flask library χρειάζονται τα παρακάτω requirements.

```
Werkzeug >= 0.15
click >= 5.1
Faker ==13.3.4
Jinja2 >= 2.10.1
itsdangerous >= 0.24
MarkupSafe >= 0.23
colorama
mysqlclient >= 1.3.7
Flask >= 0.12.4
six >= 1.5
python-dateutil >= 2.4
```

Εφόσον όλα τα παραπάνω έχουν εγκατασταθεί τότε προκειμένου να τρέξουμε την εφαρμογή μας εκτελούμε τα εξής βήματα:

1. Δημιουργούμε έναν φάκελο σε κάποιο directory του υπολογιστή μας.
2. Στον φάκελο αυτό εισάγουμε το αρχείο `app.py` που βρίσκεται στο repository και ταυτόχρονα δημιουργούμε και έναν νέο φάκελο με το όνομα `templates`.



3. Εντός του φάκελου `templates` εισάγουμε όλα τα αρχεία `html` που βρίσκονται στον φάκελο `templates` του repository μας.
4. Η εφαρμογή μας είναι τώρα έτοιμη να ξεκινήσει (εφόσον βέβαια έχουμε ενεργοποιήσει τη σύνδεση με τη βάση μέσω του XAMPP). Για τον σκοπό αυτόν ανοίγουμε το cmd και μεταφερόμαστε στο αρχικό directory στο οποίο έχουμε εισάγει τον φάκελο `templates` και το αρχείο `app.py`.
5. Εκτελούμε την εντολή `"flask run"` στο terminal και πρέπει να λάβουμε το αντίστοιχο αποτέλεσμα:

```
C:\Users\Apostolos\Desktop\app>flask run
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

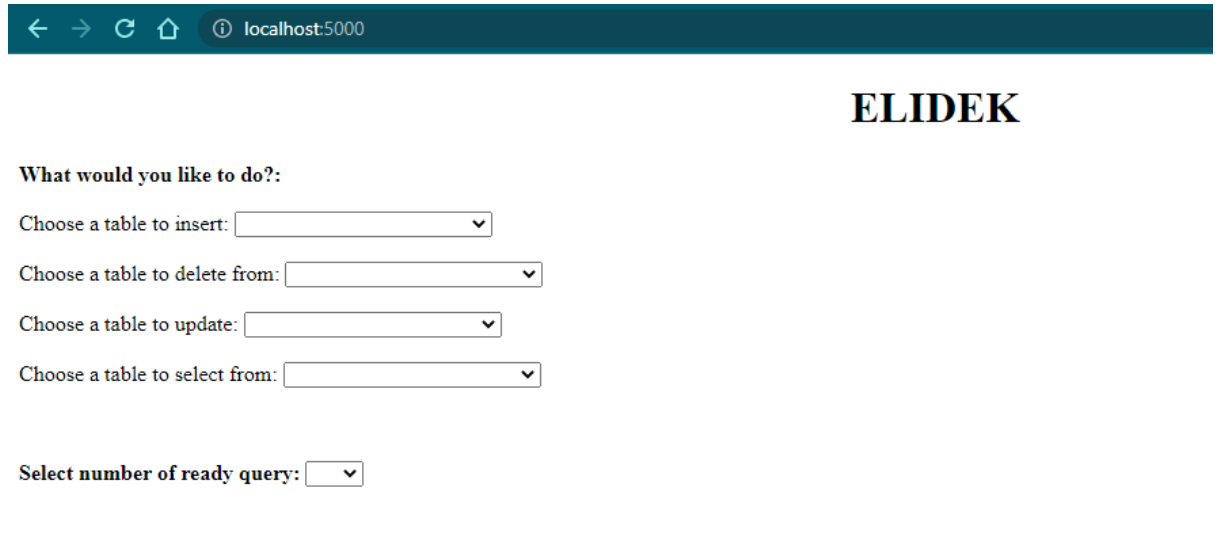
Παρατηρούμε πως ο σέρβερ μας έχει ενεργοποιηθεί και η εφαρμογή μας εκτελείται. Στο σημείο αυτό αναφέρουμε πως το port στο οποίο θα εμφανίζεται η σελίδα της εφαρμογής

μας μπορεί να επιλεγεί με χρήση κατάλληλης εντολής. Εμείς χρησιμοποιήσαμε το 5000 που είναι το default port της flask όπως φαίνεται και παρακάτω.

port 5000

By default, Flask runs on **port 5000** in development mode.

- Τέλος ανοίγουμε κάποιον browser (χωρίς να απαιτείται σύνδεση στο δίκτυο) και δίνουμε ως url το εξής: `http://localhost:5000/`. Παρατηρούμε πως η εφαρμογή μας εκτελείται κανονικά.



Να σημειωθεί ότι η εφαρμογή έχει ενσωματωμένες κατευθυντήριες γραμμές για τον χρήστη, όμως πρέπει να ξεκαθαρίσουμε μερικά λεπτά σημεία.

Υπάρχουν ορισμένοι περιορισμοί που προκύπτουν λογικά από τον τρόπο που έχει δημιουργηθεί η βάση δεδομένων. Ο χρήστης δεν μπορεί να διαγράψει, για παράδειγμα, Προγράμματα που έχουν έστω και ένα Έργο, διότι τότε θα υπήρχαν Έργα χωρίς Πρόγραμμα. Μπορεί όμως να διαγράψει έναν οργανισμό, και μαζί του διαγράφονται αυτόματα και όλα τα έργα και οι ερευνητές αυτού του οργανισμού καθώς και οι εργασιακές σχέσεις μεταξύ τους.

Επίσης, κατά την ανανέωση των δεδομένων στους πίνακες Works και Project_Field και Organization_Phone_Numbers δίνεται η δυνατότητα για update με βάση τη μία τιμή από τις δύο. Συγκεκριμένα, ο χρήστης μπορεί να επιλέξει τον Ερευνητή του οποίου θέλει να ανανεώσει κάποια εργασιακή σχέση, και κατόπιν να του αναθέσει ένα διαφορετικό έργο από τις έγκυρες επιλογές. Σε αντίθετη περίπτωση, δηλαδή σε περίπτωση που μια καταχώρηση στον πίνακα Works έχει λάθος ID_Researcher (αν πληκτρολόγησε λάθος ο χρήστης), τότε για να ενημερωθεί η σχέση αυτή, θα πρέπει να διαγραφεί και να εισαχθεί εκ νέου με τις σωστές τιμές.

Με παρόμοια λογική λειτουργούν και οι πίνακες Project_Field και Organization_Phone_Numbers με την διαφορά ότι τη θέση του ID_Researcher έχουν τώρα τα ID_Project και ID_Org αντίστοιχα.

Θεωρήσαμε αυτόν τον περιορισμό χρήσιμο καθώς καθοδηγούμε τον χρήστη σε ευκολότερη επιλογή των πεδίων που επιθυμεί να ανανεώσει αλλά κυρίως, με τον τρόπο αυτό, αποτρέπουμε την διαγραφή και εισαγωγή και των δύο πεδίων του πίνακα ταυτόχρονα μέσω ενός update statement, κάτι το οποίο κατά τη γνώμη μας αυξάνει την πολυπλοκότητα της εφαρμογής μας.

Επιπλέον, στον πίνακα Organization, ανάλογα με την κατηγορία που επιλέγει ο χρήστης (Εταιρία, Πανεπιστήμιο, Ερευνητικό Κέντρο) κατά τη διάρκεια των update ή insert statement, κάναμε τις εξής παραδοχές:

- Κανένας προϋπολογισμός δεν μπορεί να έχει αρνητική τιμή.
- Τα πεδία των προϋπολογισμών που δεν είναι έγκυρα για μια επιλεγμένη κατηγορία θέτονται ως NULL (None στην MySQL).
- Τα πεδία που είναι έγκυρα για την επιλεγμένη κατηγορία έχουν default τιμή το 0 σε περίπτωση που ο χρήστης αφήσει το πεδίο κενό ή εισάγει μη έγκυρη τιμή. Αυτό είναι απαραίτητο για τον διαχωρισμό των έγκυρων από τα μη-έγκυρα πεδία ακόμα και με εποπτική επισκόπηση του πίνακα Organization από τον χρήστη.

Στα update statements ο χρήστης μπορεί να αφήσει ένα πεδίο κενό αν δεν επιθυμεί να αλλάξει την τιμή του.

Θεωρούμε πως η εφαρμογή είναι εύχρηστη ακόμα και για κάποιον χρήστη που δεν είναι γνώστης του αντικειμένου Βάσεις Δεδομένων.

2.7 Σύνδεσμος για το Git Repository της εφαρμογής μας