Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

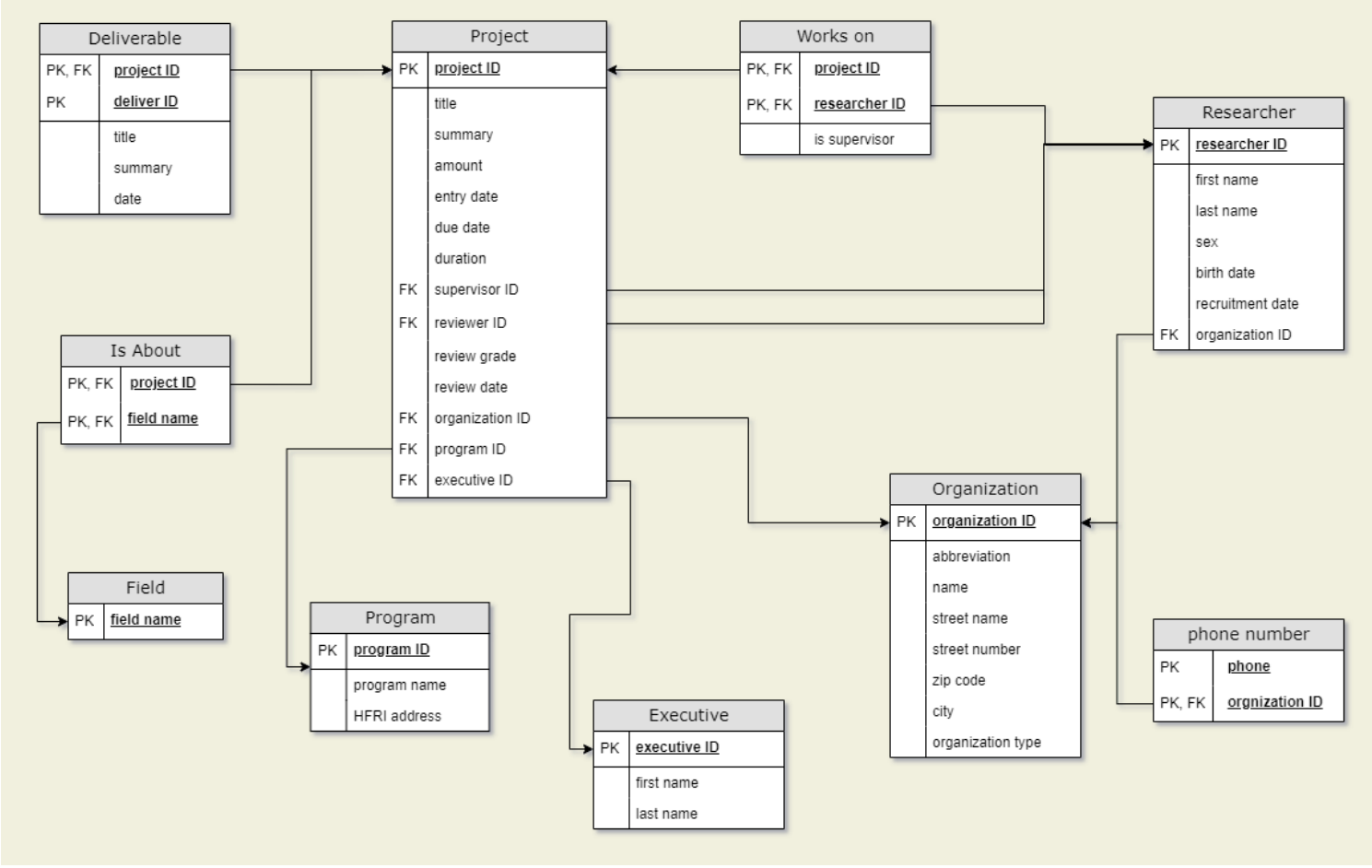
Βάσεις Δεδομένων, 2021-2022

Αναφορά Εξαμηνιαίας Εργασίας Βάσεων Δεδομένων

Στοιχεία Φοιτητών:

Παναγιώτης Γιαδικιάρογλου	**03119185**
Μιχαήλ Άγγελος Βελαλόπουλος	**03119908**

1) Το σχεσιακό διάγραμμα της Βάσης Δεδομένων που κατασκευάσαμε δίνεται παρακάτω:



Κάθε σχέση του Relational Diagram που αντιστοιχεί σε entity του ER Diagram διαθέτει ένα αυθαίρετο και μοναδικό ID το οποίο μας βοηθά για να έχουμε την μοναδικότητα του κάθε tuple που απαιτείται και να μην έχουμε duplicates. Το σχεσιακό διάγραμμα αποτελείται από τα ακόλουθα tables:

- **Field**: Το table αυτό έχει μόνο ένα attribute το οποίο είναι και το primary key του και αυτό είναι το field_name
- **Is_about**: Σχέση η οποία συνδέει τα projects με διάφορα επιστημονικά πεδία (field). Το primary key της σχέσης αυτής αποτελείται από δύο attributes, το project_id και το field_name, τα οποία είναι foreign keys
- **Program**: Με attributes το όνομα προγράμματος και τη διεύθυνση του ΕΛΙΔΕΚ στην οποία ανήκει
- **Executive**: Με attributes όνομα και επώνυμο
- **Organization**: Με attributes την συντομογραφία, το όνομα, τη διεύθυνση (οδό, αριθμό, Τ.Κ., πόλη) καθώς τα composed attributes δίνουν τα επιμέρους χαρακτηριστικά τους στην σχέση, και τέλος τον τύπο του οργανισμού (πανεπιστήμιο, ερευνητικό κέντρο, εταιρία)
- **Phone number**: Αφού το attribute phone number ήταν multivalued στο ER τότε απαιτείται να αναπαρασταθεί ως ξεχωριστό relation στο σχεσιακό διάγραμμα
- **Researcher**: Με attributes το ονοματεπώνυμο, το φύλο, την ημερομηνία γέννησης και τον οργανισμό για τον οποίο εργάζεται καθώς και την ημερομηνία πρόσληψής του από αυτόν (δεν χρειάζεται ξεχωριστή σχέση για αυτό καθώς ένας ερευνητής μπορεί, και πρέπει, να εργάζεται μόνο σε έναν οργανισμό
- **Project**: Με attributes τον τίτλο του έργου, την περίληψη, το ποσό, την ημερομηνία έναρξης και ημερομηνία λήξης, τη διάρκεια (η οποία είναι derived attribute και προκύπτει από τα δύο προηγούμενα), την αξιολόγησή του και την ημερομηνία αξιολόγησης. Επίσης εφόσον ένα έργο ανήκει σε ένα μόνο πρόγραμμα, έναν οργανισμό και διαχειρίζεται από ένα μόνο στέλεχος τότε αποφασίζουμε το project να έχει ως foreign keys τα ID’s των προαναφερθέντων πινάκων. Τέλος, αφού έχει μόνο έναν επιστημονικό υπεύθυνο και αυτός είναι ερευνητής τότε έχει ως foreign key το ID του αντίστοιχου researcher, ενώ το ίδιο ισχύει και για τον reviewer, ο οποίος όμως δεν μπορεί να ανήκει στον ίδιο οργανισμό με το project
- **Works_on**: Σχέση που αναπαριστά την εργασία ενός ερευνητή πάνω σε ένα συγκεκριμένο έργο. Διαθέτει δύο attributes για primary key, όμοια με τη σχέση is_about, τα οποία είναι τα foreign keys project_id και researcher_id. Επίσης έχουμε ένα attribute το οποίο καθορίζει αν η σχέση αφορά τον supervisor του έργου
- **Deliverable**: Διαθέτει δύο attributes ως primary key, το project_id που είναι και foreign key καθώς και το deliverable_id. Δεν χρειάζεται ενδιάμεση σχέση διότι

κάθε παραδοτέο αφορά μόνο ένα έργο. Επίσης έχουμε ως attributes και τον τίτλο, την περίληψη και την ημερομηνία παράδοσης του παραδοτέου.

Στη συνέχεια, προσθέτουμε κατάλληλα indexes προκειμένου να επιταχύνουμε την εκτέλεση των ζητούμενων queries. Δεδομένου ότι κάθε primary key έχει ενσωματωμένο εξ' αρχής κάποιο index επιλέγουμε να βάλουμε ευρετήρια στα παρακάτω attributes που χρησιμοποιούνται συχνότερα στην διάσχιση των πινάκων:

```
-- Create Indexes

CREATE INDEX idx_organization_id ON project(organization_id);
CREATE INDEX idx_entry_date ON project(entry_date);
CREATE INDEX idx_due_date ON project(due_date);
CREATE INDEX idx_birth_date ON researcher(birth_date);
```

Ένα παράδειγμα της έντονης χρήσης του project.organization_id αποτελεί το query 3.4 το οποίο βελτιστοποιείται με τη χρήση του κατάλληλου index:

```
CREATE VIEW query4_helper AS
SELECT count(*) AS num, organization_id, YEAR(entry_date) AS year
FROM project
GROUP BY YEAR(entry_date), organization_id
ORDER BY organization_id;

CREATE VIEW org_with_same_num_of_proj AS
SELECT helper1.num AS num1, helper1.organization_id, organization.name,
helper1.year AS first_year, helper2.year AS last_year
FROM query4_helper helper1
INNER JOIN query4_helper helper2 ON helper1.organization_id =
helper2.organization_id AND helper1.year = helper2.year+1
INNER JOIN organization ON organization.organization_id =
helper1.organization_id
WHERE helper1.num = helper2.num AND helper1.num >= 10;
```

2) DDL Script

```
DROP SCHEMA IF EXISTS hfri;
CREATE SCHEMA hfri;
USE hfri;

-- Create program table
CREATE TABLE program
(
    program_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    program_name VARCHAR(45) NOT NULL,
    hfri_address VARCHAR(45) NOT NULL,
    last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    PRIMARY KEY (program_id)
);

-- Create executive table
CREATE TABLE executive
(
  executive_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
  first_name VARCHAR(45) NOT NULL,
  last_name VARCHAR(45) NOT NULL,
  last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  PRIMARY KEY (executive_id)
);

-- Create organization table
CREATE TABLE organization
(
    organization_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    abbreviation VARCHAR(20) NOT NULL,
    name VARCHAR(50) NOT NULL,
    street_name VARCHAR(45) NOT NULL,
    street_number SMALLINT UNSIGNED NOT NULL,
    zip_code VARCHAR(5) NOT NULL,
    city VARCHAR(45) NOT NULL,
    organization_type ENUM('university', 'research_center', 'company') NOT
NULL,
    last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    PRIMARY KEY (organization_id)
);
```

```sql
-- Create  phone_number table
CREATE TABLE phone_number
(
    phone VARCHAR(15) NOT NULL,
    organization_id SMALLINT UNSIGNED NOT NULL,
    last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    PRIMARY KEY (phone, organization_id),
    CONSTRAINT fk_phone_number_organization FOREIGN KEY (organization_id)
REFERENCES organization (organization_id) ON DELETE CASCADE ON UPDATE
CASCADE
);

-- Create researcher table
CREATE TABLE researcher
(
    researcher_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    first_name VARCHAR(45) NOT NULL,
    last_name VARCHAR(45) NOT NULL,
    sex ENUM('male', 'female') NOT NULL,
    birth_date DATE NOT NULL,
    recruitment_date DATE NOT NULL,
    organization_id SMALLINT UNSIGNED NOT NULL,
    last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    PRIMARY KEY (researcher_id),
    CONSTRAINT fk_researcher_organization FOREIGN KEY (organization_id)
REFERENCES organization (organization_id) ON DELETE CASCADE ON UPDATE
CASCADE
);

-- Create project table. Add constraints for amount and review_date
CREATE TABLE project
(
    project_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    title VARCHAR(45) NOT NULL,
    summary TEXT,
    amount INT NOT NULL,
    entry_date DATE NOT NULL,
    due_date DATE NOT NULL,
    program_id SMALLINT UNSIGNED NOT NULL,
    executive_id SMALLINT UNSIGNED NOT NULL,
    organization_id SMALLINT UNSIGNED NOT NULL,
    supervisor_id SMALLINT UNSIGNED NOT NULL,
    reviewer_id SMALLINT UNSIGNED NOT NULL,
```

```
    review_grade ENUM('1','2','3','4','5') NOT NULL,
    review_date DATE NOT NULL,
    PRIMARY KEY (project_id),
    CONSTRAINT fk_project_program FOREIGN KEY (program_id) REFERENCES
program (program_id) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_project_executive FOREIGN KEY (executive_id) REFERENCES
executive (executive_id) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_project_organization FOREIGN KEY (organization_id)
REFERENCES organization (organization_id) ON DELETE CASCADE ON UPDATE
CASCADE,
    CONSTRAINT fk_project_supervisor FOREIGN KEY (supervisor_id)
REFERENCES researcher (researcher_id) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_project_reviewer FOREIGN KEY (reviewer_id) REFERENCES
researcher (researcher_id) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT check_amount CHECK (amount BETWEEN 100000 AND 1000000),
    CONSTRAINT check_review_date CHECK ((review_date < entry_date) AND
(review_date >= '1950-01-01'))
);

-- Add duration and active attributes to project table. Duration has to be
between 1 and 4 years
ALTER TABLE project
    ADD COLUMN duration INT AS (DATEDIFF(due_date, entry_date) DIV 365)
AFTER due_date,
    ADD COLUMN active BOOLEAN AS (IF ((CURRENT_DATE() > due_date), 0, 1))
AFTER duration,
    ADD COLUMN last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP,
    ADD CONSTRAINT check_duration CHECK (duration BETWEEN 1 AND 4);

-- Create works_on table
CREATE TABLE works_on
(
    project_id SMALLINT UNSIGNED NOT NULL,
    researcher_id SMALLINT UNSIGNED NOT NULL,
    is_supervisor BOOLEAN DEFAULT 0 NOT NULL,
    last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    PRIMARY KEY (project_id, researcher_id),
    CONSTRAINT fk_works_on_project FOREIGN KEY (project_id) REFERENCES
project (project_id) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_works_on_researcher FOREIGN KEY (researcher_id)
REFERENCES researcher (researcher_id) ON DELETE CASCADE ON UPDATE CASCADE
);
```

```sql
-- Create deliverable table
CREATE TABLE deliverable
(
    deliver_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    project_id SMALLINT UNSIGNED NOT NULL,
    title VARCHAR(45) NOT NULL,
    summary TEXT,
    deliver_date DATE NOT NULL,
    last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    PRIMARY KEY (deliver_id, project_id),
    CONSTRAINT fk_deliverable_project FOREIGN KEY (project_id) REFERENCES
project (project_id) ON DELETE CASCADE ON UPDATE CASCADE
);

-- Create field table
CREATE TABLE field
(
  field_name VARCHAR(50) NOT NULL,
  last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  PRIMARY KEY (field_name)
);

-- Create is_about table
CREATE TABLE is_about
(
    project_id SMALLINT UNSIGNED NOT NULL,
    field_name VARCHAR(50) NOT NULL,
    last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    PRIMARY KEY (project_id, field_name),
    CONSTRAINT fk_is_about_field FOREIGN KEY (field_name) REFERENCES field
(field_name) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_is_about_project FOREIGN KEY (project_id) REFERENCES
project (project_id) ON DELETE CASCADE ON UPDATE CASCADE
);
```

DML Script

```
-- TRIGGERS FOR INSERT

DELIMITER &&

-- Constraints for birth date and recruitment date of a researcher using
trigger
CREATE TRIGGER trig_birth_recruitment_date_insert BEFORE INSERT ON
researcher FOR EACH ROW BEGIN
    CALL check_birth_recruitment_date(NEW.birth_date,
NEW.recruitment_date);
END&&

-- Constraint for entry date, review date of a project.
-- We don't have a constraint for due_date because the combination of
entry_date trigger with the check_duration constraint covers us.
CREATE TRIGGER trig_project_dates_insert BEFORE INSERT ON project FOR EACH
ROW BEGIN
    CALL check_entry_date(NEW.entry_date);
    CALL check_reviewer(NEW.review_date, NEW.reviewer_id, NEW.project_id,
NEW.organization_id);
END&&

-- When inserting a project we have to add supervisor as a working
researcher on the project
CREATE TRIGGER trig_add_supervisor_insert AFTER INSERT ON project FOR EACH
ROW BEGIN
    INSERT  INTO works_on(project_id, researcher_id, is_supervisor) VALUES
(NEW.project_id, NEW.supervisor_id, 1);
END&&

-- Constraint for delivary date using trigger
CREATE TRIGGER trig_deliver_date_insert BEFORE INSERT ON deliverable FOR
EACH ROW BEGIN
    CALL check_deliver_date(NEW.deliver_date, NEW.project_id);
END&&

-- Constraints for researcher-project works_on relarion using trigger
CREATE TRIGGER trig_works_on_insert BEFORE INSERT ON works_on FOR EACH ROW
BEGIN
    CALL check_works_on(NEW.project_id, NEW.researcher_id);
END&&

DELIMITER ;
```

```
-- TRIGGERS FOR UPDATE

DELIMITER &&

-- Constraints for birth date and recruitment date of a researcher using
trigger
CREATE TRIGGER trig_birth_recruitment_date_update BEFORE UPDATE ON
researcher FOR EACH ROW BEGIN
    CALL check_birth_recruitment_date(NEW.birth_date,
NEW.recruitment_date);
END&&

-- Constraint for entry date, review date of a project.
-- We don't have a constraint for due_date because the combination of
entry_date trigger with the check_duration constraint covers us.
CREATE TRIGGER trig_entry_date_update BEFORE UPDATE ON project FOR EACH
ROW BEGIN
    CALL check_entry_date(NEW.entry_date);
    CALL check_reviewer(NEW.review_date, NEW.reviewer_id, NEW.project_id,
NEW.organization_id);
END&&

-- Constraint for delivary date using trigger
CREATE TRIGGER trig_deliver_date_update BEFORE UPDATE ON deliverable FOR
EACH ROW BEGIN
    CALL check_deliver_date(NEW.deliver_date, NEW.project_id);
END&&

-- Constraints for researcher-project works_on relarion using trigger
CREATE TRIGGER trig_works_on_update BEFORE UPDATE ON works_on FOR EACH ROW
BEGIN
    CALL check_works_on(NEW.project_id, NEW.researcher_id);
END&&


-- When updating a project we have to update supervisor on the works_on
relation
CREATE TRIGGER trig_supervisor_update BEFORE UPDATE ON project FOR EACH
ROW BEGIN
    DELETE FROM works_on WHERE (project_id = NEW.project_id) AND
(researcher_id = NEW.supervisor_id);
    DELETE FROM works_on WHERE (project_id = NEW.project_id) AND
(researcher_id = OLD.supervisor_id);
```

```
    INSERT INTO works_on(project_id, researcher_id, is_supervisor) VALUES
(NEW.project_id, NEW.supervisor_id, 1);
END&&


DELIMITER ;

-- Procedures

DELIMITER &&

CREATE PROCEDURE check_birth_recruitment_date(bDate DATE, recDate DATE)
BEGIN
    IF NOT ((YEAR(bDate) >= 1940) AND (DATEDIFF(CURRENT_DATE, bDate) >=
365 * 18)) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalide researcher
birth date.';
    END IF;
    IF NOT ((recDate <= CURRENT_DATE()) AND (YEAR(recDate) >= 1950) AND
((DATEDIFF(recDate, bDate)) DIV 365) >= 18) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalide researcher
recruitment date.';
    END IF;
END&&

CREATE PROCEDURE check_entry_date(entry DATE) BEGIN
    IF NOT ((entry <= CURRENT_DATE()) AND (YEAR(entry) >= 1950)) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalide entry date of
project.';
    END IF;
END&&

CREATE PROCEDURE check_deliver_date(deliver DATE, projID SMALLINT
UNSIGNED) BEGIN
    IF NOT (deliver BETWEEN (SELECT DISTINCT entry_date FROM project WHERE
project.project_id = projID)
                            AND (SELECT DISTINCT due_date FROM project
WHERE project.project_id = projID)) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalide deliverable
date.';
    END IF;
END&&

CREATE PROCEDURE check_works_on(projID SMALLINT UNSIGNED, researID
SMALLINT UNSIGNED) BEGIN
```

```
    IF NOT ((SELECT DISTINCT organization_id FROM project WHERE
project.project_id = projID) =
            (SELECT DISTINCT organization_id FROM researcher WHERE
researcher.researcher_id = resear ID)) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalide works_on
relation. Researcher can only work on projects that are handled by the
organization he/she works for.';
    END IF;
    IF NOT ((SELECT DISTINCT entry_date FROM project WHERE
project.project_id = projID) >=
            (SELECT DISTINCT recruitment_date FROM researcher WHERE
researcher.researcher_id = resear ID)) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalide works_on
relation. Researcher has to be recruited by an organization before working
on a project.';
    END IF;
END&&

CREATE PROCEDURE check_reviewer(rev_date DATE, reviewerID SMALLINT
UNSIGNED, projID SMALLINT UNSIGNED, orgID SMALLINT UNSIGNED) BEGIN
    IF ((SELECT DISTINCT organization_id FROM researcher WHERE
researcher_id = reviewerID) = orgID) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalide works_on
relation. The reviewer of a project cannot work for the same organization
that handles the project.';
    END IF;
    IF NOT (rev_date >= (SELECT recruitment_date FROM researcher WHERE
researcher_id = reviewerID)) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalide reviewer.
Reviewer has to be old enough to review the project.';
    END IF;

END&&
```

Queries

```
-- 3.1) Find projects and researchers on these projects where executive's
id is ...
CREATE PROCEDURE projects_based_on_executive (exec_id SMALLINT UNSIGNED)
BEGIN
SELECT project.project_id, project.title, researcher.researcher_id,
researcher.first_name, researcher.last_name
FROM researcher
INNER JOIN works_on ON researcher.researcher_id = works_on.researcher_id
INNER JOIN project ON project.project_id = works_on.project_id
INNER JOIN executive ON executive.executive_id = project.executive_id
WHERE executive.executive_id = exec_id;
END&&

-- 3.1) Find projects and researchers on these projects where duration is
...
CREATE PROCEDURE projects_based_on_duration (duration_var INT) BEGIN
SELECT project.project_id, project.title, researcher.first_name,
researcher.last_name
FROM researcher
INNER JOIN works_on ON researcher.researcher_id = works_on.researcher_id
INNER JOIN project ON project.project_id = works_on.project_id
WHERE project.duration = duration_var;
END&&

-- 3.1) Find active projects and researchers on these projects on date ...
CREATE PROCEDURE projects_based_on_date (date_var DATE) BEGIN
SELECT project.project_id, project.title, researcher.researcher_id,
researcher.first_name, researcher.last_name
FROM researcher
INNER JOIN works_on ON researcher.researcher_id = works_on.researcher_id
INNER JOIN project ON project.project_id = works_on.project_id
WHERE project.entry_date <= date_var AND project.due_date >= date_var;
END&&

-- 3.3) Find all projects and their researchers that are active and are
about a particular field
CREATE PROCEDURE projects_based_on_field (field_var VARCHAR(50)) BEGIN
SELECT project.project_id, project.title, researcher.researcher_id,
researcher.first_name, researcher.last_name, project.entry_date,
project.due_date, project.active
FROM project
INNER JOIN is_about ON is_about.project_id = project.project_id
INNER JOIN field ON field.field_name = is_about.field_name
```

```
INNER JOIN works_on ON project.project_id = works_on.project_id
INNER JOIN researcher ON researcher.researcher_id = works_on.researcher_id
WHERE (field.field_name = field_var AND project.active = 1);
END&&

DELIMITER ;

-- 3.2) Find all projects that researchers work on
CREATE VIEW researcher_project_view AS
SELECT researcher.researcher_id, researcher.first_name,
researcher.last_name, project.project_id, project.title
FROM researcher
INNER JOIN works_on ON researcher.researcher_id = works_on.researcher_id
INNER JOIN project ON project.project_id = works_on.project_id
ORDER BY researcher_id;

-- 3.2) Find all projects that organizations handle
CREATE VIEW organization_project_view AS
SELECT project.project_id, project.title, organization.organization_id,
organization.abbreviation, organization.name
FROM organization
INNER JOIN project ON project.organization_id =
organization.organization_id;

-- 3.4) Find which organizations have handled the same number of projects
for two years sequentially and those projects are at least 10 per year
CREATE VIEW query4_helper AS
SELECT count(*) AS num, organization_id, YEAR(entry_date) AS year
FROM project
GROUP BY YEAR(entry_date), organization_id
ORDER BY organization_id;

CREATE VIEW org_with_same_num_of_proj AS
SELECT helper1.num AS num1, helper1.organization_id, organization.name,
helper1.year AS first_year, helper2.year AS last_year
FROM query4_helper helper1
INNER JOIN query4_helper helper2 ON helper1.organization_id =
helper2.organization_id AND helper1.year = helper2.year+1
INNER JOIN organization ON organization.organization_id =
helper1.organization_id
WHERE helper1.num = helper2.num AND helper1.num >= 10;

-- 3.5) Find top-3 tuples of fields met in projects
CREATE VIEW top_3_tuples_of_fields AS
SELECT count(*) AS num, f1.field_name AS field_1, f2.field_name AS field_2
```

```sql
FROM is_about f1
INNER JOIN is_about f2 ON (f1.project_id = f2.project_id AND f1.field_name
!= f2.field_name)
WHERE f1.field_name > f2.field_name
GROUP BY f1.field_name, f2.field_name
ORDER BY num DESC LIMIT 3;

-- 3.6) Find all young researchers (age < 40) who work on the most active
projects and the number of the projects they work on
CREATE VIEW young_researchers AS
SELECT count(*) AS num, researcher.researcher_id, researcher.first_name,
researcher.last_name
FROM researcher
INNER JOIN works_on ON researcher.researcher_id = works_on.researcher_id
INNER JOIN project ON works_on.project_id = project.project_id
WHERE (datediff(CURRENT_DATE(), researcher.birth_date) < 40 * 365 AND
project.active = 1)
GROUP BY researcher_id
ORDER BY num DESC, researcher_id ASC;

-- 3.7) Find top-5 executives who have given the greatest amount to a
company
CREATE VIEW top_5_executives AS
SELECT executive.executive_id, executive.first_name, executive.last_name,
organization.name, project.amount
FROM executive
INNER JOIN project ON executive.executive_id = project.executive_id
INNER JOIN organization ON organization.organization_id =
project.organization_id
WHERE organization.organization_type = 'company'
ORDER BY amount DESC LIMIT 5;

-- 3.8) Find researchers who work on 5 or more projects that have no
deliverables
CREATE VIEW research_without_deliverables AS
SELECT count(*) AS num, researcher.researcher_id, researcher.first_name,
researcher.last_name
FROM researcher
INNER JOIN works_on ON works_on.researcher_id = researcher.researcher_id
INNER JOIN project ON project.project_id = works_on.project_id
WHERE project.project_id NOT IN
    (SELECT p1.project_id
     FROM project p1
     INNER JOIN deliverable ON deliverable.project_id = p1.project_id
    )
```

```
GROUP BY researcher.researcher_id HAVING num >= 5
ORDER BY num DESC, researcher_id ASC;
```

3) Για την εγκατάσταση της εφαρμογής σε συστήματα Windows 10/11 απαιτούνται τα παρακάτω βήματα:

- Εγκατάσταση του XAMPP
- Εγκατάσταση του MySQL Workbench
- Εγκατάσταση της NodeJS
- Εγκατάσταση των εξής Dependencies:
    - express
    - mysql2
    - ejs
    - express-session
    - connect-flash
    - nodemon
    - chalk
    - custom-env
    - html-js-confirm
- Τρέχουμε ως administrators το XAMPP και κάνουμε start την MySQL
- Ανοίγουμε το Workbench και προσθέτουμε ένα νέο connection μέσα στο οποίο ανοίγουμε τα αρχεία db-hfri-schema και db-hfri-insert που βρίσκονται μέσα στο folder MySQL
- Ελέγχουμε ότι τα host και user που βρίσκονται στο αρχείο database.js στο folder utils ταυτίζονται με αυτά που τρέχει η βάση μας, ελέγχοντάς το από το connection του Workbench. Σε αντίθετη περίπτωση τα τροποποιούμε κατάλληλα
- Αρχικά τρέχουμε το schema αρχείο το οποίο περιέχει και όλα τα απαραίτητα constraints, triggers, views και indexes και στη συνέχεια τρέχουμε το insert αρχείο (προσοχή να ΜΗΝ βρισκόμαστε σε safe updates mode στο Workbench, Edit -> Preferences -> SQL Editor)
- Ανοίγουμε το command window και μεταφερόμαστε στο path του folder HFRI-DB-Project-NTUA που κατεβάσαμε από το GitHub
- Εκτελούμε την εντολή npm start
- Ανοίγουμε έναν browser, δοκιμάστηκε σε Google Chrome και Microsoft Edge, και μεταφερόμαστε στη διεύθυνση localhost:3000

https://github.com/pGiad/HFRI-DB-Project-NTUA.git

Για οποιοδήποτε πρόβλημα σχετικά με την εγκατάσταση της εφαρμογής:

panosgiadi@gmail.com michael.velalopoulos@gmail.com