Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Βάσεις Δεδομένων

Εξάμηνο 6ο

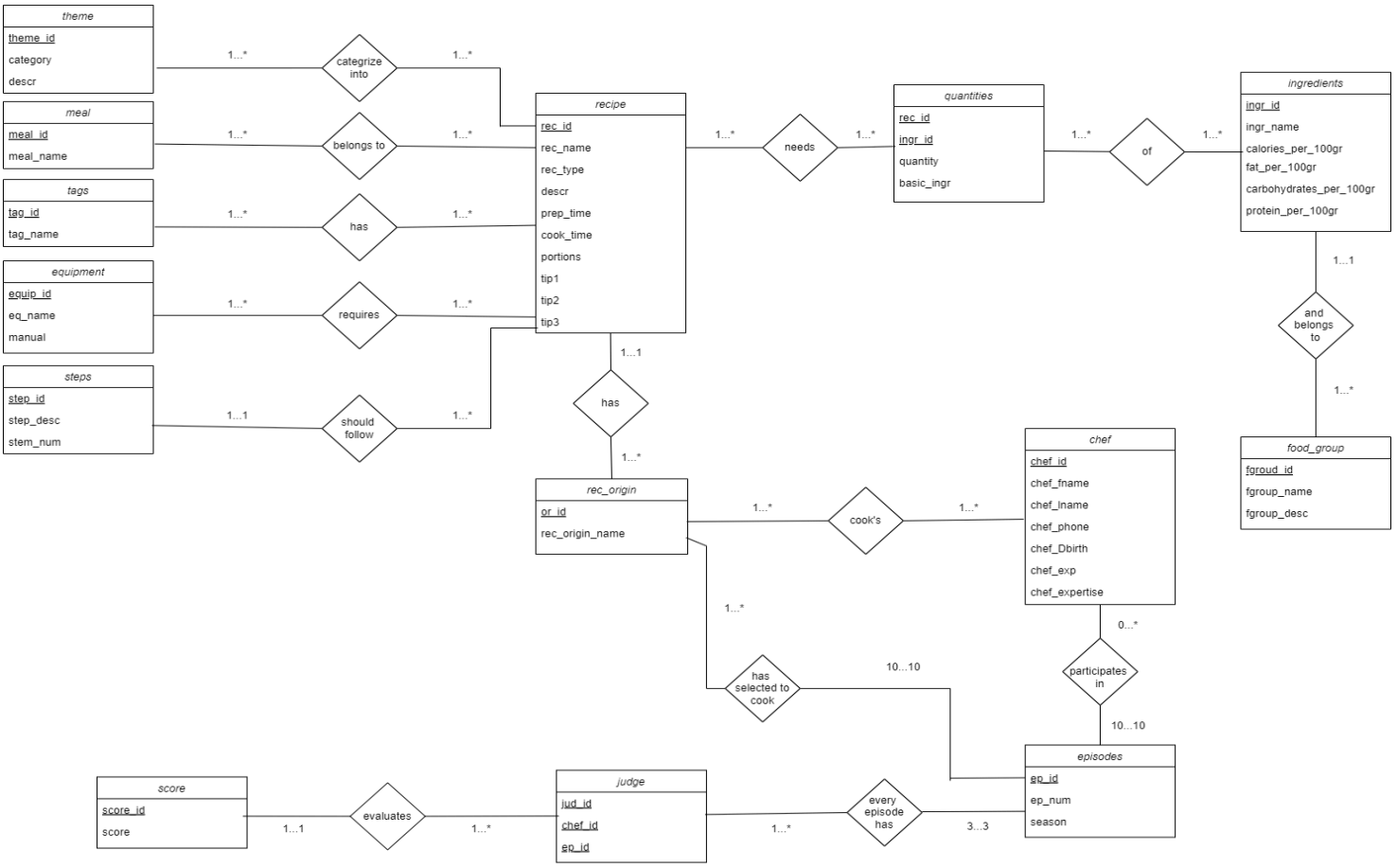
Ονόματα φοιτητών:
Ζακέο Μαρτίνα, 03121824
Γούσιος Ιωάννης, 03121065
Λαγουδάκης Εμμανουήλ, 03121219

Περιεχόμενα

1 Βάση Δεδομένων

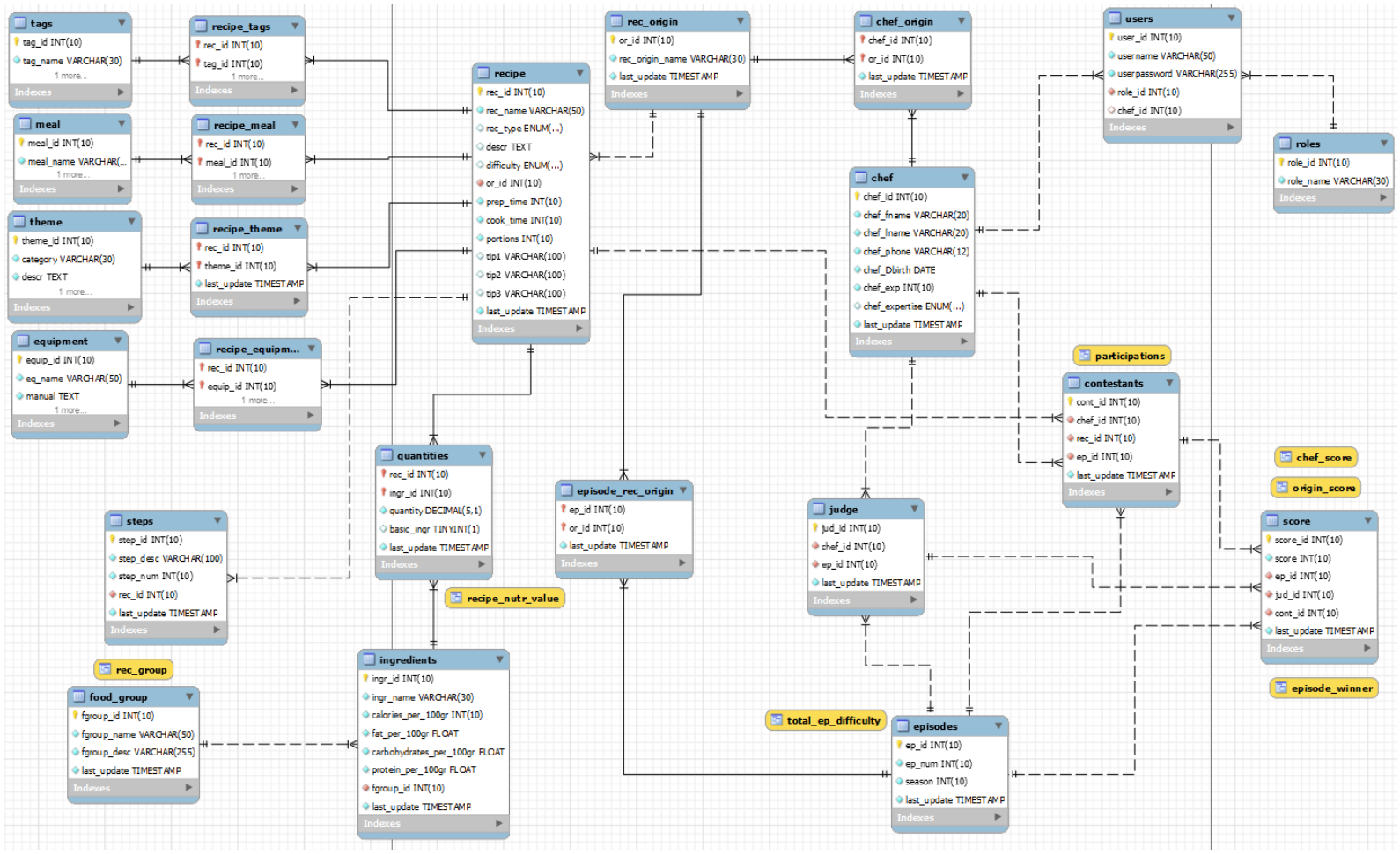
1.1 Διάγραμμα Οντοτήτων- Συσχετίσεων (Entity-Relationship Diagram)

Το ER - Diagram της βάσης δεδομένων που δημιουργήθηκε παρουσιάζεται στην παρακάτω εικόνα :



1.2 Σχεσιακό σχήμα (Relational Schema)

Χρησιμοποιώντας τα εργαλεία του '' **MySql Workbench** '', στο οποίο εργαστήκαμε για την κατασκευή της βάσης δεδομένων, προκύπτει το ακόλουθο σχεσιακό σχήμα για την βάση δεδομένων :



1.3 Ευρετήρια

Για τα attributes των πινάκων, που χρησιμοποιούνται συχνότερα δημιουργήσαμε και τα αντίστοιχα ευρετήρια (indexes). Συγκεκριμένα, επισημαίνεται πως η σχεδίαση των indexes έχει γίνει προσεγγιστικά και έχει ως στόχο την βελτιστοποίηση της απόδοσης. Σημειώνεται ότι η δημιουργία indexes στα primary keys δεν είναι απαραίτητη, καθώς δημιουργούνται αυτόματα ευρετήρια τύπου B+ tree. Αναλυτικότερα, τα indexes που δημιουργήθηκαν είναι τα εξής :

- idx_rec_or_id του table "recipe" (φιλτράρισμα συνταγών με βάση το or_id)
- idx_rec_name του table "recipe" (εύρεση ονόματος συνταγής με βάση το rec_name)
- idx_rec_portions του table "recipe" (εύρεση μερίδων συνταγής με βάση το portions)

- idx_theme του table “theme” (εύρεση θεματικής ενότητας με βάση το category)
- idx_rec_tags_rec_id του table “recipe_tags” (εύρεση ετικέτας με βάση το rec_id)
- idx_ingr_carbs του table “ingredients” (εύρεση υδατανθράκων με βάση του carbohydrates_per_100gr)
- idx_ingr_fgroup του table “ingredients” (εύρεση ομάδας τροφίμων του υλικού με βάση το fgroup_id)
- idx_quant_rec του table “quantities” (εύρεση συνταγής με βάση το rec_id)
- idx_quant_ingr του table “quantities” (εύρεση υλικού στην συνταγή με βάση το ingr_id)
- idx_step_rec του table “steps” (εύρεση βημάτων της συνταγής με βάση το rec_id)
- idx_chef_name του table “chef” (εύρεση του ονοματεπώνυμου του chef με βάση το chef_fname και του chef_lname)
- idx_chef_exp του table “chef” (εύρεση της εξειδίκευσης του chef με βάση το chef_exp)
- idx_ep_seas του table “episodes” (εύρεση του συνδυασμού επεισόδιο, σεζόν με βάση το ep_num, season)
- idx_cont_chef του table “contestants” (εύρεση του chef που συμμετείχε σε επεισόδιο με βάση το chef_id)
- idx_cont_rec του table “contestants” (εύρεση του συμμετέχοντα που μαγείρεψε την εκάστοτε συνταγή με βάση το rec_id)
- idx_cont_ep του table “contestants” (εύρεση των συμμετεχόντων σε ένα επεισόδιο με βάση το ep_id)
- idx_jud_chef του table “judge” (εύρεση του κριτή με βάση το chef_id)
- idx_jud_ep του table “judge” (εύρεση του κριτή με βάση το επεισόδιο που συμμετείχε βάσει του ep_id)
- idx_score_ep του table “score” (εύρεση βαθμολογίας - 30 αποτελέσματα- για με βάση το ep_id)
- idx_score_cont του table “score” (εύρεση της βαθμολογίας του κάθε συμμετέχοντα με βάση το cont_id)

2 Βιβλιοθήκες που χρησιμοποιήθηκαν

Η **MySQL**, είναι η βιβλιοθήκη που χρησιμοποιήθηκε για την υλοποίηση της βάσης δεδομένων.

3 DDL Script

Το αρχείο DDL το οποίο δημιουργεί τα tables και τα indexes της βάσης δεδομένων παρουσιάζεται παρακάτω :

```
/*
*
*************************** DDL ***************************
*
*/

DROP SCHEMA IF EXISTS mastercook;
CREATE SCHEMA mastercook;
USE mastercook;


/*
*
*
*****************************************   TABLES
************************************************************************
*
*
*/
CREATE TABLE rec_origin (
or_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
rec_origin_name VARCHAR(30) NOT NULL UNIQUE,
last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY(or_id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE recipe (
rec_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
rec_name VARCHAR(50) NOT NULL UNIQUE,
rec_type ENUM('Cooking', 'Pastry'),
descr TEXT DEFAULT NULL,
difficulty ENUM('1','2','3','4','5'),
or_id INT UNSIGNED NOT NULL,
prep_time INT UNSIGNED NOT NULL , -- preparation time in minutes for each recipe
cook_time INT UNSIGNED NOT NULL,  -- cooking time in minutes for each recipe
portions INT UNSIGNED NOT NULL,
tip1 VARCHAR(100),
tip2 VARCHAR(100),
tip3 VARCHAR(100),
last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY (rec_id),
CONSTRAINT fk_recipe_rec_origin FOREIGN KEY (or_id) REFERENCES rec_origin (or_id) ON DELETE
RESTRICT ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE equipment (
equip_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
eq_name VARCHAR(50) NOT NULL UNIQUE,
manual TEXT NOT NULL,
last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY (equip_id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE recipe_equipment (
rec_id INT UNSIGNED NOT NULL,
equip_id INT UNSIGNED NOT NULL,
last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY (rec_id,equip_id),
CONSTRAINT fk_recipe_equipment_recipe FOREIGN KEY (rec_id) REFERENCES recipe (rec_id) ON
DELETE RESTRICT ON UPDATE CASCADE,
CONSTRAINT fk_recipe_equipment_equipment FOREIGN KEY (equip_id) REFERENCES equipment
(equip_id) ON DELETE RESTRICT ON UPDATE CASCADE
```

```
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE theme (
theme_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
category VARCHAR(30) NOT NULL UNIQUE,
descr TEXT NOT NULL,
last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY (theme_id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE recipe_theme (
rec_id INT UNSIGNED NOT NULL,
theme_id INT UNSIGNED NOT NULL,
last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY(rec_id,theme_id),
CONSTRAINT fk_recipe_theme_recipe FOREIGN KEY (rec_id) REFERENCES recipe (rec_id) ON DELETE
RESTRICT ON UPDATE CASCADE,
CONSTRAINT fk_recipe_theme_theme FOREIGN KEY (theme_id) REFERENCES theme (theme_id) ON
DELETE RESTRICT ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE meal (
meal_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
meal_name VARCHAR(30) NOT NULL UNIQUE,
last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY (meal_id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE recipe_meal (
rec_id INT UNSIGNED NOT NULL,
meal_id INT UNSIGNED NOT NULL,
last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY (rec_id,meal_id),
CONSTRAINT fk_recipe_meal_recipe FOREIGN KEY (rec_id) REFERENCES recipe (rec_id) ON DELETE
RESTRICT ON UPDATE CASCADE,
CONSTRAINT fk_recipe_meal_meal FOREIGN KEY (meal_id) REFERENCES meal (meal_id) ON DELETE
RESTRICT ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE tags (
tag_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
tag_name VARCHAR(30) NOT NULL UNIQUE,
last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY (tag_id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE recipe_tags (
rec_id INT UNSIGNED NOT NULL,
tag_id INT UNSIGNED NOT NULL,
last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY(rec_id,tag_id),
CONSTRAINT fk_recipe_tags_recipe FOREIGN KEY (rec_id) REFERENCES recipe (rec_id) ON DELETE
RESTRICT ON UPDATE CASCADE,
CONSTRAINT fk_recipe_tags_tags FOREIGN KEY (tag_id) REFERENCES tags (tag_id) ON DELETE
RESTRICT ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE food_group (
fgroup_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
fgroup_name VARCHAR(50) NOT NULL UNIQUE CHECK(fgroup_name IN ('Vegetables', 'Fruits', 'Cereal and
Potatoes', 'Milk and Dairy Products', 'Legumes', 'Red Meat', 'White Meat', 'Eggs', 'Fish and Seafood', 'Fats, Oil,
Olives and Nuts')),
fgroup_desc VARCHAR(255) NOT NULL,
last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY(fgroup_id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE ingredients (
ingr_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
ingr_name VARCHAR(30) NOT NULL UNIQUE,
calories_per_100gr INT UNSIGNED NOT NULL ,
fat_per_100gr FLOAT NOT NULL ,
carbohydrates_per_100gr FLOAT NOT NULL ,
protein_per_100gr FLOAT NOT NULL ,
fgroup_id INT UNSIGNED NOT NULL,
last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY (ingr_id),
CONSTRAINT fk_ingredients_food_group FOREIGN KEY (fgroup_id) REFERENCES food_group (fgroup_id)
ON DELETE RESTRICT ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE quantities (
rec_id INT UNSIGNED NOT NULL,
ingr_id INT UNSIGNED NOT NULL,
quantity DECIMAL(5,1) NOT NULL,
basic_ingr BOOL DEFAULT FALSE,   -- Only one basic ingr
last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY (rec_id, ingr_id),
CONSTRAINT fk_quantities_recipe FOREIGN KEY (rec_id) REFERENCES recipe (rec_id) ON DELETE
RESTRICT ON UPDATE CASCADE,
CONSTRAINT fk_quantities_ingredients FOREIGN KEY (ingr_id) REFERENCES ingredients (ingr_id) ON
DELETE RESTRICT ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE steps (
step_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
step_desc VARCHAR(100) NOT NULL,
step_num INT UNSIGNED NOT NULL,
rec_id INT UNSIGNED NOT NULL,
last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY(step_id),
CONSTRAINT fk_steps_recipe FOREIGN KEY (rec_id) REFERENCES recipe (rec_id) ON DELETE RESTRICT
ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE chef (
chef_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
chef_fname VARCHAR(20) NOT NULL,
chef_lname VARCHAR(20) NOT NULL,
chef_phone VARCHAR(12) NOT NULL UNIQUE,
chef_Dbirth DATE NOT NULL,
chef_exp INT UNSIGNED NOT NULL DEFAULT 0,
chef_expertise ENUM('C Chef', 'B Chef', 'A Chef', 'Assistant Head Chef', 'Head Chef') DEFAULT 'C Chef',
last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY(chef_id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE chef_origin (
chef_id INT UNSIGNED NOT NULL,
or_id INT UNSIGNED NOT NULL,
CONSTRAINT fk_chef_origin_chef FOREIGN KEY (chef_id) REFERENCES chef (chef_id) ON DELETE
RESTRICT ON UPDATE CASCADE,
CONSTRAINT fk_chef_origin_rec_origin FOREIGN KEY (or_id) REFERENCES rec_origin (or_id) ON
DELETE RESTRICT ON UPDATE CASCADE,
last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY (chef_id, or_id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE episodes (
ep_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
ep_num INT UNSIGNED NOT NULL,
season INT UNSIGNED NOT NULL,
```

```
last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY (ep_id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE episode_rec_origin (
ep_id INT UNSIGNED NOT NULL,
or_id INT UNSIGNED NOT NULL,
last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY (ep_id, or_id),
CONSTRAINT fk_episode_rec_origin_episode FOREIGN KEY (ep_id) REFERENCES episodes (ep_id) ON
DELETE RESTRICT ON UPDATE CASCADE,
CONSTRAINT fk_episode_rec_origin_rec_origin FOREIGN KEY (or_id) REFERENCES rec_origin (or_id)
ON DELETE RESTRICT ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE contestants (
cont_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
chef_id INT UNSIGNED NOT NULL,
rec_id INT UNSIGNED NOT NULL,
ep_id INT UNSIGNED NOT NULL,
last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY(cont_id),
CONSTRAINT fk_contestants_chef FOREIGN KEY (chef_id) REFERENCES chef (chef_id) ON DELETE
RESTRICT ON UPDATE CASCADE,
CONSTRAINT fk_contestants_recipe FOREIGN KEY (rec_id) REFERENCES recipe (rec_id) ON DELETE
RESTRICT ON UPDATE CASCADE,
CONSTRAINT fk_contestants_episodes FOREIGN KEY (ep_id) REFERENCES episodes (ep_id) ON DELETE
RESTRICT ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE judge (
jud_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
chef_id INT UNSIGNED NOT NULL,
ep_id INT UNSIGNED NOT NULL,
last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY (jud_id),
CONSTRAINT fk_judge_chef FOREIGN KEY (chef_id) REFERENCES chef (chef_id) ON DELETE RESTRICT
ON UPDATE CASCADE,
CONSTRAINT fk_judge_episodes FOREIGN KEY (ep_id) REFERENCES episodes (ep_id)  ON DELETE
RESTRICT ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE score (
score_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
score INT UNSIGNED NOT NULL,
ep_id INT UNSIGNED NOT NULL,
jud_id INT UNSIGNED NOT NULL,
cont_id INT UNSIGNED NOT NULL,
last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY (score_id),
CONSTRAINT fk_score_episode FOREIGN KEY (ep_id) REFERENCES episodes (ep_id) ON DELETE
RESTRICT ON UPDATE CASCADE,
CONSTRAINT fk_score_judge FOREIGN KEY (jud_id) REFERENCES judge (jud_id) ON DELETE RESTRICT
ON UPDATE CASCADE,
CONSTRAINT fk_score_contestants FOREIGN KEY (cont_id) REFERENCES contestants (cont_id) ON
DELETE RESTRICT ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8;


/*
*
*
*********************************************  INDEXES
*****************************************************************************
*
*
```

```
*/


CREATE INDEX idx_rec_or_id ON recipe (or_id);
CREATE INDEX idx_rec_name ON recipe (rec_name);
CREATE INDEX idx_rec_portions ON recipe (portions);
CREATE INDEX idx_theme ON theme (category);
CREATE INDEX idx_rec_tags_rec_id ON recipe_tags(rec_id);
CREATE INDEX idx_ingr_carbs ON ingredients (carbohydrates_per_100gr);
CREATE INDEX idx_ingr_fgroup ON ingredients (fgroup_id);
CREATE INDEX idx_quant_rec ON quantities (rec_id);
CREATE INDEX idx_quant_ingr ON quantities (ingr_id);
CREATE INDEX idx_step_rec ON steps (rec_id);
CREATE INDEX idx_chef_name ON chef (chef_fname, chef_lname);
CREATE INDEX idx_chef_exp ON chef (chef_expertise);
CREATE INDEX idx_ep_seas ON episodes (ep_num, season);
CREATE INDEX idx_cont_chef ON contestants (chef_id);
CREATE INDEX idx_cont_rec ON contestants (rec_id);
CREATE INDEX idx_cont_ep ON contestants (ep_id);
CREATE INDEX idx_jud_chef ON judge (chef_id);
CREATE INDEX idx_jud_ep ON judge (ep_id);
CREATE INDEX idx_score_ep ON score (ep_id);
CREATE INDEX idx_score_cont  ON score (cont_id);

/*
*
*
********************************************   VIEWS
**************************************************************************
*
*
*/

-- View for nutritional value

CREATE VIEW recipe_nutr_value
AS
SELECT r.rec_name AS Recipe, ROUND(SUM(i.fat_per_100gr * (q.quantity /100))/r.portions,1) AS
Fat_per_portion_gr,  ROUND(SUM(i.protein_per_100gr * (q.quantity /100))/r.portions,1) AS
Protein_per_portion_gr,
                    ROUND(SUM(i.carbohydrates_per_100gr * (q.quantity /100))/r.portions,1) AS
Carbohydrates_per_portion_gr, ROUND(SUM(i.calories_per_100gr * (q.quantity /100))/r.portions,1) AS
Calories_per_portion_kcal
FROM recipe AS r INNER JOIN quantities AS q ON r.rec_id = q.rec_id
INNER JOIN ingredients AS i ON q.ingr_id = i.ingr_id
GROUP BY r.rec_id;

-- View for chef id -> score

CREATE VIEW chef_score
AS
SELECT c.chef_id AS id, CONCAT(c.chef_fname, " ", c.chef_lname) AS chef_name, e.season, e.ep_num AS
episode,  SUM(s.score) AS total_score
FROM chef AS c INNER JOIN contestants AS con ON c.chef_id = con.chef_id INNER JOIN score AS s on
s.cont_id = con.cont_id INNER JOIN episodes AS e on e.ep_id = con.ep_id
GROUP BY con.cont_id;

-- View for rec_origin -> score
CREATE VIEW origin_score
AS
SELECT ro.or_id AS id, e.season, e.ep_num AS episode,  SUM(s.score) AS total_score
FROM rec_origin as ro INNER JOIN recipe AS r ON ro.or_id = r.or_id INNER JOIN contestants AS con ON
r.rec_id = con.rec_id INNER JOIN score AS s on s.cont_id = con.cont_id INNER JOIN episodes AS e on
e.ep_id = con.ep_id
GROUP BY con.cont_id;
```

```
-- View for winner on each episode
CREATE VIEW episode_winner AS
SELECT chef_name, season, episode, total_score, chef_expertise
FROM (
   SELECT chef_name, season, episode, total_score, chef_expertise,
        ROW_NUMBER() OVER (PARTITION BY season, episode ORDER BY total_score DESC,
FIELD(chef_expertise, 'C Chef', 'B Chef', 'A Chef', 'Assistant Head Chef', 'Head Chef') DESC) AS chef_rank
   FROM chef_score
   INNER JOIN chef ON chef_score.id = chef.chef_id
) AS ranked_chefs
WHERE chef_rank = 1;

-- View for number of participations for each chef
CREATE VIEW participations AS
SELECT CONCAT(chef_fname, " ", chef_lname) AS chef_name, COUNT(cont.cont_id) AS
number_of_participations
FROM chef AS c INNER JOIN contestants AS cont ON c.chef_id = cont.chef_id
GROUP BY c.chef_id;

-- View for the total difficulty of each episode

CREATE VIEW total_ep_difficulty AS
SELECT SUM(r.difficulty) AS total_ep_diff, e.ep_id, e.season
FROM episodes AS e INNER JOIN contestants AS cont ON cont.ep_id = e.ep_id INNER JOIN recipe AS r ON
r.rec_id = cont.rec_id
GROUP BY e.ep_id;

-- View for the food group in which a recipe belongs based on its basic ingredient

CREATE VIEW rec_group AS
SELECT r.rec_name,  fg.fgroup_name
FROM recipe r INNER JOIN quantities q ON r.rec_id = q.rec_id INNER JOIN ingredients i on i.ingr_id =
q.ingr_id INNER JOIN food_group fg ON fg.fgroup_id = i.fgroup_id
WHERE q.basic_ingr = TRUE;

-- View for each chefs Recipes

CREATE VIEW chef_recipes AS
SELECT c.chef_id AS Chef_id, CONCAT(c.chef_fname, " ", c.chef_lname) AS Chef, r.rec_name AS Recipe,
r.rec_id AS Recipe_id
FROM contestants cont INNER JOIN chef c ON cont.chef_id = c.chef_id INNER JOIN recipe r ON r.rec_id =
cont.rec_id

/*
*
*
********************************************  INITIAL PROCEDURES (FIRST FIVE SEASONS)
*************************************************************************
*
*
*/


DELIMITER //

CREATE PROCEDURE InsertRecOriginInit()  -- Procedure to select 10 Receipe Origins for each episode
BEGIN
   DECLARE x INT;
   DECLARE i INT DEFAULT 1;
   DECLARE j INT DEFAULT 0;
            WHILE i<51 DO
   SET j = 0;
      WHILE j<10 DO
                                    SELECT or_id INTO x
            FROM rec_origin
            WHERE or_id NOT IN (SELECT ero.or_id
```

```
                                                                    FROM episode_rec_origin
ero
                        WHERE ero.ep_id = i )

                                                AND or_id NOT IN          (          SELECT ero1.or_id FROM
episode_rec_origin ero1
                                                                                                    JOIN
episode_rec_origin ero2 ON ero2.ep_id = i - 2 AND ero1.or_id = ero2.or_id
                                                                                                    JOIN
episode_rec_origin ero3 ON ero3.ep_id = i - 3 AND ero1.or_id = ero3.or_id
                                WHERE ero1.ep_id = i - 1)


            ORDER BY RAND()
            LIMIT 1;
            INSERT INTO episode_rec_origin(ep_id, or_id)
            VALUES (i,x);
                                                SET j = j + 1;
                        END WHILE;
                        SET i = i + 1;
             END WHILE;
END//

CREATE PROCEDURE InsertEp_Chef_RecInit()                    -- Procedure to select 10 chefs for episode
BEGIN
   DECLARE x INT;
   DECLARE y INT;
   DECLARE z INT;
   DECLARE i INT DEFAULT 1;
   DECLARE j INT DEFAULT 0;
            WHILE i<51 DO
   SET j = 0;
      WHILE j<10 DO
                                            SELECT ero.or_id INTO x
                                            FROM episode_rec_origin ero
            WHERE ero.ep_id = i AND ero.or_id NOT IN (SELECT reci.or_id
                                                        FROM recipe reci JOIN contestants conte ON
conte.rec_id = reci.rec_id
                                                        WHERE conte.ep_id = i)
            LIMIT 1;

            SELECT co.chef_id INTO y
            FROM chef_origin co INNER JOIN episode_rec_origin ero ON co.or_id = ero.or_id
            WHERE co.or_id = x AND co.chef_id NOT IN (SELECT cont.chef_id
                                                                                                    FROM
contestants cont
                                                                                                    WHERE
cont.ep_id = i)
                                            AND       co.chef_id NOT IN  (          SELECT cont1.chef_id FROM
contestants cont1
                                                                                                    JOIN
contestants cont2 ON cont2.ep_id = i - 2 AND cont1.chef_id = cont2.chef_id
                                                                                                    JOIN
contestants cont3 ON cont3.ep_id = i - 3 AND cont2.chef_id = cont3.chef_id
                                WHERE cont1.ep_id = i - 1)
            ORDER BY RAND()
            LIMIT 1;

            SELECT r.rec_id INTO z
            FROM rec_origin ro INNER JOIN episode_rec_origin ero ON ro.or_id = ero.or_id INNER JOIN
recipe r on r.or_id = ro.or_id
            WHERE ro.or_id = x AND r.rec_id NOT IN (SELECT cont.rec_id
                                                                                        FROM contestants
cont
                                                                                        WHERE cont.ep_id =
i)
```

```
                                    AND r.rec_id NOT IN  (      SELECT cont1.rec_id FROM contestants cont1
                                                                                                    JOIN
contestants cont2 ON cont2.ep_id = i - 2 AND cont1.rec_id = cont2.rec_id
                                                                                                    JOIN
contestants cont3 ON cont3.ep_id = i - 3 AND cont2.rec_id = cont3.rec_id
                            WHERE cont1.ep_id = i - 1)
          ORDER BY RAND()
          LIMIT 1;

          INSERT INTO contestants(chef_id, rec_id, ep_id)
          VALUES (y, z, i);
                                    SET j = j + 1;
                    END WHILE;
                    SET i = i + 1;
           END WHILE;
END//

CREATE PROCEDURE InsertJudgeInit()  -- Procedure to select 10 Receipe Origins for each episode
BEGIN
   DECLARE x INT;
   DECLARE i INT DEFAULT 1;
   DECLARE j INT DEFAULT 0;
            WHILE i<51 DO
   SET j = 0;
      WHILE j<3 DO
                                SELECT chef_id INTO x
          FROM chef
          WHERE chef_id NOT IN (SELECT cont.chef_id
                                                        FROM contestants cont
                    WHERE cont.ep_id = i)
                                            AND chef_id NOT IN (SELECT j.chef_id
                                                        FROM judge j
                    WHERE j.ep_id = i)
                                            AND chef_id NOT IN (SELECT j1.chef_id FROM judge j1
                                                                                    JOIN judge
j2 ON j2.ep_id = i - 2 AND j1.chef_id = j2.chef_id
                                                                                    JOIN judge
j3 ON j3.ep_id = i - 3 AND j1.chef_id = j2.chef_id
                            WHERE j1.ep_id = i - 1)
          ORDER BY RAND()
          LIMIT 1;
          INSERT INTO judge(chef_id, ep_id)
          VALUES (x, i);
                                    SET j = j + 1;
                    END WHILE;
                    SET i = i + 1;
           END WHILE;
END//

CREATE PROCEDURE CreateScoreInit()  -- Procedure to select 10 Receipe Origins for each episode
BEGIN
   DECLARE x INT;
   DECLARE y INT;
   DECLARE i INT DEFAULT 1;
   DECLARE j INT DEFAULT 1;
   DECLARE k INT DEFAULT 1;
            WHILE i<51 DO
   SET j = 1;
      WHILE j<4 DO
      SET k=1;
                            WHILE k<11 DO
                                    SELECT FLOOR(RAND() * 5 + 1) INTO x;
          INSERT INTO score(score, ep_id, jud_id, cont_id)
          VALUES (x, i, (i-1)*3 +j, (i-1)*10 + k);
                                    SET k = k + 1;
                            END WHILE;
```

```
        SET j = j + 1;
                    END WHILE;
                    SET i = i + 1;
            END WHILE;
END//

DELIMITER ;

/*
*
*
********************************************   PROCEDURES TO ADD MORE EPISODES ETC
*************************************************************************
*
*
*/

DELIMITER //

CREATE PROCEDURE InsertNextEpisode()
BEGIN
   DECLARE next_ep_id INT;
   DECLARE next_season INT;
   DECLARE ep_in_season INT;

   -- Βρίσκουμε το επόμενο ep_id
   SELECT IFNULL(MAX(ep_id), 0) + 1 INTO next_ep_id FROM episodes;

   -- Υπολογίζουμε τη σεζόν του επόμενου επεισοδίου
   SET next_season = CEIL(next_ep_id / 10);

   -- Υπολογίζουμε τον αριθμό των επεισοδίων στη σεζόν
   SET ep_in_season = next_ep_id % 10;

   -- Αν το επόμενο επεισόδιο είναι το πρώτο της νέας σεζόν
   IF ep_in_season = 0 THEN
      SET ep_in_season = 10;
   END IF;

   -- Εισάγουμε το νέο επεισόδιο στον πίνακα episodes
   INSERT INTO episodes (ep_num, season) VALUES (ep_in_season, next_season);

   -- Εισαγωγή των 10 origins για το νέο επεισόδιο
   CALL InsertRecOrigin(next_ep_id);

   -- Εισαγωγή των 10 chefs και των αντίστοιχων recipes για το νέο επεισόδιο
   CALL InsertEp_Chef_Rec(next_ep_id);

   -- Εισαγωγή των 3 judges για το νέο επεισόδιο
   CALL InsertJudge(next_ep_id);

   -- Εισαγωγή των scores για το νέο επεισόδιο
   CALL CreateScore(next_ep_id);

END//

CREATE PROCEDURE InsertRecOrigin(IN ep_id INT)
BEGIN
   DECLARE x INT;
   DECLARE j INT DEFAULT 0;
   WHILE j < 10 DO
      SET x = (SELECT or_id
            FROM rec_origin
            WHERE or_id NOT IN (SELECT ero.or_id
                        FROM episode_rec_origin ero
                        WHERE ero.ep_id = ep_id)
```

```
        AND or_id NOT IN (SELECT ero1.or_id
                  FROM episode_rec_origin ero1
                  JOIN episode_rec_origin ero2 ON ero2.ep_id = ep_id - 2 AND ero1.or_id = ero2.or_id
                  JOIN episode_rec_origin ero3 ON ero3.ep_id = ep_id - 3 AND ero1.or_id = ero3.or_id
                  WHERE ero1.ep_id = ep_id - 1)
      ORDER BY RAND()
      LIMIT 1);
    INSERT INTO episode_rec_origin(ep_id, or_id)
    VALUES (ep_id, x);
    SET j = j + 1;
  END WHILE;
END//

CREATE PROCEDURE InsertEp_Chef_Rec(IN ep_id INT)
BEGIN
  DECLARE x INT;
  DECLARE y INT;
  DECLARE z INT;
  DECLARE j INT DEFAULT 0;
  WHILE j < 10 DO
    SELECT ero.or_id INTO x
    FROM episode_rec_origin ero
    WHERE ero.ep_id = ep_id AND ero.or_id NOT IN (SELECT reci.or_id
                                FROM recipe reci
                                JOIN contestants conte ON conte.rec_id = reci.rec_id
                                WHERE conte.ep_id = ep_id)
    LIMIT 1;

    SELECT co.chef_id INTO y
    FROM chef_origin co
    INNER JOIN episode_rec_origin ero ON co.or_id = ero.or_id
    WHERE co.or_id = x AND co.chef_id NOT IN (SELECT cont.chef_id
                               FROM contestants cont
                               WHERE cont.ep_id = ep_id)
     AND co.chef_id NOT IN (SELECT cont1.chef_id
                FROM contestants cont1
                JOIN contestants cont2 ON cont2.ep_id = ep_id - 2 AND cont1.chef_id = cont2.chef_id
                JOIN contestants cont3 ON cont3.ep_id = ep_id - 3 AND cont2.chef_id = cont3.chef_id
                WHERE cont1.ep_id = ep_id - 1)
    ORDER BY RAND()
    LIMIT 1;

    SELECT r.rec_id INTO z
    FROM rec_origin ro
    INNER JOIN episode_rec_origin ero ON ro.or_id = ero.or_id
    INNER JOIN recipe r on r.or_id = ro.or_id
    WHERE ro.or_id = x AND r.rec_id NOT IN (SELECT cont.rec_id
                               FROM contestants cont
                               WHERE cont.ep_id = ep_id)
     AND r.rec_id NOT IN (SELECT cont1.rec_id
                FROM contestants cont1
                JOIN contestants cont2 ON cont2.ep_id = ep_id - 2 AND cont1.rec_id = cont2.rec_id
                JOIN contestants cont3 ON cont3.ep_id = ep_id - 3 AND cont2.rec_id = cont3.rec_id
                WHERE cont1.ep_id = ep_id - 1)
    ORDER BY RAND()
    LIMIT 1;

    INSERT INTO contestants(chef_id, rec_id, ep_id)
    VALUES (y, z, ep_id);
    SET j = j + 1;
  END WHILE;
END//

CREATE PROCEDURE InsertJudge(IN ep_id INT)
BEGIN
  DECLARE x INT;
```

```
  DECLARE j INT DEFAULT 0;
  WHILE j < 3 DO
    SELECT chef_id INTO x
    FROM chef
    WHERE chef_id NOT IN (SELECT cont.chef_id
                FROM contestants cont
                WHERE cont.ep_id = ep_id)
     AND chef_id NOT IN (SELECT j.chef_id
                FROM judge j
                WHERE j.ep_id = ep_id)
     AND chef_id NOT IN (SELECT j1.chef_id
                FROM judge j1
                JOIN judge j2 ON j2.ep_id = ep_id - 2 AND j1.chef_id = j2.chef_id
                JOIN judge j3 ON j3.ep_id = ep_id - 3 AND j1.chef_id = j2.chef_id
                WHERE j1.ep_id = ep_id - 1)
    ORDER BY RAND()
    LIMIT 1;
    INSERT INTO judge(chef_id, ep_id)
    VALUES (x, ep_id);
    SET j = j + 1;
  END WHILE;
END//

CREATE PROCEDURE CreateScore(IN ep_id INT)
BEGIN
  DECLARE x INT;
  DECLARE j INT DEFAULT 1;
  DECLARE k INT DEFAULT 1;
  WHILE j < 4 DO
    SET k = 1;
    WHILE k < 11 DO
      SELECT FLOOR(RAND() * 5 + 1) INTO x;
      INSERT INTO score(score, ep_id, jud_id, cont_id)
      VALUES (x, ep_id, (ep_id-1)*3 + j, (ep_id-1)*10 + k);
      SET k = k + 1;
    END WHILE;
    SET j = j + 1;
  END WHILE;
END//

DELIMITER ;

/*
*
*
********************************************   USERS
*************************************************************************
*
*
*/

CREATE TABLE roles (
  role_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
  role_name VARCHAR(30) NOT NULL UNIQUE,
  PRIMARY KEY (role_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO roles (role_name) VALUES ('Admin'), ('Chef');

CREATE TABLE users (
  user_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
  username VARCHAR(50) NOT NULL UNIQUE,
  userpassword VARCHAR(255) NOT NULL,
  role_id INT UNSIGNED NOT NULL,
  chef_id INT UNSIGNED,
  PRIMARY KEY (user_id),
```

```
   FOREIGN KEY (role_id) REFERENCES roles (role_id) ON DELETE RESTRICT ON UPDATE CASCADE,
   FOREIGN KEY (chef_id) REFERENCES chef (chef_id) ON DELETE SET NULL ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

/*                                                  Privileges for admin role
                                                */

GRANT ALL PRIVILEGES ON mastercook.* TO 'Admin'@'%';
/*                                                  Privileges for chef role
                                                */

DELIMITER //
CREATE PROCEDURE UpdateRecipe(
 IN p_user_id INT,
 IN p_rec_id INT,
 IN p_rec_name VARCHAR(50),
 IN p_rec_type ENUM('Cooking', 'Pastry'),
 IN p_descr TEXT,
 IN p_difficulty ENUM('1','2','3','4','5'),
 IN p_or_id INT,
 IN p_prep_time INT,
 IN p_cook_time INT,
 IN p_portions INT,
 IN p_tip1 VARCHAR(100),
 IN p_tip2 VARCHAR(100),
 IN p_tip3 VARCHAR(100)
)
BEGIN
 UPDATE recipe
 SET rec_name = p_rec_name,
    rec_type = p_rec_type,
    descr = p_descr,
    difficulty = p_difficulty,
    or_id = p_or_id,
    prep_time = p_prep_time,
    cook_time = p_cook_time,
    portions = p_portions,
    tip1 = p_tip1,
    tip2 = p_tip2,
    tip3 = p_tip3
 WHERE rec_id = p_rec_id
 AND rec_id IN (SELECT rec_id FROM chef_recipes WHERE chef_id = p_user_id);
END//
CREATE PROCEDURE UpdateChefInfo(
 IN p_chef_id INT,
 IN p_chef_fname VARCHAR(20),
 IN p_chef_lname VARCHAR(20),
 IN p_chef_phone VARCHAR(12),
 IN p_chef_Dbirth DATE,
 IN p_chef_exp INT,
 IN p_chef_expertise ENUM('C Chef', 'B Chef', 'A Chef', 'Assistant Head Chef', 'Head Chef')
)
BEGIN
 UPDATE chef
 SET chef_fname = p_chef_fname,
    chef_lname = p_chef_lname,
    chef_phone = p_chef_phone,
    chef_Dbirth = p_chef_Dbirth,
    chef_exp = p_chef_exp,
    chef_expertise = p_chef_expertise
 WHERE chef_id = p_chef_id;
END//
DELIMITER ;

GRANT EXECUTE ON PROCEDURE UpdateRecipe TO 'Chef'@'%';
GRANT EXECUTE ON PROCEDURE UpdateChefInfo TO 'Chef'@'%';
```

```
GRANT SELECT ON TABLE ingredients TO 'Chef'@'%';
GRANT SELECT ON TABLE quantities TO 'Chef'@'%';
GRANT SELECT ON TABLE steps TO 'Chef'@'%';
GRANT SELECT ON TABLE recipe TO 'Chef'@'%';
GRANT SELECT ON TABLE episodes TO 'Chef'@'%';
GRANT SELECT ON TABLE contestants TO 'Chef'@'%';
GRANT SELECT ON TABLE score TO 'Chef'@'%';
-- GRANT SELECT ON chef_recipes TO 'Chef'@'%';
-- GRANT SELECT ON recipe_nutr_value TO 'Chef'@'%';

FLUSH PRIVILEGES;
```

Σημειώνεται ότι στον σύνδεσμο του GitHub, υπάρχει αναρτημένο το πλήρες αρχείο DDL.

4 DML Script

Το αρχείο DML που εισάγει τα δεδομένα στη βάση δεδομένων βρίσκεται στο GitHub.

Τόσο το DDL όσο και το DML script, μαζί με όλα τα queries που ζητούνται από το 3ο ερώτημα της εργασίας βρίσκονται στον ακόλουθο σύνδεσμο:
https://github.com/Giannisgou/Databases_2024_Mastercook/tree/main/SQL

5 Οδηγίες για την Λειτουργία της Βάσης Δεδομένων

Τα αρχεία που απαρτίζουν τη βάση δεδομένων μας βρίσκονται στο συγκεκριμένο github repository:https://github.com/Giannisgou/Databases_2024_Mastercook
Για την ορθή λειτουργία της βάσης δεδομένων, είναι απαραίτητο τα αρχεία να εκτελεστούν με συγκεκριμένη σειρά.
Συγκεκριμένα, πρώτο πρέπει να εκτελεστεί το αρχείο “**Tables**“ , το οποίο δημιουργεί του πίνακες, τα **procedures**, τους **users**, τα **indexes** και τα **views** της βάσης δεδομένων.
Στην συνέχεια, πρέπει να εκτελεστεί το αρχείο “**Insert_Data**“, το οποίο εισάγει τα **fake data** στην βάση δεδομένων και εκτελεί επίσης τα αρχικά procedures για τα πέντε πρώτα χρόνια του διαγωνισμού (50 αρχικά επεισόδια). Στη συνέχεια δίνεται η δυνατότητα εκτέλεσης περισσότερων επεισοδίων με κλήση της procedure **InsertNextEpisode()**.
Τέλος, στο αρχείο “**Queries_ex_3**” υπάρχουν τα ζητούμενα, από την εκφώνηση της άσκησης, **queries**.