Εθνικό Μετσόβιο Πολυτεχνείο

Email: el21188@mail.ntua.gr / el22435@mail.ntua.gr / el21058@mail.ntua.gr
Ηρώων Πολυτεχνείου 9, Ζωγράφου

Βάσεις Δεδομένων

Εξαμηνιαία Εργασία

Εαρινό Εξάμηνο 2023-24

Μέλη	Διονύσιος Μαχαίρας / Φίλιππος Ξενάκης / Γεώργιος Πνευματικός	Ημερομηνία	26/05/2024
Μητρώα	03122435 / 03121188 / 03121058	Ομάδα	Ομάδα Project 44
Μάθημα	Βάσεις Δεδομένων	Εξάμηνο	6ο

ΟΔΗΓΟΣ ΕΓΚΑΤΑΣΤΑΣΗ ΕΦΑΡΜΟΓΗΣ

Βήμα 1: Κατέβασμα των αρχείων

Κατεβάστε τα αρχεία από το GitHub (https://github.com/George4423/Project-44.git)
Τα αρχεία περιέχονται και στον zip φάκελο

Βήμα 2: Δημιουργία σύνδεσης στο MySQL Workbench

Βήμα 3: Εκτέλεση του αρχείου DDL

Βήμα 4: Εκτέλεση των αρχείων DML

Βήμα 5: Επιβεβαίωση εγκατάστασης

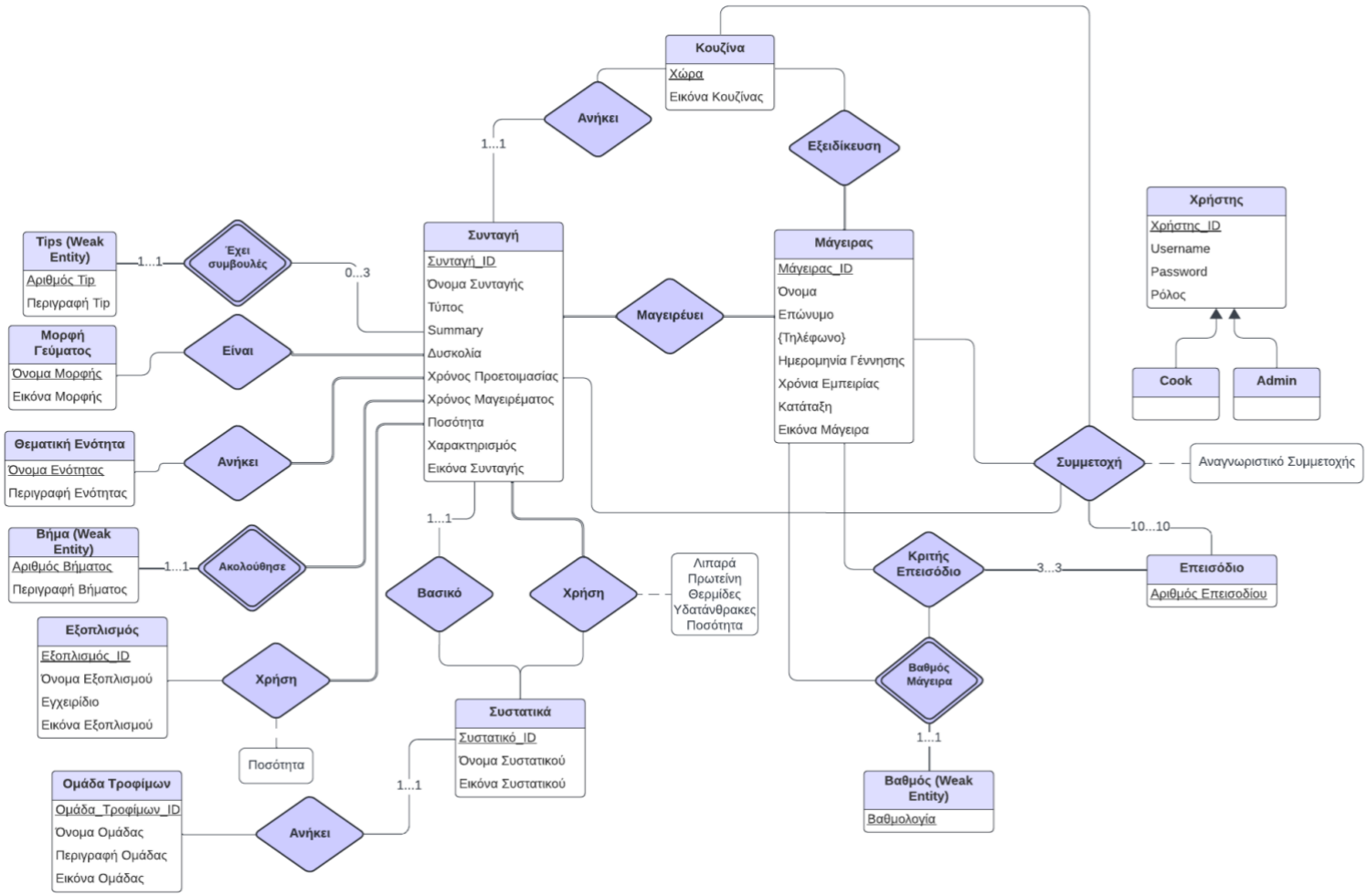
Στο αριστερό πάνελ του MySQL Workbench, ανανεώστε τα δεδομένα (δεξί κλικ και **Refresh**).

Βήμα 6: Εκτελέστε τα Αρχεία με τα Queries και την PrepareEpisode για προσθήκη επεισοδίων.

Συστήνεται η Βάση να έχει τεθεί ως default αφού γίνει import ή τρέξει για πρώτη φορά ώστε μετά κάθε query που τρέχει να αφορά την συγκεκριμένη βάση!!!

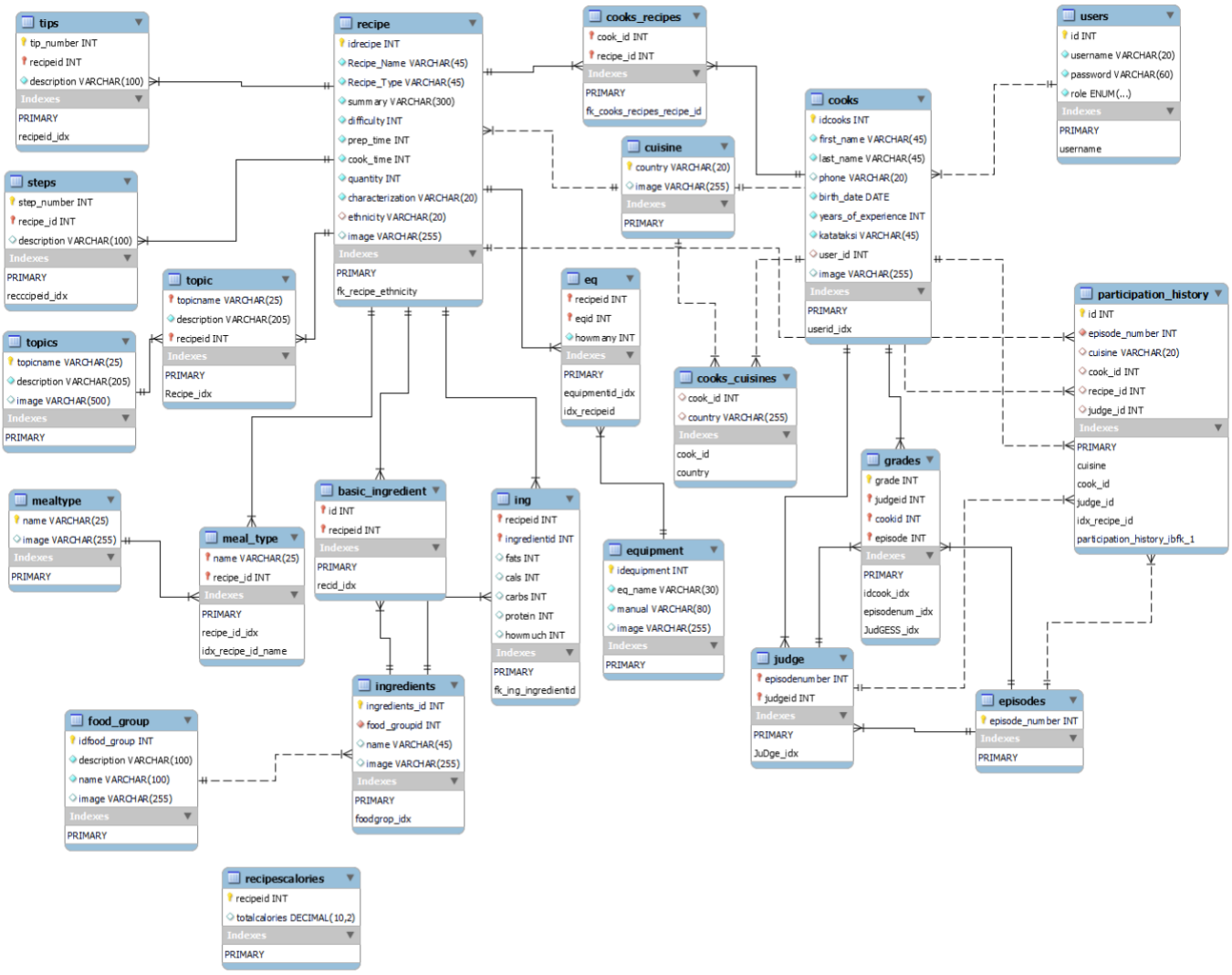
1. ER ΔΙΑΓΡΑΜΜΑ ΕΦΑΡΜΟΓΗΣ

Παρακάτω ακολουθεί το ER διάγραμμα της συγκεκριμένης εργασίας με βάση την εκφώνηση που έχει δοθεί. Κάποιες απαραίτητες επισημάνσεις είναι ότι έχουν χρησιμοποιηθεί κάποιοι περιορισμοί που δίνονται στην εκφώνηση όπως για παράδειγμα ότι κάθε συνταγή έχει από κανένα έως 3 tips (0…3) στην σχέση έχει συμβουλές μεταξύ συνταγών και tips ή ότι κάθε συνταγή ανήκει σε μία και μοναδική κουζίνα (1…1) στην σχέση ανήκει και κουζίνα. Ακόμη ως weak entities έχουν θεωρηθεί τα tips και τα βήματα όπως και οι σχέσεις που τα συνδέουν με τις συνταγές αφού χωρίς τις συνταγές από μόνα τους δεν γίνεται να υπάρχουν. Έτσι έχουμε θεωρήσει και για τον βαθμό ο οποίος χωρίς να έχει κάποιον κριτή να βαθμολογεί κάποιον μάγειρα δεν έχει ουσία ύπαρξης. Γενικά το ER περιέχει όλες τις οντότητες που έχουν δοθεί στην εκφώνηση (οι περισσότερες υπογραμμισμένες) και όλες τις σχέσεις που αναπτύσσονται μεταξύ τους και έχουν ουσία για την βάση μας.



2. ΣΧΕΣΙΑΚΟ ΔΙΑΓΡΑΜΜΑ ΕΦΑΡΜΟΓΗΣ

Υλοποιώντας το σχεσιακό διάγραμμα της βάσης μας, που ανταποκρίνεται καλά στους πίνακες που έχουμε δημιουργήσει παρατηρούμε τα εξής ενδιαφέροντα. Αρχικά όλες οι βασικές οντότητες του ER διαγράμματος μας έχουν έναν δικό τους πίνακα και απεικονίζονται έτσι στο σχεσιακό διάγραμμα της βάσης. Οι σχέσεις ανάμεσα στις οντότητες έχουν υλοποιηθεί και με πίνακες σε κάποιες περιπτώσεις όπως οι μάγειρες που μαγειρεύουν συνταγές (cooks_recipes), οι μάγειρες που έχουν εξειδίκευση σε κάποια κουζίνα (cooks_cuisines), τα υλικά που χρησιμοποιούνται από μια συνταγή (ing), το βασικό συστατικό που έχει κάθε συνταγή (basic_ingredient), ο εξοπλισμός που χρησιμοποιείται σε μια συνταγή (eq), ο κριτής που βαθμολογεί σε ένα επεισόδιο (judge) αλλά και η συμμετοχή σε κάποιο επεισόδιο (participation_history). Άλλες σχέσεις υλοποιούνται μέσω της ύπαρξης foreign keys όπως τα tips, steps και ετικέτες κυρίως επειδή αποτελούν weak entity sets. Ακόμη υπάρχει και ο πίνακας recipesandcalories ο οποίος δυναμικά υπολογίζει τις συνολικές θερμίδες που περιέχονται σε μια συνταγή.



DDL Script:

```
CREATE DATABASE IF NOT EXISTS `COOKCONTEST`;
USE `COOKCONTEST`;

DROP TABLE IF EXISTS `basic_ingredient`;

/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;

CREATE TABLE `basic_ingredient` (
  `id` int NOT NULL,
  `recipeid` int NOT NULL,
  PRIMARY KEY (`id`,`recipeid`),
  KEY `recid_idx` (`recipeid`),
  CONSTRAINT `ingid` FOREIGN KEY (`id`) REFERENCES `ingredients`
(`ingredients_id`),
  CONSTRAINT `recid` FOREIGN KEY (`recipeid`) REFERENCES `recipe` (`idrecipe`)
);

DROP VIEW IF EXISTS `cook_view`;

 CREATE VIEW `cook_view` AS SELECT
 1 AS `cook_id`,
 1 AS `first_name`,
 1 AS `last_name`,
 1 AS `phone`,
 1 AS `birth_date`,
 1 AS `years_of_experience`,
 1 AS `katataksi`,
 1 AS `recipe_id`,
 1 AS `recipe_name`,
 1 AS `recipe_type`,
 1 AS `summary`,
 1 AS `difficulty`,
 1 AS `prep_time`,
 1 AS `cook_time`,
 1 AS `quantity`,
 1 AS `characterization`,
 1 AS `ethnicity`,
 1 AS `participation_history_id`,
 1 AS `episode_number`,
 1 AS `cuisine`,
 1 AS `judge_id`;

 DROP TABLE IF EXISTS `cooks`;

CREATE TABLE `cooks` (
```

```
  `idcooks` int NOT NULL,
  `first_name` varchar(45) NOT NULL,
  `last_name` varchar(45) NOT NULL,
  `phone` varchar(20) DEFAULT NULL,
  `birth_date` date NOT NULL,
  `years_of_experience` int NOT NULL,
  `katataksi` varchar(45) NOT NULL,
  `user_id` int DEFAULT NULL,
  `image` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`idcooks`),
  KEY `userid_idx` (`user_id`),
  CONSTRAINT `fk_user_id` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`)
);

DROP TABLE IF EXISTS `cooks_cuisines`;

CREATE TABLE `cooks_cuisines` (
  `cook_id` int DEFAULT NULL,
  `country` varchar(255) DEFAULT NULL,
  KEY `cook_id` (`cook_id`),
  KEY `country` (`country`),
  CONSTRAINT `cooks_cuisines_ibfk_1` FOREIGN KEY (`cook_id`) REFERENCES
`cooks` (`idcooks`),
  CONSTRAINT `cooks_cuisines_ibfk_2` FOREIGN KEY (`country`) REFERENCES
`cuisine` (`country`)
);

DROP TABLE IF EXISTS `cooks_recipes`;

CREATE TABLE `cooks_recipes` (
  `cook_id` int NOT NULL,
  `recipe_id` int NOT NULL,
  PRIMARY KEY (`cook_id`,`recipe_id`),
  KEY `fk_cooks_recipes_recipe_id` (`recipe_id`),
  CONSTRAINT `fk_cooks_recipes_cook_id` FOREIGN KEY (`cook_id`) REFERENCES
`cooks` (`idcooks`),
  CONSTRAINT `fk_cooks_recipes_recipe_id` FOREIGN KEY (`recipe_id`)
REFERENCES `recipe` (`idrecipe`)
);

DROP TABLE IF EXISTS `cuisine`;

CREATE TABLE `cuisine` (
  `country` varchar(20) NOT NULL,
  `image` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`country`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

DROP TABLE IF EXISTS `episodes`;
```

```
CREATE TABLE `episodes` (
  `episode_number` int NOT NULL,
  PRIMARY KEY (`episode_number`)
);

DROP TABLE IF EXISTS `eq`;

CREATE TABLE `eq` (
  `recipeid` int NOT NULL,
  `eqid` int NOT NULL,
  `howmany` int NOT NULL,
  PRIMARY KEY (`recipeid`,`eqid`),
  KEY `equipmentid_idx` (`eqid`),
  KEY `idx_recipeid` (`recipeid`),
  CONSTRAINT `equipmentid` FOREIGN KEY (`eqid`) REFERENCES `equipment`
(`idequipment`),
  CONSTRAINT `reccipeid` FOREIGN KEY (`recipeid`) REFERENCES `recipe`
(`idrecipe`)
);

DROP TABLE IF EXISTS `equipment`;

CREATE TABLE `equipment` (
  `idequipment` int NOT NULL,
  `eq_name` varchar(30) NOT NULL,
  `manual` varchar(80) NOT NULL,
  `image` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`idequipment`)
);

DROP TABLE IF EXISTS `food_group`;

CREATE TABLE `food_group` (
  `idfood_group` int NOT NULL,
  `description` varchar(100) NOT NULL,
  `name` varchar(100) NOT NULL,
  `image` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`idfood_group`)
);

DROP TABLE IF EXISTS `grades`;

CREATE TABLE `grades` (
  `grade` int NOT NULL,
  `judgeid` int NOT NULL,
  `cookid` int NOT NULL,
  `episode` int NOT NULL,
  PRIMARY KEY (`grade`,`judgeid`,`cookid`,`episode`),
  KEY `idcook_idx` (`cookid`),
  KEY `episodenum_idx` (`episode`),
```

```
  KEY `JudGESS_idx` (`judgeid`),
  CONSTRAINT `episodenum` FOREIGN KEY (`episode`) REFERENCES `episodes`
(`episode_number`),
  CONSTRAINT `idcook` FOREIGN KEY (`cookid`) REFERENCES `cooks` (`idcooks`),
  CONSTRAINT `JudGESS` FOREIGN KEY (`judgeid`) REFERENCES `judge` (`judgeid`)
);

DROP TABLE IF EXISTS `ing`;

CREATE TABLE `ing` (
  `recipeid` int NOT NULL,
  `ingredientid` int NOT NULL,
  `fats` int DEFAULT NULL,
  `cals` int DEFAULT NULL,
  `carbs` int DEFAULT NULL,
  `protein` int DEFAULT NULL,
  `howmuch` int DEFAULT NULL,
  PRIMARY KEY (`recipeid`,`ingredientid`),
  KEY `fk_ing_ingredientid` (`ingredientid`),
  CONSTRAINT `fk_ing_ingredientid` FOREIGN KEY (`ingredientid`) REFERENCES
`ingredients` (`ingredients_id`),
  CONSTRAINT `fk_ing_recipeid` FOREIGN KEY (`recipeid`) REFERENCES `recipe`
(`idrecipe`)
);

DROP TABLE IF EXISTS `ingredients`;

CREATE TABLE `ingredients` (
  `ingredients_id` int NOT NULL,
  `food_groupid` int NOT NULL,
  `name` varchar(45) DEFAULT NULL,
  `image` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`ingredients_id`),
  KEY `foodgrop_idx` (`food_groupid`),
  CONSTRAINT `foodgrop` FOREIGN KEY (`food_groupid`) REFERENCES `food_group`
(`idfood_group`)
);

DROP TABLE IF EXISTS `judge`;

CREATE TABLE `judge` (
  `episodenumber` int NOT NULL,
  `judgeid` int NOT NULL,
  PRIMARY KEY (`episodenumber`,`judgeid`),
  KEY `JuDge_idx` (`judgeid`),
  CONSTRAINT `epiSodenumber` FOREIGN KEY (`episodenumber`) REFERENCES
`episodes` (`episode_number`),
  CONSTRAINT `JuDge` FOREIGN KEY (`judgeid`) REFERENCES `cooks` (`idcooks`)
);
```

```
DROP TABLE IF EXISTS `meal_type`;

DROP TABLE IF EXISTS `meal_type`;
DROP TABLE IF EXISTS `mealtype`;


CREATE TABLE `mealtype` (
  `name` varchar(25) NOT NULL,
  `image` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`name`)
);


CREATE TABLE `meal_type` (
  `name` varchar(25) NOT NULL,
  `recipe_id` int NOT NULL,
  PRIMARY KEY (`name`, `recipe_id`),
  KEY `recipe_id_idx` (`recipe_id`),
  CONSTRAINT `mealtype_fk` FOREIGN KEY (`name`) REFERENCES `mealtype`
(`name`),
  CONSTRAINT `recipe_fk` FOREIGN KEY (`recipe_id`) REFERENCES `recipe`
(`idrecipe`)
);

DROP TABLE IF EXISTS `participation_history`;

CREATE TABLE `participation_history` (
  `id` int NOT NULL AUTO_INCREMENT,
  `episode_number` int NOT NULL,
  `cuisine` varchar(20) DEFAULT NULL,
  `cook_id` int DEFAULT NULL,
  `recipe_id` int DEFAULT NULL,
  `judge_id` int DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `cuisine` (`cuisine`),
  KEY `cook_id` (`cook_id`),
  KEY `judge_id` (`judge_id`),
  KEY `idx_recipe_id` (`recipe_id`),
  CONSTRAINT `participation_history_ibfk_1` FOREIGN KEY (`episode_number`)
REFERENCES `episodes` (`episode_number`),
  CONSTRAINT `participation_history_ibfk_2` FOREIGN KEY (`cuisine`) REFERENCES
`cuisine` (`country`),
  CONSTRAINT `participation_history_ibfk_3` FOREIGN KEY (`cook_id`) REFERENCES
`cooks` (`idcooks`),
  CONSTRAINT `participation_history_ibfk_4` FOREIGN KEY (`recipe_id`)
REFERENCES `recipe` (`idrecipe`),
  CONSTRAINT `participation_history_ibfk_5` FOREIGN KEY (`judge_id`)
REFERENCES `judge` (`judgeid`)
);
DROP TABLE IF EXISTS `recipe`;
```

```
CREATE TABLE `recipe` (
  `idrecipe` int NOT NULL,
  `Recipe_Name` varchar(45) NOT NULL,
  `Recipe_Type` varchar(45) NOT NULL,
  `summary` varchar(300) NOT NULL,
  `difficulty` int NOT NULL,
  `prep_time` int NOT NULL,
  `cook_time` int NOT NULL,
  `quantity` int NOT NULL,
  `characterization` varchar(20) NOT NULL,
  `ethnicity` varchar(20) DEFAULT NULL,
  `image` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`idrecipe`),
  KEY `fk_recipe_ethnicity` (`ethnicity`),
  CONSTRAINT `fk_recipe_ethnicity` FOREIGN KEY (`ethnicity`) REFERENCES
`cuisine` (`country`)
);

DROP TABLE IF EXISTS `recipescalories`;

CREATE TABLE `recipescalories` (
  `recipeid` int NOT NULL,
  `totalcalories` decimal(10,2) DEFAULT NULL,
  PRIMARY KEY (`recipeid`)
);

DROP TABLE IF EXISTS `steps`;

CREATE TABLE `steps` (
  `step_number` int NOT NULL,
  `recipe_id` int NOT NULL,
  `description` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`step_number`,`recipe_id`),
  KEY `recccipeid_idx` (`recipe_id`),
  CONSTRAINT `recccipeid` FOREIGN KEY (`recipe_id`) REFERENCES `recipe`
(`idrecipe`)
);

DROP TABLE IF EXISTS `tips`;

CREATE TABLE `tips` (
  `tip_number` int NOT NULL,
  `recipeid` int NOT NULL,
  `description` varchar(100) NOT NULL,
  PRIMARY KEY (`tip_number`,`recipeid`),
  KEY `recipeid_idx` (`recipeid`),
  CONSTRAINT `recipeid` FOREIGN KEY (`recipeid`) REFERENCES `recipe` (`idrecipe`)
);
```

```
DROP TABLE IF EXISTS `topic`;

CREATE TABLE `topic` (
  `topicname` varchar(25) NOT NULL,
  `description` varchar(205) NOT NULL,
  `recipeid` int NOT NULL,
  PRIMARY KEY (`topicname`,`recipeid`),
  KEY `Recipe_idx` (`recipeid`),
  CONSTRAINT `Recipe` FOREIGN KEY (`recipeid`) REFERENCES `recipe` (`idrecipe`),
  CONSTRAINT `NAME` FOREIGN KEY (`topicname`) REFERENCES `topics`
(`topicname`)
);
CREATE TABLE IF NOT EXISTS topics (
    `topicname` varchar(25) NOT NULL,
  `description` varchar(205) NOT NULL,
    `image` VARCHAR(500),
    PRIMARY KEY (`topicname`)
);
DROP TABLE IF EXISTS `users`;

CREATE TABLE `users` (
  `id` int NOT NULL AUTO_INCREMENT,
  `username` varchar(20) NOT NULL,
  `password` varchar(60) NOT NULL,
  `role` enum('admin','cook') NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `username` (`username`)
);

DROP VIEW IF EXISTS `cook_view`;
CREATE VIEW `cook_view` AS select `c`.`idcooks` AS `cook_id`,`c`.`first_name` AS
`first_name`,`c`.`last_name` AS `last_name`,`c`.`phone` AS `phone`,`c`.`birth_date` AS
`birth_date`,`c`.`years_of_experience` AS `years_of_experience`,`c`.`katataksi` AS
`katataksi`,`r`.`idrecipe` AS `recipe_id`,`r`.`Recipe_Name` AS
`recipe_name`,`r`.`Recipe_Type` AS `recipe_type`,`r`.`summary` AS
`summary`,`r`.`difficulty` AS `difficulty`,`r`.`prep_time` AS `prep_time`,`r`.`cook_time` AS
`cook_time`,`r`.`quantity` AS `quantity`,`r`.`characterization` AS
`characterization`,`r`.`ethnicity` AS `ethnicity`,`ph`.`id` AS
`participation_history_id`,`ph`.`episode_number` AS `episode_number`,`ph`.`cuisine` AS
`cuisine`,`ph`.`judge_id` AS `judge_id` from ((`cooks` `c` join `participation_history` `ph`
on((`c`.`idcooks` = `ph`.`cook_id`))) join `recipe` `r` on((`ph`.`recipe_id` = `r`.`idrecipe`)))
where (`c`.`user_id` = (select `users`.`id` from `users` where (`users`.`username` =
current_user())));
-- Drop existing triggers if they exist
DELIMITER //

DROP TRIGGER IF EXISTS after_cooks_cuisines_insert//
DROP TRIGGER IF EXISTS update_total_calories_after_insert//
DROP TRIGGER IF EXISTS update_total_calories_after_update//
DROP TRIGGER IF EXISTS after_recipe_insert//
```

```
DROP TRIGGER IF EXISTS after_recipe_delete//
DROP TRIGGER IF EXISTS after_cooks_cuisines_delete//
DROP TRIGGER IF EXISTS update_total_calories_after_delete//
DROP TRIGGER IF EXISTS before_tip_insert//

DELIMITER ;

-- Create triggers one at a time
DELIMITER //

CREATE TRIGGER after_cooks_cuisines_insert
AFTER INSERT ON cooks_cuisines
FOR EACH ROW
BEGIN
   DECLARE done INT DEFAULT FALSE;
   DECLARE recipe_id INT;
   DECLARE cur CURSOR FOR
      SELECT idrecipe
      FROM recipe
      WHERE ethnicity = NEW.country;

   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

   OPEN cur;

   read_loop: LOOP
      FETCH cur INTO recipe_id;
      IF done THEN
         LEAVE read_loop;
      END IF;

      INSERT INTO cooks_recipes (cook_id, recipe_id) VALUES (NEW.cook_id,
recipe_id);
   END LOOP;

   CLOSE cur;
END//

DELIMITER ;

DELIMITER //

CREATE TRIGGER update_total_calories_after_insert
AFTER INSERT ON ing
FOR EACH ROW
BEGIN
   -- Calculate the additional calories from the new ingredient
   DECLARE new_calories DECIMAL(10, 2);
   SET new_calories = NEW.howmuch * NEW.cals / 100;
```

```
    -- Check if the recipe already exists in recipescalories
    IF EXISTS (SELECT * FROM recipescalories WHERE recipeid = NEW.recipeid) THEN
        -- Update the total calories
        UPDATE recipescalories
        SET totalcalories = totalcalories + new_calories
        WHERE recipeid = NEW.recipeid;
    ELSE
        -- Insert a new record if the recipe does not exist
        INSERT INTO recipescalories (recipeid, totalcalories)
        VALUES (NEW.recipeid, new_calories);
    END IF;
END//

DELIMITER ;

DELIMITER //

CREATE TRIGGER update_total_calories_after_update
AFTER UPDATE ON ing
FOR EACH ROW
BEGIN
    -- Calculate the calorie difference
    DECLARE old_calories DECIMAL(10, 2);
    DECLARE new_calories DECIMAL(10, 2);
    SET old_calories = OLD.howmuch * OLD.cals / 100;
    SET new_calories = NEW.howmuch * NEW.cals / 100;

    -- Update the total calories in recipescalories
    UPDATE recipescalories
    SET totalcalories = totalcalories - old_calories + new_calories
    WHERE recipeid = NEW.recipeid;
END//

DELIMITER ;

DELIMITER //

CREATE TRIGGER after_recipe_insert
AFTER INSERT ON recipe
FOR EACH ROW
BEGIN
    -- Insert cook_id and recipe_id into cooks_recipes for all matching cooks
    INSERT INTO cooks_recipes (cook_id, recipe_id)
    SELECT cook_id, NEW.idrecipe
    FROM cooks_cuisines
    WHERE country = NEW.ethnicity;
END//

DELIMITER ;
```

```
DELIMITER //

CREATE TRIGGER after_recipe_delete
AFTER DELETE ON recipe
FOR EACH ROW
BEGIN
   -- Delete cook_id and recipe_id from cooks_recipes for the deleted recipe
   DELETE FROM cooks_recipes
   WHERE recipe_id = OLD.idrecipe;
END//

DELIMITER ;

DELIMITER //

CREATE TRIGGER after_cooks_cuisines_delete
AFTER DELETE ON cooks_cuisines
FOR EACH ROW
BEGIN
   DELETE FROM cooks_recipes
   WHERE cook_id = OLD.cook_id
   AND recipe_id IN (
      SELECT idrecipe
      FROM recipe
      WHERE ethnicity = OLD.country
   );
END//

DELIMITER ;

DELIMITER //

CREATE TRIGGER update_total_calories_after_delete
AFTER DELETE ON ing
FOR EACH ROW
BEGIN
   -- Calculate the calories to be subtracted
   DECLARE old_calories DECIMAL(10, 2);
   SET old_calories = OLD.howmuch * OLD.cals / 100;

   -- Update the total calories in recipescalories
   UPDATE recipescalories
   SET totalcalories = totalcalories - old_calories
   WHERE recipeid = OLD.recipeid;
END//

DELIMITER ;

DELIMITER //
```

```
CREATE TRIGGER before_tip_insert
BEFORE INSERT ON tips
FOR EACH ROW
BEGIN
   DECLARE tip_count INT;
   SET tip_count = (SELECT COUNT(*) FROM tips WHERE recipeid = NEW.recipeid);

   IF tip_count >= 3 THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'A recipe cannot have more
than 3 tips.';
   END IF;
END//

DELIMITER ;
```

2.1. ΠΕΡΙΟΡΙΣΜΟΙ ΟΡΘΟΤΗΤΑΣ

Προκειμένου να εξασφαλίσουμε την ορθότητα της ΒΔ έχουμε ορίσει απαραίτητους περιορισμούς. Περιορισμοί ακεραιότητας αποτελούν όλα τα πρωτεύοντα κλειδιά (primary keys) τα οποία εξασφαλίζουν την μοναδικότητα και την ορισμένη τιμή (not NULL), και αντίστοιχα τα εξωτερικά κλειδιά (foreign keys) τα οποία εξασφαλίζουν την αναφορική ακεραιότητα μεταξύ των πινάκων, διασφαλίζοντας πώς οι τιμές αυτές των πινάκων αναφέρονται σε άλλον πίνακα και παρέχοντας μια ασφαλή σύνδεση. Παράλληλα σχετικά με την ακεραιότητα πεδίου τιμών, έχουμε ορίσει όλους τους απαραίτητους περιορισμούς σε κάθε πίνακα ώστε όλα τα δεδομένα να έχουν τον σωστό τύπο και ταυτόχρονα με την δική μας εγκατάσταση ορθών δεδομένων αλλά και με την εγκατάσταση triggers και procedures (που αποτελούν περιορισμούς οριζόμενοι από τους χρήστες) εξασφαλίζουμε και όλα τα υπόλοιπα ζητούμενα που είναι απαραίτητα για την ορθή λειτουργία της εφαρμογής.

Πιο συγκεκριμένα χρησιμοποιούμε ένα trigger το οποίο ενεργοποιείται κατά την εισαγωγή ενός στοιχείου στον πίνακα tips (ο οποίος δέχεται foreign key το recipeid) και σε περίπτωση που ήδη έχουμε τρία tips τότε δεν επιτρέπουμε την εισαγωγή νέου tip για την συνταγή. Έχουμε άλλα δύο triggers που εξασφαλίζουν την ορθή εισαγωγή σε «ενδιάμεσους πίνακες» που συνδέονται με τους κύριους πίνακες και παράλληλα άλλο ένα σημαντικό trigger που χρησιμοποιούμε είναι αυτό που αφορά τις συνολικές θερμίδες μιας συνταγής. Δηλαδή σε περίπτωση που εισάγουμε κάποια ποσότητα νέου συστατικού σε μια συνταγή (μπορούμε και να αφαιρέσουμε αλλά και να κάνουμε κάποιο update ποσότητας) τότε επαναυπολογίζονται οι συνολικές θερμίδες της συνταγής οι οποίες κρατούνται σε έναν «ενδιάμεσο» πίνακα που περιέχει attribute foreign key στο recipeid και περιέχει και τις συνολικές θερμίδες μιας συνταγής.

Τέλος, αξίζει να σημειωθεί πώς ο τρόπος με τον οποίο λειτουργεί η εφαρμογή μας, είναι πώς έχουμε εισάγει αρχικά (judges, Grades, Cooks, …) που πληρούν τους κανονισμούς της εκφώνησης, και στην συνέχεια μπορούμε να δημιουργήσουμε επεισόδια καλώντας την ορισμένη συνάρτηση ‘PrepareEpisode’ η οποία δημιουργεί νέα επεισόδια και εισάγει πάλι τυχαία τα στοιχεία του επεισοδίου (judges, grades, cook_participants) με τρόπο τέτοιο ώστε να διατηρούνται οι περιορισμοί της εκφώνησης. Φυσικά έχουμε ορίσει και όλους τους

περιορισμούς στις μεταβλητές που λαμβάνουν οι τιμές στα επεισόδια, όπως είναι η βαθμολογία των μάγειρων που λαμβάνει τιμές στο εύρος 1-5.

2.2 ΕΥΡΕΤΗΡΙΑ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Στο σημείο αυτό θα αναφερθούμε στα ευρετήρια που έχουμε χρησιμοποιήσει προκειμένου να λειτουργεί με πιο αποδοτικό τρόπο η ΒΔ. Όπως φαίνεται και στο DDL script πού έχουμε παρουσιάσει, όλα τα πρωτεύοντα κλειδιά primary keys) που χρησιμοποιούμε αποτελούν και πρωτεύοντα ευρετήρια των πινάκων. Παράλληλα, σε ορισμένες περιπτώσεις χρησιμοποιούνται και ως ευρετήρια φυσικά τα εξωτερικά κλειδιά (foreign keys) είτε αυτομάτως είτε με την εξωτερική παρέμβαση που γίνεται με χρήση της εντολής KEY στον εκάστοτε πίνακα, γεγονός που βοηθά σημαντικά στην βελτίωση της επίδοσης κατά την αναφορική ακεραιότητα. Ακόμα θέτουμε και ορισμένα attributes ως δευτερεύοντα ευρετήρια με χρήση της KEY. Τονίζουμε πώς όλα τα ευρετήρια βελτιώνουν την απόδοση της βάσης τόσο κατά την εισαγωγή σε ορισμένους πίνακες (triggers) όσο και στα ερωτήματα ζητούμενα ερωτήματα αλλά και στην προσομοίωση των επεισοδίων.

Παραθέτουμε και ευρετήρια που έχουμε χρησιμοποιήσει καθώς και την χρήση τους στα triggers ή στις απαντήσεις των ερωτημάτων

- 3.1 Μέσος όρος Αξιολογήσεων ανά μάγειρα και εθνική κουζίνα

Ευρετήρια:

Table participation_history(cuisine): Ενισχύει την επίδοση στο join καθώς και στο grouping με βάση την κουζίνα

Table participation_history(cook_id, episode): Ενισχύει την επίδοση στο join καθώς το συγκεκριμένο join περιλαμβάνει και τις δύο συνθήκες

Table grades(cookid, episode): Βοηθά στο join κατά το οποίο συνδέει τα grades με το participation_history στο cookid και το episode_number

- 3.2 Μάγειρες που Ανήκουν σε Εθνική Κουζίνα και μάγειρες που συμμετέχουν στον διαγωνισμό από ορισμένη εθνική κουζίνα

Table cooks_cuisines(cook_id, country): Βοηθούν στο join condition μεταξύ cooks και cooks_cuisines και στην επιτάχυνση της μεταβλητής συνθήκης where

Table cooks(idcooks): Υποστηρίζει τα join conditions

- 3.3 Νέοι Μάγειρες με τις Περισσότερες Συνταγές

  Table cooks_recipes(cook_id): Το συγκεκριμένο ευρετήριο βελτίωνει την επίδοση του join μεταξύ cooks και cooks recipes

  Table cooks(idcooks): Υποστηρίζει την γρήγορη λήψη των δεδομένων για τους μάγειρες

- 3.4 Μάγειρες που δεν έχουν υπάρξει κριτές

  Table cooks(idcooks): Υποστηρίζει την γρήγορη λήψη των δεδομένων για τους μάγειρες

  Table judge(judgeid): Ενισχύει την επίδοση του LEFT join ελέγχοντας αποδοτικά για NULL τιμές.

- 3.6 Κορυφαία Ζεύγη που Εμφανίστηκαν σε Επεισόδια

  Στο συγκεκριμένο ερώτημα έχουμε δύο υλοποιήσεις που εκτελούν τον ίδιο σκοπό. Η δεύτερη υλοποίηση είναι ακριβώς ίδια με την πρώτη, ωστόσο, περιλαμβάνει force index σε ένα pair των στοιχείων του meal_type.

  Στην πρώτη υλοποίηση χρησιμοποιούμε τα indices name και recipe_id του meal_type που επιταχύνουν φυσικά την υλοποίηση καθώς εκτελείται ταχύτερα το join και το group by

Στην δεύτερη, όμως, περίπτωση έχουμε προσδιορίσει ένα force index στο pair (name, recipe_id) του meal_type με αποτέλεσμα κατά την επεξεργασία των δεδομένων να ενθαρρύνεται σημαντικά η χρήση του. Αυτό σίγουρα βελτιώνει την επίδοση κατά το join των δύο meal_type όπου οι συνθήκες είναι τα δύο στοιχεία που πλέον εμπεριέχονται στον force index. Ωστόσο, ενδεχομένως να καθυστερήσει το πρόγραμμα κατά την διαδικασία του group by name καθώς πριν είχαμε αποκλειστικά ένα ευρετήριο στο name ενώ πλέον κατά την διαδικασία αυτή θα πρέπει να προσπελαστεί και το recipe_id του meal_type, διαδικασία που μας είναι άσκοπη.

- 3.7 Μάγειρες που συμμετείχαν τουλάχιστον πέντε λιγότερες φορές από τον μάγειρα με τις περισσότερες συμμετοχές

  Table participation_history(cook_id): Το ευρετήριο cook_id στον συγκεκριμένο πίνακα εξυπηρετεί κατά την εντολή group by(cook_id)

- 3.8 Επεισόδιο με τα περισσότερα Εξαρτήματα

Ακριβώς όπως και στο ερώτημα 3.6, έχουμε πάλι δύο υλοποιήσεις.

Στην πρώτη περίπτωση χρησιμοποιούνται τα μεμονωμένα indices του participation_history καθώς και το ευρετήριο του recipe_id του πίνακα eq.

Στην δεύτερη περίπτωση προσδιορίσουμε πάλι force index που περιλαμβάνει το pair (episode_number, recipe_id) στον πίνακα participation_history. Στην συγκεκριμένη περίπτωση αυτό δεν θα μας επωφελήσει καθώς δεν υπάρχει κάποιο σημείο στον κώδικα που να απαιτεί την χρήση της δυάδας και είναι ταχύτερη η υλοποίηση με τα μεμονωμένα ευρετήρια. Παρατηρούμε, λοιπόν, πώς θα πρέπει να χρησιμοποιούμε τα force indices με φειδώ.

- 3.15 Ομάδες Τροφίμων που δεν έχουν εμφανιστεί σε συνταγές

Τονίζουμε ότι για τα υπόλοιπα ερωτήματα που δεν εξετάσαμε, τα ευρετήρια που χρησιμοποιούνται είναι ακριβώς ίδια με αυτά που χρησιμοποιούνται στα ερωτήματα που είδαμε

Table food_group(idfood_group): Χρησιμοποιείται το πρωτεύον κλειδί του πίνακα food_group για τον ταχύτερο υπολογισμό της συνθήκης where

Table basic_ingredients (recipe_id): Χρησιμοποιείται το συγκεκριμένο ευρετήριο του πίνακα basic_ingredients για την επιτάχυνση του join me to participation_history

Προχωράμε τώρα σε ευρετήρια που βελτιώνουν την εισαγωγή στοιχείων μας στην βάση με χρήση των triggers

- Table recipe_calories:
  Χρησιμοποιούμε ως ευρετήριο το πρωτεύον κλειδί recipe_id του πίνακα για την γρήγορη αναζήτηση συγκεκριμένης συνταγής και τον υπολογισμό θερμίδων της.

- Table tips:
  Χρησιμοποιούμε ως index το recipe_id για να βρίσκουμε γρήγορα την συνταγή στην οποία επιθυμούμε να προσθέσουμε κάποιο tip

- Table ing:
  Ο πίνακας ing που ουσιαστικά παρέχει πληροφορίες για συστατικά και ποσότητες που χρησιμοποιούνται σε μια συνταγή χρησιμοποιεί ως ευρετήριο το ingredient_id για την γρήγορη αντιστοίχιση στο συστατικό που απεικονίζει.

  Στο σημείο αυτό αναφέρουμε πώς επιλέξαμε τον πρώτο μάγειρα προκειμένου αυτός μέσω του κωδικού του να έχει δυνατότητα πρόσβασης στην εφαρμογή και να μπορεί να επεξεργαστεί τα δεδομένα του, τα στοιχεία των συνταγών που του έχουν ανατεθεί καθώς και να μπορεί να προσθέσει νέα συνταγή. Ωστόσο, ο μάγειρας αυτός δεν θα έχει δικαίωμα να επεξεργαστεί συνταγή που δεν του έχει ανατεθεί.

Ο κώδικάς που υλοποιεί τον συγκεκριμένο σκοπό απεικονίζεται στην συνέχεια

```
CREATE USER 'firstCook'@'localhost' IDENTIFIED BY 'PASSWORD';

CREATE OR REPLACE VIEW `firstCookData` AS
SELECT *
FROM cooks
WHERE idcooks = 1;

GRANT SELECT, UPDATE ON `cookcontestdb2`.`firstCookData` TO
'firstCook'@'localhost';

UPDATE firstCookData
SET phone = 'newPhoneNumber'
WHERE idcooks = 1;


CREATE OR REPLACE VIEW `firstCookRecipes` AS
SELECT r.*
FROM recipe r
JOIN cooks_recipes cr ON r.idrecipe = cr.recipe_id
WHERE cr.cook_id = 1;

GRANT SELECT, INSERT, UPDATE ON ‘cookcontestdb2’. `firstCookRecipes` TO
'firstCook'@'localhost';


GRANT INSERT ON `cookcontestdb2`.`recipe` TO 'firstCook'@'localhost';

GRANT INSERT ON `cookcontestdb2`.`cooks_recipes` TO 'firstCook'@'localhost';
```

Δημιουργούμε ένα νέο USER firstCook που αναγνωρίζεται από την εφαρμογή με τον κωδικό του. Επιλέγουμε στον συγκεκριμένο χρήστη να δώσουμε δικαίωμα πρόσβασης και αλλαγής στα προσωπικά του δεδομένα. Αυτό το κάνουμε δίνοντας του ένα view στα στοιχεία του και έπειτα εκτελώντας GRANT SELECT, UPDATE στα δεδομένα του. Ενδεικτικά κάνουμε update στο τηλέφωνό του.

Στην συνέχεια δίνουμε το δικαίωμα στον συγκεκριμένο χρήστη να έχει πρόσβαση και σε όλες τις συνταγές του με αντίστοιχο τρόπο όπως πριν. Αυτήν την φορά πρώτα του δίνουμε view στο firstCookRecipes που εμπεριέχει μόνο τις συνταγές του και έπειτα του δίνουμε το δικαίωμα να προσθέσει συνταγή. Για να συμβεί αυτό φυσικά πρέπει να προσθέσει τον συνδυαστικά τόσο στον πίνακα recipe όσο και τον πίνακα cooks_recipes.