ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Βάσεις Δεδομένων

Διαγωνισμός Μαγειρικής

Εξαμηνιαία Εργασία (Ακαδημαϊκό έτος 2023 – 2024)

Ομάδα: 66

Ντουντουνάκης Γεώργιος / 03121881

Τσουρβελούδης Χρήστος / 03121059

Χρήστου Βασίλειος / 03121054

Περιεχόμενα

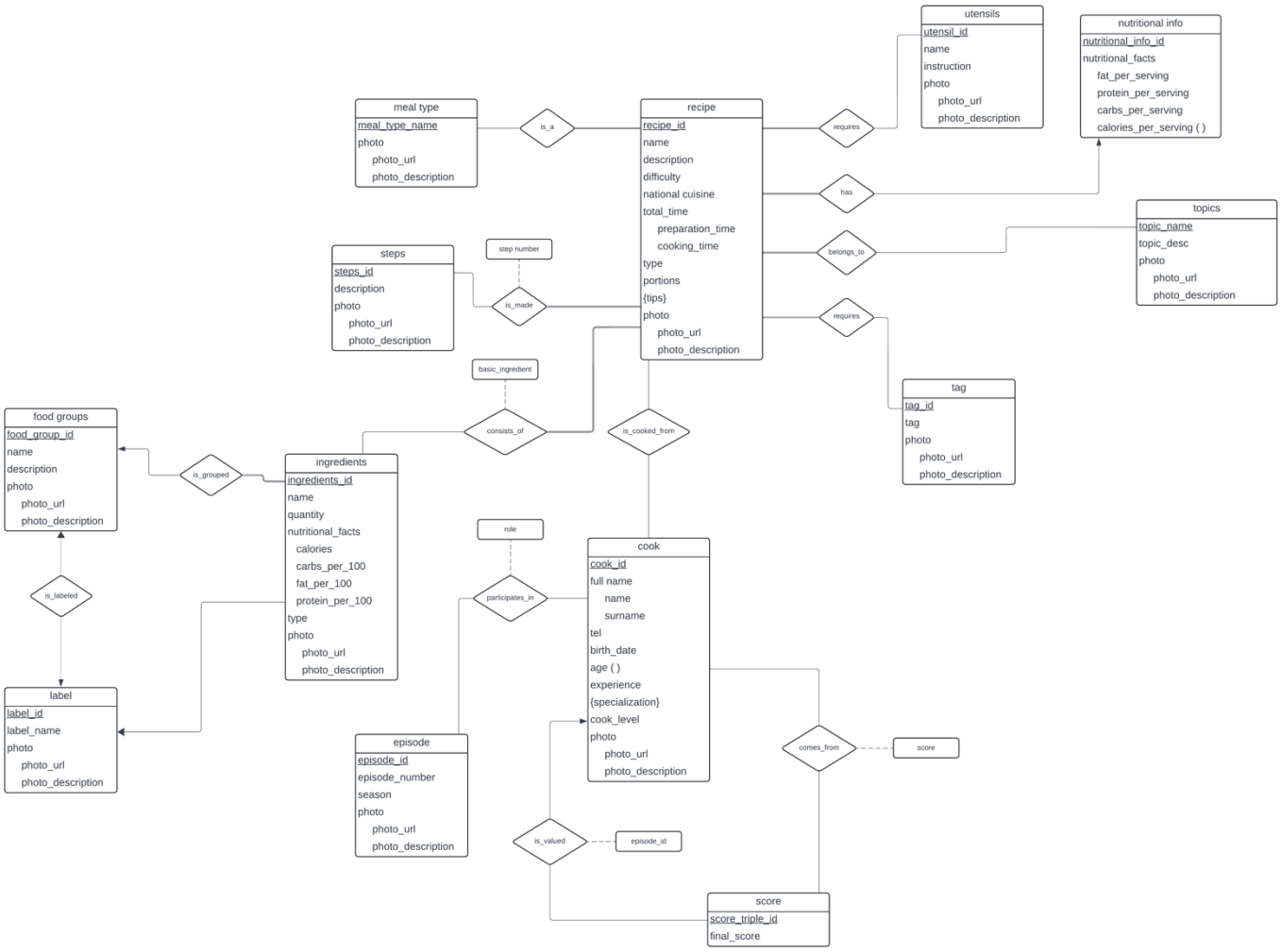
1. **Εισαγωγή**

Ζητούμενο της εργασίας είναι η σχεδίαση και υλοποίηση μιας βάσης δεδομένων για δημοφιλή διαγωνισμό μαγειρικής. Μέσω αυτής θα αποθηκεύονται και θα διαχειρίζονται απαιτούμενα για τον διαγωνισμό δεδομένα αναφορικά με τις συνταγές, τα υλικά και τον εξοπλισμό. Τα δεδομένα αυτά θα τροποποιούνται τόσο από τον διαχειριστή του διαγωνισμού όσο και από τους ίδιους τους μάγειρες, οι οποίοι όμως θα έχουν περιορισμένη πρόσβαση, μέσω συστήματος εξακρίβωσης στοιχείων (username, password).

2. **Βάση Δεδομένων**

2.1 Διάγραμμα Οντοτήτων - Συσχετίσεων (Entity – Relational Diagram)

Αρχικά, με βάση τις δοσμένες από τον διαγωνισμό απαιτούμενες προδιαγραφές σχεδιάζουμε το διάγραμμα οντοτήτων συσχετίσεων το οποίο δίνεται παρακάτω (Εικόνα 1.1):

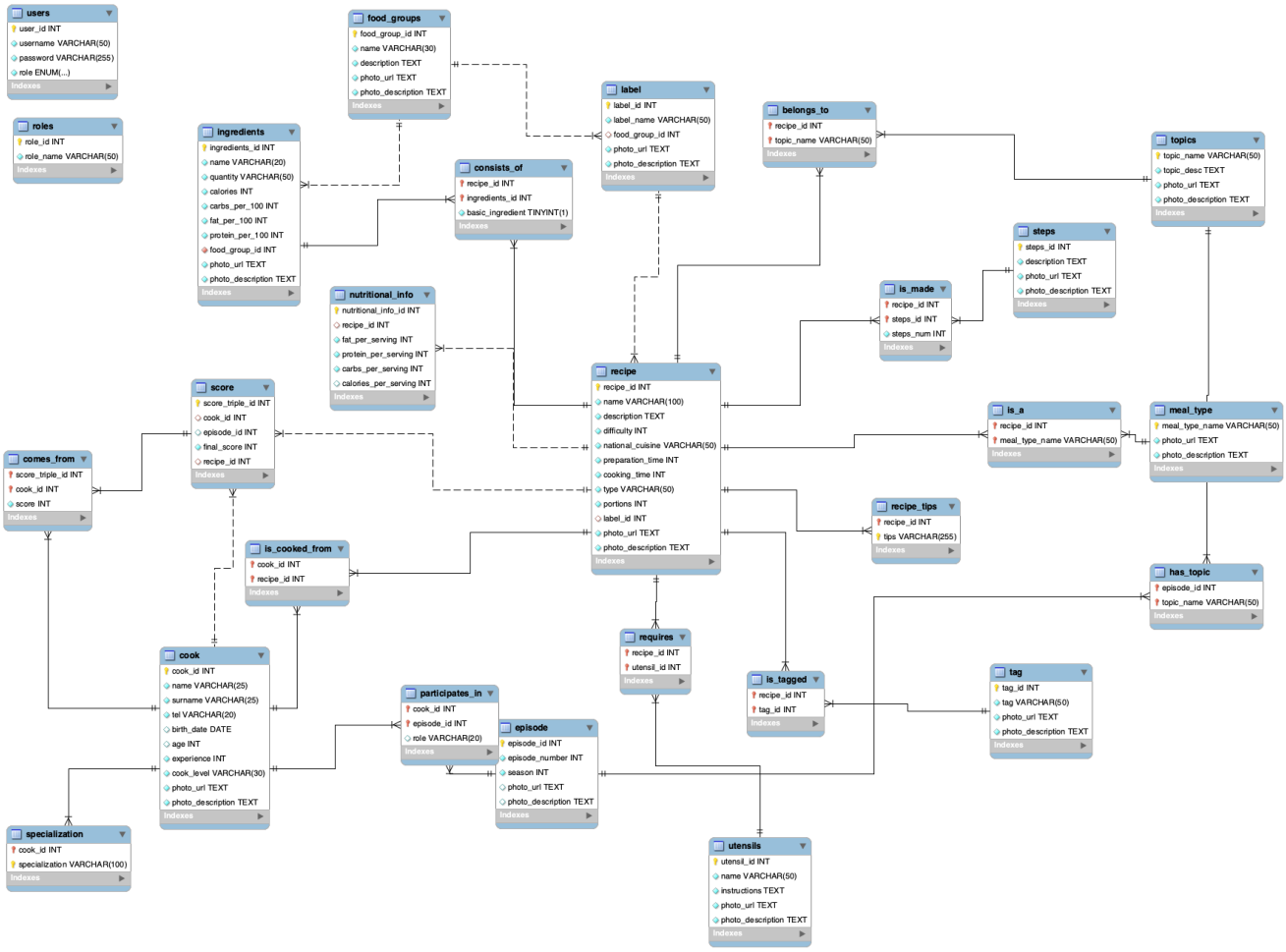


Εικόνα 2.1

Η σχεδίαση του ER διαγράμματος έγινε στο πρόγραμμα “lucid”, το διάγραμμα δίνεται μαζί με την εργασία με το όνομα “er_diagram.png”.

2.2 Σχεσιακό Σχήμα (Relational Schema)

Με βάση το παραπάνω διάγραμμα οντοτήτων - συσχετίσεων σχεδιάζουμε το σχεσιακό διάγραμμα και το παραθέτουμε (Εικόνα 2.2):



Εικόνα 2.2

Το Relational Schema σχεδιάστηκε μέσω της εφαρμογής “MySQL Workbench” και δίνεται μαζί με την εργασία με το όνομα “relational_schema.png”.

Στο σχεσιακό διάγραμμα είναι υλοποιημένες όλες οι οντότητες που παρατηρούνται και στο ER – διάγραμμα, μαζί με τις αντίστοιχες σχέσεις. Συγκεκριμένα:

- Η οντότητα recipe αναφέρεται δεδομένα που οφείλει να έχει η κάθε συνταγή, δηλαδή όνομα, περιγραφή, δυσκολία, εθνική κουζίνα, χρόνος προετοιμασίας και χρόνο μαγειρέματος, τύπο (το οποίο καθορίζει αν μια συνταγή είναι συνταγή μαγειρικής ή ζαχαροπλαστικής), τις μερίδες που προκύπτουν καθώς και δεδομένα σχετικά με τη φωτογραφία που θα την συνοδεύει (διεύθυνση φωτογραφίας και περιγραφή φωτογραφίας). Κάθε συνταγή συνοδεύεται από ένα recipe_id που την ξεχωρίζει από τις υπόλοιπες.
- Η κάθε συνταγή με βάση τις προϋποθέσεις μπορεί να συνοδεύεται από ετικέτες (απεριόριστες), οι οποίες αποθηκεύονται στην οντότητα tag, η οποία διαθέτει tag_id, την ετικέτα καθώς και δεδομένα σχετικά με τη φωτογραφία. Για να γίνει η σύνδεση μεταξύ των δύο οντοτήτων (many to many) χρησιμοποιούμε τον πίνακα is_tagged που περιέχει τα κλειδιά των δύο οντοτήτων (υπερκλειδί).
- Επίσης, η συνταγή διαθέτει χαρακτηρισμό (label), η σύνδεση της κάθε συνταγής με χαρακτηρισμό θα εξηγηθεί παρακάτω. Το κάθε διαθέτει label_id, την ομάδα τροφίμων στην οποία αναφέρεται και

δεδομένα σχετικά με τη φωτογραφία που θα το συνοδεύει (διεύθυνση φωτογραφίας και περιγραφή φωτογραφίας).

- Επίσης, καθένα από τα συστατικά της συνταγής αποθηκεύονται στην οντότητα ingredients. Αυτή περιέχει το όνομα του συστατικού, τη ποσότητα του, τις διατροφικές αξίες (θερμίδες, υδατάνθρακες, λιπαρά και πρωτεΐνη ανά 100 gr) καθώς και πληροφορίες για τις εικόνες. Κάθε συστατικό συνδέεται με τη συνταγή μέσω του πίνακα consists_of που περιέχει τα κλειδιά των δύο οντοτήτων (υπερκλειδί).
- Κάθε συστατικό ανήκει σε μια ομάδα τροφίμων. Κάθε ομάδα τροφίμων περιέχεται στην οντότητα food_groups, η οποία περιέχει το id της ομάδα τροφίμων, το όνομα της, την περιγραφή καθώς και απαραίτητα δεδομένα για τις φωτογραφίες.
- Κάθε συνταγή διαθέτει διατροφική αξία που περιέχονται στην οντότητα nutritional_info (Έχουμε κάνει την υπόθεση η κάθε συνταγή δεν πρόκειται να έχει ακριβώς τη ίδια διατροφική αξία με την άλλη). Στις διατροφικές αξίες περιέχονται οι θερμίδες (οι θερμίδες δεν εισάγονται χειροκίνητα αλλά υπολογίζονται δυναμικά όπως θα εξηγήσουμε παρακάτω), οι υδατάνθρακες, τα λιπαρά και οι πρωτεΐνη ανά μερίδα.
- Επιπλέον δημιουργούμε την οντότητα. utensils που περιέχει όλα τα σκεύη που διαθέτει ο διαγωνισμός. Κάθε σκεύος έχει id, όνομα οδηγίες χρήσης και πληροφορίες αναγκαίες για τις εικόνες. Κάθε σκεύος συνδέεται με τη συνταγή που το απαιτεί μέσω του πίνακα requires που περιέχει τα κλειδιά των δύο οντοτήτων (υπερκλειδί).
- Ο πίνακας tips περιέχει τις συμβουλές για κάθε συνταγή (έως τρεις) και το id της συνταγής στην οποία αναφέρεται. Δημιουργήθηκε διότι το attribute tips που ανήκει στην οντότητα recipe είναι multivalued (Εικόνα 2.1).
- Η οντότητα steps δηλώνει τα βήματα που απαιτούνται για τη δημιουργία της συνταγής τα οποία για κάθε συνταγή εκτελούνται σειριακά. Η οντότητα steps περιέχει το id του βήματος, την περιγραφή καθώς και πληροφορίες αναγκαίες για την εικόνα. Κάθε step συνδέεται με τη συνταγή που αναφέρεται μέσω του πίνακα is_made που περιέχει τα κλειδιά των δύο οντοτήτων (υπερκλειδί) καθώς και τον αριθμό του βήματος.
- Μια συνταγή μπορεί να ανήκει σε έναν ή περισσότερους τύπους γευμάτων. Αυτοί περιέχονται στην οντότητα meal_type που περιέχει το όνομα του τύπου γεύματος (το όποιο αποτελεί και το κλειδί) καθώς και τις απαραίτητε πληροφορίες για την εικόνα. Κάθε συνταγή συνδέεται με τους τύπους γεύματος που ανήκει μέσω του πίνακα is_a που περιέχει τα κλειδιά των δύο οντοτήτων (υπερκλειδί).
- Κάθε συνταγή ανήκει σε μια ή περισσότερες θεματικές ενότητες οι οποίες περιέχονται στην οντότητα topics. Η οντότητα αυτή περιλαμβάνει το όνομα της θεματικής ενότητας (που αποτελεί και το κλειδί) τη περιγραφή της καθώς και απαιτούμενα δεδομένα για τις πληροφορίες. Κάθε συνταγή συνδέεται με τις θεματικές ενότητες που ανήκει μέσω του πίνακα belongs_to που περιέχει τα κλειδιά των δύο οντοτήτων (υπερκλειδί).
- Τα δεδομένα για κάθε μάγειρα περιέχονται στην οντότητα cook που περιέχει το όνομα, το επώνυμο, το τηλέφωνο (θεωρούμε κάθε μάγειρας έχει μόνο ένα τηλεφωνικό αριθμό), την ημερομηνία γέννησης, την ηλικία (η οποία υπολογίζεται δυναμικά με βάση το τρέχον έτος), τα χρόνια επαγγελματικής εμπειρίας και το χαρακτηρισμό της επαγγελματικής κατάρτισης. Επιπλέον, διαθέτει εξειδίκευση σε μια η περισσότερες εθνικές κουζίνες, κάθε εξειδίκευση αποθηκεύεται στη βοηθητική οντότητα specialization (καθώς η εμπειρία είναι multivalued attribute). Η κάθε συνταγή συνδέεται με τον κάθε μάγειρα που την μαγειρεύει μέσω του πίνακα is_cooked_from που περιέχει τα κλειδιά των δύο οντοτήτων (υπερκλειδί).
- Κάθε επεισόδιο αποθηκεύεται στην οντότητα episode, η όποια περιέχει το id του επεισοδίου, τον αριθμό του επεισοδίου για συγκεκριμένη σεζόν, τον αριθμό της σεζόν (κάνουμε την σύμβαση ότι ο αριθμός της σεζόν ταυτίζεται με την χρονιά που πραγματοποιείται κάθε ένας διαγωνισμός) καθώς και τις απαιτούμενες πληροφορίες για κάθε εικόνα. Κάθε επεισόδιο διαθέτει θεματικές ενότητες, όποτε οι δύο οντότητές συνδέονται μέσω του πίνακα has_topic που περιέχει τα κλειδιά των δύο οντοτήτων (υπερκλειδί).
- Για κάθε διαγωνιζόμενο που συμμετέχει σε συγκεκριμένο επεισόδιο και για συγκεκριμένη συνταγή που μαγειρεύει του αποδίδεται ένας βαθμός (score), το οποίο αποθηκεύεται στην οντότητα score. Η οντότητα score περιέχει το id της βαθμολογίας, την μέση βαθμολογία που προκύπτει από τους τρεις κριτές (οι επιμέρους βαθμολογίες προκύπτουν από το πρόγραμμα του διαγωνισμού που θα αναφερθεί παρακάτω) καθώς και τα id της συνταγής και του μάγειρα. Ο κάθε μάγειρας – κριτής

συνδέεται με το score μέσω του πίνακα comes_from που περιέχει τα κλειδιά των δύο οντοτήτων (υπερκλειδί) καθώς και την βαθμολογία που εκείνος έβαλε. Κάθε μάγειρας συνδέεται με το επεισόδιο που συμμετέχει μέσω του πίνακα participates_in που περιέχει τα κλειδιά των δύο οντοτήτων (υπερκλειδί), καθώς και τον ρόλο που έχει στον διαγωνισμό (κριτής ή διαγωνιζόμενος).

- Οι οντότητες user και roles δεν προκύπτουν από το διάγραμμα ER αλλά έχουν δημιουργηθεί για τη ταυτοποίηση των χρηστών (Σελίδα 16).

2.3 Ευρετήρια (Indexes)

Έχοντας ως στόχο την ελαχιστοποίηση του χρόνου των queries δημιουργούμε κατάλληλα ευρετήρια (indexes). Σημειώνεται ότι στη MySQL γίνεται αυτόματα indexing για τα primary keys. Λαμβάνοντας αυτό υπόψιν, παρατηρήσαμε πόσες φορές γίνεται αναφορά σε table ή attribute από τα queries και τα triggers και επιλέξαμε να δημιουργήσουμε τα παρακάτω indexes:

```
1. CREATE INDEX idx_cook_name_surname ON cook(name, surname);
2. CREATE INDEX idx_score_cook_id ON score(cook_id);
3. CREATE INDEX idx_score_recipe_id ON score(recipe_id);
4. CREATE INDEX idx_recipe_national_cuisine ON recipe(national_cuisine);
5. CREATE INDEX idx_episode_season ON episode(season);
6. CREATE INDEX idx_cook_experience ON cook(experience);
7. CREATE INDEX idx_score_final_score ON score(final_score);
8. CREATE INDEX idx_participates_in_role ON participates_in(role);
```

Το πρώτο αποτελεί ένα composite index που δίνει τις πληροφορίες του cook για συγκεκριμένο ονοματεπώνυμο και χρησιμοποιείται ως επί το πλείστων σε όλα τα queries. Τα δύο επόμενα δημιουργούν index του score με βάση το cook_id και το recipe_id, αντίστοιχα (Query 1ο , 11ο). Το επόμενό κάνει index της συνταγής με βάση την εθνική κουζίνα (Query 1ο, 2ο, 10ο). Το 5ο κάνει indexing τα δεδομένα του επεισοδίου με βάση τη σεζόν (τη χρονιά με βάση τη δική με σύμβαση) (Query 5ο, 9ο, 10ο, 12ο). Το 6ο κάνει indexing τον cook με βάση την εμπειρία του (χρησιμοποιείται στο Query 13ο). Το 7ο κάνει indexing το score με βάση την βαθμολογία του διαγωνιζόμενου (χρησιμοποιείται στο Query 1ο). Τέλος το 8ο κάνει indexing το participates_in με βάση τον ρόλο ώστε να έχουμε γρήγορη πρόσβαση στους διαγωνιζόμενους και τους κριτές.

2.4 DDL (Data Definition Language)

Παρακάτω δίνεται το DDL Script που κατασκευάζει τους πίνακες καθώς και τα ευρετήρια που αναφέραμε παραπάνω (Παρατίθεται μαζί με την εργασία στο αρχείο με όνομα “ddl.sql”):

```
 1.     USE cook_show;
 2.
 3. SET FOREIGN_KEY_CHECKS = 0;
 4.
 5. DROP TABLE IF EXISTS recipe_tips;
 6. DROP TABLE IF EXISTS is_tagged;
 7. DROP TABLE IF EXISTS requires;
 8. DROP TABLE IF EXISTS is_a;
 9. DROP TABLE IF EXISTS is_made;
10. DROP TABLE IF EXISTS consists_of;
11. DROP TABLE IF EXISTS has_label;
12. DROP TABLE IF EXISTS belongs_to;
13. DROP TABLE IF EXISTS specialization;
14. DROP TABLE IF EXISTS is_cooked_from;
15. DROP TABLE IF EXISTS participates_in;
16. DROP TABLE IF EXISTS has_topic;
17. DROP TABLE IF EXISTS comes_from;
18. DROP TABLE IF EXISTS score;
19. DROP TABLE IF EXISTS roles;
20. DROP TABLE IF EXISTS recipe;
21. DROP TABLE IF EXISTS tag;
22. DROP TABLE IF EXISTS utensils;
23. DROP TABLE IF EXISTS meal_type;
24. DROP TABLE IF EXISTS steps;
25. DROP TABLE IF EXISTS label;
26. DROP TABLE IF EXISTS ingredients;
```

```sql
DROP TABLE IF EXISTS food_groups;
DROP TABLE IF EXISTS topics;
DROP TABLE IF EXISTS cook;
DROP TABLE IF EXISTS episode;
DROP TABLE IF EXISTS nutritional_info;
DROP TABLE IF EXISTS users;
DROP TABLE IF EXISTS user_roles;

SET FOREIGN_KEY_CHECKS = 1;

CREATE TABLE food_groups (
    food_group_id INTEGER UNSIGNED,
    name VARCHAR(30) NOT NULL,
    description TEXT NOT NULL,
    photo_url TEXT NOT NULL,
    photo_description TEXT NOT NULL,
    PRIMARY KEY (food_group_id)
);

CREATE TABLE label (
    label_id INTEGER UNSIGNED,
    label_name VARCHAR(50) NOT NULL,
    food_group_id INTEGER UNSIGNED,
    photo_url TEXT NOT NULL,
    photo_description TEXT NOT NULL,
    PRIMARY KEY (label_id),
    FOREIGN KEY (food_group_id)
        REFERENCES food_groups (food_group_id)
);

CREATE TABLE recipe (
    recipe_id INTEGER UNSIGNED AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    description TEXT NOT NULL,
    difficulty INTEGER NOT NULL,
    national_cuisine VARCHAR(50) NOT NULL,
    preparation_time INTEGER NOT NULL,
    cooking_time INTEGER NOT NULL,
    type VARCHAR(50) NOT NULL,
    portions INT NOT NULL,
    label_id INT UNSIGNED,
    photo_url TEXT NOT NULL,
    photo_description TEXT NOT NULL,
    PRIMARY KEY (recipe_id),
    FOREIGN KEY (label_id)
        REFERENCES label (label_id),
    CONSTRAINT difficulty_constraint CHECK (difficulty IN (1 , 2, 3, 4, 5)),
    CONSTRAINT portions_constraint CHECK (portions > 0),
    CONSTRAINT preparation_time_constraint CHECK (preparation_time >= 0),
    CONSTRAINT cooking_time_constraint CHECK (cooking_time >= 0),
    CONSTRAINT chk_includes_cuisine CHECK (
        LOCATE('Cuisine', national_cuisine) > 0
    ),
    CONSTRAINT chk_type CHECK (type IN ('cooking', 'pastry', 'Cooking', 'Pastry'))
);

CREATE TABLE recipe_tips (
    recipe_id INTEGER UNSIGNED,
    tips VARCHAR(255),
    PRIMARY KEY (recipe_id , tips),
    FOREIGN KEY (recipe_id)
        REFERENCES recipe (recipe_id)
);

CREATE TABLE tag (
    tag_id INTEGER UNSIGNED,
    tag VARCHAR(50) NOT NULL,
    photo_url TEXT NOT NULL,
    photo_description TEXT NOT NULL,
    PRIMARY KEY (tag_id)
```

```
 97. );
 98.
 99. CREATE TABLE is_tagged (
100.     recipe_id INTEGER UNSIGNED,
101.     tag_id INTEGER UNSIGNED,
102.     PRIMARY KEY (recipe_id , tag_id),
103.     FOREIGN KEY (recipe_id)
104.         REFERENCES recipe (recipe_id),
105.     FOREIGN KEY (tag_id)
106.         REFERENCES tag (tag_id)
107. );
108.
109. CREATE TABLE utensils (
110.     utensil_id INTEGER UNSIGNED NOT NULL,
111.     name VARCHAR(50) NOT NULL,
112.     instructions TEXT NOT NULL,
113.     photo_url TEXT NOT NULL,
114.     photo_description TEXT NOT NULL,
115.     PRIMARY KEY (utensil_id)
116. );
117.
118. CREATE TABLE requires (
119.     recipe_id INTEGER UNSIGNED,
120.     utensil_id INTEGER UNSIGNED,
121.     PRIMARY KEY (recipe_id , utensil_id),
122.     FOREIGN KEY (recipe_id)
123.         REFERENCES recipe (recipe_id),
124.     FOREIGN KEY (utensil_id)
125.         REFERENCES utensils (utensil_id)
126. );
127.
128. CREATE TABLE meal_type (
129.     meal_type_name VARCHAR(50),
130.     photo_url TEXT NOT NULL,
131.     photo_description TEXT NOT NULL,
132.     PRIMARY KEY (meal_type_name)
133. );
134.
135. CREATE TABLE is_a (
136.     recipe_id INTEGER UNSIGNED,
137.     meal_type_name VARCHAR(50),
138.     PRIMARY KEY (recipe_id , meal_type_name),
139.     FOREIGN KEY (recipe_id)
140.         REFERENCES recipe (recipe_id),
141.     FOREIGN KEY (meal_type_name)
142.         REFERENCES meal_type (meal_type_name)
143. );
144.
145. CREATE TABLE steps (
146.     steps_id INTEGER UNSIGNED,
147.     description TEXT NOT NULL,
148.     photo_url TEXT NOT NULL,
149.     photo_description TEXT NOT NULL,
150.     PRIMARY KEY (steps_id)
151. );
152.
153. CREATE TABLE is_made (
154.     recipe_id INTEGER UNSIGNED,
155.     steps_id INTEGER UNSIGNED,
156.     steps_num INTEGER UNSIGNED NOT NULL,
157.     CONSTRAINT steps_num CHECK (steps_num > 0),
158.     PRIMARY KEY (recipe_id , steps_id),
159.     FOREIGN KEY (recipe_id)
160.         REFERENCES recipe (recipe_id),
161.     FOREIGN KEY (steps_id)
162.         REFERENCES steps (steps_id)
163. );
164.
165.
166. CREATE TABLE ingredients (
```

```
    ingredients_id INTEGER UNSIGNED,
    name VARCHAR(20) NOT NULL,
    quantity varchar(50) NOT NULL,
    calories INTEGER NOT NULL,
    carbs_per_100 INTEGER NOT NULL,
    fat_per_100 INTEGER NOT NULL,
    protein_per_100 INTEGER NOT NULL,
    food_group_id INTEGER UNSIGNED NOT NULL,
    photo_url TEXT NOT NULL,
    photo_description TEXT NOT NULL,
    PRIMARY KEY (ingredients_id),
    FOREIGN KEY (food_group_id)
        REFERENCES food_groups (food_group_id),
    CONSTRAINT calories CHECK (calories >= 0),
    CONSTRAINT carbs_per_100 CHECK (carbs_per_100 >= 0),
    CONSTRAINT fat_per_100 CHECK (fat_per_100 >= 0),
    CONSTRAINT protein_per_100 CHECK (protein_per_100 >= 0)
);

CREATE TABLE consists_of (
    recipe_id INTEGER UNSIGNED,
    ingredients_id INTEGER UNSIGNED,
    basic_ingredient BOOLEAN NOT NULL,
    PRIMARY KEY (recipe_id , ingredients_id),
    FOREIGN KEY (recipe_id)
        REFERENCES recipe (recipe_id),
    FOREIGN KEY (ingredients_id)
        REFERENCES ingredients (ingredients_id)
);

CREATE TABLE nutritional_info (
    nutritional_info_id INTEGER UNSIGNED,
    recipe_id INTEGER UNSIGNED,
    fat_per_serving INTEGER NOT NULL,
    protein_per_serving INTEGER NOT NULL,
    carbs_per_serving INTEGER NOT NULL,
    calories_per_serving INTEGER,
    PRIMARY KEY (nutritional_info_id),
    CONSTRAINT calories_per_serving CHECK (calories_per_serving >= 0),
    CONSTRAINT carbs_per_serving CHECK (carbs_per_serving >= 0),
    CONSTRAINT fat_per_serving CHECK (fat_per_serving >= 0),
    CONSTRAINT protein_per_serving CHECK (protein_per_serving >= 0),
    FOREIGN KEY (recipe_id)
        REFERENCES recipe (recipe_id)
);

CREATE TABLE topics (
    topic_name VARCHAR(50),
    topic_desc TEXT NOT NULL,
    photo_url TEXT NOT NULL,
    photo_description TEXT NOT NULL,
    PRIMARY KEY (topic_name)
);

CREATE TABLE belongs_to (
    recipe_id INTEGER UNSIGNED,
    topic_name VARCHAR(50),
    PRIMARY KEY (recipe_id , topic_name),
    FOREIGN KEY (recipe_id)
        REFERENCES recipe (recipe_id),
    FOREIGN KEY (topic_name)
        REFERENCES topics (topic_name)
);

CREATE TABLE cook (
    cook_id INTEGER UNSIGNED,
    name VARCHAR(25) NOT NULL,
    surname VARCHAR(25) NOT NULL,
    tel VARCHAR(20) NOT NULL,
    birth_date DATE,
```

```
237.     age INTEGER UNSIGNED,
238.     experience INTEGER NOT NULL,
239.     cook_level VARCHAR(30) NOT NULL,
240.     photo_url TEXT NOT NULL,
241.     photo_description TEXT NOT NULL,
242.     PRIMARY KEY (cook_id),
243.     CONSTRAINT cook_level_constraint CHECK (cook_level IN ('a-cook' , 'b-cook', 'c-cook',
'chef', 'sous-chef'))
244. );
245.
246. CREATE TABLE specialization (
247.     cook_id INTEGER UNSIGNED,
248.     specialization VARCHAR(100),
249.     PRIMARY KEY (cook_id , specialization),
250.     FOREIGN KEY (cook_id)
251.         REFERENCES cook (cook_id)
252. );
253.
254. CREATE TABLE is_cooked_from (
255.     cook_id INTEGER UNSIGNED,
256.     recipe_id INTEGER UNSIGNED,
257.     PRIMARY KEY (cook_id , recipe_id),
258.     FOREIGN KEY (cook_id)
259.         REFERENCES cook (cook_id),
260.     FOREIGN KEY (recipe_id)
261.         REFERENCES recipe (recipe_id)
262. );
263.
264. CREATE TABLE episode (
265.     episode_id INTEGER UNSIGNED AUTO_INCREMENT,
266.     episode_number INTEGER NOT NULL,
267.     season INTEGER NOT NULL,
268.     photo_url TEXT ,
269.     photo_description TEXT,
270.     PRIMARY KEY (episode_id)
271. );
272.
273. CREATE TABLE participates_in (
274.     cook_id INTEGER UNSIGNED,
275.     episode_id INTEGER UNSIGNED,
276.     role VARCHAR(20),
277.     PRIMARY KEY (cook_id , episode_id),
278.     FOREIGN KEY (cook_id)
279.         REFERENCES cook (cook_id),
280.     FOREIGN KEY (episode_id)
281.         REFERENCES episode (episode_id)
282. );
283.
284. CREATE TABLE has_topic (
285.     episode_id INTEGER UNSIGNED,
286.     topic_name VARCHAR(50),
287.     PRIMARY KEY (episode_id , topic_name),
288.     FOREIGN KEY (episode_id)
289.         REFERENCES episode (episode_id),
290.     FOREIGN KEY (topic_name)
291.         REFERENCES topics (topic_name)
292. );
293.
294. CREATE TABLE score (
295.     score_triple_id INTEGER UNSIGNED AUTO_INCREMENT,
296.     cook_id INTEGER UNSIGNED,
297.     episode_id INTEGER UNSIGNED,
298.     final_score INTEGER NOT NULL,
299.     recipe_id INTEGER UNSIGNED,
300.     PRIMARY KEY (score_triple_id),
301.     FOREIGN KEY (cook_id)
302.         REFERENCES cook (cook_id),
303.     FOREIGN KEY (recipe_id)
304.         REFERENCES recipe (recipe_id),
```

```
    CONSTRAINT final_score_constraint CHECK (final_score IN (3 , 4, 5, 6, 7, 8, 9, 10,
11,12, 13, 14,15))
);

CREATE TABLE comes_from (
    score_triple_id INTEGER UNSIGNED,
    cook_id INTEGER UNSIGNED,
    score INTEGER UNSIGNED NOT NULL,
    PRIMARY KEY (score_triple_id , cook_id),
    FOREIGN KEY (score_triple_id)
        REFERENCES score (score_triple_id),
    FOREIGN KEY (cook_id)
        REFERENCES cook (cook_id),
    CONSTRAINT score_constraint CHECK (score IN (1 , 2, 3, 4, 5))
);

CREATE TABLE roles (
    role_id INT AUTO_INCREMENT PRIMARY KEY,
    role_name VARCHAR(50) NOT NULL
);

CREATE TABLE users (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    role ENUM('admin', 'cook') NOT NULL
);

CREATE INDEX idx_cook_name_surname ON cook(name, surname);
CREATE INDEX idx_score_cook_id ON score(cook_id);
CREATE INDEX idx_score_recipe_id ON score(recipe_id);
CREATE INDEX idx_recipe_national_cuisine ON recipe(national_cuisine);
CREATE INDEX idx_episode_season ON episode(season);
CREATE INDEX idx_cook_experience ON cook(experience);
CREATE INDEX idx_score_final_score ON score(final_score);
CREATE INDEX idx_participates_in_role ON participates_in(role);
```

Αναφορικά με τα constraints που έχουμε θέσει:

- Θεωρούμε ότι τα περισσότερα attributes εκτός από αυτά που προκύπτουν δυναμικά είναι NOT NULL ούτως ώστε να υπάρχει πληρότητα στα δεδομένα.
- Επιπλέων, ότι κάθε εθνική κουζίνα πρέπει να ακολουθείται από τη λέξη “Cuisine” ώστε να εγγυηθεί η ομοιομορφία και η μη ύπαρξη διπλότυπων.
- Επιπλέον, ο χαρακτηρισμός της επαγγελματικής κατάστασης γίνεται με την εξής αντιστοιχία και μόνο: Α’ μάγειρας αντιστοιχεί σε a-cook, Β’ μάγειρας αντιστοιχεί σε b-cook, Γ’ μάγειρας αντιστοιχεί σε c-cook, αρχιμάγειρας και βοηθός αρχιμάγειρα σε chef και sous-chef, αντίστοιχα.

2.5 DML Scripts (Data Manipulation Language Scripts)

Το DML Script επισυνάπτεται μαζί με την εργασία στο αρχείο “dml.sql” και περιέχει όλα τα απαραίτητα δεδομένα για την διεξαγωγή του διαγωνισμού. Συγκεκριμένα προστέθηκαν:

- 69 συνταγές
- 100 μάγειρες
- 23 ομάδες τροφίμων
- 5 σεζόν με 10 επεισόδια η κάθε μια (50 επεισόδια συνολικά)
- 100 διαφορετικά συστατικά
- 23 ετικέτες
- 25 σκεύη
- 35 ετικέτες
- 35 θεματικές ενότητες.

2.6 Triggers και βοηθητικά Queries

Το πρώτο βοηθητικό query που υλοποιήσαμε πραγματοποιεί την εξής διαδικασία: εντοπίζει το βασικό υλικό (με βάση τη Boolean μεταβλητή) και σε ποια ομάδα τροφίμων ανήκει και με βάση αυτή να προσθέτει το label_id του χαρακτηρισμού στη συνταγή. Το πρώτο βοηθητικό query παρατίθεται παρακάτω:

```
UPDATE recipe r
INNER JOIN (
    SELECT recipe_id, MAX(l.label_id) AS label_id
    FROM consists_of co
    JOIN ingredients i ON co.ingredients_id = i.ingredients_id
    JOIN label l ON l.food_group_id = i.food_group_id
    WHERE co.basic_ingredient = 1
    GROUP BY recipe_id
) t ON r.recipe_id = t.recipe_id
SET r.label_id = t.label_id;
```

Το παρακάτω trigger αφορά την ηλικία των μαγείρων. Κάθε φορά που προστίθεται ένας νέος μάγειρας το trigger προσθέτει στο attribute age την ηλικία που έχει τη χρονιά που διανύουμε. Το trigger είναι το εξής:

```
 DELIMITER //

DROP TRIGGER IF EXISTS calculate_cook_age;
CREATE TRIGGER calculate_cook_age
BEFORE INSERT ON cook
FOR EACH ROW
BEGIN
    -- Handle NULL birth_date
    IF NEW.birth_date IS NULL THEN
        SET NEW.age = NULL; -- Set age to NULL if birth_date is not provided
    ELSE
        SET NEW.age = YEAR(CURDATE()) - YEAR(NEW.birth_date);

        -- Adjust for birthdays that haven't happened yet this year
        IF (MONTH(CURDATE()) < MONTH(NEW.birth_date)) OR
           (MONTH(CURDATE()) = MONTH(NEW.birth_date) AND DAY(CURDATE()) <
DAY(NEW.birth_date)) THEN
            SET NEW.age = NEW.age - 1;
        END IF;
    END IF;
END //

DELIMITER ;

```

Το δεύτερο trigger ελέγχει τον αριθμό των tag που έχουν προστεθεί σε μια συνταγή (εφόσον το όριο είναι 3 tags σε κάθε συνταγή) και αν ξεπεράσει τον δεκτό αριθμό εμφανίζει μήνυμα σφάλματος. Το trigger είναι:

```
DROP TRIGGER IF EXISTS check_tag_limit;
DELIMITER //

CREATE TRIGGER check_tag_limit
BEFORE INSERT ON is_tagged
FOR EACH ROW
BEGIN
  DECLARE tag_count INT;


  SELECT COUNT(*)
  INTO tag_count
  FROM is_tagged
  WHERE recipe_id = NEW.recipe_id;


  IF tag_count >= 3 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Cannot add more than 3 tags to a recipe';
  END IF;
END;
```

```
22. // DELIMITER ;
```

Το τρίτο trigger υπολογίζει δυναμικά τις θερμίδες ανά μερίδα κάθε συνταγή με βάση τις θερμίδες από κάθε υλικό που αποτελείται. Έχουμε ως σύμβαση ότι αν η ποσότητα ενός συστατικού δεν είναι σαφώς ορισμένη (πχ λίγο αλάτι) τότε τα calories του συστατικού αυτού είναι ίσα με μηδέν. Το trigger δίνεται παρακάτω:

```
 1. DELIMITER //
 2.
 3. CREATE TRIGGER calculate_calories_per_serving AFTER INSERT ON consists_of
 4. FOR EACH ROW
 5. BEGIN
 6.     DECLARE total_calories_per_100 INT;
 7.     DECLARE num_portions INT;
 8.     DECLARE ingredient_quantity VARCHAR(255); -- Adjust the size as needed
 9.     DECLARE is_quantity_numeric INT;
10.
11.     -- Get the quantity from the ingredients table
12.     SELECT quantity
13.     INTO ingredient_quantity
14.     FROM ingredients
15.     WHERE ingredients_id = NEW.ingredients_id;
16.
17.     -- Check if quantity is numeric (includes numbers)
18.     SELECT IF(ingredient_quantity REGEXP '^[0-9]+$', 1, 0) INTO is_quantity_numeric;
19.
20.     -- Calculate calories only if quantity is numeric
21.     IF is_quantity_numeric = 1 THEN
22.
23.         -- Calculate the sum of (calories * quantity / 100) for the recipe
24.         SELECT SUM(i.calories * (i.quantity / 100))
25.         INTO total_calories_per_100
26.         FROM ingredients i
27.         JOIN consists_of c ON i.ingredients_id = c.ingredients_id
28.         WHERE c.recipe_id = NEW.recipe_id;
29.
30.         -- Get the number of portions for the recipe
31.         SELECT portions
32.         INTO num_portions
33.         FROM recipe
34.         WHERE recipe_id = NEW.recipe_id;
35.
36.         -- Calculate the calories_per_serving
37.         UPDATE nutritional_info ni
38.         SET ni.calories_per_serving = total_calories_per_100 / num_portions
39.         WHERE ni.recipe_id = NEW.recipe_id;
40.
41.     ELSE -- Quantity is not numeric, set calories_per_serving to 0
42.
43.         UPDATE nutritional_info ni
44.         SET ni.calories_per_serving = 0
45.         WHERE ni.recipe_id = NEW.recipe_id;
46.
47.     END IF;
48.
49. END//
50.
51. DELIMITER ;
52.
```

Το επόμενο trigger ελέγχει την ορθότητα ότι έχει προστεθεί μόνο ένα βασικό υλικό σε μια συνταγή, διαφορετικά εκτυπώνει σφάλμα και είναι:

```
 1. DELIMITER //
 2. DROP TRIGGER IF EXISTS enforce_one_basic_ingredient;
 3. CREATE TRIGGER enforce_one_basic_ingredient
 4. BEFORE INSERT ON consists_of
 5. FOR EACH
 6. BEGIN
 7.     IF NEW.basic_ingredient = TRUE THEN
```

```
 8.         -- Check for existing 'basic' ingredient for the same recipe
 9.         IF EXISTS (SELECT 1 FROM consists_of
10.                    WHERE recipe_id = NEW.recipe_id AND basic_ingredient = TRUE) THEN
11.             SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Error: Only one basic ingredient
allowed per recipe';
12.         END IF;
13.     END IF;
14. END;
15. //
16. DELIMITER ;
```

Το επόμενο trigger ελέγχει ότι τα βήματα μιας συνταγής έχουν προστεθεί σειριακά, αλλιώς εκτυπώνει μήνυμα:

```
 1. DELIMITER $$
 2. DROP TRIGGER IF EXISTS check_steps_order;
 3. CREATE TRIGGER check_steps_order
 4. BEFORE INSERT ON is_made
 5. FOR EACH ROW
 6. BEGIN
 7.     DECLARE last_step_num INT;
 8.     SELECT MAX(steps_num) INTO last_step_num
 9.     FROM is_made
10.     WHERE recipe_id = NEW.recipe_id;
11.
12.     IF last_step_num IS NOT NULL THEN
13.         IF NEW.steps_num != last_step_num + 1 THEN
14.             SIGNAL SQLSTATE '45000'
15.             SET MESSAGE_TEXT = 'Steps not inserted serially for recipe';
16.         END IF;
17.     END IF;
18. END$$
19. DELIMITER ;
```

Τα triggers όσο και το βοηθητικό querry αποθηκεύονται στο αρχείο “triggers_quary.sql”

2.7. Ζητούμενα Queries

Καθένα από τα ζητούμενα Queries αποθηκεύεται σε ένα view, για ευκολότερη πρόσβαση και αποθηκεύουμε τα queries στο αρχείο “queries.sql” που επισυνάπτεται μαζί με την εργασία

Συγκεκριμένα τώρα για το query του ερωτήματος 3.6 δημιουργούμε ένα εναλλακτικό Query Plan. Αρχικά τροποποιούμε το αρχικό query ως εξής:

```
 1. SET profiling = 1;
 2. EXPLAIN SELECT
 3.   T1.tag AS Tag1,
 4.   T2.tag AS Tag2,
 5.   COUNT(*) AS EpisodeCount
 6. FROM
 7.   is_tagged IT1
 8.   JOIN is_tagged IT2 ON IT1.recipe_id = IT2.recipe_id
 9.   JOIN recipe R ON IT1.recipe_id = R.recipe_id
10.   JOIN score S ON R.recipe_id = S.recipe_id
11.   JOIN episode E ON S.episode_id = E.episode_id
12.   JOIN tag T1 ON IT1.tag_id = T1.tag_id
13.   JOIN tag T2 ON IT2.tag_id = T2.tag_id
14. WHERE
15.   IT1.tag_id < IT2.tag_id
16. GROUP BY
17.   T1.tag, T2.tag
18.   ORDER BY
19.   EpisodeCount desc
20. LIMIT 3;
21. SHOW PROFILES;
```

Και λαμβάνουμε τις εξόδους (Εικόνα 2.3 και Εικόνα 2.4)

id	select_...	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	S	NULL	ALL	idx_score_recipe_id	NULL	NULL	NULL	1	100.00	Using where; Using temporary; Using filesort
1	SIMPLE	E	NULL	eq_ref	PRIMARY	PRIMARY	4	cook_show.S.episode_id	1	100.00	Using index
1	SIMPLE	R	NULL	eq_ref	PRIMARY	PRIMARY	4	cook_show.S.recipe_id	1	100.00	Using index
1	SIMPLE	IT1	NULL	ref	PRIMARY,tag_id	PRIMARY	4	cook_show.S.recipe_id	1	100.00	Using index
1	SIMPLE	IT2	NULL	ref	PRIMARY,tag_id	PRIMARY	4	cook_show.S.recipe_id	1	33.33	Using where; Using index
1	SIMPLE	T1	NULL	eq_ref	PRIMARY	PRIMARY	4	cook_show.IT1.tag_id	1	100.00	NULL
1	SIMPLE	T2	NULL	eq_ref	PRIMARY	PRIMARY	4	cook_show.IT2.tag_id	1	100.00	NULL

Εικόνα 2.3

Query_ID	Duration	Query
5876	0.00040400	INSERT INTO meal_type (meal_type_name, photo_url, photo_description)…
5877	0.00058500	INSERT INTO tag (tag_id, tag, photo_url, photo_description) VALUES (1, '…
5878	0.00071200	INSERT INTO nutritional_info (nutritional_info_id, recipe_id, fat_per_servin…
5879	0.00069400	INSERT INTO specialization (cook_id, specialization) VALUES (1, 'Italian…
5880	0.00424400	INSERT INTO is_tagged (recipe_id, tag_id) VALUES (1, 1), (1, 2), (2, 2), (…
5881	0.00151200	INSERT INTO requires (recipe_id, utensil_id) VALUES (1, 1), (2, 2), (3, 3),…
5882	0.00201200	INSERT IGNORE INTO consists_of (recipe_id, ingredients_id, basic_ingre…
5883	0.00285800	INSERT IGNORE INTO recipe_tips (recipe_id, tips) VALUES (1, 'Keep ref…
5884	0.00094800	INSERT INTO is_a(recipe_id, meal_type_name) VALUES (1, 'Main Cours…
5885	0.00196900	INSERT INTO is_made(recipe_id, steps_id, steps_num) VALUES (1, 5, 1),…
5886	0.00081200	INSERT INTO belongs_to (recipe_id, topic_name) VALUES (1, 'Comfort F…
5887	0.00230500	INSERT INTO label (label_id, label_name, food_group_id, photo_url, phot…
5888	0.00018800	SET profiling = 1
5889	0.00006400	SHOW WARNINGS
5890	0.00075900	EXPLAIN SELECT T1.tag AS Tag1, T2.tag AS Tag2, COUNT(*) AS E…

Εικόνα 2.4

Τώρα κάνουμε τις απαιτούμενες αλλαγές ούτως ώστε να χρησιμοποιήσουμε force indexes και έχουμε:

```
1. SET profiling = 1;
2. CREATE INDEX idx_is_tagged_recipe_id_tag_id ON is_tagged (recipe_id, tag_id);
3. CREATE INDEX idx_score_recipe_id_episode_id ON score (recipe_id, episode_id);
4.
5. EXPLAIN SELECT
6.   T1.tag AS Tag1,
7.   T2.tag AS Tag2,
8.   COUNT(*) AS EpisodeCount
9. FROM
10.   is_tagged IT1 FORCE INDEX (idx_is_tagged_recipe_id_tag_id)
11.   JOIN is_tagged IT2 FORCE INDEX (idx_is_tagged_recipe_id_tag_id) ON IT1.recipe_id =
IT2.recipe_id
12.   JOIN recipe R FORCE INDEX (PRIMARY) ON IT1.recipe_id = R.recipe_id
13.   JOIN score S FORCE INDEX (idx_score_recipe_id_episode_id) ON R.recipe_id = S.recipe_id
14.   JOIN episode E FORCE INDEX (PRIMARY) ON S.episode_id = E.episode_id
15.   JOIN tag T1 FORCE INDEX (PRIMARY) ON IT1.tag_id = T1.tag_id
16.   JOIN tag T2 FORCE INDEX (PRIMARY) ON IT2.tag_id = T2.tag_id
17. WHERE
18.   IT1.tag_id < IT2.tag_id
19. GROUP BY
20.   T1.tag, T2.tag
21. ORDER BY
22.   EpisodeCount DESC
23. LIMIT 3;
24.
25. SHOW PROFILES;
```

Λαμβάνουμε τώρα τις εξόδους (Εικόνα 2.5, 2.6):

id	select_...	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	IT1	NULL	index	idx_is_tagged_recipe_id_tag_id	idx_is_tagged_recipe_id_tag_id	8	NULL	1	100.00	Using index; Using temporary; Using filesort
1	SIMPLE	IT2	NULL	ref	idx_is_tagged_recipe_id_tag_id	idx_is_tagged_recipe_id_tag_id	4	cook_show.IT1.recipe_id	1	100.00	Using where; Using index
1	SIMPLE	R	NULL	eq_ref	PRIMARY	PRIMARY	4	cook_show.IT1.recipe_id	1	100.00	Using index
1	SIMPLE	S	NULL	ref	idx_score_recipe_id_episode_id	idx_score_recipe_id_episode_id	5	cook_show.IT1.recipe_id	1	100.00	Using where; Using index
1	SIMPLE	E	NULL	eq_ref	PRIMARY	PRIMARY	4	cook_show.S.episode_id	1	100.00	Using index
1	SIMPLE	T1	NULL	eq_ref	PRIMARY	PRIMARY	4	cook_show.IT1.tag_id	1	100.00	NULL
1	SIMPLE	T2	NULL	eq_ref	PRIMARY	PRIMARY	4	cook_show.IT2.tag_id	1	100.00	NULL

Εικόνα 2.5

Query_ID	Duration	Query
6122	0.00438700	CREATE INDEX idx_score_cook_id ON score(cook_id)
6123	0.00351500	CREATE INDEX idx_score_recipe_id ON score(recipe_id)
6124	0.01150900	CREATE INDEX idx_recipe_national_cuisine ON recipe(national_cuisine)
6125	0.00417000	CREATE INDEX idx_episode_season ON episode(season)
6126	0.00727700	CREATE INDEX idx_cook_experience ON cook(experience)
6127	0.00387400	CREATE INDEX idx_score_final_score ON score(final_score)
6128	0.00213700	CREATE INDEX idx_participates_in_role ON participates_in(role)
6129	0.00015400	SET profiling = 1
6130	0.00006700	SHOW WARNINGS
6131	0.00172400	EXPLAIN SELECT T1.tag AS Tag1, T2.tag AS Tag2, COUNT(*) AS Epi…
6132	0.00021100	SET profiling = 1
6133	0.00011700	SHOW WARNINGS
6134	0.00736500	CREATE INDEX idx_is_tagged_recipe_id_tag_id ON is_tagged (recipe_id,…
6135	0.01401800	CREATE INDEX idx_score_recipe_id_episode_id ON score (recipe_id, epi…
6136	0.00080800	EXPLAIN SELECT T1.tag AS Tag1, T2.tag AS Tag2, COUNT(*) AS Epi…

Εικόνα 2.6

Παρατηρούμε ότι τα αποτελέσματα με τα force indexes είναι χειρότερα, γεγονός που υποδεικνύει ότι η προηγούμενη λύση με τα indexes που είχαμε χρησιμοποιήσει ήταν η βέλτιστη.

Επαναλαμβάνουμε την ίδια διαδικασία και για το query του 3.8. Αρχικά, χωρίς τα force indexes έχουμε τα αποτελέσματα (Εικόνα 2.7 και Εικόνα 2.8):

```
 1. SET profiling = 1;
 2.
 3. EXPLAIN SELECT
 4.     e.episode_id,
 5.     COUNT(r.utensil_id) AS equipment_count
 6. FROM
 7.     episode e
 8. JOIN
 9.     requires r ON e.episode_id = r.recipe_id
10. GROUP BY
11.     e.episode_id
12. ORDER BY
13.     equipment_count DESC
14. LIMIT 1;
15. SHOW PROFILES;
```

id	select_...	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	e	NULL	index	PRIMARY,idx_episode_season	PRIMARY	4	NULL	1	100.00	Using index; Using temporary; Using filesort
1	SIMPLE	r	NULL	ref	PRIMARY	PRIMARY	4	cook_show.e.episode_id	1	100.00	Using index

Εικόνα 2.7

Query_ID	Duration	Query
6198	0.00804900	CREATE TABLE score (score_triple_id INTEGER UNSIGNED AUTO_I…
6199	0.00554900	CREATE TABLE comes_from (score_triple_id INTEGER UNSIGNED,…
6200	0.00197100	CREATE TABLE roles (role_id INT AUTO_INCREMENT PRIMARY KEY…
6201	0.00273000	CREATE TABLE users (user_id INT AUTO_INCREMENT PRIMARY KE…
6202	0.00609700	CREATE INDEX idx_cook_name_surname ON cook(name, surname)
6203	0.00420900	CREATE INDEX idx_score_cook_id ON score(cook_id)
6204	0.00353400	CREATE INDEX idx_score_recipe_id ON score(recipe_id)
6205	0.00882300	CREATE INDEX idx_recipe_national_cuisine ON recipe(national_cuisine)
6206	0.00359400	CREATE INDEX idx_episode_season ON episode(season)
6207	0.00561600	CREATE INDEX idx_cook_experience ON cook(experience)
6208	0.00329400	CREATE INDEX idx_score_final_score ON score(final_score)
6209	0.00221100	CREATE INDEX idx_participates_in_role ON participates_in(role)
6210	0.00021500	SET profiling = 1
6211	0.00011600	SHOW WARNINGS
6212	0.00226200	EXPLAIN SELECT e.episode_id, COUNT(r.utensil_id) AS equipmen…

Εικόνα 2.8

Παρακάτω δίνεται το νέο query καθώς και οι έξοδοι (Εικόνα 2.9 και Εικόνα 2.10)

id	select_...	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	e	NULL	index	PRIMARY	PRIMARY	4	NULL	1	100.00	Using index; Using temporary; Using filesort
1	SIMPLE	r	NULL	ref	idx_recipe_utensil	idx_recipe_utensil	4	cook_show.e.episode_id	1	100.00	Using index

Εικόνα 2.9

Query_ID	Duration	Query
6202	0.00609700	CREATE INDEX idx_cook_name_surname ON cook(name, surname)
6203	0.00420900	CREATE INDEX idx_score_cook_id ON score(cook_id)
6204	0.00353400	CREATE INDEX idx_score_recipe_id ON score(recipe_id)
6205	0.00882300	CREATE INDEX idx_recipe_national_cuisine ON recipe(national_cuisine)
6206	0.00359400	CREATE INDEX idx_episode_season ON episode(season)
6207	0.00561600	CREATE INDEX idx_cook_experience ON cook(experience)
6208	0.00329400	CREATE INDEX idx_score_final_score ON score(final_score)
6209	0.00221100	CREATE INDEX idx_participates_in_role ON participates_in(role)
6210	0.00021500	SET profiling = 1
6211	0.00011600	SHOW WARNINGS
6212	0.00226200	EXPLAIN SELECT e.episode_id, COUNT(r.utensil_id) AS equipmen…
6213	0.00045200	SET profiling = 1
6214	0.00022000	SHOW WARNINGS
6215	0.01477000	CREATE INDEX idx_recipe_utensil ON requires (recipe_id, utensil_id)
6216	0.00080100	EXPLAIN SELECT e.episode_id, COUNT(r.utensil_id) AS equipment…

Εικόνα 2.10

Πράγματι και στη περίπτωση αυτή παρατηρούνται ελαφρώς χειρότερα αποτελέσματα.

Τα queries υλοποιημένα με force index παρατίθενται μαζί με την εργασία στο αρχείο “fi_queries.sql”

3. Οδηγίες εγκατάστασης:

3.1 GitHub

Αρχικά, το repository της βάσης δεδομένων είναι το εξής: https://github.com/christsourvel/DB_PROJECT. Για να λάβετε τοπικά τα δεδομένα που περιέχονται στο repository εκτελείτε την εντολή git clone https://github.com/christsourvel/DB_PROJECT και τα αποθηκεύουμε στην επιθυμητή τοποθεσία.

3.2 Βάση Δεδομένων

Για την εγκατάσταση της βάσης δεδομένων προτείνουμε το MySQLWorkbench, στο οποίο και δουλέψαμε.

3.3 Απαιτούμενες εγκαταστάσεις για κλήρωση και χρήστες

1) Εγκατάσταση MySQL: https://dev.mysql.com/doc/refman/8.3/en/installing.html
2) Εγκατάσταση python: https://www.python.org/downloads/
3) Εγκατάσταση κατάλληλων βιβλιοθηκών python:

Στο τερματικό εκτελείτε:

```
pip install mysql-connector-python
```

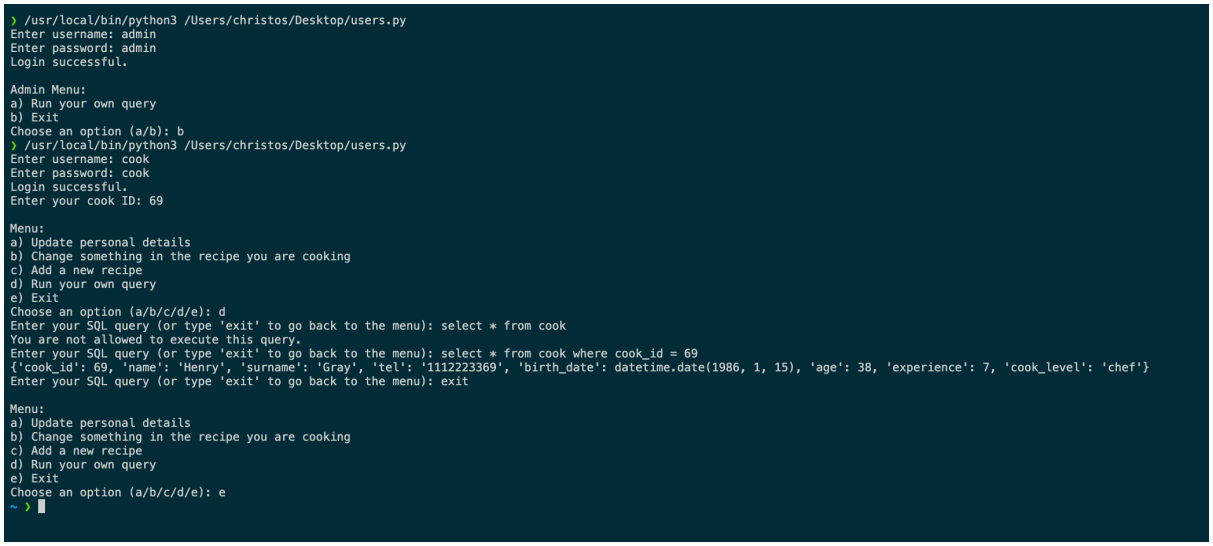
`mysql-connector-python`: Χρησιμοποιείται για τη σύνδεση και την αλληλεπίδραση με την βάση δεδομένων MySQL από την Python.
Επιπλέον χρησιμοποιείται η βιβλιοθήκη ***random*** η οποία εγκαθίσταται αυτόματα με την εγκατάσταση της python.

3.2 Κλήρωση

Η κλήρωση της εφαρμογής πραγματοποιείται μέσω του ***draw.py*** με δυναμικό τρόπο, κληρώνοντας κάθε φορά συνολικά 50 επεισόδια, 10 για κάθε σεζόν σε διάστημα 5 χρόνων. Ο αριθμός των επεισοδίων ανά σεζόν και τα χρόνια που τρέχει ο διαγωνισμός επιλέγονται από τον διαχειριστή. Η κλήρωση λαμβάνει υπόψιν πιθανόν περιορισμούς, όπως αναγράφονται στην εκφώνηση της εργασίας.

3.3 Χρήστες

Οι αυθεντικοποίηση των χρηστών και η είσοδός τους στο σύστημα της βάσης δεδομένων πραγματοποιείται μέσω του ***users.py*** το οποίο δέχεται τον χρήστη *admin* και τον χρήστη *cook (guest)*. Μετά την σύνδεση εμφανίζεται διαδραστικό μενού το οποίο δίνει στους μάγειρες τη δυνατότητα να επεξεργαστούν όλα τα στοιχεία των συνταγών που τους έχουν ανατεθεί και επίσης να προσθέσουν νέα συνταγή. Επίσης μπορούν να επεξεργαστούν τα προσωπικά τους στοιχεία. Δεν μπορούν να τροποποιήσουν άλλα στοιχεία του συστήματος πχ συνταγές που δεν έχουν ανατεθεί σε αυτούς, και σε περίπτωση που το επιχειρήσουν το σύστημα δεν τους το επιτρέπει. Αντίθετα, ο admin έχει πλήρη πρόσβαση στην βάση, καταχωρίζει και τροποποιεί όλα τα απαιτούμενα στοιχεία. Μπορεί να δημιουργήσει αντίγραφο ασφαλείας για όλη τη βάση (backup) και να επαναφέρει το σύστημα από αυτό (restore) (Εικόνα 3.1).


❯ /usr/local/bin/python3 /Users/christos/Desktop/users.py
Enter username: admin
Enter password: admin
Login successful.
Admin Menu:
a) Run your own query
b) Exit
Choose an option (a/b): b
❯ /usr/local/bin/python3 /Users/christos/Desktop/users.py
Enter username: cook
Enter password: cook
Login successful.
Enter your cook ID: 69
Menu:
a) Update personal details
b) Change something in the recipe you are cooking
c) Add a new recipe
d) Run your own query
e) Exit
Choose an option (a/b/c/d/e): d
Enter your SQL query (or type 'exit' to go back to the menu): select * from cook
You are not allowed to execute this query.
Enter your SQL query (or type 'exit' to go back to the menu): select * from cook where cook_id = 69
{'cook_id': 69, 'name': 'Henry', 'surname': 'Gray', 'tel': '1112223369', 'birth_date': datetime.date(1986, 1, 15), 'age': 38, 'experience': 7, 'cook_level': 'chef'}
Enter your SQL query (or type 'exit' to go back to the menu): exit
Menu:
a) Update personal details
b) Change something in the recipe you are cooking
c) Add a new recipe
d) Run your own query
e) Exit
Choose an option (a/b/c/d/e): e
~ ❯

Εικόνα 3.1