Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών ΕΜΠ

ΕΡΓΑΣΤΗΡΙΟ ΣΥΣΤΗΜΑΤΩΝ ΒΑΣΕΩΝ ΓΝΩΣΕΩΝ ΚΑΙ ΔΕΔΟΜΕΝΩΝ

Εξαμηνιαία εργασία στο μάθημα των βάσεων δεδομένων για το εαρινό εξάμηνο 2023-2024

Ομάδα : Project 29

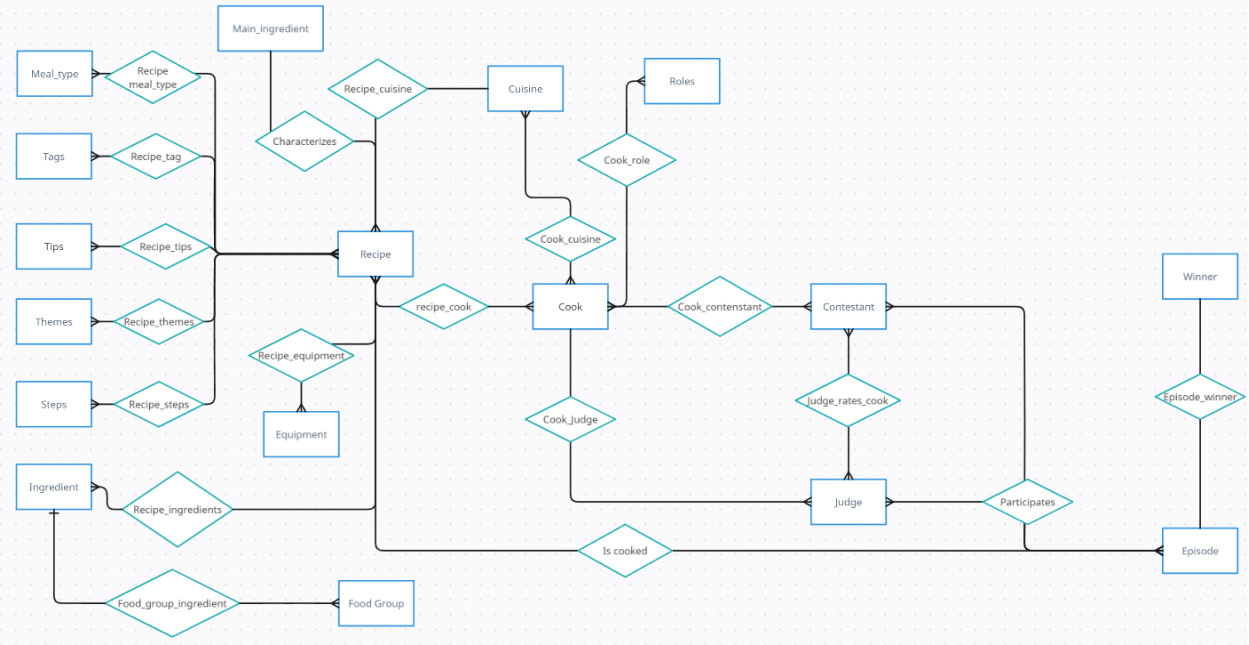
https://github.com/feezz8/DataBases_2024

Ονοματεπώνυμο : Φέζος Κωνσταντίνος / Α.Μ : 03118076
Ονοματεπώνυμο : Τσουκνίδας Οδυσσέας-Αρθούρος-Ρήγας / Α.Μ : 03120043
Ονοματεπώνυμο : Γιαννακόπουλος Γιώργος / Α.Μ : 03120828

1 Βάση δεδομένων

1.1 Διάγραμμα Οντοτήτων Συσχετίσεων(Entity – Relationship Diagram)

Ξεκινήσαμε την υλοποίηση της βάσης σχεδιάζοντας το διάγραμμα οντοτήτων-συσχετίσεων όπως φαίνεται στην εικόνα 1.



Εικόνα 1 : Entity – Relationship Diagram

1.2 Σχεσιακό σχήμα (Relational Schema)

Χρησιμοποιώντας το παραπάνω μοντέλο σχεδιάσαμε το σχεσιακό σχήμα της βάσης. Το σχήμα σχεδιάστηκε έτσι ώστε η βάση να βρίσκεται σε 3^{η} κανονική μορφή. Κάθε σύνολο attributes εξαρτάται μονάχα από το υπερκλειδί , το ολόκληρο υπερκλειδί και τίποτα άλλο πέρα από το υπερκλειδί. Το σχήμα φαίνεται παρακάτω στην εικόνα 2.

Εικόνα 2 : Relational Schema

Τα κυρίοτερα σημεία της βάσης μας είναι τα entities:

- Recipe : Περιλαμβάνει όλες τις συνταγές με τα χαρακτηριστικά που ζητούνται για την κάθε μία. Ορίζουμε tables που συνδέονται με το recipe table χρησιμοποιώντας το πρόθεμα ‘ recipe_ ‘ για να υποδηλώσουμε την σύνδεση μεταξύ τους και foreign keys για να δηλώσουμε την έξαρτηση.
- Cook : Περιλαμβάνει τους μάγειρες του διαγωνισμού μαζί με τα χαρακτηριστικά τους. Μέσα από την υλοποίηση της βάσης οποιοδήποτε στοιχείο του cook μπορεί να γίνει κριτής σε επεισόδιο ή διαγωνιζόμενος.

Για την ορθή λειτουργια της βάσης μας έχουμε κάνει κάποιες παραδοχές.

1. Για να γίνεται δυναμικά ο υπολογισμός θερμίδων έχουμε αποφασίσει να λαμβάνουμε υπ’ όψιν μονάχα τα υλικά με ρητή ποσότητα ml η γραμμαρίων. Για παράδειγμα ‘1 αυγο’ = Δεν γνωρίζουμε αριθμό θερμίδων (NULL), ενώ 120 ml γαλα = 90 θερμίδες.
2. Κάθε υλικό της βάσης ανήκει σε μία μονάχα ομάδα τροφίμων.
3. Το τελευταίο επεισόδιο μίας χρονίας με το πρώτο της επόμενης θεωρούνται συνεχόμενα.
4. Ο περιορισμός στην εμφάνιση ενός μάγειρα σε παραπάνω από 3 επεισόδια συνεχόμενα, αναφέρεται αυστηρά στην εμφάνιση του σε συγκεκριμένο ρόλο

(διαγωνιζόμενος μάγειρας ή κριτής). Συνεπώς ένας μάγειρας μπορεί να συμμετάσχει σε παραπάνω από 3 συνεχόμενα επεισόδια αρκεί να μην υπάρξουν πάνω από 3 συνεχόμενα επεισόδια στα οποία να έχει τον ίδιο ρόλο.

5. Οι κριτές δεν απαιτείται να έχουν κάποια εξειδίκευση στις κουζίνες που θα κρίνουν, ούτε όμως τις μαθαίνουν μετά το επεισόδιο.
6. Για την εξειδίκευση σε κουζίνα επιλέξαμε το εξής. Κάθε μάγειρας που γνωρίζει τουλάχιστον μία συνταγή από μία κουζίνα εχει εξειδίκευση και στην συγκεκριμένη κουζίνα. Ωστόσο αυτό δεν σημαίνει ότι δεν μπορεί να μάθει μία συνταγή από κουζίνα στην οποία δεν έχει ήδη εξειδίκευση. Σε περίπτωση που γίνει αυτό θα αποκτήσει και εξειδίκευση στην κουζίνα αυτή. Θα δούμε περισσότερα για αυτό στο τομέα των triggers και procedures.

1.3 Ευρετήρια (Indices)

Στη συνέχεια , ορίσαμε στη βάση μας ορισμένα ευρετήρια τυπου B+ Δέντρου, έτσι ώστε να βλετιστοποιήσουμε την απόδοση στην απάντηση των ερωτημάτων που μας δόθηκε. Γνωρίζουμε ήδη ότι η Βάση Δεδομένων δημιουργεί αυτόματα ευρετήρια οποτεδήποτε επιβληθεί ο περιορισμός UNIQUE. Τέλος να σημειωθεί ότι τα ευρετήρια αυτά είναι προσεγγιστικά και ίσως συγκυριακά για τον σκοπό του συγκεκριμένου πρότζεκτ. Σε μια βάση δεδομένων είναι σημαντικό να επαναξιολογούμε τα απαιτούμενα ευρετήρια βάσει της συχνότητας εμφάνισης του εκάστοτε ερωτήματος. Πιο απλά, ενδέχεται το κόστος ενημέρωσης του ευρετηριού να ξεπερνά το κέρδος στην απάντηση του ερωτήματος.

Στη βάση μας ορίσαμε τα ακόλουθα ευρετήρια στα:

- Recipe_id και equipment_id στο table recipe_equipment (Για την απάντηση του 8^{ου} ερωτήματος με force index.
- Recipe_id στο table step(Για το φιλτράρισμα των βημάτων των συνταγών)
- First και Last name στο table cook (Για το φιλτράρισμα των μαγειριών βάσει ονόματος).
- Recipe_id στο table episode_selection (Για την απάντηση του 8^{ου} ερωτήματος με force index).
- Rating στο table jusge_rates_cook(Για το φιλτράρισμα των βαθμολογιών ανα τα επεισόδια).
- Recipe_id και tag_id στο table recipe_tag(Για την απάντηση του 6^{ου} ερωτήματος

Γενικότερα η υλοποίηση της βάσης μας χρησιμοποιεί αρκετούς περιορίσμους primary key επομένως δεν είναι άμεση η ανάγκη για χρήση πολλών ευρετηρίων.

1.4 Triggers και Procedures

Για να ικανοποιήσουμε όλους τους περιορισμούς που μας ζητήθηκαν χρησιμοποιήσαμε ορισμένα triggers και procedures. Αρχικά υλοποιούμε την επιλογή 10 εθνικών κουζίνων, 10 μαγείρων και 3 κριτών ανά επεισόδιο με 2 πίνακες, τον episode_selection, που κάνει την σύνδεση μεταξύ επεισοδίου, μάγειρα και κουζίνας, και τον episode_judge που κάνει την σύνδεση μεταξύ επεισοδίου και μάγειρα-κριτή. Τέλος, ο πίνακας judge_rates_cook, συνδέει επεισόδιο, μάγειρα και κριτή, με πρόσθετη πληροφορία την βαθμολογία που δίνει ο κριτής στον μάγειρα.

Τα triggers χρησιμοποιούνται για τήρηση των business rules της βάσης μας, δεν αφήνουν δηλαδή tuples που δεν είναι σύμφωνα με τους κανόνες του διαγωνισμού να εισέλθουν στην βάση, ή επιτελούν άλλες ενέργειες ώστε η βάση μας να είναι εννοιολογικά σωστή και πλήρης. Συγκεκριμένα:

tr_recipe_difficulty: Φροντίζει το επίπεδο δυσκολίας των συνταγών να είναι μεταξύ του 1 και του 5.

tr_cal_portion_calculate: Αφού γίνει προσθήκη στον πίνακα recipe_ingredient (που συμβολίζει την χρήση ενός υλικού από μία συνταγή), υπολογίζει τις επιπλέον θερμίδες ανά μερίδα που προσθέτει το υλικό στην συνταγή και ενημερώνει κατάλληλα το αντίστοιχο attribute στον πίνακα recipe.

tr_recipe_ingredient_not_null: Φροντίζει οι ποσότητες των υλικών να έχουν μία από τις αποδεκτές μορφές.

tr_recipe_ingredient_one_main: Φροντίζει κάθε συνταγή να έχει ένα μόνο βασικό υλικό.

tr_cook_age_and_experience: Υπολογίζει δυναμικά την ηλικία ενός μάγειρα, βάση της ημερομηνίας γέννησης του, και φροντίζει να μην δεχθεί μάγειρα με περισσότερα χρόνια επαγγελματικής εμπειρίας από την ηλικία του.

tr_cook_role: Φροντίζει η επαγγελματική κατάρτιση των μαγείρων να είναι μία από τις ορισμένες από τον διαγωνισμό.

tr_episode: Φροντίζει να μην μπαίνουν πάνω από 10 επεισόδια σε κάθε σεζόν, αλλά και να μην μπαίνει το ίδιο επεισόδιο πάνω από μία φορά.

tr_tip_up_to_3: Φροντίζει να μην έχουμε πάνω από 3 χρηστικές συμβουλές ανά συνταγή.

tr_recipe_cook_learns_cuisine: Σύμφωνα με τις παραδοχές μας, αφού ένας μάγειρας μάθει μία συνταγή, φροντίζει να του δώσει (εάν δεν την έχει ήδη) την εξειδίκευση την εθνικής κουζίνας στην οποία ανήκει η συνταγή.

tr_episode_selection:

1. Φροντίζει να έχουμε μέχρι 10 επιλογές ανά επεισόδιο.
2. Φροντίζει να μην επιλέξουμε τον ίδιο μάγειρα/εθνική κουζίνα/συνταγή πάνω από 1 φορά ανά επεισόδιο.
3. Αν ο μάγειρας δεν γνωρίζει την συνταγή που του ανατίθεται, του την αναθέτει.
4. Φροντίζει να μην συμμετέχουν ένας μάγειρας, εθνική κουζίνα ή συνταγή σε πάνω από 3 συνεχόμενα επεισόδια.

tr_episode_judge_sel:

1. Φροντίζει να μην έχουμε πάνω από 3 κριτές ανά επεισόδιο
2. Φροντίζει να μην επιλεγεί ως κριτής ένας μάγειρας που διαγωνίζεται στο ίδιο επεισόδιο.
3. Φροντίζει να μην συμμετάσχει ένας κριτής σε πάνω από 3 συνεχόμενα επεισόδια.

tr_judge_rates_cook:

1. Φροντίζει η βαθμολογία των κριτών προς τους μάγειρες να είναι μεταξύ 1 και 5.
2. Φροντίζει ο κριτής και ο μάγειρας να είναι όντως κριτές και μάγειρες στο συγκεκριμένο επεισόδιο

1.5 Mock Data και Testing

Για να βεβαιωθούμε για την ορθή λειτρουγία του σχήματος χρησιμοποιήσαμε την βιβλιοθήκη faker της python για να γεμίσουμε τη βάση με δεδομένα. Ενδεικτικά στη βάση εισάγαμε 100 συνταγές, 299 υλικά, 200 μάγειρες, 100 tags, 40 tips, 40 θεματικές ενότητες, 21 ομάδες τροφίμων, 20 διαφορετικά μαγειρικά σκεύη/εξοπλισούς, 50 εθνικές κουζίνες και 60 επεισόδια.

Περισσότερα για την δημιουργία των Mock data θα αναφερθούν παρακάτω.

1.6 Ρόλοι χρηστών βάσης δεδομένων.

Καθώς η υλοποίηση της εργασίας απαιτεί μονάχα την δημιουργία της βάσης μαζί με τα απαραίτητα δεδομένα η εξουσιοδότηση των πιθανών χρηστών παίρνει πιο θεωρητικό ρόλο. Στη βάση έχουμε δημιουργήσει 1 table για όλους τους χρήστες με attributes το username και password και 2 ακόμα, ένα για τους χρήστες μάγειρες και ένα για τον διαχειριστή της βάσης δεδομένων. Σε πραγματική υλοποίησης εφαρμογής θα ορίζαμε δυο επίπεδα ένα στο API και ένα στο Web APP έτσι ώστε να επιβεβαιώναμε ότι ο κάθε χρήστης θα είχε τα κατάλληλα δικαιώματα στη βάση. Πίο συγκεκριμένα οι μάγειρες θα είχαν πρόσβαση στην επεξεργασία μονάχα δεδομένων που σχετίζονται με αυτούς (προσωπικές πληροφορίες, συνταγές που γνωρίζουν), ενώ ο διαχειριστής θα είχε τις δυνατότητες του root user όπως εμείς που υλοποιούμε την βάση.

2 DDL SCRIPT, TRIGGERS AND PROCEDURES

2.1 DDL

Παρακάτω ακολουθεί το ddl script που χρησιμοποιήθηκε για την δημιουργία της βάσης. Το αρχείο βρίσκεται επίσης στο ακόλουθο github link :
https://github.com/feezz8/DataBases_2024
στην τοποθεσία /SQL_foundations/cooking_show_schema.sql

```
DROP TABLE IF EXISTS recipe_meal_type;
DROP TABLE IF EXISTS recipe_tag;
DROP TABLE IF EXISTS recipe_equipment;
DROP TABLE IF EXISTS recipe_tip;
DROP TABLE IF EXISTS recipe_ingredient;
DROP TABLE IF EXISTS recipe_cook;
DROP TABLE IF EXISTS cook_cuisine;
DROP TABLE IF EXISTS episode_selection;
DROP TABLE IF EXISTS episode_judge;
DROP TABLE IF EXISTS judge_rates_cook;
DROP TABLE IF EXISTS cook;
```

```
DROP TABLE IF EXISTS recipe_theme;
DROP TABLE IF EXISTS step;
DROP TABLE IF EXISTS recipe;
DROP TABLE IF EXISTS cuisine;
DROP TABLE IF EXISTS meal_type;
DROP TABLE IF EXISTS tag;
DROP TABLE IF EXISTS tip;
DROP TABLE IF EXISTS equipment;
DROP TABLE IF EXISTS ingredient;
DROP TABLE IF EXISTS food_group;
DROP TABLE IF EXISTS theme;
DROP TABLE IF EXISTS cook_role;
DROP TABLE IF EXISTS episode;
DROP TABLE IF EXISTS user_cook;
DROP TABLE IF EXISTS user_admin;
DROP TABLE IF EXISTS database_user;



CREATE TABLE cuisine(
    cuisine_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    title VARCHAR(255) NOT NULL,
    PRIMARY KEY (cuisine_id)
);

CREATE TABLE recipe(
    recipe_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    baking BOOLEAN NOT NULL,
    cuisine_id INT UNSIGNED NOT NULL,
    difficulty TINYINT NOT NULL,
    title VARCHAR(255) NOT NULL,
    recipe_description TEXT NOT NULL,
    prep_time INT NOT NULL,
    cook_time INT NOT NULL,
    portions SMALLINT NOT NULL,
    fat_portion SMALLINT NOT NULL,
```

```
        protein_portion SMALLINT NOT NULL,
        carbo_portion SMALLINT NOT NULL,
        cal_portion FLOAT NOT NULL DEFAULT 0,
        picture VARCHAR(255),
        picture_description TEXT,
        PRIMARY KEY(recipe_id),
        CONSTRAINT fk_recipecuisine FOREIGN KEY(cuisine_id) REFERENCES
cuisine(cuisine_id)
    );


    CREATE TABLE meal_type(
        meal_type_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
        title VARCHAR(255) NOT NULL,
        PRIMARY KEY (meal_type_id)
    );

    CREATE TABLE recipe_meal_type(
        recipe_id INT UNSIGNED NOT NULL,
        meal_type_id INT UNSIGNED NOT NULL,
        PRIMARY KEY (recipe_id, meal_type_id),
        CONSTRAINT fk_recipe_meal_type_recipe FOREIGN KEY(recipe_id)
REFERENCES recipe(recipe_id) ON UPDATE CASCADE,
        CONSTRAINT fk_recipe_meal_type_meal_type FOREIGN
KEY(meal_type_id) REFERENCES meal_type(meal_type_id) ON UPDATE CASCADE
    );

    CREATE TABLE tag(
        tag_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
        title VARCHAR(255) NOT NULL,
        PRIMARY KEY (tag_id)
    );

    CREATE TABLE recipe_tag(
        recipe_id INT UNSIGNED NOT NULL,
        tag_id INT UNSIGNED NOT NULL,
```

```
        PRIMARY KEY (recipe_id, tag_id),    CONSTRAINT
fk_recipe_tag_recipe FOREIGN KEY (recipe_id) REFERENCES
recipe(recipe_id) ON UPDATE CASCADE,
        CONSTRAINT fk_recipe_tag_tag FOREIGN KEY (tag_id) REFERENCES
tag(tag_id) ON UPDATE CASCADE
    );

    CREATE INDEX idx_recipe_tag_recipe_id ON recipe_tag(recipe_id);
    CREATE INDEX idx_recipe_tag_tag_id ON recipe_tag(tag_id);

    CREATE TABLE tip(
        tip_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
        title VARCHAR(255) NOT NULL,
        PRIMARY KEY (tip_id)
    );

    CREATE TABLE recipe_tip(
        recipe_id INT UNSIGNED NOT NULL,
        tip_id INT UNSIGNED NOT NULL,
        PRIMARY KEY (recipe_id, tip_id),
        CONSTRAINT fk_recipe_tip_recipe FOREIGN KEY (recipe_id)
REFERENCES recipe(recipe_id) ON UPDATE CASCADE,
        CONSTRAINT fk_recipe_tip_tip FOREIGN KEY (tip_id) REFERENCES
tip (tip_id) ON UPDATE CASCADE
    );


    CREATE TABLE equipment(
        equipment_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
        title VARCHAR(255) NOT NULL,
        details TEXT NOT NULL,
        picture VARCHAR(255),
        picture_description TEXT,
        PRIMARY KEY(equipment_id)
    );
```

```
    CREATE TABLE recipe_equipment(
        recipe_id INT UNSIGNED NOT NULL,
        equipment_id INT UNSIGNED NOT NULL,
        num SMALLINT NOT NULL,
        PRIMARY KEY (recipe_id, equipment_id),
        CONSTRAINT fk_recipe_equipment_recipe FOREIGN KEY (recipe_id)
REFERENCES recipe(recipe_id) ON UPDATE CASCADE,
        CONSTRAINT fk_recipe_equipment_equipment FOREIGN KEY
(equipment_id) REFERENCES equipment(equipment_id) ON UPDATE CASCADE
    );

    CREATE INDEX idx_recipe_equipment_recipe_id ON
recipe_equipment(recipe_id);
    CREATE INDEX idx_recipe_equipment_equipment_id ON
recipe_equipment(equipment_id);

    CREATE TABLE step(
        step_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
        recipe_id INT UNSIGNED NOT NULL,
        step_order SMALLINT NOT NULL,
        details TEXT NOT NULL,
        PRIMARY KEY (step_id),
        CONSTRAINT fk_step_recipe FOREIGN KEY (recipe_id) REFERENCES
recipe (recipe_id) ON UPDATE CASCADE
    );

    CREATE INDEX idx_step_recipe_id ON step(recipe_id);

    CREATE TABLE food_group(
        food_group_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
        title VARCHAR(255) NOT NULL,
        food_group_description TEXT NOT NULL,
        char_if_main TEXT NOT NULL,
        picture VARCHAR(255),
        picture_description TEXT,
        PRIMARY KEY(food_group_id)
```

```
    );


    CREATE TABLE ingredient(
        ingredient_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
        title VARCHAR(255) NOT NULL,
        cal_gr INT NOT NULL,
        cal_ml INT NOT NULL,
        food_group_id INT UNSIGNED NOT NULL,
        picture VARCHAR(255),
        picture_description TEXT,
        PRIMARY KEY (ingredient_id),
        CONSTRAINT fk_ingredient_food_group FOREIGN KEY
(food_group_id) REFERENCES food_group (food_group_id) ON UPDATE
CASCADE
    );

    CREATE TABLE recipe_ingredient(
        recipe_id INT UNSIGNED NOT NULL,
        ingredient_id INT UNSIGNED NOT NULL,
        quantity INT,
        recipe_ingredient_description VARCHAR(255),
        main BOOLEAN NOT NULL,
        PRIMARY KEY (recipe_id, ingredient_id),
        CONSTRAINT fk_recipe_ingredient_recipe FOREIGN KEY (recipe_id)
REFERENCES recipe(recipe_id) ON UPDATE CASCADE,
        CONSTRAINT fk_recipe_ingredient_ingredient FOREIGN KEY
(ingredient_id) REFERENCES ingredient (ingredient_id) ON UPDATE
CASCADE
    );



    CREATE TABLE theme(
        theme_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
        title VARCHAR(255) NOT NULL,
```

```
        theme_description TEXT NOT NULL,
        picture VARCHAR(255),
        picture_description TEXT,
        PRIMARY KEY (theme_id)
    );

    CREATE TABLE recipe_theme(
        recipe_id INT UNSIGNED NOT NULL,
        theme_id INT UNSIGNED NOT NULL,
        PRIMARY KEY (recipe_id, theme_id),
        CONSTRAINT fk_recipe_theme_recipe FOREIGN KEY (recipe_id)
REFERENCES recipe (recipe_id) ON UPDATE CASCADE,
        CONSTRAINT fk_recipe_theme_theme FOREIGN KEY (theme_id)
REFERENCES theme (theme_id) ON UPDATE CASCADE
    );

    CREATE TABLE cook_role(
        cook_role_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
        title VARCHAR(255),
        PRIMARY KEY (cook_role_id)
    );


    CREATE TABLE cook(
        cook_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
        first_name VARCHAR(255) NOT NULL,
        last_name VARCHAR(255) NOT NULL,
        phone VARCHAR (10) NOT NULL,
        d_birth DATE NOT NULL,
        age SMALLINT,
        exp_years SMALLINT NOT NULL,
        cook_role_id INT UNSIGNED NOT NULL,
        picture VARCHAR(255),
        picture_description TEXT,
        PRIMARY KEY (cook_id),
```

```
        CONSTRAINT fk_cook_cook_role FOREIGN KEY (cook_role_id)
REFERENCES cook_role (cook_role_id) ON UPDATE CASCADE
    );

    CREATE INDEX idx_first_name ON cook(first_name);
    CREATE INDEX idx_last_name ON cook(last_name);

    CREATE TABLE recipe_cook(
        recipe_id INT UNSIGNED NOT NULL,
        cook_id INT UNSIGNED NOT NULL,
        PRIMARY KEY (recipe_id, cook_id),
        CONSTRAINT fk_recipe_cook_recipe FOREIGN KEY (recipe_id)
REFERENCES recipe (recipe_id) ON UPDATE CASCADE,
        CONSTRAINT fk_recipe_cook_cook FOREIGN KEY (cook_id)
REFERENCES cook (cook_id) ON UPDATE CASCADE
    );

    CREATE TABLE cook_cuisine(
        cook_id INT UNSIGNED NOT NULL,
        cuisine_id INT UNSIGNED NOT NULL,
        PRIMARY KEY (cook_id, cuisine_id),
        CONSTRAINT fk_cook_cuisine_cook FOREIGN KEY (cook_id)
REFERENCES cook (cook_id) ON UPDATE CASCADE,
        CONSTRAINT fk_cook_cuisine_cuisine FOREIGN KEY (cuisine_id)
REFERENCES cuisine (cuisine_id) ON UPDATE CASCADE
    );

    CREATE TABLE episode (
        episode_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
        season CHAR(4) NOT NULL,
        ep_num TINYINT NOT NULL,
        picture VARCHAR(255),
        picture_description TEXT,
        PRIMARY KEY (episode_id)
    );
```

```
    CREATE TABLE episode_selection(
        episode_id INT UNSIGNED NOT NULL,
        cook_id INT UNSIGNED NOT NULL,
        cuisine_id INT UNSIGNED,
        recipe_id INT UNSIGNED NOT NULL,
        PRIMARY KEY (episode_id, cook_id, cuisine_id, recipe_id),
        CONSTRAINT fk_episode_selection_episode FOREIGN KEY
(episode_id) REFERENCES episode (episode_id) ON UPDATE CASCADE,
        CONSTRAINT fk_episode_selection_cook FOREIGN KEY (cook_id)
REFERENCES cook (cook_id) ON UPDATE CASCADE,
        CONSTRAINT fk_episode_selection_cuisine FOREIGN KEY
(cuisine_id) REFERENCES cuisine (cuisine_id) ON UPDATE CASCADE,
        CONSTRAINT fk_episode_selection_recipe FOREIGN KEY (recipe_id)
REFERENCES recipe (recipe_id) ON UPDATE CASCADE
    );

    CREATE INDEX idx_episode_selection_recipe_id ON
episode_selection(recipe_id);

    CREATE TABLE episode_judge(
        episode_id INT UNSIGNED NOT NULL,
        cook_id INT UNSIGNED NOT NULL,
        PRIMARY KEY (episode_id, cook_id),
        CONSTRAINT fk_episode_judge_episode FOREIGN KEY (episode_id)
REFERENCES episode (episode_id) ON UPDATE CASCADE,
        CONSTRAINT fk_episode_judge_cook FOREIGN KEY (cook_id)
REFERENCES cook (cook_id) ON UPDATE CASCADE
    );

    CREATE TABLE judge_rates_cook(
        episode_id INT UNSIGNED NOT NULL,
        judge_id INT UNSIGNED NOT NULL,
        cook_id INT UNSIGNED NOT NULL,
        rating TINYINT NOT NULL,
        PRIMARY KEY (episode_id, judge_id, cook_id),
```

```
        CONSTRAINT fk_judge_rates_cook_episode FOREIGN KEY
(episode_id) REFERENCES episode (episode_id) ON UPDATE CASCADE,
        CONSTRAINT fk_judge_rates_cook_judge FOREIGN KEY (judge_id)
REFERENCES cook (cook_id) ON UPDATE CASCADE,
        CONSTRAINT fk_judge_rates_cook_cook FOREIGN KEY (cook_id)
REFERENCES cook (cook_id) ON UPDATE CASCADE
    );

    CREATE INDEX idx_judge_rates_cook_rating ON
rating(judge_rates_cook);

    CREATE TABLE database_user(
        database_user_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
        username VARCHAR(30) NOT NULL UNIQUE,
        user_password VARCHAR(50) NOT NULL UNIQUE,
        PRIMARY KEY (database_user_id)
    );

    CREATE TABLE user_cook(
        database_user_id INT UNSIGNED NOT NULL UNIQUE,
        CONSTRAINT fk_user_cook FOREIGN KEY (database_user_id)
REFERENCES database_user (database_user_id) ON DELETE CASCADE
    );

    CREATE TABLE user_admin(
        database_user_id INT UNSIGNED NOT NULL UNIQUE,
        CONSTRAINT fk_user_admin FOREIGN KEY (database_user_id)
REFERENCES database_user (database_user_id) ON DELETE CASCADE
    );
```

2.2 Triggers και Procedures

Για να αποφύγουμε την δημιουργία μίας τεράστιας δυσανάγνωστης αναφράς δε θα συμπεριλάβουμε τον πηγιαίο κώδικα για τα triggers και τα procedures. Ωστόσο στο .zip file που βρίσκεται αυτή η αναφορά θα υπάρχει το αρχείο

triggers_procedures.sql το οποίο θα περιλαμβάνει ολόκληρο των κώδικα. Ομοίως και για το DDL. Το αρχείο επίσης βρίσκεται στο github link : https://github.com/feezz8/DataBases_2024 στο /SQL_foundations/Triggers_procedures.sql

2.3 DML και Python script

Ομοίως με παραπάνω το DML βρίσκεται στο .zip file και στο github https://github.com/feezz8/DataBases_2024 στην τοποθεσία : /fake_data/fake_data.sql

Τα ίδια ισχύουν και για το python script που δημιούργησε το DML μαζί με τα απαραίτητα .txt files που περιέχουν πληροφορίες για τον κόσμο της μαγειρικής (recipes, tips, ingredients etc.).

Για να παράξει το DML το πρόγραμμα σε python οφείλουμε να αποθήκευσουμε στο ίδιο directory τον φάκελο με όνομα Fake_data_info και το fake_data.py Και τα δύο αυτά components βρισκονται στο github link στο : /fake_data. Ο φάκελος Fake_data_info περιλαμβάνει τα .txt files που χρησιμοποιήσαμε.

Τέλος για να ολοκήρώσουμε την διαδικασία κλήρωσης επεισοδίου και βαθμολόγησης χρησιμοποιούμε τα procedures που ορίσαμε παραπάνω. Πιο συγκεκριμένα τα insert_episode_selection, insert_episode_judge και insert_judge_rates_cook

3 Ερωτήσεις επάνω στη βάση

Στο .zip file συμπεριλαμβάνουμε επίσης ένα αρχείο queries.sql το οποίο περιέχει τον πηγιαίο κώδικα σε sql για την απάντηση των ερωτημάτων. Να σημειωθεί ότι το αρχείο απλά περιέχει τις απαντήσεις ξεχωριστά. Δεν αποτελεί script. Για να δωθεί σωστή απάντηση σε κάθε ερώτημα πρέπει κάθε query να εκτελεστεί μεμονωμένα.

Όσον αφορά τα ερωτήματα 3.6 και 3.8 παρατηρούμε πράγματι ότι με την χρήση των indeces που ορίσαμε παραπάνω η εκτέλεση των αντίστοιχων ερωτημάτων παρουσιάζει βελτίωση στο χρόνο εκτέλεσης.

4 Τεχνικές απαιτήσεις και βιβλιοθήκες που χρησιμοποιήσαμε

- Η υλοποίηση της βάσης έγινε σε MYSQL 8.0.
- Το script γράφθηκε σε Python 3 με χρήση της βιβλιοθήκης Faker.
- Για την υλοποίηση της βάσης όπως θα είναι στημένη στην εξέταση απαιτείται η εξής διαδικασία :
    1. Εκτέλεση του DDL script (Είτε μέσω κάποιου GUI είτε με cmd line.
    2. Εκτέλεση του script triggers_procedures.sql (Παρομοίως).
    3. Εκτέλεση του python script fake_data.py στο ίδιο directory με τον φάκελο Fake_data_info.
    4. Εκτέλεση του DML script που παράχθηκε με όνομα fake_data.sql.