ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

ΑΝΑΦΟΡΑ ΕΞΑΜΗΝΙΑΙΑΣ ΕΡΓΑΣΙΑΣ

Εαρινό εξάμηνο 2023-2024

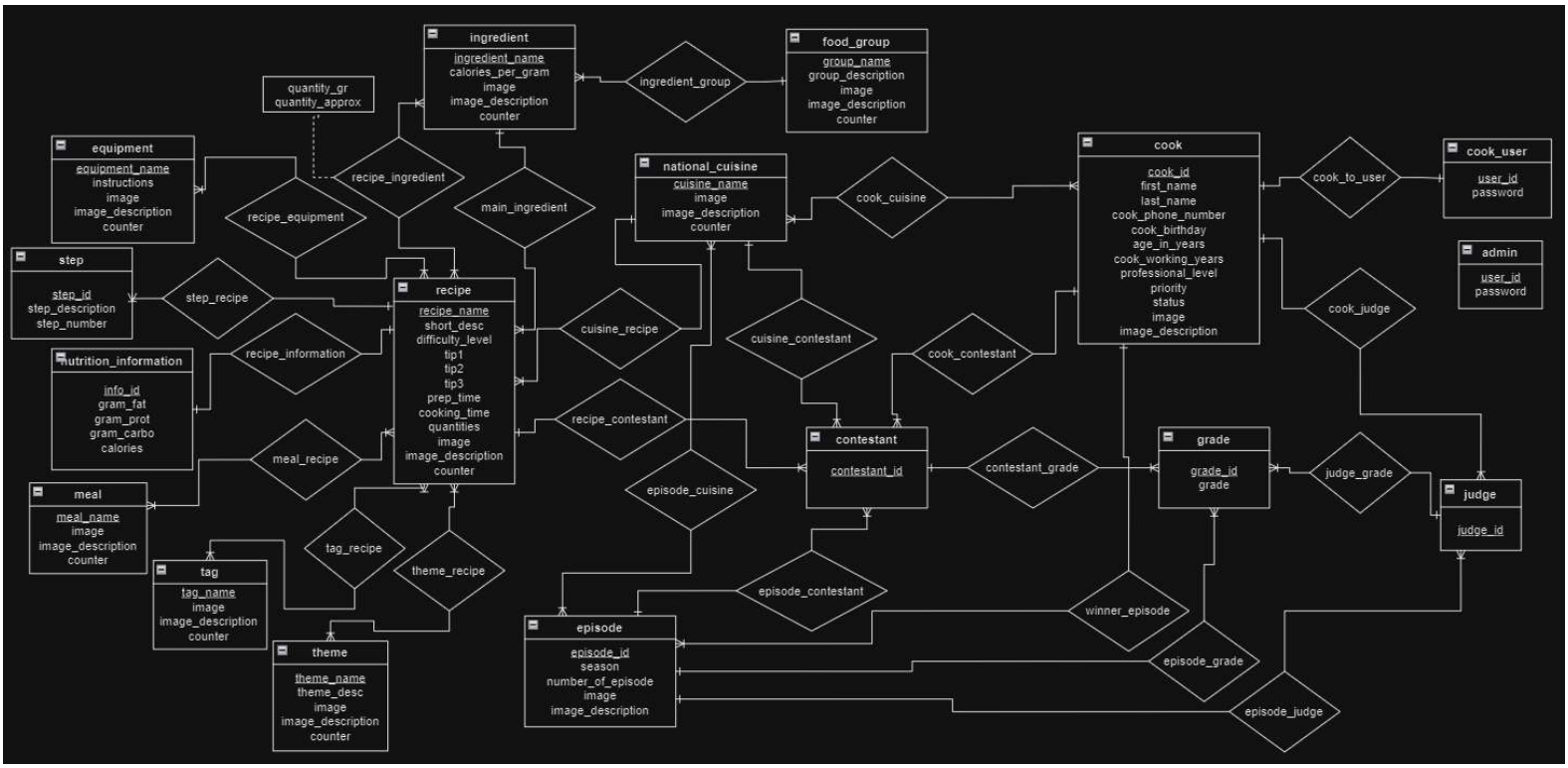
ΟΜΑΔΑ 57

Νίκη Γιαννακάκη 03121081

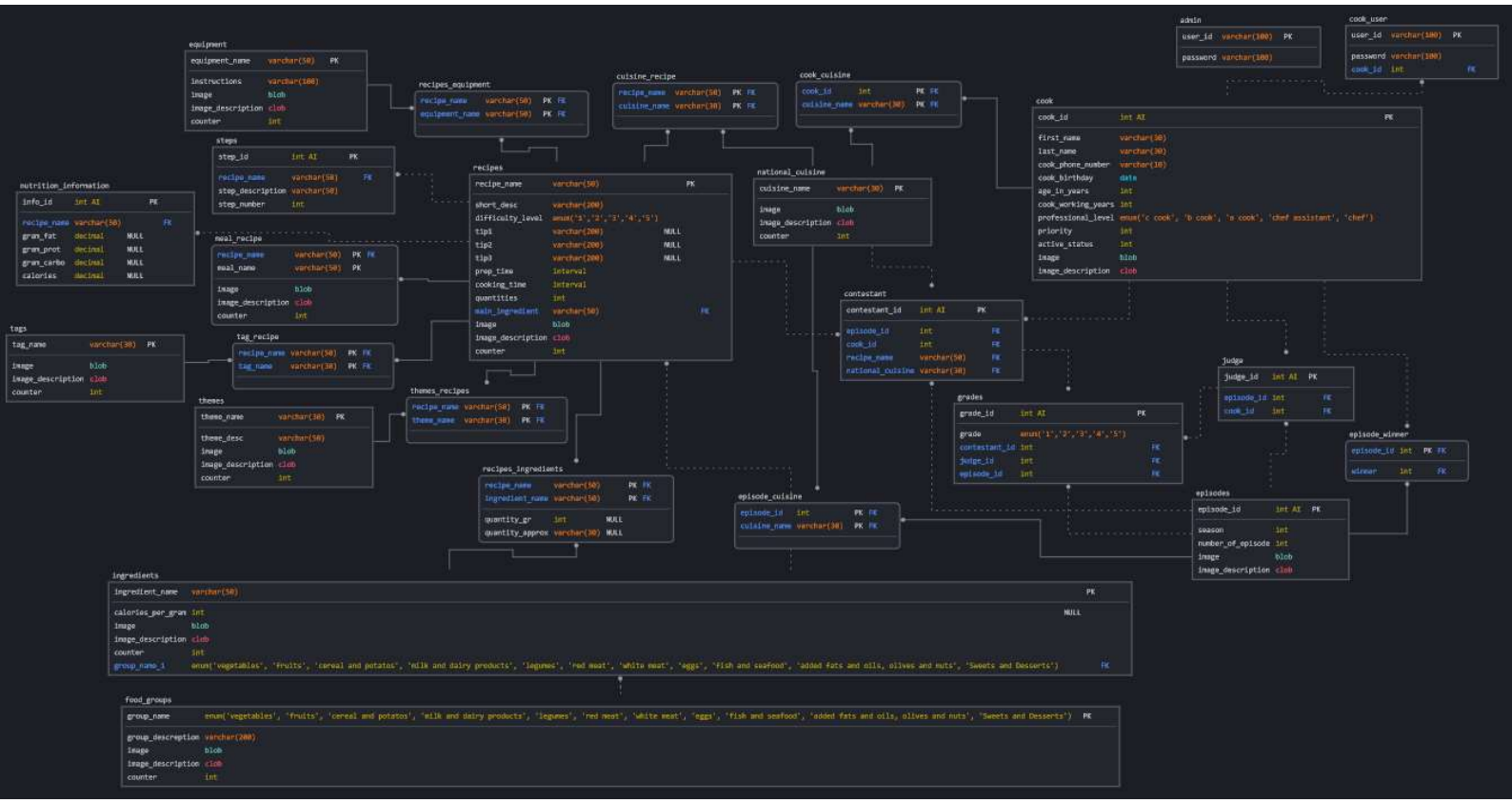
Παναγιώτης Γεωργαντάς 03121094

Πέτρος Βιτάλης 03121117

1. Διάγραμμα ER



2. Σχεσιακό Διάγραμμα



1. Περιορισμοί – Constraints

Στον πίνακα recipes_ingredients υπάρχει ένα check constraint που εξασφαλίζει πως ούτε quantity_gr, ούτε quantity_approx γίνεται να είναι ταυτόχρονα null value, αλλά ούτε και ταυτόχρονα not null. Έτσι εξασφαλίζεται ότι υπάρχει η ποσότητα του κάθε συστατικού που χρησιμοποιούμε είτε σε γραμμάρια είτε προσεγγιστικά.

Στον πίνακα cook ελέγχουμε μέσω ενός check constraint ότι το τηλέφωνο του κάθε μάγειρα είναι 10 ψηφία (κάθε ψηφίο παίρνει τιμές μεταξύ του 0 και του 9). Επίσης ελέγχουμε μέσω του UNIQUE ότι κάθε τηλέφωνο είναι διαφορετικό.

Ορίζουμε default τιμή στο attribute active_status τον αριθμό 1, που υποδηλώνει πως ο μάγειρας είναι ενεργός. Όταν θέλουμε να διαγράψουμε κάποιον μάγειρα κάνουμε update την τιμή του active_status του μάγειρα σε 0.

Κάθε μάγειρας μπορεί να έχει ηλικία μεταξύ 18 και 80 ετών.

Κάθε συνταγή, μάγειρας, εθνική κουζίνα μπορεί να συμμετέχει το πολύ τρεις φορές συνεχόμενα σε επεισόδια μίας χρονιάς, και μόνο οι μάγειρες που έχουν active_status τον αριθμό 1.

Ορίζουμε ως κλειδί στους πίνακες recipes, equipment, tags, themes, meal_recipe, ingredients, food_groups, national_cuisine το attribute counter ώστε μέσω της faker να προσθέσουμε σε αυτούς τους πίνακες εκ των υστέρων τις τιμές των attributes image, image_description.

Τα attributes που αναφέρονται στην εκφώνηση ως απαραίτητα έχουν οριστεί με τον περιορισμό NOT NULL. Τα υπόλοιπα δεν έχουν αυτόν τον περιορισμό.

Όλα τα primary keys είναι ορισμένα μαζί με NOT NULL.

Για τα foreign keys σε όλους τους πίνακες υπάρχει ο περιορισμός η τιμή τους να υπάρχει στο primary key του πίνακα στον οποίο αναφέρονται. Ορίζουμε ON UPDATE CASCADE σε κάθε ξένο κλειδί διότι θέλουμε να γίνονται οι αντίστοιχες ενημερώσεις των τιμών σε όλους τους πίνακες όταν μια αποδεκτή αλλαγή συμβαίνει. Αφήνουμε ON DELETE RESTRICT καθώς επιθυμούμε να διασφαλίζεται η ακεραιότητα της βάσης μας. Στην περίπτωση που θέλουμε να διαγράψουμε έναν μάγειρα κάνουμε update το active_status σε 0.

2. Indexing

Με βάση τα ερωτήματα που καλούμαστε να απαντήσουμε δημιουργήσαμε συγκεκριμένα ευρετήρια που η χρήση τους θα εξασφαλίσει γρηγορότερη και πιο αποδοτική αναζήτηση πινάκων, πράγμα που διαπιστώθηκε στα αντίστοιχα queries. Συγκεκριμένα φτιάξαμε ευρετήρια για τα:

- age_in_years στον πίνακα cook για το where clause του 3.3
- season στον πίνακα episodes για το where clause του 3.2, το group by του 3.9 και του 3.12 και το order by του 3.12

Για την ανάπτυξη της συγκεκριμένης βάσης δεδομένων, εκτός από τα απαιτούμενα της εκφώνησης έχουν γίνει και οι εξής παραδοχές:

Η εξειδίκευση ενός μάγειρα σε μία συγκεκριμένη εθνική κουζίνα (table cook_cuisine), συνεπάγεται ότι ο μάγειρας γνωρίζει και μπορεί να εκτελέσει όλες τις συνταγές που ανήκουν σε αυτή (table cuisine_recipe). Προφανώς στο μάγειρα δεν μπορούν να ανατεθούν συνταγές που υπάγονται σε κουζίνες που δεν έχει εξειδίκευση.

Ο περιορισμός που απαγορεύει την συμμετοχή μάγειρων/κριτών/συνταγών/κουζίνας σε παραπάνω από 3 επεισόδια συνεχόμενα εφαρμόζεται ανά σεζόν. Δηλαδή αν κάποιος μάγειρας συμμετέχει στα 3 τελευταία επεισόδια της πρώτης σεζόν, μπορεί και να συμμετέχει στο πρώτο επεισόδιο της δεύτερης.

Ως συμμετέχοντες σε ένα διαγωνισμό θεωρούμε τους αντιπρόσωπους και τους κριτές. (Στο ερώτημα 3.2 μετράμε μόνο τους διαγωνιζόμενους και όχι τους κριτές) (Στο ερώτημα 3.7 μετράμε και τους κριτές ως συμμετέχοντες στο επεισόδιο)

Στη περίπτωση που κάποιος μάγειρας αποχωρήσει από τον διαγωνισμό γίνεται ενημέρωση του attribute active_status στο table cook. Έτσι αναγνωρίζονται οι ενεργοί μάγειρες του διαγωνισμού, χωρίς να διαγράφονται τα δεδομένα των ανενεργών.

Δεν υπάρχει λόγος διαγραφής κάποιας οντότητας της βάσης μας πέρα από τους μάγειρες.

Για να ανατίθενται τα προνόμια στους διαχειριστές και τους μάγειρες κάθε φορά που κάνουμε insert στους πίνακες admin και cook_user αντίστοιχα δημιουργήσαμε triggers που καλούν procedures και με χρήση δυναμικής sql κάθε φορά παραχωρούν τα αντίστοιχα δικαιώματα πάνω στην βάση στους χρήστες. Αυτά τα triggers και procedures τα τρέχουμε από το terminal.

Για την δημιουργία καινούργιο επεισοδίου έχουμε κατασκευάσει procedure που παίρνει ως παράμετρο το πλήθος των επεισοδίων που θέλουμε να δημιουργήσουμε. Έπειτα η διαδικασία φροντίζει να επιλέξει τυχαία 10 διαγωνιζόμενους, 3 κριτές, 10 εθνικές κουζίνες και 10 συνταγές που καταχωρούνται στα πεδία στων πινάκων μας. Πάντα ελέγχεται ο περιορισμός των τριών συνεχόμενων εμφανίσεων των οντοτήτων του διαγωνισμού.

3. Ερωτήματα – Queries

1. . Μέσος Όρος Αξιολογήσεων (σκορ) ανά μάγειρα και Εθνική κουζίνα:

```
SELECT c.cook_id, CONCAT(c.first_name, ' ', c.last_name) AS cook_name, avg(g.grade) as avgcook_grade
FROM cook c
JOIN contestant co ON c.cook_id = co.cook_id
JOIN grades g ON co.contestant_id = g.contestant_id
GROUP BY c.cook_id, cook_name;

SELECT n.cuisine_name, avg(g.grade) as avgcuisine_grade
FROM national_cuisine n
JOIN contestant c ON n.cuisine_name = c.national_cuisine
JOIN grades g ON c.contestant_id = g.contestant_id
GROUP BY n.cuisine_name;
```

2. Μάγειρες που ανήκουν σε συγκεκριμένη εθνική κουζίνα και μάγειρες της κουζίνας που συμμετείχαν σε επεισόδια συγκεκριμένου έτους: (παράδειγμα με Ιαπωνική κουζίνα και 5ο έτος)

```
SELECT c.cook_id, CONCAT(c.first_name, ' ', c.last_name) AS cook_name
FROM cook c
WHERE c.cook_id IN (SELECT cc.cook_id
                    FROM cook_cuisine cc
                    WHERE cc.cuisine_name = 'Japanese');

SELECT DISTINCT *
FROM (SELECT c.cook_id, CONCAT(c.first_name, ' ', c.last_name) AS cook_name
      FROM cook c
      WHERE c.cook_id IN (SELECT cc.cook_id
                          FROM cook_cuisine cc
                          WHERE cc.cuisine_name = 'Japanese')) cu
WHERE cu.cook_id IN (SELECT con.cook_id
                     FROM contestant con
                     WHERE con.episode_id IN (SELECT e.episode_id
                                              FROM episodes e
                                              WHERE e.season = 5));
```

3. Νέοι μάγειρες (ηλικία < 30 ετών) που έχουν τις περισσότερες συνταγές:

```
SELECT
    c.cook_id,
    CONCAT(c.first_name, ' ', c.last_name) AS full_name,
    c.age_in_years,
    COUNT(DISTINCT ct.recipe_name) AS recipe_count
FROM
    cook c
JOIN
    contestant ct ON c.cook_id = ct.cook_id
WHERE
    c.age_in_years < 30
GROUP BY
    c.cook_id, full_name, c.age_in_years
ORDER BY
    recipe_count DESC;
```

4. Μάγειρες που δεν έχουν συμμετάσχει ποτέ ως κριτές σε επεισόδιο:

```sql
SELECT c.cook_id, c.first_name, c.last_name
FROM cook c
WHERE c.cook_id NOT IN (SELECT j.cook_id
                        FROM judge j);
```

5. Κριτές που έχουν συμμετάσχει στον ίδιο αριθμό επεισοδίων σε διάστημα ενός έτους με περισσότερες από 3 εμφανίσεις:

```sql
SELECT DISTINCT
    o.cook_id AS cook_id1,
    CONCAT(c1.first_name, ' ', c1.last_name) AS cook_name1,
    t.cook_id AS cook_id2,
    CONCAT(c2.first_name, ' ', c2.last_name) AS cook_name2,
    o.appearances
FROM (SELECT e.season, j.cook_id, COUNT(*) AS appearances
      FROM judge j
      INNER JOIN episodes e ON j.episode_id = e.episode_id
      GROUP BY e.season, j.cook_id
      HAVING COUNT(*) > 3) o
INNER JOIN (SELECT e.season, j.cook_id, COUNT(*) AS appearances
            FROM judge j
            INNER JOIN episodes e ON j.episode_id = e.episode_id
            GROUP BY e.season, j.cook_id
            HAVING COUNT(*) > 3) t
ON o.appearances = t.appearances
AND o.cook_id < t.cook_id
INNER JOIN cook c1 ON o.cook_id = c1.cook_id
INNER JOIN cook c2 ON t.cook_id = c2.cook_id
ORDER BY o.appearances, cook_id1, cook_id2;
```

6. Τρία κορυφαία ζεύγη tag που είναι κοινά σε συνταγές που εμφανίστηκαν σε επεισόδια:

```sql
SELECT tr1.tag_name AS tag1, tr2.tag_name AS tag2, COUNT(*) AS tot
FROM contestant con
INNER JOIN tag_recipe tr1
    ON con.recipe_name = tr1.recipe_name
INNER JOIN tag_recipe tr2
    ON con.recipe_name = tr2.recipe_name
    AND tr1.tag_name < tr2.tag_name
INNER JOIN episodes e
    ON con.episode_id = e.episode_id
GROUP BY tr1.tag_name, tr2.tag_name
ORDER BY tot DESC
LIMIT 3;
```

Παρακάτω φαίνονται πληροφορίες υλοποίησης του query:

```
MariaDB [cooking]> EXPLAIN SELECT tr1.tag_name AS tag1, tr2.tag_name AS tag2, COUNT(*) AS tot
    -> FROM contestant con
    -> INNER JOIN tag_recipe tr1 ON con.recipe_name = tr1.recipe_name
    -> INNER JOIN tag_recipe tr2 ON con.recipe_name = tr2.recipe_name AND tr1.tag_name < tr2.tag_name
    -> INNER JOIN episodes e ON con.episode_id = e.episode_id
    -> GROUP BY tr1.tag_name, tr2.tag_name
    -> ORDER BY tot DESC
    -> LIMIT 3;
+------+-------------+-------+------+-----------------------+-------------+---------+-------------------------+------+---------------------------------+
| id   | select_type | table | type | possible_keys         | key         | key_len | ref                     | rows | Extra                           |
+------+-------------+-------+------+-----------------------+-------------+---------+-------------------------+------+---------------------------------+
|    1 | SIMPLE      | e     | ALL  | PRIMARY               | NULL        | NULL    | NULL                    | 100  | Using temporary; Using filesort |
|    1 | SIMPLE      | con   | ref  | episode_fk2,recipe_fk5 | episode_fk2 | 4       | cooking.e.episode_id    | 1    |                                 |
|    1 | SIMPLE      | tr1   | ref  | recipe_name,tag_name  | recipe_name | 52      | cooking.con.recipe_name | 2    |                                 |
|    1 | SIMPLE      | tr2   | ref  | recipe_name,tag_name  | recipe_name | 52      | cooking.con.recipe_name | 2    | Using where                     |
+------+-------------+-------+------+-----------------------+-------------+---------+-------------------------+------+---------------------------------+
4 rows in set (0.001 sec)
```

Εναλλακτικό Query Plan με force index:

```
MariaDB [cooking]> EXPLAIN SELECT STRAIGHT_JOIN SQL_BUFFER_RESULT tr1.tag_name AS tag1, tr2.tag_name AS tag2, count(*) AS tot
    -> FROM episodes e
    -> INNER JOIN contestant con FORCE INDEX (idx_contestant_recipe_name) FORCE INDEX (idx_contestant_episode_id) ON con.episode_id = e.episode_id
    -> INNER JOIN tag_recipe tr1 ON con.recipe_name = tr1.recipe_name
    -> INNER JOIN tag_recipe tr2 ON con.recipe_name = tr2.recipe_name AND tr1.tag_name < tr2.tag_name
    -> GROUP BY tr1.tag_name, tr2.tag_name
    -> ORDER BY tot DESC
    -> LIMIT 3;
+------+-------------+-------+-------+-----------------------------------------------------+-------------------------+---------+-------------------------+------+-----------------------------------------------+
| id   | select_type | table | type  | possible_keys                                       | key                     | key_len | ref                     | rows | Extra                                         |
+------+-------------+-------+-------+-----------------------------------------------------+-------------------------+---------+-------------------------+------+-----------------------------------------------+
|    1 | SIMPLE      | e     | index | PRIMARY                                             | idx_episodes_season     | 4       | NULL                    | 100  | Using index; Using temporary; Using filesort  |
|    1 | SIMPLE      | con   | ref   | idx_contestant_episode_id,idx_contestant_recipe_name | idx_contestant_episode_id | 4       | cooking.e.episode_id    | 1    |                                               |
|    1 | SIMPLE      | tr1   | ref   | recipe_name,tag_name,idx_tag_recipe_tag_name        | recipe_name             | 52      | cooking.con.recipe_name | 2    |                                               |
|    1 | SIMPLE      | tr2   | ref   | recipe_name,tag_name,idx_tag_recipe_tag_name        | recipe_name             | 52      | cooking.con.recipe_name | 2    | Using where                                   |
+------+-------------+-------+-------+-----------------------------------------------------+-------------------------+---------+-------------------------+------+-----------------------------------------------+
4 rows in set (0.000 sec)
```

Η διαφορά στα δύο queries είναι πως στο δεύτερο τα δεδομένα του πίνακα episodes βρίσκονται μέσω της χρήσης του επιλεγμένου index καθώς έχουμε χρησιμοποιήσει force index. Κατά τα άλλα δεν υπάρχουν άλλες διαφορές διότι optimizer της sql έχει κάνει τις απαραίτητες βελτιστοποιήσεις για να τρέχει όσο πιο γρήγορα γίνεται το ερώτημα.

Στην συνέχεια φαίνεται το αντίστοιχο trace του δεύτερου query:


```
{
  "join_optimization": {
    "select_id": 1,
    "steps": [
      {
        "condition_processing": {
          "condition": "WHERE",
          "original_condition": "con.episode_id = e.episode_id and con.recipe_name = tr2.recipe_name and tr1.tag_name < tr2.tag_name and con.recipe_name
 = tr1.recipe_name",
          "steps": [
            {
              "transformation": "equality_propagation",
              "resulting_condition": "tr1.tag_name < tr2.tag_name and multiple equal(con.episode_id, e.episode_id) and multiple equal(con.recipe_name, t
r2.recipe_name, tr1.recipe_name)"
            },
            {
              "transformation": "constant_propagation",
              "resulting_condition": "tr1.tag_name < tr2.tag_name and multiple equal(con.episode_id, e.episode_id) and multiple equal(con.recipe_name, t
r2.recipe_name, tr1.recipe_name)"
            },
            {
              "transformation": "trivial_condition_removal",
              "resulting_condition": "tr1.tag_name < tr2.tag_name and multiple equal(con.episode_id, e.episode_id) and multiple equal(con.recipe_name, t
r2.recipe_name, tr1.recipe_name)"
            }
          ]
        }
      },
      {
        "table_dependencies": [
          {
            "table": "con",
            "row_may_be_null": false,
            "map_bit": 0,
            "depends_on_map_bits": []
          },
          {
            "table": "tr1",
            "row_may_be_null": false,
            "map_bit": 1,
            "depends_on_map_bits": ["0"]
```

7. Μάγειρες που συμμετείχαν τουλάχιστον 5 λιγότερες φορές από τον μάγειρα με τις περισσότερες συμμετοχές σε επεισόδια:

```
CREATE VIEW cook_participation AS
WITH max_part(value) AS (
    SELECT MAX(ap.tot) AS value
    FROM (
        SELECT (c.cnt + j.cnt) AS tot
        FROM (
            SELECT cook_id, COUNT(*) AS cnt
            FROM contestant
            GROUP BY cook_id
        ) c
        INNER JOIN (
            SELECT cook_id, COUNT(*) AS cnt
            FROM judge
            GROUP BY cook_id
        ) j ON c.cook_id = j.cook_id
         ) ap
    )
SELECT con.cook_id
    FROM (
        SELECT cook_id, COUNT(*) AS cnt
        FROM contestant
        GROUP BY cook_id
    ) con
    INNER JOIN (
        SELECT cook_id, COUNT(*) AS cnt
        FROM judge
        GROUP BY cook_id
    ) jud ON con.cook_id = jud.cook_id
    WHERE (con.cnt + jud.cnt) < (SELECT value FROM max_part) - 4;
```

Για το ερώτημα αυτό, αρχικά αναπτύσσουμε ένα View. Πρώτα βρίσκουμε max_part, ένας προσωρινός πίνακας που περιέχει τον μέγιστο αριθμό εμφανίσεων ενός μάγειρα στον διαγωνισμό περιλαμβάνοντας και τις εμφανίσεις του ως διαγωνιζόμενος και ως κριτής. Αποθηκεύονται στο View τα ids των μάγειρων που συμμετέχουν 5 λιγότερες φορές από τη μέγιστη συμμετοχή. Με το select βλέπουμε τα ids αυτά και το ονοματεπώνυμο του κάθε μάγειρα.

```
SELECT c.cook_id, CONCAT(c.first_name, ' ', c.last_name) AS cook_name
FROM cook c
JOIN cook_participation cp ON c.cook_id = cp.cook_id;
```

8. Επεισόδιο χρησιμοποιήθηκαν τα περισσότερα εξαρτήματα (εξοπλισμός):

```
SELECT con.episode_id, SUM(re.equip) AS sumequip
FROM (SELECT recipe_name, COUNT(*) AS equip
      FROM recipes_equipment
      GROUP BY recipe_name) re
INNER JOIN contestant con ON con.recipe_name = re.recipe_name
GROUP BY con.episode_id
ORDER BY sumequip DESC
LIMIT 1;
```

παρακάτω φαίνονται οι πληροφορίες υλοποίησης του query:

```
MariaDB [cooking]> EXPLAIN SELECT con.episode_id, SUM(re.equip) AS sumequip
    -> FROM (SELECT recipe_name, COUNT(*) AS equip
    ->       FROM recipes_equipment
    ->       GROUP BY recipe_name) re
    -> INNER JOIN contestant con ON con.recipe_name = re.recipe_name
    -> GROUP BY con.episode_id
    -> ORDER BY sumequip DESC
    -> LIMIT 1;
+------+-------------+------------------+-------+---------------+-------------+---------+---------------+------+----------------------------------------------+
| id   | select_type | table            | type  | possible_keys | key         | key_len | ref           | rows | Extra                                        |
+------+-------------+------------------+-------+---------------+-------------+---------+---------------+------+----------------------------------------------+
|    1 | PRIMARY     | <derived2>       | ALL   | NULL          | NULL        | NULL    | NULL          | 200  | Using temporary; Using filesort              |
|    1 | PRIMARY     | con              | ref   | recipe_fk5    | recipe_fk5  | 52      | re.recipe_name | 1    |                                              |
|    2 | DERIVED     | recipes_equipment | index | PRIMARY       | equipment_fk | 52      | NULL          | 200  | Using index; Using temporary; Using filesort |
+------+-------------+------------------+-------+---------------+-------------+---------+---------------+------+----------------------------------------------+
3 rows in set (0.000 sec)
```

Εναλλακτική υλοποίηση

```
MariaDB [cooking]> EXPLAIN SELECT STRAIGHT_JOIN SQL_BUFFER_RESULT con.episode_id, SUM(re.equip) AS sumequip
    -> FROM (SELECT recipe_name, COUNT(*) AS equip
    ->       FROM recipes_equipment
    ->       GROUP BY recipe_name) re
    -> INNER JOIN contestant con FORCE INDEX (idx_contestant_recipe_name) FORCE INDEX FOR GROUP BY (idx_contestant_episode_id) ON con.recipe_name = re.recipe_name
    -> GROUP BY con.episode_id
    -> ORDER BY sumequip DESC
    -> LIMIT 1;
+------+-------------+------------------+-------+-----------------------------+-----------------------------+---------+---------------+------+----------------------------------------------+
| id   | select_type | table            | type  | possible_keys               | key                         | key_len | ref           | rows | Extra                                        |
+------+-------------+------------------+-------+-----------------------------+-----------------------------+---------+---------------+------+----------------------------------------------+
|    1 | PRIMARY     | <derived2>       | ALL   | NULL                        | NULL                        | NULL    | NULL          | 200  | Using temporary; Using filesort              |
|    1 | PRIMARY     | con              | ref   | idx_contestant_recipe_name  | idx_contestant_recipe_name  | 52      | re.recipe_name | 1    |                                              |
|    2 | DERIVED     | recipes_equipment | index | PRIMARY                     | equipment_fk                | 52      | NULL          | 200  | Using index; Using temporary; Using filesort |
+------+-------------+------------------+-------+-----------------------------+-----------------------------+---------+---------------+------+----------------------------------------------+
3 rows in set (0.000 sec)
```

Όπως βλέπουμε δεν αλλάζει τίποτα με την χρήση force index καθώς ο optimizer από την αρχή υλοποιεί την βέλτιστη λύση. Βλέπουμε πως και το πρώτο χρησιμοποιεί ίδια indices με το δεύτερο για την αναζήτηση των στοιχείων του πίνακα οπότε λογικό να μην υπάρχει διαφορά στο αποτέλεσμα.

Από κάτω φαίνεται το trace του query plan:


```
{
  "join_optimization": {
    "select_id": 1,
    "steps": [
      {
        "condition_processing": {
          "condition": "WHERE",
          "original_condition": "con.recipe_name = re.recipe_name",
          "steps": [
            {
              "transformation": "equality_propagation",
              "resulting_condition": "multiple equal(con.recipe_name, re.recipe_name)"
            },
            {
              "transformation": "constant_propagation",
              "resulting_condition": "multiple equal(con.recipe_name, re.recipe_name)"
            },
            {
              "transformation": "trivial_condition_removal",
              "resulting_condition": "multiple equal(con.recipe_name, re.recipe_name)"
            }
          ]
        }
      },
      {
        "join_optimization": {
          "select_id": 2,
          "steps": [
            {
              "table_dependencies": [
                {
                  "table": "recipes_equipment",
                  "row_may_be_null": false,
                  "map_bit": 0,
                  "depends_on_map_bits": []
                }
              ]
            },
            {
              "rows_estimation": [
                {
```


9. Λίστα με μέσο όρο αριθμού γραμμάριων υδατανθράκων στο διαγωνισμό ανά έτος:

```sql
SELECT e.season, AVG(ci.carbo) AS avg_carbo
FROM (
    SELECT
        r.recipe_name,
        (ni.gram_carbo * r.quantities) AS carbo
    FROM
        nutrition_information ni
    INNER JOIN
        recipes r ON ni.recipe_name = r.recipe_name
) ci
INNER JOIN
    contestant er ON ci.recipe_name = er.recipe_name
INNER JOIN
    episodes e ON e.episode_id = er.episode_id
GROUP BY
    e.season;
```

10. Εθνικές που κουζίνες έχουν τον ίδιο αριθμό συμμετοχών σε διαγωνισμούς, σε διάστημα δύο συνεχόμενων ετών, με τουλάχιστον 3 συμμετοχές ετησίως:

```
WITH participations_per_season AS (
    SELECT
        c.national_cuisine,
        e.season,
        COUNT(*) AS participation_count
    FROM
        contestant c
        JOIN episodes e ON c.episode_id = e.episode_id
    GROUP BY
        c.national_cuisine, e.season
    HAVING
        COUNT(*) >= 3
),
consecutive_seasons AS (
    SELECT
        p1.national_cuisine,
        p1.season AS season1,
        p2.season AS season2,
        (p1.participation_count + p2.participation_count) AS total_count
    FROM
        participations_per_season p1
        JOIN participations_per_season p2 ON p1.national_cuisine = p2.national_cuisine
        AND p1.season = p2.season - 1
)
SELECT
    c1.national_cuisine,
    c2.national_cuisine,
    c1.season1,
    c1.season2,
    c1.total_count
FROM
    consecutive_seasons c1
JOIN
    consecutive_seasons c2 ON c1.total_count = c2.total_count
    AND c1.national_cuisine < c2.national_cuisine
    AND c1.season1 = c2.season1
    AND c1.season2 = c2.season2
ORDER BY
    c1.season1, c1.total_count;
```

Στον πίνακα participation_per_season βρίσκουμε τις κουζίνες που έχουν εμφανιστεί από 3 και πάνω φορές σε ένα χρόνο στον διαγωνισμό. Στον πίνακα consecutive_seasons αποθηκεύουμε τις φορές που κάθε κουζίνα από τον προηγούμενο πίνακα έχει εμφανιστεί σε δύο συνεχόμενα χρόνια. Τα ορίσματα αυτά διευκολύνουν το select που εμφανίζει τις ζητούμενες κουζίνες.

11. Top-5 κριτές που έχουν δώσει συνολικά την υψηλότερη βαθμολόγηση σε ένα μάγειρα

```
SELECT
    jc.cook_id AS judge_cook_id,
    CONCAT(jc.first_name, ' ', jc.last_name) AS judge_name,
    c.cook_id AS contestant_cook_id,
    CONCAT(c.first_name, ' ', c.last_name) AS contestant_name,
    SUM(CAST(gr.grade AS INT)) AS total_grade
FROM
    grades gr
INNER JOIN
    judge j ON gr.judge_id = j.judge_id
INNER JOIN
    contestant ct ON gr.contestant_id = ct.contestant_id
INNER JOIN
    cook c ON ct.cook_id = c.cook_id
INNER JOIN
    cook jc ON j.cook_id = jc.cook_id
GROUP BY
    jc.cook_id, ct.cook_id
ORDER BY
    total_grade DESC
LIMIT 5;
```

12. Τεχνικά δύσκολο, από πλευράς συνταγών, επεισόδιο του διαγωνισμού ανά έτος:

```
SELECT
    season,
    number_of_episode,
    total_difficulty
FROM (
    SELECT
        e.season,
        e.number_of_episode,
        SUM(CAST(r.difficulty_level AS INT)) AS total_difficulty,
        ROW_NUMBER() OVER (PARTITION BY e.season ORDER BY SUM(CAST(r.difficulty_level AS INT)) DESC) AS rank
    FROM
        episodes e
    INNER JOIN
        contestant c ON e.episode_id = c.episode_id
    INNER JOIN
        recipes r ON c.recipe_name = r.recipe_name
    GROUP BY
        e.season, e.number_of_episode
) AS ranked_episodes
WHERE
    rank = 1
ORDER BY
    season;
```

13. Το επεισόδιο που συγκέντρωσε τον χαμηλότερο βαθμό επαγγελματικής κατάρτισης (κριτές και μάγειρες):

```
SELECT gr.episode_id, ROUND(SUM(c.priority) / 3 + SUM(jc.priority) / 10)  AS professional_grade
FROM grades gr
JOIN contestant ct on gr.contestant_id = ct.contestant_id
JOIN cook c ON ct.cook_id = c.cook_id
JOIN judge j ON gr.judge_id = j.judge_id
JOIN cook jc ON j.cook_id = jc.cook_id
GROUP BY episode_id
ORDER BY professional_grade, RAND()
LIMIT 1;
```

14. Θεματική ενότητα που έχει εμφανιστεί τις περισσότερες φορές στο διαγωνισμό:

```
SELECT theme_name, COUNT(*) tot
FROM contestant con
INNER JOIN themes_recipes tr ON con.recipe_name = tr.recipe_name
GROUP BY theme_name
ORDER BY tot DESC
LIMIT 1;
```

15. Ομάδες τροφίμων δεν έχουν εμφανιστεί ποτέ στον διαγωνισμό:

```
SELECT
    fg.group_name
FROM
    food_groups fg
LEFT JOIN
    ingredients i ON fg.group_name = i.group_name
LEFT JOIN
    recipes_ingredients ri ON i.ingredient_name = ri.ingredient_name
LEFT JOIN
    contestant c ON ri.recipe_name = c.recipe_name
GROUP BY
    fg.group_name
HAVING
    COUNT(DISTINCT ri.recipe_name) = 0;
```

Οδηγίες Εγκατάστασης

Αρχικά, πρέπει να γίνει εγκατάσταση του MariaDB από το ακόλουθο link: https://mariadb.org/download/?t=mariadb&p=mariadb&r=11.3.2&os=windows&cpu=x86_64&pkg=msi&mirror=chroot-network . Μαζί θα εγκατασταθεί η εφαρμογή HeidiSQL.

Μετά τη σύνδεση στο Heidi ακολουθούν τα εξής βήματα:

1. Φορτώνουμε και εκτελούμε το αρχείο sql για τα tables(DDL_Script.sql).
2. Κλείνουμε το Heidi και το ξανά ανοίγουμε επιλέγοντας τη βάση cooking_contest κατά τη σύνδεση.
3. Φορτώνουμε και εκτελούμε τα sql αρχεία: inserts_final.sql, alter_tables_for_image.sql και από τον φάκελο python source update_statements.sql.
4. Η βάση είναι έτοιμη και μπορούμε να εκτελέσουμε τα queries από το command prompt του MariaDB.

Για την χρήση των αρχείων python απαιτείται η εγκατάσταση ενός Python Virtual Environment. Επίσης πρέπει να κατέβουν οι βιβλιοθήκες faker και random.

(pip install faker, pip install random)

Στη συνέχεια η εκτέλεση του κώδικα python γίνεται κανονικά.

Τα απαραίτητα αρχεία βρίσκονται στο git repo στο link:

https://github.com/nikigiannakaki/database-assignment-cooking-contest