Βάσεις Δεδομένων
Εξαμηνιαία Εργασία

Ομάδα 48
Κόρδας Νικόλαος 03121032, nikolaoskor8@gmail.com
Πανταζής Δημήτριος 03121208, dimpantam@gmail.com
Τσαλκιτζής Νικόλαος 03121123, nikostsalkitzhs@gmail.com

26 Μαΐου 2024

1 ER Διάγραμμα

Στην εργασία αυτή καλούμαστε να υλοποιήσουμε μία βάση δεδομένων για έναν μαγειρικό διαγωνισμό. Στην περίπτωσή μας, όπως κα σε οποιαδήποτε άλλη, πρώτα από όλα πρέπει να προσδιορίσουμε το σχήμα της βάσης μας. Για να το πετύχουμε αυτό οφείλουμε να προσδιορίσουμε ποιες οντότητες αντιπροσωπεύονται στη βάση μας και το πώς αυτές σχετίζονται μεταξύ τους. Το κατεξοχήν μέσο για την επίτευξη αυτού του στόχου είναι το μοντέλο οντοτήτων-συσχετίσεων (E-R).
Αρχικά, επιχειρήσαμε να εντοπίσουμε τα σύνολα οντοτήτων που υπάρχουν στη βάση μας. Το καταφέραμε διαβάζοντας προσεκτικά την εκφώνηση προσπαθώντας να εντοπίσουμε διακριτά αντικείμενα του κόσμου μας και τις ιδιότητες που τα χαρακτηρίζουν. Από την επεξεργασία μας προέκυψαν τα εξής σύνολα οντοτήτων:

- Συνταγές (recipes)
- Εξοπλισμός (cooking_gear)
- Υλικά (ingredients)
- Ομάδες Τροφίμων (food_groups)
- Θεματικές Ενότητες (theme_category)
- Μάγειρες (cooks)
- Επεισόδια (episodes)

Οι ιδιότητες που χαρακτηρίζουν τις οντότητες των παραπάνω συνόλων οντοτήτων φαίνονται αναλυτικά στο E-R διάγραμμα που θα ακολουθήσει. Πριν το παρουσιάσουμε, σημαντικό ήταν να εντοπίσουμε τις συσχετίσεις μεταξύ των οντοτήτων μας και άρα να δημιουργήσουμε τα κατάλληλα σύνολα συσχετίσεων. Τα σύνολα συσχετίσεων που καταλήξαμε είναι τα ακόλουθα:

- demands: recipes - cooking_gear
- belongs_to: recipes - theme_category
- basic_ingredient (kind()): ingredient - recipe
- contain (quantity, calories()): ingredient - recipe
- participant (part_id): cooks - episode *
- judge (judge_id): cooks - episode *
- grades (grade): participant - judge *
- has_to_cook: participant - recipe

Αξίζει να σχολιαστούν τα σύνολα συσχετίσεων που έχουμε επισημάνει με αστερίσκο. Προκειμένου να μοντελοποιήσουμε καλύτερα τον κόσμο μας αξιοποιήσαμε την έννοια της συνάθροισης. Δεν είναι τίποτα άλλο παρά μία αφαιρετική έννοια μέσω της οποίας χειριζόμαστε τις συσχετίσεις ως οντότητες υψηλότερου επιπέδου [1]. Εδώ, οι συσχετίσεις που χειριζόμαστε ως οντότητες είναι η participant και η judge, πράγμα λογικό, γιατί μπορεί αυτές να αποτελούν συσχέτιση μεταξύ ενός μάγειρα και του επεισοδίου που έπαιξε τον αντίστοιχο ρόλο, αλλά, ταυτόχρονα, ο μεν κριτής (judge) βαθμολογεί έναν συμμετέχοντα (participant) και ο δε participant μαγειρεύει μία συνταγή. Συμπεριφέρονται, δηλαδή, ως οντότητες.

Ακολουθεί φωτογραφία του E-R διαγράμματός μας:

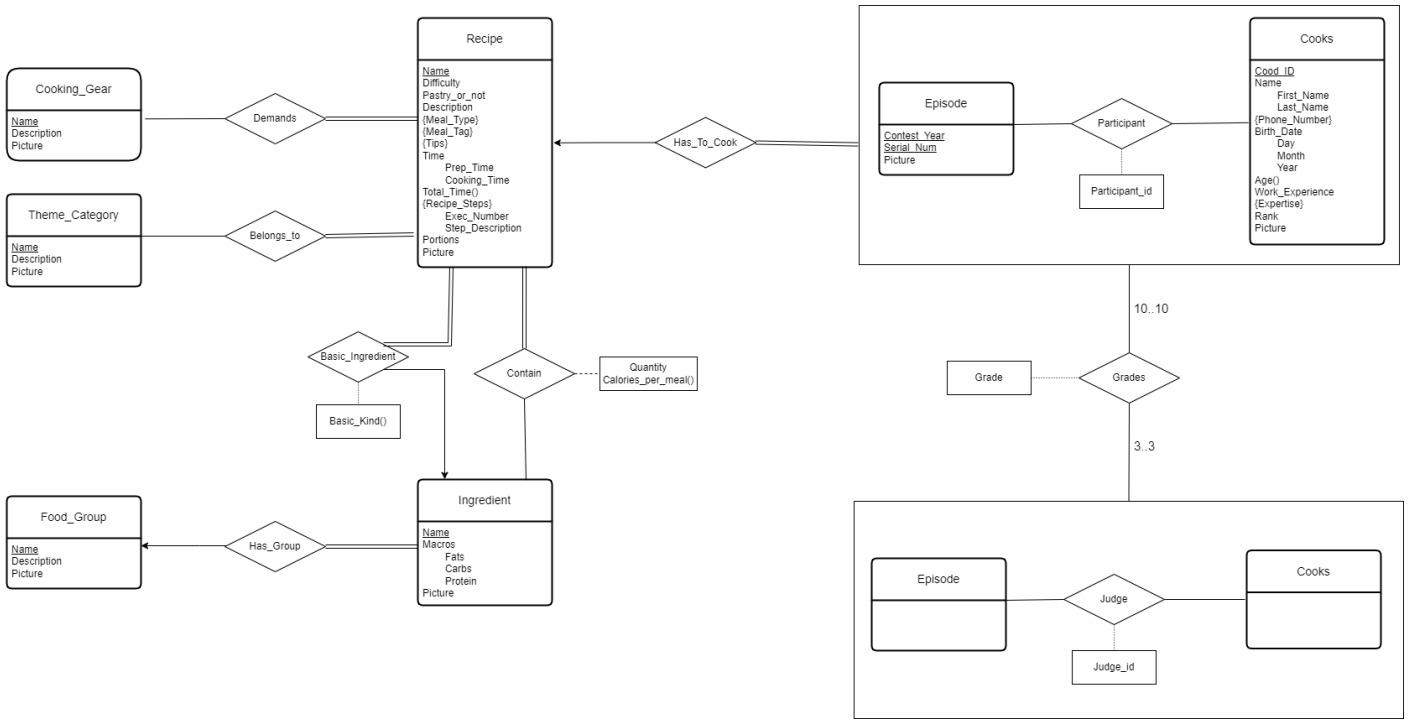

Figure 1: E-R Διάγραμμα

Το διάγραμμα φαίνεται ευκρινέστερα σε φωτογραφία που επισυνάπτεται μαζί με την παρούσα αναφορά. Εκεί, διακρίνουμε πως ορισμένες οντότητες δεν έχουν μόνο απλές ιδιότητες. Για παράδειγμα, οι ιδιότητες που περικλείονται με "{}" υποδηλώνουν μία ιδιότητα η οποία μπορεί να πάρει πάνω από μία τιμή (όπως το τηλέφωνο). Επιπλέον, υπάρχουν σύνθετες ιδιότητες που αναλύονται σε επιμέρους, όπως η Time του συνόλου οντοτήτων Recipes. Οι υπογραμμισμένες ιδιότητες αποτελούν το πρωτεύον κλειδί του εκάστοτε συνόλου οντοτήτων. Τέλος, οι ιδιότητες που ακολουθούνται από "()" είναι παραγόμενες από άλλες της οντότητας. Αυτές οι διακρίσεις μεταξύ των οντοτήτων αξιοποιούνται και πρακτικά (πέρα από σημασιολογικά) κατά την μετατροπή του E-R μοντέλου στο σχεσιακό που του αντιστοιχεί.

Πριν προχωρήσουμε στην ανάλυση του σχεσιακού μοντέλου, καλό θα ήταν να συζητήσουμε λεπτομερέστερα ορισμένες σχεδιαστικές αποφάσεις μας για το E-R, καθώς από αυτό εξαρτάται η συνέχεια της εργασίας μας. Όπως επιγραμματικά αναφέραμε παραπάνω, καθένα από τα 7 σύνολα οντοτήτων που υπάρχουν στο διάγραμμά μας επιλέχθηκε διότι ξεχώριζε σαφώς με τις ιδιότητές του από τα υπόλοιπα συστατικά του κόσμου του προβλήματος. Βρεθήκαμε όμως αντιμέτωποι με διλήμματα για το κατά πόσο πρέπει ορισμένα attributes οντοτήτων όπως το recipe_steps, meal_type ή meal_tag να ήταν δικές τους αυτοτελείς οντότητες οι οποίες θα σχετίζονταν με, το recipe εδώ, με μία ακόμα συσχέτιση. Αυτή η προοπτική απορρίφθηκε διότι οι υποψήφιες οντότητες δεν είχαν επιπλέον attributes εκτός του πρωτεύοντος κλειδιού (ονόματός τους) και έτσι το να τα θεωρήσουμε entities απλά αύξανε τον αριθμό των συσχετίσεων, χωρίς να προσφέρει ουσιαστικά στην σχεδίασή μας.

2 Σχεσιακό Μοντέλο

Το μοντέλο οντοτήτων - συσχετίσεων βρίσκεται πιο κοντά στον τρόπο με τον οποίο ο άνθρωπος αντιλλαμβάνεται και κατηγοριοποιεί τα δεδομένα του. Αυτός είναι και ο λόγος που ξεκινάμε καταστρώνοντας το E-R μοντέλο. Παρόλα αυτά, η SQL υλοποιεί σχεσιακές βάσεις δεδομένων. Με άλλα λόγια, βάσεις που αποτελούνται από ένα σύνολο πινάκων. Έτσι, η μετατροπή του E-R μοντέλου στο σχεσιακό είναι επιβεβλημένη.
Ευτυχώς, η διαδικασία της μετατροπής του ενός μοντέλου στο άλλο είναι κάπως τυποποιημένη. Αρχικά, λοιπόν, όλα τα σύνολα οντοτήτων γίνονται σχέσεις (πίνακες). Ιδιαίτερη μέριμνα πρέπει να δοθεί στις ιδιότητες τους. Οι σύνθετες ιδιότητες παραλείπονται εντελώς από τις στήλες του πίνακα, στον οποίο αναπαρίστανται μόνο οι απλές οι οποίες την αποτελούν. Έτσι, στο DDL script που παραθέτουμε, για παράδειγμα στην σχέση recipes, βλέπουμε τα attributes prep_time και cooking_time και όχι το time. Οι πλειότιμες ιδιότητες, όπως το phone_number των cooks, μετατρέπονται σε δικές τους σχέσεις. Η σχέση αυτή περιέχει το primary key της οντότητας που ανήκει η ιδιότητα, και η τιμή της ιδιότητας. Όλα τα παραπάνω απαρτίζουν το πρωτεύον κλειδί της νέας σχέσης. Οι παραγόμενες ιδιότητες δεν αναπαριστόνται καν αφού υπολογίζονται με κάποιο procedure, trigger ή και query. Τα σύνολα συσχετίσεων μετατρέπονται και αυτά σε σχέσεις (πίνακες). Περιέχουν τα πρωτεύοντα κλειδιά των σχέσεων (οντοτήτων) που συνδέουν (η ένωσή τους είναι το πρωτεύον κλειδί της νέας) καθώς και τα attributes του συνόλου συσχετίσεων. Το ίδιο ισχύει και για τις συσχετίσεις που αντιπροσωπεύουν την συνάθροιση. Κατά τον ορισμό

των πινάκων που αντιπροσωπεύουν σύνολα συσχετίσεων προσοχή πρέπει να δοθεί στην φορά των βελακίων του αντίστοιχου διαγράμματος. Με άλλα λόγια, πρέπει να φροντίσουμε το referencial integrity να υλοποιηθεί σωστά. Αυτό σημαίνει ότι στις συσχετίσεις πρέπει τα πρωτεύοντα κλειδιά των οντοτήτων που συμμετέχουν σε αυτές να επισημαίνονται ως foreign keys. Το ίδιο ισχύει και για τους πίνακες που αφορούν πλειότιμες ιδιότητες. Τους παραπάνω κανόνες αποτυπώσαμε στο DDL script που επισυνάπτουμε μαζί με αυτή την αναφορά. Στο script αυτό ορίζονται και ορισμένοι περιορισμοί ακεραιότητας που θα αναλύσουμε παρακάτω. Παρουσιάζουμε το Σχεσιακό Διάγραμμα που προέκυψε:

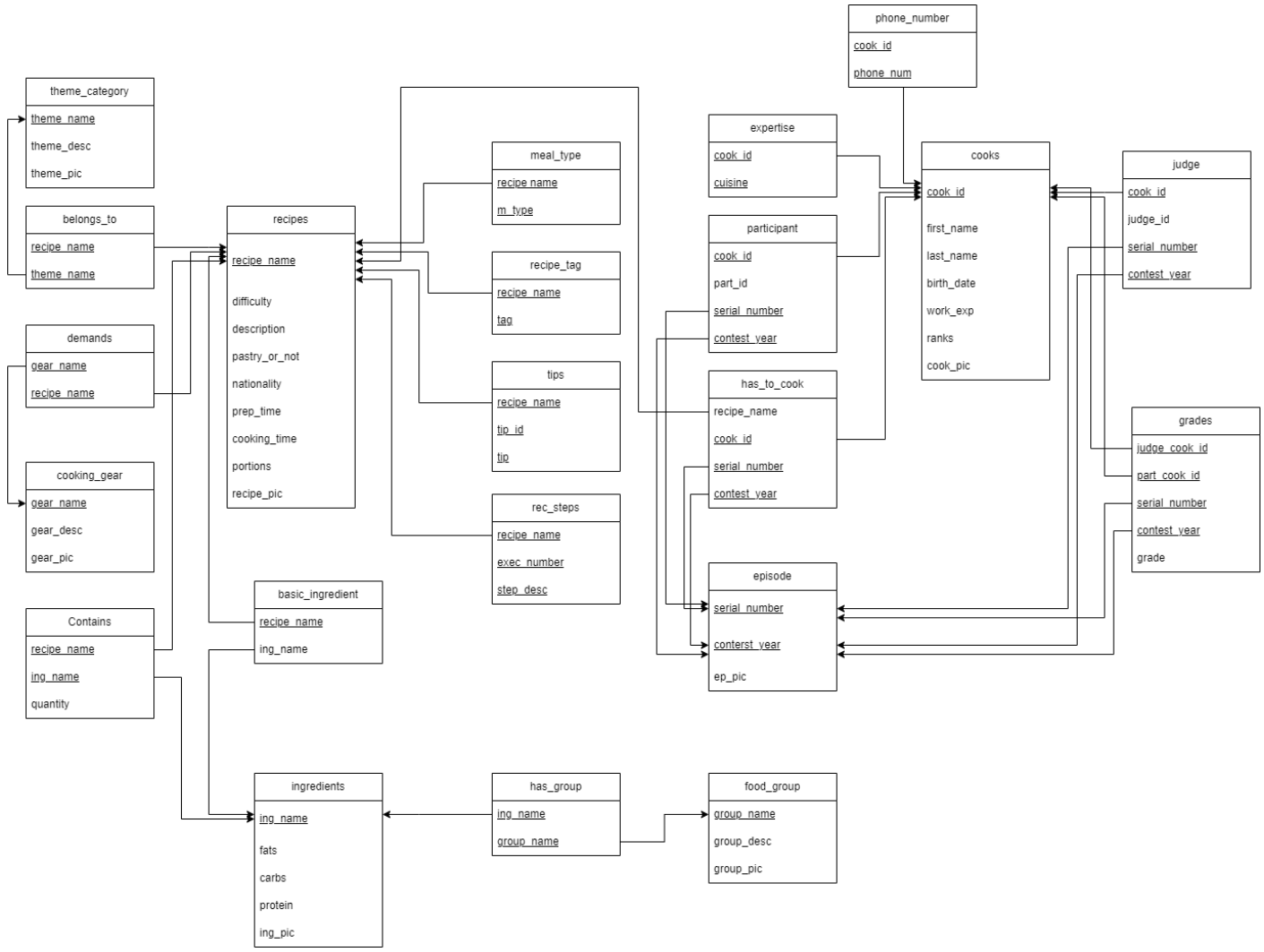

Figure 2: Σχεσιακό Διάγραμμα

2.1 Περιορισμοί (Constraints)

Τα διάφορα είδη περιορισμών σε μία βάση δεδομένων διασφαλίζουν την ορθότητα, την εγγυρότητα και την συνέπεια των δεδομένων που είναι αποθηκευμένα σε αυτή. Αρχικά, θα αναφερθούμε στους περιορισμούς ακεραιότητας. Αυτοί εγγυόνται πως οι αλλαγές που γίνονται σε βάσεις δεδομένων δεν καταλήγουν σε απώλεια συνέπειας των δεδομένων [2].

- **Περιορισμοί Not Null**
  Πρόκειται για έναν περιορισμό πεδίου τιμών. Τον αξιοποιήσαμε σε ιδιότητες πινάκων που δεν έχει νόημα να αγνοείται ή να μην υπάρχει η τιμή τους. Ένα χαρακτηριστικό παράδειγμα είναι το όνομα και το επώνυμο των μαγείρων, τα οποία δεν ανήκουν στο πρωτεύον κλειδί της σχέσης (by default not null) αλλά δεν έχει νόημα το null.

- **Περιορισμός Unique**
  Έτσι δηλώνουμε πως ένα υποσύνολο των ιδιοτήτων μιας σχέσης σχηματίζει ένα υπερκλειδί. Στην βάση μας δεν υπήρξε ανάγκη για την χρήση αυτού του περιορισμού.

- **Check**
  Ο όρος αυτός ορίζει ένα κατηγόρημα που πρέπει να ικανοποιείται από κάθε tupple της σχέσης. Μας φάνηκε ιδιαίτερα χρήσιμο για τον ορισμό user defined constraints όπως για παράδειγμα ότι η ποσότητα ενός υλικού πρέπει να είναι θετικός αριθμός.

Με τα κλειδιά έχουμε ασχοληθεί ήδη από το E-R μοντέλο. Τα πρωτεύοντα κλειδιά είναι τα ελάχιστα υπερκλειδιά που προσδιορίζουν μοναδιά μία πλειάδα και προφανώς καμία τιμή των ιδιοτήτων που τα απαρτίζουν δεν πρέπει να είναι null. Ιδιαίτερο ενδιαφέρον εμφανίζουν οι περιορισμοί με βάση τα ξένα κλειδιά, τα οποία είναι πρωτεύοντα κλειδιά σχέσεων που εμφανίζονται μέσα σε άλλες σχέσεις. Χρησιμοποιώντας τα μπορούμε να επιτύχουμε ακεραιότητα αναφορών (referencial integrity). Τι νόημα θα είχε να αποθηκεύσουμε ένα εργαλείο το οποίο δεν χρησιμοποιείται σε καμία συνταγή της βάσης μας? Ακόμα χειρότερα. Γίνεται κριτές να βαθμολογήσουν συμμετέχοντες διαφορετικών επεισοδίων? Η διασφάλιση του referencial integrity δίνει λύση σε τέτοια προβλήματα. Πολύ σημαντικό είναι να ορίσουμε και ποιες ενέργειες πρέπει να γίνουν σε περίπτωση διαγραφής, εισαγωγής ή ενημέρωσης ενός πίνακα που αναφέρεται από κάποιον άλλο. Στην περίπτωσή μας, έχουμε επιλέξει παντού σε περίπτωση ενημέρωσης ή διαγραφής το cascade (on insert/delete cascade). Αυτό εξασφαλίζει πως, παραδείγματος χάρην, όταν διαγράφεται ένας μάγειρας από τη βάση, διαγράφονται και τα τηλέφωνα που σχετίζονται με εκείνον.
Υπάρχουν και άλλοι περιορισμοί που θα συζητηθούν στο τέλος αυτής της ενότητας και υλοποιούνται με εναύσματα (triggers). Δεν έχει νόημα να αναφερθούμε στο κάθε constraint χωριστά, αφού όλα περιέχονται στο DDL sript μας.

2.2 Ευρετήρια (Indexes)

Ένα ευρετήριο σε μία ιδιότητα είναι μία δομή δεδομένων που επιτρέπει σε ένα σύστημα βάσης δεδομένων να βρίσκει αποτελεσματικά αυτές τις πλειάδες χωρίς την σάρωση όλων των tuples της σχέσης [3]. Γνωρίζουμε πως ένα DBMS δημιουργεί αυτόματα ένα ευρετήριο σε όλα τα πρωτεύοντα κλειδιά των σχέσεων που αναπαρίστανται σε αυτό. Πολλές φορές, όμως, γίνονται πάρα πολλά επερωτήματα που αφορούν ιδιότητες σχέσεων στις οποίες δεν υπάρχει ευρετήριο και άρα η αναζήτησή τους είναι αναποτελεσματική. Σε τέτοιες περιπτώσεις αξίζει να πληρώσουμε το κόστος σε χώρο αποθήκευσης που έχει η δημιουργία ενός ευρετηρίου, καθώς η κίνησή μας αυτή θα μας γλιτώσει πολύ χρόνο στην επεξεργασία των εν λόγω queries (προφανώς και δεν αλλάζουμε το primary key στο οποίο υπάρχει ήδη index). Έτσι, προκειμένου να επιλέξουμε τις ιδιότητες στις οποίες θα δημιουργήσουμε ευρετήρια διαβάσαμε τα υποερωτήματα 3 της εξαμηνιαίας εργασίας. Έτσι, εντοπίσαμε τις ιδιότητες που εμφανίζονται πιο συχνά. Αυτή είναι η recipe(nationallity) που χρησιμοποιείται σε 3 διαφορετικά queries (3.1, 3.2, 3.10). Εάν αυτό δεν υπήρχε, σε κάθε operation που αναφέρεται η Εθνική Κουζίνα, το σύστημα βάσης δεδομένων μας θα έπρεπε να σαρώνει όλες τις πλειάδες για να βρει τα αποτελέσματα που του ταιριάζουν. Τώρα κάτι τέτοιο δεν είναι απαραίτητο και άρα η ταχύτητα απόκρισης αυξάνεται δραματικά. Σαν ομάδα πήραμε την σχεδιαστική απόφαση να χρησιμοποιήσουμε μόνο ένα επιπλέον ευρετήριο καθώς αυτά είναι μεγάλες δομές και ειδικά αν είναι πολλά δεν βοηθούν στην βελτιστοποίηση της απόδοσης της βάσης μας. Εξάλλου, τις περισσότερες φορές, στα υποερωτήματα της άσκησής μας γίνονται join χρησιμοποιώντας το primary key των σχέσεων στο οποίο υπάρχει ήδη ευρετήριο.
Ορίζουμε και επιπλέον ευρετήρια για τα εναλλακτικά Query Plans των ερωτημάτων 3.6 και 3.8.

2.3 Procedures, Triggers

Πριν προχωρήσουμε στην παρουσίαση των επερωτημάτων που καταστρώσαμε για το ερώτημα 3 θα θέλαμε να αφιερώσουμε λίγο κείμενο στα procedures, triggers, ρόλους και άλλα στοιχεία της βάσης που δεν έχουν αναφερθεί ακόμα. Ξεκινάμε, λοιπόν, με τα procedures:

- **calculateCookAge()**: Η συνάρτηση αυτή είναι υπεύθυνη για τον υπολογισμό της ηλικίας του κάθε μάγειρα (ήταν παραγόμενη ιδιότητα του συνόλου οντοτήτων) χρησιμοποιώντας την ημερομηνία γέννησής του και την τρέχουσα ημερομηνία.

- **CalculateGrades()**: Η συνάρτηση αυτή επιστρέφει το άθροισμα των βαθμών κάθε συμμετέχοντα στο εκάστοτε επεισόδιο. Χρησιμεύει στην ανάδειξη του νικητή.

- **RandomlySelect()**: Επιστρέφει τυχαία n tuples από τη σχέση που θα του ζητηθεί. Με αυτή επιλέγονται τυχαία οι μάγειρες που επιλέγονται ως κριτές ή συμμετέχοντες στα επεισόδια του διαγωνισμού.

- **basic_kind()**: Επιστρέφει το παραγόμενο attribute του συνόλου συσχετίσεων basic_ingedient μεταξύ του recipes και του ingredient. Δεν είναι τίποτα άλλο παρά ένα mapping των υλικών στις κατηγορίες που ανήκουν.

- **select_episode_winners()**: Επιλέγει τον νικητή κάθε επεισοδίου φροντίζοντας να επιλύσει τις ισοβαθμίες με βάση το rank τους ή και άλλα κριτήρια με φθίνουσα σημασία.

Μπορεί κανείς εύκολα να αντιληφθεί ότι χωρίς τις παραπάνω διαδικασίες αφενός δεν θα είχαμε τρόπο να υπολογίσουμε τις παραγόμενες ιδιότητες των οντοτήτων μας, πράγμα που θα καταστρατηγούσε την σχεδίασή μας και αφετέρου η βάση θα ήταν πολύ πιο δύσχρηστη.

Συνεχίζοντας, αναφερόμαστε στα εναύσματα που ορίσαμε. Ουσιαστικά, υλοποιούν πιο σύνθετους περιορισμούς που προκύπτουν κατά την τροποποίηση της βάσης μας:

- **check_consecutive_participation**: Αναφέρεται στην περίπτωση before insertion. Η κλήρωση μπορεί να οδηγήσει στην συμμετοχή του ίδιου μάγειρα σε 3 συνεχόμενα επεισόδια, πράγμα που απαγορεύεται από τις προδιαγραφές της άσκησης. Με αυτό το trigger απαγορεύουμε το insertion ενός τέτοιου αποτελέσματος στη βάση εξασφαλίζοντας την τήρηση των απαιτήσεών μας.
- **check_consecutive_judges**: Ουσιαστικά, επιτελεί τον ίδιο σκοπό με το προηγούμενο έναυσμα αλλά για τους μάγειρες-κριτές, όχι για τους συμμετέχοντες.
- **check_expertise_before_cook_assignment**: Στον διαγωνισμό μας, κάθε μάγειρας εκτελεί συνταγές στις οποίες είναι ειδικός. Έτσι, before insertion στη σχέση has_to_cook πρέπει να εξασφαλίσουμε πως δεν καταπατούμε αυτή την προδιαγραφή.

2.4 Χρήστες Εφαρμογής (Users)

Κατά την είσοδό μας στο σύστημα βάσης δεδομένων μας, μας ζητείται να κάνουμε log in με κάποιο username και password. Αρχικά, κατά την δημιουργία της βάσης μας συνδεόμασταν πάντα ως root που είναι ο χρήστης που έχει προνόμια να κάνει οτιδήποτε επιθυμεί στη βάση. Για τις ανάγκες της εργασίας δημιουργήσαμε επιπλεόν χρήστες και μηχανισμούς εξουσιοδότησης σε ορισμένες προβολές προκειμένου να επιτευχθεί η επιθυμητή λειτουργικότητα. Πιο συγκεκριμένα:

- **Διαχειριστής**: Εκτός από την καταχώρηση και τροποποίηση δεδομένων, ο χρήστης admin πρέπει να μπορεί να δημιουργήσει backup της βάσης, καθώς και να την επαναφέρει. Για να το πετύχουμε αυτό του δώσαμε επιπλέον τα απαραίτητα προνόμια που αναφέρονται στο documentation της MySQL. (source). Ο αντίστοιχος κώδικας φαίνεται στο script cooking-contest-users.sql. Λεπτομερέστερα, οι εντολή που δημιουργεί το backup είναι η **mysqldump -u admin -p cooking_contest > backup.sql** ενώ αυτή που κάνει restore η **mysql -u admin -p < backup.sql**.
- **Μάγειρας**: Για να μπορέσουμε να ικανοποιήσουμε την απαίτηση της εκφώνησης κάθε μάγειρας να μπορεί να τροποποιεί τις συνταγές που του έχουν ανατεθεί, πρέπει να δημιουργήσουμε τόσο views όσα οι μάγειρες, καθένα από τα οποία θα περιέχει τις εκάστοτε συνταγές. Επιπλέον δημιουργούνται και άλλοι τόσο ρόλοι cook στους οποίους απονέμονται προνόμια ανάγνωσης και τροποποίησης της αντίστοιχης προβολής, αλλά και εισαγωγής στον πίνακα των συνταγών. Τέλος, ο ρόλος cook απονέμεται στον εκάστοτε μάγειρα. Ο κώδικας βρίσκεται στο αρχείο cooking-contest-users.sql.

Προφανώς τα script αυτά δεν γράφτηκαν με το χέρι. Χρησιμοποιήσαμε ορισμένα python scipts που γράψαμε οι ίδιοι προκειμένου να δημιουργήσουμε τις προβολές, τους χρήστες και τα προνόμια. Τα scripts αυτά φαίνονται στο github repo που δίνουμε στο τέλος.
Αξίζει να σημειώσουμε πως προσπαθήσουμε να αυτοματοποιήσουμε την παραπάνω διαδικασία με SQL και συγκεκριμένα ένα trigger στην είσοδο νέου μάγειρα. Το συγκεκριμένο έναυσμα θα φαίνεται ως σχόλιο στον κώδικα. Ο λόγος που δεν προτιμήθηκε αυτή η ομολογουμένως κομψότερη λύση είναι πως η MySQL απαγορεύει ρητά την χρήση dynamic SQL στις συναρτήσεις, διαδικασίες και τα εναύσματα.

3 Επερωτήματα (Queries)

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

Σε αυτό το υπερώτημα για το πρώτο σκέλος του απλώς από το relation των grades κάναμε grouping by την χαρακτηριστική ταυτότητα των συμμετεχόντων και τυπώσαμε τόσο την τελευταία όσο και την μέση βαθμολογία τους ανά τα χρόνια των διαγωνισμών.

```
select part_cook_id as Participant_Name, avg(grade) as average_grade
from grades
group by part_cook_id;
```

Figure 3: SQL κώδικας για το πρώτο σκέλος

Για το μέσο score ανά εθνική κουζίνα, πρέπει να ομαδοποιήσουμε με βάση την εθνικότητα της κουζίνας και να κάνουμε join 4 relations. Πιο συγκεκριμένα, έχουμε ότι πρέπει να χρησιμοποιήσουμε τα grades, μετά τον πινακά των participants με την σχέση has_to_cook και την recipes έτσι ώστε έχονατς ώς predicate το όνομα του συμμετέχοντα, το επεισόδιο και τον χρόνο, βρίσκουμε απο την δεύτερη σχέση την συνταγή που έχει μαγειρέψει και κάνοντας join με τις συνταγές βρίσκουμε την εθνική κουζίνα.

```sql
select r.nationality as National_Cuisine, avg(g.grade) as average_grade
from
grades g
join
    participant p on g.part_cook_id = p.cook_id and g.serial_number = p.serial_number and g.contest_year = p.contest_year
join
    has_to_cook h on p.cook_id = h.cook_id and p.serial_number = h.serial_number and p.contest_year = h.contest_year
join
    recipes r on h.recipe_name = r.recipe_name
group by
    r.nationality;
```

Figure 4: SQL κώδικας για το δεύτερο σκέλος

3.2 Για δεδομένη Εθνική κουζίνα και έτος, ποιοι μάγειρες ανήκουν σε αυτήν και ποιοι μάγειρες συμμετείχαν σε επεισόδια;

Εδώ δημιουργήσαμε δύο temporary tables καθώς αντιληφθήκαμε ότι πρόκειται για τομή των δύο αποτελεσματών. Ειδικότερα, βρήκαμε για κάθε εθνικότητα τους μάγειρες που ανήκουν σε αυτήν και μετά βρήκαμε ποιοι μάγειρες συμμετείχαν σε κάθε έτος του διαγωνισμού και αποθηκεύσαμε το αποτελέσματα στα tables cooks_by_nationality και cooks_by_contest_year, αντίστοιχα. Ακολούθως, υπερωτώντας τα temporaries tables κάνουμε inner join με βάση το όνομα του participant τυπώνουμε το τελικό αποτελέσμα.

```sql
with cooks_by_nationality as (
    select r.nationality, p.cook_id
    from has_to_cook h
    join recipes r on h.recipe_name = r.recipe_name
    join participant p on h.cook_id = p.cook_id
    group by r.nationality, p.cook_id
),
cooks_by_contest_year as (
    select p.contest_year, p.cook_id
    from participant p
    group by p.contest_year, p.cook_id
)
select n.nationality, c.contest_year, n.cook_id
from cooks_by_nationality n
inner join cooks_by_contest_year c on n.cook_id = c.cook_id;
```

Figure 5: SQL κώδικας για το ερώτημα 3.2

3.3 Βρείτε τους νέους μάγειρες (ηλικία < 30 ετών) που έχουν τις περισσότερες συνταγές.

Ουσιαστικά, για να απαντήσουμε σε αυτό το query θα χρειαστούμε την συνάρτηση η οποία μας υπολογίζει την ηλικία κάθε μάγειρα. Εν συνεχεία, θα χρειαστούμε έναν μετρητή στον οποίον κρατάμε την πληροφορία για το πλήθος των συνατγών που έχει κάθε μάγειρας. Μετά, κάνουμε join τον πίνακα των participants με το has_to_cook και τα recipes με βάση το όνομα του και την συνταγή, επιλέγουμε αυτούς με ηλικία μικρότερη των 30 ετών και τους κατατάσσουμε με φθίνουσα σειρά. Για λόγους πληρότητας επιστρέφουμε τους top-5 μάγειρες.

```sql
SELECT p.cook_id, COUNT(*) AS recipe_count
FROM participant p
JOIN has_to_cook h ON p.cook_id = h.cook_id
JOIN recipes r ON h.recipe_name = r.recipe_name
WHERE calculateCookAge(p.cook_id) < 30
GROUP BY p.cook_id
ORDER BY COUNT(*) DESC
LIMIT 5;
```

Figure 6: SQL κώδικας για το ερώτημα 3.3

3.4 Βρείτε τους μάγειρες που δεν έχουν συμμετάσχει ποτέ σε ως κριτές σε κάποιο επεισόδιο

Επιλέξαμε να εργαστούμε με joins μεταξύ του πίνακα των κριτών και των μαγείρων. Πιο συγκεκριμένα, χρησιμοποιήσαμε left join έτσι ώστε να βρούμε όλους τους κριτές μάγειρες και στην συνέχεια επιλέξαμε μόνο αυτούς με τιμή στο πεδίο j.cook_id NULL δηλώνοντας ότι δεν έχουν υπάρξει ποτέ κριτές σε κάποιο επεισόδιο.

```sql
SELECT c.cook_id, c.first_name, c.last_name
FROM cooks c
LEFT JOIN judge j ON c.cook_id = j.cook_id
WHERE j.cook_id IS NULL;
```

Figure 7: SQL κώδικας για το ερώτημα 3.4

3.5 Ποιοι κριτές έχουν συμμετάσχει στον ίδιο αριθμό επεισοδίων σε διάστημα ενός έτους με περισσότερες από 3 εμφανίσεις;

Για το πέμπτο ερώτημα χρησιμοποιήσαμε εμφολευμένα queries και το εσωτερικό μας επιστρέφει την ταυτότητα του μάγειρα και το πλήθος των εμφανίσεων του σε κάθε χρόνο, μαζί με το πλήθος των φορών που έλαβε μέρος ως κριτής. Απο αυτόν τον πίνακα που προκύπτει επιλέγουμε μονάχα τους μάγειρες με πλήθος εμφανίσεων μεγαλύτερο του 3.

```sql
SELECT judge_appearances.cook_id, judge_appearances.contest_year
FROM (
    SELECT cook_id, contest_year, COUNT(*) AS num_appearances
    FROM judge
    GROUP BY cook_id, contest_year
) AS judge_appearances
WHERE judge_appearances.num_appearances > 3;
```

Figure 8: SQL κώδικας για το ερώτημα 3.5

3.6 Πολλές συνταγές καλύπτουν περισσότερες από μια ετικέτες. Ανάμεσα σε ζεύγη πεδίων (π.χ. brunch και κρύο πιάτο) που είναι κοινά στις συνταγές, βρείτε τα 3 κορυφαία (top-3) ζεύγη που εμφανίστηκαν σε επεισόδια

Αυτό το ερώτημα SQL στοχεύει να προσδιορίσει τα τρία κορυφαία ζεύγη ετικετών γευμάτων που εμφανίζονται συχνά μαζί σε συνταγές που εμφανίζονται σε επεισόδια ενός διαγωνισμού μαγειρικής. Αυτό το επιτυγχάνει ενώνοντας τον πίνακα meal_tag με τον εαυτό του (με το ψευδώνυμο rt1 και rt2) για να βρει ζεύγη ετικετών που σχετίζονται με την ίδια συνταγή, φιλτράροντας τα διπλότυπα διασφαλίζοντας ότι το tag1 είναι μικρότερο από το tag2. Στη συνέχεια, ενώνει αυτό το αποτέλεσμα με τον πίνακα has_to_cook για να συνδέσει συνταγές με επεισόδια και, τέλος, με τον πίνακα επεισοδίων για να διασφαλίσει ότι λαμβάνονται

υπόψη μόνο οι συνταγές που έχουν μαγειρευτεί σε επεισόδια. Το ερώτημα υπολογίζει τον αριθμό των επεισοδίων για κάθε ζεύγος ετικετών και παρουσιάζει τα τρία πρώτα ζεύγη με βάση τον υψηλότερο αριθμό επεισοδίων.

```sql
WITH RecipeTags AS (
    SELECT
        rt1.tag AS tag1,
        rt2.tag AS tag2,
        COUNT(*) AS episode_count
    FROM
        meal_tag rt1
    JOIN
        meal_tag rt2 ON rt1.recipe_name = rt2.recipe_name AND rt1.tag < rt2.tag
    JOIN
        has_to_cook hc ON rt1.recipe_name = hc.recipe_name
    JOIN
        episode e ON hc.serial_number = e.serial_number AND hc.contest_year = e.contest_year
    GROUP BY
        rt1.tag, rt2.tag
)
SELECT
    tag1,
    tag2,
    episode_count
FROM
    RecipeTags
ORDER BY
    episode_count DESC
LIMIT 3;
```

Figure 9: SQL κώδικας για το ερώτημα 3.6

Για το παραπάνω query χρησιμοποιήσαμε την εντολή explain και παρήγαγε το εξής αποτέλεσμα:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	<derived2>	NULL	ALL	NULL	NULL	NULL	NULL	122	100.00	Using filesort
2	DERIVED	rt1	NULL	index	PRIMARY,recipe_tag	recipe_tag	202	NULL	184	100.00	Using index; Using temporary
2	DERIVED	rt2	NULL	ref	PRIMARY,recipe_tag	recipe_tag	202	cooking_contest.rt1.recipe_name	2	33.33	Using where; Using index
2	DERIVED	hc	NULL	ref	has_to_cook_recipe	has_to_cook_recipe	203	cooking_contest.rt1.recipe_name	1	100.00	Using index
2	DERIVED	e	NULL	eq_ref	PRIMARY	PRIMARY	6	cooking_contest.hc.serial_number,cooking_con...	1	100.00	Using index

Figure 10: Explain result για το πρώτο σκέλος του 3.6

Στην συνέχεια έχουμε ότι δημιουργήσαμε τρία indexes για την επερχόμενη εκτέλεση του query:

```sql
CREATE INDEX idx_meal_tag_recipe_name_tag ON meal_tag(recipe_name, tag);
CREATE INDEX idx_has_to_cook_recipe_serial_contest ON has_to_cook(recipe_name, serial_number, contest_year);
CREATE INDEX idx_episode_serial_contest ON episode(serial_number, contest_year);
```

Figure 11: Indexes για το ερώτημα 3.6

Ως εναλλακτικό query χρησιμοποιήσαμε τα παραπάνω force indexes και παρηγάγαμε αυτόν τον κώδικα:

```sql
explain
WITH RecipeTags AS (
    SELECT
        rt1.tag AS tag1,
        rt2.tag AS tag2,
        COUNT(*) AS episode_count
    FROM
        meal_tag rt1 FORCE INDEX (idx_meal_tag_recipe_name_tag)
    JOIN
        meal_tag rt2 FORCE INDEX (idx_meal_tag_recipe_name_tag)
        ON rt1.recipe_name = rt2.recipe_name AND rt1.tag < rt2.tag
    JOIN
        has_to_cook hc FORCE INDEX (idx_has_to_cook_recipe_serial_contest)
        ON rt1.recipe_name = hc.recipe_name
    JOIN
        episode e FORCE INDEX (idx_episode_serial_contest)
        ON hc.serial_number = e.serial_number AND hc.contest_year = e.contest_year
    GROUP BY
        rt1.tag, rt2.tag
)
SELECT
    tag1,
    tag2,
    episode_count
FROM
    RecipeTags
ORDER BY
    episode_count DESC
LIMIT 3;
```

Figure 12: Query εναλλακτικό για το ερώτημα 3.6

Τώρα με την χρήση explain για τον παραπάνω κώδικα το αποτέλεσμα του είναι :

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	<derived2>	NULL	ALL	NULL	NULL	NULL	NULL	919	100.00	Using filesort
2	DERIVED	rt1	NULL	index	idx_meal_tag_recipe_name_tag	idx_meal_tag_recipe_name_tag	284	NULL	184	100.00	Using index; U
2	DERIVED	rt2	NULL	ref	idx_meal_tag_recipe_name_tag	idx_meal_tag_recipe_name_tag	202	cooking_contest.rt1.recipe_name	2	33.33	Using where; I
2	DERIVED	hc	NULL	ref	idx_has_to_cook_recipe_serial_contest	idx_has_to_cook_recipe_serial_contest	203	cooking_contest.rt1.recipe_name	7	100.00	Using index
2	DERIVED	e	NULL	eq_ref	idx_episode_serial_contest	idx_episode_serial_contest	6	cooking_contest.hc.serial_number,cooking_con...	1	100.00	Using index

Figure 13: Explain result για το δεύτερο σκέλος του 3.6

Πριν αναλύσουμε την συμπεριφορά των παραπάνω queries, θα δείξουμε το τί επιστρέφει το explain:

- id: Ένα αναγνωριστικό για κάθε επιλεγμένο ερώτημα εντός του σχεδίου εκτέλεσης.
- Select_type: Ο τύπος του ερωτήματος επιλογής (π.χ. ΑΠΛΟ, ΚΥΡΙΟ, ΠΑΡΑΓΩΓΟ).
- Table: Ο πίνακας που εμπλέκεται στο ερώτημα επιλογής.
- Type: Ο τύπος σύνδεσης που χρησιμοποιείται (π.χ. ALL, ευρετήριο, εύρος, αναφορά).
- possible_keys: Τα indexes που μπορούν να χρησιμοποιηθούν.
- key: Το index το οποίο χρησιμοποιείται όντως
- key_len: Το μήκος του τελευταίου
- ref: Οι στήλες σε σύγκριση με το ευρετήριο
- rows: Ο αριθμός των σειρών που εξετάστηκαν
- filtered: Το ποσοστό των σειρών που φιλτράρονται από τη συνθήκη
- Extra: Πρόσθετες πληροφορίες σχετικά με το σχέδιο εκτέλεσης

Παρατηρούμε ότι με την χρήση των indexes τα rows που γίνονται examined είναι πολύ περισσότερα σε σχέση με το προηγούμενο query(919>122), γεγονός που αυξάνει κατά πολύ τον χρόνο και την πολυπλοκότητα της εμφάνισης του αποτελέσματος.

3.7 Βρείτε όλους τους μάγειρες που συμμετείχαν τουλάχιστον 5 λιγότερες φορές από τον μάγειρα με τις περισσότερες συμμετοχές σε επεισόδια.

Παρόμοια με το ερώτημα όπου χρησιμοποιήσαμε την judge_appearances έχουμε δημιουργήσει το table cook_appearances και από αυτόν επιλέγουμε τον participant με τις περισσότερες εμφανίσεις. Εν συνεχεία εφαρμόζουμε join μεταξύ των μαγείρων, των συμμετεχόντων και του max_appearances και έτσι επιστρέφουμε όνομα, επώνυμο και πλήθος συμμετεχών sorted by το τελευταίο,αφότου έχουμε ελέγχξει ότι ο τρέχων συμμετέχοντας έχει τουλάχιστον 5 συμμετοχές απο τον πιο πολυεμφανιζόμενο μάγειρα.

```
with cook_appearances as (
    select cook_id, count(*) as appearances
    from participant
    group by cook_id
),
max_appearances as (
    select max(appearances) as max_appearances
    from cook_appearances
)
select c.cook_id, c.first_name, c.last_name, ca.appearances
from cooks c
join cook_appearances ca on c.cook_id = ca.cook_id
join max_appearances ma on ca.appearances <= ma.max_appearances - 5
order by ca.appearances desc;
```

Figure 14: SQL κώδικας για το ερώτημα 3.7

3.8 Σε ποιο επεισόδιο χρησιμοποιήθηκαν τα περισσότερα εξαρτήματα (εξοπλισμός);

Για αυτό το ερώτημα χρησιμοποιήσαμε εκ νέου ένα τριπλό join μεταξύ του participant, του has_to_cook και του demands έτσι ώστε με βάση το id κάθε μάγειρα-συμμετέχοντα βρίσκουμε την συνταγή που έχει μαγειρέψει. Μετά, εκτελώντας σύνδεσμο με το demands, σχέση που δηλώνει τον απαιτούμενο εξοπλισμό για την συνταγή, grouppάρουμε με βάση τον χρόνο και το επεισόδιο, τυπώνουμε σε φθίνουσα σειρά τα 4 επεισόδια με τον περισσότερο εξοπλισμό.

```
select p.serial_number, p.contest_year, COUNT(*) as gear_count
from participant p
join has_to_cook htc on p.cook_id = htc.cook_id and p.serial_number = htc.serial_number and p.contest_year = htc.contest_year
join demands d on htc.recipe_name = d.recipe_name
group by p.serial_number, p.contest_year
order by gear_count desc
limit 5;
```

Figure 15: SQL κώδικας για το ερώτημα 3.8

Εκ νέου με παρόμοιο τρόπο με το ερώτημα 3.6 δημιουργήσαμε μέσω explain το απότελεσμα του αρχικού query.

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	htc	NULL	ref	PRIMARY,idx_has_to_cook_recipe_serial_contest	idx_has_to_cook_recipe_serial_contest	203	cooking_contest.d.recipe_name	7	100.00	Using index
1	SIMPLE	d	NULL	index	PRIMARY	demands_cooking_gear	122	NULL	92	100.00	Using index;
1	SIMPLE	p	NULL	eq_ref	PRIMARY,unique_part_id_per_episode,particip...	PRIMARY	10	cooking_contest.htc.cook_id,cooking_contest.h...	1	100.00	Using index

Figure 16: Explain για το πρώτο σκέλος του 3.8

Επιπρόσθετα, δημιουργήσαμε όπως και πριν τέσσερα ευρετήρια έτσι ώστε να τα εφαρμόσουμε στο νέο query.

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	htc	NULL	ref	PRIMARY,idx_has_to_cook_recipe_serial_contest	idx_has_to_cook_recipe_serial_contest	203	cooking_contest.d.recipe_name	7	100.00	Using index
1	SIMPLE	d	NULL	index	PRIMARY	demands_cooking_gear	122	NULL	92	100.00	Using index;
1	SIMPLE	p	NULL	eq_ref	PRIMARY,unique_part_id_per_episode,particip...	PRIMARY	10	cooking_contest.htc.cook_id,cooking_contest.h...	1	100.00	Using index

Figure 17: Indexes για το ερώτημα 3.8

Ο νέος κώδικας για το ερώτημα 3.8 είναι ο εξής:

```sql
EXPLAIN
SELECT
    p.serial_number,
    p.contest_year,
    COUNT(*) AS gear_count
FROM
    participant p FORCE INDEX (idx_participant_cook_serial_contest)
JOIN
    has_to_cook htc FORCE INDEX (idx_has_to_cook_cook_serial_contest)
    ON p.cook_id = htc.cook_id AND p.serial_number = htc.serial_number AND p.contest_year = htc.contest_year
JOIN
    demands d FORCE INDEX (idx_demands_recipe_name)
    ON htc.recipe_name = d.recipe_name
GROUP BY
    p.serial_number, p.contest_year
ORDER BY
    gear_count DESC
LIMIT 5;
```

Figure 18: Νέος κώδικας για το ερώτημα 3.8

με αποτέλεσμα του explain να είναι το ακόλουθο:

```sql
EXPLAIN
SELECT
    p.serial_number,
    p.contest_year,
    COUNT(*) AS gear_count
FROM
    participant p FORCE INDEX (idx_participant_cook_serial_contest)
JOIN
    has_to_cook htc FORCE INDEX (idx_has_to_cook_cook_serial_contest)
    ON p.cook_id = htc.cook_id AND p.serial_number = htc.serial_number AND p.contest_year = htc.contest_year
JOIN
    demands d FORCE INDEX (idx_demands_recipe_name)
    ON htc.recipe_name = d.recipe_name
GROUP BY
    p.serial_number, p.contest_year
ORDER BY
    gear_count DESC
LIMIT 5;
```

Figure 19: Explain για το δεύτερο σκέλος 3.8

3.9 Λίστα με μέσο όρο αριθμού γραμμάριων υδατανθράκων στο διαγωνισμό ανά έτος;

Το συγκεκριμένο ερώτημα είναι ένα βαρύ υποερώτημα καθώς απαιτεί πενταπλό join, διότι πρέπει να βρούμε απο τους συμμετέχοντες την συνταγή που εκτέλεσαν, από εκει μέσω του ingredients να βρούμε τα συστατικά κάθε συνταγής και εν συνεχεία μέσω του relation contain έχουμε τα τελικά αποτελέσματα, τα οποία ομαδοποιούμε ανά χρονιά του διαγωνισμού.

```sql
SELECT p.contest_year, AVG(i.carbs) AS avg_carbs_per_year
FROM contain c
JOIN recipes r ON c.recipe_name = r.recipe_name
JOIN has_to_cook htc ON c.recipe_name = htc.recipe_name
JOIN participant p ON htc.serial_number = p.serial_number AND htc.contest_year = p.contest_year
JOIN ingredients i ON c.ing_name = i.ing_name
GROUP BY p.contest_year;
```

Figure 20: SQL κώδικας για το ερώτημα 3.9

3.10 Ποιες Εθνικές κουζίνες έχουν τον ίδιο αριθμό συμμετοχών σε διαγωνισμούς, σε διάστημα δύο συνεχόμενων ετών, με τουλάχιστον 3 συμμετοχές ετησίως;

Για το 10ο ερώτημα έχουμε με την αρωγή της δεσμευμένης λέξης with δημιουργούμε τον πίνακα entry_counts, για να βρούμε τις συμμετοχές κάθε εθνικής κουζίνας σε κάθε χρόνο του διαγωνισμού. Εν συνεχεία με το consecutive_years βρίσκουμε τους διαδοχικούς χρόνους. Εν τέλει, υπερωτούμε αυτούς τους προσωρινούς πίνακες και επιστρέφουμε τα consecutive έτη του διαγωνισμού με την εθνική κουζίνα, αν οι εμφάνισεις της τελευταίας είναι περισσότερες των τριών και στα δύο χρόνια και ταυτόχρονα ίσες.

```sql
WITH entry_counts AS (
    SELECT
        r.nationality,
        p.contest_year,
        COUNT(DISTINCT htc.recipe_name) AS num_entries
    FROM
        participant p
    JOIN
        has_to_cook htc ON p.serial_number = htc.serial_number AND p.contest_year = htc.contest_year
    JOIN
        recipes r ON htc.recipe_name = r.recipe_name
    GROUP BY
        r.nationality, p.contest_year
),
consecutive_years AS (
    SELECT
        ec1.nationality,
        ec1.contest_year AS year1,
        ec2.contest_year AS year2,
        ec1.num_entries AS entries_year1,
        ec2.num_entries AS entries_year2
    FROM
        entry_counts ec1
    JOIN
        entry_counts ec2 ON ec1.nationality = ec2.nationality
                        AND ec1.contest_year = ec2.contest_year - 1)
```

Figure 21: SQL κώδικας για το ερώτημα 3.10 (πρώτο σκέλος)

```sql
SELECT
    nationality,
    year1,
    year2
FROM
    consecutive_years
WHERE
    entries_year1 >= 3
    AND entries_year2 >= 3
    AND entries_year1 = entries_year2;
```

Figure 22: SQL κώδικας για το ερώτημα 3.10 (δεύτερο σκέλος

3.11 Βρείτε τους top-5 κριτές που έχουν δώσει συνολικά την υψηλότερη βαθμολόγηση σε ένα μάγειρα. (όνομα κριτή, όνομα μάγειρα και συνολικό σκορ βαθμολόγησης)

Σε αυτό το query απλώς φτιάξαμε ένα temporary table JudgeGrades στο οποίο βρήκαμε όλες τις βαθμολογίες των κριτών στους συμμετέχοντες και εν συνεχεία επιστρέφουμε τα πλήρη ονόματα κριτών και συμμετεχόντων μαζί με την πλήρη βαθμολογία τους, σε φθίνουσα σειρά. Μέσω του limit εμφαανίζουμε τις 5 πρώτες πλειάδες.

```
WITH JudgeGrades AS (
    SELECT
        j.cook_id AS judge_cook_id,
        p.cook_id AS part_cook_id,
        j.serial_number,
        j.contest_year,
        SUM(g.grade) AS total_grade
    FROM
        judge j
    JOIN
        grades g ON j.cook_id = g.judge_cook_id AND j.serial_number = g.serial_number AND j.contest_year = g.contest_year
    JOIN
        participant p ON g.part_cook_id = p.cook_id AND j.serial_number = p.serial_number AND j.contest_year = p.contest_year
    GROUP BY
        j.cook_id, p.cook_id, j.serial_number, j.contest_year
)
```

Figure 23: SQL κώδικας για το ερώτημα 3.11 (πρώτο σκέλος)

```
SELECT
    j.first_name AS judge_first_name,
    j.last_name AS judge_last_name,
    c.first_name AS cook_first_name,
    c.last_name AS cook_last_name,
    jg.total_grade
FROM
    JudgeGrades jg
JOIN
    cooks j ON jg.judge_cook_id = j.cook_id
JOIN
    cooks c ON jg.part_cook_id = c.cook_id
ORDER BY
    jg.total_grade DESC
LIMIT 5;
```

Figure 24: SQL κώδικας για το ερώτημα 3.11 (δεύτερο σκέλος

3.12 Ποιο ήταν το πιο τεχνικά δύσκολο, από πλευράς συνταγών,επεισόδιο του διαγωνισμού ανά έτος

Το ερώτημα πρώτα υπολογίζει τη μέση δυσκολία ανά επεισόδιο και, στη συνέχεια, προσδιορίζει τη μέγιστη μέση δυσκολία για κάθε έτος διαγωνισμού. Στη συνέχεια, ενώνει αυτά τα αποτελέσματα για να ανακτήσει το αντίστοιχο έτος διαγωνισμού, τον αριθμό σειράς και τη μέγιστη μέση δυσκολία για επεισόδια με τη μεγαλύτερη δυσκολία. Τέλος, το εξώτατο ερώτημα επιλέγει διακριτούς συνδυασμούς έτους διαγωνισμού και μέγιστης μέσης δυσκολίας, αφαιρώντας ουσιαστικά τυχόν διπλότυπα. Αυτή η δομημένη προσέγγιση διασφαλίζει την εξαγωγή μοναδικών και κατάλληλων πληροφοριών σχετικά με τα πιο δύσκολα επεισόδια ανά έτος διαγωνισμού.

```sql
SELECT DISTINCT contest_year, max_avg_difficulty
FROM (
    SELECT a.contest_year, a.serial_number, a.avg_difficulty AS max_avg_difficulty
    FROM (
        SELECT e.contest_year, e.serial_number, AVG(r.difficulty) AS avg_difficulty
        FROM episode e
        JOIN has_to_cook hc ON e.contest_year = hc.contest_year AND e.serial_number = hc.serial_number
        JOIN recipes r ON hc.recipe_name = r.recipe_name
        GROUP BY e.contest_year, e.serial_number
    ) AS a
    JOIN (
        SELECT contest_year, MAX(avg_difficulty) AS max_avg_difficulty
        FROM (
            SELECT e.contest_year, e.serial_number, AVG(r.difficulty) AS avg_difficulty
            FROM episode e
            JOIN has_to_cook hc ON e.contest_year = hc.contest_year AND e.serial_number = hc.serial_number
            JOIN recipes r ON hc.recipe_name = r.recipe_name
            GROUP BY e.contest_year, e.serial_number
        ) AS avg_diff_per_episode
        GROUP BY contest_year
    ) AS b ON a.contest_year = b.contest_year AND a.avg_difficulty = b.max_avg_difficulty
) AS distinct_max_avg_difficulty;
```

Figure 25: SQL κώδικας για το ερώτημα 3.12

3.13 Ποιο επεισόδιο συγκέντρωσε τον χαμηλότερο βαθμό επαγγελματικής κατάρτισης (κριτές και μάγειρες)

Το ερώτημα SQL προσδιορίζει το επεισόδιο με τη χαμηλότερη βαθμολογία κατάταξης μεταξύ όλων των επεισοδίων υπολογίζοντας πρώτα τις βαθμολογίες κατάταξης των μαγείρων (τόσο από τους πίνακες κριτών όσο και από τους πίνακες συμμετεχόντων) με βάση τις βαθμολογίες τους (αντιστοιχίζοντας βαθμολογίες από 5 για τον chef σε 1 για τον c_cook) . Στη συνέχεια, συνδυάζει αυτές τις βαθμολογίες και τις ομαδοποιεί κατά serial_number και contest_year για να βρει την ελάχιστη βαθμολογία κατάταξης για κάθε επεισόδιο. Το κύριο ερώτημα ενώνει αυτό το αποτέλεσμα με τον πίνακα επεισοδίων για να διασφαλίσει ότι κάθε επεισόδιο συσχετίζεται με την ελάχιστη βαθμολογία κατάταξης, ταξινομεί τα επεισόδια με αυτές τις βαθμολογίες σε αύξουσα σειρά και περιορίζει την έξοδο σε μία σειρά, επιλέγοντας έτσι το επεισόδιο με την απόλυτη χαμηλότερη βαθμολογία κατάταξης.

```sql
SELECT
    e.serial_number,
    e.contest_year,
    MIN(score) AS min_rank_score
FROM
    episode e
JOIN (
    SELECT
        serial_number,
        contest_year,
        MIN(rank_score) AS score
    FROM
        (
            SELECT
                j.serial_number,
                j.contest_year,
                c.cook_id,
                (
                    CASE
                        WHEN c.ranks = 'chef' THEN 5
                        WHEN c.ranks = 'sous_chef' THEN 4
                        WHEN c.ranks = 'a_cook' THEN 3
                        WHEN c.ranks = 'b_cook' THEN 2
                        WHEN c.ranks = 'c_cook' THEN 1
                    END
                ) AS rank_score
```

Figure 26: SQL κώδικας για το ερώτημα 3.13 (πρώτο σκέλος)

```sql
            FROM
                judge j
            JOIN cooks c ON j.cook_id = c.cook_id
        UNION ALL
            SELECT
                p.serial_number,
                p.contest_year,
                c.cook_id,
                (
                    CASE
                        WHEN c.ranks = 'chef' THEN 5
                        WHEN c.ranks = 'sous_chef' THEN 4
                        WHEN c.ranks = 'a_cook' THEN 3
                        WHEN c.ranks = 'b_cook' THEN 2
                        WHEN c.ranks = 'c_cook' THEN 1
                    END
                ) AS rank_score
            FROM
                participant p
            JOIN cooks c ON p.cook_id = c.cook_id
        ) AS combined_scores
    GROUP BY
        serial_number,
        contest_year
) AS min_scores ON e.serial_number = min_scores.serial_number AND e.contest_year = min_scores.contest_year
GROUP BY
    e.serial_number,
    e.contest_year
ORDER BY
    min_rank_score ASC
LIMIT 1;
```

Figure 27: SQL κώδικας για το ερώτημα 3.13 (δεύτερο σκέλος

3.14 Ποια θεματική ενότητα έχει εμφανιστεί τις περισσότερες φορές στο διαγωνισμό;

Με τον όρο with φτιάχνουμε ένα sub query, με όνομα temp, το οποίο περιέχει στην πρώτη στήλη τα ονόματα όλων των θεματικών ενοτήτων που έχουν εμφανιστεί στον διαγωνισμό και στην δεύτερη στήλη το πόσες φορές έχουν αυτές εμφανιστεί. Έπειτα από τον πίνακα temp, κρατάμε τα ονόματα μόνο εκείνων των θεματικών ενοτήτων που έχουν εμφανιστεί περισσότερες φορές, χρησιμοποιώντας στο where clause τον όρο $\geq$ all. Το sub query του all μόνο την στήλη με το πόσες φορές εμφανίστηκε κάθε θεματική ενότητα.

```
with temp as (
select theme_name,count(theme_name) as frequency
from has_to_cook h join belongs_to b on h.recipe_name = b.recipe_name
group by theme_name
)
select theme_name
from temp
where frequency >= all (
select frequency
from temp
);
```

Figure 28: SQL κώδικας για το ερώτημα 3.14

3.15 Ποιες ομάδες τροφίμων δεν έχουν εμφανιστεί ποτέ στον διαγωνισμό;

Για αυτό το query χρειαζόμαστε την πληροφορία για το ποιες συνταγές έχουν εμφανιστεί στον διαγωνισμό, επομένως κάνουμε ένα join μεταξύ των σχέσεων has_to_cook και recipes. Έπειτα πρέπει να κάνουμε join μεταξύ αυτού του αποτελέσματος με την contain για να πάρουμε όλα τα υλικά αυτών των συνταγών. Δηλαδή όλα τα υλικά που έχουν εμφανιστεί στον διαγωνισμό. Και τέλος πρέπει να κάνουμε join το τελευταίο αποτέλεσμα με την σχέση has_group για να βρούε σε ποια ομάδα τροφίμων ανήκουν αυτά τα υλικά. Το τελικό αποτέλεσμα είναι το sub query στο all που έχουμε βάλει στο where clause. Και με την βοήθεια το τελεστή != βρίσκουμε τις ομάδες τροφήμων που δεν ανήκουν σε αυτό το σύνολο και άρα που δεν έχουν εμφανιστεί στον διαγωνισμό.

```
select group_name
from food_group f
where f.group_name != all (
select g.group_name
from ((has_to_cook h join recipes r on h.recipe_name = r.recipe_name)
join contain c on h.recipe_name = c.recipe_name)
join has_group g on g.ing_name = c.ing_name
```

Figure 29: SQL κώδικας για το ερώτημα 3.15

4 Εφαρμογή

Στην τελευταία αυτή ενότητα της αναφοράς θα συζητήσουμε ορισμένες παραδοχές που κάναμε κατά την εργασία μας, θα δόσουμε οδηγίες χρήσης της εφαρμογής αλλά και το github repo μας.

4.1 Παραδοχές

Πρώτα πρώτα πρέπει να αναφέρουμε τον τρόπο με τον οποίο καταγράφουμε τις ποσότητες των υλικών και υπολογίζουμε τις θερμίδες των γευμάτων. Το attribute του contain quantity ουσιαστικά εκφράζει την ποσότητα κάθε υλικού σε γραμμάρια. Προφανώς τα γραμμάρια αντιστοιχούν σε ml για τα υγρά, ενώ ένα μήλο για παράδειγμα λογίζεται ως ένα τεμάχιο. Για

αυτό και κατά τον υπολογισμό των θερμίδων έχουμε απλό πολλαπλασιασμό με τις θερμίδες ανά γραμμάριο των λιπών (x9), υδατανθράκων (x4) και πρωτεϊνών (x4).

4.2 Εγκατάσταση / Οδηγίες

Η εφαρμογή μας βασίζεται πάνω στο web development stack xampp. Επομένως για να εγκαταστήσει κανείς την εφαρμογή μας αρκεί να ακολουθήσει τις οδηγίες εγκατάστασης του xampp από τις διαφάνειες του εργαστηρίου του μαθήματος. Κατόπιν, πρέπει να γίνει source των αρχείων που υπάρχουν στο github repository μας. Αυτό βρίσκεται στον σύνδεσμο: GitRepo.