Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Εργαστήριο Βάσεων Δεδομένων

Εξαμηνιαία Εργασία:

Ανάπτυξη Βάσης Δεδομένων για Διαγωνισμό Μαγειρικής

Ομάδα Project: 102

Αναγνώστου Νικόλαος: 03121818

Μάκρας Ηλίας: 03121207

Μαρκουλιδάκης Γεώργιος: 03121050

Περιεχόμενα

1. Σχεδίαση της Βάσης

1.1 Εφαρμογές που χρησιμοποιήθηκαν

Η εργασία μας σχεδιάστηκε και υλοποιήθηκε με τη χρήση της MySQL, μέσω του client MySQL Workbench για την εκτέλεση των SQL scripts που δημιουργούν τη βάση και εκτελούν την κλήρωση του διαγωνισμού, καθώς και των queries.

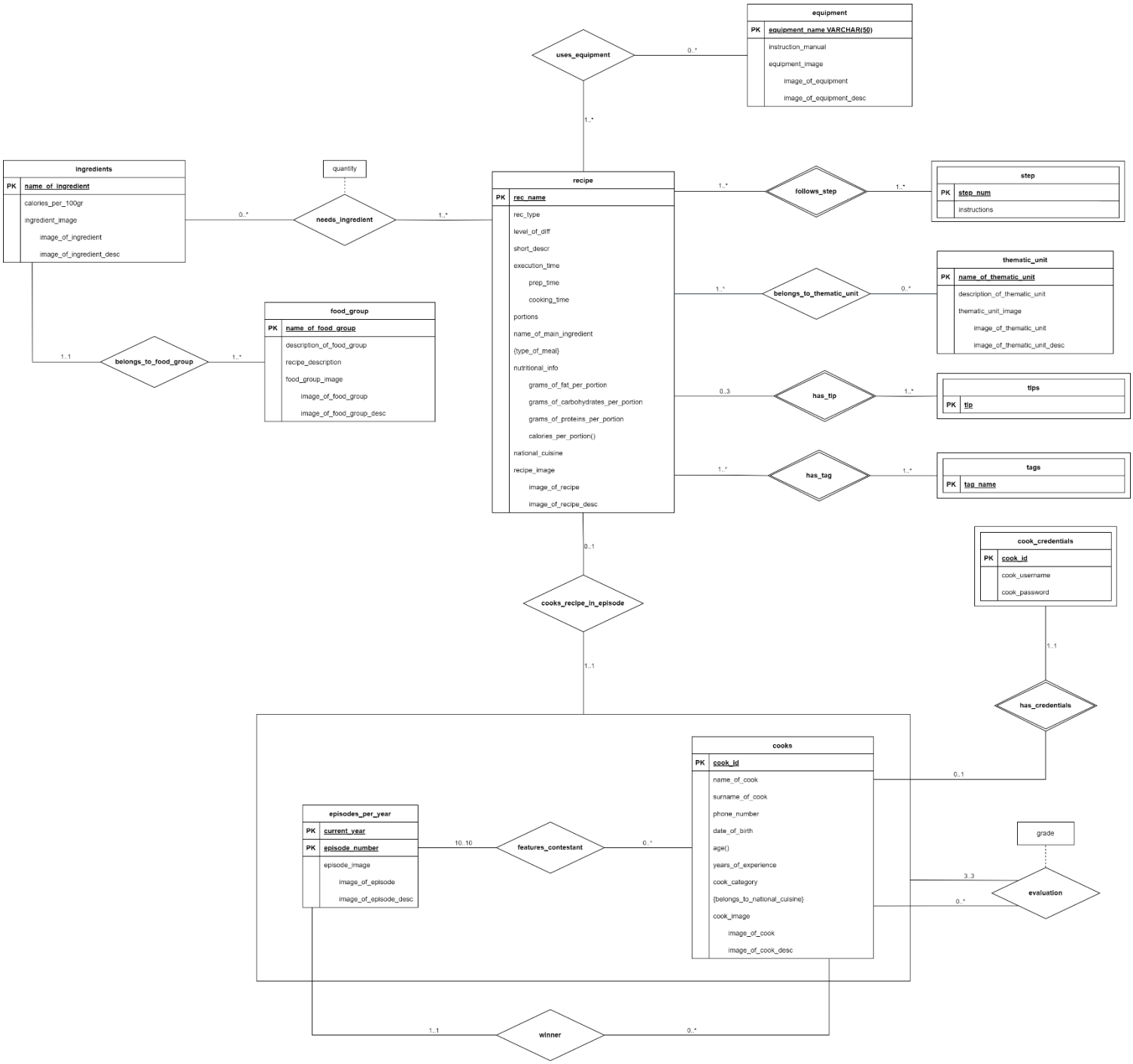
Τα δεδομένα που εισήχθησαν στην βάση δημιουργήθηκαν με κώδικα Python που περιλαμβάνει τις βιβλιοθήκες faker και faker_food, οι οποίες λειτούργησαν ως γεννήτριες dummy data.

Το διάγραμμα ER σχεδιάσθηκε μέσω του εργαλείου draw.io

Όλα τα μέρη της εργασίας και οι υλοποιήσεις τους βρίσκονται στο σύνδεσμο για το GitHub repository: *https://github.com/nikosanag/DB_Project*

1.2 Διάγραμμα ER

Για την ορθή σχεδίαση της βάσης μας δημιουργήσαμε την αναπαράσταση των δεδομένων σε μοντέλο οντοτήτων-συσχετίσεων (ER). Το μοντέλο που σχεδιάσαμε φαίνεται στην ακόλουθη εικόνα:



Η εικόνα φαίνεται καλύτερα στο αντίστοιχο αρχείο στο GitHub Repository της εργασίας με όνομα αρχείου *ER-Diagram.png*. Το αντίστοιχο αρχείο που αποθηκεύσαμε από το draw.io είναι το *ER-Diagram.drawio.crswap*.

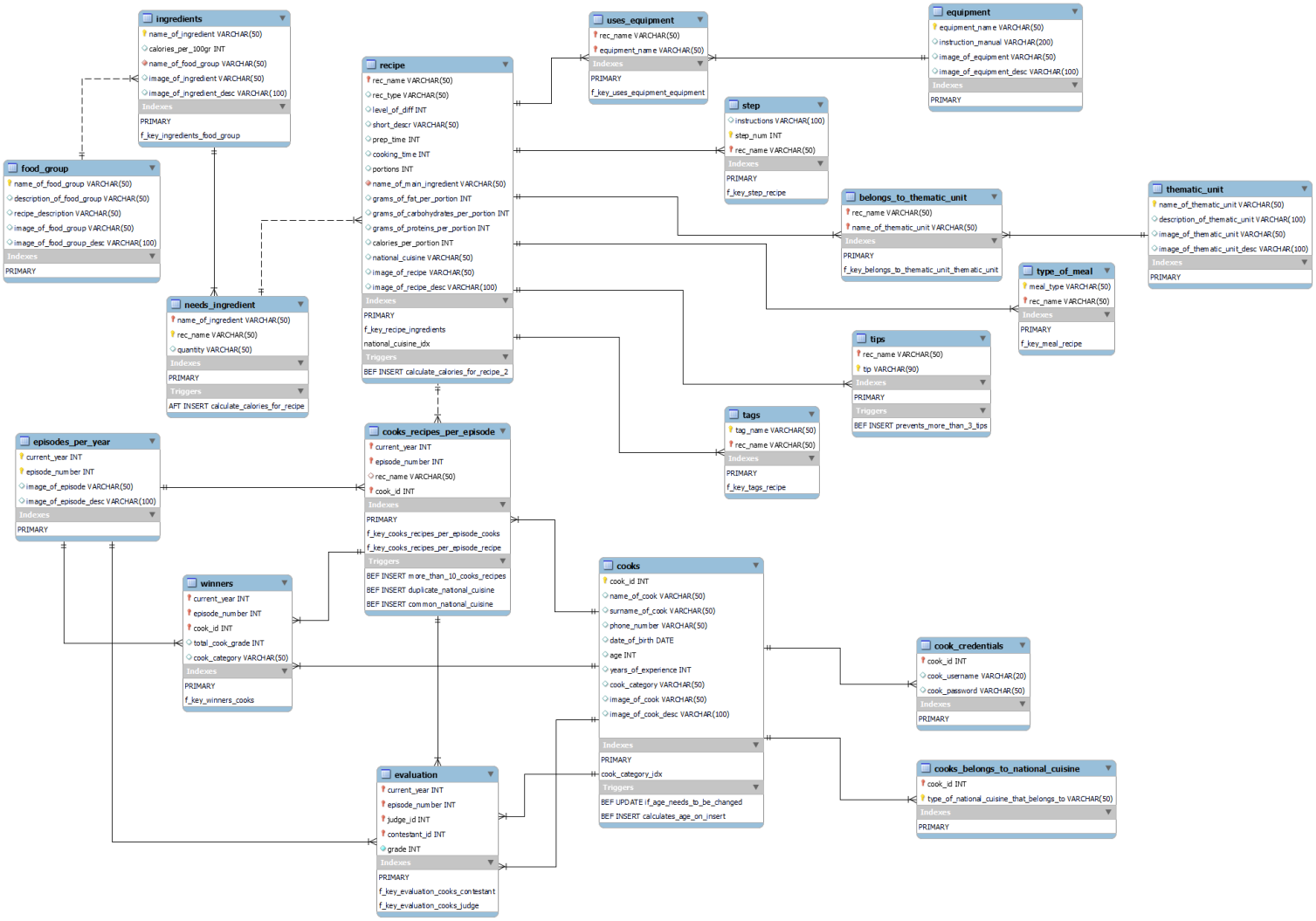
Στο παραπάνω διάγραμμα έχουμε επιλέξει να χρησιμοποιήσουμε σε όλες τις σχέσεις τους συμβολισμούς άνω και κάτω ορίου, για λόγους συνέπειας. Για τη δημιουργία του διαγράμματος έχουμε λάβει τις εξής παραδοχές:

- Οι οντότητες «thematic_unit», «equipment» και «ingredients» είναι πιθανό να μη συσχετίζονται με κάποια συνταγή.
- Οι μάγειρες δεν σχετίζονται άμεσα με τις συνταγές που μπορούν να εκτελέσουν, αλλά μπορούν να εκτελέσουν οποιαδήποτε συνταγή, αρκεί αυτή να ανήκει σε κάποια εθνική κουζίνα στην οποία αυτοί είναι εξειδικευμένοι. Επειδή η εθνική κουζίνα δεν αποτελεί κάποια οντότητα στη μοντελοποίησή μας, η πληροφορία αυτή δεν φαίνεται στο διάγραμμα ER.
- Όπως θα αναφερθεί και παρακάτω, η οντότητα «cook_credentials» χρησιμοποιείται για να συμβολίσει έναν πίνακα όπου θα αποθηκεύονται το username και το password κάθε μάγειρα και θα συνδέονται με το cook_id του. Ωστόσο, για τους σκοπούς παρουσίασης της εργασίας, έχουμε αποθηκεύσει μία μόνο γραμμή στον πίνακα αυτόν, η οποία αντιστοιχεί στις πληροφορίες ενός μόνο μάγειρα.

Αξίζει να αναφερθεί ότι, όπως φαίνεται και στο διάγραμμα και, σύμφωνα με τη μέθοδο aggregation , έχουμε θεωρήσει μία οντότητα η οποία περιλαμβάνει τις οντότητες «episodes_per_year» και «cooks», μαζί με τη σχέση «features_contestant» που τις συνδέει.

1.3 Σχεσιακό Διάγραμμα

Το σχεσιακό διάγραμμα που προκύπτει από τη σχεδίαση της βάσης μας είναι το παρακάτω:



Το αρχείο της εικόνας καθώς και το model που φτιάξαμε στο MySQL Workbench βρίσκεται στο repository της εργασίας, με ονόματα *Relational_Image.png* και *Relational.mwb* αντίστοιχα.

Στο διάγραμμα φαίνονται και τα ευρετήρια που χρησιμοποιεί κάθε πίνακας, καθώς και τα triggers που φτιάξαμε για καθέναν από αυτούς, για τα οποία θα μιλήσουμε στο μέρος 3.

2. Υλοποίηση της Βάσης

2.1 DDL

Η δημιουργία της βάσης γίνεται στο MySQL script με όνομα *Create_Cooking_DB* (βρίσκεται στο repository). Σε αυτό το script δημιουργούνται όλα τα tables, τα triggers και τα indexes που υλοποιούν τη βάση. Θα περιγράψουμε πιο αναλυτικά την λειτουργία των triggers και των indexes στο μέρος 3.

Σε όλα τα tables έχουν οριστεί τα κατάλληλα constraints, ώστε να εξασφαλίζεται η ορθότητα της βάσης δεδομένων. Τα constraints αυτά είναι τα εξής: Primary Keys, Foreign Keys, χρήση CHECK για έλεγχο ακεραιότητας του πεδίου τιμών, περιορισμός NOT NULL.

Όπως αναφέραμε και προηγουμένως, θεωρούμε ότι ο κάθε μάγειρας μπορεί να εκτελέσει αποκλειστικά όλες τις συνταγές που ανήκουν σε εθνική κουζίνα στην οποία αυτός εξειδικεύεται. Για τον λόγο αυτό δεν ορίζουμε στο DDL κάποιον πίνακα που να συνδέει τους μάγειρες με τις συνταγές που μπορούν να τους ανατεθούν. Αξίζει να σημειώσουμε πως, λαμβάνοντας υπ' όψιν την περίπτωση δύο μάγειρες να έχουν ίδιο ονοματεπώνυμο, έχουμε ορίσει το attribute «cook_id», το οποίο είναι μοναδικό για κάθε μάγειρα και αποτελεί το Primary Key του πίνακα «cooks».

Το δεύτερο μέρος του DDL απαιτεί την εκτέλεση του script με όνομα *User_Privileges.sql*. Αυτό το script καθορίζει τα δικαιώματα χειρισμού της βάσης που έχουν τόσο οι μάγειρες όσο και οι διαχειριστές. Στο script δημιουργούνται δύο νέοι ρόλοι, ένας που αφορά τους μάγειρες και ένας άλλος που αφορά τον/τους διαχειριστή/διαχειριστές.

- **Ρόλος «cook_role»**

Είναι ο ρόλος που αντιστοιχεί στον μάγειρα. Έχουμε θεωρήσει ότι ο μάγειρας έχει δικαίωμα να δει ανά πάσα στιγμή τα περιεχόμενα των πινάκων της βάσης (GRANT SELECT) για όλους τους πίνακες, εκτός από τον πίνακα που περιέχει τα usernames και passwords των άλλων μαγείρων. Έχουμε επίσης δώσει σε κάθε μάγειρα το δικαίωμα να προσθέτει νέες συνταγές στον πίνακα «recipe». Επιπλέον, έχουμε φροντίσει ώστε κάθε μάγειρας να μπορεί να κάνει UPDATE μόνο τις συνταγές που έχουν ανατεθεί στον ίδιο. Για να το πετύχουμε αυτό δημιουργούμε ένα view, το οποίο περιέχει τις συνταγές που έχουν ανατεθεί στον μάγειρα που είναι συνδεδεμένος από την αρχή του διαγωνισμού. Στο view αυτό παραχωρούμε το UPDATE privilege, ώστε να δώσουμε στον μάγειρα τη δυνατότητα επεξεργασίας του.

Όπως προαναφέραμε, για τους σκοπούς της παρουσίασης της λετουργικότητας της εφαρμογής έχουμε συμπεριλάβει στον πίνακα «cook_credentials» έναν μόνο μάγειρα με cook_id = 7 και username = “Cook”. Ωστόσο η λειτουργικότητα μπορεί να επεκταθεί για κάθε αριθμό μαγείρων, αν φυσικά δημιουργηθούν οι κατάλληλες συνδέσεις και δοθεί στους αντίστοιχους χρήστες ο ρόλος «cook_role».

- **Ρόλος «administrator_role»**

Είναι ο ρόλος που έχει κάθε διαχειριστής. Στον ρόλο αυτό παραχωρούνται δικαιώματα που επιτρέπουν τη διαχείριση όλων των δεδομένων. Ένα πολύ βασικό κομμάτι των δικαιωμάτων του διαχειριστή είναι η παραχώρηση των απαιτούμενων privileges προκειμένου να δημιουργείται backup της βάσης, μέσω της εντολής mysqldump από το τερματικό. Η χρήση της εντολής αυτής πρϋποθέτει το να έχουμε εγκατεστημένο το mysqldump (περιέχεται στο installation του MySQL Server). Χρησιμοποιήσαμε λοιπόν την εντολή GRANT για να παραχωρήσουμε στον διαχειριστή κάθε δικαίωμα που περιγράφεται στο documentation page της εντολής (*https://dev.mysql.com/doc/refman/8.0/en/mysqldump.html*) ως απαραίτητο για την δημιουργία backup. Για το κομμάτι του restore χρειάστηκαν επιπλέον τα δικαιώματα: DROP, REFERENCES, CREATE VIEW , SUPER και SYSTEM_USER. Με τα δικαιώματα αυτά ο διαχειριστής μπορεί εύκολα να δημιουργήσει backup της βάσης δεδομένων καθώς και να ανακτήσει τη βάση από αυτό, μέσω του τερματικού.

Σημειώνουμε πως σε παλαιότερες εκδόσεις του MySQL Workbench παρατηρήθηκε ένα πρόβλημα συμβατότητας, καθώς δεν αναγνωριζόταν η εντολή DEFAULT ROLE (που χρησιμοποιείται κατά τη δημιουργία χρήστη), συνεπώς το script δεν θα τρέχει στις εκδόσεις αυτές. Για να διορθώσουμε το πρόβλημα θα μπορούσαμε να δημιουργήσουμε τον χρήστη και, έπειτα, να του δώσουμε τον ρόλο με χρήση της εντολής GRANT.

2.2 DML

Το πρώτο κομμάτι του DML είναι η εισαγωγή των δεδομένων στους πίνακες που δημιουργήθηκαν στο DDL. Όλες οι εισαγωγές που δεν αφορούν τον διαγωνισμό πραγματοποιούνται με την εκτέλεση του script με όνομα *Insertions.sql*. Το script εισάγει στη βάση δεδομένων τα εξής στοιχεία με τα απαιτούμενα attributes τους:

- 250 υλικά (ingredients)
- 12 ομάδες τροφίμων (food groups)
- 100 μάγειρες (cooks)
- 200 συνταγές (recipes)

- 50 στοιχεία εξοπλισμού/εξαρτήματα (equipment)
- 30 θεματικές ενότητες (thematic units)
- 30 εθνικές κουζίνες (national cuisines)

Όπως προαναφέραμε η παραγωγή του script έγινε μέσω Python με τη χρήση των βιβλιοθηκών faker και faker_food. Το αρχείο κώδικα Python που παράγει τα insertions βρίσκεται στο repository και ονομάεται *dummy_generator.py*. Οι ζητούμενες εικόνες έχουν εισαχθεί σε μορφή συνδέσμου.

Τονίζουμε ότι τα δεδομένα δεν έχουν δημιουργηθεί για να ταιριάζουν μεταξύ τους ή για να μας πληροφορούν για τις συνταγές, αλλά για να αποτελέσουν ένα θεμέλιο στη διαχείριση της βάσης και στην δημιουργία/εκτέλεση των queries. Συνεπώς πολλές περιγραφές συνταγών, εικόνων και άλλων στοιχείων έχουν εισαχθεί ως τυχαίες προτάσεις, τα links των εικόνων οδηγούν σε dummy εικόνες κ.λ.π.

Το δεύτερο κομμάτι του DML είναι ο διαγωνισμός/κλήρωση. Το αρχείο έχει όνομα Build_Contest. Σε αυτό, σύμφωνα με τις οδηγίες της εκφώνησης, εκτελείται ανάθεση μιας συνταγής σε κάθε έναν από τους 10 μάγειρες που συμμετέχουν στο επεισόδιο. Η κλήρωση υλοποιείται μέσω ενός stored procedure με όνομα build_contest. Η διαδικασία δέχεται ως ορίσματα δύο integers, το πρώτο και το τελευταίο έτος του διαγωνισμού. Αφού η διαδικασία της κλήρωσης οριστεί, αυτή καλείται με έτος εκκίνησης 2020 και έτος λήξης 2024. Εάν ο χρήστης επιθυμεί να κάνει νέα κλήρωση, αρκεί να καλέσει τη διαδικασία με τα έτη της επιλογής του. Η τυχαιότητα εξασφαλίζεται μέσω του ORDER BY RAND() σε διάφορα queries στη διάρκεια της κλήρωσης. Αφού εκτελεστεί η κλήρωση, ο πίνακας «episodes_per_year» ενημερώνεται με εισαγωγή των εικόνων των επεισοδίων και των περιγραφών τους. Το κομμάτι κώδικα που υλοποιεί τις εισαγωγές αυτές παράχθηκε με Python (αντίστοιχα με τα insertions) και βρίσκεται στο repository της εργασίας με όνομα *dummy_episode_image.py*. Σημειώνουμε ότι οι ενημερώσεις αυτές του πίνακα είναι σωστές μόνο για επεισόδια που αντιστοιχούν στα επεισόδια των ετών 2020-2024. Σε περίπτωση εκ νέου εκτέλεσης της κλήρωσης θα πρέπει να εκτελεστεί ξανά και η εισαγωγή των εικόνων.

Για την κλήρωση κάναμε τις παρακάτω παραδοχές:

- Δεν επιτρέπεται ένας μάγειρας να συμμετέχει στο ίδιο έτος πάνω από 3 συνεχόμενες φορές, ακόμα και αν εναλλάσσεται ο ρόλος του από μάγειρα σε κριτή. Αν γινόταν αυτό θεωρητικά θα μπορούσε ένας μάγειρας να συμμετέχει σε όλα τα επεισόδια ενός έτους με εναλλασσόμενο ρόλο.
- Επιπλέον θεωρούμε ότι το σερί συμμετοχών μεταφέρεται και σε επόμενα έτη. Δηλαδή αν για παράδειγμα ένας διαγωνιζόμενος είχε εμφανιστεί στα 3 τελευταία επεισόδια ενός έτους, τότε δεν μπορεί να συμμετάσχει στο πρώτο επεισόδιο του επόμενου έτους. Αντίστοιχα αν έχει συμμετάσχει

στα 2 τελευταία επεισόδια ενός έτους και συμμετάσχει και στο πρώτο του επόμενου έτους, δε θα μπορεί να συμμετέχει στο δεύτερο επεισόδιο.

- Θεωρούμε ότι για να είναι ένας μάγειρας πιθανός κριτής, πρέπει να είναι της κατηγορίας «Σεφ». Θα ήταν παράλογο για παράδειγμα, εάν ένας μάγειρας της κατηγορίας «Α Μάγειρας» αξιολογούσε έναν «Σεφ». Η διαδικασία της κλήρωσης για αυτόν τον λόγο αλλάζει μόνο στο σημείο της επιλογής των κριτών, όπου προστίθεται απλά η συνθήκη ο υποψήφιος μάγειρας κριτής να είναι «Σεφ».
- Για να καθορίσουμε τον νικητή θεωρούμε αθροιστικό σύστημα βαθμολόγησης. Αυτό σημαίνει ότι ο νικητής σε κάθε επεισόδιο καθορίζεται, σε πρώτη φάση, με βάση το άθροισμα των τριών βαθμών που έχει λάβει ο κάθε μάγειρας από τους κριτές.

2.3 Queries

Τα ζητούμενα ερωτήματα (queries) υλοποιούνται στο αρχείο του GitHub repository της εργασίας με όνομα *Queries.sql*. Στο DDL δημιουργήθηκαν αρκετά δεδομένα έτσι ώστε όλα τα queries να επιστρέφουν στην μεγάλη πλειοψηφία των περιπτώσεων αποτέλεσμα, με ορισμένες παραδοχές τις οποίες θα αναφέρουμε παρακάτω. Ωστόσο, λόγω της τυχαιότητας της κλήρωσης, υπάρχει πάντα το ενδεχόμενο το query να επιστρέψει κενό αποτέλεσμα, κάτι το οποίο δεν είναι στον έλεγχό μας. Σημειώνουμε ότι τα εναλλακτικά Query Plans αναλύονται στο 3ο μέρος της αναφοράς.

Για τη σύνταξη των queries έγιναν οι παρακάτω παραδοχές:

- Στα queries 3.2 και 3.7 θεωρούμε ότι οι μάγειρες που συμμετείχαν στα επεισόδια είναι διαγωνιζόμενοι και όχι κριτές.
- Στο query 3.2 βρίσκουμε μόνο τους μάγειρες που ανήκουν σε μία εθνική κουζίνα και την αντιπροσώπευσαν σε ένα επεισόδιο, όχι απλά εάν ανήκουν σε αυτήν. Εάν θέλαμε να βρούμε απλά εάν ανήκουν σε αυτήν, η λογική του query θα ήταν η ίδια, απλά θα κάναμε JOIN το table cooks_belongs_to_national_cuisine αντί του recipe με την αντίστοιχη συνθήκη στο JOIN και στο WHERE.
- Στο query 3.3 βρίσκουμε τους μάγειρες που έχουν εκτελέσει τις περισσότερες συνταγές στη διάρκεια του διαγωνισμού και είναι κάτω των 30 ετών.
- Στο query 3.8 θεωρήσαμε σωστό να μετρηθούν το πόσα διαφορετικά εξαρτήματα χρησιμοποιούνται σε κάθε επεισόδιο, καθώς δε διευκρινίζεται σύμφωνα με την εκφώνηση η ποσότητα από κάθε εξάρτημα που χρειάζεται κάθε συνταγή.

- Στο query 3.11 βρίσκουμε τους top 5, ως προς συνολικό (average) βαθμό, συνδυασμούς κριτή-διαγωνιζόμενου. Είναι πιθανό ο ίδιος κριτής ή ο ίδιος διαγωνιζόμενος να βρίσκονται πάνω από 1 φορά στην πεντάδα.
- Στο query 3.12 συγκρίνουμε τους μέσους όρους (όχι το άθροισμα ενδεχομένως) δυσκολίας των συνταγών των επεισοδίων για να συγκρίνουμε τη δυσκολία των επεισοδίων.
- Στο query 3.13 δώσαμε αυθαίρετα συγκεκριμένη βαρύτητα σε κάθε κατηγορία μάγειρα:
    - ❖ 1 στον Γ Μάγειρα
    - ❖ 2 στον Β Μάγειρα
    - ❖ 3 στον Α Μάγειρα
    - ❖ 4 στον Βοηθό Σεφ
    - ❖ 5 στον Σεφ

Στη συνέχεια συγκρίναμε το άθροισμα των βαρών αυτών για κάθε επεισόδιο.

- Στο query 3.14 μετράμε ξεχωριστά την κάθε εμφάνιση της κάθε θεματικής ενότητας στο ίδιο επεισόδιο. Αν δεν το θέλουμε αυτό μπορούμε να προσθέσουμε DISTINCT σε ένα συγκεκριμένο subquery (βλέπε σχόλια κώδικα).
- Στο query 3.15 θεωρούμε ότι μια ομάδα τροφίμων εμφανίζεται σε ένα επεισόδιο, εάν μία συνταγή του επεισοδίου έχει ως κύριο υλικό, ένα υλικό που ανήκει στη συγκεκριμένη ομάδα τροφίμων.

3. Λεπτομέρειες Υλοποίησης

3.1 Triggers

Η χρησιμότητα των triggers που δημιουργήθηκαν περιγράφεται παρακάτω:

- **calculates_age_on_insert**
  Υπολογίζει την ηλικία ενός μάγειρα που εισάγεται στη βάση δυναμικά με βάση την ημερομηνία γέννησής του .
- **if_age_needs_to_be_changed**
  Υπολογίζει την ηλικία του μάγειρα σε περίπτωση που γίνει κάποιο update σε κάποιον μάγειρα, έτσι ώστε αν είχε γίνει λάθος στην ημερομηνία γέννησης του να μπορεί να διορθώνεται δυναμικά και η ηλικία του.
- **prevents_more_than_3_tips**
  Εξασφαλίζει ότι δεν πρόκειται να εισαχθούν περισσότερα από 3 tips, δηλαδή χρηστικές συμβουλές, για την ίδια συνταγή. Σε περίπτωση που αυτό συμβεί θα προκαλέσει error.
- **calculate_calories_for_recipe_on_insert_recipe**

Υπολογίζει τις θερμίδες ανά μερίδα για τη συνταγή που εισάγεται στη βάση, με βάση τις θερμίδες και τις αναλογίες των υλικών που χρειάζεται και που έχουν ήδη εισαχθεί.

- **calculate_calories_for_recipe_on_insert_needs_ingredient**
  Ανανεώνει τις θερμίδες ανά μερίδα για μία συνταγή εάν προστεθεί κάποιο νέο υλικό που χρειάζεται η συνταγή.
- **common_national_cuisine**
  Εξασφαλίζει ότι δεν πρόκειται να εισαχθεί σε κάποιο επεισόδιο συνδυασμός μάγειρα και συνταγής που δεν έχουν κοινή εθνική κουζίνα. Εάν συμβεί αυτό, τότε θα προκληθεί error.
- **duplicate_national_cuisine**
  Εξασφαλίζει ότι δεν πρόκειται να εισαχθούν σε κάποιο επεισόδιο δύο συνταγές που ανήκουν στην ίδια εθνική κουζίνα. Εάν αυτό συμβεί θα προκληθεί error.
- **more_than_10_cooks_recipes**
  Εξασφαλίζει ότι δεν πρόκειται να εισαχθούν στο ίδιο επεισόδιο πάνω από 10 συνδυασμοί συνταγών και μαγείρων. Θα προκαλέσει error στην περίπτωση που αυτό συμβεί.

3.2 Indexes/Ευρετήρια

Όπως γνωρίζουμε, για όποια attributes έχουν καθοριστεί ως κλειδιά, είτε Primary Keys είτε Foreign Keys, δημιουργούνται αυτόματα και indexes από τη MySQL. Όπως είναι κατανοητό, τα περισσότερα queries θα βασίζονται σε αυτά τα attributes και συνεπώς δε θα απαιτούν επιπρόσθετα indexes. Υπάρχουν όμως κάποια queries που θα μπορούσαν να επωφεληθούν από τη χρήση κάποιων indexes. Συγκεκριμένα στο query 3.2 έχουμε τη συνθήκη: national_cuisine = 'Ζητούμενη εθνική κουζίνα'.

```
16    -- 3.2
17 •  SELECT DISTINCT CONCAT(name_of_cook,' ',surname_of_cook) 'Cook name',
18                          national_cuisine 'National Cuisine',
19                          current_year 'Year of the episode'
20    FROM cooks
21    JOIN cooks_recipes_per_episode USING (cook_id)
22    JOIN recipe USING (rec_name)
23    WHERE current_year=2020
24    AND national_cuisine='Mordovian' -- This condition is to check if the cook actually represents this national cuisine on an episode.
25    ;
26
```

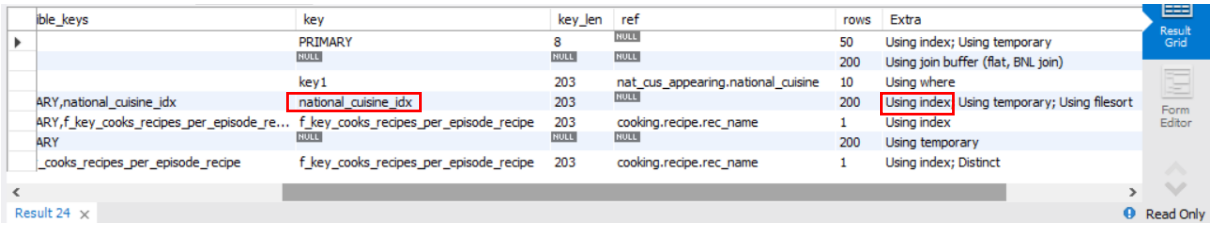
Τα indexes είναι κατάλληλα για τέτοιες συνθήκες αναζήτησης και συνεπώς θα ήταν καλή πρακτική να ορίσουμε ένα index για το table recipe στο attribute national_cuisine.

```
CREATE INDEX national_cuisine_idx ON recipe(national_cuisine);
```

Το συγκεκριμένο index χρησιμοποιείται και στο query 3.10, στο πρώτο subquery, ως συνθήκη JOIN.

```
117     -- Store the number of apps of each national cuisine for each year.
118     -- If a cuisine does not appear in a year, then number of apps will be null.
119 •   CREATE TEMPORARY TABLE nat_cus_year_apps
120     SELECT DISTINCT current_year, national_cuisine, Number_of_apps
121     FROM episodes_per_year
122     JOIN (
123         SELECT DISTINCT national_cuisine
124         FROM cooks_recipes_per_episode
125         JOIN recipe USING(rec_name)
126     ) nat_cus_appearing -- creates a table with every possible combination (year,national cuisine that has ever appeared).
127     LEFT JOIN (
128         SELECT current_year, national_cuisine, COUNT(rec_name) Number_of_apps
129         FROM(
130             SELECT current_year, rec_name, national_cuisine
131             FROM cooks_recipes_per_episode
132             JOIN recipe USING(rec_name)
133         ) nat_cus_rec_per_year
134         GROUP BY current_year, national_cuisine
135     ) apps_of_nat_cus_per_year USING (current_year, national_cuisine) -- This subquery finds the cuisines that appear each year
136                                                                       -- and their apps that year.
137     ;
```

Όπως μπορούμε να δούμε με τη βοήθεια της εντολής EXPLAIN, ο optimizer της SQL επιλέγει να χρησιμοποιήσει το συγκεκριμένο index.



ible_keys	key	key_len	ref	rows	Extra
	PRIMARY	8	NULL	50	Using index; Using temporary
	NULL	NULL	NULL	200	Using join buffer (flat, BNL join)
	key1	203	nat_cus_appearing.national_cuisine	10	Using where
ARY,national_cuisine_idx	national_cuisine_idx	203	NULL	200	Using index; Using temporary; Using filesort
ARY,f_key_cooks_recipes_per_episode_re...	f_key_cooks_recipes_per_episode_recipe	203	cooking.recipe.rec_name	1	Using index
ARY	NULL	NULL	NULL	200	Using temporary
_cooks_recipes_per_episode_recipe	f_key_cooks_recipes_per_episode_recipe	203	cooking.recipe.rec_name	1	Using index; Distinct

Result 24 ×

Result Grid

Form Editor

Read Only

Επιπλέον, στο query 3.13 κάνουμε JOIN χρησιμοποιώντας το attribute cook_category του cook.

```sql
WITH level_of_eps AS (
    SELECT current_year, episode_number, SUM(level_of_cook) level_of_episode
    FROM(
        SELECT current_year, episode_number, level_of_cook
        FROM(
            SELECT current_year, episode_number,cook_id, cook_category
            FROM cooks_recipes_per_episode
            JOIN cooks USING (cook_id)
            UNION ALL
            SELECT DISTINCT current_year, episode_number, judge_id,cook_category
            FROM evaluation
            JOIN cooks ON cook_id=judge_id
        ) cooks_categories -- This subquery finds the cook categories each episode contains (both judges and contestants)
        JOIN(
            SELECT 1 level_of_cook, 'C Cook' cook_category
            UNION
            SELECT 2 level_of_cook, 'B Cook' cook_category
            UNION
            SELECT 3 level_of_cook, 'A Cook' cook_category
            UNION
            SELECT 4 level_of_cook, "Chef's Assistant" cook_category
            UNION
            SELECT 5 level_of_cook, 'Chef' cook_category
        ) category_to_level USING (cook_category)
        -- this subquery maps each cook category to an integer representing the level of the cook.
    ) levels_for_each_episode
    GROUP BY current_year, episode_number
) -- This subquery finds the level of each episode.
SELECT current_year, episode_number, level_of_episode
```

Μπορούμε να επισπεύσουμε τη διαδικασία του JOIN αυτού μέσω ενός index στο cook_category:

```sql
CREATE INDEX cook_category_idx ON cooks (cook_category);
```

Το συγκεκριμένο index, με την ίδια ακριβώς λογική, θα βοηθήσει και σε ένα subquery της κλήρωσης το οποίο επιλέγει τον νικητή του κάθε επεισοδίου:

```
INSERT INTO winners
SELECT count_years,count_episodes,cook_id,SUM(grade) total_grade, cook_category
FROM (
SELECT contestant_id AS cook_id,grade
FROM evaluation
WHERE current_year= count_years AND episode_number = count_episodes) temp
JOIN cooks USING (cook_id)
JOIN(
        SELECT 1 level_of_cook, 'C Cook' cook_category
        UNION
        SELECT 2 level_of_cook, 'B Cook' cook_category
        UNION
        SELECT 3 level_of_cook, 'A Cook' cook_category
        UNION
        SELECT 4 level_of_cook, "Chef's Assistant" cook_category
        UNION
        SELECT 5 level_of_cook, 'Chef' cook_category
    ) tempo USING (cook_category)
GROUP BY cook_id
ORDER BY total_grade DESC,level_of_cook DESC,RAND()
LIMIT 1
;
```

3.3 Εναλλακτικά Query Plans

3.6)

```
66  -- 3.6
67  SELECT a_tag_name, b_tag_name, COUNT(*) Tag_Couple_Appearances
68  FROM(
69      SELECT a.tag_name a_tag_name, b.tag_name b_tag_name
70      FROM cooks_recipes_per_episode competition
71      JOIN tags a USING (rec_name)
72      JOIN tags b ON a.rec_name=b.rec_name AND a.tag_name<b.tag_name
73  ) possible_couples_of_tags -- this subquery finds the possible couples of tags that appeared in the competition.
74                             -- The couple is contained in the query as many times as it appears in the competition.
75  GROUP BY a_tag_name, b_tag_name
76  ORDER BY Tag_Couple_Appearances DESC
77  LIMIT 3;
```

Σε αυτό το query ο optimizer της SQL επιλέγει να χρησιμοποιήσει το index του attribute rec_name, τόσο στον πίνακα competition, όσο και στους δύο πίνακες tags. Η σειρά του JOIN επιλέγεται έτσι ώστε πρώτα να σκαναριστεί το table με τις λιγότερες γραμμές. Στα επόμενα tables του JOIN θα γίνει χρήση indexes και θα αποφευχθεί το σκανάρισμα των περισσότερων γραμμών. Παρακάτω φαίνονται τα αντίστοιχα traces που μπορούμε να δούμε μέσω της εντολής EXPLAIN και επιλέγοντας το FORMAT «JSON». Με κόκκινο φαίνονται τα indexes που χρησιμοποιούνται και με μπλε οι γραμμές που σκαναρίστηκαν από κάθε table. Όπως βλέπουμε το competition

σκανάρεται 500 φορές (όσες και οι γραμμές του), ενώ τα δύο tables tags μόλις 2 φορές.


```
{
 "query_block": {
  "select_id": 1,
  "filesort": {
   "sort_key": "count(0) desc",
   "temporary_table": {
    "table": {
     "table_name": "competition",
     "access_type": "range",
     "possible_keys": ["f_key_cooks_recipes_per_episode_recipe"],
     "key": "f_key_cooks_recipes_per_episode_recipe",
     "key_length": "203",
     "used_key_parts": ["rec_name"],
     "rows": 500,
     "filtered": 100,
     "attached_condition": "competition.rec_name is not null and competition.rec_name is not null",
     "using_index": true
    },
    "table": {
     "table_name": "a",
     "access_type": "ref",
     "possible_keys": ["PRIMARY", "f_key_tags_recipe"],
     "key": "f_key_tags_recipe",
     "key_length": "202",
     "used_key_parts": ["rec_name"],
     "ref": ["cooking.competition.rec_name"],
     "rows": 2,
     "filtered": 100,
     "using_index": true
    },
    "table": {
     "table_name": "b",
```

```
      "access_type": "ref",
      "possible_keys": ["PRIMARY", "f_key_tags_recipe"],
      "key": "f_key_tags_recipe",
      "key_length": "202",
      "used_key_parts": ["rec_name"],
      "ref": ["cooking.competition.rec_name"],
      "rows": 2,
      "filtered": 100,
      "attached_condition": "a.tag_name < b.tag_name",
      "using_index": true
    }
   }
  }
 }
}
,
```


Εάν εξαναγκάσουμε τον optimizer της SQL να αγνοήσει το index του rec_name, μέσω της εντολής IGNORE INDEX, στον πίνακα tags b, ο optimizer θα αναγκαστεί να χρησιμοποιήσει το index του Primary Key του tags, που όμως δεν είναι κατάλληλο στη συγκεκριμένη περίπτωση λόγω της συνθήκης: a.tag_name < b.tag_name.

```
292     -- 3.6
293 •   EXPLAIN format = json
294     SELECT a_tag_name, b_tag_name, COUNT(*) Tag_Couple_Appearances
295     FROM(
296         SELECT a.tag_name a_tag_name, b.tag_name b_tag_name
297         FROM cooks_recipes_per_episode competition
298         JOIN tags a USING (rec_name)
299         JOIN tags b IGNORE INDEX (f_key_tags_recipe) ON a.rec_name=b.rec_name AND a.tag_name<b.tag_name
300     ) possible_couples_of_tags -- this subquery finds the possible couples of tags that appeared in the competition.
301                                -- The couple is contained in the query as many times as it appears in the competition.
302     GROUP BY a_tag_name, b_tag_name
303     ORDER BY Tag_Couple_Appearances DESC
304     LIMIT 3;
305
```

Για αυτό θα επιλέξει ως πρώτο table στο JOIN το tags b, παρόλο που έχει περισσότερες γραμμές από το competition. Έτσι καταφέρνει να εξετάσει αυτήν τη συνθήκη μόνο 2 φορές , για τις 2 γραμμές που ικανοποίησαν τα JOINS.


```
{
 "query_block": {
  "select_id": 1,
  "filesort": {
```

```
"sort_key": "count(0) desc",
"temporary_table": {
 "table": {
  "table_name": "b",
  "access_type": "index",
  "possible_keys": ["PRIMARY"],
  "key": "PRIMARY",
  "key_length": "404",
  "used_key_parts": ["tag_name", "rec_name"],
  "rows": 1028,
  "filtered": 100,
  "using_index": true
 },
 "table": {
  "table_name": "competition",
  "access_type": "ref",
  "possible_keys": ["f_key_cooks_recipes_per_episode_recipe"],
  "key": "f_key_cooks_recipes_per_episode_recipe",
  "key_length": "203",
  "used_key_parts": ["rec_name"],
  "ref": ["cooking.b.rec_name"],
  "rows": 1,
  "filtered": 100,
  "using_index": true
 },
 "table": {
  "table_name": "a",
  "access_type": "ref",
  "possible_keys": ["PRIMARY", "f_key_tags_recipe"],
  "key": "f_key_tags_recipe",
  "key_length": "202",
  "used_key_parts": ["rec_name"],
  "ref": ["cooking.b.rec_name"],
  "rows": 2,
```

```
      "filtered": 100,
      "attached_condition": "a.tag_name < b.tag_name",
      "using_index": true
     }
    }
   }
  }
}
```


Τέλος, ένα ακόμα ενδιαφέρον query plan θα ήταν να εξαναγκάσουμε επιπλέον τον optimizer να εκτελέσει τα JOINS με τη συγκεκριμένη σειρά που έχουν γραφτεί τα tables στον κώδικα. Αυτό το επιτυγχάνουμε μέσω της εντολής STRAIGHT_JOIN.

```
99 •  EXPLAIN format = json
100   SELECT a_tag_name, b_tag_name, COUNT(*) Tag_Couple_Appearances
101   FROM(
102       SELECT STRAIGHT_JOIN a.tag_name a_tag_name, b.tag_name b_tag_name
103       FROM cooks_recipes_per_episode competition
104       JOIN tags a USING (rec_name)
105       JOIN tags b IGNORE INDEX (f_key_tags_recipe) ON a.rec_name=b.rec_name AND a.tag_name<b.tag_name
106   ) possible_couples_of_tags -- this subquery finds the possible couples of tags that appeared in the competition.
107                              -- The couple is contained in the query as many times as it appears in the competition.
108   GROUP BY a_tag_name, b_tag_name
109   ORDER BY Tag_Couple_Appearances DESC
110   LIMIT 3;
```

Αυτή τη φορά ο optimizer δεν μπορεί να αποφύγει τη χρήση του index (tag_name,rec_name) του Primary Key για το JOIN με τη συνθήκη a.tag_name < b.tag_name. Έτσι καταφεύγει στη χρήση της τεχνικής Block Nested Loop Join. Δηλαδή αποθηκεύει σε κάθε εξωτερική λούπα σε έναν buffer μερικά υποψήφια tuples των άλλων 2 tables και τα εξετάζει ένα προς ένα στην εσωτερική λούπα. Τα traces φαίνονται παρακάτω:


```
{
 "query_block": {
  "select_id": 1,
  "filesort": {
   "sort_key": "count(0) desc",
   "temporary_table": {
    "table": {
     "table_name": "competition",
     "access_type": "range",
```

```
  "possible_keys": ["f_key_cooks_recipes_per_episode_recipe"],
  "key": "f_key_cooks_recipes_per_episode_recipe",
  "key_length": "203",
  "used_key_parts": ["rec_name"],
  "rows": 500,
  "filtered": 100,
  "attached_condition": "competition.rec_name is not null",
  "using_index": true
 },
 "table": {
  "table_name": "a",
  "access_type": "ref",
  "possible_keys": ["PRIMARY", "f_key_tags_recipe"],
  "key": "f_key_tags_recipe",
  "key_length": "202",
  "used_key_parts": ["rec_name"],
  "ref": ["cooking.competition.rec_name"],
  "rows": 2,
  "filtered": 100,
  "using_index": true
 },
 "block-nl-join": {
  "table": {
    "table_name": "b",
    "access_type": "index",
    "possible_keys": ["PRIMARY"],
    "key": "PRIMARY",
    "key_length": "404",
    "used_key_parts": ["tag_name", "rec_name"],
    "rows": 1028,
    "filtered": 100,
    "using_index": true
  },
  "buffer_type": "flat",
```

```
        "buffer_size": "256Kb",
        "join_type": "BNL",
        "attached_condition": "b.rec_name = competition.rec_name and a.tag_name < b.tag_name"
      }
    }
   }
  }
 }
}
```


Φυσικά η συγκεκριμένη παρέμβαση μειώνει αρκετά την αποδοτικότητα του query, όπως φαίνεται από τις 1028 γραμμές που σκανάρονται στο τελευταίο table.

3.8)

```
92    -- 3.8
93    WITH amount AS (
94        SELECT current_year, episode_number, COUNT(*) Amount_of_Equipment
95        FROM cooks_recipes_per_episode
96        JOIN uses_equipment  USING (rec_name)
97        GROUP BY current_year, episode_number) -- This subquery finds the amount of equipment for each episode.
98    SELECT current_year, episode_number, Amount_of_Equipment
99    FROM amount
100   WHERE Amount_of_Equipment= (
101       SELECT MAX(Amount_of_Equipment)
102       FROM amount
103       ) -- This subquery finds the max amount of equipment an episode any had.
104       -- The overall query could find more than 1 episode in case of a draw.
105       -- (More than 1 episode could have the max amount of equipment)
106       ;
```

Σε αυτό το query ο optimizer επιλέγει να κάνει το JOIN του πρώτου subquery χρησιμοποιώντας το Primary Key του uses_equipment, δηλαδή index στο rec_name. Έτσι σκανάρει πρώτα το table cooks_recipes_per_episode που έχει λιγότερες γραμμές (500 όπως φαίνεται παρακάτω) και στη συνέχεια χρησιμοποιεί το index για το uses_equipment, σκανάροντας μόνο 3 γραμμές.

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	PRIMARY	<derived2>	ALL	NULL	NULL	NULL	NULL	1500	Using where
3	SUBQUERY	<derived4>	ALL	NULL	NULL	NULL	NULL	1500	
4	DERIVED	cooks_recipes_per_episode	index	f_key_cooks_recipes_per_episode_recipe	PRIMARY	12	NULL	500	Using where
4	DERIVED	uses_equipment	ref	PRIMARY	PRIMARY	202	cooking.cooks_recipes_per_episode.rec_name	3	Using index
2	DERIVED	cooks_recipes_per_episode	index	f_key_cooks_recipes_per_episode_recipe	PRIMARY	12	NULL	500	Using where
2	DERIVED	uses_equipment	ref	PRIMARY	PRIMARY	202	cooking.cooks_recipes_per_episode.rec_name	3	Using index

Ένα εναλλακτικό query plan θα ήταν να επιβάλλουμε στον optimizer να αγνοήσει το Primary Key του uses_equipment.

```
136         -- Εναλλακτικό query plan
137 •    EXPLAIN
138      WITH amount AS (
139          SELECT current_year, episode_number, COUNT(*) Amount_of_Equipment
140          FROM uses_equipment IGNORE INDEX(PRIMARY)
141          JOIN cooks_recipes_per_episode USING (rec_name)
142          GROUP BY current_year, episode_number) -- This subquery finds the amount of equipment for each episode.
143      SELECT current_year, episode_number, Amount_of_Equipment
144      FROM amount
145      WHERE Amount_of_Equipment= (
146          SELECT MAX(Amount_of_Equipment)
147          FROM amount
148          ) -- This subquery finds the max amount of equipment an episode any had.
149          -- The overall query could find more than 1 episode in case of a draw.
150          -- (More than 1 episode could have the max amount of equipment)
151         ;
```

Αυτό θα οδηγήσει τον optimizer στο να χρησιμοποιήσει το index στο rec_name του cooks_recipes_per_episode, σκανάροντας πρώτα το uses_equipment, το οποίο όπως φαίνεται παρακάτω έχει 1393 γραμμές, μειώνοντας την αποδοτικότητα του query.

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	PRIMARY	<derived2>	ALL	NULL	NULL	NULL	NULL	1393	Using where
3	SUBQUERY	<derived4>	ALL	NULL	NULL	NULL	NULL	1393	
4	DERIVED	uses_equipment	index	NULL	f_key_uses_equipment_equipment	202	NULL	1393	Using index; Using temporary; Using filesort
4	DERIVED	cooks_recipes_per_episode	ref	f_key_cooks_recipes_per_episode_recipe	f_key_cooks_recipes_per_episode_recipe	203	cooking.uses_equipment.rec_name	1	Using index
2	DERIVED	uses_equipment	index	NULL	f_key_uses_equipment_equipment	202	NULL	1393	Using index; Using temporary; Using filesort
2	DERIVED	cooks_recipes_per_episode	ref	f_key_cooks_recipes_per_episode_recipe	f_key_cooks_recipes_per_episode_recipe	203	cooking.uses_equipment.rec_name	1	Using index