

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** 2](#_Toc167647036)

[**1.1** **Εφαρμογές που χρησιμοποιήθηκαν** 2](#_Toc167647037)

[**1.2** **Διάγραμμα ER** 3](#_Toc167647038)

[**1.3** **Σχεσιακό Διάγραμμα** 5](#_Toc167647039)

[**2.** **Υλοποίηση της Βάσης** 6](#_Toc167647040)

[**2.1** **DDL** 6](#_Toc167647041)

[**2.2** **DML** 7](#_Toc167647042)

[**2.3** **Queries** 9](#_Toc167647043)

[**3.** **Λεπτομέρειες Υλοποίησης** 10](#_Toc167647044)

[**3.1** **Triggers** 10](#_Toc167647045)

[**3.2** **Indexes/Ευρετήρια** 11](#_Toc167647046)

[**3.3** **Εναλλακτικά Query Plans** 14](#_Toc167647047)

# **Σχεδίαση της Βάσης**

## **Εφαρμογές που χρησιμοποιήθηκαν**

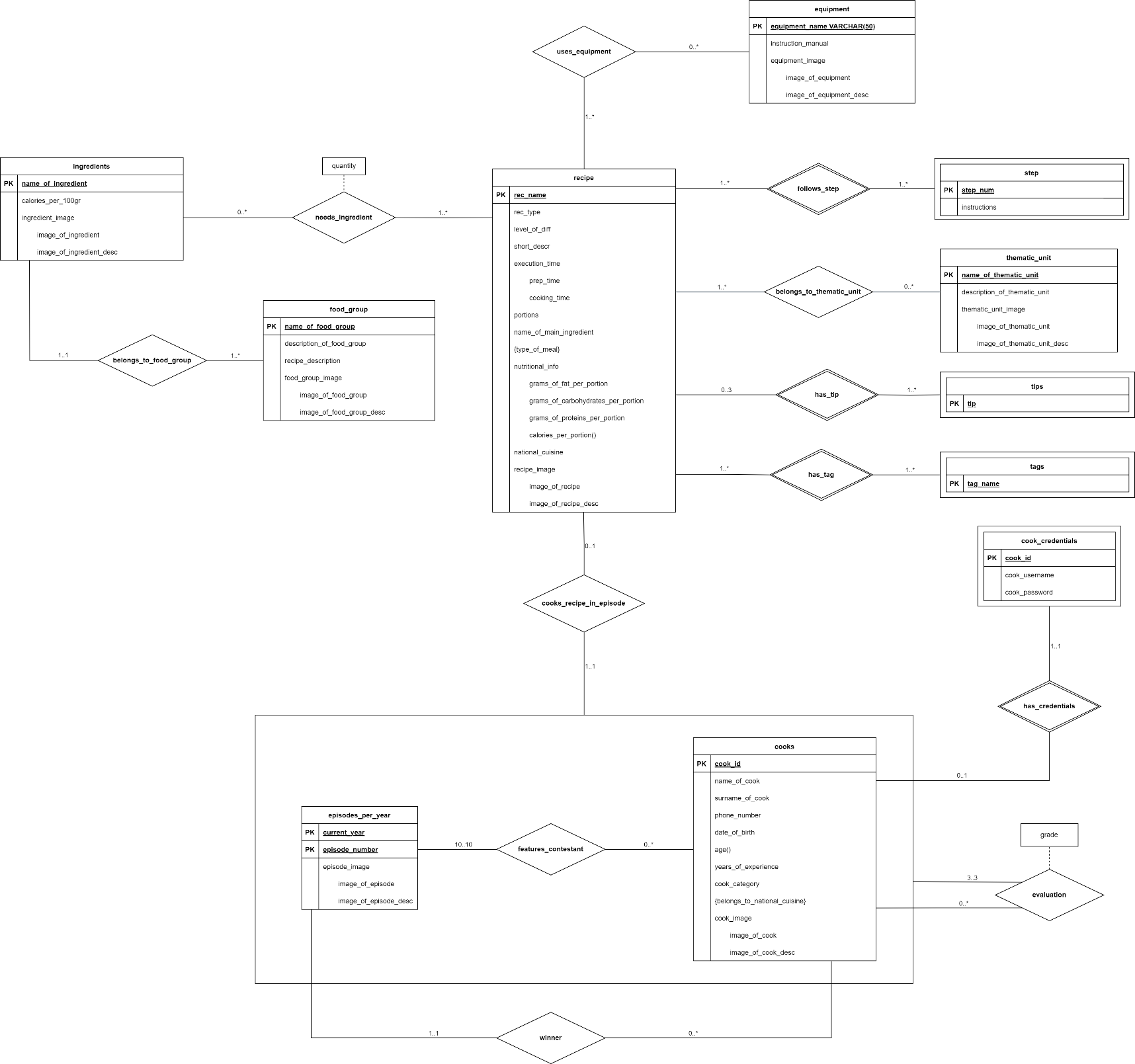
Η εργασία μας σχεδιάστηκε και υλοποιήθηκε με τη χρήση της MySQL, μέσω του client MySQL Workbench για την εκτέλεση των SQL scripts που δημιουργούν τη βάση και εκτελούν την κλήρωση του διαγωνισμού, καθώς και των queries.

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.

To 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*](https://github.com/nikosanag/DB_Project)

## **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:

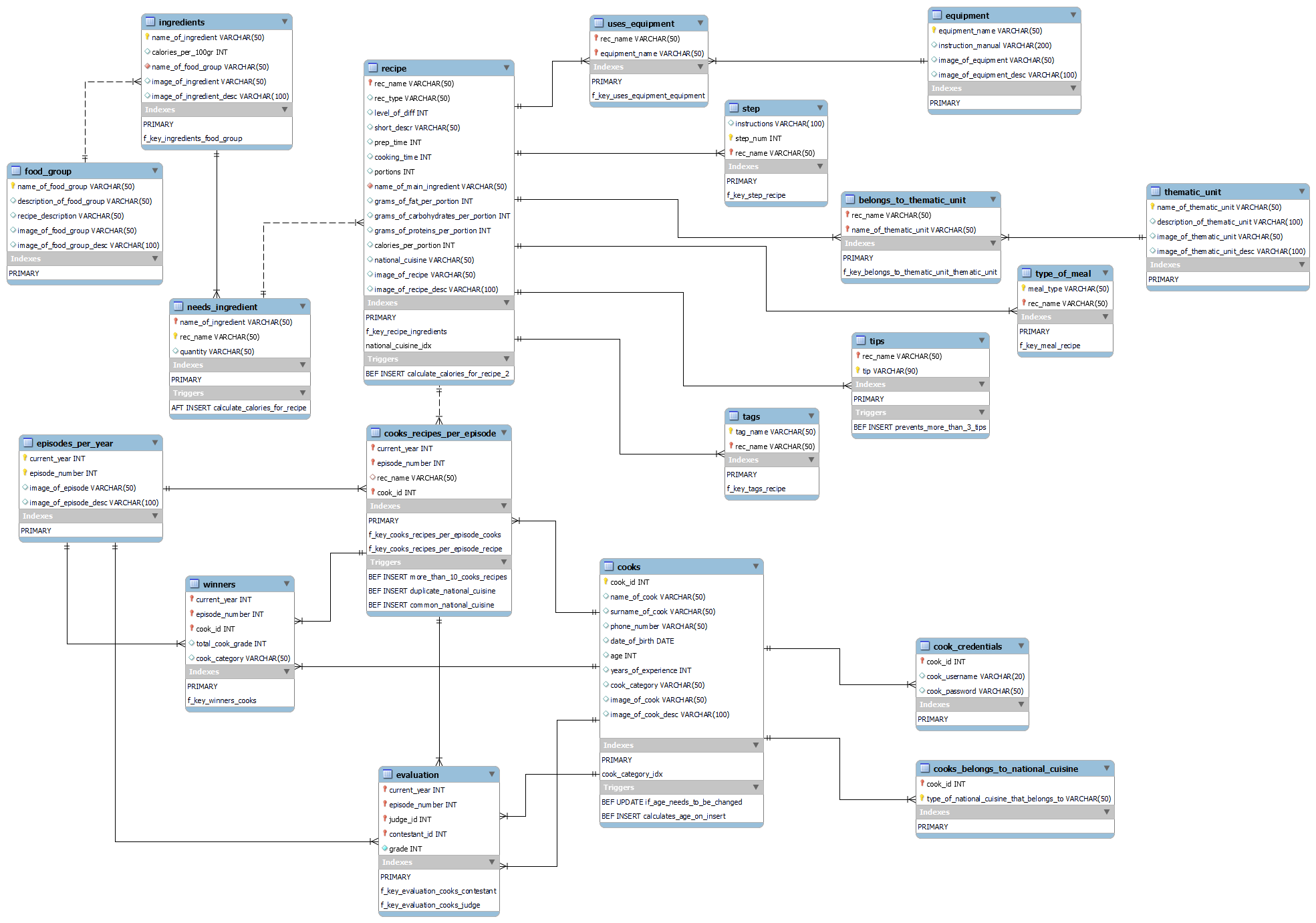
Η εικόνα φαίνεται καλύτερα στο αντίστοιχο αρχείο στο GitHub Repository της εργασίας με όνομα αρχείου *ER-Diagram.png*. Το αντίστοιχο αρχείο που αποθηκεύσαμε από το draw.io είναι το *ER-Diagram.drawio.crswap*.

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.

## **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.

# **Υλοποίηση της Βάσης**

## **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».

To 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 επίδειξης της λετουργικότητας της εργασίας 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*](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 εκδόσ

## **DML**

Το πρώτο κομμάτι του DML είναι η εισαγωγή των δεδομένων στους πίνακες που δημιουργήθηκαν στο DDL. Όλες οι εισαγωγές εκτός από την εισαγωγή των εικόνων των επεισοδίων και της περιγραφής τους πραγματοποιούνται με την εκτέλεση του script με όνομα *Insertions.sql*. To 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 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.

## **Queries**

Tα 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.

# **Λεπτομέρειες Υλοποίησης**

## **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**

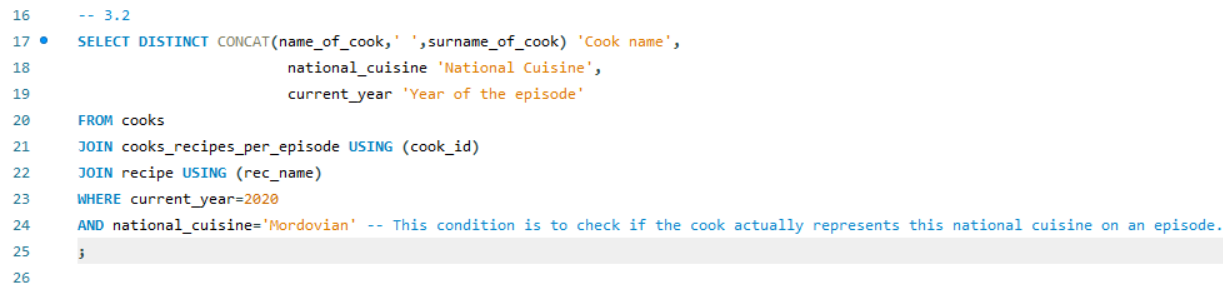
Εξασφαλίζει ότι δεν πρόκειται να εισαχθούν σε κάποιο επεισόδιο δύο συνταγές που ανήκουν στην ίδια εθνική κουζίνα. Εάν αυτό συμβεί θα προκληθεί error.

* **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.

## **Indexes/Ευρετήρια**

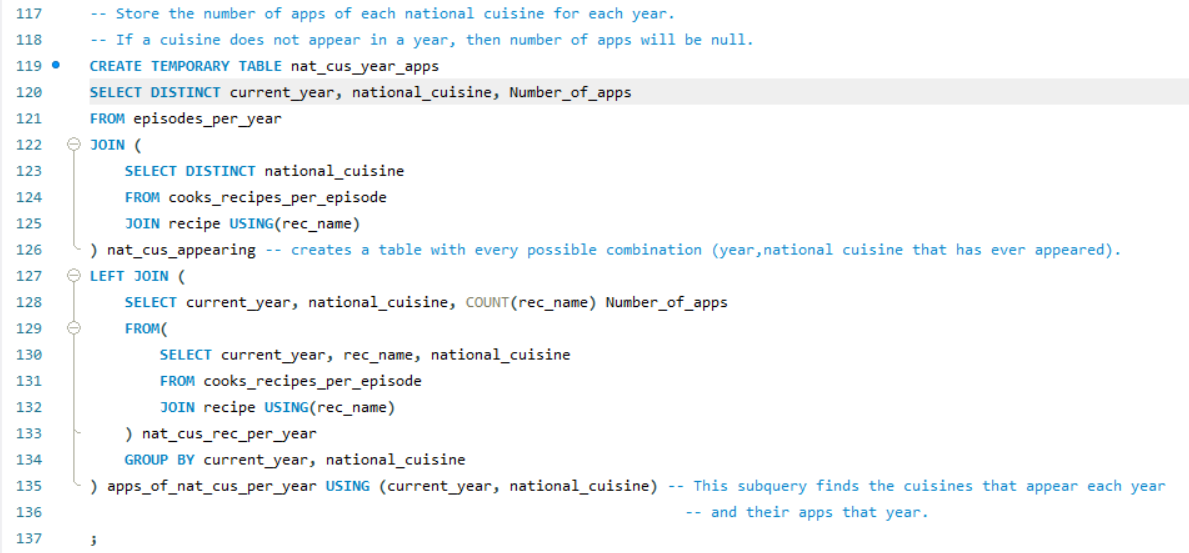
Όπως γνωρίζουμε, για όποια attributes έχουν καθοριστεί ως κλειδιά, είτε Primary Keys είτε Foreign Keys, δημιουργούνται αυτόματα και indexes από τη MySQL. Όπως είναι κατανοητό, τα περισσότερα queries θα βασίζονται σε αυτά τα attributes και συνεπώς δε θα απαιτούν επιπρόσθετα indexes. Υπάρχουν όμως κάποια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



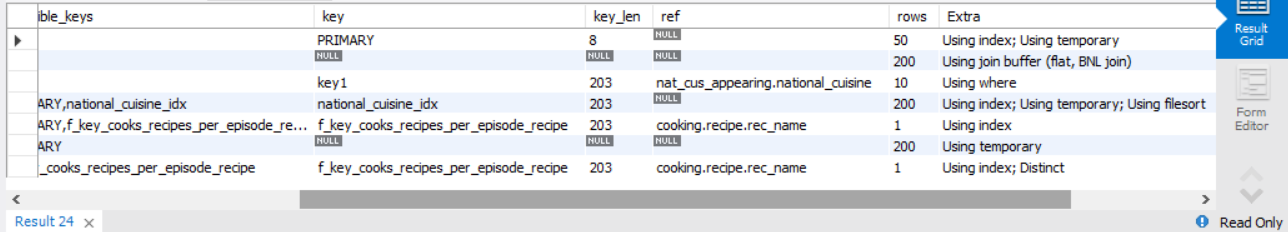
Τα indexes είναι κατάλληλα για τέτοιες συνθήκες αναζήτησης και συνεπώς θα ήταν καλή πρακτική να ορίσουμε ένα index για το table recipe στο attribute national\_cuisine.



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



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



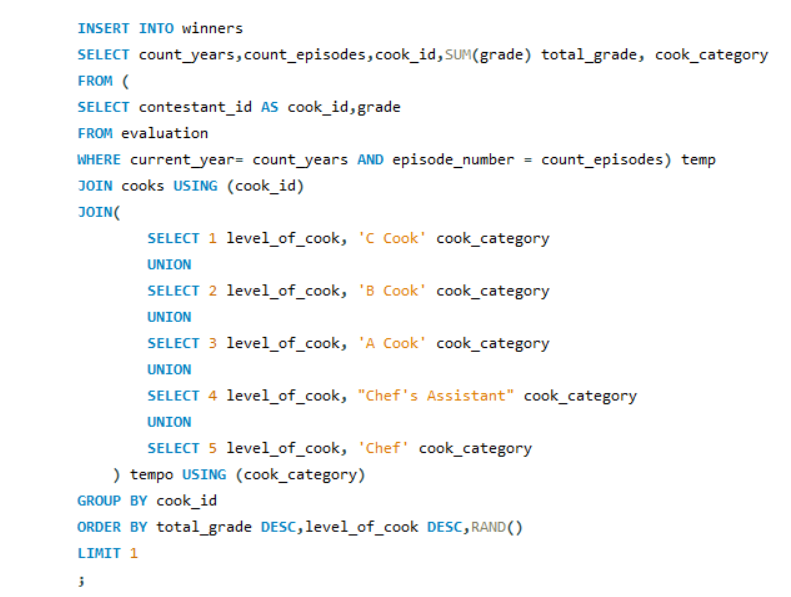
Επιπλέον, στο query 3.13 κάνουμε JOIN χρησιμοποιώντας το attribute cook\_category του cook.



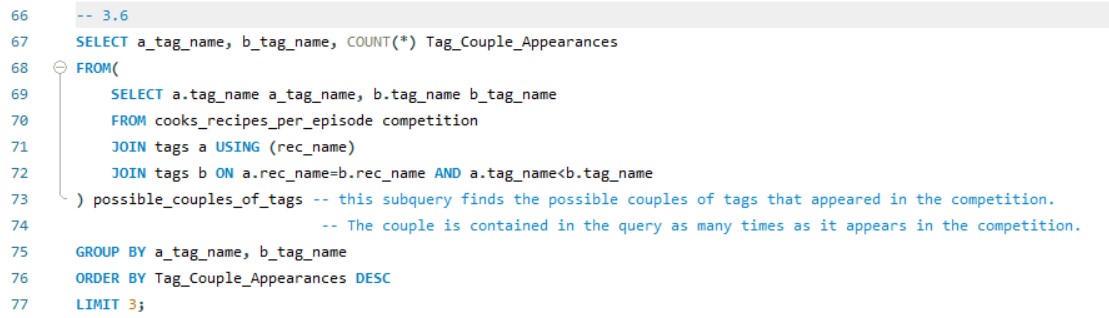
Μπορούμε να επισπεύσουμε τη διαδικασία του JOIN αυτού μέσω ενός index στο cook\_category:



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



## **Εναλλακτικά Query Plans**

3.6) 

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

}

}

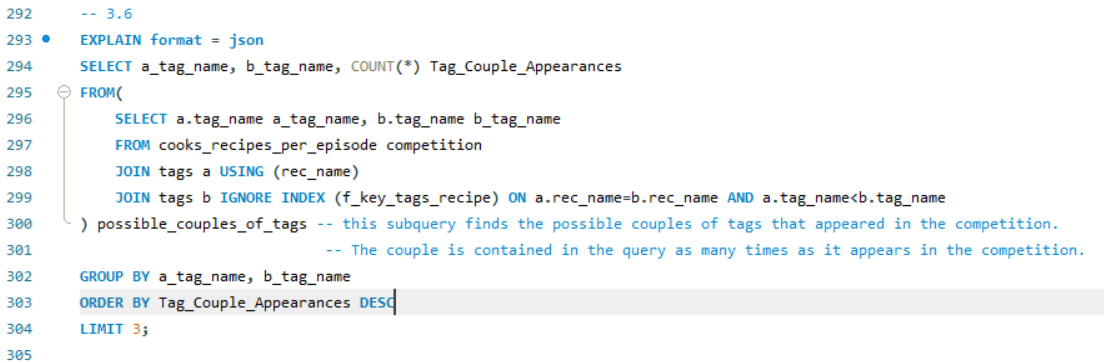
}

}

}

,

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.

 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

}

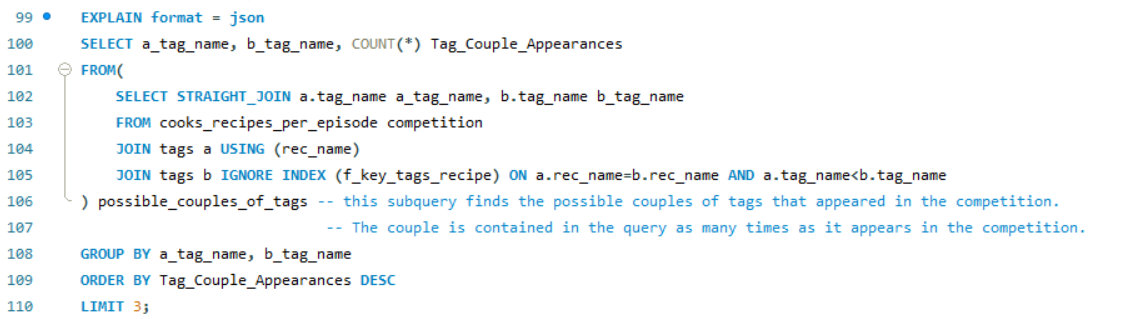
}

}

}

}

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.



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"

}

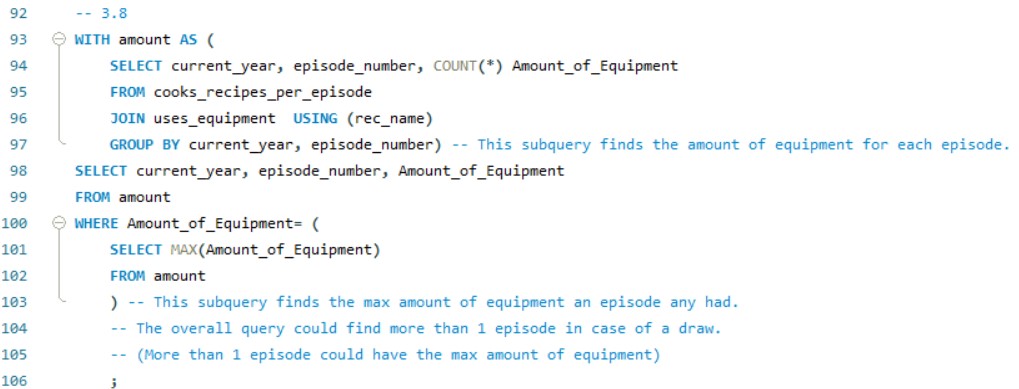
}

}

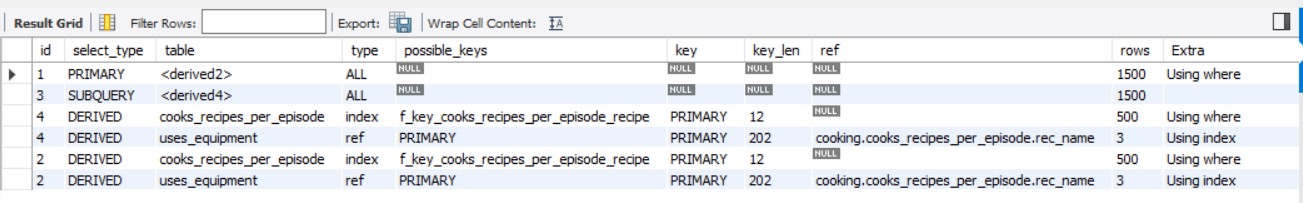
}

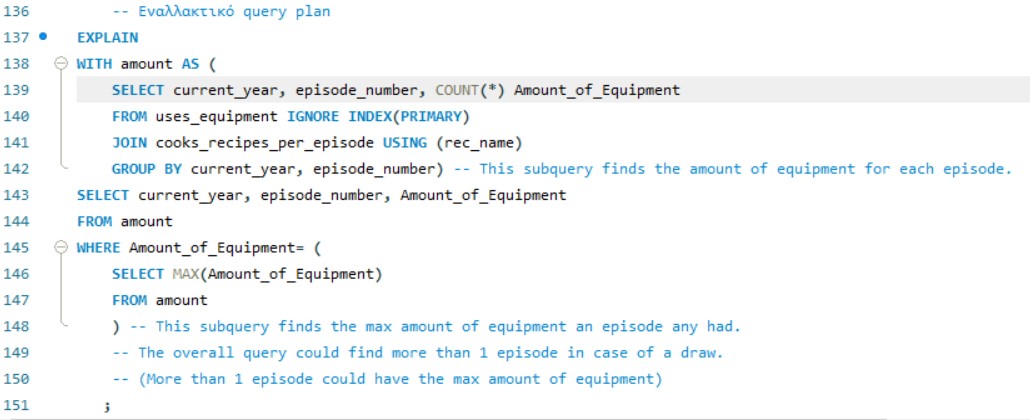
}

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

3.8) 

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.



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

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.

