ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΡΓΑΣΤΗΡΙΟ ΣΥΣΤΗΜΑΤΩΝ ΒΑΣΕΩΝ ΓΝΩΣΕΩΝ ΚΑΙ ΔΕΔΟΜΕΝΩΝ

Ακ. έτος 2023-2024, 6ο εξάμηνο

ΟΜΑΔΑ Project 127:

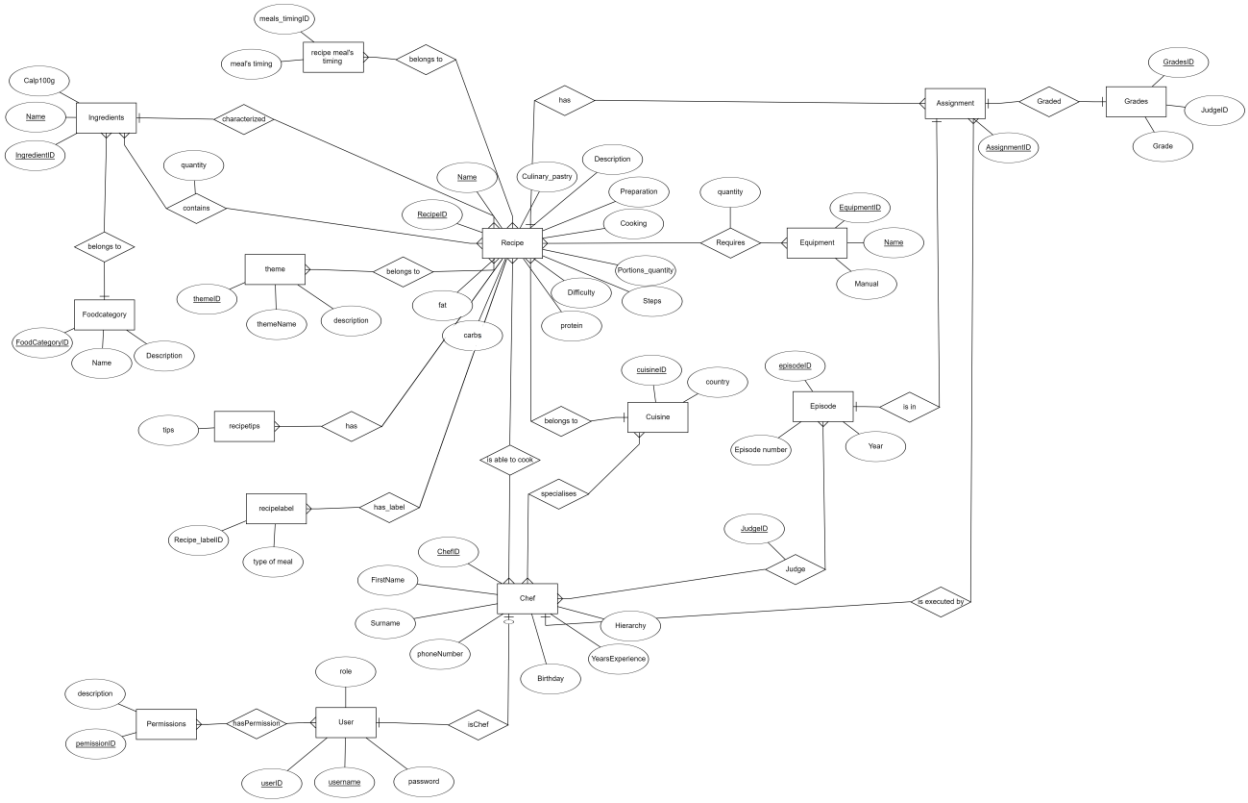
03121840 ΝΕΣΤΟΡΗ ΔΗΜΗΤΡΑ

03121154 ΠΥΡΠΙΡΗ ΠΑΡΑΣΚΕΥΗ

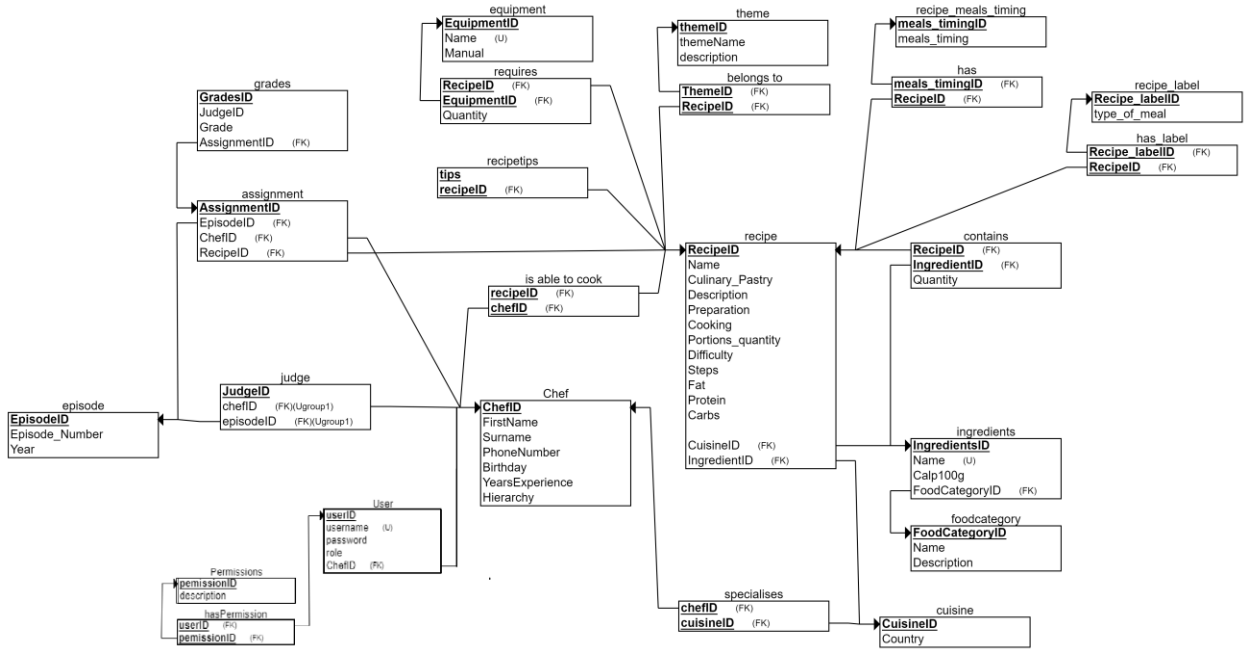
03121145 ΦΥΤΡΟΥ ΜΑΡΙΑ

Περιεχόμενα

ER Διάγραμμα της Βάσης



Relational Schema



Η έννοια στην οποία αναφέρεται κάθε οντότητα είναι προφανής με εξαίρεση τις εξής:

Η οντότητα recipe_meals_timing αναφέρεται στις έννοιες {πρωινό, πρόγευμα, γεύμα, απογευματινό, δείπνο}.

Η οντότητα foodcategory αναφέρεται στις «Ομάδες Τροφίμων».

Η οντότητα recipe_label αναφέρεται στις ετικέτες για μορφή/ τύπο γεύματος (πχ brunch, quick-lunch, κρύο πιάτο κτλ).

Η οντότητα assignment αναφέρεται στις συμμετοχές που υπάρχουν στο διαγωνισμό.

Για την μετροπή του ER diagram σε Relation Schema ακολουθήσαμε την εξής λογική:

Καθε οντότητα την μετρέψαμε σε πίνακα.

Για καρδιναλιότητα «ενα-προς-ένα» και «ένα –προς-πολλά», ορίσαμε το πρωτεύον κλειδί της οντότητας της «ένα» πλευράς ως ξένο κλειδί στην οντότητα της «πολλά» πλευράς.

Για καρδιναλιότητα «πολλά-προς-πολλά», δημιουργήσαμε έναν νέο πίνακα στον οποίο ορίσαμε ως ξένα παιδιά τα πρωτεύοντα κλειδιά των οντοτήτων που συμμετέχουν στη σχέση.

Indexes

Το RDBMS που χρησιμοποιήσαμε δημιουργεί από μόνο του indexes στα primary keys που ορίσαμε στη βάση.

Στο ερώτημα 3.8 ζητήθηκε εναλλακτικό query plan με force index, το οποίο δημιουργήσαμε όπως φαίνεται παρακάτω:

CREATE INDEX index_requires_equipment ON requires (EquipmentID);

Κάνοντας trace το query παρατηρήσαμε ότι έκανε πέρασμα σε περισσότερα rows χρησιμοποιώντας το νέο index σε σύγκριση με την περίπτωση που είχε ως index τα primary keys. Επομένως, ο αρχικός τρόπος αναζήτησης ήταν πιο αποδοτικός.

DDL script

```
CREATE TABLE `chef` (

  `ChefID` int(11) NOT NULL,

  `FirstName` varchar(25) NOT NULL,
```

```
  `Surname` varchar(25) NOT NULL,

  `PhoneNumber` varchar(15) NOT NULL,

  `Birthday` DATE NOT NULL,

  `YearsExperience` int(11) NOT NULL,

  `Hierarchy` ENUM('First Cook', 'Second Cook', 'Third Cook', 'Chef Assistant', 'Head Chef') NOT NULL,

  CONSTRAINT check_hierarchy_headchef CHECK (`YearsExperience` >= 12 OR `Hierarchy` != 'Head Chef'),

  CONSTRAINT check_hierarchy_assistant CHECK (`YearsExperience` >= 8 OR `Hierarchy` != 'Chef Assistant'),

  CONSTRAINT check_hierarchy_third CHECK (`YearsExperience` >= 6 OR `Hierarchy` != 'Third Cook'),

  CONSTRAINT check_hierarchy_second CHECK (`YearsExperience` >= 3 OR `Hierarchy` != 'Second Cook'),

  CONSTRAINT check_hierarchy_first CHECK (`YearsExperience` >= 0 OR `YearsExperience` <= 5 OR `Hierarchy` != 'First Cook'),

  PRIMARY KEY (`ChefID`)

);


CREATE TABLE `recipe` (

  `RecipeID` int(6) NOT NULL,

  `Name` varchar(60) NOT NULL,

  `Culinary_Pastry` enum('Culinary', 'Pastry') NOT NULL,

  `Description` text NOT NULL,

  `Preparation` time(0) NOT NULL DEFAULT '00:00:00',

  `Cooking` time(0) NOT NULL DEFAULT '00:00:00',

  `Portions_quantity` int(3) UNSIGNED ZEROFILL NOT NULL,

  `Difficulty` tinyint(1) NOT NULL,

  `Steps` text NOT NULL,

  `Fat` int(11) NOT NULL,
```

```
  `Protein` int(11) NOT NULL,
  `Carbs` int(11) NOT NULL,
  `IngredientsID` int(11) NOT NULL,
  `CuisineID` int(11) NOT NULL,
  CONSTRAINT `check_difficulty` CHECK (`Difficulty` BETWEEN 1 AND 5),
  PRIMARY KEY (`RecipeID`)
);

CREATE TABLE `isabletocook` (
  `recipeID` int(11) NOT NULL,
  `chefID` int(11) NOT NULL,
  CONSTRAINT `isabletocook_ibfk_1` FOREIGN KEY (`recipeID`) REFERENCES `recipe` (`RecipeID`) ON UPDATE CASCADE,
  CONSTRAINT `isabletocook_ibfk_2` FOREIGN KEY (`chefID`) REFERENCES `chef` (`ChefID`) ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `recipe_meals_timing` (
  `meals_timingID` int(11) NOT NULL,
  `meals_timing` varchar(25) NOT NULL,
   PRIMARY KEY (`meals_timingID`)
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `has` (
  `RecipeID` int(11) NOT NULL,
```

```
  `meals_timingID` int(11) NOT NULL,
  CONSTRAINT `has_fk1` FOREIGN KEY (`RecipeID`) REFERENCES recipe(`RecipeID`),
  CONSTRAINT `has_fk2` FOREIGN KEY (`meals_timingID`) REFERENCES recipe_meals_timing(`meals_timingID`)
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `recipe_label` (
  `type_of_meal` varchar(25) NOT NULL,
  `Recipe_labelID` int(11) NOT NULL,
  PRIMARY KEY (`Recipe_labelID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `has_label` (
  `RecipeID` int(11) NOT NULL,
  `Recipe_labelID` int(11) NOT NULL,
  CONSTRAINT `haslabel_fk1` FOREIGN KEY (`RecipeID`) REFERENCES recipe(`RecipeID`),
  CONSTRAINT `haslabel_fk2` FOREIGN KEY (`Recipe_labelID`) REFERENCES recipe_label(`Recipe_labelID`)
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `recipetips` (
  `tips` text NOT NULL,
  `recipeID` int(11) NOT NULL,
```

```
  CONSTRAINT `recipetips_ibfk_1` FOREIGN KEY (`recipeID`) REFERENCES `recipe` (`RecipeID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;


CREATE TABLE `theme` (
  `ThemeID` int(11) NOT NULL,
  `ThemeName` varchar(50) NOT NULL,
  `Description` varchar(250) NOT NULL,
  PRIMARY KEY (`ThemeID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;


CREATE TABLE `belongsto` (
  `ThemeID` int(11) NOT NULL,
  `RecipeID` int(11) NOT NULL,
  PRIMARY KEY (`ThemeID`, `RecipeID`),
  CONSTRAINT `fk_belongsto_theme` FOREIGN KEY (`ThemeID`) REFERENCES `theme` (`ThemeID`),
  CONSTRAINT `fk_belongsto_recipe` FOREIGN KEY (`RecipeID`) REFERENCES `recipe` (`RecipeID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;


CREATE TABLE `cuisine` (
  `CuisineID` int(11) NOT NULL,
  `Country` varchar(25) NOT NULL,
   PRIMARY KEY (`CuisineID`)
```

```
);

CREATE TABLE `foodcategory` (
  `FoodCategoryID` int(11) NOT NULL,
  `Name` varchar(40) NOT NULL,
  `Description` text NOT NULL,
  PRIMARY KEY (`FoodCategoryID`)
);

CREATE TABLE `ingredients` (
  `IngredientsID` int(11) NOT NULL AUTO_INCREMENT,
  `Name` varchar(25) NOT NULL,
  `Calp100` int(11) NOT NULL,
  `FoodCategoryID` int(11) NOT NULL,
   PRIMARY KEY(IngredientsID),
   CONSTRAINT `ingr_belogsto_cat` FOREIGN KEY (`FoodCategoryID`) REFERENCES `foodcategory`(`FoodCategoryID`),
   UNIQUE (Name)
);

CREATE TABLE `contains` (
  `RecipeID` int(11) NOT NULL,
  `IngredientsID` int(11) NOT NULL,
  `Quantity` varchar(20) NOT NULL,
  PRIMARY KEY (`IngredientsID`, `RecipeID`),
  KEY `IngredientsID` (`IngredientsID`),
```

```
  CONSTRAINT `fk_contains_ingredients` FOREIGN KEY (`IngredientsID`) REFERENCES `ingredients` (`IngredientsID`),
  CONSTRAINT `fk_contains_recipe` FOREIGN KEY (`RecipeID`) REFERENCES `recipe` (`RecipeID`)
);

CREATE TABLE `equipment` (
  `EquipmentID` int(11) NOT NULL,
  `Name` varchar(25) NOT NULL,
  `Manual` text NOT NULL,
  PRIMARY KEY (`EquipmentID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `requires` (
  `RecipeID` int(11) NOT NULL,
  `EquipmentID` int(11) NOT NULL,
  `Quantity` int(5) NOT NULL,
  PRIMARY KEY (`EquipmentID`, `RecipeID`),
  CONSTRAINT `fk_requires_equipment` FOREIGN KEY (`EquipmentID`) REFERENCES `equipment` (`EquipmentID`),
  CONSTRAINT `fk_requires_recipe` FOREIGN KEY (`RecipeID`) REFERENCES `recipe` (`RecipeID`)
);

CREATE TABLE `episode` (
  `EpisodeID` int(11) NOT NULL,
  `Episode_Number` int(11) NOT NULL,
  `Year` year(4) NOT NULL,
```

```
  PRIMARY KEY (`EpisodeID`),
  CONSTRAINT `chk_episode_number` CHECK (`Episode_Number` BETWEEN 1 AND 10)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

ALTER TABLE `episode`
  MODIFY `EpisodeID` int(11) NOT NULL AUTO_INCREMENT;

CREATE TABLE `judge` (
  `JudgeID` int(11) NOT NULL,
  `ChefID` int(11) NOT NULL,
  `EpisodeID` int(11) NOT NULL,
  PRIMARY KEY (`JudgeID`),
  KEY `ChefID` (`ChefID`),
  KEY `EpisodeID` (`EpisodeID`),
  CONSTRAINT `judge_ibfk_1` FOREIGN KEY (`ChefID`) REFERENCES `chef` (`ChefID`) ON UPDATE CASCADE,
  CONSTRAINT `judge_ibfk_2` FOREIGN KEY (`EpisodeID`) REFERENCES `episode` (`EpisodeID`) ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

ALTER TABLE `judge`
  MODIFY `JudgeID` int(11) NOT NULL AUTO_INCREMENT;

CREATE TABLE `assignment` (
  `AssignmentID` int(11) NOT NULL,
  `EpisodeID` int(11) NOT NULL,
```

```
  `RecipeID` int(11) NOT NULL,

  `ChefID` int(11) NOT NULL,

  PRIMARY KEY (`AssignmentID`),

  CONSTRAINT `assignment_ibfk_1` FOREIGN KEY (`EpisodeID`) REFERENCES `episode` (`EpisodeID`) ON UPDATE CASCADE,

  CONSTRAINT `assignment_ibfk_2` FOREIGN KEY (`ChefID`) REFERENCES `chef` (`ChefID`) ON UPDATE CASCADE,

  CONSTRAINT `assignment_ibfk_3` FOREIGN KEY (`RecipeID`) REFERENCES `recipe` (`RecipeID`) ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;


ALTER TABLE `assignment`

  MODIFY `AssignmentID` int(11) NOT NULL AUTO_INCREMENT;



CREATE TABLE `grades` (

  `GradesID` int(11) NOT NULL,

  `Grade` int(11) NOT NULL,

  `JudgeID` int(11) NOT NULL,

  `AssignmentID` int(11) NOT NULL,

  `EpisodeID` int(11) NOT NULL,

  PRIMARY KEY (`GradesID`),

  KEY `JudgeID` (`JudgeID`),

  KEY `AssignmentID` (`AssignmentID`),

  CONSTRAINT `grades_ibfk_1` FOREIGN KEY (`JudgeID`) REFERENCES `judge` (`JudgeID`) ON UPDATE CASCADE,
```

```
  CONSTRAINT `grades_ibfk_2` FOREIGN KEY (`AssignmentID`) REFERENCES `assignment` (`AssignmentID`) ON UPDATE CASCADE,
  CONSTRAINT `chk_grade` CHECK (`Grade` BETWEEN 1 AND 5)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

ALTER TABLE `grades`
  MODIFY `GradesID` int(11) NOT NULL AUTO_INCREMENT;

CREATE TABLE `specialises` (
  `CuisineID` int(11) NOT NULL,
  `ChefID` int(11) NOT NULL,
  CONSTRAINT `specialises_ibfk_1` FOREIGN KEY (`CuisineID`) REFERENCES `cuisine` (`CuisineID`) ON UPDATE CASCADE,
  CONSTRAINT `specialises_ibfk_2` FOREIGN KEY (`ChefID`) REFERENCES `chef` (`ChefID`) ON UPDATE CASCADE
);

CREATE TABLE User (
    userID INT(11) NOT NULL AUTO_INCREMENT,
    username CHAR(30) UNIQUE NOT NULL,
    password CHAR(30) NOT NULL,
    role ENUM('Admin', 'Cook') NOT NULL,
    chefID INT(11) NULL,
    PRIMARY KEY (userID),
    FOREIGN KEY (chefID) REFERENCES Chef(ChefID)
);
```

```
CREATE TABLE Permissions (
   permissionID INT(11) NOT NULL AUTO_INCREMENT,
   description TEXT NOT NULL,
   PRIMARY KEY (permissionID)
);

CREATE TABLE hasPermission (
   userID INT(11) NOT NULL,
   permissionID INT(11) NOT NULL,
   PRIMARY KEY (userID, permissionID),
   FOREIGN KEY (userID) REFERENCES User(userID),
   FOREIGN KEY (permissionID) REFERENCES Permissions(permissionID)
);
```

Περιορισμοί:

Σε attributes που λαμβάνουν αρκετά συγκεκριμένες τιμές χρησιμοποιήσαμε ENUM για να έχουμε σίγουρα έγκυρη καταχώρηση.

Στον πίνακα Chef βάλαμε περιορισμούς για την κατάρτιση του σε σχέση με τα χρόνια εμπειρίας του ώστε να αιτιολογείται ρεαλιστικά η θέση του.

Στον πίνακα recipe στον βαθμό δυσκολίας βάλαμε περιορισμό ότι είναι ένας αριθμός από το 1 έως και το 5.

Στον πίνακα Episode στον αριθμό του επεισοδίου βάλαμε περιορισμό να είναι από το 1 έως και το 10 (κάθε σεζόν έχει 10 επεισόδια).

Στον πίνακα grades βάλαμε η βαθμολογία να είναι από το 1 έως και το 5.

Ολα τα ξένα κλειδία είναι επίσης περιορισμοί.

DML script

Για την παραγωγή των δεδομένων στα assignments, judge, grades χρησιμοποιήθηκε ένα πρόγραμμα της python που δημιουργήσαμε εμείς. Σε κάθε επεισόδιο επιλέγονται αυτόματα, με τυχαίο τρόπο, από το σύστημα, 10 εθνικές κουζίνες, 10 μάγειρες αντιπρόσωποι από κάθε κουζίνα, 3 μάγειρες κριτές και 1 συνταγή από κάθε εθνική κουζίνα που ανατίθεται σε 1 μάγειρα . Στη συνέχεια παράγονται και οι βαθμολογίες για κάθε συμμετοχή. Προσέξαμε ώστε στα δεδομένα που παράγονται κάποιος μάγειρας να μην συμμετέχει συνεχόμενα σε περισσότερα από 3 επεισόδια και ένας μάγειρας διαγωνιζόμενος να μην είναι και κριτής στο ίδιο επεισόδιο.

Επίσης, χρησιμοποιήθηκαν και triggers.

- Το trigger different_cuisines_in_each_episode_trg εξασφαλίζει ότι κάθε επεισόδιο θα έχει 10 διαφορετικές κουζίνες
- Το assignments_per_episode_trg διασφαλίζει ότι κάθε επεισόδιο θα έχει 10 assignments
- Το participation_as_chef_or_judge_trg χρησιμοποιείται ώστε σε ένα επεισόδιο ένας chef να μπορεί να συμμετέχει ως διαγωνιζόμενος ή ως κριτής, αλλά να μην μπορεί να έχει ταυτόχρονα και τους δύο ρόλους.
- Το three_grades_for_assignment_trg δημιουργήθηκε ώστε κάθε συμμετοχή σε ένα επεισόδιο να βαθμολογείται με 3 βαθμούς.
- Το one_chef_participation_per_episode_trg διασφαλίζει ότι ο κάθε μάγειρας μπορεί να συμμετάσχει σε κάθε επεισόδιο μια φορά ως διαγωνιζόμενος, δηλαδή να εκτελέσει ως και μία συνταγή για αυτό το επεισόδιο.
- Το one_judge_participation_per_episode_trg έχει την ίδια χρησιμότητα με το παραπάνω αλλά ελέγχει την συμμετοχή ενός chef ως κριτή.
- Το cheff_recipe_can_cook_trg εξασφαλίζει ότι ο κάθε μάγειρας μπορεί να εκτελέσει συνταγές μόνο από τις κουζίνες στις οποίες ειδικεύεται.
- Τα check_consecutive_participation_chef_trg, check_consecutive_participation_judge_trg χ ρησιμοποιούνται ώστε να μην μπορεί ένας μάγειρας να συμμετάσχει ως διαγωνιζόμενος σε πάνω από 3 συνεχόμενα επεισόδια και αντίστοιχα ούτε ως κριτής.

Βρίσκονται στο github
cooking-contest-p127/data.txt at main · mariafytrou/cooking-contest-p127 · GitHub

Εφαρμογή

Για την εφαρμογή δημιουργήθηκε ένα προσχέδιο με τα δεδομένα που θα χρειαστούν.

Δημιουργήθηκε η οντότητα User στην οποία αποθηκεύονται: username, password, admin/cook, chefid (αν είναι cook).

Στην συνέχεια έχουμε έναν πίνακα με όλα τα δυνατά permission που μπορεί να έχει κάποιος.

MANAGE_PERSONAL_DATA: Δυνατότητα επεξεργασίας μόνο προσωπικών στοιχείων.

MANAGE_ALL_DATA: Δυνατότητα επεξεργασίας όλων των δεδομένων της βάσης.

MANAGE_OWNED_RECIPES: Δυνατότητα επεξεργασίας μόνο των συνταγών που ανήκουν στον user.

BACKUP_DATA: Δυνατότητα Backup data.

RESTORE_DATA: Δυνατότητα restore data.

Με τον πίνακα hasPermission γίνεται ο συσχετισμός ανάμεσα σε user και permissions. Ο συσχετισμός γίνεται αυτόματα με το εξής insertion:

```
INSERT INTO hasPermission (userID, permissionID)SELECT u.userID, p.permissionID
FROM User u
CROSS JOIN (
   SELECT 1 AS permissionID
   UNION ALL
   SELECT 3 AS permissionID
) p
WHERE u.role = 'Cook';


INSERT INTO hasPermission (userID, permissionID)
SELECT u.userID, p.permissionID
FROM User u
CROSS JOIN (
   SELECT 1 AS permissionID
   UNION ALL
   SELECT 2 AS permissionID
   UNION ALL
   SELECT 4 AS permissionID
   UNION ALL
   SELECT 5 AS permissionID
) p
WHERE u.role = 'Admin';
```

Στο MANAGE_OWNED_RECIPES μπορούμε να λάβουμε τις συνταγές που μπορεί να τροποποιήσει ένας σεφ με το εξής query:

```
FROM
   recipe r
JOIN
   isabletocook ic ON r.RecipeID = ic.recipeID
WHERE
   ic.chefID = “number input”;
```

Github link

Το url για την εργασία είναι:

https://github.com/mariafytrou/cooking-contest-p127