

Software Documentation

תאריך 12.01.2023

סינון ה-Data

המקורות שלנו לקוחים מאתרים הבאים -

המקור לכל הדירוגים עם הביקורות שלקוחים מאתר food.com לא נשתמש בכל המידע ונצטרך להצליב מקורות כדי להשלים את התמונה המלאה. הקבצים שהשתמשנו הם reviews.csv, recipes.csv וחילצנו מהם שדות מסוימים.

<https://www.kaggle.com/datasets/irkaal/foodcom-recipes-and-reviews>

עוד קישור שמשלים נתונים חסרים נוספים על המתכונים. אנחנו השתמשנו בקובץ לא נשתמש בכל הנתונים של Recipe1M+ dataset קישור ראשון של האתר עצמו <http://pic2recipe.csail.mit.edu/> קישור של הקובץ שנמצא באתר (צריך להרשם אליו) http://data.csail.mit.edu/im2recipe/recipes_with_nutritional_info.json

דאטסט שמכיל את כל המצרכים הרלוונטיים והערכים התזונתיים שלהם ביחידות של 100 גרם <https://www.kaggle.com/datasets/trolukovich/nutritional-values-for-common-foods-and-products>

תהליך החילוץ

נפטרו משדות מיותרים כמו שדה food standards שמופיע בקישור השני או משאר הערכים התזונתיים שיש בקישור הראשון. התמקדנו ברק 7 ערכים תזונתיים שיש גם בקישור השני, כי שם יש רק ערכים תזונתיים מסויימים. השתמשנו באלגוריתם SequenceMatcher כדי למצוא התאמה בין המצרכים שיש בקישור השני לקישור השלישי. כמו כן, השתמשנו בספריית pandas בפייתון כדי לעשות פיצול של כל מתכון ושל שמות המצרכים, כמויות והיחידות. זה מופיע בקישור השני שצויין לעיל, בנוסף נאלצנו לבצע חיתוכים כי לא כל הדאטה מגיע עם כמויות נכונות של מצרכים ושל יחידות או כמויות בכלל אז מרשימה של 500000 של מתכונים זה הצטמצם ל-29000 מתכונים בערך. הפיצול של שורת המצרכים לכל מתכון נקראת פעולת explode והיא שימושית לחץ דאטה מסוג זה. הדאטה שיש ברשותנו היא גרסה מצומצמת של הדאטה המקורי ואותו שמרנו בחשבון הגיטהאב. השתמשנו בפונקציות שנמצאות ב setup.js כמו generateRandomPassword ו updatePassword והן לא מופעלות. כמו כן, יש סקריפט בפייתון בתקיית remains רק להמחשת תהליך החילוץ (אי אפשר להריץ).

הסכמה הסופית

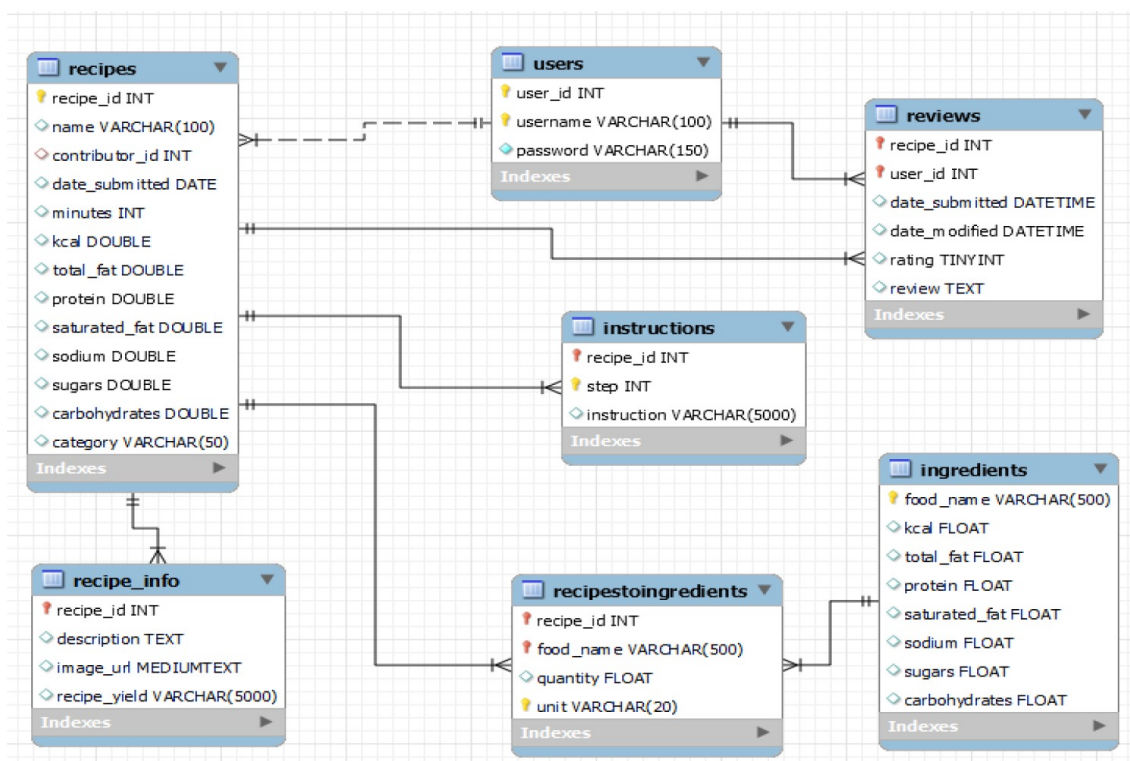


Figure 1: The scheme

הסכמה מורכבת מ-7 טבלאות ולכל טבלה נפרט את המפתחות שלה והקשרים שלה לשאר הטבלאות. יש את טבלת המשתמשים לכל משתמש יש ניק וסיסמה והוא מסוגל ליצור מספר מתכונים ולכתוב כמה ביקורות במקרה שלנו הגבלנו ביקורת אחת לכל מתכון. המפתחות שמסומנים באדום הם מפתחות שלקוחים מטבלה אחרת. בנוסף לכך, הסיסמא של היוזרים לא יכולה להיות null אבל אנו מגבילים ברמת האפליקציה בכל מקרה. כמות השורות של כל הטבלאות הוא בערך 400,000 בגלל פעולת explode שהכפילה את מספר המתכונים. בין המשתמשים לבין המתכונים או ביקורות מתקיים יחס One to Many וגם זה נכון לגבי מתכונים לביקורות. יחס זה מתקיים גם בין מתכונים להוראות כי לכל מתכון יש כמה הוראות וזה הדרך שבחרנו לאחסן את הדאטה. נשים לב כי יש יחס Many to Many במקרה של מצרכים ומתכונים כי לכל מצרך יש הרבה מתכונים ולכל מתכון יש הרבה מצרכים ולכן נוצרה טבלת אמצע שמכילה את המזהה של המתכון, המצרכים שיכולים לחזור על עצמם ביחידות מידה שונות. מכאן נאלצנו להפוך 3 שדות מתוך הטבלה של recipestoingredients למפתחות כדי שלא יוצרו חזרות. הטבלה האחרונה היא טבלת ה recipe_info שמשתמשת ב MEDIUMTEXT, כיוון שהאפליקציה מאחסנת תמונות בכך שהתוכנה ממירה את קבצי התמונה לבסיס 64 ולאחר מכן שומרת על שדה שמאפשר נפח של 16 מגה בייט לכל התמונות של המשתמש עבור אותו מתכון. לסיכום, זה הסכמה שהגענו אליה אחרי ניסוי וטעייה כמו למשל ביטלנו כפילויות של שמות המשתמשים. יתכן שיש כאן דברים שהמחבר שכח לפרט. איננו משתמשים בתאריכים חוץ מלתעד את זמן הכנסת הביקורות בשאלת הכנסה עם NOW(). לכל מתכון יש תאריך הגשה שאפשר למצוא אותו בחיפוש המתכון.

שאלות

נציין 3 שאלות מורכבות ו-3 שאלות פשוטות, כמובן שיש עוד שאלות כמו הכנסה או עדכון.

1. שאלות פשוטות שמחזירות את כל המצרכים או מתכונים ששם דומה לתפוח.
זאת שאלת טמפלט, כלומר משתמשים במשתנה searchTerm כדי לבצע חיפוש שמציג פרטים כמו בגוגל.

```
SELECT * FROM ingredients WHERE food_name LIKE '%apple%'
SELECT * FROM recipes WHERE name LIKE '%apple%'
```

2. שאלות טמפלט שמביאה את הפרטים של אותו משתמש/מתכון, במקור השאלה עם סימן שאלה בסוף.
Select * from users where username = 'DancerIO'
Select * from recipes where name = 'Apple Crisp'

3. שאלות טמפלט שמביאות את המצרכים כפול ערכי חישוב בהמשך ואז איחוד עם שאר המצרכים.
SELECT 10 * kcal, 10 * total_fat, ..., 10 * carbohydrates
FROM ingredients WHERE food_name = 'app' UNION...

A. שאלת טמפלט מורכבת שמבוססת על השאלה הראשונה היא מופעלת שיש פילטר בחיפוש המתכון
SELECT * FROM recipes WHERE name LIKE '%apple%'
AND recipe_id IN (SELECT recipe_id
FROM (SELECT recipe_id, AVG(rating) as avg_rating FROM reviews GROUP BY recipe_id)
as ratings WHERE avg_rating >= 4)
AND kcal <= 1000
AND recipe_id IN (SELECT recipe_id
FROM (SELECT recipe_id, max(step) as max_step FROM instructions GROUP BY recipe_id)
as instructions WHERE max_step <= 10)
AND recipe_id IN (SELECT recipe_id
FROM recipestoingredients WHERE food_name LIKE '%apple%')

B. שאלת טמפלט מורכבת שמחזירה את המתכונים של המשתמש בצורת דאטה ורשימות בצורת קובץ JSON כי הדאטה שאנחנו מקבלים הוא מתכון ואוסף ההוראות או ביקורות. עשינו כך כדי להציג בבת אחת את כל המתכונים בדף הבית עם כל המידע עליהם שנוכל לעבור עליהם והשאלה יחסית מהירה. הטמפלט הוא שם המשתמש ואנחנו עושים חיתוך כי אין לנו את המזהה של המשתמש, רק את שם המשתמש.

```
SELECT r.name, ri.description, ri.image_url,
(SELECT JSON_ARRAYAGG(JSON_OBJECT('rating', rv.rating, 'review', rv.review))) AS reviews,
(SELECT JSON_ARRAYAGG(JSON_OBJECT('step', ins.step, 'instruction', ins.instruction)))
FROM instructions ins WHERE r.recipe_id = ins.recipe_id GROUP BY ins.recipe_id) AS instructions
FROM recipes r
JOIN users u ON r.contributor_id = u.user_id
LEFT JOIN reviews rv ON r.recipe_id = rv.recipe_id
LEFT JOIN recipe_info ri ON r.recipe_id = ri.recipe_id
WHERE u.username = 'DancerIO'
GROUP BY r.recipe_id
```

C. שאלת מורכבת שמחזירה את 100 המתכונים הפופולרים ביותר בצורת דאטה ורשימות דומה לשאלת הקודמת רק שאנחנו מבצעים מיון ויש גם כמות הביקורות וגם פחות LEFT JOIN בשביל יעילות. השאלה מופעלת בדף הבית שמציג את כל המידע הרלוונטי על אותם מתכונים.

```

SELECT r.name, ri.description, ri.image_url,
(SELECT JSON_ARRAYAGG(JSON_OBJECT('rating', rv.rating, 'review', rv.review)))
FROM reviews rv WHERE r.recipe_id = rv.recipe_id GROUP BY rv.recipe_id) AS reviews
(SELECT JSON_ARRAYAGG(JSON_OBJECT('step', ins.step, 'instruction', ins.instruction)))
FROM instructions ins WHERE r.recipe_id = ins.recipe_id GROUP BY ins.recipe_id) AS instructions
FROM recipes r
JOIN users u ON r.contributor_id = u.user_id
LEFT JOIN recipe_info ri ON r.recipe_id = ri.recipe_id
WHERE r.recipe_id IN (SELECT test.recipe_id
FROM (SELECT recipe_id, COUNT(*) as num_reviews
FROM reviews
GROUP BY recipe_id
ORDER BY num_reviews DESC
LIMIT 100) as test)

```

כמובן יש עוד שאילתה מורכבת שמופיעה בפונקציה `getRecipeByName` בקובץ `database.js`. בהנחה סבירה שהשאילתות הקודמות נחשבות מורכבות, מכיוון שהיא דומה לאלו ומופעלת בדף החיפוש.

תיעוד הקוד

יש את הקובץ `setup.js` שהמטרה שלו היא להתקין את הפרוייקט אבל ניתן להריץ `npm install` שמתקין את כל החבילות הנדרשות ולאחר מכן מפעיל את הקובץ `setup.js` שטוען את `JSON`ים שהפקנו ב`row_data`. כמו כן, הקובץ `facecook.js` מייצג את הלוגיקה של האתר ואותו מריצים כדי להקים את האתר. הקובץ `database.js` מייצג את ההתממשקות עם הדאטהבייס והקובץ הקודם קורא לממשק הדאטהבייס כדי להפעיל פונקציות. אחרון וזה ה-GUI שנמצא בתיקיית `Public` ומכיל סקריפטים להצגת האתר וכל מיני עיצובים ב-CSS. התיקייה מכילה גם קבצי `html` שמציגים את האתר עצמו בצורה בסיסית. יש גם קוד ישן בפייטון אך הוא לא חלק מהפרוייקט ומוצג בהקשר לחילוף דאטה. רוב הקבצים שציינתי הם מתועדים כולל הקבצי סקריפטים שנמצאים ב`Public\js`. התחברות הפרוייקט מתבצעת באמצעות ספרייה `mysql2` או `promise-mysql` כי הקוד יפסיק לתפקד. `const mysql = require('mysql');` לא לשנות אותה לספרייה

בעיות אפשריות בפרוייקט

אנחנו מתחברים לשרת דרך פורט 3000 ויכול להיות שלמשתמש יהיה בעיה עם הפורט הנ"ל
`app.listen("3000", () => {console.log("Server is running on port 3000");});`
אם המשתמש נתקל בבעיית אתחול השרת בגלל שגיאת אימות בגלל גרסה מתקדמת של MySQL
`ALTER USER 'username'@'localhost' IDENTIFIED WITH mysql_native_password BY 'yourpassword';`

התחברות ויצירת סכמה, אנו משנים את זה ידנית בפונקציה `createSQLDB` ב`setup.js` בקטע הקוד
`connection.changeUser({database: 'recipesdb'}, function(err) {if (err) throw err;});`
ובפונקציה `createSchema` שנמצאת גם באותו קובץ התקנה
`await connection.query('CREATE SCHEMA IF NOT EXISTS recipesdb'...` ב`database.js` ובקובץ `database.js` אנחנו משנים את השם של הסכמה בהתאם למה שנדרש מאיתנו
`const dbConnection = mysql.createConnection({`
`host: "localhost",`
`user: secrets.DB_USERNAME,`
`password: secrets.DB_PASSWORD,`
`database: "recipesdb",});`
וקובץ `secrets.json` שיש בתוכו `{ "DB_USERNAME": "yourusername", "DB_PASSWORD": "yourpassword" }`

לסיום זה חשבון הגיטהאב שניתן לעקוב אחריו כדי להבין את הקוד דרך הקומיטים.
<https://github.com/eyalmichon/FaceCook>