



Βάσεις Δεδομένων

Αδαμόπουλος Θεόδωρος (03119099)

Καραγιάννης Ευθύμιος (03119434)

Ομάδα 11

Χειμερινό εξάμηνο 2023-2024

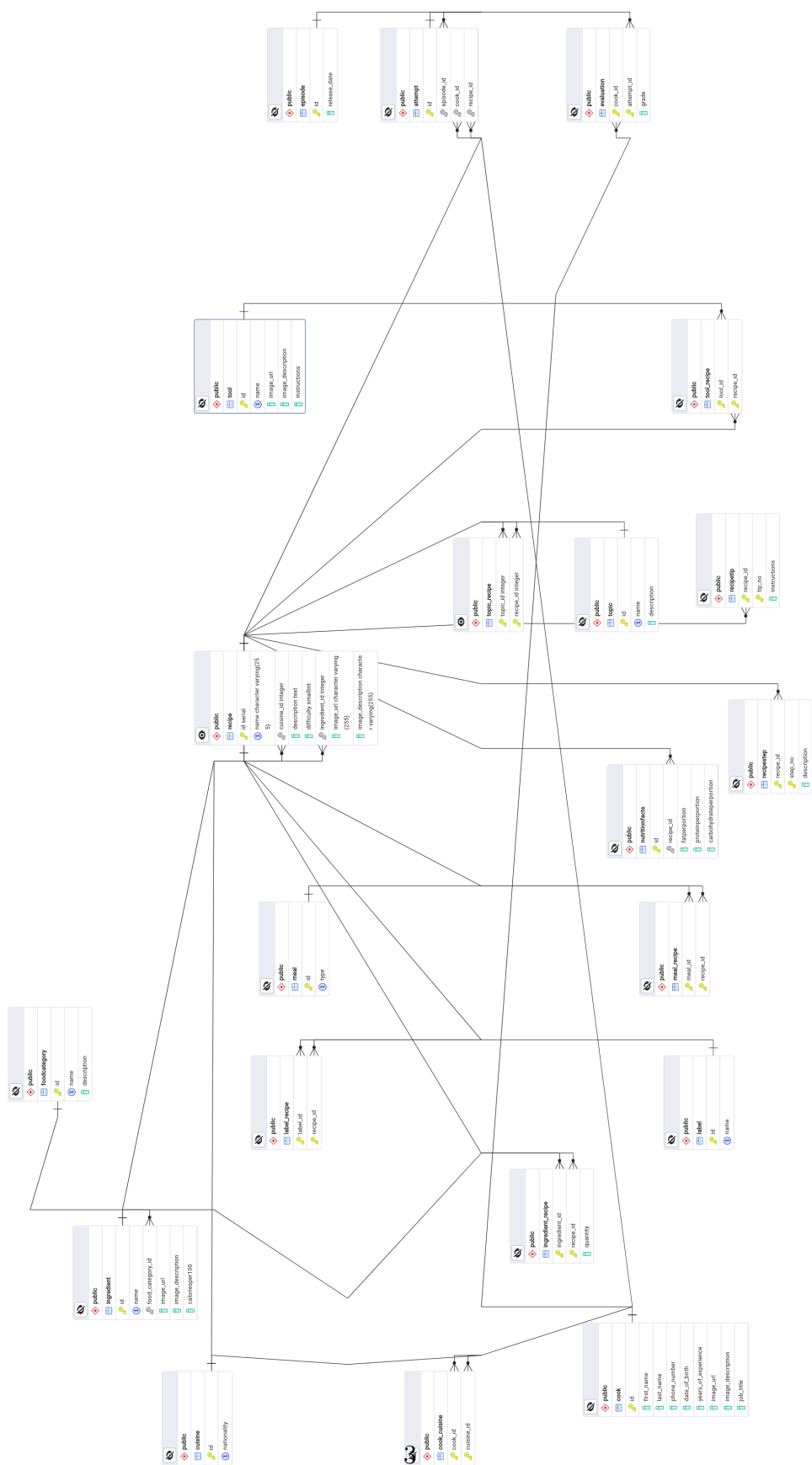
1 Εισαγωγή

Στην εργασία αυτή καλούμαστε να υλοποιήσουμε μία Βάση Δεδομένων για έναν δημοφιλή διαγωνισμό μαγειρικής. Πιο συγκεκριμένα χρειάζεται να σχεδιάσουμε και να υλοποιήσουμε το σύστημα αποθήκευσης και διαχείρισης των πληροφοριών που απαιτούνται για τη λειτουργία του διαγωνισμού αναφορικά με τις συνταγές, τα υλικά και τον εξοπλισμό που απαιτείται για την λειτουργία του διαγωνισμού.

Github repository: <https://github.com/TheodoreAdamopoulos/Database-2024/tree/master>

2 Σχεδίαση

Στη συνέχεια δίνεται το ER διάγραμμα με βάση το οποίο υλοποιήθηκε το DataBase:



3 Υλοποίηση

Cook(id, first_name, last_name, phone_number, date_of_birth, years_of_experience, image_url, image_description, job_title)
Recipe(id, name, cuisine_id, description, difficulty, ingredient_id, image_url, image_description)
RecipeStep(recipe_id, step_no, description)
RecipeTip(recipe_id, tip_no, instructions)
Cuisine(id, nationality)
Cook_Cuisine(cook_id, cuisine_id)
Episode(id, release_date)
Attempt(id, episode_id, cook_id, recipe_id)
Evaluation(cook_id, attempt_id, grade)
Ingredient(id, name, food_category_id, image_url, image_description, caloriesPer100)
Ingredient_Recipe(ingredient_id, recipe_id, quantity)
Label(id, name)
Label_Recipe(label_id, recipe_id)
FoodCategory(id, name, description)
Meal(id, type)
Meal_Recipe(meal_id, recipe_id)
NutritionFacts(id, recipe_id, sequence, fatPerPortion, proteinPerPortion, carbohydratePerPortion)
Tool(id, name, image_url, image_description, instructions)
Tool_Recipe(tool_id, recipe_id)
Topic(id, name, description)
Topic_Recipe(topic_id, recipe_id)

Σε αυτο το σημείο δίνονται τα links για τα αρχεία στο github:

DDL script: <https://github.com/TheodoreAdamopoulos/Database-2024/blob/master/sql/ddl.sql>

DML script: <https://github.com/TheodoreAdamopoulos/Database-2024/blob/master/sql/dml.sql>

3.1

Για την εξασφάλιση της ορθότητας της Βάσης Δεδομένων που υλοποιούμε, ορίζουμε τους παρακάτω περιορισμούς:

- **Primary Keys:**

Primary Keys: Each table's primary key ensures unique records.

Cuisine(id) FoodCategory(id) Ingredient(id) Recipe(id) Meal(id) Label(id) Tool(id)
NutritionFacts(id) Topic(id) Cook(id) Episode(id) Attempt(id) Evaluation(id)

Foreign Keys: Maintain relationships between tables and ensure referential integrity.
Attempt(episode_id) references Episode(id) Attempt(cook_id) references Cook(id) Attempt(recipe_id) references Recipe(id) Evaluation(cook_id) references Cook(id) Evaluation(attempt_id) references Attempt(id) RecipeTip(recipe_id) references Recipe(id) RecipeStep(recipe_id) references Recipe(id) Meal_Recipe(meal_id) references Meal(id) Meal_Recipe(recipe_id) references Recipe(id) Label_Recipe(label_id) references Label(id) Label_Recipe(recipe_id) references Recipe(id) Tool_Recipe(tool_id) references Tool(id) Tool_Recipe(recipe_id) references Recipe(id) NutritionFacts(recipe_id) references Recipe(id) Ingredient_Recipe(ingredient_id) references Ingredient(id) Ingredient_Recipe(recipe_id) references Recipe(id) Topic_Recipe(recipe_id) references Topic(id) Topic_Recipe(recipe_id) references Recipe(id) Cook_Cuisine(cook_id) references Cook(id) Cook_Cuisine(cuisine_id) references Cuisine(id)

- **Referential Integrity**

Foreign Key Constraints: Ensure valid references across tables. Attempt(episode_id) references Episode(id) Attempt(cook_id) references Cook(id) Attempt(recipe_id) references Recipe(id) Evaluation(cook_id) references Cook(id) Evaluation(attempt_id) references Attempt(id)

- **Domain Integrity**

Check Constraints: Ensure valid values within columns. Recipe.difficulty must be between 1 and 5. Evaluation.grade must be between 1 and 5.

- **User-Defined Constraints** Δημιουργούμε triggers για την εξασφάλιση των παρακάτω:

- Δεν υπάρχει ένας κριτής στο ίδιο επεισόδιο πάνω από μία φορά.
- Δεν συμμετέχει κάποιος μάγειρας/κριτής/ εθνική κουζίνα/ συνταγή, συνεχόμενα σε περισσότερα από 3 επεισόδια.
- Το πλήθος των συμβουλών για μία συνταγή είναι από 0 έως 3.
- Δεν υπάρχουν πάνω από 10 επεισόδια σε κάθε χρονιά.
-
-

3.2 Indexes

Χρησιμοποιούμε indexes στα παρακάτω σημεία:

- **Indexes on Cook Table**

Indexes:

idx_cook_last_name: Index on the last_name column.

Αιτιολόγηση : Ερωτήματα Αναζήτησης: Τα ερωτήματα συχνά αναζητούν μάγειρες με βάση το επώνυμό τους, όπως όταν εμφανίζουν μια λίστα μαγείρων αλφαβητικά ή αναζητούν έναν μάγειρα κατά επώνυμο. Ένα ευρετήριο στη στήλη last_name θα επιταχύνει αυτές τις λειτουργίες.

- **Indexes on Episode Table**

Indexes:

idx_episode_release_date: Index on the release_date column.

Αιτιολόγηση:

Ταξινόμηση και Φιλτράρισμα: Τα ερωτήματα συχνά ταξινομούν τα επεισόδια κατά ημερομηνία κυκλοφορίας ή τα φιλτράρουν βάσει εύρους ημερομηνιών. Ένα ευρετήριο στη στήλη release_date θα επιταχύνει σημαντικά αυτές τις λειτουργίες.

- **Indexes on Recipe Table**

Indexes:

idx_recipe_cuisine_id: Index on the cuisine_id column. idx_recipe_difficulty: Index on the difficulty column.

Αιτιολόγηση:

Φιλτράρισμα Ερωτημάτων: Τα ερωτήματα συχνά φιλτράρουν συνταγές κατά τύπο κουζίνας ή επίπεδο δυσκολίας. Τα ευρετήρια στις στήλες cuisine_id και difficulty θα βελτιστοποιήσουν αυτές τις λειτουργίες φιλτραρίσματος.

- **Indexes on Attempt Table**

Indexes:

idx_attempt_episode_id: Index on the episode_id column. idx_attempt_cook_id: Index on the cook_id column.

Αιτιολόγηση:

Συνενώσεις (Join) Πινάκων: Τα ερωτήματα συχνά συνενώνουν τον πίνακα Attempt με τους πίνακες Episode και Cook για να ανακτήσουν απόπειρες για συγκεκριμένα επεισόδια ή μάγειρες. Τα ευρετήρια στις στήλες episode_id και cook_id θα επιταχύνουν αυτές τις λειτουργίες συνένωσης.

- **Indexes on Evaluation Table**

Indexes:

idx_evaluation_cook_id: Index on the cook_id column. idx_evaluation_attempt_id: Index on the attempt_id column.

Αιτιολόγηση:

Συνενώσεις (Join) Πινάκων: Τα ερωτήματα συχνά συνενώνουν τον πίνακα Evaluation με τους πίνακες Cook και Attempt για να ανακτήσουν αξιολογήσεις για συγκεκριμένους κριτές ή απόπειρες. Τα ευρετήρια στις στήλες cook_id και attempt_id θα βελτιστοποιήσουν αυτές τις λειτουργίες συνένωσης.

4 Ερωτήματα

4.0.1 Ερώτημα 3.6


Τρέχουμε το ερώτημα 3.6 αρχικά χωρίς την χρήση indexes και λαμβάνουμε τα παρακάτω αποτελέσματα:

QUERY PLAN	
	text
3	Sort Key: (count(*)) DESC
4	Sort Method: top-N heapsort Memory: 25kB
5	-> HashAggregate (cost=1482.04..1780.00 rows=29796 width=16) (actual time=1.698..1.893 rows=90 loops=1)
6	Group Key: lr1.label_id, lr2.label_id
7	Batches: 1 Memory Usage: 1561kB
8	-> Merge Join (cost=317.16..1258.57 rows=29796 width=8) (actual time=0.231..1.286 rows=1096 loops=1)
9	Merge Cond: (a.recipe_id = lr1.recipe_id)
10	-> Index Only Scan using idx_attempt_recipe_id on attempt a (cost=0.15..34.65 rows=700 width=4) (actual time=0.038..0.341 rows=700 loop...)
11	Heap Fetches: 700
12	-> Materialize (cost=317.01..796.51 rows=8513 width=16) (actual time=0.177..0.521 rows=1094 loops=1)
13	-> Merge Join (cost=317.01..775.23 rows=8513 width=16) (actual time=0.173..0.384 rows=129 loops=1)
14	Merge Cond: (lr1.recipe_id = lr2.recipe_id)
15	Join Filter: (lr1.label_id < lr2.label_id)
16	Rows Removed by Join Filter: 294
17	-> Sort (cost=158.51..164.16 rows=2260 width=8) (actual time=0.097..0.111 rows=165 loops=1)
18	Sort Key: lr1.recipe_id
19	Sort Method: quicksort Memory: 32kB
20	-> Seq Scan on label_recipe lr1 (cost=0.00..32.60 rows=2260 width=8) (actual time=0.024..0.045 rows=165 loops=1)
21	-> Sort (cost=158.51..164.16 rows=2260 width=8) (actual time=0.067..0.102 rows=422 loops=1)
22	Sort Key: lr2.recipe_id
23	Sort Method: quicksort Memory: 32kB
24	-> Seq Scan on label_recipe lr2 (cost=0.00..32.60 rows=2260 width=8) (actual time=0.007..0.026 rows=165 loops=1)
25	Planning Time: 0.983 ms

Στη συνέχεια ορίζουμε τα παρακάτω indexes :

```
CREATE INDEX idx_label_recipe_recipe_id ON Label_Recipe(recipe_id);
CREATE INDEX idx_label_recipe_label_id ON Label_Recipe(label_id);
CREATE INDEX idx_attempt_recipe_id ON Attempt(recipe_id);
```

και τρέχουμε πάλι το ερώτημα παίρνοντας τα αντίστοιχα traces:

	QUERY PLAN	
	text	
1	Limit (cost=32.12..32.13 rows=3 width=16) (actual time=1.193..1.201 rows=3 loops=1)	
2	-> Sort (cost=32.12..32.60 rows=192 width=16) (actual time=1.189..1.196 rows=3 loops=1)	
3	Sort Key: (count(*)) DESC	
4	Sort Method: top-N heapsort Memory: 25kB	
5	-> HashAggregate (cost=27.72..29.64 rows=192 width=16) (actual time=1.119..1.144 rows=90 loops=1)	
6	Group Key: lr1.label_id, lr2.label_id	
7	Batches: 1 Memory Usage: 40kB	
8	-> Hash Join (cost=10.73..26.28 rows=192 width=8) (actual time=0.263..0.643 rows=1096 loops=1)	
9	Hash Cond: (a.recipe_id = lr1.recipe_id)	
10	-> Seq Scan on attempt a (cost=0.00..11.00 rows=700 width=4) (actual time=0.010..0.092 rows=700 loops=1)	
11	-> Hash (cost=10.04..10.04 rows=55 width=16) (actual time=0.238..0.243 rows=129 loops=1)	
12	Buckets: 1024 Batches: 1 Memory Usage: 15kB	
13	-> Hash Join (cost=4.71..10.04 rows=55 width=16) (actual time=0.093..0.200 rows=129 loops=1)	
14	Hash Cond: (lr1.recipe_id = lr2.recipe_id)	
15	Join Filter: (lr1.label_id < lr2.label_id)	
16	Rows Removed by Join Filter: 294	
17	-> Seq Scan on label_recipe lr1 (cost=0.00..2.65 rows=165 width=8) (actual time=0.006..0.021 rows=165 loops=1)	
18	-> Hash (cost=2.65..2.65 rows=165 width=8) (actual time=0.057..0.058 rows=165 loops=1)	
19	Buckets: 1024 Batches: 1 Memory Usage: 15kB	
20	-> Seq Scan on label_recipe lr2 (cost=0.00..2.65 rows=165 width=8) (actual time=0.005..0.019 rows=165 loop...	
21	Planning Time: 1.130 ms	
22	Execution Time: 1.272 ms	

4.0.2 Ερώτημα 3.8


Όπως και στο προηγούμενο ερώτημα, τρέχουμε το 3.8 αρχικά χωρίς την χρήση indexes και λαμβάνουμε τα παρακάτω αποτελέσματα:

QUERY PLAN		text	
1	Limit	(cost=460.87..460.88 rows=1 width=12) (actual time=2.177..2.181 rows=1 loops=1)	
2	-> Sort	(cost=460.87..466.52 rows=2260 width=12) (actual time=2.176..2.179 rows=1 loops=1)	
3	Sort Key:	(count(tr.tool_id)) DESC	
4	Sort Method:	top-N heapsort Memory: 25kB	
5	-> HashAggregate	(cost=426.97..449.57 rows=2260 width=12) (actual time=2.127..2.157 rows=70 loops=1)	
6	Group Key:	e.id	
7	Batches:	1 Memory Usage: 121kB	
8	-> Merge Join	(cost=265.27..387.42 rows=7910 width=8) (actual time=0.653..1.538 rows=2552 loops=1)	
9	Merge Cond:	(a.recipe_id = tr.recipe_id)	
10	-> Sort	(cost=106.77..108.52 rows=700 width=8) (actual time=0.545..0.622 rows=700 loops=1)	
11	Sort Key:	a.recipe_id	
12	Sort Method:	quicksort Memory: 57kB	
13	-> Hash Join	(cost=60.85..73.69 rows=700 width=8) (actual time=0.070..0.357 rows=700 loops=1)	
14	Hash Cond:	(a.episode_id = e.id)	
15	-> Seq Scan on attempt a	(cost=0.00..11.00 rows=700 width=8) (actual time=0.012..0.088 rows=700 loops=1)	
16	-> Hash	(cost=32.60..32.60 rows=2260 width=4) (actual time=0.036..0.036 rows=70 loops=1)	
17	Buckets:	4096 Batches: 1 Memory Usage: 35kB	
18	-> Seq Scan on episode e	(cost=0.00..32.60 rows=2260 width=4) (actual time=0.007..0.016 rows=70 loop...	
19	-> Sort	(cost=158.51..164.16 rows=2260 width=8) (actual time=0.102..0.285 rows=2555 loops=1)	
20	Sort Key:	tr.recipe_id	
21	Sort Method:	quicksort Memory: 38kB	
22	-> Seq Scan on tool_recipe tr	(cost=0.00..32.60 rows=2260 width=8) (actual time=0.021..0.058 rows=289 loops=...	
23	Planning Time: 0.598 ms		

Έπειτα ορίζουμε τα παρακάτω indexes :

```
CREATE INDEX idx_attempt_episode_id ON Attempt(episode_id);
CREATE INDEX idx_tool_recipe_recipe_id ON Tool_Recipe(recipe_id);
```

και επαναλαμβάνουμε τις μετρήσεις:

	QUERY PLAN	
	text	
1	Limit (cost=139.68..139.68 rows=1 width=12) (actual time=1.267..1.269 rows=1 loops=1)	
2	-> Sort (cost=139.68..142.21 rows=1012 width=12) (actual time=1.266..1.268 rows=1 loops=1)	
3	Sort Key: (count(tr.tool_id)) DESC	
4	Sort Method: top-N heapsort Memory: 25kB	
5	-> HashAggregate (cost=124.50..134.62 rows=1012 width=12) (actual time=1.238..1.255 rows=70 loops=1)	
6	Group Key: e.id	
7	Batches: 1 Memory Usage: 73kB	
8	-> Hash Join (cost=69.35..119.44 rows=1012 width=8) (actual time=0.197..0.805 rows=2552 loops=1)	
9	Hash Cond: (a.recipe_id = tr.recipe_id)	
10	-> Hash Join (cost=60.85..73.69 rows=700 width=8) (actual time=0.057..0.286 rows=700 loops=1)	
11	Hash Cond: (a.episode_id = e.id)	
12	-> Seq Scan on attempt a (cost=0.00..11.00 rows=700 width=8) (actual time=0.011..0.068 rows=700 loops=1)	
13	-> Hash (cost=32.60..32.60 rows=2260 width=4) (actual time=0.040..0.040 rows=70 loops=1)	
14	Buckets: 4096 Batches: 1 Memory Usage: 35kB	
15	-> Seq Scan on episode e (cost=0.00..32.60 rows=2260 width=4) (actual time=0.005..0.019 rows=70 loop...	
16	-> Hash (cost=4.89..4.89 rows=289 width=8) (actual time=0.135..0.135 rows=289 loops=1)	
17	Buckets: 1024 Batches: 1 Memory Usage: 20kB	
18	-> Seq Scan on tool_recipe tr (cost=0.00..4.89 rows=289 width=8) (actual time=0.012..0.052 rows=289 loops...	
19	Planning Time: 0.580 ms	
20	Execution Time: 1.337 ms	