BUSENUR GÖKLER

Ödev 4 – İleri Cypher Sorguları ile Use Case Geliştirme

Tarif ve malzeme domaini, çok sayıda varlık (malzeme, tarif, tat, teknik vs.) ve bu varlıklar arasında zengin ilişki türleri içermektedir. Bu nedenle bu domain, gerçek hayatla bağlantılı olduğundan ve anlaşılması kolay olduğundan birçok karmaşık senaryoyu modellemek için ideal bir örnektir.

Tarif ve malzeme domain'i üzerinde çalışarak, yemek tarifleri, malzemeler, lezzetler ve pişirme teknikleri arasındaki çok boyutlu ilişkileri inceleyeceğiz.

Bu ödevde, Neo4j graph database'de ileri seviye Cypher sorgu tekniklerini kullanarak use case'ler geliştirdim. Tarif(Recipe) ve malzeme(İngredients) domain'i üzerinde çalışarak, UNION ALL ile veri birleştirme, WITH, UNWIND, dinamik property erişimi, aggregation işlemleri, graph algoritmaları ve APOC gibi gelişmiş teknikleri uyguladım. EXPLAIN/PROFILE komutları ile performans analizi yaptım.

Ödevde, daha önce oluşturduğum tarif (recipe) graph database'ini kullanacağım. Bu database aşağıdaki node türlerini ve ilişkileri içermektedir:

Node Türleri:

- **Recipe**: {id, name, difficulty, time_minutes}
- **Ingredient**: {id, name, category}
- **Flavor**: {id, name}
- **Technique**: {id, name}

Relationships Türleri:

- **CONTAINS**: (Recipe) → (Ingredient)
- **HAS_FLAVOR**: (Recipe) → (Flavor)
- **USES_TECHNIQUE**: (Recipe) → (Technique)
- **CAN_REPLACE**: (Ingredient) → (Ingredient)

Hem tatlı hem de kremamsı tarifleri tek sorguda getirebilir miyiz?

MATCH (r:Recipe)-[:HAS_FLAVOR]->(f:Flavor {name: "Tatlı"}) // Tatlı tarifleri bulur.

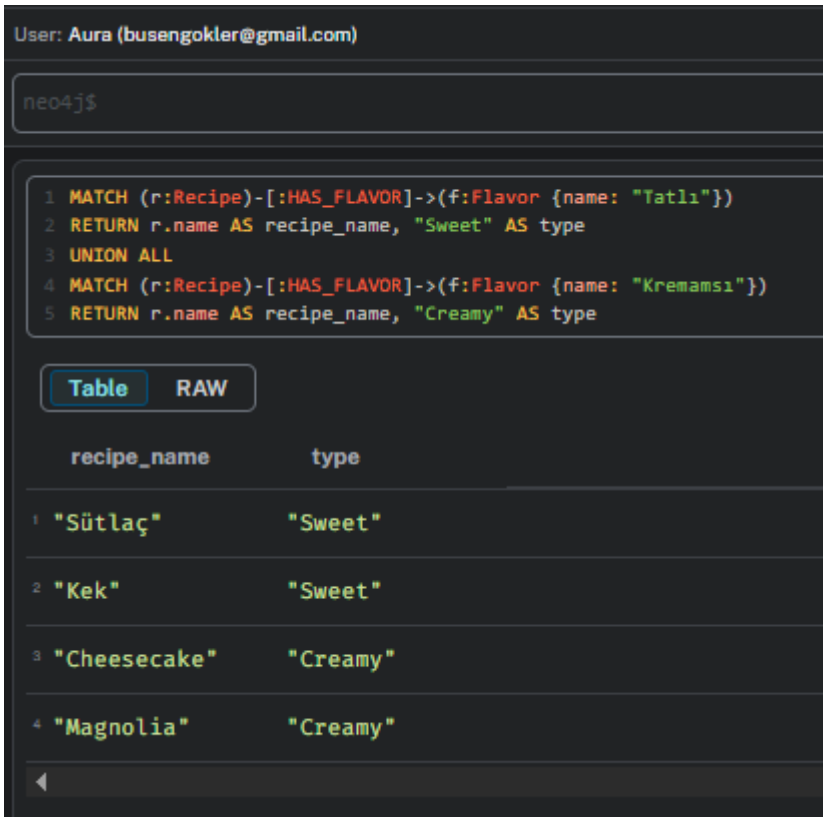
RETURN r.name AS recipe_name, "Sweet" AS type

UNION ALL // sonuclari birleştirir.

MATCH (r:Recipe)-[:HAS_FLAVOR]->(f:Flavor {name: "Kremamsı"}) // Kremamsı tarifleri bulur.

RETURN r.name AS recipe_name, "Creamy" AS type

UNION ALL: Farklı Sorgu Sonuçlarını Birleştirir.



En çok malzeme kullanan 3 tarifi bulup, bunların ortalama süresini hesaplayabilir miyiz?

MATCH (r:Recipe)-[:CONTAINS]->(i:Ingredient)

// Malzeme sayısına göre sıralanması ve ilk 3 tarifin seçilmesi

WITH r, count(i) AS ingredient_count

ORDER BY ingredient_count DESC

LIMIT 3

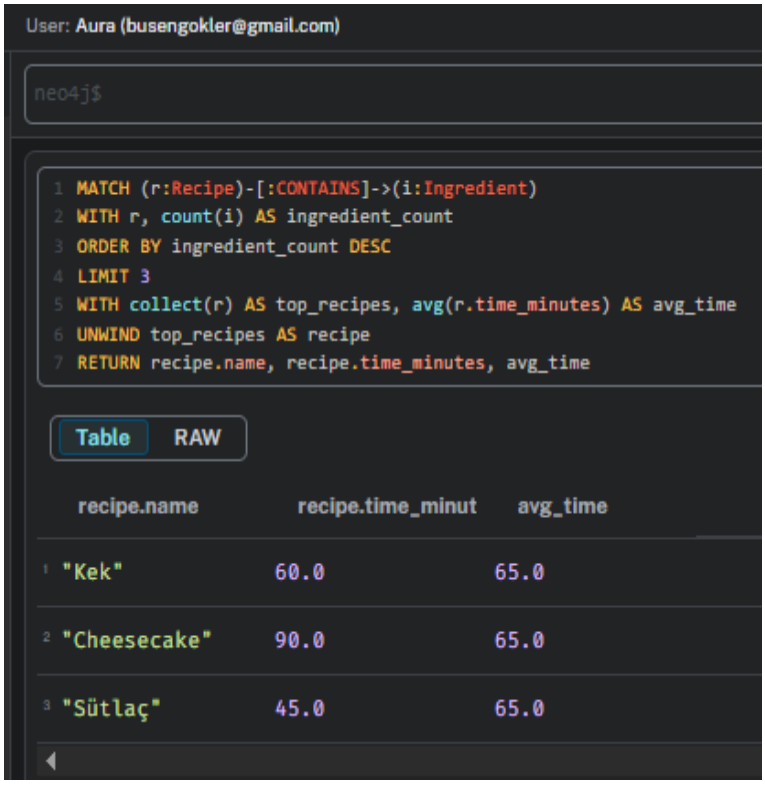
// Seçilen 3 tarifin ortalama süresini hesaplama ve tarifleri toplama

WITH collect(r) AS top_recipes, avg(r.time_minutes) AS avg_time

// Toplanan tariflerin tek tek sonuçlarının döndürülmesi

UNWIND top_recipes AS recipe

RETURN recipe.name, recipe.time_minutes, avg_time



Property nameleri dinamik olarak kullanabilir miyiz?

MATCH (r:Recipe) //Tarifleri bulur

// Her tarif için erişilecek property name listesini tanımladım.

WITH r, ['name', 'difficulty', 'time_minutes'] AS properties

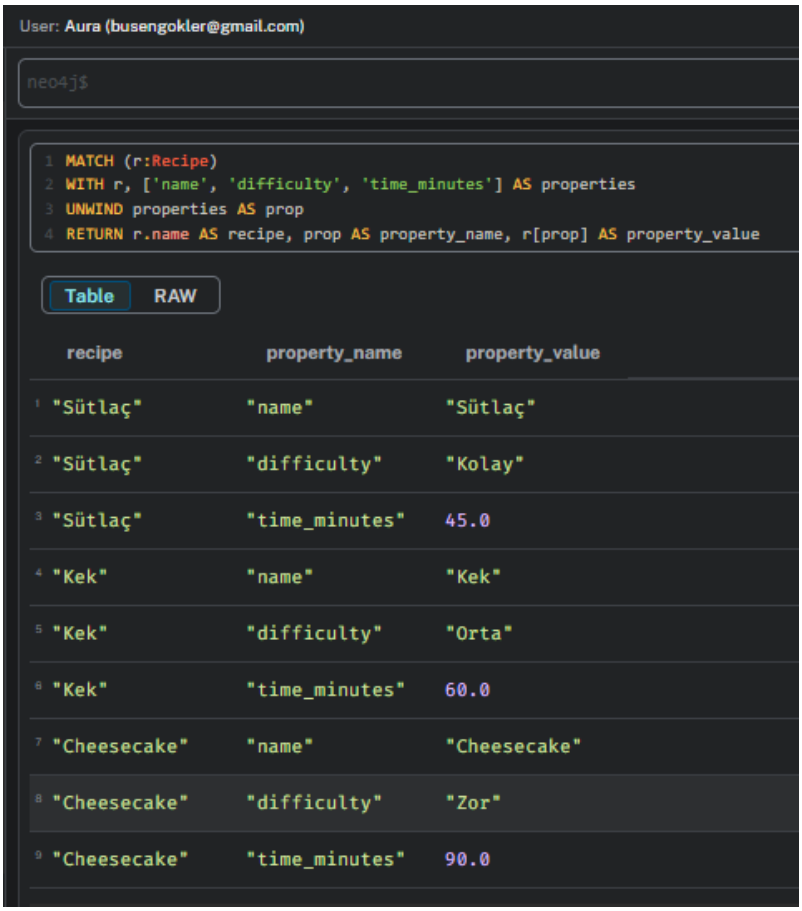
// Bu listedeki her bir elemanı ayrı bir satıra açtım

UNWIND properties AS prop

// Her tarif için dinamik olarak erişilen properties'i döndürdüm.

RETURN r.name AS recipe, prop AS property_name, r[prop] AS property_value

Burada bir tarifin property_name'leri ve onların value'lari çıktı olarak gelmektedir.



Her kategorideki malzemeleri gruplayabilir ve her grubun tarif sayısını hesaplayabilir miyiz?

MATCH (i:Ingredient)<-[:CONTAINS]-(r:Recipe)

// Kategoriye göre gruplama ve her kategori için unique tarifleri liste içinde toplama

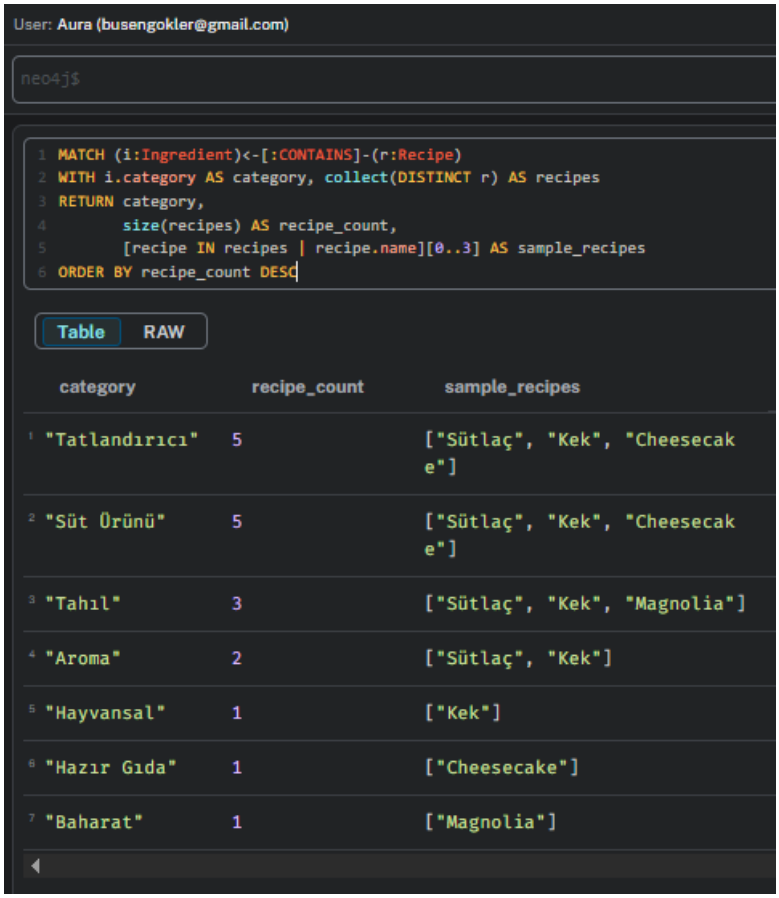
WITH i.category AS category, collect(DISTINCT r) AS recipes

RETURN category,

size(recipes) AS recipe_count,

[recipe IN recipes | recipe.name][0..3] AS sample_recipes

ORDER BY recipe_count DESC



En çok bağlantıya sahip malzemeleri bulabilir miyiz?

MATCH (i:Ingredient)

// Her malzeme kaç tarifte kullanıldı

OPTIONAL MATCH (i)<-[:CONTAINS]-(r:Recipe)

WITH i, count(r) AS recipe_connections

// Her malzemenin kaç diğer malzeme ile yer değiştirebileceği

OPTIONAL MATCH (i)-[:CAN_REPLACE]-(other:Ingredient)

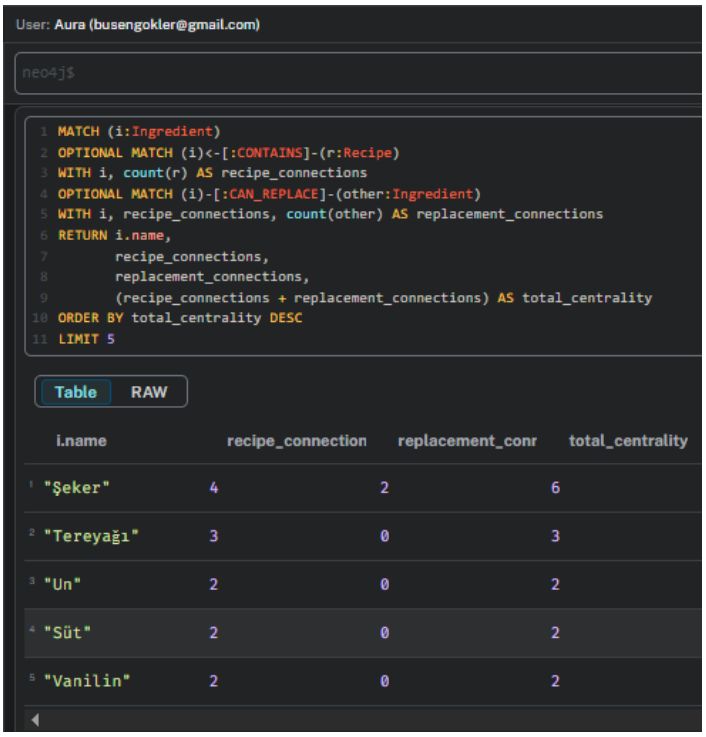
WITH i, recipe_connections, count(other) AS replacement_connections

RETURN i.name, recipe_connections, replacement_connections,

(recipe_connections + replacement_connections) AS total_centrality

ORDER BY total_centrality DESC

LIMIT 5



APOC: Neo4j için gelişmiş fonksiyonlar sağlar.

Koşullu veri ekleyebilmek için APOC gerekli.

MATCH (r:Recipe)

CALL apoc.do.when(

r.time_minutes > 60, // Koşul: Tarifin hazırlık süresi 60 dk'dan fazla mı?

"SET r.time_min = 'High' RETURN r", //doğruysa çalışacak sorgu

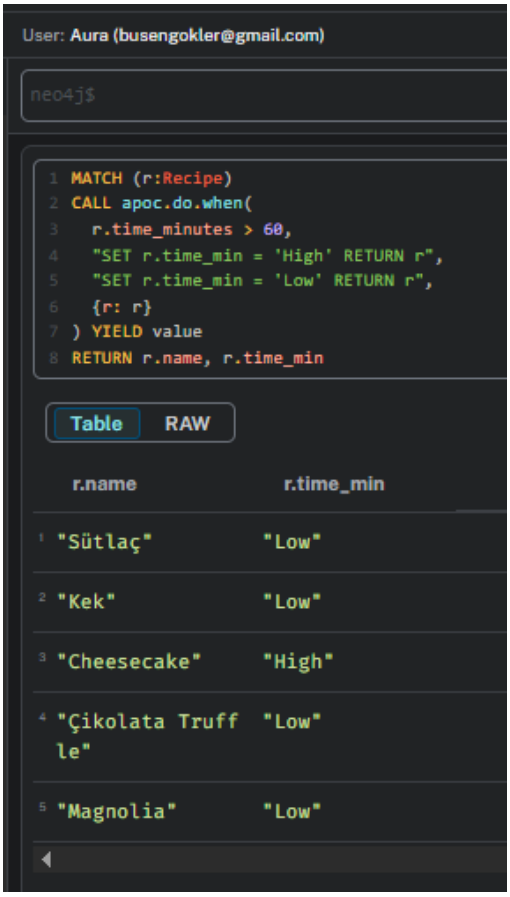
"SET r.time_min = 'Low' RETURN r", //yanlışsa çalışacak

{r: r} // params

) YIELD value // apoc.do.when'den dönen sonucu almak için

RETURN r.name, r.time_min

Tarifin adını ve yeni eklenen özelliği döndürür.



EXPLAIN ve PROFILE

Cypher sorgularının nasıl yürütüleceğini anlamak ve performans sorunlarını tespit etmek için kullanılırlar.

- **EXPLAIN:** Sorgunun *planını* gösterir. Verilerin nasıl alınacağını, hangi indekslerin kullanılacağını veya hangi işlemleri yapılacağını açıklar.
- **PROFILE:** Sorguyu *çalıştırır* ve her bir işlem için istatistikler (harcanan zaman, okunan satır sayısı vs.) gösterir.

EXPLAIN

MATCH (r:Recipe)-[:CONTAINS]->(i:Ingredient)

WHERE i.category = "Süt Ürünleri" AND r.time_minutes < 30

RETURN r.name, i.name

PROFILE

MATCH (r:Recipe)-[:CONTAINS]->(i:Ingredient)

WHERE i.category = "Süt Ürünleri" AND r.time_minutes < 30

RETURN r.name, i.name

Bu sorgu, malzemeleri (ingredients) tarayıp filtreleyerek 27 db hit (27 veritabanı erişimi) ve 28 ms ile hızlıca tamamlandı. İkinci filtrelemenin 0 db hits ile çalışması, bu adımların oldukça verimli olduğunu ve verilerin etkin bir şekilde kullanıldığını gösteriyor.