Progetto realizzato da Elena Curti (matr. 185431)

Ricette

Introduzione

In questo progetto viene realizzato un database con Neo4j. In esso verranno memorizzate delle ricette, i relativi ingredienti, autori, categorie e tipi di dieta. La struttura del database è stata realizzata partendo dallo usecase "Recipe" fornito da Neo4j.

Nel seguito, verranno inizialmente spiegate le operazioni iniziali per il corretto funzionamento dell'applicazione. Verranno poi mostrati lo schema e i dati dello use case originale. Verrà spiegato come è stato modificato lo schema e quali dati sono stati aggiunti e rimossi. Infine, verranno mostrate delle interrogazioni, modifiche e cancellazioni dei dati.

Requisiti

Questo progetto è stato realizzato con Python 3.11. E' necessaria l'installazione di Neo4j e dei seguenti pacchetti:

In [1]:
```
# %pip install py2neo neo4j-driver pandas
```

Operazioni iniziali

Per il corretto funzionamento del progetto proposto, è necessario:

- Creare un nuovo progetto Neo4j
- Creare all'interno del progetto un DMBS (scegliendo come nome "neo4j", come password "pass" e come versione 5.2.0)
- Aggiungere il plugin APOC.
- Premere su Start.

Eseguire poi le celle di codice, nell'ordine proposto.

Connessione e autenticazione

Per effettuare la connessione al database, eseguire il seguente script:

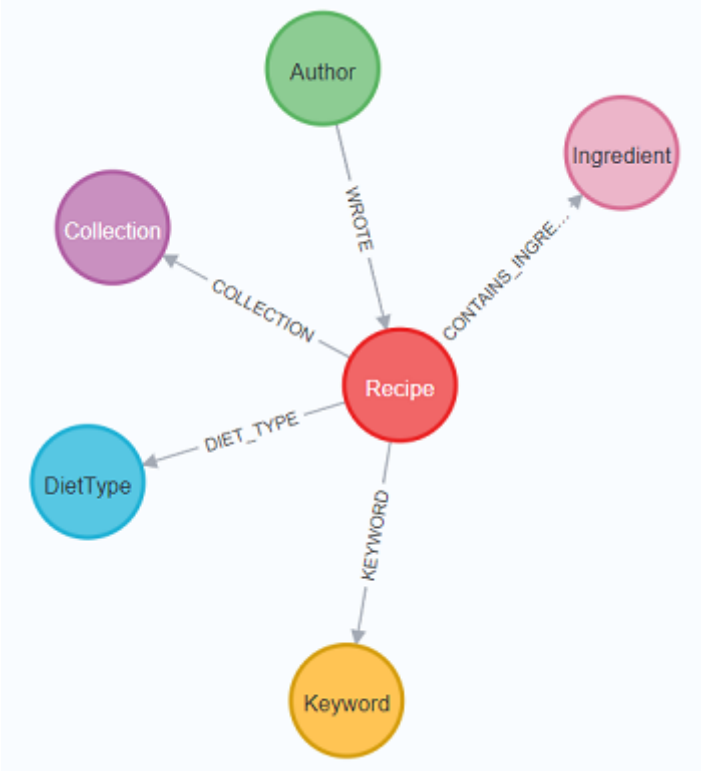
In [2]:
```
from py2neo import Graph
graph = Graph("bolt://localhost:7687",  auth=("neo4j", "pass"))

print("Connessione al database eseguita correttamente!")
```

```
Connessione al database eseguita correttamente!
```

Schema e dati originali

Lo use case originale è stato trovato sul sito Recipes Neo4j. Ha la seguente struttura:

```
call db.schema.visualization()
```



I nodi Ingredient, Author, Collection, DietType, Keyword hanno un solo campo "name" che ne rappresenta il nome.
Il nodo Recipe ha le seguenti proprietà:

- id: campo che identifica univocamente la ricetta
- cookingTime: tempo di cottura
- description: descrizione della ricetta
- name: nome della ricetta
- preparationTime: tempo di preparazione
- skillLevel: difficoltà

Gli archi non hanno proprietà.

Per importare i dati e aggiungere gli indici ho usato il seguente script.
Lo usecase originale ha previsto i seguenti indici per migliorare la performance delle cypher query:

- *id* su Recipe
- *name* su Ingredient
- *name* su Keyword
- *name* su DietType
- *name* su Author
- *name* su Collection

Descrizione dei comandi di import

Il comando `CALL apoc.load.json('path_file') YIELD value` serve per leggere il file in formato JSON in input. Se il file dato:

- è memorizzato in locale, allora va inserito nella cartella "import" del DBMS. Va inoltre inserito nella cartella "conf" un file "apoc.conf" con scritto "apoc.import.file.enabled=true" (e riavviato il database). Infine, il percorso deve essere in formato URI.
- è memorizzato in rete (come in questo caso), il comando funziona in automatico, se il computer è connesso ad Internet.

Il comando `MERGE` serve eventualmente a creare un nodo (o un arco) e a selezionarlo. E' una sorta di combinazione di `MATCH` e `CREATE`.

Il comando `FOREACH (elem IN list | codice )` serve a eseguire il codice per ogni elemento `elem` nella lista `list`.

Il comando `CREATE INDEX IF NOT EXISTS FOR (n:Recipe) ON (n.id)` è equivalente a `CREATE INDEX IF NOT EXISTS ON :Recipe(id)`. Il primo, però, è compatibile con la versione 5.2.0 di Neo4j.

In [3]:
```
print("Inserimento dei dati in corso... ")

# Ricette
graph.run("""
CALL apoc.load.json('https://raw.githubusercontent.com/neo4j-
examples/graphgists/master/browser-guides/data/stream_clean.json') YIELD value
WITH value.page.article.id AS id,
       value.page.title AS title,
       value.page.article.description AS description,
       value.page.recipe.cooking_time AS cookingTime,
       value.page.recipe.prep_time AS preparationTime,
       value.page.recipe.skill_level AS skillLevel
MERGE (r:Recipe {id: id})
SET r.cookingTime = cookingTime,
    r.preparationTime = preparationTime,
    r.name = title,
    r.description = description,
    r.skillLevel = skillLevel;
""")
graph.run(""" CREATE INDEX IF NOT EXISTS FOR (n:Recipe) ON (n.id);        //
CREATE INDEX IF NOT EXISTS ON :Recipe(id)  """)

# Autori
graph.run("""
```

```
//import authors and connect to recipes
CALL apoc.load.json('https://raw.githubusercontent.com/neo4j-
examples/graphgists/master/browser-guides/data/stream_clean.json') YIELD value
WITH value.page.article.id AS id,
       value.page.article.author AS author
MERGE (a:Author {name: author})
WITH a,id
MATCH (r:Recipe {id:id})
MERGE (a)-[:WROTE]->(r);
""")
graph.run("""CREATE INDEX IF NOT EXISTS FOR (n:Author) ON (n.name);""" )

# Ingredienti
graph.run("""//import ingredients and connect to recipes
CALL apoc.load.json('https://raw.githubusercontent.com/neo4j-
examples/graphgists/master/browser-guides/data/stream_clean.json') YIELD value
WITH value.page.article.id AS id,
       value.page.recipe.ingredients AS ingredients
MATCH (r:Recipe {id:id})
FOREACH (ingredient IN ingredients |
  MERGE (i:Ingredient {name: ingredient})
  MERGE (r)-[:CONTAINS_INGREDIENT]->(i)
);""" )
graph.run("""CREATE INDEX IF NOT EXISTS FOR (n:Ingredient) ON (n.name);""" )

# DietType
graph.run("""//import dietTypes and connect to recipes
CALL apoc.load.json('https://raw.githubusercontent.com/neo4j-
examples/graphgists/master/browser-guides/data/stream_clean.json') YIELD value
WITH value.page.article.id AS id,
       value.page.recipe.diet_types AS dietTypes
MATCH (r:Recipe {id:id})
FOREACH (dietType IN dietTypes |
  MERGE (d:DietType {name: dietType})
  MERGE (r)-[:DIET_TYPE]->(d)
);""" )
graph.run("""CREATE INDEX IF NOT EXISTS FOR (n:DietType) ON (n.name);""" )

# Collections
graph.run("""//import collections and connect to recipes
CALL apoc.load.json('https://raw.githubusercontent.com/neo4j-
examples/graphgists/master/browser-guides/data/stream_clean.json') YIELD value
```

```
WITH value.page.article.id AS id,
       value.page.recipe.collections AS collections
MATCH (r:Recipe {id:id})
FOREACH (collection IN collections |
  MERGE (c:Collection {name: collection})
  MERGE (r)-[:COLLECTION]->(c)
);""" )
graph.run("""CREATE INDEX IF NOT EXISTS FOR (n:Collection) ON (n.name);""" )

# Keyword
cq = """//import keywords and connect to recipes
CALL apoc.load.json('https://raw.githubusercontent.com/neo4j-
examples/graphgists/master/browser-guides/data/stream_clean.json') YIELD value
WITH value.page.article.id AS id,
       value.page.recipe.keywords AS keywords
MATCH (r:Recipe {id:id})
FOREACH (keyword IN keywords |
  MERGE (k:Keyword {name: keyword})
  MERGE (r)-[:KEYWORD]->(k)
);"""
# graph.run(cq)
cq = """CREATE INDEX IF NOT EXISTS FOR (n:Keyword) ON (n.name);"""
# graph.run(cq)


print("Tutti i dati sono stati importati correttamente!")
```

```
Inserimento dei dati in corso...
Tutti i dati sono stati importati correttamente!
```

Modifiche effettuate

Per arricchire il progetto ho deciso di aggiungere dei dati presi da Kaggle e di eseguire alcune modifiche allo schema originale.

In [4]:

```
# Definisco un'eccezione usata nel seguito
class MiaStopExecution(Exception):
    """Il raise di questa classe provoca l'interruzione dell'esecizione della
cella, senza interrompere il kernel"""
    def _render_traceback_(self):
        pass

# Prima di effettuare le modifiche, copio i file necessari nella cartella
```

```
"import" del DBMS, mediante il seguente script:
import os
import shutil
DATA_FOLDER ="data"+os.path.sep

cq ="""CALL dbms.listConfig() YIELD name, value
    WHERE name="server.directories.import"
    RETURN value"""
DIRECTORY_IMPORT = graph.run(cq).evaluate()
print("Cartella import del DMBS: " + DIRECTORY_IMPORT)

def copy_in_import_dir(file_name):
    source = DATA_FOLDER + file_name
    destination = DIRECTORY_IMPORT + os.path.sep + file_name
    shutil.copy(source, destination)

copy_in_import_dir("FoodData.csv")
copy_in_import_dir("gz_recipe.csv")
```

```
Cartella import del DMBS: D:\Elena\_Elena\Shared\Universita\Magistrale\Big_data\Neo4J\relate-data\dbmss\dbms-108c38bb-8fe4-4c95-b4e6-d1b3bbdad61e\import
```

1. Rimozione di Keyword

Ho deciso di rimuovere il nodo Keyword. Ho quindi evitato di eseguire le ultime due cypher query nello script precedente. Alternativamente, avrei potuto anche usare il seguente comando:

In [5]:
```
graph.run("MATCH (n:Keyword) DETACH DELETE n")
```

Out[5]: (No data)

2. Aggiunta di altre ricette

Per arricchire il dataset, ho deciso di inserire altre ricette al database. Ho quindi scaricato il file gz_recipe.csv da https://www.kaggle.com/datasets/edoardoscarpaci/italian-food-recipes, contenente delle ricette estratte dal sito GialloZafferano.
Il file scaricato contiene i seguenti campi:

- id: id che identifica univocamente la ricetta (intero da 0 a 5938).
- Nome: nome del piatto
- Categoria: categoria della ricetta (Primi, Secondi, ...)
- Link: link della ricetta
- Persone/Pezzi: numero di persone o pezzi della ricetta
- Ingredienti: lista di ingredienti e relative quantità. Ad esempio: [['Mascarpone', '750g'], ['Uova', '260g']]
- Steps: contenuto della ricetta

Per integrare questi dati ho deciso di:

- Aggiungere a Recipe le proprietà
  - "fonte" che avrà il valore "GialloZafferano" o "BBC GoodFood"
  - "persone_pezzi" per indicare il numero di persone o pezzi
- Aggiungere a CONTAINS_INGREDIENTS una proprietà "quantita", contenente la quantità di un ingrediente in una ricetta
- Aggiungere a Recipe un controllo sull'univocità degli id. Nonostante gli id delle Recipe siano univoci, non è presente nel database nessun controllo che impedisca all'utente di inserire più ricette con lo stesso id. Dovendo andare ad unire un dataset esterno, potrebbero esserci dei conflitti sull'id. Ho quindi deciso di aggiungere tale controllo aggiungendo un constraint.

In [6]:
```
# Aggiungo le ricette di GialloZafferano.

# Tolgo il "vecchio" indice. Inserisco un CONSTRAINT per verificare l'univocità
dell'id.
# Il nuovo indice su id è inserito insieme al CONSTRAINT
if len(graph.run("SHOW CONSTRAINT WHERE name='id_ricetta_univoco'").data())==0:
# Questo if serve per non avere errori se questa cella dovesse essere eseguita
più volte
    cq = "SHOW INDEXES WHERE labelsOrTypes=['Recipe'] AND properties=['id']"
    nome_indice_id_ricette = graph.run(cq).data()[0]["name"]
    # cq = "DROP CONSTRAINT id_ricetta_univoco IF EXISTS"
    # graph.run(cq)
    cq = "DROP INDEX "+nome_indice_id_ricette+" IF EXISTS"
    graph.run(cq)
    cq = "CREATE CONSTRAINT id_ricetta_univoco IF NOT EXISTS FOR (r:Recipe)
REQUIRE r.id IS UNIQUE"
    graph.run(cq)

# Controllo che non ci siano già ricette con gli stessi id
cq="""
LOAD CSV WITH HEADERS FROM 'file:///gz_recipe.csv' AS value
MATCH (r:Recipe {id:value["id"]})
WHERE r.fonte IS NULL OR r.fonte<>"GialloZafferano"
return count(r.id)
"""
num_id_comuni = graph.run(cq).evaluate()
if  num_id_comuni!= 0:
    print("Le ricette di BBC e quelle di GialloZafferano hanno
"+str(num_id_comuni)+" id in comune!")
    raise(MiaStopExecution)
```

```
# Inserisco le ricette
cq="""
LOAD CSV WITH HEADERS FROM 'file:///gz_recipe.csv' AS value
MERGE (r:Recipe {id: value["id"]})
SET r.name = value["Nome"],
    r.description = value["Steps"],
    r.persone_pezzi = value["Persone/Pezzi"],
    r.fonte = "GialloZafferano"
"""
graph.run(cq)

cq="""MATCH (r:Recipe) WHERE r.fonte IS NULL SET r.fonte="BBC GoodFood" """
graph.run(cq)

# Inserisco gli ingredienti
cq="""
LOAD CSV WITH HEADERS FROM 'file:///gz_recipe.csv' AS value
WITH value["id"] AS id, replace(value["Ingredienti"], '"', "") AS ris1
WITH id, replace(ris1, "[[", "") AS ris2
WITH id,replace(ris2, "]]", "") AS ris3
WITH id,replace(ris3, "'", "") AS ris4
WITH id,split(ris4, "], [") AS ingrs_list
MATCH (r:Recipe {id:id})
WHERE ingrs_list[0]<>"[]"   // Alcune ricette non hanno ingredienti nè
descrizione (es. Churros red velvet)
FOREACH (ingr_quantita_string IN ingrs_list |
    MERGE (i:Ingredient {name: split(ingr_quantita_string, ", ")[0]})
    MERGE (r)-[:CONTAINS_INGREDIENT {quantita: split(ingr_quantita_string, ",
")[1]}]->(i)
)
"""
graph.run(cq)


# Creo le categorie
cq="""
LOAD CSV WITH HEADERS FROM 'file:///gz_recipe.csv' AS value
MATCH (r:Recipe {id:value["id"]})
WHERE value["Categoria"] IS NOT NULL
MERGE (c:Collection {name: value["Categoria"]})
MERGE (r)-[:COLLECTION]->(c)
```

```
"""
graph.run(cq)
```

Out[6]: (No data)

3. Inserimento degli allergeni

Ho aggiunto un campo "is_allergene" al nodo Ingredient. Ho scaricato da Kaggle due file:

- allergens.json (https://www.kaggle.com/datasets/elenacivati/allergensjson): contenente un elenco non molto ampio di allergeni in italiano, inglese e altre lingue.
- FoodData.csv (https://www.kaggle.com/datasets/boltcutters/food-allergens-and-allergies): contenente un elenco più ampio ma solo in inglese

Ho quindi impostato gli ingredienti contenuti nei file con is_allergene=True. Ho invece settato is_allergene=False per i restanti ingredienti.

In [7]:
```
# Prendo dal file allergens.json i nomi degli ingredienti in inglese e italiano.
# Converto poi in csv, in modo da semplificare la lettura del file successivamente
import pandas as pd
df_allergens = pd.read_json(DATA_FOLDER + "allergens.json").transpose()
df_allergens["en"]=df_allergens.index
df_allergens["en"]=df_allergens["en"].apply(lambda x: x[3:])
df_allergens["it"]=df_allergens["name"].apply(lambda x: x["it"])
df_allergens=df_allergens[["en", "it"]]
df_allergens.to_csv(DATA_FOLDER + "allergens.csv", index=False)
copy_in_import_dir("allergens.csv")

# Setto gli allergeni
cq="""
LOAD CSV WITH HEADERS FROM 'file:///FoodData.csv' AS value
LOAD CSV WITH HEADERS FROM 'file:///allergens.csv' AS value2
MATCH (i:Ingredient)
WHERE
    toLower(i.name) CONTAINS toLower(value["Food"])  OR
    toLower(i.name) CONTAINS toLower(value2["en"])  OR
    toLower(i.name) CONTAINS toLower(value2["it"])
SET i.is_allergene=True;"""
graph.run(cq)

# Setto i restanti ingredienti come non allergeni
cq="""
```

```
MATCH (i:Ingredient)
WHERE i.is_allergene IS NULL
SET i.is_allergene=False; """
graph.run(cq)
```

Out[7]: (No data)

4. Nuovo indice

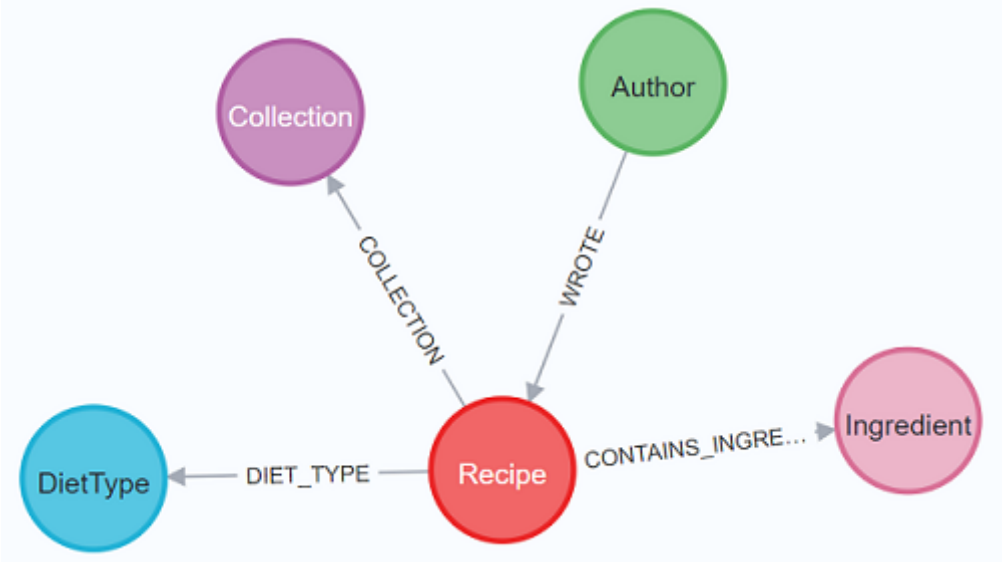
Per velocizzare in futuro la ricerca delle ricette in base al nome, ho deciso di inserire un nuovo indice sulla proprietà name di Recipe.

In [8]:
```
graph.run(" CREATE INDEX IF NOT EXISTS FOR (n:Recipe) ON (n.name);")
```

Out[8]: (No data)

Schema finale

Dopo le modifiche, lo schema finale è il seguente:



In [9]:
```
def print_first_n(cq, n=1000):
    ris = graph.run(cq)
    ris.sample_size = n
    return ris

print_first_n("CALL db.schema.nodeTypeProperties")
```

Out[9]:

nodeType	nodeLabels	propertyName	propertyTypes	mandatory
:`Author`	['Author']	name	['String']	true
:`Ingredient`	['Ingredient']	name	['String']	true
:`Ingredient`	['Ingredient']	is_allergene	['Boolean']	true
:`DietType`	['DietType']	name	['String']	true
:`Collection`	['Collection']	name	['String']	true
:`Recipe`	['Recipe']	id	['String']	true
:`Recipe`	['Recipe']	cookingTime	['Long']	false
:`Recipe`	['Recipe']	preparationTime	['Long']	false
:`Recipe`	['Recipe']	name	['String']	true
:`Recipe`	['Recipe']	description	['String']	false
:`Recipe`	['Recipe']	skillLevel	['String']	false
:`Recipe`	['Recipe']	persone_pezzi	['String']	false
:`Recipe`	['Recipe']	fonte	['String']	true

In [10]:
```
print_first_n("CALL db.schema.relTypeProperties")
```

Out[10]:

relType	propertyName	propertyTypes	mandatory
:`WROTE`	null	null	false
:`CONTAINS_INGREDIENT`	quantita	['String']	false
:`DIET_TYPE`	null	null	false
:`COLLECTION`	null	null	false

In [11]:
```
print_first_n("SHOW INDEX")
```

Out[11]:

id	name	state	populationPercent	type	entityType	labelsOrTypes	properties	indexP
8	id_ricetta_univoco	ONLINE	100.0	RANGE	NODE	['Recipe']	['id']	range-1
7	index_217e8412	ONLINE	100.0	RANGE	NODE	['Collection']	['name']	range-1
1	index_343aff4e	ONLINE	100.0	LOOKUP	NODE	null	null	token-l 1.0
6	index_6ba4d9f7	ONLINE	100.0	RANGE	NODE	['DietType']	['name']	range-1
3	index_ac0200dd	ONLINE	100.0	RANGE	NODE	['Recipe']	['name']	range-1
4	index_bf1e25cf	ONLINE	100.0	RANGE	NODE	['Author']	['name']	range-1
5	index_f492b287	ONLINE	100.0	RANGE	NODE	['Ingredient']	['name']	range-1
2	index_f7700477	ONLINE	100.0	LOOKUP	RELATIONSHIP	null	null	token-l 1.0

In [12]:
```
print_first_n("SHOW CONSTRAINT")
```

Out[12]:

id	name	type	entityType	labelsOrTypes	properties	ownedIndex
9	id_ricetta_univoco	UNIQUENESS	NODE	['Recipe']	['id']	id_ricetta_univoco

Interrogazioni

Quali ricette contengono due ingredienti (scelti dall'utente)?

In [13]:

```
# Cerco le ricette con due ingredienti
print("************ Stampa una ricetta con due ingredienti *********** ")

# Stampo qualche suggerimento
print("Qualche suggerimento per la scelta degli ingredienti: ")
cq = """
MATCH (i:Ingredient)
WITH i, rand() as num
RETURN i.name AS ingrediente, i.is_allergene AS is_allergene
ORDER BY num
LIMIT 5
"""
display(print_first_n(cq))

# Chiedo il primo ingrediente e stampo gli ingredienti più usati con esso
ingrediente1=""
while ingrediente1=="":
    ingrediente1=input("Inserisci primo ingrediente: ")
# ingrediente1="Confettura di prugne"

print("Ecco 5 ingredienti usati spesso con " +ingrediente1)
cq = """
MATCH (i_scelto:Ingredient)<-[:CONTAINS_INGREDIENT]-(:Recipe)-
[:CONTAINS_INGREDIENT]->(i2:Ingredient)
WHERE toLower(i_scelto.name) CONTAINS toLower('"""+ingrediente1+"""')
WITH i2,  COUNT(i2) AS num
RETURN i2.name AS ingrediente_raccomandato, i2.is_allergene AS is_allergene
ORDER BY num DESC
LIMIT 5
"""
display(print_first_n(cq))


# Chiedo il secondo ingrediente
```

```
ingrediente2=""
while ingrediente2=="":
    ingrediente2=input("Inserisci secondo ingrediente: ")
# ingrediente2="Uova"

# Cerco i risultati...
cq = """
MATCH (i1:Ingredient)<-[:CONTAINS_INGREDIENT]-(r:Recipe)-
[:CONTAINS_INGREDIENT]->(i2:Ingredient)
WHERE
    toLower(i1.name) CONTAINS toLower('"""+ingrediente1+"""') AND
    toLower(i2.name) CONTAINS toLower('"""+ingrediente2+"""')
RETURN r
"""
recipe_list = graph.run(cq)

# ... e li stampo
print("\nRicette trovate con gli ingredienti",ingrediente1, ", "
,ingrediente2,"\b: ")
ricetta_trovata = False
for r in recipe_list:
    ricetta_trovata = True
    node_recipe = r[r.keys()[0]]
    print("\t", node_recipe.get("name", "\b"))
if not ricetta_trovata:
    print("------ Nessuna ricetta! ------")
```

```
************ Stampa una ricetta con due ingredienti ************
Qualche suggerimento per la scelta degli ingredienti:
```

ingrediente	is_allergene
raw cacao powder	false
puffed wheat	true
Aroma allarancia	false
pain de campagne	false
rice vinegar	true

```
Ecco 5 ingredienti usati spesso con Confettura di prugne
```

ingrediente_raccomandato	is_allergene
Uova	true
Farina 00	false
Zucchero	false
Sale fino	false
Latte intero	true

```
Ricette trovate con gli ingredienti Confettura di prugne ,  uova:
         Crostata alla crema e prugne
         Treccine di sfoglia
         Girelle alla marmellata
```

Stampa i dettagli di una ricetta

In [14]:

```
# Chiedo il nome della ricetta
nome_ricetta=""
while nome_ricetta=="":
    nome_ricetta=input("Inserisci nome della ricetta: ")
print("Ricetta scelta:", nome_ricetta)
#nome_ricetta="Limoncello"

# Cerco la ricetta e tutti i relativi dati
cq = """
MATCH (r:Recipe)
WHERE toLower(r.name) = toLower('"""+nome_ricetta+"""')
OPTIONAL MATCH (a:Author)-[:WROTE]->(r)
OPTIONAL MATCH (c:Collection)<-[:COLLECTION]-(r)
OPTIONAL MATCH (d:DietType)<-[:DIET_TYPE]-(r)
OPTIONAL MATCH (i:Ingredient)<-[r_i:CONTAINS_INGREDIENT]-(r)
RETURN r AS ricetta, COLLECT(DISTINCT c) AS categorie, COLLECT(DISTINCT i) AS
ingredienti, COLLECT(DISTINCT d) AS tipi_dieta, COLLECT(DISTINCT r_i) AS
quantita_ingredienti
"""
list_recipes = graph.run(cq)

# Stampo i risultati
ricetta_trovata=False
for res in list_recipes:
    ricetta_trovata = True

    tmp = dict(res)
    print("---------------")
```

```
    if tmp.__contains__("ricetta"):
        for k,v in tmp["ricetta"].items():
            print(k, "\b:",v)

    if tmp.__contains__("categorie"):
        categorie = tmp["categorie"]
        print("Categorie:", [v for c in categorie for _,v in c.items()])

    if tmp.__contains__("ingredienti"):
        ingredienti =  [c["name"] + (" (Allergene!)" if c["is_allergene"] else
"") for c in tmp["ingredienti"]]

    if tmp.__contains__("quantita_ingredienti"):
        quantita_ingredienti = [dict(q_i).get("quantita", "") for q_i in
tmp["quantita_ingredienti"]]
        tmp2 = []
        for i in range(len(ingredienti)):
            tmp2.append( ingredienti[i] + " " + quantita_ingredienti[i])
        ingredienti = tmp2

    print("Ingredienti:", ingredienti)

if not ricetta_trovata:
    print("--- Nessuna ricetta trovata --- ")
```

```
Ricetta scelta: Limoncello
---------------
fonte: BBC GoodFood
name: Limoncello
preparationTime: 900
description: Make up a batch of this boozy lemony drink - it's great as a homemade gift or
poured over ice cream
id: 102024
skillLevel: Easy
cookingTime: 0
Categorie: ['Our top 25 drink', 'Lemon', "New Year's Eve cocktails"]
Ingredienti: ['caster sugar (Allergene!) ', 'water ', 'lemon (Allergene!) ', 'vodka ']
---------------
persone_pezzi: 10
fonte: GialloZafferano
name: Limoncello
description: Per realizzare il limoncello lavate i limoni sotto il getto dell'acqua corren
te sfregate la buccia con una spugnetta nuova per eliminare eventuali impurità, quindi asc
iugate i limoni con un canovaccio Sbucciate i limoni con un pelapatate, dovrete prelevare
solo la scorza gialla e non la parte bianca che risulterebbe amara Prendete un recipiente
di vetro con chiusura ermetica, versate all'interno l'alcol richiudete il barattolo Trasco
rsi i 30 giorni recuperate le scorze e preparate lo sciroppo: in un tegame versate l'acqua
portate al bollore lo sciroppo e, una volta raggiunto il bollore spegnete il fuoco, versat
e lo sciroppo in una brocca Agitate il barattolo per mescolare lo sciroppo, quindi lasciat
elo ancora riposare Trascorso il tempo di riposo riprendete il barattolo con il liquore, a
gitatelo e poi filtratelo attraverso un colino (13-14) e raccogliete il liquore all'intern
o di una bottiglia. Il vostro limoncello fatto in casa è pronto per essere degustato
id: 385
Categorie: ['Bevande']
Ingredienti: ['Limoni 5', 'Zucchero 600g', 'Alcol puro 500ml', 'Acqua 750ml']
```

Quali sono le ricette più veloci da preparare?

In [15]:

```
print("Scegli un'opzione. Ricetta piu' veloce in base a: \n\t1. Tempo di
preparazione\n\t2. Tempo di cottura\n\t3. Tempo totale")
# opzione = "1"
opzione=""
while opzione=="":
    opzione=input(">> ")
    opzione = "tempo_preparazione" if opzione=="1" else ("tempo_cottura" if
opzione=="2" else ("tempo_totale" if opzione=="3" else ""))
print("Opzione scelta: " + opzione)

cq = """MATCH (r:Recipe)
WHERE r.preparationTime<>0
RETURN r.name AS ricetta, r.cookingTime AS tempo_cottura, r.preparationTime as
tempo_preparazione, r.cookingTime+r.preparationTime AS tempo_totale
ORDER BY """ + opzione + " LIMIT 3"
graph.run(cq)
```

```
Scegli un'opzione. Ricetta piu' veloce in base a:
       1. Tempo di preparazione
       2. Tempo di cottura
       3. Tempo totale
Opzione scelta: tempo_preparazione
```

Out[15]:

ricetta	tempo_cottura	tempo_preparazione	tempo_totale
Lighter Cornish pasties	3000	60	3060
Christmas spice latte	180	60	240
Creamy yogurt porridge	180	60	240

Chi ha scritto più ricette?

In [16]:

```
cq = """MATCH (a:Author)-[:WROTE]->(r:Recipe)
WITH a, COUNT(r) AS num_ricette, COLLECT(r.name)[..5] AS alcune_ricette //
Seleziono solo le prime cinque ricette
RETURN a.name, num_ricette, alcune_ricette
ORDER BY num_ricette DESC
LIMIT 5
"""
display(print_first_n(cq))
```

a.name	num_ricette	alcune_ricette
Good Food	3441	['Tandoori tilapia with spicy sweet potato mash & tomato salad', 'Best ever tiramisu', 'Duck, apricot & pine nut pastilla', 'Raspberry tiramisu ', 'Classic smoked salmon crostini']
Sara Buenfeld	664	['Thai curry fish cakes with sweet chilli dressing', 'Cranberry & clementine jelly', 'Jam & white chocolate roly-poly', 'South Indian coconut & prawn curry', 'Turkey salad on rye']
Barney Desmazery	646	['Crispy bacon-basted turkey', 'Burnt butter cabbage', 'Tea-smoked salmon with herb mayonnaise', 'Poached plums', 'Creamy cucumber with gravadlax']
Cassie Best	638	['English rose cake', 'Gingerbread B\u00c3\u00bbche de No\u00c3\u00abl', 'Crunchy prawn & noodle salad', 'Spanish meatball & butter bean stew', 'Golden couscous with apricots & crispy onions']
Sarah Cook	515	['Spring chicken paella', 'Iced trifle slice', 'Potato curry with lime & cucumber raita', 'Salted caramel & macadamia pralines', 'Shredded greens salad']

Stampa le ricette suggerite

Partendo da una ricetta ('Chocolate chia pudding'), il seguente script stampa le ricette suggerite, in base alle categorie, ai tipi di dieta e all'autore/autrice comuni.

In [17]:

```
ricetta = "Chocolate chia pudding"
cq = """
MATCH (r:Recipe {name:'"""+ ricetta +"""'})-[:COLLECTION]->(c:Collection)<-
[:COLLECTION]-(r2:Recipe)
OPTIONAL MATCH (r)-[:DIET_TYPE]->(d:DietType)<-[:DIET_TYPE]-(r2)
```

```
OPTIONAL MATCH (r)<-[:WROTE]-(a:Author)-[:WROTE]->(r2)
WITH r2, COUNT(*) AS n_dati_comuni, COLLECT(DISTINCT d.name) AS
diete_comuni,COLLECT(DISTINCT c.name) AS categorie_comuni, a.name AS
autore_comune
RETURN r2.name AS ricetta_suggerita, categorie_comuni, diete_comuni,
n_dati_comuni, autore_comune
ORDER BY n_dati_comuni DESC, autore_comune IS NULL
LIMIT 5
"""
display(print_first_n(cq))
```

ricetta_suggerita	categorie_comuni	diete_comuni	n_dati_comuni	autore_comune
Quinoa stew with squash, prunes & pomegranate	['Easy vegan', 'Healthy vegan', 'Vegan gluten-free']	['Vegan', 'Dairy-free', 'Vegetarian', 'Gluten-free', 'Healthy']	15	null
Guacamole & mango salad with black beans	['Vegan summer', 'Healthy vegan', 'Vegan gluten-free']	['Vegan', 'Low-calorie', 'Gluten-free', 'Vegetarian', 'Healthy']	15	null
Chargrilled vegetable tacos with smoky salsa	['Easy vegan', 'Vegan summer', 'Healthy vegan']	['Vegan', 'Low-calorie', 'Healthy', 'Vegetarian']	12	null
Vegan chickpea curry jacket potatoes	['Easy vegan', 'Healthy vegan', 'Vegan gluten-free']	['Vegetarian', 'Vegan', 'Gluten-free', 'Healthy']	12	null
Vegan Eton mess	['Vegan summer', 'Vegan gluten-free']	['Gluten-free', 'Vegan', 'Vegetarian', 'Dairy-free', 'Egg-free']	10	Sophie Godwin

Quali ingredienti sono usati più spesso con la pasta?

In [18]:

```
cq ="""
MATCH (pasta:Ingredient)<-[:CONTAINS_INGREDIENT]-()-[:CONTAINS_INGREDIENT]->
(i:Ingredient)
WHERE toLower(pasta.name)="pasta"
WITH i, count(*) as frequenza
RETURN i.name AS ingrediente, frequenza
ORDER BY frequenza DESC
LIMIT 10
"""
display(print_first_n(cq))
```

ingrediente	frequenza
olive oil	65
parmesan	44
garlic clove	41
onion	25
lemon	23
pesto	17
parsley	14
chopped tomato	12
basil	12
tomato	11

Ricette presenti sia su GialloZafferano, sia su BBC GoodFood

In [19]:

```
def print_columns(lista, n_colonne):
    # max_len = max([len(i) for i in lista])
    if n_colonne == 2:
        for a,b in zip(lista[::2],lista[1::2]):
            print('   {:<60}{:<}'.format(a,b))
    if n_colonne == 3:
        for a,b,c in zip(lista[::3],lista[1::3],lista[2::3]):
            print('   {:<30}{:<30}{:<}'.format(a,b,c))



cq = """
MATCH (r_gz:Recipe {fonte:"GialloZafferano"}), (r_bbc:Recipe {fonte:"BBC
GoodFood"})
WITH apoc.coll.intersection(COLLECT(DISTINCT r_gz.name), COLLECT(DISTINCT
r_bbc.name)) as ricette_comuni //WHERE r_gz.name=r_bbc.name
RETURN size(ricette_comuni) AS num_ricette_comuni, ricette_comuni
"""
ris = graph.run(cq)

# Stampo il numero di risultati
diz_ris = ris.data()[0]
print("Numero di ricette comuni trovate:", diz_ris["num_ricette_comuni"])

# Stampo le ricette su tre colonne
print("Nomi delle ricette:")
lista_ricette = diz_ris["ricette_comuni"]
```

```
print_columns(lista_ricette, 3)
```

```
Numero di ricette comuni trovate: 47
Nomi delle ricette:
   Banoffee pie                 Pulled pork                  Alfajores
   Limoncello                   Pakora                       Harissa
   Hot cross buns               Colcannon                    Ceviche
   Caponata                     Pierogi                      Tarte tatin
   Acquacotta                   Coq au vin                   Fish tacos
   Ratatouille                  Cake pops                    Patatas bravas
   Key lime pie                 Quiche Lorraine              Croque madame
   Monkey bread                 Tzatziki                     Irish coffee
   Irish stew                   Raclette                     Lebkuchen
   Panettone                    Cosmopolitan                 Raita
   Tequila sunrise              Ciabatta                     Cranachan
   Baci di dama                 Bouillabaisse                Cinnamon rolls
   Macarons                     Panforte                     French toast
   Cornbread                    Lemon curd                   Lemon bars
   Spaghetti alle vongole       Crumpets                     Salmorejo
```

Quali ricette non contengono allergeni?

In [24]:

```
cq = """
MATCH (r:Recipe)-[:CONTAINS_INGREDIENT]->(i:Ingredient)
WITH r, COLLECT(i.is_allergene) AS are_allergeni, COLLECT(i.name) AS
ingredienti
WHERE all(x IN are_allergeni WHERE NOT x)
RETURN r.name AS ricetta, ingredienti
ORDER BY rand()
LIMIT 5 """
display(print_first_n(cq))
```

ricetta	ingredienti
Parfait alle mandorle	['Zucchero', 'Zucchero', 'Burro', 'Tuorli', 'Acqua', 'Acqua', 'Panna fresca liquida', 'Cioccolato fondente', 'Mandorle pelate']
Crostoni al formaggio	['Burro', 'Pepe nero', 'Panna fresca liquida', 'Rosmarino', 'Grana Padano DOP', 'Pane casereccio', 'Edamer', 'Emmentaler']
Pasteis de nata	['Zucchero', 'Farina 00', 'Burro', 'Sale fino', 'Tuorli', 'Acqua', 'Farina Manitoba', 'Panna fresca liquida']
Pettole	['Zucchero', 'Farina 00', 'Acqua', 'Lievito di birra fresco', 'Miele', 'Sale grosso', 'Olio di semi', 'Olive nere', 'Olive verdi']
Fusilloni con salsiccia, datterini gialli e provolone	['Sale fino', 'Olio extravergine doliva', 'Pepe nero', 'Vino bianco', 'Timo', 'Salsiccia', 'Scalogno', 'Provolone', 'Pomodorini datterini', 'Fusilloni']

Modifiche

Come si può notare, alcuni ingredienti che sono allergeni non sono impostati come tali nel database (es. farina). Ho quindi creato il seguente script per dare la possibilità all'utente di cambiare la property is_allergene di un certo ingrediente.

In [25]:
```
ingrediente=""
while ingrediente=="":
    ingrediente=input("Inserisci nome dell'ingrediente da modificare:
").lower()
# ingrediente="farina"

print("Ecco gli ingredienti trovati per la ricerca '"+ingrediente+"'")
cq = "MATCH (i:Ingredient) WHERE toLower(i.name) CONTAINS '"+ingrediente+"'
RETURN i.name AS ingrediente"
ris = graph.run(cq).data()
ris = [i["ingrediente"] for i in ris]
print_columns(ris, 2)

ingrediente2=""
while ingrediente2=="":
    ingrediente2=input("Inserisci il nome dell'ingrediente da modificare,
oppure inserisci 0 per modificare tutti quelli stampati: ")
# ingrediente2 = "Farina 00"

opzione_allergene=""
while opzione_allergene == "":
    opzione_allergene=input("Modificare l'ingrediente/gli ingredienti come:
\n1. Allergene/i\n2. Non allergene/i\n")
    opzione_allergene = True if opzione_allergene=="1" else (False if
opzione_allergene=="2" else "")

print("\n\nHai scelto di impostare" + \
        " come" + (" non" if not opzione_allergene else "") + " allergen" +
        ("e l'ingrediente " + ingrediente2 if ingrediente2 != "0" else "i gli
ingredienti: " + str(ris)) +
        ". "
        )

where_clause = "toLower(i.name) CONTAINS '"+ingrediente+"'"  if
ingrediente2=="0" else "i.name='"+ingrediente2+"'"
cq = "MATCH (i:Ingredient) WHERE "+where_clause+" SET
```

```
i.is_allergene="+str(opzione_allergene)
graph.run(cq)
```

```
Ecco gli ingredienti trovati per la ricerca 'farina'
   Farina 00                                                Farina Manitoba
   Farina di grano saraceno                                 Farina 0
   Farina di ceci                                           Farina di mandorle
   Farina                                                   Farina di nocciole
   Farina di mais                                           Farina di mais fioretto
   Farina di pistacchi                                      Farina di castagne
   Farina integrale                                         Tortillas di farina
   Farina di riso                                           Farina 1
   Farina di frumento                                       Farina per pinsa
   Farina per polenta taragna                               Farina di farro integrale
   Farina istantanea per polenta                            Farina di mais bramata
   Farina 2                                                 Farina di farro
   Farina di Kamut®                                         Farina di grano duro
   Farina di mais giallo                                    Farina istantanea per polen
ta taragna
   Farina di segale                                         Farina di cocco
   Farina di semi di carrube                                Farina di mais integrale pe
r polenta istantanea
   Farina di riso senza glutine                             Farina per chapati
   Farina di mais bianco                                    Farina di avena
   Farina di tapioca                                        Farina di mais e cereali pe
r polenta istantanea
   Farina di mais bramata bianca                            Farina di mais senza glutin
e
   Farina di grano saraceno senza glutine                   Farina di grano arso
   Farina di riso integrale                                 Farina di lupini
   Farina di mais fumetto                                   Farina di teff
   Farina per pane senza glutine                            Farina di riso venere
   Farina di castagne senza glutine                         Farina di amaranto
   Farina di mais integrale                                 Farina di quinoa
   Farina integrale di miglio                               Farina di enkir
   Chicchi di farina di legumi (lenticchie rosse e ceci)    Chicchi di farina di legumi
(lenticchie rosse
   Farina di canapa                                         Chicchi di farina di legumi
(lenticchie rosse e piselli)


Hai scelto di impostare come allergene l'ingrediente Farina 00.
```

Out[25]: (No data)

Cancellazioni

Ho inserito il seguente script per dare la possibilità all'utente di cancellare una ricetta. Ho inserito `DETACH` per eliminare anche gli archi legati al nodo da eliminare.

In [22]:
```
nome_ricetta=input("Inserisci nome della ricetta che desideri eliminare: ")
cq = "MATCH (r:Recipe {name: '"+nome_ricetta+"'})  DETACH DELETE (r)"
graph.run(cq)
print("Ricetta " + nome_ricetta + " eliminata correttamente!")
```

```
Ricetta Tiramisù eliminata correttamente!
```