به نام خدا

مرتضی فائزی نیا تسک مقدماتی

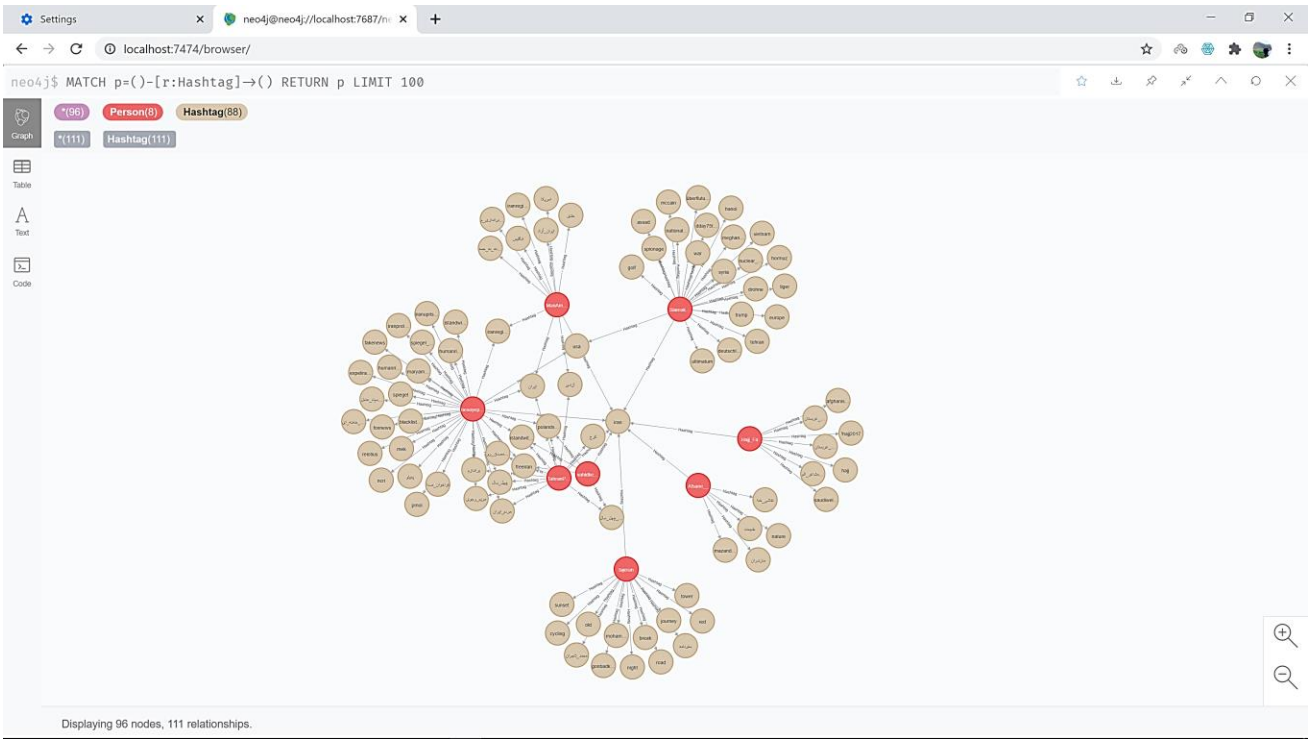
بخش اول)

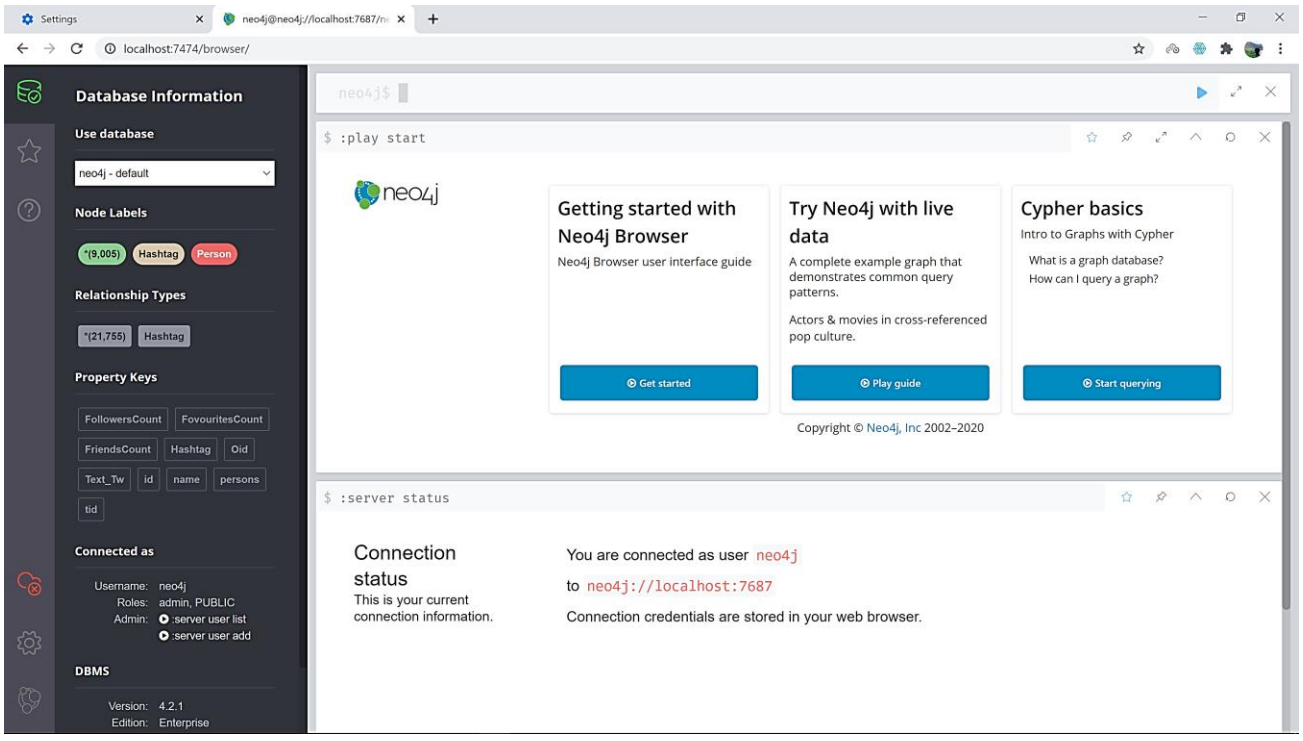
- **مراحل انجام شده**
  - تمیز سازی داده
  - استخراج داده مورد نیاز با توجه به هدف
  - ذخیره و ایجاد مدل داده ها روی پایگاه داده ***Neo4j***

در این بخش با استفاده از زبان پایتون عملیات لازم روی داده طبق مراحل بالا انجام شد که کد آن در فایل ***Part1.py*** موجود است در این کد طی تابع ***read_data*** عمل خواندن و تمیزسازی داده انجام شده است و سپس پس از رسیدن به نتایج مطلوب با استفاده از تابع ***processing*** کوئری لازم برای افزودن داده ها در پایگاه داده انجام می شود.

- **ایجاد مدل در پایگاه داده**
  - نام های کاربری افراد منتشرکننده توییت
  - هشتگ های موجود در توییت ها
  -

در این بخش دو نوع داده در نظر گرفته شده است یعنی گره هایی که کاربران و متا دیتا های آن ها عضو های آن هستند و گره های نوع دوم کل هشتگ های موجود در این مجموعه داده هستند. که این کوئری ها در تابع ***processing*** موجود اند که با استفاده از تابع تعریف شده ***_create_and_return_result*** عمل ارسال و اجرا روی پایگاه داده انجام شده است. شکل زیر ۱۰۰ گره نمونه از این پایگاه داده ***(neo4j)*** را نشان می دهد:


Settings
neo4j@neo4j://localhost:7687/
localhost:7474/browser/
neo4j$ MATCH p=()-[r:Hashtag]→() RETURN p LIMIT 100
*(96)
Person(8)
Hashtag(88)
*(111)
Hashtag(111)
Graph
Table
Text
Code
Displaying 96 nodes, 111 relationships.


Settings
neo4j@neo4j://localhost:7687/
localhost:7474/browser/
Database Information
Use database
neo4j - default
Node Labels
*(9,005)
Hashtag
Person
Relationship Types
*(21,755)
Hashtag
Property Keys
FollowersCount
FovouritesCount
FriendsCount
Hashtag
Oid
Text_Tw
id
name
persons
tid
Connected as
Username: neo4j
Roles: admin, PUBLIC
Admin: :server user list
:server user add
DBMS
Version: 4.2.1
Edition: Enterprise
neo4j$
$:play start
neo4j
Getting started with Neo4j Browser
Neo4j Browser user interface guide
Get started
Try Neo4j with live data
A complete example graph that demonstrates common query patterns.
Actors & movies in cross-referenced pop culture.
Play guide
Cypher basics
Intro to Graphs with Cypher
What is a graph database?
How can I query a graph?
Start querying
Copyright © Neo4j, Inc 2002–2020
$:server status
Connection status
This is your current connection information.
You are connected as user neo4j
to neo4j://localhost:7687
Connection credentials are stored in your web browser.

بخش دوم)

➢ مراحل انجام شده

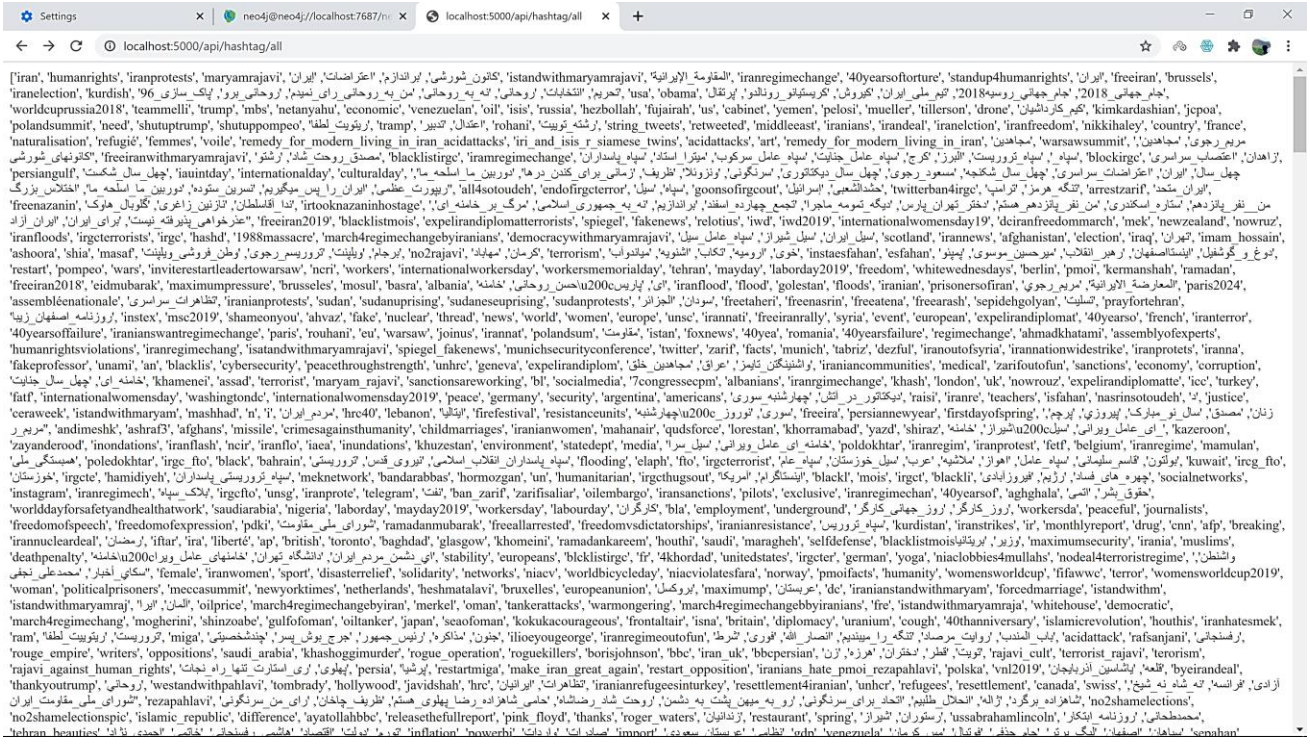
- استفاده از خروجی بخش اول برای داده ها
- راه اندازی **api** برای داده با استفاده از فریمورک **Flask**

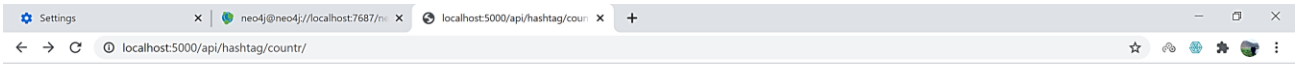
در این بخش با استفاده از زبان پایتون و فریمورک **Flask** یک **api** روی پورت ۵۰۰۰ راه اندازی شده است که با اجرای فایل **Part2_Flask.py** اجرا می شود (به منظور استفاده از داده باید کد بخش اول و پایگاه داده ران باشد).

➢ عملیات

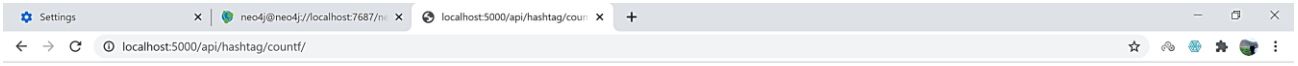
- برگشت همه هشتگ ها: http://localhost:5000/api/hashtag/all/
- برگشت ۱۰ هشتگ با بیشترین تکرار: http://localhost:5000/api/hashtag /countr/
- برگشت ۱۰ هشتگ منتشر شده از افراد مهم: http://localhost:5000/api/hashtag /countrf/

که تصویر خروجی به صورت زیر می باشد:





['iran', 'humanrights', 'iranprotests', 'maryamrajavi', 'ایران', 'اعتراضات', 'براندازم', 'کانون_شورشی', 'istandwithmaryamrajavi', 'المقاومة_الإيرانية']

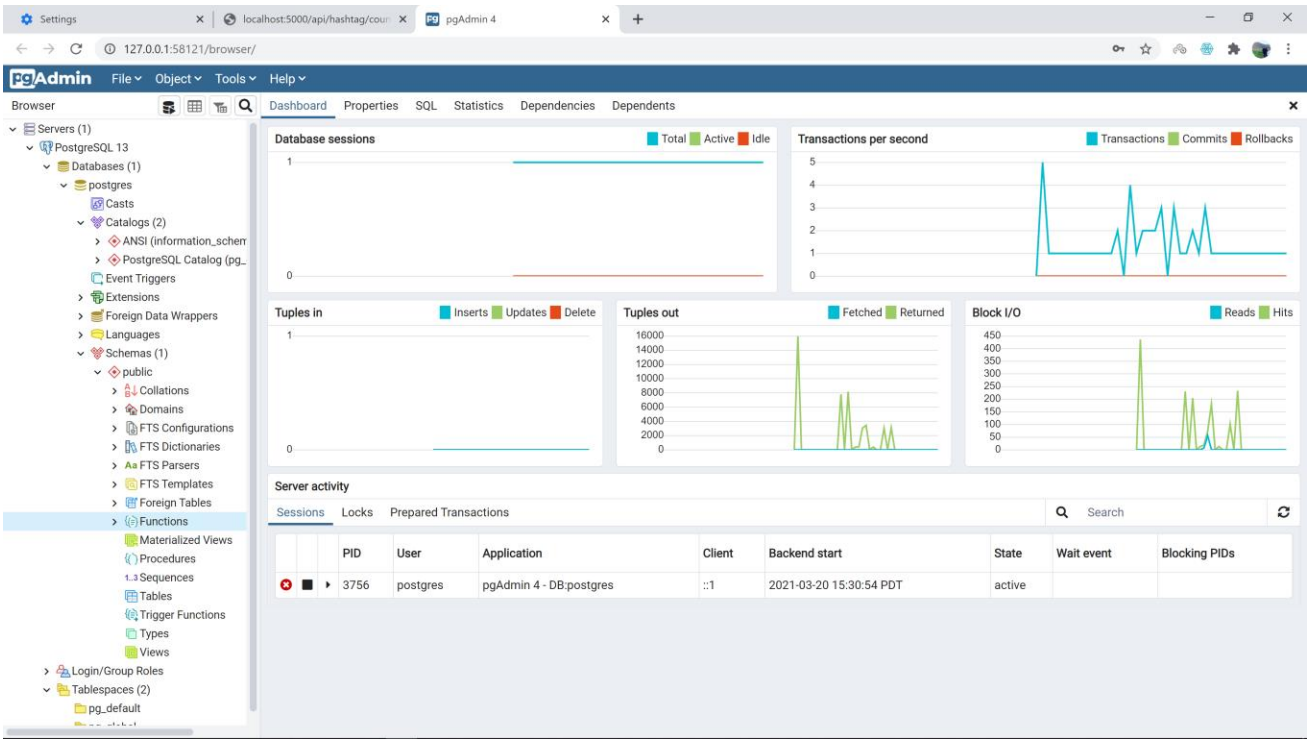


['iran', 'cuba', 'estamosunidosve', 'venezuela', 'roadtorussia', 'russia2018', 'apexlegends', 'ea', 'nowruz', 'london']

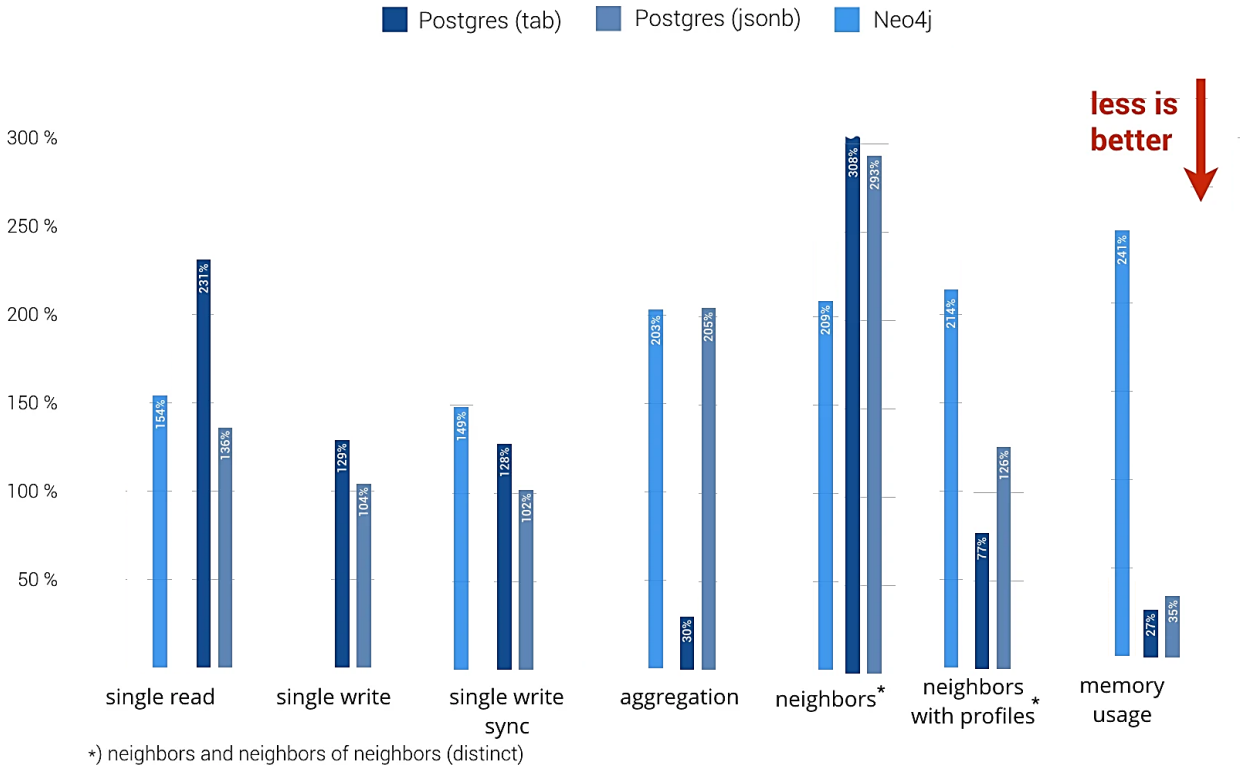
بخش سوم)

- **مراحل انجام شده**
  - تکرار بخش اول
  - مقایسه دو پایگاه داده **Neo4j** و **PostgreSQL**

در این بخش با استفاده از زبان پایتون عملیات لازم روی داده طبق مراحل بالا انجام شد که کد آن در فایل ***Part3.py*** موجود است در این کد طی تابع ***read_data*** عمل خواندن و تمیزسازی داده انجام شده است و سپس با استفاده از **creating** جداول مورد نیاز ایجاد می شوند. سپس از رسیدن به نتایج مطلوب با استفاده از تابع ***processing*** کوئری لازم برای افزودن داده ها در پایگاه داده انجام می شود.



برای مقایسه دو پایگاه داده متاسفانه کتابخانه **locust** علارغم تلاش بالا نیامد به صورت ریسرچی مقایسه این دو پایگاه داده به صورت زیر است:



*) neighbors and neighbors of neighbors (distinct)

	single read (s)	single write (s)	single write sync (s)	aggregation (s)	shortest (s)	neighbors 2nd (s)	neighbors 2nd data (s)	memory (GB)
Neo4j 3.3.1	153.65%		149.37%	203.45%	199.94%	208.96%	214.22%	240.68%
	35.73		43.22	2.18	0.83	2.99	11.04	37.00
PostGres 10.1 (tabular)	231.17%	129.03%	127.70%	29.62%		307.96%	76.87%	26.68%
	53.77	36.22	36.10	0.32		4.41	3.96	4.10
PostGres 10.1 (jsonb)	135.96%	104.34%	101.55%	204.55%		292.57%	126.14%	35.36%
	31.62	29.29	28.70	2.20		4.19	6.50	5.43