

DAO Tabanlı Oyun Yönetişiminin CSV ile Graph Veritabanına Aktarımı

Bir önceki hafta seçtiğim domain alanı olan oyun içi DAO yönetim analizi projemden devam ediyorum . Bu projeyi yaparken bir CSV dosyasından yola çıkarak yapmamıştım . Ama orada oluşturduğum mantık üzerinden giderek bugün kendime ufak bir csv dosyası hazırladım. Zaten düğümlerime, ilişkilerime ve bunların özelliklerini belirlemiştim başlangıç için . Bu özelliklerle yapay zekadan -user adları isteme gibi- sentetik verim için destek olarak geçen hafta Merge ile oluşturduğum ve Aura'da incelediğim datasetimi hem biraz daha büyütüyor hemde ödevimiz olan .csv dosyalarımı Cypher kullanarak yüklemeyi gerçekleştiriyorum. Csv'lerimi oluştururken 50 tane User tanımladım , 15 tane guild tanımladım . DAO Token ı olanları oy verebilir diye hesapladım ve oy verebilenler arasından delegate yapmaya özen gösterdim. Verilerimi yükledikten sonra, her kullanıcının sahip olduğu token miktarı ve tokenın değeri (governance power) üzerinden oy hakkının deerini hesaplayacak bir Cypher sorgusu yazarak, oy verme haklarını atayacağım.

CSV dosyalarımın linkleri

Düğümlemlerim için:

<https://raw.githubusercontent.com/jaguuai/Homeworks/main/Week4/Nodes/1.users.csv>

<https://raw.githubusercontent.com/jaguuai/Homeworks/main/Week4/Nodes/2.guilds.csv>

<https://raw.githubusercontent.com/jaguuai/Homeworks/main/Week4/Nodes/3.proposals.csv>

<https://raw.githubusercontent.com/jaguuai/Homeworks/main/Week4/Nodes/4.tokens.csv>

<https://raw.githubusercontent.com/jaguuai/Homeworks/main/Week4/Nodes/5.tokenypes.csv>

İlişkilerim için:

<https://raw.githubusercontent.com/jaguuai/Homeworks/main/Week4/Relationships/created.csv>

https://raw.githubusercontent.com/jaguuai/Homeworks/main/Week4/Relationships/delegate_to.csv

https://raw.githubusercontent.com/jaguuai/Homeworks/main/Week4/Relationships/holds_token.csv

https://raw.githubusercontent.com/jaguuai/Homeworks/main/Week4/Relationships/member_of.csv

https://raw.githubusercontent.com/jaguuai/Homeworks/main/Week4/Relationships/voted_for.csv

https://raw.githubusercontent.com/jaguuai/Homeworks/main/Week4/Relationships/has_tokenype.csv

Önce düğümlerimi ekliyorum hepsi birdende eklenebilir fakat gözlemlemek adına önce düğümlerim için Cypher kodumu yazıp Neo4j Aura içine ekledim.

Instance: Free instance Database: neo4j User: Aura (ayseeeyilmaz96@gmail.com)

Database information

Nodes (85)

- Guild
- Proposal
- Token
- TokenType
- User

Relationships (0)

Property keys

- createdAt
- creationDate
- creatorType
- endDate
- exampleUsecase
- governancePower
- guildId
- joinDate
- level
- memberCount
- name
- price
- proposalId
- reputationScore
- startDate
- status
- symbol
- title
- tokenId
- tokenType

Show all (5 more)

Last update: 2:02:04 PM

```

1 CREATE CONSTRAINT userId_constraint IF NOT EXISTS
2 FOR (u:User)
3 REQUIRE u.userId IS UNIQUE;
4
5 CREATE CONSTRAINT name_constraint IF NOT EXISTS
6 FOR (u:User)
7 REQUIRE u.name IS UNIQUE;
8
9 CREATE CONSTRAINT guildId_constraint IF NOT EXISTS
10 FOR (g:Guild)
11 REQUIRE g.guildId IS UNIQUE;
12
13 CREATE CONSTRAINT guildName_constraint IF NOT EXISTS
14 FOR (g:Guild)
15 REQUIRE g.name IS UNIQUE;
16
17 CREATE CONSTRAINT proposalId_constraint IF NOT EXISTS
18 FOR (p:Proposal)
19 REQUIRE p.proposalId IS UNIQUE;
20
21 CREATE CONSTRAINT tokenId_constraint IF NOT EXISTS

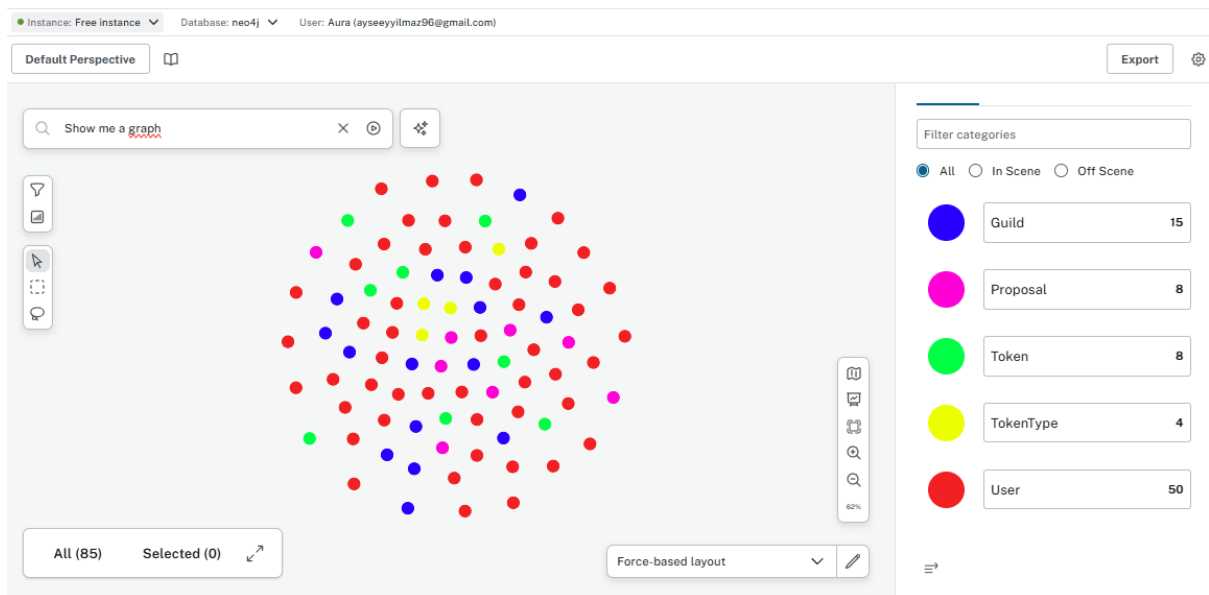
```

CREATE CONSTRAINT userId_constraint IF NOT EXISTS

CREATE CONSTRAINT name_constraint IF NOT EXISTS

CREATE CONSTRAINT guildId_constraint IF NOT EXISTS

CREATE CONSTRAINT guildName_constraint IF NOT EXISTS



CREATE CONSTRAINT userId_constraint IF NOT EXISTS
 FOR (u:User)
 REQUIRE u.userId IS UNIQUE;

CREATE CONSTRAINT name_constraint IF NOT EXISTS
 FOR (u:User)
 REQUIRE u.name IS UNIQUE;

CREATE CONSTRAINT guildId_constraint IF NOT EXISTS

```
FOR (g:Guild)
  REQUIRE g.guildId IS UNIQUE;
```

```
CREATE CONSTRAINT guildName_constraint IF NOT EXISTS
FOR (g:Guild)
  REQUIRE g.name IS UNIQUE;
```

```
CREATE CONSTRAINT proposalId_constraint IF NOT EXISTS
FOR (p:Proposal)
  REQUIRE p.proposalId IS UNIQUE;
```

```
CREATE CONSTRAINT tokenTypeId_constraint IF NOT EXISTS
FOR (tt:TokenType)
  REQUIRE tt.tokenTypeId IS UNIQUE;
```

```
CREATE CONSTRAINT tokenTypeName_constraint IF NOT EXISTS
FOR (tt:TokenType)
  REQUIRE tt.tokenType IS UNIQUE;
```

```
CREATE CONSTRAINT tokenId_constraint IF NOT EXISTS
FOR (t:Token)
  REQUIRE t.tokenId IS UNIQUE;
```

```
CREATE CONSTRAINT tokenSymbol_constraint IF NOT EXISTS
FOR (t:Token)
  REQUIRE t.symbol IS UNIQUE;
```

```
LOAD CSV WITH HEADERS
FROM
'https://raw.githubusercontent.com/jaguuai/Homeworks/main/Week4/Nodes/1.users.csv' AS users
MERGE (u:User {userId: toInteger(users.userId)})
SET u.name = users.name,
    u.level = toInteger(users.level),
    u.joinDate = datetime(replace(users.joinDate, '-', 'T'));
```

```
LOAD CSV WITH HEADERS
```

```

FROM
'https://raw.githubusercontent.com/jaguuai/Homeworks/main/Week4/Nodes/2.guilds.
csv' AS guilds
MERGE (g:Guild {guildId: toInteger(guilds.guildId)})
SET g.name = guilds.name,
    g.creationDate = datetime(replace(guilds.creationDate, '-', 'T')),
    g.memberCount = toInteger(guilds.memberCount),
    g.verified = CASE WHEN guilds.verified = 'True' THEN true ELSE false END,
    g.reputationScore = toInteger(guilds.reputationScore);

```

LOAD CSV WITH HEADERS

```

FROM
'https://raw.githubusercontent.com/jaguuai/Homeworks/main/Week4/Nodes/3.propo
sals.csv' AS proposals
MERGE (p:Proposal {proposalId: toInteger(proposals.proposalId)})
SET p.title = proposals.title,
    p.creatorType = proposals.creatorType,
    p.createdAt = datetime(replace(proposals.createdAt, '-', 'T')),
    p.status = proposals.status,
    p.voteCount = toInteger(proposals.voteCount),
    p.startDate = date(proposals.startDate),
    p.endDate = date(proposals.endDate),
    p.type = proposals.type;

```

LOAD CSV WITH HEADERS

```

FROM
'https://raw.githubusercontent.com/jaguuai/Homeworks/main/Week4/Nodes/4.tokens
.csv' AS tokens
MERGE (t:Token {tokenId: toInteger(tokens.tokenId)})
SET t.name = tokens.name,
    t.symbol = tokens.symbol,
    t.price = toFloat(tokens.price),
    t.governancePower = toInteger(tokens.governancePower),
    t.tokenTypeId = toInteger(tokens.tokenTypeId);

```

LOAD CSV WITH HEADERS

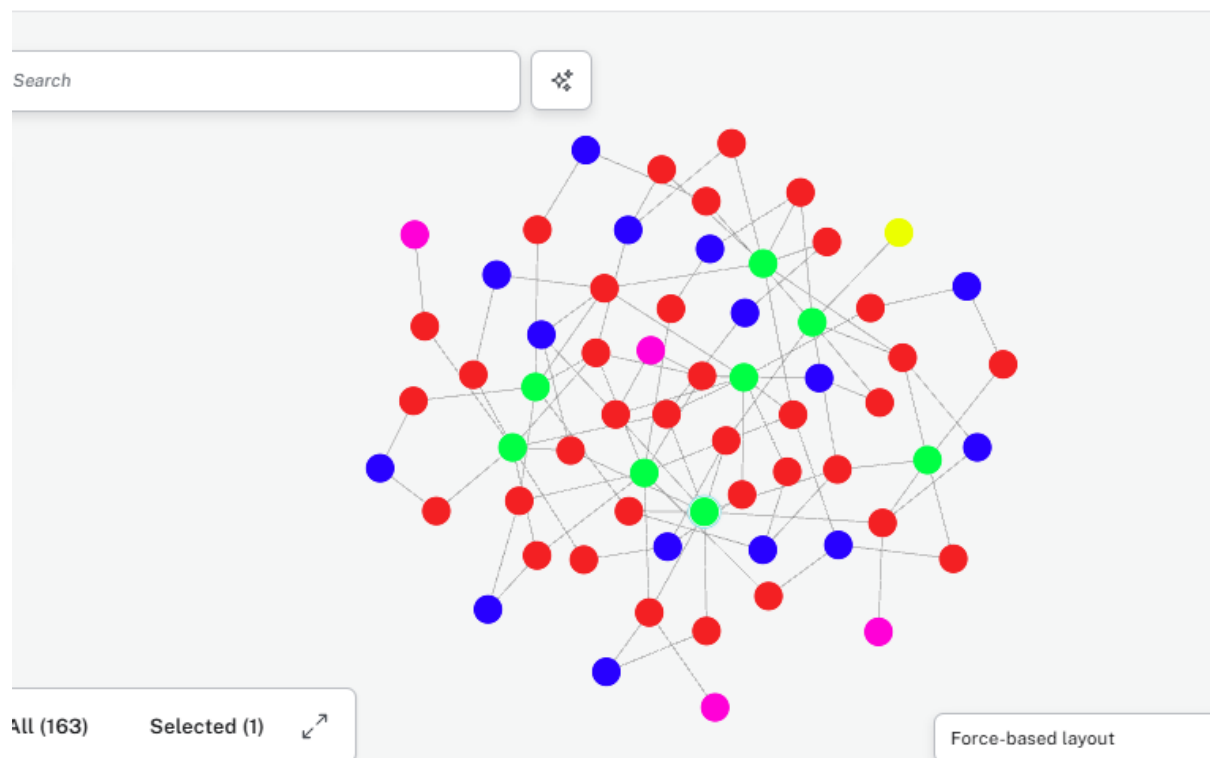
FROM

'https://raw.githubusercontent.com/jaguuai/Homeworks/main/Week4/Nodes/5.token
types.csv' AS types

MERGE (tt:TokenType {tokenId: toInteger(types.tokenTypeId)})

SET tt.tokenType = types.tokenType,
tt.exampleUsecase = types.exampleUsecase;

Daha sonra düğümlerim, ekliyorum :



LOAD CSV WITH HEADERS

FROM 'https://raw.githubusercontent.com/jaguuai/Homeworks/main/Week4/Relationships/created.csv' AS rel

MATCH (p:Proposal {proposalId: toInteger(rel.proposalId)})

WITH rel, p

WHERE rel.creatorType = 'user'

MATCH (u:User {userId: toInteger(rel.userId)})

MERGE (u)-[r:CREATED]->(p)

SET r.timestamp = datetime(rel.timestamp),

r.verified = toBoolean(rel.verified);

LOAD CSV WITH HEADERS

FROM 'https://raw.githubusercontent.com/jaguuai/Homeworks/main/Week4/Relationships/created.csv' AS rel

MATCH (p:Proposal {proposalId: toInteger(rel.proposalId)})

```

WITH rel, p
WHERE rel.creatorType = 'guild'
MATCH (g:Guild {guildId: toInteger(rel.userId)})
MERGE (g)-[r:CREATED]->(p)
SET r.timestamp = datetime(rel.timestamp),
    r.verified = toBoolean(rel.verified);

```

```

LOAD CSV WITH HEADERS
FROM 'https://raw.githubusercontent.com/jaguuai/Homeworks/main/Week4/Relationships/delegate_to.csv' AS
row
//Delegator (USER)
MATCH (delegator:User {userId: toInteger(row.delegatorId)})
//Delegatee (USER|GUILD)
MATCH (delegatee WHERE
CASE row.delegateeType
WHEN 'user' THEN delegatee:User AND delegatee.userId = toInteger(row.delegateeId)
WHEN 'guild' THEN delegatee:Guild AND delegatee.guildId = toInteger(row.delegateeId)
END)

```

```

MERGE (delegator)-[d:DELEGATED_TO]->(delegatee)
SET d.since = datetime(row.since),
    d.weight = toFloat(row.weight),
    d.reason = row.reason,
    d.isActive = toBoolean(row.isActive);

```

```

LOAD CSV WITH HEADERS
FROM 'https://raw.githubusercontent.com/jaguuai/Homeworks/main/Week4/Relationships/holds_token.csv' AS
row
//User|Guild
MATCH (holder WHERE
CASE row.holderType
WHEN 'user' THEN holder:User AND holder.userId = toInteger(row.userId)
WHEN 'guild' THEN holder:Guild AND holder.guildId = toInteger(row.userId)
END)
// Token
MATCH (token:Token {tokenId: toInteger(row.tokenId)})

```

```

MERGE (holder)-[h:HOLDS_TOKEN]->(token)
SET h.amount = toInteger(row.amount),
    h.source = row.source;

```

```

LOAD CSV WITH HEADERS
FROM 'https://raw.githubusercontent.com/jaguuai/Homeworks/main/Week4/Relationships/member_of.csv' AS
row
MATCH (u:User {userId: toInteger(row.userId)})
MATCH (g:Guild {guildId: toInteger(row.guildId)})
MERGE (u)-[m:MEMBER_OF]->(g)
SET m.joinDate = date(row.joinDate),
    m.role = row.role,
    m.status = row.status,
    m.contributionScore = toInteger(row.contributionScore);

```

LOAD CSV WITH HEADERS

```
FROM 'https://raw.githubusercontent.com/jaguuai/Homeworks/main/Week4/Relationships/voted_for.csv' AS row
MATCH (u:User {userId: toInteger(row.userId)})
MATCH (p:Proposal {proposalId: toInteger(row.proposalId)})
MERGE (u)-[v:VOTED_FOR]->(p)
SET v.timestamp = datetime(row.timestamp),
    v.delegated = toBoolean(row.delegated),
    v.voteWeight = toFloat(row.voteWeight),
    v.source = row.source;
```

LOAD CSV WITH HEADERS

```
FROM 'https://raw.githubusercontent.com/jaguuai/Homeworks/main/Week4/Relationships/has_tokenType.csv' AS row
MATCH (t:Token {tokenId: toInteger(row.tokenId)})
MATCH (tt:TokenType {tokenId: toInteger(row.tokenId)})
MERGE (t)-[:IS_TYPE]->(tt);
```

Yine oluşturduğum datasetimde usecaselerimide görmek istiyorum:

1.Kim teklif oluşturdu?

Burada hem User ım hemde Guild lerim teklif oluşturabiliyordu daha büyük datasetlerde arama kolaylığı için Creator adında yeni bir node tanımlayabilirim.

```
MATCH (u:User)-[:CREATED]->(p:Proposal)
WITH DISTINCT u
SET u:Creator;

MATCH (g:Guild)-[:CREATED]->(p:Proposal)
WITH DISTINCT g
SET g:Creator;
```

neo4j\$ MATCH (u:User)-[:CREATED]->(:Proposal) WITH DISTINCT u SET u:Creator; MATCH (g:Guild)-[:CREATED]->(:Proposal)

MATCH (u:User)-[:CREATED]->(:Proposal)

MATCH (g:Guild)-[:CREATED]->(:Proposal)

```

1 MATCH (c:Creator)
2 RETURN
3 CASE
4   WHEN 'User' IN labels(c) THEN c.userId
5   WHEN 'Guild' IN labels(c) THEN c.guildId
6 END AS id,
7 labels(c) AS types,
8 COUNT { (c)-[:CREATED]->() } AS createdProposals
9 ORDER BY createdProposals DESC;

```

id	types	createdProposals
12	["User", "Creator"]	1
21	["User", "Creator"]	1
23	["User", "Creator"]	1
35	["User", "Creator"]	1
4	["Guild", "Creator"]	1
7	["Guild", "Creator"]	1
9	["Guild", "Creator"]	1
11	["Guild", "Creator"]	1

Bu şekilde daha okunaklı kim tarafından kaç tane teklif oluşturulmuş görmüş olduk.

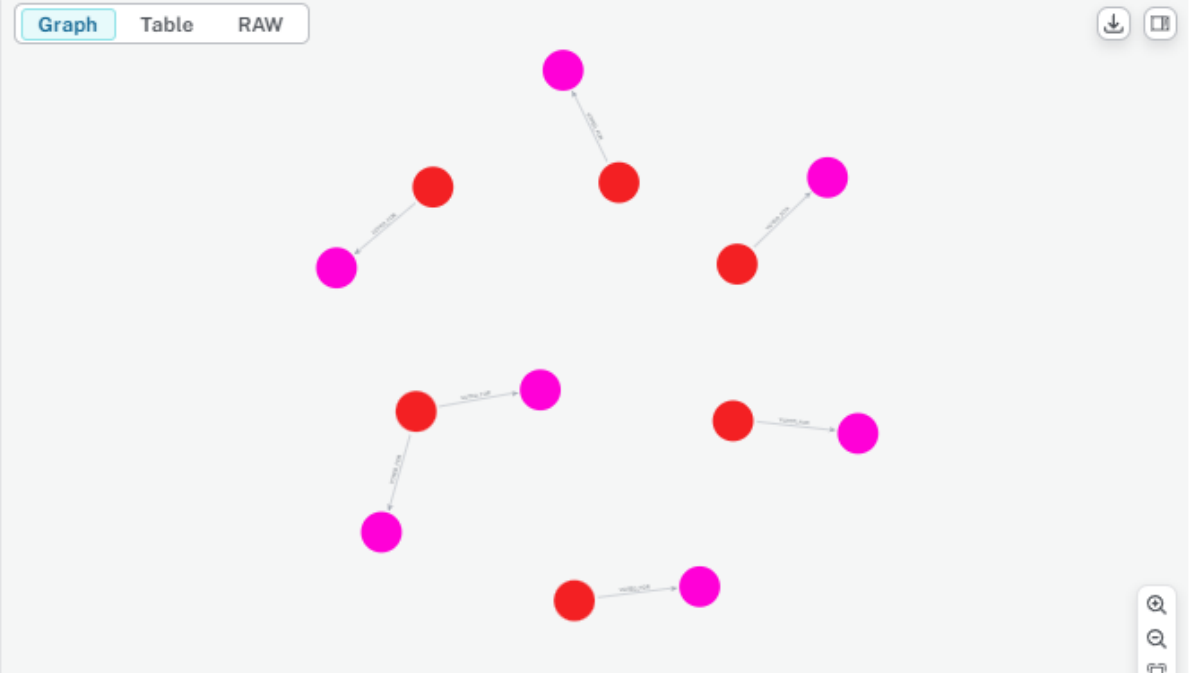
2.Kullancılar hangi tekliflere oy verdi?

neo4j\$ MATCH (u:User)-[VOTED_FOR]->(p:Proposal) RETURN u.userId AS KullanıcıID, u.name AS KullanıcıAdı, p.

KullanıcıID	KullanıcıAdı	TeklifID
10	"NovaSniper6"	1
9	"QuantumFox40"	2
1	"NovaByte82"	3
2	"PixelKing47"	5
10	"NovaSniper6"	6
7	"DragonZer034"	7
5	"GhostFury8"	8

Started streaming 7 records after 23 ms and completed after 23 ms.


```
neo4j$ MATCH p=()-[:VOTED_FOR]->() RETURN p LIMIT 25;
```



3.Oy hakkını başkasına devredenler kimler?

Bu soguda geçen seferde datalarımı eklerken typeımı vermediğim için id leri alıyordum ve eklenmesi güzel olur denmişim burda csv de bunu eklediğim için user mı yoksa guild mi görebiliyorum.

```
1 MATCH (delegator)-[d:DELEGATED_TO]->(delegatee)
2 RETURN
3   delegator.userId AS delegatorId,
4   labels(delegator)[0] AS delegatorType,
5   CASE
6     WHEN 'User' IN labels(delegatee) THEN delegatee.userId
7     WHEN 'Guild' IN labels(delegatee) THEN delegatee.guildId
8   END AS delegateeId,
9   labels(delegatee)[0] AS delegateeType,
10  d.since AS since,
11  d.weight AS weight,
12  d.reason AS reason,
13  d.isActive AS isActive
14 ORDER BY d.since DESC;
```

	delegatorId	delegatorType	delegateeId	delegateeType	since	weight	reason
1	5	"User"	9	"Guild"	2025-02-15T00:00:00Z	1.0	"Guild leadership"
2	2	"User"	1	"User"	2025-01-01T00:00:00Z	0.6	"Trusted advisor"
3	7	"User"	3	"User"	2024-12-01T00:00:00Z	0.8	"Technical expertise"

4.Kim tokenların büyük kısmını elinde tutuyor?

```

1 MATCH (u:User)-[h:HOLDS_TOKEN]->(t:Token)-[:IS_TYPE]->(tt:TokenType)
2 RETURN
3   u.userId AS userId,
4   u.name AS userName,
5   sum(h.amount * t.governancePower) AS totalVotingPower,
6   collect([
7     token: t.name,
8     amount: h.amount,
9     tokenPower: t.governancePower,
10    type: tt.tokenType
11  ]) AS tokenDetails
12 ORDER BY totalVotingPower DESC;

```

userId	userName	totalVotingPower	tokenDetails
19	"RogueGhost46"	9300000	[{ amount: 1400, tokenPower: 3500, token: "DAO Token", type: "Governance Token" }, { amount: 1100, tokenPower: 4000, token: "Blockchain Token", type: "Blockchain Token" }]
10	"NovaSniper6"	7970000	[{

Aşağıda çıktımdan ilk 3 satırı koydum burda gördüğümüz üzere analizi etmek için bu işlemi yapıp totalVotinPower hesaplamak gerekli bir durum çünkü . Token sayısı , türü ya da kaç tane olduğunu tek başına kimin tokenarın büyük kısmını tuttuğunu direkt göstermiyor o yüzden bu hesapladığımız değeri User!lara özellik olarak eklemeye karar verdim.

```

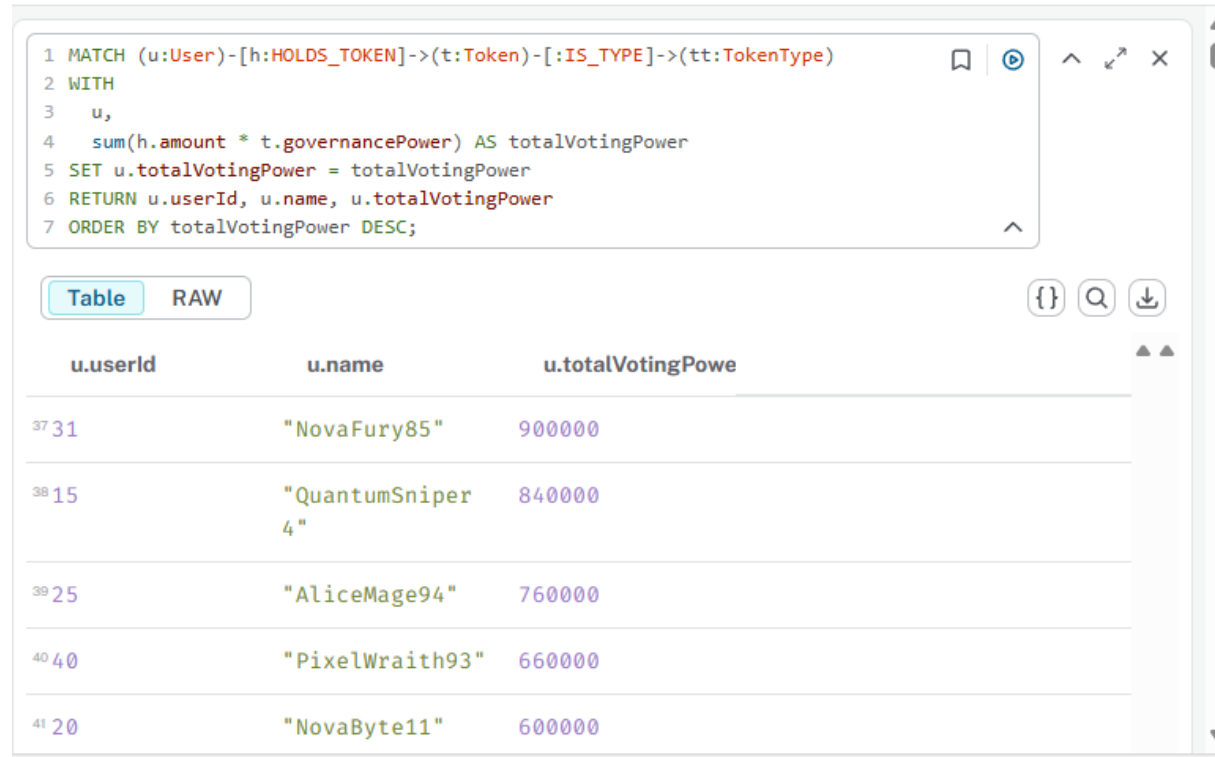
19,RogueGhost46,9300000,"[
{
  amount: 1400,
  tokenPower: 3500,
  token: DAO Token,
  type: Governance Token
},
{
  amount: 1100,
  tokenPower: 4000,
  token: Blockchain Token,
  type: Blockchain Token
}
]"
10,NovaSniper6,7970000,"[
{
  amount: 1200,
  tokenPower: 3500,
  token: DAO Token,
  type: Governance Token
},
{
  amount: 850,
  tokenPower: 1200,
  token: Slot Game Token,
  type: Utility Token
},
{
  amount: 1100,

```

```

    tokenPower: 2500,
    token: LegalTech Token,
    type: Security Token
  }
]"
21,NovaClaw7,7375000,"[
{
  amount: 1250,
  tokenPower: 3500,
  token: DAO Token,
  type: Governance Token
},
{
  amount: 1200,
  tokenPower: 2500,
  token: LegalTech Token,
  type: Security Token
}
]"

```



The screenshot shows a database query interface. At the top, a Cypher query is entered in a text area. Below the query, there are two tabs: 'Table' (selected) and 'RAW'. To the right of the tabs are icons for schema, search, and download. The results are displayed in a table with three columns: 'u.userId', 'u.name', and 'u.totalVotingPower'. The table contains five rows of data, each with a row number on the left.

```

1 MATCH (u:User)-[h:HOLDS_TOKEN]->(t:Token)-[:IS_TYPE]->(tt:TokenType)
2 WITH
3   u,
4   sum(h.amount * t.governancePower) AS totalVotingPower
5 SET u.totalVotingPower = totalVotingPower
6 RETURN u.userId, u.name, u.totalVotingPower
7 ORDER BY totalVotingPower DESC;

```

	u.userId	u.name	u.totalVotingPower
37	31	"NovaFury85"	900000
38	15	"QuantumSniper 4"	840000
39	25	"AliceMage94"	760000
40	40	"PixelWraith93"	660000
41	20	"NovaByte11"	600000

Bu şekilde artık tokenları olan kullanıcılara yeni bir özellik ekledim ve artık sadece aşağıdaki sorguyla kolayca token hakkı en yüksek olan kullanıcıları bulabiliyoruz.

neo4j\$ MATCH(u:User) RETURN u.totalVotingPower	
Table	RAW
u.totalVotingPowe	
1	5690000
2	7170000
3	6445000
4	4275000
5	4180000
6	6950000
7	7970000
8	2300000

5. Hangi kullanıcı ya da klan hangi türden tokenlara sahip?

Bu sorguyuda tüm nodelardan token lara sahip olanları bul bu tokenlarında türünü bul diyerek yapabiliyoruz. Fakat id adlarını direkt id olarak düzenlemenin daha mantıklı olacağını düşündüm bu sorguda eğer öyle yaparsak zaten her aramada referens olarak verdiğimiz harfu olsun mesela u.id RETURN ediyorduk. Aşağıdaki gibi tek sorguda bulabilmek adına bunun başka zamanlarda da işime yaracağını düşünüyorum.

<pre> 1 MATCH (n)-[:HOLDS_TOKEN]->(t:Token)-[:IS_TYPE]->(tt:TokenType) 2 RETURN DISTINCT labels(n) AS nodeLabels, n.id AS nodeId, tt.tokenType AS tokenType 3 ORDER BY nodeLabels, nodeId, tokenType; </pre>		
Table	RAW	{ } Q ↓
nodeLabels	nodeId	tokenType
8 ["User"]	null	"Blockchain Token"
9 ["User"]	null	"Governance Token"
10 ["User"]	null	"Security Token"
11 ["User"]	null	"Utility Token"
12 ["User", "Creator"]	null	"Governance Token"

Bu şekilde id ler null geliyor çünkü user ise userId yazdım guild ise guildId yazdım ve bu şekilde yazımın özellik return ederken bir artısı olmayacak bakınca. Bu değişikliği yapıp sorgumu tekrar çalıştırdım sonuç aşağıda.

// Guild node'larında guildId'yi id olarak kopyala

```
1 MATCH (n)-[:HOLDS_TOKEN]->(t:Token)-[:IS_TYPE]->(tt:TokenType)
2 RETURN DISTINCT labels(n) AS nodeLabels, n.id AS nodeId, tt.tokenType AS
  tokenType
3 ORDER BY nodeLabels, nodeId, tokenType;
```

	nodeLabels	nodeId	tokenType
10	["Guild"]	14	"Utility Token"
11	["Guild"]	15	"Utility Token"
12	["Guild", "Creator"]	4	"Governance Tok en"
13	["Guild", "Creator"]	7	"Blockchain Tok en"

6.Aynı kalan üyeleri birlikte mi hareket ediyor?

```
1 MATCH (leader:User)-[:MEMBER_OF {role: "Leader"}]->(g:Guild)-[:MEMBER_OF]->(member:User),
2   (leader)-[:VOTED_FOR]->(p:Proposal),
3   (member)-[:VOTED_FOR]->(p)
4 WHERE leader <> member
5 RETURN g.id AS guild,
6   p.title AS proposal,
7   leader.userId AS leader,
8   collect(DISTINCT member.userId) AS alignedMembers
9
```

No changes, no records

Bır önceki projemde guildlerin leaderını yazmamıştım burada member of da leader role olanları alabildim. Ve bu şekilde leader olanları oy verdiği tekliflerle memberların oy verdiği teklifleri ortak varsa getir dedim . Bu csv dosyamızda bu şekilde bır kartelleşme gözlenmemiş oldu.