



# Dota2Analysis



Felipe Pacheco Manoel - 215347  
Hugo Carvalho de Almeida Navarro - 198893  
Matheus Augusto da Silva Cândido - 241640

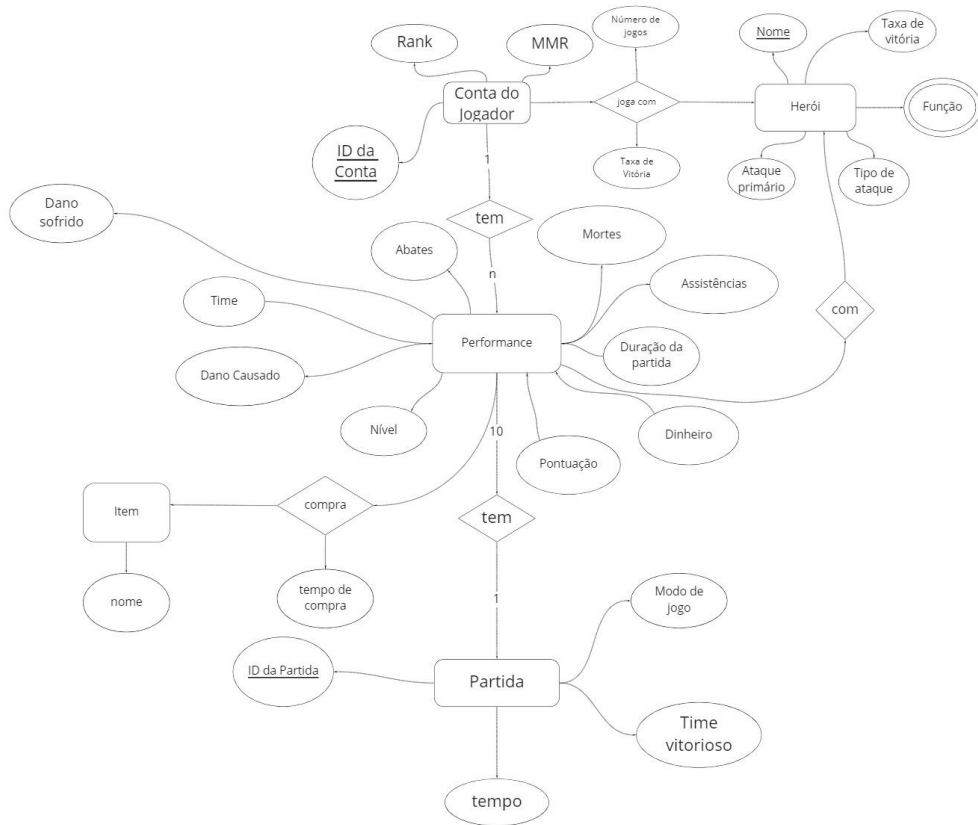


# Tema, motivação e contexto gerador

---

- Criação de um dataset baseado em informações retiradas diretamente do jogo DOTA 2, e posterior análise desse dataset.
- Crescimento dos eSports
- Busca de ferramentas de análise aprofundada
- DOTA 2 possui muita informação de qualidade pública

# Modelo Conceitual



# Modelos Lógicos

---

- A api utiliza um modelo hierárquico
- Utilização de um modelo relacional
- Utilização de um modelo de grafos

# Modelo Relacional

---

- Arquivo csv de partidas públicas, com quase 5 mil partidas públicas

Partidas(Match\_ID,Average\_MMR,Player\_1\_ID,Player\_2\_ID,Player\_3\_ID,Player\_4\_ID,Player\_5\_ID,Player\_6\_ID,Player\_7\_ID,Player\_8\_ID,Player\_9\_ID,Player\_10\_ID,Radiant\_Win,First\_Blood\_Time,Duration)

- Arquivo csv de jogadores, com pouco mais de 2700 jogadores

Jogador(Player ID,MMR, Rank,Win,Lose)

Os jogadores são adicionados a partir da análise das partidas registradas no nosso data set.

A partir de Jogador, calculamos a taxa de vitórias

# Modelo Relacional

---

- Herói(HERO\_ID,NOME,ATRIBUTO\_PRIMARIO,TIPO\_ATAQUE,WINRATE\_PORCENTAGEM)
- Heroi\_Role(nome,funcao)
- Performance(Match\_ID,Player\_ID,Radiant\_Team,Level,Hero\_ID,Hero\_Damage,Hero\_Healing,Damage\_Taken,Tower\_Damage,Kills,Assists,Deaths,KDA,Gold\_Per\_Minute,Gold\_Spent,Experience\_Per\_Minute,Last\_Hits,Item\_0,Item\_1,Item\_2,Item\_3,Item\_4,Win)

A partir de Performance e Herói calculamos a taxa de vitória de um herói.

Pergunta: Determinar a taxa de vitória de um determinado herói.

# Modelo Relacional

---

```
CREATE VIEW HERO_WIN AS  
SELECT HERO_ID,WIN FROM PERFORMANCE
```

```
DROP TABLE IF EXISTS HERO_WIN_LOSE;  
CREATE VIEW HERO_WIN_LOSE AS  
  SELECT HERO_ID,SUM(WIN) AS WINS, COUNT(*) AS MATCHES FROM HERO_WIN  
  GROUP BY (HERO_ID)
```

```
DROP TABLE IF EXISTS HERO_WINRATE;  
CREATE VIEW HERO_WINRATE AS  
  
  SELECT HERO_ID,100*WINS/MATCHES AS WINRATE_PORCENTAGEM FROM HERO_WIN_LOSE;
```

```
SELECT W.HERO_ID, H.NOME, H.ATRIBUTO_PRIMARIO, H.TIPO_ATAQUE, W.WINRATE_PORCENTAGEM  
FROM HEROI H, HERO_WINRATE W  
  WHERE H.ID = W.HERO_ID
```

1	HERO_ID	NOME	ATRIBUTO_PRIMARIO	TIPO_ATAQUE	WINRATE_PORCENTAGEM
2	1	Anti-Mage	agi	Melee	48
3	2	Axe	str	Melee	69
4	3	Bane	int	Ranged	51
5	4	Bloodseeker	agi	Melee	55
6	5	Crystal Maiden	int	Ranged	50
7	6	Drow Ranger	agi	Ranged	74
8	7	Earthshaker	str	Melee	52
9	8	Juggernaut	agi	Melee	58
10	9	Mirana	agi	Ranged	53
11	10	Morphling	agi	Ranged	50
12	11	Shadow Fiend	agi	Ranged	34
13	12	Phantom Lancer	agi	Melee	55
14	13	Puck	int	Ranged	58
15	14	Pudge	str	Melee	51
16	15	Razor	agi	Ranged	44



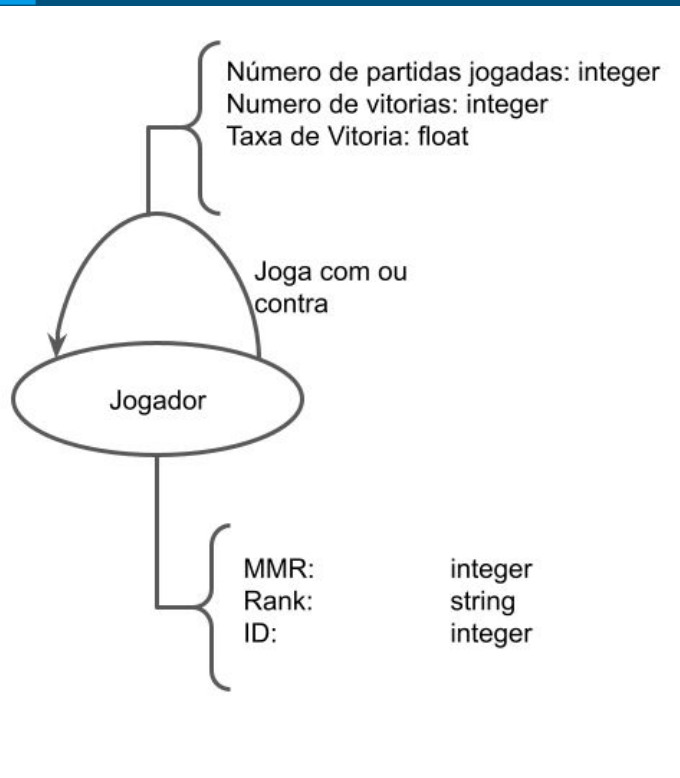
# Modelo Relacional

---

Outras queries:

- Qual herói de uma certa função tem a maior winrate?
- Qual item tem a maior winrate dado um determinado herói?
- Qual a taxa de jogadores que saem das partidas antes de seu término?
- Qual é o herói mais popular no geral? E em um rank específico?
- Como se comporta a winrate de jogadores conforme o ouro ou experiencia por minuto aumenta? Esse comportamento muda dependendo do rank?

# Modelo Grafo de Rede



## Tuplas:

Jogador:

Id , Rank ,MMR

Jogador:

partidas, vitorias , taxa

## Cypher:

```
LOAD CSV WITH HEADERS FROM {url do git do csv dos jogadores} AS line
CREATE (:Jogador {id:line.id,rank:line.rank,mmr:line.mmr})
LOAD CSV WITH HEADERS FROM {url do git do csv dos jogadores} AS line
MATCH (j1:Jogador{id:line.jogador1})
MATCH (j2:Jogador{id:line.jogador2})
CREATE (j1)-[:jogou_com]->(j2)
MATCH (j1)-[:jogou_com]->(j2)
RETURN j1,j2
```

# Modelo Grafo de Rede

---

- Como se organizam as pessoas em grupos de amigos dentro do DOTA 2?
  - Será utilizada as arestas com peso entre jogadores para tentar achar grupos em que as arestas ponderadas são mais densas entre si quando comparadas aos demais jogadores. A pergunta se encaixa na modalidade de comunidade/modularidade.

```
CALL gds.louvain.stream('myGraph', { relationshipWeightProperty: 'VEZES' })  
YIELD nodeId, communityId, intermediateCommunityIds  
RETURN gds.util.asNode(nodeId).id AS id, communityId  
ORDER BY communityId ASC
```

id	communityId
"1218009810"	0
"1218442058"	1
"1219311651"	2
"1220783114"	3
"1221755332"	4
"1225153521"	5

# Modelo Grafo de Rede

---

- Existem grupos que tendem a se encontrar mais nas filas ranqueadas? Esses grupos correspondem a pessoas com ranking e MMR similares?
- Será utilizada as arestas entre jogadores para tentar achar grupos em que as arestas são mais densas entre si comparadas aos demais jogadores, em seguida iremos checar as comunidades encontradas para ver se o MMR e o rank são similares. A pergunta se encaixa na modalidade de comunidade/modularidade.

```
CALL gds.louvain.stream('myGraph')  
YIELD nodeId, communityId, intermediateCommunityIds  
RETURN gds.util.asNode(nodeId).id AS id, communityId  
ORDER BY communityId ASC
```

# Modelo Grafo de Rede


	id	communityId	mmr
17	"838956"	26	"3001"
18	"25021817"	26	"3088"
19	"100714921"	26	"4038"
20	"125794069"	26	"2905"
21	"129336429"	26	"3252"
22	"143328344"	26	"3440"

# Modelo Grafo de Rede

---

- Existem jogadores que se destacam em relação à centralidade? Caso existam, o que os diferenciam?
  - Utilizando o algoritmo de Pagerank, vamos tentar encontrar os principais jogadores e em seguida comparar seus atributos com a média para entender o que os torna especiais. A pergunta se encaixa na modalidade de centralidade

```
CALL gds.pageRank.stream('myGraph', {  
  maxIterations: 20,  
  dampingFactor: 0.85  
})  
YIELD nodeId, score  
RETURN gds.util.asNode(nodeId).id AS id, score  
ORDER BY score DESC, id ASC
```



	id	score
1	"34697180"	1.3879479935610355
2	"33988426"	1.1159452249500235
3	"110539477"	1.0196453832030674
4	"113226336"	1.0196453832030674
5	"22393755"	1.0196453832030674
6	"29949218"	1.0196453832030674



# Operações realizadas

---

- Para o nosso dataset usamos uma api pública do jogo Dota 2
- Disponível em: <https://docs.opendota.com/>
- Através de requests na api e leitura de arquivo JSON
  - Base do dataset: GET /publicMatches
  - Análise de partida única: GET /matches/{match\_id}
- Com a utilização de vetores temporarios, verificamos a existencia de uma partida ou jogador no csv e se não existir a partida ou jogador buscada, adicionamos no arquivo csv

# Operações realizadas

---

- A partir das chaves primárias Match\_ID e Player\_ID das tabelas anteriores conseguimos atribuir uma performance de um jogador em uma partida.
- Conseguimos então ter acesso a dados mais palpáveis para análise.
  - Temos acesso a dados de decisões tomadas pelo player e como isso interferiu no seu desempenho
  - Conseguimos, com dados suficientes, extrair decisões que contribuíram para maior taxa de vitórias
- Assim, finalmente podemos tirar insights a partir do tratamento correto de dados.



# Evolução do projeto

# Obrigado!

---