Título: WS - Assignment 1

Autores: Carina Neves 90451, Eduardo Santos 93107, Pedro Bastos 93150

Data: 11/04/2023

Conteúdo

1. Introdução ao tema

Este trabalho foi desenvolvido no âmbito da Unidade Curricular de Web Semântica e visa o desenvolvimento de um sistema de informação baseado na Web.

O tema do sistema a implementar foi bastante debatido. Dada a possibilidade de escolha, o grupo procurou um tema interessante e com bastante informação relevante, sendo que o escolhido foi Fórmula 1, uma vez que se mostrou um tópico com alguma complexidade, bastante promissor e que dava ao grupo flexibilidade a nível de implementação, dada a quantidade de informação que poderia ser usada.

A Fórmula 1 é um desporto de corridas de carros, projetados para atingirem altas velocidades em circuitos fechados. Envolve diversas equipas, normalmente com dois pilotos cada, a competir num Campeonato Mundial anual.

As corridas são geralmente realizadas ao fim-de-semana e incluem várias etapas, sendo as mais importantes:

- Treinos livres: onde os pilotos e as equipas têm a oportunidade de ajustar os carros e testar a pista;
- Qualificação: onde os pilotos competem para definir a ordem de largada na corrida;
- Corrida: os pilotos competem por várias voltas ao redor da pista, tentando ultrapassar seus adversários e chegar à linha de chegada em primeiro lugar.

Nos últimos anos surgiu uma nova etapa, o *Sprint* Qualificatório, corrida com um terço da distância da corrida principal, onde a posição de largada é a definida pela Qualificação e o resultado do mesmo determina a ordem de largada na Corrida principal.

A Fórmula 1 é considerada um dos desportos mais emocionantes e tecnologicamente avançados, onde as equipas estão constantemente a tentar inovar em busca de vantagens competitivas.

2. Dados

Para a realização deste trabalho o grupo tinha ao seu dispor dados de Formula 1, desde 1950 até a atualidade[1] relativos a:

- Equipas;
- Pilotos;
- Circuitos;
- Qualificações;
- Sprints;
- Corridas.

No entanto apenas foram usados dados de 2019 até 2022 referentes a Equipas, Pilotos e Corridas.

Na pasta */datasets* pode encontrar-se o ficheiro *.zip* com o dataset completo bem como os ficheiros filtrados usados para a criação da base de dados, os *.csv*'s. A imagem a baixo, também presente na pasta mencionada, corresponde ao esquema de base de dados implementado, salientando que tal como pedido pelo docente foi utilizado *Resource Descripion Framework*(RDF). Os dados utilizados foram, após processados, guardados no formato **N-Triples**.

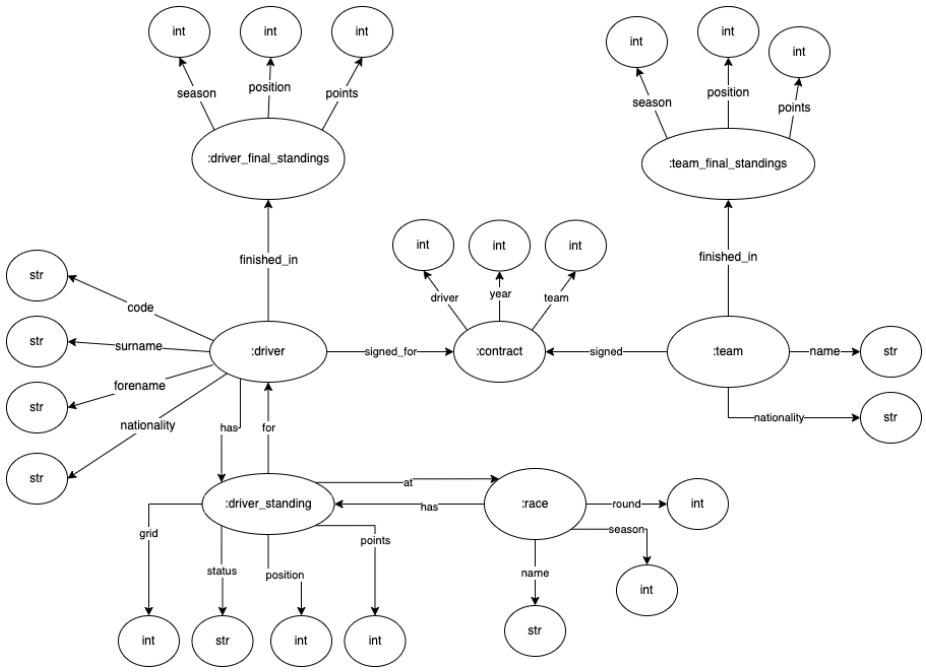

Figura 1: Esquema da base de dados

Um dos principais desafios na projeção deste esquema foi a passagem de uma base de dados relacional, onde as entidades eram representadas pelos diversos *csv's* e as relações pelas colunas presentes nos mesmos, para um modelo de grafo, onde as entidades são representadas por nós e as suas relações por arcos.

Foram definidas então 7 entidades:

- As entidades de *Driver, Team* e *Race*, são por si só explicativas, representam os pilotos, equipas e corridas, respetivamente e têm como objetos de valor literal os atributos que os definem, como por exemplo no caso dos pilotos, o primeiro e último nome, o código por o qual são conhecidos e a nacionalidade.
- O *Driver Final Standings* e *Team Final Standings*, são entidades que têm como objetos literais a posição(*position*) e pontos(*points*) em que o piloto e equipa, respetivamente, terminaram numa dada época(*season*). Estas entidades estão associadas aos pilotos e equipas, pelos predicados de *finished_in*.
- O *Diver Standings* permite a relação de: piloto X - acaba em Y - na corrida Z, pelo predicado *has*, bem como a relação no sentido contrária, pelo predicado *for*, permitindo saber os resultados de todas as corridas.
- O *Contract* permite relacionar o *Driver*, a *Season* e a *Team*, de forma que seja possivel saber por que pilotos uma equipa foi representada em cada ano, recorrendo ao predicado *signed*, bem como para que equipa correu um piloto em cada ano, recorrendo ao predicado *signed_for*.

De forma a passar dos ficheiros *.csv* para o ficheiro *f1.nt*, presente também na pasta */datasets*, foi elaborado o script *processing_data.py*, que pode ser encontrado na pasta */scripts*. Este *script* é composto por duas partes, a primeira para carregar a informação base como: equipas, pilotos, corridas e os diversos

estados em que um piloto pode terminar uma corrida. E a segunda para criar os triplos que serão usados na criação da RDF, com as relações necessárias.

Na fase de carregamento, a informação é guardada em dicionários auxiliares, que serão posteriormente usados na fase de criação. Na imagem a baixo pode ver-se o carregamento da informação relativa aos pilotos, para as restantes entidades o processo foi bastante similar, acedendo apenas ao ficheiro *.csv* correspondente.

```
### Load drivers
drivers_dict = {}
with open('../datasets/drivers.csv', 'r') as file:
    file.readline()
    reader = csv.reader(file)

    for row in reader:
        driver_id = row[0]
        row.pop(0)
        drivers_dict[driver_id] = row
```

Figura 2: Carregamento de informação relativa aos pilotos

Na fase de criação dos tuplos, são associadas às respetivas entidades os devidos predicados e objetos, tal como mostrado na figura abaixo. Após todas as entidades serem processadas, os triplos são escritos no ficheiro *f1.nt*.

```
with open("../datasets/team_final_standings.csv") as file:
    file.readline()
    reader = csv.reader(file)

    for row in reader:
        team_base_rdf = "<{}/team".format(base_rdf)
        team_final_standing_base_rdf = "<{}/team_final_standing".format(base_rdf)

        team_final_standing_id = "{}/id/{}>".format(team_final_standing_base_rdf, row[0])

        team_final_standing_season_pred = "{}/pred/season>".format(team_final_standing_base_rdf)
        team_final_standing_season = "\"{}\"".format(races_dict[row[1]][0])
        team_final_standing_triples.add("{} {} {} .".format(team_final_standing_id, team_final_standing_season_pred, team_final_standing_season))

        team_final_standing_position_pred = "{}/pred/position>".format(team_final_standing_base_rdf)
        team_final_standing_position = "\"{}\"".format(row[4])
        team_final_standing_triples.add("{} {} {} .".format(team_final_standing_id, team_final_standing_position_pred, team_final_standing_position))

        team_final_standing_points_pred = "{}/pred/points>".format(team_final_standing_base_rdf)
        team_final_standing_points = "\"{}\"".format(row[3])
        team_final_standing_triples.add("{} {} {} .".format(team_final_standing_id, team_final_standing_points_pred, team_final_standing_points))

        team_id = "{}/id/{}>".format(team_base_rdf, row[2])
        team_final_standing_pred = "{}/pred/finished_in>".format(team_base_rdf)
        team_triples.add("{} {} {} .".format(team_id, team_final_standing_pred, team_final_standing_id))
```

Figura 3: Criação dos tuplos relativos aos team final standings

3. Operações sobre os dados

Como foi dito na secção anterior, foi utilizado GraphDB dado que a base de dados era de grafos. Foi importado o ficheiro *f1.nt*, que resultou da transformação da base de dados relacional para grafos.

Quanto às queries feitas à base de dados, um dos pontos fulcrais deste projeto, estão todas contidas no diretório *"f1_app/f1_app/queries"*. As queries estão divididas por ficheiros, cada um focado numa parte da BD:

- *standings_queries.py* - queries de standings finais;
- *races_queries.py* - queries de corridas específicas;
- *drivers_queries.py* - queries de pilotos;
- *teams_queries.py* - queries de equipas;
- *curiosities.py* - queries de inferências;
- *admin_queries.py* - queries de operações CRUD de ADMIN.

Todas as chamadas feitas a qualquer um destes ficheiros, são feitas através das **Views** do Django, de maneira à operação ser feita de forma segura:

```
150  def races(request, season):
151      if request.user.is_authenticated:
152          races = races_queries.races_by_season(season)
```

Figura 4: Exemplo de uma chamada feita a uma querie, dentro de uma view.

As queries contêm todo o tipo de operações sobre os dados.

3.1. standings_queries.py

```
239  query = """
240  PREFIX driver: <http://f1/driver/pred/>
241  PREFIX driver_final_standings: <http://f1/driver_final_standing/pred/>
242  PREFIX contract: <http://f1/contract/pred/>
243  PREFIX team: <http://f1/team/pred/>
244
245  SELECT ?driver_code ?forename ?surname ?nationality ?team_name ?position ?points WHERE
246  {
247      ?driver_id driver:code ?driver_code.
248      ?driver_id driver:nationality ?nationality.
249      ?driver_id driver:forename ?forename.
250      ?driver_id driver:surname ?surname.
251
252      ?driver_id driver:finished_in ?dfs.
253      ?dfs driver_final_standings:season ?season.
254      ?dfs driver_final_standings:position ?position.
255      ?dfs driver_final_standings:points ?points.
256
257      ?driver_id driver:signed_for ?contract.
258
259      ?contract contract:year ?year.
260
261      ?team_id team:signed ?contract.
262      ?team_id team:name ?team_name.
263
264      FILTER regex(?season, "SEASON_YEAR" , "i")
265      FILTER regex(?year, "SEASON_YEAR" , "i")
266
267  }
268
269  ORDER BY ASC( xsd:long ( STR(?position) ) )
270
271  """
272
273  query = query.replace("SEASON_YEAR", str(season))
```

Este exemplo retorna a tabela final de classificações para uma certa season, utilizando o **FILTER** para filtrar por ano e o **ORDER BY** para ordenar pela posição. As queries deste ficheiro baseiam-me em classificações por época de pilotos, equipas, número de campeonatos de pilotos/equipas, etc.

3.2. races_queries.py

```
149     query = """
150     PREFIX driver: <http://f1/driver/pred/>
151     PREFIX driver_standing: <http://f1/driver_standing/pred/>
152     PREFIX race: <http://f1/race/pred/>
153     PREFIX contract: <http://f1/contract/pred/>
154     PREFIX team: <http://f1/team/pred/>
155
156     SELECT ?code ?forename ?surname ?nationality ?season ?race_name ?round ?grid ?status ?position ?points ?team_name WHERE
157     {
158         ?driver_id driver:code ?code.
159         ?driver_id driver:forename ?forename.
160         ?driver_id driver:surname ?surname.
161         ?driver_id driver:nationality ?nationality.
162         ?driver_id driver:has ?driver_standing.
163
164         ?driver_standing driver_standing:grid ?grid.
165         ?driver_standing driver_standing:status ?status.
166         ?driver_standing driver_standing:position ?position.
167         ?driver_standing driver_standing:points ?points.
168         ?driver_standing driver_standing:at ?race_id.
169
170         ?race_id race:name ?race_name.
171         ?race_id race:season ?season.
172         ?race_id race:round ?round.
173
174         ?driver_id driver:signed_for ?contract.
175         ?contract contract:year ?season.
176         ?team_id team:signed ?contract.
177         ?team_id team:name ?team_name.
178
179         FILTER (regex(?season, "SEASON_YEAR" , "i") && regex(?race_name, "RACE_NAME", "i"))
180
181     }
182
183     ORDER BY ASC( xsd:long ( STR(?position) ) )
184
185     """
186
187     query = query.replace("SEASON_YEAR", str(season))
188     query = query.replace("RACE_NAME", race_name)
```

Este exemplo retorna a tabela de classificações de uma certa corrida de uma certa season, utilizando, mais uma vez, o **FILTER** para filtrar por corrida e por ano e o **ORDER BY** para ordenar pela posição. As queries deste ficheiro são semelhantes ao do anterior, mas mais específicas às corridas em si.

3.3. drivers_queries.py & teams_queries.py

```
27      query = """
28      PREFIX driver: <http://f1/driver/pred/>
29
30      SELECT ?driver_code ?forename ?surname ?nationality WHERE
31      {
32      {
33      SELECT DISTINCT ?driver_code ?forename ?surname ?nationality
34      WHERE{
35          ?driver_id driver:code ?driver_code.
36          ?driver_id driver:nationality ?nationality.
37          ?driver_id driver:forename ?forename.
38          ?driver_id driver:surname ?surname.
39          FILTER regex(?forename, "DRIVER_NAME" , "i")
40      }
41      }
42      UNION
43      {
44      SELECT DISTINCT ?driver_code ?forename ?surname ?nationality
45      WHERE{
46          ?driver_id driver:code ?driver_code.
47          ?driver_id driver:nationality ?nationality.
48          ?driver_id driver:forename ?forename.
49          ?driver_id driver:surname ?surname.
50          FILTER regex(?surname, "DRIVER_NAME" , "i")
51      }
52      }
53      }
54
55      """
56
57      query = query.replace("DRIVER_NAME", name)
```

Esta query retorna a informação de um piloto dado o seu nome, utilizando **FILTER** para filtrar pelo nome e **UNION** para procurar tanto no nome, como no sobrenome. Este ficheiro tem queries relativas a pilotos, como listar todos, porcurar pilotos por season, as equipas por onde já passou um piloto, etc. O ficheiro das equipas tem queries bastante semelhantes, mas para as equipas.

3.4. admin_queries.py

Este ficheiro contém queries para as operações CRUD do ADMIN, como criar/apagar pilotos, equipas, etc.

3.5. curiosities.py

```
291     query = """
292     PREFIX driver: <http://f1/driver/pred/>
293     PREFIX driver_standing: <http://f1/driver_standing/pred/>
294     PREFIX race: <http://f1/race/pred/>
295 
296     SELECT ?driver_code ?forename ?surname ?nationality (COUNT(DISTINCT ?race_id) as ?count) WHERE
297     {
298         ?driver_id driver:code ?driver_code.
299         ?driver_id driver:forename ?forename.
300         ?driver_id driver:surname ?surname.
301         ?driver_id driver:nationality ?nationality.
302         ?driver_id driver:has ?driver_standing.
303 
304         ?driver_standing driver_standing:grid ?grid.
305         ?driver_standing driver_standing:status ?status.
306         ?driver_standing driver_standing:position ?position.
307         ?driver_standing driver_standing:points ?points.
308         ?driver_standing driver_standing:at ?race_id.
309 
310         ?race_id race:name ?race_name.
311         ?race_id race:season ?season.
312         ?race_id race:round ?round.
313 
314         FILTER (regex(?season, "SEASON_YEAR" , "i"))
315         FILTER (?status != 'Accident' && ?status != 'Finished')
316 
317     }
318 
319     GROUP BY ?driver_code ?forename ?surname ?nationality
320     ORDER BY DESC(?count)
321 
322     """
323 
324     query = query.replace("SEASON_YEAR", str(season))
```

Neste ficheiro estão presentes as queries de inferências. Como podemos ver neste exemplo, aqui é retornado os pilotos com mais carros retirados da corrida, por season. Mais uma vez, é utilizado os **FILTERS**, acompanhados de **COUNT DISTINCT**, **GROUP BY** e **ORDER BY**, para obter uma contagem correta, agrupada por pilotos e em ordem decrescente.

4. Funcionalidades da Aplicação

A aplicação implementada esta dividida em três partes, a área de utilizador comum, de administrador e pública.

Na área de utilizador comum podem ser consultados cinco separadores:

- *Drivers*: onde é possivel visualizar a listagem de todos os pilotos que competiram entre 2019 e 2022, o respetivo nome, nacionalidade, quantos campeonatos ganharam nesse intervalo de tempo e, ao carregar em *Teams History*, as equipas que estes representaram nas diferentes épocas; é ainda possível pesquisar por um determinado piloto.
- *Teams*: onde são listadas todas as equipas presentes nos campeonatos, a sua nacionalidade, o número de campeonatos ganhos e, ao selecionar *Drivers History*, os pilotos que a representaram nas diversas

épocas.

- *Season Results*: onde podem ser consultadas as classificações finais de cada época.
- *Races Results*: onde podemos encontrar informação relativa a todas as corridas realizadas, estando estas agrupadas por época; para cada corrida pode ver-se o seu nome, onde foi realizada, em que ronda aconteceu, e, ao carregar em *Modal for results*, os resultados da corrida em questão.
- *Curiosities*: onde podemos consultar informações relativas às inferencias implementadas:
  - Os pilotos mais frequentes nos pódios, com mais carros retirados e com mais acidentes, por época;
  - Os pilotos considerados experientes;
  - A distribuição das nacionalidades dos pilotos;
  - Os melhores pilotos e equipas.

Na área de administrador, é possivel visualizar todos os separadores descritos anteriormente. No entanto, existe uma nova secção, que permite criar e eliminar pilotos/equipas.

Na área pública, é apenas possível consultar o separador das curiosidades.

De forma a facilitar a perceção das diferentes funcionalidades, foi gravado um curto video de demonstração que pode ser consultado em `https://drive.google.com/file/d/1MIYf3hqjhHpNv4Rl7NKHm4cQIVXQ7yZS/view`.

5. Como executar a aplicação

A aplicação pode ser executada seguindo os seguintes passos:

1. Instalar os requirements presentes no ficheiro *requirements.txt*.
2. Inicializar o GraphDB, criar um novo repositório *"db"* e importar o ficheiro *f1.nt* presente na pasta */datasets*.
3. No diretório */f1_app*, executar o seguinte comando:

```
python manage.py migrate
```

4. Por fim, para iniciar a app, executar o seguinte comando:

```
python3 manage.py runserver
```

Caso se pretenda criar um *user admin*, deve-se executar o seguinte comando:

```
python manage.py createsuperuser
```

6. Ideias de trabalho futuro

Como mencionado no inicio deste relatório, o tema escolhido é bastante abrangente e o *dataset* possuía uma vasta quantidade de dados. Como trabalho futuro, os seguintes pontos foram destacados:

- Incluir mais anos nas opções já implementadas;

- Tratar e permitir a visualização de informação referente às corridas de Qualificação e Sprint;
- Inferir as melhores pistas de um dado piloto;
- Inferir os pilotos com mais trocas de equipa, e quais os pares de pilotos mais duradouros;
- Permitir a comparação direta de pilotos e equipas;

7. Conclusão

Concluindo, este trabalho mostrou-se bastante proveitoso para consolidar os conhecimentos adquiros na Unidade Curricular, relativos à utilização de *Django*, *Resource Description Framework*(RDF), *GraphDB* e *SPARQL*. E, uma vez que o tema escolhido era do interesse geral do grupo, o trabalho tornou-se mais apelativo e fumentou o debate em equipa.

Os principais desafios foram o mapeamento da base de dados relacional encontrada para *GraphDB* e a utilização do *SPARQL*.

Referências

[1] VOPANI. *Formula 1 World Championship (1950 - 2023)*. Dataset Utilizado. URL: https://www.kaggle.com/datasets/rohanrao/formula-1-world-championship-1950-2020?resource=download.