**Relatório Trabalho Prático**

Football Data from Transfermarkt

14829, José Pedro Gomes

15708, Pedro Carvalho

15709, José Carreira

Barcelos, dezembro 2023

Índice

[1. Introdução 3](#_Toc155732241)

[1.1. Objetivos do Projeto 3](#_Toc155732242)

[1.2. Lista de recursos da Plataforma 4](#_Toc155732243)

[1.3. Volume de dados Plataforma 5](#_Toc155732244)

[2. Modelo de dados relacional — *SQL Server* 6](#_Toc155732245)

[2.1. Modelo de dados 6](#_Toc155732246)

[2.2. Importação de dados 8](#_Toc155732247)

[2.3. Ajustes e testes ao Modelo de dados 12](#_Toc155732248)

[2.4. Operações de manipulação de dados 17](#_Toc155732249)

[2.4.1. INSERT 17](#_Toc155732250)

[2.4.2. UPDATE 18](#_Toc155732251)

[2.4.3. DELETE 19](#_Toc155732252)

[3. Modelo de dados – MongoDB 22](#_Toc155732253)

[3.1. Modelo de Dados 22](#_Toc155732254)

[3.1.1 Alteração de abordagem 23](#_Toc155732255)

[3.2. Importação de dados 24](#_Toc155732256)

[3.2.1. Novo modelo de dados 24](#_Toc155732257)

[3.1. Ajustes e testes ao Modelo de dados 25](#_Toc155732258)

[3.2. Ajustes e testes ao Modelo de dados 28](#_Toc155732259)

[3.3. Operações de manipulação de dados 32](#_Toc155732260)

[3.3.1. INSERT 32](#_Toc155732261)

[3.3.2. UPDATE 33](#_Toc155732262)

[3.3.3. DELETE 34](#_Toc155732263)

[4. Graph-based data model — Neo4J 35](#_Toc155732264)

[4.1. Data model design 35](#_Toc155732265)

[4.2. Data model tuning and testing 35](#_Toc155732266)

[5. Discussion and conclusion 35](#_Toc155732267)

[References 35](#_Toc155732268)

[Contributions 35](#_Toc155732269)

[Annexes 37](#_Toc155732270)

[Annex 01 – <<Nome do Annex 01>> 37](#_Toc155732271)

[Annex 02 – <<Nome do Annex 02>> 37](#_Toc155732272)

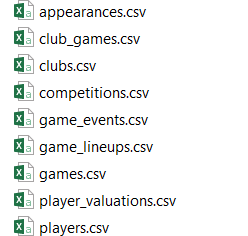
# Introdução

Este relatório é apresentado no âmbito do trabalho prático da unidade curricular de Bases de Dados Avançadas, do Mestrado de Engenharia Informática, lecionada pelo Professor Joaquim Silva. Esta UC tem como objetivo criar competências para o desenvolvimento e otimização de sistemas de armazenamento de dados relacionais e não relacionais, assim com análise de *performance* e persistência de dados. Também se pretende aumentar o conhecimento em relação a modelos e sistemas de dados orientados a documentos, *key/value* egrafos.

O desafio do trabalho prático consiste em criar bases de dados de tipologias diferentes: relacional, não relacional e grafos, modelar as bases de dados, fazer o *design*, implementar e monitorizar a sua *performance*. Pretende-se descrever o processo de cada base de dados, fazer comparações de resultados, conseguindo perceber as diferenças entre as BDs e quais seriam as melhores formas de otimizar cada uma delas, conseguindo identificar qual seria a melhor escolha para responder a determinadas perguntas.

## Objetivos do Projeto

O tema escolhido pelo grupo foi um dataset (constituído por vários ficheiros CSV) de estatísticas de futebol do Transfermarkt obtido no Kaggle (<https://www.kaggle.com/datasets/davidcariboo/player-scores>) com o nome “Football data from Transfermarkt”, com informação de várias competições nacionais e europeias desde 2012.

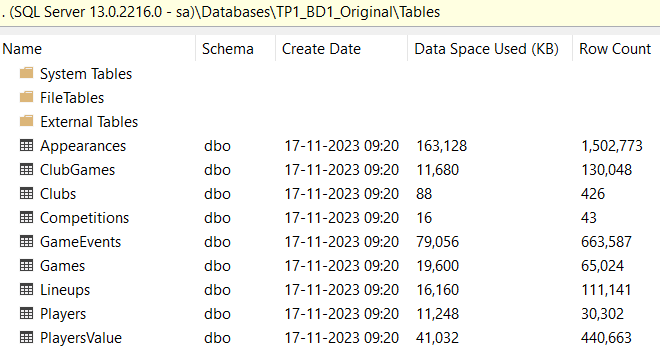


O objetivo passa por integrar todos os ficheiros *CSV,* pois cada ficheiro corresponde a determinada parte da informação, analisar o seu conteúdo e de seguida importar a informação para uma base de dados em *SQL*, para se conseguir ter melhor perceção do potencial, ou eventuais falhas, dos dados. Depois da análise de toda a informação disponível será normalizada a BD, serão validados e importados os dados e será testada a *performance* da BD, este trabalho também será efetuado em *MongoDB* e *Neo4j*.

## Lista de recursos da Plataforma

De seguida será explicada a estrutura de informação disponível, que deverá ser capaz de dar respostas às necessidades de uma plataforma de resultados, com informação detalhada dos jogos, jogadores e equipas.

A estrutura do *dataset* original foi importada para uma base de dados em *SQL* (“TP1\_BD1\_Original”), tendo sido criadas tabelas diretamente a partir de cada ficheiro *CSV* disponível. Como é visível na imagem abaixo, o *dataset* abrange informação variada em relação aos jogos, competições e intervenientes, contém alguma complexidade a relacionar as várias informações disponíveis e possui um volume de registos relativamente elevado, o que deverá permitir as análises pretendidas.



* *Appearances*, detalhe de cada jogador presente no jogo (golos, cartões, etc.);
* *ClubGames,* informação de cada jogo de determinada equipa;
* *Clubs,* informação de cada equipa;
* *Competitions,* competições disponíveis (campeonatos, taças nacionais, competições europeias, etc.);
* *GameEvents,* detalhe deeventos do jogo (golos, substituições, cartões, etc.);
* *Games,* informação de cada jogo;
* *Lineups,* onze inicial e substitutos que entraram no jogo;
* *Players,* informação relacionada com os jogadores;
* *PlayersValue,* detalhe das avaliações de cada jogador.

A partir desta informação pretende-se ser capaz efetuar análises detalhadas de competições, jogos, equipas, jogadores, treinadores, árbitros e estádios. O objetivo do grupo passará por estruturar de forma correta toda esta informação, para se conseguir desenvolver uma base de dados capaz de dar as respostas necessárias, de forma rápida e assertiva.

## Volume de dados Plataforma

O *dataset* depois de ser importado para *SQL* na BD “TP1\_BD1\_Original” permitiu ter a noção da dimensão de cada tabela. A análise da informação disponível permitiu chegar a algumas conclusões em relação aos totais de registos por época (tendo por base os registos disponíveis desde 2012):

* Cada época tem entre 5.5k e 6k registos de jogos;
* Os registos de eventos por jogo variam entre 46k e 73k registos;
* A informação relacionada com os detalhes do jogador em cada jogo varia entre 120k e 145k.

As operações mais frequentes deverão estar relacionadas com informação dos jogos, equipas, jogadores e eventos do jogo. Pelo que uma correta modelação da BD, armazenamento dos dados e a otimização no acesso a estes dados seja fundamental.

# Modelo de dados relacional — *SQL Server*

Depois de se analisar os dados chegou-se à conclusão que havia muita redundância de informação e que seria necessário partir a informação em várias entidades, de forma a garantir a integridade e a normalização dos dados.

Foram identificadas as principais tabelas:

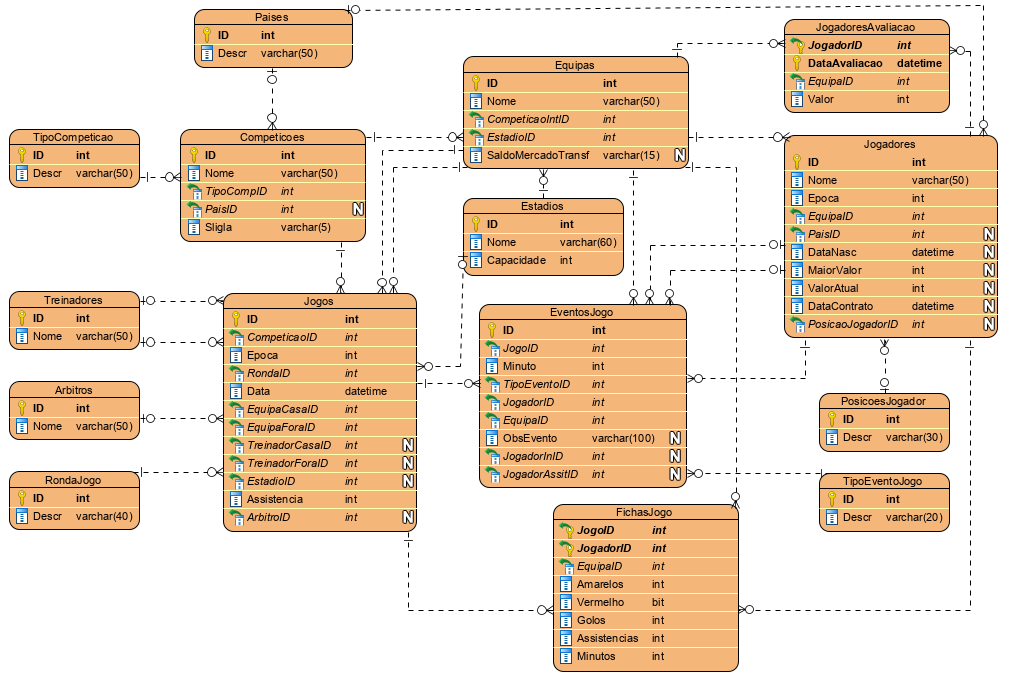
* Competições;
* Equipas;
* Jogos;
* Eventos do Jogo;
* Ficha de Jogo;
* Jogadores;
* Avaliação dos Jogadores;

Como tabelas auxiliares foram identificadas as seguintes tabelas:

* Árbitros;
* Estádios;
* Países;
* Posições de Jogador;
* Ronda/Jornada do Jogo;
* Tipo de Competição;
* Tipo de evento do Jogo.

## Modelo de dados

Foi efetuado o processo de normalização da estrutura do *dataset,* com o objetivo de eliminar redundância e eventuais anomalias do modelo de dados. O seguinte diagrama ER deu origem à base de dados “TP1\_BD1”, estruturada de forma a permitir ao modelo de dados dar as respostas necessárias ao negócio.



Foram definidas chaves primárias em todas as tabelas e algumas chaves estrangeiras. Alguns campos que não são obrigatórios foram definidos como *null*, houve especial atenção na definição do tipo de dados e tamanho dos campos *varchar,* para garantir a integridade e consistência de dados, e controlar o crescimento do tamanho de armazenamento da base de dados.

Em relação aos *IDs* das tabelas Equipas, Jogadores e Jogos, optou-se por continuar a utilizar o valor de *ID* (correspondente à informação em causa) disponível no *dataset* para se conseguir relacionar melhor a informação, em vez dos nomes/descrições, pois foram detetados casos em que o mesmo nome existe mais do que uma vez, aumentando significativamente a margem de erro no cruzamento da informação. Nas restantes tabelas os *IDs* são todos do tipo *int* com *identity increment* de1.

## Importação de dados

Como explicado no capítulo anterior, a informação disponível nos CSV foi importada para a BD “TP1\_BD1\_Original”, esta base de dados serviu como fonte de dados para a base de dados normalizada “TP1\_BD1”. Depois de profunda análise dos dados, identificou-se algumas particularidades nos registos:

* Jogadores com o mesmo nome (usou-se o *playerID* como ID de tabela);
* Mesmo estádio utilizado por mais do que uma equipa, sendo que as equipas mencionam capacidade diferente (usou-se a maior capacidade indicada);
* Estádios que apesar de não terem a capacidade disponível, têm a informação da afluência nos jogos (nesses casos utilizou-se o maior valor);
* Equipas que são mencionadas em jogos, mas não existem na tabela das equipas, provavelmente equipas de escalões inferiores em jogos de taça (estes jogos não foram importados);
* Competições com o mesmo nome em diferentes países (utilizou-se a sigla da competição para efetuar as comparações);
* Nem todos os jogadores têm avaliação disponível;
* Tabela *Lineups* só tem dados disponíveis desde 2021 (excluiu-se os dados desta tabela no modelo de dados final).

De seguida será explicado detalhadamente o processo de popular as tabelas da base de dados “TP1\_BD1”.

As primeiras tabelas a serem populadas logicamente foram as auxiliares:

* Paises



* TipoCompeticao



* Estadios



* Treinadores



* Arbitros



* RondaJogo



* PosicoesJogador



* TipoEventoJogo



*Queries* utilizadas na importação de dados para as tabelas principais:

* Competicao



* Equipas



* Jogadores



* JogadoresAvaliacao



* Jogos



* FichasJogo

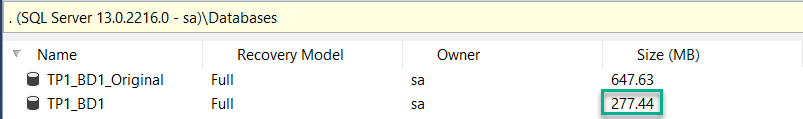


* EventosJogo



O processo de importação de dados ajudou a identificar problemas e oportunidades, daí surgiram pequenos ajustes na estruturação das tabelas: tipos de dados, tamanho máximo definido, campos não obrigatórios e em alguns casos (já explicados anteriormente) deixar de usar ID de auto incremento.

O que foi possível constatar de imediato, e que não foi surpresa para o grupo, foi a diferença de tamanho no armazenamento da informação entre as duas bases de dados, a original (após importação dos *CSV*) e a normalizada:



A base de dados “TP1\_BD1” ficou com 43% do tamanho em comparação com a BD original.

## Ajustes e testes ao Modelo de dados

O modelo de dados desenvolvido deverá ser capaz de dar resposta a uma plataforma de consulta de resultados, detalhes de jogos, equipa, jogadores, etc. daí terem sido testadas algumas *queries* para dar resposta a algumas questões que previsivelmente poderão ser colocadas por utilizadores, desde as questões mais simples às consultas mais criativas ou originais. Algumas das questões possíveis poderão ser:

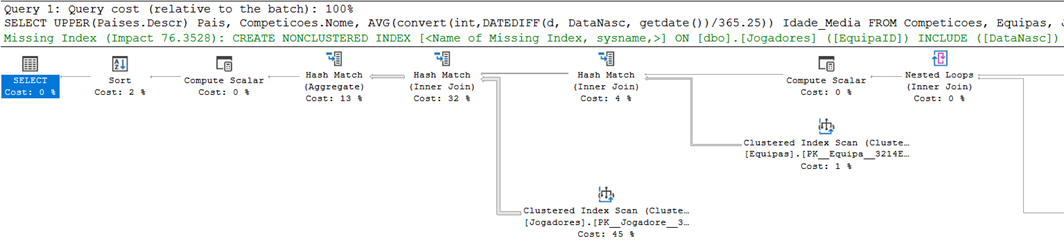
* Qual a idade média de jogadores das várias ligas nacionais? (Apresentar os resultados mais baixos primeiro)



Executando a *query* acima representada obtemos o seguinte resultado:



Ao analisar o *execution plan* desta *query* podemos verificar que indica a necessidade de criação de *index:*



Como sugerido no *execution plan* foi criado um *non clustered index:*

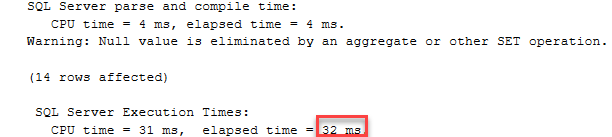


Para se conseguir verificar o tempo de execução da *query* em milissegundos, antes da sintaxe da *query* (para calcular a média de idades) colocou-se o seguinte comando:

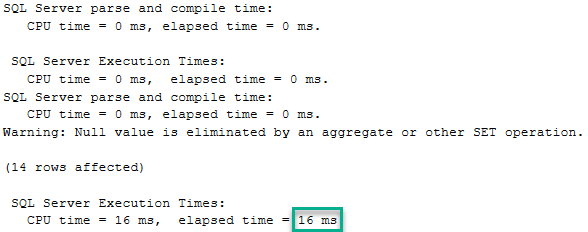


Isso permitiu obter um resultado mais detalhado em relação ao tempo de execução. Podemos abaixo verificar a diferença do tempo de execução das duas queries:

* Resultados de *query* **sem** *index* criado:



* Resultados de *query* **com** *index* criado:



Neste caso concreto a criação do *index* resultou num aumento de *performance* da *query*, pois o tempo de execução diminuiu (passou de 32 para 16 ms).

Outras possíveis questões que este modelo de dados deverá conseguir responder são:

* Da lista de 100 Jogadores com maior valor de mercado qual tem menos presenças em jogos nos últimos 2 anos?





* Por época qual o nº médio de golos marcado em campeonatos nacionais, desde a época 2020/2021?





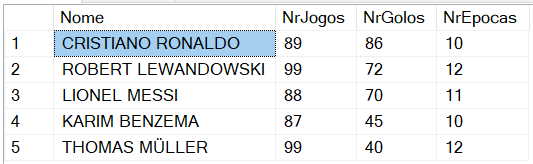
* Qual o arbitro (com pelos menos 10 jogos) que mostrou mais cartões vermelhos na Liga Europa?





* Qual o top 5 de jogadores com mais golos na *Champions* (mas que nunca jogaram o *qualifying* da *Champions*)?





* Qual o jogador que a entrar como substituto tem melhor média de golos (nos jogos em que entrou) por época?





* Qual o jogador que jogou mais jogos ao lado do Cristiano Ronaldo?





* Quais as competições onde a média de substituições por jogo é menor?





## Operações de manipulação de dados

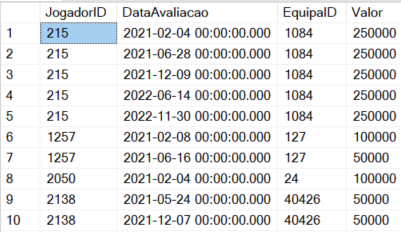
Além dos testes na execução de *queries*, também se testou as operações manipulação de dados: *Insert, Update* e *Delete*.

O desafio assumido pelo grupo foi efetuar estes testes sem afetar os dados disponíveis na base de dados (pois já estão validados e são fidedignos), pelo que se usou uma estratégia que passou por selecionar os registos da tabela “JogadoresAvalicao” as avaliações disponíveis referentes a 2021 e 2022 (a este intervalo de tempo estão alocados 90.149 registos).

Query desenvolvida para identificar os registos em causa:



Resultados obtidos:

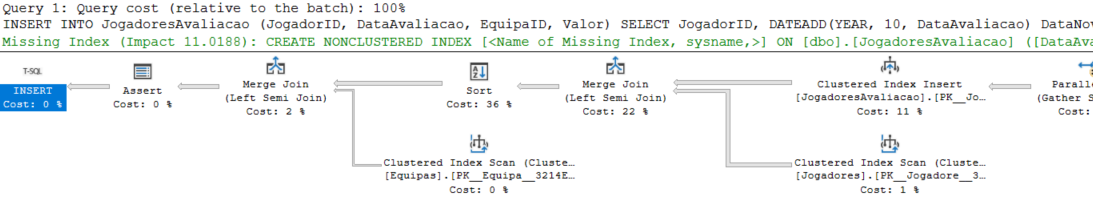


## INSERT

Inserir os registos identificados acima somando 10 anos à data de avaliação do registo, a *query* desenvolvida foi a seguinte:



A análise do *execution plan* da *query* identificou anecessidade de criar um *index*:

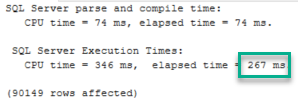


Sintaxe da criação do index:

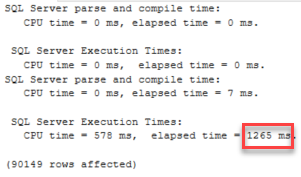


Foram ambas as situações testadas, *INSERT* sem *index* criadoe com *index* criado, de forma a conseguir-se perceber quais as diferenças de *performance*.

Sem *index* criado:



Com *index* criado:



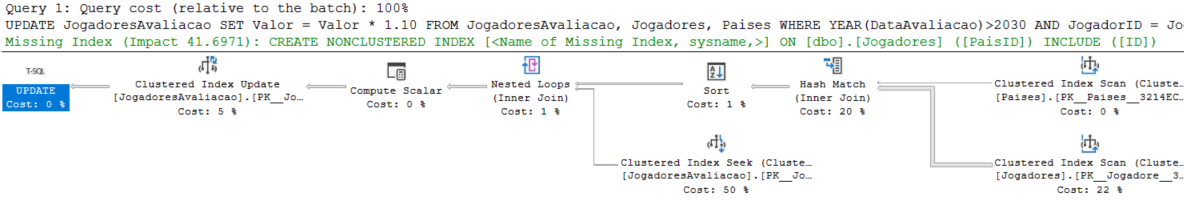
Perante os resultados obtidos pudemos comprovar que a inserção de registos tem tempos de execução diferentes com e sem *index*, como seria de esperar. Esta operação é mais rápida quando a tabela não tem *index* criado, o que já era expectável pois quando as tabelas têm *indexes,* além deinserir os dados, também tem de atualizar os *indexes* da tabela, para garantir a consistência de dados na tabela e em todos os *indexes* relacionados. No final desta operação foi eliminado o *index* para não afetar os restantes testes.

## UPDATE

Utilizando os mesmos registos gerados no ponto anterior (registos inseridos com datas de 2031 e 2032) foi testada a operação *UPDATE*, desta vez cruzando com a tabela dos Jogadores e Países, pois o objetivo será alterar (para os jogadores provenientes de Portugal) o valor da avaliação e aumentá-lo em 10%:



A análise do *execution plan* da *query* identificou anecessidade de criar um *index*:

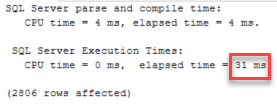


Sintaxe da criação do *index:*

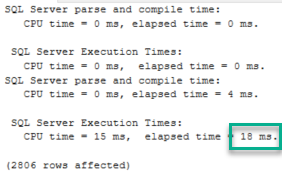


Testou-se novamente a execução da *query* com e sem a criação do *index*:

Sem *index* criado:



Com *index* criado:



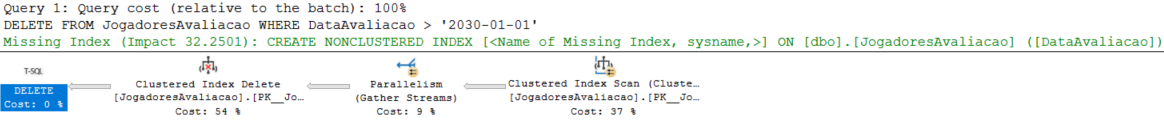
A criação do *index* resultou numa melhoria de performance da *query,* passando de 31 para 18 milissegundos após a criação do *index*. No final desta operação foi eliminado o index para não afetar os restantes testes.

## DELETE

Novamente utilizando os registos obtidos no ponto 2.4.1, desta vez o objetivo será testar a operação *DELETE*, de forma a remover os registos inseridos e posteriormente alterados, para que a tabela volte a ter os dados originais. Sintaxe utilizada para eliminar os registos:



A análise do *execution plan* da *query* identificou anecessidade de criar um *index*:

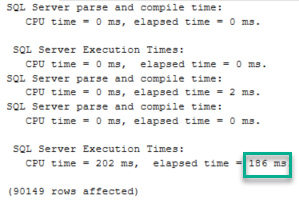


Sintaxe da criação do *index*:

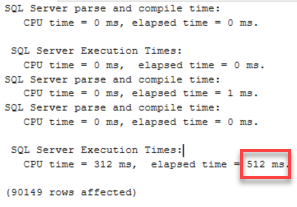


A execução da *query* foi testada com e sem a criação do *index*:

Sem *index* criado:



Com *index* criado:



A obtenção destes resultados permitiu comprovar que a eliminação de registos tem tempos de execução diferentes com e sem *index*, sem *index* tem melhor *performance* do que com.

Enquanto que se a tabela não tiver *index* criado, o SQL apenas procede à eliminação dos registos. Enquanto que se a tabela tiver *index* criado, além da eliminação dos registos também os *indexes* terão de ser atualizados, de forma a garantir a consistência de dados na tabela e em todos os *indexes* relacionados. Isso pode fazer com que a operação seja mais lenta pois existe o trabalho adicional de reorganização de *indexes*.

# Modelo de dados – *Mongo*DB

## Modelo de Dados

Em *MongoDB*, os dados são armazenados em documentos no formato BSON (Binary JSON), que são estruturas de dados semelhantes a JSON. Cada documento contém um id único criado pela BD e pertence a uma *collection*. Uma vez que foram utilizadas várias *collections* alguns ids foram mantidos para ser possível relacionar as diferentes *collections*.

Optou-se por criar as seguintes *collections*:

* Jogos;
* Jogadores;
* EventosJogo;
* FichasJogo;
* JogadoresAvaliacao.

Na imagem seguinte está representada a estrutura do documento para a *collection* Jogos.

Uma imagem com texto, captura de ecrã, software, Página web

Descrição gerada automaticamente

Os campos desta *collection* são:

* \_id: Campo criado automaticamente que referência o documento.
* Competicao: Campo do tipo subdocumento que contem campos com informações da competição (Nome, Tipo, Sigla e Pais).
* Estadio: Campo do tipo subdocumento que que contem campos com informação do estádio (Nome e capacidade).
* JogoId: Campo simples que contem um id de jogo para ser possível relacionar com as outras *collections*.
* Ronda: Campo simples.
* Epoca: Campo simples.
* Arbitro: Campo simples.
* Assistencia: Campo simples.
* Treinadores: Campo do tipo subdocumento que que contem campos com informação dos treinadores (casa e fora).
* “Equipas”: Campo do tipo subdocumento que que contem campos com informação das equipas (casa e fora).

## 3.1.1 Alteração de abordagem

Após todos os testes efetuados nos pontos anteriores o desempenho das *querys* **não**correspondia à nossa expectativa face aos resultados obtidos no modelo relacional. Depois de análise mais detalhada sobre o modo de funcionamento de bases de dados não relacionais, entendeu-se que o modo como os dados estavam relacionados e o tipo de queries utilizadas eram fatores determinantes na performance de bases de dados *NOSQL*.

Sendo o *MongoDB* uma base de dados orientada a documentos decidiu-se alterar o modo como os dados estavam armazenadas e o tipo de consultas realizadas.

Foi decidido reduzir o número de *collections* e adicionar os dados de forma “aninhada” em cada *collection*. A imagem abaixo representada mostra as *collections* que resultaram das operações de tratamento e aninhamento realizadas. Este processo será explicado em mais detalhe na seção 3.2.

Uma imagem com texto, Tipo de letra, captura de ecrã, design

Descrição gerada automaticamente

## Importação de dados

Uma vez que houve tratamento de dados no modelo relacional, foi utilizada a base de dados relacional como fonte de origem para a importação, para que os dados utilizados nas diferentes bases de dados sejam consistentes assim como os resultados obtidos fiáveis.

Foi utilizada a ferramenta *Navicat* para a importação direta, a base de dados ficou com o nome “BDA1\_Original”.

A ferramenta *NaviCat* permitiu configurar várias origens de dados diferentes. Neste caso foi configurada uma ligação ao *SQL* e uma outra ao *MongoDB*. Após estarem as duas ligações configuradas foi utilizada a ferramenta de importação direta de *SQL* para *Mongo*. Esta importação direta transforma cada tabela da base de dados de origem (*SQL*) numa *collection* na base de dados de destino (*MongoDB*).

*Uma imagem com captura de ecrã, texto, software, Software de multimédia

Descrição gerada automaticamente*

## Novo modelo de dados

A nova abordagem teve como objetivo evitar carregar e consumir os dados de forma “*bulk*”, mas fazer algum tratamento aos mesmos quando os importamos para *collection*s.

A *collection* “Competicao” foi carregada a partir do script em anexo ao relatório com o nome “*Competicao.txt*”. Na estrutura de dados foram adicionados dois campos calculados, a média de idade dos jogadores e a média de golos por competição (de modo a reduzir o processamento necessário para obter estes resultados)

A *collection* “Jogos” passou a ter mais dados “aninhados”, foram adicionados os eventos dos jogos assim como as fichas de jogo. O *script* utilizado para a carga dos dados está em anexo com o nome “Jogos.txt”.

A *collection* “Jogadores” passou a conter também dados como equipa, pais e dados de avaliação. O *script* utilizado para carga de dados está em anexo com o nome “Jogadores.txt”.

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

## Ajustes e testes ao Modelo de dados

Para testar a performance da base de dados utilizamos as mesmas queries que foram utilizadas para o modelo relacional como termo de comparação. Foram escolhidas as primeiras 3 questões:

* Qual a idade média de jogadores das várias ligas nacionais? (Apresentar os resultados mais baixos primeiro)

Para responder à pergunta executamos a *query* abaixo (em anexo ao relatório com o nome *Pergunta1\_v2.txt*).

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Após a execução podemos verificar que tempo passou de 42 segundos (abordagem anterior) para 0.26 segundos. Uma diferença **bastante** significativa face à abordagem inicial, o que demonstra novamente que em bases de dados NO*SQL* a maneira que os dados estão organizados tem um impacto profundo na sua velocidade de obtenção.

* Da lista de 100 Jogadores com maior valor de mercado qual tem menos presenças em jogos nos últimos 2 anos?

Uma imagem com captura de ecrã, texto, Software de multimédia, software

Descrição gerada automaticamentePara responder a esta pergunta, foi criada uma tabela com os 100 jogadores mais valiosos que depois foi cruzada com a restante query (em anexo com o nome *Pergunta2\_v2.txt*).

Após executar a query o tempo foi de 4.7 segundos.

Foram criados índices utilizando os seguintes comandos:



Uma imagem com texto, software, Software de multimédia, Software gráfico

Descrição gerada automaticamenteVoltou-se a correr a query e tempo reduziu para 3.4 segundos.

Antes da alteração de abordagem o tempo obtido para responder a esta pergunta era de 6.4 segundos.

* Por época qual o nº médio de golos marcado em campeonatos nacionais, desde a época 2020/2021?

O script para responder a esta pergunta está em anexo com nome “Pergunta3\_v2.txt”, após correr o script, o tempo de execução foi de 0.249 segundos. Também aqui a criação de indexes não melhora a performance.

Uma imagem com texto, captura de ecrã, software, ecrã

Descrição gerada automaticamente

## Operações de manipulação de dados

Além dos testes na execução de *queries*, também se testou as operações manipulação de dados: *Insert, Update* e *Delete* tal como no modelo relacional, de forma a testar a velocidade destas operações.

Para garantir que o volume de dados testado é igual ao modelo relacional foi utilizada uma estratégia que passou por selecionar da tabela “Jogadores” as avaliações disponíveis referentes a 2021 e 2022 (a este intervalo de tempo estão alocados 90.149 registos, o mesmo numero de registos obtidos no modelo relacional).

Uma imagem com software, Software de multimédia, texto, eletrónica

Descrição gerada automaticamenteNa imagens abaixo representadas podemos ver a *query* desenvolvida para obter o volume de dados considerado e os resultados obtidos.

## INSERT

Inserir os registos identificados acima somando 10 anos à data de avaliação do registo, a *query* desenvolvida foi a seguinte:

Uma imagem com texto, software, Software de multimédia, Software gráfico

Descrição gerada automaticamente

O tempo de execução foi de 83 segundos. De seguida foi criado um *índex* para verificar as diferenças de *performance*.

Sintaxe de criação do *index*:

Uma imagem com texto, Tipo de letra, captura de ecrã

Descrição gerada automaticamente

Após a criação do *index* voltou-se a fazer o *INSERT* e desta vez o tempo de execução da tarefa aumentou para 104 segundos. O mesmo comportamento já tinha sido verificado no modelo relacional, o aumento de tempo é devido ao trabalho extra que a base de dados tem após inserir dados de manter o *index* atualizado.

Uma imagem com texto, software, Software de multimédia, ecrã

Descrição gerada automaticamente

No final desta operação foi eliminado o *index* para não afetar os restantes testes.

## UPDATE

Utilizando os mesmos registos gerados no ponto anterior (registos inseridos com datas de 2031 e 2032) foi testada a operação *UPDATE*.O objetivo será alterar (para os jogadores com nacionalidade Portuguesa) o valor da avaliação e aumentá-lo em 10%:

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamenteFoi criada uma *collection* temporária com os dados filtrados já por país. O *script* utilizado para construir a mesma foi o seguinte:

Apos estar criada a *collection* foi executado o *update,* no primeiro caso sem *index* criado, e no segundo com *index* criado. Os tempos de execução são muito idênticos, com vantagem para o *update* com *index.*

Sintaxe de criação de *index*:

Uma imagem com texto, Tipo de letra, captura de ecrã

Descrição gerada automaticamente

Sem *index* criado:

Uma imagem com captura de ecrã, Software de multimédia, software, texto

Descrição gerada automaticamente

Com *index* criado:

Uma imagem com texto, captura de ecrã, Software de multimédia, software

Descrição gerada automaticamente

## DELETE

Novamente utilizando os registos obtidos no ponto 3.4.1, desta vez o objetivo será testar a operação *DELETE* (de forma a remover os registos inseridos e posteriormente alterados) para que a *collection* volte a ter os dados originais. A execução da *query* foi testada com e sem a criação do *índex*.

Uma imagem com texto, Software de multimédia, software, captura de ecrã

Descrição gerada automaticamenteSem *index* criado:

Uma imagem com texto, Software de multimédia, software, captura de ecrã

Descrição gerada automaticamenteCom *index* criado:

A obtenção destes resultados permitiu comprovar que a eliminação de registos tem tempos de execução diferentes com e sem *índex*. Verificou-se que sem *index* é possível obter melhor *performance* do que com *index*.

A operação com *index* criado é mais lenta pois existe o trabalho adicional de manter o mesmo atualizado aquando da eliminação de registos.

# Graph-based data model — Neo4J

<<Include here the tasks performed and its results. >>

## Data model design

<< Identify the entities and their relationships. **Design and** **implement** the Graph-based data model. **Populate the database** with meaningful data. The database should have enough size to draw conclusions about the performance and suitability of the data model.>>

## Data model tuning and testing

<<Test and evaluate some scenarios for the database system. **Execute** operations previously described. **Implement solutions** to improve the operations performance, e. g., the creation of indexes. Results, performance metrics, and the conclusions **should be properly documented**.>>

# Discussion and conclusion

<< Taking into account the results and conclusions, please provide a grounded recommendation for the database model that should be adopted. >>

# References

<< In this section, should be presented in APA format the list of bibliographic sources that were relevant the execution of the work. All the references must be cited on the report.>>

# Contributions

<<Here must be reported the **individual contributions** of each student. For each project task, identify the students that have actively participated.>>

# Annexes

<< The annexes should be used for the inclusion of additional information needed for a better understanding of the report or to complement it. The annexes created should be numbered, start at the top of a new page and be cited in the body text of the report.>>

## Annex 01 – <<Nome do Annex 01>>

## Annex 02 – <<Nome do Annex 02>>

Etc.