**Relatório Trabalho Prático**

Football Data from Transfermarkt

14829, José Pedro Gomes

15708, Pedro Carvalho

15709, José Carreira

Barcelos, dezembro 2023

Índice

[1. Introdução 3](#_Toc151383806)

[1.1. Objetivos do Projeto 3](#_Toc151383807)

[1.2. Lista de recursos da Plataforma 4](#_Toc151383808)

[1.3. Volume de dados Plataforma 5](#_Toc151383809)

[2. Modelo de dados relacional — *SQL Server* 6](#_Toc151383810)

[2.1. Modelo de dados 6](#_Toc151383811)

[2.2. Importação de dados 8](#_Toc151383812)

[2.3. Ajustes e testes ao Modelo de dados 11](#_Toc151383813)

[2.4. Operações de manipulação de dados 14](#_Toc151383814)

[2.4.1. INSERT 15](#_Toc151383815)

[2.4.2. UPDATE 16](#_Toc151383816)

[2.4.3. DELETE 16](#_Toc151383817)

[3. Document-based data model — MongoDB 16](#_Toc151383818)

[3.1. Data model design 16](#_Toc151383819)

[3.2. Data model tuning and testing 16](#_Toc151383820)

[4. Graph-based data model — Neo4J 17](#_Toc151383821)

[4.1. Data model design 17](#_Toc151383822)

[4.2. Data model tuning and testing 17](#_Toc151383823)

[5. Discussion and conclusion 17](#_Toc151383824)

[References 17](#_Toc151383825)

[Contributions 17](#_Toc151383826)

[Annexes 18](#_Toc151383827)

[Annex 01 – <<Nome do Annex 01>> 18](#_Toc151383828)

[Annex 02 – <<Nome do Annex 02>> 18](#_Toc151383829)

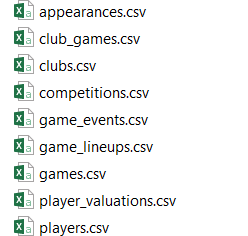
# Introdução

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*](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.

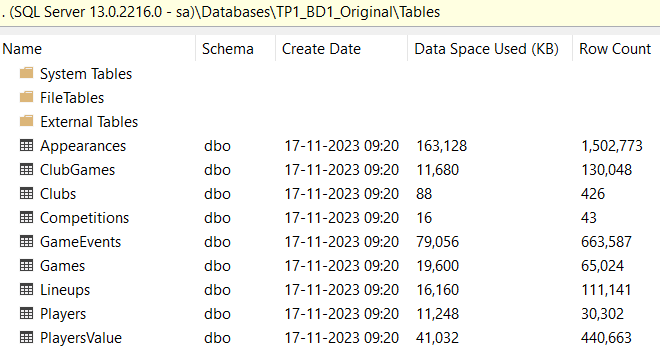


O objetivo passa por integrar todos os ficheiros *CSV,* pois cada ficheiro corresponde a determinada parte da informação, analisar o seu conteúdo e de seguida importar a informação para uma base de dados em *SQL*, para se conseguir ter melhor perceção do potencial, ou eventuais falhas, dos dados. Depois 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 de seguida 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” permite ter a noção da dimensão de cada tabela. Depois de se analisar a informação chegou-se a algumas conclusões em relação aos totais 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:

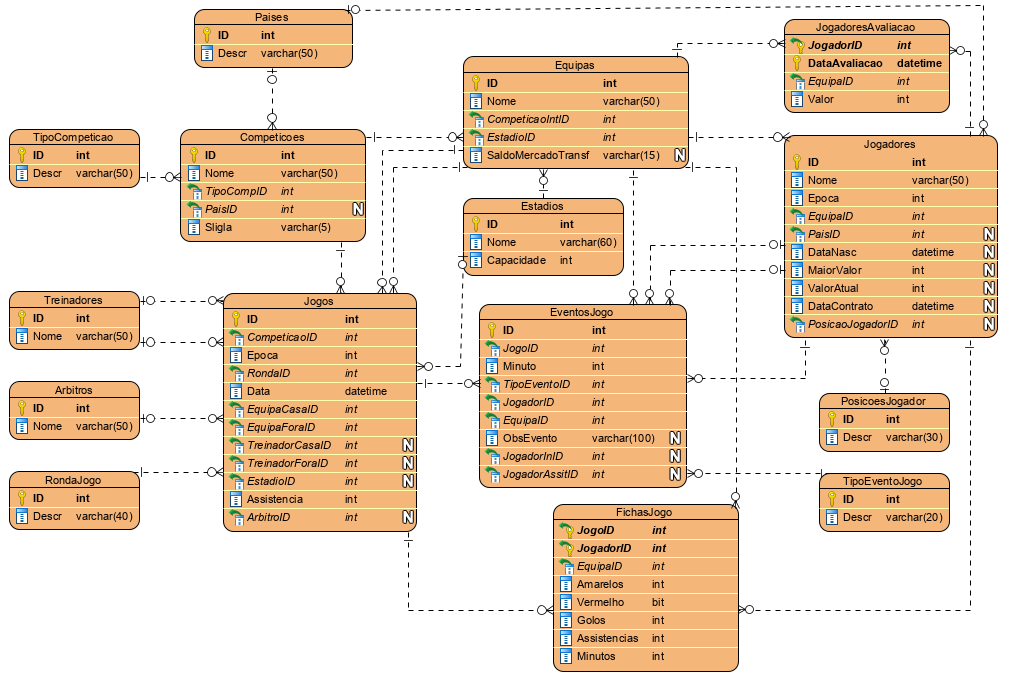
* Competições;
* Equipas;
* Jogos;
* Eventos do Jogo;
* Ficha de Jogo;
* Jogadores;
* Avaliação dos Jogadores;

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 *ID*s 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 existiam 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 nossa 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, tinham 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 que permitiu popular as tabelas da base de dados “TP1\_BD1”.

As primeiras tabelas a serem populadas logicamente foram as auxiliares:

* Países

INSERT INTO TP1\_BD1.dbo.Paises (Descr)

SELECT DISTINCT Country\_name

FROM Competitions

WHERE country name IS NOT NULL

UNION

SELECT DISTINCT country\_of\_citizenship

FROM Players

WHERE country\_of\_citizenship IS NOT NULL

* TipoCompeticao

INSERT INTO TP1\_BD1.dbo.TipoCompeticao (Descr)

SELECT DISTINCT sub\_type

FROM Competitions

* Estadios

--Popular tabela "Estadios" (clubes a mencionar o mesmo estádio com diferente capacidade - Génova)

INSERT INTO TP1\_BD1.dbo.Estadios (Nome, Capacidade)

SELECT UPPER(stadium\_name), MAX(stadium\_seats) Lugares

FROM Clubs

GROUP BY stadium\_name

INSERT INTO TP1\_BD1.dbo.Estadios (Nome, Capacidade)

SELECT UPPER(stadium), MAX(ISNULL(attendance,0)) Lugares

FROM Games

WHERE stadium IS NOT NULL

AND NOT EXISTS (SELECT \* FROM TP1\_BD1.dbo.Estadios WHERE nome = stadium)

GROUP BY stadium

* Treinadores

INSERT INTO TP1\_BD1.dbo.Treinadores (Nome)

SELECT DISTINCT home\_club\_manager\_name Treinador

FROM Games

WHERE home\_club\_manager\_name IS NOT NULL

UNION

SELECT DISTINCT away\_club\_manager\_name Treinador

FROM Games

WHERE away\_club\_manager\_name IS NOT NULL

* Arbitros

INSERT INTO TP1\_BD1.dbo.Arbitros (Nome)

SELECT DISTINCT referee

FROM Games

WHERE referee IS NOT NULL

* RondaJogo

INSERT INTO TP1\_BD1.dbo.RondaJogo (Descr)

SELECT DISTINCT round

FROM Games

WHERE round IS NOT NULL

* PosicoesJogador

INSERT INTO TP1\_BD1.dbo.PosicoesJogador (Descr)

SELECT DISTINCT sub\_position

FROM Players

WHERE sub\_position IS NOT NULL

* TipoEventoJogo

INSERT INTO TP1\_BD1.dbo.TipoEventoJogo (Descr)

SELECT DISTINCT type

FROM GameEvents

WHERE type IS NOT NULL

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

* Competicao

INSERT INTO TP1\_BD1.dbo.Competicoes (Nome, TipoCompID, PaisID, Sigla)

SELECT DISTINCT UPPER(name), TipoCompeticao.ID, Paises.ID, competition\_id

FROM TP1\_BD1.dbo.TipoCompeticao, Competitions

LEFT JOIN TP1\_BD1.dbo.Paises ON country\_name = Paises.Descr

WHERE TipoCompeticao.Descr = sub\_type

* Equipas

INSERT INTO TP1\_BD1.dbo.Equipas (ID, Nome, CompeticaoIntID, EstadioID, SaldoMercadoTransf)

SELECT DISTINCT club\_id, UPPER(Clubs.name), Competicoes.ID, Estadios.ID, net\_transfer\_record

FROM Clubs, Competitions, TP1\_BD1.dbo.Competicoes, TP1\_BD1.dbo.Estadios

WHERE domestic\_competition\_id = competition\_id AND Estadios.Nome = stadium\_name

AND Competitions.competition\_id = Sigla

* Jogadores

INSERT INTO TP1\_BD1.dbo.Jogadores (ID, Nome, Epoca, EquipaID, PaisID, DataNasc, MaiorValor, ValorAtual, DataContrato, PosicaoJogadorID)

SELECT player\_id, UPPER(Players.name), Players.last\_season, Equipas.ID, Paises.ID, date\_of\_birth, highest\_market\_value\_in\_eur, market\_value\_in\_eur, contract\_expiration\_date, PosicoesJogador.ID

FROM Players

LEFT JOIN Clubs ON club\_id = current\_club\_id

LEFT JOIN TP1\_BD1.dbo.Equipas ON Clubs.club\_id = Equipas.ID

LEFT JOIN TP1\_BD1.dbo.Paises ON Paises.Descr = country\_of\_citizenship

LEFT JOIN TP1\_BD1.dbo.PosicoesJogador ON PosicoesJogador.Descr = sub\_position

* JogadoresAvaliacao

--Popular tabela "JogadoresAvaliacao" (há ID jogador com avaliação que não existe na tabela Jogadores)

INSERT INTO TP1\_BD1.dbo.JogadoresAvaliacao (JogadorID, DataAvaliacao, EquipaID, Valor)

SELECT Jogadores.ID, datetime, Equipas.ID, PlayersValue.market\_value\_in\_eur

FROM PlayersValue

LEFT JOIN Players ON Players.player\_id = PlayersValue.player\_id

LEFT JOIN TP1\_BD1.dbo.Equipas ON Players.current\_club\_id = Equipas.ID

LEFT JOIN TP1\_BD1.dbo.Jogadores ON Jogadores.ID = Players.player\_id

WHERE Jogadores.ID IS NOT NULL

* Jogos

--Popular tabela "Jogos" (existem jogos cujos equipas não estão na tabela)

INSERT INTO TP1\_BD1.dbo.Jogos (ID, CompeticaoID, Epoca, RondaID, Data, EquipaCasaID, EquipaForaID, TreinadorCasaID, TreinadorForaID, EstadioID, Assistencia, ArbitroID)

SELECT game\_id, Competicoes.ID, season, RondaJogo.ID, date, Casa.ID, Fora.ID, TCasa.ID, TFora.ID, Estadios.ID, ISNULL(attendance,0) Assist, Arbitros.ID

FROM Games

LEFT JOIN TP1\_BD1.dbo.Competicoes ON competition\_id = sigla

LEFT JOIN TP1\_BD1.dbo.RondaJogo ON round = RondaJogo.Descr

LEFT JOIN TP1\_BD1.dbo.Equipas Casa ON home\_club\_id = Casa.ID

LEFT JOIN TP1\_BD1.dbo.Equipas Fora ON away\_club\_id = Fora.ID

LEFT JOIN TP1\_BD1.dbo.Treinadores TCasa ON home\_club\_manager\_name = TCasa.Nome

LEFT JOIN TP1\_BD1.dbo.Treinadores TFora ON away\_club\_manager\_name = TFora.Nome

LEFT JOIN TP1\_BD1.dbo.Estadios ON stadium = Estadios.Nome

LEFT JOIN TP1\_BD1.dbo.Arbitros ON referee = Arbitros.Nome

WHERE Casa.ID IS NOT NULL AND Fora.ID IS NOT NULL

* FichasJogo

--Popular tabela "FichasJogo" (preencher somente para os jogos, equipas e jogadores existentes)

INSERT INTO TP1\_BD1.dbo.FichasJogo (JogoID, JogadorID, EquipaID, Amarelos, Vermelho, Golos, Assistencias, Minutos)

SELECT Appearances.game\_id, Appearances.player\_id, player\_club\_id, yellow\_cards, red\_cards, goals, assists, minutes\_played

FROM Appearances

INNER JOIN Games ON Games.game\_id = Appearances.game\_id

INNER JOIN TP1\_BD1.dbo.Jogos ON Jogos.ID = Appearances.game\_id

INNER JOIN Players ON Players.player\_id = Appearances.player\_id

INNER JOIN Clubs ON Clubs.club\_id = Appearances.player\_club\_id

* EventosJogo

--Popular tabela "EventosJogo" (preencher somente para os jogos e jogadores existentes)

INSERT INTO TP1\_BD1.dbo.EventosJogo (JogoID, Minuto, TipoEventoID, JogadorID, EquipaID, ObsEvento, JogadorInID, JogadorAssitID)

SELECT game\_id, minute, TipoEventoJogo.ID, player\_id, club\_id, description, player\_in\_id, player\_assist\_id

FROM GameEvents

INNER JOIN TP1\_BD1.dbo.TipoEventoJogo ON TipoEventoJogo.Descr = type

INNER JOIN TP1\_BD1.dbo.Jogos ON Jogos.ID = GameEvents.game\_id

INNER JOIN TP1\_BD1.dbo.Jogadores ON Jogadores.ID = GameEvents.player\_id

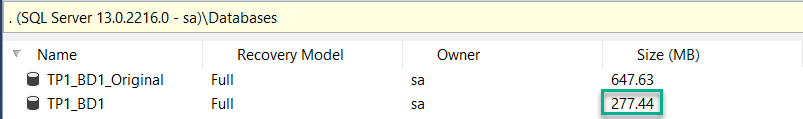
LEFT JOIN TP1\_BD1.dbo.Jogadores JogIN ON JogIN.ID = GameEvents.player\_in\_id

LEFT JOIN TP1\_BD1.dbo.Jogadores JogAssist ON JogAssist.ID = GameEvents.player\_assist\_id

WHERE (GameEvents.player\_in\_id IS NULL OR JogIN.ID IS NOT NULL) AND (GameEvents.player\_assist\_id IS NULL OR JogAssist.ID IS NOT NULL)

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)

SELECT UPPER(Paises.Descr) Pais, Competicoes.Nome, AVG(convert(int,DATEDIFF(d, DataNasc, getdate())/365.25)) Idade\_Media

FROM Competicoes, Equipas, Jogadores, Paises, TipoCompeticao

WHERE Competicoes.ID = CompeticaoIntID AND EquipaID = Equipas.ID AND Competicoes.PaisID = Paises.ID

AND TipoCompeticao.Descr = 'first\_tier' AND TipoCompID = TipoCompeticao.ID

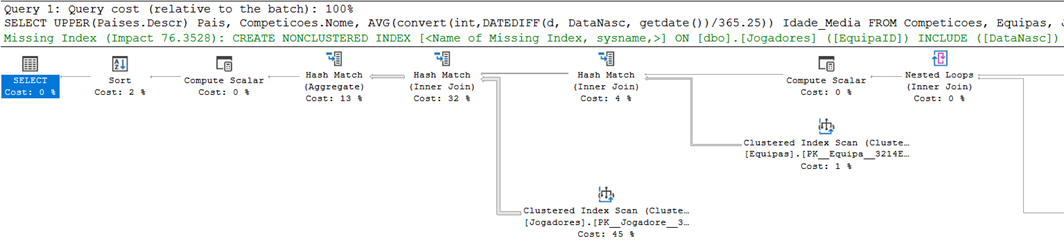
GROUP BY UPPER(Paises.Descr), Competicoes.Nome

ORDER BY Idade\_Media

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:*

USE [TP1\_BD1]

GO

CREATE NONCLUSTERED INDEX [Ind001]

ON [dbo].[Jogadores] ([EquipaID])

INCLUDE ([DataNasc])

GO

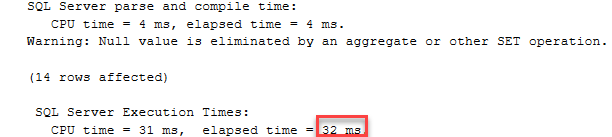
Para se conseguir verificar o tempo de execução da *query* em milissegundos, antes da sintaxe da *query* colocou-se o seguinte comando:

SET STATISTICS TIME ON

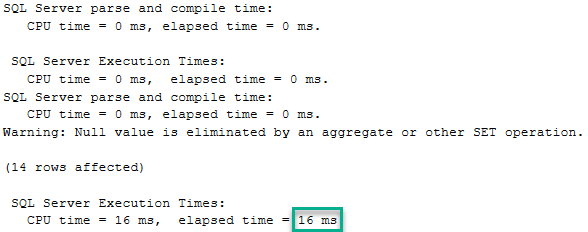
GO

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?
* 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 3 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 media de golos por época?
* Qual o defesa maior média de amarelos (em épocas em que nunca foi substituído) nas taças nacionais?
* Qual o estádio em que a equipa da casa perde mais jogos e tem maior taxa de ocupação?
* Qual o treinador que ganha mais jogos fora (tendo treinado pelo menos 2 equipas num país onde trabalhou)?
* Quais as competições onde a média de substituições por jogo é menor?
* Qual o jogador mais vezes substituído (em 2022/2023) com melhor avaliação antes do primeiro jogo que fez na época 2023/2024?

## 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).

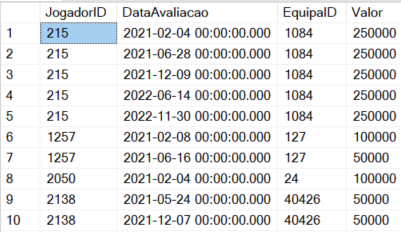
Query desenvolvida para identificar os registos em causa:

SELECT JogadorID, DATEADD(YEAR, 10, DataAvaliacao) DataNova, EquipaID, Valor

FROM JogadoresAvaliacao

WHERE DataAvaliacao BETWEEN '2021-01-01' AND '2022-12-31'

Resultados obtidos:



## INSERT

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

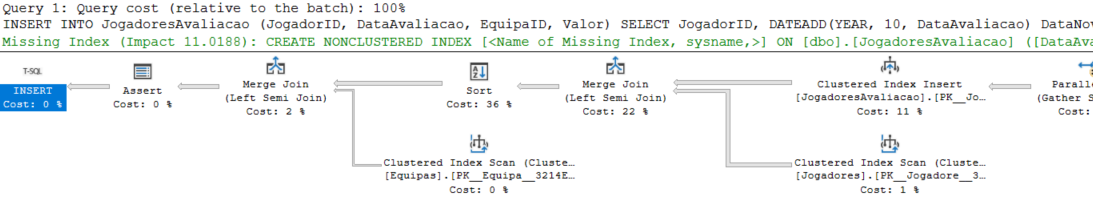
INSERT INTO JogadoresAvaliacao (JogadorID, DataAvaliacao, EquipaID, Valor)

SELECT JogadorID, DATEADD(YEAR, 10, DataAvaliacao) DataNova, EquipaID, Valor

FROM JogadoresAvaliacao

WHERE DataAvaliacao BETWEEN '2021-01-01' AND '2022-12-31'

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



Sintaxe da criação do index:

USE [TP1\_BD1]

GO

CREATE NONCLUSTERED INDEX [Ind001]

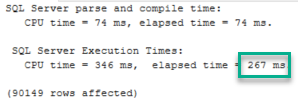
ON [dbo].[JogadoresAvaliacao] ([DataAvaliacao])

INCLUDE ([JogadorID],[EquipaID],[Valor])

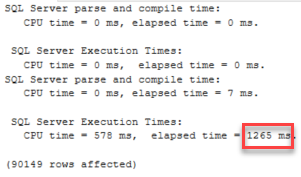
GO

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

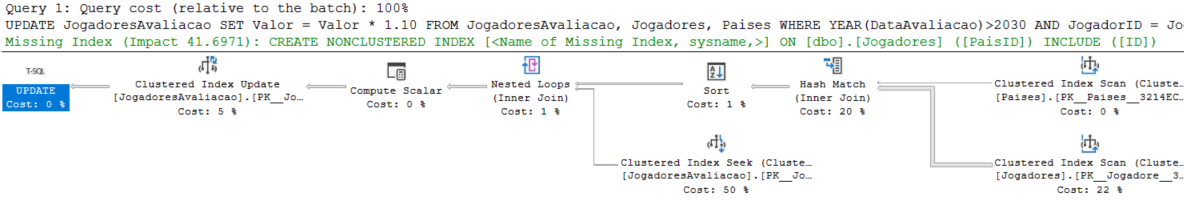
Utilizando os mesmos registos gerados no ponto anterior (registos inseridos com datas de 2031 e 2032) foi testada a operação *UPDATE*, desta vez cruzando com a tabela dos Jogadores e Países, pois o objetivo será alterar (para os jogadores provenientes de Portugal) o valor da avaliação e aumentá-lo em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

UPDATE JogadoresAvaliacao SET Valor = Valor \* 1.10

FROM JogadoresAvaliacao, Jogadores, Paises

WHERE YEAR(DataAvaliacao)>2030 AND JogadorID = Jogadores.ID AND PaisID = Paises.ID AND Descr = 'Portugal'

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



Sintaxe da criação do *index:*

USE [TP1\_BD1]

GO

CREATE NONCLUSTERED INDEX [Ind001]

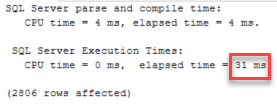
ON [dbo].[Jogadores] ([PaisID])

INCLUDE ([ID])

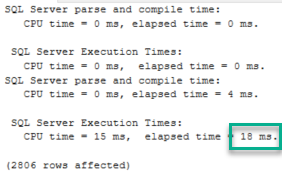
GO

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.

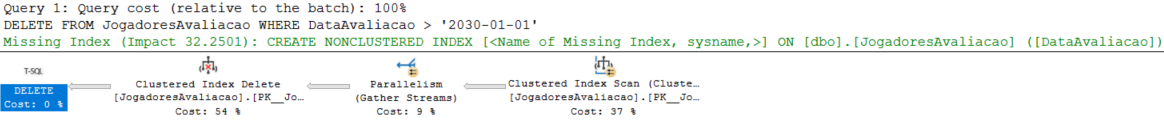
## DELETE

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:

DELETE FROM JogadoresAvaliacao

WHERE DataAvaliacao > '2030-01-01'

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



Sintaxe da criação do *index*:

USE [TP1\_BD1]

GO

CREATE NONCLUSTERED INDEX [Ind002]

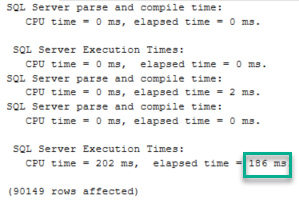
ON [dbo].[JogadoresAvaliacao] ([DataAvaliacao])

INCLUDE ([JogadorID])

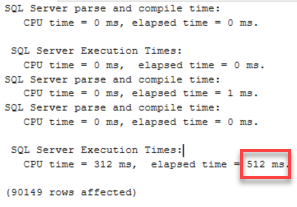
GO

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*.

# Document-based data model — MongoDB

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

## Data model design

<< Identify the entities and their relationships. **Design and** **implement** the data model – Collection-Relationship Diagram. **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**.>>

# 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.