U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Base de Dados

Campeonato Português de Futebol

2LEIC08 – Grupo 801

2022/2023

João Alves	**up202108670**@fe.up.pt
Daniel Bernardo	**up202108667**@fe.up.pt
Tiago Oliveira	**up202009302**@fe.up.pt

Descrição

Este projeto baseia-se na criação de uma base de dados, e todo o processo adjacente, para guardar toda a informação necessária sobre uma época, neste caso da época 2021/2022, da primeira liga de futebol portuguesa. Nesta base de dados tem apenas uma amostra dos dados oficiais da época referida, tendo apenas dados até meio da época desportivo, ou seja, até à décima sétima jornada.

Em primeiro lugar, consideremos a classe **Clube** e as suas interligações. Cada **Clube** está ligado diretamente a todas a outras classes, exceto à classe **Época**. Todos os **Clubes** têm associados **Jogadores**, um **Estádio**, os **Golos** que vão marcando ao longo da época, os **Jogos** realizados e acesso a **Competições** internacionais ou à **Descida** de divisão, para além de terem como atributos o seu nome, a morada e o ano de fundação. A classe **Clube** também apresenta uma ligação à classe **Estádio**, de modo a definir que cada **Clube** terá um **Estádio** associado, onde ir jogar quando for o clube visitado.

Cada **Clube** tem associados diversos **Jogadores**, tal como referido anteriormente. A estes é lhes atribuída uma quantidade considerável de informação, tal como, o nome, a nacionalidade, a posição no terreno de jogo, o seu pé preferencial e a sua altura. Para além disso, a classe Jogador apresenta interligações com a classe **Golo** de forma a atribuir a marcação de um **Golo** a um **Jogador**, e com **Clube**, pois cada Jogador tem de estar devidamente associado a um **Clube**.

De seguida examinemos a classe **Jogo**, que possui como atributos a jornada e a data de um jogo de futebol. **Jogo** apresenta enumeras interligações com outras classes sendo essas: uma interligação com **Golo**, para a base de dados associar os **Golos** marcados em cada **Jogo** às restantes classes, uma ligação dupla a **Clube**, pois cada **Jogo** é sempre disputado por dois clubes, sendo sempre um visitante e um visitado e por último uma ligação a **Estádio** para associar um **Jogo** sempre ao **Estádio** do clube visitado.

Um **Golo** é, evidentemente, sempre associado a um **Jogador**, a um **Clube** para qual o **Golo** seja a favor (para o caso de um golo for na própria baliza, este irá ser associado ao clube adversário) e também será obviamente associado a **Jogo**, de modo a ser posteriormente possível obter os resultados dos **Jogos**.

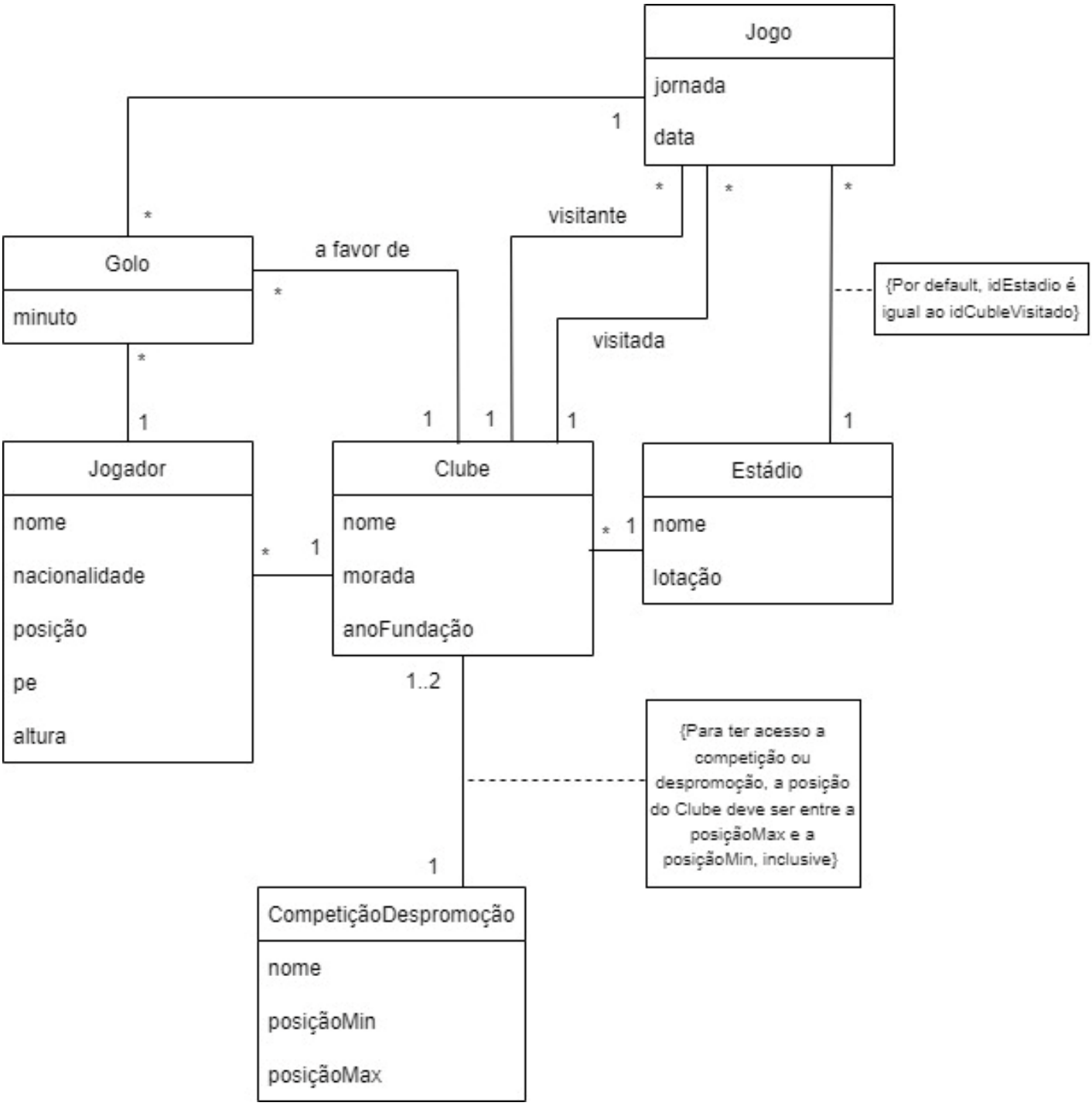
Descrição (continuação)

Olhando agora para a classe **Estádio**, vemos que esta possui como atributos, o nome e a sua lotação, que são as características principais de um estádio. Esta classe apresenta uma interligação com **Clube**, pois cada **Estádio** está sempre associado a um **Clube** e outra interligação com **Jogo**, pois cada jogo da liga tem de estar associado ao **Estádio** da equipa da visitada.

Por último, a classe **CompetiçãoDespromoção**, será a classe que contém o nome de uma competição internacional ou despromoção, a posição mínima e posição máxima na tabela classificativa para obter acesso a esta. Esta classe apresenta uma ligação com **Clube** de modo a associar os clubes às competições ou despromoção.

Em relação à primeira entrega, decidimos retirar a classe **Época** do diagrama de classes UML, aceitando a recomendação do professor avaliador do projeto, visto que a classe **Época** era irrelevante na nossa base de dados, visto que esta só se refere aos dados de somente uma época desportiva (época 2021/2022).

Diagrama de Classes - UML



Para visualização mais nítida, o diagrama UML também está disponível em
https://drive.google.com/file/d/1HSj8Uke920celPJTiGdZbdENDXIH388L/view?usp=share_link

Esquema Relacional

Jogo (idJogo, Jornada, Data, idEstádio -> Estádio, IdClubeVisitado -> Clube, IdClubeVisitante -> Clube)

Golo (idGolo, Minuto, idJogo -> Jogo, idJogador -> Jogador, idClube -> Clube)

Clube (idClube, Nome, Morada, AnoFundação, idEstádio -> Estádio, idCompetiçãoDespromoção -> CompetiçãoDespromoção)

Jogador (idJogador, Nome, Nacionalidade, Posição, Pé, Altura, idClube -> Clube)

Estádio (idEstádio, Nome, Lotação)

CompetiçãoDespromoção (idCompetiçãoDespromoção, Nome, PosiçãoMin, PosiçãoMax)

Dependências Funcionais

Jogo (idJogo, Jornada, Data, idEstádio -> Estádio, IdClubeVisitado -> Clube, IdClubeVisitante -> Clube)

1. {idJogo} -> {Jornada, Data, idEstádio -> Estádio, IdClubeVisitado -> Clube, IdClubeVisitante -> Clube}

Golo (idGolo, Minuto, idJogo -> Jogo, idJogador -> Jogador, idClube -> Clube)

1. { idGolo } -> { Minuto, idJogo -> Jogo, idJogador -> Jogador, idClube -> Clube }

Clube (idClube, Nome, Morada, AnoFundação, idEstádio -> Estádio, idCompetiçãoDespromoção -> CompetiçãoDespromoção)

1. { idClube } -> { Nome, Morada, AnoFundação, idEstádio -> Estádio, idCompetiçãoDespromoção -> CompetiçãoDespromoção }
2. { Nome } -> { idClube, Morada, AnoFundação, idEstádio -> Estádio, idCompetiçãoDespromoção -> CompetiçãoDespromoção }
3. { Morada } -> { idClube, Nome, AnoFundação, idEstádio -> Estádio, idCompetiçãoDespromoção -> CompetiçãoDespromoção }

Jogador (idJogador, Nome, Nacionalidade, Posição, Pé, Altura, idClube -> Clube)

1. { idJogador } -> { Nome, Nacionalidade, Posição, Pé, Altura, idClube -> Clube }

Estádio (idEstádio, Nome, Lotação)

1. { idEstádio } -> { Nome, Lotação }
2. { Nome } -> { idEstádio, Lotação }

CompetiçãoDespromoção (idCompetiçãoDespromoção, Nome, PosiçãoMin, PosiçãoMax)

1. { idCompetiçãoDespromoção } -> { Nome, PosiçãoMin, PosiçãoMax }

Formas Normais

Para concluirmos que uma relação se apresenta na Terceira Forma Normal, é necessário assegurar a regra de não existir transitividade. Se, por algum motivo, esta regra não se mostrar verdadeira para qualquer relação, então esta também não se irá encontrar na forma de Boyce-Codd, pois a forma BCNF é apenas uma versão ligeiramente mais restrita do que a Terceira Forma Normal.

Como se pode verificar, todas as relações presentes no modelo relacional seguem a forma de Boyce-Codd, tal como a Terceira Forma Normal. Analisando as definições das duas formas concluímos que:

Uma relação está na Terceira Forma Normal se para cada S -> A não trivial, S é uma superkey ou uma key da relação ou A consiste em apenas em atributos que são membros, de pelo menos, uma chave da relação;

Uma relação está na forma de Boyce-Codd se para cada S -> A não trivial, S é uma superkey ou key;

Como para todas as relações existentes, a partir da parte da esquerda de todas as dependências funcionais (o conjunto de atributos S explicitado nas definições da formas) se consegue obter todos os atributos presentes em cada relação, então com isto dá para concluir que em todos os casos, o atributo S do exemplo, é sempre ou uma superkey ou uma key da relação, e deste modo é possível justificar que todas as relações se apresentam tanto na forma de Boyce-Codd (ou BCNF) tanto como na Terceira Forma Normal.

Lista e Implementação das Restrições

Para haver um correto funcionamento da base de dados, é necessário fornecer a esta restrições para assegurar mais segurança durante a sua utilização. Deste modo, nesta base de dados recorreu-se à utilização de diversas restrições do tipo chave, do tipo chave estrageira e do tipo NOT NULL, CHECK e UNIQUE.

Começando por explicitar o uso da restrição **CHECK**, esta foi sempre usada com a ideia de restringir os valores que certos atributos poderiam obter, de modo a obter um melhor controlo sobre os dados inseridos na base de dados. A lista inclui:

- Atributo lotação na classe Estádio tem um limite mínimo de 5000 (lotação >= 5000), que é a lotação mínima que um estádio de uma equipa da primeira liga portuguesa deve dispor de modo a poder receber jogos no seu estádio;
- Atributo minuto na classe Golo tem um limite mínimo de 0 (minuto > 0), pois os golos de jogos de futebol são sempre contabilizados no minuto imediatamente a seguir, mal a contagem dos segundos volte a chegar a zero. Ou seja, caso um golo fosse marcado aos dez segundos de jogo, o golo seria contabilizado no minuto um, e nunca no mínimo zero ou inferior.
- Atributo jornada na classe Jogo tem um número no intervalo de valores de 1 até 34 (jornada > 0 e jornada <= 34).

A restrição **UNIQUE** tem como vista garantir que não existe nenhum atributo com igual valor ao mesmo atributo de outro objeto existente na base de dados. Esta foi a aplicada a:

- Atributo nome da classe Clube – não podem existir dois clubes com exatamente ao mesmo nome;
- Atributo morada da classe Clube – a base de dados não permite o registo de dois clubes com exatamente a mesma morada (neste caso, a morada fiscal apresentada no site dos clubes não pode ser igual);
- Atributo nome da classe Estádio – não existem dois estádios com precisamente o mesmo nome.

De seguida, a restrição **NOT NULL** que quando é associada a um atributo, torna-se obrigatória a existência desse atributo para a base de dados conseguir criar um objeto do tipo da classe à qual o atributo com NOT NULL está associado. A lista de uso desta restrição é a seguinte:

- Os atributos idJogo, idJogador e idClube da classe Golo são declarados com NOT NULL, pois sem eles seriam impossível associar um golo ao seu marcador, ao jogo em que aconteceu ou ao clube que favoreceu;
- Os atributos básicos de um Jogador também são todos declarados como NOT NULL, devido ao facto de que não faz sentido esses atributos serem nulos quando se refere a um jogador de futebol profissional em Portugal. Esses atributos são: o nome, a nacionalidade, a posição, o pé preferencial, altura e o idClube do Clube que representam;

Lista e Implementação das Restrições (continuação)

- Os atributos jornada, idÉpoca, idEstádio, idClubeVisitante e idClubeVisitado da classe Jogo também são todos classificados como NOT NULL, pois se estes fossem nulos era impossível definir com precisão quando os jogos tinham acontecido, onde tinham acontecido e quais os intervenientes do jogo;

- Os atributos anoFundação e idEstádio da classe Clube também são restringidos como NOT NULL, pois todos os clubes têm quer ter associados a si um estádio no qual irão jogar os jogos como estatuto de visitado e todos os clubes têm de possuir uma data de fundação;

- O atributo nome, posiçãoMin e posiçãoMax da classe CompetiçãoDespromoção possuem a restrição NOT NULL, pois todas as competições internacionais têm um nome, e também, como é evidente, a despromoção e o playoff de despromoção têm nomes associados, que são estes que referi agora. Para além disso, as competições têm sempre de ter associados os lugares necessários a ser ocupados por um clube na tabela classificativa, de modo a terem o acesso;

Agora veremos as **PRIMARY KEYS**, que foram aplicadas a cada classe, de modo a garantir mais segurança aos utilizadores da base de dados, impedindo a criação de dois objetos como a mesma PRIMARY KEY. A lista é muito simples e trivial, pois para cada classe foi criado um atributo do tipo id<NomeDaClasse>, para servir de chave primária:

- Para as classes Época, Estádio, CompetiçãoDespromoção, Clube, Jogo, Jogador e Golo foi criado um atributo do tipo INT idÉpoca, idEstádio, idCompetiçãoDespromoção, idClube, idJogo, idJogador e idGolo, associados respetivamente às classes, para servir como chave primária.

Por último, relativamente às restrições do tipo **FOREIGN KEY**, aplicamos chaves estrangeiras a classes diretamente relacionadas com outras. Todas estas relações possuem também uma restrição do tipo ON UPDATE CASCADE, para o caso de que se algo mudar numa classe um, por exemplo, a classe dois que possui a chave estrangeira da classe um, também será atualizada com os novos valores da classe um. A lista de utilização é a seguinte:

- Na classe Clube foi usada a FOREIGN KEY idEstádio, de modo, a ser possível associar um Clube ao Estádio no qual joga os jogos disputados no estatuto de visitado.
- Em Jogo foram usadas quatro FOREIGN KEYS: idEstádio, idÉpoca, idClubeVisitado e idClubeVisitante. Desta maneira, conseguimos associar em que Época é que um jogo é disputado, qual o Clube Visitante e qual o Clube Visitado e em que Estádio é que o jogo decorreu.
- Na classe Jogador, foi usada uma FOREIGN KEY, idClube. Assim conseguimos implementar a relação de um jogador tem um contrato com um Clube.
- De seguida, na classe Golo, foram utilizadas três FOREIGN KEYS, sendo estas: idJogo, idJogador e idClube. Assim, é possível ligar um golo sempre ao Jogador que o marcou, em que Jogo é que aconteceu e qual o Clube que beneficiou (pois pode ser um golo normal ou um autogolo).

Interrogações

Aqui apresenta-se a lista de interrogações que se achou pertinente perguntar à base de dados, não só de modo a mostrar o seu correto funcionamento, como também para mostrar um pouco dos dados inseridos no ficheiro povoar.sql.

1. Quantidade de marcadores distintos de cada clube: nesta pergunta são verificadas as interligações entre Jogo, Jogador e Clube, dando uso às funções count, inner join, distinct, group by e order by.

2. Melhores marcadores de equipas que não são consideradas "grandes" no campeonato português (todas as equipas exceto, Porto, Benfica, Sporting e Sporting de Braga): aqui são verificadas as ligações entre Jogo, Jogador e Clube, e são usadas as funções (inner) join, group by, order by, having e not in em conjunto, de modo a simplificar um pouco a pergunta, selecionando todos os clubes que não tenham como idClube 1, 2, 3 ou 4, que são os idClube de Porto, Benfica, Sporting e Sporting de Braga, respetivamente.

3. Media de golos por nacionalidade dos jogadores com altura entre 1,70 e 1,90 metros que são avançados, ou dos jogadores que têm como pé preferencial o esquerdo e que os seus golos tenham sido marcados apenas na segunda parte (minuto > 45): nesta interrogação é usada uma sub-query, em que dentro da qual existe uma união de tabelas usando a função union, de modo a tornar mais explicita a visualização da pergunta; são usadas ainda as funções round, avg, count, natural join, between, like, inner join, group by e order by.

4. Jogos em que ambas equipas marcam e a diferença de golos do jogo tenha sido entre 3 e 7 golos, são ainda mostrados os dados de onde o jogo foi disputado: nesta interrogação à base de dados, é usada uma sub-query de modo a conseguirmos contar a quantidade de golos de cada clube num jogo em que foi disputado entre as duas equipas; aqui são usadas as funções left join, count, abs, between, order by e group by.

5. Resultados dos jogos de uma jornada inteira (primeira jornada neste caso): nesta query mostramos os resultados de apenas uma jornada de modo que o output produzido para pergunta não ser demasiado extenso. Aqui usamos, mais uma vez, uma sub-query para contar os golos de cada equipa; são ainda usadas as funções left outer join e count. Por mais que não seja uma pergunta com uma complexidade muito diferente da pergunta anterior, achamos pertinente mostrar os resultados de uma jornada inteira, que é uma coisa, normalmente muito requisitada quando se trata de um liga de futebol.

6. Tabela classificativa, com posição atual na liga, número de vitórias, derrotas e empates, e também o número de pontos obtidos até ao momento: esta é a nossa pergunta mais complexa à base de dados. Esta apresenta até três sub-queries umas dentro das outras de modo a ser possível contar os golos de cada clube em cada jogo, depois comparar o número de golos de um clube com o outro, de modo a saber se o clube venceu, empatou ou perdeu um jogo e assim então fazer a contagem do número de derrotas, vitórias e empates de cada clube. Após isso, na query principal, é calculado o número de pontos de cada clube e atribuída uma posição na tabela classificativa consoante a ordenação descendente pelo número de pontos. Aqui são usadas bastantes sub-queries, row_number () over, count, left outer join, order by e group by.

Interrogações (continuação)

7. Golos marcados fora e em casa por cada clube: esta é uma interrogação bastante pertinente, pois os golos marcados fora é um dos critérios de desempate entre clubes com os mesmos pontos. Esta pergunta faz uso de uma sub-query e das funções count, inner join, group by e order by.

8. Quantidade de jogos realizados em casa no intervalo de tempo apresentado (em dias) e as caraterísticas do estádio onde foram disputados os jogos: esta interrogação faz uso de uma sub-query, para além de também fazer uso à função de tempo julianday, e uso das funções min, max, count, group by e order by.

9. Clubes com acesso a competições ou à despromoção na presente jornada: esta é uma pergunta muito pertinente à nossa base de dados, mas que irá ser inevitavelmente parecida com a pergunta 6, pois para sabermos os clubes com acesso a competições ou à despromoção, temos de saber o estado da tabela classificativa. Aqui usamos o mesmo método usado na pergunta 6 para contar o número de vitórias e de empates (aqui as derrotas não são desnecessárias) com recurso a sub-queries para obter os resultados dos jogos e então conseguirmos calcular a ordem na tabela classificativa. Após isso testamos a ligações com a classe CompetiçãoDespromoção de modo a vermos se a posição que o clube ocupa, o leva a alguma situação extraordinária e qual o nome dela. Aqui são usadas as funções row_number () over, count, left outer join, group by e order by. Esta query irá produzir o resultado inicial presente na base, caso não seja atualizada com recurso ao nosso trigger gatilho3. Quando utilizado o gatilho3, este vai atualizar a tabela consoante os golos inseridos e irá automaticamente atualizar o resultado produzido por este query.

10. Golos sofridos por cada clube: nesta última interrogação à base de dados é usada uma sub-query dentro de outra sub-query, de modo a ser possível contar os golos ocorridos em todos os jogos em que um certo clube participou, exceto aqueles que foram marcados pelo próprio clube; aqui são usadas também as funções count, left outer join, except, group by e order by.

Gatilhos

Por fim foram definidos três gatilhos que são úteis para haver uma boa monitorização e manutenção da base de dados.

1. **InsertGoloEquipa**

No intuito de inserir um novo golo na base de dados, são necessários os dados de minuto no qual este foi marcado, o id deste, o marcador, o clube que o golo favoreceu e o jogo em que acorreu. Mas até agora nada impedia de ser inserido um golo a favorecer uma equipa que nem sequer era interveniente do jogo. Deste modo, este trigger foi criado de maneira a impedir esta situação.

Deixamos por criar um trigger que verificaria se o golo inserido seria de um jogador presente no jogo, ou seja, que representaria uma das equipas envolvidas no jogo. Tal como pedido no enunciado do trabalho, que diz que caso haja mais do que alguma violação a uma inserção, neste caso, só implementaríamos um dos gatilhos e o outro(s) seriam indicados apenas em relatório.

2. **DeleteJogo**

Após ser eliminado um jogo, caso este não ter sido válido por alguma razão, os golos marcados nesse jogo deixam de fazer qualquer sentido e necessitam de ser eliminados antes do próprio jogo ser eliminado (before delete). Deste modo, ao eliminarmos um jogo este trigger irá ser ativado e irá apagar todos os golos que foram marcados no jogo que se pretende.

3. **updateCompetiçãoDespromoção**

Após ser inserido um golo na base de dados, este trigger vai calcular a posição em que um clube fica na tabela classificativa com este golo. Assim, atualiza (after insert) a competição ou a despromoção do clube beneficiário do golo, ou não, caso o golo inserido não mude o estado em que o clube está. Por exemplo, se o clube não tem acesso a nenhuma competição e com o golo o clube não muda de posição na tabela, a competição associada a este clube não irá mudar, mas, se, no entanto, o golo contribuir para uma subida de posição na tabela classificativa para algum lugar que dá acesso a alguma competição, a competição associada ao clube irá mudar. Obviamente, caso um clube suba de lugar com um golo, e ultrapasse um clube, os clubes ultrapassados podem perder posições na tabela classificativa, e assim obterem também outra competição ou despromoção.

Autoavaliação

Primeira entrega

Opinião Daniel Bernardo:

- Daniel: 33%
- Tiago: 32%
- João: 35%

Opinião Tiago Oliveira:

- Daniel: 35%
- Tiago: 25%
- João: 40%

Opinião João Alves:

- Daniel: 37,5%
- Tiago: 25%
- João: 37,5%

Segunda entrega

Opinião Daniel Bernardo:

- Daniel: 47,5%
- Tiago: 5%
- João: 47,5%

Opinião Tiago Oliveira:

- Daniel: 42,5%
- Tiago: 15%
- João: 42,5%

Opinião João Alves:

- Daniel: 50%
- Tiago: 0%
- João: 50%