UNIVERSIDADE DO VALE DO RIO DOS SINOS - UNISINOS
UNIDADE ACADÊMICA DE GRADUAÇÃO
CURSO DE ENGENHARIA DA COMPUTAÇÃO

FELIPE GRINGS.

ESTUDO DE CASO GA DE BANCO DE DADOS

SÃO LEOPOLDO
2019

RESUMO

Este documento descreve um estudo sobre o funcionamento do projeto de estudo de caso realizado na cadeira de Banco de Dados para Engenharia. Através da utilização de diagrama ER, modelagem SQL, como tabelas, tuplas e visões, utilizando o SGBD MYSQL, assim finalizando com diversos testes para validar as objeções e restrições criadas na modelagem do diagrama ER. O estudo foi realizado sobre estatísticas de futebol, onde o elemento principal é o desempenho do jogador, assim todas as análises sendo relacionadas ao mesmo.

Palavras-chaves: Banco de dados, MYSQL, futebol, diagrama ER, modelagem.

Sumário

1 INTRODUÇÃO

O objetivo deste trabalho prático de modelagem, construção do banco de dados e análise ao desempenho de jogadores de futebol é aprimorar os conhecimento adquiridos sobre modelagem ER, SGBD's e SQL, desenvolvendo, testando e validando nossas soluções propostas para o problema e requisitos apresentados.

Este trabalho, além da aplicação dos conhecimentos adquiridos em aula, também exigiu e incentivou muita pesquisa, em livros, na internet e debates com colegas de curso.

Para realização deste trabalho necessitamos seguir as seguintes especificações.

Parte 1:

Elabore um diagrama ER de um banco de dados para armazenar informações sobre jogos de futebol, onde devem ser armazenados os dados sobre os atletas, times, campeonatos (separados por ano), e dados de desempenho de atleta por partida.

Os dados de desempenho que devem ser armazenados são:
Itens positivos:

• RB – Roubadas de bolas
• G - Gol
• A – Assistência
• SG - Jogos sem sofrer gols
• FS – Falta sofrida
• FF – Finalização para fora
• FD – Finalização defendida
• FT – Finalização na trave
• DD – Defesa difícil
• DP – Defesa de pênalti Itens negativos:
• GC – Gol contra
• CV – Cartão Vermelho
• CA – Cartão Amarelo
• GS – Gol sofrido
• PP – Pênalti perdido
• FC – Falta cometida
• I – Impedimento
• PE - Passes errados

Parte 2:
Monte o script SQL (para MySQL) para a implementação do banco de dados modelado na parte 1.
Esta implementação deve possuir as seguintes visões:
• GolsPorTime: Retorna o nome do atleta, e a quantidade de gols realizados por ele naquele time.

• GolsPorCampeonatoAtleta: Retorna o nome do atleta, o nome do time, e a quantidade de gols realizados pelo atleta neste campeonato.

• GolsPorCampeonatoTime: Retorna o nome do time, o campeonato e a quantidade de gols feitos e gols sofridos, com o saldo de gols.

• ItensPorJogador: Retorna o nome do time, o campeonato, e os itens positivos e negativos do jogador

2 LEVANTAMENTO DE REQUISITOS

Esta etapa trata da primeira parte do projeto que descreve a compreensão e levantamento de todos os requisitos impostos na descrição do projeto.

Requisitos:

- A análise deverá ser feita a partir de cada jogador, levando em consideração a partida jogada por ele.
- O desempenho do jogador é independente do desempenho do time.
- O desempenho do jogador A é independe do desempenho do jogador B.
- Cada jogador deve participar apenas de um time por vez
- Deverá ser mantido o histórico de participação de cada jogador nos respectivos times.
- Uma partida só poderá existir se pertencer a algum campeonato.
- O jogador não necessariamente precisa de um time.
- Caso um campeonato seja excluído automaticamente o histórico de partidas e desempenho dos jogadores participantes será excluído.
- Uma partida sempre deverá conter dois times diferentes.

3 DIAGRAMA ER

3.1 Definição de entidades

A descrição do projeto deixa claro a existência de algumas entidades, porém não é explicito seus atributos e relacionamentos, assim devemos realizar o levantamento de todas as entidades, atributos e relacionamentos existentes.

Começando pelos mais simples temos;

- Jogadores
- Partidas
- Times
- Campeonatos

Com estas quatro entidades abrangemos o necessário, porém para relaciona-lôs necessitamos de mais definições.

Cada jogador pertence a um time, e necessita guardar histórico de seus contratos anteriores com o mesmo time, logo devemos ter um relacionamento ternário onde a chave primária é o id do contrato.

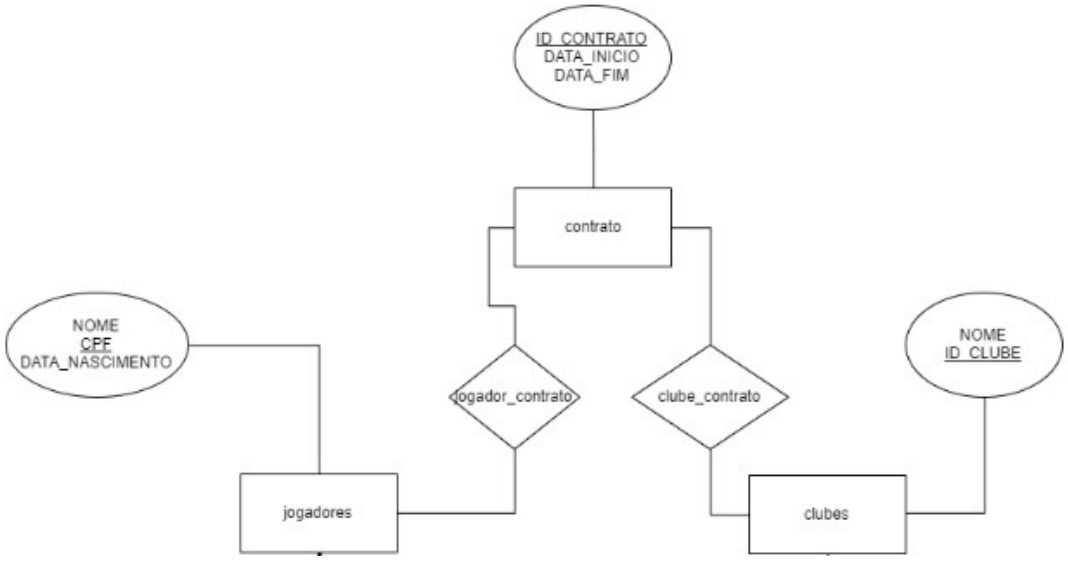


Figura 1: Relacionamento Jogador - Clube

Para obter o desempenho de cada jogador por partida é viável que tenhamos uma entidade especifica no qual é fruto de um relacionamento fraco com jogador e partida.

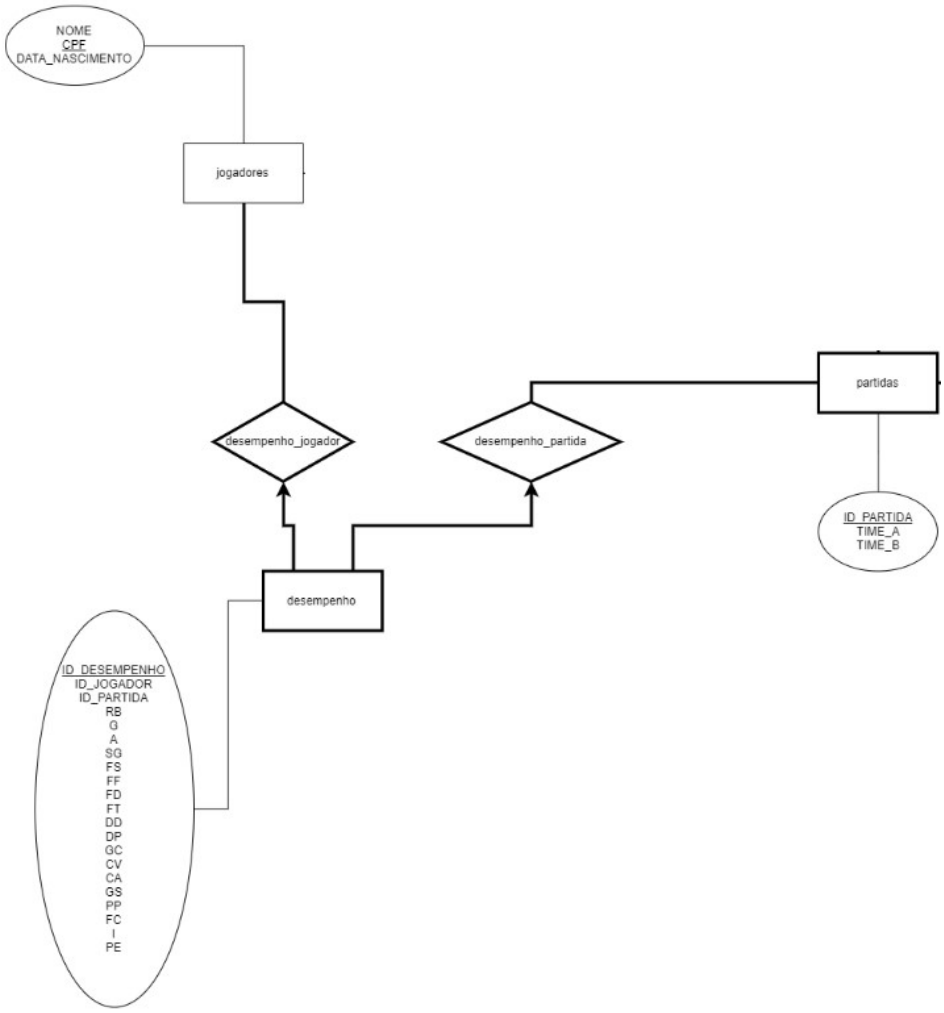


Figura 2: Relacionamento Desempenho - Jogador- Partida

Como só faz sentido existir um desempenho de uma partida existente, caso a partida ou o jogador sejam excluídos o desempenho também será.

Cada partida é executada por dois times diferentes, logo devemos ter também um relacionamento entre partidas e clubes. Partidas também pertencem a um campeonato logo;

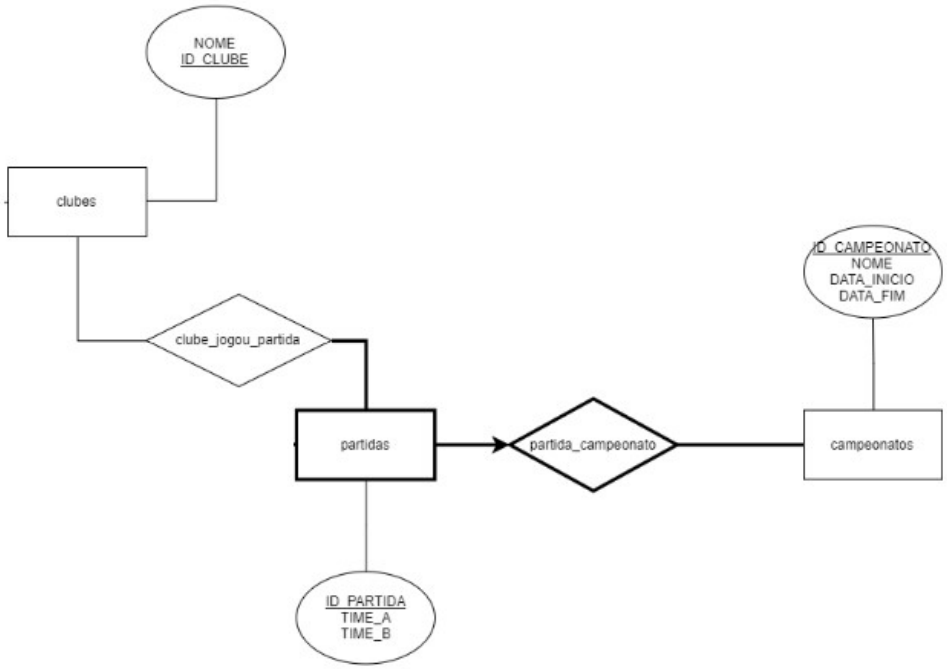


Figura 3: Relacionamento Partidas - Clubes - Campeonatos

Assim como apresentado no anexo 1 a modelagem final do diagrama ER, abrangendo todos os requisitos levantados.

4 MODELAGEM SQL

4.1 Entidades e relacionamentos

Para realizar a modelagem SQL do diagrama proposto acima, podemos desconsiderar as tabelas de relacionamento simples, no qual não apresentam nenhum atributo próprio, assim resumindo o diagramo em apenas 6 tabelas, como descrevo a seguir.

Para tabelas jogadores não foi realizada nenhuma alteração, apenas a implementação dos atributos previamente descritos, sendo deles o CPF utilizado como chave primária.

Nome foi definido como VARCHAR pelo motivo de ser apenas caracteres. Porém CPF é necessariamente VARCHAR pois caso o jogador tenha como CPF “03160960080”, se for armazenado como inteiro o número 0 será cortado pois é indiferente para um número iniciar com 0, assim armazenando apenas “3160960080”. Data_nascimento é uma data.

```
CREATE TABLE jogadores
(
    nome VARCHAR(40) NOT NULL,
    CPF VARCHAR(10),
    data_nascimento DATE,
    PRIMARY KEY (cpf)
);
```

Partidas é uma tabela que utiliza diversas FOREIGN KEY’s para poupar a criação de tabelas de relacionamento. Partida utiliza apenas ID_PARTIDA como Primary Key pois pode conter diversos outros jogos coms os mesmos times no mesmo campeonato.

```
CREATE TABLE partidas
(
    id_partida INT,
    PRIMARY KEY (id_partida),
    time_a INT,
    time_b INT,
    id_campeonato INT,
    FOREIGN KEY (id_campeonato) REFERENCES campeonatos (id_campeonato),
    FOREIGN KEY (time_a) REFERENCES clubes (id_clube),
    FOREIGN KEY (time_b) REFERENCES clubes (id_clube) ON DELETE CASCADE
);
```

Contrato é uma tabela de relacionamento necessária pois tem como principal característica armazenar o histórico de contratos entre jogador de clubes. Assim utilizando de chave estrangeira o CPF do jogador e id_clube, porém a chave primária é apenas id_contrato.

```sql
CREATE TABLE contrato
(
    id_contrato INT,
    PRIMARY KEY(id_contrato),
    CPF VARCHAR(40),
    id_clube INT,
    data_inicio DATE,
    data_fim DATE,
    FOREIGN KEY (id_clube) REFERENCES clubes (id_clube),
    FOREIGN KEY (CPF) REFERENCES jogadores (CPF)
);
```

E para finalizar temos a tabela Desempenho que é a representação mais significativa contendo todos os dados a serem analisados. Para a criação da tabela foi necessário utilizar como chave estrangeira CPF e id_partida, assim podendo localizar facilmente o jogador e a partida/time no qual estava jogando.

```sql
CREATE TABLE desempenho
(
    id_desempenho INT,
    CPF VARCHAR(40),
    id_partida INT,
    PRIMARY KEY (id_desempenho, CPF, id_partida),
    RB INT,
    G INT,
    A INT,
    SG INT,
    FS INT,
    FF INT,
    FD INT,
    FT INT,
    DD INT,
    DP INT,
    GC INT,
    CV INT,
    CA INT,
    GS INT,
    PP INT,
    FC INT,
    I INT,
    PE INT,
    FOREIGN KEY (CPF) REFERENCES jogadores (CPF),
    FOREIGN KEY (id_partida) REFERENCES partidas (id_partida)
);
```

4.2 Visões

Para criação das visões foi utilizado as ferramentas de INNER JOIN para realizar a consulta agrupada das tabelas relacionadas, e GROUP BY HAVING agrupando as tuplas repetidas dentro do exigido.

A visão GolsPorTime necessitava trazer os gols realizados por cada jogador em cada time, assim foi necessário realizar a consulta em desempenho onde tem o histórico de gols de cada jogador, juntando com a tabela jogadores para podermos obter o nome do jogador em especifico, juntamente com contrato para seguirmos para a consulta de qual time está relacionado com o jogador que teve tal desempenho. Então com Contrato conseguimos relacionar os clubes por contrato, a seguir temos a partida jogada por desempenho, por fim relacionando com campeonatos para obtermos a data e qual time o jogador estava jogando naquele período.

A cláusula WHERE foi utilizada para separar as buscas de jogadores que tiveram suas datas de contrato dentro do período do campeonato.
Group By foi utilizado para agrupar os jogadores por seus clubes juntamente com SUM realizando a somatória de Gols realizados por clube.

```
74 CREATE VIEW GolsPorTime
75 AS
76     SELECT SUM(d.g) Gols, j.nome Jogador, c.nome Clube
77     FROM desempenho d
78         INNER JOIN jogadores j ON j.cpf = d.cpf
79         INNER JOIN contrato jc ON jc.cpf = j.cpf
80         INNER JOIN clubes c ON c.id_clube = jc.id_clube
81         INNER JOIN partidas p ON p.id_partida = d.id_partida
82         INNER JOIN campeonatos camp ON camp.id_campeonato = p.id_campeonato
83     WHERE (jc.data_inicio < camp.data_inicio AND camp.data_inicio < jc.data_fim)
84     OR (jc.data_inicio < camp.data_fim AND camp.data_fim < jc.data_fim)
85    GROUP BY
86                 Jogador, Clube
87     HAVING
88                 COUNT(*) > 0;
89
```

As visões de GolsPorCampeonatoAtleta, GolsPorCampeonatoTime e ItensPorJogador seguiram a mesma lógica apenas alterando os componentes dentro do Group By e a diferença entre dois somatórios sendo eles Gols Feitos e Gols Sofridos. Como podemos ver no anexo 2.

5 TESTES REALIZADOS

5.1 Selecionando as visões

Após o banco construído podemos começar com as validações, assim sendo os primeiros testes em relação as visões criadas. Obtive os seguintes resultados;

GolsPorTime:

#	Gols	Jogador	Clube
1	4	Armando	Sao Paulo
2	0	Elvis	Gremio
3	5	Elvis	Vitoria
4	6	Felipe Grings	Gremio
5	0	João	Flamengo
6	5	Jonas	Inter

GolsPorCampeonatoAtleta:

#	GOLS	Campeonato	Jogador	nome
1	4	Brasileirão 2012	Armando	Sao Paulo
2	5	Brasileirão 2012	Elvis	Vitoria
3	0	Brasileirão 2013	Elvis	Gremio
4	6	Brasileirão 2012	Felipe Grings	Gremio
5	0	Brasileirão 2012	João	Flamengo
6	5	Brasileirão 2012	Jonas	Inter

GolsPorCampeonatoTime:

#	Gols_feitos	Gols_sofridos	Saldo_de_gols	Campeonato	Jogador	Clube
1	4	3	1	Brasileirão 2012	Armando	Sao Paulo
2	5	8	-3	Brasileirão 2012	Elvis	Vitoria
3	0	1	-1	Brasileirão 2013	Elvis	Gremio
4	6	12	-6	Brasileirão 2012	Felipe Grings	Gremio
5	0	6	-6	Brasileirão 2012	João	Flamengo
6	5	9	-4	Brasileirão 2012	Jonas	Inter

ItensPorJogador:

#	Campeonato	Jogador	Clube	Roubadas_de_Bola	Gols	Assistencia
1	Brasileirão 2012	Armando	Sao Paulo	27	4	5
2	Brasileirão 2012	Elvis	Gremio	19	5	8
3	Brasileirão 2012	Elvis	Vitoria	19	5	8
4	Brasileirão 2013	Elvis	Gremio	19	0	1
5	Brasileirão 2013	Elvis	Vitoria	19	0	1
6	Brasileirão 2012	Felipe Grings	Gremio	74	6	16
7	Brasileirão 2012	Felipe Grings	Inter	74	6	16
8	Brasileirão 2012	João	Flamengo	22	0	4
9	Brasileirão 2012	Jonas	Inter	57	5	6

5.2 Análise de tabelas separadamente

- Inserir item em Jogadores sem nome

```
mysql> mysql> INSERT INTO jogadores VALUES( null, "100", '2000-05-05');
ERROR 1048 (23000): Column 'nome' cannot be null
mysql>
```

- Inserir Jogadores com CPF repetido

```
mysql> INSERT INTO jogadores VALUES( "Felipe Grings", "1", '2000-05-05');
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
mysql>
```

- Alterar desempenho de jogadores sem afetar o jogador analisado

  Desempenho do jogador Elvis alterado ao longe de sua carreira sem alterar o desempenho dos demais atletas

#	Gols_feitos	Gols_sofridos	Saldo_de_gols	Campeonato	Jogador	Clube
1	4	3	1	Brasileirão 2012	Armando	Sao Paulo
2	6	8	-2	Brasileirão 2012	Elvis	Vitoria
3	2	1	1	Brasileirão 2013	Elvis	Gremio
4	6	12	-6	Brasileirão 2012	Felipe Grings	Gremio
5	0	6	-6	Brasileirão 2012	João	Flamengo
6	5	9	-4	Brasileirão 2012	Jonas	Inter

- Histórico de contratos

#	id_contrato	CPF	id_clube	data_inicio	data_fim
1	1	1	1	2012-01-01	2012-12-12
2	2	1	2	2013-01-01	2015-12-12
3	3	2	2	2012-01-01	2015-12-03
4	4	3	3	2012-01-01	2015-12-03
5	5	4	4	2012-01-01	2015-12-03
6	6	5	5	2012-01-01	2012-12-03
7	7	5	1	2013-01-01	2015-12-12

- Adicionar partida sem campeonato

```
mysql> INSERT INTO partidas VALUES (23, 1, 2,null);
ERROR 1048 (23000): Column 'id_campeonato' cannot be null
mysql>
```

- Ao deletar o campeonato deleta todos os respectivos desempenhos e partidas

```
mysql> DELETE FROM campeonatos WHERE id_campeonato = 1;
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM GolsPorTime;
Empty set (0.01 sec)
```

O controle de dois diferentes times por partida e apenas um time ativo por jogador fica por responsabilidade do software que utilizará a base de dados.

6 ANÁLISE DE RESULTADOS

Ao final da modelagem e implementação do projeto conseguimos chegar a algo muito interessante abrangendo grande parte dos requisitos levantados como principais para o desenvolvimento do projeto e caracterizar algumas limitações encontradas ao longo do desenvolvimento do projeto. Apesar de poucos dados cadastrados o principio de funcionamento é o mesmo, levando assim ao êxito da execução da modelagem.

Pontos a melhorar:

Criação de restrições para melhor segurança dos dados impedindo do usuário adicionar mais de um time por jogados ativo, e uma partida onde os dois times participantes são o mesmo.

Pontos a destacar:

A utilização de histórico de contratos e busca por datas para a análise de jogadores acredito ser algo bem interessante, assim podendo trabalhar focado em contratos assim que atualizados os resultados já serão capturados pelas visões. Evitando ao máximo a duplicação de dados, assim gerando uma melhor confiabilidade e usabilidade dos dados disponíveis.

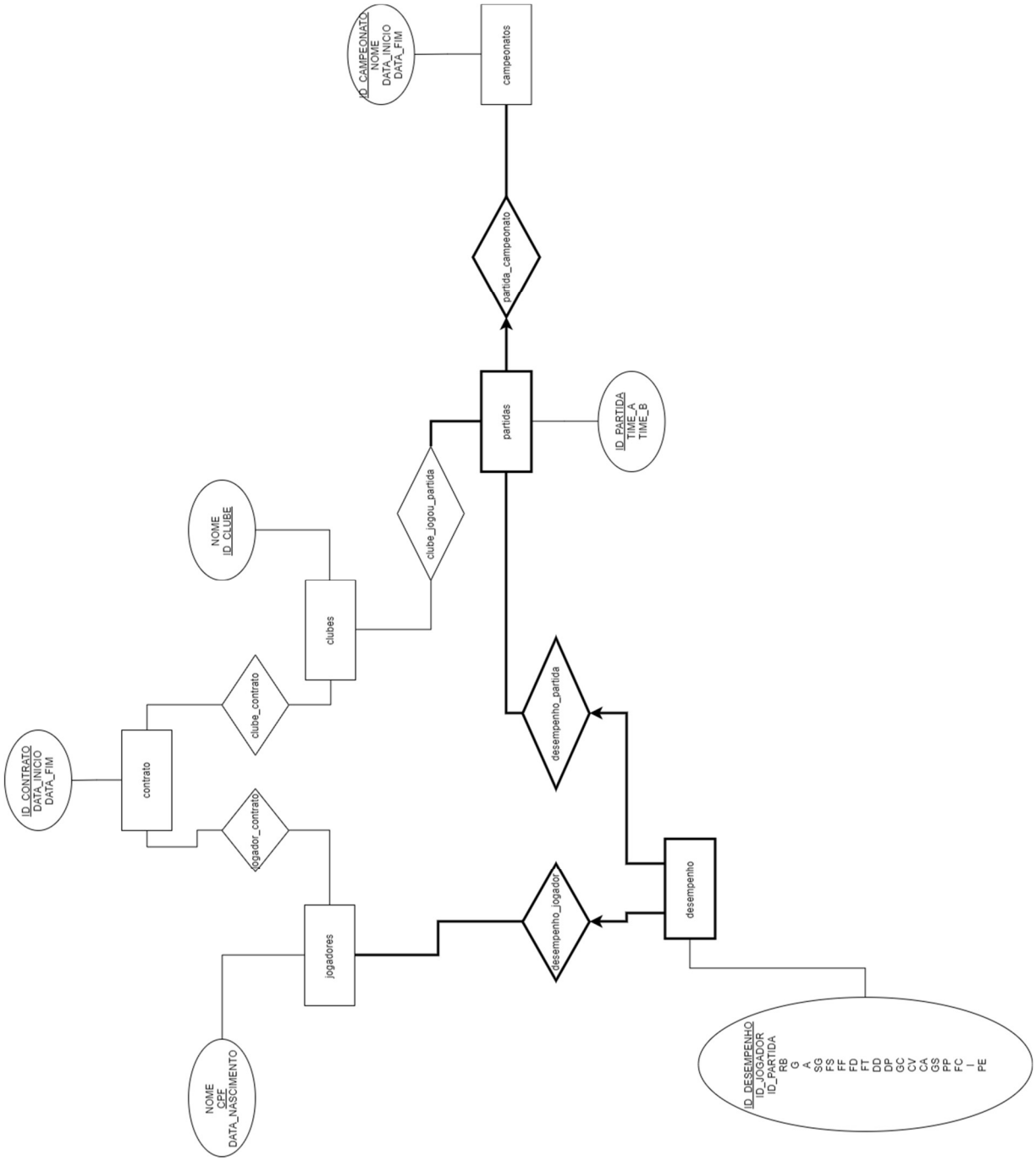
7 CONCLUSÃO

Como pode ser visto nos resultados, a modelagem proposta funcionou de acordo com os requisitos levantados atingindo os objetivos para as visões propostas, assim desenvolvendo novas habilidades para pesquisa, desenvolvimento e modelagem de base de dados, aprimorando os conhecimentos estudados durante o primeiro período do semestre.

Além disso, o desenvolvimento deste trabalho me mostrou problemas que normalmente só são observados na prática, como adicionar dois times em datas conflitantes a um jogador acarreta de duplicar todas as tuplas resultantes de uma pesquisa por inner join, assim duplicando os resultados de GolsPorTime, trazendo algo que não condiz com a realidade.

O trabalho exigiu diversos momentos de concentração e pesquisas para melhor solução do mesmo.

ANEXO 1 – MODELAGEM DIAGRAMA ER



ANEXO 2 – VISÕES PARA CONSULTA

```sql
CREATE VIEW GolsPorCampeonatoAtleta
AS
    SELECT SUM(d.g) GOLS, c.nome Campeonato, j.nome Jogador, cl.nome
    FROM campeonatos c
        INNER JOIN partidas p ON p.id_campeonato = c.id_campeonato
        INNER JOIN desempenho d ON d.id_partida = p.id_partida
        INNER JOIN jogadores j ON j.cpf = d.cpf
        INNER JOIN contrato jc ON jc.cpf = j.cpf
        INNER JOIN clubes cl ON cl.id_clube = jc.id_clube
    WHERE (jc.data_inicio < c.data_inicio AND c.data_inicio < jc.data_fim)
    OR (jc.data_inicio < c.data_fim AND c.data_fim < jc.data_fim)
    GROUP BY
 j.nome, c.nome, c.id_campeonato, cl.nome
    having
 count(*) > 0;
```

```sql
CREATE VIEW GolsPorCampeonatoTime
AS
    SELECT SUM(d.g) Gols_feitos, SUM(d.gs) Gols_sofridos, (SUM(d.g) - SUM(d.gs)) Saldo_de_gols, c.nome
    FROM campeonatos c
        INNER JOIN partidas p ON p.id_campeonato = c.id_campeonato
        INNER JOIN desempenho d ON d.id_partida = p.id_partida
        INNER JOIN jogadores j ON j.cpf = d.cpf
        INNER JOIN contrato jc ON jc.cpf = j.cpf
        INNER JOIN clubes cb ON cb.id_clube = jc.id_clube
    WHERE (jc.data_inicio < c.data_inicio AND c.data_inicio < jc.data_fim)
    OR (jc.data_inicio < c.data_fim AND c.data_fim < jc.data_fim)
  GROUP BY
 j.nome, c.nome, cb.nome
    HAVING
CREATE VIEW ItensPorJogador
AS
    SELECT c.nome Campeonato, j.nome Jogador, cb.nome Clube, SUM(d.rb) Roubadas_de_Bola, SUM(d.g) Gols, SUM(d.
    FROM campeonatos c
        INNER JOIN partidas p ON p.id_campeonato = c.id_campeonato
        INNER JOIN desempenho d ON d.id_partida = p.id_partida
        INNER JOIN jogadores j ON j.cpf = d.cpf
        INNER JOIN contrato jc ON jc.cpf = j.cpf
        INNER JOIN clubes cb ON cb.id_clube = jc.id_clube
    GROUP BY
 j.nome, c.nome, cb.nome
    HAVING
 count(*) >1;
```