

UNIVERSIDADE DO VALE DO RIO DOS SINOS

UNIDADE ACADÊMICA DE GRADUAÇÃO

CURSO DE ENGENHARIA DA COMPUTAÇÃO

ELVIS FERNANDES BUGS

JOÃO GILBERTO HOFFMANN

RELATÓRIO DE BANCO DE DADOS:

Desempenho de atleta

São Leopoldo

2019

1 INTRODUÇÃO

A boa modelagem de um banco de dados possibilita ao sistema implementado que fará uso do mesmo, uma melhor eficiência na manipulação das informações, principalmente protegendo esses dados de possíveis erros de codificação e mau uso do sistema.

O banco de dados apresentado neste relatório apresenta um modelo implementado para organizar informações de desempenho de jogadores de futebol por partida realizada, assim como dados pessoais, de clubes de futebol, de campeonatos e de partidas.

2 PREMISSAS E MODELO ER

Implementado no SGBD MySQL, O banco de dados foi construído partindo de algumas premissas, que devem ser respeitadas para funcionamento do mesmo, elas são:

- Somente pode haver um cadastro para cada atleta;
- Não é permitido ter mais de um mesmo atleta com mesmo nome completo;
- O atleta somente pode participar de um clube por vez, indiferente do campeonato;
- Toda partida possui dois clubes, o resultado da partida será o produto do atributo Resultado de cada time. Este dado não é guardado pelo banco de dados;
- Um jogador pode ter diferentes desempenhos para cada campeonato e time que jogar;
- Os itens positivos e negativos do jogador serão apresentados nas suas respectivas visões em sua totalidade, para acessar os dados de forma específica (time, campeonato), devem ser utilizadas outras views;
- Todo contrato precisa referenciar um jogador.

2.1 MODELO ER

Para isso foi realizado a modelagem através de um ER contendo todas as entidades, seus atributos e as forma como se relacionam entre si.

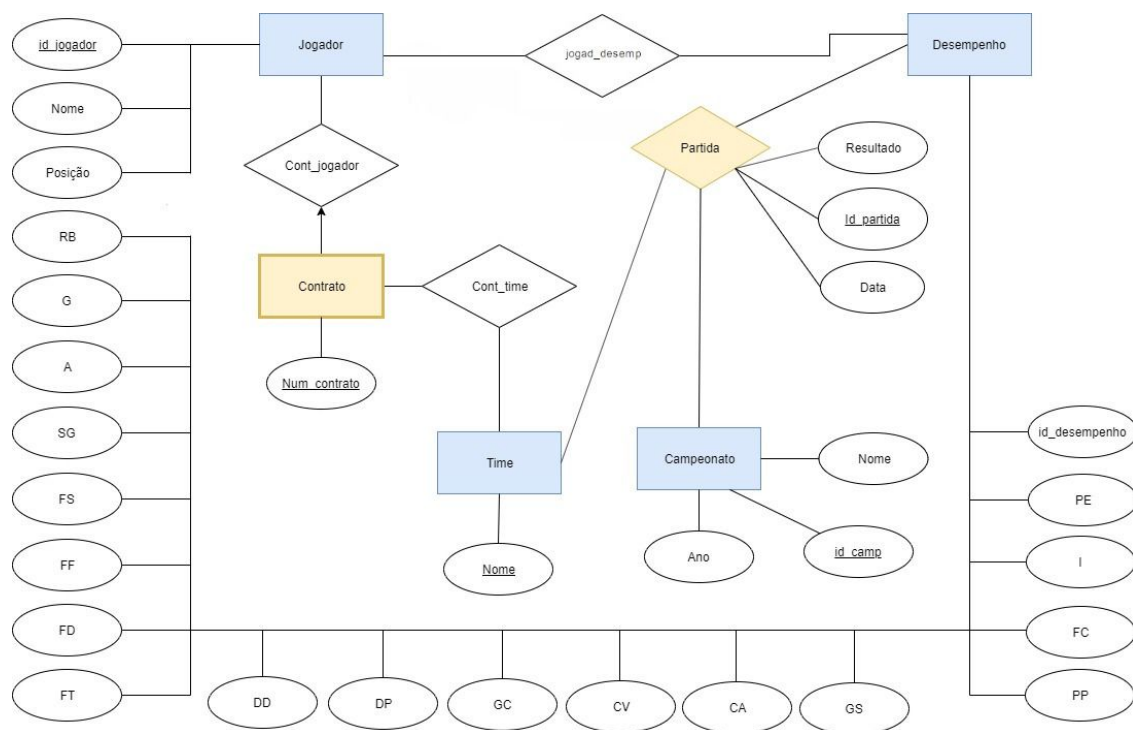


Figura 1: Modelo ER (elaborado pelos autores)

Conforme figura 1, em azul estão apresentadas às entidades do banco de dados, e em amarelo as relações que possuem tabelas. 'Contrato' apesar de ser uma entidade está em amarelo pois possui características de relacionamento.

3 BANCO DE DADOS

Após a modelagem ER, foi dado início a implementação do banco de dados, conforme semântica do MySQL, os dados numéricos são do tipo inteiro(INTEGER) e os dados de texto são do tipo caracter(varchar). Alguns atributos possuem características especiais como:

PRIMARY KEY: atributo chave, apenas pode haver um atributo com o mesmo valor por tabela. Caso uma tabela possua mais de um atributo como primary key, a combinação destes atributos acabam se tornando a chave da tabela, por exemplo, considere a tabela 'sample', que possui uma chave primária 'id_sample', este atributo não poderá ter valores repetidos pois a chave é única. Agora considere a tabela 'sample_2' onde esta possui duas primary key 'id_sample_1' e 'id_sample_2', nestas condições, a combinação destas chaves se torna a chave da tabela permitindo que um dos valores seja repetido desde que o outro não seja, exemplo:







Linha	Atributo_1	Valor_1	Atributo_2	Valor_2	Validação
1	'id_sample_1'	1	'id_sample_2'	1	
2	'id_sample_1'	1	'id_sample_2'	2	
3	'id_sample_1'	2	'id_sample_2'	1	
4	'id_sample_1'	2	'id_sample_2'	2	
5	'id_sample_1'	2	'id_sample_2'	2	
6	'id_sample_1'	1	'id_sample_2'	2	

Tabela 1: Exemplo de combinação de atributos para formar uma nova key válida(Elaborado pelos autores).

Na linha 2 da Tabela 1 podemos ver que 'id_sample_1' manteve o mesmo valor da linha 1, porém 'id_sample_2' recebeu um novo valor criando uma nova combinação das chaves tornando possível a inserção destes dados. O mesmo

acontece para linha 3, onde 'id_sample_2' repete o valor da linha 1, mas desta vez 'id_sample_1' recebe um novo valor, criando também uma nova combinação que ainda não existe na tabela e permitindo também a inserção destes dados na tabela.

Na linha 4, o valor de ambas as chaves já existem na tabela, porém a combinação destes valores ainda não havia sido utilizado, possibilitando a inserção destes dados. Já na linha 5 e 6, além de repetir os valores, a combinação também foi repetido impossibilitando a inserção dos dados na tabela.

Isto foi utilizado a nosso favor na tabela partida, onde as primary keys da tabela eram 'id_partida' e 'nome_time', desta maneira é possível referenciar dois times para uma mesma partida, de outra forma só seria possível um time jogar uma partida sozinho (o que não faria sentido).

NOT NULL: atributo não pode receber valor nulo, assim obriga que alguns valores importantes sejam sempre inseridos para certas tabelas, como por exemplo o nome do jogador na tabela 'Jogador', nome e ano do campeonato na tabela 'Campeonato', entre outros.

AUTO_INCREMENT: atributo será auto incrementado de forma sequencial quando for inserido um novo dado na tabela, assim não há a necessidade de inserir valores manualmente para certos dados. Esta cláusula foi utilizado em chaves primárias que podem ter valores arbitrados (desde que seja único) sem afetar o funcionamento do banco de dados, como por exemplo: 'id_jogador', 'id_camp', 'id_desempenho'.

UNIQUE: o atributo não poderá ser repetido de nenhuma maneira, serve para quando um atributo não é uma chave mas também não pode ser repetido. Isto foi útil para implementarmos, por exemplo, a tabela contrato, onde dizemos que o atributo 'id_jogador' é UNIQUE, pois caso não fosse, um jogador poderia ter mais de um contrato ativo ao mesmo tempo com mais de um time, o que não condiz com a realidade.

UPDATE CASCADE: esta cláusula foi utilizado em todas as foreign keys, pois quando forem atualizadas em suas tabelas de origem, também serão atualizadas nas tabelas onde estiverem referenciadas.

DELETE ON CASCADE: foi utilizado nas foreign keys para implementar a entidade fraca, pois quando for deletado um dado na tabela de origem, também é

deletado das demais tabelas que possuem relação fraca e referenciam essa foreign key.

3.1 ENTIDADES E ATRIBUTOS

A entidade 'Time_' possui apenas um dado de texto para guardar o nome do clube, portanto sendo seu atributo chave. Optamos por fazer assim pois não pode existir mais de dois times com exatamente o mesmo nome, mesmo que um time possua parte do nome de outro, sempre é adicionado uma variante ao nome, exemplo: Barcelona de Guayaquil, Atlético Paranaense, Atlético Mineiro, Atlético Goianiense, Grêmio Barueri, etc....

```
CREATE TABLE Time_ (  
    Nome_time varchar(255) PRIMARY KEY  
);
```

A entidade 'Jogador' possui dados como nome, posição e id, sendo o último um número e os demais em texto, o id foi escolhido como atributo chave. O nome do jogador não pode ser nulo pois não faz sentido ter um jogador sem nome e deve ser único (nome completo) para não permitir jogadores duplicados.

```
CREATE TABLE Jogador (  
    id_jogador INTEGER PRIMARY KEY AUTO_INCREMENT,  
    Nome_jogador varchar(255) NOT NULL UNIQUE,  
    Posicao varchar(255)  
);
```

A entidade 'Campeonato' possui dados como nome, id e ano, nome em texto e os demais como números, seu id e ano foram escolhidos como atributos chave. Os dados nome e ano não podem ser nulos pois todo campeonato precisa ter um nome e ser referente a algum ano específico, eles também podem ser repetidos pensando em que um mesmo campeonato pode ocorrer mais de uma vez ao decorrer do mesmo ano.

```
CREATE TABLE Campeonato (
    Nome_camp varchar(255) NOT NULL,
    id_camp INTEGER AUTO_INCREMENT,
    Ano integer NOT NULL,
    PRIMARY KEY (id_camp, Ano)
);
```

A tabela 'Desempenho' possui todos os dados de desempenho do atleta por partida(inclusive dados históricos de outros times que ele jogou no passado) em valores numéricos, assim como seu id, escolhido como atributo chave da tabela.

Ainda possui o id da partida que faz a relação com 'partida', id do jogador para fazer a relação com 'jogador' e nome do time que faz a relação com 'time_'.

```
CREATE TABLE Desempenho (
    RB INTEGER,
    G INTEGER,
    A INTEGER,
    SG INTEGER,
    FS INTEGER,
    FF INTEGER,
    FD INTEGER,
    FT INTEGER,
    DD INTEGER,
    DP INTEGER,
    GC INTEGER,
    CV INTEGER,
    CA INTEGER,
    GS INTEGER,
    PP INTEGER,
    FC INTEGER,
    I INTEGER,
    PE INTEGER,
    id_desempenho INTEGER PRIMARY KEY AUTO_INCREMENT,
    id_partida INTEGER NOT NULL,
    id_jogador INTEGER NOT NULL,
    Nome_time varchar(255) NOT NULL,
    CONSTRAINT jogad_desemp FOREIGN KEY (id_jogador) REFERENCES Jogador(id_jogador) on
UPDATE CASCADE,
    CONSTRAINT part_desemp FOREIGN KEY (id_partida) REFERENCES Partida (id_partida) on UPDATE
CASCADE,
    CONSTRAINT desemp_time FOREIGN KEY (Nome_time) REFERENCES Time_ (Nome_time) on
UPDATE CASCADE
);
```


3.2 RELAÇÕES

A relação 'Partida' possui dados id, resultado como numéricos, data do tipo date e recebe dados de outras entidades como nome de clubes (chave de tabela 'Time_') e o id de 'Campeonato' fazendo o relacionamento entre estas tabelas. Optamos por implementar desta forma pois assim fica mais legível e organizado o relacionamento. Duas chaves foram escolhidas para esta tabela, o id da partida sendo um atributo próprio da tabela, e o nome do time como chave complementar. A escolha de ter duas primary key foi feita pois desta maneira (como explicado anteriormente) podemos referenciar dois times a uma mesma partida.

Todos os atributos da tabela são not null pois são informações importantes para utilizar em um sistema real, assim como todas as chaves estrangeiras são not null para obrigar a referenciar as outras tabelas.

```
CREATE TABLE Partida (  
    Data_ date NOT NULL,  
    Resultado INTEGER NOT NULL,  
    id_partida INTEGER,  
    id_camp INTEGER NOT NULL,  
    Nome_time varchar(255) NOT NULL,  
    PRIMARY KEY(id_partida,Nome_time),  
    CONSTRAINT part_camp FOREIGN KEY (id_camp) REFERENCES Campeonato (id_camp) on  
UPDATE CASCADE,  
    CONSTRAINT part_time FOREIGN KEY (Nome_time) REFERENCES Time_ (Nome_time) on  
UPDATE CASCADE  
);
```

A entidade 'Contrato' armazena as relações entre as entidades 'Jogador' e 'Time_', ela possui o número do contrato como atributo chave, recebe o nome do clube e id do jogador como atributos externos. Nesta tabela restringimos o id jogador (como já explicado anteriormente).

```
CREATE TABLE Contrato (  
    Num_contrato INTEGER PRIMARY KEY AUTO_INCREMENT,  
    Nome_time varchar(255) NOT NULL,  
    id_jogador INTEGER NOT NULL UNIQUE,  
    CONSTRAINT cont_jogador FOREIGN KEY (id_jogador) REFERENCES Jogador(id_jogador) ON  
DELETE CASCADE on UPDATE CASCADE,  
    CONSTRAINT cont_time FOREIGN KEY (Nome_time) REFERENCES Time_(Nome_time) ON  
DELETE CASCADE on UPDATE CASCADE  
);
```

3.3 VISÕES

Antes de explicar cada visão, é importante explicar como funciona a cláusula INNER JOIN e porque escolhemos utilizar, pois todas as visões acabaram usando.

O INNER JOIN serve para associar duas tabelas na interseção destas duas a partir de um atributo semelhante para ambas as entidades. Desta forma podemos criar uma visão mais genérica, pois ela não fica estática para um determinado valor, e na verdade se torna dinâmica com a tabela que quisermos (uma ou mais tabelas), tirando a necessidade de ter que atualizar essas visões conforme o banco vai crescendo.

A visão 'GolsPorTime' retorna a quantidade de gols e nome de um jogador por time registrado na tabela 'Desempenho'. Essa visão foi implementado com um select que realiza a soma total da quantidade de gols e agrupa as informações por 'id_jogador' e 'nome_time', a tabela resultante desse select ainda é organizada pelos nomes dos jogadores em ordem alfabética.

```
CREATE VIEW GolsPorTime(Gols, Nome_jogador, Nome_time) as
  SELECT sum(Desempenho.G) as Gols, jogador.Nome_jogador, Desempenho.Nome_time
  FROM (Desempenho
  INNER JOIN jogador ON Desempenho.id_jogador = jogador.id_jogador)
  GROUP BY Desempenho.id_jogador, Desempenho.nome_time
  ORDER BY Nome_jogador asc;
```

A visão 'GolsPorCampeonatoAtleta' retorna o somatório de gols, o nome do jogador, o nome do time e campeonato por jogador registrado na tabela desempenho.

Para implementarmos essa visão foi feito um select que realiza o soma total dos gols e agrupa por 'id_jogador', 'id_camp' e 'nome_time', o resultado desse select é organizado por nome do jogador de forma alfabética.

```
CREATE VIEW GolsPorCampeonatoAtleta(Gols, Nome_jogador, Nome_time, Nome_camp) as
  SELECT sum(Desempenho.G) as Gols, Jogador.Nome_jogador, Desempenho.Nome_time,
  Campeonato.Nome_camp
  FROM (((Desempenho
  INNER JOIN Partida ON Desempenho.id_partida = Partida.id_partida)
  INNER JOIN Campeonato ON partida.id_camp = Campeonato.id_camp)
  INNER JOIN jogador ON Desempenho.id_jogador = jogador.id_jogador)
  GROUP BY Desempenho.id_jogador, partida.id_camp, Desempenho.nome_time
  ORDER BY Nome_jogador asc;
```

A visão 'GolsPorCampeonatoTime' retorna o somatório de gols, o somatório de gols sofridos e o saldo de gols do time por campeonato.

Para implementarmos essa visão foi feito um select que realiza o soma total dos gols, a soma total dos gols sofridos e faz a soma da diferença da quantidade de gols por gols sofridos (saldo de gols do time) a tabela é agrupada por 'id_camp' e 'nome_time'.

```
CREATE VIEW GolsPorCampeonatoTime(Nome_time, Nome_camp, GolsFeitos, GolsSofridos, SaldoGols) as
SELECT
    Desempenho.Nome_time
    , Campeonato.Nome_camp
    , sum(Desempenho.G) as Gols
    , sum(Desempenho.GS) as GolsSofridos
    , sum(Desempenho.G-Desempenho.GS) as SG
FROM ((Desempenho
INNER JOIN Partida ON Desempenho.id_partida = Partida.id_partida)
INNER JOIN Campeonato ON partida.id_camp = Campeonato.id_camp)
GROUP BY Desempenho.nome_time, Campeonato.id_camp;
```

A visão 'ItensPorJogador' retorna o somatório de cada item da tabela desempenho por jogador separando diferentes desempenhos para cada time e campeonato que ele jogou.

Para implementarmos essa visão foi feito um select que realiza o soma dos atributos de desempenho do jogador, o nome do time, nome do jogador e nome do campeonato, a tabela é agrupada por 'id_jogador', 'id_camp' e 'nome_time'.

```
CREATE VIEW ItensPorJogador(G, RB, A, SG, FS, FF, FD, FT, DD, DP, GC, CV, CA, GS, PP, FC, I, PE, Nome_time,
Nome_jogador, Nome_camp) as
SELECT sum(Desempenho.G) as G
    , sum(Desempenho.RB) as RB
    , sum(Desempenho.A) as A
    , sum(Desempenho.SG) as SG
    , sum(Desempenho.FS) as FS
    , sum(Desempenho.FF) as FF
    , sum(Desempenho.FD) as FD
    , sum(Desempenho.FT) as FT
    , sum(Desempenho.DD) as DD
    , sum(Desempenho.DP) as DP
    , sum(Desempenho.GC) as GC
    , sum(Desempenho.CV) as CV
    , sum(Desempenho.CA) as CA
    , sum(Desempenho.GS) as GS
    , sum(Desempenho.PP) as PP
    , sum(Desempenho.FC) as FC
    , sum(Desempenho.I) as I
    , sum(Desempenho.PE) as PE
    , Desempenho.Nome_time
```

```
,jogador.Nome_jogador  
,Campeonato.Nome_camp  
FROM (((Desempenho  
INNER JOIN Partida ON Desempenho.id_partida = Partida.id_partida)  
INNER JOIN jogador ON Desempenho.id_jogador = jogador.id_jogador)  
INNER JOIN Campeonato ON partida.id_camp = Campeonato.id_camp)  
GROUP BY Desempenho.id_jogador, partida.id_camp, Desempenho.nome_time;
```

4 RESULTADOS

Ao longo do desenvolvimento do banco de dados, fomos encontrando divergências no nosso diagrama ER e fomos concertando para ficar mais dentro da realidade e que fizesse mais sentido nos conformes de um sistema como foi requisitado. As tabelas e visões citadas nos tópicos anteriores foram as tabelas geradas ao final do trabalho. Abaixo temos uma imagem das tabelas da database que foi gerada:

```
mysql> show tables;
+-----+
| Tables_in_ta_elvis_joao |
+-----+
| campeonato               |
| contrato                 |
| desempenho               |
| golsporcampeonatoatleta  |
| golsporcampeonatotime   |
| golsportime               |
| itensporjogador          |
| jogador                  |
| partida                  |
| time_                     |
+-----+
10 rows in set (0.00 sec)
```

Figura 2: Tabelas geradas para o projeto(elaborado pelos autores)

Com isso inserimos alguns dados nas tabelas para avaliarmos nosso projeto, abaixo pode ser visto a tabela campeonato com dados inseridos:

```
mysql> select * from campeonato;
+-----+-----+-----+
| Nome_camp | id_camp | Ano |
+-----+-----+-----+
| Brasileiro | 1 | 2019 |
| Libertadores | 2 | 2019 |
| Copa do Brasil | 3 | 2019 |
| Copa do Brasil | 4 | 2021 |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

Figura 3: Tabela campeonato com alguns dados de teste inseridos(elaborado pelos autores)

Como pode ser visto na Figura 3 os campeonatos podem repetir nome desde que varie o 'ano', abaixo pode ser visto o que acontece caso tente se inserir um campeonato já existente:

```
mysql> INSERT INTO Campeonato(Nome_camp,Ano,id_camp) VALUES ("Brasileiro",2019,1);
ERROR 1062 (23000): Duplicate entry '1-2019' for key 'PRIMARY'
mysql> _
```

Figura 4: Mensagem de erro ao tentar duplicar dado na tabela campeonato(elaborado pelos autores)

Abaixo vamos mostrar alguns o restante das tabelas com os dados já inseridos:

```
mysql> select * from contrato;
```

Num_contrato	Nome_time	id_jogador
1	Inter	1
2	Grêmio	2
3	Palmeiras	3
4	São Paulo	4
5	Flamengo	5
6	Flamengo	6
7	Inter	7

```
7 rows in set (0.00 sec)
```

Figura 5: Tabela contrato com dados inseridos(elaborado pelos autores)

Nesta tabela podemos ver que o MySQL tem alguns problema com unicode não nos permitindo utilizar acentuação de forma correta.

```
mysql> select * from desempenho;
```

RB	G	A	SG	FS	FF	FD	FT	DD	DP	GC	CV	CA	GS	PP	FC	I	PE	id_desempenho	id_partida	id_jogador	Nome_time
1	4	3	4	5	6	7	8	9	0	7	8	9	1	3	4	42	3	1	1	1	Inter
1	1	3	4	5	6	7	8	9	8	7	8	9	7	3	4	42	3	2	1	2	Grêmio
1	2	3	4	5	6	7	8	9	3	7	8	9	1	3	4	42	3	3	2	3	Palmeiras
1	0	3	4	5	6	7	8	9	1	7	8	9	200	3	4	42	3	4	2	4	São Paulo
1	1	3	4	5	6	7	8	9	0	7	8	9	7	3	4	42	3	5	2	7	Inter
1	1	3	4	5	6	7	8	9	0	7	8	9	7	3	4	42	3	6	3	7	Inter

```
6 rows in set (0.00 sec)
```

Figura 6: Tabela desempenho com dados inseridos(elaborado pelos autores)

Alguns dados foram exagerados para realizarmos testes nas visões.

```
mysql> select * from jogador;
```

id_jogador	Nome_jogador	Posicao
1	Elvis Bugs	Zagueiro
2	João Hoffmann	Atacante
3	Armando Keller	Meio-campo
4	Joaquim Uzumaki	Goleiro
5	Gabriel Becker	Atacante
6	Renato Uchiha	Lateral
7	Paolo Guerreiro	Atacante

```
7 rows in set (0.00 sec)
```

Figura 7: Tabela jogador com dados inseridos(elaborado pelos autores)

```
mysql> select * from partida;
```

Data_	Resultado	id_partida	id_camp	Nome_time
2019-10-02	0	1	1	Grêmio
2019-10-02	2	1	1	Inter
2019-10-02	3	2	1	Palmeiras
2019-10-02	0	2	1	São Paulo
2019-10-02	3	3	3	Inter

```
5 rows in set (0.00 sec)
```

Figura 8: Tabela partida com dados inseridos(elaborado pelos autores)

```
mysql> select * from time_;
```

Nome_time
Atlético-Mineiro
Atlético-Paranaense
Botafogo
Corinthians
Cruzeiro
Flamengo
Fluminense
Grêmio
Inter
Palmeiras
São Paulo
Santos
Vasco

```
13 rows in set (0.00 sec)
```

Figura 9: Tabela time_ com dados inseridos(elaborado pelos autores)

Após os dados estarem inseridos nas tabelas, podemos ver se as visões estão funcionando corretamente, nas figuras a seguir podemos analisar cada visão:

```
mysql> select * from golsportime;
```

Gols	Nome_jogador	Nome_time
2	Armando Keller	Palmeiras
4	Elvis Bugs	Inter
1	João Hoffmann	Grêmio
0	Joaquim Uzumaki	São Paulo
2	Paolo Guerreiro	Inter

```
5 rows in set (0.00 sec)
```

Figura 10: Visão GolsPorTime (elaborado pelos autores)

Podemos ver na Figura 10 que os dados foram agrupados de forma correta como esperado, mostrando o somatório de gols dos jogadores por time.


```
mysql> select * from golsporcampeonatoatleta;
```

Gols	Nome_jogador	Nome_time	Nome_camp
4	Armando Keller	Palmeiras	Brasileiro
8	Elvis Bugs	Inter	Brasileiro
2	João Hoffmann	Grêmio	Brasileiro
0	Joaquim Uzumaki	São Paulo	Brasileiro
2	Paolo Guerreiro	Inter	Brasileiro
1	Paolo Guerreiro	Inter	Copa do Brasil

6 rows in set (0.00 sec)

Figura 11: Visão GolsPorCampeonatoAtleta (elaborado pelos autores)

Podemos ver na Figura 11 que os dados foram agrupados de forma correta como esperado, mostrando o somatório de gols dos jogadores por campeonato e time.

```
mysql> select * from golsporcampeonatotime;
```

Nome_time	Nome_camp	GolsFeitos	GolsSofridos	SaldoGols
Inter	Brasileiro	10	16	-6
Grêmio	Brasileiro	2	14	-12
Palmeiras	Brasileiro	4	2	2
São Paulo	Brasileiro	0	400	-400
Inter	Copa do Brasil	1	7	-6

5 rows in set (0.00 sec)

Figura 12: Visão GolsPorCampeonatoTime (elaborado pelos autores)

Podemos ver na Figura 12 que os dados foram agrupados de forma correta como esperado, mostrando o somatório de gols, o somatório dos gols sofridos e o saldo de gols de cada time por campeonato.

```
mysql> select * from itensporjogador;
```

G	RB	A	SG	FS	FF	FD	FT	DD	DP	GC	CV	CA	GS	PP	FC	I	PE	Nome_time	Nome_jogador	Nome_camp
8	2	6	8	10	12	14	16	18	0	14	16	18	2	6	8	84	6	Inter	Elvis Bugs	Brasileiro
2	2	6	8	10	12	14	16	18	16	14	16	18	14	6	8	84	6	Grêmio	João Hoffmann	Brasileiro
4	2	6	8	10	12	14	16	18	6	14	16	18	2	6	8	84	6	Palmeiras	Armando Keller	Brasileiro
0	2	6	8	10	12	14	16	18	2	14	16	18	400	6	8	84	6	São Paulo	Joaquim Uzumaki	Brasileiro
2	2	6	8	10	12	14	16	18	0	14	16	18	14	6	8	84	6	Inter	Paolo Guerreiro	Brasileiro
1	1	3	4	5	6	7	8	9	0	7	8	9	7	3	4	42	3	Inter	Paolo Guerreiro	Copa do Brasil

6 rows in set (0.00 sec)

Figura 13: Visão ItensPorJogador (elaborado pelos autores)

Por último temos a Figura 13 que nos mostra os dados de desempenho de cada jogador por time e campeonato.

5 CONCLUSÃO

Apresentados os resultados, ficou claro que um bom planejamento realizado através do estudo dos requisitos e modelagem do banco de dados colabora na implementação do mesmo. O estudo do MySQL também influenciou os resultados, pois a primeira tentativa foi realizada em outro SGBD e problemas com os tipos de dados apareceram.

Com o domínio do SGBD e testes realizados, foi observado que melhorias poderiam ser realizadas, depois de tais modificações chegamos a um resultado satisfatório.