Universidade Federal de São Carlos
Bacharelado em Ciência da Computação
Banco de Dados

Grupo 12 - Esportes
Fase Final

Prof. Dra. Sahudy Montenegro González

Nome: Lucas Alexandre Occaso RA: 620505
Nome: Ricardo Mendes Leal Junior RA: 562262
Nome: Vitor Pratalli Camilo RA: 620181

SOROCABA
8 de Dezembro de 2017

1. Especificação do problema

Temos que um clube esportivo está desenvolvendo suas categorias de base e aderindo a novos esportes, com isso a quantidade de novos atletas, profissionais contratados e o aumento de campeonatos disputados fez com que a quantidade de dados aumentasse consideravelmente, assim tendo a necessidade da criação de um sistema para conseguir armazenar, manipular e recuperar os dados de forma eficiente.

1.1. Objetivo do sistema

O objetivo do trabalho é a modelagem de um banco de dados afim de resolver o problema com a organização dos dados de um clube esportivo. O sistema deverá armazenar as informações de um atleta, como nome, idade e desempenho também deverá armazenar as informações de categoria e esporte referente a um time.

O sistema deverá guardar as informações divisão e nome dos campeonatos do qual o clube participa, também deverá guardar as informações, especificação, cargo e nome dos profissionais que trabalham em um time.

Um profissional faz parte de uma comissão de um ou mais times através de um contrato, um time é formado por vários atletas e um atleta faz parte de apenas um time e um time está inscrito em um ou mais campeonatos.

Um atleta precisará ter um código, que será gerado automaticamente quando um novo atleta for cadastrado, deverá armazenar seu nome, sua idade que deverá ser maior que 15 anos e seu desempenho que deverá estar entre 0 e 100, além do código do time que ele participa.

Um time precisará ter um código, que será gerado automaticamente quando um novo time for cadastrado, deverá armazenar sua categoria e seu esporte.

Um campeonato deverá armazenar os códigos dos times que participam deste campeonato, sua divisão e seu nome.

Um profissional precisará ter um código, que será gerado automaticamente quando um novo profissional for cadastrado, além de seu nome, sua especialidade, seu cargo e seus telefones no formato +pp(xx)nnnnn-nnnn.

E deverá armazenar um contrato, que é formado por um profissional e por um time, além do número do novo contrato cadastrado que será gerado automaticamente pelo banco.

2. Requisito de dados

Nesta seção serão apresentados os requisitos de dados, especificando quais as consultas que o sistema deverá realizar.

R01. O sistema deverá recuperar o todos os dados de um determinado atleta.

R02. O sistema deverá recuperar todos os campeonatos e divisões, dado um esporte e uma categoria do time.

R03. O sistema deverá recuperar a quantidade de profissionais que fazer parte de uma comissão

R04. O sistema deverá recuperar todos os esportes que o clube participa.

R05. O sistema deverá recuperar o desempenho total de um time, baseado no desempenho individual de cada atleta, dado seu esporte e categoria.

R06. O sistema deverá recuperar quantos profissionais fazem parte da comissão de um time.

3. Projeto conceitual

Nesta seção será apresentado o projeto conceitual, que consta do Modelo Entidade-Relacionamento (MER) e também a descrição do tipo de cada atributo por entidade.

3.1. Modelo Entidade-Relacionamento

Abaixo, na Figura 1, se encontra o MER que é usado para descrever, de maneira conceitual, como será o armazenamento dos dados no sistema.

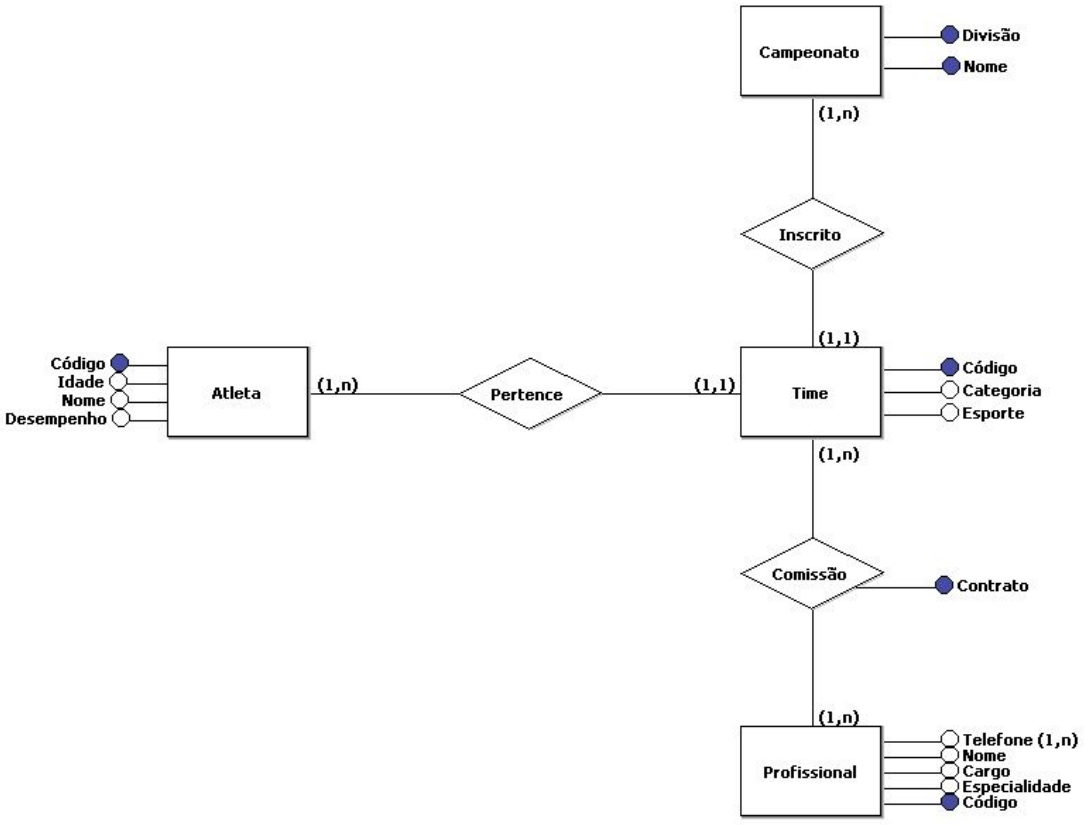

Figura 1 - Modelo Entidade Relacionamento

3.2. Descrição do tipo de atributos por tipo-entidade

A seguir será mostrado nas tabelas de 1 a 5 os atributos de cada Tipo-Entidade e Tipo-Relacionamento, suas descrições de tipo e suas restrições.

Atributo	Tipo	Restrição
cod_atleta	Identificador	Obrigatório
nome_atleta	Monovalorado	Obrigatório
idade	Monovalorado	Obrigatório e idade >= 15
desempenho	Monovalorado	Obrigatório e 0 < desempenho < 100
cod_time	Monovalorado	Obrigatório

Tabela 1: Tipo-Entidade Atleta

Atributo	Tipo	Restrição
cod_time	Identificador	Obrigatório

categoria	Monovalorado	Obrigatório
esporte	Monovalorado	Obrigatório

Tabela 2: Tipo-Entidade Time

Atributo	Tipo	Restrição
cod_time	Identificador	Obrigatório
divisao	Identificador	Obrigatório
nome_camp	Monovalorado	Obrigatório

Tabela 3: Tipo-Entidade Campeonato

Atributo	Tipo	Restrição
cod_prof	Identificador	Obrigatório
nome_prof	Monovalorado	Obrigatório
cargo	Monovalorado	Obrigatório
telefone	Multivalorado	Obrigatório e no formato +pp(xx)nnnnn-nnnn
especialidade	Monovalorado	Obrigatório

Tabela 4: Tipo-Entidade Profissional

Atributo	Tipo	Restrição
cod_prof	Identificador	Obrigatório
cod_time	Identificador	Obrigatório
contrato	Identificador	Obrigatório

Tabela 5: Tipo-Relacionamento comissão

4. Projeto Lógico

Abaixo, na figura 2, temos o conjunto de relações que especificam o banco de dados que será implementado, a transformação do MER para o modelo lógico foi feita com o auxílio da ferramenta BrModelo.

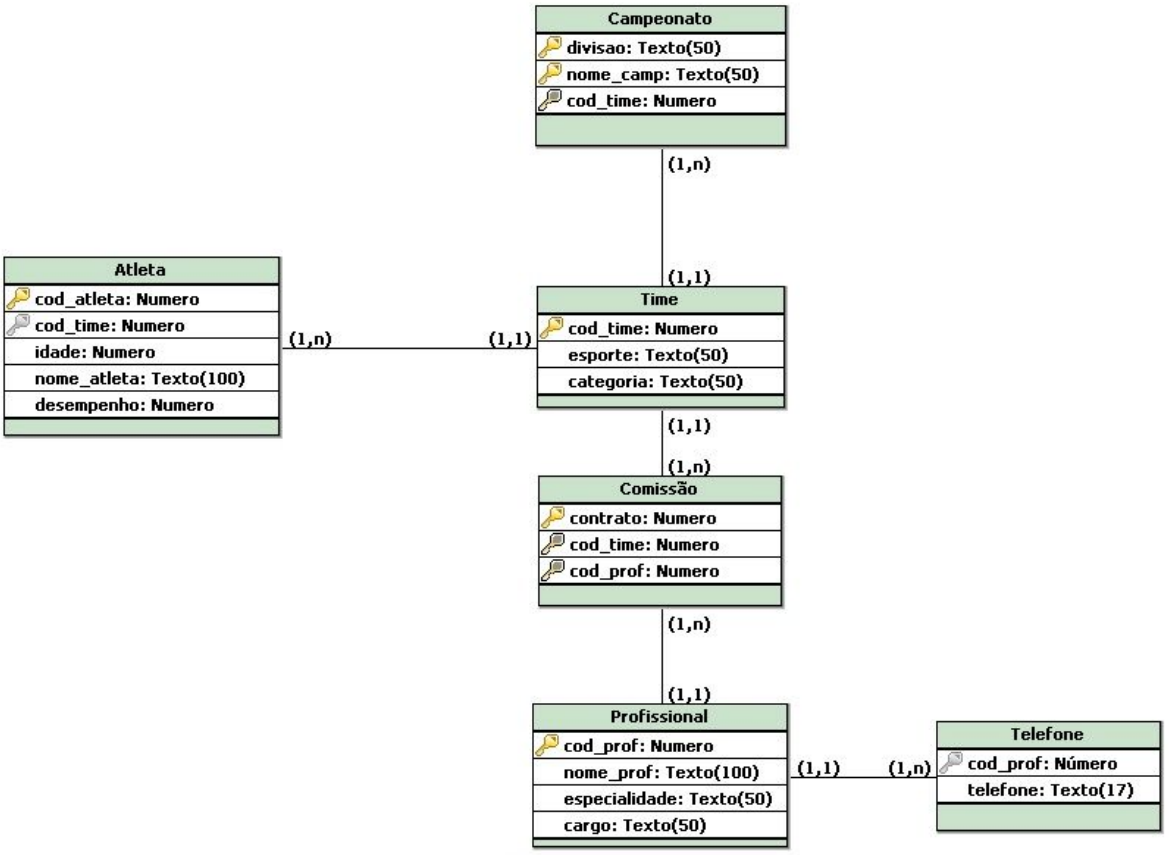

Figura 2 - Modelo Lógico

Nosso modelo, respeita as seguintes formas normais:

1.FN

Está na primeira forma normal pois, no nosso esquema não há nenhum atributo multivalorado, o atributo que era multivalorado no MER era o campo telefone do profissional, ele foi removido normalizando o banco, criando uma nova tabela para o campo telefone que herda a chave primária do profissional.

2.FN

Está na segunda forma normal, pois no nosso esquema não há nenhum atributo não chave que não dependa parcialmente da super chave ou chave primária.

3.FN

Está na terceira forma normal, pois no nosso esquema não há nenhuma dependência funcional transitiva a partir da super chave ou chave primária.

5. Projeto Físico do Banco de Dados

A criação do banco de dados acontece, primeiramente, com a criação das tabelas através do SQL. As tabelas geradas são: **time**, **campeonato**, **atleta**, **profissional**, **telefone** e **comissão**. Os scripts gerados para a criação dos banco de dados pode ser encontrado anexado no arquivo *esquema.sql.*

Em relação às políticas de integridade dos dados em nosso sistema, consiste em cláusula como **NOT NULL**, **PRIMARY KEY**, **FOREIGN KEY** e **CASCADE**.

Na tabela time, temos o atributo **cod_time** como **PRIMARY KE**y, que é o código único para identificar um time, esporte que é um texto onde obrigatoriamente deve ser inserido um texto, atribuímos **NOT NULL** a este atributo para que isso ocorra. O atributo categoria também deve ser inserido um texto obrigatoriamente, desta maneira, este é **NOT NULL**.

Na tabela campeonato, os atributos **divisao** e **nome_camp** são **NOT NULL**, pois é necessário inserir um texto ao inserir dados nestes campos. Além disso, temos o atributo **cod_time** que é **FOREIGN KEY** e faz referência a tabela **time**, onde há atualização deste campo quando é mudado ou deletado algum dado deste atributo, desta forma, utilizamos a cláusula de **CASCADE**. É importante ressaltar que que todos os atributos nesta tabela, são uma chave primária (**PRIMARY KEY**) composta.

Na tabela **atleta**, o atributo **cod_atleta**, é um identificador único para cada atleta, assim, é **PRIMARY KEY** nesta tabela. Os atributos **idade**, **nome_atleta** e **desempenho** têm cláusulas de **NOT NULL**, pois é obrigatório inserir dados nestes campos. O atributo cod_time é chave estrangeira (**FOREIGN KEY**), que faz referência a tabela **time**. Deste modo, usamos a cláusula **CASCADE**, para atualizar este atributo nesta tabela também, caso haja um atualização ou exclusão de algum dado deste atributo.

Na tabela profissional, o atributo **cod_prof** que é único para cada profissional, é a **PRIMARY KEY**, além de os atributos **nome_prof**, **desempenho** e **cargo** tem cláusulas **NOT NULL**, pois é obrigatório inserir dados nestes atributos.

Na tabela **telefone** que foi gerado de um campo multivalorado da tabela profissional, tem como atributo o **cod_prof** que é **FOREIGN KEY** que faz referência a tabela profissional e o outro atributo é o próprio **telefone**, do qual tem a cláusula **not null**, pois é obrigado inserir dado neste atributo.

Por fim, temos a tabela **comissão**, que tem um atributo **contrato** e os atributos **cod_time** e **cod_prof**, que são chaves estrangeiras e fazem referência as tabelas **time** e **profissional**, respectivamente. Além disso, é importante lembrar que são chaves estrangeiras têm a cláusula **CASCADE**, caso haja alguma mudança em uma atualização ou exclusão de algum dado destes atributos. Assim, temos que todos os atributos desta tabela são uma **PRIMARY KEY** composta.

É importante ressaltar que os casos de integridade de idade > 5, desempenho entre 0 e 100 e o telefone em um determinado formato serão testados através das triggers, como pode ser visto na seção 7 deste documento.

Em relação a alimentação do banco de dados, foi necessário criar vários nomes para os atletas, campeonatos e profissionais, além de criar valores aleatórios mas que fazem sentidos para os outros campos. Os scripts de inserção de dados no banco de dados se encontram no arquivo `dados.sql`.

6. Especificação de Consultas em Álgebra Relacional e na SQL

Consulta 1 - Listar todas as informações de um atleta.

AR

$$\pi_{\text{nome\_atleta, idade, desempenho}}(\sigma_{\text{nome\_atleta = <nome>}}(\text{atleta}))$$

SQL

```
SELECT nome_atleta, idade, desempenho
FROM atleta AS a
WHERE a.nome_atleta = <nome>;
```

Consulta 2 - Listar os nomes e divisões, dado um esporte e uma categoria de um time.

AR

$$t_1 \leftarrow \sigma_{\text{esporte = <esporte> } \wedge \text{ categoria = <categoria>}}(\text{time})$$

$$\pi_{\text{nome\_camp, divisao}}(t_1 \bowtie \text{campeonato})$$

SQL

```
SELECT DISTINCT c.nome_camp, c.divisao
FROM time AS t, campeonato AS c
WHERE t.esporte = <esporte> AND t.categoria = <categoria>;
```

Consulta 3 - Listar todas as informações de um profissional.

AR

$$t_1 \leftarrow \sigma_{\text{nome\_prof = <nome>}}(\text{profissional})$$

$$t_2 \leftarrow \sigma\ t_1 \bowtie \text{telefone}$$

$$\pi_{\text{nome\_prof, especialidade, cargo, telefone}}(t_2)$$

SQL

```
SELECT nome_prof, especialidade, cargo, telefone
FROM telefone AS t, profissional AS p
WHERE p.nome_prof = <nome> AND p.cod_prof = t.cod_prof;
```

Consulta 4 - Listar todos os esportes cadastrados no clube.

AR

$$\pi_{\text{esporte}}(\text{time})$$

SQL

```
SELECT DISTINCT esporte
FROM time;
```

Consulta 5 - Mostrar o desempenho do time baseado no desempenho individual de cada atleta, dado seu esporte e categoria.

AR

$$t_1 \leftarrow \sigma_{\text{esporte = <esporte> } \wedge \text{ categoria = <categoria>}}(\text{time})$$

$t_2 \leftarrow t_1 \bowtie$ atleta

$\mathcal{F}_{avg(desempenho)}(t2)$

SQL

```
SELECT avg(desempenho) as d
FROM time AS t, atleta AS a
WHERE t.esporte = <esporte> AND t.categoria = <categoria> AND
t.cod_time = a.cod_time;
```

Consulta 6 - Contar quantos profissionais fazem parte da comissão de um time.

AR

$t_1 \leftarrow \sigma_{esporte = <esporte> \wedge categoria = <categoria>}(time)$

$t_2 \leftarrow t_1 \bowtie$ comissao

$\mathcal{F}_{count(cod\_prof)}(t_2)$

SQL

```
SELECT count(cod_prof) as qtdd_prof
FROM time AS t, comissao AS c
WHERE t.esporte = <esporte> AND t.categoria = <categoria> AND
t.cod_time = c.cod_time;
```

7. Trigger

Foram feitas três triggers, uma para verificar quando está havendo um **UPDATE** ou **INSERT** na tabela **atleta**, para garantir que a nova tupla a ser inserida ou a tupla que está sendo modificada tenha os dados dentro dos padrões estabelecidos, no caso, o atleta deve ter idade maior ou igual a 15 anos, ela pode ser observada na Figura 3.

```
1  -- Trigger para verificar os novos dados do atleta
2  CREATE OR REPLACE FUNCTION verifica_atleta() RETURNS trigger AS $verifica_idade$
3      BEGIN
4          -- Verifica se a idade a ser inserida é maior ou igual a 15
5          IF NEW.idade < 15 THEN
6              RAISE EXCEPTION 'A idade não pode ser menor que 15';
7          END IF;
8          RETURN NEW;
9      END;
10 $verifica_idade$ LANGUAGE plpgsql;
11
12 CREATE TRIGGER verifica_idade BEFORE INSERT OR UPDATE ON atleta
13     FOR EACH ROW EXECUTE PROCEDURE verifica_idade();
```

Figura 3 - Trigger de restrição de integridade na tabela atleta no campo idade

A segunda trigger foi feita para verificar quando está havendo um **UPDATE** ou **INSERT** na tabela **atleta**, para garantir que a nova tupla a ser inserida ou a tupla que está

sendo modificada tenha os dados dentro dos padrões estabelecidos, no caso, o atleta deve possuir desempenho entre 0 e 100, a trigger pode ser observada na Figura 4.

```
-- Trigger para verificar o desempenho
CREATE OR REPLACE FUNCTION verifica_desempenho() RETURNS trigger AS $verifica_desempenho$
    BEGIN
        -- Verifica se o desempenho a ser inserido está entre 0 e 100
        IF NEW.desempenho < 0 OR NEW.desempenho > 100 THEN
            RAISE EXCEPTION 'O desempenho deve estar entre 0 e 100';
        END IF;
        RETURN NEW;
    END
$verifica_desempenho$ LANGUAGE plpgsql;

CREATE TRIGGER verifica_desempenho BEFORE INSERT OR UPDATE ON atleta
    FOR EACH ROW EXECUTE PROCEDURE verifica_desempenho();
```

Figura 4 - Trigger de restrição de integridade na tabela atleta no campo desempenho

A terceira trigger foi feita para verificar quando está havendo um **`UPDATE`** ou **`INSERT`** na tabela **`telefone`**, para garantir que a nova tupla a ser inserida ou a tupla que está sendo modificada tenha os dados dentro dos padrões estabelecidos, no caso, o telefone deve ter o formato +pp(xx)nnnnn-nnnn, a trigger pode ser observada na Figura 5.

```
-- Trigger para verificar o formato do telefone
CREATE OR REPLACE FUNCTION verifica_telefone() RETURNS trigger AS $verifica_telefone$
    BEGIN
        -- Verifica se o telefone está no padrão desejado
        IF NOT NEW.telefone ~ '^\+[0-9]{2}\([0-9]{2}\)[0-9]{5}\-[0-9]{4}$' THEN
            RAISE EXCEPTION 'O telefone deve estar no formato +pp(dd)nnnnn-nnnn';
        END IF;
        RETURN NEW;
    END;
$verifica_telefone$ LANGUAGE plpgsql;

CREATE TRIGGER verifica_telefone BEFORE INSERT OR UPDATE ON telefone
    FOR EACH ROW EXECUTE PROCEDURE verifica_telefone();
```

Figura 5 - Trigger de restrição de integridade na tabela telefone

8. Considerações finais

Conforme o idealizado, na fase 3 deste projeto foi desenvolvido o projeto físico do banco de dados onde conceitualmente foi desenvolvida a Álgebra Relacional para cada consulta e então implementado em SQL, além, é claro, da implementação dos códigos que geram as tabelas do banco com suas devidas restrições de dados.

Para fins de testes também foram criadas rotinas de inserção de dados em cada tabela do Banco, sendo que esses dados possuem informações que fazem sentido, não somente qualquer cadeia de caracteres aleatórios.

Também foi necessária a utilização de Triggers para verificar se os dados estão nos padrões pré-estabelecidos e então manter a consistência dos dados das tabelas Atleta e Telefone quando houver um Insert ou Update.

As dificuldades desse processo giraram em torno da criação da terceira Trigger, onde é verificado o campo telefone na tabela Telefone, onde foi necessário utilizar expressões regulares, sendo essa a maior dificuldade do grupo.