TECNOLOGIA SETÚBAL
ESCOLA SUPERIOR POLITÉCNICO SETÚBAL

Bases de Dados

2023/2024

Licenciatura em Engª. Informática

Relatório Técnico (Época Exame)

Turma: 5
Horário de Laboratório: 10:30 Sexta-Feira
Docente: Luis Damas
201902549, Thiers de Mesquita

Índice

1. Sumário

O projeto visa desenvolver um sistema de gestão de vôlei que centraliza e organiza informações sobre jogadores, equipes, eventos, estádios, treinadores, países, patrocinadores e entidades organizadoras. O sistema será uma ferramenta essencial para clubes, federações, organizadores de torneios e patrocinadores, facilitando a administração e acompanhamento das atividades relacionadas ao vôlei.

O sistema de gestão de vôlei oferece cadastro detalhado de atletas e treinadores, gestão de equipes, planejamento de eventos, administração de estádios, gerenciamento de resultados, controle de patrocínios e supervisão por entidades organizadoras. Essas funções integradas garantem uma administração eficiente e transparente, beneficiando todos os envolvidos no esporte.

2. Definição do domínio do problema

O projeto desenvolvido é voltado para a gestão de vôlei, abrangendo a administração de jogadores, equipes, eventos, estádios, patrocinadores e entidades organizadoras. A seguir, detalhamos a lógica esperada para o funcionamento do sistema, incluindo os requisitos funcionais e de negócio necessários

ID	Descrição
R01	O sistema deverá permitir o cadastro detalhado de atletas e treinadores
R02	O sistema deverá possibilitar a criação e gerenciamento de equipes.
R03	O sistema deverá organizar eventos e partidas, agendando datas e locais
R04	O sistema deverá mostrar informações de estádios, incluindo capacidade.
RM01	O sistema deverá gerar relatórios analíticos para suporte à tomada de decisões.

3. Consultas

Atletas por Equipe

Descrição: Listar todos os atletas que pertencem a uma determinada equipe.
Exemplo: "Listar todos os atletas da equipe 'Vôlei Stars'."

Estádios Disponíveis

Descrição: Listar todos os estádios disponíveis em uma determinada data.
Exemplo: "Listar todos os estádios disponíveis para o dia 20 de setembro de 2024."

Treinadores por Especialidade

Descrição: Listar todos os treinadores que possuem uma especialidade específica.
Exemplo: "Listar todos os treinadores especializados no “Físico do Atleta."

Patrocínios Ativos

Descrição: Listar todos os patrocinadores ativos e suas contribuições financeiras.
Exemplo: "Listar todos os patrocinadores ativos e o valor das suas contribuições."

Essas são algumas consultas para obter informações precisas e úteis do sistema no gerenciamento do vôlei.

4. Modelo Entidade Relação

4.1 Levantamento das entidades, atributos e relacionamento

Entidades e Atributos:

FichaTecnica:
- **Chave Primária**: id_FichaTecnica
- **Atributos**: DataNascimento, Altura, Peso, Status

Pessoa:
- **Chave Primária**: IdPessoa
- **Atributos**: Nome, Cidade, Morada, Bairro

Atleta:
- **Chave Primária**: IdAtleta
- **Atributos**: Posicao, Numero da Camiseta, Capitão

Treinador:
- **Chave Primária**: IdTreinador
- **Atributos**: Anos de Experiencia, Especialidade

Equipe:
- **Chave Primária**: IdEquipa
- **Atributos**: NomeEquipa, IdadeDaEquipa

País:
- **Chave Primária**: IdPais
- **Atributos**: Nome

Estádio:
- **Chave Primária**: IdEstadio
- **Atributos**: Nome, Capacidade

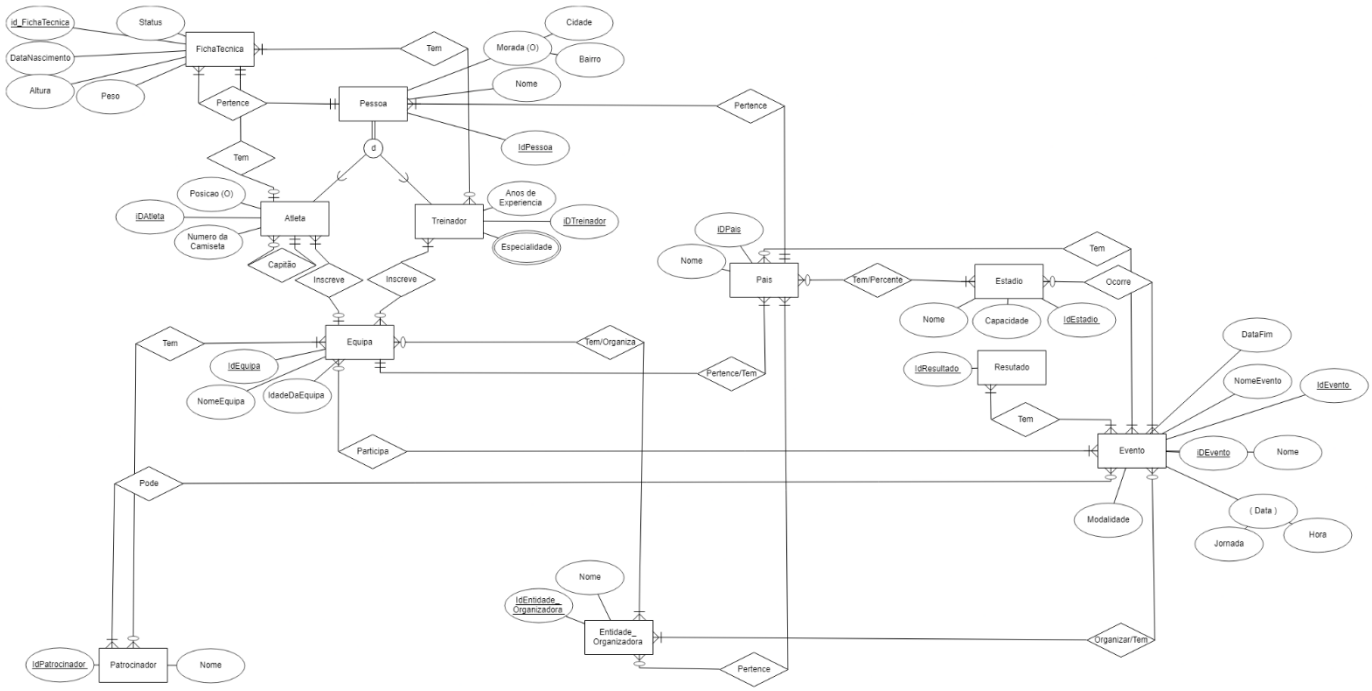
Evento:
- **Chave Primária**: IdEvento
- **Atributos**: NomeEvento, Nome, Data, Jornada, Hora, Modalidade

Patrocinador:
- **Chave Primária**: IdPatrocinador
- **Atributos**: Nome

Entidade Organizadora:
- **Chave Primária**: IdEntidade_Organizadora
- **Atributos**: Nome

Resultado:
- **Chave Primária**: IdResultado
- **Atributos:** Descricao

4.2 Diagrama do Modelo Entidade Relação



5. Modelo Relacional

5.1 Especificação do Modelo Relacional

Tabelas e Atributos:

FichaTecnica:

Chave Primária: id_FichaTecnica

Atributos: DataNascimento, Altura, Peso, Status

Chave Estrangeira: IdPessoa (referencia Pessoa)

Pessoa:

Chave Primária: IdPessoa

Atributos: Nome, Cidade, Morada, Bairro

Atleta:

Chave Primária: IdAtleta

Atributos: Posicao, Numero da Camiseta, Capitão

Chave Estrangeira: IdPessoa (referencia Pessoa)

Treinador:

Chave Primária: IdTreinador

Atributos: Anos de Experiencia, Especialidade

Chave Estrangeira: IdPessoa (referencia Pessoa)

Equipe:

Chave Primária: IdEquipa

Atributos: NomeEquipa, IdadeDaEquipa

Chave Estrangeira: IdPais (referencia Pais)

Pais:

Chave Primária: IdPais

Atributos: Nome

Estadio:

Chave Primária: IdEstadio

Atributos: Nome, Capacidade

Evento:

Chave Primária: IdEvento

Atributos: NomeEvento, DataFim, Jornada, Hora

Chave Estrangeira: IdEstadio (referencia Estadio), IdResultado (referencia Resultado), IdPais (referencia Pais)

Patrocinador:

Chave Primária: IdPatrocinador

Atributos: Nome

Entidade_Organizadora:

Chave Primária: IdEntidade_Organizadora

Atributos: Nome

Chave Estrangeira: IdPais (referencia Pais)

Resultado:

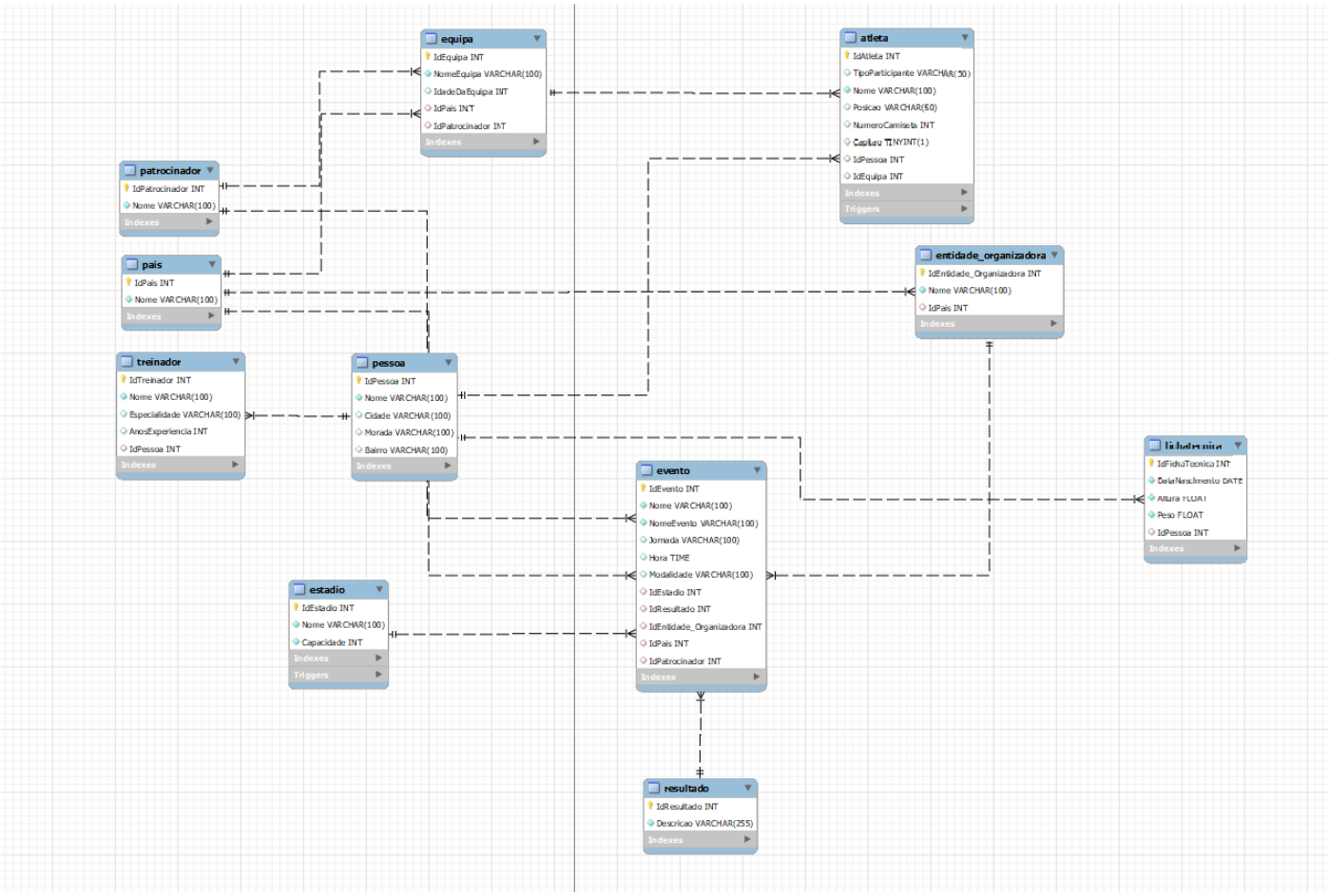
Chave Primária: IdResultado

Atributos: Descricao

Relacionamentos

FichaTecnica -> **Pessoa**: Cada ficha técnica pertence a uma pessoa.
Pessoa -> **Atleta** ou **Treinador**: Uma pessoa pode ser um atleta ou um treinador.
Atleta -> **Equipa**: Um atleta inscreve-se em uma equipe.
Treinador -> **Equipa**: Um treinador inscreve-se em uma equipe.
Equipa -> **Evento**: Uma equipe participa de eventos.
Pais -> **Equipa**: Um país tem várias equipes.
Pais -> **Evento**: Um país organiza eventos através de entidades organizadoras.
Estadio -> **Evento**: Um evento ocorre em um estádio.
Evento -> **Resultado**: Um evento tem um resultado associado.
Patrocinador -> **Equipa**: Um patrocinador pode patrocinar uma equipe.
Entidade_Organizadora -> **Pais**: Uma entidade organizadora pertence a um país e organiza eventos.

5.2 Diagrama do Modelo Relacional



6. Consultas à Base de Dados

6.1 Consultas de resposta ao enunciado

Comando SQL (SELECT)	**Descrição**	**Critério**
SELECT Nome FROM Atleta;	Lista de participantes.	***1.1***
SELECT NomeEvento, Jornada, Hora, Modalidade FROM Evento WHERE IdEntidade_Organizadora = 1;	Lista de eventos organizados pela entidade com ID 1.	**1.2**
SELECT NomeEquipa, Pais FROM EquipePorPais;	Lista de equipes por país.	**1.3**
SELECT Nome, Idade, Categoria FROM AtletasStatus;	Lista de atletas com idade e categoria.	**1.4**
SELECT IdEquipa, NomeEquipa, MediaAltura FROM MediaAlturaDeTodasEquipa;	Média de altura de todas as equipes.	**1.5**

6.2 Outras Consultas

Comando SQL (SELECT)	**Descrição**
SELECT * FROM Pais;	Verifica a inserção de dados na tabela País.
SELECT * FROM Entidade_Organizadora;	Verifica a inserção de dados na tabela Entidade Organizadora.
SELECT * FROM Estadio;	Verifica a inserção de dados na tabela Estádio.
SELECT * FROM Resultado;	Verifica a inserção de dados na tabela Resultado.
SELECT * FROM Pessoa;	Verifica a inserção de dados na tabela Pessoa.
SELECT * FROM FichaTecnica;	Verifica a inserção de dados na tabela Ficha Técnica.
SELECT * FROM Treinador	Verifica a inserção de dados na tabela Treinador.
SELECT * FROM Equipa;	Verifica a inserção de dados na tabela Equipa.
SELECT * FROM Atleta;	Verifica a inserção de dados na tabela Atleta.
SELECT * FROM Evento;	Verifica a inserção de dados na tabela Evento.
SELECT * FROM Patrocinador;	Verifica a inserção de dados na tabela Patrocinador.
SELECT MediaAlturaEquipa(1) AS MediaAlturaEquipa1;	Verifica a função armazenada para calcular a média de altura dos atletas de uma equipe específica.
SELECT TotalAtletasEquipa(1) AS TotalAtletasEquipa1;	Verifica a função armazenada para contar o total de atletas em uma equipe específica.
SELECT * FROM EquipePorPais;	Verifica a view que lista todas as equipes, indicando a qual país pertencem.
SELECT * FROM AtletasStatus;	Verifica a view que lista todos os atletas com suas idades e categorias (Junior/Senior).
SELECT * FROM MediaIdadeEquipa;	Verifica a view que calcula a média de idade dos atletas em cada equipe.
SELECT * FROM MediaAlturaDeTodasEquipa;	Verifica a view que calcula a média de altura dos atletas em todas as equipes.
SELECT * FROM Evento WHERE Nome = 'Prova de Teste';	Verifica se a prova foi criada corretamente.
SELECT * FROM Evento WHERE IdEvento = 1;	Verifica se a prova foi removida corretamente.
SELECT * FROM Evento WHERE IdEvento = 2;	Verifica se a prova foi removida corretamente com força.
SELECT * FROM Evento WHERE NomeEvento LIKE '%COPIA%';	Verifica se a prova foi clonada corretamente.

6.3 Views

Apresentar as views implementadas e descrever sucintamente o seu objetivo.

Nome View	Descrição
CREATE VIEW EquipePorPais AS *SELECT* *NomeEquipa,* *(CASE* *WHEN IdPais = 1 THEN 'Brasil'* *WHEN IdPais = 2 THEN 'Portugal'* *ELSE 'Outro'* *END) AS Pais* *FROM Equipa;*	Lista todas as equipes indicando a qual país pertencem.
CREATE VIEW AtletasStatus AS *SELECT* *Atleta.Nome,* *TIMESTAMPDIFF(YEAR, FichaTecnica.DataNascimento, CURDATE()) AS Idade,* *(CASE* *WHEN TIMESTAMPDIFF(YEAR, FichaTecnica.DataNascimento, CURDATE()) > 18 THEN 'Senior'* *ELSE 'Junior'* *END) AS Categoria* *FROM Atleta* *JOIN FichaTecnica ON Atleta.IdPessoa = FichaTecnica.IdPessoa;*	Lista todos os atletas com suas idades e categorias (Junior/Senior).
CREATE OR REPLACE VIEW MediaIdadeEquipa AS *SELECT* *Equipa.IdEquipa,* *Equipa.NomeEquipa,* *AVG(YEAR(CURDATE()) - YEAR(FichaTecnica.DataNascimento)) AS MediaIdade* *FROM Atleta* *JOIN FichaTecnica ON Atleta.IdPessoa = FichaTecnica.IdPessoa* *JOIN Equipa ON Atleta.IdEquipa = Equipa.IdEquipa* *GROUP BY Equipa.IdEquipa, Equipa.NomeEquipa;*	Calcula a média de idade dos atletas em cada equipe.
CREATE VIEW MediaAlturaDeTodasEquipa AS *SELECT* *Equipa.IdEquipa,* *Equipa.NomeEquipa,* *MediaAlturaEquipa(Equipa.IdEquipa) AS MediaAltura* *FROM* *Equipa;*	Calcula a média de altura dos atletas em todas as equipes.

7. Programação

7.1 Stored Procedures

SP implementado	SP enunciado
CREATE PROCEDURE sp_criar_prova(*IN pNome VARCHAR(100),* *IN pNomeEvento VARCHAR(100),* *IN pJornada VARCHAR(100),* *IN pHora TIME,* *IN pModalidade VARCHAR(100),* *IN pIdEstadio INT,* *IN pIdResultado INT,* *IN pIdEntidade_Organizadora INT,* *IN pIdPais INT* *)* *BEGIN* *INSERT INTO Evento (Nome, NomeEvento, Jornada, Hora, Modalidade, IdEstadio, IdResultado, IdEntidade_Organizadora, IdPais)* *VALUES (pNome, pNomeEvento, pJornada, pHora, pModalidade, pIdEstadio, pIdResultado, pIdEntidade_Organizadora, pIdPais);* *END*	Cria um evento com as informações fornecidas.
CREATE PROCEDURE sp_adicionar_participante(*IN pIdEvento INT,* *IN pIdAtleta INT* *)* *BEGIN* *INSERT INTO ProvaParticipante (IdEvento, IdAtleta)* *VALUES (pIdEvento, pIdAtleta);* *END*	Adiciona um atleta a um evento/prova específico.
CREATE PROCEDURE sp_registar_resultado(*IN pIdEvento INT,* *IN pIdAtleta INT,* *IN pIdResultado INT* *)* *BEGIN* *UPDATE ProvaParticipante* *SET IdResultado = pIdResultado* *WHERE IdEvento = pIdEvento AND IdAtleta = pIdAtleta;* *END*	Registra o resultado de um atleta em um evento específico.

CREATE PROCEDURE sp_remover_prova(*IN pIdEvento INT,* *IN pForce BOOLEAN* *)* *BEGIN* *DECLARE resultCount INT;* *SELECT COUNT(*) INTO resultCount FROM ProvaParticipante WHERE IdEvento = pIdEvento;* *IF resultCount = 0 OR pForce = TRUE THEN* *DELETE FROM Evento WHERE IdEvento = pIdEvento;* *ELSE* *SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Não é possível remover o Evento porque existem resultados associados.';* *END IF;* *END*	Remove um evento, podendo forçar a remoção mesmo que existam resultados associados.
CREATE PROCEDURE sp_clonar_prova(*IN pIdEvento INT* *)* *BEGIN* *DECLARE pNomeEvento VARCHAR(100);* *DECLARE pJornada VARCHAR(100);* *DECLARE pHora TIME;* *DECLARE pModalidade VARCHAR(100);* *DECLARE pIdEstadio INT;* *DECLARE pIdResultado INT;* *DECLARE pIdEntidade_Organizadora INT;* *DECLARE pIdPais INT;* *SELECT NomeEvento, Jornada, Hora, Modalidade, IdEstadio, IdResultado, IdEntidade_Organizadora, IdPais* *INTO pNomeEvento, pJornada, pHora, pModalidade, pIdEstadio, pIdResultado, pIdEntidade_Organizadora, pIdPais* *FROM Evento WHERE IdEvento = pIdEvento;* *SET pNomeEvento = CONCAT(pNomeEvento, ' COPIA do Evento');* *INSERT INTO Evento (Nome, NomeEvento, Jornada, Hora, Modalidade, IdEstadio, IdResultado, IdEntidade_Organizadora, IdPais)* *VALUES ('Evento Clonado', pNomeEvento, pJornada, pHora, pModalidade, pIdEstadio, pIdResultado, pIdEntidade_Organizadora, pIdPais);* *END*	Clona um evento existente, criando uma cópia com os mesmos detalhes.
CREATE PROCEDURE ListarAtletasPorEquipa(equipaId INT) *BEGIN* *DECLARE done INT DEFAULT 0;* *DECLARE atletaNome VARCHAR(100);* *DECLARE atletaCursor CURSOR FOR SELECT Nome FROM Atleta WHERE IdEquipa = equipaId;*	

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1; *OPEN atletaCursor;* *read_loop: LOOP* *FETCH atletaCursor INTO atletaNome;* *IF done THEN* *LEAVE read_loop;* *END IF;* *SELECT atletaNome;* *END LOOP;* *CLOSE atletaCursor;* *END*	Lista todos os atletas de uma equipe específica.
CREATE PROCEDURE MostrarMediaAlturaEquipa(equipaId INT) *BEGIN* *DECLARE media FLOAT;* *SET media = MediaAlturaEquipa(equipaId);* *SELECT media AS MediaAltura;* *END*	Mostra a média de altura dos atletas de uma equipe, chamando a função
CREATE PROCEDURE ListarEventosPorEntidade(entidadeId INT) *BEGIN* *SELECT NomeEvento, Jornada, Hora, Modalidade* *FROM Evento* *WHERE IdEntidade_Organizadora = entidadeId;* *END*	Lista todos os eventos organizados por uma entidade específica.
CREATE PROCEDURE InserirNovoEstadio(IN nome VARCHAR(100), IN capacidade INT) *BEGIN* *IF capacidade < 0 THEN* *SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'A capacidade do estádio não pode ser negativa';* *ELSE* *INSERT INTO Estadio (Nome, Capacidade) VALUES (nome, capacidade);* *END IF;* *END*	Insere um novo estádio, verificando se a capacidade não é negativa.
CREATE PROCEDURE InserirNovoTreinador(IN nome VARCHAR(100), IN especialidade VARCHAR(100), IN anos_experiencia INT, IN id_pessoa INT) *BEGIN* *DECLARE EXIT HANDLER FOR SQLEXCEPTION* *BEGIN* *ROLLBACK;* *SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Erro ao inserir novo treinador.';* *END;* *START TRANSACTION;* *INSERT INTO Treinador (Nome, Especialidade, AnosExperiencia, IdPessoa) VALUES (nome, especialidade, anos_experiencia, id_pessoa); COMMIT;END*	Tenta inserir um novo treinador e retorna uma mensagem de erro em caso de falha.

CREATE FUNCTION MediaAlturaEquipa(id_equipa INT) RETURNS FLOAT *READS SQL DATA* *BEGIN* *DECLARE media_altura FLOAT;* *SELECT AVG(FichaTecnica.Altura) INTO media_altura* *FROM Atleta* *JOIN FichaTecnica ON Atleta.IdPessoa = FichaTecnica.IdPessoa* *WHERE Atleta.IdEquipa = id_equipa;* *RETURN media_altura;* *END*	Calcula a média de altura dos atletas de uma equipe específica.
CREATE FUNCTION TotalAtletasEquipa(id_equipa INT) RETURNS INT *READS SQL DATA* *BEGIN* *DECLARE total_atletas INT;* *SELECT COUNT(*) INTO total_atletas* *FROM Atleta* *WHERE Atleta.IdEquipa = id_equipa;* *RETURN total_atletas;* *END*	Retorna o número total de atletas em uma equipe específica.

7.2 Functions

FN	**Descrição**
CREATE FUNCTION MediaAlturaEquipa(id_equipa INT) RETURNS FLOAT READS SQL DATA BEGIN DECLARE media_altura FLOAT; SELECT AVG(FichaTecnica.Altura) INTO media_altura FROM Atleta JOIN FichaTecnica ON Atleta.IdPessoa = FichaTecnica.IdPessoa WHERE Atleta.IdEquipa = id_equipa; RETURN media_altura; END	Calcula a média de altura dos atletas de uma equipe específica.
CREATE FUNCTION TotalAtletasEquipa(id_equipa INT) RETURNS INT *READS SQL DATA* *BEGIN* *DECLARE total_atletas INT;* *SELECT COUNT(*) INTO total_atletas* *FROM Atleta* *WHERE Atleta.IdEquipa = id_equipa;* *RETURN total_atletas;* *END*	Retorna o número total de atletas em uma equipe específica.

7.3 Triggers

7.3.1 Monitorização de falhas

Trigger	Trigger enunciado
CREATE TRIGGER BeforeInsertAtleta *BEFORE INSERT ON Atleta* *FOR EACH ROW* *BEGIN* *DECLARE pessoa_exists INT;* *-- Verifica se a pessoa existe na tabela Pessoa* *SELECT COUNT(*) INTO pessoa_exists* *FROM Pessoa* *WHERE IdPessoa = NEW.IdPessoa;* *-- Se a pessoa não existe, lançar um erro* *IF pessoa_exists = 0 THEN* *SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Pessoa referenciada não existe na tabela Pessoa';* *END IF;* *END*	Verifica se a pessoa já existe na tabela Pessoa antes de inserir um novo atleta. Se a pessoa não existir, lança um erro.
CREATE TRIGGER BeforeInsertEstadio *BEFORE INSERT ON Estadio* *FOR EACH ROW* *BEGIN* *IF NEW.Capacidade < 0 THEN* *SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'A capacidade do estádio não pode ser negativa';* *END IF;* *END*	Verifica se a capacidade de um novo estádio não é negativa antes de inserir. Se a capacidade for negativa, lança um erro.

8. Conclusões

O projeto envolveu a criação de várias stored procedures, functions, triggers e views para um sistema de gestão de Voleibol. O objetivo principal foi implementar funcionalidades que permitissem a gestão de dados relacionados a equipes, atletas, eventos e patrocinadores de maneira eficiente.

Ao longo do desenvolvimento, enfrentei alguns desafios significativos. A criação das stored procedures exigiu várias tentativas para garantir que todos os dados fossem corretamente manipulados e que as restrições de integridade fossem respeitadas. Além da criação de triggers. A escalabilidade pode se tornar um problema à medida que a quantidade de dados cresce, exigindo a otimização contínua das Queries e Stored Procedures para manter a performance.

O trabalho envolveu uma enorme dificuldade na criação do Modelo Entidade-Relacionamento (MER). A complexidade do sistema, com suas múltiplas entidades e relações, exigiu um planejamento cuidadoso para garantir que todas as interações fossem capturadas corretamente. A definição das chaves primárias e estrangeiras, bem como a criação de relações adequadas entre as tabelas, foram tarefas que demandaram muito esforço gigante.

Este projeto demonstrou a importância de um planejamento cuidadoso e testes exaustivos para garantir a funcionalidade e integridade de um sistema de gestão de dados complexo. Com melhorias contínuas e manutenção adequada, ele pode servir como uma base sólida para futuras expansões e funcionalidades adicionais.