Universidade do Minho
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Lectivo de 2024/2025

Jogos Olímpicos

Alexis Correia, João Fonseca, Ricardo Vilaça

Janeiro, 2025

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Jogos Olímpicos

Alexis Correia, João Fonseca, Ricardo Vilaça

Janeiro, 2025

Resumo

Este projeto tem como objetivo desenvolver um Sistema de Banco de Dados (SBD) relacional para a "Gestão de Atletas, Equipas e Resultados em uma Competição Desportiva Multi-modalidade", inspirado em eventos como os Jogos Olímpicos. O sistema foi projetado para organizar e facilitar o acesso a informações detalhadas sobre atletas, equipes, modalidades esportivas e resultados de competições, garantindo eficiência e integridade na gestão dos dados.

A elaboração do banco de dados envolveu várias etapas do ciclo de vida de desenvolvimento de sistemas, incluindo análise de requisitos, modelagem conceitual, modelação física, implementação e validação. Utilizando ferramentas de modelagem como MySQL Workbench, foram construídos diagramas ER (Entity-Relationship) e representações do modelo de dados para estruturar e documentar as entidades, relacionamentos e atributos necessários.

O sistema permite registar e consultar informações essenciais para os diferentes perfis de utilizadores, como gestores, treinadores e analistas esportivos, que podem, assim, monitorar o desempenho de atletas e equipes de forma centralizada e confiável. A implementação do SBD em um Sistema de Gerenciamento de Banco de Dados Relacional (SGBDR) garante segurança, escalabilidade e suporte a consultas complexas, oferecendo vantagens operacionais significativas, como a minimização de redundâncias e a melhoria na precisão dos dados.

A normalização do banco de dados foi verificada para assegurar a consistência e otimização das operações, garantindo um sistema robusto e eficiente para a gestão de eventos esportivos de grande escala.

Área de Aplicação: Eventos desportivos de grande escala.

Palavras-Chave: Base de Dados (BD), Entidade, Atributo, Relacionamento, Modelo Conceptual, Modelo Lógico, Modelo Físico, *MySQL, SQL.*

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

1.1 Contextualização

Em grandes eventos esportivos como as Olimpíadas, gerenciar as comunicações sobre atletas, equipes, mudanças e resultados é uma parte difícil, mas importante do evento. Milhares de jogadores de vários desportos participam nestas competições, que exigem que os dados sejam processados, classificados e calculados para serem precisos e úteis para equipas, especialistas, treinadores e observadores. Além disso, a necessidade de segurança e proteção das informações é importante, pois uma grande quantidade é gerada durante o evento. Este projeto apresenta o desenvolvimento de um sistema de informação (SBD) que pode gerenciar as informações mais importantes sobre diferentes desafios, apoiando decisões e monitorando resultados ao longo do tempo.

1.2 Motivação

A principal razão para o desenvolvimento deste SBD é que as competições desportivas complexas requerem sistemas de gestão de dados fortes e centralizados, onde dados como resultados, dados de equipas e jogadores devem ser processados e acedidos de forma eficiente. Muitos sistemas atualmente utilizados em grandes eventos desportivos sofrem de falta de dados, dificuldade em obter informações rapidamente e falta de fiabilidade. A criação de um SBD virtual pode resolver esse problema, melhorar a precisão do gerenciamento de dados e ajudar administradores e professores a planejar com mais eficiência. Com um banco de dados relacional padronizado, a sobreposição é reduzida e os detalhes e análises de atletas e equipes são suportados.

1.3 Objetivo

O objetivo do projeto é criar uma base de dados relacional para a gestão de informação de qualidade sobre jogadores, equipas, competições desportivas e seus resultados. O sistema deve permitir a entrada, consulta e modificação instantânea de dados, melhorando a gestão de

eventos e garantindo a precisão e integridade dos dados ao longo da campanha. Também visa fornecer informações detalhadas e análises do desempenho dos atletas e das equipes, acompanhar a distribuição de prêmios entre os países e facilitar a discussão pública de testes competitivos e estatísticos.

1.4 Viabilidade

O estudo de viabilidade deste projeto mostra que a utilização de sistemas de informação (SBD) para gestão de competições desportivas é boa e útil. O SBD pode ser utilizado para coordenar e melhorar muitas informações sobre jogadores, times, partidas e resultados, tornando-as mais precisas e acessíveis aos usuários. Os benefícios incluem custos mais baixos, informações mais detalhadas e a facilidade de criação de relatórios abrangentes que apoiam uma melhor tomada de decisões e monitoramento de desempenho. Os recursos utilizados economizam tempo e reduzem erros na verificação e atualização de dados, já que o sistema não é eficiente devido à exposição manual e erros. O processo de conclusão do projeto inclui etapas: escrever código, criar documentos conceituais e conceituais, planejar e testar e gerenciar dados. Isso torna o sistema eficiente e torna a concorrência justa e eficiente, atendendo às necessidades dos diferentes usuários.

2. Modelo Conceptual

Tudo relacionado ao levantamento e análise de requesitos poder ser visualizado na seguinte ligação: DocRequesitos.

No que toca ao modelo conceptual, o grupo decidiu elaborá-lo utilizando o software ***brModelo*** devido à sua interface intuitiva e funcionalidades específicas para a criação de diagramas Entidade-Relacionamento (ER). Essa ferramenta permitiu representar de forma clara as entidades, atributos e relacionamentos identificados durante a análise de requisitos, facilitando o entendimento geral do sistema pelos integrantes do grupo. A escolha foi motivada também pela capacidade do ***brModelo*** de garantir a padronização e a organização do diagrama, otimizando o processo de modelagem e assegurando que todos os aspectos conceituais fossem contemplados antes da transição para as etapas de modelagem lógica e física.

2.1 Identificação e caracterização das entidades

Entidade	Descrição	Ocorrência
Competição	Uma competição desportiva; possui um nome, local e datas (início e fim); Com diferentes esportes/modalidades; Composta por diferentes delegações e equipas.	Pode haver diferentes competições no BD.
Atleta	Pessoa que competirá nos diversos eventos; cada atleta compete em 1 ou mais modalidades de um determinado esporte; informações como nome, e descrição física (altura, peso, género, etc)	cada atleta compõe apenas 1 equipa e (consequentemente) 1 delegação.
Equipa	Composta por vários atletas; cada equipa treina para um esporte; porém, pode haver mais de uma modalidade;	Várias equipas compõem uma delegação; cada equipa só pode estar associada a uma delegação.
Delegação	Composta por, pelo menos, 1 equipa; cada delegação representa um país que participará na competição;	Várias delegações compõem a competição.
Esporte	Um esporte possui nome; os esportes podem ser individuais ou não;	Diferentes esportes são realizados durante a competição.

Modalidade	Cada esporte pode ter diferentes modalidades (categorias de peso, divisão de género, individual ou em equipe, etc); possui uma descrição;	Existem diferentes modalidades para cada um dos múltiplos esportes.
Evento	Eventos são os momentos em que os atletas disputam; ocorrem num determinado dia e hora e local (previamente definidos);	Acontece muitos eventos; cada evento refere-se a uma modalidade de um esporte.
Resultado	Cada atleta possui um resultado no evento em que ele participou; possui posição e se ele se classificou ou não; pode possuir outros dados como tempo ou pontos;	Existe apenas um resultado para cada atleta em cada evento; logo existem múltiplos resultados por atletas/eventos.
Funcionário	São as pessoas que trabalham para a competição; eles possuem nome e tipo (função que eles exercem)	A competição é organizada/realizada por inúmeros funcionários.
Tipo	Descreve os diferentes tipos de trabalhos que os funcionários podem realizar;	Há diferentes cargos, mas cada funcionário só possui um cargo (tipo),
Treinador	Pessoa responsável por treinar os atletas; cada treinador faz parte de apenas uma equipa (consequentemente, apenas uma delegação); possui um nome;	Existe pelo menos um treinador por equipa; logo há múltiplos treinadores por delegação;

Tabela 1 – Caracterização das entidades

2.2 Identificação e caracterização dos relacionamentos

Nome Entidade A	Multiplicidade	Relacionamento	Nome Entidade B	Multiplicidade
Delegação	N	participa	Competição	1
Competição	1	inclui	Desporto	N
Funcionário	N	do	Tipo	1
Funcionário	N	trabalha	Competição	1
Atleta	N	realiza	Modalidade	N
Atleta	N	forma	Equipe	1
Equipe	N	constitui	Delegação	1
Equipe	N	pratica	Desporto	1
Modalidade	N	do	Desporto	1
Evento	N	da	Modalidade	1
Treinador	N	treina	Equipe	1
Atleta	1	obteve	Resultado	N
Resultado	N	no	Evento	1

Tabela 2 – Caracterização dos relacionamentos

2.3 Identificação e caracterização da associação dos atributos com as entidades

Nome Entidade	Atributo	Descrição Atributo	Tipo	Nulo (S/N)	Chave Primária (S/N)	Auto-incremento(S/N)
Competição	idCompeticao	nº de identificação da comp.	INT	N	S	S
Competição	nome	nome da competição	VARCHAR(50)	N	N	N
Competição	dataInicio	data do começo da competição	DATE	N	N	N
Competição	dataFim	data do encerramento da comp.	DATE	N	N	N
Competição	local	local onde se realizará a comp.	VARCHAR(50)	N	N	N
Atleta	idAtleta	nº de identificação do atleta	INT	N	S	N
Atleta	nome	nome do atleta	VARCHAR(50)	N	N	N
Atleta	idade	idade do atleta	INT	N	N	N
Atleta	genero	género do atleta (Masc / Fem)	VARCHAR(1)	N	N	N
Atleta	peso	peso do atleta em Kg	DECIMAL(5,2)	S	N	N
Atleta	altura	altura do atleta em cm	DECIMAL(5,2)	S	N	N
Equipa	idEquipa	nº de identificação da equipa	INT	N	S	S
Delegação	idDelegacao	nº de identificação da delegação	INT	N	S	S
Delegação	pais	país de origem da delegação	VARCHAR(50)	N	N	N
Esporte	idEsporte	nº de identificação do esporte	INT	N	S	N
Esporte	nome	nome do esporte	VARCHAR(50)	N	N	N
Modalidade	idModalidade	nº de id da modalidade	INT	N	S	N
Modalidade	descricao	descrição da modalidade	VARCHAR(100)	N	N	N
Evento	idEvento	nº de identificação do evento	INT	N	S	N
Evento	dataHora	data e hora da realização do evento	DATETIME	N	N	N
Evento	local	local onde se realizará o evento	VARCHAR(50)	N	N	N
Resultado	idResultado	id do resultado	INT	N	S	N

Resultado	posicao	posição final do atleta	INT	N	N	N
Resultado	pontos	pontos marcados pelo atleta no evento	INT	S	N	N
Resultado	tempo	tempo marcado pelo atleta	TIME	S	N	N
Resultado	classificado	diz se o atleta se classificou ou não	TINYINT	N	N	N
Funcionário	idFunc	nº de identificação do funcionário	INT	N	S	S
Funcionário	nome	nome do funcionário	VARCHAR(50)	N	N	N
Tipo	idTipo	nº de identificação do tipo	INT	N	S	S
Tipo	descricao	descrição do tipo	VARCHAR(20)	N	N	N
Treinador	idTreinador	nº de identificação do treinador	INT	N	S	N
Treinador	nome	nome do treinador	VARCHAR(50)	N	N	N

Tabela 3 – Caracterização dos atributos

2.4 Apresentação do diagrama ER

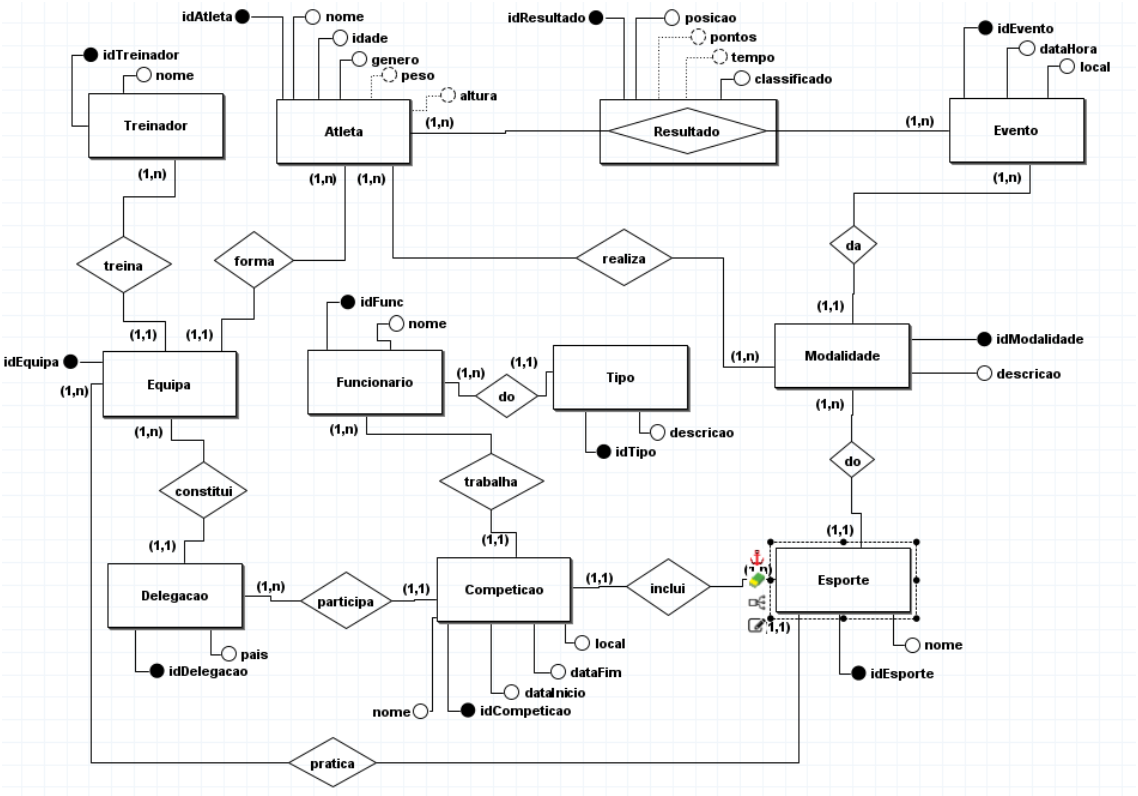

Figura 1 – Ilustração do modelo conceptual

3. Modelo Lógico

Para se dar conversão do modelo de dados conceptual para o modelo de dados lógico nós optamos por usar o *MySQL* devido à sua capacidade de transformar estruturas lógicas em tabelas, atributos, chaves primárias e chaves estrangeiras, garantindo integridade e consistência dos dados. Além disso, a interface *MySQL* e suas ferramentas de suporte, como *MySQL Workbench*, suportam a criação e validação de modelos lógicos, permitindo que a equipe atenda aos requisitos dos padrões e simule melhor o design do banco de dados antes de sua implementação.

3.1 Construção e validação do modelo de dados lógico

Chave Primária	Atributos	Tipo	Chave Estrangeira
idTreinador	idTreinador	INT	idEquipa
	Nome	VARCHAR(45)	
	idEquipa	INT	

Tabela 4 – Tabela Treinador

Chave Primária	Atributos	Tipo	Chave Estrangeira
idEquipa	idEquipa	INT	
	idEsporte	INT	idEsporte
	idDelegacao	INT	idDelegacao

Tabela 5 – Tabela Equipa

Chave Primária	Atributos	Tipo	Chave Estrangeira
idDelegacão	idDelegacão	INT	idCompeticão
	país	VARCHAR(45)	
	idCompeticão	INT	

Tabela 6 – Tabela Delegacão

Chave Primária	Atributos	Tipo	Chave Estrangeira
idAtleta	idAtleta	INT	idEquipa
	nome	VARCHAR(45)	
	idade	INT	
	genero	VARCHAR(1)	
	peso	DECIMAL(5,2)	
	altura	DECIMAL(3,2)	
	idEquipa	INT	

Tabela 7 – Tabela Atleta

Chave Primária	Atributos	Tipo	Chave Estrangeira
idFuncionario	idFuncionario	INT	idTipo idCompeticao
	nome	VARCHAR(45)	
	idTipo	INT	
	idCompeticao	INT	

Tabela 8 – Tabela Funcionario

Chave Primária	Atributos	Tipo
idCompeticao	idCompeticao	INT
	dataInicio	DATE
	dataFim	DATE
	local	VARCHAR(45)
	nome	VARCHAR(45)

Tabela 9 – Tabela Competicao

Chave Primária	Atributos	Tipo	Chave Estrangeira
idResultado	idRealiza	INT	idAtleta idEvento
	posição	INT	
	pontos	INT	
	tempo	TIME	
	classificado	TINYINT	
	idAtleta	INT	
	idEvento	INT	

Tabela 10 – Tabela Resultado

Atributos	Tipo	Chave Estrangeira
idAtleta	INT	idAtleta
idMod	INT	idMod

Tabela 11 – Tabela Realiza

Chave Primária	Atributos	Tipo
idTipo	idTipo	INT
	descricao	VARCHAR(45)

Tabela 12 – Tabela Tipo

Chave Primária	Atributos	Tipo	Chave Estrangeira
idEvento	idEvento	INT	idMod
	dataHora	DATETIME	
	local	VARCHAR(45)	

	idMod	INT	

Tabela 13 – Tabela Evento

Chave Primária	Atributos	Tipo	Chave Estrangeira
idModalidade	idModalidade	INT	idEsporte
	descricao	VARCHAR(100)	
	idEsporte	INT	

Tabela 14 – Tabela Modalidade

Chave Primária	Atributos	Tipo	Chave Estrangeira
idEsporte	idEsporte	INT	idCompeticao
	nome	VARCHAR(45)	
	idCompeticao	INT	

Tabela 15 – Tabela Esporte

3.2 Apresentação do modelo lógico

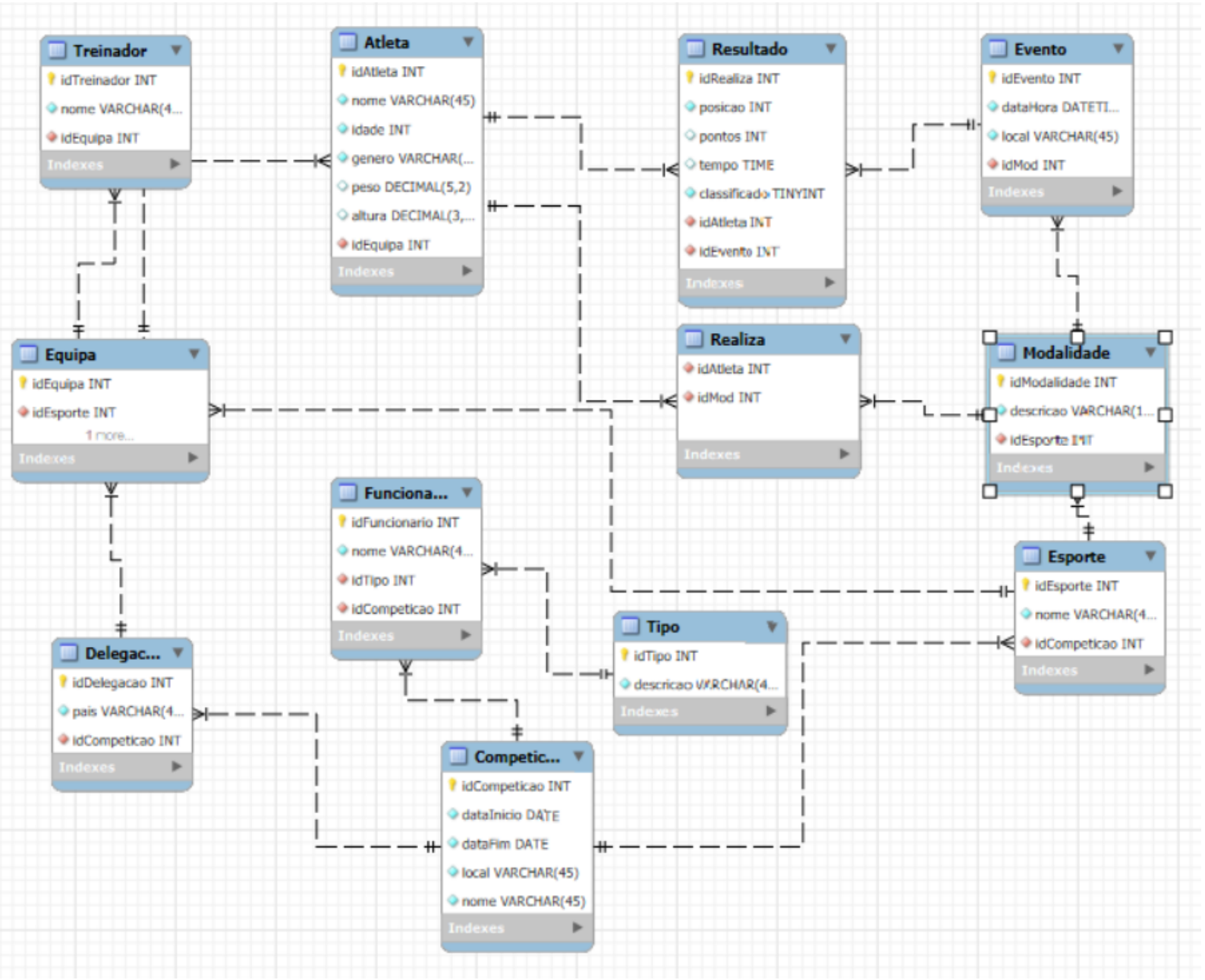

Figura 2 – Ilustração do modelo lógico

Modelo Físico

4.1 Apresentação e explicação da base de dados implementada

De forma a iniciar o desenvolvimento do sistema de gestão de base de dados, decidimos criar a implementação mySQL de cada tabela apresentada anteriormente no modelo lógico.

Dá-se início ao processo com duas instruções SQL essenciais: a primeira é responsável pela criação da base de dados com o nome atribuído EventoDesportivo, a segunda indica ao sistema qual a base de dados que deve ser utilizada para as operações executadas.

```
CREATE DATABASE IF NOT EXISTS  EventoDesportivo;
USE EventoDesportivo;
```

Figura 3 – Instrução SQL para criar e utilizar a base de dados

A tabela Competição armazena informações essenciais sobre os eventos competitivos. A coluna idCompeticao é do tipo “INT” e serve como “PRIMARY KEY”, sendo “NOT NULL” para garantir que cada competição possua um identificador único. Além disso, utiliza “AUTO_INCREMENT” para que o sistema atribua automaticamente um identificador sequencial.

“Data_Inicio” é um campo do tipo “DATE” e “NOT NULL”, que regista a data de início de cada competição. De forma complementar, “Data_Fim” é também do tipo “DATE” e “NOT NULL”, indicando a data de término do evento. A coluna “Local” é um “VARCHAR(45)” e “NOT NULL”, permitindo armazenar até 45 caracteres com a descrição do local onde a competição ocorre. Por fim, a coluna “Nome” é um “VARCHAR(45)” e também “NOT NULL”, garantindo que cada competição tenha um nome descritivo de até 45 caracteres.

Figura 4 – Tabela Lógica Competição

```
CREATE TABLE IF NOT EXISTS Competicao (
  idCompeticao INT NOT NULL,
  dataInicio DATE NOT NULL,
  dataFim DATE NOT NULL,
  localidade VARCHAR(45) NOT NULL,
  nome VARCHAR(45) NOT NULL,
  PRIMARY KEY (idCompeticao)
);
```

Figura 5 - Código SQL para criar tabela Competicao

A tabela Tipo regista os diferentes tipos ou funções associadas aos funcionários. A coluna idTipo é do tipo “INT”, definida como “PRIMARY KEY” e “NOT NULL”, garantindo identificadores únicos para cada tipo.

A coluna descricao é um “VARCHAR(45)” e “NOT NULL”, permitindo armazenar descrições com até 45 caracteres. Esta tabela é referenciada pela tabela Funcionário através da coluna idTipo, assegurando que cada funcionário esteja associado a um tipo específico.

Figura 6 – Tabela Lógica Tipo

```sql
CREATE TABLE IF NOT EXISTS Tipo (
  idTipo INT NOT NULL,
  descricao VARCHAR(45) NOT NULL,
  PRIMARY KEY (idTipo)
);
```

Figura 7 – Código SQL para criar a tabela Tipo

A tabela Funcionário armazena informações básicas sobre os colaboradores. A coluna idFuncionario é do tipo “INT”, definida como “PRIMARY KEY” e “NOT NULL”, garantindo um identificador único para cada funcionário. Nome é um “VARCHAR(45)” e “NOT NULL”, permitindo armazenar até 45 caracteres.

A coluna idTipo é um “INT” e “NOT NULL”, funcionando como uma chave estrangeira que referencia a tabela Tipo, associando o funcionário ao respetivo tipo. Da mesma forma, idCompeticao é um “INT” e “NOT NULL”, servindo como chave estrangeira que referencia a tabela Competição, indicando a qual competição o funcionário está associado.

Figura 8 – Tabela Lógica Funcionário

```sql
CREATE TABLE IF NOT EXISTS Funcionario (
  idFuncionario INT NOT NULL,
  nome VARCHAR(45) NOT NULL,
  idTipo INT NOT NULL,
  idCompeticao INT NOT NULL,
  PRIMARY KEY (idFuncionario),
  FOREIGN KEY (idTipo) REFERENCES Tipo (idTipo),
  FOREIGN KEY (idCompeticao) REFERENCES Competicao (idCompeticao)
);
```

Figura 9 - Código SQL para criar tabela Funcionário

A tabela Esporte regista os desportos associados a cada competição. A coluna idEsporte é do tipo “INT”, definida como “PRIMARY KEY” e “NOT NULL”, garantindo um identificador único para cada desporto.

A coluna nome é um “VARCHAR(45)” e “NOT NULL”, permitindo armazenar nomes de desportos com até 45 caracteres. Por fim, idCompeticao é um “INT” e “NOT NULL”, funcionando como uma chave estrangeira que referencia a tabela Competição, associando cada desporto a uma competição específica no sistema.

Figura 10 – Tabela Lógica Esporte

```sql
CREATE TABLE IF NOT EXISTS Esporte (
  idEsporte INT NOT NULL,
  nome VARCHAR(45) NOT NULL,
  idCompeticao INT NOT NULL,
  PRIMARY KEY (idEsporte),
  FOREIGN KEY (idCompeticao) REFERENCES Competicao (idCompeticao)
);
```

Figura 11 – Código SQL para criar tabela Esporte

A tabela Modalidade regista as diferentes modalidades associadas a cada desporto. A coluna idModalidade é do tipo “INT”, definida como “PRIMARY KEY” e “NOT NULL”, garantindo um identificador único para cada modalidade.

A coluna descricao é um “VARCHAR(100)” e “NOT NULL”, permitindo armazenar descrições detalhadas das modalidades com até 100 caracteres. Já a coluna idEsporte é um “INT” e “NOT NULL”, funcionando como uma chave estrangeira que referencia a tabela Esporte, estabelecendo a relação entre cada modalidade e o desporto correspondente no sistema.

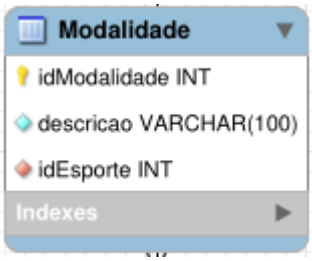

Figura 12 – Tabela Lógica Modalidade

```sql
CREATE TABLE IF NOT EXISTS Modalidade (
  idModalidade INT NOT NULL,
  descricao VARCHAR(100) NOT NULL,
  idEsporte INT NOT NULL,
  PRIMARY KEY (idModalidade),
  FOREIGN KEY (idEsporte) REFERENCES Esporte (idEsporte)
);
```

Figura 13 – Código SQL para criar tabela Modalidade

A tabela Delegacao regista as delegações associadas a cada competição. A coluna idDelegacao é do tipo “INT”, definida como “PRIMARY KEY” e “NOT NULL”, garantindo um identificador único para cada delegação.

A coluna país é um “VARCHAR(45)” e “NOT NULL”, permitindo armazenar o nome do país da delegação com até 45 caracteres. A coluna idCompeticao é do tipo “INT” e “NOT NULL”, funcionando como uma chave estrangeira que referencia a tabela Competicao, associando cada delegação a uma competição específica no sistema.

Figura 14 – Tabela Lógica Delegação

```sql
CREATE TABLE IF NOT EXISTS Delegacao (
  idDelegacao INT NOT NULL,
  pais VARCHAR(45) NOT NULL,
  idCompeticao INT NOT NULL,
  PRIMARY KEY (idDelegacao),
  FOREIGN KEY (idCompeticao) REFERENCES Competicao (idCompeticao)
);
```

Figura 15 – Código SQL para criar tabela Delegação

A tabela Equipa regista as equipas associadas aos desportos e às delegações. A coluna idEquipa é do tipo “INT”, definida como “PRIMARY KEY” e “NOT NULL”, garantindo um identificador único para cada equipa. A coluna idEsporte é do tipo “INT” e “NOT NULL”, funcionando como uma chave estrangeira que referencia a tabela Esporte, associando cada equipa a um desporto específico. A coluna idDelegacao é do tipo “INT” e “NOT NULL”, funcionando como uma chave estrangeira que referencia a tabela Delegacao, associando cada equipa a um país.

Figura 16 – Tabela Lógica Equipa

```sql
CREATE TABLE IF NOT EXISTS Equipa (
  idEquipa INT NOT NULL,
  idEsporte INT NOT NULL,
  idDelegacao INT NOT NULL,
  PRIMARY KEY (idEquipa),
  FOREIGN KEY (idDelegacao) REFERENCES Delegacao (idDelegacao),
  FOREIGN KEY (idEsporte) REFERENCES Esporte (idEsporte)
);
```

Figura 17 – Código SQL para criar tabela Equipa

A tabela Treinador regista os treinadores associados às equipas. A coluna idTreinador é do tipo “INT”, definida como “PRIMARY KEY” e “NOT NULL”, garantindo um identificador único para cada treinador. A coluna nome é do tipo “VARCHAR(45)” e “NOT NULL”, permitindo armazenar o nome do treinador com até 45 caracteres. A coluna idEquipa é do tipo “INT” e “NOT NULL”, funcionando como uma chave estrangeira que referencia a tabela Equipa, associando cada treinador a uma equipa específica no sistema.

Figura 18 – Tabela Lógica Treinador

```sql
CREATE TABLE IF NOT EXISTS Treinador (
  idTreinador INT NOT NULL,
  nome VARCHAR(45) NOT NULL,
  idEquipa INT NOT NULL,
  PRIMARY KEY (idTreinador),
  FOREIGN KEY (idEquipa) REFERENCES Equipa (idEquipa)
);
```

Figura 19 – Código SQL para criar tabela Treinador

A tabela Atleta regista os atletas que pertencem às equipas. A coluna idAtleta é do tipo “INT”, definida como “PRIMARY KEY” e “NOT NULL”, garantindo um identificador único para cada atleta. A coluna nome é do tipo “VARCHAR(45)” e “NOT NULL”, permitindo armazenar o nome do atleta com até 45 caracteres. A coluna idade é do tipo “INT” e “NOT NULL”, armazenando a idade do atleta. A coluna genero é do tipo “VARCHAR(1)” e “NOT NULL”, armazenando o género do atleta, representado por uma letra (por exemplo, "M" para masculino, "F" para feminino). As colunas peso e altura são do tipo “DECIMAL”, permitindo armazenar valores numéricos com precisão (peso com até 5 dígitos, 2 após a vírgula e altura com até 3 dígitos, 2 após a vírgula), sendo ambas opcionais (NULL). A coluna idEquipa é do tipo “INT” e “NOT NULL”, funcionando como uma chave estrangeira que referencia a tabela Equipa, associando cada atleta a uma equipa específica.

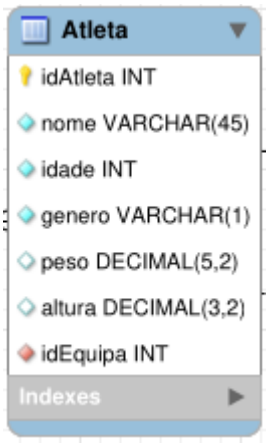

Figura 20 – Tabela Lógica Atleta

```sql
CREATE TABLE IF NOT EXISTS Atleta (
  idAtleta INT NOT NULL,
  nome VARCHAR(45) NOT NULL,
  idade INT NOT NULL,
  genero VARCHAR(1) NOT NULL,
  peso DECIMAL(5,2) NULL,
  altura DECIMAL(3,2) NULL,
  idEquipa INT NOT NULL,
  PRIMARY KEY (idAtleta),
  FOREIGN KEY (idEquipa) REFERENCES Equipa (idEquipa)
);
```

Figura 21 – Código SQL para criar tabela Atleta

A tabela Evento regista os eventos associados a uma modalidade específica. A coluna idEvento é do tipo “INT”, definida como “PRIMARY KEY” e “NOT NULL”, garantindo um identificador único para cada evento. A coluna dataHora é do tipo DATETIME e “NOT NULL”, permitindo armazenar a data e a hora do evento. A coluna localidade é do tipo VARCHAR(45) e “NOT NULL”, permitindo armazenar o nome da localidade onde o evento ocorre, com até 45 caracteres. A coluna idMod é do tipo INT e “NOT NULL”, funcionando como uma chave estrangeira que referencia a tabela Modalidade, associando cada evento a uma modalidade específica.

Figura 22 – Tabela Lógica Evento

```sql
CREATE TABLE IF NOT EXISTS Evento (
  idEvento INT NOT NULL,
  dataHora DATETIME NOT NULL,
  localidade VARCHAR(45) NOT NULL,
  idMod INT NOT NULL,
  PRIMARY KEY (idEvento),
  FOREIGN KEY (idMod) REFERENCES Modalidade (idModalidade)
);
```

Figura 23 – Código SQL para criar tabela Evento

A tabela Resultado regista os resultados de cada atleta em eventos. A coluna idRealiza é do tipo "INT", definida como "PRIMARY KEY" e "NOT NULL", garantindo um identificador único para cada resultado. A coluna posicao é do tipo "INT" e "NOT NULL", indicando a posição do atleta no evento. A coluna pontos é do tipo "INT" (opcional), registrando a pontuação, caso se aplique. A coluna tempo é do tipo TIME (opcional), armazenando o tempo do atleta, se relevante. classificado é do tipo "TINYINT" e "NOT NULL", indicando se o atleta se classificou ou não. As colunas idAtleta e idEvento são chaves estrangeiras que referenciam as tabelas Atleta e Evento, associando o resultado ao atleta e ao evento correspondentes

Figura 24 – Tabela Lógica Resultado

```sql
CREATE TABLE IF NOT EXISTS Resultado (
  idRealiza INT NOT NULL,
  posicao INT NOT NULL,
  pontos INT NULL,
  tempo TIME NULL,
  classificado TINYINT NOT NULL,
  idAtleta INT NOT NULL,
  idEvento INT NOT NULL,
  PRIMARY KEY (idRealiza),
  FOREIGN KEY (idAtleta) REFERENCES Atleta (idAtleta),
  FOREIGN KEY (idEvento) REFERENCES Evento (idEvento)
);
```

Figura 25 – Código SQL para criar tabela Resultado

A tabela Realiza regista a relação entre os atletas e as modalidades que praticam. A coluna idAtleta é uma chave estrangeira que referencia a tabela Atleta, associando cada registo a um atleta específico. A coluna idMod é uma chave estrangeira que referencia a tabela Modalidade, indicando a modalidade associada ao atleta. Ambas as colunas são do tipo INT e NOT NULL.

Figura 26 – Tabela Lógica Realiza

```
CREATE TABLE IF NOT EXISTS Realiza (
  idAtleta INT NOT NULL,
  idMod INT NOT NULL,
  FOREIGN KEY (idAtleta) REFERENCES Atleta (idAtleta),
  FOREIGN KEY (idMod) REFERENCES Modalidade (idModalidade)
);
```

Figura 27 – Código SQL para criar tabela Realiza

4.2. Cálculo do espaço da base de dados (inicial e taxa de crescimento anual)

De forma a perceber o espaço que a base de dados pode vir a ocupar, calculamos inicialmente o mesmo quando existe apenas um registo por tabela e, numa fase final, estimou-se o espaço total necessário para armazenar toda a informação contida na base de dados, assim como a informação que surgirá com um determinado crescimento anual esperado.

Assim, representa-se, a seguir, uma tabela que evidencia os tipos de dados usados no sistema, acompanhados pelos seus tamanhos respetivos em bytes, de acordo com o manual de referência do MySQL.

Tipo de Dados	Tamanho (bytes)
INT	4
DATE	3
VARCHAR(M)	L + 1 bytes caso a coluna necessite 0 − 255 bytes
DECIMAL (M,D)	M + 2
TINYINT	1
DATETIME	8
TIME	6

Tabela 16 : Tipos de dados utilizados e o seu respetivo tamanho em bytes

Seguidamente, representam-se os espaços ocupados por cada tabela, em bytes, para quando existe apenas um registo na mesma.

Tabela	Nome da coluna	Tipo de dado	Tamanho (bytes)
Competição	idCompeticao	INT	4
	dataInicio	DATE	3
	dataFim	DATE	3
	localidade	VARCHAR(45)	46
	nome	VARCHAR(45)	46
Total	**-**	**-**	**102**

Tabela 17: Espaço ocupado pela tabela Competição com um registo

Tabela	Nome da coluna	Tipo de dado	Tamanho (bytes)
Funcionário	idFuncionário	INT	4
	nome	VARCHAR(45)	46
	idTipo	INT	4
	idCompeticao	INT	4
Total	**-**	**-**	**58**

Tabela 18: Espaço ocupado pela tabela Funcionário com um registo

Tabela	Nome da coluna	Tipo de dado	Tamanho (bytes)
TIpo	idTipo	INT	4
	descricao	VARCHAR(45)	46
Total	**-**	**-**	**50**

Tabela 19: Espaço ocupado pela tabela Tipo com um registo

Tabela	Nome da coluna	Tipo de dado	Tamanho (bytes)
Esporte	idEsporte	INT	4
	nome	VARCHAR(45)	46
	idCompeticao	INT	4
Total	**-**	**-**	**54**

Tabela 20: Espaço ocupado pela tabela Esporte com um registo

Tabela	Nome da coluna	Tipo de dado	Tamanho (bytes)
Modalidade	idModalidade	INT	4
	descricao	VARCHAR(100)	101
	idEsporte	INT	4
Total	**-**	**-**	**109**

Tabela 21: Espaço ocupado pela tabela Modalidade com um registo

Tabela	Nome da coluna	Tipo de dado	Tamanho (bytes)
Delegacao	idDelegacao	INT	4
	pais	VARCHAR(45)	46
	idCompeticao	INT	4
Total	**-**	**-**	**54**

Tabela 22: Espaço ocupado pela tabela Delegação com um registo

Tabela	Nome da coluna	Tipo de dado	Tamanho (bytes)
Equipa	idEquipa	INT	4
	idEsporte	INT	4
	idDelegacao	INT	4
Total	**-**	**-**	**12**

Tabela 23: Espaço ocupado pela tabela Equipa com um registo

Tabela	Nome da coluna	Tipo de dado	Tamanho (bytes)
Treinador	idTreinador	INT	4
	nome	VARCHAR(45)	46
	idEquipa	INT	4
Total	**-**	**-**	**54**

Tabela 24: Espaço ocupado pela tabela Treinador com um registo

Tabela	Nome da coluna	Tipo de dado	Tamanho (bytes)
Atleta	idAtleta	INT	4
	nome	VARCHAR(45)	46
	idade	INT	3
	genero	VARCHAR(1)	2
	peso	DECIMAL(5,2)	7
	altura	DECIMAL(3,2)	5
	idEquipa	INT	4
Total	**-**	**-**	**72**

Tabela 25: Espaço ocupado pela tabela Atleta com um registo

Nota: Acrescenta-se 1 byte ao total devido ao peso e altura poderem ser Null.

Tabela	Nome da coluna	Tipo de dado	Tamanho (bytes)
Evento	idEvento	INT	4
	dataHora	DATETIME	46
	localidade	VARCHAR(45)	8
	idMod	INT	4
Total	**-**	**-**	**62**

Tabela 26: Espaço ocupado pela tabela Evento com um registo

Tabela	Nome da coluna	Tipo de dado	Tamanho (bytes)
Realiza	idAtleta	INT	4
	idMod	INT	4
Total	**-**	**-**	**8**

Tabela 27: Espaço ocupado pela tabela Realiza com um registo

Tabela	Nome da coluna	Tipo de dado	Tamanho (bytes)
Resultado	idRealiza	INT	4
	posicao	INT	4
	pontos	INT	4
	tempo	TIME	6
	classificado	TINYINT	1
	idAtleta	INT	4
	idEvento	INT	4
Total	**-**	**-**	**28**

Tabela 28: Espaço ocupado pela tabela Resultado com um registo

Nota: Acrescenta-se 1 byte ao total devido a pontos e tempo poderem ser Null.

Com esta configuração, o espaço total ocupado pelas tabelas, preenchidas com um registo cada, seria igual a 663 bytes.

Após o povoamento inicial e com ainda 4 anos para os próximos Jogos Olimpicos, a base de dados conta com 1 Competiçao que é esta próxima edição dos Jogos Olimpicos, com 206 delegações a competir. e para garantir o bom funcionamento desta competição temos 500 funcionários com 10 funções distintas. Teremos 32 esportes com, em média, 5 modalidades cada, ou seja com um total de 160 modalidades.

Tabela	Quantidade de registos	Tamanho da tabela (bytes)
Competicao	1	102
Tipo	10	500
Funcionario	500	29 000
Esporte	32	1 728
Modalidade	160	17 440
Delegacao	206	11 124
Equipa	0	0
Treinador	0	0
Atleta	0	0
Evento	330	20 460
Resultado	0	0
Realiza	0	0
Total	**-**	**80 354**

Tabela 29: Espaço total ocupado pelas tabelas com o povoamento inicial

Embora os valores do povoamento inicial sejam fixos, depois de uma processo de qualificação árduo alheio aos Jogos Olimpicos, e por sua vez à sua base de dados, é expectável que se qualifique em média 10 equipas por delegação, com cada equipa a ter um treinador e em média 5 atletas.

Cada evento vai ser competido por cerca de 30 atletas (realiza 9 900) , no qual apenas os 8 finalistas terão direito a ter o seu resultado registado na base de dados.

Tabela	Quantidade de registos	Tamanho da tabela (bytes)
Competicao	1	102
Tipo	10	500
Funcionario	500	29 000
Esporte	32	1 728
Modalidade	160	17 440
Delegacao	206	11 124
Equipa	+2060	24 720
Treinador	+2060	111 240
Atleta	+10 500	756 000
Evento	330	20 460
Resultado	+2 640	73 920
Realiza	+9 900	79 200
Total	**-**	**+ 1 045 080**

Tabela 30: Variações do espaço ocupado pelas tabelas após quatro anos

Como foi evidenciado nas tabelas, o espaço necessário mínimo para a primeira implementação da base de dados seria de 78,47 Kbytes ou 0,0766 Mbytes, com um crescimento no fim de quatro anos de 1200%, em que é traduzido por um aumento de 1 020 Kbytes ou 0,997 Mbytes.

4.3 Realização do povoamento da base de dados

Após a criação da estrutura de base de dados, avançamos para a fase de povoamento, onde serão inseridos manualmente os dados necessários para dar suporte às operações e análises previstas. Este passo será conduzido de forma criteriosa, respeitando as relações estabelecidas entre as tabelas e garantindo a consistência das informações. A seguir, detalhamos o processo de inserção de dados nas diferentes tabelas, seguindo uma ordem lógica que assegure a integridade referencial do sistema.

O preenchimento dos dados inicia-se pelas tabelas Competicao e Tipo, pois são tabelas independentes, ou seja, não afetam diretamente as demais tabelas que estão sendo preenchidas. A tabela Competição é especialmente importante porque serve de base para outras tabelas, como a tabela Esporte, Delegacao e Evento, que dependem do seu preenchimento para estabelecer a conexão correta. Já a tabela Tipo é importante na distribuição das informações utilizadas em outras partes do sistema. Ao iniciar um conjunto base com estas tabelas, garantimos um sistema estável e consistente, permitindo que outras tabelas sejam públicas regularmente, evitando problemas de compatibilidade.

```
INSERT INTO Competicao
    (idCompeticao, dataInicio, dataFim, localidade, nome)
VALUES
    (1, '2024-06-10', '2024-06-24', 'Rio de Janeiro', 'Olimpíadas 2024'),
    (2, '2024-07-01', '2024-07-15', 'São Paulo', 'Jogos Panamericanos');

INSERT INTO Tipo
    (idTipo, descricao)
VALUES
    (1, 'Organizador'),
    (2, 'Juiz'),
    (3, 'Assistente Técnico');
```

Figura 28 – Povoamento das tabelas Competição e Tipo

Após o povoamento das tabelas Competicao e Tipo, o próximo passo é preencher as tabelas Funcionario, Esporte e Delegacao, pois elas já possuem uma relação direta com as tabelas previamente preenchidas. A tabela Esporte depende de Competicao, já que cada esporte está vinculado a uma competição específica. Portanto, ao povoar Esporte, garantimos que todos os esportes sejam associados corretamente às competições já existentes. Já a tabela Delegacao também depende de Competicao, uma vez que cada delegação é atribuída a uma competição específica. O preenchimento desta tabela é essencial para estabelecer os vínculos entre as delegações participantes e as competições que elas integram.

Por fim, a tabela Funcionario acaba por depender, não só da tabela Competicao, mas também da tabela Tipo. Ao preencher esta tabela, vinculamos os funcionários às competições, criando a estrutura necessária para associá-los a funções específicas dentro do sistema. Este passo é fundamental para preparar o caminho para o povoamento das tabelas ainda mais dependentes, como Equipe e Modalidade.

```sql
INSERT INTO Funcionario
    (idFuncionario, nome, idTipo, idCompeticao)
VALUES
    (1, 'Carlos Silva', 1, 1),
    (2, 'Ana Paula', 2, 1),
    (3, 'João Souza', 3, 2);

INSERT INTO Esporte
    (idEsporte, nome, idCompeticao)
VALUES
    (1, 'Atletismo', 1),
    (2, 'Natação', 1),
    (3, 'Vôlei', 2);

INSERT INTO Delegacao
    (idDelegacao, pais, idCompeticao)
VALUES
    (1, 'Brasil', 1),
    (2, 'Estados Unidos', 1),
    (3, 'Argentina', 2);
```

Figura 29 – Povoamento das tabelas Funcionario, Esporte e Delegação

Depois de preencher as tabelas Esporte, Delegacao e Funcionario, o próximo passo é povoar as tabelas Modalidade e Equipe. Isso acontece porque elas dependem diretamente das tabelas anteriores para que a organização do sistema faça sentido. A tabela Modalidade depende da tabela Esporte, pois cada modalidade está ligada a um esporte específico. Ou seja, para preencher corretamente as modalidades, precisamos garantir que os esportes já estejam cadastrados, pois a modalidade precisa estar associada a um esporte. Já a tabela Equipe depende tanto da tabela Esporte quanto da tabela Delegacao. Cada equipe está vinculada a um esporte e a uma delegação específica, então é necessário garantir que essas tabelas estejam povoadas primeiro. Isso assegura que as equipes sejam corretamente associadas tanto ao esporte em questão quanto à delegação de que fazem parte. Portanto, povoar primeiro as tabelas Esporte e Delegacao é essencial, pois permite preencher corretamente as tabelas Modalidade e Equipe. Isso cria uma base sólida para o preenchimento das próximas tabelas, como Atleta e Treinador, que vão depender dessas informações para definir a participação dos atletas nas competições e suas respetivas equipes.

```sql
INSERT INTO Modalidade
    (idModalidade, descricao, idEsporte)
VALUES
    (1, '100m Rasos', 1),
    (2, 'Salto em Altura', 1),
    (3, '50m Livre', 2),
    (4, 'Vôlei de Praia', 3);

INSERT INTO Equipa
    (idEquipa, idEsporte, idDelegacao)
VALUES
    (1, 1, 1),
    (2, 2, 1),
    (3, 3, 2);
```

Figura 30 – Povoamento das tabelas Modalidade e Equipa

Em seguida é feito o preenchimento das tabelas Treinador, Atleta e Evento. O motivo pelo qual as tabelas Treinador, Atleta e Evento são preenchidas após as tabelas anteriores acaba por seguir a mesma lógica referida nas etapas anteriores, elas dependem das informações contidas nas tabelas Equipe e Modalidade para garantir a integridade e a coesão dos dados. Primeiro, a tabela Treinador é povoada porque cada treinador deve ser vinculado a uma equipe, e essa equipe já precisa estar associada a uma modalidade. Dessa forma, ao preencher a tabela de treinadores, podemos assegurar que cada um deles estará ligado à equipe certa e à modalidade apropriada. A relação entre treinadores e equipes é essencial para garantir que os treinadores saibam a que grupo estão orientando e qual esporte estão praticando. Em seguida, a tabela Atleta é preenchida, pois os atletas devem ser associados a uma equipe, e, como a tabela de equipes já foi preenchida, podemos vincular facilmente cada atleta à sua respetiva equipe e modalidade. Este preenchimento é crucial, pois permite a organização de informações sobre os atletas e a sua participação nas competições dentro de uma determinada equipe e modalidade. Sem essa conexão, seria impossível gerir o desempenho e a participação dos atletas em eventos ou competições.

```
INSERT INTO Treinador
    (idTreinador, nome, idEquipa)
VALUES
    (1, 'Fernando Costa', 1),
    (2, 'Mariana Lima', 2),
    (3, 'Pedro Oliveira', 3);

INSERT INTO Atleta
    (idAtleta, nome, idade, genero, peso, altura, idEquipa)
VALUES
    (1, 'Lucas Santos', 25, 'M', 70.5, 1.80, 1),
    (2, 'Maria Clara', 22, 'F', 60.0, 1.65, 2),
    (3, 'João Pedro', 28, 'M', 85.0, 1.90, 3);

INSERT INTO Evento
    (idEvento, dataHora, localidade, idMod)
VALUES
    (1, '2024-06-11 10:00:00', 'Estádio Olímpico', 1),
    (2, '2024-06-12 14:00:00', 'Estádio Olímpico', 2),
    (3, '2024-06-13 09:00:00', 'Piscina Olímpica', 3);
```

Figura 31 – Povoamento das tabelas Treinador, Atleta e Evento

Para finalizar, o próximo passo é o povoamento das tabelas Resultado e Realiza. Esse processo é essencial visto que estas duas tabelas dependem diretamente das informações das tabelas anteriores para garantir que a estrutura do banco de dados esteja devidamente organizada e funcional.

A tabela Resultado é responsável por armazenar os resultados das competições, como as classificações ou pontuações de atletas ou equipes em eventos específicos. Para garantir que esses resultados sejam corretamente atribuídos, é fundamental que as tabelas Atleta, Treinador e Evento já estejam preenchidas. Isso ocorre porque, sem as informações dos participantes e dos eventos, não seria possível registar corretamente os resultados. Ou seja, a tabela Resultado só pode ser povoada após garantir que os eventos e os atletas envolvidos nas competições já estão devidamente registados no banco de dados. Por outro lado, a tabela Realiza serve para registrar as participações dos Atletas e Treinadores em Eventos específicos. Ela faz a conexão entre as competições e as pessoas que participaram delas. Para que isso seja feito de forma precisa, é necessário que as tabelas Atleta e Treinador já estejam preenchidas, assim como a tabela Evento. Somente dessa maneira conseguimos associar corretamente quem participou de quais eventos. Portanto, a tabela Realiza deve ser preenchida depois que as informações sobre os atletas, treinadores e eventos já estiverem completas, garantindo que as relações entre esses elementos sejam devidamente mapeadas.

Em resumo, o preenchimento das tabelas Resultado e Realiza ocorre após o povoamento das tabelas anteriores porque essas duas tabelas dependem das informações sobre os participantes (atletas e treinadores) e os eventos. Ao garantir que essas tabelas estejam completas e corretas, é possível associar de maneira precisa os resultados e as participações dos indivíduos nas competições. Esse processo é fundamental para manter a integridade e a organização do banco de dados.

```
INSERT INTO Resultado
    (idRealiza, posicao, pontos, tempo, classificado, idAtleta, idEvento)
VALUES
    (1, 1, 10, '00:10:25', 1, 1, 1),
    (2, 2, 8, '00:10:50', 1, 2, 2),
    (3, 3, NULL, NULL, 0, 3, 3);

INSERT INTO Realiza
    (idAtleta, idMod)
VALUES
    (1, 1),
    (2, 3),
    (3, 4);
```

Figura 32 – Povoamento das tabelas Resultado e Realiza

4.4 Tradução das interrogações do utilizador para SQL

Depois de realizar o povoamento das tabelas, comecamos a elaborar e a executar algumas queries para testar a estrutura e verificar se os dados estavam organizados de forma correta. Essas consultas foram desenvolvidas com o objetivo de explorar as informações armazenadas, garantindo que a modelagem atenda às necessidades propostas inicialmente.

Tendo isto dito apresentamos aqui algumas das queries que utiliazamos como testes para a base de dados.

```
SELECT A.nome, M.descricao, C.nome AS Competicao
FROM Atleta A
JOIN Realiza R ON A.idAtleta = R.idAtleta
JOIN Modalidade M ON R.idMod = M.idModalidade
JOIN Esporte E ON M.idEsporte = E.idEsporte
JOIN Competicao C ON E.idCompeticao = C.idCompeticao;
```

Figura 33 – Listagem de Atletas por Competição e Modalidade

A querie apresentada tem como objetivo listar os atletas, a modalidade esportiva em que participam e a competição em que a modalidade está inserida. Para isso, são utilizadas diversas tabelas interligadas.

A relação entre atletas e modalidades é estabelecida por meio da tabela Realiza (R), que associa cada atleta à modalidade em que participa, utilizando o campo idAtleta. Em seguida, faz-se o JOIN entre Realiza e Modalidade para associar o atleta à modalidade correta. Cada modalidade pertence a um Esporte,daí ser feito o JOIN entre Modalidade e Esporte através do valor idEsporte. Finalmente, a tabela Esporte (E) está relacionada a uma Competição (representada pela tabela Competicao (C)), que indica em qual competição o esporte está inserido. A querie então associa o Esporte à Competição usando o campo idCompeticao.

O resultado final da querie é uma lista que exibe o nome do atleta, a descrição da modalidade em que participa e o nome da competição em que essa modalidade ocorre, criando uma visão detalhada sobre onde e como o atleta está competindo.

Apresentamos abaixo outra querie que segue uma lógica semelhante.

```
SELECT E.dataHora, E.localidade, C.nome AS Competicao
FROM Evento E
JOIN Modalidade M ON E.idMod = M.idModalidade
JOIN Esporte S ON M.idEsporte = S.idEsporte
JOIN Competicao C ON S.idCompeticao = C.idCompeticao;
```

Figura 34 – Listagem de Eventos por localidade e competição

```sql
SELECT A.nome, SUM(R.pontos) AS Pontuacao_Total
FROM Atleta AS A
JOIN Resultado AS R ON A.idAtleta = R.idAtleta
GROUP BY A.nome;
```

Figura 35 – Pontuação total por Atleta em todos os Eventos

A querie acima tem como objetivo calcular a pontuação total de cada atleta com base nos resultados obtidos em diversas competições ou eventos.

A consulta começa com a seleção dos nomes dos atletas (A.nome) e a soma dos pontos acumulados por cada um dos atletas, utilizando a função agregadora SUM(R.pontos), que calcula o total de pontos de cada atleta. A tabela Resultado é associada à tabela Atleta através do JOIN entre o valor idAtleta de ambas as tabelas, garantindo que os pontos de cada atleta sejam somados corretamente.

Para garantir que os pontos sejam somados por atleta, a consulta utiliza a cláusula GROUP BY A.nome, que agrupa os registros de acordo com o nome de cada atleta. Isso significa que, para cada atleta, a soma de seus pontos será calculada e apresentada como uma única linha por nome de atleta.

```sql
SELECT A.nome, M.descricao, R.classificado
FROM Atleta A
JOIN Resultado R ON A.idAtleta = R.idAtleta
JOIN Evento E ON R.idEvento = E.idEvento
JOIN Modalidade M ON E.idMod = M.idModalidade
WHERE R.classificado = 1;
```

Figura 36 – Atletas classificados em modalidades especificas

Esta querie lista os atletas classificados em eventos esportivos, junto com a modalidade dos eventos em que participaram. Para isso, ela utiliza as tabelas Atleta (A), Resultado (R), Evento (E) e Modalidade (M), que estão interligadas para fornecer as informações necessárias.

A querie seleciona o nome dos atletas (A.nome), a descrição da modalidade (M.descricao) e o indicador de classificação (R.classificado). A tabela Atleta é relacionada à tabela Resultado pelo campo idAtleta, identificando os resultados de cada atleta. Em seguida, a tabela Resultado é conectada à tabela Evento através do campo idEvento, para associar os

resultados aos eventos esportivos. Finalmente, a tabela Evento é conectada à tabela Modalidade pelo campo idMod, permitindo identificar a modalidade associada a cada evento.

A cláusula “WHERE R.classificado = 1” filtra os dados, garantindo que apenas os atletas que foram classificados (com o campo classificado igual a 1) sejam incluídos no resultado.

4.5 Definição e caracterização de vistas de utilização em SQL

Após a elaboração e execução das consultas SQL para extrair informações específicas e detalhadas das tabelas do banco de dados, passamos agora para a próxima etapa: a definição e caracterização das vistas de utilização. Este capítulo tem como objetivo criar estruturas que consolidem os dados mais relevantes e frequentemente consultados, simplificando o acesso e a reutilização dessas informações. Neste capítulo, exploraremos como as vistas podem ser criadas para representar cenários específicos de análise, combinando dados de várias tabelas e agrupando-os de forma lógica.

```
CREATE VIEW View_Atletas_Equipas AS
SELECT A.idAtleta, A.nome AS nomeAtleta, A.idade, A.genero,
       Eq.idEquipa, Es.nome AS nomeEsporte, D.pais AS paisDelegacao
    FROM Atleta AS A
    JOIN Equipa AS Eq ON A.idEquipa = Eq.idEquipa
    JOIN Esporte AS Es ON Eq.idEsporte = Es.idEsporte
    JOIN Delegacao AS D ON Eq.idDelegacao = D.idDelegacao;
```

Figura 37 – Vista View_Atletas_Equipas

Essa vista SQL, com nome *View_Atletas_Equipas*, é criada para consolidar e simplificar a consulta de informações relacionadas a atletas, equipas, esportes e delegações. Ela combina dados de várias tabelas através de JOINs, resultando em uma estrutura lógica que permite acessar os dados relevantes de maneira organizada e eficiente.

A criação da vista começa com a instrução CREATE VIEW View_Atletas_Equipas AS, seguida de uma consulta SELECT que define os campos que farão parte da vista. Esses campos incluem o identificador do atleta (A.idAtleta), o nome do atleta (A.nome, renomeado como nomeAtleta), a idade e o gênero do atleta (A.idade, A.genero), além de informações adicionais como o identificador da equipa (Eq.idEquipa), o nome do esporte (Es.nome, renomeado como nomeEsporte), e o país da delegação (D.pais, renomeado como paisDelegacao).

O conjunto de JOINs conecta as tabelas necessárias para extrair todas essas informações. Primeiro, a tabela Atleta (A) é ligada à tabela Equipa (Eq) por meio da chave idEquipa, que indica a equipa a que cada atleta pertence. Em seguida, a tabela resultante é conectada à tabela Esporte (Es) pelo campo idEsporte, relacionando a equipa ao esporte que pratica. Por fim, faz-se também a junção à tabela Delegacao (D) pelo campo idDelegacao, que identifica a delegação nacional ou regional associada à equipa.

O resultado da vista é uma estrutura lógica que consolida todas essas informações em uma única fonte de dados, permitindo que consultas futuras sejam feitas de forma mais simples, sem a necessidade de repetir os JOINs ou lidar diretamente com a complexidade das tabelas subjacentes. Assim, View_Atletas_Equipas facilita o acesso a informações como a lista de atletas, suas equipas, os esportes que praticam e os países ou delegações que representam, tudo isso em uma única consulta à vista.

```
CREATE VIEW View_Resultados_Eventos AS
SELECT E.idEvento, E.dataHora, E.localidade, M.descricao AS modalidade,
       A.nome AS nomeAtleta, R.posicao, R.pontos, R.tempo, R.classificado
    FROM Resultado AS R
    JOIN Evento AS E ON R.idEvento = E.idEvento
    JOIN Atleta AS A ON R.idAtleta = A.idAtleta
    JOIN Modalidade AS M ON E.idMod = M.idModalidade;
```

Figura 38– Vista View_Resultados_Evento

A vista *View_Resultados_Eventos* é criada com o objetivo de consolidar informações completas e detalhadas sobre os resultados de eventos esportivos, facilitando o acesso a dados que envolvem os eventos, os atletas participantes, as modalidades esportivas e os resultados obtidos. Ela reúne informações provenientes de várias tabelas relacionadas, permitindo uma visão integrada que elimina a necessidade de consultas complexas repetidas no banco de dados.

A criação de qualquer vista começa com a instrução CREATE VIEW {nome da vista} AS (neste caso *View_Resultados_Eventos*), tendo isto dito fez-se de seguida uma consulta SELECT que define os campos incluídos na vista. Entre esses campos, estão o identificador do evento (E.idEvento), a data e hora em que ocorreu (E.dataHora), e a localidade do evento (E.localidade). Além disso, são adicionados detalhes como a descrição da modalidade esportiva (M.descricao, renomeada como modalidade), o nome do atleta que participou (A.nome, renomeado como nomeAtleta), e o desempenho do atleta no evento, que inclui a posição final (R.posicao), a pontuação obtida (R.pontos), o tempo registrado (R.tempo) e um indicador de classificação (R.classificado).

A integração desses dados é feita por meio de JOINs entre as tabelas do banco de dados. A tabela Resultado (R), que regista o desempenho de cada atleta, é o ponto central. Ela é conectada à tabela Evento (E) através do campo idEvento, relacionando os resultados aos

eventos correspondentes. Em seguida, a tabela Resultado (R) é associada à tabela Atleta (A) pelo campo idAtleta, para identificar o atleta responsável pelo resultado. Finalmente, a tabela Evento (E) é ligada à tabela Modalidade (M) pelo campo idMod, permitindo identificar a modalidade esportiva de cada evento.

Temos aqui outra vista que segue exatamente a mesma lógica.

```
CREATE VIEW View_Agenda_Eventos AS
SELECT Ev.idEvento, Ev.dataHora, Ev.localidade,
    M.descricao AS modalidade, C.nome AS nomeCompeticao
    FROM Evento AS Ev
    JOIN Modalidade AS M ON Ev.idMod = M.idModalidade
    JOIN Esporte AS Es ON M.idEsporte = Es.idEsporte
    JOIN Competicao AS C ON Es.idCompeticao = C.idCompeticao;
```

Figura 39 – Vista View_Agenda_Eventos

```
CREATE VIEW View_Desempenho_Delegacoes AS
SELECT D.pais, SUM(R.pontos) AS totalPontos,
       COUNT(CASE WHEN R.classificado = 1 THEN 1 END) AS totalClassificados
    FROM Resultado AS R
    JOIN Atleta AS A ON R.idAtleta = A.idAtleta
    JOIN Equipa AS E ON A.idEquipa = E.idEquipa
    JOIN Delegacao AS D ON E.idDelegacao = D.idDelegacao
    GROUP BY D.pais;
```

Figura 40 – Vista View_Desempenho_Delegacoes

A vista *View_Desempenho_Delegacoes* é criada para fornecer uma análise consolidada do desempenho das delegações em competições esportivas, calculando o total de pontos obtidos por cada delegação e o número de atletas classificados. Essa vista é uma ferramenta poderosa para avaliar a performance geral de cada país ou delegação, utilizando dados provenientes de diversas tabelas do banco de dados.

A criação dessa vista começa com o comando CREATE VIEW View_Desempenho_Delegacoes AS, seguido de uma consulta SELECT que especifica as informações desejadas. A consulta inclui o país de cada delegação (D.pais), a soma total dos pontos obtidos pelos atletas de cada delegação (SUM(R.pontos)), e uma contagem dos atletas

classificados, calculada com a expressão COUNT(CASE WHEN R.classificado = 1 THEN 1 END), que contabiliza somente os casos em que o indicador de classificação é igual a 1.

Para consolidar essas informações, a consulta faz uso de diversos JOINs para conectar as tabelas envolvidas. A tabela Resultado (R), que armazena os resultados dos atletas, é associada à tabela Atleta (A) pelo campo idAtleta, permitindo identificar os atletas responsáveis pelos resultados. A tabela Atleta está ligada à tabela Equipa (E) através do campo idEquipa, relacionando os atletas às suas equipas. Por fim, a tabela Equipa é conectada à tabela Delegacao (D) pelo campo idDelegacao, que permite identificar o país ou delegação a que cada equipa pertence. A cláusula GROUP BY D.pais agrupa os dados pelo país de cada delegação, garantindo que a soma dos pontos e o número de classificados sejam calculados separadamente para cada delegação. O resultado é uma visão consolidada onde cada linha representa o desempenho de uma delegação específica, com seu país, a soma dos pontos acumulados por seus atletas e o total de classificados.

A vista *View_Desempenho_Delegacoes* é especialmente útil para gerar relatórios ou análises que avaliem a performance de cada delegação em competições.

4.6 Definição dos perfis de utilização para cada utilizador da base de dados

Em seguida, a próxima etapa é definir os perfis de utilização dos futuros utilizadores desta base de dados. O primeiro passo foi definir o administrador da base de dados em questão, pois é ele quem terá todos os privilégios do sistema e poderá inclusive atribuir ou revogar as permissões dos outros utilizadores.

```
CREATE USER 'adminBD'@'localhost' IDENTIFIED BY 'P4ssW0rd@';
GRANT ALL PRIVILEGES ON eventodesportivo.* TO 'adminBD'@'localhost';
```

Figura 41 – Definição do utilizador “adminBD”

Em seguida podemos definir outros usuários. No entanto, dado que, num contexto real, haveria dezenas talvez centenas de utilizadores, faz sentido começar por definir alguns grupos de usuários. Dessa forma, é mais fácil de controlar quais usuários possuem quais permissões bem como alterar em larga escala de maneira rápida e fácil.

Os grupos que criamos foram os: Organizadores, Juiz, Gerente de delegações e Usuário. Estes grupos são apenas alguns exemplos do que é possível fazer em SQL e não representam os perfis de utilização necessários na gestão de uma base de dados real em sua totalidade.

```
CREATE ROLE Organizadores;
GRANT SELECT, INSERT, UPDATE, DELETE ON EventoDesportivo.* TO Organizadores;
```

Figura 42 – Definição do grupo “Organizadores”

```
CREATE ROLE Juiz;
GRANT SELECT ON EventoDesportivo.Atleta TO Juiz;
GRANT SELECT ON EventoDesportivo.Evento TO Juiz;
GRANT SELECT ON EventoDesportivo.Modalidade TO Juiz;
GRANT INSERT, UPDATE ON EventoDesportivo.Resultado TO Juiz;
```

Figura 43 – Definição do grupo “Juiz”

```
CREATE ROLE GerenteDelegacao;
GRANT SELECT ON EventoDesportivo.Atleta TO GerenteDelegacao;
GRANT SELECT ON EventoDesportivo.Equipa TO GerenteDelegacao;
GRANT SELECT ON EventoDesportivo.Delegacao TO GerenteDelegacao;
GRANT SELECT ON EventoDesportivo.Resultado TO GerenteDelegacao;
```

Figura 44 – Definição do grupo “GerenteDelegacao”

```
CREATE ROLE Usuario;
GRANT EXECUTE ON PROCEDURE
    eventodesportivo.ListarResultadosAtleta TO Usuario;
GRANT EXECUTE ON PROCEDURE
    eventodesportivo.ContarAtletasPorDelegacao TO Usuario;
```

Figura 45 – Definição do grupo “Usuario”

O próximo passo foi criar alguns usuários e apenas atribuir a qual grupo eles pertencem. Esta forma de definir utilizadores é rápida e evita que um utilizador receba permissões indevidas.

```
CREATE USER 'org1'@'localhost' IDENTIFIED BY 'org123@Comp456';
GRANT Organizadores TO 'org1'@'localhost';
```

Figura 46 – Criação do usuário “org1” e a atribuição do grupo “Organizadores”

```
CREATE USER 'judge1'@'localhost' IDENTIFIED BY 'pass_judge_2025';
CREATE USER 'juizA'@'localhost' IDENTIFIED BY 'juizA2025senha';
GRANT Juiz TO 'judge1'@'localhost';
GRANT Juiz TO 'juizA'@'localhost';
```

Figura 47 – Definição dos usuários “judge1” e “juizA”, ambos pertencentes ao grupo “Juiz”

```
CREATE USER 'gestorDeleg1'@'localhost' IDENTIFIED BY 'Senha2025Delegacao1';
GRANT GerenteDelegacao TO 'gestorDeleg1'@'localhost';
```

Figura 48 – Definição do “gestorDeleg1”, um usuário do grupo “GerenteDelegacao”

Além do grupo (Role) e das permissões vistas anteriormente, também podemos limitar a frequência de uso de um usuário. Por exemplo, limitando o número de “queries” que podem ser realizadas. Como podemos ver na Figura 32.

```
CREATE USER 'user1'@'localhost' IDENTIFIED BY 'PASSWORD123'
    WITH MAX_QUERIES_PER_HOUR 100;
GRANT Usuario TO 'user1'@'localhost';
```

Figura 49 – Definição do usuário “user1”

Por fim, é preciso utilizar o comando “Flush privileges;” para atualizar os perfis de usuários e suas permissões.

4.7 Indexação do Sistema de Dados

Nesta secção aborda-se a necessidade de implementar uma metodologia de indexação na base de dados para melhorar a eficiência das operações de pesquisa e manipulação de dados. O primeiro passo envolve a identificação das tabelas e colunas com maior frequência de acesso, uma vez que estas necessitarão de índices para otimizar o desempenho.

Após uma análise cuidadosa, verificou-se que as tabelas Atleta, Resultado, Evento, Modalidade, Competição e Delegação desempenham um papel central no sistema e estarão sujeitas a consultas frequentes. Para garantir a eficiência, os índices foram criados nas seguintes colunas:

Tabela Atleta: Um índice foi criado na coluna nome para acelerar a pesquisa de atletas pelo seu nome.

```
CREATE INDEX idx_anome ON Atleta(nome);
```

Figura 50 - Aplicação de indexação na tabela Atleta

Tabela Resultado: Um índice foi criado nas colunas idAtleta e idEvento para otimizar as operações relacionadas a resultados associados a atletas e eventos.

```
CREATE INDEX idx_resultados ON Resultado(idAtleta, idEvento);
```

Figura 51 - Aplicação de indexação na tabela Resultado

Tabela Evento: A coluna dataHora recebeu um índice para facilitar a consulta por data e hora dos eventos.

```
CREATE INDEX idx_eventoS on Evento(dataHora);
```

Figura 52 - Aplicação de indexação na tabela Evento

Tabela Modalidade: A coluna descricao foi indexada para permitir buscas rápidas por descrições de modalidades.

```
CREATE INDEX idx_modalidades ON Modalidade(descricao);
```

Figura 53- Aplicação de indexação na tabela Modalidade

Tabela Competição: As colunas nome e dataInicio foram indexadas para melhorar a pesquisa por competições específicas e suas datas de início.

```
CREATE INDEX idx_competicoes ON Competicao(nome,dataInicio);
```

Figura 54 - Aplicação de indexação na tabela Competição

Tabela Delegação: Um índice foi aplicado na coluna pais para otimizar consultas relacionadas aos países das delegações.

```
CREATE INDEX idx_delegacoes ON Delegacao(pais);
```

Figura 55 - Aplicação de indexação na tabela Delegação

A criação destes índices é justificada por três motivos principais:

1. **Melhoria do desempenho nas consultas:** Os índices permitem localizar rapidamente os registos necessários, garantindo tempos de resposta mais curtos em operações de leitura.
2. **Redução do processamento desnecessário:** Com índices, o sistema pode aceder diretamente aos registos relevantes, evitando a necessidade de percorrer toda a tabela.
3. **Escalabilidade do sistema:** Embora o volume de dados vá crescer consideravelmente (1200% em quatro anos), o crescimento é previsível e gerenciável devido à estrutura e restrições do sistema. O uso de índices garante que, mesmo com esse aumento, as operações de consulta permaneçam eficientes.

No contexto do sistema, os índices foram aplicados estrategicamente para alinhar-se com os padrões de uso previstos, assegurando que o desempenho da base de dados seja otimizado. É importante notar que não foi necessário criar índices para chaves primárias, pois estas são automaticamente indexadas no momento da sua definição.

4.8 Procedimentos Implementados

Numa base de dados completa, com diversos usuários utilizando frequentemente, a criação de procedimentos torna-se necessária para garantir o desempenho e, muitas vezes, a segurança da base de dados. Para este trabalho, criamos algumas queries com o objetivo de demonstrar a capacidade da nossa base de dados, por exemplo, o procedimento "cadastrarAtleta" permite automatizar a operação de *INSERT* na tabela atleta.

```
DELIMITER $$
CREATE PROCEDURE CadastrarAtleta (
    IN in_nome VARCHAR(45),
    IN in_idade INT,
    IN in_genero CHAR(1),
    IN in_peso DECIMAL(5,2),
    IN in_altura DECIMAL(3,2),
    IN in_idEquipa INT
)
BEGIN
    IF EXISTS (SELECT 1 FROM Equipa WHERE idEquipa = in_idEquipa) THEN
        INSERT INTO Atleta (nome, idade, genero, peso, altura, idEquipa)
        VALUES (in_nome, in_idade, in_genero, in_peso, in_altura, in_idEquipa);
    ELSE
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'A equipa especificada não existe.';
    END IF;
END
$$
```

Figura 56 – Definição do procedimento: "cadastrarAtleta"

Outro exemplo, seria o precedimento a seguir "AtualizarResultadoEvento". Como o nome sugere, este procedimento facilita a alteração de uma entrada da tabela Resultado.

```
DELIMITER $$
CREATE PROCEDURE AtualizarResultadoEvento (
    IN in_idAtleta INT,
    IN in_idEvento INT,
    IN in_posicao INT,
    IN in_pontos INT,
    IN in_tempo TIME,
    IN in_classificado TINYINT
)
BEGIN
    IF EXISTS (SELECT 1 FROM Resultado WHERE idAtleta = in_idAtleta AND idEvento = in_idEvento) THEN
        UPDATE Resultado
        SET posicao = in_posicao,
            pontos = in_pontos,
            tempo = in_tempo,
            classificado = in_classificado
        WHERE idAtleta = in_idAtleta AND idEvento = in_idEvento;
    ELSE
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'O resultado especificado não existe.';
    END IF;
END
$$
```

Figura 57 – Definição do procedimento: “AtualizarResultadoEvento”

Podemos também definir procedimentos com queries mais complexas que utilizam da operação de junção (JOIN) de tabelas. O procedimento abaixo, “ListarResultadosAtleta”, é um exemplo disto, no qual listamos todos os resultados de um atleta específico nos diferentes eventos.

```
DELIMITER $$
CREATE PROCEDURE ListarResultadosAtleta (
    IN in_idAtleta INT
)
BEGIN
    SELECT
        E.idEvento, E.dataHora, E.localidade, M.descricao AS modalidade,
        R.posicao, R.pontos, R.tempo, R.classificado
    FROM Resultado AS R
    JOIN Evento AS E ON R.idEvento = E.idEvento
    JOIN Modalidade AS M ON E.idMod = M.idModalidade
    WHERE R.idAtleta = in_idAtleta;
END
$$
```

Figura 58 – Definição do procedimento: “listarResulatdosAtleta”

O precedimento abaixo, “ContarAtletasPorDelegacao”, segue a mesma lógica.

```
DELIMITER $$
CREATE PROCEDURE ContarAtletasPorDelegacao (
    IN in_idCompeticao INT
)
BEGIN
    SELECT
        Delegacao.pais, COUNT(A.idAtleta) AS totalAtletas
    FROM Atleta AS A
    JOIN Equipa AS E ON A.idEquipa = E.idEquipa
    JOIN Delegacao AS D ON E.idDelegacao = D.idDelegacao
    WHERE D.idCompeticao = in_idCompeticao
    GROUP BY Delegacao.pais;
END
$$
```

Figura 59 – Definição do procedimento: “ContarAtletasPorDelegacao”

4.9 Plano de Segurança e Recuperação de Dados

O último passo da Implementação Física desta Base de Dados é o Plano de Segurança e Recuperação de Dados. No contexto do trabalho, no qual a base de dados trata-se de eventos desportivos multi-modalidades, a segurança e recuperação de dados são aspectos essenciai devido à importância e sensibilidade das informações, como detalhes dos eventos, participantes, horários e afins. Um bom plano de backup e recuperação minimiza o risco de perda de dados em caso de falhas técnicas, ataques cibernéticos ou erros humanos.

Quanto a etapa de backup, uma opção é a de backups incrementais diários e completos semanais. Os backups incrementais salvam apenas os dados alterados desde o último backup, seja este backup completo ou não, o que economiza espaço e tempo. Já os backups completos, realizados semanalmente, garantem uma cópia integral dos dados, assegurando que nada será perdido. Além disso, backups podem ser armazenados em mais de um disco rígido (hard drive) ou até em mais do que um local para maior segurança.

A recuperação pode ser feita restaurando os arquivos de backup completos e aplicando, em seguida, os backups incrementais para atualizar os dados ao estado mais recente. Esse processo pode ser lento, dependendo de quantos backups incrementais foram realizados, e deve ser documentado e testado regularmente para garantir sua eficiência em situações de emergência.

5. Conclusões e Trabalho Futuro

O trabalho realizado, abrangendo os modelos conceitual, lógico e físico, foi concluído, destacando-se pela clareza na definição das entidades e seus relacionamentos, além da transição eficiente entre as etapas, culminando na implementação em SQL. A utilização do brModelo na modelagem conceitual proporcionou uma visão clara e bem estruturada do sistema, enquanto a modelagem lógica no MySQL assegurou uma base sólida para a implementação física.

Com a implementação física já concluída, incluindo índices otimizados e a criação de views e outros recursos avançados, o projeto alcançou um nível de maturidade robusto. Para o futuro, pode-se considerar a integração com outras ferramentas, explorar técnicas de análise de dados ou até mesmo expandir o sistema para atender a novas demandas.