UNIVERSIDADE DE SÃO PAULO

INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

Monitoramento de Atletas Olímpicos

Entrega Final

SCC0640 – Bases de Dados

Profa. Dra. Elaine Parros Machado de Sousa

Monitor PAE: André Moreira Souza

Jean Carlos Pereira Cassiano - 13864008

João Marcelo Ferreira Battaglini - 13835472

João Victor de Almeida - 13695424

Lucas Corlete Alves de Melo - 13676461

Lucas Sales Duarte - 11734490

São Carlos – SP

07/12/2024

Sumário

1. Introdução

A Comissão Olímpica Internacional (COI) propõe a criação de um banco de dados abrangente com o objetivo de coletar e armazenar uma vasta quantidade de informações sobre atletas, competições e resultados esportivos. Essa iniciativa visa impulsionar a pesquisa e o desenvolvimento de novas tecnologias aplicadas ao esporte, especialmente no campo da inteligência artificial.
A partir do banco de dados, o objetivo principal está associado ao enriquecimento de informações para o treinamento de modelos de inteligência artificial, por meio aplicação de *machine learning*, por exemplo, de modo que seja possível a integração com usuários a nível de aplicação.

1.1 Descrição da Problemática

A prática esportiva de alto rendimento, especialmente no contexto das Olimpíadas, envolve a interação de múltiplos fatores que influenciam diretamente o desempenho dos atletas. Estes fatores incluem aspectos físicos, técnicos, táticos, psicológicos, genéticos e ambientais. No entanto, a coleta e análise detalhada desses fatores ainda enfrentam desafios significativos, principalmente devido à falta de integração e centralização dos dados disponíveis.

Atualmente, os dados dos atletas estão dispersos em diferentes plataformas, sistemas e formatos, o que torna difícil para treinadores, equipes médicas, pesquisadores e analistas esportivos realizar uma análise holística do desempenho e da saúde dos atletas ao longo do tempo. Além disso, muitos dados, como vídeos de desempenho e exames médicos, são armazenados de forma isolada, limitando sua utilidade para a criação de modelos de inteligência artificial que poderiam ajudar a identificar padrões, tendências e correlações que impactam diretamente a performance esportiva.

A falta de um banco de dados centralizado e abrangente também dificulta a personalização de treinos e a prevenção de lesões, pois as informações não estão facilmente acessíveis ou analisáveis em conjunto. Este problema se agrava ainda mais pela falta de padronização na coleta e armazenamento de dados, o que pode resultar em informações imprecisas, incompletas ou inconsistentes, comprometendo o desenvolvimento de estudos científicos sólidos e a aplicação de tecnologias avançadas, como o machine learning, para o aprimoramento do desempenho esportivo.

Nesse contexto, surge a necessidade de desenvolver um banco de dados robusto e centralizado que integre diferentes tipos de informações — como vídeos de desempenho, exames médicos, dados genéticos, estatísticas de jogo, e informações biométricas — de maneira organizada e acessível. Este banco de dados deve ser projetado para capturar a complexidade das variáveis que afetam o desempenho dos atletas e ser capaz de oferecer insights valiosos para todas as partes interessadas, desde treinadores e equipes médicas até analistas de desempenho e cientistas esportivos.

O desafio central é, portanto, criar uma infraestrutura de banco de dados que permita:

- A coleta de dados de múltiplas fontes em diferentes formatos.
- A integração e centralização desses dados de maneira organizada.
- A análise eficiente de grandes volumes de dados para identificar padrões e tendências que possam ser utilizados para otimização do desempenho e prevenção de lesões.
- A aplicação de técnicas de inteligência artificial para a geração de insights personalizados, otimizando assim a tomada de decisão no treinamento e na gestão da saúde dos atletas.

Ao abordar esses desafios, o banco de dados proposto contribuirá significativamente para a evolução da ciência do esporte, para o desenvolvimento de novas tecnologias que visam melhorar a performance atlética e a saúde dos atletas e para o incentivo às atividades esportivas das pessoas.

1.2 Motivação

A principal motivação para a construção desse banco de dados é a possibilidade de realizar estudos aprofundados sobre o desempenho atlético, identificando padrões, correlações e tendências que possam auxiliar no aprimoramento do treinamento, prevenção de lesões e otimização do desempenho esportivo e, então, ser possível sugerir em aplicação ao usuário . Nesse sentido, a coleta e análise desses dados permitirão:

- **Evolução individual dos atletas:** Acompanhar o desenvolvimento de cada atleta ao longo de sua carreira.
- **Influência genética:** Analisar a relação entre o genoma e o desempenho esportivo.
- **Biomecânica e desempenho:** Estudar a relação entre os movimentos e o resultado das competições.
- **Prevenção de lesões:** Identificar fatores de risco e desenvolver estratégias para prevenir lesões.
- **Desenvolvimento de novas tecnologias:** Treinar modelos de inteligência artificial para fornecer insights personalizados - e cada vez melhores - para atletas, treinadores e usuários comuns.
- **Acompanhamento da Saúde e Bem-Estar dos Atletas**: a garantia de que o bem-estar dos atletas profissionais está sendo monitorado incentiva a participação no esporte de atletas em potencial, especialmente jovens, visto que podem ver a prática de alto rendimento de maneira mais saudável e segura.
- **Popularização do Esporte Através da Tecnologia**: com acesso mais difundido de tecnologias que melhoram o desempenho esportivo, há, consequentemente, um aumento no

interesse pela prática esportiva pelo público, tal como aplicativos de construção de rotina de treinamento para dispositivos *mobile*.

Logo, o tema deste estudo está profundamente conectado à promoção e ao incentivo do esporte, tanto em nível profissional quanto amador. Quando os atletas e suas equipes percebem os benefícios do monitoramento, como melhora contínua no desempenho e redução de lesões, isso incentiva a adoção de práticas esportivas mais eficientes. Ao verem seus próprios progressos, os atletas e outras pessoas de fora do esporte profissional se sentem motivados a se dedicar ao esporte, buscando sempre construir uma vida ativa e aprimorar resultados.

1.3 Sobre o Banco de Dados

O banco de dados da COI armazenará diversas informações, incluindo:

- **Pessoas:** Armazenará dados pessoais dos Atletas do campeonato, os treinadores de cada time e todos os médicos que estarão aptos a realizarem exames nos atletas.
- **Vídeos:** Vídeos de alta qualidade de cada atleta durante uma partida, capturados em diferentes ângulos e momentos da competição.
- **Exames médicos:** Resultados de diversos exames, como exames de sangue, imagem, cardiológicos, entre outros, analisados devidamente por um grupo de médicos organizado pela COI.
- **Dados genéticos:** Sequenciamento genético dos atletas.
- **Estatísticas de desempenho:** Dados quantitativos sobre o desempenho dos atletas em cada competição (pontos, rebotes, assistências, etc.).
- **Informações biométricas:** Dados sobre a composição corporal, força, resistência e outras características físicas dos atletas.
- **Times**: Equipes compostas por pelo menos um atleta, para cada esporte das competições, que serão de fato as entidades que participam das partidas, de modo que cada time também terá um treinador qualificado selecionado.

Desse modo, o intuito do banco de dados é armazenar todas essas informações para que possam ser utilizadas futuramente para estudos de caso e aplicação à nível de usuário, como: evolução de um atleta ao longo de todo o campeonato; influência de genoma para certas estatísticas do atleta na partida (número de gols, passes, assistências e quaisquer outros dados que dependem da modalidade); relação de formas de se locomover, movimentos específicos em momentos decisivos, seja em um momento de pontuação do time ou de uma lesão do atleta.

1.3.1 Requisitos de Dados

No que diz respeito aos dados específicos, o banco deve armazenar informações detalhadas dos atletas, incluindo, CPF, nome, data de nascimento, função, endereço genéticos, exames médicos, dados biométricos e histórico de desempenho. Também deve coletar dados de desempenho e treinamento, como estatísticas de jogos, movimentos específicos durante as competições e vídeos para análise de performance e evolução dos atletas ao longo do tempo. Além disso, é necessário registrar informações detalhadas sobre cada evento esportivo, como data, local, participantes e resultados, permitindo uma análise cruzada com os dados dos atletas. Dados de suporte médico, incluindo registros de exames realizados, tratamentos administrados e recomendações médicas, são fundamentais para a prevenção e o gerenciamento de lesões. Informações sobre condições ambientais, como clima e ambiente durante as competições, também devem ser coletadas para entender sua influência no desempenho dos atletas.

O banco de dados deve facilitar a integração com outras plataformas e sistemas de gerenciamento de competições e ferramentas de análise, utilizando APIs e formatos de dados padronizados para interoperabilidade. Além disso, é essencial assegurar que o banco de dados esteja em conformidade com as regulamentações de proteção de dados, garantindo o uso responsável e seguro das informações dos atletas.

Estes requisitos são essenciais para criar um banco de dados robusto e centralizado que suporte a análise e otimização do desempenho dos atletas olímpicos, contribuindo para o desenvolvimento da ciência esportiva e o aprimoramento da gestão de saúde e treinamento dos atletas.

1.3.2 Principais Funcionalidades

Considerando a organização interna da COE, responsável pelo gerenciamento do banco de dados, e os usuários de consulta (médicos, por exemplo) o sistema deve possuir as seguintes funcionalidades:

Inserção de Dados:

- Inserção de médicos e treinadores, com suas devidas identificações e funções.
- Inserção de informações detalhadas dos atletas, incluindo dados pessoais, biométricos, genéticos e resultados de exames médicos.
- Registro de vídeos de desempenho de atletas durante as competições, com metadados como

data, local e dispositivo de captura.

- Adição de novas partidas, incluindo detalhes como equipes, local, data e hora.

Busca e Consulta:

- Busca por atletas específicos com filtros por nome, nacionalidade, idade, e estatísticas de desempenho.
- Consulta de vídeos de treinos e competições por esporte, atleta, data ou tipo de movimento.
- Pesquisa por exames médicos realizados, com possibilidade de filtrar por tipo de exame, médico responsável ou data.

Remoção de Dados:

- Remoção de registros de atletas, incluindo todas as informações associadas (dados biométricos, genéticos, vídeos, etc.).
- Exclusão de vídeos específicos que não atendem aos critérios de qualidade ou foram carregados incorretamente.
- Eliminação de registros de exames médicos desatualizados ou duplicados.

Edição e Atualização de Dados:

- Edição de informações do perfil dos atletas, como dados pessoais, atualização de dados biométricos e histórico de lesões.
- Atualização de diversos metadados dos vídeos (como ângulo de captura ou tipo de movimento) para melhorar a precisão na análise de IA.
- Alteração de detalhes de eventos esportivos, como mudanças no cronograma ou substituições de participantes.

Relatórios e Estatísticas:

- Geração de relatórios personalizados sobre o desempenho de atletas ao longo de um período.
- Análise estatística de padrões de lesões, identificando os fatores de risco mais comuns.
- Relatórios de evolução de performance por atleta, considerando dados genéticos, exames médicos e vídeos de treinamento.

Interação com Médicos:

- Interface para que médicos possam registrar novos exames, adicionar observações ou recomendar tratamentos.
- Acesso restrito para que médicos possam consultar o histórico médico completo de um atleta,

incluindo vídeos de incidentes durante as competições.

- Ferramenta de comunicação para que médicos possam alertar os treinadores sobre potenciais riscos ou condições que exigem atenção imediata.

Interface para Treinamento de IA:

- Upload de vídeos formatados para treinamento de modelos de IA, com integração direta para ferramentas de *machine learning*.
- Anotação manual de vídeos para treinar redes neurais em diferentes tipos de movimentos e padrões de desempenho.
- Interface para ajustar parâmetros de modelos de IA em tempo real, com base nos dados coletados.

Gerenciamento de Eventos Esportivos:

- Criação e edição de eventos esportivos, incluindo a possibilidade de definir regras, condições ambientais, e restrições específicas.
- Notificação automática para treinadores e atletas sobre alterações de cronograma ou local.

Monitoramento em Tempo Real:

- Visualização em tempo real de dados biométricos dos atletas durante as competições.
- Monitoramento contínuo das condições ambientais (temperatura, umidade) e seu impacto no desempenho.

Ferramentas de Análise e Visualização:

- Gráficos e visualizações de dados sobre a evolução do desempenho dos atletas.
- Mapas de calor para analisar movimentos específicos durante partidas ou treinos.

Integração com Dispositivos de Monitoramento:

- Sincronização automática com dispositivos de monitoramento de saúde (smartwatches, cintas cardíacas) para importar dados em tempo real.

Alertas e Notificações Personalizadas:

- Notificações automáticas sobre a necessidade de novos exames médicos, mudanças na performance, ou alertas de possíveis lesões.

1.3.3 Modelo Entidade Relacionamento (MER)

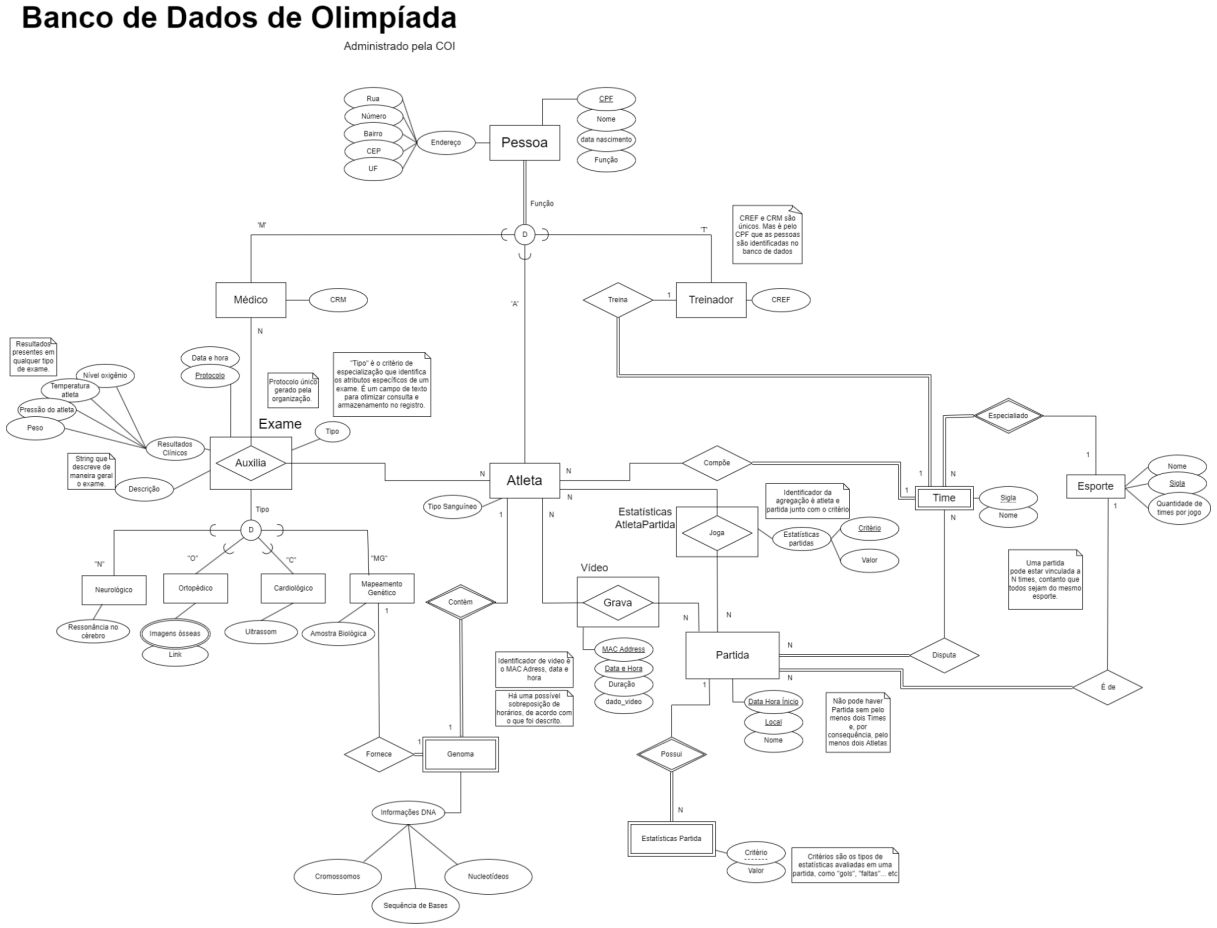


Figura 1 - Modelo Entidade Relacionamento

1.3.4 Análise de Informações Lógicas

- **Generalização de Pessoa**

Especialização total das pessoas dado o contexto do campeonato. Isso porque o **Médico**, que faz parte da comissão, não pode treinar e nem competir. Assim como para **Treinador** e **Atleta**, desempenhando papéis únicos na competição.

- **Exame**

O exame é conceituado pela agregação de atletas com os médicos, sendo que um atleta pode ser atendido por N médicos e qualquer médico pode atender qualquer atleta. Esses exames são únicos e diferenciados pelos protocolos. Contém alguns atributos que, idealmente, devem ser preenchidos na

maioria dos exames. No entanto, esses atributos podem ser nulos em algumas situações, uma vez que, ocasionalmente, pode não haver tempo suficiente ou critérios específicos para sua coleta durante o procedimento. Os principais atributos incluem: nível de oxigênio no sangue, temperatura corporal, pressão arterial e peso do atleta. Por fim, há o atributo **descrição**, que consiste em uma *string* extensa, destinada a armazenar as observações detalhadas registradas pelo médico.

- **Generalização de Exame**

Os exames podem ser do tipo bem simples, como apenas um exame local em campo, ou uma consulta simples com exames complementares, mas também podem ser alguns mais sérios e rigorosos, que caem nas especializações como cardiológica, ortopédica, neurológica ou mapeamento genético. Essas quatro, geram exames específicos com imagens e dados específicos, que são guardados no banco de dados - Os "links" em exame ortopédico são informações armazenadas no banco que referenciam a armazenamentos externos apenas para imagens (Google Drive, Dropbox, ICloud etc) -, de modo que, por exemplo, diferentes exames de um mesmo atleta podem requerer determina imagem, logo, faz sentido haver links repetidos -. No caso específico de exames ortopédicos, haverá um mapeamento só para armazenar as múltiplas imagens deste exame.

- **Ciclo entre exame Genoma-Atleta**

O atleta pode realizar até N exames, incluindo o Mapeamento Genético, que fornece dados ao genoma do atleta. E, por sua vez, o genoma armazena essas informações genéticas específicas, vinculadas exclusivamente ao atleta.

- **Relação Atleta-Time-Treinador em "treina" e "compõe"**

Ao inserir um novo time, já deve ter sido alocado no banco o treinador e os atletas correspondentes previamente – assim como o esporte já cadastrado no sistema –, pois não tem como criar o time sem que estes existam (participação total de ambos). em aplicação, é checada essa informação e assim que o time é criado, a relação "compõe" também é criada

As cardinalidades são: todo atleta pode ter apenas um time no campeonato e cada time pode ter de 1 a N atletas (no mínimo 1, ou seja, pode haver time de apenas um atleta). Cada treinador só pode treinar um time - tal como regulamentado pela COI - e cada time só pode ter um treinador (1:1). Cada time só pode ser composto por um esporte (N:1) de modo que podemos ter vários times para um mesmo esporte (divisão entre as equipes masculinas e femininas é um exemplo em que se aplica).

- **Vídeo**

Com inserção de vídeos, separados por possíveis segundos diferentes podem ser inseridos com o mesmo *MAC address*, em tempos diferentes, o que é bom, pois a ideia é que se possa haver mais de

um vídeo por mesmo aparelho, mas nunca no mesmo instante. Pode haver uma dupla camada de verificação (pertinente à aplicação) que averigua se não há sobreposição de horários e duração de vídeos - para esta maior segurança, há o armazenamento da duração do vídeo -. A cardinalidade: todo atleta pode participar de N partidas e cada partida pode haver N atletas jogando. E isso também serve para a mesma relação “joga”.

- **Ciclo Atleta-Partida-Time**

N atletas podem participar da mesma partida, desde que hajam times participantes, e também de diferentes partidas ao longo das Olimpíadas. N times podem competir entre si em uma única partida que se diferencia entre data, hora e local.

- **Ciclo entre partida vídeo e atleta**

Pode haver uma partida que o atleta jogou sem que haja nenhum vídeo. Mas não pode haver vídeo do atleta sem que esse atleta tenha obrigatoriamente jogado a partida, o que o ciclo acaba deixando acontecer. Como a inserção do vídeo no banco vem depois do atleta estar lá e a partida também, caberia à aplicação evitar isso. N atletas podem jogar N partidas ao longo de todo o campeonato (N:N)

- **Genoma**

O Genoma é uma entidade fraca pois usa atributos do atleta. Isso faz o genoma só existir caso o atleta exista, e o inverso não é verdade. a cardinalidade é 1:1 pois cada genoma só é atrelado a um atleta e vice versa. Além disso, cabe ressaltar que Genoma também é obtido de uma entidade especializada de Exame: Mapeamento Genético, e não há problemas de redundância neste formato, visto que Exame é uma agregação que tem como requisição Atleta, logo, não haverá um caso de genoma obtido por um exame sem que haja um atleta.

- **Relacionamento entre Partida Time e Esporte**

Uma partida é representada como uma entidade forte, mas que dependente logicamente de time - o que é claramente uma limitação da modelagem -, com um relacionamento de N:N. A cardinalidade indica que um time pode participar de várias partidas, e cada partida pode envolver múltiplos times. Essa configuração é válida em quase todos os contextos, pois só é possível ter partida havendo no mínimo 2 times jogando ela, isso engloba desde partidas com múltiplos times, até os esportes mais tradicionais, onde é um contra um.

Considerando que, no modelo atual, um time está associado a apenas um esporte, é importante observar que cada time deve competir somente contra times do mesmo esporte. Portanto, uma partida pode estar vinculada a N times, contanto que todos sejam do mesmo esporte.

- **Time**

Time é uma entidade fraca pois depende de Esporte, afinal, um time só faz sentido dentro do contexto de um esporte específico. Ou seja, um time é formado para competir em um esporte, e sem o esporte não há razão para o time existir. Isso significa que um Time só pode existir se estiver associado a um Esporte, mas o inverso não é verdade: um Esporte pode existir sem um Time. A cardinalidade da relação entre Esporte e Time é de 1:N, ou seja, um Esporte pode ter múltiplos Times, mas cada Time pertence a um único Esporte. A escolha de modelar Time como uma entidade fraca de Esporte garante que os times estejam sempre vinculados a um esporte, assegurando dependência lógica e integridade referencial. Isso evita a criação de times sem um esporte associado.

- **Ciclo Partida, Time e Esporte**

Este ciclo, por mais que seja um ciclo ruim e de dependência, acaba sendo necessário para que se averigue que as partidas sejam duplamente verificadas que tanto só coloquem diferentes times, times de esportes iguais, a partida em si também seja de um esporte. A relação com esporte de partida é usada para se buscar o atributo “quantidade de times por partida”, para se poder conferir em aplicação que está sendo armazenado no banco de dados a quantidade certa de times por partida

- **Estatísticas - Estatística de partida e Estatística de jogador**

Toda **partida** contém uma tabela externa que armazena as estatísticas de tal partida. Essas estatísticas são armazenadas através da chave (“partida”,” critério”,“valor”) através da entidade fraca. “critério” em si é avaliado na aplicação se é um critério válido para ser inserido no Banco de Dados.

- Partida_A , Falta_partida, número de faltas;
- Partida_A , Gols_da_partida. número de gols;
- Partida_A , Tempo_do_jogo, tempo

Paralelamente, em parecida análise, criou-se uma tabela que armazena, para cada **atleta**, em cada **partida**, suas estatísticas em jogo (“partida”,”jogador”, “critério","valor ").

- Partida_X , Jogador_A , Falta, número de faltas;
- Partida_X , Jogador_B ,Gols. número de gols;
- Partida_Y , Jogador_A ,Tempo_em_jogo, tempo

1.3.5 Restrições de Integridade

- **Inconsistência entre Partida - Time - Esporte**

Durante a análise do MER proposto, foi identificada uma possível inconsistência na definição das entidades Partida, Time e Esporte. No modelo atual, temos que Partida se relaciona com Time em um relacionamento N:N. A inconsistência encontra-se no fato de que ao associar uma partida a um time de um esporte qualquer, é irregular associar a mesma partida a um ou mais times especializados em outro esporte, o que nosso modelo atual permite. Em suma, uma primeira limitação da modelagem é que pode-se organizar uma partida com times de diferentes esportes, o que, pela regulamentação da COI, não é permitido.

Outra possível incoerência, é o fato de que é permitido associar muitos times a uma mesma partida de maneira ilógica, pois há esportes que tem um número fixo de times que disputam uma partida. Desta forma nosso modelo permite com que seja inserido uma partida com menos times do que normalmente disputam uma partida de um esporte qualquer, ou mesmo, mais times do que o esporte permite em uma partida. Por exemplo, uma partida de futebol com três times.

Essa restrição pode ser resolvida introduzindo uma lógica de validação na aplicação, fixando uma quantidade mínima e máxima de times que podem participar de uma partida para a primeira vulnerabilidade, por exemplo e garantindo que todas as partidas sejam realizadas apenas entre times que praticam o mesmo esporte, de acordo com as regras de cada disputa, por exemplo.

Adicionar a ideia de que existe um ciclo aqui (ANTES NÃO TINHA, ISSO FOI ADICIONADO NA CORREÇÃO) mas esse ciclo de dependência é importante que tenha pois faz se garantir que partida tem único esporte, e possa-se buscar a quantidade de times por partida daquela modalidade de esporte

- **Análise de ciclo Atleta-Time-Partida**

Na modelagem, há o problema de poder haver o caso em que um atleta que "joga" uma partida, mas que não está em nenhum time. O que não pode acontecer dado o contexto da aplicação. Obrigatoriamente cada atleta tem de representar (nem que seja o único atleta) um time. Esse problema tem que ser verificado no momento de inserção, pois procura-se inserir as estatísticas individuais daquele atleta. Ou seja, para jogar, o atleta tem de estar em um time obrigatoriamente.

- **Análise de ciclo Genoma-Atleta-Exame**

A princípio, olhando apenas para a modelagem, o ciclo não causa inconsistência. Isso podemos obter um genoma a partir de um atleta, mas também podemos pela entidade especializada de Exame: Mapeamento Genético. Essa entidade especializada é um Exame, que por ser uma entidade abstrata exige necessariamente a participação de um atleta. Ou seja, independente do caminho percorrido,

obtemos dados de genoma e atleta - de modo que podemos, ou não, requisitar os atributos de Exame -.

O que de fato é uma limitação, e um problema, do MER é que, no cadastro de novos atletas, é fato que ele ou ela, como ser humano, possui genoma, mas só pode ser obtido através de um exame médico. Ou seja, só será possível cadastrar no banco um novo genoma se necessariamente houver um atleta que tenha feito o exame.

Genoma armazena as informações do DNA de certo jogador através do monovalorado em cromossomos, sequência de bases e nucleotídeos.

2. Fluxograma de inserção no Banco de Dados

Inicialmente, é essencial entender a inserção de novos dados no banco, de modo que todas as informações estejam consistentes e de acordo com as requisições de dados:

1. Cria-se todas as pessoas e suas devidas especializações (com possibilidade de serem inseridas a qualquer momento depois): Pessoas e suas devidas especializações, que não podem se sobrepor, logo nenhum treinador pode ser médico ou atleta dentro desse campeonato, nem nesse banco;
2. Cada exame que for feito, em qualquer momento da olimpíada, o médico e o atleta já têm de estar cadastrados no banco. Esse exame pode ser do tipo simples (sem tipo) ou específico, como o exame neurológico, ortopédico, cardiológico e mapeamento genético, onde cada tipo de exame contém como atributo, ressonância no cérebro, imagens ósseas, ultrassom, amostra biológica, respectivamente, além do mais, é fornecido para o mapeamento genético o genoma do atleta, onde há todas as informações de seu DNA, como cromossomos, sequência de bases e nucleotídeos;
3. O genoma pode ser cadastrado a qualquer momento do campeonato, desde que haja o atleta que tenha feito um mapeamento genético e que seja feito uma única vez. O genoma é entidade fraca de atleta, pois depende de um atleta como *Owner* para a sua existência, portanto não há possibilidade de haver um genoma sem um atleta em específico;
4. Cadastra-se todas as modalidades de esportes que haverão na edição da olimpíada, ambos diferenciados por uma sigla única, discriminada pela COI.
5. Ao criar um novo time, ele pode ter atletas associados a ele, e deve conter um treinador (que já devem estar no banco anteriormente) e com relação fraca com a sigla do esporte, de modo que cada time tenha sua identificação única composta por sua sigla e a sigla do esporte

associado.

6. Ao longo do campeonato, são inseridas partidas, individualmente, que relacionam os N possíveis times. Cada partida é identificada pela chaves *Data hora início e local*. Além de haver uma participação total da partida para o time.
7. Já existindo a partida no banco, cria-se uma relação 1:1 de um atleta para essa partida em que, pela agregação, tem-se vídeos, identificados e diferenciados pela data e hora, e pelo *MAC address* do dispositivo de captura.
8. Paralelamente e independente à relação anterior, há o "joga" que relaciona os N possíveis atletas e N partidas, 1 a 1 com o armazenamento das estatísticas do atleta naquela partida. Ou seja, cada par entre "atleta" e "partida" deve ser único e deve guardar suas estatísticas.

Há a possibilidade de cruzar os dados para analisar estatísticas como vitórias da equipe pela análise individual do atleta através de seu desempenho na partida (pelo vídeo), de sua saúde física (através dos exames feitos ao longo do campeonato por médicos especializados), seu genoma através de um exame de DNA, todos para se traçar alguma correlação e estudo de casos específicos. Assim como treinamento de redes neurais para serem Inteligências Artificiais específicas e poderem ajudar em futuros acompanhamentos médicos. A importância da análise do genoma está na sua capacidade de detectar possíveis anomalias genéticas que podem oferecer vantagens ou desvantagens em relação a determinados aspectos do esporte praticado; o que também pode ser explorado por Inteligência Artificial.

Para viabilizar esse projeto, será por parte da COI o investimento de haver um sistema que grave individualmente cada atleta por jogo, como em uma partida de ping pong, onde haverá 1, 2 ou mais câmeras gravando em paralelo cada atleta. Ou pode ser uma ultra camera que grava os dois ao mesmo tempo na hora do campeonato etc. Entretanto, na hora da inserção dos vídeos no banco de dados, o vídeo deve ser recortado, individualizando assim o banco de dados com vídeos únicos de cada atleta - de modo de ajudar a rede neural a encontrar padrões únicos ao atleta. A parte de preparação de vídeos fica fora do contexto do banco. Ao serem inseridos, têm de estarem formatados para esse modelo correto.

Ademais, haverá uma enorme equipe médica acompanhando os atletas fazendo os exames, como os simples, como uma luxação em campo, ou um eventual exame cardiológico, por exemplo. Ao fim, ter-se-á, um conjunto de exames para serem analisados estatisticamente, para parametrizar relações como a qualidade do atendimento e a sua eficiência em resultados do atleta e/ou do time.

2.1 Alterações Relacionadas à Primeira Entrega

Foram encontradas inconsistências, que foram resolvidas uma a uma para melhorar a continuidade do projeto. Ajustes foram realizados em cada tópico do fluxograma de inserção.

- **Modificações:**

  Subtópico 1:

  Primeiramente, foi acrescentada uma descrição completa de todos os atributos e especializações solicitadas, visando a um melhor entendimento do relatório.

  Subtópico 2:

  Foi ajustado a interpretação sobre entidade fraca e dependência, corrigindo um equívoco cometido pelo grupo na parte teórica. Usamos um ciclo de dependência em esporte com partida e time

  Subtópico 3:

  Foi modificado para que a descrição do relatório ficasse condizente com o que está no MER.

  Subtópico 4:

  Fizemos ajustes na sintaxe e na explicação sobre o conjunto de chaves, além de corrigir as estatísticas inerentes, pois o grupo havia incluído atributos inexistentes no MER.

  Subtópico 5:

  Inicialmente, organizamos as descrições do projeto para eliminar o espalhamento das informações, de modo que o relatório ficasse mais coerente e alinhado com os tópicos. Além disso, corrigimos a especificação dos dados pessoais no MER e detalhamos, no relatório, tópico 1.3.1, quais dados pessoais estavam sendo utilizados, já que anteriormente não estava claro.

  Subtópico 6:

  Realizamos os acréscimos solicitados, adicionando notas explicativas para esclarecer algumas nuances na entidade "Pessoa". Também incluímos a descrição do atributo "Estatística de Partida" e removemos o atributo redundante "Id_Partida". Adicionamos o relacionamento "Joga" agregado com seus atributos monovalorados, "Estatísticas Partidas", além do novo atributo "Dado_Video". Foram corrigidos os valores (anteriormente) multivalorados em

genoma, e nos dois tipos de estatística, em partida e em joga. Assim, armazena-se corretamente, por critério, das estatísticas correspondentes, seja ao atleta ou à partida.

Subtópico 7:

Foi acrescentada uma explicação no subtópico "Inconsistência entre Partida - Time - Esporte", onde foram solicitados dados que definem os times em uma partida. Em resposta, adicionamos o atributo "Dado_Video" no MER. Além disso, removemos o tópico Vulnerabilidade do Exame, que apresentava uma redação pouco coesa. Por fim, fizemos um ajuste no relacionamento "Genoma-Atleta-Exame" para garantir sua consistência com o que está descrito no MER.

Subtópico 8:

Este subtópico aborda as principais mudanças no Modelo Entidade-Relacionamento (MER), focando nas entidades fracas. "Estatística de partida" foi redefinida como uma entidade fraca, dependente de "partida", e "genoma" foi caracterizado como uma entidade fraca do mapeamento genético. Essas alterações, em relação à versão inicial, trazem maior clareza nas dependências das entidades e fortalecem a coerência e robustez do modelo de banco de dados.

Ademais, foram ajustados os multivalorados.

Subtópico 9:

Modificamos o atributo “Imagens Ósseas” para tratar-se de um multivalorado. Desta forma, permitindo armazenar várias imagens associadas a um único registro sem necessidade de repetição de dados, melhorando a organização, reduzindo redundância, e facilitando a consulta e atualização dos dados.

3. Modelo Relacional

Nesta seção, introduziremos o modelo relacional da base de dados proposta, que possui como função relacionar as diversas tabelas existentes a nível físico. Além de apenas construir o esquema, também será discutido sobre os diversos mapeamentos feitos, de modo a indicar suas vantagens e desvantagens. A Figura 2 representa a imagem com o esquema relacional (ER), as tabelas são representadas por blocos e sendo relacionadas a partir das setas.

3.1 Esquema Relacional

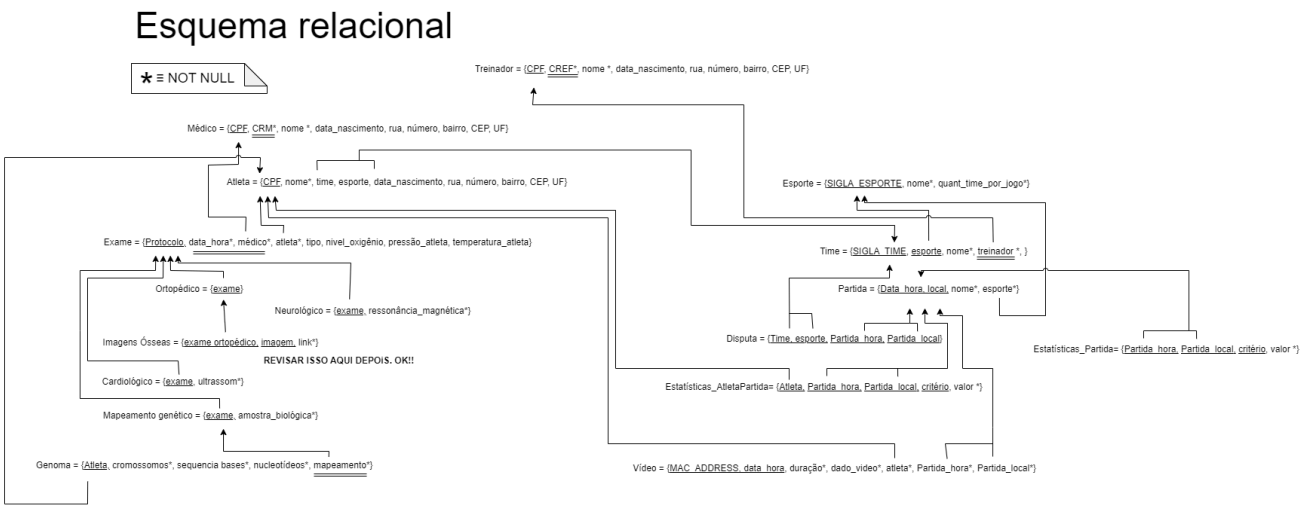


Figura 2 - Esquema Relacional

3.2 Justificativas para o mapeamento para o Modelo Relacional

3.2.1 Mapeamento de Pessoa e suas entidades especializadas

- **Solução adotada:** Visto que há participação total de Pessoa, - isto é, toda entidade especializada existente está no MER, então, toda Pessoa é Médico, Treinador ou Atleta -, optou-se por mapear apenas as entidades especializadas. Além disso, outros dois pontos importantes levados em consideração foram: a entidade genérica não se relaciona com nenhuma outra entidade que não sejam as suas especializadas; e essas possuem poucos atributos específicos.
- **Vantagens:** Tal mapeamento é útil principalmente para consulta de dados que frequentemente envolvem dados específicos das entidades especializadas, afinal, como as tabelas são independentes, as consultas podem ser mais diretas e eficientes, sem a necessidade de junção (buscar dados adicionais da entidade generalizada).
- **Desvantagens:** Uma vez que o critério de especialização (atributo ‘função’) não é mapeado nas tabelas das entidades especializadas, a solução não garante disjunção, já que não há ferramenta estrutural que impeça de um mesmo CPF, por exemplo, aparecer na tabela de Treinador e Médico. Desse modo, é necessário realizar uma checagem no SQL para que não seja possível inserir no banco de dados informações replicadas em diferentes tabelas desse contexto.
- **Soluções Alternativas:** Outro mapeamento possível que levamos em consideração foi mapear, junto com a nossa solução proposta, uma tabela de duas colunas contendo a chave de cada entidade como identificador (chave primária) e o atributo referente ao critério de especialização. Apesar deste mapeamento ser útil quando pensamos em manter a consistência

de dados, afinal, não haveria um mesmo CPF com funções distintas, por exemplo, perdemos em temos de memória e na consulta de dados, isso porque a criação de outra tabela ocupa mais espaço em disco (o que pode ser um problema grande sim a depender da escalabilidade do sistema) e torna-se necessária junções na busca de certos dados - por exemplo, consultar o endereço e de todos os médicos -, é custoso e pode acarretar na diminuição de desempenho do nosso sistema. Além disso, essa solução ainda carrega os problemas da disjunção e participação total, afinal, nada na estrutura impede, por exemplo, o CPF "123" de estar tanto na tabela em que lhe define a função de médico, quanto na tabela de Atleta, e de haver um CPF que não está associado a nenhuma das três especializadas descritas na nossa modelagem - por exemplo, CPF "123" aparece em Atleta mas não na tabela adicional criada -.

Além disso, cabe citar que há mais soluções alternativas possíveis quando consideramos a literatura: poderíamos ter mapeado em tabelas distintas a entidade generalizada, contendo o critério de especialização, e as especializadas com apenas o atributo identificador e seus específicos, mas o principal problemas desta solução está na questão da participação total, que não pode ser garantido, tão menos a disjunção quando consideramos apenas a estrutura; poderíamos ter mapeado apenas Pessoa com todos os atributos possíveis, o que parece ser uma solução inviável dado que cada entidade especializada se relaciona com alguma outra entidade do diagrama - inclusive, especializadas relacionam-se entre si - e aparecerão muitos valores nulos na tabela - por exemplo, para todo Atleta, o atributo CRM seria *null* para que a consistência de dados seja mantida -; nessa mesma linha, também poderíamos construir a tabela Pessoa com atributos *booleanos* que indicam a especialização, apesar de garantir a disjunção (considerando que criamos a tabela com todos os *booleanos* associados ao banco de dados) ainda nos deparamos com o problemas de memória e a presença de muitos campos nulos na tabela.

3.2.2 Mapeamento da agregação Exame e suas especializadas

- **Solução adotada:** Para esta agregação, que também será uma generalização dos tipos de exames, de modo que seu protocolo já é suficiente para identificá-lo, afinal, é um identificador único. Nesse sentido, optamos por apenas deixar os identificadores, associados a um exame, do médico e do atleta como atributos identificadores não nulos (chave secundária composta), afinal, cada tupla *médico-atleta-data_hora* compõe um exame, mas um único atleta pode realizar N exames com um mesmo médico, assim como um médico pode fazer N exames com N atletas. Nesse sentido, uma vez que temos a entidade generalizada Exame mapeada com todos os atributos, podemos mapear, abaixo, as suas entidades especializadas: Neurológico, Ortopédico, Cardiológico e Mapeamento Genético. Apesar de haverem poucos atributos

específicos, optamos por mapear as especializadas principalmente pelo relacionamento existente para Mapeamento Genético e a entidade Genoma. Isto é, há uma entidade especializada que se relaciona com outro conjunto de entidades do MER.

- **Vantagens:** A combinação de médico, atleta e data_hora como chave secundária garante a identificação única de cada exame, mesmo para múltiplos exames do mesmo par, de modo que contribui para a normalização do banco de dados, evitando redundância de informações. A entidade generalizada permite o reuso de informações comuns a todos os tipos de exames. Ao separar desta forma os exames de exames específicos (como ortopédico, Neurológico, etc), garantimos que a maior parte dos exames carreguem consigo os dados padrões, como nível de oxigênio, pressão arterial, dentre outros, contendo, na suma maioria dos exames, apenas dados leves de exames simples e corriqueiros ao longo do campeonato. Enquanto um exame específico, que é bem menos comum, pois são exames mais complexos e devem ser armazenado em uma outra tabela, guardando informações pesadas, imagens, ressonâncias etc Assim, quando se precisar de buscas e acessos nesses dados que são pouco usados ou raramente buscado deverá se fazer uma junção apenas quando for necessário, pois trata-se de casos mais específicos. Desta forma, evitando uma quantidade maior de nulos nas tabelas, quando tratar-se de exames gerais e não específicos.
- **Desvantagens:** A principal desvantagem associada a este tipo de mapeamento está na necessidade de realização de um junção com a entidade genérica quando necessárias informações completas de uma entidade especializada. Além disso, não é garantida a disjunção.
- **Soluções alternativas:** A outra solução que levamos em consideração se tratou de mapear apenas as entidades especializadas, já que os únicos relacionamentos associados à entidade generalizada são os que compõem a agregação. Ou seja, mapeia-se todas as entidades generalizadas com seus atributos específicos e genéricos (critério de especialização não é mapeado). Não haverá necessidade de junção de tabelas para realização de consultas, mas a abordagem por si só não é a melhor que selecionamos. Além disso, caso a solução fosse apenas armazenar a generalizada, haveriam muitos nulos pois deveriam haver campos para cada tipo de exame específico. Ademais, seriam raros de serem preenchidos pelo fato de a grande maioria dos exames estarem sendo simples. Outro problema seria o relacionamento de “mapeamento genético” com o genoma, o que seria direto com o exame, e não com o exame genético especializado.

3.2.3 Mapeamento do atributo multivalorado

- **Solução adotada:** Nossa modelagem apresenta apenas um único atributo multivalorado que é "imagens ósseas" na entidade Ortopédico, e seu mapeamento, na verdade é de imediato: uma vez que este tipo de exame apresenta uma quantidade dinâmica de imagens que podem ser geradas. Deste modo, optamos por construir uma tabela adicional chamada de "Imagens Ósseas" de dois campos identificadores (chave primária composta), um para referenciar o exame ortopédico e o "valor" da imagem. Tal mapeamento permite que para um mesmo exame possam ser associadas diversas imagens como resultado. Além disso, cabe ressaltar que a tabela Ortopédico necessariamente só contém um único atributo para que nossa solução seja coerente.
- **Vantagens:** Permite adicionar ou remover imagens de um exame de forma dinâmica e a chave primária composta garante a integridade referencial entre o exame e suas imagens.
- **Desvantagens:** Para obter informações sobre as imagens de um exame, é necessário realizar uma junção entre as tabelas, o que pode impactar o desempenho em consultas complexas.
- **Soluções alternativas:** Dada a natureza dinâmica de imagens ósseas para um resultado de um exame ortopédico, não foram identificadas alternativas para este mapeamento.

3.2.4 Mapeamento da entidade fraca Genoma

- **Solução adotada:** Para esta entidade fraca, foi mapeado tal como o previsto na literatura para esse tipo de entidade, de modo que, além dos seus atributos, foi mapeado apenas com o identificador de seu *owner* (Atleta) como chave primária, pois já é suficiente para identificar o genoma unicamente (cardinalidade 1:1 em entidade fraca). Além disso, é importante explicitar a presença do campo "mapeamento": nesse campo há uma chave estrangeira para o identificador de "Mapeamento Genético", haja vista o relacionamento entre as duas entidades, e tal atributo foi selecionado como chave secundária não nula para garantir a cardinalidade 1:1 e a participação total, tal como no DER.
- **Vantagens:** Poderia-se armazenar o genoma diretamente dentro de atleta, já que é único de cada atleta. Porém, esta decisão agregaria muitos dados por tupla de atleta. Portanto, compensa mais separar os dados cruciais de de atleta na tabela "atleta" e passar os dados pesados e pouco acessados de seu genoma para "genoma". Assim, quando se quiser de fato ter acesso aos dados genéticos, fazer-se-á uma junção entre

estes, e na imensa maioria das vezes em que não será necessária essa informação, não se precisará carregar uma tupla de atleta carregando vários dados pouco úteis (não carrega uma tupla gigante com todos os dados genéticos por tupla atleta).

- **Desvantagens:** A maior desvantagem é introduzir agora a necessidade de se fazer junções a cada pesquisa de genoma com atleta quando se quiser saber tanto os dados de um atleta e seu respectivo genoma. A inclusão da chave estrangeira para "Mapeamento Genético" pode introduzir redundância de dados, caso a mesma informação já esteja presente em outra tabela. Além disso, o genoma torna-se dependente da entidade "Mapeamento Genético", o que pode limitar a flexibilidade do modelo em cenários futuros.
- **Soluções alternativas:** Por se tratar de uma entidade fraca, não há outro tipo de mapeamento para Genoma quando tentamos relacionar com seu *owner*. Entretanto, o levamos em consideração como alternativa não mapear o campo "mapeamento". Isso também é possível para a cardinalidade 1:1, de modo que seria necessário, então, mapear o Genoma na tabela Mapeamento Genético, o que, na verdade, não seria a melhor alternativa em termos de estrutura do nosso modelo, afinal, não conseguimos garantir a participação total e uma boa análise semântica (nem todo exame do tipo Mapeamento Genético fornece um genoma, ou seja, há chances de aparecerem muitos campos nulos na tabela).

3.2.5 Mapeamento da agregação Vídeo

- **Solução adotada:** No mapeamento Vídeo, as chaves de Atleta e Partida funcionam como identificadores para a agregação, mas não é necessário, visto que *MAC_Address* + *Data_Hora* também identifica univocamente um vídeo. Nesse sentido, mapeamos essa entidade agregada da maneira mais simples possível de modo que traga todas as informações que precisamos dela - isso também porque não há relacionamentos com a entidade agregada.
- **Vantagens:** O mapeamento escolhido permite associar cada vídeo a um contexto específico (atleta e partida), de modo a facilitar a realização de consultas complexas, como encontrar todos os vídeos de um determinado atleta em uma determinada partida. Além disso, garante integridade referencial dos dados, evitando a existência de vídeos sem um atleta ou partida associados. Não obstante, tal solução permite que sejam armazenados os vídeos no banco de dados, que é o objetivo principal do Banco de Dados.
- **Desvantagens:** Tal mapeamento significa a criação de uma nova tabela que traz os problemas de espaço em memória (principalmente se o problema escalar muito) e de

potenciais junções em consultas. Outro problema inato é a inserção que pode ser feita com vídeos diferentes em diferentes horários, mas que se sobrepõem, o que não tem sentido semântico para o contexto dos vídeos únicos, pois cada máquina fotográfica tem um Mac address único e não pode gravar dois vídeos ao mesmo tempo. Portanto há uma necessidade a mais de, em aplicação, verificar sobreposição dos vídeos através da "data_hora" e "duração".

- **Soluções alternativas:** Daria-se para criar "vídeo" a partir apenas do relacionamento de 1 atleta e 1 partida, porém, dessa forma, se perderia a possibilidade de haver mais de 1 vídeo por atleta naquela mesma partida, o que é bem plausível, dada a quantidade de fotógrafos ser variável em uma partida no contexto do olimpíada. A solução alternativa encontrada é, para além de mapear a entidade agregada, também mapear o relacionamento entre as entidades que compõem a agregada - ou seja, teriamos uma tabela a mais da forma: *Grava = {atleta, partida}*, uma vez que o relacionamento N:N não possui atributos, de modo que essa nova chave seria a referenciada para a alguns dos identificadores da tabela da entidade agregada -. Essa solução ainda nos permite flexibilizar que há, por exemplo, mais de um vídeo por atleta, mas perdemos principalmente em questão de desempenho dada a necessidade de mais uma junção na realização de consultas, o que foi um dos fatores principais para não optarmos por tal solução.

3.2.6 Mapeamento da agregação Estatísticas AtletaPartida

- **Solução adotada:** A justificativa para este mapeamento é similar a do item anterior 3.2.5. Os identificadores de Atleta e Partida, juntamente com o critério, formam um identificador único para tal, de modo que pode ser utilizado para armazenar um histórico completo das estatísticas de um atleta ao longo de várias partidas, afinal permite-se armazenar estatísticas detalhadas para cada combinação única de atleta e partida com base em um critério. Além disso, apesar do identificador para esta tabela ser uma chave primária composta bem extensa, contendo quatro campos, e três deles com chaves estrangeiras para outras duas tabelas (Atleta e Partida), tal solução traz toda informação semântica que precisamos para identificar a entidade, ou seja, também justifica-se não haver necessidade de um identificador sintético.
- **Vantagens:** A chave primária, mesmo extensa, permanece estável visto que os atributos que a compõem não têm propensão de serem atualizados/alterados com certa frequência, isto é, não são dados dinâmicos no contexto de identificação da tupla. Isso evita problemas de integridade de dados e facilita a manutenção do banco de dados.

- **Desvantagens:** As principais desvantagens de tal mapeamento envolvem, principalmente, o fato de haver muitas combinações possíveis de atleta, partida, local e critério, de modo que a tabela pode se tornar muito grande e redundante, impactando o desempenho das consultas, e a gestão pode ficar complexa com base na escalabilidade do sistema. Além disso, a existência de vários campos que referenciam diferentes tabelas, há o problema claro da necessidade de várias junções na realização de consultas - lembrando que a junção com Partida, por exemplo, será em chave primária composta.
- **Soluções alternativas:** Uma solução alternativa cogitada envolve a criação de um *Id* sintético para essa tabela, de modo que ele seria identificador único e o que antes compunha uma chave primária composta, agora se trata de uma chave secundária de campos não nulos, para manter o sentido lógico da modelagem. Essa abordagem permite maior independência e flexibilidade para alterações futuras na estrutura dos dados, afinal, *Id 's* sintéticos não possuem significado semântico, e permanece estável mesmo que os valores de outros atributos sejam atualizados/alterados. Por outro lado, mesmo que essa solução não carregue o problema da necessidade de muitas junções, tal abordagem foi descartada visto que a chave da nossa melhor solução, mesmo que que tenha muitos atributos, é estável - o que foi prioritário quando levamos em consideração o grande volume de dados do sistema -.

3.2.7 Mapeamento da entidade fraca Time e alguns relacionamentos

- **Solução adotada:** Mapeamento direto para uma entidade fraca, de modo que sua chave primária é composta pela chave fraca da entidade e o identificador único de seu *owner* (Esporte). Dada a cardinalidade 1:1, mapeamos um atributo “treinador” (chave secundária), cuja chave estrangeira aponta para o identificador de Treinador, e é não nulo para garantir a participação total de Time.
- **Vantagens:** A relação deixa claro que um “Time” só pode existir dentro do contexto de um “Esporte”, ou seja, garante que time é uma entidade fraca, com *owner* a entidade “Esporte”**.** A estrutura modular facilita a manutenção dos dados (troca de treinadores, inclusão de novos atletas, novas partidas) sem comprometer a integridade da base de dados. Tal estrutura garante diferentes times jogando diferentes esportes, como “Brasil, futebol” e “Brasil, vôlei” serem identificadores diferentes, por exemplo, isso devido a necessidade semântica.
- **Desvantagens:** Se um time participa de mais de um esporte, essa estrutura pode não suportar essa situação sem ajustes, já que o “Time” está amarrado a um único “Esporte”.

- **Soluções alternativas:** Uma vez que o mapeamento de entidade fraca e o relacionamento 1:1 são imediatos, poderíamos pensar em uma alternativa, na verdade, para o relacionamento entre Time e Atleta, de maneira que criaria-se uma tabela adicional que mapeia o relacionamento entre essas entidades. Isso permitiria garantir a participação total de forma mais explícita, mas a presença de junção para realizar descredibiliza tal solução quando optamos pelo desempenho da busca/consulta.

3.2.8 Mapeamento do relacionamento entre Time e Atleta

- **Solução adotada:** Quanto ao relacionamento N:1 com Atleta, "time" foi mapeado na tabela Atleta (chave estrangeira que aponta para identificador de Time) como justificativa da análise semântica que garante a cardinalidade - mas não é possível garantir estruturalmente a participação total de Time neste caso, sendo necessário, então, garantir tal requisito em aplicação.
- **Vantagens:** Consultas para obter informações sobre a disciplina de um time são mais diretas, pois a informação já está na mesma tabela Atleta. Além disso, para obter informações sobre atletas e seus times, são necessárias poucas junções entre tabelas, o que pode melhorar o desempenho em consultas complexas**.**
- **Desvantagens:** A garantia da participação total no relacionamento com Atleta é feita em aplicação, e não por restrições de integridade de dados no banco de dados. Isso pode levar a inconsistências se o tratamento não for adequado.
- **Soluções alternativas:** Um mapeamento alternativo para este caso que levamos em consideração foi mapear o relacionamento entre Atleta e Time, de modo que o identificador de Atleta iria compor a chave primária, enquanto que o campo que referencia Time teria a restrição de não nulo para ter sentido semântico (cardinalidade). Essa solução, apesar de aparentar ter maior flexibilidade - caso se queira adicionar informações adicionais entre um atleta e seu time -, ela não foi considerada tão vantajosa quanto a selecionada nos quesitos de desempenho, afinal, a depender das consultas, seriam necessárias mais junções; complexidade, devido a presença de uma entidade associativa; e por questões de armazenamento considerando a escalabilidade do sistema.

3.2.9 Mapeamento do relacionamento entre Time e Partida

- **Solução adotada:** Nesta solução, uma partida pode envolver múltiplos times e um

time pode participar de múltiplas partidas.

- **Vantagens:** A relação entre “Time” e “Disputa” permite que um time esteja associado a múltiplas partidas, sem replicar as informações do time em cada uma delas. Mantendo, assim, os dados centralizados.
- **Desvantagens:** As principais desvantagens em se criar uma nova tabela envolvem armazenamento em disco dos dados, que aumentaram, e a necessidade de mais junções na hora da busca/consulta. Além disso, esse tipo de mapeamento, apesar de garantir a cardinalidade N:N, não garante a participação total de “Partida”, sendo esta necessária de se garantir apenas em aplicação.
- **Soluções alternativas:** Não foram identificadas alternativas para este mapeamento.

3.2.10 Mapeamento do relacionamento entre Partida e Esporte

- **Solução adotada:** Na solução adotada, “Esporte” é um campo não nulo da entidade ”Partida”, o que significa que cada partida deve estar associada a um esporte específico. Não é tratado como uma chave, ou seja, não é único - afinal, uma partida entre os times Corinthians e o Palmeiras pode ocorrer tanto no futebol quanto no basquete, por exemplo -, mas como uma característica obrigatória que identifica o tipo de esporte relacionado àquela partida.
- **Vantagens:** Como o esporte é obrigatório, garante-se que nenhuma partida seja cadastrada sem um esporte associado, o que melhora a integridade dos dados. Ao manter o esporte como atributo, é possível filtrar rapidamente as partidas por esporte em consultas simples, o que agiliza o processamento de dados em relatórios Como o esporte está diretamente presente na entidade partida, não há necessidade de fazer uma junção entre as tabelas para obter o esporte relacionado a uma partida. Ademais, pode-se ajudar na aplicação ao montar as partidas para que cada esporte tenha uma quantidade predefinida de times. Assim, amarra na estrutura do banco os times em quantidade correta a uma partida.
- **Desvantagens:** A maior desvantagem é a redundância dos dados, visto que cada vez que uma nova partida é inserida, o campo “esporte” será repetido, ocupando mais espaço no banco de dados. Ademais, há o ciclo de dependência entre Esporte, Time e Partida, mas que é essencial para amarrar uma partida sendo inserida no banco tal que seus times participantes sejam de mesmo esporte que o esporte na tupla da partida. Porém, isso gera *overhead* na aplicação, a qual tem que verificar essas condições para a inserção condizente e consistente.
- **Soluções alternativas:** O relacionamento entre partida e esporte poderia ser removido com objetivo de eliminar o ciclo de dependência - que mudaria o relacional pois a estrutura da tabela Partida seria alterada -, mas perderia-se o controle. Então,

dada a natureza do relacionamento, não foi identificado nenhuma solução alternativa.

3.2.11 Mapeamento da entidade fraca Estatísticas Partida

- **Solução adotada:** Mapeamento imediato para uma entidade fraca de cardinalidade N:1 que não apresenta outros relacionamentos com entidades além de seu *owner***.** Nesse sentido, sua chave primária composta é justificada pelo identificador fraco e o identificador de Partida. Cabe citar que essa a única solução logicamente, logo, não vantagens/desvantagens ou soluções alternativas para levar em consideração.
- **Vantagens:** Não se aplica.
- **Desvantagens:** Não se aplica.
- **Soluções alternativas:** Não se aplica.

3.3 Alterações Relacionadas à Segunda Entrega

Para esta última entrega, alguns pontos levantados foram corrigidos. A motivação para essas modificações, além de fixar pequenos erros que passam despercebidos, leva em consideração a maior familiaridade do grupo em relação ao de mapeamento. Desse modo:

- **Modificações:**

  Subtópico 1:

  Uma vez que a participação no relacionamento entre Atleta e Esporte é parcial, de fato houve um erro de mapear os atributos “time” e “esporte” na tabela Atleta, Nesse sentido, tais atributos foram alterados, agora “podendo ser atribuídos valores nulos” - mas na verdade sabemos que isso não irá acontecer se estiver referenciando a tabela Time -.

  Subtópico 2:

  Nas tabelas Médico e Atleta, respectivamente, os atributos “CRM” e “CREF” foram transformados em chaves secundárias não nulas - anteriormente só possuíam a restrição de NOT NULL, que não é suficiente dada a semântica do nosso banco -, visto que são identificadores únicos mas que não compõem chave primária.

  Subtópico 3:

  Na tabela Exame, houve apenas um erro de atenção, de modo que o atributo “nivel_oxigenio” estava duplicado. Tal atributo duplicado foi retirado do mapeamento.

Subtópico 4:

Foram consertadas algumas inconsistências entre o mapeamento e o DER: foi adicionado ao diagrama um atributo “nome” em Partida, tal como no mapeamento; foi inserido um atributo “link” na entidade especializada exame Ortopédico, cuja função é armazenar links que referenciam armazenamentos externos para se obter as imagens ósseas - isso resultou em atualizações da seção 1.3.4, no campo sobre a entidade Exame e suas entidades especializadas; foi retirado do mapeamento um atributo “id_atleta” sintético, inconsistente com a nossa construção do diagrama.

Subtópico 5:

No item 3.2.2 foi explicitado mais claramente que para a solução adotada não há garantia de disjunção. Além disso, tal fato também se aplica à solução alternativa.

Subtópico 6:

No item 3.2.5, a solução alternativa foi atualizada para um mapeamento tanto da entidade agregada quanto do relacionamento N:N entre as entidades que compõem a agregada.

Subtópico 7:

No item 3.2.6, foram atualizadas as justificativas da solução adotada e suas principais desvantagens, de modo que também foi comentado o porquê do não uso de um *id* sintético, mas apenas como um ponto a mais na justificativa.

Subtópico 8:

No item 3.2.7, as justificativas para o mapeamento do relacionamento entre Time e Atleta foi retirado do item 3.2.7, de modo que foi construído uma sessão própria para a análise desse relacionamento, que será item 3.2.8 - como consequência disso, os resto dos itens da seção 3.2 foram incrementados, totalizando, no final, 11 itens ao invés de 10, como foi para a segunda entrega -.

4. Implementação

Nesta seção, introduziremos a implementação do banco de dados e o protótipo de aplicação plataforma de monitoramento de atletas olímpicos.

A aplicação foi implementada utilizando a linguagem Python, enquanto para o banco de

dados foi utilizado o SGBD relacional Oracle Developer. A implementação com todo o conteúdo criado para esse projeto pode ser visualizado no repositório do Github desenvolvido pelo grupo.

Nesta etapa do projeto, foi montado o esquema da base de dados (***esquema.sql***) - consistindo em todas as tabelas, com seus devidos tratamentos de consistência -, um script de alimentação da base de dados (***dados.sql***) e seis consultas - de complexidade média e alta - que foram formuladas para teste (***consultas.sql***) e um protótipo inicial, integrado à base de dados (***app.py***).

4.1. Esquema

O esquema SQL detalhado para a criação e manutenção do banco de dados define as entidades, seus atributos e os relacionamentos entre elas, fornecendo uma base sólida para armazenar e gerenciar as informações relacionadas aos atletas, treinadores, equipes, partidas, exames e os outros dados relevantes.

Inicialmente, observamos que todos os atributos que compõem as chaves primárias de todas as tabelas criadas foram implementados em NOT NULL, apenas para manter um padrão. Para Oracle, isso é optativo, já que toda primária, por *default*, é não nula e, portanto, o gerenciador não permite a inserção sem esse campos em específico.

```
---------------------- Criação de médico --------------------
CREATE TABLE MEDICO (
    CPF CHAR (14) NOT NULL, -- 11 dígitos para o cpf e 4 para ponto e
hífen
    CRM CHAR (6) NOT NULL, -- 6 dígitos para formar um CRM, segundo a
Organização Brasileira de Médicos
    NOME VARCHAR2 (50) NOT NULL, -- Espera-se que 50 dígitos sejam
suficientes para armazenar um nome
    DATA DATE, -- Data de nascimento do cidadão
    RUA VARCHAR2 (50) , -- nome da rua
    NUMERO NUMBER (4) , -- número da residência (0 à 9999)
    BAIRRO VARCHAR2 (25) , -- Bairro do endereço - espera-se que 25
caracteres seja suficiente para indicar o bairro
    CEP VARCHAR (9) , -- CEP do endereço com um formato especifico de 8
dígitos e 1 traço
    UF CHAR (2) , -- Estado do cidadão 2 caracteres são suficiente para
cobrir todos os estados brasileiros
    CONSTRAINT PK_CPF_MEDICO PRIMARY KEY (CPF) , -- declaração de chave
primária para o CPF
    CONSTRAINT SK_UNIQUE_MEDICO UNIQUE (CRM) , -- Garantir chave
secundária
    CONSTRAINT CK_CPF_MEDICO CHECK (REGEXP_LIKE (CPF,
'[0-9]{3}\.[0-9]{3}\.[0-9]{3}\-[0-9]{2}') ) , -- Formatação de CPF no
devido formato
```

```
    CONSTRAINT CK_CRM CHECK (REGEXP_LIKE (CRM, '[0-9]{6}') ) , --
Formatação de CRM no devido formato
    CONSTRAINT CK_CEP_MEDICO CHECK (REGEXP_LIKE (CEP,
'[0-9]{5}\-[0-9]{3}') ) , -- formatação de CEP sob "XXXXX-YYY"
    CONSTRAINT CK_UF_MEDICO CHECK (UF IN ('AC', 'AL', 'AP', 'AM', 'BA',
'CE', 'DF', 'ES', 'GO', 'MA', 'MT', 'MS', 'MG', 'PA', 'PB', 'PR', 'PE',
'PI', 'RJ', 'RN', 'RS', 'RO', 'RR', 'SC', 'SP', 'SE', 'TO') ) --
garantir a inserção correta do estado da pessoa
);
```

```
---------------------- Criação de treinador --------------------
CREATE TABLE TREINADOR (
    CPF CHAR (14) NOT NULL, -- 11 dígitos para o cpf e 4 para ponto e
hífen
    CREF CHAR (6) NOT NULL, -- 6 dígitos para formar um CREF, segundo o
concelho do CREF
    NOME VARCHAR2 (50) NOT NULL, -- Espera-se que 50 dígitos sejam
suficientes para armazenar um nome
    DATA DATE, -- Data de nascimento do cidadão
    RUA VARCHAR2 (50) , -- nome da rua
    NUMERO NUMBER (4) , -- número da residência (0 à 9999)
    BAIRRO VARCHAR2 (25) , -- Bairro do endereço - espera-se que 25
caracteres seja suficiente para indicar o bairro
    CEP VARCHAR (9) , -- CEP do endereço com um formato especifico de 8
dígitos e 1 traço
    UF CHAR (2) , -- Estado do cidadão 2 caracteres são suficiente para
cobrir todos os estados brasileiros
    CONSTRAINT PK_CPF_TREINADOR PRIMARY KEY (CPF) , -- declaração de
chave primária para o CPF
    CONSTRAINT SK_UNIQUE_TREINADOR UNIQUE (CREF) , -- Garantir chave
secundária
    CONSTRAINT CK_CPF_TREINADOR CHECK (REGEXP_LIKE (CPF,
'[0-9]{3}\.[0-9]{3}\.[0-9]{3}\-[0-9]{2}') ) , -- Formatação de CPF no
devido formato
    CONSTRAINT CK_CREF CHECK (REGEXP_LIKE (CREF, '[0-9]{6}') ) , --
Formatação de CREF no devido formato
    CONSTRAINT CK_CEP_TREINADOR CHECK (REGEXP_LIKE (CEP,
'[0-9]{5}\-[0-9]{3}') ) , -- formatação de CEP sob "XXXXX-YYY"
    CONSTRAINT CK_UF_TREINADOR CHECK (UF IN ('AC', 'AL', 'AP', 'AM',
'BA', 'CE', 'DF', 'ES', 'GO', 'MA', 'MT', 'MS', 'MG', 'PA', 'PB', 'PR',
'PE', 'PI', 'RJ', 'RN', 'RS', 'RO', 'RR', 'SC', 'SP', 'SE', 'TO') ) --
garantir a inserção correta do estado da pessoa
);
```

```
---------------------- Criação de esporte --------------------
CREATE TABLE ESPORTE (
    SIGLA_ESPORTE CHAR (3) NOT NULL, -- Sigla única que mapeia até 999
esportes no campeonato, o que é suficiente
    NOME VARCHAR2 (50) NOT NULL, -- Nome do esporte
    QUANT_TIME_POR_JOGO NUMBER (2) NOT NULL, -- Armazena a quantidade de
times possíveis em uma competição (99 é mais que suficiente)
    CONSTRAINT CK_ESPORTE PRIMARY KEY (SIGLA_ESPORTE) -- garantia de
chave primária
);
```

```
---------------------- Criação de time --------------------
CREATE TABLE TIME (
    SIGLA_TIME CHAR (5) NOT NULL, -- Sigla única que mapeia até 99999
TIMES no campeonato, o que é mais que suficiente
    SIGLA_ESPORTE CHAR (3) NOT NULL, -- Chave estrangeira para esporte
    NOME VARCHAR2 (50) NOT NULL, -- Nome do time
    CPF_TREINADOR CHAR (14) NOT NULL, -- Chave estrangeira para o cpf do
treinador
    CONSTRAINT CK_UNIQUE_TIME UNIQUE (CPF_TREINADOR) , -- Garantir chave
secundária
    CONSTRAINT PK_TIME PRIMARY KEY (SIGLA_TIME, SIGLA_ESPORTE) , --
Chave primária da tabela
    CONSTRAINT FK_TIME_ESPORTE FOREIGN KEY (SIGLA_ESPORTE) REFERENCES
ESPORTE (SIGLA_ESPORTE) ON DELETE CASCADE, -- chave estrangeira para o
esporte. Caso seja excluido, o time não faz mais sentido estar na base
    CONSTRAINT FK_TIME_TREINADOR FOREIGN KEY (CPF_TREINADOR) REFERENCES
TREINADOR (CPF) -- chave estrangeira para o treinador. Caso o treinador
saia, o time deve continuar
);
```

```
---------------------- Criação de ATLETA --------------------
CREATE TABLE ATLETA (
    CPF CHAR (14) NOT NULL, -- 11 dígitos para o cpf e 4 para ponto e
hífen
    NOME VARCHAR2 (50) NOT NULL, -- Espera-se que 50 dígitos sejam
suficientes para armazenar um nome
    ATL_SIGLA_TIME CHAR (5), -- chave estrangeira para a sigla de time
    ATL_SIGLA_ESPORTE CHAR (3), -- Chave estrangeira para sigla esporte
de time
    DATA DATE, -- Data de nascimento do cidadão
    RUA VARCHAR2 (50) , -- nome da rua
    NUMERO NUMBER (4) , -- número da residência (0 à 9999)
    BAIRRO VARCHAR2 (25) , -- Bairro do endereço - espera-se que 25
```

```
caracteres seja suficiente para indicar o bairro
    CEP VARCHAR (9) , -- CEP do endereço com um formato especifico de 8
dígitos e 1 traço
    UF CHAR (2) , -- Estado do cidadão 2 caracteres são suficiente para
cobrir todos os estados brasileiros
    CONSTRAINT PK_CPF_ATLETA PRIMARY KEY (CPF) , -- declaração de chave
primária para o CPF
    CONSTRAINT CK_CPF_ATLETA CHECK (REGEXP_LIKE (CPF,
'[0-9]{3}\.[0-9]{3}\.[0-9]{3}\-[0-9]{2}') ) , -- Formatação de CPF no
devido formato
    CONSTRAINT FK_ATL FOREIGN KEY (ATL_SIGLA_TIME, ATL_SIGLA_ESPORTE)
REFERENCES TIME (SIGLA_TIME, SIGLA_ESPORTE) ON DELETE SET NULL, -- caso
o time ou o esporte seja removido da Base, as informações do atleta
devem permanecer no banco
    CONSTRAINT CK_CEP_ATLETA CHECK (REGEXP_LIKE (CEP,
'[0-9]{5}\-[0-9]{3}') ) , -- formatação de CEP sob "XXXXX-YYY"
    CONSTRAINT CK_UF_ATLETA CHECK (UF IN ('AC', 'AL', 'AP', 'AM', 'BA',
'CE', 'DF', 'ES', 'GO', 'MA', 'MT', 'MS', 'MG', 'PA', 'PB', 'PR', 'PE',
'PI', 'RJ', 'RN', 'RS', 'RO', 'RR', 'SC', 'SP', 'SE', 'TO') ) --
garantir a inserção correta do estado da pessoa
);
```

A partir deste ponto, é importante dizer que o grupo levou em consideração que a decisão de utilizar os métodos ON DELETE CASCADE e ON DELETE SET NULL depende diretamente da lógica e das regras de integridade do nosso sistema.

Nesse sentido, em caso de *delete* de time, optou-se por manter o registro do atleta, mas definir como nulo o campo da chave estrangeira para o time. Isso seria útil se o atleta for transferido para outro time ou se a informação sobre o time anterior for importante para análise histórica. Dessa forma, se a existência de uma entidade não depende diretamente da existência de outra, mas há uma relação entre elas, o uso do SET NULL acabou por ser mais adequado. Por sinal, foi o único ON DELETE SET NULL que o grupo julgou ser válido dentre as dependências às chaves estrangeiras das tabelas acima.

```
---------------------- Criação de PARTIDA --------------------
CREATE TABLE PARTIDA (
    DATA DATE NOT NULL, -- DATA E HORÁRIO DE COMEÇO DA PARTIDA
    LOCAL VARCHAR (50) NOT NULL, -- NOME DO LOCAL ONDE SERÁ FEITA A
COMPETIÇÃO DAQUELE ESPORTE
    NOME VARCHAR2 (100) NOT NULL, -- NOME DESCRITIVO DA PARTIDA
    SIGLA_ESPORTE CHAR (3) NOT NULL, -- nome do esporte o qual joga o
time
    CONSTRAINT PK_PAR PRIMARY KEY (DATA, LOCAL) , --  chave primária
```

```
desta tabela
    CONSTRAINT FK_PAR_ESPORTE FOREIGN KEY (SIGLA_ESPORTE) REFERENCES
ESPORTE (SIGLA_ESPORTE) -- FOI ESCOLHIDO não usar CASCADE POIS é
interessante manter os dados de histórico mesmo que uma tupla de esporte
seja excluida
);
```

```
---------------------- Criação de ESTATISTICA DE PARTIDA
--------------------
CREATE TABLE ESTAT_PARTIDA (
    EST_DATA DATE NOT NULL, -- horario relativo à respectiva partida
    EST_LOCAL VARCHAR2 (50) NOT NULL, -- local respectivo à partida em
que essa estatistica foi coletada
    CRITERIO VARCHAR (20) NOT NULL, -- nome da estatística avaliada
àquela partida
    VALOR NUMBER (4) NOT NULL, -- Acredita-se que as estátisticas aqui
sejam todas suficientes até o valor 9999
    CONSTRAINT PK_ESTAT_PARTIDA PRIMARY KEY (EST_DATA, EST_LOCAL,
CRITERIO) , --
    CONSTRAINT FK_ESTAT_DATA FOREIGN KEY (EST_DATA, EST_LOCAL)
REFERENCES PARTIDA (DATA, LOCAL) ON DELETE CASCADE -- faz sentido exluir
pois não tem sentido guardar a estat caso a partida seja exluida
);
```

Agora na análise de exclusão em cascata, um exemplo claro que podemos observar envolve uma disputa e uma partida: se um disputa for excluída do banco, é lógico que a partida identificada por tal, que está unicamente vinculada a essa disputa, também seja excluída, pois não faria sentido lógico no contexto da competição. Em suma, se a existência de uma entidade depende diretamente da existência de outra, o uso do CASCADE garante a consistência dos dados, já que, por exemplo, os dados de hora e local de uma partida só fazem sentido se a partida existir.

```
---------------------- Criação de DISPUTA --------------------
CREATE TABLE DISPUTA (
    SIGLA_TIME CHAR (5) NOT NULL, -- Chave estrangeira para TIME
    SIGLA_ESPORTE CHAR (3) NOT NULL, -- Chave estrangeira para esporte
    EST_DATA DATE NOT NULL, -- Chave estrangeira para DATA E HORA DA
PARTIDA
    EST_LOCAL VARCHAR2 (50) NOT NULL, -- Chave estrangeira para LOCAL
    CONSTRAINT PK_DIPUTA PRIMARY KEY (SIGLA_TIME, SIGLA_ESPORTE,
EST_DATA, EST_LOCAL) , -- chave primária
    CONSTRAINT FK_DISPUTA_TIME FOREIGN KEY (SIGLA_TIME, SIGLA_ESPORTE)
REFERENCES TIME (SIGLA_TIME, SIGLA_ESPORTE) , -- se querer guardar
```

```
histórico
    CONSTRAINT FK_DISPUTA_DATA FOREIGN KEY (EST_DATA, EST_LOCAL)
REFERENCES PARTIDA (DATA, LOCAL) ON DELETE CASCADE -- Aqui já não faz
mais sentido guardar caso a partida em si tenha sido excluida
);
```

A partir deste ponto, no qual já entendemos a lógica das implementações CASCADE e SET NULL, cabe ressaltar que a remoção do método ON DELETE na construção de várias *constraints* do esquema foi uma decisão que exigiu uma análise cuidadosa por parte do grupo, principalmente no contexto de preservar histórico de informações. Quando pensamos em manter registros mesmo após a exclusão de outros relacionados, foi fundamental garantir que certas restrições não implementaram o método: por exemplo, caso estejamos interessados em armazenar as estatísticas de um jogador de uma partida eliminada do banco de dados - essa lógica faz sentido no contexto do sistema quando expandimos que tais dados vão ser aplicados a análises e tendências envolvendo o jogador e seu vídeo (que, nessa mesma lógica, pode ser armazenado no histórico do banco caso uma partida seja exclusiva, mas não se o atleta for excluído) -. Ou seja, é possível manter um vídeo de uma partida, com as estatísticas de um atleta, mesmo que tal partida não esteja mais contida no banco. Um contraexemplo de armazenamento. agora, envolve as estatísticas de uma partida, afinal, precisamos implementar o método de ON DELETE CASCADE neste caso uma vez que nao eh logico armazenar tais estatísticas se a partida não existe mais.

Além disso, em caso de exclusão acidental ou necessidade de restaurar dados, a presença de registros históricos é crucial.

Outrossim, quando pensamos em flexibilidade, a remoção do ON DELETE permite a inclusão de novos tipos de dados relacionados sem a necessidade de modificar as *constraints* existentes. Por exemplo, ao adicionar um novo tipo de exame médico, não será necessário alterar as constraints das tabelas relacionadas. Não podemos esquecer, também, que ao longo do tempo, as regras envolvendo as Olimpíadas podem mudar e a ausência de ON DELETE oferece maior flexibilidade para adaptar o banco de dados a essas mudanças sem a necessidade de redefinir as *constraints*.

```
---------------------- Criação de ESTÁTISTICA DE ATLETA NA PARTIDA
--------------------
CREATE TABLE ESTAT_ATLETA_PARTIDA (
    ATLETA CHAR (14) NOT NULL, -- 11 dígitos para o cpf e 4 para ponto e
hífen
    EST_DATA DATE NOT NULL, -- horario relativo à respectiva partida
    EST_LOCAL VARCHAR2 (50) NOT NULL, -- local respectivo à partida em
que essa estatistica foi coletada
    CRITERIO VARCHAR (20) NOT NULL, -- nome da estatística avaliada
```

```
àquela partida
    VALOR NUMBER (4) NOT NULL, -- Acredita-se que as estátisticas aqui sejam todas suficientes até o valor 9999
    CONSTRAINT FK_ESTAT_ATLETA_ATLETA FOREIGN KEY (ATLETA) REFERENCES ATLETA (CPF) ON DELETE CASCADE, -- não faz sentido guardar a estatistica de um atleta que não exista na base
    CONSTRAINT FK_ESTAT_ATLETA_PARTIDA_DATA FOREIGN KEY (EST_DATA, EST_LOCAL) REFERENCES PARTIDA (DATA, LOCAL) , -- é interessante guardar os dados do atleta, mesmo que a partida tenha sido removida
    CONSTRAINT PK_ESTAT_ATLETA_PARTIDA PRIMARY KEY (ATLETA, EST_DATA, EST_LOCAL, CRITERIO) -- chave primária de estatistica atleta partida
);
```

```
---------------------- Criação de VIDEO --------------------
CREATE TABLE VIDEO (
    MAC_ADDRESS CHAR (17) NOT NULL, -- MAC address que segue o formato XX-XX-XX-XX-XX-XX
    DATA_HORA DATE NOT NULL, -- Data e hora de início do vídeo
    DURACAO NUMBER (2) NOT NULL, -- Acredita-se que qualquer vídeo seja menor que 1h39 minutos. Caso necessite de mais tempo, o vídeo deve ser dividido antes de entrar na Base de Dados
    DADO_VIDEO VARCHAR2 (255) , -- é um link para o video armazenado em outro repositório. Esta escolha é feita por escalabilidade, praticidade e custo em uma aplicação real
    ATLETA CHAR (14) NOT NULL, -- 11 dígitos para o cpf e 4 para ponto e hífen
    PARTIDA_DATA DATE NOT NULL, -- DATA E HORÁRIO DE COMEÇO DA PARTIDA
    PARTIDA_LOCAL VARCHAR (50) NOT NULL, -- NOME DO LOCAL ONDE SERÁ FEITA A COMPETIÇÃO DAQUELE ESPORTE
    CONSTRAINT PK_VIDEO PRIMARY KEY (MAC_ADDRESS, DATA_HORA) , -- chave Primária desta tabela
    CONSTRAINT FK_VIDEO_ATLETA FOREIGN KEY (ATLETA) REFERENCES ATLETA (CPF) ON DELETE CASCADE, -- deseja-se que, quando o atleta for removido, o video não exista, afinal, não faz sentido armazena-lo no banco
    CONSTRAINT FK_VIDEO_PARTIDAHORA FOREIGN KEY (PARTIDA_DATA, PARTIDA_LOCAL) REFERENCES PARTIDA (DATA, LOCAL) , -- deseja-se que, mesmo que o partida seja removido, o video continue existindo
    CONSTRAINT CK_MAC_INSERT CHECK (REGEXP_LIKE (MAC_ADDRESS, '[0-9A-F]{2}\-[0-9A-F]{2}\-[0-9A-F]{2}\-[0-9A-F]{2}\-[0-9A-F]{2}\-[0-9A-F]{2}') ) -- formatar a entrada obrigatoriamente para ser do formato: XX-XX-XXXX-XX-XX
);
```

```
---------------------- Criação de EXAME GENÉRICO --------------------
```

```
CREATE TABLE EXAME (
    PROTOCOLO CHAR (13) NOT NULL, -- PROTOCOLO no formato XXXXXXXXX-YYY
para armazenar unicamente o protocolo -  9 letras MAIUSCULAS seguidas de
3 digitos numéricos
    DATA_HORA DATE NOT NULL, -- DATA E HORA DO EXAME
    MEDICO CHAR (14) NOT NULL, -- 11 dígitos para o cpf e 4 para ponto e
hífen
    ATLETA CHAR (14) NOT NULL, -- 11 dígitos para o cpf e 4 para ponto e
hífen
    TIPO CHAR (1) NOT NULL, -- VALOR CHAR para identificar o tipo do
exame em si
    NIVEL_OXIGENIO NUMBER (2) , -- (pode ser nulo) NIVEL DE OXIGÊNIO EM
PORCENTAGEM 0 - 99%
    PRESSAO_ATLETA CHAR (7) , -- (pode ser nulo) 2 valores, uma para a
pressão baixa e outro para a pressão alta Pressão Arterial Diastólica
(PAD) e Pressão Arterial Sistólica (PAS) formato: (XXX/YYY)
    TEMPERATURA_ATLETA CHAR (4) , -- (pode ser nulo) 0.0 a 99.9 é
suficiente para armazenar a o valor em ºC de temperatura do cidadão
    CONSTRAINT PK_EXAME PRIMARY KEY (PROTOCOLO) , -- CHAVE PRIMÁRIA DA
TABELA
    CONSTRAINT SK_EXAME UNIQUE (DATA_HORA, MEDICO, ATLETA) , -- CHAVE
SECUNDÁRIA DA TABELA
    CONSTRAINT CK_EXAME_PROTOCOLO CHECK (REGEXP_LIKE (PROTOCOLO,
'[a-zA-Z]{9}\-[0-9]{3}') ) , -- PROTOCOLO DO EXAME SEGUE UM PADRÃO DE
IDENTITFICAÇÃO DE 9 LETRAS E 3 DÍGITOS
    CONSTRAINT CK_EXAME_TIPO CHECK (TIPO IN ('S', 'N', 'O', 'C', 'M') )
, -- EXAME PODE SER DE 5 TIPOS S: simples,'N'- NEUROLÓGICO, 'O' -
ORTOPÉDICO, 'C' - Cardiaco, 'M' - mapeamento genético
    CONSTRAINT CK_EXAME_PRESSAO CHECK (REGEXP_LIKE (PRESSAO_ATLETA,
'[0-9]{3}\/[0-9]{3}') ) , -- Pressão tem que seguir o formato XXX-YYY
(PAD-PAS)
    CONSTRAINT CK_EXAME_TEMP CHECK (REGEXP_LIKE (TEMPERATURA_ATLETA,
'[0-9]{2}\.[0-9]{1}') ) -- TEMPERATURA tem que seguir o formato XX.X
);
```

Sobre a construção da tabela Exame, é importante citar que o tipo de exame sempre tem que existir por causa da descrição de tal exame, que só pode ser inserida quando existir um tipo de exame (um único caractere que determina a entidade especializada ou se é exame simples), de modo que pode assumir: “S” (simples), “N” (Neurológico), “O” (Ortopédico), “C” (Cardiológico) ou “M” (Mapeamento Genético). Optamos por construir dessa maneira - sem usar o valor nulo para especificar o exame simples - ao considerar possíveis mudanças no sistema, na inserção de um novo tipo de exame, por exemplo.

```
--------------- Criação de EXAME ESPECIALIZADO ORTOPÉDICO -------------
CREATE TABLE ORTOPEDICO (
    EXAME CHAR (13) NOT NULL, -- PROTOCOLO no formato XXXXXXXXX-YYY para armazenar unicamente o protocolo -  9 letras seguidas de 3 digitos numéricos
    CONSTRAINT PK_EXAME_ORTOPEDICO PRIMARY KEY (EXAME) , -- CHAVE PRIMÁRIA DA TABELA
    CONSTRAINT FK_ORTOPEDICO FOREIGN KEY (EXAME) REFERENCES EXAME (PROTOCOLO) ON DELETE CASCADE -- NÃO FAZ SENTIDO EXISTIR O EXAME ESPECIALIZADO ORTOPÉDICO SE O EXAME ORIGINAL TIVER SIDO EXCLUÍDO
);
```

```
--------------- Criação de EXAME ESPECIALIZADO NEUROLÓGICO -------------
CREATE TABLE NEUROLOGICO (
    EXAME CHAR (13) NOT NULL, -- PROTOCOLO no formato XXXXXXXXX-YYY para armazenar unicamente o protocolo -  9 letras seguidas de 3 digitos numéricos
    RESSONANCIA_MAGNETICA VARCHAR2 (255) NOT NULL, -- LINK PARA O ENDEREÇO ONDE ESTÁ O ARQUIVO VOLUMOSO DE DADOS DE RESSONCIA MAGNÉTICA DESTE EXAME ESPECIALIZADO
    CONSTRAINT PK_EXAME_NEUROLOGICO PRIMARY KEY (EXAME) , -- CHAVE PRIMÁRIA DA TABELA
    CONSTRAINT FK_NEUROLOGICO FOREIGN KEY (EXAME) REFERENCES EXAME (PROTOCOLO) ON DELETE CASCADE -- NÃO FAZ SENTIDO EXISTIR O EXAME ESPECIALIZADO NEUROLÓGICO SE O EXAME ORIGINAL TIVER SIDO EXCLUÍDO
);
```

```
---------------- Criação de EXAME ESPECIALIZADO CARDIOLÓGICO ---------------
CREATE TABLE CARDIOLOGICO (
    EXAME CHAR (13) NOT NULL, -- PROTOCOLO no formato XXXXXXXXX-YYY para armazenar unicamente o protocolo -  9 letras seguidas de 3 digitos numéricos
    ULTRASSOM VARCHAR2 (255) NOT NULL, -- LINK PARA O ENDEREÇO ONDE ESTÁ O ARQUIVO VOLUMOSO DE DADOS DA ULTRASSOM DESTE EXAME ESPECIALIZADO
    CONSTRAINT PK_EXAME_CARDIOLOGICO PRIMARY KEY (EXAME) , -- CHAVE PRIMÁRIA DA TABELA
    CONSTRAINT FK_CARDIOLOGICO FOREIGN KEY (EXAME) REFERENCES EXAME (PROTOCOLO) ON DELETE CASCADE -- NÃO FAZ SENTIDO EXISTIR O EXAME ESPECIALIZADO CARDIOLÓGICO SE O EXAME ORIGINAL TIVER SIDO EXCLUÍDO
);
```

```
-------- Criação de EXAME ESPECIALIZADO MAPEAMENTO_GENETICO ----------
```

```
CREATE TABLE MAPEAMENTO_GENETICO (
    EXAME CHAR (13) NOT NULL, -- PROTOCOLO no formato XXXXXXXXX-YYY para armazenar unicamente o protocolo -  9 letras seguidas de 3 digitos numéricos
    AMOSTRA_BIOLOGICA VARCHAR2 (255) NOT NULL, -- LINK PARA O ENDEREÇO ONDE ESTÁ O ARQUIVO VOLUMOSO DE DADOS DA AMOSTRA BIOLOGICA DESTE EXAME ESPECIALIZADO
    CONSTRAINT PK_EXAME_MAPEAMENTO_GENETICO PRIMARY KEY (EXAME) , -- CHAVE PRIMÁRIA DA TABELA
    CONSTRAINT FK_MAPEAMENTO_GENETICO FOREIGN KEY (EXAME) REFERENCES EXAME (PROTOCOLO) ON DELETE CASCADE -- NÃO FAZ SENTIDO EXISTIR O EXAME ESPECIALIZADO MAPEAMENTO GENÉTICO SE O EXAME ORIGINAL TIVER SIDO EXCLUÍDO
);
```

```
---------------------- Criação de GENOMA --------------------
CREATE TABLE GENOMA (
    ATLETA CHAR (14) NOT NULL, -- 11 dígitos para o cpf e 4 para ponto e hífen
    GENOMA VARCHAR2 (255) NOT NULL, -- LINK PARA O ENDEREÇO ONDE ESTÁ O ARQUIVO VOLUMOSO DE DADOS DO GENOMA DO ATLETA
    MAPEAMENTO CHAR (13) NOT NULL, -- PROTOCOLO no formato XXXXXXXXX-YYY para armazenar unicamente o protocolo -  9 letras seguidas de 3 digitos numéricos
    CONSTRAINT PK_GENOMA PRIMARY KEY (ATLETA) , -- NÃO FAZ SENTIDO EXISTIR O GENOMA DEputa grupo heterogêneo UMA ATLETA FORA DA BASE DE DADOS
    CONSTRAINT FK_GENOMA_ATLETA FOREIGN KEY (ATLETA) REFERENCES ATLETA (CPF) ON DELETE CASCADE, -- NÃO FAZ SENTIDO EXISTIR O GENOMA DEputa grupo heterogêneo UMA ATLETA FORA DA BASE DE DADOS
    CONSTRAINT UNIQUE_MAP UNIQUE (MAPEAMENTO) , -- só pode haver um mapeamento relacionado com um genoma, pois é único
    CONSTRAINT FK_GENOMA_MAPEAMENTO FOREIGN KEY (MAPEAMENTO) REFERENCES MAPEAMENTO_GENETICO (EXAME) ON DELETE CASCADE -- NÃO FAZ SENTIDO EXISTIR O GENOMA DE UMA ATLETA FORA DA BASE DE DADOS
);
```

```
---------------------- Criação de IMAGEM ÓSSEA --------------------
CREATE TABLE IMAGEM_OSSEA (
    EXAME_ORTOPEDICO CHAR (13) NOT NULL, -- PROTOCOLO no formato XXXXXXXXX-YYY para armazenar unicamente o protocolo -  9 letras seguidas de 3 digitos numéricos
    IMAGEM NUMBER (2) NOT NULL, -- espera-se que haja até 99 imagens por exame ortopédico ao máximo
```

```
    LINK VARCHAR2 (255) NOT NULL, -- LINK PARA O ENDEREÇO ONDE ESTÁ O
ARQUIVO VOLUMOSO DE DADOS DA ULTRASSOM DESTE EXAME ESPECIALIZADO
    CONSTRAINT PK_IMAGEM_OSSEA PRIMARY KEY (EXAME_ORTOPEDICO, IMAGEM) ,
-- CHAVE PRIMÁRIA DA TABELA
    CONSTRAINT FK_IMAGEM_OSSEA FOREIGN KEY (EXAME_ORTOPEDICO) REFERENCES
ORTOPEDICO (EXAME) ON DELETE CASCADE -- NÃO FAZ SENTIDO EXISTIR O EXAME
ESPECIALIZADO CARDIOLÓGICO SE O EXAME ORIGINAL TIVER SIDO EXCLUÍDO
);
```

Por fim, foram projetadas instruções de eliminação de tabelas, caso necessárias - claro, apenas comentadas no script para evitar *commits* após uma tabela (ou tabelas) ser deletada do banco.

```
-- -- Tabelas que dependem de outras (child tables)

 DROP TABLE ESTAT_ATLETA_PARTIDA CASCADE CONSTRAINTS;
 DROP TABLE DISPUTA CASCADE CONSTRAINTS;
 DROP TABLE ESTAT_PARTIDA CASCADE CONSTRAINTS;
 DROP TABLE ATLETA CASCADE CONSTRAINTS;

-- -- Tabelas que servem como referência para outras (parent tables)
 DROP TABLE TIME CASCADE CONSTRAINTS;
 DROP TABLE PARTIDA CASCADE CONSTRAINTS;
 DROP TABLE ESPORTE CASCADE CONSTRAINTS;
 DROP TABLE TREINADOR CASCADE CONSTRAINTS;
 DROP TABLE MEDICO CASCADE CONSTRAINTS;

 DROP TABLE video CASCADE CONSTRAINTS;
 DROP TABLE EXAME CASCADE CONSTRAINTS;
 DROP TABLE ORTOPEDICO CASCADE CONSTRAINTS;
 DROP TABLE NEUROLOGICO CASCADE CONSTRAINTS;
 DROP TABLE CARDIOLOGICO CASCADE CONSTRAINTS;
 DROP TABLE MAPEAMENTO_GENETICO CASCADE CONSTRAINTS;
 DROP TABLE GENOMA CASCADE CONSTRAINTS;
 DROP TABLE IMAGEM_OSSEA CASCADE CONSTRAINTS;
```

4.2. Consultas

4.2.1. Consulta 1

```
SELECT P.NOME AS PARTIDA, EXTRACT (MONTH FROM P.DATA) AS MÊS, A.NOME AS
ATLETA, A.DATA AS ANIVERSARIO_ATLETA, A.RUA, A.NUMERO, A.BAIRRO, A.CEP,
A.UF, G.GENOMA
FROM PARTIDA P
   JOIN DISPUTA D ON P.DATA = D.EST_DATA
   AND P.LOCAL = D.EST_LOCAL
   JOIN TIME T ON D.SIGLA_ESPORTE = T.SIGLA_ESPORTE
   AND T.SIGLA_TIME = D.SIGLA_TIME
   JOIN ATLETA A ON A.ATL_SIGLA_TIME = T.SIGLA_TIME
   LEFT JOIN GENOMA G ON G.ATLETA = A.CPF
WHERE (P.SIGLA_ESPORTE = '111'
   OR P.SIGLA_ESPORTE = '222')
   AND (P.LOCAL = 'PRAÇA DA LIBERDADE')
   AND (EXTRACT (MONTH FROM P.DATA) BETWEEN 6
   AND 11);
```

A consulta apresenta todas as partidas de futebol realizadas na Praça da Liberdade entre os meses de julho e novembro de anos olímpicos, bem como os nomes dos atletas brasileiros que participaram dessas partidas. Além disso, exibe informações adicionais como a data de nascimento, endereço e, se disponível, o link para o genoma do atleta. O uso de JOIN conecta as tabelas PARTIDA, DISPUTA, TIME, ATLETA e GENOMA com base em suas chaves estrangeiras. A cláusula LEFT JOIN garante que, mesmo que um atleta não tenha genoma registrado, ele ainda será incluído na consulta. A filtragem por EXTRACT(MONTH) limita os resultados para os meses especificados. A cláusula WHERE considera os códigos de esporte 111 e 222 para abranger futebol masculino e feminino.

4.2.2. Consulta 2

```
SELECT A.NOME AS ATLETA, A.CPF, T.NOME AS TIME, TRUNC (MONTHS_BETWEEN
(SYSDATE, A.DATA) / 12) AS IDADE, A.RUA, A.BAIRRO, A.NUMERO, A.CEP,
A.UF
FROM ATLETA A
   JOIN TIME T ON A.ATL_SIGLA_TIME = T.SIGLA_TIME
   JOIN ESTAT_ATLETA_PARTIDA E ON A.CPF = E.ATLETA
WHERE (E.EST_DATA, E.EST_LOCAL) IN (
```

```
        SELECT V.PARTIDA_DATA, V.PARTIDA_LOCAL
        FROM VIDEO V
        GROUP BY V.PARTIDA_DATA, V.PARTIDA_LOCAL
        HAVING COUNT ( * ) > 2
    )
GROUP BY A.NOME, A.CPF, T.NOME, TRUNC (MONTHS_BETWEEN (SYSDATE, A.DATA)
/ 12) , A.RUA, A.BAIRRO, A.NUMERO, A.CEP, A.UF
HAVING SUM (E.VALOR) > 3
ORDER BY A.NOME;
```

A consulta tem como objetivo identificar atletas que somaram mais de 3 gols em partidas e que possuem pelo menos 3 vídeos associados a essas mesmas partidas. Ela retorna informações detalhadas sobre os atletas, como nome, CPF, time, idade e endereço. Para isso, a consulta conecta as tabelas ATLETA, TIME e ESTAT_ATLETA_PARTIDA utilizando junções baseadas em chaves estrangeiras.

Uma subconsulta identifica partidas com mais de 2 vídeos, agrupando os registros na tabela VIDEO por data e local da partida e filtrando com a cláusula HAVING COUNT(*) > 2. Esses resultados são usados para verificar se as estatísticas de cada atleta pertencem a essas partidas qualificadas. Em seguida, a cláusula HAVING na consulta principal soma os valores de gols para cada atleta e filtra apenas aqueles cuja soma seja maior que 3.

A consulta também calcula a idade dos atletas usando a função MONTHS_BETWEEN e organiza os resultados por nome em ordem alfabética. Ao combinar junções, subconsultas e filtros, a consulta realiza uma análise detalhada, retornando dados completos e relevantes para avaliar o desempenho e a exposição dos atletas em partidas específicas.

4.2.3. Consulta 3

```
SELECT E.PROTOCOLO, E.DATA_HORA, M.NOME AS MÉDICO, A.NOME AS ATLETA,
E.NIVEL_OXIGENIO, E.PRESSAO_ATLETA, E.TEMPERATURA_ATLETA, C.ULTRASSOM
FROM EXAME E
   JOIN MEDICO M ON E.MEDICO = M.CPF
   JOIN ATLETA A ON E.ATLETA = A.CPF
   LEFT JOIN CARDIOLOGICO C ON C.EXAME = E.PROTOCOLO
WHERE E.NIVEL_OXIGENIO > 79
   AND TO_NUMBER (E.TEMPERATURA_ATLETA) BETWEEN 36 AND 37
ORDER BY E.TIPO;
```

Essa consulta retorna os exames realizados que possuem Nível de oxigênio acima de 79. Temperatura corporal entre 36 e 37 graus Celsius. Se o exame for cardiológico, também exibe o link para o ultrassom correspondente. Além disso, traz informações detalhadas do exame, como o nome do médico responsável, o atleta avaliado e a data do exame. As tabelas EXAME, MEDICO, ATLETA e CARDIOLOGICO são combinadas para trazer dados relevantes de exames e especializações. O uso da função TO_NUMBER converte a temperatura armazenada como string para um formato numérico, permitindo realizar comparações precisas. A ordenação por E.TIPO categoriza os resultados, facilitando a análise de exames por tipo.

4.2.4. Consulta 4

```
SELECT T.NOME AS MODALIDADE, T.SIGLA_TIME, COUNT ( * ) AS
QUANT_PESSOAS_MAIORDE30
FROM TIME T
   JOIN ATLETA A ON A.ATL_SIGLA_TIME = T.SIGLA_TIME
   AND A.ATL_SIGLA_ESPORTE = T.SIGLA_ESPORTE
WHERE TRUNC (MONTHS_BETWEEN (SYSDATE, A.DATA) / 12) > 30
GROUP BY T.NOME, T.SIGLA_TIME
HAVING COUNT ( * ) > 1;
```

Esta consulta determina o time com o maior número de jogadores com mais de 30 anos e que tenha pelo menos 2 jogadores nessa faixa etária. A idade dos atletas é calculada usando a função MONTHS_BETWEEN(SYSDATE, A.DATA) / 12. A consulta combina as tabelas TIME e ATLETA para verificar a idade de cada atleta associado ao time. O agrupamento (GROUP BY) por nome e sigla do time permite contar os atletas maiores de 30 anos em cada time. A consulta filtra os times que possuem pelo menos dois jogadores acima de 30 anos.

4.2.5. Consulta 5

```
SELECT T.SIGLA_TIME, T.NOME, COUNT ( * ) AS PARTIDAS
FROM ESTAT_PARTIDA E
   JOIN PARTIDA P ON P.DATA = E.EST_DATA
   AND P.LOCAL = E.EST_LOCAL
   JOIN DISPUTA D ON P.DATA = D.EST_DATA
   AND P.LOCAL = D.EST_LOCAL
   RIGHT JOIN TIME T ON T.SIGLA_TIME = D.SIGLA_TIME
   AND T.SIGLA_ESPORTE = D.SIGLA_ESPORTE
WHERE P.SIGLA_ESPORTE = '222'
```

```
   AND E.CRITERIO = 'GOLS'
   AND E.VALOR > 1
GROUP BY T.SIGLA_TIME, T.NOME
HAVING COUNT (E.VALOR) <= 2;
```

Essa consulta retorna os times que participaram de pelo menos duas partidas com menos de 2 gols por jogo. A resposta inclui o nome do time, sua sigla e a contagem de partidas que atendem a esse critério. O saldo de gols por partida é obtido da tabela ESTAT_PARTIDA, com o critério GOLS. A junção entre as tabelas ESTAT_PARTIDA, PARTIDA, DISPUTA e TIME garante que todas as informações relevantes sejam conectadas. O filtro E.VALOR > 1 identifica partidas com menos de dois gols. A cláusula HAVING limita os resultados aos times que participaram de mais de duas partidas nesse critério.

4.2.6. Consulta 6

```
SELECT P.SIGLA_ESPORTE, ES.NOME AS ESPORTE, P.NOME AS NOME_PARTIDA, MAX
(E.VALOR) AS MAX_VALOR
FROM PARTIDA P
   JOIN ESTAT_PARTIDA E ON P.DATA = E.EST_DATA
   AND P.LOCAL = E.EST_LOCAL
   JOIN ESPORTE ES ON ES.SIGLA_ESPORTE = P.SIGLA_ESPORTE
WHERE E.CRITERIO = 'GOLS'
   AND (P.LOCAL = 'PRAÇA DA LIBERDADE'
   OR P.LOCAL = 'CORONEL CERVANTES')
GROUP BY P.SIGLA_ESPORTE, ES.NOME, P.NOME
HAVING MAX (E.VALOR) >= 4;
```

Essa consulta seleciona partidas com saldo de gols maior que 3 que ocorreram na Praça da Liberdade ou na Coronel Cervantes. Exibe o nome do esporte, nome da partida, local, data e o saldo máximo de gols. A junção com a tabela ESPORTE permite adicionar o nome do esporte às informações da partida. O filtro na cláusula WHERE especifica os critérios para saldo de gols (E.CRITERIO = 'GOLS') e os locais permitidos. A cláusula HAVING MAX(E.VALOR) >= 4 garante que apenas partidas com saldo de gols significativo sejam retornadas.

4.3. Aplicação

4.3.1 Descrição Geral da Implementação

O código implementa uma aplicação baseada em interface gráfica usando a biblioteca Tkinter para interagir com um banco de dados Oracle, através do módulo oracledb. Seu objetivo é fornecer um sistema para gerenciar informações relacionadas a um sistema esportivo, permitindo inserção e consulta de dados de diversas tabelas, como MEDICO, ATLETA, TIME, entre outras.

Essa aplicação está estruturada para ser amigável ao usuário e modular, com cada funcionalidade dividida em funções específicas. O sistema também inclui tratamento de erros e uso dinâmico de SQL para se adaptar a diferentes tabelas.

4.3.2. Estrutura de Navegação

A navegação é gerenciada principalmente por meio de funções que exibem diferentes menus e formulários.

Menu Principal

- A função `exibir_menu` é o ponto de entrada da aplicação.
- Ela cria a janela inicial com botões para:
- **Inserção de Dados:** Direciona o usuário para um submenu onde ele escolhe a tabela para inserir dados.
- **Consulta de Dados:** Redireciona para funcionalidades de consulta (não detalhado no código).

```
def exibir_menu():
      limpar_tela()
      Button(janela, text="Inserir Dados",
command=mostrar_submenu_insercao).pack()
      Button(janela, text="Consultar Dados").pack()
      Button(janela, text="Editar Dados (Desabilitado)",
state=DISABLED).pack()
```

4.3.3. Inserção de Dados

A inserção é o principal foco da aplicação e está organizada em três etapas principais:

- **Escolha da Tabela:**

O submenu de inserção (`mostrar_submenu_insercao`) apresenta as opções de tabelas disponíveis para inserir dados. Cada tabela está associada a uma função específica que cria o formulário correspondente.

- **Formulários de Inserção:**

  Funções como `criar_formulario_medico`, `criar_formulario_atleta`, e similares criam formulários personalizados para cada tabela.

  Os campos de entrada são gerados dinamicamente e armazenados em dicionários.

Exemplo de um formulário (para médicos):

```
def criar_formulario_medico():
      limpar_tela()
      campos = {
      "CPF": Entry(janela),
      "CRM": Entry(janela),
      "NOME": Entry(janela),
      "DATA": Entry(janela),
      }
      for label, widget in campos.items():
      Label(janela, text=label).pack()
      widget.pack()
      Button(janela, text="Salvar", command=lambda:
inserir_dados_no_banco("MEDICO", campos)).pack()
```

4.3.4. Inserção no Banco:

- A função `inserir_dados_no_banco` recebe o nome da tabela e os dados do formulário como um dicionário.
- Um comando SQL dinâmico é criado, adaptando-se à tabela selecionada.
- A conexão com o banco é gerenciada de forma robusta, com tratamento de erros via *try-except*.

```
def inserir_dados_no_banco(tabela, campos):
      valores = {chave: campo.get() for chave, campo in campos.items()}
      placeholders = ', '.join(f':{i + 1}' for i in range(len(valores)))
      colunas = ', '.join(valores.keys())
      try:
      cursor.execute(f"INSERT INTO {tabela} ({colunas}) VALUES
```

```
({placeholders})", list(valores.values()))
      conexao.commit()
      except Exception as e:
      conexao.rollback()
```

4.3.5. Consultas de Dados

A consulta é gerida por funções que exibem os dados de tabelas específicas ou realizam buscas condicionais.

Exemplo de consulta genérica:

Consultar todos os registros de uma tabela pode ser implementado com SELECT *.

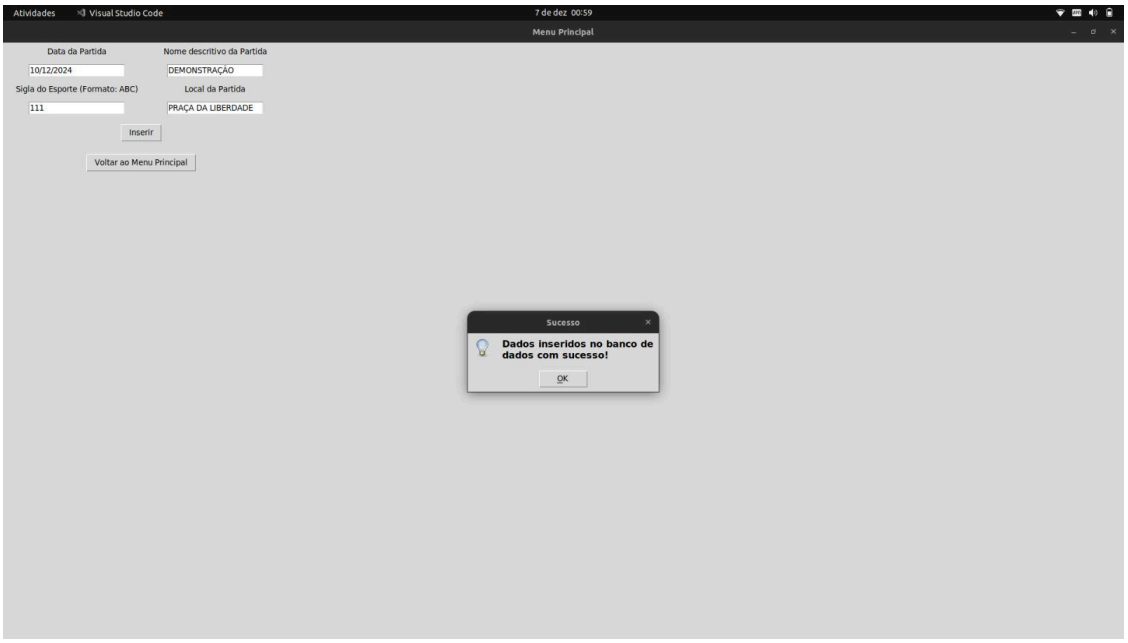
```
def exibir_selecao(tabela):
      limpar_tela()
      cursor.execute(f"SELECT * FROM {tabela}")
      resultados = cursor.fetchall()
      for linha in resultados:
      Label(janela, text=str(linha)).pack()
```

O código contém tratamento de erros básico, que é acionado principalmente durante a inserção de dados. Caso um erro ocorra, o *rollback* garante que nenhuma alteração parcial seja feita no banco.

4.3.6. Screenshots do protótipo

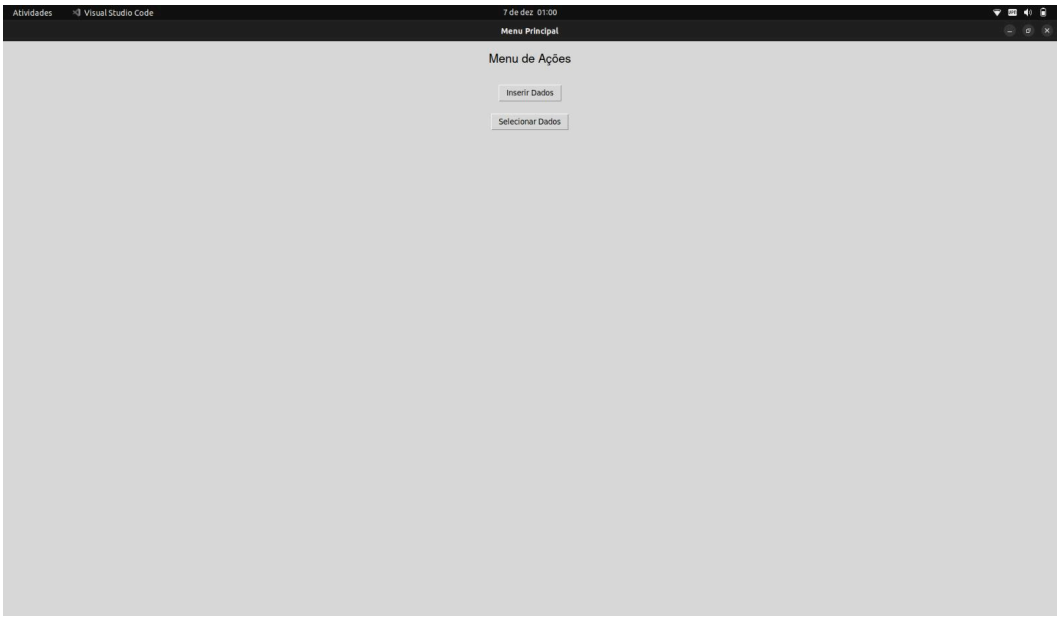
A seguir serão apresentadas algumas imagens da aplicação para exemplificar um pouco do que foi implementado. A figura 3 abaixo demonstra a inserção bem sucedida no banco de dados, relacionando com o citado acima em relação a inserção de dados.

Figura 3: Inserção de Dados



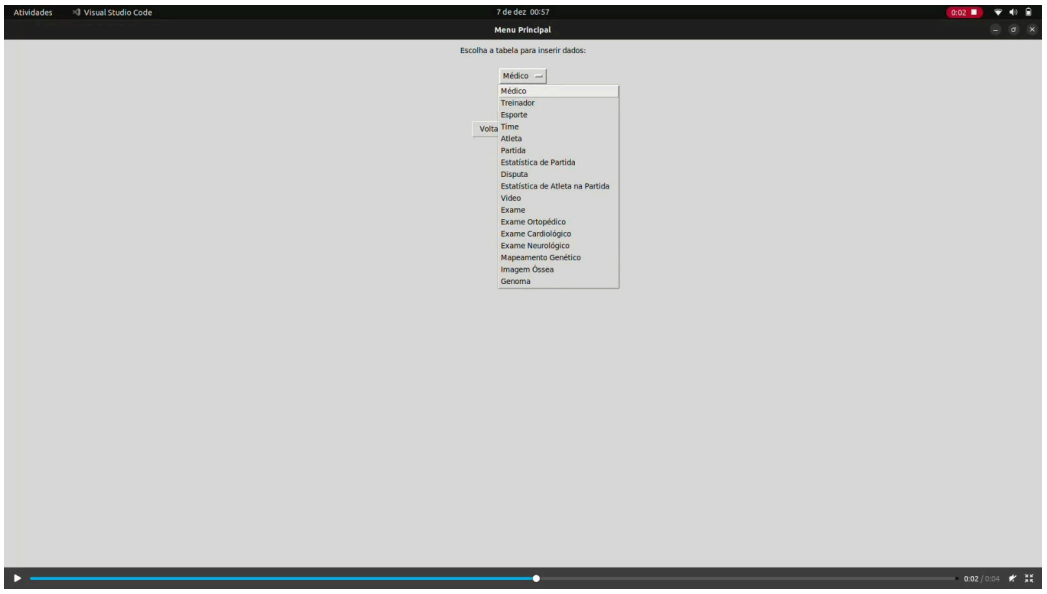
Para a figura 4, temos o layout do menu principal, onde pode- se selecionar a tabela que será inserida e também selecionar o dado que vai ser aplicado.

Figura 4: Menu principal de ações



Abaixo, na figura 5, há mais um exemplo em relação às tabelas criadas, sendo selecionadas.

figura 5: Selecionando tabelas



As figuras a seguir mostram uma seleção de todos os atletas de futebol que têm vídeos disponíveis em todas as partidas em que seu time jogou. Além disso, se o atleta possuir um genoma cadastrado, ele também será exibido. A primeira, figura 6, demonstra que não há nenhum jogador que bate com a descrição. Em seguida, são adicionados novos vídeos e, então, o resultado é exibido novamente na figura 7, agora com jogadores que cumprem os requisitos.

figura 6: resultado quando não há jogadores com vídeos cadastrados

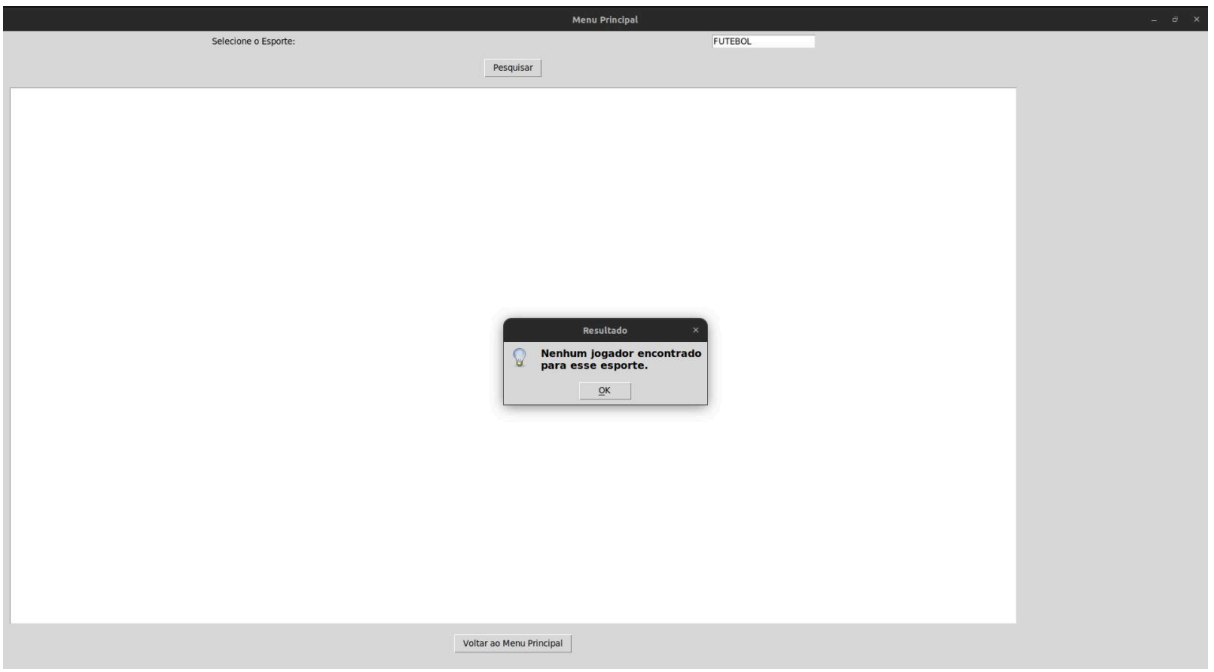
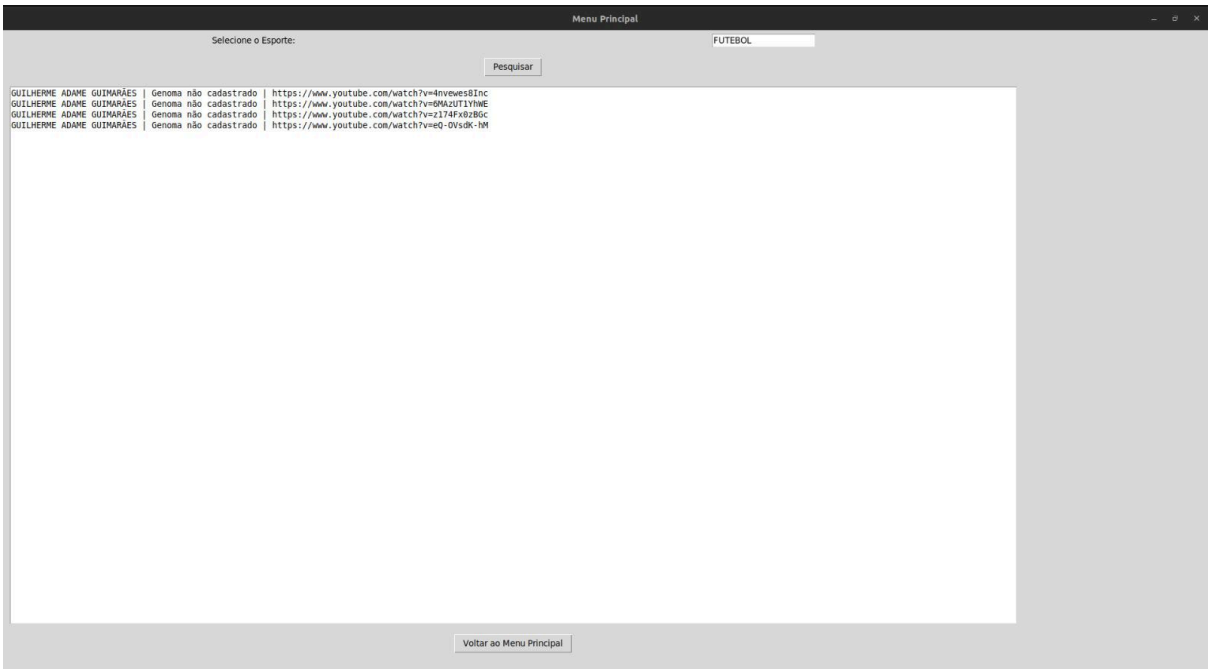


figura 7: novo resultado após inserção de vídeos



4.3.7. Problemas resolvidos exclusivamente na aplicação

Durante a estruturação da base de dados, foi identificado um problema que só poderia ser resolvido na aplicação. O problema ocorre quando dois vídeos gravados pela mesma câmera (mesmo *MacAddress*) apresentam sobreposição de horário. Embora isso seja fisicamente impossível, a base de dados permite a inserção de ambos os vídeos, pois utiliza o horário de início da gravação como chave primária. Dessa forma, vídeos que ocupam o mesmo intervalo de tempo, mas não começam exatamente no mesmo segundo, são inseridos sem conflito. A solução foi implementada na aplicação, onde uma lógica foi incorporada para verificar e impedir a inserção de vídeos com sobreposição de horário para a mesma câmera.

Outra problemática diz respeito à quantidade de times cadastrados em uma mesma partida. O banco de dados não possui uma maneira de verificar quantos times ou competidores estão associados a uma partida e se já atingiu o limite, como no caso de partidas de futebol, que requerem dois times. Para resolver isso, foi implementada uma lógica na aplicação, que ao selecionar o esporte para inserção, verifica o número máximo de times permitido para cada modalidade e checa se esse limite já foi atingido. Essa validação é feita antes da inserção de novos times, garantido a integralidade e coerência da tabela de partidas.

5. Conclusão

Durante o desenvolvimento deste projeto, foram implementadas as diversas etapas de criação e gestão de um banco de dados relacional voltado ao monitoramento de atletas olímpicos, com foco em centralizar informações complexas e diversificadas. O projeto partiu da concepção teórica de um modelo entidade-relacionamento (MER) robusto, avançando para sua tradução em um modelo relacional eficiente, até culminar na implementação prática do sistema no banco Oracle Developer.

O banco de dados foi estruturado para integrar dados de atletas, treinadores, médicos, exames, partidas, vídeos de desempenho, estatísticas e informações genéticas. A partir dessa integração, foi possível executar consultas complexas que demonstraram o potencial do sistema em responder a questões críticas do esporte de alto rendimento, como análises de desempenho, saúde e até mesmo influência genética no desempenho esportivo.As principais realizações incluem, Design e Mapeamento, Consultas Complexas, Implementação da Aplicação, Aprendizados Técnicos, Impacto no Conhecimento.

Ao fim, o projeto demonstrou o valor de uma solução integrada e centralizada para a gestão de informações esportivas, com grande potencial para suporte à tomada de decisão e desenvolvimento tecnológico no esporte. Os desafios superados e as lições aprendidas ao longo deste percurso consolidaram uma base sólida de conhecimento técnico que será de grande valia para projetos futuros.