Faculdade de Engenharia da Universidade do Porto

Jogos Olímpicos 2020

Grupo 907:

Afonso Abreu up202008552@edu.fe.up.pt

André Ávila up202006767@edu.fe.up.pt

João Malva up202006605@edu.fe.up.pt

Professor:

Michel Ferreira mpferrei@fc.up.pt

Carla Lopes ctl@fe.up.pt

Índice

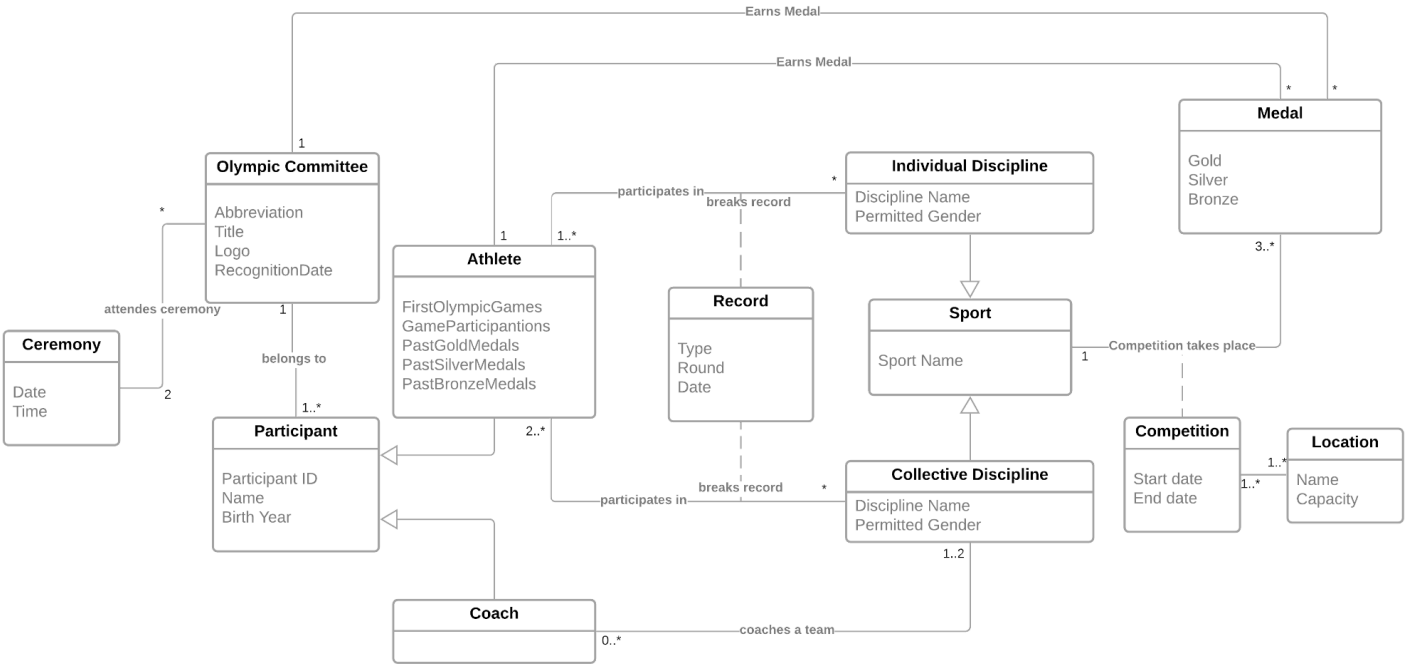
Contexto

O nosso projeto aborda o tema “Jogos Olímpicos 2020”, que decorreram em 2021, em Tóquio, no Japão. É um tema que consideramos apelativo devido ao imenso público que atrai em vários países independentemente da cultura que neles vigora. Também o facto dos Jogos Olímpicos serem um sistema de vencedores e perdedores é algo que combina com um modelo de base de dados e de registo dos mesmos, podendo incrementar o potencial deste projeto.

Este trabalho vai abordar os resultados finais das modalidades, sendo desprezados os eventos dos quais não se atribuíram medalhas, ou seja, estão apenas presentes na base dados as medalhas que um atleta ganhou e a modalidade.

A nossa base de dados deve armazenar dados sobre as várias modalidades (sports), os treinadores (coach), os atletas (athlete) o local onde decorreram os eventos (location), as cerimónias (ceremony) e os recordes batidos (records). Dos vários comités olímpicos queremos ter os atletas e os treinadores que os representam nas respetivas modalidades, para além de alguns dados únicos e de referência de cada um. Pretendemos guardar dados dos participantes das atividades Olímpicas, atletas e treinadores, como idade, género, medalhas obtidas em jogos anteriores entre outros (sendo este segundo importante devido à divisão de atletas pelo mesmo em certas categorias).

Diagrama UML



Esquema Relacional

Ceremony(idCeremony, date, time)

OlympicCommittee(abbreviation, title, logo, recognitionDate)

AttendesCeremony(abbreviation -> OlympicCommittee, idCeremony -> Ceremony)

Participant(idParticipant, abbreviation -> OlympicCommittee, name, birthYear)

Athlete(idParticipant -> Participant, firstOlympicGames, gameParticipations, pastGoldMedals, pastSilverMedals, pastBronzeMedals)

Coach(idParticipant -> Participant)

Sport(idSport, sportName)

Coaches(idParticipant -> Participant, idSport -> Sport)

ParticipatesIn(idParticipant -> Participant, idSport -> Sport)

Record(idParticipant -> Participant, idSport -> Sport, recordType, round, date)

IndividualDiscipline(idSport -> Sport, disciplineName, permittedGender)

CollectiveDiscipline(idSport -> Sport, disciplineName, permittedGender)

Competition(idSport -> Sport, startDate, endDate)

CompetitionLocation(idCompetition -> Competition, idLocation -> Location)

Location(idLocation, name, capacity)

Medal(idMedal, gold, silver, bronze)

AthleteEarnsMedal(idMedal -> Medal, idParticipant -> Participant)

CommitteeEarnsMedal(idMedal -> Medal, abbreviation -> OlympicCommittee)

Dependências Funcionais e Formas Normais

Ceremony(idCeremony, date, time)

idCeremony -> {date, time} : Obedece à 3ª forma normal e à forma de Boyce-Codd.

OlympicCommittee(abbreviation, title, logo, recognitionDate)

abbreviation -> {title, logo, recognitionDate} : Obedece à 3ª forma normal.

title -> {abbreviation, logo, recognitionDate} : Obedece à 3ª forma normal.

logo -> {title, abbreviation, recognitionDate} : Obedece à 3ª forma normal.

Participant(idParticipant, abbreviation, name, birthYear)

idParticipant -> {abbreviation, name, birthYear} : Obedece à 3ª forma normal e à forma normal de Boyce-Codd.

Athlete(idParticipant, firstOlympicGames, gameParticipations, pastGoldMedals, pastSilverMedals, pastBronzeMedals)

idParticipant -> {firstOlympicGames, gameParticipations, pastGoldMedals, pastSilverMedals, pastBronzeMedals} : Obedece à 3ª forma normal de Boyce-Codd.

Record(idParticipant, idSport, recordType, round, date)

{idParticipant, idSport} -> {recordType, round, date} : Obedece à 3ª forma normal.

Sport(idSport, sportName)

idSport -> sportName : Obedece à 3ª forma normal e a forma normal de Boyce-Codd.

IndividualDiscipline(idSport, disciplineName, permittedGender)

idSport -> {disciplineName, permittedGender} : Obedece à 3ª forma normal e à forma normal de Boyce-Codd.

CollectiveDiscipline(idSport, sportName, disciplineName, permittedGender)

idSport -> {disciplineName, permittedGender} : Obedece à 3ª forma normal e à forma normal de Boyce-Codd.

Competition(idSport, startDate, endDate)

idSport -> {startDate, endDate} : Obedece à 3ª forma normal e à forma normal de Boyce-Codd.

Location(idLocation, name, capacity)

idLocation -> {name, capacity} : Obedece à 3ª forma normal e à forma normal de Boyce-Codd.

CompetitionLocation(idCompetition, idLocation)

idSport -> {disciplineName, permittedGender} : Obedece à 3ª forma normal e à forma normal de Boyce-Codd.

Medal(idMedal, gold, silver, bronze)

idMedal -> {gold, silver, bronze} : Obedece à 3ª forma normal e à forma normal de Boyce-Codd.

AthleteEarnsMedal(idMedal, idParticipant)

idMedal -> idParticipant : Obedece à 3ª forma normal e à forma normal de Boyce-Codd.

CommitteeEarnsMedal(idMedal, abbreviation)

idMedal -> abbreviation : Obedece à 3ª forma normal e à forma normal de Boyce-Codd.

Todas estas relações obedecem à 3ª forma normal porque os seus atributos têm um nome diferente e são atómicos (1ª forma normal) e não existem dependências parciais (2ª forma normal) nem dependências transitivas.

A relações que obedecem à forma de Boyce-Codd obedecem pois contêm uma superclasse (parte esquerda da dependência funcional) que não depende de nenhum outro atributo.

Restrições

Ceremony

- idCeremony: PRIMARY KEY -> Cada cerimónia é identificada por um valor único;
- date: NOT NULL -> São conhecidas todas as datas em que as cerimónias se deram;
- time: NOT NULL -> Todas as cerimónias ocorreram a uma determinada hora conhecida.

OlympicCommittee

- abbreviation: PRIMARY KEY -> Expressão derivada do título, diferente em cada comité;
- title: NOT NULL, UNIQUE -> Todos os comités têm um título que os distingue dos outros;
- logo: NOT NULL, UNIQUE -> Os comités são todos representados pelo seu próprio logotipo.

AttendesCeremony

- idCeremony: NOT NULL, FOREIGN KEY -> Relaciona a presente classe com a classe Ceremony;
- OlympicCommitteeAbbreviation: NOT NULL, FOREIGN KEY -> Relaciona a presente classe com a classe OlympicCommittee.

Participant

- idParticipant: PRIMARY KEY -> Número de ID único do participante;
- abbreviation: NOT NULL, FOREIGN KEY -> Todos os participantes representam o Comité Olímpico a qual pertencem;
- name: NOT NULL -> Todos os participantes possuem um nome.

Athlete

- idParticipant: PRIMARY KEY, FOREIGN KEY -> Todos os atletas são participantes e, por isso, estão relacionados com a classe Participant;
- pastGOLDmedals: NOT NULL -> O atleta possui sempre 0 ou mais medalhas de ouro;
- pastSILVERmedals: NOT NULL -> O atleta possui sempre 0 ou mais medalhas de prata;
- pastBRONZEmedals: NOT NULL -> O atleta possui sempre 0 ou mais medalhas de bronze.

Coach

- idParticipant INT PRIMARY KEY, FOREIGN KEY -> Todos os treinadores também são mebros da classe Participant.

Sport

- idSport: PRIMARY KEY -> Cada desporto é identificado por um valor único;
- sportName: NOT NULL -> Cada desporto tem um nome que o representa.

Coaches

- idParticipant: NOT NULL, FOREIGN KEY -> Um participante treina pelo menos num desporto;
- idSport: NOT NULL, FOREIGN KEY -> Num desporto existe pelo menos um treinador.

ParticipatesIn

- idParticipant: NOT NULL, FOREIGN KEY -> Cada atleta participa em pelo menos um desporto;
- idSport: NOT NULL, FOREIGN KEY -> Num desporto compete pelo menos um atleta.

Record

- idParticipant: NOT NULL, FOREIGN KEY -> É sempre um atleta que bate um determinado recorde;
- idSport: NOT NULL, FOREIGN KEY -> Um recorde é associado a um desporto;
- recordType: NOT NULL -> Existe sempre um tipo de recorde batido (ex: recorde mundial, olímpico, etc.);
- round: NOT NULL -> O recorde é associado a uma ronda de uma certa modalidade;
- date: NOT NULL -> Um recorde é batido num determinado dia conhecido.

IndividualDiscipline

- idSport: NOT NULL, FOREIGN KEY -> Uma disciplina individual faz parte de um desporto e portanto, pertence à superclasse Sport;
- disciplineName: NOT NULL -> Cada modalidade tem um nome específico.

CollectiveDiscipline

- idSport: NOT NULL, FOREIGN KEY -> Uma modalidade coletiva faz parte de um desporto e portanto, pertence à superclasse Sport;
- disciplineName: NOT NULL -> Cada modalidade tem um nome específico.

Competition

- idSport: PRIMARY KEY ,FOREIGN KEY -> Cada competição está associada a um desporto. Pertence à classe Sport.

Location

- idLocation: PRIMARY KEY -> Uma localização é identificada por um valor único;
- name: NOT NULL -> Um certo nome está associado com uma localização.

CompetitionLocation

- idCompetition: NOT NULL, FOREIGN KEY -> Número não nulo que indica a competição respetiva. Pertence à classe Competition;
- idLocation: NOT NULL, FOREIGN KEY -> Uma competição é realizada num certo local. Pertence à classe Location.

Medal

- idMedal: PRIMARY KEY -> Cada medalha tem um valor único;
- gold, silver, bronze: CHECK -> Apenas vai ser atribuído o número ‘1’ a um tipo de medalha (gold ou silver ou bronze). Uma medalha não pode ser de dois tipos nem ter 0 tipos associados.

AthleteEarnsMedal

- idMedal: NOT NULL, FOREIGN KEY -> Medalha única associada ao atleta. Pertence à classe Medal;
- idParticipant: NOT NULL, FOREIGN KEY -> Um atleta pode ganhar uma ou mais medalhas (ou nenhuma). Pertence à superclasse Participant.

CommitteeEarnsMedal

- idMedal: NOT NULL, FOREIGN KEY -> Medalha única associada ao comité olímpico. Pertence à classe Medal;
- abbreviation: NOT NULL, FOREIGN KEY -> Um comité olímpico pode ganhar uma ou mais medalhas (ou nenhuma). Pertence à classe OlympicCommittee.

Interrogações

As queries criadas para este modelo tendo em conta o tema foram:

1- Atletas mais medalhados: Os nomes dos atletas mais medalhados e as suas medalhas, ordenado por ordem crescente do número de medalhas, sendo que há prioridade de ouro (gold) para prata (silver) e por fim para bronze (bronze). Nesta query destaca-se o uso da function sum() e de um case dentro da mesma, isto deve-se à necessidade de contar três colunas ao mesmo tempo, portanto o sum faz a função do count apenas quando o case lhe dá o valor 1.

2- Número de atletas de cada Comité Olímpico: As abreviaturas dos comités olímpicos seguidas do seu número de atletas, ordenado por ordem crescente das abreviaturas. Nesta query destaca-se o uso da function count().

3- Número de recordes quebrados: Número de recordes quebrados nesta edição dos Jogos Olímpicos

4- Estreantes que ganharam uma medalha de ouro: O nome dos atletas que se estrearam nesta edição dos Jogos Olímpicos e que ganharam pelo menos uma medalha de ouro, ordenado pela ordem decrescente dos seus respetivos nomes.

5- Duração dos eventos: O nome dos desportos e a duração dos seus eventos, ou seja, a diferença entre o início da primeira competição e o fim da última (caso o desporto tenha várias modalidades), ordenado pela ordem decrescente da duração e crescente do início da primeira competição em caso de empate. Nesta query destaca-se o uso da function min(), max(), cast() e julianday(), necessárias para calcular a diferença entre datas.

6- Comités reconhecidos após o nascimento de pelo menos um dos seus atletas: O título dos Comités Olímpicos em que pelo menos um dos seus atletas nasceu após a data de reconhecimento do comité ordenado pela ordem crescente do ano de reconhecimento.

7- Género nas modalidades individuais: O nome do desporto e se tem pelo menos uma modalidade em que o género permitido é, primeiramente “Men”, depois “Women” e por fim “Mixed”, cada um na sua coluna, representado por “TRUE” caso exista, “FALSE” caso não exista. É ordenado pela ordem crescente dos nomes dos desportos.

8- O primeiro atleta a bater um recorde nesta edição: O nome do primeiro atleta (ou dos primeiros caso os seus recordes sejam quebrados no mesmo dia) a bater um recorde nesta edição dos Jogos Olímpicos, ordenado pelo nome dos atletas caso haja empate.

9- A idade média dos atletas que ganharam ouro: A idade média dos atletas que ganharam pelo menos uma medalha de ouro nesta edição dos Jogos Olímpicos arredondado às centésimas. Nesta query destaca-se o uso da function round() e avg() usadas de modo a retornar o valor desejado.

10- Desportos com recordes quebrados: O nome dos desportos que tiveram pelo menos um dos seus recordes quebrados, quantos foram quebrados em modalidades individuais dos desportos, quantos foram quebrados em modalidades coletivas dos desportos e o total de recordes quebrados nos desportos. É ordenado pela ordem decrescente do total.

Gatilhos

Foram também adicionados na nossa base de dados 4 gatilhos (2 no mesmo ficheiro), que têm como função o correto funcionamento do programa e a coerência entre os dados já implementados e aqueles que serão futuramente incluídos.

1. SportElimination

Antes de ser eliminado um determinado desporto da base de dados, este gatilho irá ser ativado, ou seja, este é um gatilho do tipo *Before Delete*.

De forma a que este objeto seja eliminado, é necessário fazer alterações nas tabelas que fazem uso desse desporto primeiro, caso contrário o *sqlite* dá erro. Dessa maneira, removemos o id do respetivo objeto associado ao desporto presente nessas tabelas. Finalizando este processo, o desporto escolhido irá ser eliminado sem qualquer erro.

2. ParticipantAdittion

Neste gatilho, é verificado se um certo participante pode ou não ser adicionado à base de dados dependendo das informações que lhe atribuímos. É por isso um gatilho do tipo *Before Insert*.

Um participante tem como uns dos seus atributos o seu id e o comité olímpico associado, representado por três letras maiúsculas. Neste gatilho, o que é verificado é se o seu id está entre os id's já implementados (0 <= id atribuído <= id máximo) e também se a abreviação do seu comité de facto existe (está entre os comités existentes na base de dados). Se estas condições forem cumpridas o participante irá ser adicionado.

3. MedalUpdate e EmptyMedalUpdate

Antes da atualização do valor das medalhas, é necessária uma verificação do valor dado e é justamente essa a função deste gatilho. É, portanto, um gatilho do tipo *Before Update*.

Para uma medalha ter um valor válido, esta apenas pode conter o valor de 1 num dos três tipos de medalhas (ouro, prata e bronze) ou seja, um exemplo de um valor válido seria: gold-1, silver-0, bronze-0; e um exemplo de um valor inválido seria: gold-1, silver-1, bronze-0; pois não é possível ter uma medalha de dois tipos simultaneamente, levantando, por isso um erro no sqlite.

Também foi adicionado neste ficheiro um outro gatilho do tipo *Before Update.* No entanto, este verifica se a nossa atualização fez com que a medalha em questão não ficasse de nenhum tipo, ou seja: gold-0, silver-0, bronze-0. Se tal acontecer, a medalha é apagada da base de dados e nas tabelas nela associadas.