Armazenamento e Acesso a Dados

Instituto Politécnico do Cávado e do Ave
Escola Técnica Superior Profissional
CTeSP – Tecnologia e Inovação Informática
Ano Letivo: 2021/2022

Relatório
Trabalho Prático

Joana Freitas Pimenta – 22999

Ludgero Miguel Simões – 23135

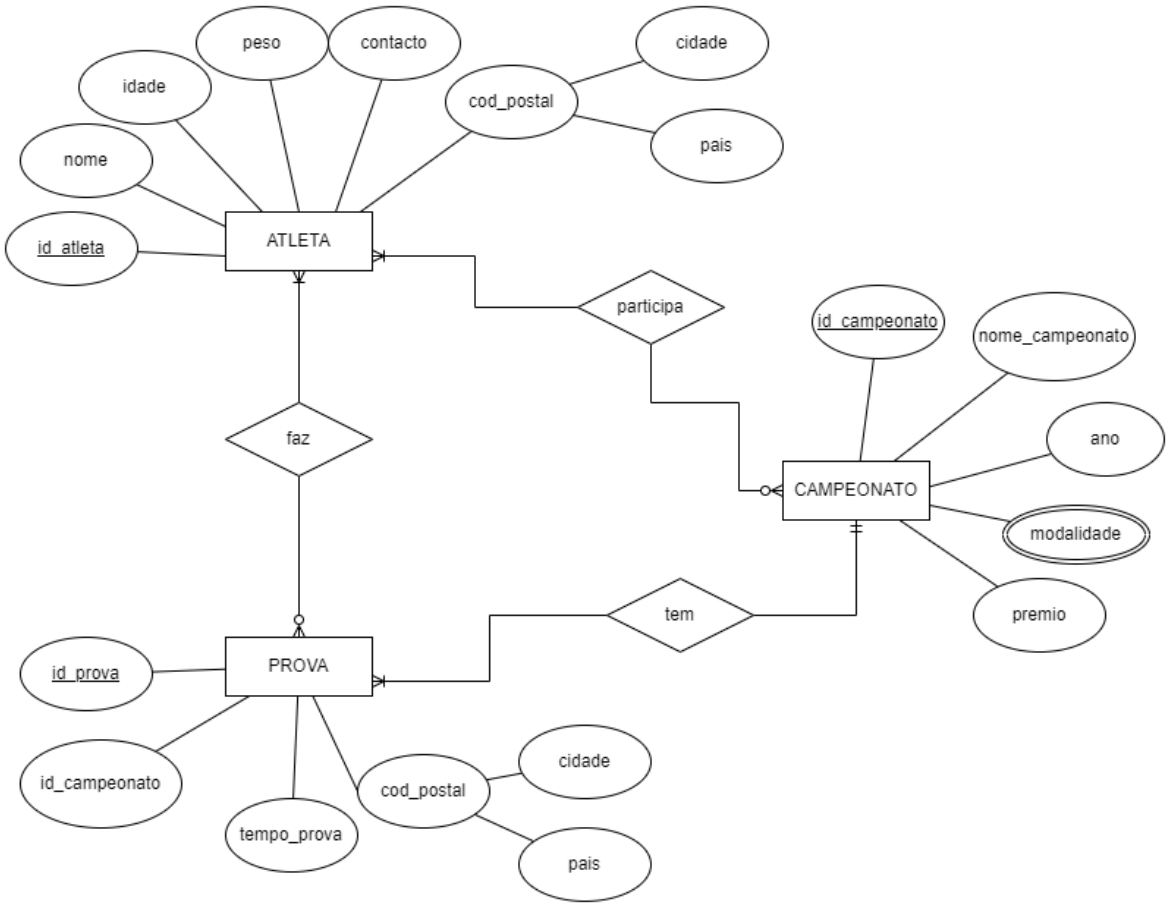
INTRODUÇÃO

Neste trabalho prático, foi nos proposto modelar e criar uma base dados segundo um tema a escolha e desenvolver instruções SQL para cumprir os requisitos definidos.

O tema que escolhemos foi um campeonato de surf, que foi desenhado, implementado e trabalhado por nós no decorrer do projeto.

Os requisitos da base dados era essencialmente ter funcionalidades tais como: obter informações do atleta, informações do campeonato, informações da prova, quais os atletas que participam em cada campeonato, entre outras funcionalidades.

DESENHO INICIAL



Acima podemos ver o diagrama ER que representa a nossa modelização inicial do problema, que consiste nas tabelas:

ATLETA(id_ateta, nome, idade, peso, contacto, cod_postal(cidade, pais), pontuacao)

CAMPEONATO(id_campeonato, nome_campeonato, ano, {modalidade}, premio)

PROVA(id_prova, id_campeonato, tempo_prova, cod_postal(cidade, pais))

NORMALIZAÇÃO

1FN

ATLETA(id_ateta, nome, idade, peso, contacto, pontuação, cod_postal)

CAMPEONATO(id_campeonato, nome_campeonato, ano, modalidade, premio)

PROVA(id_prova, id_campeonato, tempo_prova, cod_postal)

COD_POSTAL(cod_postal, cidade, pais)

2FN

ATLETA(id_ateta, nome, idade, peso, contacto, pontuacao, cod_postal)

CAMPEONATO(id_campeonato, nome_campeonato, ano, modalidade, premio)

PROVA(id_prova, id_campeonato, tempo_prova, cod_postal)

COD_POSTAL(cod_postal, cidade, pais)

3FN

ATLETA(id_ateta, nome, idade, peso, contacto, cod_postal)

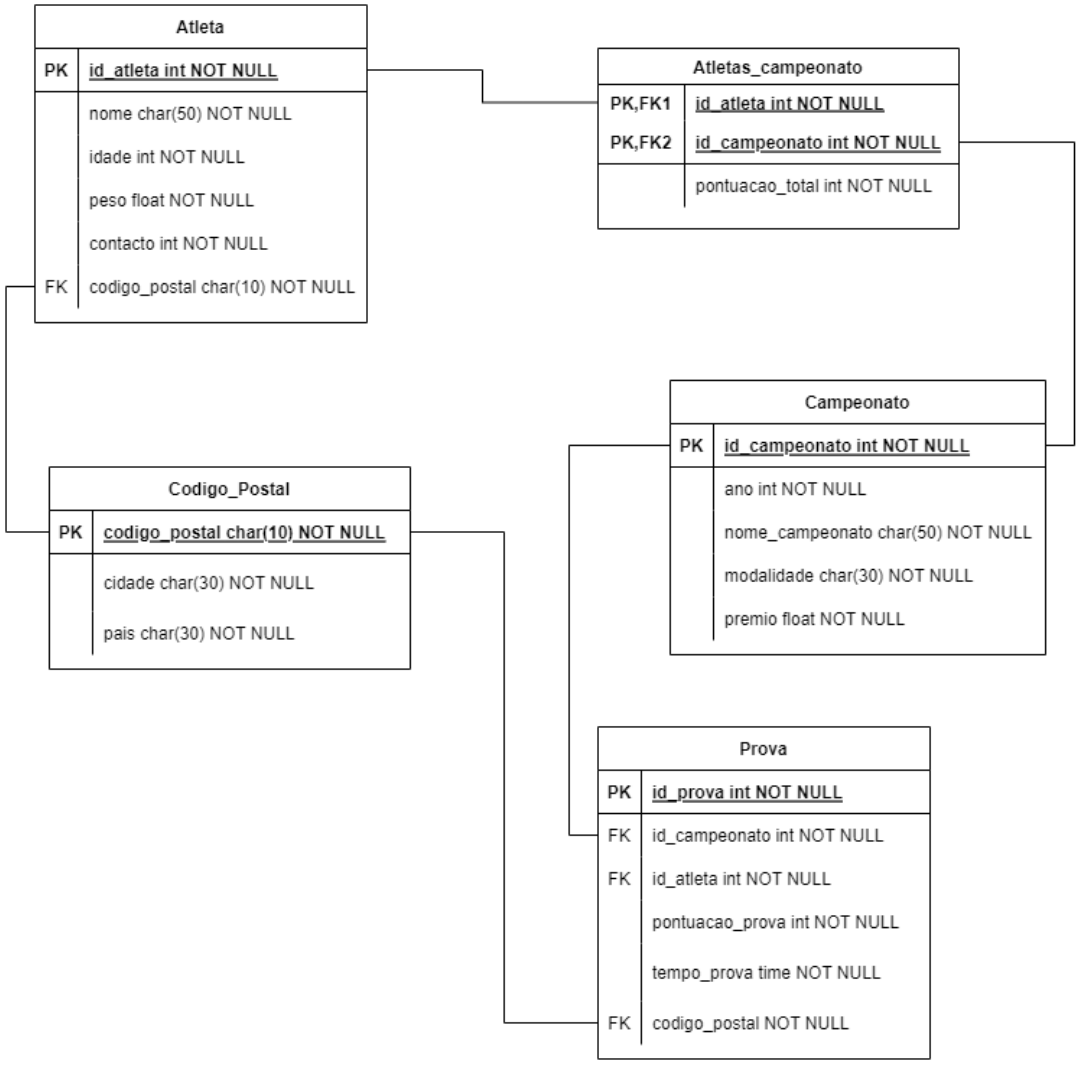
CAMPEONATO(id_campeonato, nome_campeonato, ano, modalidade, premio)

PROVA(id_prova, id_campeonato, tempo_prova, cod_postal)

COD_POSTAL(cod_postal, cidade, pais)

ATLETA_CAMPEONATO(id atleta, id_campeonato, pontuacao)

MODELO REPRESENTACIONAL



EXPLICAÇÃO DO CÓDIGO

```
1 •  DROP schema if EXISTS campeonatos_de_surf;
2 •  CREATE schema campeonatos_de_surf default character set utf8;
3 •  USE campeonatos_de_surf;
```

Elimina o esquema se existir, de seguida cria novamente, e chama-o para a área de trabalho através do USE, para dizermos ao mysql que queremos trabalhar sobre este esquema nas próximas linhas de código. A eliminação e a criação é se no caso executarmos o código novamente, este não dará erros caso o esquema já esteja criado.

```
CREATE TABLE atleta (
    id_atleta int(5) not null auto_increment,
    nome char(50) not null,
    idade int(3) not null,
    peso float(5,2) not null,
    contacto int(15) not null,
    cod_postal char(10) not null,
    primary key(`id_atleta`)
);
```

```
CREATE TABLE codigo_postal (
    cod_postal char(10) not null,
    cidade char(30) not null,
    pais char(30) not null,
    primary key(`cod_postal`)
);
```

```sql
CREATE TABLE campeonato (
    id_campeonato int(5) not null auto_increment,
    ano int(4) not null,
    nome_campeonato char(100) not null,
    modalidade char(30) not null,
    premio float(15,2) not null,
    primary key(`id_campeonato`)
);

CREATE TABLE atletas_campeonato (
    id_atleta int(5) not null,
    id_campeonato int(5) not null,
    pontuacao_total int(10) DEFAULT 0,
    primary key(`id_atleta`,`id_campeonato`)
);


CREATE TABLE prova (
    id_prova int(5) not null auto_increment,
    id_campeonato int(5) not null,
    id_atleta int(5) not null,
    cod_postal char(10) not null,
    tempo_prova time not null,
    pontuacao_prova int(4) not null,
    primary key(`id_prova`)
);
```

Criação das 5 tabelas. A tabela do código postal está presente para determinar o local de onde o atleta vive e de onde a prova será realizada, e assim através do simples código postal conseguimos saber a cidade, país associados.

Num campeonato participam vários atletas, que disputam várias provas ao longo deste. Em cada prova, os atletas que nela participam recebem uma pontuação que posteriormente é atualizada também na tabela atletas_campeonato, que é a tabela que tem a pontuação total para aquele atleta naquele campeonato.

Na tabela atletas_campeonato é essencialmente para determinar quais os atletas que participam em qual campeonato, com uma coluna também para a pontuação obtida naquele campeonato, como explicado em cima.

Na tabela prova, temos que a prova só pertence a um campeonato e prontos tem a sua pontuação para aquele prova para aquele atleta naquele campeonato.

A tabela atleta é intuitiva e são atributos de um atleta, não havendo muito para explicar.

```sql
ALTER TABLE atletas_campeonato
    ADD CONSTRAINT atletas_campeonato_fk_atleta
    FOREIGN KEY (id_atleta)
    REFERENCES atleta(id_atleta)
    ON DELETE CASCADE;

ALTER TABLE atletas_campeonato
    ADD CONSTRAINT atletas_campeonato_fk_campeonato
    FOREIGN KEY (id_campeonato)
    REFERENCES campeonato(id_campeonato);

ALTER TABLE atleta
    ADD CONSTRAINT atleta_fk_codigo_postal
    FOREIGN KEY  (cod_postal)
    REFERENCES codigo_postal(cod_postal);
```

```sql
ALTER TABLE prova
    ADD CONSTRAINT prova_fk_codigo_postal
    FOREIGN KEY  (cod_postal)
    REFERENCES codigo_postal(cod_postal);

ALTER TABLE prova
    ADD CONSTRAINT prova_fk_campeonato_e_atleta
    FOREIGN KEY  (id_campeonato,id_atleta)
    REFERENCES atletas_campeonato(id_campeonato,id_atleta);
```

Nas chaves estrangeiras, o que acho que deve ser melhor explicado é no último alter table para a tabela prova, em que temos uma chave estrangeira dupla por assim dizer, ou seja um registo só é aceite na tabela prova, se na tabela atletas_campeonato houver a conjunção do id_campeonato e id_atleta. Ou seja, é a chave se não for cumprido o requisito dará erro a inserção. Assim prevenimos de

não estar a preencher um registo para a prova e não estar antes definida na tabela atletas_campeonato, ou seja, não corremos o risco de os dados não serem coerentes, de por exemplo um atleta fez aquela prova naquele campeonato mas depois vamos a ver e outra tabela nem aparece que o atleta participa/participou no campeonato.

```
-- Criação da view para os campeonatos de um atleta ordenados
CREATE VIEW `lista_campeonatos_atleta`
AS (SELECT atleta.id_atleta, atleta.nome, campeonato.*, atletas_campeonato.pontuacao_total
    FROM atleta
    INNER JOIN atletas_campeonato
        ON atletas_campeonato.id_atleta = atleta.id_atleta
    INNER JOIN campeonato
        ON atletas_campeonato.id_campeonato = campeonato.id_campeonato
        ORDER BY id_atleta, id_campeonato);
```

View para os campeonatos que um atleta já participou, vindo com informações sobre o atleta, sobre o campeonato e claro a sua pontuação. Dá este output:

```
608 •  SELECT * FROM lista_campeonatos_atleta;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id_atleta	nome	id_campeonato	ano	nome_campeonato	modalidade	premio	pontuacao_total
1	Gaye	1	2014	vestibulum velit id pretium iaculis diam erat ferm...	Skate	12600.76	524
2	Mervin	1	2014	vestibulum velit id pretium iaculis diam erat ferm...	Skate	12600.76	0
2	Mervin	2	2022	hac habitasse platea dictumst etiam faucibus cu...	Bodysurf	21382.11	1650
3	Billy	5	2018	nullam molestie nibh in lectus pellentesque at nu...	Skate	176776.86	200
3	Billy	13	2012	nam ultrices libero non mattis pulvinar nulla ped...	Stand up paddle	133080.34	0
5	Colet	1	2014	vestibulum velit id pretium iaculis diam erat ferm...	Skate	12600.76	0
6	Amargo	6	2018	est phasellus sit amet erat nulla tempus	Longboard	391497.47	0
7	Curran	5	2018	nullam molestie nibh in lectus pellentesque at nu...	Skate	176776.86	0
7	Curran	7	2021	nulla eget eros elementum pellentesque quisque...	Kneeboarding	70627.03	548
8	Deva	6	2018	est phasellus sit amet erat nulla tempus	Longboard	391497.47	0

```
-- Ciração de um stored procedure para listar as pontuações por cada atleta
delimiter $$
create procedure get_pontuacoes_por_atleta()
BEGIN
    SELECT atleta.id_atleta, atleta.nome, Sum(atletas_campeonato.pontuacao_total) AS total_pontuacoes
        FROM atleta
        INNER JOIN atletas_campeonato
            ON atleta.id_atleta = atletas_campeonato.id_atleta
        GROUP BY id_atleta;
END $$
delimiter ;
```

Aqui temos um stored procedure que lista as pontuações obtidas por cada atleta individualmente trazendo a soma das pontuações em todos os campeonatos que ele participou no campo 'total_pontuacoes'. Produz este output:

```
call get_pontuacoes_por_atleta;
```

Result Grid | Filter Rows: | Export:

id_atleta	nome	total_pontuacoes
1	Gaye	524
2	Mervin	1650
3	Billy	200
5	Colet	0
6	Amargo	0
7	Curran	548
8	Deva	0
9	Cory	0
10	professor Marco	0
11	Ginni	0

```
-- Check no peso e idade para entrar nas conformidades
delimiter $$
CREATE TRIGGER idade_check BEFORE INSERT
    ON atleta
    FOR EACH ROW
        IF NEW.idade < 18 THEN
            SIGNAL SQLSTATE '50001' SET MESSAGE_TEXT = 'O atleta tem de ter no minímo 18 anos.';
        ELSEIF NEW.idade > 70 THEN
            SIGNAL SQLSTATE '50001' SET MESSAGE_TEXT = 'O atleta não pode ter mais que 70 anos.';
        END IF $$
CREATE TRIGGER peso_check BEFORE INSERT
    ON atleta
    FOR EACH ROW
        IF NEW.peso < 30 THEN
            SIGNAL SQLSTATE '50001' SET MESSAGE_TEXT = 'O atleta tem de ter pelo menos 30kg.';
        ELSEIF NEW.peso > 100 THEN
            SIGNAL SQLSTATE '50001' SET MESSAGE_TEXT = 'O atleta tem de ter no máximo 100kg.';
        END IF $$
delimiter ;
```

Aqui temos um trigger que sempre que for inserido um registo na tabela dos atletas, antes este vai verificar se o peso e a idade estão na conformidade. Caso não estejam apresenta uma mensagem de erro bem intuitiva do motivo da falha.

Por exemplo vou adicionar um registo de pura demonstração com a idade de 120:

```
614 • insert into atleta (id_atleta, nome, idade, peso, contacto, cod_postal) values (1, 'Gaye', 120, 90.08, 964158810, '8843-715');
```

Output

Action Output

#	Time	Action	Message
10312	00:56:21	SELECT * FROM atletas_campeonato LIMIT 0, 1000	30 row(s) returned
10313	00:56:21	SELECT * FROM lista_campeonatos_atleta LIMIT 0, 1000	30 row(s) returned
10314	00:56:21	call get_pontuacoes_por_atleta	23 row(s) returned
10315	00:56:31	SELECT * FROM lista_campeonatos_atleta LIMIT 0, 1000	30 row(s) returned
10316	01:00:14	call get_pontuacoes_por_atleta	23 row(s) returned
10317	01:07:44	insert into atleta (id_atleta, nome, idade, peso, contacto, cod_postal) values (1, 'Gaye', 120, 90.08, 964158810, '8843-715')	Error Code: 1644. O atleta não pode ter mais que 70 anos.

```
-- Se inserirem uma prova, a pontuacao será também alterada na tabela dos atletas_campeonatos.
delimiter $$
CREATE TRIGGER atualizar_pontuacoes_totais_i AFTER INSERT
    ON prova
    FOR EACH ROW
    begin
        DECLARE v_pontuacao_prova double;
        DECLARE v_pontuacao_total double;

        SELECT pontuacao_total into v_pontuacao_total from atletas_campeonato WHERE id_atleta = NEW.id_atleta AND id_campeonato = NEW.id_campeonato;
        set v_pontuacao_total = v_pontuacao_total + NEW.pontuacao_prova;

        UPDATE atletas_campeonato set pontuacao_total = v_pontuacao_total where id_atleta = NEW.id_atleta AND id_campeonato = NEW.id_campeonato;
    end $$
delimiter ;

-- Se deletarem uma das provas, a pontuacao será também alterada na tabela dos atletas_campeonatos.
delimiter $$
CREATE TRIGGER atualizar_pontuacoes_totais_d AFTER DELETE
    ON prova
    FOR EACH ROW
    begin
        DECLARE v_pontuacao_prova double;
        DECLARE v_pontuacao_total double;

        SELECT pontuacao_total into v_pontuacao_total from atletas_campeonato WHERE id_atleta = OLD.id_atleta AND id_campeonato = OLD.id_campeonato;
        set v_pontuacao_total = v_pontuacao_total - OLD.pontuacao_prova;

        UPDATE atletas_campeonato set pontuacao_total = v_pontuacao_total where id_atleta = OLD.id_atleta AND id_campeonato = OLD.id_campeonato;
    end $$
delimiter ;
```

Aqui temos dois triggers para quando é feito alguma alteração na tabela da prova, este vai dar update na tabela atletas_campeonato no campo da pontuação_total.

Então, no insert usamos o “new”, para explicar que é o valor que é novo, antes do insert e vamos pegar nesse valor e adicionar ao que já temos na tabela da pontuacao_total que está na tabela que queremos dar update.

No caso de delete usamos o “old”, porque era o valor antigo e temos que pegar na pontuação da prova que foi eliminada e subtrair à pontuação_total do campeonato para aquele atleta.

São triggers iguais só que recíprocos.

Por exemplo:

```
598     -- Código para explicação do trigger delete na atualização das pontuações
599 •   SELECT *
600         FROM atletas_campeonato;
601
602     -- DELETE from prova where id_prova = 1;
603 •   SELECT * FROM atletas_campeonato;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Con

id_atleta	id_campeonato	pontuacao_total
1	1	524
2	1	0
2	2	1650
3	5	200
3	13	0
5	1	0
6	6	0
7	5	0
7	7	548
8	6	0

A pontuação do atleta 1, no campeonato 1 está de 524, porém se eu deletar um dos registos da prova que corresponda ao atleta 1 e campeonato 1 como por exemplo a prova com o id 2, este ficará 524 – 250 = 274 na pontuação total.

```
601 •   SELECT *
602         FROM prova;
603
```

Result Grid | Filter Rows: | Edit: | Export/Import:

id_prova	id_campeonato	id_atleta	cod_postal	tempo_prova	pontuacao_prova
2	1	1	7783-151	00:17:01	250
3	2	2	1018-981	01:55:46	1000
4	7	7	0265-762	00:49:51	30
5	6	19	1941-591	00:43:54	200
6	5	3	5729-175	01:34:10	200
7	4	27	8728-500	00:46:28	150
8	7	7	5329-912	01:46:20	128
9	7	7	5120-259	01:01:09	90
10	7	7	9249-452	01:06:36	150
11	7	7	4453-841	01:21:30	150

prova 83

Output final:

```
605 •   DELETE from prova where id_prova = 2;
606 •   SELECT * FROM atletas_campeonato;
607
```

Result Grid | Filter Rows: | Edit:

id_atleta	id_campeonato	pontuacao_total
1	1	274
2	1	0
2	2	1650
3	5	200
3	13	0
5	1	0
6	6	0
7	5	0
7	7	548
8	6	0

```
-- Se o nome vier com a primeira letra minúscula este trigger antes de o inserir na tabela ele coloca a primeira letra capitalizada.
delimiter $$
CREATE TRIGGER nome_capitalizado BEFORE INSERT
    ON atleta
    FOR EACH ROW
    begin
        set new.nome = CONCAT(UCASE(LEFT(new.nome, 1)), LCASE(SUBSTRING(new.nome, 2)));
    end $$
delimiter ;
```

Aqui tenho outro trigger para caso na inserção do atleta, escrevem o nome com a primeira letra minúscula, o trigger vai corrigir e colocar com maiúscula e as outras posteriormente tudo minúscula.

```
594 •   insert into atleta values (32, 'rui', 25, 66, 962316798, '2359-523');
595 •   SELECT *
596         FROM atleta
597         WHERE id_atleta = 32;
598
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap C

id_atleta	nome	idade	peso	contacto	cod_postal
32	Rui	25	66.00	962316798	2359-523

Neste update em que dou novos dados para o atleta com o id de 10:

```
583   -- ----------------------------------------------------------------------------------------
584   UPDATE atleta
585   SET nome = 'Professor Marco', idade= 35, peso = 85.60, contacto = 914391391, cod_postal = '2945-430'
586   WHERE id_atleta = 10;
587
588   SELECT *
589       FROM atleta
590       WHERE id_atleta = 10;
591
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id_atleta	nome	idade	peso	contacto	cod_postal
10	Professor Marco	35	85.60	914391391	2945-430

```
593   DELETE FROM atletas_campeonato WHERE id_atleta = 10;
594   DELETE FROM atleta WHERE id_atleta = 10;
595   SELECT * FROM atleta;
596
```

Result Grid | Filter Rows: | Edit: | Export/Import:

id_atleta	nome	idade	peso	contacto	cod_postal
1	Gaye	19	90.08	964158810	8843-715
2	Mervin	61	50.47	965024828	6494-304
3	Billy	18	56.80	962227514	2871-249
4	Oralia	54	47.79	96397020 963970204	9
5	Colet	18	43.80	963044573	5350-397
6	Amargo	49	87.50	962332460	3149-606
7	Curran	59	58.52	962316798	3995-396
8	Deva	32	39.14	967313804	0980-495
9	Cory	67	60.17	968849599	9820-678
11	Ginni	46	63.78	960441576	2940-580

Podemos verificar que o id do atleta foi removido com sucesso. Para remover o atleta tive primeiro que remover aonde o atleta estava registado como chave estrangeira e só depois remover a primária. Em vez disso na declaração também podia ter declarado “ON DELETE CASCADE” que prontos se tiver confiança no trabalho que fiz uso isso e aonde o atleta estiver com o id = 10 em alguma doutras tabelas elimina, varrendo tudo.

```
599 •  SELECT  b.id_atleta,a.id_campeonato, a.pontuacao_total
600        FROM atleta b
601        LEFT JOIN atletas_campeonato a
602        ON b.id_atleta = a.id_atleta
603        ORDER BY id_atleta;
604
605
606
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id_atleta	id_campeonato	pontuacao_total
11	1	0
12	1	0
13	NULL	NULL
14	1	0
14	20	0
15	NULL	NULL
16	NULL	NULL
17	1	0
18	NULL	NULL
19	6	200

Select para apresentar todos os atletas independentemente se já participam em algum campeonato ou não. Para isso uso o left join.

```
605 •  SELECT  b.id_atleta,a.id_campeonato, a.pontuacao_total
606        FROM atleta b
607        INNER JOIN atletas_campeonato a
608        ON b.id_atleta = a.id_atleta
609        ORDER BY id_atleta;
610
611
612
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id_atleta	id_campeonato	pontuacao_total
1	1	1024
2	1	0
2	2	1650
3	5	200
3	13	0
5	1	0
6	6	0
7	5	0
7	7	548
8	6	0

Com o inner join já consigo trazer só os valores que se intercetam, por assim dizer, ou seja já não me aparecem atletas que nunca participaram em nenhum campeonato.

```
SELECT  b.id_atleta,a.id_campeonato, a.pontuacao_total
    FROM atleta b
    left join atletas_campeonato a
    ON b.id_atleta = a.id_atleta
    WHERE a.id_campeonato IS NULL
    ORDER BY id_atleta;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

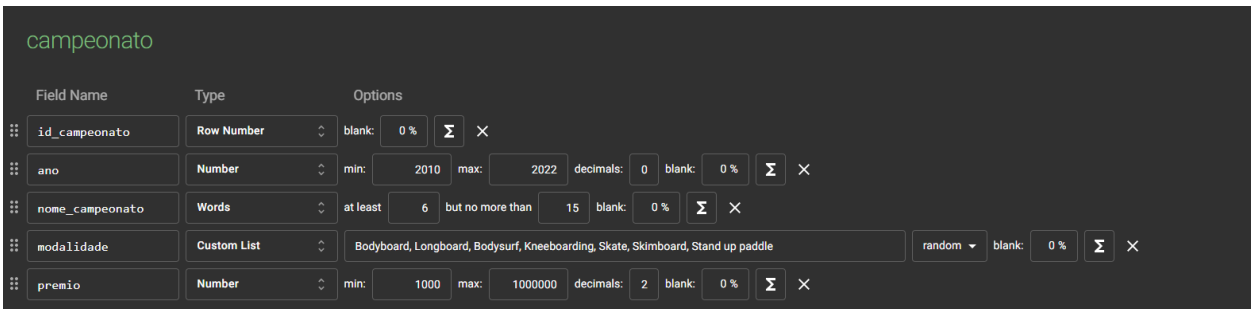
id_atleta	id_campeonato	pontuacao_total
4	NULL	NULL
13	NULL	NULL
15	NULL	NULL
16	NULL	NULL
18	NULL	NULL
21	NULL	NULL
28	NULL	NULL

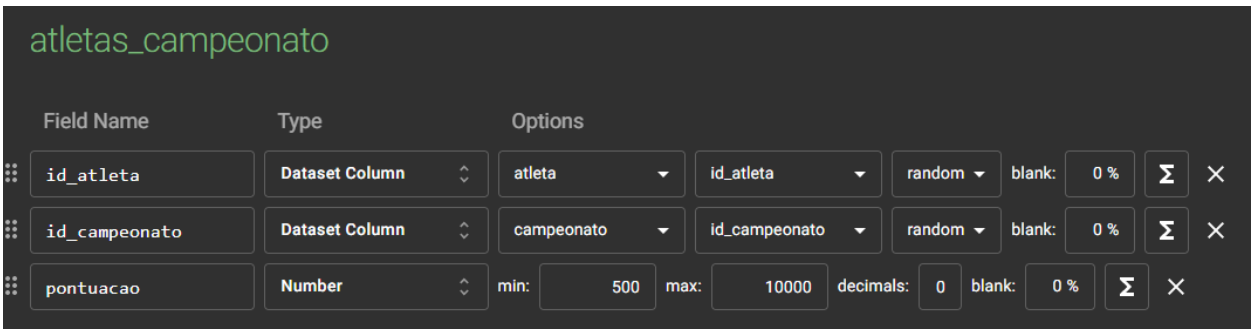
Usando a 'clause where is null', consigo obter os atletas que nunca participaram em nenhum torneio.

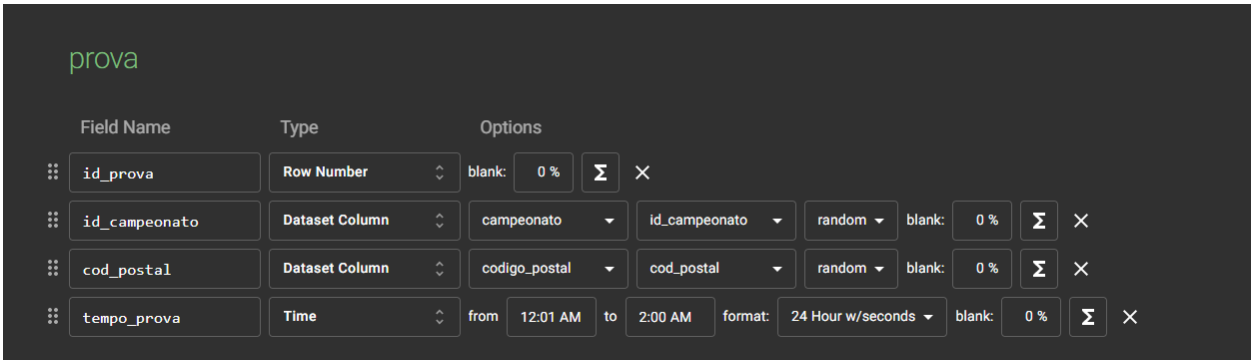
Para a inserção de dados usamos o "**mockaroo"**, onde criamos 5 esquemas, para as nossas 5 tabelas, começando por registar primeiro a tabela dos códigos postais, para poder chamar como chave estrangeira noutras tabelas através de datasets.

Inserção foi feita por esta ordem:

1. codigo_postal
2. atleta (FK do código postal)
3. campeonato
4. atletas_campeonato (FK de atleta e de campeonato)
5. prova (FK do código postal, campeonato e de atleta)







Exemplos de inserts na base de dados:

```
INSERT INTO codigo_postal(cod_postal,cidade,pais) VALUES ('9252-635','Bayan Hot','China');
INSERT INTO codigo_postal(cod_postal,cidade,pais) VALUES ('7834-451','Fryčovice','Czech Republic');


insert into atleta (id_atleta, nome, idade, peso, contacto, cod_postal) values (1, 'Gaye', 19, 90.08, 964158810, '8843-715');
insert into atleta (id_atleta, nome, idade, peso, contacto, cod_postal) values (2, 'Mervin', 61, 50.47, 965024828, '6494-304');
insert into atleta (id_atleta, nome, idade, peso, contacto, cod_postal) values (3, 'Billy', 18, 56.8, 962227514, '2871-249');
insert into atleta (id_atleta, nome, idade, peso, contacto, cod_postal) values (4, 'Oralia', 54, 47.79, 963970204, '4726-489');
insert into atleta (id_atleta, nome, idade, peso, contacto, cod_postal) values (5, 'Colet', 18, 43.8, 963044573, '5350-397');
insert into atleta (id_atleta, nome, idade, peso, contacto, cod_postal) values (6, 'Amargo', 49, 87.5, 962332460, '3149-606');
insert into atleta (id_atleta, nome, idade, peso, contacto, cod_postal) values (7, 'Curran', 59, 58.52, 962316798, '3995-396');
insert into atleta (id_atleta, nome, idade, peso, contacto, cod_postal) values (8, 'Deva', 32, 39.14, 967313804, '0980-495');
insert into atleta (id_atleta, nome, idade, peso, contacto, cod_postal) values (9, 'Cory', 67, 60.17, 968849599, '9820-678');
```

CONCLUSÃO

De maneira geral, pusemos em prática os conhecimentos adquiridos em contexto de aula, aprofundando os mesmos, assim como adquirindo novos conhecimentos através de pesquisas e esclarecimento de dúvidas, atingimos todos os objetivos propostos para este projeto e ultrapassamos as dificuldades que surgiram.