**PROJETO E MODELAGEM DE BANCO DE DADOS**

**ALUNO: DIEGO PRESTES DE SOUSA – RESPOSTAS NO FINAL DO DOCUMENTO.**

Existem diversas ferramentas de modelagem de dados. Experimente neste exercício utilizar algumas delas: SQL Power Architect, DBDesigner 4, Astah Professional, MySQL Workbench.

1. Considere as seguintes alternativas de implementação de um banco de dados relacional:

Alternativa 1:

Aluno (CodAl, Nome, CodCurso, Endereco)

Alternativa 2

Aluno (CodAl, Nome, CodCurso)

EnderecoAluno (CodAl, Endereco)

CodAl referencia Aluno

Em ambos os casos está sendo representado um conjunto de alunos e informações (código, nome, código de curso, endereço) a ele referentes. Considerando os princípios que baseiam as regras de um modelo relacional, aponte os pontos positivos e negativos de cada uma delas. (dois argumentos no máximo)

1. A partir do script SQL abaixo elabore o modelo físico e comente o código.

CREATE TABLE instrutores (

id INT NOT NULL AUTO\_INCREMENT,

nome VARCHAR(50) NOT NULL,

email VARCHAR(50) NOT NULL,

valor\_hora INTEGER UNSIGNED NULL,

certificados VARCHAR(255) NULL,

PRIMARY KEY(id)

);

CREATE TABLE cursos (

id INTEGER UNSIGNED NOT NULL AUTO\_INCREMENT,

nome VARCHAR(50) NOT NULL,

requisito VARCHAR(255) NULL,

carga\_horaria SMALLINT UNSIGNED NULL,

preco DOUBLE UNSIGNED NULL,

PRIMARY KEY(id)

);

CREATE TABLE alunos (

id INT NOT NULL AUTO\_INCREMENT,

cpf CHAR(11) NOT NULL,

nome VARCHAR(50) NOT NULL,

email VARCHAR(50) NOT NULL,

fone CHAR(14) NOT NULL,

data\_nascimento DATE NULL,

PRIMARY KEY(id)

);

CREATE TABLE turmas (

id INTEGER UNSIGNED NOT NULL AUTO\_INCREMENT, -- UNSIGNED está depreciado e será removido em versões futuras (aviso do MySQL Workbench)

instrutores\_id INT NOT NULL,

cursos\_id INTEGER UNSIGNED NOT NULL,

data\_inicio DATE NULL,

data\_final DATE NULL,

carga\_horaria SMALLINT UNSIGNED NULL,

PRIMARY KEY(id),

INDEX turmas\_FKIndex1(cursos\_id),

INDEX turmas\_FKIndex2(instrutores\_id),

FOREIGN KEY(cursos\_id)

REFERENCES cursos(id)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

FOREIGN KEY(instrutores\_id)

REFERENCES instrutores(id)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

CREATE TABLE matriculas (

id INTEGER UNSIGNED NOT NULL AUTO\_INCREMENT,

turmas\_id INTEGER UNSIGNED NOT NULL,

alunos\_id INT NOT NULL,

data\_matricula DATE NULL,

PRIMARY KEY(id),

INDEX matriculas\_FKIndex1(alunos\_id),

INDEX matriculas\_FKIndex3(turmas\_id),

FOREIGN KEY(alunos\_id)

REFERENCES alunos(id)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

FOREIGN KEY(turmas\_id)

REFERENCES turmas(id)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

* 1. Execute o script do banco de dados
  2. Execute o comando DROP table.
  3. Execute o comando TRUNCATE table. Pesquise a diferença entre DROP e TRUNCATE.

1. Implemente a base da dados para o modelo:

Navio(nome, capacidade)

NavioEmbarque(Navio, carga)

Carga(numero, peso)

CargaRecepcaoAgente(Carga, Agente)

CargaPerecivel(Carga, datavalidade)

CargaSensivel(Carga, temperatura\_maxima)

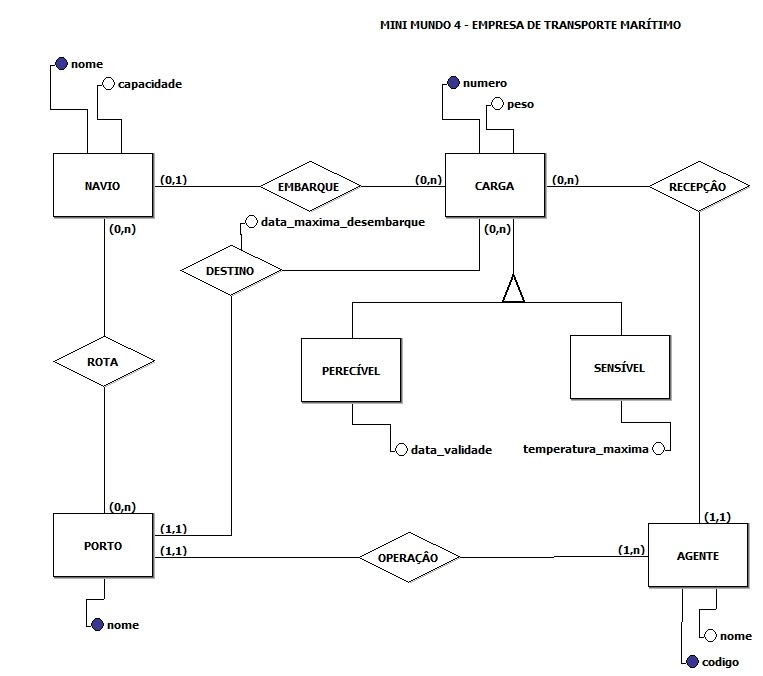
Porto(nome)

PortoFazRota(Porto, Navio)

PortoDestinoCarga(Porto, Carga, data\_maxima\_desembarque)

PortoOperacaoAgente(Porto, Agente)

Agente(nome, codigo)



* 1. Elabore o modelo físico
  2. Implemente o script SQL para a criação do banco de dados
  3. Insira dados fictícios em cada uma das tabelas
  4. Construa as queries e escreva a álgebra relacional associada
     1. Selecione as cargas dos navios por porto
     2. Selecione as cargas agrupadas como perecível e sensível
     3. Selecione as cargas perecíveis com data de validade expirada
     4. Selecione as cargas recepcionadas por agente com data de desembarque inferior a data atual

1. Considerando o case abaixo:

*“Uma empresa deseja contratar você para desenvolver um sistema de controle as competições estaduais dos times de futebol.*

*Cada time é composto por um número de jogadores, dos quais não são todos que participam de um determinado jogo. Os times são acompanhados pelo treinador, por um médico e por um nutricionista. Cada jogador possui uma dieta de acordo com as suas necessidades de desempenho em campo. Todos os jogadores também possuem uma rígida rotina de exercícios.*

*O médico é responsável por solicitar os exames e acompanhar a saúde dos jogadores. Caso um jogador não tenha condições físicas, o médico pode impedir sua escala no jogo.*

*O sistema deve controlar a rotina de cada jogador e sua escala nos jogos de cada time, as posições em que jogam e o resultado do jogo.”*

Elabore o projeto **o Modelo Conceitual e o Modelo Fisico**.

Com base no **Modelo Relacional** elaborado, descreva as instruções utilizando os comandos de SELECT, INSERT, UPDATE e DELETE, conforme o caso, para as seguintes situações.

1. Inclua informações sobre em todas as entidades do modelo.

* Demonstre os comandos de inserção.
* Apresente a tabela com os dados dados populados.

1. Um jogador foi vendido e agora ele pertence ao time adversário.

* Demonstre o comando para atualização das informações do jogador e do time.

1. O técnico da equipe declarou que após 10 partidas sem ganhar, está se aposentando.
   * Demonstre o comando para exclusão das informações do técnico.
2. Os jornalistas estão solicitando informações para publicar na mídia. Precisam saber o ranking de times e sua classificação considerando o resultado dos jogos.
   * Demonstre o(s) comando(s).
   * Apresente a tabela com os dados resultantes.
3. Os técnicos gostariam de saber quais jogadores estão aptos a realizar partidas conforme a orientação dos médicos.
   * Demonstre o(s) comando(s).
   * Apresente a tabela com os dados resultantes.

TODAS AS RESPOSTAS SE ENCONTRAM A PARTIR DAQUI:

1:

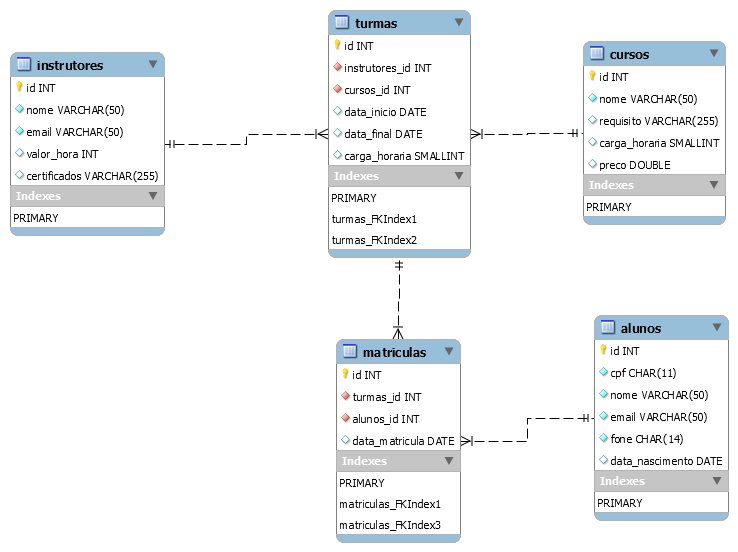
Pontos positivos Alternativa 1: Está simples de compreender e fácil de fazer buscas (SELECT); outras operações também são facilitadas com os dados numa só tabela como INSERT, UPDATE e DELETE. Tudo feito numa só tabela. (Aqui estamos ignorando boas práticas, mas conhecendo a sintaxe do MySQL, por exemplo).

Pontos negativos Alternativa 1: O endereço deve ser separado em uma tabela com mais detalhes e armazenado somente a chave estrangeira dele na tabela aluno; CodCurso deveria estar como chave estrangeira, assim poderia facilmente puxar todos os dados do curso.

Pontos positivos Alternativa 2: Está separando o endereço em outra tabela; novamente, utilizar CodAl em ambas tabelas da alternativa facilita a busca e outras operações no banco de dados.

Pontos negativos Alternativa 2: A tabela endereço não está dividida em rua, bairro, cidade, CEP... Basicamente não auxiliou em nada tirar esse único dado da tabela aluno, pois ele deve ser melhor dividido; EnderecoAluno deveria ter uma chave primaria como "CodEnd" e puxar essa chave como chave estrangeira na tabela Aluno, poderia facilmente buscar os dados utilizando WHERE Aluno.CodEnd = EnderecoAluno.CodEnd.

2:



2a: Script executado com sucesso:

id INTEGER UNSIGNED NOT NULL AUTO\_INCREMENT, -- UNSIGNED está depreciado e será removido em versões futuras (aviso do MySQL Workbench)

-- COMENTARIOS GERAIS

-- Eu consideraria colocar algo como "nometabela\_id" para não confundir os ids;

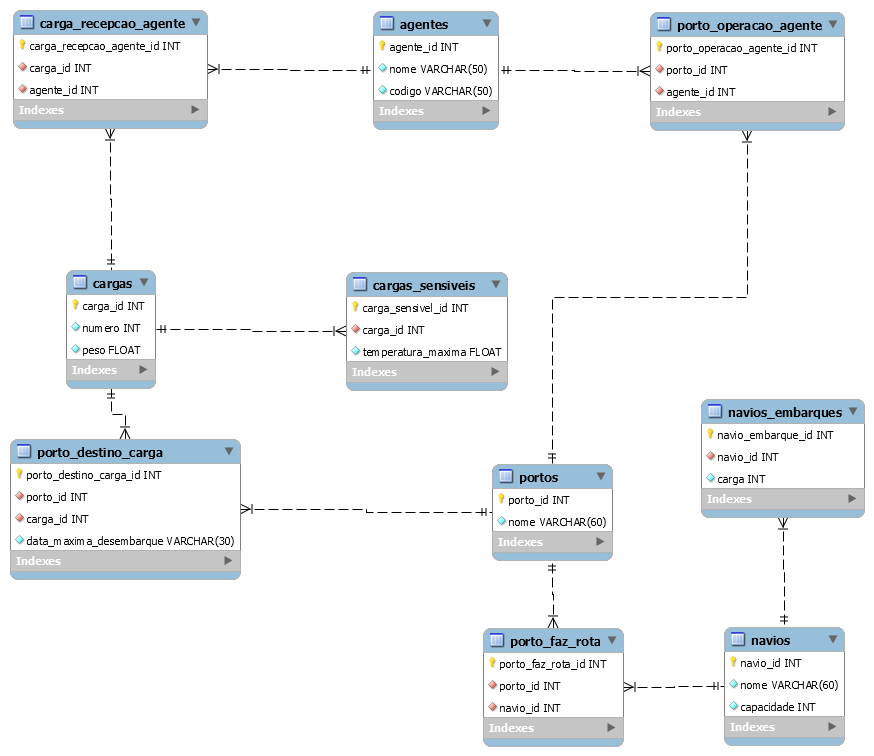
-- DELETE E UPDATE eu colocaria ação de CASCADE e não "NO ACTION", assim os dados de outras tabelas se auto atualizariam e alguns dados importantes poderiam ser devidamente tratados (exemplo: cliente deleta sua conta, tudo será deletado, porém o prestador de serviço daquele cliente ainda teria registrado que ele atendeu determinado cliente, e os dados do serviço DAQUELE determinado cliente que deletou sua conta);

-- Não conheço INDEX, mas pelo que pesquisei está tudo correto. Não apareceu nenhum erro para criar as tabelas (tirando o aviso de UNSIGNED que comentei dentro do código) ou para popular elas com alguns dados fictícios.

2b: DROP TABLE matriculas, outras tabelas deram erro, pois as chaves estrangeiras estão “ligadas” com a tabela matriculas. Para executar o DROP TABLE em todas as tabelas é preciso fazer na ordem certa, levando em consideração as chaves estrangeiras. MySQL não deixa você estragar a relação.

2c: A diferença de DROP e TRUCATE é que enquanto TRUNCATE TABLE deleta todas as linhas de uma tabela o DROP TABLE deleta de fato a tabela, removendo a definição dela e seu conteúdo; DROP TABLE libera espaço em memória e o TRUCATE TABLE não; em DROP TABLE as restrições de integridade (não sei exatamente ao que se refere) será removida e o TRUCATE TABLE não as remove; TRUCATE TABLE é mais rápido que DROP TABLE, e que DROP TABLE também pode ter mais complicação no momento de executar. Lembrando que nenhum dos comandos usa WHERE, pois é um comando para a tabela inteira.

3a:



3b:

CREATE DATABASE fundamentosDB\_trab2\_3;

USE fundamentosDB\_trab2\_3;

CREATE TABLE navios (

navio\_id INT NOT NULL AUTO\_INCREMENT,

nome VARCHAR(60) NOT NULL,

capacidade INT NOT NULL,

PRIMARY KEY(navio\_id)

);

CREATE TABLE navios\_embarques (

navio\_embarque\_id INT NOT NULL AUTO\_INCREMENT,

navio\_id INT NOT NULL,

carga\_id INT NOT NULL,

PRIMARY KEY(navio\_embarque\_id),

FOREIGN KEY(navio\_id)

REFERENCES Navios(navio\_id)

);

CREATE TABLE cargas (

carga\_id INT NOT NULL AUTO\_INCREMENT,

numero INT NOT NULL, -- se isso aqui for um identificador da carga talvez devesse ser varchar, mas vou manter int

peso FLOAT NOT NULL,

PRIMARY KEY(carga\_id)

);

CREATE TABLE agentes (

agente\_id INT NOT NULL AUTO\_INCREMENT,

nome VARCHAR(50) NOT NULL,

codigo VARCHAR(50) NOT NULL,

PRIMARY KEY(agente\_id)

);

CREATE TABLE carga\_recepcao\_agente (

carga\_recepcao\_agente\_id INT NOT NULL AUTO\_INCREMENT,

carga\_id INT NOT NULL,

agente\_id INT NOT NULL,

PRIMARY KEY(carga\_recepcao\_agente\_id),

FOREIGN KEY(carga\_id)

REFERENCES cargas(carga\_id),

FOREIGN KEY(agente\_id)

REFERENCES agentes(agente\_id)

);

CREATE TABLE cargas\_pereciveis (

carga\_perecivel\_id INT NOT NULL AUTO\_INCREMENT,

carga\_id INT NOT NULL,

data\_validade VARCHAR(20) NOT NULL,

PRIMARY KEY(carga\_perecivel\_id),

FOREIGN KEY(carga\_id)

REFERENCES cargas(carga\_id)

);

CREATE TABLE cargas\_sensiveis (

carga\_sensivel\_id INT NOT NULL AUTO\_INCREMENT,

carga\_id INT NOT NULL,

temperatura\_maxima FLOAT NOT NULL,

PRIMARY KEY(carga\_sensivel\_id),

FOREIGN KEY(carga\_id)

REFERENCES cargas(carga\_id)

);

CREATE TABLE portos (

porto\_id INT NOT NULL AUTO\_INCREMENT,

nome VARCHAR(60) NOT NULL,

PRIMARY KEY(porto\_id)

);

CREATE TABLE porto\_faz\_rota (

porto\_faz\_rota\_id INT NOT NULL AUTO\_INCREMENT,

porto\_id INT NOT NULL,

navio\_id INT NOT NULL,

PRIMARY KEY(porto\_faz\_rota\_id),

FOREIGN KEY(porto\_id)

REFERENCES portos(porto\_id),

FOREIGN KEY(navio\_id)

REFERENCES navios(navio\_id)

);

CREATE TABLE porto\_destino\_carga (

porto\_destino\_carga\_id INT NOT NULL AUTO\_INCREMENT,

porto\_id INT NOT NULL,

carga\_id INT NOT NULL,

data\_maxima\_desembarque VARCHAR(30) NOT NULL,

PRIMARY KEY(porto\_destino\_carga\_id),

FOREIGN KEY(porto\_id)

REFERENCES portos(porto\_id),

FOREIGN KEY(carga\_id)

REFERENCES cargas(carga\_id)

);

CREATE TABLE porto\_operacao\_agente (

porto\_operacao\_agente\_id INT NOT NULL AUTO\_INCREMENT,

porto\_id INT NOT NULL,

agente\_id INT NOT NULL,

PRIMARY KEY(porto\_operacao\_agente\_id),

FOREIGN KEY(porto\_id)

REFERENCES portos(porto\_id),

FOREIGN KEY(agente\_id)

REFERENCES agentes(agente\_id)

);

3c:

INSERT INTO navios (nome, capacidade)

VALUES ("x", 300);

INSERT INTO navios (navio\_id, carga)

VALUES (12, 300);

INSERT INTO cargas(numero, peso)

VALUES(12, 666);

INSERT INTO agentes (nome, codigo)

VALUES ("name of the agent", "666");

INSERT INTO carga\_recepcao\_agente(carga\_id, agente\_id)

VALUES(12, 666);

INSERT INTO cargas\_pereciveis(carga\_id, data\_validade)

VALUES(12, "1999-04-17");

INSERT INTO cargas\_sensiveis(carga\_id, temperatura\_maxima)

VALUES(12, 66.44);

INSERT INTO porto\_faz\_rota(porto\_id, navio\_id)

VALUES(12, 21);

INSERT INTO porto\_destino\_carga(porto\_id, carga\_id, data\_maxima\_desembarque)

VALUES(12, 21, ”2020-01-01”);

INSERT INTO porto\_operacao\_agente(porto\_id, agente\_id)

VALUES(12, 21);

3d:

USE fundamentosDB\_trab2\_3;

-- SELECT 1

SELECT porto\_destino\_carga.porto\_id AS porto\_id, porto\_destino\_carga.carga\_id AS porto\_carga\_id, navios\_embarques.navio\_id AS embarque\_navio\_id, navios\_embarques.carga\_id AS embarque\_carga\_id

FROM porto\_destino\_carga, navios\_embarques

WHERE porto\_carga\_id = embarque\_carga\_id

GROUP BY porto\_id;

-- SELECT 2

SELECT cargas\_pereciveis.carga\_id AS carga\_perecivel, cargas\_sensiveis.carga\_id AS carga\_sensivel

FROM cargas\_pereciveis, cargas\_sensiveis

GROUP BY carga\_perecivel AND carga\_sensivel;

-- SELECT 3

SELECT cargas\_pereciveis.carga\_id AS carga\_perecivel

FROM cargas\_pereciveis

WHERE DATE(cargas\_pereciveis.data\_validade) < DATE("2020-11-17");

-- SELECT 4

SELECT porto\_destino\_carga.carga\_id, porto\_destino\_carga.data\_maxima\_desembarque

FROM porto\_destino\_carga

INNER JOIN carga\_recepcao\_agente

ON carga\_recepcao\_agente.carga\_id = porto\_destino\_carga.carga\_id

WHERE DATE(data\_maxima\_desembarque) < DATE("1999-04-17");

**ALGEBRA RELACIONAL DAS QUERIES ACIMA:**

**SELECT 1:**

π porto\_id, porto\_carga\_id, embarque\_navio\_id, embarque\_carga\_id

σ (porto\_carga\_id = embarque\_carga\_id)

(porto\_destino\_carga, navios\_embarques)

**SELECT 2:**

π carga\_perecivel, carga\_sensivel (cargas\_pereciveis, cargas\_sensiveis)

**SELECT 3:**

*OBS: não encontrei um método correto que expressasse datas em algebra relacional, então resolvi fazer dessa forma.*

π carga\_perecivel

σ (DATE(cargas\_pereciveis.data\_validade) < DATE(2020-11-17))

(cargas\_pereciveis)

**SELECT 4:**

π porto\_destino\_carga.carga\_id, porto\_destino\_carga.data\_maxima\_desembarque

σ (DATE(data\_maxima\_desembarque) < DATE("1999-04-17"))

(porto\_destino\_carga |X| carga\_recepcao\_agente)

carga\_recepcao\_agente.carga\_id = porto\_destino\_carga.carga\_id

4: SCRIPT DE CRIAÇÃO DO BANCO E MODELOS CONCEITUAL E FÍSICO.

CREATE DATABASE fundBD\_ativ4;

USE fundBD\_ativ4;

CREATE TABLE enderecos (

endereco\_id INT NOT NULL AUTO\_INCREMENT,

estado VARCHAR(50) NOT NULL,

cidade VARCHAR(50) NOT NULL,

bairro VARCHAR(50) NOT NULL,

rua VARCHAR(50) NOT NULL,

cep VARCHAR(50) NOT NULL,

PRIMARY KEY(endereco\_id)

);

CREATE TABLE times (

time\_id INT NOT NULL AUTO\_INCREMENT,

endereco\_id INT NOT NULL,

nome VARCHAR(150) NOT NULL,

data\_inicio VARCHAR(30) NOT NULL,

ON DELETE CASCADE,

ON UPDATE CASCADE,

PRIMARY KEY(time\_id),

FOREIGN KEY(endereco\_id)

REFERENCES enderecos(endereco\_id)

);

CREATE TABLE jogadores (

jogador\_id INT NOT NULL AUTO\_INCREMENT,

time\_id INT NOT NULL,

nome VARCHAR(50) NOT NULL,

data\_nascimento VARCHAR(30),

num\_camisa INT(30),

posicao VARCHAR(30),

ON DELETE CASCADE,

ON UPDATE CASCADE,

PRIMARY KEY(jogador\_id),

FOREIGN KEY(time\_id)

REFERENCES times(time\_id)

);

CREATE TABLE treinadores (

treinador\_id INT NOT NULL AUTO\_INCREMENT,

time\_id INT NOT NULL,

nome VARCHAR(100) NOT NULL,

data\_inicio\_time VARCHAR(30) NOT NULL,

ON DELETE CASCADE,

ON UPDATE CASCADE,

PRIMARY KEY(treinador\_id),

FOREIGN KEY(time\_id)

REFERENCES times(time\_id)

);

CREATE TABLE treinos (

treino\_id INT NOT NULL AUTO\_INCREMENT,

treinador\_id INT NOT NULL,

objetivo VARCHAR(100) NOT NULL,

data\_treino VARCHAR(30) NOT NULL,

descricao VARCHAR(500),

ON DELETE CASCADE,

ON UPDATE CASCADE,

PRIMARY KEY(treino\_id),

FOREIGN KEY(treinador\_id)

REFERENCES treinadores(treinador\_id)

);

CREATE TABLE jogador\_treino (

jogador\_treino\_id INT NOT NULL AUTO\_INCREMENT,

jogador\_id INT NOT NULL,

treino\_id INT NOT NULL,

ON DELETE CASCADE,

ON UPDATE CASCADE,

PRIMARY KEY(jogador\_treino\_id),

FOREIGN KEY(jogador\_id)

REFERENCES jogadores(jogador\_id),

FOREIGN KEY(treino\_id)

REFERENCES treinos(treino\_id)

);

CREATE TABLE nutricionistas (

nutricionista\_id INT NOT NULL AUTO\_INCREMENT,

time\_id INT NOT NULL,

nome VARCHAR(100) NOT NULL,

data\_inicio\_time VARCHAR(30) NOT NULL,

ON DELETE CASCADE,

ON UPDATE CASCADE,

PRIMARY KEY(nutricionista\_id),

FOREIGN KEY(time\_id)

REFERENCES times(time\_id)

);

CREATE TABLE refeicoes (

refeicao\_id INT NOT NULL AUTO\_INCREMENT,

vitaminas\_principais VARCHAR(200),

proteina FLOAT,

carboidrato FLOAT,

gordura FLOAT,

PRIMARY KEY(refeicao\_id)

);

CREATE TABLE dietas (

dieta\_id INT NOT NULL AUTO\_INCREMENT,

nutricionista\_id INT NOT NULL,

jogador\_id INT NOT NULL,

valor\_energetico\_diario FLOAT NOT NULL,

num\_refeicoes\_dia INT NOT NULL,

objetivo VARCHAR(50) NOT NULL,

ON DELETE CASCADE,

ON UPDATE CASCADE,

PRIMARY KEY(dieta\_id),

FOREIGN KEY(nutricionista\_id)

REFERENCES nutricionistas(nutricionista\_id),

FOREIGN KEY(jogador\_id)

REFERENCES jogadores(jogador\_id)

);

CREATE TABLE refeicao\_na\_dieta (

refeicao\_na\_dieta\_id INT NOT NULL AUTO\_INCREMENT,

refeicao\_id INT NOT NULL,

dieta\_id INT NOT NULL,

ON DELETE CASCADE,

ON UPDATE CASCADE,

PRIMARY KEY(refeicao\_na\_dieta\_id),

FOREIGN KEY(refeicao\_id)

REFERENCES refeicoes(refeicao\_id),

FOREIGN KEY(dieta\_id)

REFERENCES dietas(dieta\_id)

);

CREATE TABLE medicos (

medico\_id INT NOT NULL AUTO\_INCREMENT,

time\_id INT NOT NULL,

nome VARCHAR(100) NOT NULL,

data\_inicio\_time VARCHAR(30) NOT NULL,

ON DELETE CASCADE,

ON UPDATE CASCADE,

PRIMARY KEY(medico\_id),

FOREIGN KEY(time\_id)

REFERENCES times(time\_id)

);

CREATE TABLE avaliacao\_fisica (

avaliacao\_fisica\_id INT NOT NULL AUTO\_INCREMENT,

jogador\_id INT NOT NULL,

medico\_id INT NOT NULL,

data\_avaliacao VARCHAR(30) NOT NULL,

apto\_para\_jogar BOOLEAN NOT NULL,

observacoes VARCHAR(500),

ON DELETE CASCADE,

ON UPDATE CASCADE,

PRIMARY KEY(avaliacao\_fisica\_id),

FOREIGN KEY(jogador\_id)

REFERENCES jogadores(jogador\_id),

FOREIGN KEY(medico\_id)

REFERENCES medicos(medico\_id)

);

CREATE TABLE jogos (

jogo\_id INT NOT NULL AUTO\_INCREMENT,

time\_da\_casa INT NOT NULL,

time\_adversario INT NOT NULL,

pontuacao\_time\_da\_casa INT NOT NULL,

pontuacao\_time\_adversario INT NOT NULL,

PRIMARY KEY(jogo\_id)

);

CREATE TABLE time\_jogo\_casa (

time\_jogo\_casa\_id INT NOT NULL AUTO\_INCREMENT,

time\_id INT NOT NULL,

jogo\_id INT NOT NULL,

ON DELETE CASCADE,

ON UPDATE CASCADE,

PRIMARY KEY(time\_jogo\_casa\_id),

FOREIGN KEY(time\_id)

REFERENCES times(time\_id),

FOREIGN KEY(jogo\_id)

REFERENCES jogos(jogo\_id)

);

CREATE TABLE time\_jogo\_adversario (

time\_jogo\_adversario\_id INT NOT NULL AUTO\_INCREMENT,

time\_id INT NOT NULL,

jogo\_id INT NOT NULL,

ON DELETE CASCADE,

ON UPDATE CASCADE,

PRIMARY KEY(time\_jogo\_adversario\_id),

FOREIGN KEY(time\_id)

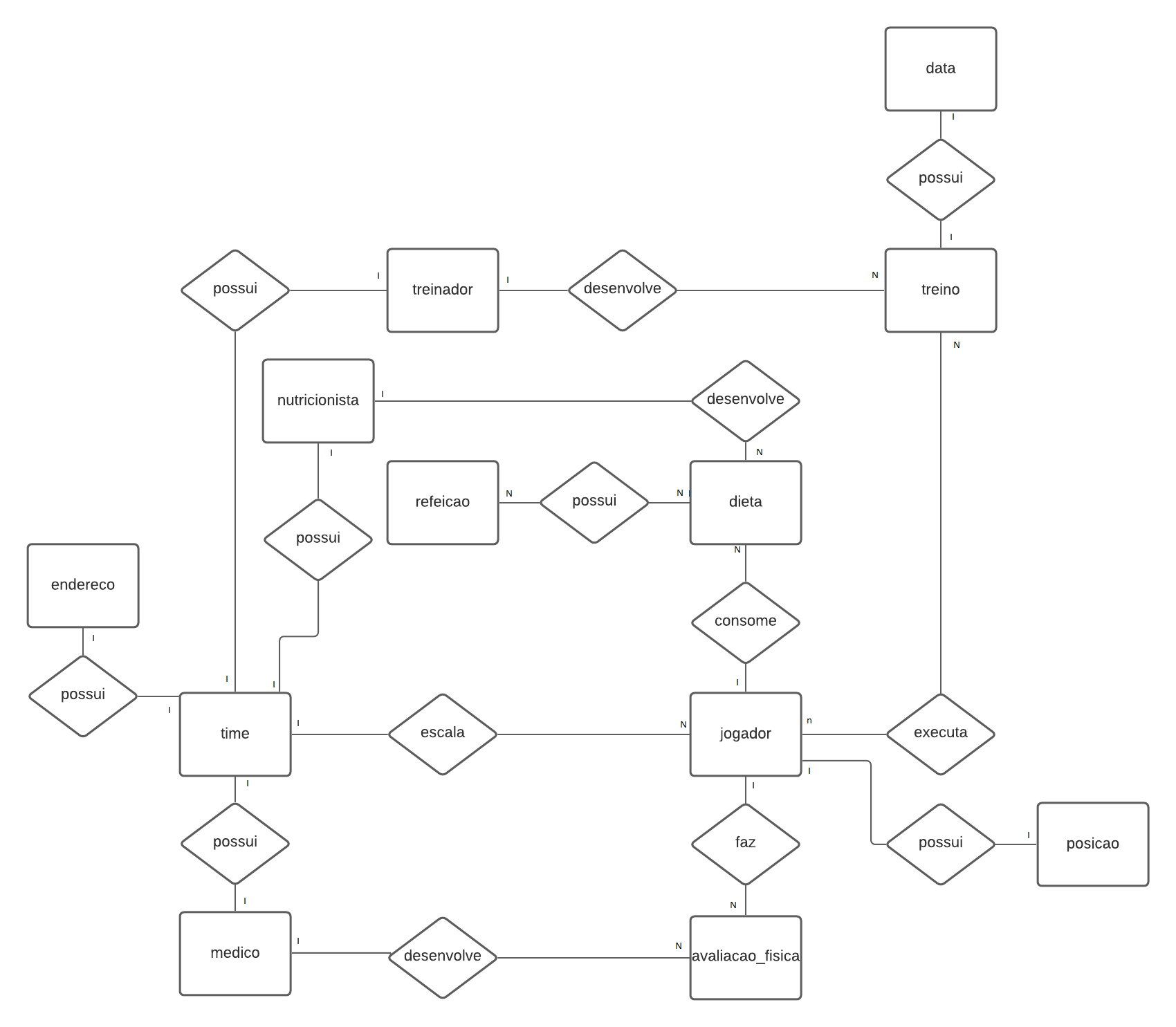
REFERENCES times(time\_id),

FOREIGN KEY(jogo\_id)

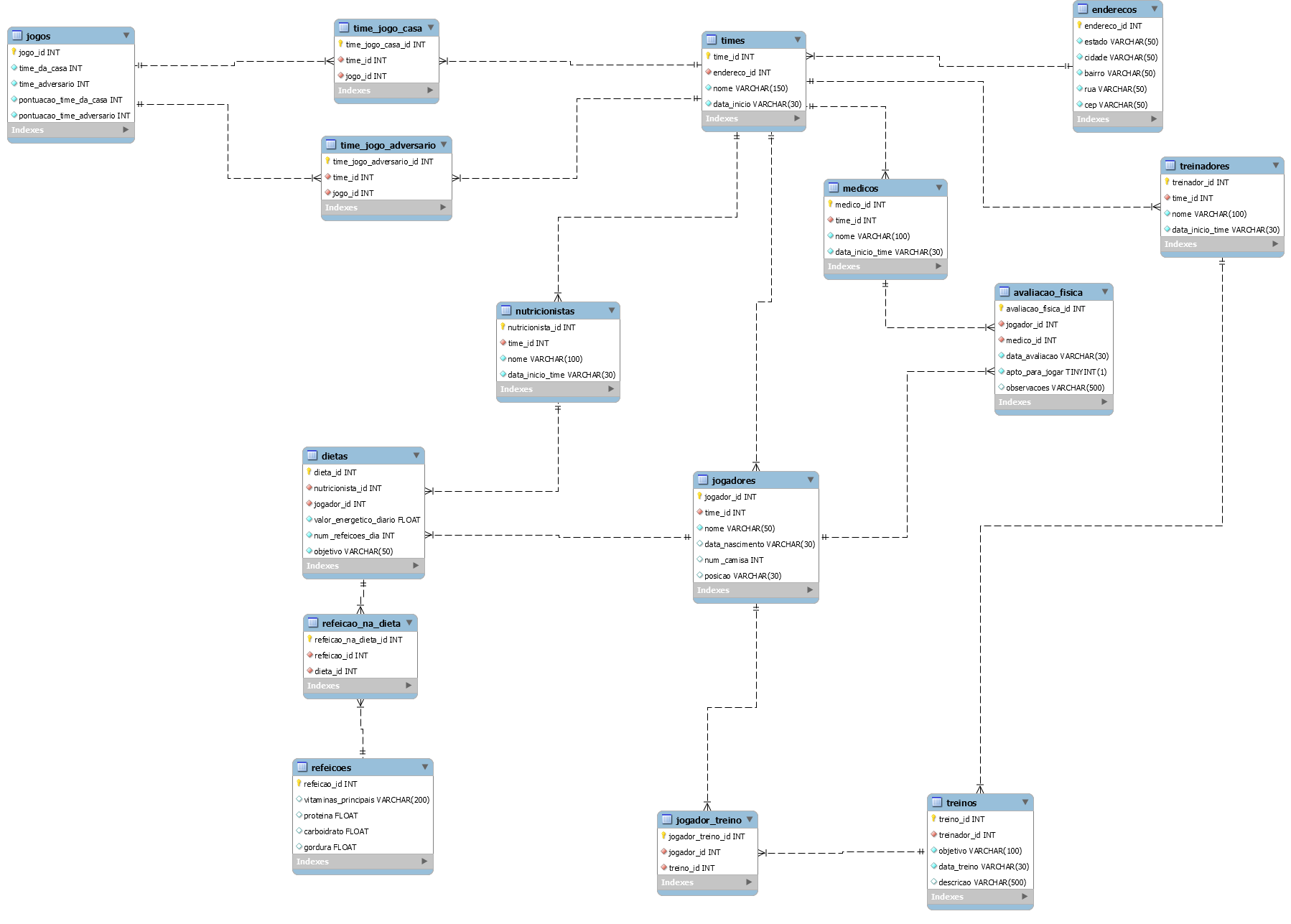
REFERENCES jogos(jogo\_id)

);

MODELO CONCEITUAL:



MODELO FÍSICO:



4A: INSERT

-- DADOS PURAMENTE FICTÍCIOS.

INSERT INTO enderecos(estado, cidade, bairro, rua, cep) VALUES ("SC", "Floripa", "Coqueiros", "Rua das ruas", "88886660");

INSERT INTO enderecos(estado, cidade, bairro, rua, cep) VALUES ("SC", "Floripa", "Lá longe", "Uma rua qualquer", "3353212");

INSERT INTO times(endereco\_id, nome, data\_inicio) VALUES(1, "Figueira", "1500-01-01");

INSERT INTO times(endereco\_id, nome, data\_inicio) VALUES(2, "Avai", "1800-01-01");

INSERT INTO jogadores(time\_id, nome, data\_nascimento, num\_camisa, posicao) VALUES(1, "Rafinha", "1699-04-04", 666, "atacante");

INSERT INTO jogadores(time\_id, nome, data\_nascimento, num\_camisa, posicao) VALUES(2, "Douglinhas", "1639-04-04", 333, "zagueiro");

INSERT INTO treinadores(time\_id, nome, data\_inicio\_time) VALUES(1, "Zacarias", "1790-02-13");

INSERT INTO treinadores(time\_id, nome, data\_inicio\_time) VALUES(2, "Dodo", "1360-02-13");

INSERT INTO treinos(treinador\_id, objetivo, data\_treino, descricao) VALUES(1, "Treinar ataque e gol partindo do escanteio", "1980-02-12", "time evoluindo");

INSERT INTO treinos(treinador\_id, objetivo, data\_treino, descricao) VALUES(2, "Treinar defesa espartana 300 AUUU", "1980-02-12", "time evoluindo muito");

INSERT INTO jogador\_treino(jogador\_id, treino\_id) VALUES(1, 3);

INSERT INTO jogador\_treino(jogador\_id, treino\_id) VALUES(2, 4);

INSERT INTO nutricionistas(time\_id, nome, data\_inicio\_time) VALUES(1, "Rolland da cozinha", "1687-12-21");

INSERT INTO nutricionistas(time\_id, nome, data\_inicio\_time) VALUES(2, "Cocorico da cozinha", "1634-12-21");

INSERT INTO refeicoes(vitaminas\_principais, proteina, carboidrato, gordura) VALUES("A, C e MAGNEZIO", 12.4, 31.2, 0.12);

INSERT INTO dietas(nutricionista\_id, jogador\_id, valor\_energetico\_diario, num\_refeicoes\_dia, objetivo) VALUES(1, 1, 3666, 8, "Ficar monstro");

INSERT INTO dietas(nutricionista\_id, jogador\_id, valor\_energetico\_diario, num\_refeicoes\_dia, objetivo) VALUES(2, 2, 1642, 4, "Ficar mais magrin");

-- MESMA REFEIÇÃO EM 2 DIETAS DIFERENTES

INSERT INTO refeicao\_na\_dieta(refeicao\_id, dieta\_id) VALUES(1, 1);

INSERT INTO refeicao\_na\_dieta(refeicao\_id, dieta\_id) VALUES(1, 2);

INSERT INTO medicos(time\_id, nome, data\_inicio\_time) VALUES(1, "Dr. Ray", "1945-12-31");

INSERT INTO medicos(time\_id, nome, data\_inicio\_time) VALUES(2, "Dr. Mitsunamy", "1934-02-11");

INSERT INTO avaliacao\_fisica(jogador\_id, medico\_id, data\_avaliacao, apto\_para\_jogar) VALUES(1, 1, "2020-12-12", TRUE);

INSERT INTO avaliacao\_fisica(jogador\_id, medico\_id, data\_avaliacao, apto\_para\_jogar) VALUES(2, 2, "2020-12-12", TRUE);

INSERT INTO jogos(time\_da\_casa, time\_adversario, pontuacao\_time\_da\_casa, pontuacao\_time\_adversario) VALUES(1, 2, 10, 0);

INSERT INTO jogos(time\_da\_casa, time\_adversario, pontuacao\_time\_da\_casa, pontuacao\_time\_adversario) VALUES(2, 1, 15, 2);

INSERT INTO time\_jogo\_casa(time\_id, jogo\_id) VALUES(1, 1);

INSERT INTO time\_jogo\_casa(time\_id, jogo\_id) VALUES(2, 2);

INSERT INTO time\_jogo\_adversario(time\_id, jogo\_id) VALUES(1, 2);

INSERT INTO time\_jogo\_adversario(time\_id, jogo\_id) VALUES(2, 1);

4B: UPDATE

UPDATE jogadores

SET time\_id = 2

WHERE jogador\_id = 1;

4C: DELETE

DELETE FROM treinadores

WHERE treinador\_id = 1;

4D: SELECT

SELECT times.nome, jogos.\* FROM times, jogos;

-- essa aqui eu vou ficar devendo ;-;

4E: SELECT

SELECT \* FROM jogadores, avaliacao\_fisica

WHERE avaliacao\_fisica.apto\_para\_jogar = TRUE;

