Conversion to Relational Schema

Evento(IdEvento, titulo NN, capa, descricao, localizacao NN, dataInicio NN, duracao NN, publico NN D{true})
Utilizador(IdUtilizador, nome NN, username NN UK, password NN, foto, email NN UK, IdPais → Pais NN)
Sondagem(IdSondagem, descricao, data NN, escolhaMultipla NN D{false}, IdEvento → Evento NN)
Opcao(IdOpcao, descricao NN, IdSondagem → Sondagem NN)
Comentario(IdComentario, texto NN, data NN, IdComentador → Utilizador NN, IdEvento → Evento NN, IdComentarioPai → Comentario)
Pais(IdPais, nome NN)
Seguidor(IdSeguidor → Utilizador, IdSeguido → Utilizador, data NN)
Participacao(IdEvento → Evento, IdParticipante → Utilizador, classificacao, comentario)
Convite(IdEvento → Evento, IdConvidado → Utilizador, data NN, resposta)
Anfitrião(IdEvento → Evento, IdAnfitriao → Utilizador)
UtilizadorOpcao(IdUtilizador → Utilizador, IdOpcao → Opcao)
ComentarioVoto(IdComentario → Comentario, IdVotante → Utilizador, positivo NN)
Administrador(IdAdministrador, username NN UK, email NN UK, password NN)
Album(IdAlbum, nome NN, descricao, IdEvento → Evento NN)
Imagem(IdImagem, caminho NN, data NN, IdAlbum → Album NN)

Dependências Funcionais

Evento

Chaves candidatas
{IdEvento}

Dependências funcionais
IdEvento → titulo, capa, descriçao, localizacao, dataInicio, duracao, publico

Utilizador

Chaves candidatas
{IdUtilizador}
{username}
{email}

Dependências funcionais
IdUtilizador → nome, username, password, foto, email, IdPais
username → IdUtilizador, nome, password, foto, email, IdPais
email → IdUtilizador, nome, username, password, foto, IdPais

Sondagem

Chaves candidatas
{IdSondagem}
Dependências funcionais
IdSondagem → descricao, data, escolhaMultipla, IdEvento

Opcao

Chaves candidatas
{IdOpcao}
Dependências funcionais
IdOpcao → descricao, IdSondagem

Comentario

Chaves candidatas
{IdComentario}
Dependências funcionais
IdComentario → texto, data, IdUtilizador, IdEvento, IdComentarioPai

Pais

Chaves candidatas
{IdPais}
Dependências funcionais
IdPais → nome

Seguidor

Chaves candidatas
{IdSeguidor, IdSeguido}
Dependências funcionais
IdSeguidor → IdUtilizador, IdSeguido → IdUtilizador, data

Participacao

Chaves candidatas
{IdEvento, IdParticipante}
Dependências funcionais
IdEvento, IdParticipante → classificacao, comentario

Convite

Chaves candidatas
{IdEvento, IdConvidado}
Dependências funcionais
IdEvento, IdConvidado → data, resposta

ComentarioVoto

Chaves candidatas
{IdComentario, IdVotante}
Dependências funcionais
IdComentario, IdVotante → positivo

Administrador

Chaves candidatas
{IdAdministrador}
{username}
{email}
Dependências funcionais
IdAdministrador → username, email, password
username → IdAdministrador, email, password
email → IdAdministrador, username, password

Album

Chaves candidatas
{IdAlbum}
Dependências funcionais
IdAlbum → nome, descricao, IdEvento

Imagem

Chaves candidatas
{IdImagem}
Dependências funcionais
IdImagem → caminho, data, IdAlbum

Formas Normais

O mapeamento do nosso modelo concetual para o *schema* relacional está na 1ª forma normal visto todos os atributos serem atómicos. Na 2ª forma normal porque em todas as chaves primárias compostas nenhum subconjunto dessa chave identifica um atributo não chave (não existem dependências parciais). Também está na 3ª forma normal uma vez que não existem dependências transitivas. E na Boyce-Codd porque todos os lados esquerdos das dependências funcionais são chaves candidatas.

Domínios

Classificação	ENUM ('0', '1', '2', '3', '4', '5')

Checks

Data	Data >= CURRENT_DATE
DataInicio	Data >= CURRENT_DATE
email	'^[^\s@]+@[^\s@]+\.[^\s@.]+$'
password	${#password} >= 8

SQL Database Creation

g22_creates.sql

```
DROP TABLE IF EXISTS Evento CASCADE;
DROP TABLE IF EXISTS Utilizador CASCADE;
DROP TABLE IF EXISTS Sondagem CASCADE;
DROP TABLE IF EXISTS Opcao CASCADE;
DROP TABLE IF EXISTS Comentario CASCADE;
DROP TABLE IF EXISTS Pais CASCADE;
DROP TABLE IF EXISTS Seguidor CASCADE;
DROP TABLE IF EXISTS Participacao CASCADE;
DROP TABLE IF EXISTS Convite CASCADE;
DROP TABLE IF EXISTS Anfitriao CASCADE;
DROP TABLE IF EXISTS UtilizadorOpcao CASCADE;
DROP TABLE IF EXISTS ComentarioVoto CASCADE;
DROP TABLE IF EXISTS Administrador CASCADE;
DROP TABLE IF EXISTS Album CASCADE;
DROP TABLE IF EXISTS Imagem CASCADE;

DROP DOMAIN IF EXISTS DMClassificacao;

CREATE DOMAIN DMClassificacao AS INTEGER CHECK( VALUE <= 5 AND VALUE >=
);

CREATE TABLE Evento (
  idEvento SERIAL PRIMARY KEY,
  titulo VARCHAR(100) NOT NULL,
  capa TEXT,
  descricao TEXT,
  localizacao TEXT NOT NULL,
  dataInicio TIMESTAMP NOT NULL CHECK (dataInicio >=
CURRENT_TIMESTAMP),
  duracao INTEGER NOT NULL,
  publico BOOLEAN DEFAULT TRUE NOT NULL
```

```
);

CREATE TABLE Pais(
  idPais SERIAL PRIMARY KEY,
  nome TEXT NOT NULL
);

CREATE TABLE Utilizador(
  idUtilizador SERIAL PRIMARY KEY,
  nome TEXT NOT NULL,
  username VARCHAR(100) UNIQUE NOT NULL,
  password VARCHAR(100) NOT NULL CHECK( LENGTH(password) >= 8),
  foto TEXT,
  email VARCHAR(100) UNIQUE NOT NULL CHECK ( email ~*
'^[^\s@]+@[^\s@]+\.[^\s@.]+$'),
  idPais INTEGER NOT NULL REFERENCES Pais(idPais)
);

CREATE TABLE Sondagem(
  idSondagem SERIAL PRIMARY KEY,
  descricao TEXT,
  "data" TIMESTAMP NOT NULL CHECK ("data" >= CURRENT_TIMESTAMP),
  escolhaMultipla BOOLEAN DEFAULT FALSE NOT NULL,
  idEvento INTEGER NOT NULL REFERENCES Evento(idEvento)
);

CREATE TABLE Opcao(
  idOpcao SERIAL PRIMARY KEY,
  descricao TEXT NOT NULL,
  idSondagem INTEGER NOT NULL REFERENCES Sondagem(idSondagem)
);

CREATE TABLE Comentario(
  idComentario SERIAL PRIMARY KEY,
  texto TEXT NOT NULL,
  "data" TIMESTAMP NOT NULL CHECK ("data" >= CURRENT_TIMESTAMP),
  idComentador INTEGER NOT NULL REFERENCES Utilizador(idUtilizador),
  idEvento INTEGER NOT NULL REFERENCES Evento(idEvento),
  idComentarioPai INTEGER REFERENCES Comentario(idComentario)
);

CREATE TABLE Seguidor(
  idSeguidor INTEGER REFERENCES Utilizador(idUtilizador),
  idSeguido INTEGER REFERENCES Utilizador(idUtilizador),
  "data" TIMESTAMP NOT NULL CHECK ("data" >= CURRENT_TIMESTAMP),
  PRIMARY KEY(idSeguidor, idSeguido)
);

CREATE TABLE Participacao(
  idEvento INTEGER REFERENCES Evento(idEvento),
  idParticipante INTEGER REFERENCES Utilizador(idUtilizador),
```

```
  classificacao DMClassificacao,
  comentario TEXT,
  PRIMARY KEY(idEvento, idParticipante)
);

CREATE TABLE Convite(
  idEvento INTEGER REFERENCES Evento(idEvento),
  idConvidado INTEGER REFERENCES Utilizador(idUtilizador),
  "data" TIMESTAMP NOT NULL CHECK ("data" >= CURRENT_TIMESTAMP),
  resposta BOOLEAN,
  PRIMARY KEY(idEvento, idConvidado)
);

CREATE TABLE Anfitriao(
  idEvento INTEGER REFERENCES Evento(idEvento),
  idAnfitriao INTEGER REFERENCES Utilizador(idUtilizador),
  PRIMARY KEY(idEvento, idAnfitriao)
);

CREATE TABLE UtilizadorOpcao(
  idUtilizador INTEGER REFERENCES Utilizador(idUtilizador),
  idOpcao INTEGER REFERENCES Opcao(idOpcao),
  PRIMARY KEY(idUtilizador, idOpcao)
);

CREATE TABLE ComentarioVoto(
  idComentario INTEGER REFERENCES Comentario(idComentario),
  idVotante INTEGER REFERENCES Utilizador(idUtilizador),
  positivo BOOLEAN NOT NULL,
  PRIMARY KEY(idComentario, idVotante)
);

CREATE TABLE Administrador(
  idAdministrador SERIAL PRIMARY KEY,
  username VARCHAR(100) UNIQUE NOT NULL,
  password VARCHAR(100) NOT NULL CHECK( LENGTH(password) >= 8),
  email VARCHAR(100) UNIQUE NOT NULL CHECK ( email ~*
'^[^\s@]+@[^\s@]+\.[^\s@.]+$')
);

CREATE TABLE Album(
  idAlbum SERIAL PRIMARY KEY,
  nome TEXT NOT NULL,
  descricao TEXT,
  idEvento INTEGER NOT NULL REFERENCES Evento(idEvento)
);

CREATE TABLE Imagem(
  idImagem SERIAL PRIMARY KEY,
  caminho TEXT NOT NULL,
  "data" TIMESTAMP NOT NULL CHECK ("data" >= CURRENT_TIMESTAMP),
```

```
  idAlbum INTEGER NOT NULL REFERENCES Album(idAlbum)
);
```

[EventBook]