SQL-LINGUAGEM DE DEFINIÇÃO DE DADOS

1. A BASE DE DADOS DAS ELEIÇÕES

Esta base de dados tem como contexto a informação relativa às eleições para a Assembleia da República, de 10 de Outubro de 1999.

O país está dividido em círculos eleitorais, aqui designados por distritos, embora nas ilhas o círculo eleitoral coincida com a região. Os partidos apresentam listas nos círculos eleitorais a que concorrem, sendo a esse nível que são obtidos os mandatos de deputado. Não são considerados os círculos dos emigrantes, pelo que o total de mandatos é apenas de 226.

No entanto, as votações em partidos são registadas com granularidade mais fina, ao nível de freguesia. As freguesias pertencem a concelhos, os quais por sua vez pertencem a distritos.

O total de votos em partidos, mais os votos em branco e os nulos, constituem os votantes. Somando votantes com abstenções obtém-se os inscritos. A informação sobre inscritos, abstenções, brancos e nulos só está presente ao nível de distrito.

Os partidos podem concorrer só a alguns distritos.

Modelo de dados

Esta base de dados tem 7 tabelas que irão ser povoadas com dados reais.

As tabelas **PARISHES**, **MUNICIPALITIES** e **DISTRICTS** dizem respeito às divisões administrativas e círculos eleitorais. Para cada freguesia sabe-se o seu código, o nome e o código do concelho a que pertence. Para cada concelho tem-se informação do seu código, nome e código do distrito (círculo eleitoral) a que pertence. Para cada distrito tem-se informação sobre o seu código, nome e região a que pertence ('C' - Continente, 'A' - Açores, 'M' - Madeira).

A tabela **LISTS** possui informação sobre o número de mandatos obtido por cada partido nos distritos em que concorreu. O valor 0 significa que concorreu mas não obteve nenhum mandato.

A tabela **VOTINGS** guarda o resultado obtido pelos diferentes partidos nas freguesias. Um valor nulo em votos significa que o partido concorreu mas não houve contagem de votos (boicote).

A tabela **PARTICIPATIONS** tem, para cada código de distrito, a informação relativa aos números de inscritos, votantes, abstenções, votos nulos e votos em branco. Note que:

inscritos = votantes + abstenções

votantes = soma de votos no distrito + brancos + nulos.

A tabela **PARTIES** guarda a sigla e designação dos 12 partidos concorrentes.

Modelo Relacional

```
DISTRICTS(code,name,region)
MUNICIPALITIES(code, name, district->DISTRICTS)
PARTICIPATIONS(district->DISTRICTS,registered_voters,voters,abstentions,blank_votes,null_votes)
PARTIES(acronym,designation)
LISTS(district->DISTRICTS,party->PARTIES,seats)
PARISHES(code,name,municipality->MUNICIPALITIES)
VOTINGS(parish->PARISHES,party->PARTIES,votes)
```

A. Implemente em SQL no SGBD SQLite o modelo relacional ilustrado acima, com as chaves primárias e externas indicadas.

   Comece pela tabela DISTRICTS e inclua as seguintes restrições nos seus atributos:

   - O atributo region apenas pode conter os valores ('C','M' ou 'A'), sendo o valor por defeito de 'C';
   - O atributo code apenas pode conter valores inteiros no intervalo [1,40];
   - O atributo name não pode ser nulo.

B. Implemente agora a tabela MUNICIPALITIES. Na declaração da chave externa, especifique o comportamento ON DELETE SET NULL e ON UPDATE CASCADE.

C. Usando o comando INSERT do SQL, insira o distrito com nome 'Porto', código 13, com o valor por defeito na região.

D. Insira o concelho com nome 'Matosinhos', com código 1308 e pertencendo ao distrito do 'Porto'. Experimente mudar o código do distrito do 'Porto' para 14 através do seguinte comando SQL:

   ```
   update DISTRICTS set code = 14 where code = 13;
   ```

   O que aconteceu ao atributo district do concelho de 'Matosinhos' na tabela de MUNICIPALITIES? Verifique correndo o seguinte comando SQL:

   ```
   select * from MUNICIPALITIES;
   ```

E. Usando o comando DELETE do SQL, elimine todos os tuplos da tabela DISTRICTS. O que aconteceu ao tuplo do concelho de 'Matosinhos' na tabela MUNICIPALITIES?

F. Implemente agora a tabela PARTICIPATIONS. Inclua as seguintes restrições:

   1. *registered_voters, voters, abstentions, blank_votes, null_votes >= 0*
   2. *registered_voters = voters + abstentions*
   3. *Poderia ter uma condição na criação desta tabela que validasse que os voters correspondem ao número total de votos no distrito + blank_votes + null_votes? Porquê?*

G. É correto dizer que na tabela PARTICIPATIONS o atributo district é simultaneamente chave primária e chave externa?

H. Implemente agora a tabela PARTIES e insira os dois seguintes tuplos nesta tabela, com uma única instrução de insert:

```
('PS','Partido Socialista')
('PPDPSD'.'Partido Social Democrata')
```

I. Implemente agora a tabela LISTS, declarando o comportamento ON UPDATE CASCADE para a chave externa em PARTIES. Inclua ainda uma restrição sobre o atributo seats, de forma ao respectivo valor estar no intervalo [0,48].

J. Insira o tuplo `(13,'PPDPSD',3)` na tabela LISTS. Corra agora o seguinte comando para modificar a sigla de 'PPDPSD' na tabela PARTIDOS para 'PSD':

```
update PARTIES set acronym = 'PSD' where acronym = 'PPDPSD';
```

O que aconteceu ao tuplo que tinha inserido na tabela LISTS? Verifique com o comando `select * from LISTS;`

K. Quantas chaves externas tem a tabela LISTS? E quantas chaves primárias?

L. Crie as restantes duas tabelas, PARISHES e VOTINGS.

2. RED BULL AIR RACE

Pretende-se armazenar informação relativa a uma época da competição Red Bull Air Race. Considere o seguinte esquema relacional.

Team (<u>name</u>, country)

Aircraft (<u>model</u>, horsepower, topspeed, width, height, weight)

Pilot (<u>num</u>, firstname, surname, nationality, birthday,
team -> Team, aircraft -> Aircraft)

Race (<u>location</u>, <u>edition</u>, country, date, gates, eliminations)

Participation (<u>pilot</u> -> Pilot, [<u>location, edition</u>] -> Race,
trainingtime, trainingpos, trainingpenalty,
qualificationtime, qualificationpos, qualificationpenalty,
eliminationtime, eliminationpos, eliminationpenalty)

Duel (<u>pilot1</u> -> Pilot, <u>pilot2</u> -> Pilot, [<u>location, edition</u>] -> Race, dueltype,
timepilot1, timepilot2, penaltypilot1, penaltypilot2)

Utilizando SQL, crie as tabelas necessárias, incluindo as respetivas regras de integridade, para este esquema relacional. Considere como restrições adicionais:

- Não pode haver duas corridas na mesma data;
- Todas as posições de partida em cada etapa (treino, qualificação e eliminação) têm que ser ≥ 1;
- Todos os tempos têm que ser positivos;
- Em cada participação, cada piloto só pode ter um tempo (de treino, de qualificação ou de eliminação) se tiver posição de partida respetiva.

3. OFICINA

Um concessionário de automóveis pretende informatizar o seu serviço de reparações em oficina. Considere o seguinte esquema relacional.

Marca (idMarca, nome)

Modelo (idModelo, nome, idMarca -> Marca)

CodPostal (codPostal1, localidade)

Cliente (idCliente, nome, morada, codPostal1 -> CodPostal, codPostal2, telefone)

Carro (idCarro, matricula, idModelo -> Modelo, idCliente -> Cliente)

Reparacao (idReparacao, dataInicio, dataFim,
idCliente -> Cliente, idCarro -> Carro)

Peca (idPeca, codigo, designacao, custoUnitario, quantidade)

ReparacaoPeca (idReparacao -> Reparacao, idPeca -> Peca, quantidade)

PecaModelo (idPeca -> Peca, idModelo -> Modelo)

Especialidade (idEspecialidade, nome, custoHorario)

Funcionario (idFuncionario, nome, morada, codPostal1 -> CodPostal, codPostal2,
telefone, idEspecialidade -> Especialidade)

FuncionarioReparacao (idFuncionario -> Funcionario,
idReparacao -> Reparacao, numHoras)

Utilizando SQL, crie as tabelas necessárias, incluindo as respetivas regras de integridade, para este esquema relacional. Antes de criar as tabelas verifique se as relações obtidas se encontram na Forma Normal de Boyce-Codd. Considere como restrições adicionais:

- Não pode haver quantidades, nem custos nem números de horas negativos;
- Uma reparação não pode terminar antes de começar;
- A matrícula de um carro é única;
- O código de uma peça é único.

4. FACULDADE

Considere a BD das classificações obtidas nas várias provas realizadas pelos alunos nas cadeiras de um ou mais cursos, com as tabelas e instâncias de seguida apresentadas:

ALUNO

nr	Nome
100	João
110	Manuel
120	Rui
130	Abel
140	Fernando
150	Ismael

PROF

sigla	Nome
ECO	Eugénio
FNF	Fernando
JLS	João

CADEIRA

cod	Design	curso	regente
TS1	Teoria dos Sistemas 1	IS	FNF
BD	Bases de Dados	IS	ECO
EIA	Estruturas de Informação e Algoritmos	IS	ECO
EP	Electrónica de Potência	AC	JLS
IE	Instalações Eléctricas	AC	JLS

PROVA

nr	cod	data	nota
100	TS1	92-02-11	8
100	TS1	93-02-02	11
100	BD	93-02-04	17
100	EIA	92-01-29	16
100	EIA	93-02-02	13
110	EP	92-01-30	12
110	IE	92-02-05	10
110	IE	93-02-01	14
120	TS1	93-01-31	15
120	EP	93-02-04	13
130	BD	93-02-04	12
130	EIA	93-02-02	7
130	TS1	92-02-11	8
140	TS1	93-01-31	10
140	TS1	92-02-11	13
140	EIA	93-02-02	11
150	TS1	92-02-11	10
150	EP	93-02-02	11
150	BD	93-02-04	17
150	EIA	92-01-29	16
150	IE	93-02-02	13

Note que a chave da tabela PROVA é constituída pelos atributos nr, cod e data, permitindo guardar o resultado de mais do que uma prova por cadeira para um mesmo aluno.

Utilizando SQL, crie as tabelas necessárias, incluindo as respetivas regras de integridade, para o modelo relacional dado. Preencha as tabelas com as instâncias apresentadas. [Baseado num exercício de Gabriel David]