Sistemas de Informação e Bases de Dados

Enunciado da Etapa 2 do Projeto de 2014/15

Ponto de partida

Considere como base de referência para esta segunda etapa do projeto o seguinte esquema relacional para a Fórmula E, especificado em SQL-DDL, correspondente a uma simplificação dos conceitos definidos para a primeira etapa:

```
DROP TABLE participa;
DROP TABLE corrida;
DROP TABLE piloto;
DROP TABLE equipa;

-- ---

CREATE TABLE equipa (
  sigla    CHAR(4),      -- Identificador mais curto de equipa.
  nome     VARCHAR2(80)  CONSTRAINT nn_equipa_nome     NOT NULL,
  pais     VARCHAR2(50)  CONSTRAINT nn_equipa_pais     NOT NULL,
  fundacao NUMBER(4)     CONSTRAINT nn_equipa_fundacao NOT NULL, -- Só ano.
--
  CONSTRAINT pk_equipa
    PRIMARY KEY (sigla),
--
  CONSTRAINT un_equipa_nome
    UNIQUE (nome),
--
  CONSTRAINT ck_equipa_fundacao
    CHECK (fundacao > 1800)
);

CREATE TABLE piloto (
  id         NUMBER(4),
  nome       VARCHAR2(80) CONSTRAINT nn_piloto_nome       NOT NULL,
  pais       VARCHAR2(50) CONSTRAINT nn_piloto_pais       NOT NULL,
  nascimento NUMBER(4)    CONSTRAINT nn_piloto_nascimento NOT NULL, -- Só ano.
  genero     CHAR(1)      CONSTRAINT nn_piloto_genero     NOT NULL,
  equipa                  CONSTRAINT nn_piloto_equipa     NOT NULL,
--
  CONSTRAINT pk_piloto
    PRIMARY KEY (id),
--
  CONSTRAINT fk_piloto_equipa
    FOREIGN KEY (equipa)
    REFERENCES equipa(sigla),
--
  CONSTRAINT ck_piloto_id
    CHECK (id > 0),
  CONSTRAINT ck_piloto_nascimento
    CHECK (nascimento > 1900),
  CONSTRAINT ck_piloto_genero
    CHECK (genero IN ('F', 'M'))
);
```

```
CREATE TABLE corrida (
  ordem  NUMBER(2),   -- Número de ordem da corrida na temporada.
  cidade VARCHAR2(50) CONSTRAINT nn_corrida_cidade NOT NULL,
  pais   VARCHAR2(50) CONSTRAINT nn_corrida_pais   NOT NULL,
  voltas NUMBER(2)    CONSTRAINT nn_corrida_voltas NOT NULL,
--
  CONSTRAINT pk_corrida
    PRIMARY KEY (ordem),
--
  CONSTRAINT un_corrida_cidade_pais
    UNIQUE (cidade, pais),
--
  CONSTRAINT ck_corrida_ordem
    CHECK (ordem BETWEEN 1 AND 10),
  CONSTRAINT ck_corrida_voltas
    CHECK (voltas > 0)
);

CREATE TABLE participa (
  piloto,
  corrida,
  pontos   NUMBER(2) CONSTRAINT nn_participa_pontos NOT NULL,
--
  CONSTRAINT pk_participa
    PRIMARY KEY (piloto, corrida),
--
  CONSTRAINT fk_participa_piloto
    FOREIGN KEY (piloto)
    REFERENCES piloto(id),
  CONSTRAINT fk_participa_corrida
    FOREIGN KEY (corrida)
    REFERENCES corrida(ordem),
--
  CONSTRAINT un_participa_corrida_pontos
    UNIQUE (corrida, pontos),
--
  CONSTRAINT ck_participa_pontos
    CHECK (pontos IN (1, 2, 4, 6, 8, 10, 12, 15, 18, 25))
);
```

As simplificações incluem o ser considerada apenas uma temporada, cada corrida ter apenas uma fase, que é a da corrida em si, existirem apenas pilotos principais associados às equipas, e o serem guardados os pontos conquistados por cada piloto em vez de tempos e número de voltas.

Um elemento novo no esquema é a chave primária sigla em equipa. Embora não possam existir duas equipas com o mesmo nome, é mais eficiente usar siglas curtas para referenciar equipas do que nomes potencialmente longos, para além de ser poupado espaço em disco para armazenar essas referências.

Objetivos

1. Codificar um pacote PL/SQL de suporte à gestão de dados da Fórmula E. A especificação das funcionalidades vem na secção seguinte.
2. Preparar uma demonstração de gestão de dados da Fórmula E, em forma de *script* contendo invocações das funções e procedimentos do pacote PL/SQL.
3. Traduzir para interrogações SQL os pedidos de dados indicados em secção própria, mais à frente.

Especificação de funcionalidades

As funções e procedimentos seguintes devem fazer parte de um pacote PL/SQL chamado `formula_e`.

`regista_equipa(sigla, nome, pais, fundacao)`
: Regista uma equipa com sigla, nome, país, e ano de fundação. O número máximo de equipas que podem ser registadas é dez.

`regista_piloto(nome, pais, nascimento, genero, equipa) -> id`
: Regista um piloto com nome, país e ano de nascimento, e género, atribuindo-lhe um ID que deve ser devolvido. Também inscreve o piloto na equipa, até ao limite de dois por equipa.

`regista_corrida(ordem, cidade, pais, voltas)`
: Regista uma corrida com número de ordem na temporada, cidade e país, e número de voltas ao circuito. O número máximo de corridas da temporada é dez e não podem existir duas corridas na mesma cidade e país, mesmo que com números de ordem diferentes.

`regista_participacao(piloto, corrida, pontos)`
: Regista os pontos conquistados por um piloto numa corrida. Cada piloto só pode participar uma vez em cada corrida e em cada corrida só um piloto pode obter cada valor da escala de pontuação. Por exemplo, numa corrida só um piloto pode conquistar os 25 pontos do primeiro lugar.

`remove_equipa(nome)`
: Remove os dados da equipa, dos seus pilotos, e das participações dos seus pilotos nas corridas.

`remove_piloto(id)`
: Remove os dados do piloto e das suas participações em corridas.

`remove_corrida(ordem)`
: Remove os dados da corrida e das participações de pilotos nessa corrida.

`remove_participacao(piloto, corrida)`
: Remove os dados da participação do piloto na corrida.

`posicao(pontos) -> posicao`
: Calcula a posição de chegada no final da corrida a partir dos pontos obtidos por um piloto. Por exemplo, `posicao(25) -> 1`.

As funcionalidades de atualização dos dados das equipas, pilotos, corridas, e participações em corridas *não* são necessárias para esta etapa do projeto.

Pedidos de dados

1. Anos de fundação e nomes das equipas do Japão com pelo menos um piloto nascido após 1990. *Nota 1:* pretende-se uma interrogação com apenas um SELECT, isto é, sem sub-interrogações. *Nota 2:* o resultado deve vir ordenado pelo ano de fundação de forma descendente e pelo nome da equipa de forma ascendente.
2. IDs e nomes das mulheres piloto que nasceram após 1990, ou então que participaram em pelo menos uma corrida e nasceram antes de 1980. Nota: pode usar construtores de conjuntos.
3. Nomes dos pilotos e suas equipas, nascidos depois de 1990, e que pontuaram nas duas primeiras corridas da temporada. *Nota 1:* pretende-se uma interrogação sem sub-interrogações. *Nota 2:* o resultado deve vir ordenado pelo nome do piloto e pelo nome da equipa.
4. Nomes dos pilotos e suas equipas que nunca participaram em corridas no país da equipa.

5. Nomes das equipas cujos pilotos (todos) participaram em pelo menos uma corrida com mais de 20 voltas ao circuito.
6. Soma dos pontos obtidos por cada equipa em cada corrida. Nota: o resultado deve vir ordenado pelo nome da equipa e pelo número de ordem da corrida, ambos de forma ascendente.
7. Nomes das equipas com mais pontos por corrida.
8. Média dos pontos obtidos por pilotos que nasceram depois de 1990 e que participaram em todas as corridas do seu país.

Extra:

9. Crie duas vistas, chamadas `tabela_classificativa_pilotos` e `tabela_classificativa_equipas` com os pontos acumulados pelos pilotos e equipas nas várias corridas da temporada.

A ter em conta

- Em caso de erro, as funções e procedimentos devem lançar exceções com texto explicativo.
- O pacote PL/SQL deve estar comentado, as variáveis e parâmetros devem ter nomes inteligíveis, e devem ser respeitadas as regras de alinhamento dos blocos de código.
- As interrogações SQL devem ser inteligíveis, seguindo a mesma apresentação dos exemplos das aulas teóricas, e sem terem mudanças bruscas de linha.
- Antes de cada interrogação deve escrever, em forma de comentário SQL, o pedido de dados em português, incluindo o respetivo número.
- As interrogações SQL devem ser o mais simples possível, evitando, por exemplo, a inclusão de tabelas desnecessárias na cláusula FROM ou o uso de sub-interrogações.
- Em caso de dúvida sobre a interpretação de valores duplicados no resultado, pode desambiguar acrescentando atributos à cláusula SELECT.
- O enunciado pode vir a ser alterado pontualmente, sendo esse facto indicado na página da disciplina e nas aulas.

Estrutura e entrega do relatório

- No cabeçalho *de cada folha* deve estar a sigla da disciplina, a data, a etapa do projeto a que se refere, o grupo, e o nome, número, e turma dos alunos.
- A especificação e código do pacote PL/SQL vem no início do relatório.
- A seguir vem um *script* com comandos de invocação das funções e procedimentos, demonstrando as várias operações possíveis sobre os dados.
- Depois vêm as interrogações SQL para responder aos pedidos de dados.
- *Não deve ser entregue folha de rosto nem deve ser usado qualquer tipo de encadernação.*
- As folhas devem ser agrafadas no canto superior esquerdo.
- O relatório é entregue em papel na caixa de correio do professor da turma teórica-prática respetiva e também em formato PDF através de opção disponível na página de entrada da disciplina.
- O prazo de entrega do relatório é o final do dia 12 de dezembro de 2014.

Bom trabalho na etapa 2 do projeto!