Prova de Fundamentos de Bancos de Dados

1^a Prova

Prof. Carlos A. Heuser

Setembro de 2010

Nome do aluno:

Prova *sem* consulta – duas horas de duração

1. *(Peso 2)*

   Deseja-se projetar uma base de dados para uma empresa de telefonia móvel. Nesta base de dados estão armazenadas informações sobre os planos que os clientes adquiriram.

   Para cada cliente, é necessário conhecer seu CPF, seu nome e os planos que ele adquiriu.

   Um plano recebe um número identificador e uma denominação. Para cada plano, a empresa deseja conhecer o seu preço, a data em que começou e a data em que terminou de ser vendido, juntamente com o número de minutos de fala adquiridos através do plano. Opcionalmente, um plano pode estar associado a um determinado tipo de aparelho que é oferecido aos adquirentes daquele plano. Obviamente, um plano pode ser adquirido por um grande número de clientes.

   Cada tipo de aparelho recebe um número interno de identificação e é necessário conhecer sua marca e seu modelo. Um mesmo tipo de aparelho pode constar de diversos planos.

   Para cada plano adquirido por um cliente, a base de dados deve armazenar a data de aquisição e o número de telefone habilitado naquele plano.

   Projete uma base de dados relacional para armazenar os dados acima sem redundância de dados. Enumere as tabelas, suas colunas, as chaves primárias e a as chaves estrangeiras. Não devem ser criadas colunas artificiais, além das apresentadas no enunciado. Apresente o esquema na notação textual ou diagramática vistas em aula.

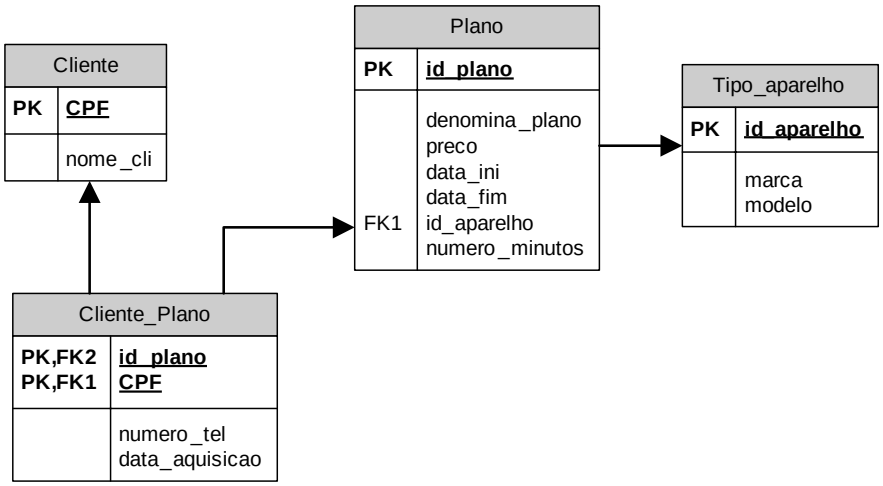

Figura 1: Resultado da Questão 1

2. Considere a seguinte base de dados, usada por uma oficina de manutenção de automóveis.

   /* tabela de clientes cadastrados na oficina – `cpf_conjuge` é o código do cônjuge do marido, caso este esteja na base de dados */

   ```
   CLIENTE (cpf, nome_cli, cpf_conjuge)
   ```

   /* tabela com dados dos automóveis dos clientes da oficina */

   ```
   AUTOMOVEL (placa, no_chassis, modelo, cpf);
      (cpf) references CLIENTE
   ```

   /* tabela com as revisões periódicas programadas e feitas – para cada automóvel, a oficina cadastra todas revisões programadas –
   `Km` e `data_programada` são a quilometragem e a data em que deve ser feita a revisão –
   `data_ultim_telef` serve para informar quando o pessoal da oficina ligou para o cliente lembrando da provável necessidade de fazer a revisão – caso o cliente não tenha sido chamado, este campo contém a cadeia vazia (″)
   `data_executada` e `Km_executada` informa a data e a quilometragem de uma revisão que já foi executada – caso a revisão não tenha sido chamado, estes campos contém a cadeia vazia (″) */

   ```
   REVISAO (placa, Km, data_programada, data_ultim_telef,
            data_executada, Km_executada)
      (placa) references AUTOMOVEL
   ```

   /* tabela com as peças usadas em cada revisão */

   ```
   PECA_REVISAO (placa, Km, cod_peca, quantidade)
      (placa, Km) references REVISAO
      (cod_peca) references PECA
   ```

/* tabela com as descrições das peças */

```
PECA (cod_peca, descricao_peca)
```

Sobre esta base de dados, resolver as consultas que seguem usando *álgebra relacional*. Não usar mais tabelas que o estritamente necessário.

a) *(Peso 1,33...)*
Obter o código e a descrição de cada peça que foi utilizada em uma revisão do automóvel de número de chassis 125678.

i. Resolver usando produto cartesiano.
Solução:

```
π PECA.cod_peca,
  PECA.descricao_peca
  (σ PECA.cod_peca = PECA_REVISAO.cod_peca AND
   PECA_REVISAO.placa = AUTOMOVEL.placa AND
   AUTOMOVEL.no_chassis = 125678
      (PECA ×
       PECA_REVISAO ×
       AUTOMOVEL
      )
  )
```

ii. Resolver usando junções. Se possível, usar junção natural, senão, usar equi-junção e em último caso usar theta-junção.
Solução:

```
π PECA.cod_peca,
  PECA.descricao_peca
  (σ  AUTOMOVEL.no_chassis = 125678
      (PECA ⋈
         (PECA_REVISAO ⋈
          AUTOMOVEL
         )
      )
  )

  )
```

b) *(Peso 1,33...)*
Para cada cliente que possui cônjuge cadastrado na base de dados, obter o nome do cliente, seguido do nome de seu cônjuge.

Solução:

```
π CLIENTE.nome_cli,
  CONJUGE.nome_cli
     (CLIENTE
      ⋈ (CLIENTE.cpf_conjuge = CONJUGE.CPF)
      (ρ CONJUGE (CLIENTE)
      )
     )
```

c) *(Peso 1,33...)*
Obter **uma única** tabela contendo as seguintes colunas:

i. CPF e nome de cada cliente;
ii. placa de cada automóvel registrado em seu nome;
iii. quilometragem em que foi executada cada revisão neste automóvel (lembrar que `Km_executada` é diferente da cadeia vazia (″), quando a revisão tiver sido executada).

Caso o cliente não tenha automóveis, as duas últimas colunas devem estar vazias. Caso um automóvel não tenha sido revisado ainda, a última coluna deve constar como vazia.

Solução:

```
π CLIENTE.CPF,
   CLIENTE.nome_cli,
   AUTOMOVEL.placa,
   REVISAO.Km_executada
    ((CLIENTE
      ⟕
      AUTOMOVEL
      )
      ⟕
           (σ Km_executada <> ''
             (REVISAO)
           )
      )
```

d) *(Peso 1,33...)*
Sabe-se que algumas peças, como filtro de ar, podem ter sido usadas em todas revisões já feitas. Obter o código e a descrição de cada peça usada em todas revisões que constam na base de dados.

Solução:

```
(   (π placa, Km, cod_peca (PECA_REVISAO))
    ÷
    (π placa, Km
      (σ Km_executada <> '' (REVISAO))
)
⋈
PECA
```

e) *(Peso 1,33...)*
Obter o CPF e o nome de cada cliente que não possui automóveis cadastrados na base de dados.

Solução:

```
π CPF, nome_cli (CLIENTE)
−
(π CPF, nome_cli
       (CLIENTE ⋈ AUTOMOVEL)
)
```

3. *(Peso 1,33...)*
   Considere a seguinte consulta em SQL:

```
SELECT CLIENTE.nome
FROM CLIENTE,
     AUTOMOVEL,
     PECA_REVISAO
WHERE
     CLIENTE.cpf = AUTOMOVEL.cpf AND
     PECA_REVISAO.placa = AUTOMOVEL.placa AND
     PECA_REVISAO.quantidade = 200 AND
     AUTOMOVEL.modelo = 'Packard conversível'
```

   Mostre a consulta equivalente em álgebra relacional, depois, mostre a representação da consulta em forma de arvore e após, mostre cada um dos passos da otimização algébrica.