Lista de Exercícios 5

Beatriz Rodrigues de Oliveira Paiva

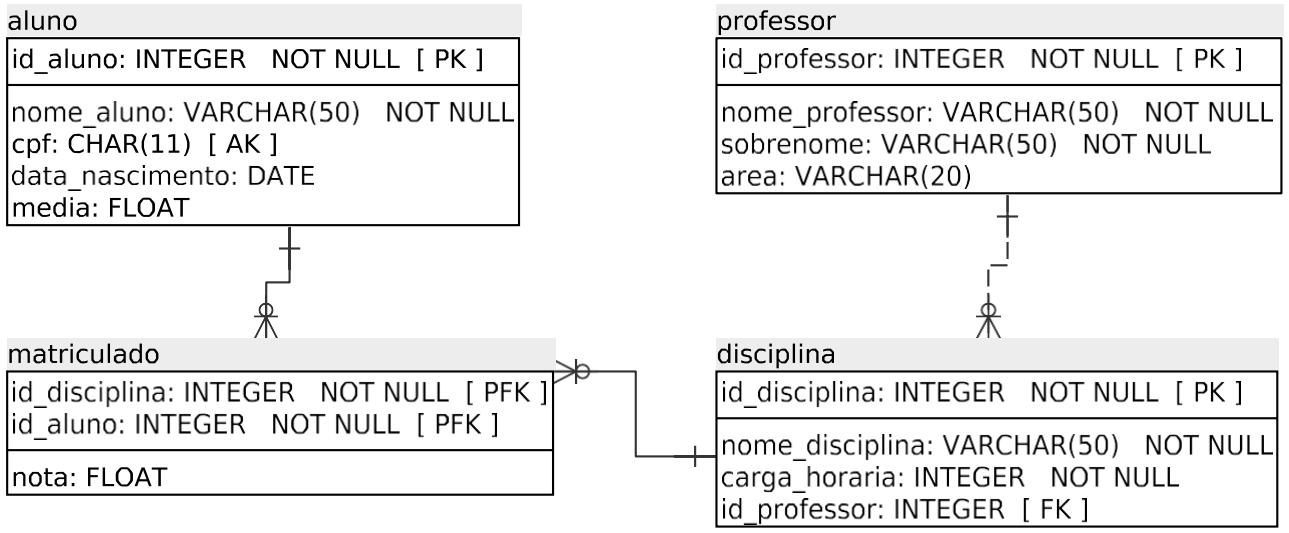

Figura 1: Banco de dados acadêmico

Exercício 1:

Considere agora o banco de dados *Acadêmico* mostrado na Figura 1. Escreva as instruções SQL para executar as seguintes ações:

(a) Obter o nome do aluno, o nome da disciplina e a nota obtida pelo aluno na disciplina;

```
select a.nome_aluno,
   d.nome_disciplina,
   m.nota
from aluno as a,
   disciplina as d,
   matriculado as m
where a.id_aluno = m.id_aluno and d.id_disciplina = m.id_disciplina
      ORDER BY (a.id_aluno);
```

(b) Obter a quantidade de disciplinas ministradas por cada professor;

```
select p.nome_professor,
   count (*) as num_disciplinas
from professor as p,
   disciplina as d
where p.id_professor = d.id_professor
GROUP BY p.id_professor
UNION
select distinct p.nome_professor,
   0
from professor as p,
   disciplina as d
where not exists(
```

```
   select d.id_professor
   from disciplina as d
   where p.id_professor = d.id_professor
);
```

(c) Obter os nomes completos de todos os professores com suas carga horária total;

```
select p.nome_professor ||' '|| p.sobrenome as Nome_Completo,
   SUM(d.carga_horaria) as carga_total
from professor as p,
   disciplina as d
where p.id_professor = d.id_professor
GROUP BY p.id_professor
UNION
select p.nome_professor ||' '|| p.sobrenome,
   0
FROM professor as p,
   disciplina as d
WHERE NOT EXISTS (
   SELECT d.id_professor
   FROM disciplina AS d
   WHERE p.id_professor = d.id_professor);
```

(d) Obter a nota média para cada disciplina;

```
select d.nome_disciplina,
   AVG(m.nota)
from disciplina as d,
   matriculado as m
where d.id_disciplina = m.id_disciplina
GROUP BY d.id_disciplina;
```

(e) Obter a maior e a menor nota para cada uma das disciplinas;

```
select d.nome_disciplina,
   MAX(m.nota),
   MIN(m.nota)
FROM disciplina as d,
   matriculado as m
where d.id_disciplina = m.id_disciplina
GROUP BY d.id_disciplina;
```

(f) Obter as disciplinas que o aluno *José* está matriculado e que possuam pelo menos 2 alunos matriculados.

```
select d.nome_disciplina
from disciplina as d,
   matriculado as m
where (d.id_disciplina = m.id_disciplina)
   and (m.id_aluno = (select id_aluno from aluno where nome_aluno='José'))
INTERSECT
select d.nome_disciplina
from disciplina as d,
   matriculado as m
where d.id_disciplina = m.id_disciplina
GROUP BY d.id_disciplina HAVING count(*) >=2;
```

(g) Obter os alunos matriculados nas disciplinas com com carga horária maior ou igual a 60;

```
select distinct a.nome_aluno
from aluno as a,
   matriculado as m,
   disciplina as d
where (a.id_aluno = m.id_aluno)
   and (m.id_disciplina = d.id_disciplina)
   and (d.carga_horaria >=60);
```

(h) Obter a média das notas de cada aluno em ordem decrescente pela média;

```
select nome_aluno,
   media
from aluno
ORDER BY media DESC;
```

(i) Obter os nomes dos alunos matriculado em disciplinas de professores da área de *Computação*;

```
CREATE TEMPORARY TABLE media AS
SELECT id_aluno,
    AVG(nota) AS media
FROM matriculado
GROUP BY id_aluno;

UPDATE aluno AS a
SET media = m.media
FROM media AS m WHERE a.id_aluno = m.id_aluno;
```

(j) Obter a carga horária total de cada professor de acordo com as disciplinas ministradas;

```
select distinct a.nome_aluno
from aluno as a,
   matriculado as m,
   disciplina as d
where (m.id_aluno = a.id_aluno)
   and (d.id_disciplina = m.id_disciplina)
   and (d.id_professor IN (
      select p.id_professor
      from professor as p
      where p.area ='Computação'));
```

(k) Obter a quantidade de alunos matriculados em cada disciplinas;

```
select p.nome_professor,
   SUM(d.carga_horaria) as carga_total
from professor as p,
   disciplina as d
where p.id_professor = d.id_professor
GROUP BY p.id_professor
UNION
select p.nome_professor,
   0
FROM professor as p,
   disciplina as d
WHERE NOT EXISTS (
```

```
    SELECT d.id_professor
    FROM disciplina AS d
    WHERE p.id_professor = d.id_professor);
```

(l) Obter os alunos que cursaram mais de 100 horas de disciplinas;

```
select d.nome_disciplina,
    count(*)
from disciplina as d,
    matriculado as m
where (d.id_disciplina = m.id_disciplina)
GROUP BY (d.nome_disciplina);
```

(m) Obter o nome dos alunos, nome das disciplinas, nome dos professores da disciplinas e a nota obtida pelos alunos;

```
select a.nome_aluno
from disciplina as d,
   matriculado as m,
   aluno as a
where (m.id_aluno=a.id_aluno)
   and (m.id_disciplina = d.id_disciplina)
GROUP BY (a.id_aluno)
HAVING (sum(d.carga_horaria)>100);
```

(n) Obter as disciplinas sem professor;

```
select distinct a.nome_aluno,
    d.nome_disciplina,
    p.nome_professor,
    m.nota
from aluno as a,
    disciplina as d,
    professor as p,
    matriculado as m
where (m.id_aluno = a.id_aluno)
    and (m.id_disciplina = d.id_disciplina)
    and (p.id_professor = d.id_professor)
ORDER BY (nome_aluno);
```

(o) Obter os professores sem disciplina;

```
select d.nome_disciplina
from disciplina as d
where d.id_professor is null;
```

(p) Obter possíveis duplas combinando todos os nomes de alunos, mas sem combinar um aluno com ele mesmo;

```
select distinct p.nome_professor
FROM professor as p,
   disciplina as d
WHERE NOT EXISTS (
   SELECT d.id_professor
   FROM disciplina AS d
   WHERE p.id_professor = d.id_professor);
```

(q) Obter as disciplinas sem nenhum aluno matriculado;

```
select a1.nome_aluno, a2.nome_aluno
from aluno as a1, aluno as a2
where a1.id_aluno<a2.id_aluno
order by a1.id_aluno;
```

(r) Matricular todos os alunos nas disciplinas sem nenhum aluno matriculado.

```
select d.id_disciplina,
   d.nome_disciplina
from disciplina as d,
   matriculado as m
where not exists(
   select m.id_disciplina
   from matriculado as m
   where (d.id_disciplina = m.id_disciplina))
   GROUP BY (d.id_disciplina);
```

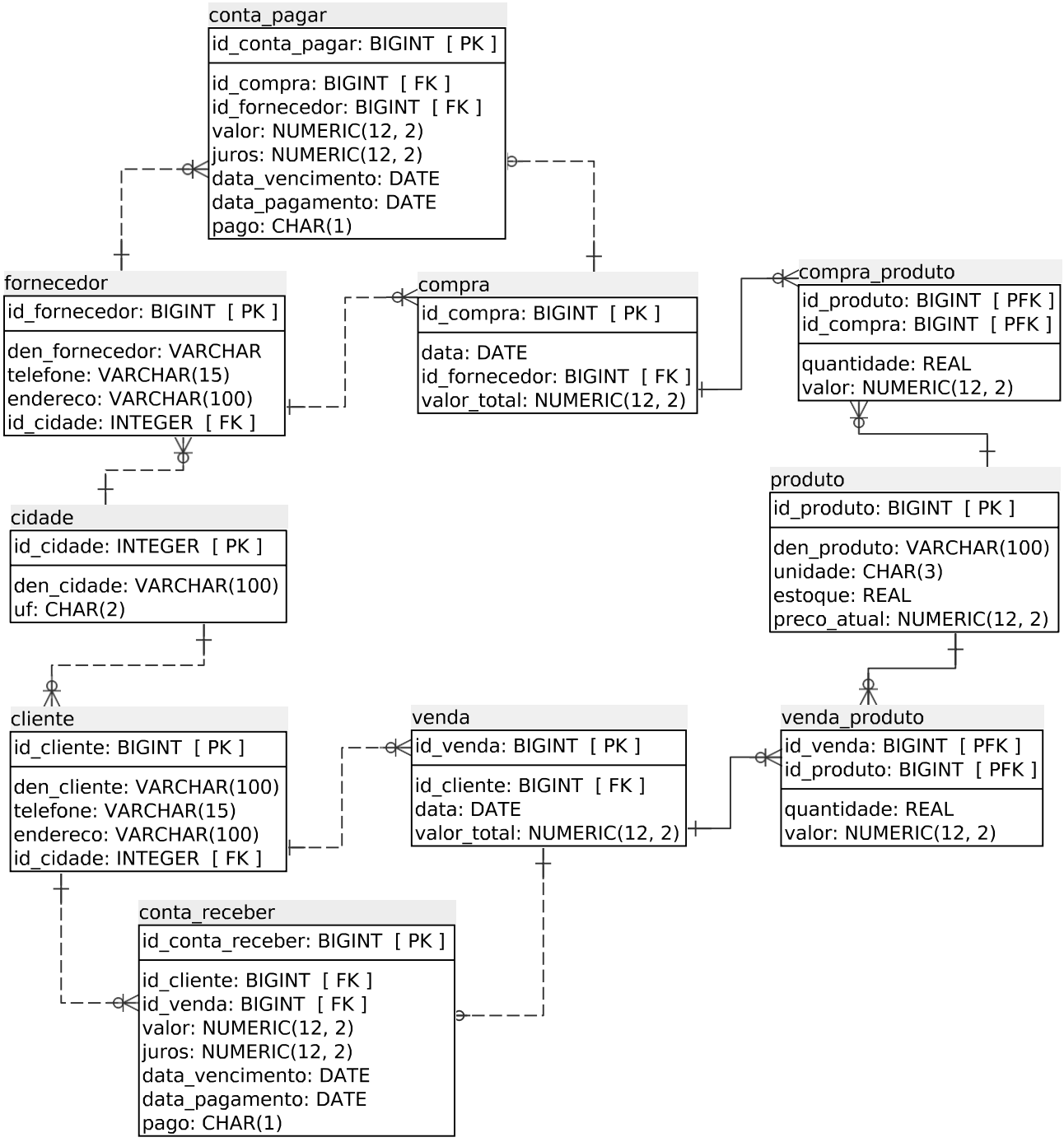

Figura 2: Banco de dados de uma empresa de varejo

Exercício 2:

Considere o banco de dados de uma empresa de varejo cujo esquema lógico é apresentado na Figura 2. Escreva as instruções SQL para executar as seguintes ações:

(a) Obter o valor total comprado de cada fornecedor;

```
select f.den_fornecedor, SUM(c.valor_total)
from fornecedor as f,
   compra as c
where (c.id_fornecedor = f.id_fornecedor)
GROUP BY (f.id_fornecedor);
```

(b) O valor total vendido para cada cidade;

```
select ci.id_cidade,
   ci.den_cidade,
   SUM(v.valor_total)
from cliente as c,
   venda as v,
   cidade as ci
where (v.id_cliente = c.id_cliente)
   and (c.id_cidade = ci.id_cidade)
GROUP BY(ci.id_cidade);
```

(c) O valor total de cada produto vendido para cada cidade;

```
select ci.den_cidade,
   p.den_produto,
   SUM(vp.quantidade * vp.valor)
from cidade as ci,
   venda_produto as vp,
   cliente as c,
   venda as v,
   produto as p
where (vp.id_produto = p.id_produto)
   and (vp.id_venda = v.id_venda)
   and (v.id_cliente = c.id_cliente)
   and (c.id_cidade = ci.id_cidade)
GROUP BY(ci.id_cidade, p.id_produto)
ORDER BY (ci.id_cidade);
```

(d) A quantidade, o valor total e o valor médio de cada produto comprado de cada estado (UF);

```
select p.id_produto,
   p.den_produto,
   ci.uf,
   sum(cp.quantidade) as quantidade,
   sum(cp.valor * cp.quantidade) as total,
   avg(cp.valor) as valor_medio
from cidade as ci,
   fornecedor as f,
   compra as co,
   produto as p,
   compra_produto as cp
where co.id_fornecedor = f.id_fornecedor
   and f.id_cidade = ci.id_cidade
   and co.id_compra = cp.id_compra
```

```
   and cp.id_produto = p.id_produto
group by p.id_produto, ci.uf
order by ci.uf;
```

(e) Listar as cidades com suas respectivas quantidades de cadastros (um cadastro pode ser um cliente ou um fornecedor);

```
create temporary table cf as(
   select id_cidade from fornecedor
   union all
   select id_cidade from cliente
);
select c.id_cidade,
   c.den_cidade,
   count(*)as cadastros
from cidade as c, cf
where c.id_cidade = cf.id_cidade
group by c.id_cidade;
```

(f) Listar as contas a pagar vencidas até dezembro de 2011 e que não foram pagas;

```
select id_conta_pagar
from conta_pagar
where (data_vencimento <= '2011-12-31')
   and pago <>'S';
```

(g) Listar os fornecedores que possuem mais de 10 contas a pagar;

```
select f.id_fornecedor,
   f.den_fornecedor,
   count(*) as contas_a_pagar
from fornecedor as f,
   conta_pagar as cp
where f.id_fornecedor = cp.id_fornecedor
GROUP BY f.id_fornecedor
HAVING count(*)>10;
```

(h) Listar o total devido por cada cliente;

```
select * from((select c.id_cliente,
   SUM(cr.valor) as valor
from cliente as c,
   conta_receber as cr
where (c.id_cliente = cr.id_cliente)
   and (cr.pago = 'N')
GROUP BY (c.id_cliente)
ORDER BY (c.id_cliente))
```

```
UNION ALL
((select id_cliente, 0
from cliente)
EXCEPT ALL
(select c.id_cliente, 0
from cliente as c,
   conta_receber as cr
where (c.id_cliente = cr.id_cliente)
   and (cr.pago = 'N')
GROUP BY (c.id_cliente))))
as consulta
ORDER BY (consulta.id_cliente);
```

(i) Listar os 10 produtos com maior movimentação (considerando compras e vendas);

```
create temporary table movimentacao as (
   select id_produto,
      quantidade
   from venda_produto
   union all
   select id_produto,
      quantidade
   from compra_produto
);
select p.id_produto,
   p.den_produto,
   sum(m.quantidade) as quantidade
from produto as p,
   movimentacao as m
where p.id_produto = m.id_produto
group by (p.id_produto)
order by quantidade DESC
limit 10;
```

(j) Listar o faturamento (vendas) mensal de todos os meses;

```
SELECT EXTRACT(year FROM data) || '-' || DATE_PART('month', data) AS mes,
       SUM(vp.quantidade) AS Faturamento_mensal
  FROM venda AS v,
      venda_produto AS vp
  WHERE v.id_venda = vp.id_venda
  GROUP BY EXTRACT(year FROM data) || '-' || DATE_PART('month', data);
```

(k) Listar os produtos que foram vendidos, mas não foram comprados em janeiro de 2011;

```
CREATE TEMP TABLE media AS
WITH mm as (
  SELECT vp.id_produto,
        EXTRACT(year FROM data) || '-' || DATE_PART('month', data) AS mes,
        AVG(vp.quantidade) AS media
  FROM venda AS v,
      venda_produto AS vp
  WHERE v.id_venda = vp.id_venda
  GROUP BY id_produto, EXTRACT(year FROM data) || '-' || DATE_PART('month', data)
)
SELECT id_produto,
      AVG(media) AS media
FROM mm
GROUP BY id_produto;

ALTER TABLE produto ADD estoque_minimo REAL;

UPDATE produto AS p
SET estoque_minimo = m.media * 0.5
FROM media AS m
WHERE m.id_produto = p.id_produto;
```

(l) Listar os produtos que foram comprados em 2010, sem repetições;

```
select distinct p.id_produto
from produto as p,
    venda_produto as vp
where (p.id_produto = vp.id_produto)
EXCEPT
select distinct p.id_produto
from produto as p,
    compra_produto as cp,
    compra as c
where (p.id_produto = cp.id_produto)
    and (c.id_compra = cp.id_compra)
    and (c.data >='2011-01-01')
    and (c.data <='2011-01-31');
```

(m) Listar as vendas de 2011 contendo mais de 5 produtos que ainda não foram pagas;

```
select distinct p.den_produto
from produto as p,
   compra_produto as cp,
   compra as c
```

```
where (p.id_produto = cp.id_produto)
   and (cp.id_compra = c.id_compra)
   and (c.data>='2010-01-01')
   and (c.data<='2010-12-31');
```

(n) Listar o total vendido e o total comprado de cada produto (pode acontecer de um produto ter sido comprado e não ter sido vendido e vice-versa);

```
select v.id_venda
from produto as p,
   venda_produto as vp,
   venda as v,
   conta_receber as cr
where (v.id_venda = vp.id_venda)
   and (vp.id_produto = p.id_produto)
   and (v.data >= '2011-01-01')
   and (v.data <= '2011-12-31')
   and (cr.id_venda = v.id_venda)
GROUP BY (v.id_venda)
HAVING (count(*)>=5);
```

(o) Listar o mês e ano com a maior quantidade de vendas de cada produto.

```
select p.den_produto,
   (select SUM(vp.quantidade) as total_vendido
   from venda_produto as vp
   where (p.id_produto = vp.id_produto)
   GROUP BY (p.id_produto)),
   (select SUM(cp.quantidade) as total_comprado
   from compra_produto as cp
   where (p.id_produto = cp.id_produto)
   GROUP BY (p.id_produto))
from produto as p;
```