PROVAS DE TI
TUDO PARA VOCÊ PASSAR

Banco de Dados

Exercícios da banca Cesgranrio

Rodrigo Adur
rodrigoadurti@gmail.com

ITnerante

TIMASTERS

Apresentação

ITnerante

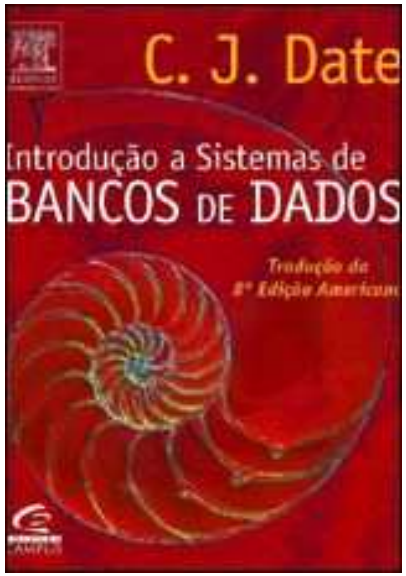
- **Professor Rodrigo Adur**
  - **Formação**
    - Bacharelado em Sistemas de Informações (FILC)
    - Especialista em Sistemas de Informações (NCE/UFRJ)
  - **Experiência Profissional**
    - Desenvolvimento de Sistemas
    - Analista de Sistemas do SERPRO

Estrutura do Módulo

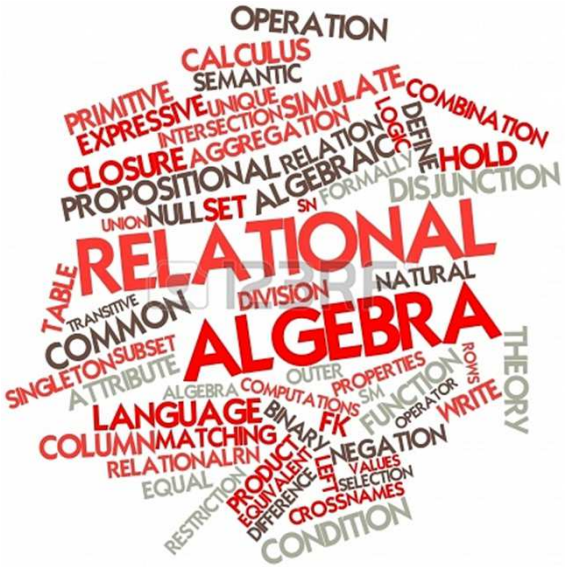
- **Módulo 2**
  - Álgebra relacional
    - Principais operadores e suas características
  - SQL (Structured Query Language)

Bibliografia

Sistemas de bancos de dados
Navathe
6ª Edição



Introdução a Sistemas de bancos de dados
Date
8ª Edição



Álgebra relacional

- **Visão geral**
  - O conjunto básico de operações para o modelo relacional é a álgebra relacional, que oferece um alicerce formal para as operações do modelo relacional.
  - Conjunto de operações desenvolvidos especificamente para banco de dados relacionais:
    - PROJEÇÃO
    - SELEÇÃO
    - JUNÇÃO
    - ...
  - Conjunto de operações da teoria de conjunto da matemática:
    - UNIÃO
    - INTERSECÇÃO
    - DIFERENÇA
    - PRODUTO CARTESIANO
  - **OBS: As operações da álgebra produzem novas relações.**

- **PROJEÇÃO (π pi)**
  - Se tivermos interessados apenas em certos atributos de uma relação, usamos a operação PROJEÇÃO para projetar a relação apenas por esses atributos.
  - O resultado da operação PROJEÇÃO pode ser visualizado como uma partição vertical da relação em duas relações: uma contém as colunas necessárias (resultado da operação), e a outra contém as colunas descartadas.
  - Operador unário.
  - **Sintaxe:** $\pi_{\texttt{<lista de atributos>}}(R)$

    Onde:
    - π - representa a PROJEÇÃO na álgebra.
    - `<lista de atributos>` - lista desejada de atributos de R.
    - `R` - expressão da álgebra cujo resultado seja uma relação.

- **PROJEÇÃO (π pi)**
  - A operação PROJEÇÃO remove qualquer tupla duplicada, de modo que o resultado dessa operação é um conjunto de tuplas distintas.
  - O número de tuplas em uma relação resultante de uma PROJEÇÃO é sempre menor ou igual ao número de tuplas em R.
  - Correspondência com SQL:

$$\pi_{\text{endereco, telefone}}(\text{FUNCIONARIO})$$

```
SELECT DISTINCT endereco, telefone
FROM FUNCIONARIO;
```

- **PROJEÇÃO (π pi)**
  - Exemplo:

FUNCIONARIO

nome	endereco	telefone	salario
Pedro	Rua XPTO	1234-1111	5000
João	Rua XYZ	1234-2222	2200
Maria	Rua XPTO	1234-1111	3000

$\pi_{\text{nome, telefone}}$ (FUNCIONARIO)

nome	telefone
Pedro	1234-1111
João	1234-2222
Maria	1234-1111

$\pi_{\text{endereco, telefone}}$ (FUNCIONARIO)

endereco	telefone
Rua XPTO	1234-1111
Rua XYZ	1234-2222

Questão 01

[2012 - Liquigás - Infraestrutura (60)]

Na Álgebra Relacional, o número de tuplas resultante de uma operação de projeção (projection) sobre uma relação R é sempre

(A) igual ou menor que o número de tuplas da relação R

(B) igual ao número de tuplas da relação R

(C) maior que o número de tuplas da relação R

(D) metade do número de tuplas da relação R

(E) o dobro do número de tuplas da relação R

- **SELEÇÃO (σ sigma)**
  - Pode-se considerar que essa operação seja um filtro que mantém apenas as tuplas que satisfazem uma condição qualificadora.
  - A operação de SELEÇÃO também pode ser visualizada como uma partição horizontal da relação em dois conjuntos de tuplas - aquelas que satisfazem a condição (são selecionadas), e aquelas que não satisfazem a condição (são descartadas).
  - Operador unário.
  - **Sintaxe:** $\sigma_{\texttt{<condição de seleção>}}(R)$

    Onde:
    - σ - representa a SELEÇÃO na álgebra.
    - **<condição de seleção>** - expressão booleana.
    - **R** - expressão da álgebra cujo resultado seja uma relação.

- **SELEÇÃO (σ sigma)**
  - O grau da relação resultante de uma operação SELEÇÃO é o mesmo que o da relação R.
  - O número de tuplas da relação resultante é sempre menor ou igual ao número de tuplas em R.
  - Correspondência com SQL:

$$\sigma_{\text{salario} > 2900}(\text{FUNCIONARIO})$$

```
SELECT *
FROM FUNCIONARIO
WHERE salario > 2900;
```

- **SELEÇÃO (σ sigma)**
  - Exemplo:

FUNCIONARIO

nome	endereco	telefone	salario
Pedro	Rua XPTO	1234-1111	5000
João	Rua XYZ	1234-2222	2200
Maria	Rua XPTO	1234-1111	3000

$$\sigma_{\texttt{salario > 2900}}(\texttt{FUNCIONARIO})$$

nome	endereco	telefone	salario
Pedro	Rua XPTO	1234-1111	5000
Maria	Rua XPTO	1234-1111	3000

Questão 02

ITnerante

[2012 - Petrobras- Processos de Negócio (62)]

Considere a relação chamada Cidade a seguir:

nome	estado
Água Branca	Alagoas
Água Branca	Minas Gerais
Água Branca	Piauí
Bom Jesus	Piauí
Bom Jesus	Rio Grande do Sul
Bom Jesus	Rio Grande do Norte
Cruzeiro do Sul	Acre
Cruzeiro do Sul	Rio Grande do Sul
Feira Nova	Sergipe

Seja a seguinte operação da álgebra relacional:

$$\pi_{nome}\ (\sigma_{estado>'R'}\ (Cidade))$$

Quantas tuplas e atributos terá a relação resultante após a execução dessa operação?

(A) 3 tuplas, cada uma com 1 atributo

(B) 3 tuplas, cada uma com 2 atributos

(C) 4 tuplas, cada uma com 1 atributo

(D) 4 tuplas, cada uma com 2 atributos

(E) 7 tuplas, cada uma com 1 atributo

Questão 03

ITnerante

[2012 - BNDES - Suporte (68)]

No contexto de banco de dados relacionais, dada a relação investimento e a relação Y resultante da operação X:

investimento

Tipo de fundo	Nome do fundo	Valor aplicado
Rendafixa	Aquarius	22500
Ações	Mercurio	27850
Multimercado	Complexo	10000
Rendafixa	Aquarius	22500
Ações	Mercurio	18600
Multimercado	Complexo	10000

Y

nome_fundo	valor_aplicado
Aquarius	22500
Mercurio	27850
Complexo	10000
Mercurio	18600

A operação X é:

(A) $\pi_{\text{nome\_fundo, valor\_aplicado}}$ (investimento)

(B) $\pi_{\text{nome\_fundo = valor\_aplicado}}$ (investimento)

(C) $\sigma_{\text{tipo de fundo e valor aplicado}}$ (investimento)

(D) $\sigma_{\text{investimento = nome do fundo e valor aplicado}}$

(E) $\pi_{\text{investimento (nome fundo, valor\_aplicado)}}$

- **União (U)**
  - O resultado dessa operação é uma relação que inclui todas as tuplas que fazem parte das relações participantes.
  - Operação binária.
  - Elimina qualquer tupla duplicada.
  - É necessário que as relações envolvidas tenham o mesmo tipo de tupla, ou seja, tenham compatibilidade de tipo.
  - **Sintaxe: R U S**

União (U)

- Correspondência com SQL:

```
ALUNO U PROFESSOR
```

```
SELECT * FROM ALUNO
UNION
SELECT * FROM PROFESSOR;
```

- Exemplo:

ALUNO

pri_nome	ult_nome
Pedro	Silva
João	Santos
Maria	Sá

PROFESSOR

pri_nome	ult_nome
José	Lima
João	Santos

ALUNO U PROFESSOR

pri_nome	ult_nome
Pedro	Silva
João	Santos
Maria	Sá
José	Lima

- **INTERSECÇÃO (∩)**
  - O resultado dessa operação é uma relação que inclui todas as tuplas que estão nas duas relações participantes.
  - Operação binária.
  - É necessário que as relações envolvidas tenham o mesmo tipo de tupla, ou seja, tenham compatibilidade de tipo.
  - **Sintaxe: R ∩ S**

- **INTERSECÇÃO (∩)**
  - Correspondência com SQL:

```
ALUNO ∩ PROFESSOR
```

```
SELECT * FROM ALUNO
INTERSECT
SELECT * FROM PROFESSOR;
```

  - Exemplo:

ALUNO

pri_nome	ult_nome
Pedro	Silva
João	Santos
Maria	Sá

PROFESSOR

pri_nome	ult_nome
José	Lima
João	Santos

ALUNO ∩ PROFESSOR

pri_nome	ult_nome
João	Santos

- **DIFERENÇA (-)**
  - O resultado dessa operação é uma relação que inclui todas as tuplas que estão na primeira relação (relação que antecede o operador) menos as que constam na segunda relação.
  - Operação binária.
  - É necessário que as relações envolvidas tenham o mesmo tipo de tupla, ou seja, tenham compatibilidade de tipo.
  - **Sintaxe:** `R - S`

- **DIFERENÇA (-)**
  - Correspondência com SQL:

```
ALUNO - PROFESSOR
```

```
SELECT * FROM ALUNO
EXCEPT
SELECT * FROM PROFESSOR;
```

  - Exemplo:

ALUNO

pri_nome	ult_nome
Pedro	Silva
João	Santos
Maria	Sá

PROFESSOR

pri_nome	ult_nome
José	Lima
João	Santos

ALUNO - PROFESSOR

pri_nome	ult_nome
Pedro	Silva
Maria	Sá

Questão 04

[2012 - Petrobras - Infra (66)]

A Álgebra Relacional define várias operações. Algumas delas operam apenas uma relação (unárias), outras operam com duas relações (binárias). As operações project (projeção), union (união) e select (seleção) são, respectivamente, operações

(A) unária, unária, unária
(B) binária, unária, binária
(C) binária, binária, unária
(D) unária, binária, unária
(E) unária, binária, binária

Questão 05

ITnerante

[2012 - Petrobras - Infraestrutura (65)]

Considere o seguinte esquema:

```
Assiste (criança, canal de TV)
Programa (canal de TV, desenho)
Gosta (criança, desenho)
```

A tabela Assiste indica os canais de TV a que a criança assiste. A tabela Programa indica os desenhos que são apresentados por cada canal de TV. A tabela Gosta indica os desenhos dos quais a criança gosta.
Qual consulta expressa em álgebra relacional que identifica quais as crianças que gostam dos desenhos P ou Q?

(A) $\pi_{\text{criança}} (\sigma_{\text{desenho} = P} (\text{Gosta})) \cup \pi_{\text{criança}} (\sigma_{\text{desenho} = Q} (\text{Gosta}))$;

(B) $X \leftarrow \sigma_{\text{criança}} (\sigma_{\text{desenho} = P} (\text{Gosta}))$; $Y \leftarrow \sigma_{\text{criança}} (\sigma_{\text{desenho} = Q} (\text{Gosta}))$; $\pi_{\text{criança}} (X \cap Y)$

(C) $X \leftarrow {}_{\text{criança}} (\sigma_{\text{desenho} = P \cup Q} (\text{desenho}))$; $\pi_{\text{criança}} (X)$

(D) $X \leftarrow \sigma_{\text{desenho}} ({}_{\text{criança} = P} (\text{Gosta}))$; $Y \leftarrow \sigma_{\text{desenho}} (\pi_{\text{criança} = Q} (\text{Gosta}))$; $\pi_{\text{criança}} (X \cup Y)$

(E) $X \leftarrow \pi_{\text{criança}} (\sigma_{\text{Gosta}}(\text{desenho} = P))$; $Y \leftarrow \pi_{\text{criança}} (\sigma_{\text{Gosta}}(\text{desenho} = Q))$; $\pi_{\text{criança}} (X \cap Y)$

PRODUTO CARTESIANO (X)

- Essa operação de conjunto produz uma nova relação combinando cada tupla de uma relação com cada tupla da outra relação.
- Operação binária.
- A relação resultante dessa operação possui grau correspondente a soma do grau das relações participantes.
- A relação resultante tem uma tupla para cada combinação de tuplas das relações participantes. Logo, a relação resultante terá número de tuplas equivalente ao produto das quantidades de tuplas das relações participantes.
- **Sintaxe:** `R X S`

PRODUTO CARTESIANO (X)

- Correspondência com SQL:

```
ALUNO X DISCIPLINA
```

```
SELECT * FROM ALUNO
CROSS JOIN
SELECT * FROM DISCIPLINA;
```

- Exemplo:

ALUNO

pri_nome	ult_nome
Pedro	Silva
João	Santos

DISCIPLINA

codigo	descricao
001	BD
002	RLM

ALUNO x DISCIPLINA

pri_nome	ult_nome	codigo	descricao
Pedro	Silva	001	BD
Pedro	Silva	002	RLM
João	Santos	001	BD
João	Santos	002	RLM

- **JUNÇÃO (⋈)**
  - A operação junção é usada para combinar tuplas relacionadas de duas relações em uma única tupla maior. Essa consulta nos permite processar relacionamentos entre as relações.
  - Operação binária.
  - A restrição de integridade referencial desempenha um papel importante para que haja tuplas combinando na relação referenciada.
  - **Sintaxe:** `R ⋈ <condição de junção> S`

    Onde:
    - R - expressão da álgebra cujo resultado seja uma relação.
    - ⋈ - representa a JUNÇÃO na álgebra.
    - <condição de junção> - expressão booleana.
    - S - expressão da álgebra cujo resultado seja uma relação.

- **JUNÇÃO (⋈)**
  - Correspondência com SQL:

```
FUNCIONARIO ⋈ mat=mat_func DEPENDENTE
```

```
SELECT *
FROM FUNCIONARIO INNER JOIN DEPENDENTE
ON mat=mat_func;
```

- **JUNÇÃO ($\bowtie$)**
  - Exemplo:

FUNCIONARIO

mat	nomef	salario
11111	Pedro	1000
22222	João	2000
33333	Maria	3000

DEPENDENTE

cod	nomed	mat_func
1	Paula	11111
2	Pelé	11111
3	José	22222

FUNCIONARIO $\bowtie_{mat=mat\_func}$ **DEPENDENTE**

mat	nomef	salario	cod	nomed	mat_func
11111	Pedro	1000	1	Paula	11111
11111	Pedro	1000	2	Pelé	11111
22222	João	2000	3	José	22222

Questão 06

ITnerante

[2013 - Liquigás - Analista de Sistemas (43)]

Seja a seguinte sequência de operações da Álgebra Relacional:

$$\pi_{A1,A2} (\sigma_{A1=5} (A \bowtie_{A1=B3} B))$$

Considerando-se essa sequência da esquerda para a direita, que operações foram empregadas?

(A) Junção, projeção e seleção
(B) Junção, seleção e projeção
(C) Projeção, junção e seleção
(D) Projeção, seleção e junção
(E) Seleção, projeção e junção

Questão 07

ITnerante

[2012 - Liquigás - Administração de Banco de Dados (21)]

Nas relações a seguir, os atributos pertencentes às chaves primárias aparecem sublinhados.

PESSOA(<u>CPF</u>, Nome, Idade)
SALA(<u>Numero</u>, Telefone)
ALOCA(<u>CPF, Numero</u>)

Considerando que a primeira relação descreve pessoas, a segunda descreve salas de uma empresa e o telefone da sala, e a terceira descreve em que sala a pessoa fica durante o expediente, qual a expressão em Álgebra Relacional que gera uma relação com duas colunas, sendo a primeira o nome da pessoa e a segunda o telefone de sua sala?

(A) $\sigma_{Nome,Telefone}$ ((PESSOA ⋈ ALOCA) ⋈ SALA)

(B) $\sigma_{Nome,Telefone}$ ((PESSOA × ALOCA) × SALA)

(C) $\sigma_{(PESSOA \bowtie ALOCA) \bowtie SALA}$ (NOME, TELEFONE)

(D) $\pi_{Nome,Telefone}$ ((PESSOA ⋈ ALOCA) ⋈ SALA)

(E) $\pi_{(PESSOA \times ALOCA)}$ × SALA (NOME, TELEFONE)

Questão 08

[2012 - BNDES - Desenvolvimento de Sistemas (57)]

T

T1	T2	T3
10	5	ab
15	8	xy
20	17	ab
30	5	xy

V

V1	V2	V3
5	x	15
6	y	20
7	w	10
8	z	20

Seja a seguinte sequência de operações da Álgebra Relacional:

$$\pi_{T1,T3} (\sigma_{V1>5} (T \bowtie_{T1=V3} V)) - \pi_{T1,T3} (\sigma_{T2<8} (T))$$

Qual é o resultado dessa sequência de operações?

(A)

T1	T3
10	ab
20	ab
30	xy

(B)

T1	T3
30	xy

(C)

T1	T3
20	ab
20	ab

(D)

T1	T3
20	ab

(E)

T1	T3
10	ab

Questão 09

[2012 - BNDES - Desenvolvimento de Sistemas (58)]

T

T1	T2	T3
10	5	ab
15	8	xy
20	17	ab
30	5	xy

V

V1	V2	V3
5	x	15
6	y	20
7	w	10
8	z	20

A relação R a seguir foi obtida pela aplicação de uma sequência de operações da Álgebra Relacional sobre as relações T e V.

R

R1	R2
20	6
20	8

(A) $R(R1,R2) \leftarrow \pi_{T1,T2}(T) - \pi_{T1,T2}(T \bowtie_{T1>V3} V)$

(B) $P(R1,R2) \leftarrow \pi_{T1,T2}(T) \cup \pi_{V3,V1}(V)$
$R \leftarrow \sigma_{R1=20}(P)$

(C) $P(R1,R2) \leftarrow \pi_{T1,V1}(T \times V) \cap \pi_{V3,V1}(V)$
$R \leftarrow \sigma_{R1>15}(P)$

(D) $R(R1,R2) \leftarrow \pi_{T1,V1}((\sigma_{T1>15}(T)) \bowtie_{T2>V1} (\sigma_{V2='x' \text{ ou } V2='y'}(V)))$

(E) $P(R1,R2) \leftarrow \pi_{T1,T2}(T) - \pi_{V3,V1}(V)$
$R \leftarrow \sigma_{R2=17}(P)$

- **DIVISÃO (÷)**
  - A operação DIVISÃO é útil para um tipo especial de consulta que as vezes ocorre nas aplicações de banco de dados. Um exemplo é recuperar a matricula dos alunos que estão inscritos em todas as disciplinas.
  - Operação binária.
  - A partir da divisão de duas relações R(Z) e S(X), deriva-se uma terceira relação T(Y).
    - É necessário que X seja um subconjunto de Z.
    - T(Y) é formada pelo conjunto dos atributos de R que não são atributos de S (Y = Z - X).
    - O estado da relação resultante é formado por valores oriundos de tuplas da primeira relação que combinem com todas as tuplas da segunda relação.
  - **Sintaxe: R ÷ S**

- **DIVISÃO (÷)**
  - Exemplo:

INSCRICAO

matricula	codigo
11111	D1
22222	D1
11111	D2
11111	D3
22222	D2
33333	D3

DISCIPLINA

codigo
D1
D2
D3

Quais as matriculas dos alunos que estão inscritos em <u>todas</u> as disciplinas?

INSCRICAO ÷ DISCIPLINA

matricula
11111

Questão 10

[2010 - Petrobras - Processos de Negócio (1)]

O cabeçalho em cada uma dessas instâncias de relações apresenta os respectivos nomes das colunas.

R1

sno	pno
1	1
1	2
1	3
1	4
2	1
2	2
3	2
4	2
4	4

R2

pno
1
2

No contexto da Álgebra Relacional, o resultado da divisão relacional de R1 por R2 é:

(A)

pno
1
2
4

(B)

pno
1
2
3
4

(C)

pno
1
4

(D)

pno
1
2

(E)

pno
3
4

Questão 11

ITnerante

[2011 - BNDES - Desenvolvimento de Sistemas (48)]

Considere um banco de dados sobre clientes de uma empresa que realiza vendas pela Internet. CLIENTES, VENDAS e CIDADES são algumas das tabelas desse banco de dados. A estrutura dessas tabelas está representada a seguir, onde os itens sublinhados representam colunas participantes da chave primária, e os itens em negrito representam colunas que participam em chaves estrangeiras.

```
VENDAS (IDVenda, IDCliente, Data, Valor_Total)
CLIENTES(ID, Nome, DataNascimento, Endereco, Complemento, IDCidade)
CIDADES (IDCid, Nome, UF)
```

A expressão da álgebra relacional que atribui a RES a sigla da UF dos clientes que realizaram compras com Valor_Total superior a 5.000 é:

(A) $RES \leftarrow \pi_{UF} (\sigma_{Valor\_Total > 5000 \text{ E } IDCid = IDCidade \text{ E } IDCliente = ID} (CIDADES))$

(B) $RES \leftarrow \pi_{UF} (\sigma_{Valor\_Total > 5000 \text{ E } IDCid = IDCidade} (CIDADES \cup (VENDAS \bowtie_{IDCliente = ID} CLIENTES)))$

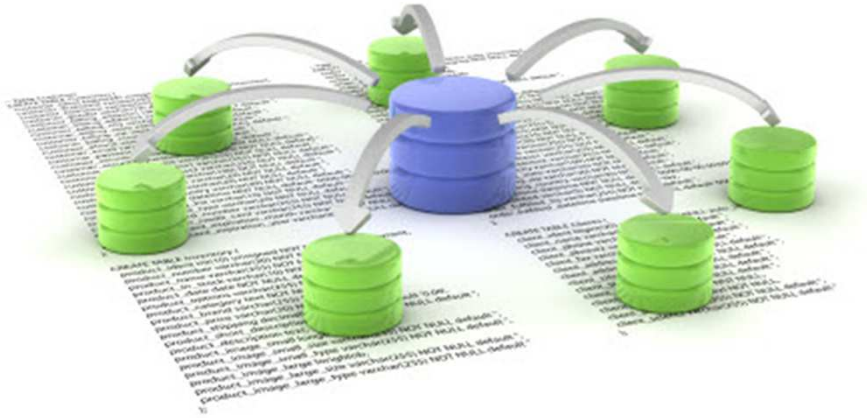
(C) $RES \leftarrow \pi_{UF} (\sigma_{Valor\_Total > 5000 \text{ E } IDCid = IDCidade} (VENDAS \times (CLIENTES \times CIDADES)))$

(D) $RES \leftarrow \pi_{UF} (\sigma_{Valor\_Total > 5000 \text{ E } IDCid = IDCidade} (CIDADES \times (VENDAS \bowtie_{IDCliente = ID} CLIENTES)))$

(E) $RES \leftarrow \pi_{UF} (\sigma_{Valor\_Total > 5000 \text{ E } IDCid = IDCidade} ((CLIENTES \div VENDAS) \times CIDADES))$

Gabarito

1 – A
2 – A
3 – A
4 – D
5 – A
6 – D
7 – D
8 – D
9 – C
10 – D
11 – D



SQL

- **SQL (Structured Query Language)**
  - A linguagem SQL pode ser considerada um dos principais motivos para o sucesso dos bancos de dados comerciais.
  - Tornou-se um padrão para esse tipo de banco de dados, deixando os usuário menos preocupados com a migração de suas aplicações.
  - A linguagem SQL oferece uma interface de linguagem declarativa de alto nível, de modo que o usuário apenas especifica qual deve ser o resultado, deixando as decisões de como executar as consultas para o SGBD.
  - SQL é uma linguagem de banco de dados abrangente tendo instruções para definição, consultas e atualização dos dados.

Criação de tabelas

- O comando CREATE TABLE é usado para especificar uma nova relação, dando-lhe um nome e especificando seus atributos e restrições iniciais.
- Os atributos são especificados primeiro, e cada um deles recebe um nome, um tipo de dado para especificar seu domínio de valores e quaisquer restrições de atributo.

- **Sintaxe:**

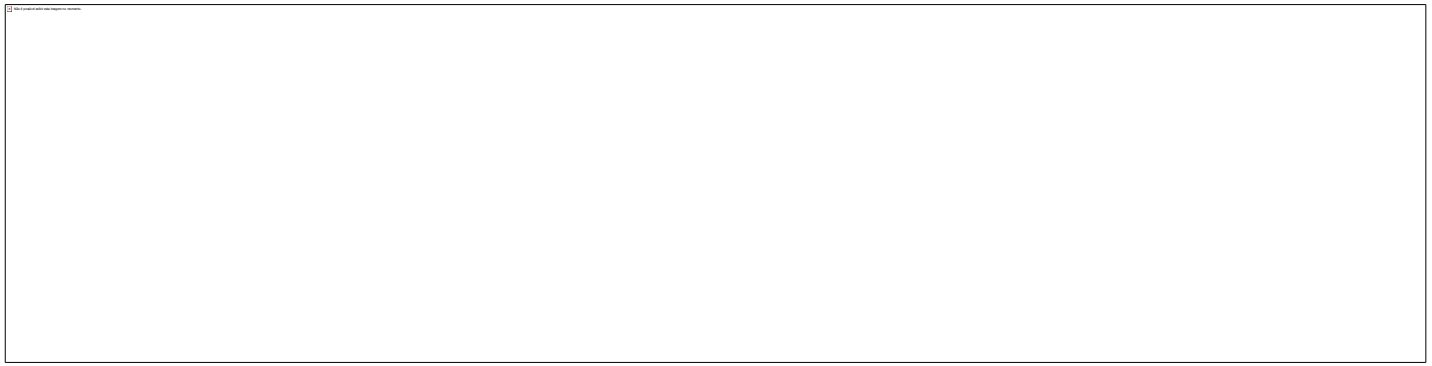
```
CREATE TABLE table_name (
    column_name1 data_type,
    column_name2 data_type,
    column_name3 data_type,
    ....
);
```

- **Exemplo:**

```
CREATE TABLE FUNCIONARIO (
    MAT      INT,
    NOME     VARCHAR(15),
    ENDERECO VARCHAR(30),
    TELEFONE VARCHAR(10),
    SALARIO  DECIMAL(10,2)
);
```

Especificação de restrições

- A seguir temos algumas restrições básicas que podem ser especificadas em SQL:



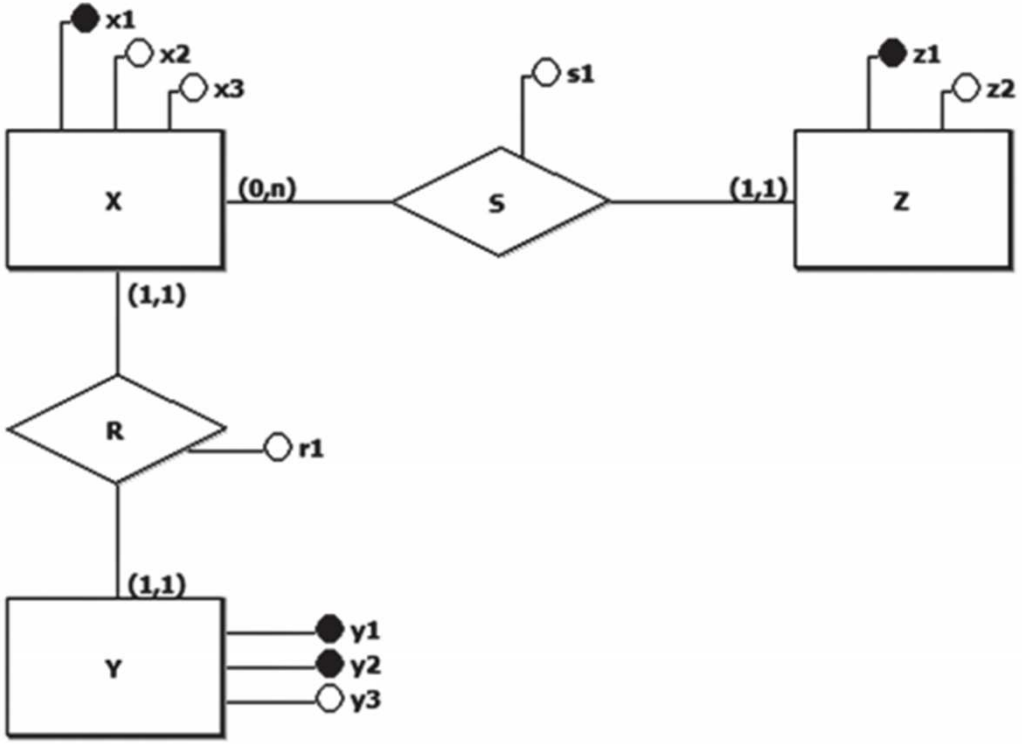
- **Exemplo:**

```
CREATE TABLE FUNCIONARIO (
    MAT      INT PRIMARY KEY,
    NOME     VARCHAR(15) NOT NULL,
    ENDERECO VARCHAR(30),
    TELEFONE VARCHAR(10),
    SALARIO  DECIMAL(10,2) DEFAULT 3000,
    DEPTO    VARCHAR(10),
    CONSTRAINT FKDEPTO FOREIGN KEY (DEPTO) REFERENCE DEPARTAMENTO(COD)
);
```

Questão 12

[2012 - Petrobras - Engenharia de Software (34)]

Seja o modelo E-R em que todas as colunas foram definidas como INTEGER, pois os tipos de dados são irrelevantes para o problema.



Qual modelo lógico relacional preserva a semântica do modelo acima?

Questão 12

(A)
```
CREATE TABLE XY (
    Y1 INTEGER NOT NULL,
    Y2 INTEGER NOT NULL,
    Y3 INTEGER,
    X1 INTEGER NOT NULL,
    X2 INTEGER,
    X3 INTEGER,
    R1 INTEGER,
    S1 INTEGER,
    Z1 INTEGER NOT NULL,
    CONSTRAINT XY_PK PRIMARY KEY (Y1,Y2),
    CONSTRAINT XY_UK1 UNIQUE (X1),
    CONSTRAINT XY_FK1 FOREIGN KEY (Z1) REFERENCES Z (Z1))

CREATE TABLE Z (
    Z1 INTEGER NOT NULL,
    Z2 INTEGER,
    CONSTRAINT Z_PK PRIMARY KEY (Z1))
```

(B)
```
CREATE TABLE X (
    X1 INTEGER NOT NULL,
    X2 INTEGER,
    X3 INTEGER,
    CONSTRAINT X_PK PRIMARY KEY (x1))

    CREATE TABLE Y (
    Y1 INTEGER NOT NULL,
    Y2 INTEGER NOT NULL,
    Y3 INTEGER,
    X1 INTEGER NOT NULL,
    R1 INTEGER,
    CONSTRAINT Y_PK PRIMARY KEY (Y1,Y2),
    CONSTRAINT Y_UK1 UNIQUE (X1),
    CONSTRAINT Y_FK1 FOREIGN KEY (X1) REFERENCES X (X1))

    CREATE TABLE Z (
    Z1 INTEGER NOT NULL,
    Z2 INTEGER,
    CONSTRAINT Z_PK PRIMARY KEY (Z1))

    CREATE TABLE S (
    X1 INTEGER NOT NULL,
    Z1 INTEGER NOT NULL,
    S1 INTEGER,
    CONSTRAINT S_PK PRIMARY KEY (X1,Z1),
    CONSTRAINT S_FK1 FOREIGN KEY (X1) REFERENCES X (X1),
    CONSTRAINT S_FK2 FOREIGN KEY (Z1) REFERENCES Z (Z1))
```

Questão 12

(C)
```
CREATE TABLE X (
    X1 INTEGER NOT NULL,
    X2 INTEGER,
    X3 INTEGER,
    Z1 INTEGER,
    S1 INTEGER,
    CONSTRAINT X_PK PRIMARY KEY (x1),
    CONSTRAINT X_FK FOREIGN KEY (Z1) REFERENCES Z (Z1))

   CREATE TABLE Y (
    Y1 INTEGER NOT NULL,
    Y2 INTEGER NOT NULL,
    Y3 INTEGER,
    X1 INTEGER NOT NULL,
    R1 INTEGER,
    CONSTRAINT Y_PK PRIMARY KEY (Y1,Y2),
    CONSTRAINT Y_UK1 UNIQUE (X1),
    CONSTRAINT Y_FK1 FOREIGN KEY (X1) REFERENCES X (X1))

    CREATE TABLE Z (
    Z1 INTEGER NOT NULL,
    Z2 INTEGER,
    CONSTRAINT Z_PK PRIMARY KEY (Z1))
```

(D)
```
CREATE TABLE XY (
    Y1 INTEGER NOT NULL,
    Y2 INTEGER NOT NULL,
    Y3 INTEGER,
    X1 INTEGER NOT NULL,
    X2 INTEGER,
    X3 INTEGER,
    R1 INTEGER,
    S1 INTEGER,
    Z1 INTEGER NOT NULL,
    CONSTRAINT XY_PK PRIMARY KEY (Y1,Y2,X1),
    CONSTRAINT XY_FK1 FOREIGN KEY (Z1) REFERENCES Z (Z1))

    CREATE TABLE Z (
    Z1 INTEGER NOT NULL,
    Z2 INTEGER,
    CONSTRAINT Z_PK PRIMARY KEY (Z1))
```

Questão 12

(E) CREATE TABLE X (
X1 INTEGER NOT NULL,
X2 INTEGER,
X3 INTEGER,
CONSTRAINT X_PK PRIMARY KEY (x1))
CREATE TABLE Y (
Y1 INTEGER NOT NULL,
Y2 INTEGER NOT NULL,
Y3 INTEGER,
X1 INTEGER,
R1 INTEGER,
CONSTRAINT Y_PK PRIMARY KEY (Y1,Y2),
CONSTRAINT Y_UK1 UNIQUE (X1),
CONSTRAINT Y_FK1 FOREIGN KEY (X1) REFERENCES X (X1))

CREATE TABLE Z (
Z1 INTEGER NOT NULL,
Z2 INTEGER,
CONSTRAINT Z_PK PRIMARY KEY (Z1))

CREATE TABLE S (
X1 INTEGER NOT NULL,
Z1 INTEGER NOT NULL,
S1 INTEGER,
CONSTRAINT S_PK PRIMARY KEY (X1),
CONSTRAINT S_FK1 FOREIGN KEY (X1) REFERENCES X (X1),
CONSTRAINT S_FK2 FOREIGN KEY (Z1) REFERENCES Z (Z1))

Alteração de tabelas

- O comando ALTER TABLE é usado para alterar a definição de uma tabela da base.
- As tabelas da base podem sofrer as seguintes alterações: acrescentar ou remover uma coluna, alterar uma definição de coluna e acrescentar ou remover restrições da tabela.
- **Sintaxe:**

```
ALTER TABLE table_name ADD column_name datatype;
ALTER TABLE table_name DROP column_name;
```

- **Exemplo:**

```
ALTER TABLE FUNCIONARIO ADD SEXO CHAR;
ALTER TABLE FUNCIONARIO DROP TELEFONE;
```

Questão 13

[2013 - IBGE - Suporte Operacional (56)]

Em um banco de dados, a tabela Pessoa foi criada com a seguinte instrução:

```
CREATE TABLE Pessoa (
    PessoaID int,
    Nome varchar(255),
    Sobrenome varchar(255),
    Endereco varchar(255),
    Cidade varchar(255)
);
```

Que instrução SQL acrescenta um campo CEP do tipo varchar(9) a essa tabela?

(A) `ADD COLUMN CEP varchar(9) INTO TABLE`
(B) `ALTER TABLE Pessoa ADD CEP varchar(9)`
(C) `ALTER TABLE Pessoa INSERT COLUMN CEP varchar(9)`
(D) `ALTER TABLE Pessoa ALTER COLUMN CEP varchar(9)`
(E) `ALTER TABLE Pessoa MODIFY COLUMN ADD CEP varchar(9)`

Consultas em tabelas

- Sintaxe de uma consulta:

```
SELECT      <lista atributos e função>
FROM        <lista tabela>
[WHERE      <condição>]
[GROUP BY   <atributos(s) de agrupamento>]
[HAVING     <condição de grupo>]
[ORDER BY   <lista de atributos>];
```

Onde:

- **SELECT** – lista os atributos ou funções a serem recuperadas.
- **FROM** – especifica todas as tabelas necessárias na consulta.
- **WHERE** – especifica as condições para selecionar as tuplas dessas relações.
- **GROUP BY** – especifica atributos de agrupamento.
- **HAVING** – especifica uma condição sobre os grupos selecionados, em vez das tuplas individuais.
- **ORDER BY** – especifica uma ordem para exibir o resultado de uma consulta.

Funções de agregação em SQL

- As funções de agregação são usadas para resumir informações de várias tuplas em uma síntese de tupla única.
- Existem diversas funções de agregação:

COUNT	Retorna o número de tuplas ou valores conforme especificado em uma consulta.
SUM	Retorna a soma de um conjunto de valores numéricos.
MAX	Retorna o maior valor de um conjunto de valores numéricos.
MIN	Retorna o menor valor de um conjunto de valores numéricos.
AVG	Retorna a média de um conjunto de valores numéricos.

- Exemplo:

```
SELECT AVG(SALARIO) FROM FUNCIONARIO;
SELECT MAX(SALARIO) FROM FUNCIONARIO;
```

Exemplos de consultas

- **Exemplo 1:** Obter a matricula, o nome e o salário de todos os funcionários que ganham mais do que 3000. Ordenar o resultado por nome de forma crescente.

```
SELECT MAT, NOME, SALARIO
FROM FUNCIONARIO
WHERE SALARIO>3000
ORDER BY NOME;
```

- **Exemplo 2:** Obter, para cada departamento, a média salarial dos funcionários com mais de 65 anos. Somente exibir os departamentos com média superior a 5000.

```
SELECT DEPARTAMENTO, AVG(SALARIO)
FROM FUNCIONARIO
WHERE idade>65
GROUP BY DEPARTAMENTO
HAVING AVG(SALARIO)>5000;
```

Questão 14

ITnerante

[2012 - EPE - Tecnologia da Informação (35)]

Tabela R

id	nome	cidade
1234	João	Rio de Janeiro
1235	Luiz	Fortaleza
1236	Leonardo	Porto Alegre
1237	Lúcio	Manaus
1238	Maria	Cuiabá

Tabela S

id	cargo	salario
1234	Engenheiro	3200
1235	Engenheiro	1250
1236	Projetista	5000
1237	Arquiteto	2000

A consulta "listar todos os cargos em ordem alfabética e a respectiva média salarial de cada um deles" é representada, em SQL ANSI, por

(A) `select cargo, sum(salario)/count(*) from S order by cargo;`
(B) `select cargo, sum(salario)/count(*) from S sort by cargo;`
(C) `select cargo from S having avg(cargo) order by cargo;`
(D) `select cargo, avg(salario) from S group by cargo sort by cargo;`
(E) `select cargo, avg(salario) from S group by cargo order by cargo;`

Consultas aninhadas

- Algumas consultas precisam que os valores existentes no banco de dados sejam buscados e depois usados em uma condição de comparação. Essas consultas podem ser formuladas usando consultas aninhadas.
- Consultas aninhadas são blocos `SELECT-FROM-WHERE` completos dentro da cláusula `WHERE` de outra consulta.
- Observação para teste de igualdade:
  - Se uma consulta aninhada retornar um único atributo e uma única tupla, o resultado da consulta será um único valor. Nesse caso podemos fazer a comparação com o operador =.
  - De outra forma, retornando uma tabela, devemos comparar o resultado com o operador IN.

- **Consultas aninhadas**
  - **Exemplo 1:** Obter o nome e o telefone de todos os funcionário que trabalham no mesmo departamento que o funcionário cuja matricula seja 2213.

```
SELECT F1.NOME, F1.TEFEFONE
FROM FUNCIONARIO F1
WHERE F1.DEPTO=( SELECT F2.DEPTO
                 FROM FUNCIONARIO F2 WHERE
                 F2.MAT=2213);
```

  - **Exemplo 2:** Obter o nome e o telefone de todos os funcionário que trabalham em algum departamento da superintendência de desenvolvimento.

```
SELECT NOME, TEFEFONE
FROM FUNCIONARIO
WHERE DEPTO IN ( SELECT DEPTO
                 FROM DEPARTAMENTO
                 WHERE SUPER='Desenvolvimento');
```

Questão 15

ITnerante

[2012 - CHESF - Analista de Sistemas (24)]

Em um banco de dados de empregados, a tabela EMP contém os campos CPF, NOME, SALARIO e DEPTNO dos empregados, sendo o último uma chave estrangeira para o departamento.
Que consulta SQL fornece o CPF, NOME, SALARIO e DEPTNO de todos os funcionários que ganham mais do que qualquer empregado do departamento 5?

(A)
```
SELECT CPF, NOME, MAX(SALARIO), DEPTNO FROM EMP
WHERE SALARIO > MAX(SALARIO) AND DEPTNO = 5;
```

(B)
```
SELECT CPF, NOME, MAX(SALARIO), DEPTNO FROM EMP
WHERE SALARIO > (SELECT MAX (SALARIO) FROM EMP WHERE DEPTNO = 5);
```

(C)
```
SELECT CPF, NOME, SALARIO, DEPTNO FROM EMP
WHERE SALARIO > (SELECT MAX (SALARIO) FROM EMP WHERE DEPTNO = 5);
```

(D)
```
SELECT CPF, NOME, SALARIO, DEPTNO FROM EMP
WHERE SALARIO > (SELECT MAX (SALARIO) FROM EMP) AND DEPTNO = 5;
```

(E)
```
SELECT CPF, NOME, SALARIO, DEPTNO FROM EMP
WHERE SALARIO > MAX(SALARIO) AND DEPTNO = 5;
```

Questão 16

ITnerante

[2013 - IBGE - Desenvolvimento de Sistemas (32)]

Considere o seguinte código SQL:

```
CREATE TABLE EMPREG (ID INTEGER PRIMARY KEY, NOME CHAR(20), SOBRENOME CHAR(60),
                     SALARIO REAL);

INSERT INTO EMPREG VALUES (44, 'William', 'Simpson', 6387.01);
INSERT INTO EMPREG VALUES (11, 'Fulano', 'Brasil', 3045.78);
INSERT INTO EMPREG VALUES (22, 'Beltrano', 'da Silva', 4046.79);
INSERT INTO EMPREG VALUES (33, 'Carlos', 'da Silva', 13040.78);

CREATE TABLE COMISSAO ( ID INTEGER REFERENCES EMPREG(ID), MES INTEGER CHECK (MES BETWEEN
                        1 AND 12), VALOR_COMISS REAL, PRIMARY KEY (ID, MES));

INSERT INTO COMISSAO VALUES (22,1,1001.67);
INSERT INTO COMISSAO VALUES (22,6,1001.67);
INSERT INTO COMISSAO VALUES (44,5,2338.67);
INSERT INTO COMISSAO VALUES (11,1,400.67);
INSERT INTO COMISSAO VALUES (33,9,2340.00);
INSERT INTO COMISSAO VALUES (44,12,2940.67);
```

O resultado da consulta será:

```
SELECT NOME FROM EMPREG WHERE 2340.00 < (SELECT AVG(VALOR_COMISS)
FROM COMISSAO WHERE EMPREG.ID = COMISSAO.ID);
```

(A) William (C) da Silva (E) 67

(B) Fulano (D) Carlos

Operadores de comparação

Há diversos operadores de comparação disponíveis, a baixo suas características:

OPERADOR	DESCRIÇÃO
BETWEEN	Verifica se um atributo esta compreendido em um intervalo (inclusive). `SELECT NOME, IDADE FROM FUNCIONARIO WHERE IDADE BETWEEN 18 AND 65;`

IN	Verifica se o atributo é igual a um valor contido na lista.
	`SELECT F1.NOME, F1.IDADE , F1.TELEFONE FROM FUNCIONARIO F1`
	`WHERE F1.IDADE IN (SELECT MAX(F2.IDADE) FROM FUNCIONARIO F2);`

ANY (SOME)	Verifica se um dos valores satisfaz uma condição especificada.
	`SELECT F1.NOME, F1.SALARIO FROM FUNCIONARIO F1`
	`WHERE F1.SALARIO > ANY (SELECT F2.SALARIO`
	`FROM FUNCIONARIO F2 WHERE F2.UF='SP');`

ALL	Verifica se todos os valores satisfazem uma condição especificada.
	`SELECT F1.NOME, F1.IDADE FROM FUNCIONARIO F1`
	`WHERE F1.IDADE > ALL (SELECT F2.IDADE`
	`FROM FUNCIONARIO F2 WHERE F2.UF='RJ');`

Questão 17

[2010 - Petrobras - Processos de Negócio (2)]

R1

sno	pno
1	1
1	2
1	3
1	4
2	1
2	2
3	2
4	2
4	4

R2

pno
1
2

Considere a expressão em SQL a seguir.

```
SELECT R1.sno
FROM R1
WHERE R1.pno >= ALL (SELECT R2.pno FROM R2)
```

O resultado dessa consulta é:

(A)

sno
1
2
2
2
4
4

(B)

sno
1
1
1
2
2
3
4

(C)

sno
1
2
3
4
4
4

(D)

sno
1
2
3
3
4
4

(E)

sno
1
1
1
2
3
4
4

Questão 18

[2012 - BNDES - Suporte (67)]

O modelo relacional a seguir representa um banco de dados simplificado de uma empresa de comércio. As chaves estão sublinhadas.

```
CLIENTE(NomeC, EnderecoC)
PRODUTO(NomeP)
FORNECEDOR(NomeF)
PRODUZ(NomeF,NomeP,Preco)
PEDIDO(NomeC,NomeF,NomeP,Quantidade)
```

Se o dono da empresa deseja saber quais clientes nunca pediram um produto do fornecedor cujo nome é "Barateira", que consulta SQL deve fazer?

(A)
```
SELECT * FROM CLIENTE WHERE CLIENTE.
NOMEC IN (SELECT NOMEC FROM PEDIDO WHERE
NOMEF="Barateira")
```

(B)
```
SELECT * FROM CLIENTE WHERE CLIENTE.
NOMEC NOT IN (SELECT NOMEC FROM PEDIDO
WHERE NOMEF="Barateira")
```

(C)
```
SELECT * FROM CLIENTE WHERE CLIENTE.
NOMEC=PEDIDO.NOMEC AND CLIENTE.NOMEC
NOT IN (SELECT NOMEC FROM PEDIDO WHERE
NOMEF="Barateira")
```

(D)
```
SELECT * FROM CLIENTE,PEDIDO WHERE
CLIENTE.NOMEC=PEDIDO.NOMEC AND CLIENTE.
NOMEC IN (SELECT NOMEC FROM PEDIDO WHERE
NOMEF="Barateira")
```

(E)
```
SELECT * FROM CLIENTE,PEDIDO WHERE CLIENTE.
NOMEC=PEDIDO.NOMEC AND NOMEF<>"Barateira"
```

Questões 19

As tabelas a seguir pertencem ao esquema de um banco de dados de atletas de salto em distância.

```
CREATE TABLE ATLETA (
        COD NUMBER(5) NOT NULL,
        NOME VARCHAR2(50) NOT NULL,
        NUM_INSCR NUMBER(7),
        TELEFONE VARCHAR2(8) NOT NULL,
        CONSTRAINT ATLETA_PK PRIMARY KEY (COD)
)

CREATE TABLE PROVA (
        COD_ATL NUMBER(5) NOT NULL,
        COD_PROVA NUMBER(5) NOT NULL,
        MARCA NUMBER(5,2),
        CONSTRAINT PROVA_PK PRIMARY KEY (COD_ATL,COD_PROVA),
        CONSTRAINT PROVA_FK FOREIGN KEY (COD_ATL) REFERENCES
        ATLETA(COD)
)
```

Observação:

• A coluna MARCA, localizada na tabela PROVA, contém a distância, em metros, saltada por um atleta em uma prova.

Questão 19

[2013 - Liquigás - Analista de Sistemas (54)]

Seja **u** a média das marcas de todos os saltos realizados em uma prova.
Qual consulta permite exibir o código da prova e a média dos saltos (u) relativos às provas em que 7,55 ≤ **u** ≤ 7,60?

(A)
```
SELECT COD_PROVA, AVG(MARCA)
    FROM PROVA
    GROUP BY COD_PROVA
    HAVING AVG(MARCA) BETWEEN 7.55 AND 7.60
```

(B)
```
SELECT COD_PROVA, AVG(MARCA)
    FROM PROVA
    GROUP BY COD_PROVA, COD_ATL
    HAVING AVG(MARCA) BETWEEN 7.55 AND 7.60
```

(C)
```
SELECT COD_PROVA, AVG(MARCA)
    FROM PROVA
    GROUP BY COD_PROVA
    HAVING AVG(MARCA)>= 7.55 AND <= 7.60
```

(D)
```
SELECT COD_PROVA, AVG(MARCA)
    FROM PROVA
    WHERE AVG(MARCA)>= 7.55 AND AVG(MARCA)<= 7.60
```

(E)
```
SELECT COD_PROVA, AVG(MARCA)
    FROM PROVA
    GROUP BY COD_PROVA, COD_ATL
    HAVING AVG(MARCA)>= 7.55 AND AVG(MARCA)<= 7.60
```

- **Tabelas de junção**
  - A tabela de junção permite ao usuário especificar uma tabela resultante que conecta duas tabelas.
  - Os atributos da tabela de junção são formados por todos os atributos das tabelas participantes, começando com os atributos da primeira tabela especificada, seguido pelos atributos da segunda tabela.

- **Junção Interna**
  - Fornece uma tabela resultante que conecta duas tabelas retornando apenas as linhas que possuam correspondência nas mesmas (satisfaz a condição da junção).
  - Tem as mesma características da operação junção estudada na álgebra relacional.

Junção Interna

- Exemplo:

```
SELECT *
FROM FUNCIONARIO INNER JOIN DEPARTAMENTO
ON DEPARTAMENTO=DEPTO;

SELECT *
FROM FUNCIONARIO, DEPARTAMENTO
WHERE DEPARTAMENTO=DEPTO;
```

- Correspondência com Álgebra Relacional:

$$\text{FUNCIONARIO} \bowtie_{\text{DEPARTAMENTO=DEPTO}} \text{DEPARTAMENTO}$$

Questão 20

As tabelas a seguir pertencem ao esquema de um banco de dados de atletas de salto em distância.

```
CREATE TABLE ATLETA (
        COD NUMBER(5) NOT NULL,
        NOME VARCHAR2(50) NOT NULL,
        NUM_INSCR NUMBER(7),
        TELEFONE VARCHAR2(8) NOT NULL,
        CONSTRAINT ATLETA_PK PRIMARY KEY (COD)
)

CREATE TABLE PROVA (
        COD_ATL NUMBER(5) NOT NULL,
        COD_PROVA NUMBER(5) NOT NULL,
        MARCA NUMBER(5,2),
        CONSTRAINT PROVA_PK PRIMARY KEY (COD_ATL,COD_PROVA),
        CONSTRAINT PROVA_FK FOREIGN KEY (COD_ATL) REFERENCES
        ATLETA(COD)
)
```

Observação:

• A coluna MARCA, localizada na tabela PROVA, contém a distância, em metros, saltada por um atleta em uma prova.

Questão 20

ITnerante

[2013 - Liquigás - Analista de Sistemas (52)]

O técnico da equipe de saltos solicitou que fosse elaborada uma consulta SQL que realize o seguinte:

• Exibir, para cada atleta, o nome e a quantidade de saltos cujas marcas foram abaixo de 7,5 m.

• A consulta deve exibir uma linha para cada atleta.

• Devem ser exibidos apenas os nomes dos atletas que realizaram pelo menos um salto abaixo de 7,5 m.

(A)
```
SELECT A.NOME, COUNT(*)
  FROM ATLETA A,PROVA P
  WHERE A.COD=P.COD_ATL AND P.MARCA < 7.5
  GROUP BY A.NOME
```

(B)
```
SELECT A.COD, COUNT(*)
  FROM ATLETA A,PROVA P
  WHERE A.COD=P.COD_ATL AND P.MARCA < 7.5
  GROUP BY A.COD,P.COD_PROVA
```

(C)
```
SELECT A.COD, COUNT(*)
  FROM ATLETA A,PROVA P
  WHERE A.COD=P.COD_ATL AND P.MARCA < 7.5
  GROUP BY A.COD
```

(D)
```
SELECT A.NOME, COUNT(*)
  FROM ATLETA A,PROVA P
  WHERE A.COD=P.COD_ATL AND P.MARCA < 7.5
  GROUP BY A.COD
```

(E)
```
SELECT A.NOME, COUNT(*)
  FROM ATLETA A,PROVA P
  WHERE P.MARCA < 7.5
  GROUP BY A.NOME
```

Questão 21

ITnerante

As tabelas são utilizadas para descrever um banco de dados que armazena dados sobre linhas de ônibus, motoristas e viagens por eles realizadas.

```
CREATE TABLE MOTORISTA (
     MATRICULA NUMBER(7,0) NOT NULL,
     NOME VARCHAR2(50) NOT NULL,
     CPF NUMBER(11,0) NOT NULL,
     CNH VARCHAR2(15) NOT NULL,
     CONSTRAINT MOTORISTA_PK PRIMARY KEY (MATRICULA),
     CONSTRAINT MOTORISTA_UK1 UNIQUE (CPF),
     CONSTRAINT MOTORISTA_UK2 UNIQUE (CNH)
)

CREATE TABLE LINHA (
     NUMERO CHAR(5) NOT NULL,
     ORIGEM VARCHAR2(50) NOT NULL,
     DESTINO VARCHAR2(50) NOT NULL,
     CONSTRAINT LINHA_PK PRIMARY KEY (NUMERO)
)

CREATE TABLE VIAGEM (
     MAT_MOT NUMBER (7,0) NOT NULL,
     NUM_LINHA CHAR(5) NOT NULL,
     INICIO DATE NOT NULL,
     FINAL DATE,
     CONSTRAINT VIAGEM_PK PRIMARY KEY (MAT_MOT,NUM_LINHA,INICIO),
     CONSTRAINT VIAGEM_FK1 FOREIGN KEY (MAT_MOT) REFERENCES MOTORISTA (MATRICULA),
     CONSTRAINT VIAGEM_FK2 FOREIGN KEY (NUM_LINHA) REFERENCES LINHA (NUMERO)
)
```

Questão 21

[2012 - Petrobras - Engenharia de Software (31)]

Considerando a possibilidade de que dois ou mais pares (ORIGEM,DESTINO) tenham o mesmo número de viagens, qual consulta permite exibir o par que possui o maior número de viagens (concluídas ou não) registradas no banco de dados?

(A)
```
SELECT L.ORIGEM,L.DESTINO
  FROM VIAGEM V,LINHA L
  WHERE V.NUM_LINHA=L.NUMERO
  GROUP BY L.ORIGEM,L.DESTINO
  HAVING COUNT(*)=(SELECT MAX(COUNT(*))
  FROM VIAGEM V,LINHA L
  WHERE V.NUM_LINHA=L.NUMERO
  GROUP BY L.ORIGEM,L.DESTINO)
```

(B)
```
SELECT L.ORIGEM,L.DESTINO
  FROM VIAGEM V,LINHA L
  WHERE V.NUM_LINHA=L.NUMERO
  GROUP BY V.NUM_LINHA
  HAVING COUNT(*)=(SELECT MAX(COUNT(NUM_LINHA))
  FROM VIAGEM V,LINHA L
  HERE V.NUM_LINHA=L.NUMERO
  GROUP BY L.ORIGEM,L.DESTINO)
```

(C)
```
SELECT L.ORIGEM,L.DESTINO
  FROM VIAGEM V,LINHA L
  WHERE V.NUM_LINHA=L.NUMERO
  GROUP BY L.ORIGEM,L.DESTINO
  HAVING COUNT(*)=(SELECT MAX(COUNT(DISTINCT NUM_LINHA))
  FROM VIAGEM V,LINHA L
  WHERE V.NUM_LINHA=L.NUMERO
  GROUP BY L.ORIGEM,L.DESTINO)
```

(D)
```
SELECT L.ORIGEM,L.DESTINO
  FROM VIAGEM V,LINHA L
  WHERE V.NUM_LINHA=L.NUMERO
  GROUP BY L.ORIGEM,L.DESTINO
  ORDER BY COUNT(*) DESC
```

(E)
```
SELECT L.ORIGEM,L.DESTINO
  FROM VIAGEM V,LINHA L
  WHERE V.NUM_LINHA=L.NUMERO AND COUNT(*)=(SELECT MAX(COUNT(*))
        FROM VIAGEM V,LINHA L
        WHERE V.NUM_LINHA=L.NUMERO
        GROUP BY L.ORIGEM,L.DESTINO)
  GROUP BY L.ORIGEM,L.DESTINO
```

Inserções em tabelas

- O comando INSERT, em sua forma mais simples, é usado para acrescentar uma linha em uma tabela.
- Sintaxe:

```
INSERT INTO table_name
VALUES (value1,value2,value3,...);
```

- De acordo com a sintaxe acima, todos os valores devem ser listados na mesma ordem em que os atributos correspondentes foram declarados no comando CREATE TABLE.
- Exemplo:

```
INSERT INTO FUNCIONARIO
VALUES (001, 'Gabriel', 'rua XPTO', NULL, 3500, 10);
```

Inserções em tabelas

- Uma forma alternativa de realizar essa operação é informar explicitamente o nome dos atributos correspondentes aos valores. Sintaxe:

```
INSERT INTO table_name (attr1, attr2, attr3,…)
VALUES (value1,value2,value3,...);
```

- Exemplo:

```
INSERT INTO FUNCIONARIO
(MAT, NOME, DEPTO)
VALUES (002, 'Pedro', 10);
```

Inserções em tabelas

- Uma variação do comando INSERT inclui várias linhas em uma tabela a partir do resultado de uma consulta.
- Sintaxe:

```
INSERT INTO table_name
SELECT-FROM-WHERE
```

- Exemplo:

```
INSERT INTO FUNCIONARIO_AUMENTO_TEMP
SELECT * FROM FUNCIONARIO WHERE SALARIO<3200;
```

Questão 22

As tabelas a seguir pertencem ao esquema de um banco de dados de atletas de salto em distância.

```
CREATE TABLE ATLETA (
     COD NUMBER(5) NOT NULL,
     NOME VARCHAR2(50) NOT NULL,
     NUM_INSCR NUMBER(7),
     TELEFONE VARCHAR2(8) NOT NULL,
     CONSTRAINT ATLETA_PK PRIMARY KEY (COD)
)

CREATE TABLE PROVA (
     COD_ATL NUMBER(5) NOT NULL,
     COD_PROVA NUMBER(5) NOT NULL,
     MARCA NUMBER(5,2),
     CONSTRAINT PROVA_PK PRIMARY KEY (COD_ATL,COD_PROVA),
     CONSTRAINT PROVA_FK FOREIGN KEY (COD_ATL) REFERENCES
     ATLETA(COD)
)
```

Observação:

• A coluna MARCA, localizada na tabela PROVA, contém a distância, em metros, saltada por um atleta em uma prova.

Questão 22

[2013 - Liquigás - Analista de Sistemas (51)]

Qual comando SQL irá inserir corretamente uma nova linha na tabela de atletas?

(A)
```
INSERT INTO ATLETA
VALUES(12345,'JOAO DO PULO','2345-6789')
```

(B)
```
INSERT INTO ATLETA(NOME,COD,TELEFONE)
VALUES('JOAO DO PULO',12345,'23456789')
```

(C)
```
INSERT INTO ATLETA(NOME,COD,TELEFONE,NUM_INSCR)
VALUES('JOAO DO PULO',12345,7777777)
```

(D)
```
INSERT INTO ATLETA
VALUES(12345,'JOAO DO PULO','2345-6789',7777777)
```

(E)
```
INSERT INTO ATLETA(NOME,COD,NUM_INSCR)
VALUES('JOAO DO PULO',12345,7777777)
```

Questão 23

[2014 - EPE - Tecnologia da Informação (50)]

Pessoa			
Id	Nome	Idade	Sexo
1	Ana	70	F
2	Beto	65	M
3	Carlos	45	M
4	Débora	40	F
5	Ênio	22	M
6	Fabiana	20	F
7	Guto	1	M
8	Hilda	52	F

Parentesco	
PaiMae	FilhoFilha
1	3
1	4
2	3
2	4
5	7
6	7
3	5
4	5

Que comando SQL inclui a informação de que Hilda é mãe de Fabiana?

(A)
```
INSERT INTO Parentesco SELECT F.Id,P.Id FROM Pessoa AS P , Pessoa AS F
WHERE P.Nome='Hilda' AND F.Nome='Fabiana‘
```

(B)
```
INSERT INTO Parentesco SELECT P.Id,F.Id FROM Pessoa AS P , Pessoa AS F
WHERE P.Nome='Hilda' AND F.Nome='Fabiana‘
```

(C)
```
INSERT INTO Parentesco SELECT P.Id,F.Id FROM Pessoa AS P , Pessoa AS F
WHERE P.Nome='Fabiana' AND F.Nome='Hilda‘
```

(D)
```
INSERT INTO Parentesco VALUES SELECT F.Id,P.Id FROM Pessoa AS P , Pessoa AS F
WHERE P.Nome='Hilda' AND F.Nome='Fabiana‘
```

(E)
```
INSERT INTO Parentesco VALUES SELECT P.Id,F.Id FROM Pessoa AS P , Pessoa AS F
WHERE P.Nome='Hilda' AND F.Nome='Fabiana
```

Remoções em tabelas

- O comando DELETE remove linhas de uma tabela.
- Sintaxe:

```
DELETE FROM table_name
WHERE <condição>;
```

- Exemplo:

```
DELETE FROM FUNCIONARIO
WHERE NOME='Pedro';
```

Ação referencial

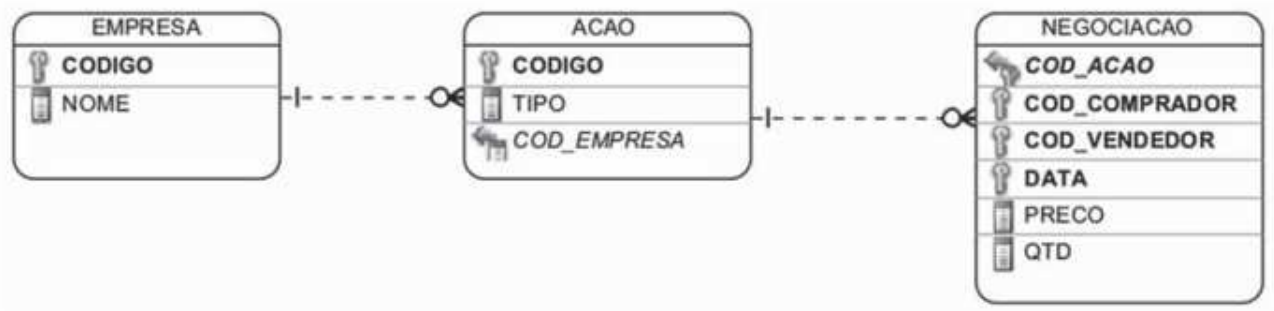
- Ao definir uma tabela, o projetista pode definir uma ação referencial a qualquer restrição de chave estrangeira.
- Uma opção deve ser qualificada com ON DELETE ou ON UPDATE.
- As opções incluem
  - SET NULL – os valores dos atributos de referência serão mudados para NULL.
  - CASCADE – propagação de eliminação/valores.
  - SET DEFAULT - os valores dos atributos de referência serão mudado para o valor padrão especificado.
- Exemplo:

```
CREATE TABLE FUNCIONARIO (
    MAT       INT PRIMARY KEY,
    ...

    DEPTO     VARCHAR(10),
    CONSTRAINT FKDEPTO FOREIGN KEY (DEPTO)
        REFERENCE DEPARTAMENTO(COD) ON DELETE CASCADE
);
```

Questão 24

O diagrama e as tabelas são utilizados para descrever um banco de dados que contém informações sobre negociações de ações em uma bolsa de valores.




```
CREATE TABLE EMPRESA (
 CODIGO NUMBER(7) NOT NULL,
 NOME VARCHAR2(50) NOT NULL,
 CONSTRAINT EMPRESA_PK PRIMARY KEY (CODIGO)
)

CREATE TABLE NEGOCIACAO (
 COD_ACAO CHAR(5) NOT NULL,
 COD_COMPRADOR NUMBER(7) NOT NULL,
 COD_VENDEDOR NUMBER(7) NOT NULL,
 DATA DATE NOT NULL,
 PRECO NUMBER(9,2) NOT NULL,
 QTD NUMBER(9,0) NOT NULL,
 CONSTRAINT NEGOCIACAO_PK PRIMARY KEY
 (COD_ACAO,COD_COMPRADOR,COD_VENDEDOR,DATA),
 CONSTRAINT NEGOCIACAO_FK FOREIGN KEY (COD_ACAO)
 REFERENCES “ACAO” (“CODIGO”)
)
```

```
CREATE TABLE ACAO (
 CODIGO CHAR(5) NOT NULL,
 TIPO CHAR(2) NOT NULL,
 COD_EMPRESA NUMBER(7) NOT NULL,
 CONSTRAINT ACAO_PK PRIMARY KEY (CODIGO),
 FOREIGN KEY (COD_EMPRESA)
     REFERENCES EMPRESA (CODIGO)
)
```

Questão 24

ITnerante

[2012 - Liquigás - Analise de Sistemas (51)]

Qual sequência de comandos SQL deve ser executada para excluir do banco de dados as informações sobre a empresa cujo nome é PGP PETROLEO?

(A)
```
DELETE FROM EMPRESA WHERE NOME = 'PGP PETROLEO'
DELETE FROM ACAO WHERE COD_EMPRESA = (SELECT CODIGO FROM EMPRESA
WHERE NOME= 'PGP PETROLEO')
DELETE FROM NEGOCIACAO WHERE COD_ACAO = (SELECT CODIGO FROM ACAO
WHERE COD_EMPRESA =(SELECT CODIGO FROM EMPRESA WHERE NOME= 'PGP PETROLEO'))
```

(B)
```
DELETE FROM EMPRESA WHERE NOME = 'PGP PETROLEO'ON CASCADE
```

(C)
```
DELETE FROM NEGOCIACAO WHERE COD_ACAO IN (SELECT CODIGO FROM ACAO
WHERE COD_EMPRESA =(SELECT CODIGO FROM EMPRESA WHERE NOME='PGP PETROLEO'))
DELETE FROM ACAO WHERE COD_EMPRESA = (SELECT CODIGO FROM EMPRESA WHERE NOME='PGP PETROLEO')
DELETE FROM EMPRESA WHERE NOME='PGP PETROLEO'
```

(D)
```
DELETE FROM ACAO WHERE COD_EMPRESA = (SELECT CODIGO FROM EMPRESA WHERE NOME='PGP PETROLEO')
DELETE FROM NEGOCIACAO WHERE COD_ACAO = (SELECT CODIGO FROM ACAO
WHERE COD_EMPRESA =(SELECT CODIGO FROM EMPRESA WHERE NOME='PGP PETROLEO'))
DELETE FROM EMPRESA WHERE NOME=’PGP PETROLEO’
```

(E)
```
DELETE FROM NEGOCIACAO WHERE COD_ACAO = (SELECT CODIGO FROM ACAO
WHERE COD_EMPRESA =(SELECT CODIGO FROM EMPRESA WHERE NOME='PGP PETROLEO'))
DELETE FROM ACAO WHERE COD_EMPRESA = (SELECT CODIGO FROM EMPRESA WHERE NOME='PGP PETROLEO')
DELETE FROM EMPRESA WHERE NOME='PGP PETROLEO'
```

Atualizações em tabelas

- O comando UPDATE é usado para modificar valores de atributo de uma ou mais tuplas selecionadas.
- Sintaxe:

```
UPDATE table_name
SET column1=value1, column2=value2, ...
WHERE <condição>
```

- Exemplo:

```
UPDATE FUNCIONARIO
SET SALARIO=10000
WHERE NOME='Pedro';
```

Questão 25

As tabelas abaixo pertencem ao esquema de um banco de dados de um supermercado.

```
CREATE TABLE PRODUTO (
     COD NUMBER(5) NOT NULL,
     DESCRICAO VARCHAR2(100) NOT NULL,
     PRECO NUMBER(8,2) NOT NULL,
     QTD_ESTOQUE NUMBER(5) ,
     TIPO NUMBER(1) NOT NULL,
     CONSTRAINT PRODUTO_PK PRIMARY KEY (COD))

CREATE TABLE ITEM (
     NUM_SERIE NUMBER(7) NOT NULL,
     COR VARCHAR2(20) NOT NULL,
     VOLTAGEM NUMBER(5) NOT NULL,
     COD_PROD NUMBER(5) NOT NULL,
     CONSTRAINT ITEM_PK PRIMARY KEY (NUM_SERIE),
     CONSTRAINT ITEM_FK FOREIGN KEY (COD_PROD)
     REFERENCES PRODUTO (COD))
```

Observações:

• A empresa comercializa produtos controlados por quantidade (por exemplo, caixa de sabão em pó, pacote de biscoito e lata de extrato de tomate) e produtos controlados por unidade (por exemplo, televisor, máquina de lavar roupa e liquidificador).

• A quantidade em estoque de um produto controlado por quantidade (TIPO=1) é obtida diretamente da coluna QTD_ESTOQUE.

• A quantidade em estoque de um produto controlado por unidade (TIPO=2) NÃO pode ser obtida diretamente da coluna QTD_ESTOQUE, pois, para esse tipo de produto, esta coluna irá conter o valor NULL.

• Cada linha da tabela ITEM contém informações sobre um item existente no estoque da empresa relativo a um tipo de produto controlado por unidade.

Questão 25

ITnerante

[2013 - FINEP - Desenvolvimento de Sistemas (45)]

O analista de suporte de banco de dados do supermercado solicitou que a coluna QTD_ESTOQUE passasse a conter a quantidade de itens em estoque de produtos do tipo 2. Embora ele reconheça que isso resultará em redundância, os relatórios de performance mostram que existe um desperdício de recursos computacionais significativo com o cálculo recorrente do total de itens em estoque de produtos do tipo 2.

Qual comando SQL irá atualizar corretamente a coluna QTD_ESTOQUE com a quantidade de itens em estoque relativa a cada um dos produtos do tipo 2 comercializados pelo supermercado?

(A)
```
UPDATE PRODUTO P SET QTD_ESTOQUE=(SELECT COUNT(*) FROM ITEM
WHERE COD_PROD=P.COD)
```

(B)
```
UPDATE PRODUTO P SET QTD_ESTOQUE=(SELECT COUNT(*) FROM ITEM
WHERE COD_PROD=P.COD AND P.TIPO=2)
```

(C)
```
UPDATE PRODUTO P SET QTD_ESTOQUE=(SELECT COUNT(DISTINCT COD_PROD)
FROM ITEM WHERE COD_PROD=P.COD AND P.TIPO=2)
```

(D)
```
UPDATE PRODUTO P SET QTD_ESTOQUE=(SELECT COUNT(COD_PROD) FROM ITEM
WHERE COD_PROD=P.COD) WHERE TIPO = 2
```

(E)
```
UPDATE PRODUTO P SET QTD_ESTOQUE=(SELECT COUNT(DISTINCT COD_PROD)
FROM ITEM WHERE COD_PROD=P.COD) WHERE TIPO=2
```

- **UNION**
  - O resultado de uma operação de união é uma tabela que inclui todas as linhas que fazem parte das tabelas participantes.
  - Por padrão, essa operação elimina as linhas duplicadas. Para manter as linhas duplicadas devemos usar `UNION ALL`.
  - É necessário que as tabelas envolvidas tenham compatibilidade de tipo.
  - Correspondência com Álgebra Relacional:

```
SELECT * FROM ALUNO
UNION
SELECT * FROM PROFESSOR;
```

```
ALUNO U PROFESSOR
```

- **INTERSECT**
  - O resultado dessa operação é uma tabela que inclui todas as linhas que fazem parte das duas tabelas participantes.
  - Por padrão, essa operação elimina as linhas duplicadas. Para manter as linhas duplicadas devemos usar `INTERSECT ALL`.
  - É necessário que as tabelas envolvidas tenham compatibilidade de tipo.
  - Correspondência com Álgebra Relacional:

```
SELECT * FROM ALUNO
INTERSECT
SELECT * FROM PROFESSOR;
```

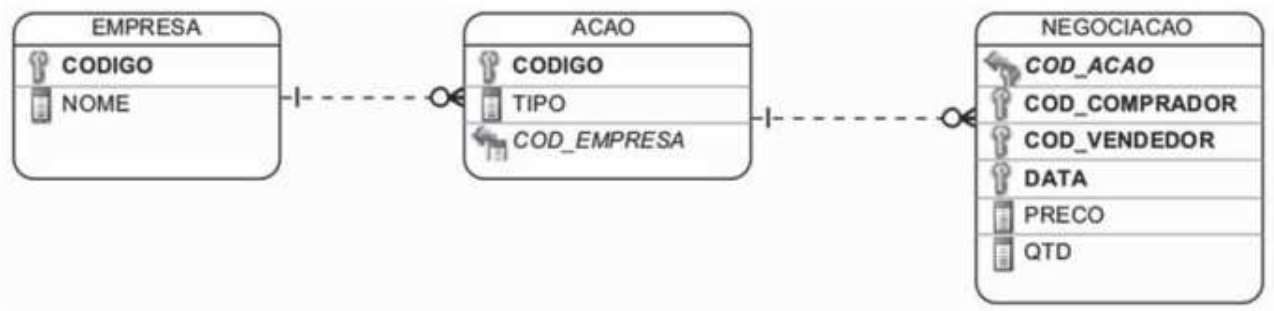
- **EXCEPT / MINUS**
  - O resultado dessa operação é uma tabela que inclui todas as linhas que estão na primeira tabela menos as que constam na segunda tabela.
  - Por padrão, essa operação elimina as linhas duplicadas. Para manter as linhas duplicadas devemos usar **EXCEPT ALL**.
  - É necessário que as tabelas envolvidas tenham compatibilidade de tipo.
  - Correspondência com Álgebra Relacional:

```
SELECT * FROM ALUNO
EXCEPT
SELECT * FROM PROFESSOR;
```

```
ALUNO - PROFESSOR
```

Questão 26

O diagrama e as tabelas são utilizados para descrever um banco de dados que contém informações sobre negociações de ações em uma bolsa de valores.




```
CREATE TABLE EMPRESA (
 CODIGO NUMBER(7) NOT NULL,
 NOME VARCHAR2(50) NOT NULL,
 CONSTRAINT EMPRESA_PK PRIMARY KEY (CODIGO)
)

CREATE TABLE NEGOCIACAO (
 COD_ACAO CHAR(5) NOT NULL,
 COD_COMPRADOR NUMBER(7) NOT NULL,
 COD_VENDEDOR NUMBER(7) NOT NULL,
 DATA DATE NOT NULL,
 PRECO NUMBER(9,2) NOT NULL,
 QTD NUMBER(9,0) NOT NULL,
 CONSTRAINT NEGOCIACAO_PK PRIMARY KEY
 (COD_ACAO,COD_COMPRADOR,COD_VENDEDOR,DATA),
 CONSTRAINT NEGOCIACAO_FK FOREIGN KEY (COD_ACAO)
 REFERENCES “ACAO” (“CODIGO”)
)
```

```
CREATE TABLE ACAO (
 CODIGO CHAR(5) NOT NULL,
 TIPO CHAR(2) NOT NULL,
 COD_EMPRESA NUMBER(7) NOT NULL,
 CONSTRAINT ACAO_PK PRIMARY KEY (CODIGO),
 FOREIGN KEY (COD_EMPRESA)
     REFERENCES EMPRESA (CODIGO)
)
```

Questão 26

ITnerante

[2012 - Liquigás - Analise de Sistemas (52)]

A consulta que permite exibir o nome de todas as empresas que NÃO tiveram ações negociadas na bolsa de valores no dia 25-01-2012 é

(A)
```
SELECT NOME FROM EMPRESA
MINUS
SELECT E.NOME FROM EMPRESA E, ACAO A, NEGOCIACAO N
WHERE E.CODIGO=A.COD_EMPRESA AND A.CODIGO=N.COD_ACAO AND
N.DATA='25-01-2012'
```

(B)
```
SELECT NOME FROM EMPRESA
INTERSECT
SELECT E.NOME FROM EMPRESA E, ACAO A, NEGOCIACAO N
WHERE E.CODIGO=A.COD_EMPRESA AND A.CODIGO=N.COD_ACAO AND
N.DATA='25-01-2012'
```

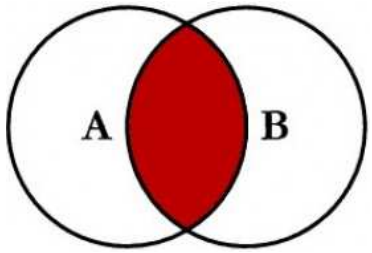
(C)
```
SELECT NOME FROM EMPRESA
UNION
SELECT E.NOME FROM EMPRESA E, ACAO A, NEGOCIACAO N
WHERE E.CODIGO=A.COD_EMPRESA AND A.CODIGO=N.COD_ACAO AND
N.DATA='25-01-2012'
```

(D)
```
SELECT E.NOME FROM EMPRESA E, ACAO A, NEGOCIACAO N
WHERE E.CODIGO=A.COD_EMPRESA AND A.CODIGO=N.COD_ACAO AND
N.DATA != '25-01-2012'
```

(E)
```
SELECT E.NOME FROM EMPRESA E WHERE E.CODIGO IN
(SELECT E.CODIGO FROM EMPRESA E, ACAO A, NEGOCIACAO N
WHERE E.CODIGO=A.COD_EMPRESA AND A.CODIGO=N.COD_ACAO AND
N.DATA = '25-01-2012')
```

Junção Interna

- Fornece uma tabela resultante que conecta duas tabelas, retornando apenas pares de linhas que combinem com a condição de junção.



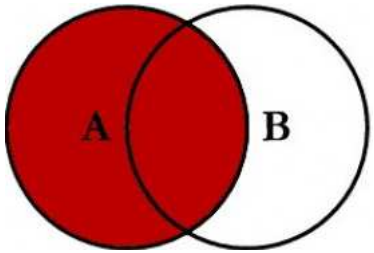

```
SELECT <select_list>
FROM A INNER JOIN B
ON A.KEY=B.KEY;
```

- Exemplo:

```
SELECT *
FROM FUNCIONARIO AS A INNER JOIN DEPARTAMENTO AS B
ON A.DEPTO= B.CODIGO;
```

Junção Externa à Esquerda

- Fornece uma tabela resultante que contêm todas as linhas da tabela da esquerda.
- Apenas as linhas da tabela da direita que possuírem correspondência com as linhas da tabela da esquerda que irão compor a tabela resultante.
- Se não houver correspondência na tabela da direita, os atributos são preenchidos com valores NULL.



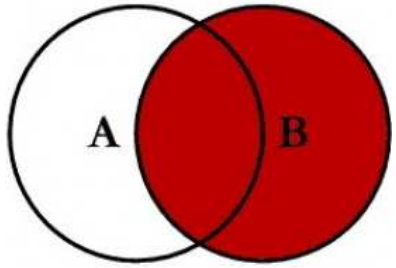

```
SELECT <select_list>
FROM A LEFT OUTER JOIN B
ON A.KEY=B.KEY
```

- Exemplo:

```
SELECT *
FROM FUNCIONARIO AS A LEFT OUTER JOIN DEPARTAMENTO AS B
ON A.MAT=B.GERENTE;
```

Junção Externa à Direita

- Fornece uma tabela resultante que contêm todas as linhas da tabela da direita.
- Apenas as linhas da tabela da esquerda que possuírem correspondência com as linhas da tabela da direita que irão compor a tabela resultante.
- Se não houver correspondência na tabela da esquerda, os atributos são preenchidos com valores NULL.



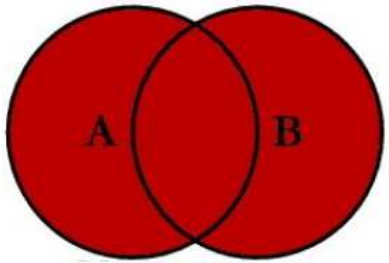

```
SELECT <select_list>
FROM A RIGHT OUTER JOIN B
ON A.KEY=B.KEY
```

- Exemplo:

```
SELECT *
FROM DEPARTAMENTO AS A RIGHT OUTER JOIN FUNCIONARIO AS B
ON A.GERENTE=B.MAT;
```

Junção Externa Completa

- Fornece uma tabela resultante que contêm todas as linhas da duas tabelas participantes.
- Caso haja correspondência entre as linhas das duas tabelas, elas irão compor uma linha da tabela resultante.
- Se não houver correspondência, os outros atributos são preenchidos com NULL.




```
SELECT <select_list>
FROM A FULL OUTER JOIN B
ON A.KEY=B.KEY
```

- Exemplo:

```
SELECT *
FROM FUNCIONARIO AS A FULL OUTER JOIN DEPARTAMENTO AS B
ON A.DEPTO= B.CODIGO;
```

Questão 27

[2011 – BR Distribuidora - Infraestrutura (39)]

Considere o resultado da consulta SQL a seguir, quaisquer que sejam os dados disponíveis nas tabelas Cidades e Veículos.

```
SELECT * FROM Cidades, Veiculos WHERE Cidades.Nome=Veiculos.Nome;
```

Para manter o resultado final dessa consulta usando a notação de JOIN, a instrução SQL deve ser a seguinte:

(A) `SELECT * FROM Veiculos CROSS JOIN Cidades`

(B) `SELECT * FROM Veiculos LEFT JOIN Cidades ON Cidades.Nome=Veiculos.Nome`

(C) `SELECT * FROM Veiculos OUTER JOIN Cidades ON Cidades.Nome=Veiculos.Nome`

(D) `SELECT * FROM Veiculos INNER JOIN Cidades ON Cidades.Nome=Veiculos.Nome`

(E) `SELECT * FROM Veiculos RIGHT JOIN Cidades ON Cidades.Nome=Veiculos.Nome`

Questão 28

ITnerante

[2012 - EPE - Tecnologia da Informação (34)]

Tabela R

id	nome	cidade
1234	João	Rio de Janeiro
1235	Luiz	Fortaleza
1236	Leonardo	Porto Alegre
1237	Lúcio	Manaus
1238	Maria	Cuiabá

Tabela S

id	cargo	salario
1234	Engenheiro	3200
1235	Engenheiro	1250
1236	Projetista	5000
1237	Arquiteto	2000

Considere o comando em SQL ANSI abaixo.

```
select R.*, S.*
from R left outer join S on S.id = R.id
```

A execução do comando SQL acima resulta em uma relação que possui, respectivamente, uma quantidade de linhas e de colunas iguais a

(A) 4 e 5
(B) 4 e 6
(C) 5 e 6
(D) 20 e 2
(E) 20 e 6

Questão 29

As tabelas abaixo pertencem ao esquema de um banco de dados de um supermercado.

```
CREATE TABLE PRODUTO (
     COD NUMBER(5) NOT NULL,
     DESCRICAO VARCHAR2(100) NOT NULL,
     PRECO NUMBER(8,2) NOT NULL,
     QTD_ESTOQUE NUMBER(5) ,
     TIPO NUMBER(1) NOT NULL,
     CONSTRAINT PRODUTO_PK PRIMARY KEY (COD))

CREATE TABLE ITEM (
     NUM_SERIE NUMBER(7) NOT NULL,
     COR VARCHAR2(20) NOT NULL,
     VOLTAGEM NUMBER(5) NOT NULL,
     COD_PROD NUMBER(5) NOT NULL,
     CONSTRAINT ITEM_PK PRIMARY KEY (NUM_SERIE),
     CONSTRAINT ITEM_FK FOREIGN KEY (COD_PROD)
     REFERENCES PRODUTO (COD))
```

Observações:

• A empresa comercializa produtos controlados por quantidade (por exemplo, caixa de sabão em pó, pacote de biscoito e lata de extrato de tomate) e produtos controlados por unidade (por exemplo, televisor, máquina de lavar roupa e liquidificador).

• A quantidade em estoque de um produto controlado por quantidade (TIPO=1) é obtida diretamente da coluna QTD_ESTOQUE.

• A quantidade em estoque de um produto controlado por unidade (TIPO=2) NÃO pode ser obtida diretamente da coluna QTD_ESTOQUE, pois, para esse tipo de produto, esta coluna irá conter o valor NULL.

• Cada linha da tabela ITEM contém informações sobre um item existente no estoque da empresa relativo a um tipo de produto controlado por unidade.

Questão 29

[2013 - FINEP - Desenvolvimento de Sistemas (43)]

Qual consulta SQL irá exibir o código, a descrição e a quantidade em estoque relativos a cada um dos produtos comercializados pelo supermercado?

(A)
```
SELECT COD, DESCRICAO, QTD_ESTOQUE
FROM PRODUTO
WHERE TIPO = 1
UNION
SELECT P.COD, P.DESCRICAO, COUNT(I.COD_PROD)
FROM PRODUTO P,ITEM I
WHERE TIPO = 2 AND P.COD=I.COD_PROD
GROUP BY P.COD, P.DESCRICAO
```

(B)
```
SELECT COD, DESCRICAO, QTD_ESTOQUE
FROM PRODUTO
WHERE TIPO = 1
UNION
SELECT P.COD, P.DESCRICAO, COUNT(I.COD_PROD)
FROM PRODUTO P
LEFT JOIN ITEM I
ON P.COD=I.COD_PROD
WHERE P.TIPO = 2
GROUP BY P.COD, P.DESCRICAO
```

(C)
```
SELECT P.COD, P.DESCRICAO, COUNT(DISTINCT
P.TIPO)
FROM PRODUTO P
LEFT OUTER JOIN ITEM I
ON P.COD=I.COD_PROD
GROUP BY P.COD, P.DESCRICAO
```

(D)
```
SELECT P.COD, P.DESCRICAO, SUM (DISTINCT
P.TIPO)
FROM PRODUTO P
INNER JOIN ITEM I
ON P.COD=I.COD_PROD
GROUP BY P.COD, P.DESCRICAO
```

(E)
```
SELECT COD, DESCRICAO, QTD_ESTOQUE
FROM PRODUTO
WHERE TIPO = 1
UNION
SELECT P.COD, P.DESCRICAO, COUNT(I.COD_PROD)
FROM PRODUTO P
RIGHT JOIN ITEM I
ON P.COD=I.COD_PROD
WHERE P.TIPO = 2
GROUP BY P.COD,P.DESCRICAO
```

A função EXISTS

- A função EXISTS em SQL costuma ser usada para verificar se o resultado de uma consulta aninhada correlacionada é vazio ou não.
- O resultado de EXISTS é um valor booleano TRUE se o resultado da consulta aninhada tiver pelo menos uma tupla, ou FALSE, se o resultado da consulta aninhada não tiver tuplas.
- Exemplo:

```
SELECT FUNC.NOME, FUNC.TELEFONE
FROM FUNCIONARIO FUNC
WHERE EXISTS (   SELECT *
                 FROM DEPENDENTE DEP
                 WHERE DEP.FCPF=FUNC.CPF );
```

Questão 30

ITnerante

[2012 - Transpetro - Infraestrutura (63)]

Tabela **Lojas**

nome_loja	vendas
Barra	1500
Ipanema	250
Copacabana	300
Tijuca	700

Tabela **Regioes**

nome_regiao	nome_loja
Sul	Ipanema
Sul	Copacabana
Oeste	Barra
Norte	Tijuca

Considerando as duas tabelas apresentadas, a consulta SQL

```
SELECT SUM( vendas ) FROM Lojas
WHERE EXISTS ( SELECT * FROM Regioes WHERE nome_regiao = 'Sul' )
```

apresentará o seguinte resultado:

(A) 250
(B) 350
(C) 550
(D) 2250
(E) 2750

Questão 31

ITnerante

[2012 - Petroquímica Suape - Analista de sistemas (45)]

Considere um conjunto de tabelas que representa um banco de dados relacional.

```
CONSULTOR (id, nome, email, nomeConsultoria)
PROJETO (id, descricao, investimento)
ATUACAO (idConsultor, idProjeto)
```

E as seguintes restrições:

- Os campos sublinhados representam as chaves primárias das relações.
- O campo idConsultor da tabela ATUACAO referencia o campo id da tabela CONSULTOR.
- O campo idProjeto da tabela ATUACAO referencia o campo id da tabela PROJETO.

```
SELECT C.nome
FROM Consultor AS C
WHERE NOT EXISTS (
                        SELECT 1
                        FROM Atuacao A
                        WHERE A.idConsultor = C.idConsultor)
```

A consulta acima, escrita em SQL, retorna o(s)

(A) nome de todos os consultores, e, caso estejam alocados em projetos, o nome do projeto.

(B) consultores que atuam em projetos.

(C) consultores que atuam em todos os projetos.

(D) consultores que não atuam em projetos.

(E) projetos que tenham consultores atuando.

Divisão em SQL

- A seguir temos a representação da operação divisão na linguagem SQL.

ALUNO

matricula	nome
11111	Vivian
22222	Gabriel
33333	Aline

INSCRICAO

matricula	codigo
11111	D1
22222	D1
11111	D2
11111	D3
22222	D2
33333	D3

DISCIPLINA

codigo
D1
D2
D3

Quais alunos estão inscritos em <u>todas</u> as disciplinas?

Exemplo:

```
SELECT * FROM ALUNO A
WHERE NOT EXISTS ( SELECT * FROM DISCIPLINA D
                   WHERE NOT EXISTS (  SELECT * FROM INSCRICAO I
                                       WHERE I.MATRICULA=A.MATRICULA AND
                                             I.CODIGO=D.CODIGO);
```

Questão 32

ITnerante

[2011 - BNDES - Desenvolvimento de Sistemas (53)]

As tabelas Projeto, Funcionario e Participacao_Projeto participam da base de dados de um sistema de controle de projetos.
A estrutura dessas tabelas é a seguinte:

```
Projeto (IDProjeto, Nome, DataInicio, DataFim)
Funcionario (Matricula, Nome, DataNascimento)
Participacao_Projeto (IDProjeto, Matricula)
```

Colunas sublinhadas participam da chave primária
Colunas em negrito participam de chaves estrangeiras
O comando SQL que retorna o nome somente dos funcionários que participaram de TODOS os projetos é:

Questão 32

(A)
```
SELECT DISTINCT F.NOME FROM FUNCIONARIO F
  INNER JOIN PARTICIPACAO_PROJETO PP
  ON F.MATRICULA = PP.MATRICULA
    INNER JOIN PROJETO P
    ON P.IDPROJETO = PP.IDPROJETO
```

(B)
```
SELECT DISTINCT F.NOME FROM FUNCIONARIO F
   INNER JOIN PARTICIPACAO_PROJETO PP
      ON F.MATRICULA = PP.MATRICULA
   WHERE EXISTS (SELECT 1 FROM PROJETO P
     WHERE P.IDPROJETO = PP.IDPROJETO)
```

(C)
```
SELECT F.NOME FROM FUNCIONARIO F
  WHERE MATRICULA IN
  (SELECT MATRICULA FROM PARTICIPACAO_PROJETO)
```

(D)
```
SELECT F.NOME FROM FUNCIONARIO F
  WHERE NOT EXISTS
   (SELECT 1 FROM PROJETO P
    WHERE NOT EXISTS (SELECT 1 FROM
       PARTICIPACAO_PROJETO PP
       WHERE PP.IDPROJETO = P.IDPROJETO
       AND PP.MATRICULA = F.MATRICULA))
```

(E)
```
SELECT F.NOME FROM FUNCIONARIO F
  WHERE NOT EXISTS
    (SELECT 1 FROM PARTICIPACAO_PROJETO PP
     WHERE PP.MATRICULA = F.MATRICULA
     AND NOT EXISTS (SELECT 1 FROM PROJETO P
         WHERE PP.IDPROJETO = P.IDPROJETO))
```

Gabarito

12 – A
13 – B
14 – E
15 – C
16 – A
17 – E
18 – B
19 – A
20 – A

21 – A
22 – B
23 – B
24 – C
25 – D
26 – A
27 – D
28 – C
29 – B

30 – E
31 – D
32 – D

PROVAS DE TI

TUDO PARA VOCÊ PASSAR

FIM

Rodrigo Adur
rodrigoadurti@gmail.com

ITnerante

TIMASTERS