



### **Título da prática:**

Modelagem e implementação de um banco de dados simples, utilizando como base o SQL Server.

**CAMPUS:** Polo Planalto – BH – MG

**DISCIPLINA:** BackEnd sem banco não tem!

**TURMA:** 2025.1

**SEMESTRE LETIVO:** Primeiro Semestre (2025)

**ALUNO:** Bruno Ricardo Viana Venturelli

**Matrícula:** 202401226726

### **LINK DO MEU GITHUB:**

[https://github.com/DevBrN01/trabalho\\_n2\\_BackEnd](https://github.com/DevBrN01/trabalho_n2_BackEnd)

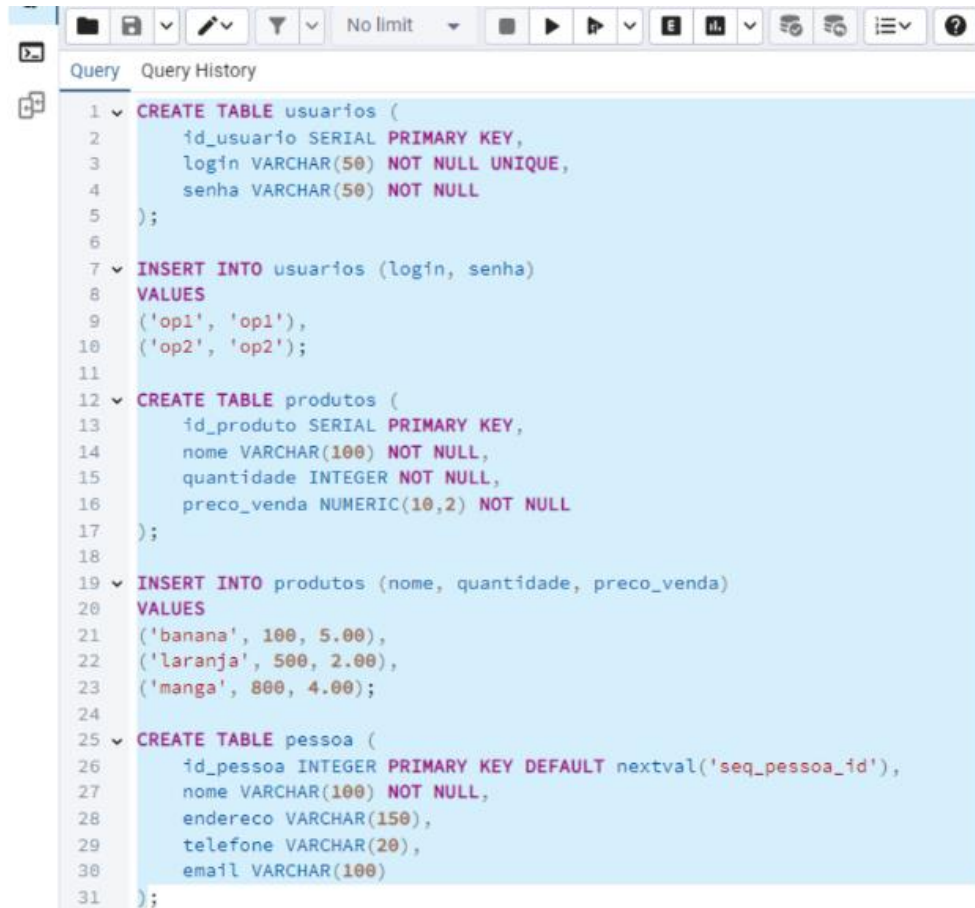
### **Objetivo da Prática**

1. Identificar os requisitos de um Sistema e transformá-los no modelo adequado.
2. Utilizar ferramentas de modelagem para bases de dados relacionais.
3. Explorar a sintaxe SQL na criação das estruturas do banco (DDL).
4. Explorar a sintaxe SQL na consulta e manipulação de dados (DML).

## Segundo Procedimento:

### Alimentando a Base

### Códigos usados neste roteiro



```
1 CREATE TABLE usuarios (  
2     id_usuario SERIAL PRIMARY KEY,  
3     login VARCHAR(50) NOT NULL UNIQUE,  
4     senha VARCHAR(50) NOT NULL  
5 );  
6  
7 INSERT INTO usuarios (login, senha)  
8 VALUES  
9 ('op1', 'op1'),  
10 ('op2', 'op2');  
11  
12 CREATE TABLE produtos (  
13     id_produto SERIAL PRIMARY KEY,  
14     nome VARCHAR(100) NOT NULL,  
15     quantidade INTEGER NOT NULL,  
16     preco_venda NUMERIC(10,2) NOT NULL  
17 );  
18  
19 INSERT INTO produtos (nome, quantidade, preco_venda)  
20 VALUES  
21 ('banana', 100, 5.00),  
22 ('laranja', 500, 2.00),  
23 ('manga', 800, 4.00);  
24  
25 CREATE TABLE pessoa (  
26     id_pessoa INTEGER PRIMARY KEY DEFAULT nextval('seq_pessoa_id'),  
27     nome VARCHAR(100) NOT NULL,  
28     endereco VARCHAR(150),  
29     telefone VARCHAR(20),  
30     email VARCHAR(100)  
31 );
```

```
Query Query History

32
33 CREATE TABLE pessoa_fisica (
34     id_pessoa INTEGER PRIMARY KEY,
35     cpf VARCHAR(14) NOT NULL UNIQUE,
36     FOREIGN KEY (id_pessoa) REFERENCES pessoa(id_pessoa)
37 );
38
39 CREATE TABLE pessoa_juridica (
40     id_pessoa INTEGER PRIMARY KEY,
41     cnpj VARCHAR(18) NOT NULL UNIQUE,
42     FOREIGN KEY (id_pessoa) REFERENCES pessoa(id_pessoa)
43 );
44
45 SELECT nextval('seq_pessoa_id');
46
47 INSERT INTO pessoa (id_pessoa, nome, endereco, telefone, email)
48 VALUES (1, 'João Silva', 'Rua das Flores, 123', '(11) 98765-4321', 'joao@email.com');
49
50 INSERT INTO pessoa_fisica (id_pessoa, cpf)
51 VALUES (1, '123.456.789-00');
52
53 INSERT INTO pessoa (id_pessoa, nome, endereco, telefone, email)
54 VALUES (2, 'Empresa XYZ Ltda.', 'Av. Central, 5000', '(11) 99876-5432', 'contato@empresa.com');
55
56 INSERT INTO pessoa_juridica (id_pessoa, cnpj)
57 VALUES (2, '12.345.678/0001-99');
58
59 CREATE TABLE movimentacoes (
60     id_movimentacao SERIAL PRIMARY KEY,
61     id_produto INTEGER NOT NULL,
62     id_usuario INTEGER NOT NULL,
63     tipo CHAR(1) NOT NULL, -- 'E' para Entrada (compra), 'S' para Saída (venda)
64     quantidade INTEGER NOT NULL,
65     preco_unitario NUMERIC(10,2) NOT NULL,
66     data_movimentacao TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
67     FOREIGN KEY (id_produto) REFERENCES produtos(id_produto),
68     FOREIGN KEY (id_usuario) REFERENCES usuarios(id_usuario)
69 );
```

```
Query Query History
70
71 -- Entrada (Compra) de bananas
72 v INSERT INTO movimentacoes (id_produto, id_usuario, tipo, quantidade, preco_unitario)
73 VALUES (1, 1, 'E', 100, 4.00);
74
75 -- Saída (Venda) de bananas
76 v INSERT INTO movimentacoes (id_produto, id_usuario, tipo, quantidade, preco_unitario)
77 VALUES (1, 2, 'S', 30, 5.00);
78
79 -- Entrada (Compra) de laranjas
80 v INSERT INTO movimentacoes (id_produto, id_usuario, tipo, quantidade, preco_unitario)
81 VALUES (2, 1, 'E', 500, 1.50);
82
83 -- Saída (Venda) de laranjas
84 v INSERT INTO movimentacoes (id_produto, id_usuario, tipo, quantidade, preco_unitario)
85 VALUES (2, 2, 'S', 200, 2.00);
86
87 -- Entrada (Compra) de mangas
88 v INSERT INTO movimentacoes (id_produto, id_usuario, tipo, quantidade, preco_unitario)
89 VALUES (3, 1, 'E', 800, 3.50);
90
91 v INSERT INTO movimentacoes (id_produto, id_usuario, tipo, quantidade, preco_unitario)
92 VALUES (3, 2, 'S', 500, 4.00);
93
94 -- Consultas SQL conforme enunciado:
95
96 -- Dados completos de pessoas físicas
97 v SELECT p.id_pessoa, p.nome, p.endereco, p.telefone, p.email, pf.cpf
98 FROM pessoa p
99 JOIN pessoa_fisica pf ON p.id_pessoa = pf.id_pessoa;
100
101 -- Dados completos de pessoas jurídicas
102 v SELECT p.id_pessoa, p.nome, p.endereco, p.telefone, p.email, pj.cnpj
103 FROM pessoa p
104 JOIN pessoa_juridica pj ON p.id_pessoa = pj.id_pessoa;
```

Query Query History

```

93
94 -- Consultas SQL conforme enunciado:
95
96 -- Dados completos de pessoas físicas
97 SELECT p.id_pessoa, p.nome, p.endereco, p.telefone, p.email, pf.cpf
98 FROM pessoa p
99 JOIN pessoa_fisica pf ON p.id_pessoa = pf.id_pessoa;
100
101 -- Dados completos de pessoas jurídicas
102 SELECT p.id_pessoa, p.nome, p.endereco, p.telefone, p.email, pj.cnpj
103 FROM pessoa p
104 JOIN pessoa_juridica pj ON p.id_pessoa = pj.id_pessoa;
105
106 -- Movimentações de entrada (compra):
107 -- . Produto
108 -- . Fornecedor (pessoa jurídica)
109 -- . Quantidade
110 -- . Preço unitário
111 -- . Valor total (quantidade * preço unitário)
112 SELECT
113     m.id_movimentacao,
114     pr.nome AS produto,
115     pe.nome AS fornecedor,
116     m.quantidade,
117     m.preco_unitario,
118     (m.quantidade * m.preco_unitario) AS valor_total
119 FROM movimentacoes m
120 JOIN produtos pr ON m.id_produto = pr.id_produto
121 JOIN pessoa_juridica pj ON pj.id_pessoa = p.j.id_pessoa
122 JOIN pessoa pe ON pj.id_pessoa = pe.id_pessoa
123 WHERE m.tipo = 'E';
124
125 -- Movimentações de saída (venda):
126 -- . Produto
127 -- . Comprador (pessoa física)
128 -- . Quantidade
129 -- . Preço unitário
130 -- . Valor total
131 SELECT
132     m.id_movimentacao,

```

Data Output Messages Notifications

	id_pessoa	nome	endereco	telefone	email	cpf
	integer	character varying (100)	character varying (150)	character varying (20)	character varying (100)	character varying (14)
1	1	João Silva	Rua das Flores, 123	(11) 98765-4321	joao@email.com	123.456.789-00

```

Query      Query History
100
101 -- Dados completos de pessoas jurídicas
102 SELECT p.id_pessoa, p.nome, p.endereco, p.telefone, p.email, pj.cnpj
103 FROM pessoa p
104 JOIN pessoa_juridica pj ON p.id_pessoa = pj.id_pessoa;
105
106 -- Movimentações de entrada (compra):
107 -- , Produto
108 -- , Fornecedor (pessoa jurídica)
109 -- , Quantidade
110 -- , Preço unitário
111 -- , Valor total (quantidade * preço unitário)
112 SELECT
113     m.id_movimentacao,
114     pr.nome AS produto,
115     pe.nome AS fornecedor,
116     m.quantidade,
117     m.preco_unitario,
118     (m.quantidade * m.preco_unitario) AS valor_total
119 FROM movimentacoes m
120 JOIN produtos pr ON m.id_produto = pr.id_produto
121 JOIN pessoa_juridica pj ON pj.id_pessoa = p.id_pessoa
122 JOIN pessoa pe ON pj.id_pessoa = pe.id_pessoa
123 WHERE m.tipo = 'E';
124
125 -- Movimentações de saída (venda):
126 -- , Produto
127 -- , Comprador (pessoa física)
128 -- , Quantidade
129 -- , Preço unitário
130 -- , Valor total
131 SELECT
132     m.id_movimentacao,
133     pr.nome AS produto,
134     pe.nome AS comprador,
135     m.quantidade,
136     m.preco_unitario,
137     (m.quantidade * m.preco_unitario) AS valor_total
138 FROM movimentacoes m
139 JOIN produtos pr ON m.id_produto = pr.id_produto

```

Data Output Messages Notifications

id_pessoa	nome	endereco	telefone	email	cnpj
integer	character varying (100)	character varying (150)	character varying (20)	character varying (100)	character varying (18)
1	2 Empresa XYZ Ltda	Av. Central, 5000	(11) 99978-5432	contato@empresa.com	12.345.678/0001-99



Query History

Execute script

```

124
125 -- Movimentações de saída (venda)
126 -- .Produto
127 -- .Comprador (pessoa física)
128 -- .Quantidade
129 -- .Preço unitário
130 -- .Valor total
131
132 SELECT
133     m.id_movimentacao,
134     pr.nome AS produto,
135     pe.nome AS comprador,
136     m.quantidade,
137     m.preco_unitario,
138     (m.quantidade * m.preco_unitario) AS valor_total
139 FROM movimentacoes m
140 JOIN produtos pr ON m.id_produto = pr.id_produto
141 JOIN pessoa_fisica pf ON pf.id_pessoa = m.id_pessoa
142 WHERE m.tipo = 'S';
143
144 -- Valor total das entradas agrupadas por produto:
145 SELECT
146     pr.nome AS produto,
147     SUM(m.quantidade * m.preco_unitario) AS valor_total_entrada
148 FROM movimentacoes m
149 JOIN produtos pr ON m.id_produto = pr.id_produto
150 WHERE m.tipo = 'E';
151 GROUP BY pr.nome;
152
153 -- Valor total das saídas agrupadas por produto:
154 SELECT
155     pr.nome AS produto,
156     SUM(m.quantidade * m.preco_unitario) AS valor_total_saida
157 FROM movimentacoes m
158 JOIN produtos pr ON m.id_produto = pr.id_produto
159 WHERE m.tipo = 'S';
160 GROUP BY pr.nome;
161
162 -- Operadores que não efetuaram movimentações de entrada (compra):
163 SELECT u.id_usuario, u.login

```

Data Output Messages Notifications

SQL

id_movimentacao	produto	comprador	quantidade	preco_unitario	valor_total
integer	character varying (100)	character varying (100)	integer	numeric (10,2)	numeric
1	2 banana	João Silva	30	5.00	150.00
2	4 laranja	João Silva	200	2.00	400.00
3	5 manga	João Silva	500	4.00	2000.00



Query Query History

Execute script F3

```
143
144 -- Valor total das entradas agrupadas por produto:
145 SELECT
146     pr.nome AS produto,
147     SUM(m.quantidade * m.preco_unitario) AS valor_total_entrada
148 FROM movimentacoes m
149 JOIN produtos pr ON m.id_produto = pr.id_produto
150 WHERE m.tipo = 'E'
151 GROUP BY pr.nome;
152
153 -- Valor total das saidas agrupadas por produto:
154 SELECT
155     pr.nome AS produto,
156     SUM(m.quantidade * m.preco_unitario) AS valor_total_saida
157 FROM movimentacoes m
158 JOIN produtos pr ON m.id_produto = pr.id_produto
159 WHERE m.tipo = 'S'
160 GROUP BY pr.nome;
161
162 -- Operadores que não efetuaram movimentações de entrada (compra):
163 SELECT u.id_usuario, u.login
164 FROM usuarios u
165 WHERE u.id_usuario NOT IN (
166     SELECT DISTINCT id_usuario
167     FROM movimentacoes
168     WHERE tipo = 'E'
169 );
170
171 -- Valor total de entrada agrupado por operador:
172 SELECT
173     u.login,
174     SUM(m.quantidade * m.preco_unitario) AS valor_total_entrada
175 FROM movimentacoes m
176 JOIN usuarios u ON m.id_usuario = u.id_usuario
177 WHERE m.tipo = 'E'
178 GROUP BY u.login;
179
180 -- Valor total de saída agrupado por operador:
181 SELECT
182     u.login,
```

Data Output Messages Notifications

	produto character varying (100)	valor_total_entrada numeric
1	banana	400.00
2	laranja	750.00
3	manga	2800.00

Query Query History

Execute script

```
1552
1553 -- Valor total das saídas agrupadas por produto:
1554 SELECT
1555     pr.nome AS produto,
1556     SUM(m.quantidade * m.preco_unitario) AS valor_total_saida
1557 FROM movimentacoes m
1558 JOIN produtos pr ON m.id_produto = pr.id_produto
1559 WHERE m.tipo = 'S'
1560 GROUP BY pr.nome;
1561
1562 -- Operadores que não efetuaram movimentações de entrada (compra):
1563 SELECT u.id_usuario, u.login
1564 FROM usuarios u
1565 WHERE u.id_usuario NOT IN (
1566     SELECT DISTINCT id_usuario
1567     FROM movimentacoes
1568     WHERE tipo = 'E'
1569 );
1570
1571 -- Valor total de entrada agrupado por operador:
1572 SELECT
1573     u.login,
1574     SUM(m.quantidade * m.preco_unitario) AS valor_total_entrada
1575 FROM movimentacoes m
1576 JOIN usuarios u ON m.id_usuario = u.id_usuario
1577 WHERE m.tipo = 'E'
1578 GROUP BY u.login;
1579
1580 -- Valor total de saída agrupado por operador:
1581 SELECT
1582     u.login,
1583     SUM(m.quantidade * m.preco_unitario) AS valor_total_saida
1584 FROM movimentacoes m
1585 JOIN usuarios u ON m.id_usuario = u.id_usuario
1586 WHERE m.tipo = 'S'
1587 GROUP BY u.login;
1588
1589 -- Valor médio de venda por produto (média ponderada):
1590 SELECT
1591     pr.nome AS produto,
```

Data Output Messages Notifications

	produto character varying (100)	valor_total_saida numeric
1	Banana	130.00
2	Laranja	400.00
3	Manga	2000.00

Query Query History

```

161
162 -- Operadores que não efetuaram movimentações de entrada (compra):
163 SELECT u.id_usuario, u.login
164 FROM usuarios u
165 WHERE u.id_usuario NOT IN (
166     SELECT DISTINCT id_usuario
167     FROM movimentacoes
168     WHERE tipo = 'E'
169 );
170
171 -- Valor total de entrada agrupado por operador:
172 SELECT
173     u.login,
174     SUM(m.quantidade * m.preco_unitario) AS valor_total_entrada
175 FROM movimentacoes m
176 JOIN usuarios u ON m.id_usuario = u.id_usuario
177 WHERE m.tipo = 'E'
178 GROUP BY u.login;
179
180 -- Valor total de saída agrupado por operador:
181 SELECT
182     u.login,
183     SUM(m.quantidade * m.preco_unitario) AS valor_total_saida
184 FROM movimentacoes m
185 JOIN usuarios u ON m.id_usuario = u.id_usuario
186 WHERE m.tipo = 'S'
187 GROUP BY u.login;
188
189 -- Valor médio de venda por produto (média ponderada):
190 SELECT
191     pr.nome AS produto,
192     SUM(m.quantidade * m.preco_unitario) / SUM(m.quantidade) AS media_ponderada
193 FROM movimentacoes m
194 JOIN produtos pr ON m.id_produto = pr.id_produto
195 WHERE m.tipo = 'S'
196 GROUP BY pr.nome;
197
198
199
200

```

Data Output Messages Notifications

id_usuario (PK) integer	login character varying (50)
1	op2

```
180 -- Operadores que não efetuaram movimentações de entrada (compra):
181 SELECT u.id_usuario, u.login
182 FROM usuarios u
183 WHERE u.id_usuario NOT IN (
184     SELECT DISTINCT id_usuario
185     FROM movimentacoes
186     WHERE tipo = 'E'
187 );
188
189 -- Valor total da entrada agrupado por operador:
190 SELECT
191     u.login,
192     SUM(m.quantidade * m.preco_unitario) AS valor_total_entrada
193 FROM movimentacoes m
194 JOIN usuarios u ON m.id_usuario = u.id_usuario
195 WHERE m.tipo = 'E'
196 GROUP BY u.login;
197
198 -- Valor total de saída agrupado por operador:
199 SELECT
200     u.login,
201     SUM(m.quantidade * m.preco_unitario) AS valor_total_saida
202 FROM movimentacoes m
203 JOIN usuarios u ON m.id_usuario = u.id_usuario
204 WHERE m.tipo = 'S'
205 GROUP BY u.login;
206
207 -- Valor médio do valor por produto (média ponderada):
208 SELECT
209     pr.nome AS produto,
210     SUM(m.quantidade * m.preco_unitario) / SUM(m.quantidade) AS media_peso
211 FROM movimentacoes m
212 JOIN produtos pr ON m.id_produto = pr.id_produto
213 WHERE m.tipo = 'E'
214 GROUP BY pr.nome;
```

Data Output Messages Notifications

id_usuario	login
1	adm2

Queries Query History

```

171 -- Valor total de entrada agrupado por operador:
172 SELECT
173     u.login,
174     SUM(m.quantidade * m.preco_unitario) AS valor_total_entrada
175 FROM movimentacoes m
176 JOIN usuarios u ON m.id_usuario = u.id_usuario
177 WHERE m.tipo = 'E'
178 GROUP BY u.login;
179
180 -- Valor total de saída agrupado por operador:
181 SELECT
182     u.login,
183     SUM(m.quantidade * m.preco_unitario) AS valor_total_saida
184 FROM movimentacoes m
185 JOIN usuarios u ON m.id_usuario = u.id_usuario
186 WHERE m.tipo = 'S'
187 GROUP BY u.login;
188
189 -- Valor médio de venda por produto (média ponderada):
190 SELECT
191     pr.nome AS produto,
192     SUM(m.quantidade * m.preco_unitario) / SUM(m.quantidade) AS media_ponderada_venda
193 FROM movimentacoes m
194 JOIN produtos pr ON m.id_produto = pr.id_produto
195 WHERE m.tipo = 'V'
196 GROUP BY pr.nome;
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211

```

Data Output Messages Notifications

logs	valor_total_entrada
(varchar(32))	numeric
cel	2830.00

Query Query History

Execute script (F5)

```

179 -- Valor total de saída agrupado por operador:
180
181 SELECT
182     u.login,
183     SUM(m.quantidade * m.preco_unitario) AS valor_total_saida
184 FROM movimentacoes m
185 JOIN usuarios u ON m.id_usuario = u.id_usuario
186 WHERE m.tipo = 'S'
187 GROUP BY u.login;
188
189 -- Valor médio de venda por produto (média ponderada):
190 SELECT
191     pr.nome AS produto,
192     SUM(m.quantidade * m.preco_unitario) / SUM(m.quantidade) AS media_ponderada_venda
193 FROM movimentacoes m
194 JOIN produtos pr ON m.id_produto = pr.id_produto
195 WHERE m.tipo = 'S'
196 GROUP BY pr.nome;
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218

```

Data Output Messages Notifications

logs	valor_total_saida
character varying (50)	numeric
op2	2550.00

Query Query History

Execute script (F5)

```

185 JOIN usuarios u ON m.id_usuario = u.id_usuario
186 WHERE m.tipo = 'S'
187 GROUP BY u.login;
188
189 -- Valor médio de venda por produto (média ponderada):
190 SELECT
191     pr.nome AS produto,
192     SUM(m.quantidade * m.preco_unitario) / SUM(m.quantidade) AS media_ponderada_venda
193 FROM movimentacoes m
194 JOIN produtos pr ON m.id_produto = pr.id_produto
195 WHERE m.tipo = 'S'
196 GROUP BY pr.nome;
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224

```

Data Output Messages Notifications

produto	media_ponderada_venda
character varying (100)	numeric
1 banana	3.0000000000000000
2 laranja	2.0000000000000000
3 manga	4.0000000000000000

#### **Análise e conclusão:**

##### **Quais as diferenças no uso de sequence e identity?**

As diferenças entre o uso de SEQUENCE e IDENTITY no SQL Server estão relacionadas à forma como os valores únicos são gerados para identificadores (como chaves primárias). Ambos os recursos têm como objetivo fornecer valores incrementais automaticamente, mas eles funcionam de maneiras diferentes e são adequados para cenários distintos.

##### **Qual a importância das chaves estrangeiras para a consistência do banco?**

As chaves estrangeiras (ou foreign keys ) desempenham um papel fundamental na garantia da consistência e integridade dos dados em um banco de dados relacional. Elas estabelecem relacionamentos entre tabelas, assegurando que os dados armazenados sejam válidos e consistentes com as regras de negócio definidas.

##### **Quais operadores do SQL pertencem à álgebra relacional e quais são definidos no cálculo relacional?**

A álgebra relacional e o cálculo relacional são duas abordagens formais para consultar e manipular dados em bancos de dados relacionais. Embora ambos sejam equivalentes em termos de poder expressivo (ou seja, qualquer consulta que pode ser escrita em uma pode ser traduzida para a outra), eles diferem na forma como representam as operações.

#### **Análise e conclusão:**

##### **Como é feito o agrupamento em consultas, e qual requisito é obrigatório?**

O agrupamento em consultas SQL é realizado usando a cláusula GROUP BY. Ele é usado para organizar linhas de uma tabela em grupos com base nos valores de uma ou mais colunas. O agrupamento é especialmente útil quando você deseja aplicar funções de agregação (como COUNT, SUM, AVG, MIN, MAX) a cada grupo separadamente.