Relações em Banco de Dados Relacional

Em um banco de dados relacional, os dados são armazenados em tabelas e essas tabelas podem se relacionar entre si de diferentes formas. As três principais formas de relacionamento são:

1. Relação 1:1 (um para um)
   - Um registro em uma tabela está relacionado com no máximo um registro em outra tabela.
   - Exemplo: Pessoa e CPF. Cada pessoa tem um único CPF e cada CPF pertence a uma única pessoa.

2. Relação 1:N (um para muitos)
   - Um registro em uma tabela pode estar relacionado com vários registros em outra tabela.
   - Exemplo: Cliente e Pedido. Um cliente pode fazer vários pedidos, mas cada pedido pertence a um único cliente.

3. Relação N:N (muitos para muitos)
   - Vários registros de uma tabela podem estar relacionados com vários registros de outra.
   - Exemplo: Alunos e Disciplinas. Um aluno pode se matricular em várias disciplinas e uma disciplina pode ter vários alunos.
   - Requer uma tabela intermediária (de associação).

Exemplo de Tabelas com Relacionamento

Exemplo: Sistema de Cursos

Tabelas:
- alunos (id, nome, email)
- cursos (id, nome, carga_horaria)
- matriculas (id_aluno, id_curso, data_matricula)

Relacionamentos:
- alunos <-> matriculas: 1:N
- cursos <-> matriculas: 1:N
- alunos <-> cursos: N:N (via tabela matriculas)

Código SQL:

```
CREATE TABLE alunos (
    id INT PRIMARY KEY,
    nome VARCHAR(100),
    email VARCHAR(100)
);

CREATE TABLE cursos (
    id INT PRIMARY KEY,
    nome VARCHAR(100),
    carga_horaria INT
);

CREATE TABLE matriculas (
    id_aluno INT,
    id_curso INT,
    data_matricula DATE,
    PRIMARY KEY (id_aluno, id_curso),
    FOREIGN KEY (id_aluno) REFERENCES alunos(id),
    FOREIGN KEY (id_curso) REFERENCES cursos(id)
);
```

JOINS em SQL

JOINs permitem consultar dados de várias tabelas relacionadas.

1. INNER JOIN: retorna registros que têm correspondência em ambas as tabelas.
2. LEFT JOIN: retorna todos os registros da tabela da esquerda e os correspondentes da direita (ou NULL).
3. RIGHT JOIN: retorna todos os registros da tabela da direita e os correspondentes da esquerda (ou NULL).

Exemplos:

```
-- INNER JOIN
SELECT a.nome, c.nome
FROM alunos a
INNER JOIN matriculas m ON a.id = m.id_aluno
INNER JOIN cursos c ON c.id = m.id_curso;

-- LEFT JOIN
SELECT a.nome, c.nome
FROM alunos a
LEFT JOIN matriculas m ON a.id = m.id_aluno
LEFT JOIN cursos c ON c.id = m.id_curso;

-- RIGHT JOIN
SELECT c.nome, a.nome
FROM cursos c
RIGHT JOIN matriculas m ON c.id = m.id_curso
RIGHT JOIN alunos a ON a.id = m.id_aluno;
```

Exemplos de SELECT com JOIN

1. Quais alunos estão matriculados em quais cursos?

```
SELECT a.nome AS aluno, c.nome AS curso
FROM alunos a
JOIN matriculas m ON a.id = m.id_aluno
JOIN cursos c ON c.id = m.id_curso;
```

2. Quais alunos não estão matriculados em nenhum curso?

```
SELECT a.nome
FROM alunos a
LEFT JOIN matriculas m ON a.id = m.id_aluno
WHERE m.id_curso IS NULL;
```

3. Quais cursos não possuem nenhum aluno matriculado?

```
SELECT c.nome
```

```
FROM cursos c
LEFT JOIN matriculas m ON c.id = m.id_curso
WHERE m.id_aluno IS NULL;

4. Mostrar todos os cursos e os alunos (mesmo que estejam vazios)
SELECT c.nome AS curso, a.nome AS aluno
FROM cursos c
LEFT JOIN matriculas m ON c.id = m.id_curso
LEFT JOIN alunos a ON a.id = m.id_aluno;

5. Mostrar todos os alunos e seus cursos (ou NULL)
SELECT a.nome AS aluno, c.nome AS curso
FROM alunos a
RIGHT JOIN matriculas m ON a.id = m.id_aluno
RIGHT JOIN cursos c ON c.id = m.id_curso;

6. Mostrar todas as combinações possíveis (cuidado!)
SELECT *
FROM alunos a
CROSS JOIN cursos c;
```

Situação para Exercício de Modelagem e JOINs

Contexto: Loja de Vendas Online

Crie as tabelas a seguir com os seguintes campos:

1. clientes (id, nome, email)
2. enderecos (id, id_cliente, rua, numero, cidade, estado)
3. produtos (id, nome, preco)
4. pedidos (id, id_cliente, data_pedido)
5. itens_pedido (id_pedido, id_produto, quantidade)
6. entregas (id, id_pedido, id_funcionario, data_entrega)
7. funcionarios (id, nome, cargo)

Regras implícitas:

- Clientes podem ter vários endereços
- Um cliente pode fazer vários pedidos
- Um pedido pode ter vários produtos e um produto pode estar em vários pedidos (tabela de associação: itens_pedido)
- Cada pedido tem uma entrega (um para um)
- Um funcionário pode fazer várias entregas

Crie SELECTs para responder:

1. Listar pedidos com nome do cliente e data.
2. Mostrar todos os produtos de um pedido.
3. Listar os pedidos entregues por um funcionário específico.
4. Mostrar todos os pedidos e seus respectivos endereços de entrega.
5. Listar os clientes com seus endereços.
6. Mostrar os produtos que nunca foram pedidos.

Gabarito - Relacionamentos entre Tabelas

1. clientes <-> enderecos -> 1:N
2. clientes <-> pedidos -> 1:N
3. pedidos <-> itens_pedido <-> produtos -> N:N
4. pedidos <-> entregas -> 1:1
5. funcionarios <-> entregas -> 1:N