Guia Completo: Subqueries e JOINs em SQL

Baseado em fontes atualizadas da internet (2024-2025)

🎯 PARTE 1: SUBQUERIES (SUBCONSULTAS)

O que são Subqueries?

Subqueries (também conhecidas como **subconsultas**, **inner queries** ou **nested queries**) são instruções SELECT dentro de outras instruções SQL. Elas permitem realizar operações em múltiplas etapas que seriam extremamente complicadas ou impossíveis de fazer de outra forma.

> **Fonte:** Mode Analytics, DevMedia, GeeksforGeeks

🔍 Conceito Fundamental

Uma subquery é executada **primeiro** pelo banco de dados, e seu resultado é usado pela consulta externa (outer query). É como resolver um problema matemático complexo: você resolve as partes entre parênteses primeiro!

SQL

```sql
-- Estrutura básica
SELECT coluna1, coluna2
FROM tabela1
WHERE coluna3 = (SELECT coluna4 FROM tabela2 WHERE condição);
```

📍 Onde Usar Subqueries

1. Na cláusula WHERE (Mais comum)

SQL

```sql
-- Encontrar livros mais caros que a média
SELECT titulo, preco
FROM Livros
WHERE preco > (SELECT AVG(preco) FROM Livros);
```

2. Na cláusula FROM (Como uma tabela temporária)

SQL

```sql
-- Usar subquery como fonte de dados
SELECT sub.genero, sub.total_livros
FROM (
    SELECT genero, COUNT(*) AS total_livros
    FROM Livros
    GROUP BY genero
) sub
WHERE sub.total_livros > 5;
```

3. Na cláusula SELECT (Para cálculos)

SQL

```sql
-- Mostrar cada livro e a média geral de preços
SELECT titulo, preco,
       (SELECT AVG(preco) FROM Livros) AS media_geral
FROM Livros;
```

🎯 Tipos de Subqueries

1. Subquery Escalar (Retorna 1 valor)

SQL

```sql
-- Livros da editora mais cara
SELECT titulo
FROM Livros
WHERE id_editora = (
    SELECT id_editora
    FROM Livros
    GROUP BY id_editora
    ORDER BY AVG(preco) DESC
    LIMIT 1
);
```

2. Subquery de Múltiplas Linhas (Retorna vários valores)

SQL

```sql
-- Usuários que pegaram livros de ficção
SELECT nome_usuario
FROM Usuarios
WHERE id_usuario IN (
    SELECT DISTINCT id_usuario
    FROM Emprestimos e
    JOIN Livros l ON e.id_livro = l.id_livro
    WHERE l.genero = 'Ficção'
);
```

3. Subquery Correlacionada (Depende da consulta externa)

SQL

```sql
-- Livros mais caros que a média da sua editora
SELECT titulo, preco
FROM Livros l1
WHERE preco > (
    SELECT AVG(preco)
    FROM Livros l2
    WHERE l2.id_editora = l1.id_editora
);
```

💡 Dicas para Dominar Subqueries

1. **Sempre teste a subquery isoladamente primeiro**
2. **Use aliases para clareza** (`sub` , `inner_query` , etc.)
3. **Indente o código** para facilitar a leitura
4. **Subqueries na cláusula WHERE não precisam de alias**
5. **Subqueries na cláusula FROM SEMPRE precisam de alias**

🔗 PARTE 2: JOINs (JUNÇÕES)

O que são JOINs?

JOINs são comandos que permitem combinar dados de duas ou mais tabelas baseado em uma relação entre elas. É como "colar" informações de diferentes tabelas que têm algo em comum.

> **Fonte:** GeeksforGeeks, W3Schools, Mode Analytics

🎯 Quando Usar Cada Tipo de JOIN

1. INNER JOIN - "Só o que existe em ambas"

Use quando: Quiser apenas registros que têm correspondência em ambas as tabelas.

SQL

```sql
-- Mostrar apenas livros que foram emprestados
SELECT l.titulo, u.nome_usuario, e.data_emprestimo
FROM Livros l
INNER JOIN Emprestimos e ON l.id_livro = e.id_livro
INNER JOIN Usuarios u ON e.id_usuario = u.id_usuario;
```

Resultado: Só livros que foram emprestados + dados do usuário

2. LEFT JOIN - "Tudo da esquerda + o que combina da direita"

Use quando: Quiser todos os registros da primeira tabela, mesmo que não tenham correspondência na segunda.

SQL

```sql
-- Mostrar TODOS os livros, emprestados ou não
SELECT l.titulo, e.data_emprestimo
FROM Livros l
LEFT JOIN Emprestimos e ON l.id_livro = e.id_livro;
```

Resultado: Todos os livros + data de empréstimo (NULL se nunca foi emprestado)

3. RIGHT JOIN - "Tudo da direita + o que combina da esquerda"

Use quando: Quiser todos os registros da segunda tabela, mesmo que não tenham correspondência na primeira.

SQL

```sql
-- Mostrar TODOS os empréstimos, mesmo de livros que foram deletados
SELECT l.titulo, e.data_emprestimo
FROM Livros l
RIGHT JOIN Emprestimos e ON l.id_livro = e.id_livro;
```

Resultado: Todos os empréstimos + título do livro (NULL se livro foi deletado)

4. FULL OUTER JOIN - "Tudo de ambas as tabelas"

Use quando: Quiser todos os registros de ambas as tabelas, combinados quando possível.

SQL

```sql
-- Mostrar TODOS os livros E TODOS os empréstimos
SELECT l.titulo, e.data_emprestimo
FROM Livros l
FULL OUTER JOIN Emprestimos e ON l.id_livro = e.id_livro;
```

Resultado: Todos os livros + todos os empréstimos, com NULLs onde não há correspondência

🧠 Como Pensar em JOINs

Analogia da Festa:

- **INNER JOIN:** Só casais que vieram juntos
- **LEFT JOIN:** Todos os homens + suas esposas (se vieram)
- **RIGHT JOIN:** Todas as mulheres + seus maridos (se vieram)
- **FULL OUTER JOIN:** Todos que estão na festa, casais ou solteiros

🔄 Relacionando Tabelas - Passo a Passo

Passo 1: Identifique a Relação

SQL

```sql
-- Livros ⟷ Editoras (através de id_editora)
-- Emprestimos ⟷ Livros (através de id_livro)
-- Emprestimos ⟷ Usuarios (através de id_usuario)
```

Passo 2: Escolha o Tipo de JOIN

- Quer só dados que existem em ambas? → **INNER JOIN**
- Quer preservar todos da primeira tabela? → **LEFT JOIN**
- Quer preservar todos da segunda tabela? → **RIGHT JOIN**
- Quer preservar todos de ambas? → **FULL OUTER JOIN**

Passo 3: Escreva a Condição ON

SQL

```sql
-- SEMPRE: tabela1.chave_estrangeira = tabela2.chave_primaria
ON Livros.id_editora = Editoras.id_editora
ON Emprestimos.id_livro = Livros.id_livro
```

🎯 Exemplos Práticos Progressivos

Nível 1: JOIN Simples

SQL

```sql
-- Livros com nome da editora
SELECT l.titulo, e.nome_editora
FROM Livros l
INNER JOIN Editoras e ON l.id_editora = e.id_editora;
```

Nível 2: Múltiplos JOINs

SQL

```sql
-- Empréstimos completos: usuário, livro, editora
SELECT u.nome_usuario, l.titulo, e.nome_editora, emp.data_emprestimo
FROM Emprestimos emp
INNER JOIN Usuarios u ON emp.id_usuario = u.id_usuario
INNER JOIN Livros l ON emp.id_livro = l.id_livro
INNER JOIN Editoras e ON l.id_editora = e.id_editora;
```

Nível 3: JOINs com Agregação

SQL

```sql
-- Quantos livros cada editora tem emprestados
SELECT e.nome_editora, COUNT(emp.id_emprestimo) AS total_emprestimos
FROM Editoras e
LEFT JOIN Livros l ON e.id_editora = l.id_editora
LEFT JOIN Emprestimos emp ON l.id_livro = emp.id_livro
GROUP BY e.nome_editora;
```

🆚 SUBQUERIES vs JOINs: Quando Usar Cada Um?

Use SUBQUERIES quando:

- Precisar de um valor específico para comparação
- A lógica for "encontre registros onde X é igual ao resultado de Y"
- Quiser filtrar baseado em agregações de outras tabelas

Use JOINs quando:

- Precisar combinar colunas de múltiplas tabelas
- Quiser mostrar dados relacionados lado a lado
- A performance for crítica (JOINs são geralmente mais rápidos)

Exemplo Comparativo:

Com SUBQUERY:

SQL

```sql
-- Usuários que pegaram livros de "Stephen King"
SELECT nome_usuario
FROM Usuarios
WHERE id_usuario IN (
    SELECT id_usuario
    FROM Emprestimos e
    JOIN Livros l ON e.id_livro = l.id_livro
    WHERE l.autor = 'Stephen King'
);
```

Com JOIN:

SQL

```sql
-- Usuários que pegaram livros de "Stephen King" (com detalhes)
SELECT DISTINCT u.nome_usuario, l.titulo, e.data_emprestimo
FROM Usuarios u
INNER JOIN Emprestimos e ON u.id_usuario = e.id_usuario
INNER JOIN Livros l ON e.id_livro = l.id_livro
WHERE l.autor = 'Stephen King';
```

🚀 Exercícios Práticos para Fixação

Subqueries:

1. Encontre livros mais caros que a média

2. Liste usuários que nunca pegaram livros emprestados
3. Mostre editoras que publicaram livros acima da média de preço

JOINs:

1. Liste todos os livros com nome da editora (INNER JOIN)
2. Mostre todos os usuários e seus empréstimos, se houver (LEFT JOIN)
3. Crie um relatório completo: usuário, livro, editora, data (múltiplos JOINs)

💡 Dicas Finais para a Prova

1. **Sempre desenhe** o relacionamento entre tabelas antes de escrever
2. **Teste subqueries isoladamente** antes de usar na consulta principal
3. **Use aliases** para deixar o código mais legível
4. **Lembre-se:** LEFT JOIN preserva a tabela da esquerda, RIGHT JOIN preserva a da direita
5. **Na dúvida entre SUBQUERY e JOIN:** se precisar de colunas de múltiplas tabelas, use JOIN

Fontes consultadas:

- Mode Analytics SQL Tutorial (2024)
- GeeksforGeeks SQL Joins Guide (2025)
- DevMedia Subquery Tutorial
- W3Schools SQL Reference