Professores
Celso Rodrigo Giusti
Daniel Manoel Filho
Marlon P. F. Rodrigues

Comandos DML
JOINS e Agrupamentos

SENAI

BANCO DE DADOS

Por que Consultas Avançadas?

Até agora, fizemos consultas em **uma tabela** de cada vez (ex: SELECT * FROM tbl_livro).

Mas os dados de um sistema são **relacionais**.
O nome do livro está na tbl_livro.
O nome do autor está na tbl_autor.
A ligação entre eles está na tbl_autor_livro.

Como criar uma consulta que mostre o **título do livro E o nome do seu autor**?

Para isso, usamos JOINS (Associações) e GROUP BY (Agrupamentos).

BANCO DE DADOS

Agrupamento (GROUP BY)

O *GROUP BY* é usado em conjunto com **Funções de Agregação** (como *COUNT, AVG, SUM*) que vimos na última aula.

Ele "comprime" todas as linhas que têm o mesmo valor em uma coluna específica, permitindo que a função de agregação seja aplicada a *cada grupo* separadamente.

```sql
SELECT editora, COUNT(isbn) AS quantidade_livros
FROM tbl_livro
GROUP BY editora;
```

BANCO DE DADOS

Filtrando Grupos: O HAVING

Já sabemos usar o *WHERE* para filtrar linhas. Mas o que fazer se quisermos filtrar o *resultado do grupo*?

WHERE filtra *antes* do agrupamento.
HAVING filtra *depois* do agrupamento.

Exemplo: Mostrar apenas as editoras que têm 2 ou mais livros cadastrados.

```sql
SELECT editora, COUNT(isbn) AS quantidade_livros
FROM tbl_livro
GROUP BY editora
HAVING COUNT(isbn) >= 2;
```

BANCO DE DADOS

União (UNION)

O operador *UNION* combina o resultado de dois ou mais *SELECT* em um único conjunto de resultados.
Ele "empilha" os resultados verticalmente.

Regras:
Os *SELECT* devem ter o mesmo número de colunas.
As colunas devem estar na mesma ordem e ser de tipos de dados compatíveis.

Exemplo: Criar uma lista única de "Contatos" (Autores e Membros).

BANCO DE DADOS

União (UNION)

O operador *UNION* combina o resultado de dois ou mais *SELECT* em um único conjunto de resultados.

Ele "er

Regras

Os *SEL*

As colu

Exemp

```
SELECT nome_autor AS nome, 'Autor' AS tipo
FROM tbl_autor

UNION

SELECT nome_membro AS nome, 'Membro' AS tipo
FROM tbl_membro;
```

BANCO DE DADOS

Associações de Tabelas (JOINS)

O JOIN é o comando usado para cruzar dados entre tabelas usando suas chaves (PK/FK).

Nosso objetivo: Buscar o **Título do Livro** e o **Nome do Autor**.

- *tbl_livro* (Tem *titulo_livro* e *isbn*)
- *tbl_autor* (Tem *nome_autor* e *id_autor*)
- *tbl_autor_livro* (A "ponte": tem *isbn* e *id_autor*)

Precisamos "juntar" as três tabelas usando as chaves que elas têm em comum.

BANCO DE DADOS

CROSS JOIN (Produto Cartesiano)

O *CROSS JOIN* acontece quando você junta duas tabelas sem especificar uma condição (*ON* ou *WHERE*).
O resultado é *todas as combinações possíveis* de linhas.

```
SELECT L.titulo_livro, A.nome_autor
FROM tbl_livro L
CROSS JOIN tbl_autor A;
```

Se você tem 5 livros e 4 autores, o resultado terá 5 x 4 = 20 linhas. A maioria delas não fará sentido (ex: "Harry Potter - Machado de Assis").

Lembre-se: Quase sempre que você obtém um *CROSS JOIN*, é por acidente (esqueceu o *WHERE* ou *ON*).

BANCO DE DADOS

INNER JOIN (A Forma Correta)

O *INNER JOIN* (Associação Interna) é o padrão de mercado. Ele retorna **apenas** as linhas que têm correspondência em **ambas** as tabelas.

A condição de ligação é feita na cláusula *ON*.

Exemplo (2 Tabelas): Juntando Livro com a tabela "ponte".

```
SELECT L.titulo_livro, AL.id_autor
FROM tbl_livro L
INNER JOIN tbl_autor_livro AL
    ON L.isbn = AL.isbn;
```

BANCO DE DADOS

INNER JOIN (Juntando 3 Tabelas)

Para buscar o Título e o Nome do Autor, encadeamos os *JOINs*:

```
SELECT L.titulo_livro, A.nome_autor
FROM tbl_livro L

/* Passo 1: Junte Livro com a tabela "ponte" */
INNER JOIN tbl_autor_livro AL
    ON L.isbn = AL.isbn

/* Passo 2: Junte o resultado com a tabela Autor */
INNER JOIN tbl_autor A
    ON AL.id_autor = A.id_autor;
```

BANCO DE DADOS

OUTER JOIN (LEFT e RIGHT)

O *INNER JOIN* só mostra quem tem correspondência. Mas e se eu quiser ver...

...*todos* os livros, mesmo os que não têm autor cadastrado?
...*todos* os autores, mesmo os que não têm livro cadastrado?

Para isso, usamos *OUTER JOIN* (Associação Externa).

LEFT JOIN: Retorna **TUDO** da tabela da **esquerda** e o que encontrar na da direita.
RIGHT JOIN: Retorna **TUDO** da tabela da **direita** e o que encontrar na da esquerda.

BANCO DE DADOS

Subconsultas (Subqueries)

Uma Subconsulta (ou Subquery) é um *SELECT dentro de outro SELECT (ou INSERT, UPDATE, DELETE).*

Elas são usadas quando precisamos de um valor que depende de outra consulta.

Exemplo: Mostrar os títulos dos livros escritos por autores brasileiros. *Primeiro, precisamos dos IDs dos autores brasileiros: (SELECT id_autor FROM tbl_autor WHERE nacionalidade = 'Brasileira')*

Agora, usamos isso como filtro:

BANCO DE DADOS

Subconsultas (Subqueries)

Uma Subconsulta (ou Subquery) é um *SELECT dentro de outro SELECT (ou INS*

Elas
cons

Exe
prec
WH

Ago

```
SELECT titulo_livro
FROM tbl_livro
WHERE isbn IN (
    SELECT isbn FROM tbl_autor_livro WHERE id_autor IN (
        SELECT id_autor FROM tbl_autor
        WHERE nacionalidade = 'Brasileira'
    )
);
```

BANCO DE DADOS

Subconsultas com IN e NOT IN

O IN é a forma mais comum de usar subqueries. Ele verifica se um valor *está na lista* retornada pela subconsulta.

Exemplo: Selecionar o nome de todos os autores que **JÁ** têm um livro cadastrado.

```
SELECT nome_autor
FROM tbl_autor
WHERE id_autor IN (
    /* Subquery: Retorna a lista de IDs de autores
       que estão na tabela "ponte"
    */
    SELECT DISTINCT id_autor FROM tbl_autor_livro
);
```

BANCO DE DADOS

Subconsultas com EXISTS

O EXISTS é usado para verificar se a subconsulta retorna *qualquer* linha. Se retornar pelo menos uma, a condição é Verdadeira.

É considerado mais rápido que o IN em grandes volumes de dados, pois ele para de procurar assim que encontra *um* resultado.

Exemplo: Selecionar autores para os quais *EXISTE* um livro.

```sql
SELECT nome_autor
FROM tbl_autor A
WHERE EXISTS (
    SELECT 1 FROM tbl_autor_livro AL
    WHERE AL.id_autor = A.id_autor
);
```

BANCO DE DADOS

Subconsultas com ANY e ALL

ANY e *ALL* são usados com operadores relacionais (=, >, <) após uma subconsulta que retorna uma lista.

ANY: Maior que *pelo menos um* da lista (ou seja, maior que o MÍNIMO).
ALL: Maior que *todos* da lista (ou seja, maior que o MÁXIMO).

Exemplo: Buscar livros publicados *antes* de *qualquer* livro da editora 'Aleph'.

```sql
SELECT titulo_livro, ano_publicacao
FROM tbl_livro
WHERE ano_publicacao < ANY (
    SELECT ano_publicacao FROM tbl_livro
    WHERE editora = 'Aleph'
);
```

BANCO DE DADOS

RESUMO DA AULA

SENAI

✅ **GROUP BY** = Agrupa linhas com valores iguais para que funções de agregação (COUNT, AVG) atuem em cada grupo.
✅ **HAVING** = Filtra o resultado *depois* que os grupos são formados (diferente do WHERE, que filtra *antes*).
✅ **UNION** = "Empilha" resultados de dois SELECTs (as colunas devem ser compatíveis).
✅ **INNER JOIN** = (Associação Interna) Mostra apenas linhas que têm correspondência em *ambas* as tabelas (usa ON para a condição).
✅ **LEFT JOIN** = (Associação Externa) Mostra *todas* as linhas da tabela da ESQUERDA, e preenche com NULL onde não há correspondência na direita.
✅ **Subquery** = Um SELECT dentro de outro SELECT. Útil para filtros complexos baseados em listas (IN, EXISTS) ou comparações (ANY, ALL).

BANCO DE DADOS

Exercícios (Setup)

Para os exercícios de JOIN e GROUP BY fazerem sentido, precisamos de dados mais complexos. Execute os comandos abaixo:

```
/* 1. Cadastre um autor que ainda não tem livro */
INSERT INTO tbl_autor (nome_autor, nacionalidade)
VALUES ('Frank Herbert', 'Americano');

/* 2. Cadastre exemplares para nossos livros */
/* (IDs 101 e 102 para o livro '978-85-325-3078-3') */
INSERT INTO tbl_exemplar (id_exemplar, status_exemplar, isbn)
VALUES (101, 'Disponível', '978-85-325-3078-3'),
       (102, 'Emprestado', '978-85-325-3078-3'),
       (103, 'Disponível', '978-85-7126-061-0');

/* 3. Cadastre um empréstimo para o exemplar 102
   para o membro 101 (Ana Silva) */
INSERT INTO tbl_emprestimo (id_emprestimo, data_emprestimo, data_devolucao, data_devolucao_efetiva, id_exemplar, id_membro)
VALUES (501, '2024-10-01', '2024-10-15', NULL, 102, 101);
```

BANCO DE DADOS

Exercícios (Setup)

Exercício 2: GROUP BY
Pergunta: Quantos exemplares (cópias) nós temos de cada livro?
O que fazer: Escreva um SELECT na tbl_exemplar que conta (COUNT) os exemplares, agrupando (GROUP BY) pelo isbn.

Exercício 3: INNER JOIN (O Grande Relatório)
Pergunta: Queremos ver um relatório de empréstimos. Mostre:

- O nome do membro (tbl_membro)
- O título do livro (tbl_livro)
- A data em que o empréstimo vence (tbl_emprestimo)

O que fazer: Você precisará "juntar" 4 tabelas.

- tbl_membro -> tbl_emprestimo (por id_membro)
- tbl_emprestimo -> tbl_exemplar (por id_exemplar)
- tbl_exemplar -> tbl_livro (por isbn)

BANCO DE DADOS

Exercícios (Setup)

Exercício 4: LEFT JOIN (Relatório de "Faltantes")
Pergunta: Queremos ver *todos* os autores e quantos livros (títulos) cada um tem cadastrado em nosso sistema.
O que fazer:

Use tbl_autor como a tabela da **esquerda** (para garantir que todos apareçam).
Use LEFT JOIN para conectar com tbl_autor_livro.
Use COUNT(AL.isbn) para contar os livros e GROUP BY A.nome_autor.

Resultado Esperado: 'Frank Herbert' (que inserimos no Ex 1) deve aparecer na lista com a contagem 0.

Exercício 5: Subquery com IN
Pergunta: Mostre o nome de todos os membros que *atualmente* têm um livro emprestado (ou seja, que estão na tbl_emprestimo com data_devolucao_efetiva IS NULL).
O que fazer:

Escreva um SELECT na tbl_membro.
Use WHERE id_membro IN (...).
A subquery (...) deve buscar os id_membro da tbl_emprestimo que atendem à condição data_devolucao_efetiva IS NULL.

BANCO DE DADOS

Referências

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de banco de dados**. 6. ed. São Paulo: Pearson Addison Wesley, 2011.

MACHADO, Felipe N. R. **Banco de dados**: projeto e implementação. 4. ed. São Paulo: Erica, 2014.

SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL (SENAI). Departamento Regional de São Paulo. **PLANO DE CURSO**: Técnico em Desenvolvimento de Sistemas. São Paulo: SENAI-SP, 2023.

Escola SENAI "Italo Bologna"

Av. Goiás, 139 – Itu/SP

Telefone
(11) 2396-1999

Instagram
@senai.itu

Facebook
/senai.itu

Site
https://sp.senai.br/unidade/itu/