INSTITUTO FEDERAL
Rio Grande do Sul
Canoas

BANCO DE DADOS

Subqueries

Cristiane Silva Garcia
2022

O que vimos até agora?

- Como criar tabelas: `CREATE TABLE`
- Como inserir, alterar e excluir registros: `INSERT`, `UPDATE` e `DELETE`
- Como fazer consultas simples: `SELECT-FROM-WHERE`
- Operadores de comparação: `<`, `<=`, `>`, `>=`, `=` e `<>`.

GitHub
cristianesgarcia/
subqueries

Exercícios da aula anterior: dúvidas?

BANCO DE DADOS

1 Introdução

INTRODUÇÃO

Existem casos em que precisamos que os valores armazenados no banco de dados sejam buscados e depois usados em uma condição de comparação.

Essas consultas podem ser formuladas usando **consultas aninhadas**, também conhecidas como subconsultas ou subqueries.

Uma **subquery** é uma expressão `SELECT-FROM-WHERE` aninhada dentro de outra query.

A consulta mais “de fora” é chamada de **consulta externa** e a consulta mais “de dentro” é chamada de **consulta interna**.

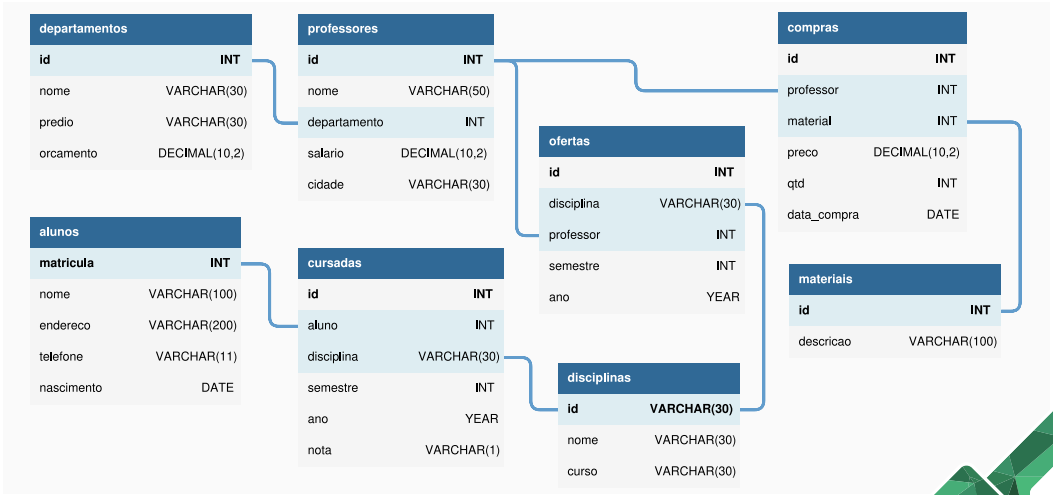
INTRODUÇÃO

As subqueries são bastante utilizadas na cláusula `WHERE` para:

- realizar testes de pertinência;
- fazer comparação de conjuntos;
- verificar se a relação retornada pela subquery é vazia.

Contudo, as subqueries podem ser empregadas em outras cláusulas como, por exemplo, nas cláusulas `FROM` e `SELECT`.

INTRODUÇÃO
Banco de dados da aula



INTRODUÇÃO
Exemplo

Exemplo: Encontrar as disciplinas ofertadas em 2021/2 **e** em 2020/1.

Sugestões?

INTRODUÇÃO
Exemplo

Exemplo: Encontrar as disciplinas ofertadas em 2021/2 **e** em 2020/1.

Podemos fazer as duas consultas:

Arquivo: consulta1.sql

```sql
SELECT DISTINCT disciplina
FROM ofertas
WHERE semestre = 1
AND ano = 2020;
```

Arquivo: consulta2.sql

```sql
SELECT DISTINCT disciplina
FROM ofertas
WHERE semestre = 2
AND ano = 2021;
```

INTRODUÇÃO
Exemplo

Exemplo: Encontrar as disciplinas ofertadas em 2021/2 **e** em 2020/1.

Retorno das duas consultas:

```
+------------+
| disciplina |
+------------+
| HIS01      |
| INF01      |
+------------+
```

```
+------------+
| disciplina |
+------------+
| FIN02      |
| HIS01      |
| HIS02      |
| INF01      |
+------------+
```

Como comparamos os resultados?

INTRODUÇÃO
Exemplo

Exemplo: Encontrar as disciplinas ofertadas em 2021/2 **e** em 2020/1.

Retorno das duas consultas:

```
+------------+
| disciplina |
+------------+
| HIS01      |
| INF01      |
+------------+
```

```
+------------+
| disciplina |
+------------+
| FIN02      |
| HIS01      |
| HIS02      |
| INF01      |
+------------+
```

Como comparamos os resultados?

Podemos comparar no software, aplicativo, página PHP, por exemplo.

INTRODUÇÃO
Exemplo

Exemplo: Encontrar as disciplinas ofertadas em 2021/2 **e** em 2020/1.

Retorno das duas consultas:

```
+------------+
| disciplina |
+------------+
| HIS01      |
| INF01      |
+------------+
```

```
+------------+
| disciplina |
+------------+
| FIN02      |
| HIS01      |
| HIS02      |
| INF01      |
+------------+
```

Como comparamos os resultados?

Podemos comparar diretamente na consulta usando subqueries.

2 Subqueries na Cláusula WHERE

SUBQUERIES NA CLÁUSULA WHERE
Estrutura básica

```
SELECT colunas
FROM tabelas
WHERE colunas operador (SELECT colunas
                        FROM tabelas
                        WHERE ...)
```

Observe que:

- Podemos ter mais de uma subquery na mesma query externa, basta adicionarmos operadores `AND`, `OR`, etc.
- Podemos ter vários níveis de subqueries

SUBQUERIES NA CLÁUSULA WHERE
Pertinência

Operador `IN`

Compara um valor v com um conjunto (ou multiconjunto) de valores S e avalia como verdadeiro se v for um dos elementos de S.

Geralmente empregado em subqueries que retornam múltiplas linhas e/ou colunas.

SUBQUERIES NA CLÁUSULA WHERE
Pertinência

Exemplo: Encontrar as disciplinas ofertadas em 2021/2 **e** em 2020/1.

Arquivo: exemploIn.sql

```sql
SELECT DISTINCT disciplina
FROM ofertas
WHERE semestre = 2
AND ano = 2021
AND disciplina IN (SELECT disciplina
                   FROM ofertas
                   WHERE semestre = 1
                   AND ano = 2020);
```

```
+------------+
| disciplina |
+------------+
| INF01      |
| HIS01      |
+------------+
```

SUBQUERIES NA CLÁUSULA WHERE
Pertinência

Operador `NOT IN`

Compara um valor *v* com um conjunto (ou multiconjunto) de valores *S* e avalia como verdadeiro se *v* não for um dos elementos de *S*.

SUBQUERIES NA CLÁUSULA WHERE
Pertinência

Exemplo: Encontrar as disciplinas ofertadas em 2021/2 mas **não** ofertadas em 2020/1.

Arquivo: exemploNotIn.sql

```sql
SELECT DISTINCT disciplina
FROM ofertas
WHERE semestre = 2
AND ano = 2021
AND disciplina NOT IN (SELECT disciplina
                       FROM ofertas
                       WHERE semestre = 1
                       AND ano = 2020);
```

```
+------------+
| disciplina |
+------------+
| FIN02      |
| HIS02      |
+------------+
```

SUBQUERIES NA CLÁUSULA WHERE
Pertinência

Os operadores `IN` e `NOT IN` podem ser usados também com conjuntos enumerados, ou seja, com tuplas escritas *inline*. Por exemplo:

```
SELECT *
FROM ofertas
WHERE disciplina IN ('INF01', 'HIS02');
```

SUBQUERIES NA CLÁUSULA WHERE
Pertinência

Veja que, até então, nós testamos a pertinência a uma relação usando um único atributo. Contudo, poderíamos testar usando múltiplos atributos, no formato de uma tupla.[1]

Exemplo: Encontrar o número de alunos que cursaram disciplinas com o professor cujo ID é 1000.

Arquivo: exemploTupla.sql

```sql
SELECT COUNT(DISTINCT aluno) AS "Quantidade"
FROM cursadas
WHERE (disciplina, semestre, ano)
   IN (SELECT disciplina, semestre, ano
       FROM ofertas
       WHERE professor = 1000);
```

```
+------------+
| Quantidade |
+------------+
|          4 |
+------------+
```

[1] Não funciona em todos os bancos de dados.

SUBQUERIES NA CLÁUSULA WHERE
Comparação de conjuntos

Operador `SOME`

Retorna todos os registros para os quais pelo menos um registro retornado na subquery atende à restrição. Em SQL o operador `SOME` é sinônimo de `ANY`.

Comparações permitidas:

`< SOME`, `<= SOME`, `> SOME`, `>= SOME`, `= SOME` e `<> SOME`.

Note que `= SOME` é o mesmo que `IN`, mas `<> SOME` não é o mesmo que `NOT IN`.

SUBQUERIES NA CLÁUSULA WHERE
Comparação de conjuntos

Exemplo: Todos que recebem mais do que **alguém** do departamento 3.

Arquivo: exemploSome.sql

```sql
SELECT nome, departamento
FROM professores
WHERE salario > SOME (SELECT salario
                      FROM  professores
                      WHERE departamento = 3);
```

```
+-------------+--------------+
| nome        | departamento |
+-------------+--------------+
| Maria Silva |            3 |
| Ana Luz     |            2 |
+-------------+--------------+
```

SUBQUERIES NA CLÁUSULA WHERE
Comparação de conjuntos

Operador `ALL`

Retorna todos os registros para os quais todos os registros retornados na subquery atendem à restrição.

As comparações permitidas são as mesmas de antes:

`< ALL`, `<= ALL`, `> ALL`, `>= ALL`, `= ALL` e `<> ALL`.

Note que `<> ALL` é o mesmo que `NOT IN`, mas `= ALL` não é o mesmo que `IN`.

SUBQUERIES NA CLÁUSULA WHERE
Comparação de conjuntos

Exemplo: Todos que recebem mais do que **qualquer um** do departamento 2.

Arquivo: exemploAll.sql

```
1 SELECT nome, departamento
2 FROM professores
3 WHERE salario > ALL (SELECT salario
4                      FROM  professores
5                      WHERE departamento = 2);
```

```
1 +-------------+--------------+
2 | nome        | departamento |
3 +-------------+--------------+
4 | Maria Silva |            3 |
5 +-------------+--------------+
```

SUBQUERIES NA CLÁUSULA WHERE
Relações vazias

Operador `EXISTS`

Empregado para verificar se o resultado de uma subquery não é vazio (contém ao menos uma tupla). Portanto, o resultado de `EXISTS` é um valor booleano verdadeiro se a subquery retornar ao menos uma tupla, ou falso, se o resultado da subquery for vazio.

Podemos testar se o retorno da subquery é vazio usando o operador `NOT EXISTS`.

SUBQUERIES NA CLÁUSULA WHERE
Relações vazias

Exemplo: Encontrar os professores que ministraram disciplinas no segundo semestre de 2021.

Arquivo: exemploExists.sql

```sql
SELECT nome
FROM professores AS p
WHERE EXISTS (SELECT *
              FROM ofertas AS o
              WHERE o.professor = p.id
              AND o.ano = 2021
              AND o.semestre = 2);
```

```
+------------------+
| nome             |
+------------------+
| Pedro Dias       |
| João dos Santos  |
| Maria Silva      |
| Luiza Meireles   |
| Ana Luz          |
+------------------+
```

Veja que as consultas são **correlacionadas**: a subquery referencia um atributo de uma relação declarada na consulta externa.

3 Subqueries na Cláusula FROM

SUBQUERIES NA CLÁUSULA FROM

Veja que qualquer expressão `SELECT-FROM-WHERE` retorna uma relação como resultado. Portanto, ela pode ser inserida em outra expressão `SELECT-FROM-WHERE` em qualquer lugar que uma relação pode aparecer.

Por conta disso, podemos utilizar uma subquery na cláusula `FROM`.

SUBQUERIES NA CLÁUSULA FROM

Exemplo: Encontrar a média dos salários dos professores (por departamento) cuja média por departamento é maior do que 3000.

Arquivo: exemploSubqueryFrom.sql

```sql
SELECT d.id, FORMAT(d.media,2) AS media
FROM (SELECT departamento AS id
           , avg(salario) AS media
      FROM professores
      GROUP BY departamento) AS d
WHERE d.media > 3000;
```

```
+----+----------+
| id | media    |
+----+----------+
|  2 | 4,400.00 |
|  3 | 4,900.00 |
+----+----------+
```

4 Subqueries Escalares

SUBQUERIES ESCALARES

Uma subquery escalar é uma subquery que retorna apenas uma tupla contendo um único atributo, ou seja, um escalar.

Por isso, tais subqueries podem ocorrer em qualquer lugar onde é permitida uma expressão que retorna um valor. Por exemplo, uma subquery escalar pode aparecer na cláusula `SELECT`.

Veja que uma subquery escalar pode aparecer também na cláusula `WHERE`. Nesse caso, utilizaremos os operadores de comparação que vimos anteriormente: `<`, `<=`, `>`, `>=`, `=` e `<>`.

SUBQUERIES ESCALARES

Exemplo: Encontrar o número de professores em cada departamento.

Arquivo: exemploSubquerySelect.sql

```sql
SELECT d.nome AS Depto,
(SELECT COUNT(*)
 FROM professores AS p
 WHERE p.departamento = d.id) AS Num
FROM departamentos AS d;
```

```
+--------------+------+
| Depto        | Num  |
+--------------+------+
| Finanças     |    2 |
| História     |    2 |
| Informática  |    2 |
+--------------+------+
```

SUBQUERIES ESCALARES

Em alguns casos, uma query pode requerer um cálculo sem precisar fazer referência a uma relação. De modo similar, certas subqueries podem conter uma cláusula `FROM` sem que a query externa necessite de uma cláusula dessas.[2]

Exemplo: Encontrar a razão entre o número total de disciplinas ofertadas e o número total de professores.

Arquivo: exemploEscalar.sql

```sql
SELECT (SELECT COUNT(*) FROM ofertas)
       /
       (SELECT COUNT(*) FROM professores) AS valor;
```

[2] Pode haver diferenças na sintaxe, dependendo do banco de dados.

5 Concluindo

Arquivo: ofertas.txt

disciplina	professor	semestre	ano
FIN01	1001	2	2019
HIS01	1005	2	2019
INF01	1000	1	2019
INF01	1000	2	2019
INF03	1000	1	2019
HIS01	1002	1	2020
HIS01	1005	1	2020
INF01	1000	1	2020
FIN02	1001	2	2021
FIN02	1001	1	2021
HIS01	1005	2	2021
HIS02	1002	2	2021
INF01	1003	2	2021
INF01	1000	2	2021
INF02	1003	1	2021
INF03	1003	1	2021
INF02	1003	1	2022

Arquivo: professores.txt

id	nome	departamento	salario	cidade
1000	Pedro Dias	1	2500.00	Porto Alegre
1001	João dos Santos	2	3500.00	Canoas
1002	Maria Silva	3	5500.00	Guaíba
1003	Luiza Meireles	1	1800.00	Gramado
1004	Wu Zen	3	4300.00	Dois Irmãos
1005	Ana Luz	2	5300.00	Canela

Arquivo: cursadas.txt

id	aluno	disciplina	semestre	ano	nota
1	1000	INF01	1	2019	A
2	1001	INF01	1	2019	D
3	1002	INF01	1	2019	C
4	1003	INF01	1	2019	E
5	1000	FIN01	2	2019	A
6	1000	HIS01	1	2020	B
7	1001	INF01	1	2020	A
8	1003	INF01	1	2020	B
9	1003	INF02	2	2021	C
10	1003	INF03	1	2019	C
11	1000	FIN02	1	2021	D
12	1000	FIN02	2	2021	A

Arquivo: departamentos.txt

```
+----+--------------+---------+------------+
| id | nome         | predio  | orcamento  |
+----+--------------+---------+------------+
|  1 | Informática  | Setor 1 | 1000000.00 |
|  2 | Finanças     | Setor 2 | 3000000.00 |
|  3 | História     | Setor 1 | 2000000.00 |
+----+--------------+---------+------------+
```

CONCLUINDO
Materiais complementares

Materiais disponíveis gratuitamente na internet:

- Vídeo do youtube:
  https://youtu.be/2qCLpE1NZ8c
- Artigo do Devmedia:
  https://www.devmedia.com.br/trabalhando-com-subqueries/40134

CONCLUINDO
Recapitulando

O que vimos na aula de hoje:

- O que é uma subquery e sua utilização
- Como empregar subqueries na cláusula `WHERE`
  - Operadores de pertinência `IN` e `NOT IN`
  - Operadores de comparação `SOME`, `ANY` e `ALL`
  - Testar por relações não vazias `EXISTS` e vazias `NOT EXISTS`
- Como empregar subqueries na cláusula `FROM`
- Como empregar subqueries escalares

CONCLUINDO
Próxima aula

Veremos como utilizar subqueries nas cláusulas:

- `INSERT`
- `UPDATE`
- `DELETE`
- `HAVING`

6 Exercícios

EXERCÍCIOS
Exercícios de fixação

1. Encontre as disciplinas que foram ministradas no primeiro semestre de 2019 e no primeiro semestre de 2020 e no segundo semestre de 2021.
2. Faça uma consulta que retorne o nome do aluno, disciplina e nota de todos os alunos que foram aprovados nas disciplinas, ou seja, cuja nota é diferente de “D” e “E”.
3. Faça uma consulta que busque pela matrícula, nome do aluno, disciplina e nota, das disciplinas ministradas pelo professor cujo número de identificação (ID) é igual a 1000. Além disso, ordene os resultados pelo nome do aluno. (Dica: use `EXISTS`).
4. Faça uma consulta que retorna todas as compras realizadas pelos professores cuja quantidade de produtos comprados é maior do que alguma das médias de produtos comprados por departamento. Use as tabelas “compras” e “materiais”.

EXERCÍCIOS
Problemas para entregar

1. Faça uma consulta que retorna todas as compras realizadas pelos professores cuja quantidade de produtos comprados é menor do que qualquer uma das médias de produtos comprados pelos departamentos.
2. Faça uma consulta que retorna o departamento e a média das compras feitas (por departamento) cuja média é maior do que 20.
3. Explique com um exemplo por que `= SOME` é o mesmo que `IN`, mas `<> SOME` não é o mesmo que `NOT IN`.
4. Explique com um exemplo por que `<> ALL` é o mesmo que `NOT IN`, mas `= ALL` não é o mesmo que `IN`.
5. Pesquise sobre a cláusula `WITH` e veja como ela pode ser empregada algumas vezes como uma alternativa às subqueries.

REFERÊNCIAS

Bibliografia básica

ELMASRI, R.; NAVATHE, S. B. Sistemas de Bancos de Dados. 6.ed. São Paulo: Pearson Addison, 2011.

SILBERSCHATZ , A.; KORTH, H. F.; SUDARSHAN, S. Sistema de Banco de Dados. 7.ed. Rio de Janeiro: GEN LTC, 2020.

Bibliografia complementar

DATE, C. J. Introdução a Sistemas de Bancos de Dados. Rio de Janeiro: Campus, 2000.

OLIVEIRA, C.H.C. SQL: Curso Prático. São Paulo: Novatec, 2002.

CARVALHO, V. MySQL: Comece com o principal banco de dados open source do mercado. São Paulo: Casa do Código, 2015.