Banco de Dados

Prof. Esp. Victor Pedroso

victor.pedroso@unicesumar.edu.br

AGENDA

- Consultas envolvendo NULL.
- Consultas utilizando JOINs.
- Consultas com funções de agregação.
- Comandos de alteração de schema.

RESTRIÇÃO NULL

Valores Null e Not Null

- Um valor NULL indica que o valor é desconhecido.
- Um valor NULL é diferente de um valor vazio ou zero.
- Dois valores nulos não são iguais.
- Testar para obter valores null em uma consulta, use IS NULL ou IS NOT NULL na cláusula WHERE.

RESTRIÇÃO NULL - EXEMPLO

- Listar todos os Professores que não possuem telefone cadastrado:

```
SELECT *
   FROM PROFESSOR
  WHERE FONE IS NULL;
```

- Listar o nome e o cargo de todos os Funcionários que foram demitidos:

```
SELECT NOME, CARGO
   FROM FUNCIONARIOS
  WHERE DATA_DEMISSAO IS NOT NULL
```

CONSULTAS ANINHADAS OU SUBQUERY

- Uma consulta aninhada é quando ela está dentro de outra consulta SQL.

- Pode-se utilizar os comparadores **IN**, **NOT IN**, **EXISTS** E **NOT EXISTS,** que comparam um valor com um conjunto de valores.

CONSULTAS ANINHADAS – UTILIZANDO IN

- **Exemplo Consulta Normal:**

```
SELECT telefone
FROM telefone,
contato
WHERE contato.id =
Telefone.contato_fk
AND sobrenome =
'Machado'
```

- **Utilizando as Subqueries:**

```
SELECT telefone
FROM telefone
WHERE contato_fk IN
(
SELECT id
FROM contato
WHERE sobrenome =
'Machado'
```

CONSULTAS ANINHADAS – UTILIZANDO EXISTS

- **Exemplo Consulta Normal:**

```
SELECT f.nome,
f.sobrenome
FROM funcionário as
f, suborninado as s
WHERE f.id =
s.superior_fk AND
f.nome = s.nome
```

- **Utilizando as Subqueries:**

```
SELECT f.nome,
f.sobrenome FROM
funcionario AS f
WHERE EXISTS
(SELECT * FROM
subordinado AS s
WHERE  f.nome =
s.nome)
```

OUTRO MODELO

funcionario

id	nome	sobrenome	cargo

subordinado

id	nome	sobrenome	superior_fk

EXEMPLO 1

Selecione o nome e sobrenome de todos os funcionários que possuem subordinados com o mesmo nome.

```
SELECT f.nome, f.sobrenome
FROM funcionario AS f, subordinado
AS s
WHERE f.id = s.superior_fk AND
f.nome = s.nome;
```

EXEMPLO 1 COM SUBQUERY CORRELACIONADA

```
SELECT f.nome, f.sobrenome
FROM funcionario AS f
WHERE id IN(
SELECT superior_fk
FROM subordinado AS s
WHERE f.nome = s.nome
);
```

EXEMPLO 1 COM SUBQUERY CORRELACIONADA E EXISTS

```
SELECT f.nome, f.sobrenome
FROM funcionario AS f
WHERE EXISTS (
SELECT *
FROM subordinado AS s
WHERE f.nome = s.nome
);
```

EXEMPLO 2

Selecione o nome e o sobrenome de todos os funcionários que não possuem subordinados.

```
SELECT f.nome, f.sobrenome
FROM funcionario AS f
WHERE NOT EXISTS (
SELECT *
FROM subordinado AS s
WHERE s.superior_fk = f.id
);
```

Comando JOIN - DEFINIÇÕES

- São as maneiras de se ligar as tabelas em uma instrução SQL.
- Tem a função básica de agregar tabelas mediante um campo que faça sentido às mesmas.

TIPOS DE JOIN

- INNER JOIN
- FULL OUTER JOIN ou FULL JOIN
- LEFT OUTER JOIN ou LEFT JOIN
- RIGHT OUTER JOIN ou RIGHT JOIN

TIPOS DE JOIN – INNER JOIN

INNER JOIN:

Retorna apenas as linhas que atendam à condição de junção na cláusula Where (estilo antigo). São selecionadas apenas linha com valores correspondentes.

TIPOS DE JOIN – INNER JOIN - Exemplo

```
SELECT *

FROM  T1, T2

WHERE T1.C1 = T2.C1
```

OU

```
SELECT *

FROM T1 INNER JOIN T2 ON

T1.C1 = T2.C1
```

TIPOS DE JOIN – FULL JOIN

- FULL OUTER JOIN ou FULL JOIN:

  Retorna linhas com valores correspondentes e inclui todas as colunas da tabela de ambas as tabelas (T1 e T2), sem valores correspondentes.

TIPOS DE JOIN – FULL JOIN - Exemplo

```
SELECT *

FROM T1 FULL JOIN T2 ON

T1.C1 = T2.C1
```

TIPOS DE JOIN – LEFT JOIN

- LEFT OUTER JOIN ou LEFT JOIN:

  Retorna linhas com valores correspondentes e inclui todas as colunas da tabela à esquerda (T1), sem valores correspondentes.

TIPOS DE JOIN – LEFT JOIN - Exemplo

```
SELECT *

FROM T1 LEFT JOIN T2 ON

T1.C1 = T2.C1
```

TIPOS DE JOIN – RIGHT JOIN

- RIGHT OUTER JOIN ou RIGHT JOIN:

  Retorna linhas com valores correspondentes e inclui todas as colunas da tabela à direita (T2), sem valores correspondentes.

TIPOS DE JOIN – RIGHT JOIN - Exemplo

```
SELECT *

FROM T1 RIGHT JOIN T2 ON

T1.C1 = T2.C1
```

EXEMPLO 3

Selecione o nome de todos os contatos, cujo telefone inicie com '44'.

```
SELECT nome
FROM contato, telefone
WHERE contato.id =
telefone.contato_fk and
telefone.telefone LIKE '44%';
```

EXEMPLO 3 UTILIZANDO JOIN

```
SELECT nome
FROM contato JOIN telefone ON
contato.id = telefone.contato_fk
WHERE telefone.telefone LIKE '44%';
```

EXEMPLO 4

Selecione todos os nomes de contatos que iniciem com a letra 'A' e seus respectivos telefones. Se o contato não tiver um telefone, mostre somente o nome e NULL como o valor do telefone.

```
SELECT nome, telefone
FROM contato LEFT JOIN telefone ON
contato.id = telefone.contato_fk
WHERE contato.nome LIKE 'A%';
```

FUNÇÕES MAX E MIN

As funções **MAX** e **MIN** ajudam a encontrar os maiores e menores valores em uma consulta.

EXEMPLO 5

Selecione o peso mínimo e máximo de todos os contatos.

```
SELECT MAX(peso), MIN(peso)
FROM contato;
```

FUNÇÃO COUNT

A função **COUNT** é utilizada para contar o número de valores não nulos de um atributo.

EXEMPLO 6

Selecione o número total de contatos cujo peso > 80;

```
SELECT COUNT(*)
FROM contato
WHERE peso > 80;
```

FUNÇÃO DISTINCT

A função **DISTINCT** é utilizada para produzir uma lista que contém apenas os valores diferentes uns dos outros.

EXEMPLO 7

Selecione a quantidade de pesos distintos de todos os contatos.

SELECT **COUNT**(**DISTINCT** peso)
FROM contato;

FUNÇÃO GROUP BY

A cláusula **GROUP BY** geralmente é utilizada quando se tem colunas de atributos combinadas com funções agregadas no comando SELECT.

EXEMPLO 8

Selecione o sobrenome e a quantidade de contatos que possuem o mesmo sobrenome.

```
SELECT sobrenome, COUNT(*)
FROM contato
GROUP BY sobrenome;
```

Cláusula HAVING

A cláusula HAVING é a condição aplicada ao resultado de uma operação GROUPY BY.

EXEMPLO 9

Selecione o sobrenome e quantidade de contatos que possuem o mesmo sobrenome, desde que haja, pelo menos, dois contatos com o mesmo sobrenome.

```
SELECT sobrenome, COUNT(*)
FROM contato
GROUP by sobrenome
HAVING COUNT(*) > 1;
```

EXEMPLO 10

Remova o *schema* agenda do banco de dados.

```
DROP SCHEMA agenda;
```

EXEMPLO 11

Remova a tabela telefone do *schema*.

```
DROP TABLE telefone;
```

EXEMPLO 12

Adicione uma coluna apelido na tabela contato contendo 15 caracteres.

ALTER TABLE contato **ADD COLUMN** apelido VARCHAR(15);

EXEMPLO 13

Adicione uma coluna apelido na tabela contato contendo 15 caracteres e com valor padrão de 'Senhor'.

```
ALTER TABLE contato ADD COLUMN
apelido VARCHAR(15) DEFAULT
'Senhor';
```

EXEMPLO 14

Altere o tamanho da coluna apelido para 25 caracteres.

```
ALTER TABLE contato ALTER COLUMN
apelido VARCHAR(25);
```

EXEMPLO 15

Remova a coluna apelido da tabela contato.

```
ALTER TABLE contato DROP COLUMN
apelido;
```

Exemplo 16

Supondo que ainda não houvesse uma integridade referencial entre a tabela telefone e a tabela contato, adicione-a.

```
ALTER TABLE telefone ADD FOREIGN KEY (contato_fk) REFERENCES contato(id);
```

Banco de Dados

Prof. Esp. Victor Pedroso
victor.pedroso@unicesumar.edu.br