

# Banco de Dados

## Linguagem SQL

DML – Data Manipulation Language – Outras Operações

---

**Prof. Clovis Ferraro**  
cferraro@unicid.edu.br



# Filtrando Linhas da Tabela

# Seleção – Cláusula WHERE

- Filtra as linhas que atendem às condições lógicas especificadas.
- Uma consulta sem a cláusula WHERE retornará TODAS as linhas da tabela.
- Como o desenvolvimento CLIENTE/SERVIDOR preconiza trafegar o menor número possível de dados, procure **SEMPRE** filtrar suas consultas, sobretudo em tabelas grandes.

# Seleção – Cláusula WHERE

## • Operadores de Comparação

- **Relacionais:** >, >=, <, <=, <>, = : Maior que, Maior ou igual que, Menor que, Menor ou igual que, Diferente de e Igual a.

❖ **Exemplo:** WHERE CODIGO = 18

WHERE PRECO >= 25.15

- **Intervalo: BETWEEN**
- **Listas: IN e NOT IN**
- LIKE e NOT LIKE: %
- IS NULL e IS NOT NULL

<b>between...and</b>	<b>Entre dois valores</b>
<b>IN(valores)</b>	<b>Sequência de busca</b>
<b>LIKE</b>	<b>Busca por padrão</b>
<b>IS NULL</b>	<b>É um valor nulo</b>

# Seleção – Cláusula WHERE

## • Operadores Lógicos

AND	Retorna TRUE se ambas as condições forem verdadeiras
OR	Retorna TRUE se uma das condições for verdadeira
NOT	Retorna TRUE se a condição seguinte for falsa

# Seleção – Cláusula WHERE

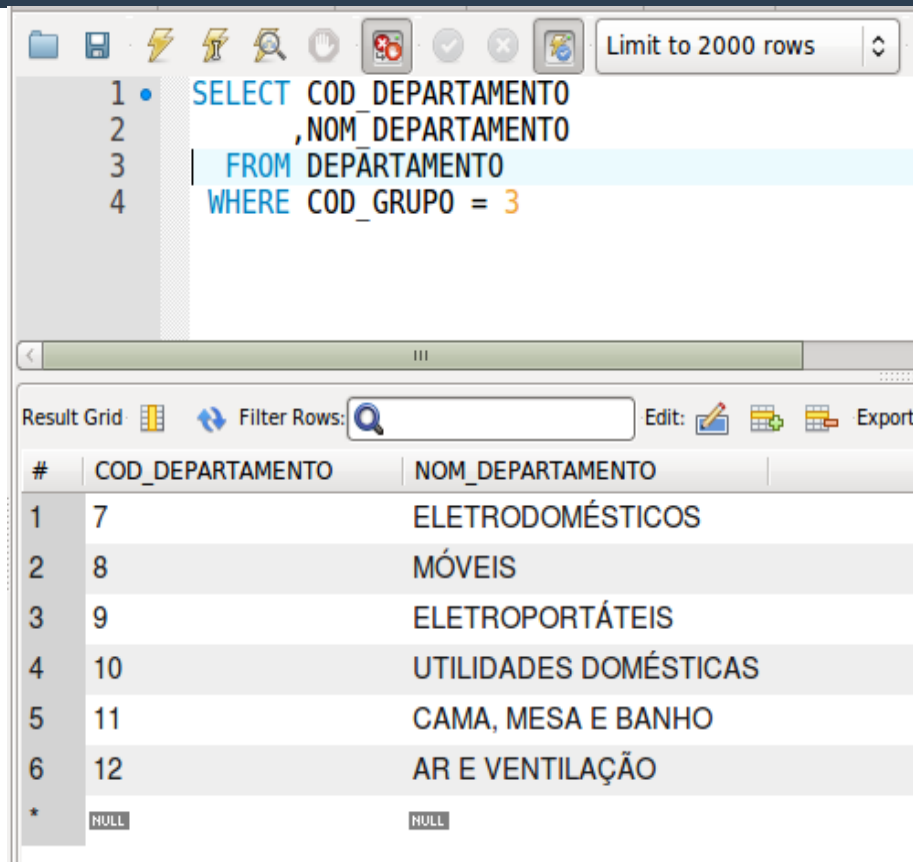
- **Operadores Lógicos**

- AND – operador 'E'
- OR – operador 'OU'
- NOT – operador de negação

- **Exemplo:**

```
SELECT cod_departamento,  
       nom_departamento FROM  
LOJA.departamento WHERE  
COD_DEPARTAMENTO = 5
```

**OR** NOM\_DEPARTAMENTO LIKE 'A%'



The screenshot shows a database query editor with a toolbar at the top. The SQL query is as follows:

```
1 • SELECT COD_DEPARTAMENTO  
2      ,NOM_DEPARTAMENTO  
3      FROM DEPARTAMENTO  
4      WHERE COD_GRUPO = 3
```

Below the query editor is a 'Result Grid' showing the results of the query. The grid has three columns: '#', 'COD\_DEPARTAMENTO', and 'NOM\_DEPARTAMENTO'. The results are as follows:

#	COD_DEPARTAMENTO	NOM_DEPARTAMENTO
1	7	ELETRODOMÉSTICOS
2	8	MÓVEIS
3	9	ELETROPORTÁTEIS
4	10	UTILIDADES DOMÉSTICAS
5	11	CAMA, MESA E BANHO
6	12	AR E VENTILAÇÃO
*	NULL	NULL

# Exclusão DELETE

# Remoção

- Duas instruções para remover linhas de uma tabela:
  - DELETE.
  - TRUNCATE.
- Com DELETE você remove linhas de uma tabela e opcionalmente você especifica um predicado para restringir as linhas a serem removidas.

DELETE FROM <TABELA>

WHERE <predicado>;

**Obs.: Sempre use a cláusula WHERE para evitar a perda de dados da tabela!**



# Remoção

- Ex.:

DELETE FROM

ADMINISTRACAO.EMPREGADO

WHERE

CPF = 159753456 OR CPF = 559753456;

Apaga os registros que contém a restrição na cláusula WHERE.

- Ex.:

DELETE FROM ADMINISTRACAO.EMPREGADO;

Apaga a tabela EMPREGADO.

# Remoção

- Com TRUNCATE você remove todas as linhas de uma tabela.

TRUNCATE TABLE <TABELA>;

O comando TRUNCATE TABLE permite **remover todas as linhas de uma tabela em uma única operação**, sem registrar as exclusões de linhas individuais.

É como executar a instrução DELETE, porém sem usar a cláusula WHERE. Portanto, é usada para apagar completamente o conteúdo de uma tabela.

Entretanto, a cláusula TRUNCATE TABLE é mais rápida e utiliza menos recursos de sistema e log de transações durante sua execução.

**\*\* DROP = Exclui uma tabela do banco de dados e todas as linhas num só comando!**

# Atualização UPDATE



# Atualização

- Instrução UPDATE.

UPDATE <tabela alvo>

SET <col 1> = <expressão 1>,

<col 2> = <expressão 2>,

...

<col n> = <expressão n>

WHERE <predicado>;

# Atualização

- Instrução UPDATE

```
UPDATE ADMINISTRACAO.EMPREGADO
```

```
SET
```

```
    Mnome = 'C',
```

```
    Salario = 27000
```

```
WHERE
```

```
    CPF = 159753456;
```

# Atualização

- Predicado:
  - Filtra as linhas que serão realizadas as modificações.
  - Se não tiver cláusula WHERE: Aplica para todas as linhas da tabela.
- Ex.: Aplicar aumento de 10% para todos os empregados:

```
UPDATE ADMINISTRACAO.EMPREGADO
```

```
SET
```

```
    Salario = 1.1 * Salario
```

```
WHERE dependente = 'S';
```

# Atualização

- Predicado:
  - Filtra as linhas que serão realizadas as modificações.
  - Se não tiver cláusula WHERE: Aplica para todas as linhas da tabela.
- Ex.: Aplicar aumento de 10% para todos os empregados:

```
UPDATE ADMINISTRACAO.EMPREGADO
```

```
SET
```

```
    Salario = 1.1 * Salario
```

```
WHERE dependente = 'S';
```



# Atualização - Exemplo

- Ex.: Aplicar aumento de 10% para todos os produtos de cod\_categoria igual a 3:

```
SELECT pco_produto from produto;
```

```
UPDATE LOJA.produto
```

```
SET
```

```
    pco_produto = pco_produto * 1.1
```

```
WHERE cod_categoria = 3;
```

```
SELECT pco_produto from produto
```

# FUNÇÕES *DE STRING*

- **CASE (primeira forma)**
- Compara uma expressão ou coluna, fornecendo um valor escolhido:

```
SELECT cod_condicao_pagamento,  
       CASE cod_condicao_pagamento  
           WHEN 1 THEN 'A vista'  
           WHEN 2 THEN 'Pagto parcelado'  
       END  
as Tipo_Pagamento FROM compra ;
```

# FUNÇÕES *DE STRING*

- **CASE (segunda forma)**

- Compara uma expressão ou coluna, fornecendo um valor previamente escolhido:

CASE

WHEN expression THEN b

WHEN expression THEN d

WHEN expression THEN f

ELSE w

END

# FUNÇÕES *DE STRING*

- **CASE** (segunda forma)

- Compara uma expressão ou coluna, fornecendo um valor previamente escolhido:

CASE

WHEN **expression** THEN b

WHEN **expression** THEN d

WHEN **expression** THEN f

ELSE w

END

# FUNÇÕES *DE STRING*

- CASE (segunda forma)

```
SELECT          cod_produto, NOM_PRODUTO, idf_ativo FROM  
PRODUTO WHERE  cod_categoria = 8 AND idf_ativo = 's';
```

# FUNÇÕES *DE STRING*

- CASE (segunda forma)

```
SELECT NOM_PRODUTO,  
       CASE IDF_ATIVO  
         WHEN 'S' THEN 'SIM'  
         WHEN 'N' THEN 'NÃO'  
       END  
FROM PRODUTO WHERE COD_FABRICANTE = 4  
ORDER BY NOM_PRODUTO;
```

# Ordenação

## (Order By)

# Seleção – Cláusula ORDER BY

- Quando uma consulta tem ordem garantida?

- Ex.:

```
SELECT
```

```
    COD_PRODUTO,NOM_PRODUTO
```

```
FROM PRODUTO
```

```
WHERE COD_CATEGORIA = 8;
```

	cod_produto	NOM_PRODUTO
▶	14	Impressora Monocromática Laser - Samsung ML-2165
	15	Impressora Jato de Tinta HP Designjet T120 ePrinter Series CQ891A#B1K
	16	Impressora HP LaserJet Pro P1102w Wireless com ePrint
	17	Impressora HP LaserJet Pro P1102 CE651A#696 - Branco/Cinza

- Ao executar várias vezes “parece” que a ordem de devolução é sempre a mesma.
  - Ordem da inserção dos dados?
  - Ordem da chave primária?
  - Ordem de algum índice?



# Seleção – Cláusula ORDER BY

- O SGBD pode alterar a ordem da consulta.
  - A execução dessa consulta **não possui garantia** de ordem nas linhas devolvidas.
- Cláusula ORDER BY.
  - Única maneira de garantir que as linhas devolvidas em uma consulta tenham uma ordem.
  - Ordena o resultado da pesquisa pelas colunas indicadas.

# Seleção – Cláusula ORDER BY

```
SELECT NOM_PRODUTO,  
       CASE IDF_ATIVO  
         WHEN 'S' THEN 'SIM'  
         WHEN 'N' THEN 'NÃO'  
       END SITUACAO  
FROM   PRODUTO  
WHERE  COD_FABRICANTE = 4  
ORDER BY NOM_PRODUTO;
```

NOM_PRODUTO	SITUACAO
Ultrabook Acer Aspire com Intel® Core™ i5	SIM
Ultrabook Acer Aspire M5-481T-6195 com Intel® Core™ i5-3317U	SIM
Ultrabook Acer Aspire M5-481T-6417 com Intel® Core™ i5-3317U	SIM
Ultrabook Acer Aspire M5-481T-6650 com Intel® Core™ i3-3227U	SIM

- Pode especificar a direção do ordenamento: **ASC** ou **DESC**.
  - Se não for especificado o padrão é ASC.

# Seleção – Cláusula ORDER BY

```
SELECT NOM_PRODUTO,  
       CASE IDF_ATIVO  
         WHEN 'S' THEN 'SIM'  
         WHEN 'N' THEN 'NÃO'  
       END SITUACAO  
FROM   PRODUTO  
WHERE  COD_FABRICANTE = 4  
ORDER BY NOM_PRODUTO DESC;
```

NOM_PRODUTO	SITUACAO
Ultrabook Acer Aspire com Intel® Core™ i5	SIM
Ultrabook Acer Aspire M5-481T-6195 com Intel® Core™ i5-3317U	SIM
Ultrabook Acer Aspire M5-481T-6417 com Intel® Core™ i5-3317U	SIM
Ultrabook Acer Aspire M5-481T-6650 com Intel® Core™ i3-3227U	SIM

- Pode especificar a direção do ordenamento: **ASC** ou **DESC**.
  - Se não for especificado o padrão é ASC.

# Seleção – Cláusula ORDER BY

```
SELECT NOM_PRODUTO, PCO_PRODUTO FROM PRODUTO  
ORDER BY PCO_PRODUTO DESC;
```

# Seleção – Cláusula ORDER BY

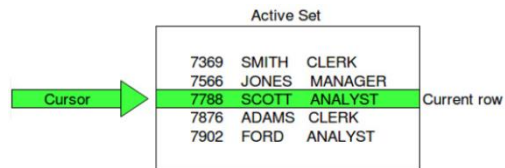
- Só **garantimos** a ordem das colunas especificadas na cláusula **ORDER BY**:

```
SELECT NOM_PRODUTO,  
       CASE IDF_ATIVO  
         WHEN 'S' THEN 'SIM'  
         WHEN 'N' THEN 'NÃO'  
       END SITUACAO  
FROM   PRODUTO  
WHERE  COD_FABRICANTE = 4  
ORDER BY NOM_PRODUTO DESC;
```

NOM_PRODUTO	SITUACAO
Ultrabook Acer Aspire M5-481T-6650 com Intel® Core™ i3-3227U	SIM
Ultrabook Acer Aspire M5-481T-6417 com Intel® Core™ i5-3317U	SIM
Ultrabook Acer Aspire M5-481T-6195 com Intel® Core™ i5-3317U	SIM
Ultrabook Acer Aspire com Intel® Core™ i5	SIM

# Seleção – Cláusula ORDER BY

- Ordenação de valores nulos:
  - ✓ SQL padrão: NULLs devem ser agrupados juntos.
  - ✓ SQL Server: Antes dos dados não nulos (ASC).
- De acordo com o SQL padrão, uma consulta com ORDER BY devolve um cursor e não uma relação.



The diagram shows a table titled 'Active Set' with five rows of employee data. A green arrow labeled 'Cursor' points to the third row, which is highlighted in green. To the right of the table, the text 'Current row' is present.

7369	SMITH	CLERK
7566	JONES	MANAGER
7788	SCOTT	ANALYST
7876	ADAMS	CLERK
7902	FORD	ANALYST

**Cursor** = estrutura de controle, é um ponteiro para uma linha em um conjunto de resultados ou uma tabela.

**Relação** = são as associações estabelecidas entre duas ou mais tabelas. As relações são baseadas em campos comuns de mais de uma tabela, envolvendo muitas vezes chaves primárias e estrangeiras.

# Trabalhando com Conjuntos

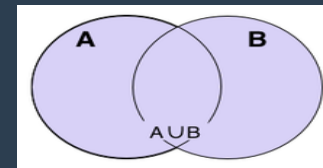
# DML – Conjuntos

## . Operações com Conjuntos

- União (UNION)
- Intersecção (INTERSECT/IN)
- Diferença (MINUS/EXCEPT/NOT IN)



# DML – Conjuntos: UNION



- UNION

- É a união de duas seleções.
- O resultado parece ser de uma única seleção.

# DML – Conjuntos: UNION

## • UNION

- O **operador UNION** combina os resultados de duas ou mais queries (consultas) em um **único result set**, retornando todas as linhas pertencentes a todas as queries envolvidas na execução.
- Assim, uma UNION combina (“une”) duas ou mais declarações **SELECT**. O resultado de cada SELECT deve possuir o mesmo número de colunas, e o tipo de dado de cada coluna correspondente deve ser compatível.

```
SELECT declaração_1  
UNION [ALL]  
SELECT declaração_2...  
[ORDER BY colunas]
```

# DML – Conjuntos: UNION

## . UNION

Existem dois tipos de operador UNION:

- **UNION**: por default, executa o equivalente a um `SELECT DISTINCT` no result set final. Em outras palavras, ele combina o resultado de execução das duas queries e então executa um `SELECT DISTINCT` a fim de eliminar as linhas duplicadas. Este processo é executado mesmo que não haja registros duplicados.
- **UNION ALL**: O operador **UNION ALL** tem a mesma funcionalidade do **UNION**, porém, não executa o ***SELECT DISTINCT*** no result set final e apresenta todas as linhas, inclusive as **linhas duplicadas**.

# DML – Conjuntos: UNION – ORDER BY

## . UNION – ORDER BY

- Para ordenar o resultado de uma operação UNION, podemos usar uma declaração **ORDER BY** após o último SELECT codificado.
- Neste ORDER BY somente é possível usar os nomes de colunas que foram declarados no primeiro SELECT da declaração UNION completa. Os nomes de colunas no resultado final são tirados deste primeiro SELECT.

# DML – Conjuntos: UNION

## . UNION – EXEMPLO

```
CREATE TABLE t1 (id INT NOT NULL, PRIMARY KEY (id));
```

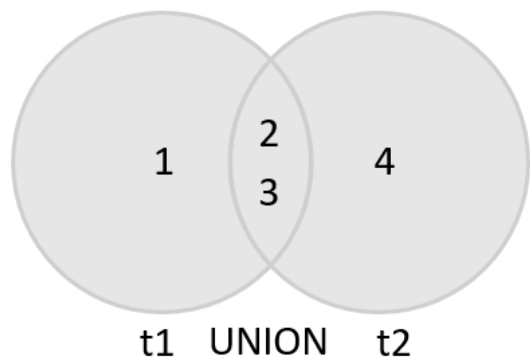
```
CREATE TABLE t2 (id INT NOT NULL, PRIMARY KEY (id));
```

```
INSERT INTO t1 VALUES (1),(2),(3);
```

```
INSERT INTO t2 VALUES (2),(3),(4);
```

```
SELECT id  
  FROM t1  
UNION  
  SELECT id  
    FROM t2;
```

	id
▶	1
	2
	3
	4



# DML – Conjuntos: UNION ALL

## . UNION ALL – EXEMPLO

```
SELECT id  
  FROM t1  
UNION ALL  
  SELECT id  
    FROM t2;
```

	id
▶	1
	2
	3
	2
	3
	4

# DML – Conjuntos: UNION

## . UNION

USE sakila;

```
SELECT * from customer; 599 row(s) returned  
SELECT * FROM actor; 200 row(s) returned
```

```
SELECT  
    first_Name, last_Name  
FROM actor  
UNION ALL  
    SELECT      first_Name, last_Name  
FROM customer;
```

	first_Name	last_Name
►	PENELOPE	GUINNESS
	NICK	WAHLBERG
	ED	CHASE
	JENNIFER	DAVIS
	JOHNNY	LOLLOBRIGIDA
	BETTE	NICHOLSON
	GRACE	MOSTEL
	MATTHEW	JOHANSSON
	JOE	SWANK
	CHRISTIAN	GABLE

799 row(s) returned

# DML – Conjuntos: UNION

## • UNION – Com ALIAS e concatenação

```
SELECT  
    CONCAT(first_name, ' ', last_name) fullname  
FROM actor  
UNION  
SELECT  
    CONCAT(first_name, ' ', last_name)  
FROM customer;
```

	fullname
▶	PENELOPE GUINNESS
	NICK WAHLBERG
	ED CHASE
	JENNIFER DAVIS
	JOHNNY LOLLOBRIGIDA
	BETTE NICHOLSON
	GRACE MOSTEL
	MATTHEW JOHANSSON
	JOE SWANK
	CHRISTIAN GABLE

797 row(s) returned



# DML – Conjuntos: UNION

- **UNION – Com ALIAS, concatenação e ORDER BY**

```
SELECT
    CONCAT(first_name, ' ', last_name) fullname
FROM actor
UNION
SELECT
    CONCAT(first_name, ' ', last_name)
FROM customer;
ORDER BY fullname;
```

	fullname
▶	AARON SELBY
	ADAM GOOCH
	ADAM GRANT
	ADAM HOPPER
	ADRIAN CLARY
	AGNES BISHOP
	AL GARLAND
	ALAN DREYFUSS
	ALAN KAHN
	ALBERT CROUSE

797 row(s) returned

# DML – Conjuntos: UNION

## • UNION – Recomendações

1. Se não existe a possibilidade de haver registros duplicados em suas tabelas ou se não houver problemas para a aplicação que o record set final apresente duplicações, utilize o operador **UNION ALL**. A vantagem é que este operador **não executa a função SELECT DISTINCT**, utiliza menos recursos e como consequência, melhora a performance da aplicação.
1. Não utilize o operador **UNION** em conjunto com a função **SELECT DISTINCT** pois o resultado final será exatamente o mesmo, porém, estará executando a mesma operação duas vezes, causando queda de desempenho.

# DML – Conjuntos: UNION

## . UNION - Recomendações

3. Uma query com uma ou mais cláusula **OR** pode ser reescrita utilizando o operador **UNION ALL**, melhorando significativamente resultado final.

```
SELECT employeeID, firstname, lastname  
FROM names  
WHERE dept = "prod" or city = "Orlando" or division = "food"
```

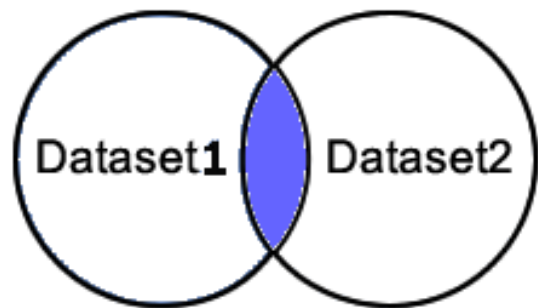
```
SELECT employeeID, firstname, lastname FROM names WHERE dept = "prod"  
UNION ALL  
SELECT employeeID, firstname, lastname FROM names WHERE city = "Orlando"  
UNION ALL  
SELECT employeeID, firstname, lastname FROM names WHERE division = "food"
```

# DML – Conjuntos: IN

## . IN

- O **operador IN** combina os resultados de duas ou mais queries (consultas) em um **único result set**, retornando as linhas pertencentes às duas queries envolvidas na execução.

```
SELECT declaração_1  
WHERE coluna IN  
    (SELECT declaração_2...)  
[ORDER BY colunas]
```



# DML – Conjuntos: UNION ALL

## . IN – EXEMPLO

```
SELECT id  
  FROM t1  
WHERE id IN  
  (SELECT id  
   FROM t2);
```

# DML – Conjuntos: UNION ALL

## . IN – EXEMPLO

```
SELECT id  
  FROM t1  
WHERE id NOT IN  
  (SELECT id  
   FROM t2);
```

# DML – Conjuntos: IN

## . IN

```
CREATE TABLE DemoTable (Number1 int);  
INSERT INTO DemoTable VALUES(100);  
INSERT INTO DemoTable VALUES(200);  
INSERT INTO DemoTable VALUES(300);  
SELECT * FROM DemoTable;
```

Number 1
100
200
300

```
CREATE TABLE DemoTable2 (Number2 int);  
INSERT INTO DemoTable2 VALUES(100);  
INSERT INTO DemoTable2 VALUES(300);  
INSERT INTO DemoTable2 VALUES(400);  
SELECT * FROM DemoTable2;
```

Number 2
100
300
400

```
SELECT Number1 FROM DemoTable  
WHERE Number1 IN (SELECT Number2 FROM DemoTable2);
```

Number 1
100
300

# DML – Conjuntos: NOT IN

## • NOT IN

- O **operador NOT IN** combina os resultados de duas ou mais queries (consultas) em um **único result set**, retornando somente as linhas pertencentes a **primeira query** envolvida na execução.

```
SELECT declaração_1  
WHERE coluna NOT IN  
    (SELECT declaração_2...)  
[ORDER BY colunas]
```





# DML – Conjuntos: NOT IN

## • NOT IN

```
CREATE TABLE DemoTable (Number1 int);  
INSERT INTO DemoTable VALUES(100);  
INSERT INTO DemoTable VALUES(200);  
INSERT INTO DemoTable VALUES(300);  
SELECT * FROM DemoTable;
```

```
CREATE TABLE DemoTable2 (Number2 int);  
INSERT INTO DemoTable2 VALUES(100);  
INSERT INTO DemoTable2 VALUES(300);  
INSERT INTO DemoTable2 VALUES(400);  
SELECT * FROM DemoTable2;
```

```
SELECT Number1 FROM DemoTable  
    WHERE Number1 NOT IN (SELECT Number2 FROM DemoTable2);  
SELECT Number2 FROM DemoTable2  
    WHERE Number2 NOT IN (SELECT Number1 FROM DemoTable);
```

Number1
100
200
300

Number2
100
300
400

Number1
200

Number2
400

# Referências

# Referências

- ELMASRI, Ramez; NAVATHE, Shamkant B.. Parte 2: Modelo de Dados Relacional e SQL: SQL Básica. In: ELMASRI, Ramez; NAVATHE, Shamkant B.. **Sistema de Banco de Dados**. 6. ed. São Paulo: Pearson Education, 2011. Cap. 4. p. 57-75.
- RAMAKRISHNAN, Raghu; GEHRKE, Johannes. SQL, Consultas, Restrições, Gatilhos. In: RAMAKRISHNAN, Raghu; GEHRKE, Johannes. **Sistemas de Gerenciamento de Banco de Dados**. 3. ed. São Paulo: Mc Graw Hill, 2008. Cap. 5. p. 110-147. Tradução da Terceira Edição.
- RANGEL, Alexandre Leite. DML: Data Manipulation Language. In: RANGEL, Alexandre Leite. **InterBase 7: Desenvolvendo e Administrando Bancos de Dados**. Rio de Janeiro: Alta Books, 2003. Cap. 2. p. 78-102.
- RANGEL, Alexandre Leite. **LINGUAGEM SQL**. Batatais: Claretiano, 2016. 97 p.

# Referências

- RANGEL, Alexandre Leite. Manipulação de Dados em MySQL. In: RANGEL, Alexandre Leite. **MySQL** - Projeto, Modelagem e Desenvolvimento de Bancos de Dados. Rio de Janeiro: Alta Books, 2004. Cap. 6. p. 74-96.
- SILBERSCHATZ, Abraham; KORTH, Henry F.; SUNDARSHAN, S.. Introdução à SQL. In: SILBERSCHATZ, Abraham; KORTH, Henry F.; SUNDARSHAN, S.. **Sistemas de Banco de Dados**. 5. ed. Rio de Janeiro: Campus, 2006. Cap. 3. p. 37-67.
- W3SCHOOLS. **SQL INSERT INTO Statement**. Disponível em: <[https://www.w3schools.com/sql/sql\\_insert.asp](https://www.w3schools.com/sql/sql_insert.asp)>. Acesso em: 19 jun. 2019.
- \_\_\_\_\_. **SQL INSERT INTO SELECT Statement**. Disponível em: <[https://www.w3schools.com/sql/sql\\_insert\\_into\\_select.asp](https://www.w3schools.com/sql/sql_insert_into_select.asp)>. Acesso em: 19 jun. 2019.

Obrigado!

Prof. Clovis Ferraro

(Adaptado de Prof.Dr. Alexandre Rangel)