OPERADORES DE CONJUNTO

Aula 7

OPERADORES DE CONJUNTO

- Combinar resultados de duas ou mais consultas em uma.

UNION

- Combina todas as linhas de duas ou mais tabelas participante da operação UNION, eliminando as linhas duplicatas resultantes.

UNION

- Combina todas as linhas de duas ou mais tabelas participante da operação UNION, eliminando as linhas duplicatas resultantes.

UNION

```
tpch=> SELECT c_acctbal
FROM customer
UNION
SELECT p_retailprice
FROM part;
```

Todos as contas dos clientes e todos os preços de varejo não duplicados

UNION

```
tpch=> SELECT c_acctbal
FROM customer
UNION
SELECT p_retailprice
FROM part;
```

```
(158391 rows)
```

UNION ALL

- Combina todas as linhas de duas ou mais tabelas participante da operação UNION.

UNION ALL

```
tpch=> SELECT c_acctbal
FROM customer
UNION ALL
SELECT p_retailprice
FROM part;
```

```
(350000 rows)
```

Todos as contas dos clientes e todos os preços de varejo.

QUANDO UTILIZAR UNION E UNION ALL

- UNION ALL sem duplicidade no resultados;

- DISTINCT está implícito em UNION.

- Uma consulta com um ou mais condições de OR podem ser reescrita utilizando UNION ALL.

UNION ALL /OR

```
tpch=> SELECT ps_availqty, ps_supplycost
FROM partsupp
WHERE ps_availqty=9999
OR ps_supplycost=399;
```

ps_availqty	ps_supplycost
9999	34.79
9999	790.31
9999	80.52
9999	458.20
9999	851.63
9999	569.39
9999	570.82
9999	173.46
6907	399.00
635	399.00

QUANDO UTILIZAR UNION E UNION ALL

```
tpch=> SELECT ps_availqty,ps_availqty
FROM partsupp
WHERE ps_availqty=9999
UNION ALL
SELECT ps_availqty,ps_supplycost
FROM partsupp
WHERE ps_supplycost=399;
```

ps_availqty	ps_availqty
9999	9999
9999	9999
9999	9999
9999	9999
9999	9999
9999	9999

6907	399.00
635	399.00
3648	399.00
7012	399.00
3583	399.00
6510	399.00

(80 rows)

DIFERENÇA ENTRE UNION ALL E OR

As duas consultas exibirão o mesmo resultado. Porém, se uma das colunas tiver um índice, mas as outras não, a primeira consulta fará o Table Scan. Na segunda consulta, o índice será utilizado em parte da consulta, melhorando a performance em geral.

INTERSECT

- Retorna todas as linhas comuns retornadas por duas ou mais consultas participantes do INTERSECT.

INTERSECT

```
tpch=> SELECT c_acctbal
FROM customer
INTERSECT
SELECT p_retailprice
FROM part;
```

A consulta exibirá o conta do cliente que também é preço de varejo.

```
(2695 rows)
```

EXCEPT

- Retorna as linhas da primeira consulta que não existem na segunda consulta.

EXCEPT

```
tpch=> SELECT c_acctbal as UNION
 FROM customer
EXCEPT
SELECT  p_retailprice
FROM part;
```

A consulta exibirá o conta do cliente que NÃO é preço de varejo.

```
(137492 rows)
```

FUNÇÕES ÚTEIS

FUNÇÕES TRIGONOMÉTRICAS

- acos(x) : cosseno inverso.

```
tpch=> SELECT acos(0), acos(1), acos(-1);
```

FUNÇÕES TRIGONOMÉTRICAS

- acos(x) : cosseno inverso.

```
tpch=> SELECT acos(0), acos(1), acos(-1);
```

```
       acos          | acos |         acos          | acos
---------------------+------+-----------------------+------
 1.5707963267949 |    0 | 3.14159265358979 |    0
(1 row)
```

FUNÇÕES TRIGONOMÉTRICAS

- asin(x) : seno inverso.

```
tpch=> SELECT asin(0), asin(1), asin(-1);
 asin |       asin        |        asin
------+-------------------+--------------------
    0 | 1.5707963267949 | -1.5707963267949
(1 row)
```

FUNÇÕES TRIGONOMÉTRICAS

- atan(x) : tangente inverso.

```
tpch=> SELECT atan(0), atan(1), atan(-1);
 atan |        atan         |         atan
------+---------------------+----------------------
    0 | 0.785398163397448 | -0.785398163397448
(1 row)
```

FUNÇÕES TRIGONOMÉTRICAS

- atan(x,y) : tangente inverso de x/ y.

```
tpch=> SELECT atan2(0,3), atan2(1,3), atan2(-1,-3);
 atan2 |       atan2        |       atan2
-------+--------------------+--------------------
     0 | 0.321750554396642 | -2.81984209919315
(1 row)
```

FUNÇÕES TRIGONOMÉTRICAS

- cos(x) : cosseno.

```
tpch=> SELECT cos(pi()) as pi, cos(1) as cos_1;
 pi |       cos_1
----+--------------------
 -1 | 0.54030230586814
(1 row)
```

FUNÇÕES TRIGONOMÉTRICAS

- sen(x) : seno.

```
tpch=> SELECT sin(pi()) as pi, sin(1) as sin_1;
          pi           |       sin_1
-----------------------+--------------------
 1.22464679914735e-16 | 0.841470984807897
(1 row)
```

FUNÇÕES TRIGONOMÉTRICAS

- tang(x) : tangente.

```
tpch=> SELECT tan(pi()) as pi, tan(1) as tan_1;
          pi             |       tan_1
-------------------------+--------------------
 -1.22464679914735e-16 | 1.5574077246549
(1 row)
```

FUNÇÕES TRIGONOMÉTRICAS

- tang(x) : tangente.

```
tpch=> SELECT tan(pi()) as pi, tan(1) as tan_1;
          pi             |       tan_1
-------------------------+--------------------
 -1.22464679914735e-16 | 1.5574077246549
(1 row)
```

FUNÇÕES TRIGONOMÉTRICAS

- cot(x) : cotangente.

```
tpch=> SELECT cot(pi()) as pi, cot(1) as cot_1;
          pi                |        cot_1
----------------------------+----------------------
 -8.16561967659768e+15 | 0.642092615934331
(1 row)
```

FUNÇÕES TRIGONOMÉTRICAS

- ln(x) : logaritmo natural.

```
tpch=> SELECT ln(200) as ln_100, ln(100) as ln_1;
      ln_100       |       ln_1
-------------------+-------------------
 5.29831736654804 | 4.60517018598809
(1 row)
```

FUNÇÕES TRIGONOMÉTRICAS

- log(b,n) : logaritmo qualquer base.

```
tpch=> SELECT log(2,200), log(3.100);
          log           |                log
------------------------+---------------------------
 7.643856189774724 | 0.49136169383427267967
(1 row)
```

FUNÇÕES TRIGONOMÉTRICAS

- log(n) : logaritmo base 10.

```
tpch=> SELECT log(200), log(100);
        log          | log
---------------------+-----
 2.30102999566398 |   2
(1 row)
```

FUNÇÕES DE MATRIZES

- array_cat(anyarray):concatena duas matrizes

```
tpch=> SELECT array_cat(ARRAY[1,2], ARRAY[2,3]);
 array_cat
-----------
 {1,2,2,3}
(1 row)
```

FUNÇÕES DE MATRIZES

- array_append(anyarray, anyelement): adiciona um elemento no final da matriz

```
tpch=> SELECT array_append(ARRAY[1,2], 3);
 array_append
--------------
 {1,2,3}
(1 row)
```

FUNÇÕES DE MATRIZES

- array_prepend(anyelement,anyarray): adiciona um elemento no inicio da matriz

```
tpch=> SELECT array_prepend(3, ARRAY[1,2]);
 array_prepend
---------------
 {3,1,2}
(1 row)
```

FUNÇÕES DE MATRIZES

- array_dims (anyarray): retona as dimensões

```
tpch=> SELECT array_dims(ARRAY[3,1,2]);
 array_dims
------------
 [1:3]
(1 row)
```

```
tpch=> SELECT array_dims(ARRAY[[3,1,2],[2,5,6]]);
 array_dims
------------
 [1:2][1:3]
(1 row)
```

FUNÇÕES DE MATRIZES

- array_lower (anyarray, integer): retorna o limite inferior da dimensão especificada da matriz

```
tpch=> SELECT array_lower(ARRAY[3,1,2],1);
 array_lower
-------------
           1
(1 row)
```

FUNÇÕES DE MATRIZES

- array_upper (anyarray, integer): retorna o limite superior da dimensão especificada da matriz

```
tpch=> SELECT array_upper(ARRAY[3,1,2],1);
 array_upper
-------------
           3
(1 row)
```

FUNÇÕES DE MATRIZES

- array_to_string (anyarray, text): concatena os elementos da matriz utilizando o delimitador especificado

```
tpch=> SELECT array_to_string(ARRAY[3,1,2],'^');
 array_to_string
-----------------
 3^1^2
(1 row)
```

FUNÇÕES DE MATRIZES

- string_to_array (text, text): divide uma cadeia de caracteres em elementos de matriz utilizando o delimitador especificado

```
tpch=> SELECT string_to_array('3^1^2','^');
 string_to_array
-----------------
 {3,1,2}
(1 row)
```

O QUE
HÁ DE NOVO SQL 2016?

MATCH_RECONGNIZE

- Encontrando séries de eventos consecutivos;
- Correspondência de padrões: inversão de tendências, eventos periódicos,…
- Top N por grupo;

MATCH_RECONGNIZE

```
SELECT
    ename,
    hiredate,
    sal,
    trend
FROM
    emp
MATCH_RECOGNIZE (
  ORDER BY hiredate
  MEASURES CLASSIFIER () AS TREND
  ALL ROWS PER MATCH
  PATTERN (FIRST * Better * Worst * Same *)
  DEFINE FIRST AS ROWNUM = 1,
  Better AS Better.sal > PREV (sal),
  Same AS Same.sal = PREV (sal),
  Worst AS Worst.sal < PREV (sal));
```

reconhecer tendências em salários de EMPs sobre a data de contrataçã o.

MATCH_RECONGNIZE

ENAME	HIREDATE	SAL	TREND
SMITH	1980-12-17	800	FIRST
ALLEN	1981-02-20	1600	BETTER
WARD	1981-02-22	1250	WORST
JONES	1981-04-02	2975	BETTER
BLAKE	1981-05-01	2850	WORST
CLARK	1981-06-09	2450	WORST
TURNER	1981-09-08	1500	WORST
MARTIN	1981-09-28	1250	WORST
KING	1981-11-17	5000	BETTER
JAMES	1981-12-03	950	WORST
FORD	1981-12-03	3000	BETTER
MILLER	1982-01-23	1300	WORST
SCOTT	1987-04-19	3000	BETTER
ADAMS	1987-05-23	1100	WORST

MATCH_RECONGNIZE

ENAME	HIREDATE	SAL	TREND
SMITH	1980-12-17	800	FIRST
ALLEN	1981-02-20	1600	BETTER
WARD	1981-02-22	1250	WORST
JONES	1981-04-02	2975	BETTER
BLAKE	1981-05-01	2850	WORST
CLARK	1981-06-09	2450	WORST
TURNER	1981-09-08	1500	WORST
MARTIN	1981-09-28	1250	WORST
KING	1981-11-17	5000	BETTER
JAMES	1981-12-03	950	WORST
FORD	1981-12-03	3000	BETTER
MILLER	1982-01-23	1300	WORST
SCOTT	1987-04-19	3000	BETTER
ADAMS	1987-05-23	1100	WORST

JOIN USANDO 'AS'

- Uma relação de junção pode ser renomeada usando AS.

```
tpch=> SELECT r_name, n_name
tpch-> FROM region
tpch-> INNER JOIN nation
tpch-> ON (r_regionkey=n_regionkey) as NomeJunção;
```

Não suportado ainda pela versão que utilizamos na aula

EXERCÍCIOS

1. Crie uma consulta para exibir o preço total (o_totalprice) dos clientes do BRAZIL e o preço de varejo que não é duplicado .

```
(80950 rows)
```

EXERCÍCIOS

2. Crie uma consulta para exibir o preço total (o_totalprice) dos clientes do BRAZIL e o preço de varejo.

```
(260137 rows)
```

EXERCÍCIOS

3. Crie uma consulta para exibir as contas dos clientes que são preços de varejo e preço total de encomendas.

```
(118 rows)
```

EXERCÍCIOS

4. Crie uma consulta para as datas de envio que são datas de encomenda.

```
(2405 rows)
```