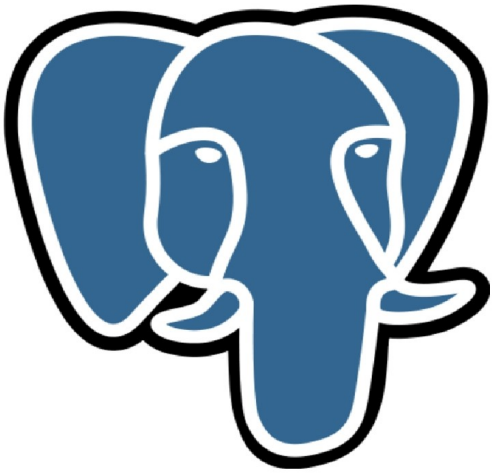

PostgreSQL Essentials

POLIBRÁS NET

Objetivos do Curso

*** Funções**

- **Conceito de função**
- **Funções matemáticas**
- **Funções de data e hora**
- **Máscaras de data e hora**
- **Funções de texto**
- **Funções de conversão de tipos**
- **Outras funções**
- **Funções de Agregação**
- **Cláusula GROUP BY**
- **Cláusula HAVING**
- **Funções de janela (Window Function)**

*** Sub-consultas**

- **Operadores de sub-consultas**
- **Sub-consultas no FROM**
- **Sub-consultas no SELECT**
- **Sub-consultas complexas**

Notação Utilizada no Curso

Itálico **: Utilizamos para nomes de objetos de banco de dados, arquivos e outros elementos que devem ser fornecidos pelo usuário.**

`Courier New` **: Os comandos, palavras-chave e saídas de comandos são grafados no tipo Courier.**

`Courier New` **: Utilizamos Courier New em negrito para enfatizar uma palavra-chave apresentada pela primeira vez e valores padrão.**

[ ] : Utilizamos colchetes sempre que uma palavra-chave for opcional.

{ } : Utilizamos chaves para delimitar uma lista de itens onde um deles deva ser escolhido.

... : Utilizamos três pontos para indicar que aquele item pode se repetir. Quando utilizado nos exemplos, indica uma parte não importante da informação foi removida.

Capítulo 12

Funções

- Uma função é um identificador que instrui o PostgreSQL a realizar uma operação dentro do comando SQL.

- O PostgreSQL possui diversos tipos de funções pré-definidas:

Funções matemáticas
Manipulação de data e hora
Manipulação de string
Conversão de tipos
Funções de sistema
Funções agregadas

Funções Matemáticas

- Funções matemáticas mais usuais:

abs(x) valor absoluto
ceil(x) menor inteiro maior ou igual ao argumento
floor(x) maior inteiro menor ou igual ao argumento
cos(x) cosseno
sin(x) seno
log(x) logaritmo de x na base 10
mod(x,y) resto inteiro da divisão de x por y
pow(x,y) eleva o número x à potência y
random() retorna um número aleatório entre 0.1 e 1.0
round(x [,y]) arredonda para a precisão informada
sign(x) retorna o sinal do argumento (-1, 0, 1)
sqrt(x) raiz quadrada
trunc(x [,y]) trunca o número até a casa decimal especificada

Funções Matemáticas

- Funções Matemáticas (exemplos):

```
SELECT pi(), log(53);
        pi        |       log
------------------+-----------------
 3.14159265358979 | 1.72427586960079


SELECT sin(30), pow(2, 5);
        sin         | pow
--------------------+-----
 -0.988031624092862 |  32


SELECT random(), random(), random();
    random     |     random     |     random
---------------+----------------+---------------
0.449414308857 | 0.842947382938 | 0.260392530291
```

Exemplos de Funções

- Funções matemáticas:

SELECT 9.4,ceil(9.4),round(9.4),sqrt(9.4);

```
 ?column? | ceil | round |             sqrt
----------+------+-------+------------------
      9.4 |    10|     9 | 3.06594194335118
```

- Também podemos utilizar “funções aninhadas”:

```
SELECT trunc(sqrt(floor(9.4)),2)^2;

 ?column?
--------------------
 9.0000000000000000
```

Funções de Data e Hora

- Funções de manipulação de data e hora mais usuais:

current_date	retorna a data atual no formato DATE
current_time	retorna a data atual no formato TIME
current_timestamp	retorna a data atual no formato TIMESTAMP
to_char(dt, masc)	converte a data para a máscara dada
date_part(s,t)	retorna a data ou hora t no formato s
date_trunc(s,t)	retorna a data ou hora t truncada no nível s
now()	retorna data e hora no formato TIMESTAMP
timeofday()	retorna data e hora no formato texto
age(dt, dt)	calcula o intervalo de tempo entre as datas

- Argumentos do date_part e date_trunc:

day	century	hour
minute	month	second
millisecond	week	year

Máscaras de Data e Hora

- Formato de máscaras de data e hora mais usuais:

HH	hora do dia (01-12)
HH12	hora do dia (01-12)
HH24	hora do dia (00-23)
MI	minuto (00-59)
SS	segundo (00-59)
MS	milisegundo (000-999)
US	microsegundo (000000-999999)
SSSS	segundo depois da meia-noite (0-86399)
AM ou PM	indicador do meridiano
YYYY	ano
YY	ano 2 últimos dígitos
Y	último dígito
BC ou AD	indicador de era
MONTH	nome completo do mês (maiúsculo)
MM	mês (01-12)
DAY	nome completo do dia (maiúsculo)
DD	dia do mês (01-31)
D	dia da semana (1-7, domingo=1)
DDD	dia do ano (1-366)
WW	semana do ano (1-53)

Usando Funções de Data e Hora

- Exemplo de funções de data e hora:

```
SELECT current_date + 1;

    date
------------
 2003-03-08

SELECT now();

              now
-------------------------------
 2003-03-08 12:01:32.057475-03

SELECT date_trunc('hour', now());

       date_trunc
-------------------------------
 2006-06-08 12:00:00-03

SELECT to_char(now(), 'DD-MM-YY HH24:MI:SS');

      to_char
-----------------------------
 08-06-06 12:13:22
```

Usando Funções de Data e Hora

- Exemplo de funções de data e hora:

```
SELECT u_conexao,
       u_codmaq,
       t_data,
       to_char(age(now(), t_data), 'YY "Years" MM "Meses" DD "dias" HH
"horas e" MI "minutos"')
    FROM pedcab ORDER BY u_conexao LIMIT 10;
```

```
  u_conexao | u_codmaq |        t_data         |                    to_char
-----------+----------+-----------------------+-----------------------------------------------
          0 |        8 | 2010-11-21 21:00:00-03 | 00 Years 09 Meses 14 dias 04 horas e 39 minutos
   23692942 |        3 | 2009-07-03 16:28:05-03 | 02 Years 02 Meses 02 dias 09 horas e 11 minutos
   23693815 |        3 | 2009-07-03 17:07:34-03 | 02 Years 02 Meses 02 dias 08 horas e 32 minutos
   23694456 |        3 | 2009-07-03 17:32:58-03 | 02 Years 02 Meses 02 dias 08 horas e 06 minutos
```

```
   SELECT current_date - '1 month'::interval,
          current_date - '1 month 8 hours'::interval;

      ?column?       |      ?column?
---------------------+---------------------
 2011-08-06 00:00:00 | 2011-08-05 16:00:00
```

Funções de Texto

- Funções de manipulação de texto mais usuais:

ascii(s)	retorna o código ASCII do caracter s
trim(s [,t])	elimina o caracter t (ou branco) no início e no final da string s
chr(n)	retorna o caracter correspondente ao número n
initcap(s)	torna maiúscula a primeira letra de cada palavra da string s
length(s)	comprimento da string s
lower(s)	retorna a string s em minúsculo
upper(s)	retorna a string s em maiúsculo
overlay(s placing b from x for y)	substitui substring da posição x com o comprimento y por b
position(b in s)	retorna a posição da string b na string s
repeat(s,n)	repete o caracter s n vezes

Funções de Texto

- Funções de manipulação de texto mais usuais:

lpad(s, n, b)	preenche a esquerda a string s com a substring b até o comprimento n
rpad(s, n, b)	preenche a direita a string s com a substring b até o comprimento n
split_part(s, d, n)	divide a string s pelo delimitador d e devolve o campo da coluna n
strpos(s, b)	retorna a posição da substring b na string s
substr(s,n,l)	retorna a string s começando em n e com comprimento l
translate(s, o, n)	retorna a string s substituindo as ocorrências de o por n

- Funções de manipulação de strings:

```
SELECT length(s_nome), upper(s_nome) FROM t_cliente ORDER BY 1;

length  |         upper
--------+---------------------
     19 | ACOUGUE DUNGAO LTDA
     20 | ROSILEIA PEREIRA ME

SELECT substr(s_nome, 1, 3) FROM t_cliente ;

 substr
 --------
 IGU
 CAS

SELECT  translate(s_nome, 'AEIOU', '43105')
  FROM t_cliente ;

translate
--------------------
1G54S5P3R LTD4
3 P D0S S4NT0S P4N1F1C4D0R4
C4S4 D3 C4RN3S D14S 3 D14S LTD4
```

Funções de Texto

- Funções de manipulação de strings:

```
SELECT split_part('Curso:PostgreSQL:Essencial', ':', 2);

 split_part
------------
 PostgreSQL

SELECT trim('*******Linux*******', '*');

 btrim
-------
 Linux

SELECT nome, lpad(s_nome, 20, ' ')
 FROM t_cliente;
                    s_nome                         |         lpad
---------------------------------------------------+----------------------
 IGUASUPER LTDA                                    |       IGUASUPER LTDA
 E P DOS SANTOS PANIFICADORA                       | E P DOS SANTOS PANIF
```

Funções de Conversão de Tipos

- O PostgreSQL oferece funções específicas para conversão de tipos de dados:

to_char(dt, masc)	converte data para texto conforme masc
to_char(num, masc)	converte número para texto conforme masc
to_timpestamp(str, masc)	converte texto para data conforme masc
to_number(str, masc)	converte texto para número conforme masc

- Máscaras para conversão numérica:

9	Valor com o número determinado de dígitos
0	Valor preenchendo início com zeros
D	Separador decimal
G	Separador de milhar
RN	Converte para romano
S	Sinal negativo
SG	Sinal negativo ou positivo

Funções de Conversão de Tipos

- Exemplos:

```
SELECT numero, to_char(data, 'dd/mm/yyyy')
 FROM concurso;
 numero |  to_char
--------+------------
      1 | 27/06/2006
      2 | 04/07/2006

SELECT numero, to_char(arrecadacao, '000G000G000D99SG')
 FROM concurso;
 numero |     to_char
--------+-----------------
      1 | 000,000,004,00+
      2 | 000,000,000,00+

SELECT    numero,    to_char(numero,    'RN')    FROM
concurso_numeros;
 numero |     to_char
--------+-----------------
     10 |                X
     12 |              XII
     11 |               XI
```

Informações de sessão

O PostgreSQL oferece várias funções para a obtenção de informações de sessão:

current_database()	retorna o nome do banco que a sessão acessa
current_schema()	retorna o nome do schema atual
current_user	retorna o nome do usuário atual
user	retorna o nome do usuário atual
version()	retorna a versão atual do banco
current_setting(str)	retorna o valor do parâmetro de configuração str

Funções de Agregação

- As funções de agregação transformam um conjunto de dados (linhas) num único valor.
- O PostgreSQL oferece as seguintes funções de agregação:

COUNT(coluna)	Número de valores não nulos na coluna
SUM(coluna)	Soma dos valores da coluna
AVG(coluna)	Média dos valores da coluna
MAX(coluna)	Maior valor da coluna
MIN(coluna)	Menor valor na coluna
COUNT(*)	Número de registros
STDDEV(coluna)	Desvio padrão
VARIANCE(coluna)	Calcula a variância estatística

Funções de Agregação

- Qual é o total de pedidos?

```
SELECT count(*) FROM pedcab;

 count
-------
  57647
```

- Qual é o maior valor de verba de um pedido?

```
SELECT max(d_cverba) FROM pedcab;

 max
-------
 4127.35
```

- Qual é o desvio padrão da verba de um pedido?

```
SELECT stddev(d_cverba) FROM  pedcab ;

         stddev
--------------------
    238.755213551943
```

A Cláusula GROUP BY

- A cláusula GROUP BY é utilizada numa query para agrupar colunas que tenham o mesmo valor em seus campos.
- O objetivo desta cláusula é reduzir um conjunto de linhas a um grupo que represente todas as linhas. Isto é feito para eliminar a redundância no resultado e para aplicar funções de agregação, a estes grupos.
- Sintaxe:

```
SELECT column list
FROM ...
[WHERE ...]
GROUP BY column list
```

Funções de Agregação com GROUP BY

- Como obter a quantidade de apostas por agente lotérico? Para isso utilizamos o GROUP BY e neste caso a função de agregação count.

```
SELECT s_codcliente, count(*)
 FROM t_prazo_cliente
 GROUP BY s_codcliente;
```

```
 s_codcliente | count
--------------+-------
 66200        |    10
 16876        |    60
 15525        |    15
 17531        |    20
 82042        |    50
 27828        |    20
 67147        |    10
 80211        |    20
 20408        |    10
 25871        |    20
 12189        |    30
 28062        |    20
 80946        |    20
 12521        |    20
 14927        |    30
 23575        |    12
```

A Cláusula HAVING

- Se uma tabela (ou várias) foi agrupada pelo GROUP BY e somente parte dos grupos resultantes são de interesse, a cláusula HAVING pode ser utilizada para eliminar parte dos grupos do resultado final.

No caso de restrição com valores agrupados, não poderíamos utilizar o WHERE para a comparação.

- Sintaxe:

```
SELECT column_list
FROM ...
[WHERE ...]
GROUP BY ...
HAVING boolean_expression
```

A cláusula HAVING está para o GROUP BY assim como o WHERE está para o SELECT.

- Exemplo de utilização da cláusula HAVING:

Selecionar os agentes lotéricos cuja quantidade de apostas seja maior que 200:

```
SELECT agente, count(*)
FROM aposta
GROUP BY agente
HAVING count(*) > 200;

agente | count
--------+-------
6 | 201
2 | 401
1 | 300
(3 rows)
```

Exercícios

```
Quantos Pedidos tem valor < 1000 Reais ?

Faça uma consulta que me retorne o vendedor e seu total em real vendido entre o
mês corrente e o mês anterior.
```

Window Function

Windows functions são um subconjunto de funções chamadas de “set functions” (funções de conjunto), ou seja, funções que são aplicadas em um conjunto de linhas. A palavra window (janela) se refere a um conjunto de linhas que a função trabalha.

Windows Functions são funções que foram introduzidas no padrão SQL:2003 controlado pela ISO e mais detalhadas no padrão SQL:2008. Já há algum tempo vários bancos de dados possuem suporte a Windows Functions, bem como: Oracle, Teradata, DB2 e até bancos gratuitos como PostgreSQL e Firebird (ainda tentando implementar).

- As funções de janela (window functions) são formas de consultar dados que complementam as tradicionais funções de agregação do SQL: SUM, COUNT, AVG, MAX e MIN.
- Possibilitam novas formas de acesso a um menor custo computacional e com uma sintaxe mais apropriada.
- Com as windows functions é possível utilizar somente um select para consultas, que se executadas sem esse recurso, exigiriam o uso de sub-consultas.
- Estas windows functions também ampliam o suporte do PostgreSQL para aplicações do tipo B.I. (Business Intelligence)

As operações realizadas pelas Window Functions estão relacionadas a:

Operações de numeração de registros (ROW_NUMBER()),

Classificação e ranqueamento (RANK(), DENSE_RANK(), PERCENT_RANK()),

Criação de subdivisões de uma partição (NTILE(), LAG() e LEAD()),

Recuperação de primeiro, último ou enésimo registro de uma window (FIRST_VALUE(), LAST_VALUE(), NTH_VALUE()),

E medida de distância ou posição relativa em relação ao início de uma partição para cada registro (CUME_DIST()), de acordo com a sua window.

Sintaxe para criação da Window Function:
function_name ([expression [, expression ...]]) OVER (window_definition)
function_name ([expression [, expression ...]]) OVER window_name
function_name (*) OVER (window_definition)
function_name (*) OVER window_name

onde window_definition:
[existing_window_name]
[PARTITION BY expression [, ...]]
[ORDER BY expression [ASC | DESC | USING operator]
[NULLS { FIRST | LAST }] [, ...]]
[frame_clause]

Exemplos:

Sem Window Function:

```
CREATE TEMP SEQUENCE rownum_seq;
SELECT NEXTVAL('rownum_seq'),
               s_codcliente,
               s_nome
   FROM t_cliente
    ORDER BY s_codcliente
;
```

```
 nextval | s_codcliente |                      s_nome
---------+--------------+-------------------------------------------------
       1 | 10000        | IGUASUPER LTDA
       2 | 10002        | E P DOS SANTOS PANIFICADORA
       3 | 10005        | CASA DE CARNES DIAS E DIAS LTDA
       4 | 10027        | SUPERMERCADO VALE VERDE LTDA
       5 | 10028        | POLIANAS COMERCIAL DE ALIMENTOS LTDA
       6 | 10033        | OSVALDO DA SILVA MEDEIROS
       7 | 10036        | UELINTON BUGE ME
       8 | 10037        | LACERDA OLIVEIRA DA SILVA ME
       9 | 10042        | COMERCIAL LOBO E PEREIRA LTDA
      10 | 10043        | DU CHEF COM E IND CARNES E DERIV LTDA
```

Com Window Function:

```
SELECT row_number() OVER (ORDER BY s_codcliente),
       s_codcliente,
       s_nome
  FROM t_cliente;
```

```
 row_number | s_codcliente | s_nome
------------------------------------------------
          1 | 10000        | IGUASUPER LTDA
          2 | 10002        | E P DOS SANTOS PANIFICADORA
          3 | 10005        | CASA DE CARNES DIAS E DIAS LTDA
          4 | 10027        | SUPERMERCADO VALE VERDE LTDA
          5 | 10028        | POLIANAS COMERCIAL DE ALIMENTOS LTDA
          6 | 10033        | OSVALDO DA SILVA MEDEIROS
          7 | 10036        | UELINTON BUGE ME
          8 | 10037        | LACERDA OLIVEIRA DA SILVA ME
          9 | 10042        | COMERCIAL LOBO E PEREIRA LTDA
         10 | 10043        | DU CHEF COM E IND CARNES E DERIV LTDA
```

```
SELECT t1.u_codmaq,
       t1.u_numero_polibras,
       t2.media_preco
  FROM pedcab t1
 INNER JOIN (SELECT u_codmaq,
                    u_numero_polibras,
                    TRUNC(AVG((d_quantidade*d_preco)::numeric),2) media_preco
               FROM pedcorp
              GROUP BY u_numero_polibras, u_codmaq) t2 ON t1.u_numero_polibras
= t2.u_numero_polibras
              ;

 u_codmaq | u_numero_polibras | media_preco
----------+-------------------+-------------
        3 |                 1 |      124.23
        3 |                 2 |      121.20
        3 |                 3 |       90.00
        3 |                 4 |      147.96
        3 |                 5 |       50.50
        3 |                 6 |      546.56
        3 |                 7 |      126.91
        3 |                 8 |       72.38
        3 |                 9 |      279.60
        3 |                10 |      244.80
        3 |                11 |      118.91
        3 |                12 |       83.78
(12 rows)
```

Exemplos:

Com Window Function:

```
SELECT t1.u_codmaq,
       t1.u_numero_polibras,
       AVG((t2.d_quantidade * t2.d_preco)::numeric) OVER (PARTITION BY
t1.u_numero_polibras)
  FROM pedcab t1
 INNER JOIN pedcorp t2 ON t1.u_numero_polibras = t2.u_numero_polibras
 GROUP BY 1,2;

 u_codmaq | u_numero_polibras |          avg
----------+-------------------+------------------------
        3 |                 1 |   124.2300000000000000
        3 |                 2 |   121.2000000000000000
        3 |                 3 |    90.0000000000000000
        3 |                 4 |   147.9650000000000000
        3 |                 5 |    50.5000000000000000
        3 |                 6 |   546.5600000000000000
        3 |                 7 |   126.9150000000000000
        3 |                 8 |    72.3800000000000000
        3 |                 9 |   279.6000000000000000
        3 |                10 |   244.8000000000000000
        3 |                11 |   118.9100000000000000
        3 |                12 |    83.7800000000000000
```

Capítulo 13

Sub-consultas

Sub-consultas

O PostgreSQL suporta sub-consultas desde a versão 6.3. Este recurso oferece flexibilidade para os comandos SQL.

O uso de sub-consultas ocorre quando os resultados de uma consulta são utilizados para restringir uma outra consulta.

Veremos a estrutura deste tipo de montagem, bem como mais alguns operadores importantes em seu uso.

O PostgreSQL permite o uso de sub-consultas na cláusula WHERE, FROM e SELECT.

Formato:
SELECT COL1, COL2, (SELECT...) ...
FROM TAB1, (SELECT ...) AS TAB2
WHERE condicao (SELECT ...)

Para saber quais concursos tiveram arrecadação acima da média de arrecadação:

```
SELECT s_codcliente
   FROM t_orgven_vendedor_cliente
 WHERE s_codvendedor in (SELECT s_codigo FROM t_vendedor WHERE u_codmaq = 1)
;
```

“A sub-consulta cria um conjunto de valores e o operador restringe os registros da pesquisa mais externa àqueles que estejam no conjunto formado pela sub-consulta.”

Operadores em Sub-consultas

O PostgreSQL oferece os seguintes operadores para a utilização em sub-consultas:

EXISTS (subconsulta)

O argumento do operador EXISTS é uma sub-consulta. Caso a sub-consulta retorne no mínimo uma linha, o resultado do EXISTS é true, caso contrário é false.

```
SELECT s_codcliente
   FROM t_orgven_vendedor_cliente t1
 WHERE EXISTS (SELECT 1 FROM t_vendedor WHERE u_codmaq = t1.u_codmaq AND
t1.u_codmaq = 1)
;
```

Expressao operador { ANY | SOME } (sub-consulta)

- Os operadores ANY e SOME são sinônimos.
- A expressão é comparada com todos as colunas resultantes da sub-consulta e retornará true se a comparação com qualquer valor da sub-consulta for bem sucedida.

Exemplo: Quantidade de acertos por aposta:

```
  SELECT s_codcliente
    FROM t_orgven_vendedor_cliente t1
   WHERE s_codvendedor = ANY (SELECT s_codigo FROM t_vendedor WHERE u_codmaq =
t1.u_codmaq AND t1.u_codmaq = 1)
       ;
```

Expressao operador ALL (sub-consulta)

A expressão é comparada com todos as colunas resultantes da sub-consulta e retornará true se todas as comparações forem bem sucedidas.

```
SELECT COUNT(*) FROM (
  SELECT s_codcliente
    FROM t_orgven_vendedor_cliente t1
   WHERE s_codvendedor <> ALL (SELECT s_codigo FROM t_vendedor WHERE u_codmaq =
t1.u_codmaq AND t1.u_codmaq = 1)
       ) A ;
```

Exercícios

1. Selecionar o numero total de pedidos dos top 10 clientes, mostrar o numero de pedidos e o nome do cliente.

2. Quantos produtos o vendedor codmaq 1 deverá conter na sua tabela ftsl_prod no PALM ?
Leve em consideração apenas produto, grupo, grupo_vendedor e grupo_produto.

3. Gere uma lista por data do mês anterior com os valores faturados por dia, pode ser usado a function generate_series para gerar todos os dias do mês, caso não tenha venda no dia deverá ser retornado zero

4. Gerar o valor de venda em percentual dos vendedores do mês anterior e quem vendeu mais, ordenar decrescente