7) Reuso de Código: Utilizando Funções

7.1) Introdução às Funções - 1
7.2) Utilizando Funções matemáticas - 2
7.3) Utilizando Funções de data e hora - 4
7.4) Utilizando Funções de Texto (Strings) - 7
7.5) Utilizando Funções de Conversão de Tipos (Casts) - 10
7.6) Utilizando Outras Funções - 11
7.7) Entendendo as Funções de Agregação - 14

7.1) Introdução às Funções

O PostgreSQL fornece um grande número de funções e operadores para os tipos de dado nativos. Os usuários também podem definir suas próprias funções e operadores, conforme descrito na Parte V. Os comandos \df e \do do psql podem ser utilizados para mostrar a lista de todas as funções e operadores disponíveis, respectivamente.

Havendo preocupação quanto à portabilidade, deve-se ter em mente que a maioria das funções e operadores descritos neste capítulo, com exceção dos operadores mais triviais de aritmética e de comparação, além de algumas funções indicadas explicitamente, não são especificadas pelo padrão SQL. Algumas das funcionalidades estendidas estão presentes em outros sistemas gerenciadores de banco de dados SQL e, em muitos casos, estas funcionalidades são compatíveis e consistentes entre as várias implementações.

Detalhes em: http://pgdocptbr.sourceforge.net/pg80/functions.html

7.2) Utilizando Funções matemáticas

Operadores Matemáticos
+, -, *, /, % (somar, subtrair, multiplicar, dividir, módulo (resto de divisão de inteiros)),
^(potência),
!(fatorial),
@(valor absoluto)
| / - rais quadrada (| / 25.0 = 5)
| | / - raiz cúbica (| | / 27.0 = 3)

Algumas funções Matemáticas
ABS(x) - valor absoluto de x
CEIL(numeric) - arredonda para o próximo inteiro superior
DEGREES(valor) - converte valor de radianos para graus
FLOOR(numeric) - arredonda para o próximo inteiro inferior
MOD(x,y) - resto da divisão de x por y
PI() - constante PI (3,1415...)
POWER(x,y) - x elevado a y
RADIANS(valor) - converte valor de graus para radianos
RANDOM() - valor aleatório entre 0 e 1
ROUND(numeric) - arredonda para o inteiro mais próximo
ROUND(v, d) - arredonda v com d casas decimais
SIGN(numeric) - retorna o sinal da entrada, como -1 ou +1
SQRT(X) - Raiz quadrada de X
TRUNC (numeric) - trunca para o nenhuma casa decimal
TRUNC (v numeric, s int) - trunca para s casas decimais

Operadores Lógicos:

AND, OR e NOT. TRUE, FALSE e NULL

Operadores de Comparação:

<, >, <=, >=, =, <> ou !=
a BETWEEN x AND y
a NOT BETWEEN x AND y
expressão IS NULL
expressão IS NOT NULL
expressão IS TRUE
expressão IS NOT TRUE
expressão IS FALSE
expressão IS NOT FALSE
expressão IS UNKNOWN
expressão IS NOT UNKNOWN

GREATEST - **Retorna o maior valor de uma lista - SELECT GREATEST(1,4,6,8,2); - - 8**

LEAST **- Retorna o menor valor de uma lista.**

Todos os valores da lista devem ser do mesmo tipo e nulos são ignorados.
Obs.: Ambas as funções acima não pertencem ao SQL standard, mas são uma extensão do PostgreSQL.

7.3) Utilizando Funções de data e hora
Operações com datas:

timestamp '2001-09-28 01:00' + interval '23 hours' -> timestamp '2001-09-29 00:00'
date '2001-09-28' + interval '1 hour' -> timestamp '2001-09-28 01:00'
date '01/01/2006' – date '31/01/2006'
time '01:00' + interval '3 hours'time -> '04:00'
interval '2 hours' - time '05:00' -> time '03:00:00'

Função age (retorna Interval) - Diferença entre datas
age(timestamp)interval (Subtrai de hoje)
age(timestamp '1957-06-13') -> 43 years 8 mons 3 days
age(timestamp, timestamp)interval Subtrai os argumentos
age('2001-04-10', timestamp '1957-06-13') -> 43 years 9 mons 27 days

Função extract (retorna double)
Extrai parte da data: ano, mês, dia, hora, minuto, segundo.
select extract(year from age('2001-04-10', timestamp '1957-06-13'))
select extract(month from age('2001-04-10', timestamp '1957-06-13'))
select extract(day from age('2001-04-10', timestamp '1957-06-13'))

Data e Hora atuais (retornam data ou hora)
SELECT CURRENT_DATE;
SELECT CURRENT_TIME;
SELECT CURRENT_TIME(0);
SELECT CURRENT_TIMESTAMP;

SELECT CURRENT_TIMESTAMP(0);

Somar dias e horas a uma data:
SELECT CAST('06/04/2006' AS DATE) + INTERVAL '27 DAYS' AS Data;

Função now (retorna timestamp with zone)
now() - Data e hora corrente (timestamp with zone);
Não usar em campos somente timestamp.

Função date_part (retorna double)
SELECT date_part('day', TIMESTAMP '2001-02-16 20:38:40');
Resultado: 16 (day é uma string, diferente de extract)

Obtendo o dia da data atual:
SELECT DATE_PART('DAY', CURRENT_TIMESTAMP) AS dia;

Obtendo o mês da data atual:
SELECT DATE_PART('MONTH', CURRENT_TIMESTAMP) AS mes;

Obtendo o ano da data atual:
SELECT DATE_PART('YEAR', CURRENT_TIMESTAMP) AS ano;

Função date_trunc (retorna timestamp)
SELECT date_trunc('year', TIMESTAMP '2001-02-16 20:38:40');
Retorna 2001-02-16 00:00:00

Convertendo (CAST)
select to_date('1983-07-18', 'YYYY-MM-DD')
select to_date('19830718', 'YYYYMMDD')

Função timeofday (retorna texto)
select timeofday() -> Fri Feb 24 10:07:32.000126 2006 BRT

Interval
interval [(*p*)]
to_char(interval '15h 2m 12s', 'HH24:MI:SS')
date '2001-09-28' + interval '1 hour'
interval '1 day' + interval '1 hour'
interval '1 day' - interval '1 hour'
900 * interval '1 second'
Interval trabalha com as unidades: second, minute, hour, day, week, month, year, decade, century, millenium ou abreviaturas ou plurais destas unidades.
Se informado sem unidades '13 10:38:14' será devidamente interpretado '13 days 10 hours 38 minutes 14 seconds'.

CURRENTE_DATE - INTERVAL '1' day;
TO_TIMESTAMP('2006-01-05 17:56:03', 'YYYY-MM-DD HH24:MI:SS')

Operações com Datas

- A Diferença entre dois Timestamps é sempre um Interval

TIMESTAMP '2001-12-31' – TIMESTAMP '2001-12-11' = INTERVAL '19 days'

- Adicionar ou subtrair um Interval com um Timestamp produzirá um Timestamp

TIMESTAMP '2001-12-11' + INTERVAL '19 days' = TIMESTAMP '2001-12-30'

- Adicionar ou suntrair dois Interval acarreta em outro Interval:

INTERVAL '1 month' + INTERVAL '1 month 3 days' = INTERVAL '2 months 3 days'
Dica: Para a tradução podemos usar algumas funções que trabalham com string para varrer o resultado traduzindo.

- Não podemos efetuar operações de Adição, Multiplicação ou Divisão com dois Timestamps:

Causará Erro.

- A diferença entre dois valores tipo Date é um Integer representando o número de dias:

DATE '2001-12-30' – DATE '2001-12-11' = INTEGER 19

- Adicionando ou subtraindo Integer para um Date produzirá um Date:

DATE '2001-12-13' + INTEGER 7 = DATE '2001-12-20'

- Não podemos efetuar operações de Adição, Multiplicação ou Divisão com dois Dates:

Causará Erro.

Como testar estes exemplos no PostgreSQL?

SELECT DATE '2001-12-30' – DATE '2001-12-11';

7.4) Utilizando Funções de Texto (Strings)

***Concatenação de Strings* - dois || (pipes)**
SELECT 'ae' || 'io' || 'u' AS vogais; --vogais -------- aeiou
SELECT CHR(67)||CHR(65)||CHR(84) AS "Dog"; -- Dog CAT

Quantidade de Caracteres de String
char_length - retorna o número de caracteres
SELECT CHAR_LENGTH('Evolução'); - -Retorna 8

Ou SELECT LENGTH('Database'); - - Retorna 8

Converter para minúsculas
SELECT LOWER('UNIFOR');

Converter para maiúsculas
SELECT UPPER('universidade');

Posição de caractere
SELECT POSITION ('@' IN 'ribafs@gmail.com'); -- Retorna 7
Ou SELECT STRPOS('Ribamar' ,'mar'); - - Retorna 5

Substring
SUBSTRING(string [FROM inteiro] [FOR inteiro])
SELECT SUBSTRING ('Ribamar FS' FROM 9 FOR 10); - - Retorna FS
SUBSTRING(string FROM padrão);
SELECT SUBSTRING ('PostgreSQL' FROM '.......'); - - Retorna Postgre
SELECT SUBSTRING ('PostgreSQL' FROM '...$'); - -Retorna SQL

Primeiros ..... e últimos ...$

Ou
SUBSTR ('string', inicio, quantidade);
SELECT SUBSTR ('Ribamar', 4, 3); - - Retorna mar

Substituir todos os caracteres semelhantes
SELECT TRANSLATE(string, velho, novo);
SELECT TRANSLATE('Brasil', 'il', 'ão'); - - Retorna Brasão
SELECT TRANSLATE('Brasileiro', 'eiro', 'eira');

Remover Espaços de Strings
SELECT TRIM('   SQL - PADRÃO   ');

Calcular MD5 de String
SELECT MD5('ribafs'); - - Retorna 53cd5b2af18063bea8ddc804b21341d1

Repetir uma string n vezes
SELECT REPEAT('SQL-', 3); - - Retorna SQL-SQL-SQL-

Sobrescrever substring em string
SELECT REPLACE ('Postgresql', 'sql', 'SQL'); - - Retorna PostgreSQL

Dividir Cadeia de Caracteres com Delimitador
SELECT SPLIT_PART('PostgreSQL', 'gre', 2); - -Retorna SQL
SELECT SPLIT_PART('PostgreSQL', 'gre', 1); - -Retorna Post

<------gre----->

Iniciais Maiúsculas

INITCAP(text) - INITCAP ('olá mundo') - - Olá Mundo

Remover Espaços em Branco
TRIM ([leading | trailing | both] [characters] from string)- remove caracteres da direita e da esquerda. trim (both 'b' from 'babacatebbbb'); - - abacate
RTRIM (string text, chars text) - Remove os caracteres chars da direita (default é espaço)
rtrim('removarrrr', 'r') - - remova
LTRIM - (string text, chars text) - Remove os caracteres chars da esquerda
ltrim('abssssremova', 'abs') - - remova

Detalhes no item 9.4 do Manual:
http://pgdocptbr.sourceforge.net/pg80/functions-string.html

Like e %
SELECT * FROM FRIENDS WHERE LASTNAME LIKE 'M%';

O ILIKE é case INsensitive e o LIKE case sensitive.

~~ equivale ao LIKE
~~* equivale equivale ao ILIKE
!~~ equivale ao NOT LIKE
!~~* equivale equivale ao NOT ILIKE

... LIKE '[4-6]_6%' -- Pegar o primeiro sendo de 4 a 6,
-- o segundo qualquer dígito,
-- o terceiro sendo 6 e os demais quaisquer

% similar a *
_ similar a ? (de arquivos no DOS)

Correspondência com um Padrão
O PostgreSQL disponibiliza três abordagens distintas para correspondência com padrão: o operador LIKE tradicional do SQL; o operador mais recente SIMILAR TO (adicionado ao SQL:1999); e as expressões regulares no estilo POSIX. Além disso, também está disponível a função de correspondência com padrão substring, que utiliza expressões regulares tanto no estilo SIMILAR TO quanto no estilo POSIX.

SELECT substring('XY1234Z', 'Y*([0-9]{1,3})'); - - Resultado: 123
SELECT substring('XY1234Z', 'Y*?([0-9]{1,3})'); - - Resultado: 1

SIMILAR TO
O operador SIMILAR TO retorna verdade ou falso conforme o padrão corresponda ou não à cadeia de caracteres fornecida. Este operador é muito semelhante ao LIKE, exceto por interpretar o padrão utilizando a definição de expressão regular do padrão SQL.

'abc' SIMILAR TO 'abc' verdade
'abc' SIMILAR TO 'a' falso
'abc' SIMILAR TO '%(b|d)%' verdade
'abc' SIMILAR TO '(b|c)%' falso

SELECT 'abc' SIMILAR TO '%(b|d)%'; -- Procura b ou d em 'abc' e no caso retorna TRUE
REGEXP

SELECT 'abc' ~ '.*ab.*';

~ distingue a de A
~* não distingue a de A
!~ distingue expressões distingue a de A
!~* distingue expressões não distingue a de A
'abc' ~ 'abc' -- TRUE
'abc' ~ '^a' -- TRUE
'abc' ~ '(b|j)' -- TRUE
'abc' ~ '^(b|c)' -- FALSE

7.5) Utilizando Funções de Conversão de Tipos (Casts)

Conversão Explícita de Tipos (CAST)

CAST (expressão AS tipo) AS apelido; -- Sintaxe SQL ANSI

Outra forma:

Tipo (expressão);

Exemplo:

SELECT DATE '10/05/2002' - DATE '10/05/2001'; -- Retorna a quantidade de dias
- -entre as duas datas

Para este tipo de conversão devemos:
Usar float8 ao invés de double precision;
Usar entre aspas alguns tipos como interval, time e timestamp
Obs.: aplicações portáveis devem evitar esta forma de conversão e em seu lugar usar o CAST explicitamente.

A função CAST() é utilizada para converter explicitamente tipos de dados em outros.
SELECT CAST(2 AS double precision) ^ CAST(3 AS double precision) AS "exp";
SELECT ~ CAST('20' AS int8) AS "negativo"; - Retorna -21
SELECT round(CAST (4 AS numeric), 4); - Retorna 4.0000
SELECT substr(CAST (1234 AS text), 3);
SELECT 1 AS "real" UNION SELECT CAST('2.2' AS REAL);

7.6) Utilizando Outras Funções

Tipos Geométricos:
CREATE TABLE geometricos(ponto POINT, segmento LSEG, retangulo BOX, poligono POLYGON, circulo CIRCLE);

ponto (0,0),
segmento de (0,0) até (0,1),
retângulo (base inferior (0,0) até (1,0) e base superior (0,1) até (1,1)) e
círculo com centro em (1,1) e raio 1.
INSERT INTO geometricos VALUES ('(0,0)','((0,0),(0,1))', '((0,0),(0,1))', '((0,0),(0,1),(1,1),(1,0))','((1,1),1)');

Tipos de Dados para Rede:
Para tratar especificamente de redes o PostgreSQL tem os tipos de dados cidr, inet e macaddr.

cidr – para redes IPV4 e IPV6
inet – para redes e hosts IPV4 e IPV6
macaddr – endereços MAC de placas de rede

Assim como tipos data, tipos de rede devem ser preferidos ao invés de usar tipos texto para guardar IPs, Máscaras ou endereços MAC.

Veja um exemplo em Índices Parciais e a documentação oficial para mais detalhes.

Formatação de Tipos de Dados

TO_CHAR - Esta função deve ser evitada, pois está prevista sua descontinuação.

TO_DATE
date TO_DATE(text, text); Recebe dois parâmetros text e retorna date.
Um dos parâmetros é a data e o outro o formato.

SELECT TO_DATE('29032006','DDMMYYYY'); - Retorna 2006-03-29

TO_TIMESTAMP
tmt TO_TIMESTAMP(text,text) - Recebe dois text e retorna timestamp with zone

SELECT TO_TIMESTAMP('29032006 14:23:05','DDMMYYYY HH:MI:SS'); - Retorna 2006-03-29 14:23:05+00

TO_NUMBER
numeric TO_NUMBER(text,text)
SELECT TO_NUMBER('12,454.8-', '99G999D9S'); Retorna -12454.8
SELECT TO_NUMBER('12,454.8-', '99G999D9'); Retorna 12454.8
SELECT TO_NUMBER('12,454.8-', '99999D9'); Retorna 12454

Detalhes no item 9.8 do manual oficial.

Funções Diversas
SELECT CURRENT_DATABASE();
SELECT CURRENT_SCHEMA();
SELECT CURRENT_SCHEMA(boolean);
SELECT CURRENT_USER;
SELECT SESSION_USER;
SELECT VERSION();

SELECT CURRENT_SETTING('DATESTYLE');
SELECT HAS_TABLE_PRIVILEGE('usuario','tabela','privilegio');
SELECT HAS_TABLE_PRIVILEGE('postgres','nulos','insert'); - - Retorna: t
SELECT HAS_DATABASE_PRIVILEGE('postgres','testes','create'); - - Retorna: t
SELECT HAS_SCHEMA_PRIVILEGE('postgres','public','create'); - - Retorna: t

SELECT relname FROM pg_class WHERE pg_table_is_visible(oid);

Arrays

SELECT ARRAY[1.1,2.2,3.3]::INT[] = ARRAY[1,2,3];
SELECT ARRAY[1,2,3] = ARRAY[1,2,8];
SELECT ARRAY[1,3,5] || ARRAY[2,4,6];
SELECT 0 || ARRAY[2,4,6];

Array de char com 48 posições e cada uma com 2:
campo char(2) [48]

Funções Geométricos

area(objeto) - - area(box '((0,0), (1,1))');
center(objeto) - - center(box '((0,0), (1,2))');
diameter(circulo double) - - diameter(circle '((0,0), 2.0)');
height(box) - - height(box '((0,0), (1,1))');
length(objeto) - - length(path '((-1,0), (1,0))');
radius(circle) - - radius(circle '((0,0), 2.0)');
width(box) - - width(box '((0,0), (1,1))');

Funções para Redes

Funções cidr e inet

host(inet) - - host('192.168.1.5/24') - - 192.168.1.5
masklen(inet) - - masklen('192.168.1.5/24') - - 24
netmask(inet) - - netmask('192.168.1.5/24') - - 255.255.255.0
network(inet) - - network('192.168.1.5/24') - - 192.168.1.0/24

Função macaddr

trunt(macaddr) - - trunc(maraddr '12:34:34:56:78:90:ab') - - 12:34:56:00:00:00

Funções de Informação do Sistema

current_database()
current_schema()
current_schemas(boolean)
current_user()
inet_client_addr()
inet_client_port()
inet_server_addr()
inet_server_port()
pg_postmaster_start_time()
version()
has_table_privilege(user, table, privilege) - dá privilégio ao user na tabela

has_table_privilege(table, privilege) - dá privilégio ao usuário atual na tabela
has_database_privilege(user, database, privilege) - dá privilégio ao user no banco
has_function_privilege(user, function, privilege) - dá privilégio ao user na função
has_language_privilege(user, language, privilege) - dá privilégio ao user na linguagem
has_schema_privilege(user, schema, privilege) - dá privilégio ao user no esquema
has_tablespace_privilege(user, tablespace, privilege) - dá privilégio ao user no tablespace
current_setting(nome) - valor atual da configuração
set_config(nome, novovalor, is_local) - seta parâmetro de retorna novo valor

pg_start_backup(label text)
pg_stop_backup()
pg_column_size(qualquer)
pg_tablespace_size(nome)
pg_database_size(nome)
pg_relation_size(nome)
pg_total_relation_size(nome)
pg_size_pretty(bigint)

pg_ls_dir(diretorio)
pg_read_file(arquivo text, offset bigint, tamanho bigint)
pg_stat_file(arquivo text)

7.7) Entendendo as Funções de Agregação (Agrupamento)

As funções de agrupamento são usadas para contar o número de registros de uma tabela.

avg(expressão)
count(*)
count(expressão)
max(expressão)
min(expressão)
stddev(expressão)
sum(expressão)
variance(expressão)

Onde expressão, pode ser "ALL expressão" ou “DISTINCT expressão”.

count(distinct expressão)

As funções de Agrupamento (agregação) não podem ser utilizadas na cláusula WHERE. Devem ser utilizadas entre o SELECT e o FROM.

Dica: Num SELECT que usa uma função agregada, as demais colunas devem fazer parte da cláusula GROUP BY. Somente podem aparecer após o SELECT ou na cláusula HAVING. De uso proibido nas demais cláusulas.

Dica2: Ao contar os registros de uma tabela com a função COUNT(campo) e esse campo for nulo em alguns registros, estes registros não serão computados, por isso cuidado com os nulos também nas funções de agregação. Somente o count(*) conta os nulos.

A cláusula HAVING normalmente vem precedida de uma cláusula GROUP BY e obrigatoriamente contém funções de agregação.

ALERTA: Retornam somente os registros onde o campo pesquisado seja diferente de NULL.

NaN - Not a Number (Não é um número)

UPDATE tabela SET campo1 = 'NaN';

Exemplos:

SELECT MIN(campo) AS "Valor Mínimo" FROM tabela;

Caso tenha problema com esta consulta use:
SELECT campo FROM tabela ORDER BY campo ASC LIMIT 1; -- trará o menor

SELECT MAX(campo) AS "Valor Máximo" FROM tabela;

Caso tenha problema com esta consulta use:
SELECT campo FROM tabela ORDER BY campo DESC LIMIT 1; -- trará o maior