PostgreSQL

Funcions

A. Moll

Ací teniu un extracte del gran ventall de funcions incorporades que ofereix PostgreSQL

1. FUNCIONS MATEMÀTIQUES

Function	Return Type	Description	Example	Result
abs(x)	(same as input)	absolute value	abs(-17.4)	17.4
ceil(dp or numeric)	(same as input)	smallest integer not less than argument	ceil(-42.8)	-42
div(y numeric, x numeric)	numeric	integer quotient of y/x	div(9,4)	2
exp(dp or numeric)	(same as input)	exponential	exp(1.0)	2.71828182845905
floor(dp or numeric)	(same as input)	largest integer not greater than argument	floor(-42.8)	-43
mod(y, x)	(same as argument types)	remainder of y/x	mod(9,4)	1
pi()	dp	"π" constant	pi()	3.14159265358979
power(a dp, b dp)	dp	a raised to the power of b	power(9.0, 3.0)	729
random()	dp	random value in the range 0.0 <= x < 1.0	random()	
round(dp or numeric)	(same as input)	round to nearest integer	round(42.4)	42
round(v numeric, s int)	numeric	round to s decimal places	round(42.4382, 2)	42.44
sqrt(dp or numeric)	(same as input)	square root	sqrt(2.0)	1.4142135623731
trunc(dp or numeric)	(same as input)	truncate toward zero	trunc(42.8)	42
trunc(v numeric, s int)	numeric	truncate to s decimal places	trunc(42.4382, 2)	42.43
ETC....		Funcions trigonomètriques etc.		

2. DE CADENES

Before PostgreSQL 8.3, these functions would silently accept values of several non-string data types as well, due to the presence of implicit coercions from those data types to `text`. Those coercions have been removed because they frequently caused surprising behaviors.

Function	Description	Example	Result
`lower(string)`	Convert string to lower case	`lower('TOM')`	`tom`
`upper(string)`	Convert string to upper case	`upper('tom')`	`TOM`
`length(string)`	Number of characters in `string`	`length('jose')`	`4`
`position(substring in string)`	Location of specified substring	`position('om' in 'Thomas')`	`3`
`substring(string from pattern)`	Extract substring matching POSIX regular expression.	`substring('Thomas' from '...$')`	`mas`
`substring(string from pattern for escape)`	Extract substring matching SQL regular expression..	`substring('Thomas' from '%#"o_a#"_' for '#')`	`oma`
`substr(string, from [, count])`	Extract substring (same as `substring(string from from for count)`)	`substr('alphabet', 3, 2)`	`ph`
`ascii(string)`	ASCII code of the first character of the argument.	`ascii('x')`	`120`
`chr(int)`	Character with the given code.	`chr(65)`	`A`
`initcap(string)`	Convert the first letter of each word to upper case and the rest to lower case. Words are sequences of alphanumeric characters separated by non-alphanumeric characters.	`initcap('hi THOMAS')`	`Hi Thomas`
`left(str text, n int)`	Return first `n` characters in the string. When `n` is	`left('abcde', 2)`	`ab`

	negative, return all but last |n| characters.		
right(str text, n int)	Return last n characters.	right('abcde', 2)	de
rpad(string text, length int [, fill text])	Fill up the string to length length by appending the characters fill (a space by default). If the string is already longer than length then it is truncated.	rpad('hi', 5, 'xy')	hixyx
lpad(string text, length int [, fill text])	Fill up the string to length length by prepending the characters fill (a space by default). If the string is already longer than length then it is truncated (on the right).	lpad('hi', 5, 'xy')	xyxhi
rtrim(string text [, characters text])	Remove the longest string containing only characters from characters (a space by default) from the end of string	rtrim('trimxxxx', 'x')	trim
ltrim(string text [, characters text])	Remove the longest string containing only characters from characters (a space by default) from the start of string	ltrim('zzzytrim', 'xyz')	trim
btrim(string text [, characters text])	Remove the longest string consisting only of characters in characters (a space by default) from the start and end of string	btrim('xyxtrimyyx', 'xy')	trim
pg_client_encoding()	Current client encoding name	pg_client_encoding()	SQL_ASCII
reverse(str)	Return reversed string.	reverse('abcde')	edcba
md5(string)	Calculates the MD5 hash of string, returning the result in hexadecimal	md5('abc')	900150983cd24fb0 d6963f7d28e17f72
translate(string text, from text, to text)	Any character in string that matches a character in the from set is replaced by the corresponding character in the to set. If from is longer than to, occurrences of the extra characters in from are removed.	translate('12345', '143', 'ax')	a2x5

3. FUNCIONS I OPERADORS DE DATA I HORA

3.1. Funcions

Function	Return Type	Description	Example	Result
`age(timestamp, timestamp)`	`interval`	Subtract arguments, producing a "symbolic" result that uses years and months	`age(timestamp '2001-04-10', timestamp '1957-06-13')`	`43 years 9 mons 27 days`
`age(timestamp)`	`interval`	Subtract from `current_date` (at midnight)	`age(timestamp '1957-06-13')`	`43 years 8 mons 3 days`
`current_date`	`date`	Current date		
`current_time`	`time with time zone`	Current time of day		
`current_timestamp`	`timestamp with time zone`	Current date and time (start of current transaction		
`date_trunc(text, timestamp)`	`timestamp`	Truncate to specified precision	`date_trunc('hour', timestamp '2001-02-16 20:38:40')`	`2001-02-16 20:00:00`
`extract(field from timestamp)`	`double precision`	Get subfield	`extract(hour from timestamp '2001-02-16 20:38:40')`	`20`
`extract(field from interval)`	`double precision`	Get subfield	`extract(month from interval '2 years 3 months')`	`3`
`now()`	`timestamp with time zone`	Current date and time (start of current transaction)		

3.2. Date/Time Operators

Operator	Example	Result
+	date '2001-09-28' + integer '7'	date '2001-10-05'
+	date '2001-09-28' + interval '1 hour'	timestamp '2001-09-28 01:00:00'
+	date '2001-09-28' + time '03:00'	timestamp '2001-09-28 03:00:00'
+	timestamp '2001-09-28 01:00' + interval '23 hours'	timestamp '2001-09-29 00:00:00'
+	time '01:00' + interval '3 hours'	time '04:00:00'
-	- interval '23 hours'	interval '-23:00:00'
-	date '2001-10-01' - date '2001-09-28'	integer '3' (days)
-	date '2001-10-01' - integer '7'	date '2001-09-24'
-	date '2001-09-28' - interval '1 hour'	timestamp '2001-09-27 23:00:00'
-	time '05:00' - time '03:00'	interval '02:00:00'
-	time '05:00' - interval '2 hours'	time '03:00:00'
-	timestamp '2001-09-28 23:00' - interval '23 hours'	timestamp '2001-09-28 00:00:00'
-	interval '1 day' - interval '1 hour'	interval '1 day -01:00:00'
-	timestamp '2001-09-29 03:00' - timestamp '2001-09-27 12:00'	interval '1 day 15:00:00'

4. CONVERSIÓ DE TIPUS

Function	Return Type	Example
to_char(timestamp, text)	text	to_char(current_timestamp, 'HH12:MI:SS')
to_char(interval, text)	text	to_char(interval '15h 2m 12s', 'HH24:MI:SS')
~~to_char(int, text)~~	~~text~~	~~to_char(125, '999')~~
~~to_char(double precision, text)~~	~~text~~	~~to_char(125.8::real, '999D9')~~
~~to_char(numeric, text)~~	~~text~~	~~to_char(-125.8, '999D99S')~~
to_date(text, text)	date	to_date('05 Dec 2000', 'DD Mon YYYY')
to_number(text, text)	numeric	to_number('12,454.8-', '99G999D9S')
to_timestamp(text, text)	timestamp with time zone	to_timestamp('05 Dec 2000', 'DD Mon YYYY')
to_timestamp(double precision)	timestamp with time zone	to_timestamp(1284352323)

Amplieu informacio sobre els patrons que es fan servir en aquestes funcions. Per exemple:

- **To_char (485, ´9 9 9´) → '4 8 5'**
- **to_char(to_date('20140527','YYYYMMDD') , 'MONTH') → *MAY***