Tipos de dato en postgresql

Tipo numéricos

Los tipos numéricos consisten en enteros de dos, cuatro y ocho bytes, números de punto flotante de cuatro y ocho bytes y decimales de precisión seleccionable.

Name	Storage Size	Description	Range
smallint	2 bytes	small-range integer	-32768 to +32767
integer	4 bytes	typical choice for integer	-2147483648 to +2147483647
bigint	8 bytes	large-range integer	-9223372036854775808 to +9223372036854775807
decimal	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
numeric	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
real	4 bytes	variable-precision, inexact	6 decimal digits precision
double precision	8 bytes	variable-precision, inexact	15 decimal digits precision
smallserial	2 bytes	small autoincrementing integer	1 to 32767
serial	4 bytes	autoincrementing integer	1 to 2147483647
bigserial	8 bytes	large autoincrementing integer	1 to 9223372036854775807

Los tipos *smallint*, *integer* y *bigint* almacenan números enteros, es decir, números sin componentes fraccionarios, de varios rangos. Los intentos de almacenar valores fuera del rango permitido darán lugar a un error.

El tipo *integer* es la elección común, ya que ofrece el mejor equilibrio entre rango, tamaño de almacenamiento y rendimiento. El tipo *smallint* se utiliza generalmente sólo si el espacio en el disco es escaso. El tipo *bigint* está diseñado para ser utilizado cuando el rango del tipo *integer* es insuficiente.

SQL sólo especifica los tipos enteros *integer* (o int), *smallint* y *bigint*. Los nombres de los tipos *int2*, *int4* e *int8* son extensiones, que también son utilizadas por algunos otros sistemas de bases de datos SQL.

Números de precisión arbitraria

El tipo *numeric* puede almacenar números con un gran número de dígitos. Se recomienda especialmente para almacenar importes monetarios y otras cantidades que requieran exactitud. Los cálculos con valores numéricos dan resultados exactos siempre que sea posible, por ejemplo, sumas, restas o multiplicaciones. Sin embargo, los cálculos con valores numéricos son muy lentos en comparación con los tipos enteros, o con los tipos de punto flotante que se describen en la siguiente sección.

Float

Los tipos de datos *real* y *double precision* son tipos numéricos inexactos de precisión variable. Inexacto significa que algunos valores no pueden ser convertidos exactamente al

formato interno y se almacenan como aproximaciones, por lo que el almacenamiento y la recuperación de un valor pueden mostrar ligeras discrepancias.

Los tipos de datos **smallserial**, **serial** y **bigserial** no son verdaderos tipos de dato, sino simplemente una conveniencia notacional para crear columnas de identificadores únicos

Tipo carácter

Name	Description
`character varying(n)`, `varchar(n)`	variable-length with limit
`character(n)`, `char(n)`	fixed-length, blank padded
`text`	variable unlimited length

Se definen character *varying*(n) y *character*(n), donde n es un número entero positivo. Ambos tipos pueden almacenar cadenas de hasta n caracteres (no bytes) de longitud. Un intento de almacenar una cadena más larga en una columna de estos tipos dará lugar a un error, a menos que los caracteres sobrantes sean todos espacios, en cuyo caso la cadena se truncará hasta la longitud máxima. (Si la cadena que se va a almacenar es más corta que la longitud declarada, los valores de tipo character serán rellenados con espacios; los valores de tipo character varying simplemente almacenarán la cadena más corta.

Si se convierte explícitamente un valor en un carácter *varying*(n) o en un *character*(n), un valor de longitud superior se truncará a n caracteres sin que se produzca un error.

Las notaciones *varchar*(n) y *char*(n) son alias para character *varying*(n) y *character*(n), respectivamente. *character* sin especificador de longitud es equivalente a *character*(1). Si se utiliza *character varying* sin especificador de longitud, el tipo acepta cadenas de cualquier tamaño.

Tipo fecha

Las fechas se cuentan según el calendario gregoriano, incluso en años anteriores a la introducción de dicho calendario

Time, *timestamp* e *interval* aceptan un valor opcional de precisión que especifica el número de dígitos fraccionarios retenidos en el campo de los segundos. Por defecto, no hay un límite explícito en la precisión. El rango permitido de p es de 0 a 6.

El tipo intervalo tiene una opción adicional, que es restringir el conjunto de campos almacenados escribiendo una de estas frases:

```
YEAR
MONTH
DAY
HOUR
MINUTE
SECOND
YEAR TO MONTH
DAY TO HOUR
DAY TO MINUTE
DAY TO SECOND
HOUR TO MINUTE
HOUR TO SECOND
MINUTE TO SECOND
```

Tenga en cuenta que si se especifican tanto campos como p, los campos deben incluir *SECOND*, ya que la precisión se aplica sólo a los segundos.

Times

Los tipos de hora del día son la hora [(p)] sin huso horario y la hora [(p)] con huso horario. la hora sola equivale a la hora sin huso horario.

La entrada válida para estos tipos consiste en una hora del día seguida de un huso horario opcional

Times stamps

La entrada válida para los tipos *Times stamps* consiste en la concatenación de una fecha y una hora, seguida de una zona horaria opcional, seguida de un AD o BC opcional

Input String	Valid Types	Description
epoch	date, timestamp	1970-01-01 00:00:00+00 (Unix system time zero)
infinity	date, timestamp	later than all other time stamps
-infinity	date, timestamp	earlier than all other time stamps
now	date, time, timestamp	current transaction's start time
today	date, timestamp	midnight (00:00) today
tomorrow	date, timestamp	midnight (00:00) tomorrow
yesterday	date, timestamp	midnight (00:00) yesterday
allballs	time	00:00:00.00 UTC