# Universidad Nacional Autónoma de México
# Facultad de Ingeniería


## Base de datos

## Profesor:

## Ing. Fernando Arreola Franco.

## Grupo: 01

## Alumna:

## Mondragón Hernández Andrea Quetzalli

## CARACTERES

- Tipos de caracteres

| Nombre | Descripción |
|---|---|
| character varying(*n*),varchar(*n*) | longitud variable con límite |
| character(*n*),char(*n*) | largo fijo, acolchado en blanco |
| text | longitud ilimitada variable |

Se puede observar que PostgreSQL proporciona tres tipos de caracteres principales donde n un entero positivo. Tanto char(n) y varchar(n) pueden almacenar hasta n caracteres. Si intenta almacenar una cadena que tiene más de n caracteres, PostgreSQL generará un error, sin embargo, una excepción es que, si los caracteres excesivos son todos espacios, PostgreSQL trunca los espacios a la longitud máxima (n) y almacena los caracteres. PostgreSQL truncará la cadena en n caracteres antes de insertarla en la tabla. La única ventaja de especificar el especificador de longitud para el varchar de datos es que PostgreSQL generará un error si intenta insertar una cadena que tenga más de n caracteres en la varchar(n)columna.

El text puede almacenar una cadena con una longitud ilimitada. Si no especifica el entero n para el tipo varchar de datos, se comporta como el tipo text. El rendimiento de varchar (sin el tamaño n) y text son los mismos.

A diferencia varchar, el carácter o char sin el especificador de longitud (n) es lo mismo que character (1) o char (1). A diferencia de otros sistemas de bases de datos, en PostgreSQL no hay diferencia de rendimiento entre los tres tipos de caracteres. En la mayoría de los casos, debe usar text o varchar. y usa varchar(n) cuando quiere que postgresql verifique la longitud.

## TIPOS NUMÉRICOS

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

| Nombre | Tamaño de almacenamiento | Descripción | Rango |
|---|---|---|---|
| smallint | 2 bytes | entero de rango pequeño | -32768 a +32767 |
| integer | 4 bytes | opción típica para entero | -2147483648 al +2147483647 |
| bigint | 8 bytes | entero de rango grande | -9223372036854775808 al +9223372036854775807 |
| decimal | variable | precisión especificada por el usuario, exacta | hasta 131072 dígitos antes del punto decimal; hasta 16383 dígitos después del punto decimal |
| numeric | variable | precisión especificada por el usuario, exacta | hasta 131072 dígitos antes del punto decimal; hasta 16383 dígitos después del punto decimal |
| real | 4 bytes | de precisión variable, inexacta | precisión de 6 dígitos decimales |
| double precision | 8 bytes | de precisión variable, inexacta | precisión de 15 dígitos decimales |
| smallserial | 2 bytes | pequeño entero autoincrementable | 1 a 32767 |
| serial | 4 bytes | autoincrementing integer | 1 al 2147483647 |
| bigserial | 8 bytes | entero autoincrementable grande | 1 al 9223372036854775807 |

Los tipos numéricos tienen un conjunto completo de funciones y operadores aritméticos correspondientes.

- Tipos de enteros
  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 generarán un error.

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

  SQL solo especifica los tipos enteros integer(int) smallint, y bigint.

- Números de precisión arbitraria
  El tipo numeric puede almacenar números con una gran cantidad de dígitos. Está especialmente recomendado para almacenar cantidades monetarias y otras cantidades donde se requiera exactitud. Los cálculos con numeric valores producen resultados exactos cuando es posible, por ejemplo, suma, resta, multiplicación. Sin embargo, los cálculos de numeric valores son muy lentos en comparación con los tipos de números enteros o los tipos de coma flotante.

## TIPO BOOLEANO

El tipo boolean puede tener varios estados: “verdadero”, “falso”, y un tercer estado, “desconocido”, que está representado por el valor nulo de SQL.

| Nombre | Tamaño de almacenamiento | Descripción |
|---|---|---|
| boolean | 1 byte | estado de verdadero o falso |

Las constantes booleanas se pueden representar en consultas SQL mediante las palabras clave de SQL TRUE, FALSE y NULL.

La función de entrada de tipo de datos para el tipo boolean acepta estas representaciones de cadena para el estado " verdadero “:

- true
- yes
- on
- 1

y estas representaciones para el estado “falso”:

- false
- no
- off
- 0

También se aceptan prefijos únicos de estas cadenas, por ejemplo, t o n. Los espacios en blanco iniciales o finales se ignoran y no importan las mayúsculas y minúsculas.
La función de salida de tipo de datos para type boolean siempre emite t o f.

```
---+---------
t | Está
f | no es
```

Las palabras clave TRUE y FALSE son el método preferido (compatible con SQL) para escribir constantes booleanas en consultas SQL.

## TIPOS MONETARIOS

El moneytipo almacena una cantidad de moneda con una precisión fraccionaria fija. La precisión fraccionaria está determinada por la configuración lc_monetary de la base de datos. La entrada se acepta en una variedad de formatos, incluidos los literales enteros y de punto flotante, así como el formato de moneda típico, como '$1,000.00'. La salida es generalmente en la última forma, pero depende de la configuración regional.

| Nombre | Tamaño de almacenamiento | Descripción | Rango |
|---|---|---|---|
| money | 8 bytes | cantidad de moneda | -92233720368547758.08 al +92233720368547758.07 |

Dado que la salida de este tipo de datos es sensible a la configuración regional, es posible que no funcione para cargar money datos en una base de datos que tenga una configuración diferente de lc_monetary. Para evitar problemas, antes de restaurar un volcado en una base de datos nueva se debe asegurar de que lc_monetary tenga el mismo valor o un valor equivalente al de la base de datos que se volcó.
Los valores de los tipos numeric de datos int, y bigint se pueden convertir a money.

## TIPOS FECHAS (Date Type)

Soporta casi cualquier formato de entrada, podemos modificar el modo de entrada para las fechas con el parámetro DataStyle:

| MODO | FORMATO |
|---|---|
| MDY | MES-DÍA-AÑO |
| DMY | DÍA-MES-AÑO |
| YDM | AÑO-DÍA-MES |

Dependiendo del formato y del modo que se configure el parámetro DataStyle, los valores de las fechas pueden variar.

Ejemplo:

| EJEMPLO | FORMATO | VALOR |
|---|---|---|
| 1999-01-08 | ISO 8601 | DÍA 8 DEL MES ENERO DEL AÑO 1999 |
| 1/8/1999 | MDY | DÍA 8 DEL MES ENERO DEL AÑO 1999 |
| 1/8/1999 | DMY | DÍA 1 DEL MES AGOSTO DEL AÑO 1999 |
| 01/02/03 | MDY | DÍA 2 DEL MES ENERO DEL AÑO 2003 |
| 01/02/03 | DMY | DÍA 1 DEL MES FEBRERO DEL AÑO 2003 |
| 01/02/03 | YMD | DÍA 3 DEL MES FEBRERO DEL AÑO 2001 |

**Bibliografías:**


- *PostgreSQL Character Types: CHAR, VARCHAR, And TEXT*. (s. f.). PostgreSQL Tutorial - Learn PostgreSQL from Scratch. https://www.postgresqltutorial.com/postgresql-char-varchar-text/
- *PostgreSQL data types, tipos de datos para fechas - TodoPostgreSQL*. (s. f.). TodoPostgreSQL. https://www.todopostgresql.com/postgresql-data-types-tipos-de-datos-para-fechas/
- *PostgreSQL: Documentation: 10: Data Types*. (s. f.). Postgres Professional. https://postgrespro.com/docs/postgresql/10/datatype