

## TIPOS DE DATOS PARA POSTGRESQL

Salgado Miranda Jorge

Bases de Datos

### TIPOS DE DATOS NUMÉRICOS

## 8.1. Numeric Types

### 8.1.1. Integer Types

### 8.1.2. Arbitrary Precision Numbers

### 8.1.3. Floating-Point Types

### 8.1.4. Serial Types

Numeric types consist of two-, four-, and eight-byte integers, four- and eight-byte floating-point numbers, and selectable-precision decimals. **Table 8.2** lists the available types.

**Table 8.2. Numeric Types**

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

Both the maximum precision and the maximum scale of a `numeric` column can be configured. To declare a column of type `numeric` use the syntax:

```
NUMERIC(precision, scale)
```

The precision must be positive, the scale zero or positive. Alternatively:

```
NUMERIC(precision)
```

selects a scale of 0. Specifying:

```
NUMERIC
```

Infinity

-Infinity

NaN

valores de los números flotantes

## 8.1.4. Serial Types

### Note

This section describes a PostgreSQL-specific way to create an autoincrementing column. Another way is to use the SQL-standard identity column feature, described at [CREATE TABLE](#).

The data types `smallserial`, `serial` and `bigserial` are not true types, but merely a notational convenience for creating unique identifier columns (similar to the `AUTO_INCREMENT` property supported by some other databases). In the current implementation, specifying:

```
CREATE TABLE tablename (  
    colname SERIAL  
);
```

is equivalent to specifying:

```
CREATE SEQUENCE tablename_colname_seq AS integer;  
CREATE TABLE tablename (  
    colname integer NOT NULL DEFAULT nextval('tablename_colname_seq')  
);  
ALTER SEQUENCE tablename_colname_seq OWNED BY tablename.colname;
```

### TIPOS DE DATOS FECHA

Table 8.9. Date/Time Types

Name	Storage Size	Description	Low Value	High Value	Resolution
timestamp [ (p) ] [ without time zone ]	8 bytes	both date and time (no time zone)	4713 BC	294276 AD	1 microsecond
timestamp [ (p) ] with time zone	8 bytes	both date and time, with time zone	4713 BC	294276 AD	1 microsecond
date	4 bytes	date (no time of day)	4713 BC	5874897 AD	1 day
time [ (p) ] [ without time zone ]	8 bytes	time of day (no date)	00:00:00	24:00:00	1 microsecond
time [ (p) ] with time zone	12 bytes	time of day (no date), with time zone	00:00:00+1559	24:00:00-1559	1 microsecond
interval [ fields ] [ (p) ]	16 bytes	time interval	-178000000 years	178000000 years	1 microsecond

time, timestamp, and interval accept an optional precision value *p* which specifies the number of fractional digits retained in the seconds field. By default, there is no explicit bound on precision. The allowed range of *p* is from 0 to 6.

Table 8.10. Date Input

Example	Description
1999-01-08	ISO 8601; January 8 in any mode (recommended format)
January 8, 1999	unambiguous in any datestyle input mode
1/8/1999	January 8 in MDY mode; August 1 in DMY mode
1/18/1999	January 18 in MDY mode; rejected in other modes
01/02/03	January 2, 2003 in MDY mode; February 1, 2003 in DMY mode; February 3, 2001 in YMD mode
1999-Jan-08	January 8 in any mode
Jan-08-1999	January 8 in any mode
08-Jan-1999	January 8 in any mode
99-Jan-08	January 8 in YMD mode, else error
08-Jan-99	January 8, except error in YMD mode
Jan-08-99	January 8, except error in YMD mode
19990108	ISO 8601; January 8, 1999 in any mode
990108	ISO 8601; January 8, 1999 in any mode
1999.008	year and day of year
J2451187	Julian date
January 8, 99 BC	year 99 BC

Table 8.11. Time Input

Example	Description
04:05:06.789	ISO 8601
04:05:06	ISO 8601
04:05	ISO 8601
040506	ISO 8601
04:05 AM	same as 04:05; AM does not affect value
04:05 PM	same as 16:05; input hour must be <= 12
04:05:06.789-8	ISO 8601, with time zone as UTC offset
04:05:06-08:00	ISO 8601, with time zone as UTC offset
04:05-08:00	ISO 8601, with time zone as UTC offset
040506-08	ISO 8601, with time zone as UTC offset
040506+0730	ISO 8601, with fractional-hour time zone as UTC offset
040506+07:30:00	UTC offset specified to seconds (not allowed in ISO 8601)
04:05:06 PST	time zone specified by abbreviation
2003-04-12 04:05:06 America/New_York	time zone specified by full name

Table 8.12. Time Zone Input

Example	Description
PST	Abbreviation (for Pacific Standard Time)
America/New_York	Full time zone name
PST8PDT	POSIX-style time zone specification
-8:00:00	UTC offset for PST
-8:00	UTC offset for PST (ISO 8601 extended format)
-800	UTC offset for PST (ISO 8601 basic format)
-8	UTC offset for PST (ISO 8601 basic format)
zulu	Military abbreviation for UTC
Z	Short form of zulu (also in ISO 8601)

8.5.1.3. Time Stamps

Valid input for the time stamp types consists of the concatenation of a date and a time, followed by an optional time zone, followed by an optional AD or BC. (Alternatively, AD/BC can appear before the time zone, but this is not the preferred ordering.) Thus:

```
1999-01-08 04:05:06
```

and:

```
1999-01-08 04:05:06 -8:00
```

are valid values, which follow the ISO 8601 standard. In addition, the common format:

```
January 8 04:05:06 1999 PST
```

is supported.

The SQL standard differentiates `timestamp without time zone` and `timestamp with time zone` literals by the presence of a "+" or "-" symbol and time zone offset after the time. Hence, according to the standard,

```
TIMESTAMP '2004-10-19 10:23:54'
```

8.5.1.4. Special Values

PostgreSQL supports several special date/time input values for convenience, as shown in **Table 8.13**. The values `infinity` and `-infinity` are specially represented inside the system and will be displayed unchanged; but the others are simply notational shorthands that will be converted to ordinary date/time values when read. (In particular, `now` and related strings are converted to a specific time value as soon as they are read.) All of these values need to be enclosed in single quotes when used as constants in SQL commands.

Table 8.13. Special Date/Time Inputs

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

The following SQL-compatible functions can also be used to obtain the current time value for the corresponding data type: `CURRENT_DATE`, `CURRENT_TIME`, `CURRENT_TIMESTAMP`, `LOCALTIME`, `LOCALTIMESTAMP`. (See **Section 9.9.5**.) Note that these are SQL functions and are *not* recognized in data input strings.

Table 8.14. Date/Time Output Styles

Style Specification	Description	Example
ISO	ISO 8601, SQL standard	1997-12-17 07:37:16-08
SQL	traditional style	12/17/1997 07:37:16.00 PST
Postgres	original style	Wed Dec 17 07:37:16 1997 PST
German	regional style	17.12.1997 07:37:16.00 PST

Table 8.15. Date Order Conventions

datestyle Setting	Input Ordering	Example Output
SQL, DMY	<i>day/month/year</i>	17/12/1997 15:37:16.00 CET
SQL, MDY	<i>month/day/year</i>	12/17/1997 07:37:16.00 PST
Postgres, DMY	<i>day/month/year</i>	Wed 17 Dec 07:37:16 1997 PST

## Table 8.16. ISO 8601 Interval Unit Abbreviations

Abbreviation	Meaning
Y	Years
M	Months (in the date part)
W	Weeks
D	Days
H	Hours
M	Minutes (in the time part)
S	Seconds

Table 8.17. Interval Input

Example	Description
1-2	SQL standard format: 1 year 2 months
3 4:05:06	SQL standard format: 3 days 4 hours 5 minutes 6 seconds
1 year 2 months 3 days 4 hours 5 minutes 6 seconds	Traditional Postgres format: 1 year 2 months 3 days 4 hours 5 minutes 6 seconds
P1Y2M3DT4H5M6S	ISO 8601 "format with designators": same meaning as above
P0001-02-03T04:05:06	ISO 8601 "alternative format": same meaning as above

### TIPOS DE DATOS CADENA



Table 8.4. Character Types

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

Table 8.5. Special Character Types

Name	Storage Size	Description
"char"	1 byte	single-byte internal type
name	64 bytes	internal type for object names

#### TIPOS DE DATOS BINARIOS

Table 8.6. Binary Data Types

Name	Storage Size	Description
bytea	1 or 4 bytes plus the actual binary string	variable-length binary string

Table 8.7. bytea Literal Escaped Octets

Decimal Octet Value	Description	Escaped Input Representation	Example	Hex Representation
0	zero octet	'\000'	'\000'::bytea	\x00
39	single quote	'''' or '\047'	''''::bytea	\x27
92	backslash	'\\' or '\134'	'\\'::bytea	\x5c
0 to 31 and 127 to 255	"non-printable" octets	'\xxx' (octal value)	'\001'::bytea	\x01

Table 8.8. `bytea` Output Escaped Octets

Decimal Octet Value	Description	Escaped Output Representation	Example	Output Result
92	backslash	\\	'\134'::bytea	\\
0 to 31 and 127 to 255	“non-printable” octets	\xxx (octal value)	'\001'::bytea	\001
32 to 126	“printable” octets	client character set representation	'\176'::bytea	~