PostgreSQL Data Types

Numeric Data Types:

There are mainly Nine numeric data types that PostgreSQL supports. These are listed below:

Data Type	Range	Storage Size	Description
INTEGER	-2147483648 to 2147483647 (signed) 0 to 4294967295 (unsigned)	4 Bytes	Normal sized integer value.
SMALLINT	-32768 to 32767 (signed) 0 to 65535 (unsigned)	2 Bytes	Small integer value.
BIGINT	-9223372036854775808 to 9223372036854775807 (signed) 0 to 18446744073709551615 (unsigned)	8 Bytes	Large integer value.
DOUBLE PRECISION	15 Decimal digits precision.	8 Bytes	Variable precision floating point number.
DECIMAL	131072 before decimal; 16383 after decimal	variable	User Specified precision fixed point number.
NUMERIC	131072 before decimal; 16383 after decimal	variable	User Specified precision fixed point number.
REAL	6 Decimal digits precision.	4 Bytes	Variable precision floating point number.
SERIAL	1 to 2147483647	4 Bytes	Auto Incrementing Integer.
BIGSERIAL	1 to 9223372036854775807	8 Bytes	Large Auto Incrementing Integer

Date and Time Data Types:

There are mainly five date and time data types that PostgreSQL supports. These are listed below:

Data Type	Size	Range	Resolution	Description
TIMESTAMP [(p)] [without time zone]	8 Bytes	4713 BC to 294276 AD	1 microsecond / 14 digits	Both date and time
TIMESTAMP [(p)] [with time zone]	8 Bytes	4713 BC to 294276 AD	1 microsecond / 14 digits	Both date and time
DATE	4 Bytes	4713 BC to 5874897 AD	1 day	Date (no time of day)
TIME [(p)] [without time zone]	8 Bytes	00:00:00 to 24:00:00	1 microsecond / 14 digits	Time of day (no date)
TIME [(p)] [with time zone]	12 Bytes	00:00:00+1459 to 24:00:00-1459	1 microsecond / 14 digits	Time of day (no date)

INTERVAL [fields] [(p)]	12 Bytes	-178000000 years to 178000000 years	1 microsecond / 14 digits	Time interval

String Data Types:

There are mainly five string data types that PostgreSQL supports. These are listed below:

Data Type	Maximum Size	Description
CHAR(size)	255 characters	Here, size is equal to the number of characters to store. It is used for fixed-length strings with space padded on right to equal size characters.
VARCHAR(size)	255 characters	Here, size is equal to the number of characters to store. It is used for variable-length strings.
TEXT(size)	65,535 characters	Here, size is equal to the number of characters to store.
CHARACTER(size)	255 characters	Here, size is equal to the number of characters to store. It is used for fixed-length strings with space padded on right to equal size characters.
CHARACTER VARYING(size)	255 characters	Here, size is equal to the number of characters to store. It is used for variable-length strings.

Boolean Data Types:

There are mainly one boolean data types that PostgreSQL supports.

Data Type	Size	Description
BOOLEAN	1 Byte	Specifies the state of True or False.

Monetary Data Types:

There are mainly one monetary data types that PostgreSQL supports.

Data Type	Size	Range	Description
MONEY	8 Bytes	-92233720368547758.08 to +92233720368547758.07	Specifies the Currency Amount.

Geometric Data Types:

There are mainly eight geometric data types that PostgreSQL supports. These are listed below:

Data Type	Size	Description
POINT	16 Bytes	Point on a plane.
LINE	32 Bytes	Infinite line.
LSEG	32 Bytes	Finite line segment.

BOX	32 Bytes	Rectangular box.
PATH	16+16n Bytes	Closed path.
PATH	16+16n Bytes	Open path
POLYGON	40+16n Bytes	Polygon.
CIRCLE	24 Bytes	Circle.

JSON:

PostgreSQL supports 2 types of JSON types namely JSON and JSONB(Binary JSON). The JSON data type is used to store plain JSON data that get parsed every time it’s called by a query. Whereas the JSONB data type is used to store JSON data in a binary format. It is one hand makes querying data faster whereas slows down the data insertion process as it supports indexing of table data.

UUID:

The UUID data type allows you to store Universal Unique Identifiers defined by RFC 4122. The UUID values guarantee a better uniqueness than SERIAL and can be used to hide sensitive data exposed to the public such as values of id in URL.

The UUID stands for Unique Universal Identifiers. These are used to give a unique ID to a data that is unique throughout the database. The UUID data type are used to store UUID of the data defined by RFC 4122. These are generally used to protect sensitive data like credit card informations and is better compared to SERIAL data type in the context of uniqueness.