|  |
| --- |
| **Лабораторная работа № 3**  Работа с датами |

PostgreSQL

**Типы для работы с датами и временем**

* **timestamp**: хранит дату и время. Занимает 8 байт. Для дат самое нижнее значение - 4713 г до н.э., самое верхнее значение - 294276 г н.э.
* **timestamp with time zone**: то же самое, что и timestamp, только добавляет данные о часовом поясе.
* **date**: представляет дату от 4713 г. до н.э. до 5874897 г н.э. Занимает 4 байта.
* **time**: хранит время с точностью до 1 микросекунды без указания часового пояса. Принимает значения от 00:00:00 до 24:00:00. Занимает 8 байт.
* **time with time zone**: хранит время с точностью до 1 микросекунды с указанием часового пояса. Принимает значения от 00:00:00+1459 до 24:00:00-1459. Занимает 12 байт.
* **interval**: представляет временной интервал. Занимает 16 байт.

Распространенные форматы дат:

* yyyy-mm-dd - 2023-01-08
* Month dd, yyyy - January 8, 2023
* mm/dd/yyyy - 1/8/2023

Распространенные форматы времени:

* hh:mi - 13:21
* hh:mi am/pm - 1:21 pm
* hh:mi:ss - 1:21:34

Чтобы посмотреть все типы данных, которые поддерживает PostgreSql, нам потребуется команда:

select typname, typlen, typtype from pg\_type;

**LOCALTIMESTAMP()**

Функция LOCALTIMESTAMP() возвращает текущие дату и время без локального часового пояса. Функция может иметь необязательный аргумент, называемый точностью. Базовый синтаксис:

SELECT LOCALTIMESTAMP;

Первая часть результата представляет дату, за которой следует время в 24-часовом формате. Поскольку точность не указана, возвращается 6 десятичных знаков.  
  
  
  
В этом примере число 2 в аргументе задает точность, следовательно на выходе будет 2 десятичных знака.

SELECT LOCALTIMESTAMP(2);

В мире программирования функция LOCALTIMESTAMP часто используется для вставки значений по умолчанию в столбцы, которые отслеживают текущее время вставки записей, например, в столбцах аудита.  
  
Давайте рассмотрим следующий пример, чтобы понять сценарий. В этой таблице registration\_datetime является столбцом аудита.

CREATE TABLE registration(

student\_id serial PRIMARY KEY,

student\_name varchar(255) NOT NULL,

registration\_datetime TIMESTAMP DEFAULT LOCALTIMESTAMP

);

Теперь вставим запись в эту таблицу

INSERT INTO registration(student\_name)

VALUES('James Watt');

select \* from registration;

Видим, что наряду с предоставленными данными пользователя, столбец аудита так же заполнился текущим значением временной метки.

**LOCALTIME()**

Функция LOCALTIME() возвращает текущее время без локального часового пояса. Функция принимает необязательный аргумент, называемый точностью. Базовый синтаксис:

SELECT LOCALTIME;

Результат представляет собой время в 24-часовом формате. Поскольку точность не указывалась, выводится 6 десятичных знаков.  
  
  
  
В этом примере в качестве аргумента точности передается значение 2, поэтому выводится 2 десятичных знака.

SELECT LOCALTIME(2);

Подобно функции LOCALTIMESTAMP, функция LOCALTIME также часто используется для вставки значений по умолчанию в столбцы, которые отслеживают текущее время вставки записей - столбцах аудита.  
  
В следующем примере столбцы таблицы registration\_date и registration\_time являются столбцами аудита (DEFAULT – по умолчанию).

CREATE TABLE registration(

student\_id serial PRIMARY KEY,

student\_name varchar(255) NOT NULL,

registration\_date DATE DEFAULT CURRENT\_DATE,

registration\_time TIME DEFAULT LOCALTIME

);

Давайте вставим теперь запись в таблицу:

INSERT INTO registration(student\_name)

VALUES('Shivayan Mukherjee');

select \* from registration;

Видим, что столбцы аудита заполняются текущими датой и временем, наряду с предоставленными данными о пользователе.

**TO\_DATE**

Функция TO\_DATE() помогает преобразовать строку в дату и, возможно, является одной из наиболее важных и широко используемых функций даты/времени PostgreSQL в мире программирования. Функция TO\_DATE() конвертирует строковый литерал в значение даты. Базовый синтаксис:  
  
TO\_DATE(текст, формат\_даты);  
  
Функция TO\_DATE() принимает два параметра. Первый параметр - это строка, которую вы хотите конвертировать в дату. Второй параметр - формат даты. Функция TO\_DATE() возвращает значение даты.  
  
**Пример 1:**  
  
В этом примере входная строка преобразуется в дату в соответствии с входным форматом 'YYYYMMDD', где YYYY - год четырьмя цифрами, MM - месяц двумя цифрами и DD - день двумя цифрами.

SELECT TO\_DATE('20210910','YYYYMMDD');

**Пример 2:**  
  
Важно заметить, что входной текст должен соответствовать указанному формату даты. Будет работать следующий запрос:

SELECT TO\_DATE('10 Sep 2021', 'DD Mon YYYY');

Этот запрос вызовет ошибку, т.к. месяц и год в строке не соответствуют форматированию.

SELECT TO\_DATE('10 Sep 2021', 'DD YYYY Mon');

**TO\_TIMESTAMP()**

Функция TO\_TIMESTAMP() помогает конвертировать строку в временную метку. функция TO\_TIMESTAMP() преобразует строковый литерал в метку времени на основе указанного формата. Базовый синтаксис:  
  
TO\_TIMESTAMP(текст, формат\_метки\_времени)  
  
Функция TO\_TIMESTAMP() принимает два параметра - строку, которую требуется преобразовать к метке времени, и соответствующий формат. Функция TO\_TIMESTAMP() возвращает временную метку с часовым поясом.  
  
**Пример 1:**  
  
В этом примере входная строка преобразуется к дате в соответствии с входным форматом 'YYYY-MM-DD HH:MI:SS', где YYYY - год, заданный четырьмя цифрами, MM - месяц двумя цифрами и DD - день двумя цифрами с последующим форматом часов, минут и секунд.

SELECT TO\_TIMESTAMP(

'2021-09-10 11:30:20',

'YYYY-MM-DD HH:MI:SS'

);

**Пример 2: время в 24-часовом формате**  
  
Выражение времени в первом примере поддерживает 12-часовой формат времени. Для поддержки 24-часового формата времени должно использоваться выражение 'HH24', как показано в этом примере.

SELECT TO\_TIMESTAMP(

'2021-09-10 21:30:20',

'YYYY-MM-DD HH24:MI:SS'

);

Результат:

Категории:[PostgreSQL](https://sql-ex.ru/blogs/?/categories/PostgreSQL)

**Текущая дата/время**

Postgres Pro предоставляет набор функций, результат которых зависит от текущей даты и времени. Все следующие функции соответствуют стандарту SQL и возвращают значения, отражающие время начала текущей транзакции:

CURRENT\_DATE

CURRENT\_TIME

CURRENT\_TIMESTAMP

CURRENT\_TIME(***точность***)

CURRENT\_TIMESTAMP(***точность***)

LOCALTIME

LOCALTIMESTAMP

LOCALTIME(***точность***)

LOCALTIMESTAMP(***точность***)

CURRENT\_TIME и CURRENT\_TIMESTAMP возвращают время с часовым поясом. В результатах LOCALTIME и LOCALTIMESTAMP нет информации о часовом поясе.

CURRENT\_TIME, CURRENT\_TIMESTAMP, LOCALTIME и LOCALTIMESTAMP могут принимать необязательный параметр точности, определяющий, до какого знака после запятой следует округлять поле секунд. Если этот параметр отсутствует, результат будет иметь максимально возможную точность.

Несколько примеров:

***Задание 1:***

SELECT NOW();

SELECT CURRENT\_TIME;

SELECT CURRENT\_DATE;

SELECT CURRENT\_TIMESTAMP;

SELECT CURRENT\_TIMESTAMP(2);

SELECT LOCALTIMESTAMP;

Так как эти функции возвращают время начала текущей транзакции, во время транзакции эти значения не меняются. Это считается не ошибкой, а особенностью реализации: цель такого поведения в том, чтобы в одной транзакции «текущее» время было одинаковым и для разных изменений в одной транзакций записывалась одна отметка времени.

SELECT CURRENT\_TIMESTAMP;

SELECT now();

SELECT TIMESTAMP 'now';

***Задание 2:***

У нас есть два типа данных, которые мы можем использовать:  
  
timestamp

* timestamp с часовым поясом (или timestamptz)

Тип *timestamp* содержит только дату и время, никакой другой информации.

select typname, typlen from pg\_type where typname ~ '^timestamp';

Каждый раз, когда речь идет о значениях timestamptz, если часовой пояс не указан, то PostgreSQL использует заранее сконфигурированное время. И вы можете конфигурировать его разными способами:  
  
**select** now();

**set** timezone = 'America/New\_York';

**select** now();

*SET TIMEZONE='Europe/Samara';*

**show** timezone;

**Временные интервалы**

При вычитании одной даты из другой результатом является временной интервал. Временной интервал существует в Postgres как самостоятельный тип данных.

Даты складывать нельзя, но к дате прибавить временной интервал можно. Например, к 31.01.2023 12:00 добавить 3 часа - вполне законная операция с понятным результатом 31.01.2023 15:00.

Выполним эту операцию на практике:

***Задание 3:***

SELECT timestamp '2023-01-31 12:00' + interval '3 hours' AS result

Вычитать временной интервал тоже можно:

SELECT timestamp '2023-01-31 12:00' - interval '3 hours' AS result

**Ввод интервала**

Временной интервал можно ввести в форме:

[@] количество единица [количество единица...] [направление]

где единица может принимать значения:

* microsecond - микросекунды;
* millisecond - миллисекунды;
* second - секунды;
* minute - минуты;
* hour - часы;
* day - дни;
* week - недели;
* month - месяцы;
* year - годы;
* decade - десятилетия;
* century - века;
* millennium - тысячелетия.

SELECT interval '1 day 12 hours 59 min 10 sec'

Направление может принимать значение ago (назад) или быть пустым.

SELECT interval '1 day 12 hours 59 min 10 sec ago'

Знак @ является необязательным.

Количество может быть отрицательным числом.

SELECT interval '12 hours -10 minutes'

**Извлечь EXTRACT, date\_part**

EXTRACT(***field*** FROM ***source***)

Функция extract получает из значений даты/времени поля, такие как год или час. Здесь ***источник*** — значение типа timestamp, time или interval. (Выражения типа date приводятся к типу timestamp, так что допускается и этот тип.) Указанное ***поле*** представляет собой идентификатор, по которому из источника выбирается заданное поле. Функция extract возвращает значения типа double precision. Допустимые поля: century

***Задание 4:***

Век:

SELECT EXTRACT(CENTURY FROM TIMESTAMP '2023-12-16 12:21:13');

Первый век начался 0001-01-01 00:00:00, хотя люди в то время и не считали так. Это определение распространяется на все страны с григорианским календарём. Века с номером 0 не было; считается, что 1 наступил после -1. Если такое положение вещей вас не устраивает, направляйте жалобы по адресу: Ватикан, Собор Святого Петра, Папе.

day

Для значений timestamp это день месяца (1 - 31), для значений interval — число дней

SELECT EXTRACT(DAY FROM TIMESTAMP '2023-02-16 20:38:40');

SELECT EXTRACT(DAY FROM INTERVAL '40 days 1 minute');

Год, делённый на 10

SELECT EXTRACT(DECADE FROM TIMESTAMP '2023-02-16 20:38:40');

День недели, считая с воскресенья (0) до субботы (6)

SELECT EXTRACT(DOW FROM TIMESTAMP '2023-02-16 20:38:40');

Заметьте, что в extract дни недели нумеруются не так, как в функции to\_char(..., 'D').

doy

День года (1 - 365/366)

SELECT EXTRACT(DOY FROM TIMESTAMP '2023-02-16 20:38:40');

Для значений timestamp with time zone это число секунд с 1970-01-01 00:00:00 UTC (отрицательное для предшествующего времени); для значений date и timestamp — номинальное число секунд с 1970-01-01 00:00:00 без учёта часового пояса, переходов на летнее время и т. п.; для значений interval — общее количество секунд в интервале

SELECT EXTRACT(EPOCH FROM TIMESTAMP WITH TIME ZONE '2023-02-16 20:38:40.12-08');

SELECT EXTRACT(EPOCH FROM TIMESTAMP '2023-02-16 20:38:40.12');

SELECT EXTRACT(EPOCH FROM INTERVAL '5 days 3 hours');

Преобразовать время эпохи назад, в значение timestamp with time zone, с помощью to\_timestamp можно так:

SELECT to\_timestamp(982384720.12);

Имейте в виду, что применяя to\_timestamp к времени эпохи, извлечённому из значения date или timestamp, можно получить не вполне ожидаемый результат: эта функция подразумевает, что изначальное значение задано в часовом поясе UTC, но это может быть не так.

Час (0 - 23)

SELECT EXTRACT(HOUR FROM TIMESTAMP '2023-02-16 20:38:40');

День недели, считая с понедельника (1) до воскресенья (7)

SELECT EXTRACT(ISODOW FROM TIMESTAMP '2023-02-18 20:38:40');

Результат отличается от dow только для воскресенья. Такая нумерация соответствует ISO 8601.

Год по недельному календарю ISO 8601, в который попадает дата (неприменимо к интервалам)

SELECT EXTRACT(ISOYEAR FROM DATE '2023-01-01');

SELECT EXTRACT(ISOYEAR FROM DATE '2023-01-02');

Год по недельному календарю ISO начинается с понедельника недели, в которой оказывается 4 января, так что в начале января или в конце декабря год по ISO может отличаться от года по григорианскому календарю.

*Юлианская дата*, соответствующая дате или дате/времени (для интервала не определена). Значение будет дробным, если заданное время отличается от начала суток по местному времени.

SELECT EXTRACT(JULIAN FROM DATE '2023-01-01');

SELECT EXTRACT(JULIAN FROM TIMESTAMP '2023-01-01 12:00');

Значение секунд с дробной частью, умноженное на 1 000 000; заметьте, что оно включает и целые секунды

SELECT EXTRACT(MICROSECONDS FROM TIME '17:12:28.5');

Тысячелетие

SELECT EXTRACT(MILLENNIUM FROM TIMESTAMP '2023-02-16 20:38:40');

Годы 20 века относятся ко второму тысячелетию. Третье тысячелетие началось 1 января 2001 г.

milliseconds

Значение секунд с дробной частью, умноженное на 1 000; заметьте, что оно включает и целые секунды.

SELECT EXTRACT(MILLISECONDS FROM TIME '17:12:28.5');

minute

Минуты (0 - 59)

SELECT EXTRACT(MINUTE FROM TIMESTAMP '2023-02-16 20:38:40');

month

Для значений timestamp это номер месяца в году (1 - 12), а для interval — остаток от деления числа месяцев на 12 (в интервале 0 - 11)

SELECT EXTRACT(MONTH FROM TIMESTAMP '2023-02-16 20:38:40');

SELECT EXTRACT(MONTH FROM INTERVAL '2 years 3 months');

SELECT EXTRACT(MONTH FROM INTERVAL '2 years 13 months');

quarter

Квартал года (1 - 4), к которому относится дата

SELECT EXTRACT(QUARTER FROM TIMESTAMP '2023-02-16 20:38:40');

Секунды, включая дробную часть (0 - 59[[7]](https://postgrespro.ru/docs/postgrespro/10/functions-datetime#ftn.id-1.5.8.14.13.5.11.17.2.1.1))

SELECT EXTRACT(SECOND FROM TIMESTAMP '2023-02-16 20:38:40');

SELECT EXTRACT(SECOND FROM TIME '17:12:28.5');

***Задание 5:***

**Составные типы**

CREATE TYPE microphone AS (

waterproof boolean,

speaker boolean,

earpiece boolean

);

CREATE TABLE radio (

radioname VARCHAR(50),

mic microphone

);

INSERT INTO radio (mic,radioname) VALUES ('(1,1,0)','Yaesu 8900');

SELECT \* FROM radio;

## **Тип JSON**

CREATE TABLE employee (

id integer NOT NULL,

age integer NOT NULL,

data jsonb

);

INSERT INTO employee VALUES (1, 35, '{"name": "Tom Price", "tags": ["Motivated", "SelfLearner"], "onboareded": true}');

select \* from employee;