|  |
| --- |
| **Лабораторная работа № 3**  Работа с датами и TimeStamp как с интервальными типами. |
| Типы данных TIMESTAMP и INTERVAL в СУБД Oracle  Сегодня в СУБД Oracle есть несколько типов данных для хранения дат и времени. Самый старый из них - тип date - совершенно точно был еще в Oracle 7 (с более ранними версиями СУБД я не работал). Некоторые [интересные вещи про тип date](http://torofimofu.blogspot.ru/2017/04/xiii.html) я уже рассказывал. В версии Oracle 9 появились новые типы для дат и времени timestamp, timestamp with local time zone и timestamp with time zone, а также интервальные типы interval year to month и interval day to second, работающие вместе с новыми типами и типом date.  Типы timestamp, timestamp with local time zone и timestamp with time zone привнесли два новшества, по сравнению с типом date:   * возможность работать со временем с точностью до наносекунд, * возможность работать с часовыми поясами (time zones).   Ниже мы поработаем с типами timestamp и interval, обращая внимание на задание значений этих типов с помощью литералов и на их арифметику. Затем обратимся к различиям между типами timestamp, timestamp with local time zone и timestamp with time zone, и рассмотрим, какой из них предпочтительно использовать в какой ситуации.   |  | | --- | | Настройка SQL. Необходимо включить режим ECHO и вывести протокол лаб. работы в файл *<Фамилия студента>.txt*. Этот файл является отчетом о проделанной лаб. работе.  SQL> set echo on  SQL> spool c:\spool. txt  ……………………………………..  Завершить протокол лаб. работы (команда spool off) и направить результаты преподавателю.  SQL> spool off |   Для начала, получим текущую системную метку времени (timestamp) с помощью функции systimestamp:  SQL> select systimestamp from dual;  SYSTIMESTAMP  --------------------------------------------------------------------------------  18-JUN-17 02.28.40.558345 PM +10:00  Видим 6 цифр для долей секунды, что меньше максимальной точности, на которую способен тип timestamp. Формат отображения типа timestamp нуждается в настройке:  SQL> show parameter timestamp  NAME TYPE VALUE  ------------------------------------ ----------- ------------------------------  nls\_timestamp\_format string DD-MON-RR HH.MI.SSXFF AM  nls\_timestamp\_tz\_format string DD-MON-RR HH.MI.SSXFF AM TZR  SQL> -- добавим в формат наносекунды  SQL> alter session set nls\_timestamp\_format = 'YYYY-MM-DD HH24:MI:SS.FF9';  Session altered  SQL> alter session set nls\_timestamp\_tz\_format = 'YYYY-MM-DD HH24:MI:SS.FF9 TZR';  Session altered  SQL> select systimestamp from dual;  SYSTIMESTAMP  --------------------------------------------------------------------------------  2017-06-18 14:35:23.468638000 +10:00  Теперь разряды единиц, десятков и сотен наносекунд отображаются, но в них нули - оттого, что операционная система, под которой работает СУБД Oracle, не поддерживает наносекундную точность.  Зададим наносекунды с помощью литерала:  SQL> select timestamp '2017-06-18 14:00:00.123456789 +10:00' from dual;  TIMESTAMP'2017-06-1814:00:00.1  --------------------------------------------------------------------------------  18-JUN-17 02.00.00.123456789 PM +10:00  Все три типа меток времени в SQL и PL/SQL по умолчанию хранят время с точностью до микросекунд. Для того, чтобы работать с наносекундами, нужно при создании столбца таблицы или переменной PL/SQL соответствующего типа явно указать точность 9 знаков.  Продемонстрирую это для SQL:  SQL> create table tstable (  2 ts timestamp,  3 tstz timestamp with time zone,  4 tsltz timestamp with local time zone,  5 ts9 timestamp(9)  6 );  Table created  SQL> desc tstable  Name Null? Type  -------------------- -------- ----------------------------------  TS TIMESTAMP(6)  TSTZ TIMESTAMP(6) WITH TIME ZONE  TSLTZ TIMESTAMP(6) WITH LOCAL TIME ZONE  TS9 TIMESTAMP(9)  SQL> insert into tstable (ts, ts9)  2 values (  3 timestamp '2017-06-18 14:00:00.123456789',  4 timestamp '2017-06-18 14:00:00.123456789'  5 );  1 row inserted  SQL> select ts, ts9 from tstable;  TS TS9  ---------------------------------- ----------------------------------  18-JUN-17 02.00.00.123457 PM 18-JUN-17 02.00.00.123456789 PM  SQL> rollback;  Как видим, при сохранении значения в столбце ts наносекунды потерялись, а в столбце ts9 сохранились. (Другие столбцы таблицы tstable понадобятся нам в дальнейшем.)  Аналогично, для PL/SQL:  SQL> set serveroutput on  SQL> declare  2 l\_ts timestamp;  3 l\_ts9 timestamp(9);  4 begin  5 l\_ts := timestamp '2017-06-18 14:00:00.123456789';  6 l\_ts9 := timestamp '2017-06-18 14:00:00.123456789';  7 dbms\_output.put\_line(l\_ts);  8 dbms\_output.put\_line(l\_ts9);  9 end;  10 /    18-JUN-17 02.00.00.123457 PM  18-JUN-17 02.00.00.123456789 PM  PL/SQL procedure successfully completed  Теперь сравним операции вычитания дат и вычитания меток времени. Результат вычитания дат - число, выражающее разницу между датами как количество суток, а результат вычитания меток времени - интервал.  SQL> select  2 to\_date('2017-06-18 14:00:00', 'yyyy-mm-dd hh24:mi:ss') -  3 to\_date('2017-05-18 14:00:00', 'yyyy-mm-dd hh24:mi:ss')  4 from dual;  TO\_DATE('2017-06-1814:00:00','  ------------------------------  31  SQL> select timestamp '2017-06-18 14:00:00' - timestamp '2017-05-18 14:00:00' from dual;  TIMESTAMP'2017-06-1814:00:00'-  -------------------------------------------------------------------------------  +000000031 00:00:00.000000000  Последний запрос вернул нам значение типа interval day to second.  СУБД Oracle предлагает два интервальных типа:   * interval year to month - для задания интервалов длительностью годы (2 разряда по умолчанию) и месяцы, * interval day to second - для задания интервалов длительностью от дней (2 разряда по умолчанию) до секунд (2 разряда по умолчанию) и долей секунд (6 разрядов по умолчанию).   Вот литералы, задающие интервалы:  SQL> select interval '10' year,  2 interval '100' year(3),  3 interval '1001' year(4)  4 from dual;  INTERVAL'10'YEAR INTERVAL'100'YEAR(3) INTERVAL'1001'YEAR(4)  ------------------------------ ------------------------------ ------------------------------  +10-00 +100-00 +1001-00  SQL> select interval '2' month,  2 interval '24' month,  3 interval '100' month(3)  4 from dual;  INTERVAL'2'MONTH INTERVAL'24'MONTH INTERVAL'100'MONTH(3)  ------------------------------ ------------------------------ ------------------------------  +00-02 +02-00 +008-04  SQL> select interval '15' day,  2 interval '50' day,  3 interval '100' day(3)  4 from dual;  INTERVAL'15'DAY INTERVAL'50'DAY INTERVAL'100'DAY(3)  ------------------------------ ------------------------------ ------------------------------  +15 00:00:00 +50 00:00:00 +100 00:00:00  SQL> select interval '21' hour,  2 interval '50' hour,  3 interval '100' hour(3)  4 from dual;  INTERVAL'21'HOUR INTERVAL'50'HOUR INTERVAL'100'HOUR(3)  ------------------------------ ------------------------------ ------------------------------  +00 21:00:00 +02 02:00:00 +004 04:00:00  SQL> select interval '5' minute,  2 interval '90' minute,  3 interval '100' minute(3)  4 from dual;  INTERVAL'5'MINUTE INTERVAL'90'MINUTE INTERVAL'100'MINUTE(3)  ------------------------------ ------------------------------ ------------------------------  +00 00:05:00 +00 01:30:00 +000 01:40:00  SQL> select interval '10' second,  2 interval '100' second(3),  3 interval '10.000000001' second,  4 interval '10.000000001' second(2,9)  5 from dual;  INTERVAL'10'SECOND INTERVAL'100'SECOND(3) INTERVAL'10.000000001'SECOND INTERVAL'10.000000001'SECOND(2  ------------------------------ ------------------------------ ------------------------------ ------------------------------  +00 00:00:10.000000 +000 00:01:40.000000 +00 00:00:10.000000 +00 00:00:10.000000001  Для обоих типов interval year to month и interval day to second можно задавать составные литералы, комбинирующие годы и месяцы, дни, часы, минуты и секунды:  SQL> select interval '10-7' year to month  2 from dual;  INTERVAL'10-7'YEARTOMONTH  -------------------------------------------------------------------------------  +10-07  SQL> select interval '13:20' hour to minute,  2 interval '13:20:22' hour to second,  3 interval '13:20:22.123456' hour to second  4 from dual;  INTERVAL'13:20'HOURTOMINUTE INTERVAL'13:20:22'HOURTOSECOND INTERVAL'13:20:22.123456'HOURT  ------------------------------ ------------------------------ ------------------------------  +00 13:20:00 +00 13:20:22.000000 +00 13:20:22.123456  Вы заметили, что значения "нормализуются" при преобразовании литерала во внутреннее представление? Например, минуты от 60-ти и выше пересчитываются в часы, часы от 24-х и выше - в дни:  SQL> select interval '90' minute from dual;  INTERVAL'90'MINUTE  -------------------------------------------------------------------------------  +00 01:30:00  SQL> select interval '25:00:59' hour to second from dual;  INTERVAL'25:00:59'HOURTOSECOND  -------------------------------------------------------------------------------  +01 01:00:59.000000  Интервалы одного типа складываются и вычитаются; бывают отрицательные интервалы:  SQL> select  2 interval '5 10' day to hour - interval '2 7' day to hour,  3 interval '2 7' day to hour - interval '5 10' day to hour  4 from dual;  INTERVAL'510'DAYTOHOUR-INTERVA INTERVAL'27'DAYTOHOUR-INTERVAL  ---------------------------------- ----------------------------------  +000000003 03:00:00.000000000 -000000003 03:00:00.000000000  SQL> select interval '-5' day + interval '-5' day from dual;  INTERVAL'-5'DAY+INTERVAL'-5'DA  -------------------------------------------------------------------------------  -000000010 00:00:00.000000000  Но агрегатная функция sum не умеет суммировать интервалы:  SQL> select sum(interval '1' second) from dual connect by level <= 10;  ORA-00932: inconsistent datatypes: expected NUMBER got INTERVAL DAY TO SECOND  Зато агрегатные функции min и max с интервалами работают:  SQL> with i as (select to\_dsinterval('0 00:00:'||to\_char(level,'00')) i from dual connect by level <= 10)  2 select min(i), max(i) from i;  MIN(I) MAX(I)  ---------------------------------- ----------------------------------  +000000000 00:00:01.000000000 +000000000 00:00:10.000000000  (Правда, напоминает знаменитое ["Здесь играем, здесь не играем, здесь рыбу заворачивали"](https://www.google.ru/search?q=%D0%97%D0%B4%D0%B5%D1%81%D1%8C%20%D0%B8%D0%B3%D1%80%D0%B0%D0%B5%D0%BC,%20%D0%B7%D0%B4%D0%B5%D1%81%D1%8C%20%D0%BD%D0%B5%20%D0%B8%D0%B3%D1%80%D0%B0%D0%B5%D0%BC,%20%D0%B7%D0%B4%D0%B5%D1%81%D1%8C%20%D1%80%D1%8B%D0%B1%D1%83%20%D0%B7%D0%B0%D0%B2%D0%BE%D1%80%D0%B0%D1%87%D0%B8%D0%B2%D0%B0%D0%BB%D0%B8)?)  Нельзя складывать и вычитать разные интервальные типы:  SQL> select  2 interval '1-0' year to month + interval '1 00:00:00' day to second  3 from dual;  ORA-30081: invalid data type for datetime/interval arithmetic  Здесь мы попробовали сложить годы с днями, но в Oracle нет типа данных, который мог бы представить интервал 1 год и 1 день. Отсюда ошибка. Зато можно прибавить интервал 1 год и интервал 1 день, например, к текущей дате:  SQL> select  2 sysdate + interval '1-0' year to month + interval '1 00:00:00' day to second  3 from dual;  SYSDATE+INTERVAL'1-0'YEARTOMON  ------------------------------  19.06.2018 16:41:35  Прибавим к метке времени интервал в 1 наносекунду. Чтобы это сработало, нужно для интервального литерала обязательно указать разрядность 9 для долей секунды:  SQL> select  2 timestamp '2017-06-17 23:59:59.999999999' + interval '0.000000001' second,  3 timestamp '2017-06-17 23:59:59.999999999' + interval '0.000000001' second(2,9)  4 from dual;  TIMESTAMP'2017-06-1723:59:59.9 TIMESTAMP'2017-06-1723:59:59.9  ---------------------------------- ----------------------------------  17-JUN-17 11.59.59.999999999 PM 18-JUN-17 12.00.00.000000000 AM  Рассмотрим различия между типами timestamp, timestamp with local time zone и timestamp with time zone. Для этого воспользуемся таблицей tstable, созданной выше.  В столбцы ts, tstz и tsltz вставим значение systimestamp и посмотрим, что содержится в таблице:  SQL> insert into tstable (ts, tstz, tsltz) values (systimestamp, systimestamp, systimestamp);  1 row inserted  SQL> select ts, tstz, tsltz from tstable;  TS TSTZ TSLTZ  -------------------------------- ----------------------------------- ---------------------------------  18-JUN-17 02.18.37.843928 PM 18-JUN-17 02.18.37.843928 PM +10:00 18-JUN-17 02.18.37.843928 PM  Мы видим, что   * в поле ts типа timestamp нет часового пояса, * в поле tstz типа timestamp with time zone (в соответствии с названием) есть часовой пояс, * в поле tsltz типа timestamp with local time zone нет часового пояса.   В остальном значения полей совпадают.  Изменим часовой пояс сеанса и выберем строку из таблицы еще раз:  SQL> select sessiontimezone from dual;  SESSIONTIMEZONE  --------------------------------------------------------------------------------  +10:00  SQL> alter session set time\_zone = '+3:00';  Session altered.  SQL> select sessiontimezone from dual;  SESSIONTIMEZONE  --------------------------------------------------------------------------------  +03:00  SQL> select ts, tstz, tsltz from tstable;  TS TSTZ TSLTZ  -------------------------------- ----------------------------------- ---------------------------------  18-JUN-17 02.18.37.843928 PM 18-JUN-17 02.18.37.843928 PM +10:00 18-JUN-17 07.18.37.843928 AM  Значения полей ts и tstz остались прежними. А значение в поле tsltz типа timestamp with local time zone изменилось с 18-JUN-17 02.18.37.843928 PM на 18-JUN-17 07.18.37.843928 AM, то есть, стало на 7 часов меньше!  Значения типа timestamp with local time zone при их извлечении всегда представляются в часовом поясе пользователя (сеанса). Тогда как хранится значение этого типа всегда в часовом поясе базы данных, который можно узнать при помощи функции dbtimezone:  SQL> select dbtimezone from dual;  DBTIMEZONE  ----------  +11:00  Из этого следует, что если в вашей БД хранятся данные в столбцах типа timestamp with local time zone, то вы не можете безнаказанно поменять часовой пояс БД. Если это сделать, данные из столбцов типа timestamp with local time zone будут представляться неправильно - в соответствии с новым часовым поясом БД. А ведь ранее, при сохранении, они были приведены к прежнему часовому поясу БД.  (Кстати, функция localtimestamp показывает текущее время, приведенное к часовому поясу БД. А функция current\_timestampпоказывает текущее время, приведенное к часовому поясу сеанса.)  Какой же тип метки времени использовать в каком случае?  Очевидно, что если вам не важен часовой пояс, то используйте тип timestamp. Если с вашим приложением работают пользователи из разных часовых поясов и вам необходимо хранить часовой пояс пользователя в метке времени, то вам нужен тип timestamp with time zone. Это единственный из типов данных БД Oracle, который хранит часовой пояс. Если часовой пояс пользователя не важен, а важно лишь правильно регистрировать хронологическую последовательность изменений, вносимых пользователями из разных часовых поясов, то вам подойдет тип timestamp with local time zone.  Функция dump позволяет заглянуть "внутрь" хранимых в столбцах значений:  SQL> select dump(ts) from tstable;  DUMP(TS)  --------------------------------------------------------------------------------  Typ=180 Len=11: 120,117,6,22,15,19,38,50,77,81,192  SQL> select dump(tstz) from tstable;  DUMP(TSTZ)  --------------------------------------------------------------------------------  Typ=181 Len=13: 120,117,6,22,5,19,38,50,77,81,192,30,60  SQL> select dump(tsltz) from tstable;  DUMP(TSLTZ)  --------------------------------------------------------------------------------  Typ=231 Len=11: 120,117,6,22,16,19,38,50,77,81,192  Как видим, значения типов timestamp и timestamp with local time zone имеют длину 11, а для хранения значений типа timestamp with time zone нужно на 2 байта больше. В них-то и хранится часовой пояс.  Еще одна, заключительная, демонстрация того, что сохраняется и как извлекаются значения трех изучаемых типов:  SQL> insert into tstable (ts, tstz, tsltz)  2 values (  3 timestamp '2017-06-18 14:00:00 +5:00',  4 timestamp '2017-06-18 14:00:00 +5:00',  5 timestamp '2017-06-18 14:00:00 +5:00'  6 );  1 row inserted  SQL> select ts, tstz, tsltz from tstable;  TS TSTZ TSLTZ  -------------------------------- ----------------------------------- ---------------------------------  18-JUN-17 02.18.37.843928 PM 18-JUN-17 02.18.37.843928 PM +10:00 18-JUN-17 02.18.37.843928 PM  18-JUN-17 02.00.00.000000 PM 18-JUN-17 02.00.00.000000 PM +05:00 18-JUN-17 12.00.00.000000 PM  При сохранении значения timestamp '2017-06-18 14:00:00 +5:00'   * в столбце типа timestamp часовой пояс был отброшен и утерян, * в столбце типа timestamp with time zone часовой пояс сохранился, * в столбце типа timestamp with local time zone значение было приведено к часовому поясу БД (+11:00), а сам часовой пояс был отброшен и утерян.   Приведу основные характеристики типов для хранения даты и времени:   |  |  | | --- | --- | | date | хранит до секунд  не хранит часовой пояс - отбрасывает при сохранении | | timestamp | хранит до наносекунд  не хранит часовой пояс - отбрасывает при сохранении | | timestamp with time zone | хранит до наносекунд  хранит часовой пояс | | timestamp with local time zone | хранит до наносекунд  не хранит часовой пояс  при сохранении приводит значение к dbtimezone  при извлечении приводит значение к sessiontimezone | | interval year to month | хранит количество лет и месяцев | | interval day to second | хранит количество дней, часов, минут, секунд и долей секунды до наносекунд |   Функция from\_tz преобразует значение типа timestamp в значение типа timestamp with time zone; по сути, добавляет информацию о часовом поясе, оставляя дату и время без изменения:  SQL> select from\_tz(timestamp '2017-06-18 14:00:01', 'UTC') from dual;  FROM\_TZ(TIMESTAMP'2017-06-1814  --------------------------------------------------------------------------------  18-JUN-17 02.00.01.000000000 PM UTC  SQL> select from\_tz(timestamp '2017-06-18 14:00:01', '+3:00') from dual;  FROM\_TZ(TIMESTAMP'2017-06-1814  --------------------------------------------------------------------------------  18-JUN-17 02.00.01.000000000 PM +03:00  Конструкция at [time zone | local] позволяет привести значение timestamp with time zone к нужному часовому поясу:  SQL> select systimestamp at time zone 'UTC' from dual;  SYSTIMESTAMPATTIMEZONE'UTC'  --------------------------------------------------------------------------------  18-JUN-17 02.30.54.411649 AM UTC  SQL> select systimestamp at time zone 'Europe/Paris' from dual;  SYSTIMESTAMPATTIMEZONE'EUROPE/  --------------------------------------------------------------------------------  18-JUN-17 04.30.54.468217 AM EUROPE/PARIS  Конструкция systimestamp at local дает тот же результат, что и current\_timestamp:  SQL> select systimestamp at local, current\_timestamp from dual;  SYSTIMESTAMPATLOCAL CURRENT\_TIMESTAMP  ----------------------------------- -----------------------------------  18-JUN-17 03.11.54.130781 PM +10:00 18-JUN-17 03.11.54.130786 PM +10:00  На этом закончим упражнения с типами timestamp и interval. В заключение, удаляю ненужную более таблицу:  SQL> drop table tstable;  Table dropped |

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;

**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;