

3. Работа с датами



Давайте попробуем теорию про основные типы данных перевести в практическую плоскость. Посмотрим, какие есть варианты дат и когда какие применяются.

PostgreSQL поддерживает полный набор типов даты и времени SQL. Все даты считаются по Григорианскому календарю, даже для времени до его введения.

Типы даты/времени

Имя	Размер	Описание	Наименьшее значение	Наибольшее значение	Точность
<code>timestamp [(p)] [without time zone]</code>	8 байт	дата и время (без часового пояса)	4713 до н. э.	294276 н. э.	1микросекунда / 14 цифр

timestamp [(p)] with time zone	8 байт	дата и время (с часовым поясом)	4713 до н. э.	294276 н. э.	1микросекунда / 14 цифр
date	4 байта	дата (без времени суток)	4713 до н. э.	5874897 н. э.	1 день
time [(p)] [without time zone]	8 байт	время суток (без даты)	00:00:00	24:00:00	1микросекунда / 14 цифр
time [(p)] with time zone	12 байт	только время суток (с часовым поясом)	00:00:00+155 9	24:00:00-1559	1микросекунда / 14 цифр
interval [поле] [(p)]	16 байт	временной интервал	-178000000 лет	178000000 лет	1микросекунда / 14 цифр

Примечание

Стандарт SQL требует, чтобы тип timestamp подразумевал timestamp without time zone (время без часового пояса), и Postgres Pro следует этому. Для краткости timestamp with time zone можно записать как timestamptz; это расширение Postgres Pro.

Важно помнить, что при вводе значений их нужно заключать в одинарные кавычки, как и текстовые строки.

Типы time, timestamp и interval принимают необязательное значение точности **p**, определяющее, сколько знаков после запятой должно сохраняться в секундах. По умолчанию точность не ограничивается. Для типов timestamp и interval **p** может принимать значения от 0 до 6.

Для типа time **p** может принимать значения от 0 до 6 при хранении типа в восьмибайтном целом и от 0 до 10 при хранении в числе с плавающей точкой.

Тип interval дополнительно позволяет ограничить набор сохраняемых **полей** следующими фразами:

YEAR

MONTH

DAY

HOUR

MINUTE

SECOND

YEAR TO MONTH

DAY TO HOUR

DAY TO MINUTE

DAY TO SECOND

HOUR TO MINUTE

HOUR TO SECOND

MINUTE TO SECOND

Заметьте, что если указаны и **поле**, и точность **p**, указание **поля** должно включать SECOND, так как точность применима только к секундам.

Локаль и восприятие формата

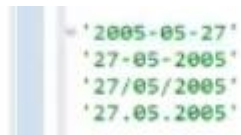
Значения даты и времени принимаются практически в любом разумном формате, включая ISO 8601, SQL-совместимый, традиционный формат POSTGRES и другие. В некоторых форматах порядок даты, месяца и года во вводимой дате неоднозначен и поэтому поддерживается явное определение формата. Для этого предназначен параметр `DateStyle`. Когда он имеет значение MDY, выбирается интерпретация месяц-день-год, значению DMY соответствует день-месяц-год, а YMD — год-месяц-день.

Локаль - набор параметров, идентифицирующих язык пользователя, страну и любые другие специальные параметры, относящиеся к языку (например: национальная валюта, клавиатурная раскладка, формат даты и времени и пр.). Обычно локаль состоит из идентификатора языка и идентификатора региона.

Нужна локаль для различных программ, чтобы они смогли правильно выбрать кодировку или показать пользовательский интерфейс на том языке, который указан в локали, при условии, что имеется перевод этого интерфейса на искомый язык.

Если вы находитесь в странах Атлантического союза либо ваша операционная система настроена на локаль, отличную от русский, то формат “день, месяц, год” восприниматься не будет. В этом регионе формат должен быть “год, месяц, число”.

В российской локали можно дату вводить в любом формате - все будет воспринято:



```
'2005-05-27'  
'27-05-2005'  
'27/05/2005'  
'27.05.2005'
```

В таблице приведены некоторые допустимые значения типа date.



```
select customer_id, rental_date  
from rental  
where rental_date between '27-05-2005' and '28-05-2005'  
  
'2005-05-27'
```

Пример	Описание
1999-01-08	ISO 8601; 8 января в любом режиме (рекомендуемый формат)
January 8, 1999	воспринимается однозначно в любом режиме datestyle
1/8/1999	8 января в режиме MDY и 1 августа в режиме DMY
1/18/1999	18 января в режиме MDY; недопустимая дата в других режимах
01/02/03	2 января 2003 г. в режиме MDY; 1 февраля 2003 г. в режиме DMY и 3 февраля 2001 г. в режиме YMD
1999-Jan-08	8 января в любом режиме
Jan-08-1999	8 января в любом режиме
08-Jan-1999	8 января в любом режиме

99-Jan-08	8 января в режиме YMD; ошибка в других режимах
08-Jan-99	8 января; ошибка в режиме YMD
Jan-08-99	8 января; ошибка в режиме YMD
19990108	ISO 8601; 8 января 1999 в любом режиме
990108	ISO 8601; 8 января 1999 в любом режиме
1999.008	год и день года
J2451187	дата по юлианскому календарю
January 8, 99 BC	99 до н. э.

Для хранения времени суток без даты предназначены типы time [(p)] without time zone и time [(p)] with time zone. Тип time без уточнения эквивалентен типу time without time zone.

Допустимые вводимые значения этих типов состоят из записи времени суток и необязательного указания часового пояса. Если в значении для типа time without time zone указывается часовой пояс, он просто игнорируется. Так же будет игнорироваться дата, если её указать, за исключением случаев, когда в указанном часовом поясе принят переход на летнее время, например America/New_York. В данном случае указать дату необходимо, чтобы система могла определить, применяется ли обычное или летнее время. Соответствующее смещение часового пояса записывается в значении time with time zone.

Вводимое время

Пример	Описание
--------	----------

04:05:06.789	ISO 8601
04:05:06	ISO 8601
040506	ISO 8601
04:05 AM	то же, что и 04:05; AM не меняет значение времени
04:05 PM	то же, что и 16:05; часы должны быть <= 12
04:05:06.789-8	ISO 8601
04:05:06-08:00	ISO 8601
04:05-08:00	ISO 8601
040506-08	ISO 8601
04:05:06 PST	часовой пояс задаётся аббревиатурой
2003-04-12 04:05:06 America/New_York	часовой пояс задаётся полным названием

Вводимый часовой пояс

Пример	Описание
PST	аббревиатура (Pacific Standard Time, Стандартное тихоокеанское время)

America/New_York	полное название часового пояса
PST8PDT	указание часового пояса в стиле POSIX
-8:00	смещение часового пояса PST по ISO-8601
-800	смещение часового пояса PST по ISO-8601
-8	смещение часового пояса PST по ISO-8601
zulu	принятое у военных сокращение UTC
z	краткая форма zulu

Стандарт SQL различает константы типов timestamp without time zone и timestamp with time zone по знаку «+» или «-» и смещению часового пояса, добавленному после времени. Следовательно, согласно стандарту, записи

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

должен соответствовать тип timestamp without time zone, а `TIMESTAMP '2004-10-19 10:23:54+02'` тип timestamp with time zone. PostgreSQL никогда не анализирует содержимое текстовой строки, чтобы определить тип значения, и поэтому обе записи будут обработаны как значения типа timestamp without time zone. Чтобы текстовая константа обрабатывалась как timestamp with time zone, укажите этот тип явно:

`TIMESTAMP WITH TIME ZONE '2004-10-19 10:23:54+02'`

В константе типа timestamp without time zone PostgreSQL просто игнорирует часовой пояс. То есть результирующее значение вычисляется только из полей даты/времени и не подстраивается под указанный часовой пояс.

Значения timestamp with time zone внутри всегда хранятся в UTC (Universal Coordinated Time, Всемирное скоординированное время или время по Гринвичу, GMT). Вводимое значение, в котором явно указан часовой пояс, переводится в UTC с учётом смещения данного часового пояса. Если во входной строке не указан часовой пояс,

подразумевается часовой пояс, заданный системным параметром TimeZone и время так же пересчитывается в UTC со смещением timezone.

Когда значение timestamp with time zone выводится, оно всегда преобразуется из UTC в текущий часовой пояс timezone и отображается как локальное время. Чтобы получить время для другого часового пояса, нужно либо изменить timezone, либо воспользоваться конструкцией AT TIME ZONE.

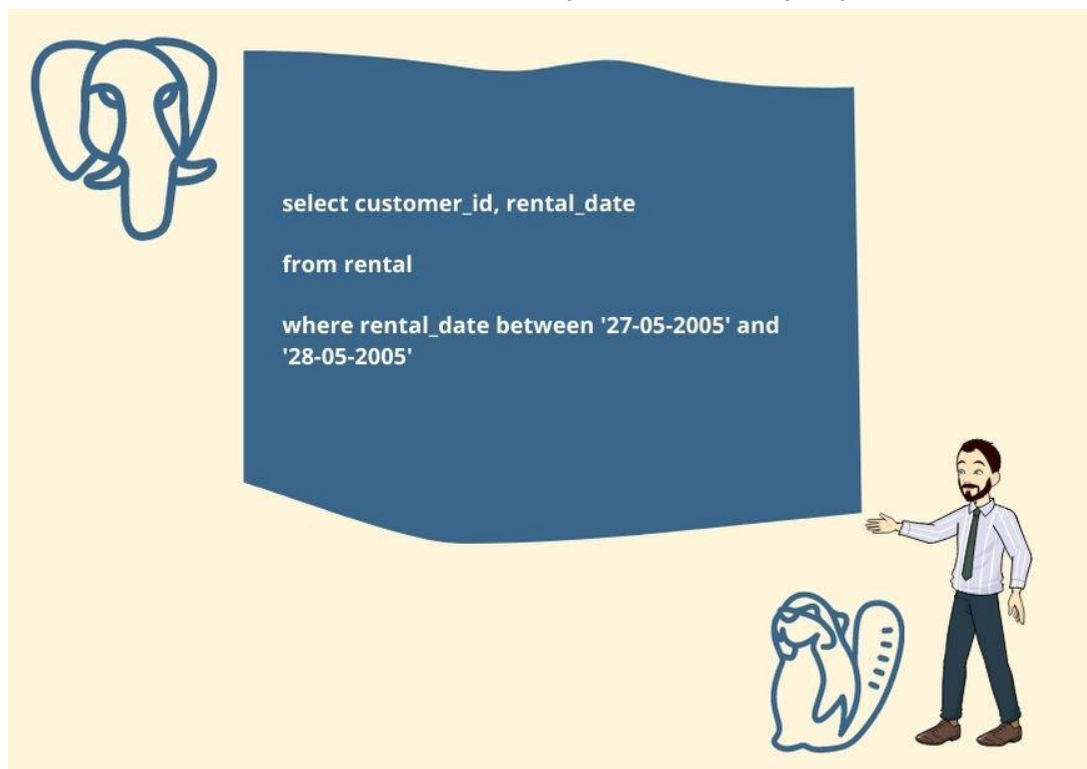
В преобразованиях между timestamp without time zone и timestamp with time zone обычно предполагается, что значение timestamp without time zone содержит местное время (для часового пояса timezone). Другой часовой пояс для преобразования можно задать с помощью AT TIME ZONE.

Правило записи

Чтобы «сказать» СУБД, что введенное значение является датой, а не простой символьной строкой, мы использовали операцию *приведения типа*. В PostgreSQL она оформляется с использованием двойного символа «двоеточие» и имени того типа, к которому мы приводим данное значение. Важно учесть, что при выполнении приведения типа производится проверка значения на соответствие формату целевого типа и множеству его допустимых значений.

Как SQL воспринимает дату?

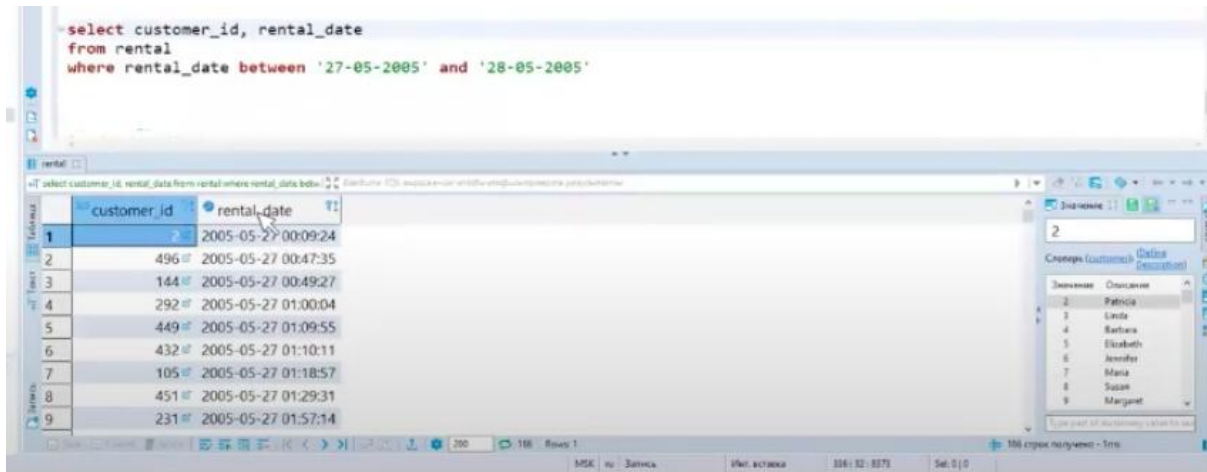
Пример: брали ли пользователи в аренду фильмы между двумя датами:



```
select customer_id, rental_date  
  
from rental  
  
where rental_date between '27-05-2005' and  
'28-05-2005'
```


- использование between

Обратите внимание, “between” означает, что выборка включает начальную и конечную даты (т.е. учитываются и 27, и 28 мая).



customer_id	rental_date
1	2005-05-27 00:09:24
2	496 2005-05-27 00:47:35
3	144 2005-05-27 00:49:27
4	292 2005-05-27 01:00:04
5	449 2005-05-27 01:09:55
6	432 2005-05-27 01:10:11
7	105 2005-05-27 01:18:57
8	451 2005-05-27 01:29:31
9	231 2005-05-27 01:57:14

Customer list from right pane:

customer_id	customer_name
1	Patricia
2	Linda
4	Barbara
5	Elizabeth
6	Jennifer
7	Maria
8	Susan
9	Margaret

Для наглядности отсортировали и увидели, что 28 мая никто фильмы в аренду не брал. Так ли это?

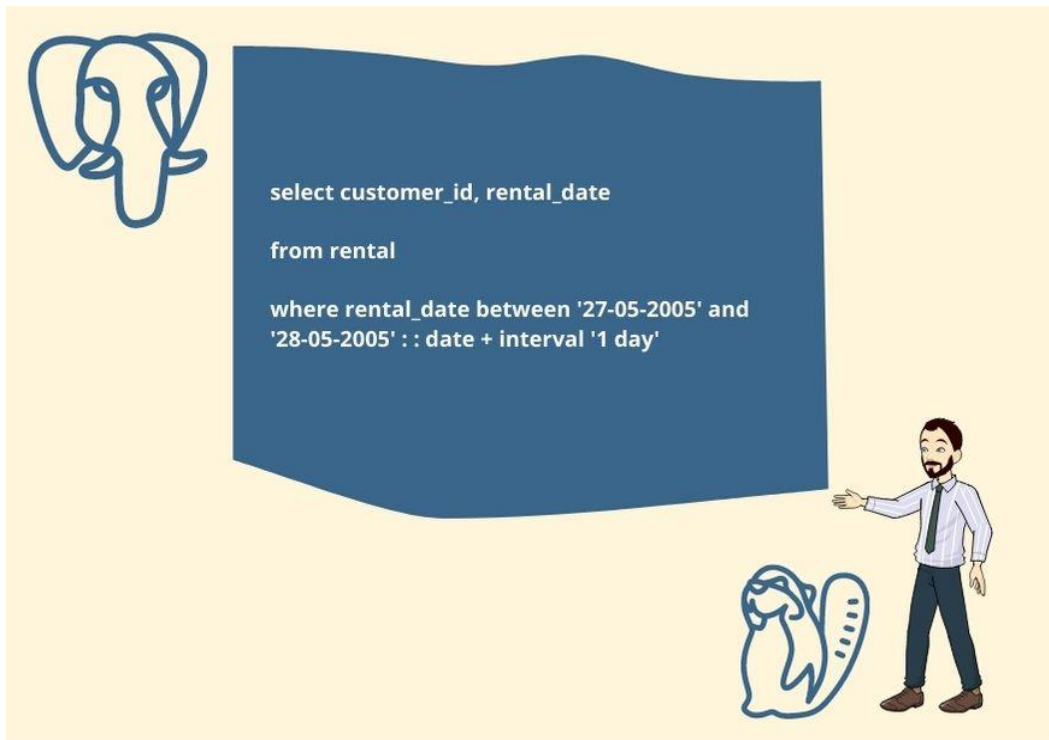
Но: по умолчанию SQL дату воспринимает как ‘28-05-2005 00:00:00’, поэтому в примере фактически 28 мая оказалось не учтено - учтены полные сутки 27 мая и в 00:00:00 часов 28 мая счетчик остановился.

Как с этим бороться?

- использование interval

Возможно несколько способов, зависит от того, что мы хотим получить:

- указать ‘28-05-2005 23:59:59’
- добавить интервал один день (interval ‘1 day’)



```
--select customer_id, rental_date  
from rental  
where rental_date between '27-05-2005' and '28-05-2005'::date + interval '1 day'  
order by rental_date desc
```

- можно напрямую указать '29-05-2005', но представьте, что это не май 2005 года, а февраль 2020 - какое число идет за 28 февраля? Если делать вручную, можно ошибиться. Т.е. так делать нежелательно.

Еще один пример: нам надо вывести платежи, поступившие после какой-то даты (т.е. не включая ее).

6* Вывести платежи поступившие после 30-04-2007

- используйте ER - диаграмму, чтобы найти подходящую таблицу
- > - строгое больше (< - строгое меньше)

```
select payment_id, payment_date
from payment p
where payment_date > '30-04-2007'
order by payment_date
```

	payment_id	payment_date
1	26588	2007-04-30 00:03:27
2	27081	2007-04-30 00:04:45
3	26293	2007-04-30 00:08:32
4	26926	2007-04-30 00:08:40
5	28870	2007-04-30 00:10:14
6	29352	2007-04-30 00:10:29
7	29478	2007-04-30 00:10:35
8	26439	2007-04-30 00:10:47
9	26347	2007-04-30 00:12:00

30.04 попадает в такую выборку. Чтобы справиться с этой задачей, мы можем пойти несколькими путями:

- добавить интервал один день (interval '1 day');
- payment_date приведем к дате и укажем '30-04-2007' в скобках:

```
select payment_id, payment_date
from payment p
where payment_date > '30-04-2007'::date + interval '1 day'
order by payment_date

select payment_id, payment_date
from payment p
where payment_date > date('30-04-2007')
order by payment_date
```

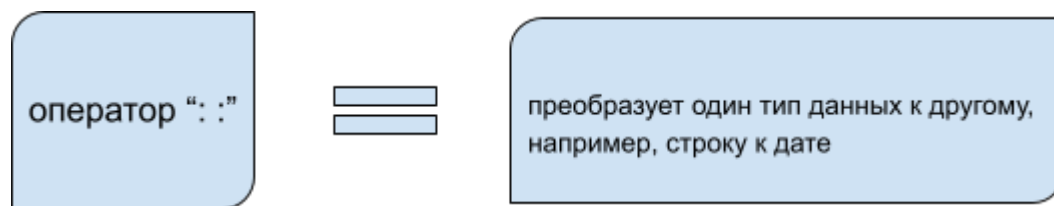
А можем из rental_date сделать дату, убрав часы, и работать именно с датой:

```
select '28/02/2020'::date + interval '1 day'

select customer_id, rental_date
from rental
where rental_date::date between '27-05-2005' and '28-05-2005'
order by rental_date desc
```

Но для этого обязательно тот столбец, с которым работаем, должны привести к дате.

Оператор для перевода:



Вывод даты/времени

В качестве выходного формата типов даты/времени можно использовать один из четырёх стилей: ISO 8601, SQL (Ingres), традиционный формат POSTGRES (формат date в Unix) или German. По умолчанию выбран формат ISO. (Стандарт SQL требует, чтобы использовался именно ISO 8601. Другой формат называется «SQL» исключительно по историческим причинам.) Примеры всех стилей вывода перечислены в таблице. Вообще со значениями типов date и time выводилась бы только часть даты или времени из показанных примеров, но со стилем POSTGRES значение даты без времени выводится в формате ISO.

Стили вывода даты/время

Стиль	Описание	Пример
ISO	ISO 8601, стандарт SQL	1997-12-17 07:37:16-08
SQL	традиционный стиль	12/17/1997 07:37:16.00 PST
Postgres	изначальный стиль	Wed Dec 17 07:37:16 1997 PST
German	региональный стиль	17.12.1997 07:37:16.00 PST

Примечание

ISO 8601 указывает, что дата должна отделяться от времени буквой T в верхнем регистре. Postgres Pro принимает этот формат при вводе, но при выводе вставляет вместо T пробел, как показано выше. Это сделано для улучшения читаемости и для совместимости с RFC 3339 и другими СУБД.

В стилях SQL и POSTGRES день выводится перед месяцем, если установлен порядок DMY, а в противном случае месяц выводится перед днём.

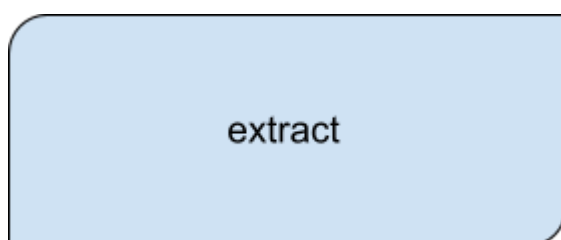
Соглашения о порядке компонентов даты

Стиль даты/времени пользователь может выбрать с помощью команды SET datestyle, параметра DateStyle в файле конфигурации postgresql.conf или переменной окружения PGDATESTYLE на сервере или клиенте.

Параметр datestyle	Порядок при вводе	Пример вывода
SQL, DMY	день/месяц/год	17/12/1997 15:37:16.00 CET
SQL, MDY	месяц/день/год	12/17/1997 07:37:16.00 PST
Postgres, DMY	день/месяц/год	Wed 17 Dec 07:37:16 1997 PST

Для большей гибкости при форматировании выводимой даты/времени можно использовать функцию to_char.

Основные функции, которые нужны при работе с датами.



Функция **extract** “вытаскивает” то, что мы передаем (в примере - год, месяц, неделю, день):

```
select extract(year from '28/02/2020'::date)
select extract(month from '28/02/2020'::date)
select extract(week from '28/02/2020'::date)
select extract(day from '28/02/2020'::date)
```

Терминология

Конкатенация - операция, которая позволяет соединить несколько текстовых строк в одну.

Используя конкатенацию, можно красиво вывести сразу месяц и год:

```
select extract(year from '28/02/2020'::date) || ' - ' || extract(month from '28/02/2020'::date)
select extract(month from '28/02/2020'::date)
select extract(week from '28/02/2020'::date)
select extract(day from '28/02/2020'::date)
```

Results 11

```
select extract(year from '28/02/2020'::date) || ' - ' || extract(month from '28/02/2020'::date)
```

column?
1 2020 - 2

Как получить платежи, поступавшие после 18-00?

Здесь так же можем использовать **extract**:

```
select payment_id, payment_date
from payment p
where extract(hour from payment_date) >= 18
order by payment_date
```

payment 11

```
select payment_id, payment_date, extract(hour from payment_date)
```

	payment_id	payment_date	date_part
1	17 793	2007-02-14 21:21:59	21
2	18 173	2007-02-14 21:23:39	21
3	19 399	2007-02-14 21:29:00	21
4	18 441	2007-02-14 21:41:12	21
5	18 698	2007-02-14 21:44:52	21
6	19 498	2007-02-14 21:44:53	21
7	18 686	2007-02-14 21:45:29	21
8	18 051	2007-02-14 22:03:35	22
9	19 036	2007-02-14 22:11:22	22

Как получить платежи по нечетным датам?

Используем остаток от деления на 2 - если 0 - четная дата, если не равен нулю - нечетная дата.

```

select payment_id, payment_date
from payment p
where extract(day from payment_date)::int % 2 = 0
order by payment_date

```

	payment_id	payment_date
1	17 793	2007-02-14 21:21:59
2	18 173	2007-02-14 21:23:39
3	19 399	2007-02-14 21:29:00
4	18 441	2007-02-14 21:41:12
5	18 698	2007-02-14 21:44:52
6	19 498	2007-02-14 21:44:53
7	18 686	2007-02-14 21:45:29
8	18 051	2007-02-14 22:03:35
9	19 036	2007-02-14 22:11:22

date_part

Extract - это часть функции **date_part**, у которой всего навсего немного другой синтаксис: указание, что надо получить и откуда:

```

select extract(year from '28/02/2020'::date) || ' - ' || extract(month from '28/02/2020'::date)
select extract(month from '28/02/2020'::date)
select extract(week from '28/02/2020'::date)
select extract(day from '28/02/2020'::date)
select date_part('year', '28/02/2020'::date)

```

Когда вы работаете с **date_part**, вы работаете с конкретным значением, с конкретным числом, которое “выдергиваете”. И здесь часто возникает ошибка. Например, вам нужно получить данные за апрель месяц за несколько лет (апрель 2020, апрель 2019, апрель 2018 года). И когда вы будете работать с **date_part**, вы получаете просто апрель. Вам нужно выдергивать 2 столбца - именно год и месяц, а это неудобно.

date_trunc

Для того, чтобы работать с конкретным значением в рамках нужного года, можно использовать функцию **date-trunc**. Она имеет такой же синтаксис и если мы напишем 'февраль', то получим 1 февраля 2020 года.

```
select extract(year from '28/02/2020'::date) || ' - ' || extract(month from '28/02/2020'::date)
select extract(month from '28/02/2020'::date)
select extract(week from '28/02/2020'::date)
select extract(day from '28/02/2020'::date)
select date_part('year', '28/02/2020'::date), date_part('month', '28/02/2020'::date)
select date_trunc('month', '28/02/2020'::date)
```

Функция **date-trunc** будет выводить при запросе месяца - на 1 число месяца, при запросе года - на первое января., т.е. конкретное число. Применяется при сравнении на конкретную дату.

now

Если вы хотите получить текущую дату и время, используйте функцию **now**.

```
select now()
now
2021-03-26 21:45:08
```

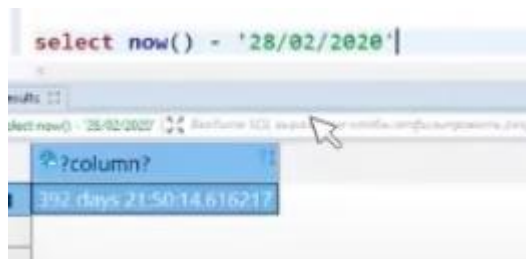

Как создать колонку которая по умолчанию будет содержать текущее время?

Используйте CURRENT_TIMESTAMP:

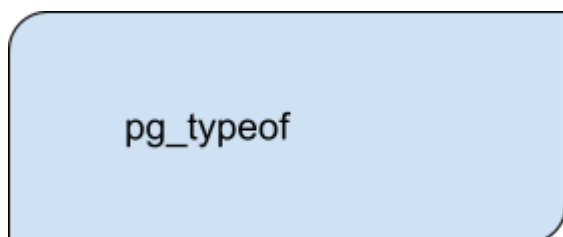
```
CREATE TABLE test (x int, modtime TIMESTAMP DEFAULT CURRENT_TIMESTAMP );
```

Входные значения now, today, tomorrow и yesterday вполне корректно работают в интерактивных SQL-командах, но когда команды сохраняются для последующего выполнения, например в подготовленных операторах, представлениях или определениях функций, их поведение может быть неожиданным. Такая строка может преобразоваться в конкретное значение времени, которое затем будет использоваться гораздо позже момента, когда оно было получено. В таких случаях следует использовать одну из SQL-функций. Например, CURRENT_DATE + 1 будет работать надёжнее, чем 'tomorrow'::date.

Как найти, сколько дней прошло между датами (например, сегодняшней датой и 28 февраля 2020 года)?

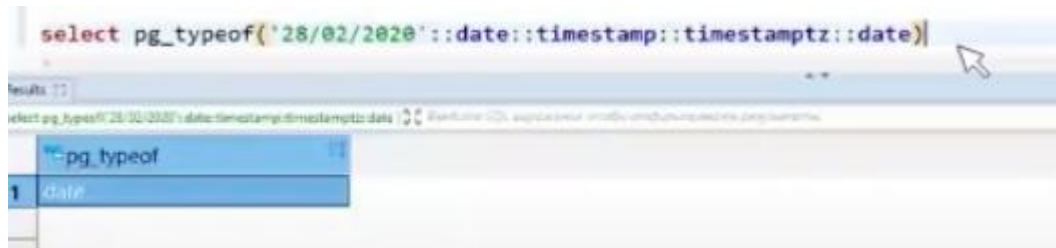


А вот чтобы вывести не в днях, а в годах или месяцах, придется писать сложные запросы (“костылить”), просто перевести в нужный формат, к сожалению, не получится.

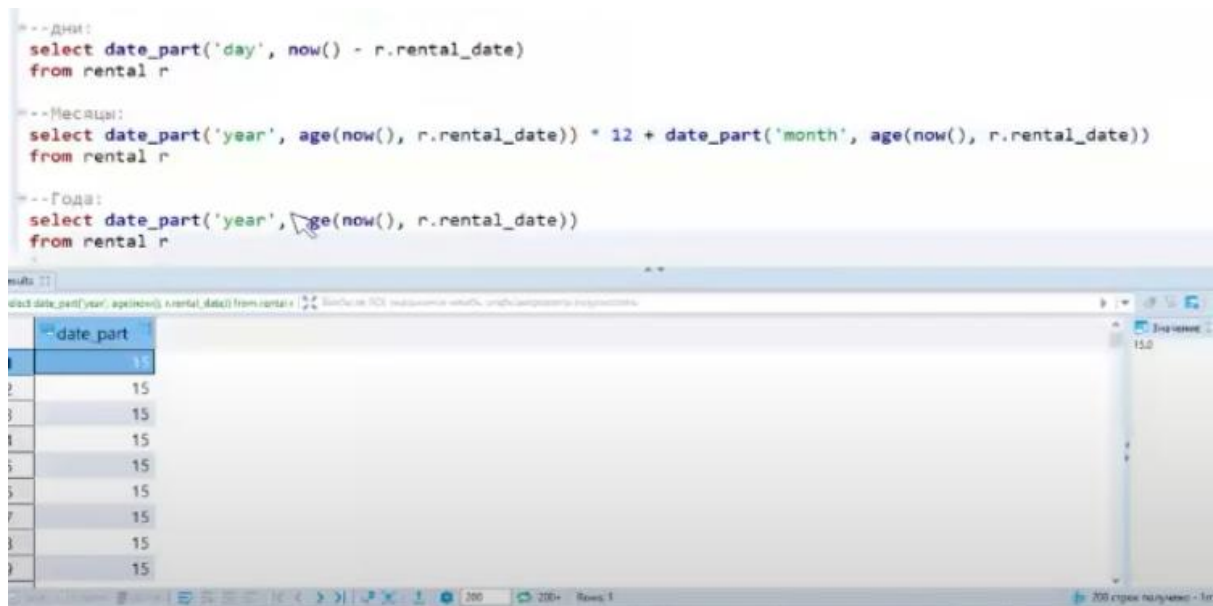


Можете переводить данные из просто даты в дату со временем, далее - дату со временем и с часовым поясом, потом - обратно.

Для этого используется функция **pg_typeof**.

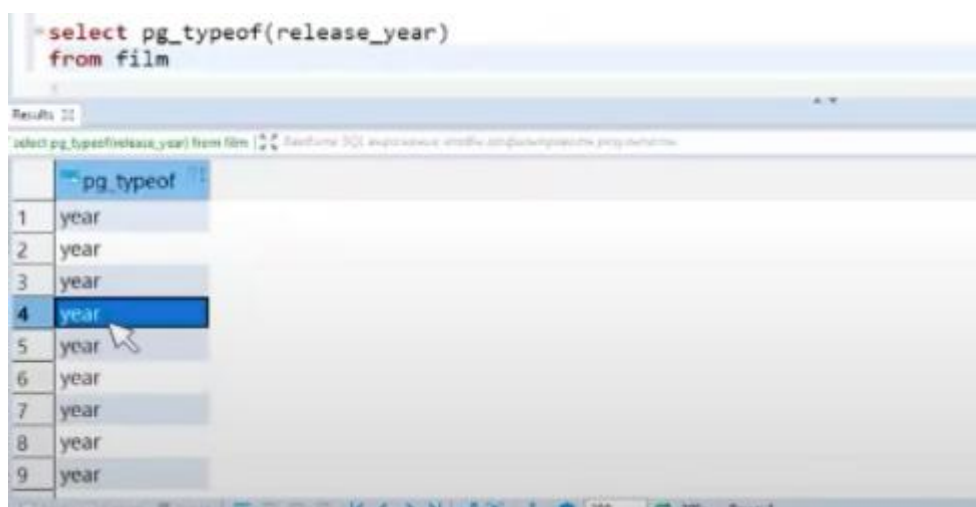


Как сделать классическую разбивку по дням, месяцам, годам (часто бывает нужно для кадровиков):



Всегда, когда видите какие-то непонятные типы данных, проверяйте, что это за данные.

Например: чем является “год выпуска” (film.release_year)?



Есть вообще в этих данных год? Что такое year?

На самом деле, тип данных здесь - integer - пользовательский тип. Это легко проверяется в “Типе данных”.

The screenshot shows the PostgreSQL pgAdmin interface. On the left, the 'public' schema is expanded, and the '123 year' domain is selected. The right pane displays the 'Свойства' (Properties) for the 'year' domain. The 'Name' is 'year', 'ID' is '16397', 'Тип типа' (Type type) is 'Domain', 'Категория типа' (Type category) is 'Numeric', 'Владелец' (Owner) is 'postgres', and 'Описание' (Description) is empty. Below the properties, the 'Исходный код' (Source code) tab is active, showing the SQL code: `-- DROP DOMAIN year;` and `CREATE DOMAIN year AS integer CONSTRAINT CHECK (VALUE >= 1901 AND VALUE <= 2155);`. The 'CREATE DOMAIN year AS integer' line is highlighted with a red box.

This screenshot is identical to the one above, showing the PostgreSQL pgAdmin interface with the 'year' domain selected. The 'Свойства' (Properties) pane shows the same details: Name 'year', ID '16397', Type 'Domain', Category 'Numeric', Owner 'postgres', and an empty description. The 'Исходный код' (Source code) tab displays the SQL code: `-- DROP DOMAIN year;` and `CREATE DOMAIN year AS integer CONSTRAINT CHECK (VALUE >= 1901 AND VALUE <= 2155);`. The 'CREATE DOMAIN year AS integer' line is highlighted with a red box.