Manejo de fechas en Postgresql

1. Tipos de fecha / hora

PostgreSQL admite el conjunto completo de tipos de fecha y hora de SQL , que se muestran en la siguiente tabla. Las operaciones disponibles en estos tipos de datos se describen en las siguientes secciones. Las fechas se cuentan de acuerdo con el calendario gregoriano, incluso en los años anteriores a la introducción de ese calendario.

Nombre	Tamaño de almacenamiento	Descripción	Valor mínimo	Valor máximo	Resolución
`timestamp [ (p) ] [without time zone]`	8 bytes	tanto la fecha como la hora (sin zona horaria)	4713 a. C.	294276 AD	1 microsegundo
`timestamp [ (p) ] with time zone`	8 bytes	tanto la fecha como la hora, con la zona horaria	4713 a. C.	294276 AD	1 microsegundo
`date`	4 bytes	fecha (sin hora del día)	4713 a. C.	5874897 AD	1 día
`time [ (p) ] [without time zone]`	8 bytes	hora del día (sin fecha)	00:00:00	24:00:00	1 microsegundo
`time [ (p) ] with time zone`	12 bytes	hora del día (sin fecha), con zona horaria	00:00:00+1559:	24:00:00-1559	1 microsegundo
`interval [ fields ] [ (p) ]`:	16 bytes	intervalo de tiempo	-178000000 años	178000000 años	1 microsegundo

`time`, `timestamp` e `interval` aceptan un valor de precisión opcional `p` que especifica el número de dígitos fraccionarios retenidos en el campo de segundos. De forma predeterminada, no hay un límite explícito de precisión. El rango permitido de `p` es de 0 a 6.

El tipo `interval` tiene una opción adicional, que es restringir el conjunto de campos almacenados escribiendo una de estas frases:

```
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
```

Ten en cuenta que si se especifican `fields` y `p` a la vez, el `fields` debe incluir `SECOND`, ya que la precisión se aplica solo a los segundos.

2. Entrada de fechas / horas

La entrada de un valor de fecha/hora se acepta en casi cualquier formato razonable, incluyendo ISO 8601. En algunos casos puede resultar ambiguo. Se puede definir un parámetro, `DateSytle` con una máscara de formato para indicar el orden de elementos de la fecha (día, mes y año).

Algunos ejemplos de entrada de fechas podrína ser:

- `1999-01-08`: ISO 8601.
- `January 8, 1999`
- `1999-Jan-08`
- `19990108`: ISO 8601
- `990108`: ISO 8601
- `J2451187`: calendario Juliano (usado aun en algunos países)
- ...

Algunos ejemplos de horas podrían ser:

- `04:05:06.789`: ISO 8601
- `04:05:06`: ISO 8601
- `04:05`: ISO 8601
- `040506`: ISO 8601
- ...

Algunos ejemplos de zonas horarias podrían ser:

- `PST`: Abreviatura (para Pacific Standard Time)
- `America/New_York`: Nombre completo de zona horaria
- `PST8PDT`: Estilo POSIX para especificar zona horaria
- `-8:00`: Desplazamiento relativo en ISO-8601 para PST
- `-800`: Desplazamiento relativo en ISO-8601 para PST
- `-8`: Desplazamiento relativo en ISO-8601 para PST
- `zulu`: Abreviatura militar para UTC
- `z`: Abreviatura corta para zulu

En el caso de las marcas de tiempo `timestamp`, la entrada consiste en la concatenación de una fecha y una hora, seguida de una zona horaria (opcionalmente). Se podrían combinar elementos de los vistos anteriormente:

- `1999-01-08 04:05:06`
- `1999-01-08 04:05:06 -8:00`
- `2004-10-19 10:23:54+02`
- ...

> Si quieres saber más sobre la interpretación de fechas/horas, puedes consultar aquí: https://www.postgresql.org/docs/current/datetime-appendix.html

2.1 Valores especiales

Postgresql admite algunos valores especiales para mayor comodidad de los usuarios:

- `epoch` (`date`, `timestamp`): `1970-01-01 00:00:00+00` (hora cero del sistema Unix)
- `infinity` (`date`, `timestamp`): más tarde que todas las marcas de tiempo.
- `-infinity` (`date`, `timestamp`): antes que todas las marcas de tiempo.
- `now` (`date`, `time`, `timestamp`): hora de inicio de la transacción actual.
- `today` (`date`, `timestamp`): medianoche de hoy.
- `tomorrow` (`date`, `timestamp`): medianoche de mañana.
- `yesterday` (`date`, `timestamp`): medianoche de ayer.
- `allballs` (`time`): `00:00:00.00 UTC`

Los siguientes funciones del estándar SQL también se pueden utilizar para obtener el valor de tiempo actual para el tipo de datos correspondiente: CURRENT_DATE, CURRENT_TIME, CURRENT_TIMESTAMP, LOCALTIME, LOCALTIMESTAMP. **Ten en cuenta que son funciones, y no cadenas de caracteres, por lo que para la entrada de datos no hay que entrecomillarlas**.

2.2 Intervalos

Los valores de intervalo se pueden escribir mediante la siguiente sintaxis:

```
[@] quantity unit [quantity unit...] [direction]
```

Donde `quantity` es un número (que puede tener signo), `unit` es una unidad (`microsecond`, `millisecond`, `second`, `minute`, `hour`, `day`, `week`, `month`, `year`, `decade`, `century`, `millennium`, o abreviaturas o plurales de estas unidades), y `direction`, que puede valer `ago` o estar vacío. La arroba es un símbolo opcional.

Los intervalos de días, horas, minutos y segundos pueden especificarse de forma abreviada. Por ejemplo, `' 1 12:59:10'` es equivalente a `'1 day 12 hours 59 min 10 sec'`. También la combinación de años y meses puede expresarse separándolos con un guión. Por ejemplo, `'200-10'` es igual a `'200 years 10 months'`.

> También se puede utilizar el formato ISO8601. Para más información puedes visitar el siguiente enlace: https://www.postgresql.org/docs/current/datatype-datetime.html#DATATYPE-INTERVAL-INPUT

3. Salidas de fecha / hora

El formato de salida de los tipos de fecha / hora se puede establecer en uno de estos cuatro: ISO8601, SQL (Ingres), POSTGRES tradicional (formato de fecha Unix) o alemán. El formato predeterminado es ISO. (El estándar SQL requiere el uso del formato ISO 8601. El nombre del formato de salida "SQL" es un accidente histórico). A continuación tenemos ejemplos de cada estilo de salida. La salida de los tipos `date` y `time` es generalmente solo la parte de fecha u hora de acuerdo con los ejemplos dados. Sin embargo, el estilo POSTGRES genera valores de solo fecha en formato ISO.

- ISO (ISO 8601, tradicional SQL): `1997-12-17 07:37:16-08`
- SQL (estilo tradicional): `12/17/1997 07:37:16.00 PST`
- Postgres (estilo Unix): `Wed Dec 17 07:37:16 1997 PST`
- Alemán: `17.12.1997 07:37:16.00 PST`

> Nosotros usaremos siempre el estilo ISO 8601, aunque posiblemente adaptado al estilo de España: `17/12/1997 07:37:16`.

3.1 Salida de intervalo

La salida de un intervalo también puede ofrecerse en varios formatos: sql estándar, postgres, postgres verbose e ISO 8601. El formato por defecto es postgres.

En la siguiente tabla podemos ver algunos ejemplos:

Estilo	Intervalo año-mes	Intervalo Día-Hora	Intervalo Mixto
`sql_standard`	`1-2`	`3 4:05:06`	`-1-2 +3 -4:05:06`
`postgres`	`1 year 2 mons`	`3 days 04:05:06`	`-1 year -2 mons +3 days -04:05:06`
`postgres_verbose`	`@ 1 year 2 mons`	`@ 3 days 4 hours 5 mins 6 secs`	`@ 1 year 2 mons -3 days 4 hours 5 mins 6 secs ago`
`iso_8601`	`P1Y2M`	`P3DT4H5M6S`	`P-1Y-2M3DT-4H-5M-6S`

4. Funciones y operadores de fecha / hora

Resumir https://www.postgresql.org/docs/current/functions-datetime.html

La siguiente lista muestra las funciones disponibles para el procesamiento de valores de fecha/hora, con detalles que aparecen a continuación en este apartado. Antes, la tabla ilustra los comportamientos de los

operadores aritméticos básicos (+, *, etc.). Para conocer las funciones de formato, puedes ver el apartado siguiente.

Todas las funciones y operadores que se describen a continuación que toman entradas `time` o `timestamp` en realidad tienen en dos variantes: una que toma `time with time zone` o `timestamp with time zone` y otra que toma `time without time zone` o `timestamp without time zone`. Por brevedad, estas variantes no se muestran por separado. Además, los operadores `+` y `*` vienen en pares conmutativos (por ejemplo, fecha + entero y entero + fecha) se muestran una sola vez.

Operador	Descripción	Ejemplo
`date + integer → date`	Agregar varios días a una fecha	`date '2001-09-28' + 7 → 2001-10-05`
`date + interval → timestamp`	Agregar un intervalo a una fecha	`date '2001-09-28' + interval '1 hour' → 2001-09-28 01:00:00`
`date + time → timestamp`	Agregar una hora del día a una fecha	`date '2001-09-28' + time '03:00' → 2001-09-28 03:00:00`
`interval + interval → interval`	Agregar intervalos	`interval '1 day' + interval '1 hour' → 1 day 01:00:00`
`timestamp + interval → timestamp`	Agregar un intervalo a una marca de tiempo	`timestamp '2001-09-28 01:00' + interval '23 hours' → 2001-09-29 00:00:00`
`time + interval → time`	Agregar un intervalo a un tiempo	`time '01:00' + interval '3 hours' → 04:00:00`
`- interval → interval`	Negar un intervalo	`- interval '23 hours' → -23:00:00`
`date - date → integer`	Reste fechas, produciendo el número de días transcurridos	`date '2001-10-01' - date '2001-09-28' → 3`
`date - integer → date`	Restar varios días de una fecha	`date '2001-10-01' - 7 → 2001-09-24`
`date - interval → timestamp`	Restar un intervalo de una fecha	`date '2001-09-28' - interval '1 hour' → 2001-09-27 23:00:00`
`time - time → interval`	Restar tiempos	`time '05:00' - time '03:00' → 02:00:00`
`time - interval → time`	Restar un intervalo de un tiempo	`time '05:00' - interval '2 hours' → 03:00:00`

Operador	Descripción	Ejemplo
`timestamp - interval → timestamp`	Restar un intervalo de una marca de tiempo	`timestamp '2001-09-28 23:00' - interval '23 hours' → 2001-09-28 00:00:00`
`interval - interval → interval`	Restar intervalos	`interval '1 day' - interval '1 hour' → 1 day -01:00:00`
`timestamp - timestamp → interval`	Restar marcas de tiempo (convertir intervalos de 24 horas en días, de manera similar a `justify_hours()`)	`timestamp '2001-09-29 03:00' - timestamp '2001-07-27 12:00' → 63 days 15:00:00`

`interval * double precision → interval`	Multiplica un intervalo por un escalar	`interval '1 second' * 900 → 00:15:00`
		`interval '1 day' * 21 → 21 days`
		`interval '1 hour' * 3.5 → 03:30:00`

`interval / double precision → interval`	Dividir un intervalo por un escalar	`interval '1 hour' / 1.5 → 00:40:00`

Todas estas son las funciones para manejo de fechas/horas:

- `age (timestamp, timestamp) → interval`: Restar argumentos, lo que produce un resultado "simbólico" que utiliza años y meses, en lugar de solo días.
  `age(timestamp '2001-04-10', timestamp '1957-06-13') → 43 years 9 mons 27 days`
- `age ( timestamp ) → interval`: Restar argumento de `current_date` (a la medianoche)
  `age(timestamp '1957-06-13') → 62 years 6 mons 10 days`
- `clock_timestamp ( ) → timestamp with time zone`: Fecha y hora actual
  `clock_timestamp() → 2019-12-23 14:39:53.662522-05`
- `current_date → date`: Fecha actual
  `current_date → 2019-12-23`
- `current_time → time with time zone`: Hora actual del día
  `current_time → 14:39:53.662522-05`
- `current_time (integer) → time with time zone`: Hora actual del día, con precisión limitada
  `current_time(2) → 14:39:53.66-05`
- `current_timestamp → timestamp with time zone` :Fecha y hora actuales
  `current_timestamp → 2019-12-23 14:39:53.662522-05`
- `current_timestamp ( integer ) → timestamp with time zone`: Fecha y hora actuales
  `current_timestamp(0) → 2019-12-23 14:39:53-05`

- `date_part ( text, timestamp ) → double precision`: Obtener el subcampo de marca de tiempo (equivalente a `extract`)
  `date_part('hour', timestamp '2001-02-16 20:38:40') → 20`

- `date_part ( text, interval ) → double precision`: Obtener subcampo de intervalo (equivalente a `extract`)
  `date_part('month', interval '2 years 3 months') → 3`

- `date_trunc ( text, timestamp ) → timestamp`: Truncar a la precisión especificada
  `date_trunc('hour', timestamp '2001-02-16 20:38:40') → 2001-02-16 20:00:00`

- `date_trunc ( text, timestamp with time zone, text ) → timestamp with time zone`: Truncar a la precisión especificada en la zona horaria especificada
  `date_trunc('day', timestamptz '2001-02-16 20:38:40+00', 'Australia/Sydney') → 2001-02-16 13:00:00+00`

- `date_trunc ( text, interval ) → interval`: Truncar a la precisión especificada
  `date_trunc('hour', interval '2 days 3 hours 40 minutes') → 2 days 03:00:00`

`extract ( ) → field from timestamp double precision`: Obtener el subcampo de marca de tiempo
`extract(hour from timestamp '2001-02-16 20:38:40') → 20`

- `extract ( ) → field from interval double precision`: Obtener subcampo de intervalo
  `extract(month from interval '2 years 3 months') → 3`

- `isfinite (date) → boolean`: Prueba para fecha finita (no +/- infinito)
  `isfinite(date '2001-02-16') → true`

- `isfinite (timestamp) → boolean`: Prueba de marca de tiempo finita (no +/- infinito)
  `isfinite(timestamp 'infinity') → false`

- `isfinite ( interval ) → boolean`: Prueba de intervalo finito (actualmente siempre es cierto)
  `isfinite(interval '4 hours') → true`

- `justify_days ( interval ) → interval`: Ajustar el intervalo para que los períodos de 30 días se representen como meses
  `justify_days(interval '35 days') → 1 mon 5 days`

- `justify_hours ( interval ) → interval`: Ajustar el intervalo para que los períodos de 24 horas se representen como días
  `justify_hours(interval '27 hours') → 1 day 03:00:00`

- `justify_interval ( interval ) → interval`: Ajuste el intervalo usando justify_daysy justify_hours, con ajustes de señal adicionales
  `justify_interval(interval '1 mon -1 hour') → 29 days 23:00:00`

- `localtime → time`: Hora actual del día
  `localtime → 14:39:53.662522`

- `localtime (integer) → time`: Hora actual del día, con precisión limitada
  `localtime(0) → 14:39:53`

`localtimestamp → timestamp`: Fecha y hora actuales
`localtimestamp → 2019-12-23 14:39:53.662522`

- `localtimestamp ( integer ) → timestamp`: Fecha y hora actuales, con precisión limitada
  `localtimestamp(2) → 2019-12-23 14:39:53.66`
- `make_date ( year int, month int, day int ) → date`: Crear fecha a partir de campos de año, mes y día
  `make_date(2013, 7, 15) → 2013-07-15`
- `make_interval ( [ years int [, months int [, weeks int [, days int [, hours int [, mins int [, secs double precision ]]]]]] ] ) → interval` Crea campos de intervalo de años, meses, semanas, días, horas, minutos y segundos, cada uno de los cuales puede predeterminarse a cero
  `make_interval(days => 10) → 10 days`
- `make_time ( hour int, min int, sec double precision ) → time`: Cree el tiempo a partir de los campos de hora, minutos y segundos
  `make_time(8, 15, 23.5) → 08:15:23.5`
- `make_timestamp ( year int, month int, day int, hour int, min int, sec double precision ) → timestamp`: Crea una marca de tiempo a partir de los campos de año, mes, día, hora, minuto y segundo
  `make_timestamp(2013, 7, 15, 8, 15, 23.5) → 2013-07-15 08:15:23.5`
- `make_timestamptz ( year int, month int, day int, hour int, min int, sec double precision [, timezone text ] ) → timestamp with time zone`: Crea una marca de tiempo con la zona horaria de los campos de año, mes, día, hora, minuto y segundos; si timezoneno se especifica, se utiliza la zona horaria actual
  `make_timestamptz(2013, 7, 15, 8, 15, 23.5) → 2013-07-15 08:15:23.5+01`
- `now ( ) → timestamp with time zone`: Fecha y hora actuales.
  `now() → 2019-12-23 14:39:53.662522-05`
- `statement_timestamp ( ) → timestamp with time zone`: Fecha y hora actual
  `statement_timestamp() → 2019-12-23 14:39:53.662522-05`
- `timeofday ( ) → text`: Fecha y hora actuales (como clock_timestamp, pero como una cadena de texto)
  `timeofday() → Mon Dec 23 14:39:53.662522 2019 EST`
- `transaction_timestamp ( ) → timestamp with time zone`: Fecha y hora actuales
  `transaction_timestamp() → 2019-12-23 14:39:53.662522-05`
- `to_timestamp ( double precision ) → timestamp with time zone`: Convierta una época de Unix (segundos desde 1970-01-01 00: 00: 00 + 00) a marca de tiempo con zona horaria.
  `to_timestamp(1284352323) → 2010-09-13 04:32:03+00`

Además de todas estas funciones, Postgresql añade el operador `OVERLAPS` para saber si dos fechas/horas se solapan.

```
(start1, end1) OVERLAPS (start2, end2)
(start1, length1) OVERLAPS (start2, length2)
```

Esta expresión da como resultado verdadero cuando dos períodos de tiempo (definidos por sus puntos finales) se superponen, falso cuando no se superponen. Los puntos finales se pueden especificar como pares de fechas, horas o marcas de tiempo; o como una fecha, hora o sello de tiempo seguido de un intervalo. Cuando se proporciona un par de valores, primero se puede escribir el inicio o el final; `OVERLAPS` toma automáticamente el valor anterior del par como inicio. Se considera que cada período de tiempo representa el intervalo semiabierto , a menos que y sean iguales, en cuyo caso representa ese único instante de tiempo. Esto significa, por ejemplo, que dos períodos de tiempo con solo un punto final en común no se superponen (`start <= time < endstartend`)

```
SELECT (DATE '2001-02-16', DATE '2001-12-21') OVERLAPS
       (DATE '2001-10-30', DATE '2002-10-30');
Result: true
SELECT (DATE '2001-02-16', INTERVAL '100 days') OVERLAPS
       (DATE '2001-10-30', DATE '2002-10-30');
Result: false
SELECT (DATE '2001-10-29', DATE '2001-10-30') OVERLAPS
       (DATE '2001-10-30', DATE '2001-10-31');
Result: false
SELECT (DATE '2001-10-30', DATE '2001-10-30') OVERLAPS
       (DATE '2001-10-30', DATE '2001-10-31');
Result: true
```

> Si quieres saber más sobre la función `EXTRACT` puedes encontrar toda la documentación aquí: https://www.postgresql.org/docs/current/functions-datetime.html#FUNCTIONS-DATETIME-EXTRACT

5. Funciones de transformación entre fechas/horas y otros tipos de datos.

Las funciones de formato de PostgreSQL proporcionan un poderoso conjunto de herramientas para convertir varios tipos de datos (fecha / hora, entero, punto flotante, numérico) en cadenas formateadas y para convertir de cadenas formateadas a tipos de datos específicos. La siguiente los enumera. Todas estas funciones siguen una convención de llamada común: el primer argumento es el valor a formatear y el segundo argumento es una plantilla que define el formato de salida o entrada.

Función	Descripción	Ejemplo
`to_char ( timestamp, text ) → text` `to_char ( timestamp with time zone, text ) → text`	Convierte la marca de tiempo en una cadena de acuerdo con el formato dado.	`to_char(timestamp '2002-04-20 17:31:12.66', 'HH12:MI:SS') → 05:31:12`

Función	Descripción	Ejemplo
`to_char ( interval, text ) → text`	Convierte el intervalo en una cadena de acuerdo con el formato dado.	`to_char(interval '15h 2m 12s', 'HH24:MI:SS') → 15:02:12`
`to_char ( numeric_type, text ) → text`	Convierte el número en una cadena de acuerdo con el formato dado; disponible para `integer`, `bigint`, `numeric`, `real`, `double precision`.	`to_char(125, '999') → 125` `to_char(125.8::real, '999D9') → 125.8` `to_char(-125.8, '999D99S') → 125.80-`
`to_date ( text, text ) → date`	Convierte la cadena a la fecha de acuerdo con el formato dado.	`to_date('05 Dec 2000', 'DD Mon YYYY') → 2000-12-05`
`to_number ( text, text ) → numeric`	Convierte una cadena en numérica según el formato dado	`to_number('12,454.8-', '99G999D9S') → -12454.8`
`to_timestamp ( text, text ) → timestamp with time zone`	Convierte una cadena en una marca de tiempo según el formato dado.	to_timestamp`('05 Dec 2000', 'DD Mon YYYY') → 2000-12-05 00:00:00-05`

Los patrones de plantilla de formato para fecha hora son los siguientes:

- HH: hora del día (01-12)
- HH12: hora del día (01-12)
- HH24: hora del día (00-23)
- MI: minuto (00–59)
- SS: segundo (00–59)
- MS: milisegundos (000–999)
- US: microsegundo (000000–999999)
- FF1: décimo de segundo (0–9)
- FF2: centésima de segundo (00–99)
- FF3: milisegundos (000–999)
- FF4: décimo de milisegundo (0000–9999)
- FF5: centésima de milisegundo (00000–99999)
- FF6: microsegundo (000000–999999)
- SSSS, SSSSS: segundos después de la medianoche (0–86399)
- AM, am, PM o pm: indicador de antes de mediodía/después de mediodía (sin puntos)
- A.M., a.m., P.M.Op.m.: indicador de antes de mediodía/después de mediodía (con puntos)
- Y,YYY: año (4 o más dígitos) con coma
- YYYY: año (4 o más dígitos)
- YYY: últimos 3 dígitos del año
- YY: últimos 2 dígitos del año
- Y: último dígito del año
- IYYY: Año de numeración semanal ISO 8601 (4 o más dígitos)

- IYY: últimos 3 dígitos del año de numeración semanal ISO 8601
- IY: últimos 2 dígitos del año de numeración de la semana ISO 8601
- I: último dígito del año de numeración semanal ISO 8601
- BC, bc, AD o ad: indicador de era (sin puntos)
- B.C., b.c., A.D. o a.d.: indicador de era (con puntos)
- MONTH: nombre completo del mes en mayúsculas (relleno en blanco a 9 caracteres)
- Month: nombre del mes completo en mayúscula (relleno en blanco hasta 9 caracteres)
- month: nombre completo del mes en minúsculas (relleno en blanco a 9 caracteres)
- MON: nombre del mes en mayúsculas abreviado (3 caracteres en inglés, las longitudes localizadas varían)
- Mon: nombre del mes abreviado en mayúscula (3 caracteres en inglés, las longitudes localizadas varían)
- mon: nombre del mes en minúsculas abreviado (3 caracteres en inglés, las longitudes localizadas varían)
- MM: número de mes (01-12)
- DAY: nombre completo del día en mayúsculas (relleno en blanco a 9 caracteres)
- Day: nombre del día completo en mayúscula (relleno en blanco a 9 caracteres)
- day: nombre completo del día en minúsculas (relleno en blanco a 9 caracteres)
- DY: nombre del día en mayúsculas abreviado (3 caracteres en inglés, las longitudes localizadas varían)
- Dy: nombre del día abreviado en mayúscula (3 caracteres en inglés, las longitudes localizadas varían)
- dy: nombre del día abreviado en minúsculas (3 caracteres en inglés, las longitudes localizadas varían)
- DDD: día del año (001–366)
- IDDD: día del año de numeración de semanas ISO 8601 (001–371; el día 1 del año es el lunes de la primera semana ISO)
- DD: día del mes (01–31)
- D: día de la semana, domingo (1) a sábado (7)
- ID: ISO 8601 día de la semana, lunes (1) a domingo (7)
- W: semana del mes (1–5) (la primera semana comienza el primer día del mes)
- WW: semana número del año (1–53) (la primera semana comienza el primer día del año)
- IW: número de semana del año de numeración de semanas ISO 8601 (01–53; el primer jueves del año corresponde a la semana 1)
- CC: siglo (2 dígitos) (el siglo XXI comienza en 2001-01-01)
- J: Día juliano (días enteros desde el 24 de noviembre de 4714 a. C. a la medianoche UTC)
- Q: trimestre
- RM: mes en mayúsculas en números romanos (I – XII; I = enero)
- rm: mes en números romanos en minúscula (i – xii; i = enero)
- TZ: abreviatura de la zona horaria en mayúsculas (solo se admite en to_char)
- tz: abreviatura de zona horaria en minúscula (solo admitida en to_char)
- TZH: horas de zona horaria
- TZM: minutos de zona horaria
- OF: Desplazamiento de zona horaria de UTC (solo admitido en to_char)

Se pueden utilizar algunos modificadores para los patrones anteriores:

Modificador	Descripción	Ejemplo
FM prefijo	modo de relleno (suprime los ceros iniciales y los espacios en blanco de relleno)	`FMMonth`

Modificador	Descripción	Ejemplo
TH sufijo	sufijo de número ordinal en mayúsculas	`DDTH`, p.ej, `12TH`
th sufijo	sufijo de número ordinal en minúscula	`DDth`, p.ej, `12th`
FX prefijo	opción global de formato fijo	`FX Month DD Day`
TM prefijo	modo de traducción (use nombres de días y meses localizados basados en lc_time)	`TMMonth`

> Los patrones para las funciones numéricas los pues encontrar en https://www.postgresql.org/docs/current/functions-formatting.html