IT'S A DATE

Jed Rembold

Wednesday, October 16, 2024

ANNOUNCEMENTS

- Homework 6 due tomorrow night!
- I'll have HW5 feedback to you by tomorrow
- We'll be starting in on Chapter 13 on Monday
- Midterm 2 is two weeks from today
  - HW7 will on the Ch 10 materials (ALTER/UPDATE/DELETE), which will be the last of the testable material
- Polling today: **polling.jedrembold.prof**

A transaction is best used to accomplish all of the following except for which?

A) Recovering from a possible drive failure mid-query

B) Prevent two simultaneous database accesses from getting different values

C) Transfer information from one table to another

D) Correcting a mistake

REMINDERS

- We have about 4 current date or time related data types
- Date time types:
  - `DATE` : holds a single individual day
  - `TIME` : holds a single individual time
  - `TIMESTAMP` or variants with `TIMESTAMPZ` : holds a combination of date and time, along with a potential time zone
- Interval types:
  - `INTERVAL` : holds a duration of time

EXTRACTING PIECES

- Having all the information in one value is convenient, but sometimes you only need pieces
  - The hour from the time, or the month from the date
- These can be particularly important with aggregates!
- Two methods to extract pieces of any datetime or interval type:
  - SQL standard: `EXTRACT( piece FROM datetime_value )`
  - Postgres specific: `date_part( text, datetime_value )`
- Both will return a `DOUBLE PRECISION` value of whatever part was requested

PARTS OF EXTRACT

○ You have a wide variety of what you can extract

text	Description
century	What century the date is in. 1st century starts 0001-01-01★
day	What day of the month
decade	The year divided by 10
dow	The day of the week (0-6, starting with Sunday)
doy	The day of the year
epoch	Number of seconds since 1970-01-01
hour	The current hour (0-23)
microseconds	The number of microseconds

text	Description
milliseconds	The number of milliseconds
minute	The minute
month	The month (1-12)
quarter	What quarter of the year (1-4)
second	The number of seconds
timezone	The timezone offset in seconds
timezone_hour	The timezone offset in hours
week	What week of the year. ISO weeks start on Monday
year	The year

★ – If you disagree with this, please write your complaint to: Pope, Cathedral Saint-Peter of Roma, Vatican.

☰ ‹ ›

REVERSING IT

- Often times existing data sets have already separated out different aspects of the date or time
  - Year, month, and day might be in different columns for example
- It can be useful to "stitch" these together into an actual datetime type for further use.
- Postgres gives you a handful of functions to do so:
  - `make_date( year, month, day )` : Returns a new `DATE` type value
  - `make_time( hour, minute, seconds )` : Returns a new `TIME` type value (with no timezone)
  - `make_timestampz(year,month,day,hour,minute,second,time zone)` : Returns a new `TIMESTAMPZ` type value
  - `make_timestamp` and `make_interval` also exist

≡   〈 〉

AGING WELL

- Subtracting two `DATE` type values will give just an `INT` (in days)

- Subtracting two `TIMESTAMP` type values will give an `INTERVAL`, with the biggest "unit" in days

- Using Postgres's `age()` function can smooth over both and give units larger than days

  - `age( datetime1, datetime2 )`: Subtracts datetime2 from datetime1

- This can **still** give you awkward interval units at times though, so also consider using `justify_interval( interval )`, which breaks intervals into divisions that don't exceed a categories max

  - Hours would always be between 0 and 23 for instance, or months between 1 and 12

  - Especially if you want to extract a particular part, this is highly recommended

WHAT TIME IS IT?

◯ Standard SQL also provides constants for grabbing the current system time and date

function	description
`current_date`	Returns the current date
`current_time`	Returns the current time with timezone
`localtime`	Returns the current time without timezone
`current_timestamp` ★	Returns the current date and time with timezone
`localtimestamp`	Returns the current date and time without timezone

★ – Postgres also offers the shorter `now( )` *function* to do the same thing

≡   〈 〉

CURRENT VS CLOCK

- Any query using `current_timestamp` has it computed once at the start of a query

  - This is frequently desired for logging, so that you just get 1 consistent time for any records added from a single query

- If you **want** record-by-record time keeping, you should use `clock_timestamp()` instead, which will work the same way but be updated before every value written to the table

TIME ZONES

- Dealing with time zones can be a headache, and it is a very nice feature that Postgres can work with them smoothly

- By default, Postgres will *display* any timestamp with a time zone with the time as you would measure it in your current system timezone

- What is your current system timezone?

  - ```
    SHOW timezone;
    ```

- Getting general information about timezones:

  - Getting abbreviations:

    - ```
      SELECT * FROM pg_timezone_abbrevs;
      ```

  - Getting full names:

    - ```
      SELECT * FROM pg_timezone_names;
      ```

TELEPORTATION

- It can sometimes be useful to switch your "current" time zone
  - Maybe it is easier to compare times to someone else living in that time zone
- Several methods to make the switch:
  - Change your `postgressql.conf` file, which controls your Postgres server. Only recommended if you have permanently moved elsewhere and the database time zone has not updated appropriately.
  - Set future queries in a single session to be from a new timezone:
    ```
    SET timezone TO time_zone_name_or_abbrv;
    ```
    - This will also adjust what values your `localtime` or `localtimestamp` report!
  - Transform a single query to be reported in a different time zone:
    ```
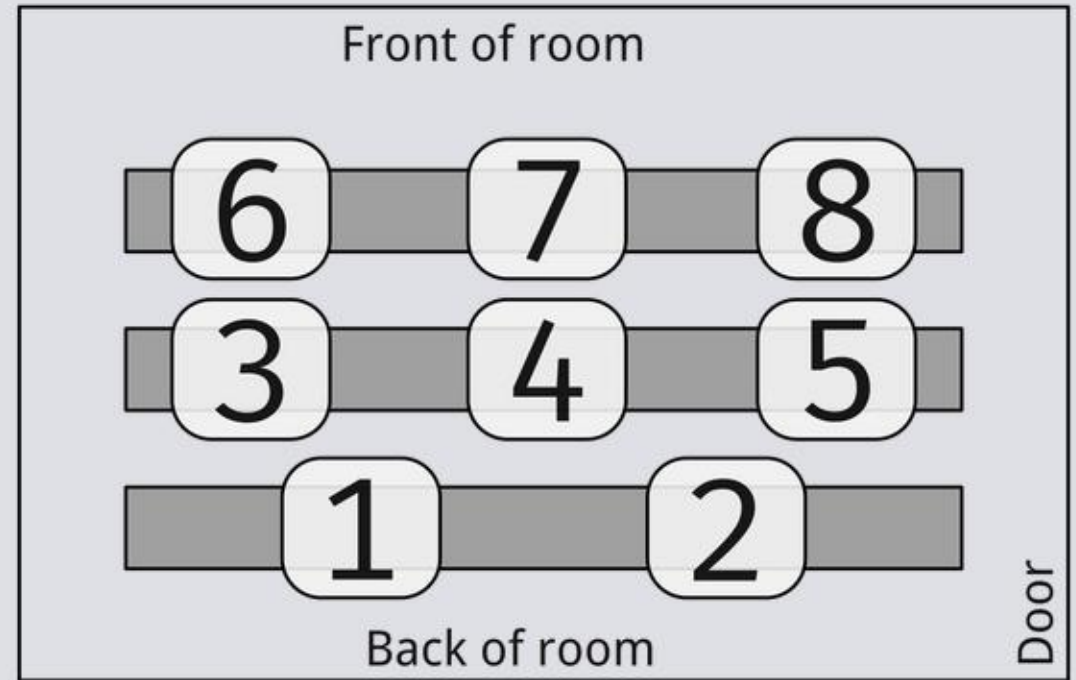    SELECT dt_col_name AT TIME ZONE tz_name_or_abbrv
    FROM tablename;
    ```

≡ ⟨ ⟩

ACTIVITY

- Using the taxi rides dataset see if you can:
  - Compute the total number of rides given each hour of the day over the month
  - Compute the average cost of rides each day of the month
  - Compute the median cost of rides over each day of the week
  - Compute the average duration of rides (in min) over each hour of the day
- In your groups:
  - Only one person typing and working with the database
  - Other folks can have documentation or slides up on their computers for reference
  - Rotate who is typing every 5-6 minutes

GROUPS

- Group 1: Matthew, Myles, Mallory
- Group 2: Hannah, Evan, Marcus
- Group 3: Jordan, Nick, Sam J
- Group 4: Harleen, Aurora, Tippy
- Group 5: AJ, Finn, Sergio
- Group 6: Haley, Sam H, Grace
- Group 7: Tiffany, Greg, Connor, Jerrick
- Group 8: Dayton, Jack, Michael

Front of room

6	7	8
3	4	5
1	2	

Back of room

Door

Group Areas