Дата и время
Что с этим делать

PROFESSIONAL Postgres

postgrespro.ru

Фролков Иван

Какие проблемы возникают и почему?

Работа с датой и временем нередко вызывает вопросы и долгие мучительные разбирательства

Неясно время наступления события:

- Сервера в Москве и Новосибирске пишут общий лог со своим местным временем, в результате сложно разобраться, что когда было
  - Злокачественный вариант — когда еще и выставляется непонятная таймзона в сессиях

Невозможно построить корректный отчет

- Сверка отчетов с географически удаленными контрагентами: отчетный период начинается и заканчивается у каждого по местному времени
- И т.п.

Заря приходит с Востока

У нас утро наступает раньше, чем в Лондоне, поэтому у нас таймзона +03, +04, +05 и т. д.

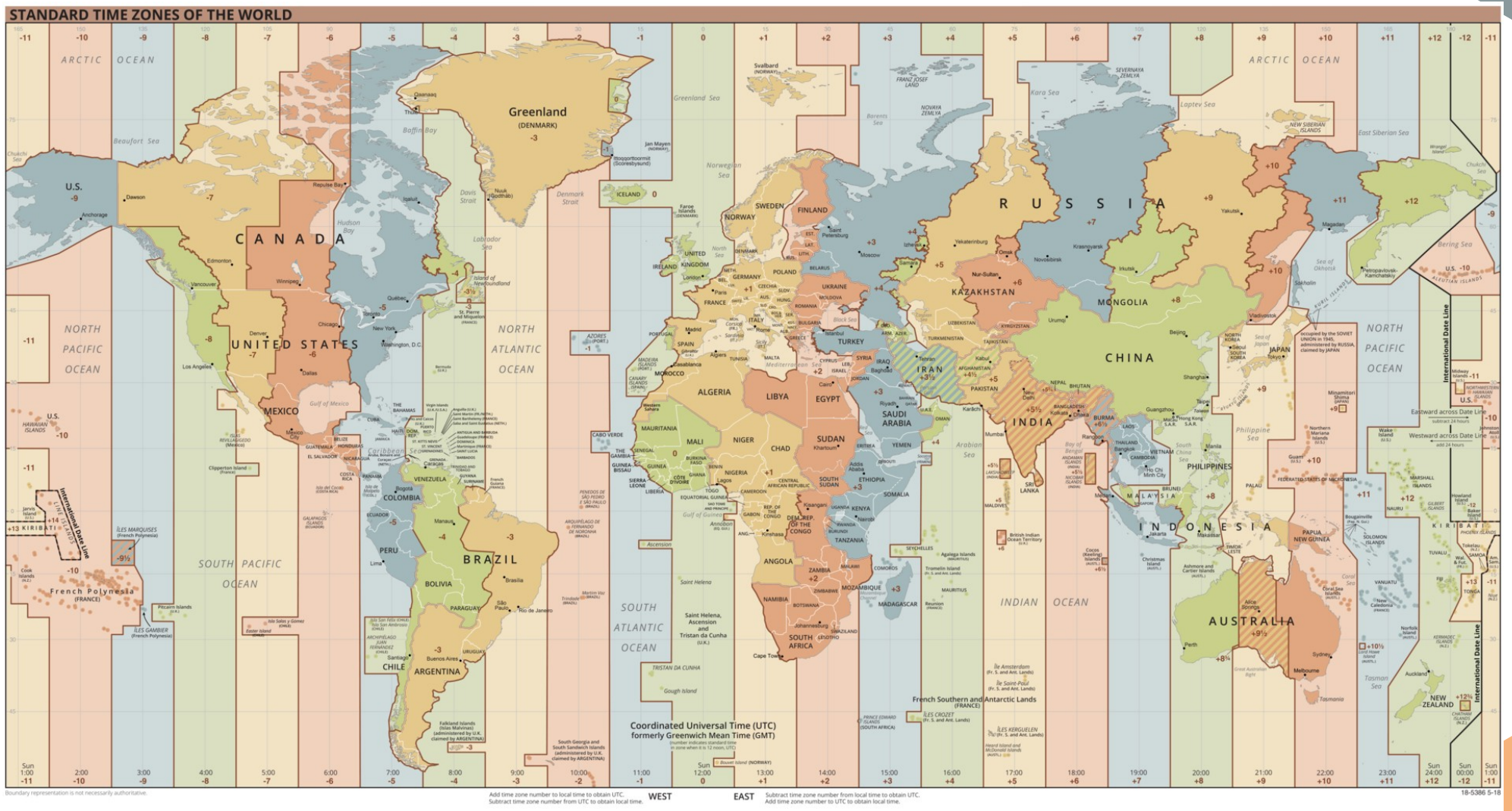
- В Китае и Японии утро наступает еще раньше
- А в Америках еще ночь. Там -03 (Бразилия), -04, -05 и т. д. -01, -02 — таймзоны каких-то мутных островов в Атлантическом океане, в реальной жизни такое встречается редко и если вдруг появится — следует обратить внимание, это может быть ошибкой

Лондонское время и UTC — не одно и то же! В Лондоне есть зимнее и летнее время. (Хотя сейчас и малоактуально)

- И не только в Лондоне

Постгрес знает обо всех этих особенностях при преобразовании. Обновления учитывают возможные изменения

Мировые таймзоны



Кто виноват и что делать?

Виновата планета — на каждой долготе свое время, свой часовой пояс

Что делать?

- Использовать единое время (UTC)

Как?

- Самый прямолинейный подход — плюнуть на все и хранить в виде unixtime в int/bigint
- Это работает, и иногда неплохо
- Недостатки — легко прибавить 3600 секунд, но прибавить месяц и десять рабочих дней уже не так просто

Использовать типы данных СУБД — timestamp/timestamptz и др.

Что вообще бывает

Типы данных

- date — н., дата рождения
- time — время начала, «каждый день в 14:00…»
  - time with timezone — довольно странное изобретение
- timestamp, timestamp without timezone — время события «вообще»
- timestamptz, timestamp with timezone — время события, приведенное к UTC
- interval — временной промежуток между событиями

timestamp и timestamptz

- timestamp without time zone
  - Это просто дата и время с точностью до микросекунды
  - 8 байт
- Timestamp with time zone
  - Это просто дата и время с точностью до микросекунды **в UTC**
    - Т.е. это тот самый unixtime и есть, extract(epoch from ts), to_timestamp(unixtime)
  - те же 8 байт
  - При сохранении/получении данных преобразуется в соответствии с таймзоной сессии
- При преобразовании литерала в timestamp информация о таймзоне может быть отброшена

timestamp ИЛИ timestamptz?

- Разница минимальная:
  - Использовать ли таймзону сессии при сохранении/получении timestamp?
- Что делать, если нет возможности контролировать таймзону сессии?
  - timestamp AT TIME ZONE/timezone(tz,ts)
- Лучше всегда указывать таймзону явно
  - `make_timestamptz ( year int, month int, day int, hour int, min int, sec double precision [, timezone text ] )`

Виды таймзон

- Имена
  - Europe/Moscow, America/Chicago, Africa/Cairo и т.д.
- Сокращения имен
  - MSK, PST и т.д.
- Смещение относительно UTC
  - +03,-05 и т.д.
- POSIX
  - STD offset [DST [dstoffset] [, rule]]
    - CET-1CEST,M3.5.0,M10.5.0/3 — Париж
  - Описано в документации

timestamp ИЛИ timestamptz?

Если внешний источник данных — timestamp

- Он может отдавать данные в своей таймзоне, нередко загадочной, так что лучше не придумывать. Что отдали — то и используем
- Если необходимо согласовывать с собственными данными - timestamptz

Если свой источник данных — тоже timestamptz

Если есть дата-время с неизвестной таймзоной и это надо как-то согласовывать со своими данными — можно считать, что это просто мусор

Проблемы с timestamptz

Еще раз: главное отличие timestamp от timestamptz - при передаче на клиента и обратно происходит преобразование в/из UTCв соответствии с таймзоной сессии.

```
$psql -c "set time zone 'UTC';select now()"

              now
-------------------------------
 2022-02-01 10:53:51.149318+00
(1 row)

$psql -c "select now()"
              now
-------------------------------
 2022-02-01 13:55:25.670707+03
(1 row)
```

AT TIME ZONE

`timestamp without time zone at time zone` → `timestamp with time zone`
`timestamp with time zone at time zone` → `timestamp without time zone`
`now()::timestamp` → просто отбрасывает таймзону

```
work=# select now(), now()::timestamp, now() at time zone 'UTC';
              now               |            now             |          timezone
--------------------------------+----------------------------+----------------------------
 2022-02-08 18:54:43.350146+03 | 2022-02-08 18:54:43.350146 | 2022-02-08 15:54:43.350146
(1 row)
```

А еще установки time zone сессии...

- И тут разработчик ощущает, что у него ум заходит за разум

И еще чуть-чуть

Есть некоторые… особенности

время начала транзакции (это фича!)

- `CURRENT_DATE`, `CURRENT_TIME`, `CURRENT_TIMESTAMP`, `transaction_timestamp()` — timestamptz
- `LOCALTIME`, `LOCALTIMESTAMP` — timestamp
- now() — timestamptz

`clock_timestamp()` — timestamptz, истинное время

`statement_timestamp()` - timestamptz, время начала выполнения выполняемого запроса

Не только timestamp-ы

DATE

- Просто дата, без таймзоны. Год-месяц-день, 2022-02-02 и т. п. 4 байта
- make_date(year int,month int,day int)

TIME

- Просто время. С таймзоной (12 байт) или без (8 байт)

INTERVAL

- Interval ’2 hours’
- ’2 hours’::interval
- make_interval(secs:=,minutes:=,hours:=,days:=,months:=,years:=)
  - make_interval(secs:=3600)
- 16 байт
- Имеет знак

Операции

Temporal value ± interval

- select now()+make_interval(months:=1) → timestamp
- select now()+make_interval(days:=10)-now() → interval

Прекрасно описано в документации

Java

`timestamp → java.time.LocalDateTime`

`timestamptz → java.time.OffsetDateTime` (HE ZonedDateTime!)

```
var i = (org.postgresql.util.PGInterval)
rs.getObject("interval_column");
```

Postgres Professional

http://postgrespro.ru/

+7 495 150 06 91

info@postgrespro.ru

postgrespro.ru