Introduction to date and time data types

TIME SERIES ANALYSIS IN POSTGRESQL

Jasmin Ludolf
Content Developer, DataCamp

datacamp

Date and time data types

- **Date**, **Time**, **DateTime** data types

In **PostgreSQL**:

- `DATE` : date data only
- `TIME` : time information without time zone
- `TIMESTAMP` : combines date and time without time zone
- `TIMESTAMPTZ` : `TIMESTAMP` with time zones
- `INTERVAL` : time between two points

Date and time values

- `DATE` : YYYY-MM-DD (ISO 8601 international standard)
- `TIME` : HH:MM:SS (seconds can be decimals, 12 or 24 hour clock)
- `TIMESTAMP` : YYYY-MM-DD HH:MM:SS
- `TIMESTAMPTZ` : YYYY-MM-DD HH:MM:SS+/-HH (+01:00 or CET)
- `INTERVAL` : one input example is 1 02:30:04
  - Interval of 1 day, 2 hours, 30 minutes, 4 seconds

Date and time in a table

```
CREATE TABLE timetable (
    date_info DATE,
    time_info TIMESTAMP,
    time_with_zone TIMESTAMPTZ,
    interval_length INTERVAL);
```

```
INSERT INTO timetable(
    date_info,
    time_info,
    time_with_zone,
    interval_length)
VALUES (
    'January 23 2013',
    '2023-01-20 18:00:00',
    '2023-01-20 18:00:00 EST',
    '1 02:03:04');
```

Date and time in a table

```
SELECT *
FROM timetable;
```

```
| date_info  | time_info           | time_with_zone            | interval_length |
| ---------- | ------------------- | ------------------------- | --------------- |
| 2023-01-20 | 2023-01-20 18:00:00 | 2023-01-20 00:00:00+01:00 | 1 day, 2:03:04  |
```

Partial or incorrect data

```
INSERT INTO timetable(time_info)
VALUES ('2020-02-20 12:00:00');
```

```
SELECT *
FROM timetable;
```

```
| date_info  | time_info           | time_with_zone             | interval_length |
| ---------- | ------------------- | -------------------------- | --------------- |
| 2023-01-20 | 2023-01-20 18:00:00 | 2023-01-20 00:00:00+01:00 | 1 day, 2:03:04  |
| null.      | 2020-02-20 12:00:00 | null                       | null            |
```

Partial or incorrect data

```
INSERT INTO timetable(time_info)
VALUES ('2020-02-20 02:00:00 EST');
```

```
SELECT *
FROM timetable;
```

```
| date_info  | time_info           | time_with_zone             | interval_length |
| ---------- | ------------------- | -------------------------- | --------------- |
| 2023-01-20 | 2023-01-20 18:00:00 | 2023-01-20 00:00:00+01:00 | 1 day, 2:03:04  |
| null.      | 2020-02-20 02:00:00 | null                       | null            |
```

Unix time

- **Unix time** : seconds since the Unix epoch
- **Unix epoch** : January 1 1970 00:00:00, UTC

```
|unix_time |
|----------|
|1483444800|
```

That's January 3 2017!

Let's practice!

TIME SERIES ANALYSIS IN POSTGRESQL

Working with time zone information

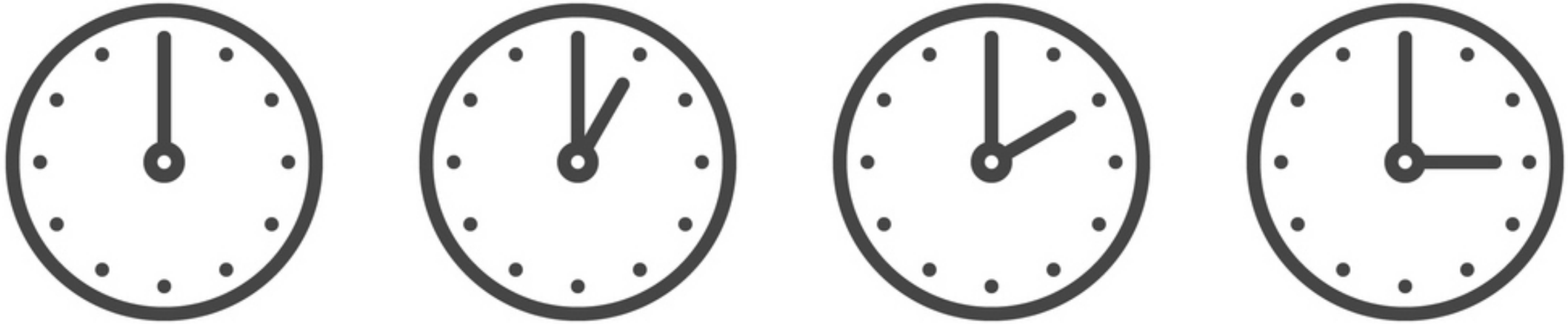
TIME SERIES ANALYSIS IN POSTGRESQL

Jasmin Ludolf
Content Developer, DataCamp

datacamp

Time zone data

- `TIMESTAMP` : date and time information without time zone
  - Fine when using a single time zone
  - All times are in UTC
- `TIMESTAMPTZ` : date and time with time zones



Time zone names

```
SELECT * FROM pg_timezone_names;
```

```
| name              |abbrev|utc_offset|is_dst|
|-------------------|------|----------|------|
|Africa/Abidjan     |GMT   |  00:00:00|false |
|Africa/Accra       |GMT   |  00:00:00|false |
|Africa/Addis_Ababa|EAT   |   3:00:00|false |
...
```

- `is_dst` : whether the location is observing daylight savings

Show the time zone

```
SHOW TIMEZONE;
```

```
|TimeZone       |
|---------------|
|Europe/Brussels|
```

Time zone data in tables

- `datetimes` table has the following two fields:
  - `datetime` : `TIMESTAMP`
  - `datetimetz` : `TIMESTAMPTZ`

```
INSERT INTO datetimes (datetime, datetimetz)
VALUES('2023-01-03 12:00:00','2023-01-03 12:00:00');
```

```
|datetime           |datetimetz                |
|-------------------|--------------------------|
|2023-01-03 12:00:00|2023-01-03 12:00:00+01:00|
```

Verify the data type

- `pg_typeof(field_name)` : check the data type of the field

```
SELECT
    pg_typeof(datetime) AS "type of(datetime)",
    pg_typeof(datetimetz) AS "type of(datetimetz)"
FROM datetimes;
```

```
|type of(datetime)           |type of(datetimetz)     |
|----------------------------|------------------------|
|timestamp without time zone|timestamp with time zone|
```

Using time zone information

```
INSERT INTO datetimes (datetime, datetimetz)
VALUES('2023-01-03 12:00:00+00','2023-01-03 12:00:00+00');
```

```
SELECT * FROM datetimes;
```

```
|datetime           |datetimetz                |
|-------------------|--------------------------|
|2023-01-03 12:00:00|2023-01-03 12:00:00+01:00|
|2023-01-03 12:00:00|2023-01-03 13:00:00+01:00|
```

Adding time zone information

- `AT TIME ZONE` : Add, change, or remove time zone information
  - Converting the type from `TIMESTAMP` to `TIMESTAMPTZ` and vice versa

Adding a time zone:

```
SELECT
    TIMESTAMP '2020-12-31 23:59:59' AT TIME ZONE 'Europe/London' AS added;
```

```
|added                      |
|---------------------------|
|2021-01-01 00:59:59+01:00|
```

This query: interprets the timestamp, creates `TIMESTAMPTZ` , displays the default

Changing and removing time zones

```
SELECT
    '2020-12-31 23:59:59+00'::TIMESTAMPTZ AT TIME ZONE 'Europe/Paris'
    AS shifted;
```

```
|shifted             |
|--------------------|
|2021-01-01 00:59:59|
```

This query: shifts the timestamp to the desired time zone, removes the time zone designation

Let's practice!

TIME SERIES ANALYSIS IN POSTGRESQL

Converting between date, time, and text

TIME SERIES ANALYSIS IN POSTGRESQL

Jasmin Ludolf
Content Developer, DataCamp

datacamp

Format strings

- YYYY : four digit year
- mm : two digit numerical month
- DD : two digit numerical day
- HH24 : two digit hour for a 24 hour clock
- HH12 : two digit hour for a 12 hour clock
- HH : two digit hour of day using 12 hour clock
- MI : minute
- SS : second

Convert strings into dates

- `TO_DATE()` : converts strings into dates
- `TO_DATE('text string', 'format string')`
- The wrong format string may result in incorrect data or an error

```
SELECT
    TO_DATE('2023-01-15', 'YYYY-MM-DD') AS date_1,
    TO_DATE('20231501', 'YYYYDDMM') AS date_2,
    TO_DATE('Jan 15, 2023', 'Mon DD, YYYY') AS date_3;
```

```
|date_1    |date_2    |date_3    |
|----------|----------|----------|
|2023-01-15|2023-01-15|2023-01-15|
```

Convert strings into dates

- `CAST()` : converts one data type into another
- Use `CAST()` for any data type
- `CAST('string' AS data_type)`

```
SELECT
    CAST('2023-01-15' AS DATE) AS date_1,
    CAST('20230115' AS DATE) AS date_2,
    CAST('Jan 15, 2023' AS DATE) AS date_3;
```

```
|date_1    |date_2    |date_3    |
|----------|----------|----------|
|2023-01-15|2023-01-15|2023-01-15|
```

The cast operator

- `::` the cast operator
- Works the same way as `CAST()`
- Only in PostgreSQL

```
SELECT
    '2023-01-15'::DATE AS date_1,
    '20230115'::DATE AS date_2,
    'Jan 15, 2023'::DATE AS date_3;
```

```
|date_1    |date_2    |date_3    |
|----------|----------|----------|
|2023-01-15|2023-01-15|2023-01-15|
```

Convert into dates and times

- `TO_TIMESTAMP()` : converts strings into dates and times
- `TO_TIMESTAMP('string', 'format string')`
- Includes dates, times, and time zones

```
SELECT
    TO_TIMESTAMP('Jan 15, 2023 14:02:01', 'Mon DD, YYYY HH24:MI:SS') AS date_time;
```

```
|date_time                 |
|--------------------------|
|2023-01-15 14:02:01+01:00|
```

Converting unix time

- Unix time can be converted using `TO_TIMESTAMP()`

```
SELECT
     TO_TIMESTAMP(1483444800) AT TIME ZONE 'UTC' as datetime;
```

```
|datetime            |
|--------------------|
|2017-01-03 12:00:00|
```

Extracting unix time

- `EXTRACT()` : retrieves values
- With Unix time, it calculates the time elapsed from the epoch to the time stamp.

```
SELECT
    EXTRACT(epoch FROM TIMESTAMP '2017-01-03 12:00:00') AS unix_time;
```

```
|unix_time |
|----------|
|1483444800|
```

Converting fields

Uniform fields

-> use `TO_DATE` or `TO_TIMESTAMP`

date
Jan 15, 2023
Jan 16, 2023
Jan 17, 2023

```
SELECT TO_DATE(date, 'Mon DD, YYYY');
```

```
|date      |
|----------|
|2023-01-15|
|2023-01-16|
|2023-01-17|
```

Converting fields

Varied fields

-> use `CAST()` or `::`

date
2023-01-15
20230116
Jan 17, 2023

```
SELECT CAST(date AS DATE);
```

```
|date      |
|----------|
|2023-01-15|
|2023-01-16|
|2023-01-17|
```

Convert date or time into text

- `TO_CHAR()` : converts datetime data into text
- `TO_CHAR(date or time, 'format string')`

```
SELECT
    timestamp_field,
    TO_CHAR(timestamp_field, 'YYYY-mm-DD HH12:MI:SS') AS timestamp_text
FROM timetable;
```

```
|timestamp_field    |timestamp_text     |
|-------------------|-------------------|
|2015-07-14 11:49:00|2015-07-14 11:49:00|
|2020-10-18 20:53:50|2020-10-18 08:53:50|
|2020-12-31 12:59:59|2020-12-31 12:59:59|
```

Verify the data types

```
SELECT
    pg_typeof(timestamp_field) AS "type of(timestamp_field)",
    pg_typeof(TO_CHAR(timestamp_field, 'YYYY-mm-DD HH12:MI:SS'))
    AS "type of(timestamp_text)"
FROM timetable;
```

```
|type of(timestamp_field)   |type of(timestamp_text)|
|---------------------------|-----------------------|
|timestamp without time zone|text                   |
```

Different delimiters

```
SELECT
    TO_CHAR(timestamp_field, 'YYYY/mm/DD') AS slashes,
    TO_CHAR(timestamp_field, 'YYYY.mm.DD') AS dots,
    TO_CHAR(timestamp_field, '"Year": YYYY "Month": mm "Day": DD') AS labels
FROM timetable;
```

```
|slashes   |dots      |labels                          |
|----------|----------|--------------------------------|
|2015/07/14|2015.07.14|Year: 2015 Month: 07 Day: 14|
|2020/10/18|2020.10.18|Year: 2020 Month: 10 Day: 18|
|2020/12/31|2020.12.31|Year: 2020 Month: 12 Day: 31|
|2021/01/01|2021.01.01|Year: 2021 Month: 01 Day: 01|
```

Non-numeric text

```
SELECT
    TO_CHAR(timestamp_field, 'Dy, Mon DD, YYYY') AS date
FROM timetable;
```

```
|date              |
|------------------|
|Tue, Jul 14, 2015|
|Sun, Oct 18, 2020|
|Thu, Dec 31, 2020|
|Fri, Jan 01, 2021|
```

Custom formats

```
SELECT TO_CHAR(time_field, 'HH24:MI') AS "HH:MM"
FROM timetable;
```

```
|HH:MM|
|-----|
|11:49|
|20:53|
|12:59|
|00:01|
```

AM and PM

```
SELECT
    TO_CHAR(time_field, 'HH12:MI:SS AM') AS "AM",
    TO_CHAR(time_field, 'HH12:MI:SS pm') AS pm
FROM timetable;
```

```
|AM          |pm          |
|------------|------------|
|11:49:00 AM|11:49:00 am|
|08:53:50 PM|08:53:50 pm|
|12:59:59 PM|12:59:59 pm|
|12:01:02 AM|12:01:02 am|
```

MDY format

```
SELECT
    TO_CHAR(date_field, 'MM/DD/YYYY') AS "MDY Numeric",
    TO_CHAR(date_field, 'Mon DD, YYYY') AS "MDY Expanded",
    TO_CHAR(timetz_field, 'HH24:MI:SS TZ') AS upper_case,
    TO_CHAR(timetz_field, 'HH24:MI:SS tz') AS lower_case,
    TO_CHAR(timetz_field, 'HH24:MI:SS OF') AS utc_offset
FROM timetable;
```

```
|MDY Numeric|MDY Expanded|upper_case  |lower_case  |utc_offset  |
|-----------|------------|------------|------------|------------|
|12/31/2020 |Dec 31, 2020|18:49:00 UTC|18:49:00 utc|18:49:00 +00|
```

Let's practice!

TIME SERIES ANALYSIS IN POSTGRESQL