Time Travelling PostgreSQL

PGDay/MED 2024-12-05

Stefanie Janine Stölting

About Me

Owner of Tallinn/Estonia based ProOpenSource OÜ

Website: www.ProOpenSource.eu

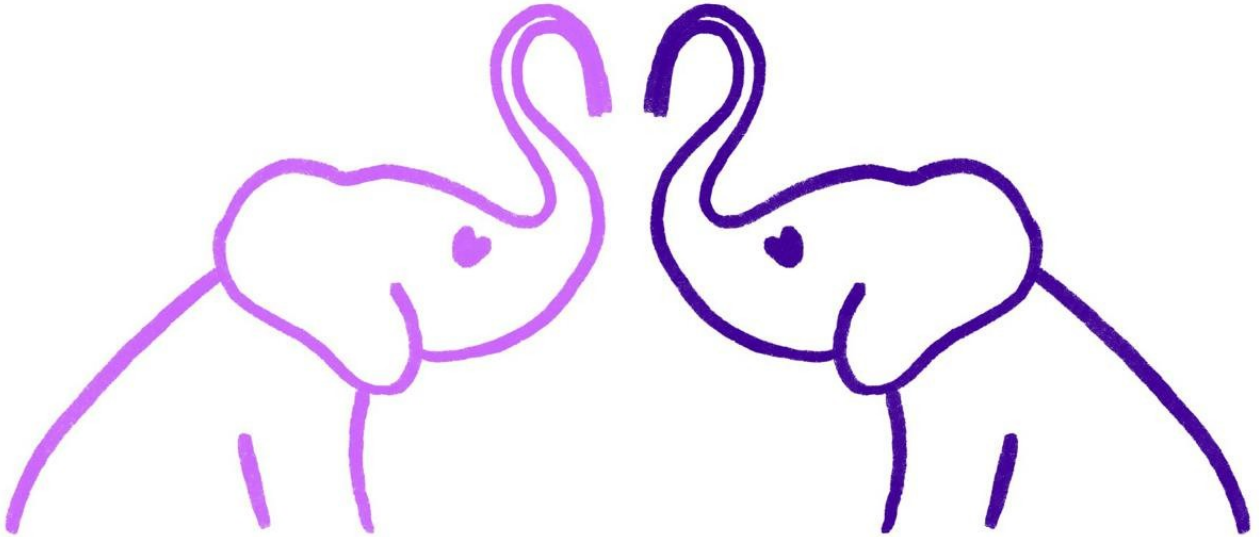
Blog: www.ProOpenSource.it

Mastodon: https://digitalcourage.social/@sjstoelting

Telegram: https://t.me/postgreschat

About Me

Member of the PGEU Diversity Committee



What is Time Travelling

Photo by Lukas Tennie on Unsplash

Definition

Being able to return data as it has been at a certain point in time.

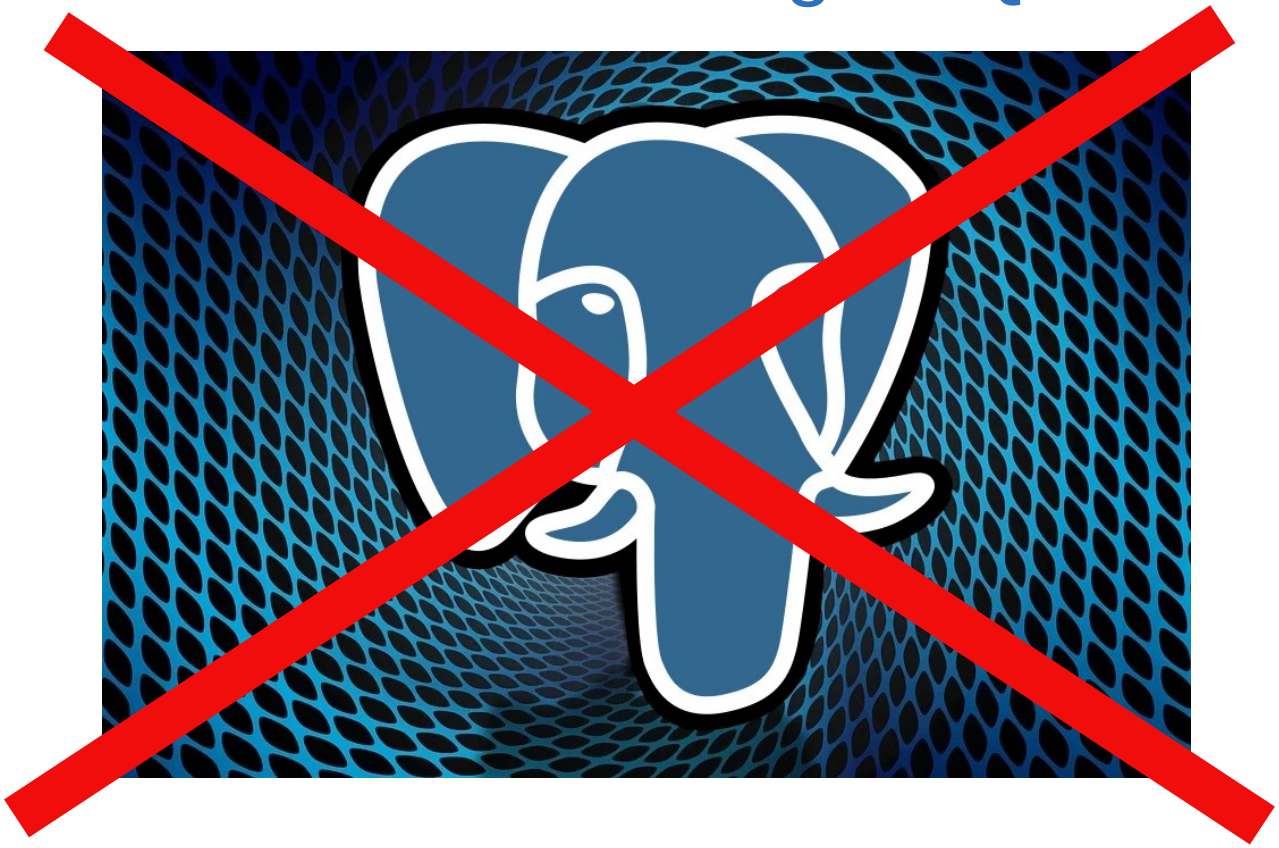
That can obviously only be data saved in the past.

And we need dates and times stored with data.

Photo by Tim Green

History

PostgreSQL supported time travel as it was implemented in its core up to version 6.2. It has been removed in PostgreSQL 6.3.



Other Ideas

Hans-Jürgen Schönig from CYBERTEC has written about implementing time travelling in his blog post in 2019 about AS OF-queries.

I have tested them, but they seem not to work with PostgreSQL 17.

The creation of the trigger fails.

Other Ideas

The problem here is:

```
CREATE FUNCTION version_trigger() RETURNS trigger AS
$
…
…
END;
$ LANGUAGE plpgsql;
```

The solution with PostgreSQL 17 is:

```
CREATE FUNCTION version_trigger()
  RETURNS TRIGGER
  LANGUAGE PLPGSQL
AS
$$
…
…
END;
$$
```

What is covered?

- Table partitioning by range
- Primary key with partitioned tables
- Updating records with the possibility to access the previous records as they have been at a certain point in time
- Deleting records with the possibility to access the previous records as they have been at a certain point in time

Let’s start

Photo by Wim van 't Einde on Unsplash

Primary Key - Partitioning

PostgreSQL does not support unique indexes or primary keys over partitioned tables.

But there is a solution to solve the primary key problem without any extensions in PostgreSQL.

Photo by Ylanite Koppens on StockSnap

Primary Key - Table

The solution is somewhat easy.

In addition to the partitioned table we also need a table, that will hold the primary key.

The partitioned table needs a foreign key referencing the primary key table.

Preparation

I stated we do not need an extension.

Well, we need one to get speedy access to the data: btree_gist.

btree_gist is an index that is part of the contrib package and available on all supported platforms.

```
CREATE EXTENSION IF NOT EXISTS btree_gist;
```

Time Travel Tables


```
CREATE TABLE timetravel_pk (
        timetravelid BIGINT GENERATED ALWAYS AS IDENTITY,
        created timestamp with time zone NOT NULL,
        CONSTRAINT timetravle_pk_pk PRIMARY KEY (timetravelid)
);

CREATE TABLE timetravel (
        timetravelid BIGINT NOT NULL,
        changed TSTZRANGE NOT NULL DEFAULT tstzrange(clock_timestamp(), 'INFINITY', '[)'),
        data_text TEXT,
        data_json JSONB,
        deleted BOOLEAN NOT NULL DEFAULT FALSE,
        CONSTRAINT timetravelid_fk FOREIGN KEY (timetravelid)
              REFERENCES timetravel_pk(timetravelid)
) PARTITION BY RANGE (lower(changed));
```

Indexes


```
CREATE INDEX timetravel_changed_idx
  ON timetravel
  USING gist
  (changed)
;

CREATE INDEX timetravel_timetravelid_idx
  ON timetravel
  USING btree
  (timetravelid)
;

CREATE INDEX timetravel_not_deleted_idx
  ON timetravel
  USING btree
  (deleted)
  WHERE NOT deleted
;
```

Partition Management Table


```sql
CREATE TABLE timetravel_part_vals (
  part_year SMALLINT NOT NULL,
  start_value TIMESTAMP WITH TIME ZONE NOT NULL,
  end_value TIMESTAMP WITH TIME ZONE NOT NULL,
  CONSTRAINT timetravel_part_vals_pk PRIMARY KEY (part_year)
);
```

Partition Management Table

This table is needed to handle partition information. It will later be used to check, if a partition has to be created.

A function will use that table to create partitions.

The function will be executed manually.

In real life I would use a cron job, for example with pg_cron.

Show Time

Showing slides with SQL is a bit boring.

Link List

Slide and sources are available on GitLab and GitHub:

https://gitlab.com/sjstoelting/talks/

https://github.com/sjstoelting/talks/

Time Traveling PostgreSQL