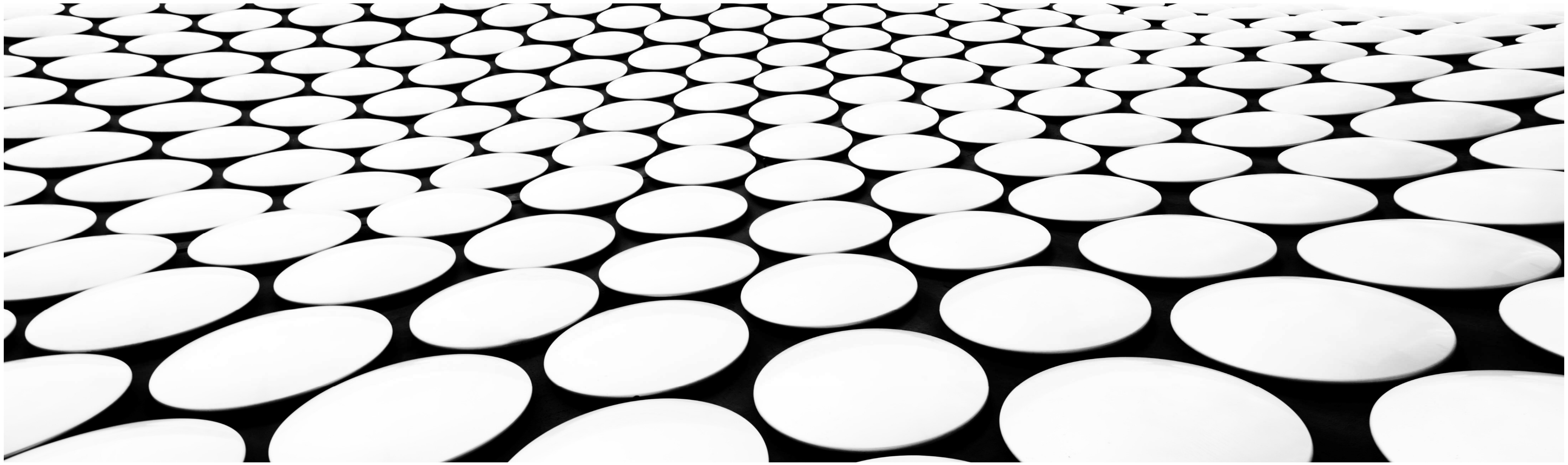
PARTITION ON THE FLY

NINA BELYAVSKAYA



WHO I AM

My name is Nina Belyavskaya

I worked with PostgreSQL
as a fullstack developer
and as DBA.

PostgreSQL PARTITIONING

Table partitioning is a great feature in PostgreSQL.

- Speed up query execution
- Prevent table bloating
- Easy archive old data

DECLARATIVE PARTITIONING

Since version 10 we have declarative partitioning,

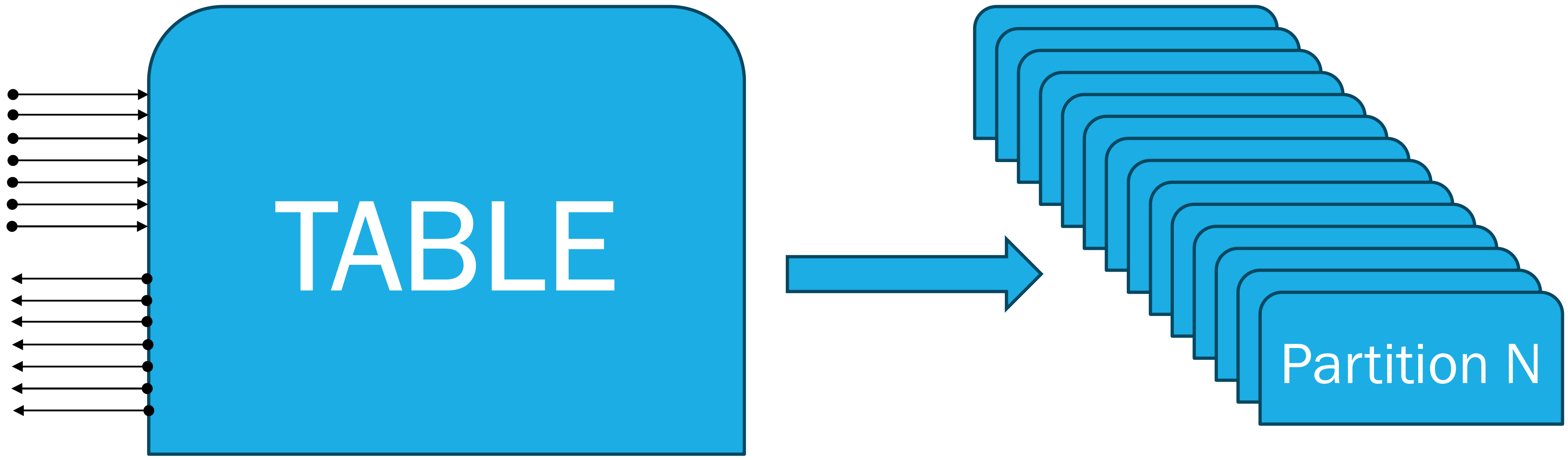
which is a good way to create and arrange partitioned tables.

```
CREATE TABLE measurement (
     city_id int not null,
     logdate date not null,
     peaktemp int,
     unitsales int )
PARTITION BY RANGE (logdate);

CREATE TABLE measurement_y2024m02 PARTITION OF measurement
    FOR VALUES FROM ('2024-02-01') TO ('2024-03-01');

CREATE TABLE measurement_y2024m03 PARTITION OF measurement
    FOR VALUES FROM ('2024-03-01') TO ('2024-04-01');
```

BUT… IF HUGE TABLE ALREADY EXISTS?



EXAMPLE

```
CREATE TABLE events (
    event_id serial PRIMARY KEY,
    event_place_id integer NOT NULL,
    created_at timestamp,
    comment text,
    username name NOT NULL
);

CREATE TABLE places (
    place_id integer PRIMARY KEY,
    place_name text
);
```

```
ALTER TABLE events
    ADD CONSTRAINT events_place_fk
    FOREIGN KEY (event_place_id)
    REFERENCES places (place_id);

CREATE INDEX events_place_idx
    ON events(event_place_id);

CREATE FUNCTION events_username() RETURNS trigger
LANGUAGE plpgsql
    AS $$
            BEGIN
                NEW.username := current_user;
                RETURN NEW;
            END;
    $$;
CREATE TRIGGER events_username BEFORE INSERT ON events
    FOR EACH ROW EXECUTE FUNCTION events_username();
```

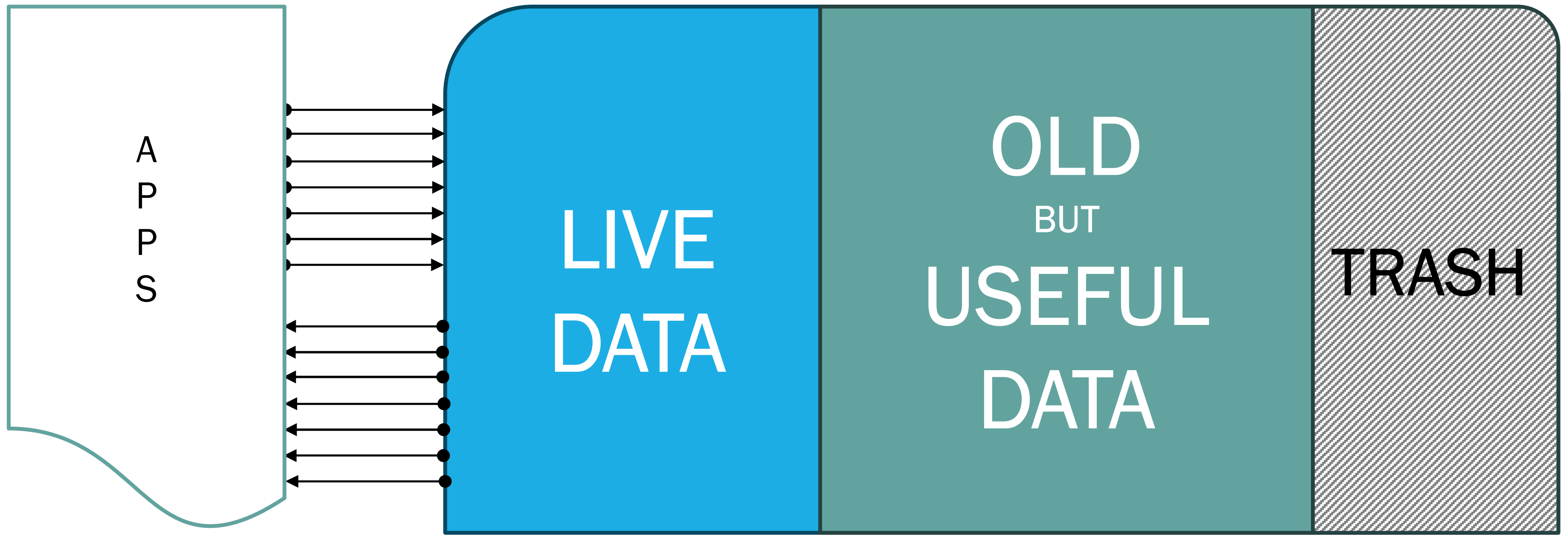
```
INSERT INTO places (place_id, place_name)
(SELECT id, format('Place %s', id)
 FROM generate_series(1, 1000) AS id);

INSERT INTO events (event_place_id)
( SELECT floor(random()*1000+1)
  FROM generate_series(1,1000000) );

INSERT INTO events (event_place_id, created_at)
( SELECT floor(random()*1000+1), created_at
  FROM generate_series('2023-01-01',now(),'1 min') AS created_at );
```

IMPORTANT DECISIONS

- What data do we need continuous, non-downtime access to?
- What data do we need to store, e.g. for analytical purposes?



ASSUME

We want to:

- access to this year data (`created_at >= '2024-01-01'`)
- keep all data (including nulls)
- There is no FK that references `events`

PREPARATION – PARTITIONED TABLE

```
CREATE TABLE events_new (
   LIKE events INCLUDING DEFAULTS  INCLUDING CONSTRAINTS )
    PARTITION BY RANGE (created_at);

~~ALTER TABLE events_new ADD PRIMARY KEY(event_id, created_at);~~
CREATE UNIQUE INDEX ON events_new(event_id, created_at);
```

PREPARATION – PARTITIONED TABLE

```
CREATE INDEX events_new_place_idx ON events_new(event_place_id);

ALTER TABLE events_new
    ADD CONSTRAINT events_new_place_fk FOREIGN KEY (event_place_id)
    REFERENCES places (place_id);

CREATE TRIGGER events_username BEFORE INSERT ON events_new
    FOR EACH ROW EXECUTE FUNCTION events_username();

ALTER TABLE events_new ALTER COLUMN created_at SET DEFAULT now();
```

PREPARATION – PARTITIONED TABLE

```
\d events_new
                                     Partitioned table "public.events_new"
    Column     |            Type             | Collation | Nullable |                  Default
---------------+-----------------------------+-----------+----------+-------------------------------------------
 event_id      | integer                     |           | not null | nextval('events_event_id_seq'::regclass)
 event_place_id | integer                     |           | not null |
 created_at    | timestamp without time zone |           |          | now()
 comment       | text                        |           |          |
 username      | name                        |           | not null |
Partition key: RANGE (created_at)
Indexes:
    "events_new_event_id_created_at_idx" UNIQUE, btree (event_id, created_at)
    "events_new_place_idx" btree (event_place_id)
Foreign-key constraints:
    "events_new_place_fk" FOREIGN KEY (event_place_id) REFERENCES places(place_id)
Triggers:
    events_username BEFORE INSERT ON events_new FOR EACH ROW EXECUTE FUNCTION events_username()
Number of partitions: 0
```

PREPARATION – PARTITIONS FOR NULLS AND OLD VALUES

```
CREATE TABLE events_null ( LIKE events );
ALTER TABLE events_null
    ADD CONSTRAINT events_created_at_null CHECK ( created_at IS NULL );

CREATE TABLE events_old ( LIKE events );
ALTER TABLE events_old
    ADD CONSTRAINT events_created_at_old CHECK ( created_at < '2024-01-01' );
```

PREPARATION – PARTITION FOR ACTUAL VALUES

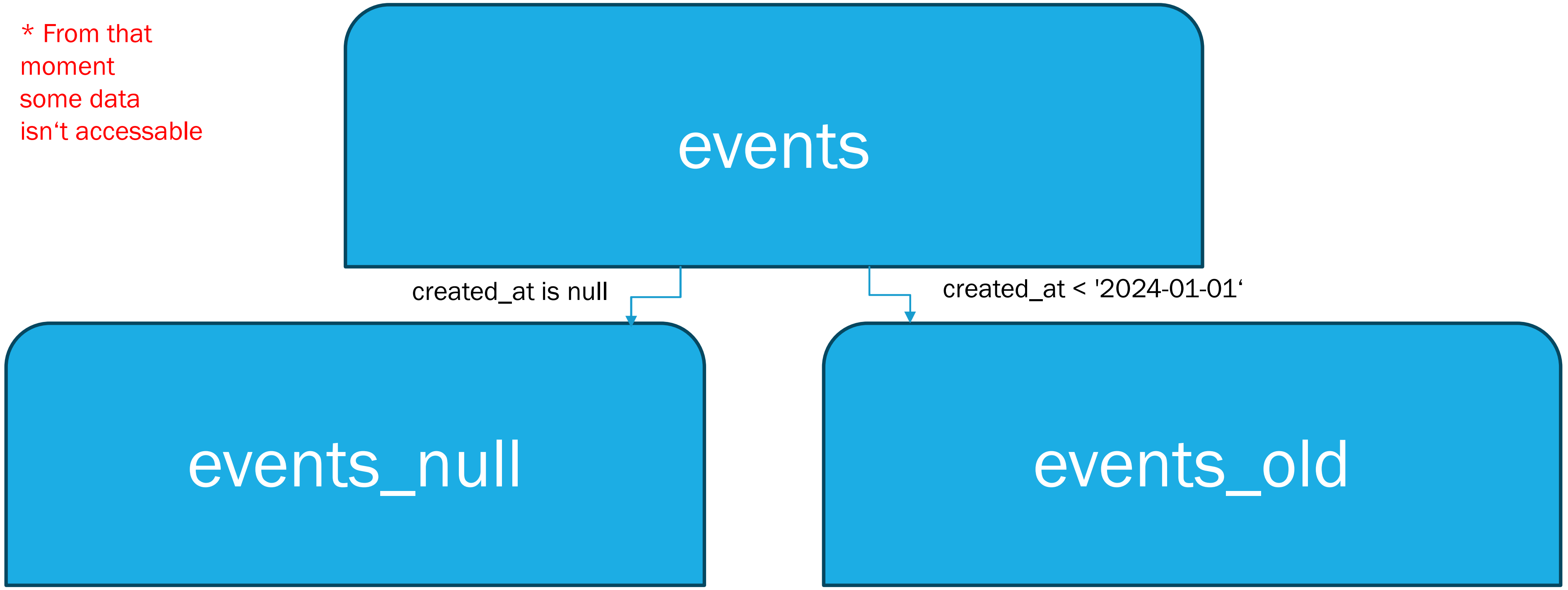
```
ALTER TABLE events ALTER COLUMN created_at SET DEFAULT now();

ALTER TABLE events
    ADD CONSTRAINT events_created_at_not_null
    CHECK ( created_at IS NOT NULL ) NOT VALID;

ALTER TABLE events
    ADD CONSTRAINT events_created_at_2024
    CHECK ( created_at>='2024-01-01' ) NOT VALID;
```

PREPARATION - MOVE OLD DATA TO OTHER TABLES

* From that moment some data isn‘t accessable



ATTACH PARTITIONS

- ALTER TABLE events_new
  ```
  ATTACH PARTITION events_null DEFAULT;
  ```

- ALTER TABLE events_new
  ```
  ATTACH PARTITION events_old
  FOR VALUES FROM (minvalue) TO ('2024-01-01');
  ```

VALIDATE CONSTRAINTS

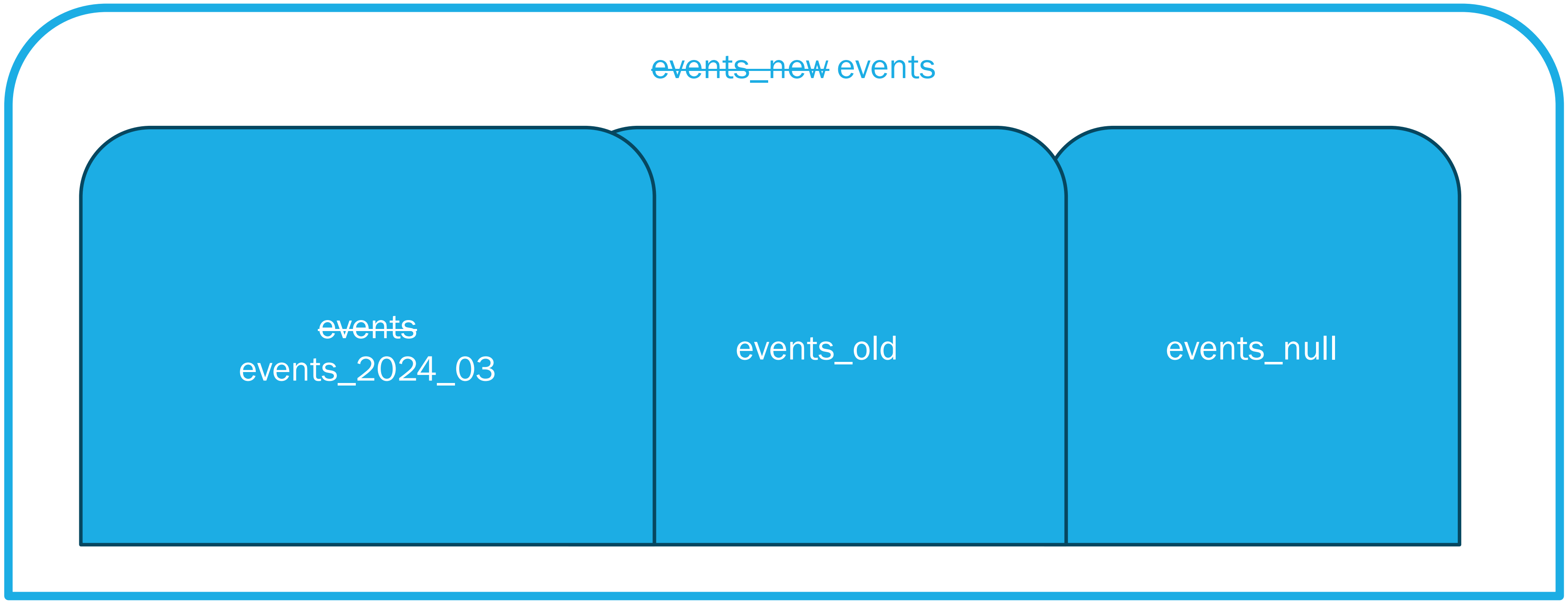
- ALTER TABLE events
  VALIDATE CONSTRAINT events_created_at_not_null;

- ALTER TABLE events
  VALIDATE CONSTRAINT events_created_at_2024;

PARTITIONED TABLE ASSEMBLY

```
BEGIN;
  ALTER TABLE events RENAME TO events_2024_03;
  ALTER TABLE events_new RENAME TO events;
  DROP TRIGGER events_username ON events_2024_03;
  ALTER TABLE events ATTACH PARTITION events_2024_03
       FOR VALUES FROM ('2024-01-01') TO ('2024-04-01‘);
COMMIT;
```

PARTITIONED TABLE ASSEMBLY



```
 Partitioned table "public.events"
    Column      |            Type             | Collation | Nullable |                  Default
----------------+-----------------------------+-----------+----------+-------------------------------------------
 event_id       | integer                     |           | not null | nextval('events_event_id_seq'::regclass)
 event_place_id | integer                     |           | not null |
 created_at     | timestamp without time zone |           |          |
 comment        | text                        |           |          |
 username       | name                        |           | not null |
Partition key: RANGE (created_at)
Indexes:
    "events_new_event_id_created_at_idx" UNIQUE, btree (event_id, created_at)
    "events_new_place_idx" btree (event_place_id)
Foreign-key constraints:
    "events_new_place_fk" FOREIGN KEY (event_place_id) REFERENCES places(place_id)
Triggers:
    events_username BEFORE INSERT ON events FOR EACH ROW EXECUTE FUNCTION events_username()
Partitions: events_2024_03 FOR VALUES FROM ('2024-01-01 00:00:00') TO ('2024-04-01 00:00:00'),
            events_old FOR VALUES FROM (MINVALUE) TO ('2024-01-01 00:00:00'),
            events_null DEFAULT
```

Don‘t forget:

```
\d events_event_id_seq
…
Owned by: public.events_2024_03.event_id
```

```
ALTER SEQUENCE events_event_id_seq
    OWNED BY events.event_id;
```

Q & A

MY CONTACTS

https://www.linkedin.com/in/nina-belyavskaya/

ninaobel@gmail.com