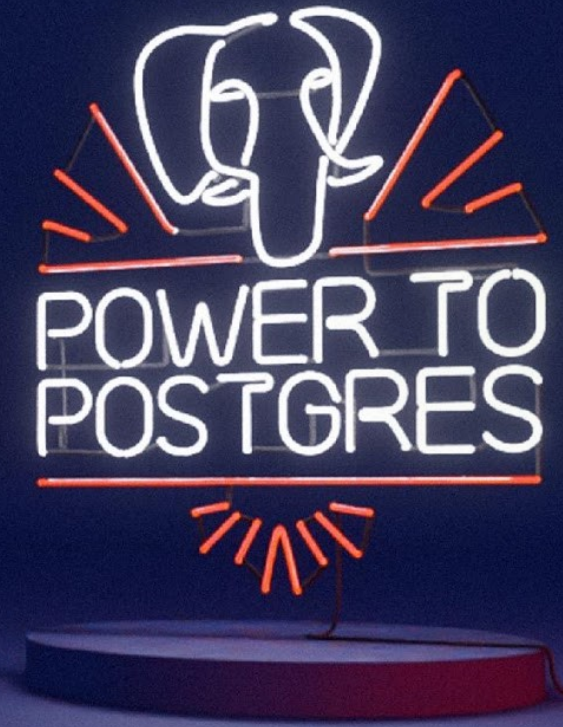
Table Partitioning Transparent but No Magic

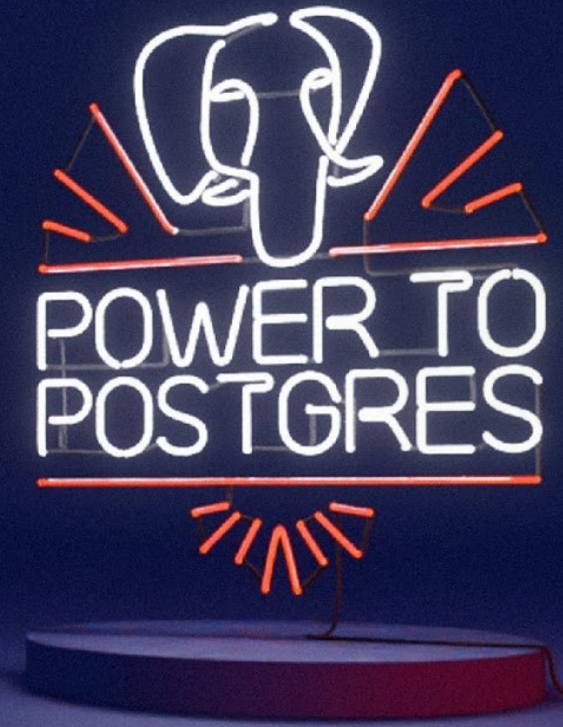


Boriss Mejías
Holistic System Software Engineer
Air Guitar Player

28 October 2022 – PgConf EU - Berlin

EDB

Table Partitioning Transparent but No Magic



Boriss Mejías
Solution Architect
Air Guitar Player

28 October 2022 – PgConf EU - Berlin

EDB @tchorix

It is all about data
a lot of data


Meyers Konversations-Lexikon.
1. A bis Atlantiden.
Meyers Konversations-Lexikon.
2. Atlantis bis Blatthornkäfer.
Meyers Konversations-Lexikon.
3. Blattkäfer bis Chimbote.
Meyers Konversations-
4. bis
Meyers Konversations-Lexikon.
5. Distanzgeschäft bis Faidherbe.
Meyers Konversations-Lexikon.
6. Faidit bis Gehilfe.
Meyers Konversations-Lexikon.
7. Gehirn bis Hainichen.
Meyers Konversations-Lexikon.
8. Hainleite bis Iriartea.
Meyers Konversations-Lexikon.
9. Irideen bis Königsgrün.
Meyers Konversations-Lexikon.
10. Königshofen bis Luzon.
Meyers Konversations-Lexikon.
11. Luzula bis Nathanael.
Meyers Konversations-Lexikon.
12. Nathusius bis Phlegmone.
Meyers Konversations-Lexikon.
13. Phlegon bis Rubinstein.
Meyers Konversations-Lexikon.
14. Rüböl bis Sodawasser.
Meyers Konversations-Lexikon.
15. Sodbrennen bis Uralit.
Meyers Konversations-Lexikon.
16. Uralsk bis Bz.


Español
Aula Internacional
Esselte
Quartier français


PostgreSQL
madhu
PGConfEU 2022
NARVA
Ola
Abu
PGConfEU 2022


Check In Queue
First (given) name:
N – Z
Please have your ticket ready to scan!
Check In Queue
First (given) name:
G – M
Please have your ticket ready to scan!
A – F

A large table does not scale and that’s why we want table partitioning

Brutally Extreme Radio Listening Interface

BERLIN
Brutally Extreme Radio Listening Interface

Online music player collecting stats

Register every time a song is played

- User
- Timestamp
- Song
- Artist
- Metadata

Note: Users opt-in to send stats

Online music player collecting stats

```
CREATE TABLE play_counts (
    user         text
  , played_on    timestamptz
  , song         text
  , artist       text
  , metadata     jsonb

);
```

Online music player collecting stats

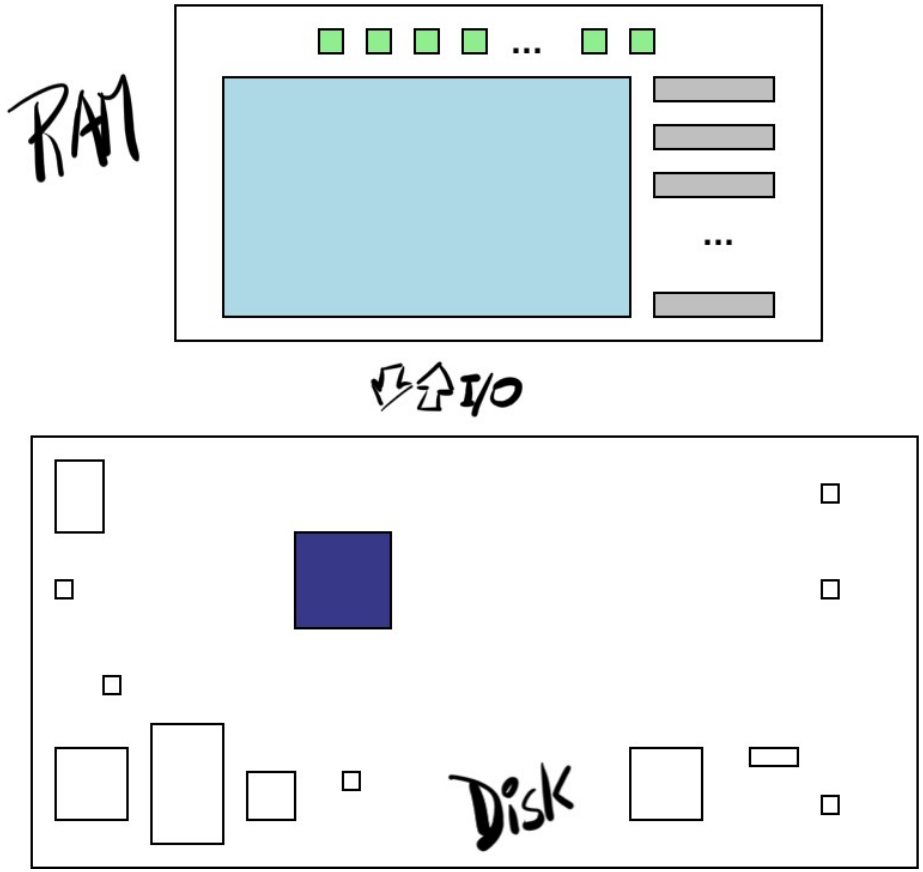
```
CREATE TABLE play_counts (
    user_id     bigint      REFERENCES users(user_id)
  , played_on   timestamptz
  , song_id     bigint      REFERENCES songs(song_id)
  , artist_id   bigint      REFERENCES artists(artist_id)
  , metadata    jsonb

);
```

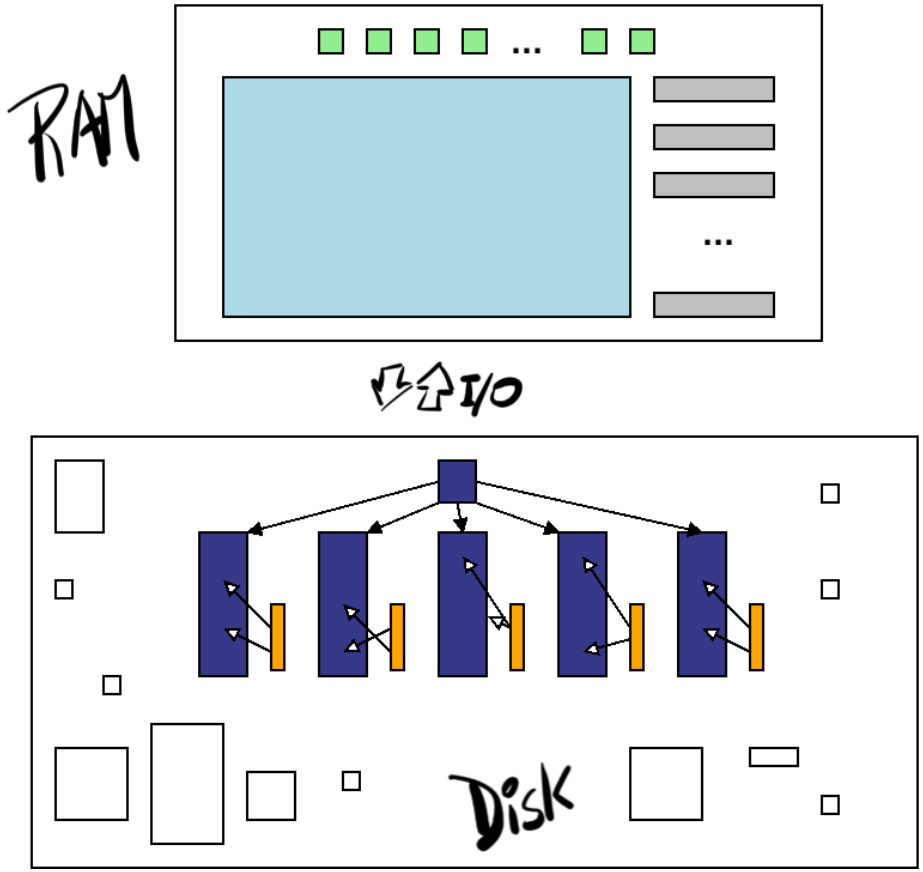
Online music player collecting stats

```
CREATE TABLE play_counts (
    user_id     bigint      REFERENCES users(user_id)
  , played_on   timestamp(0) WITH TIME ZONE
  , song_id     bigint      REFERENCES songs(song_id)
  , artist_id   bigint      REFERENCES artists(artist_id)
  , metadata    jsonb

);
```

Online music player collecting stats

```
CREATE TABLE play_counts (
    user_id     bigint      REFERENCES users(user_id)
  , played_on   timestamp(0) WITH TIME ZONE
  , song_id     bigint      REFERENCES songs(song_id)
  , artist_id   bigint      REFERENCES artists(artist_id)
  , metadata    jsonb
  , PRIMARY KEY (user_id, played_on)
);
```


RAM
I/O
Disk

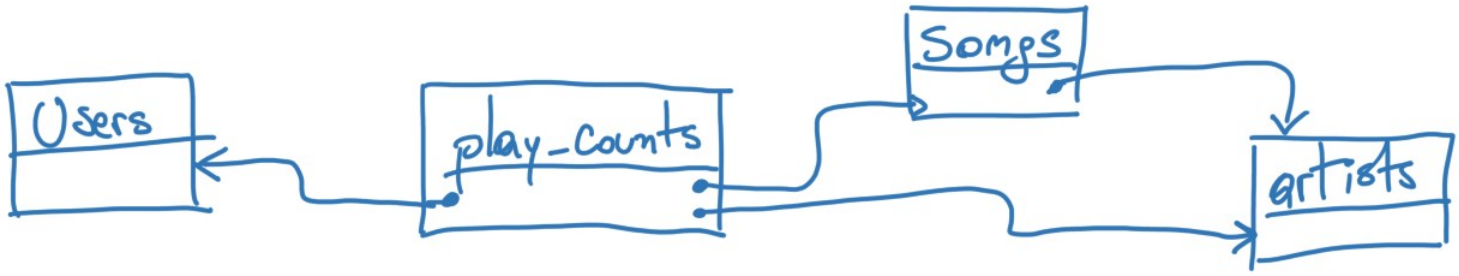

RAM
I/O
Disk

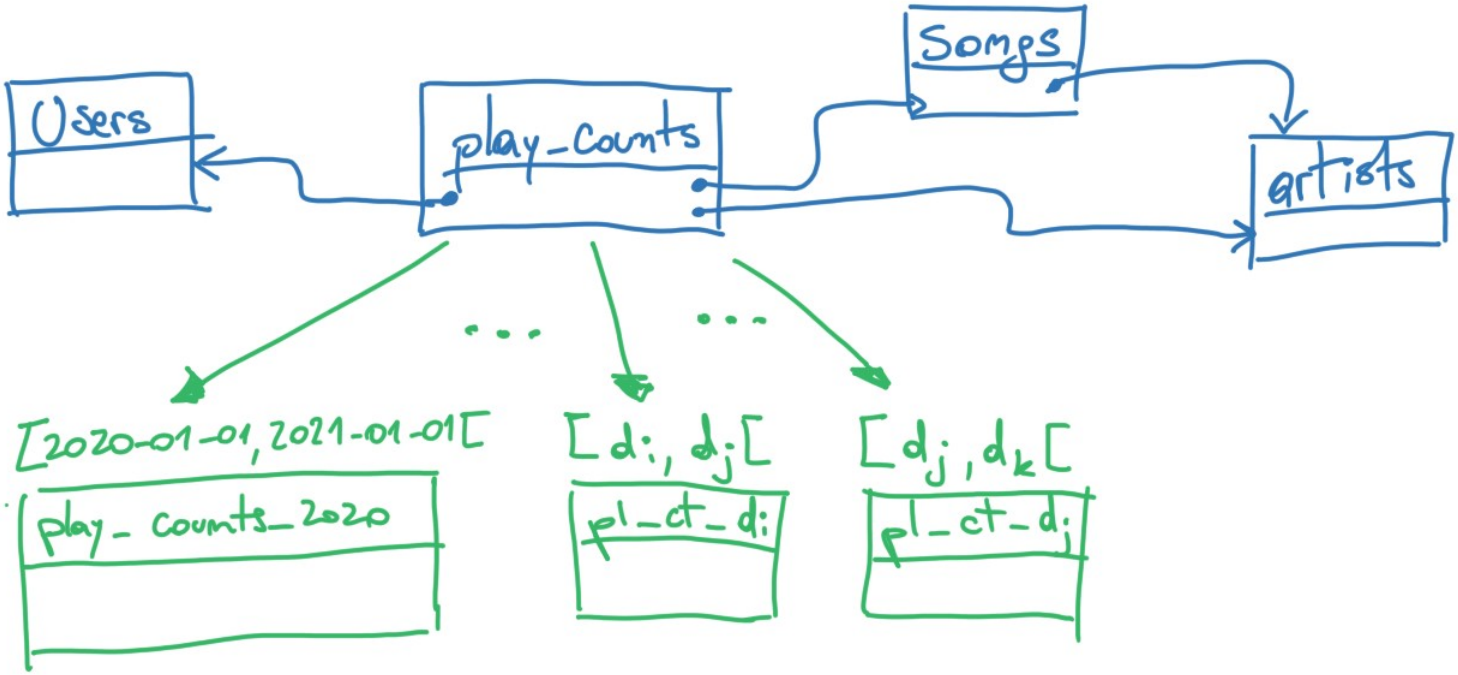
Online music player collecting stats

```
CREATE TABLE play_counts (
    user_id     bigint      REFERENCES users(user_id)
  , played_on   timestamp(0) WITH TIME ZONE
  , song_id     bigint      REFERENCES songs(song_id)
  , artist_id   bigint      REFERENCES artists(artist_id)
  , metadata    jsonb
  , PRIMARY KEY (user_id, played_on)
);
```

Online music player collecting stats

```
CREATE TABLE play_counts (
    user_id     bigint      REFERENCES users(user_id)
  , played_on   timestamp(0) WITH TIME ZONE
  , song_id     bigint      REFERENCES songs(song_id)
  , artist_id   bigint      REFERENCES artists(artist_id)
  , metadata    jsonb
  , PRIMARY KEY (user_id, played_on)
) PARTITION BY RANGE (played_on);
```


Users
play_counts
Songs
artists


Users
play_counts
Songs
artists
...
...
[2020-01-01, 2021-01-01[
play_counts_2020
[d_i, d_j[
pl_ct_d_i
[d_j, d_k[
pl_ct_d_j

Online music player collecting stats

```
CREATE TABLE play_counts (
    user_id     bigint      REFERENCES users(user_id)
  , played_on   timestamp(0) WITH TIME ZONE
  , song_id     bigint      REFERENCES songs(song_id)
  , artist_id   bigint      REFERENCES artists(artist_id)
  , metadata    jsonb
  , PRIMARY KEY (user_id, played_on)
) PARTITION BY RANGE (played_on);
```

Creating partitions

```
CREATE TABLE play_counts_2020 PARTITION OF play_counts
FOR VALUES FROM ('2020-01-01') TO ('2021-01-01');
```

Creating partitions

```
CREATE TABLE play_counts_2020 PARTITION OF play_counts
FOR VALUES FROM ('2020-01-01') TO ('2021-01-01');

CREATE TABLE play_counts_2021q1 PARTITION OF play_counts
FOR VALUES FROM ('2021-01-01') TO ('2021-04-01');

CREATE TABLE play_counts_2021q2 PARTITION OF play_counts
FOR VALUES FROM ('2021-04-01') TO ('2021-07-01');

CREATE TABLE play_counts_2021q3 PARTITION OF play_counts
FOR VALUES FROM ('2021-07-01') TO ('2021-10-01');

CREATE TABLE play_counts_2021q4 PARTITION OF play_counts
FOR VALUES FROM ('2021-10-01') TO ('2022-01-01');
```

Let’s see it working

Understanding what is going on

```
SELECT COUNT(*)
FROM play_counts;
```

Understanding what is going on

```
EXPLAIN ANALYZE
SELECT COUNT(*)
FROM play_counts;
```

Understanding what is going on

```
EXPLAIN (ANALYZE, BUFFERS)
SELECT COUNT(*)
FROM play_counts;
```

Understanding what is going on

```
EXPLAIN (ANALYZE, BUFFERS)
SELECT COUNT(*)
FROM play_counts
WHERE played_on >= now() - '1 month'::interval;
```

Postgres can prune tables based on the partition key `played_on`

Postgres can prune tables based on the partition key `played_on`

What about queries based on the user?

Query across all partitions

```
EXPLAIN ANALYZE
SELECT *
FROM play_counts
WHERE user_id = 10;
```

Dynamically prune partitions

```
EXPLAIN ANALYZE
SELECT *
FROM play_counts
WHERE user_id = 10
ORDER BY played_on DESC
LIMIT 200;
```

Getting the Query right is Fundamental

Getting the Query right is Fundamental

From

```
SELECT COUNT(*)
FROM play_counts p
JOIN artists a ON p.artist = a.artist_id
WHERE a.name='In Extremo';
```

Getting the Query right is Fundamental

From

```
SELECT COUNT(*)
FROM play_counts p
JOIN artists a ON p.artist = a.artist_id
WHERE a.name='In Extremo';
```

To

```
SELECT COUNT(*)
FROM play_counts p
WHERE artist = 22780
AND played_on >= '2021-11-21';
```

Consider CTEs – WITH Queries

```
WITH artist AS (
    SELECT artist_id, added_on
    FROM artists WHERE name = 'In Extremo'
)
SELECT COUNT(*)
FROM play_counts
WHERE artist = (SELECT artist_id FROM artist)
AND played_on >= (SELECT added_on FROM artist);
```

What about indexes?

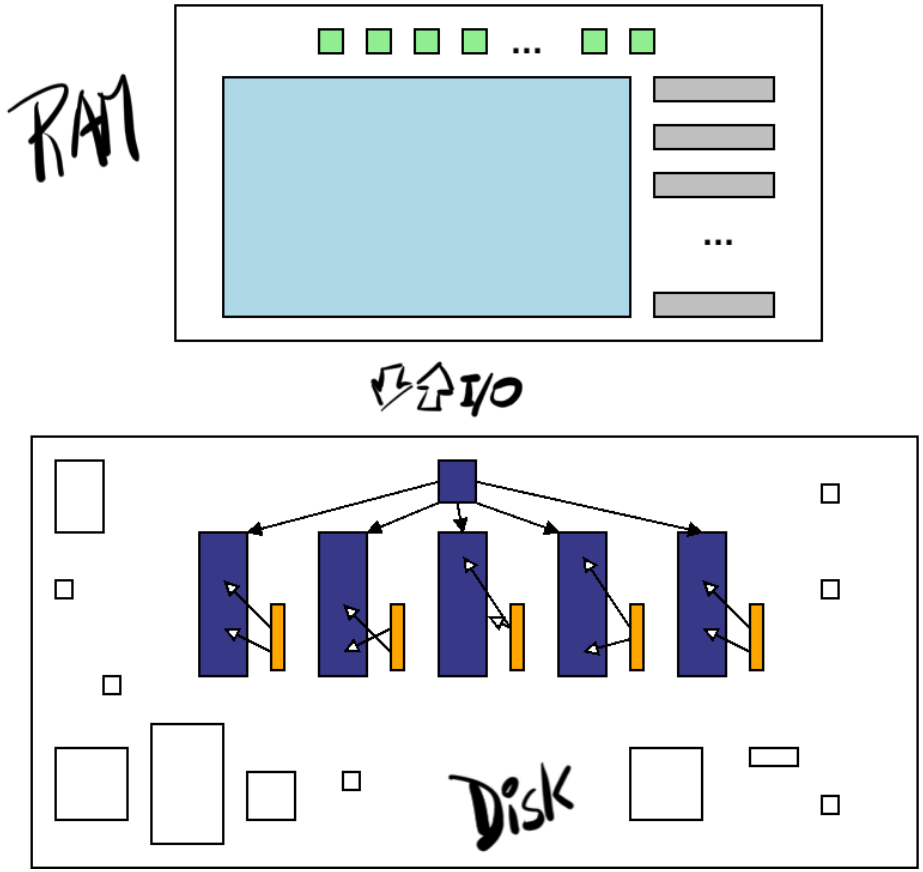
Create index at the parent table

```
CREATE INDEX play_counts_artist_idx
    ON play_counts (artist_id);
```

Create index at the parent table

```
CREATE INDEX play_counts_artist_idx
    ON play_counts (artist_id);
```

NOTE: This is NOT a global index!


RAM
I/O
Disk

Before we continue

Partition large tables when they have a logical partition key

Make sure your queries match the way tables are partitioned

More partition strategies


Meyers Konversations-Lexikon.
1. A bis Atlantiden.
2. Atlantis bis Blatthornkäfer.
3. Blattkäfer bis Chimbote.
4.
5. Distanzgeschäft bis Faidherbe.
6. Faidit bis Gehilfe.
7. Gehirn bis Hainichen.
8. Hainleite bis Iriartea.
9. Irideen bis Königsgrün.
10. Königshofen bis Luzon.
11. Luzula bis Nathanael.
12. Nathusius bis Phlegmone.
13. Phlegon bis Rubinstein.
14. Rüböl bis Sodawasser.
15. Sodbrennen bis Uralit.
16. Uralsk bis Z.


Español
Aula Internacional
Esselte
Quartier français

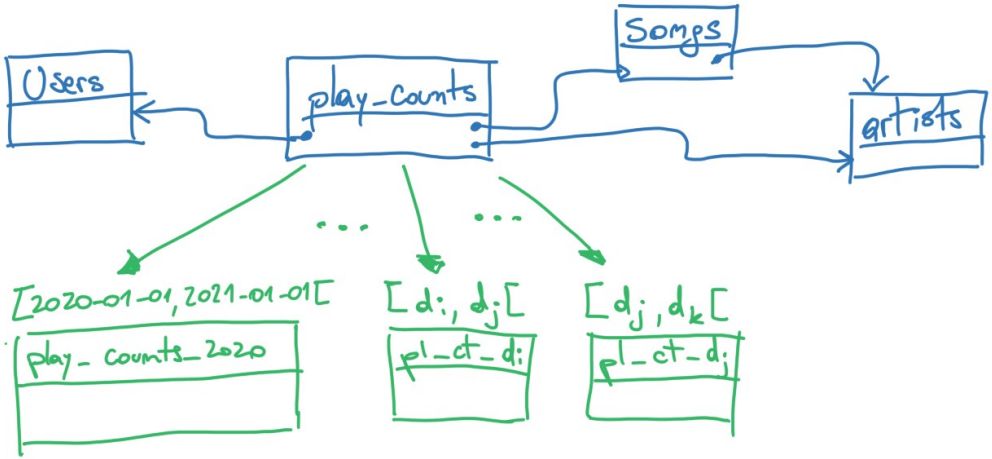
Table partitioning by HASH value

```
CREATE TABLE artists (
    artist_id   bigint GENERATED ALWAYS AS IDENTITY
  , name        text
  , added_on    date
  , axed_on     date
  , PRIMARY KEY (artist_id)
) PARTITION BY HASH (artist_id);
```

Table partitioning by HASH value

```
CREATE TABLE artists_0 PARTITION OF artists
FOR VALUES WITH (modulus 3, remainder 0);

CREATE TABLE artists_1 PARTITION OF artists
FOR VALUES WITH (modulus 3, remainder 1);

CREATE TABLE artists_2 PARTITION OF artists
FOR VALUES WITH (modulus 3, remainder 2);
```

Complete support for Foreign Keys



Complete support for Foreign Keys

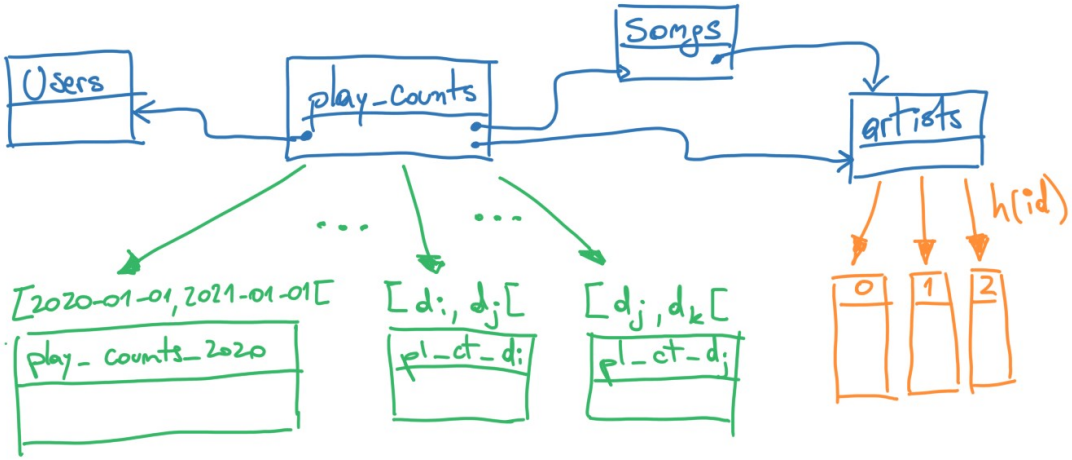


Table partitioning by LIST

```
CREATE TABLE playlists (
    id          uuid
  , name        text
  , category    text
  , tracks      jsonb
) PARTITION BY LIST (category);
```

Table partitioning by LIST

```
CREATE TABLE playlists_kids PARTITION OF playlists
FOR VALUES IN ('kids');

CREATE TABLE playlists_16 PARTITION OF playlists
FOR VALUES IN ('16+', 'explicit', 'heartbreaker');
```

The DEFAULT Partition

```
CREATE TABLE playlists_kids PARTITION OF playlists
FOR VALUES IN ('kids');

CREATE TABLE playlists_16 PARTITION OF playlists
FOR VALUES IN ('16+', 'explicit', 'heartbreaker');

CREATE TABLE playlists_default PARTITION OF playlists
DEFAULT;
```

Inserting data

```
INSERT INTO playlists
VALUES (gen_random_uuid(), 'baumhaus', 'kids', NULL);
-- goes into playlists_kids
```

Inserting and Updating data

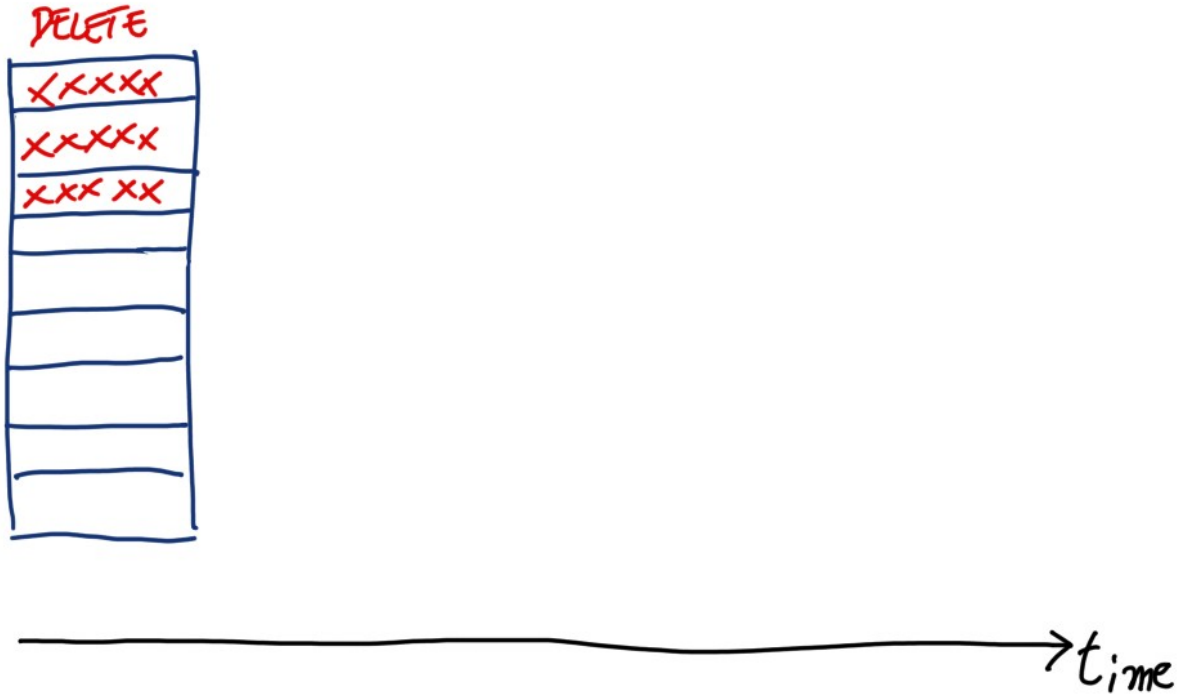
```
INSERT INTO playlists
VALUES (gen_random_uuid(), 'baumhaus', 'kids', NULL);
-- goes into playlists_kids

UPDATE playlists
SET category = 'explicit' WHERE name='baumhaus';
-- from playlists_kids to playlists_16
```

Inserting more data

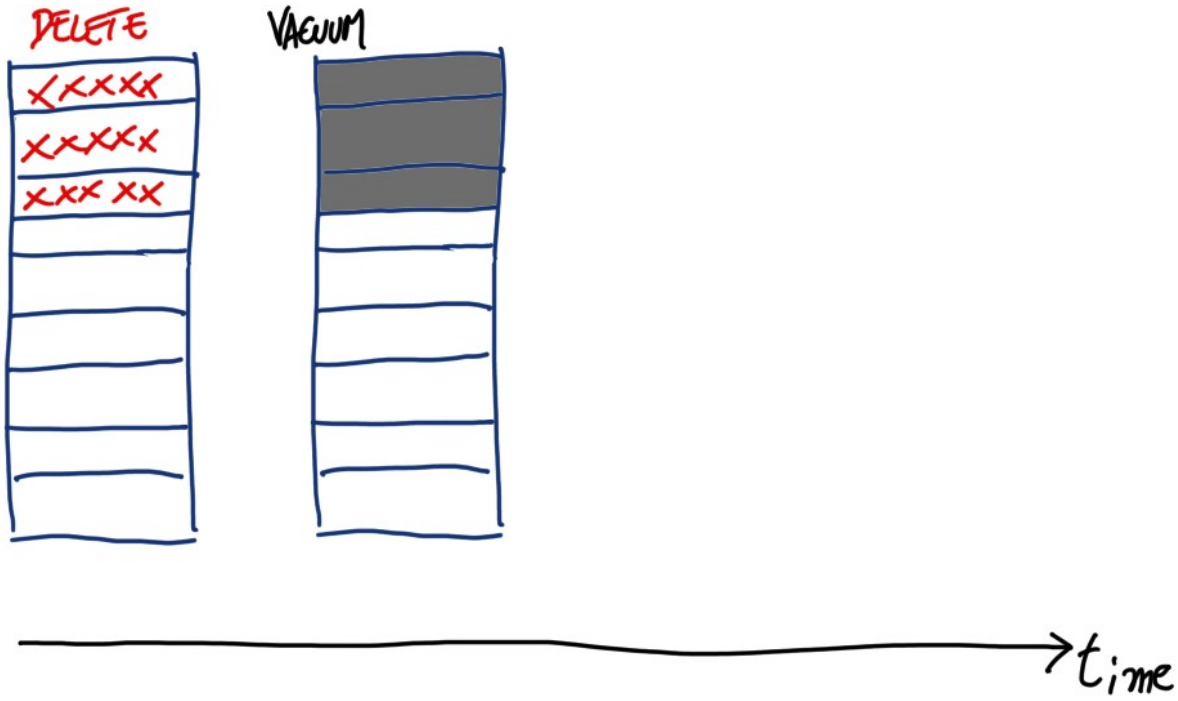
```
INSERT INTO playlists
VALUES (gen_random_uuid(), 'la plage', 'senior', NULL);
-- goes into playlists_default
```

Maintenance

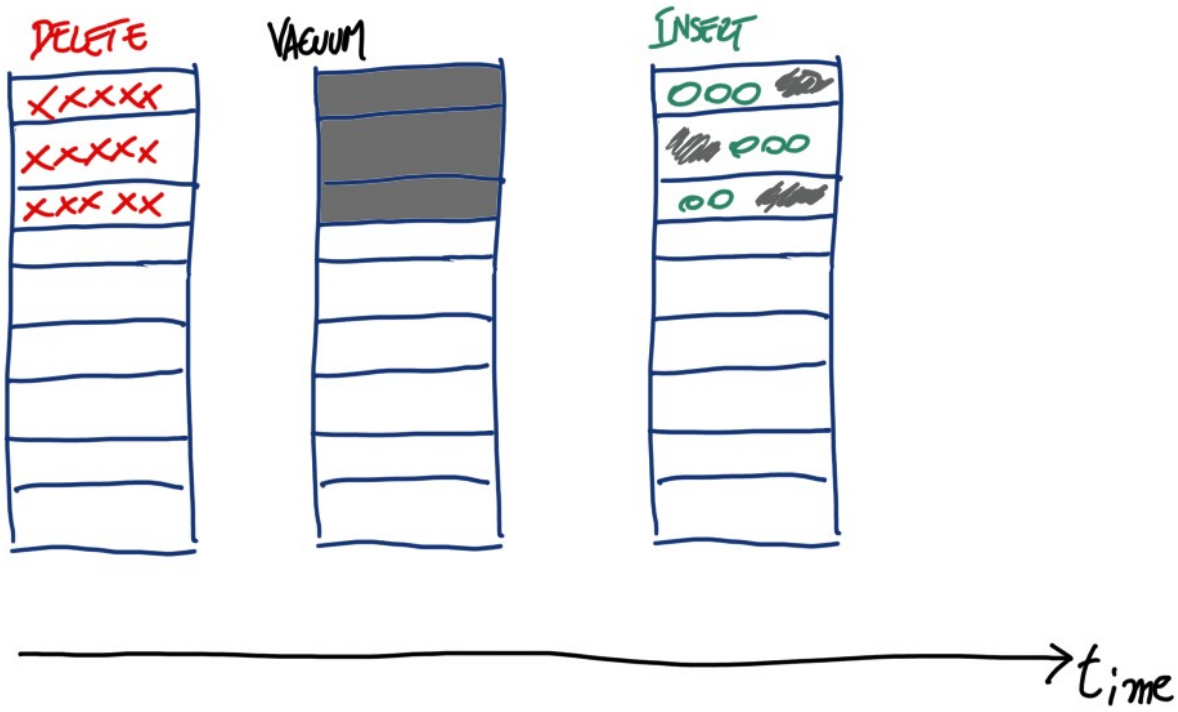
Delete data in Postgres



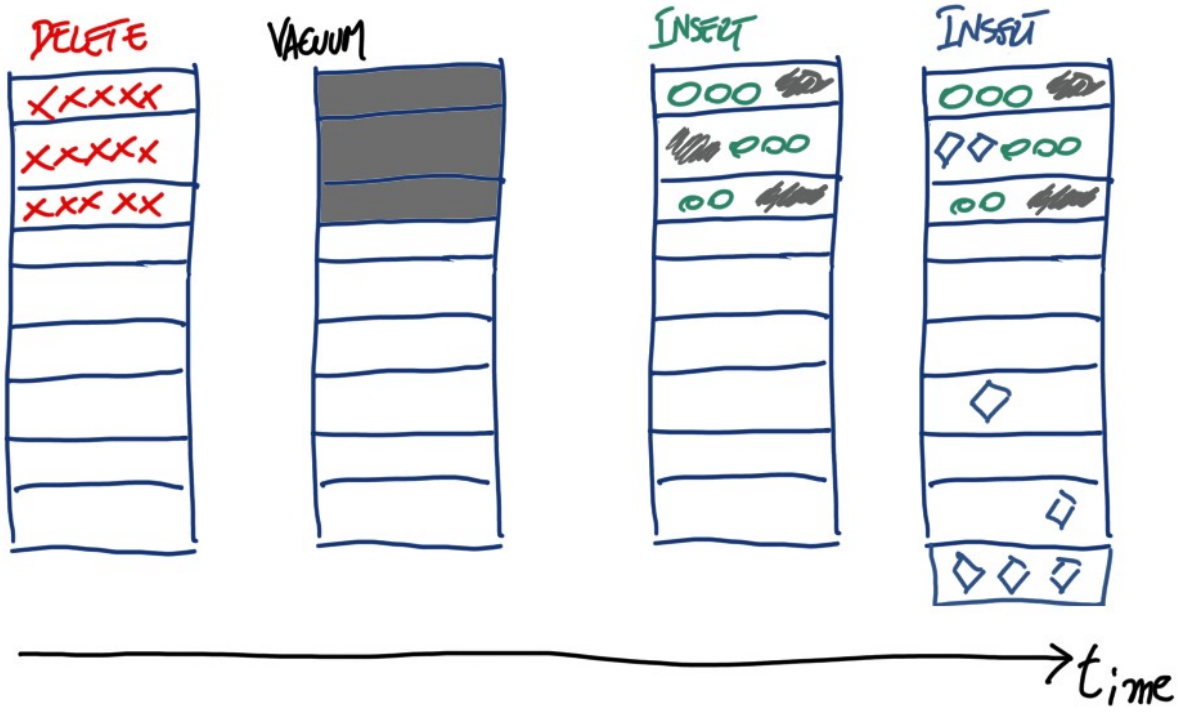
Delete data in Postgres



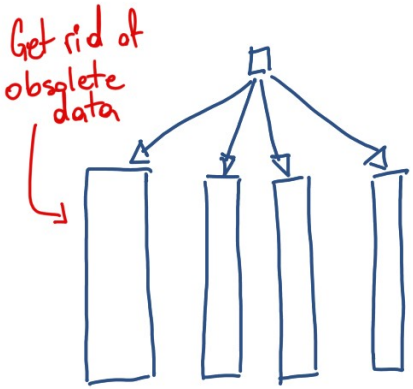
Delete data in Postgres



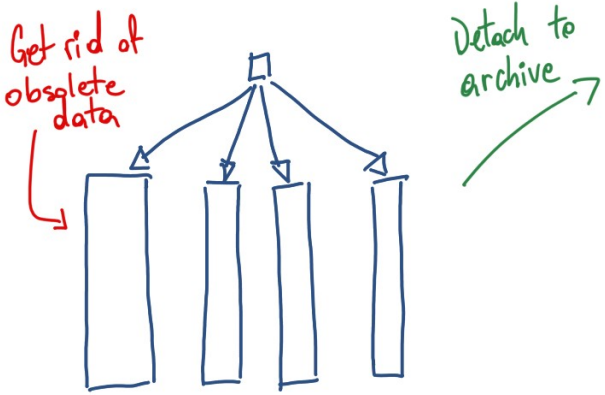
Delete data in Postgres

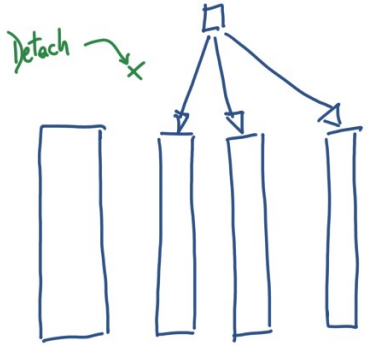


Detach or Drop partition

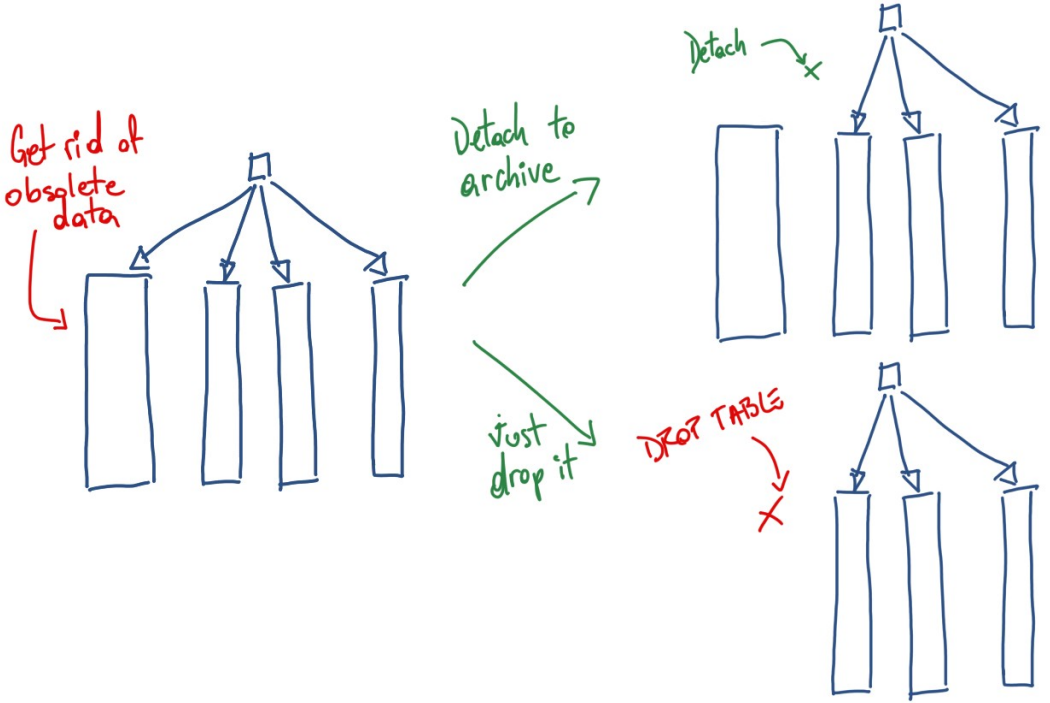


Detach or Drop partition





Detach or Drop partition



Detach a partition

```
ALTER TABLE play_counts DETACH PARTITION play_counts_2019;
```

Dropping a partition is dropping a table

```
DROP TABLE play_counts_2019;
```

Getting rid of obsolete data

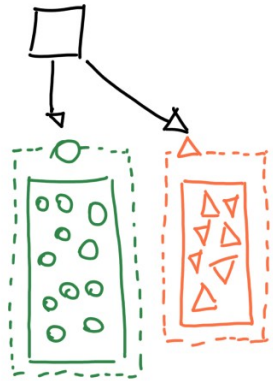
```
ALTER TABLE play_counts DETACH PARTITION play_counts_2019;

DROP TABLE play_counts_2019;
```

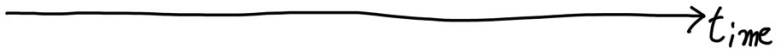
Create new partitions

```
CREATE TABLE play_counts_202211 PARTITION OF play_counts
FOR VALUES FROM ('2022-11-01') TO ('2022-12-01');
```

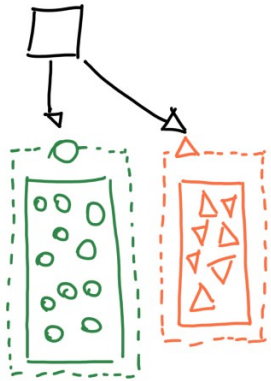
What about importing a full partition? What about bulk uploads?

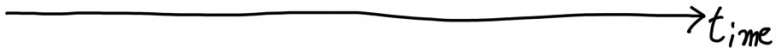
Attach partition with data



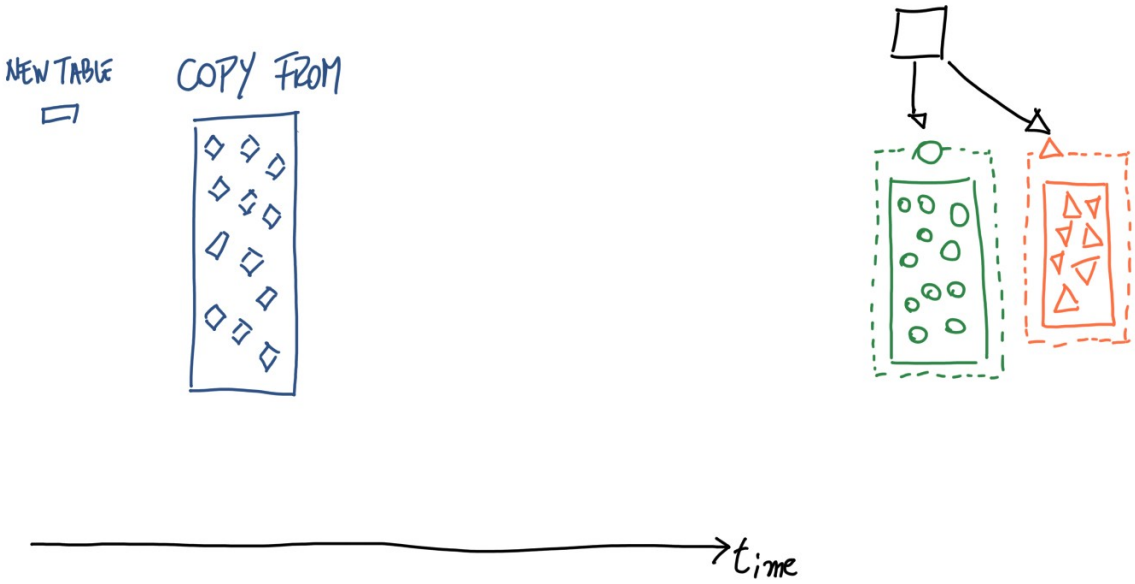


Attach partition with data

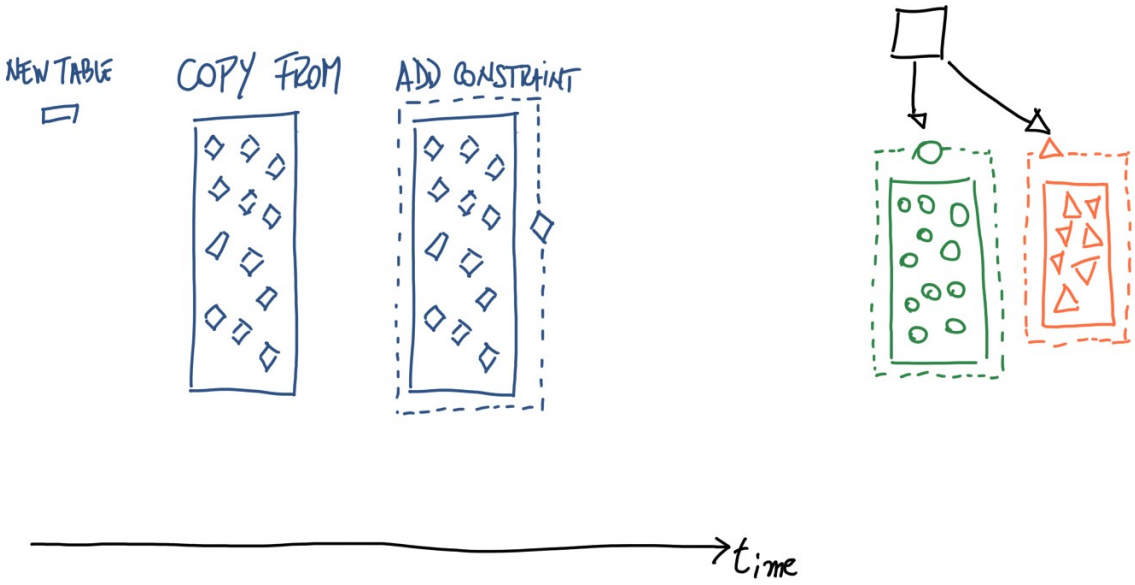




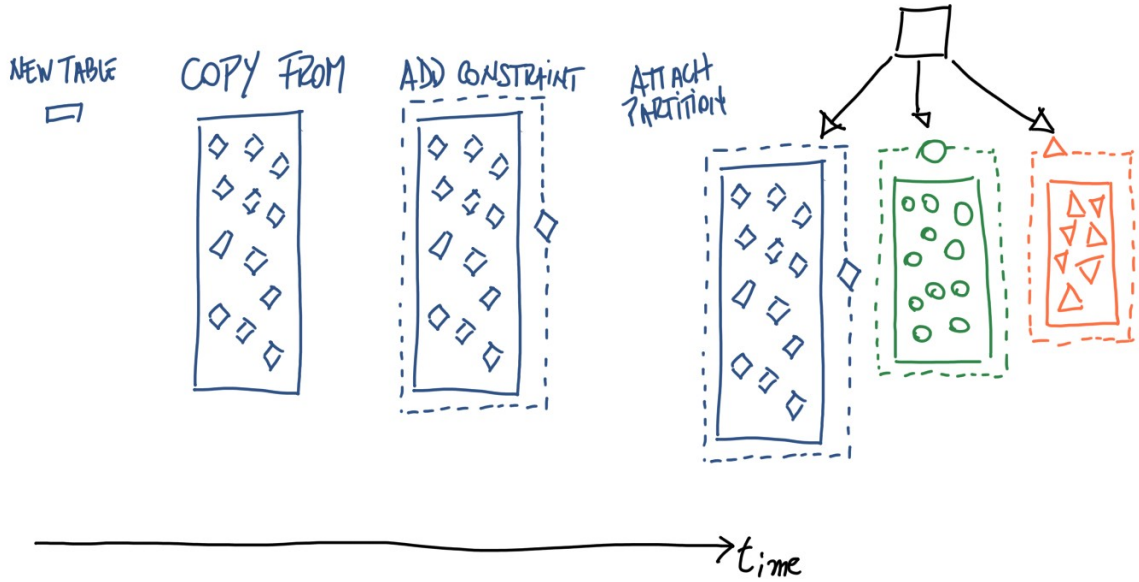
Attach partition with data



Attach partition with data

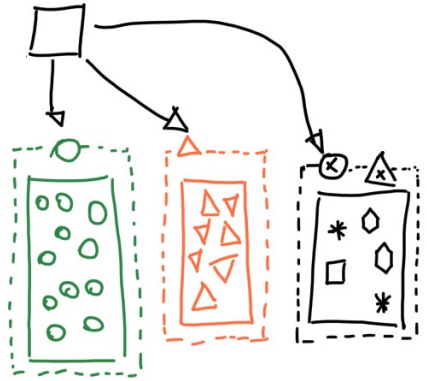


Attach partition with data

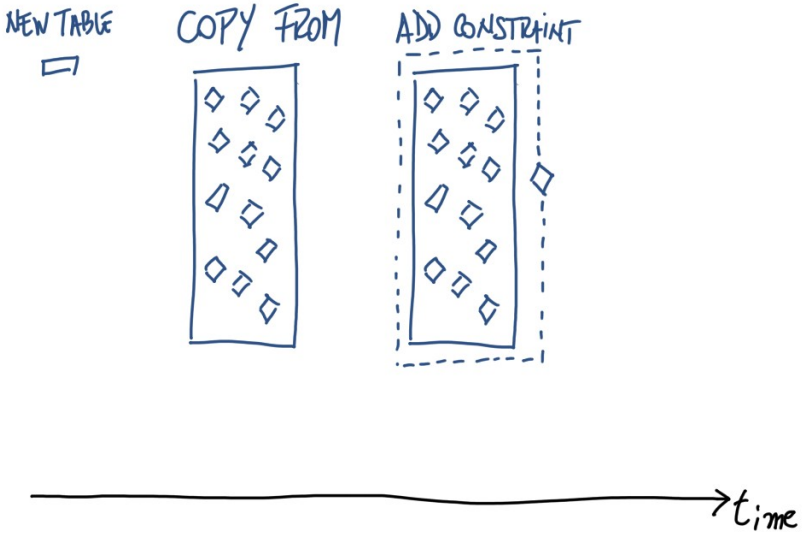


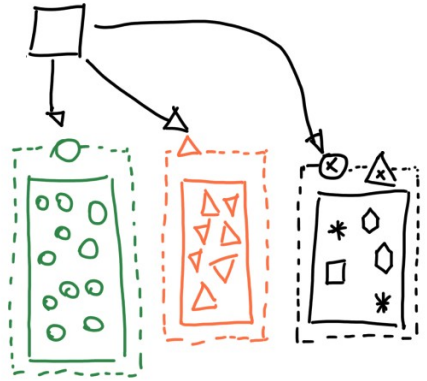
Remember
No Magic

The price of Default partitions

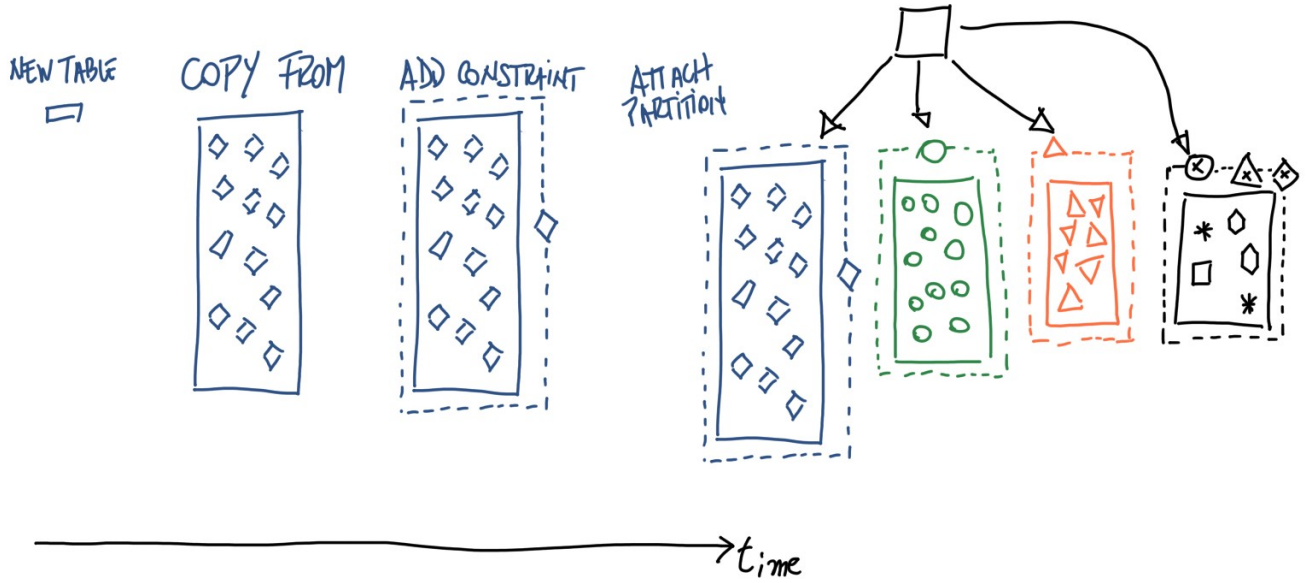


The price of Default partitions





The price of Default partitions



Import new partition

```
CREATE TABLE playlists_headbanging
    (LIKE playlists including all);

\COPY playlists_headbanging
FROM '/tmp/headbanging_postgres.txt' CSV DELIMITER ',';
```

Add Constraint before attaching

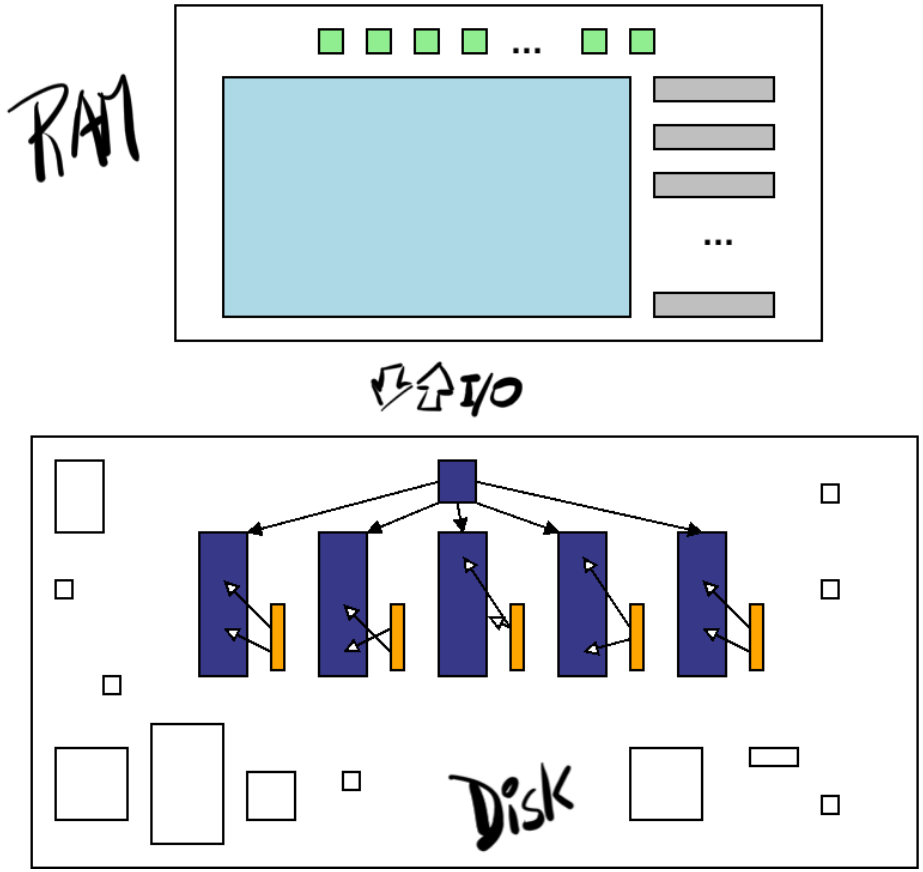
```
ALTER TABLE playlist_headbang
ADD CONSTRAINT headbanger
CHECK ((category IS NOT NULL) AND
       (category = ANY (ARRAY['rock', 'metal'])));
```

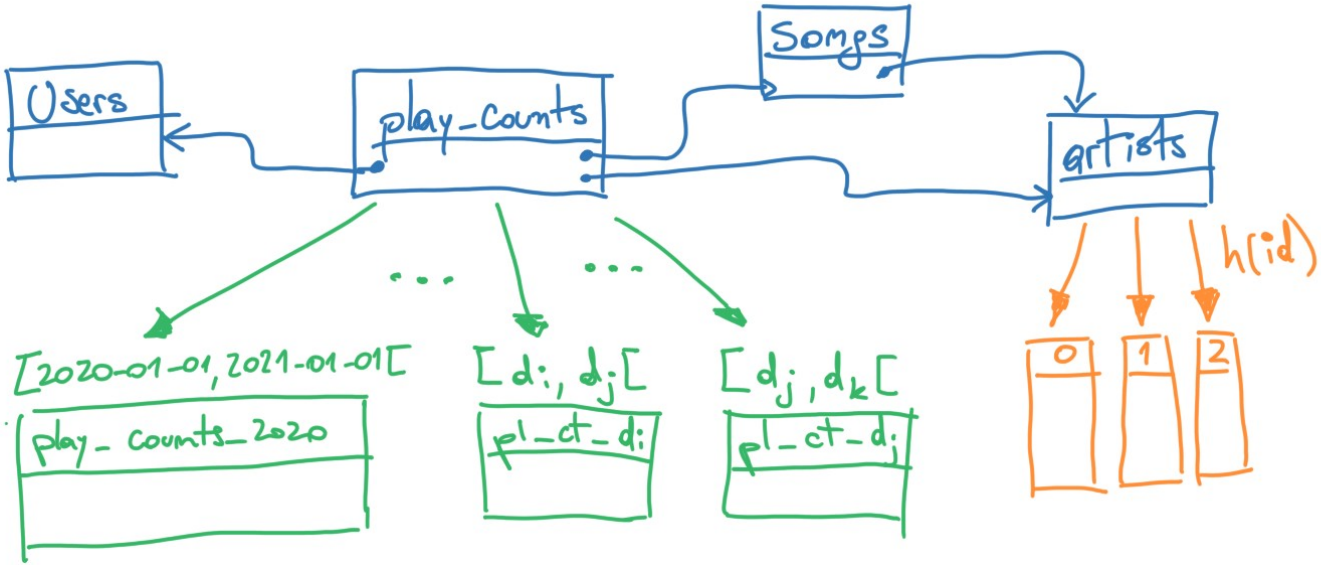
And attach

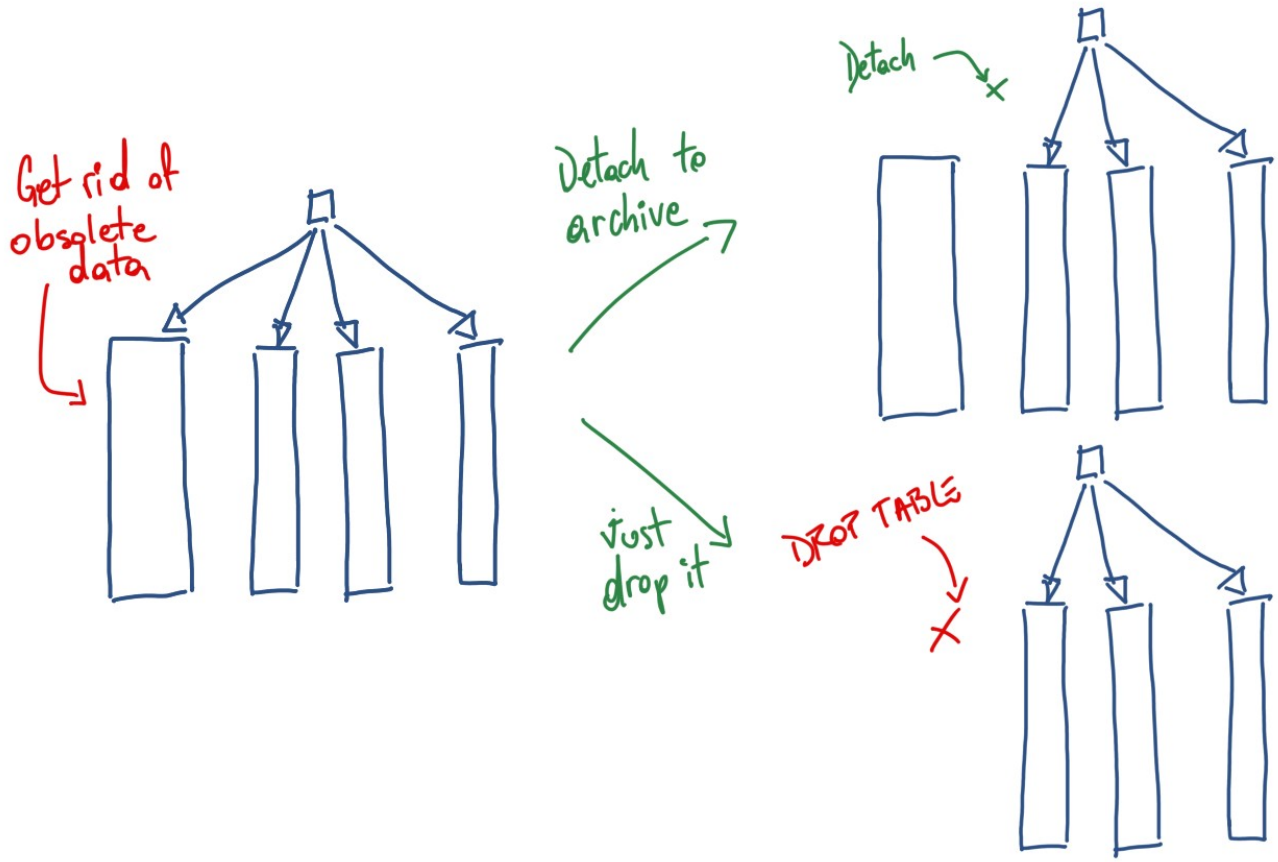
```
ALTER TABLE playlist_headbang
ADD CONSTRAINT headbanger
CHECK ((category IS NOT NULL) AND
       (category = ANY (ARRAY['rock', 'metal'])));

ALTER TABLE playlists
ATTACH PARTITION playlist_headbang
FOR VALUES IN ('metal', 'rock');
```

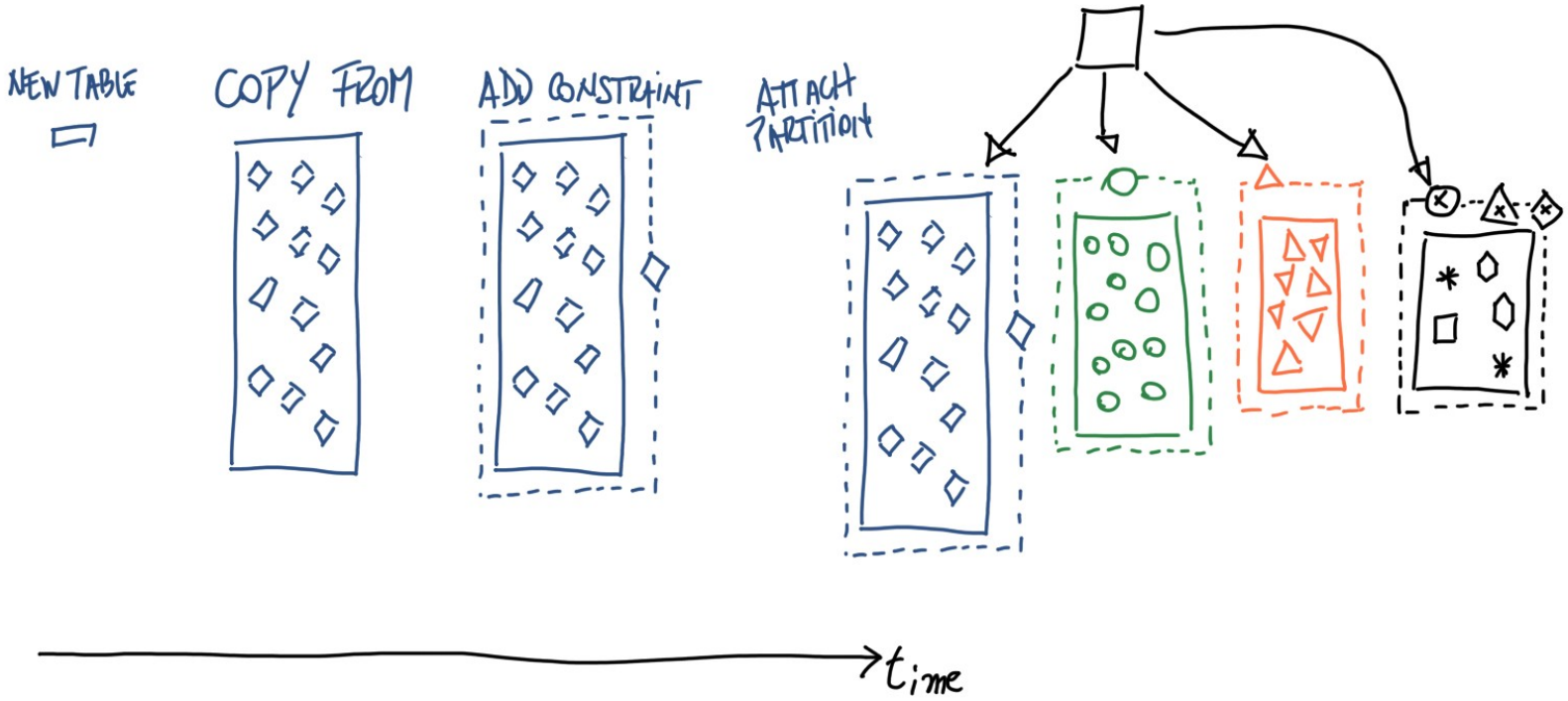
Remember
No Magic

Declarative Table Partitioning


RAM
I/O
Disk


Users
play_counts
Songs
artists
h(id)
0
1
2
[2020-01-01, 2021-01-01[
play_counts_2020
[d_i, d_j [
pl_ct_d_i
[d_j, d_k [
pl_ct_d_j


Get rid of obsolete data
Detach to archive
Just drop it
Detach
DROP TABLE


NEW TABLE
COPY FROM
ADD CONSTRAINT
ATTACH PARTITION
time

And don’t forget to
EXPLAIN and ANALYZE
your queries.
You may need to redesign.

Table Partitioning Transparent but No Magic

Danke schön

boriss.mejias@enterprisedb.com
@tchorix

EDB