2025-03-18 Nordics PGDay

Why and how to partition

This was a no-slide presentation. This deck summarises My main speaking topics.

adyen

engineered
for ambition

Why to partition

Reasons to partition your tables

1. Physical table limit of 32TB
2. Table maintenance is way easier with smaller table
3. Data life cycle management

Why to partition

Reasons to partition your tables

1. Physical table limit of 32TB

You are too late with partitioning, but lets start today

Why to partition

Reasons to partition your tables

2. Table maintenance

By the time your table reaches 100GB it is time to start thinking about partitioning. You might get issues with vacuum, maintaining (gin) indexes. Don’t just upgrade to better hardware.

Why to partition

Reasons to partition your tables

3. Data life cycle management

Everybody creates tables to store data in the database. Many people think about normalisation of the data and sometimes they denormalise data to improve performance.

Only few people think of data life cycle management. Data looses its value over time and partitioning makes it really easy to clean old data. You simply drop the partition.

How to partition

How to partition your tables

1. List
2. Hash
3. Range

Whatever type of range partitioning you pick, don’t use default partitions. They are the root of all partitioning evil.

How to partition

How to partition your tables

1. List

Doesn’t scale nor does it help with life cycle management

How to partition

How to partition your tables

2. Hash

Hash partitioning helps, but not as well as you might think. When adding new partitions you will have to re-write all your data. As the hashes are random, you can’t use this method for data life cycle management.

How to partition

How to partition your tables

3. Range

The range partitions that satisfy all three reasons to partitions your tables

1. Date -> Nice for PII data
2. Integer -> easy because your primary keys are most likely some integer types. Hard to search on date ranges though. You will have to scan over all your partitions.
3. UUID. Use pg_uuidv7. Provides the same as the combination of both date and integer partitioning.

Joining partitioned tables

Joining partitioned tables might be hard.

To set yourself up for success

- Partition ALL your tables on the same column (leading figure)
- Have the same partition boundaries for all your partitions
- Play around with enable_partitionwise_join and enable_partitionwise_aggregates to enable partition by partition joins instead of parent by parent joins.

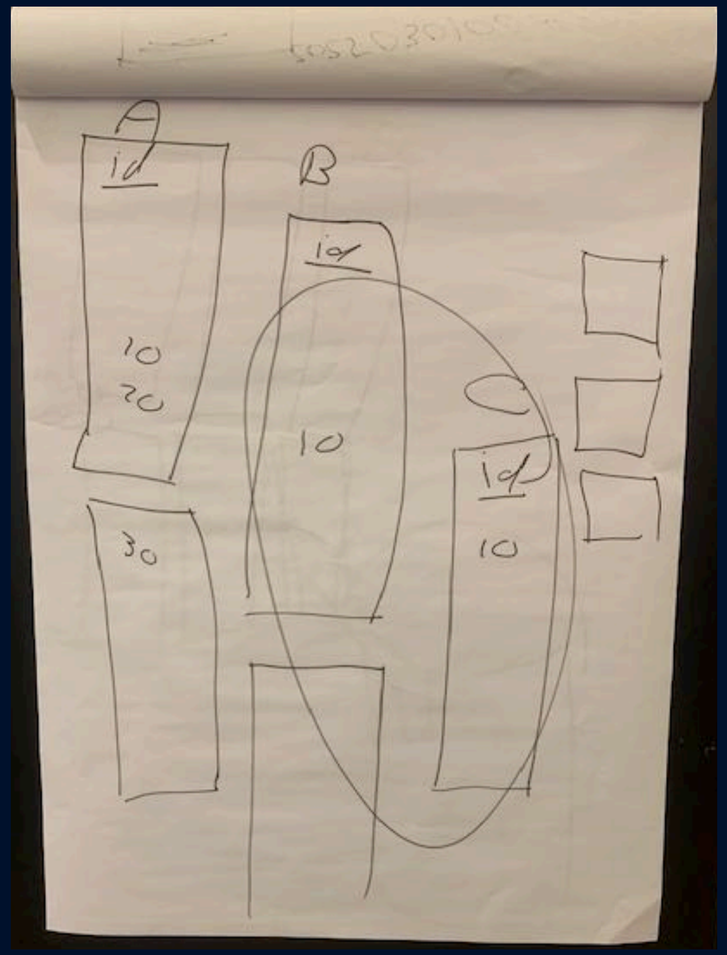
Archiving

You can move partitions out of your main database and store the data in some 2nd tier solution. From your main database you can create a foreign data wrapper to this 2nd tier solution to make the data transparently available to your application.

Bonus

Every time you create a new partition you can reorder the columns within the table as long as you keep the names and types the same. This way you can optimise the padding within your table and get better performance.

20250301000000
20250401000000


A
id
10
20
30
B
id
10
C
id
10