Declarative Partitioning Has Arrived!

Amit Langote (NTT OSS Center)

Ashutosh Bapat (EnterpriseDB)

@PGConf.ASIA 2017, Tokyo

Copyright©2017 NTT Corp. All Rights Reserved.

Outline

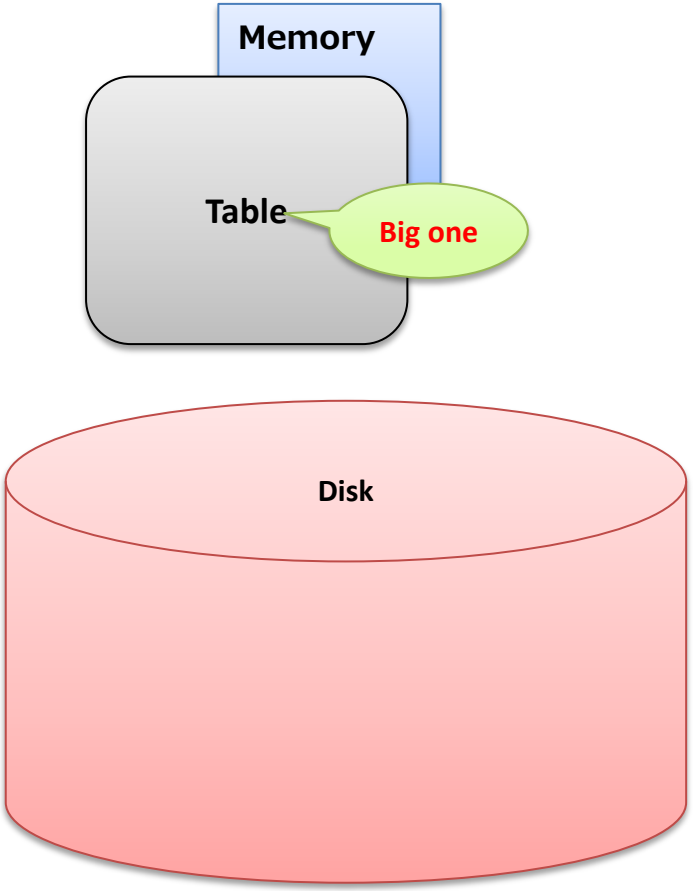
Innovative R&D by NTT

- Introduction of declarative partitioning in PostgreSQL 10 with examples
- A look at some limitations of new partitioning and plan to fix them
- Some commentary on improvements that declarative partitioning makes over table inheritance
- Introduction to partitioning planner improvements (Ashutosh)

NTT

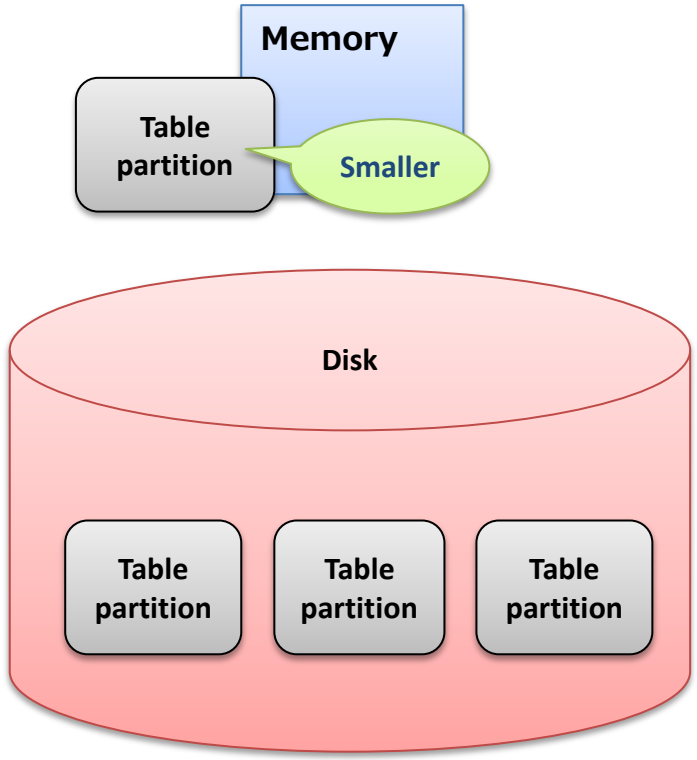
Copyright©2017 NTT Corp. All Rights Reserved.

Basics



Copyright©2017 NTT Corp. All Rights Reserved.

Basics



Copyright©2017 NTT Corp. All Rights Reserved.

Declarative Partitioning

Innovative R&D by NTT

From PostgreSQL 10 release notes

E.1.3.4. Utility Commands

- Add table partitioning syntax that automatically creates partition constraints and handles routing of tuple insertions and updates

  The syntax supports range and list partitioning.

NTT

Copyright©2017 NTT Corp. All Rights Reserved.

Declarative Partitioning

```
create table users (name text)
    partition by range (lower(left(name, 1)));

create table user_a_to_l
    partition of users for values from ('a') to ('m');

create table user_m_to_z
    partition of users for values from ('m') to ('z')
    partition by range (lower(left(name, 1)));

create table user_m_to_z_1
    partition of user_m_to_z for values from ('m') to ('t');

create table user_m_to_z_2
    partition of user_m_to_z for values from ('t') to ('z');
```

Copyright©2017 NTT Corp. All Rights Reserved.

Declarative Partitioning

```
insert into users values (‘Timmy’), (‘Andy’), (‘Molly’);

select tableoid::regclass as partition, * from users;
   partition   |     name
---------------+---------------
 user_a_to_l   | Andy
 user_m_to_z_1 | Molly
 user_m_to_z_2 | Timmy
(2 rows)

explain (costs off) select * from users where lower(left(name, 1)) >= 'm';
                      QUERY PLAN
-----------------------------------------------------
 Append
   ->  Seq Scan on user_m_to_z_1
         Filter: (lower("left"(name, 1)) >= 'm'::text)
   ->  Seq Scan on user_m_to_z_2
         Filter: (lower("left"(name, 1)) >= 'm'::text)
(5 rows)
```

Copyright©2017 NTT Corp. All Rights Reserved.

What's missing

- *Hash* partitioning (*in Postgres 11, thanks to Amul Sul*)
- A *default* partition to capture data without a pre-created partition (*in Postgres 11, thanks to Jeevan Ladhe*)
- **Create partitioned indexes (*maybe in Postgres 11, thanks to Alvaro Herrera*)**
  - **Related, *UNIQUE* constraint and hence *PRIMARY KEY* on partitioned tables**
- Handle UPDATE statement that causes data to change partition (*maybe in Postgres 11, thanks to Amit Khandekar*)
- Routing tuples to foreign partitions (*maybe in Postgres 11, thanks to Etsuro Fujita*)

Copyright©2017 NTT Corp. All Rights Reserved.

What's missing

- **Cannot use foreign key to/from partitioned tables (*in Postgres 1x, thanks to You?*)**
- Cannot define row-level triggers on partitioned tables (*in Postgres 1x, thanks to You?*)
- Ability to change partitioning of data after-the-fact by *splitting* a partition or by *merging* partitions (*in Postgres 1x, thanks to You?*)
- Automatic creation of partitions for incoming data (*in Postgres 1x, thanks to You?*)

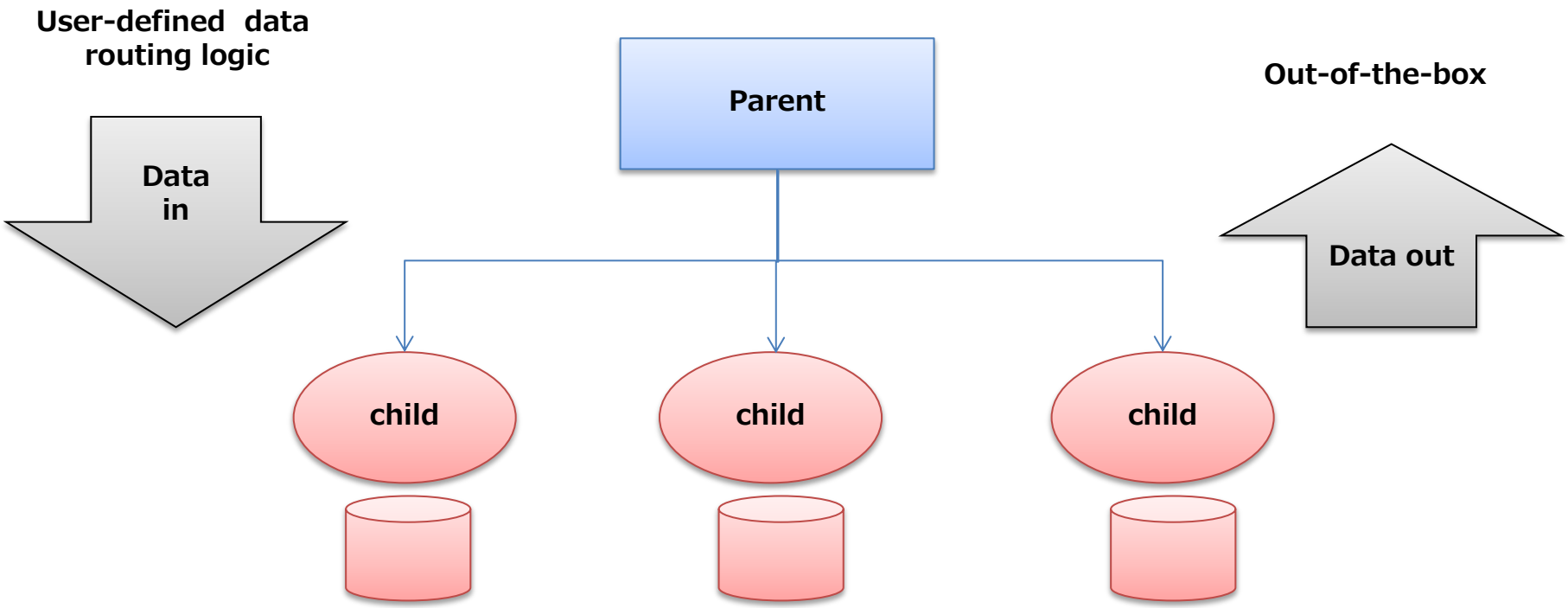
Copyright©2017 NTT Corp. All Rights Reserved.

Inheritance and Partitioning

- Table inheritance feature is enough for an application to set up basic table partitioning
- But, the application will need to implement itself important details such as non-overlapping partitions, data-routing, etc.
  - If such details are implemented in the application, database doesn't know about it, so it cannot optimize for partitioned data
  - Maybe, application could implement all the partitioning optimizations itself, but that won't be very productive in the long run

Copyright©2017 NTT Corp. All Rights Reserved.

Inheritance and Partitioning



Copyright©2017 NTT Corp. All Rights Reserved.

Inheritance and Partitioning

- Application needs to insert data into right partitions, for which it can use database facilities like rules/triggers
- PostgreSQL has enough infrastructure to provide the “data out” feature out-of-the-box
  - Keep schema in sync between parent and child tables
  - Planner translates queries applied to parent to include child tables
- PostgreSQL also has a planner feature to optimize access to partitioned data called constraint exclusion
  - Although, application needs to add the correct CHECK constraint to child tables

Copyright©2017 NTT Corp. All Rights Reserved.

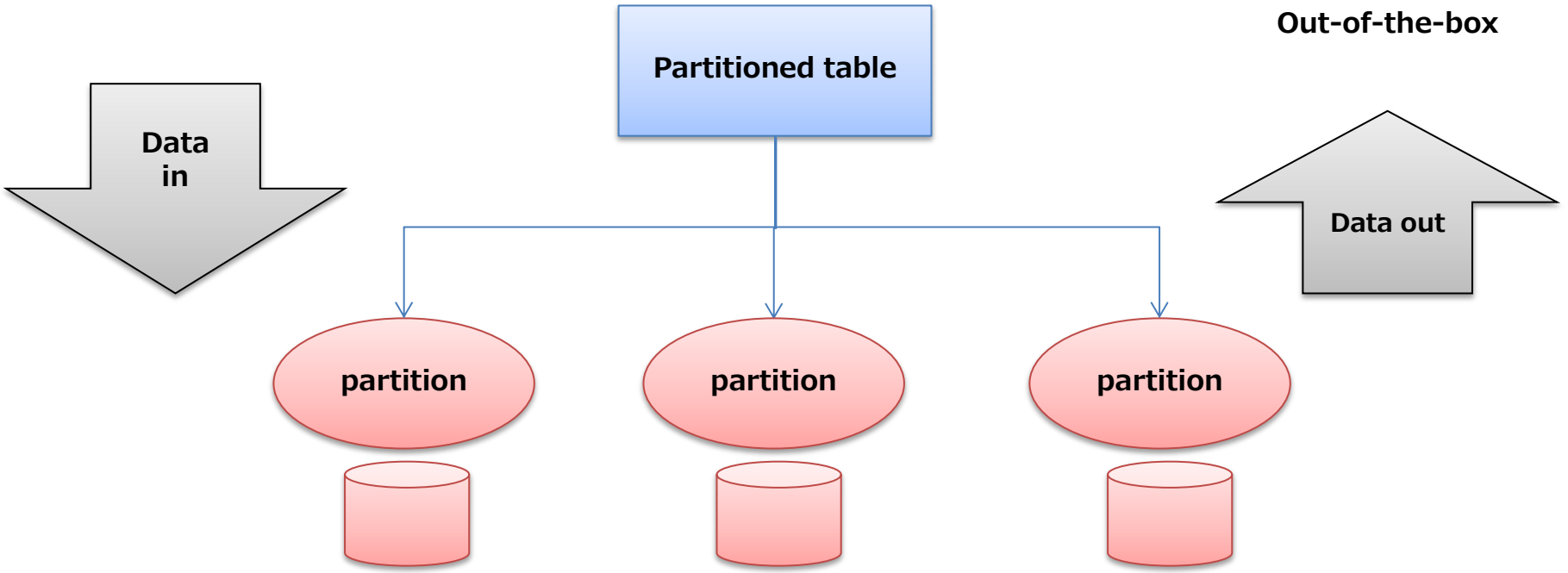
Inheritance and Partitioning

- The new table partitioning feature uses most of that infrastructure, without implementing it all from scratch
- So, overall, it doesn’t look much different to the user, except some new syntax and certain features provided out-of-the-box

Copyright©2017 NTT Corp. All Rights Reserved.

Inheritance and Partitioning

Out-of-the-box



Copyright©2017 NTT Corp. All Rights Reserved.

Inheritance and Partitioning

- Stopping here is not an option!
- Partitioning inside the database offers an opportunity to do more, especially on the planner side, because of all the metadata that's now available

NTT

Copyright©2017 NTT Corp. All Rights Reserved.

Questions?

Copyright©2017 NTT Corp. All Rights Reserved.