FDW-based Sharding Update and Future

NTT Open Source Software Center
Masahiko Sawada

PGConf Russia 2017 (16th March)

Who am I?

- **Masahiko Sawada**
  - Twitter : @sawada_masahiko
  - GitHub: MasahikoSawada

- **PostgreSQL Contributor**
  - Freeze Map(PG9.6)
  - Multiple Synchronous Replication(PG9.6)
  - Quorum-based Synchronous Replication(PG10)

- **PostgreSQL Technical Support**
  - pg_repack committer

Agenda

1. What is database sharding

2. What is FDW-based sharding

3. Demonstration

4. Use cases

5. Challenges and key techniques

6. Conclusion

Scale-up and Scale-out

- **Scale-up**
  - Vertical scaling
  - Simple
  - Price
  - Not safe against hardware failure

- **Scale-out**
  - Horizontal scaling
  - Easier to run fault-tolerantly
  - More complex

What is database sharding

- A scale-out technique
- The tables are divided and distributed into multiple servers
  - Row based
  - Column based
- A database shard can be placed on separate hardware

Pros and Cons

- **Pros**
  - Write scale out (Horizontal scaling)
  - Reduce I/O on each shard, by splitting data across shard
  - Access only required shard

- **Cons**
  - Node management
  - Cross-shard transaction could be cause of slow query
  - Downtime might be required when changing the sharding layout

Challenges

- Reliability
  - Backups of individual database shards
  - Replication of database shards
  - Automated failover
- Distributed queries
- Avoidance of cross-shard joins
- Auto-increment key, like sequence
- Distributed transactions

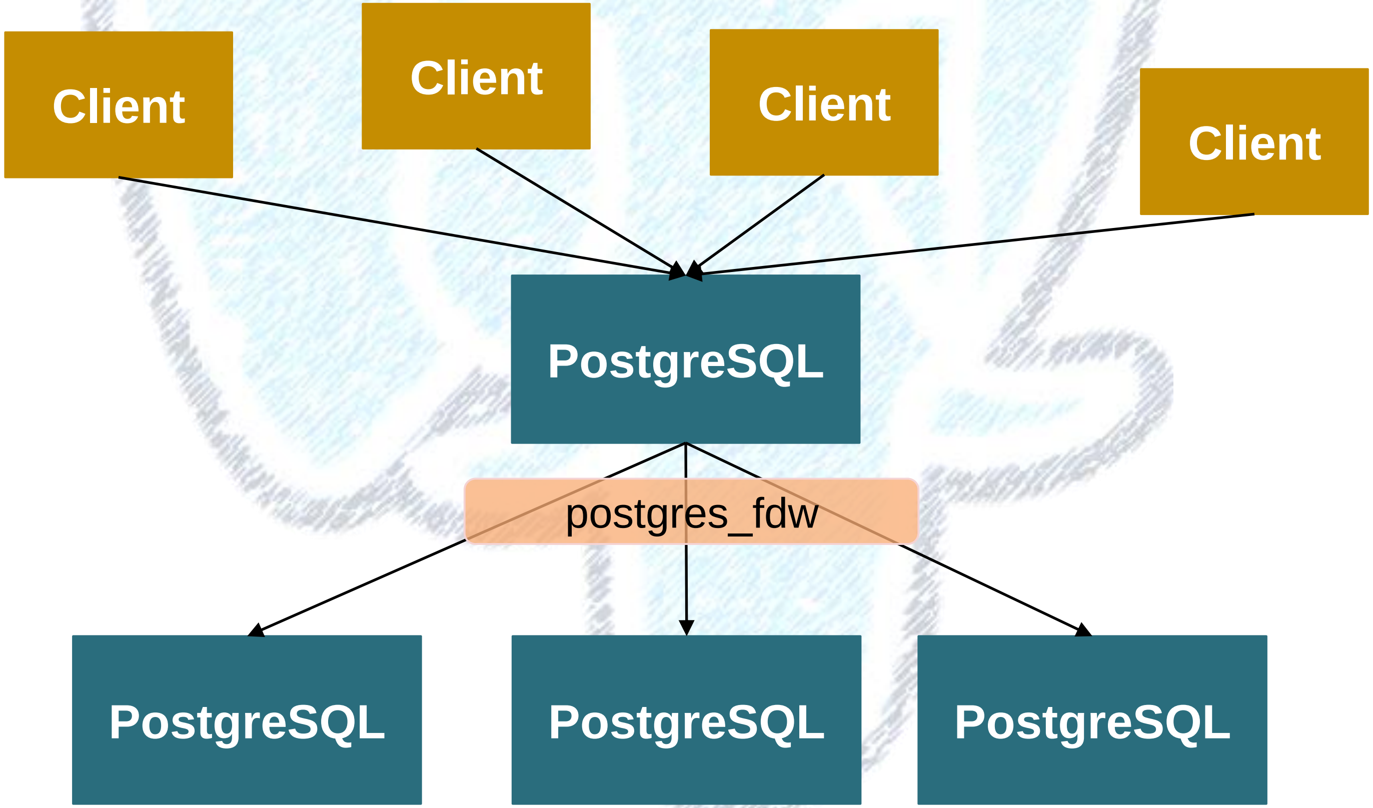
Well-known Products

- Postgres-XC by NTT, EDB
- Postgres-XL by 2ndQuadrant
- Postgres Cluster by Postgres Professional
- Greenplum by Pivotal
- pg_shard by CitusData

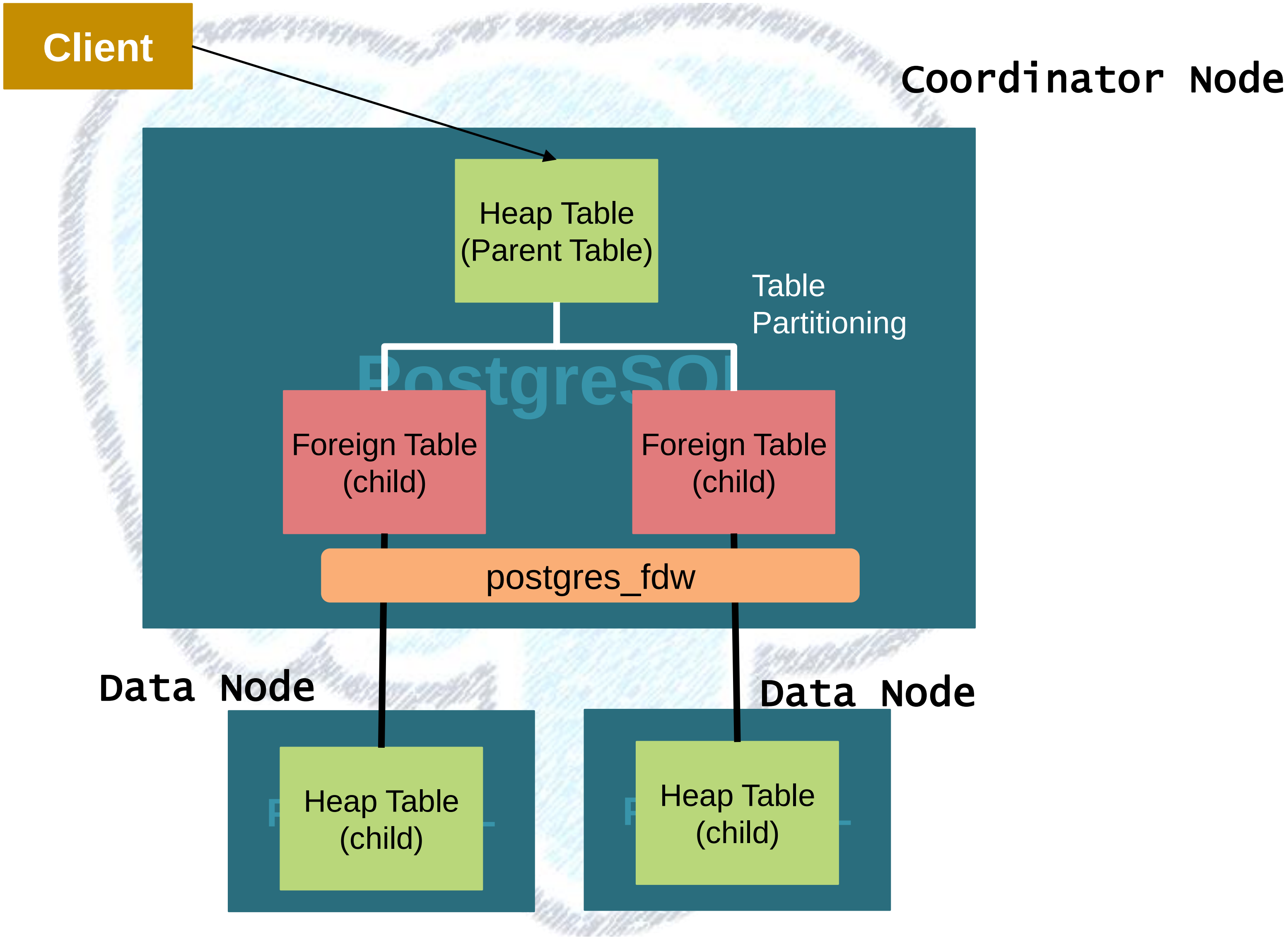
- Other than PostgreSQL,
  - VoltDB
  - MySQL Cluster
  - Spanner
  - etc

What is FDW-based sharding

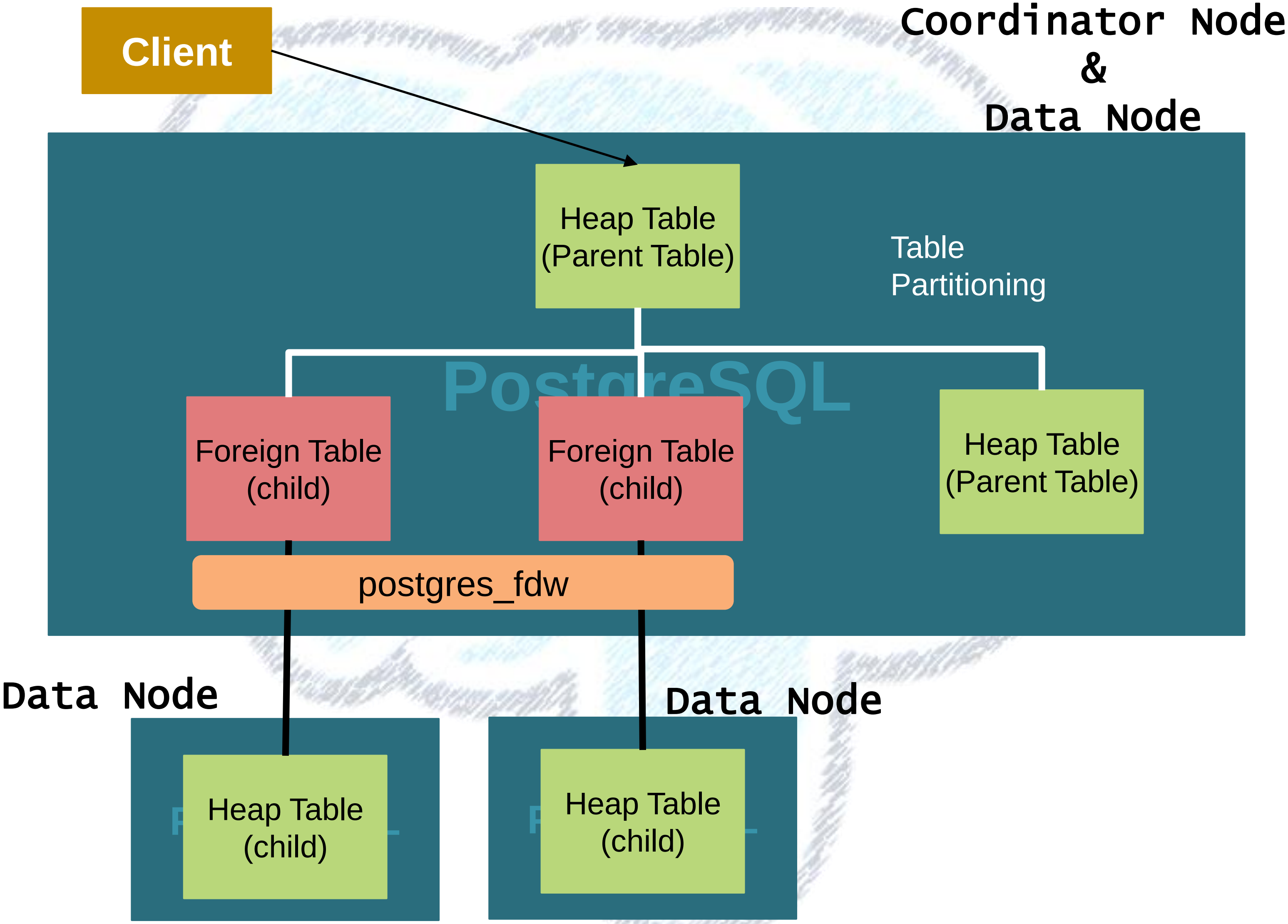
- FDW-based sharding is a database sharding techniques using mainly **FDW (Foreign-Data-Wrapper)** and **Table Partitioning**
- Our goal is providing a sharding solution as a **Built-in feature**.



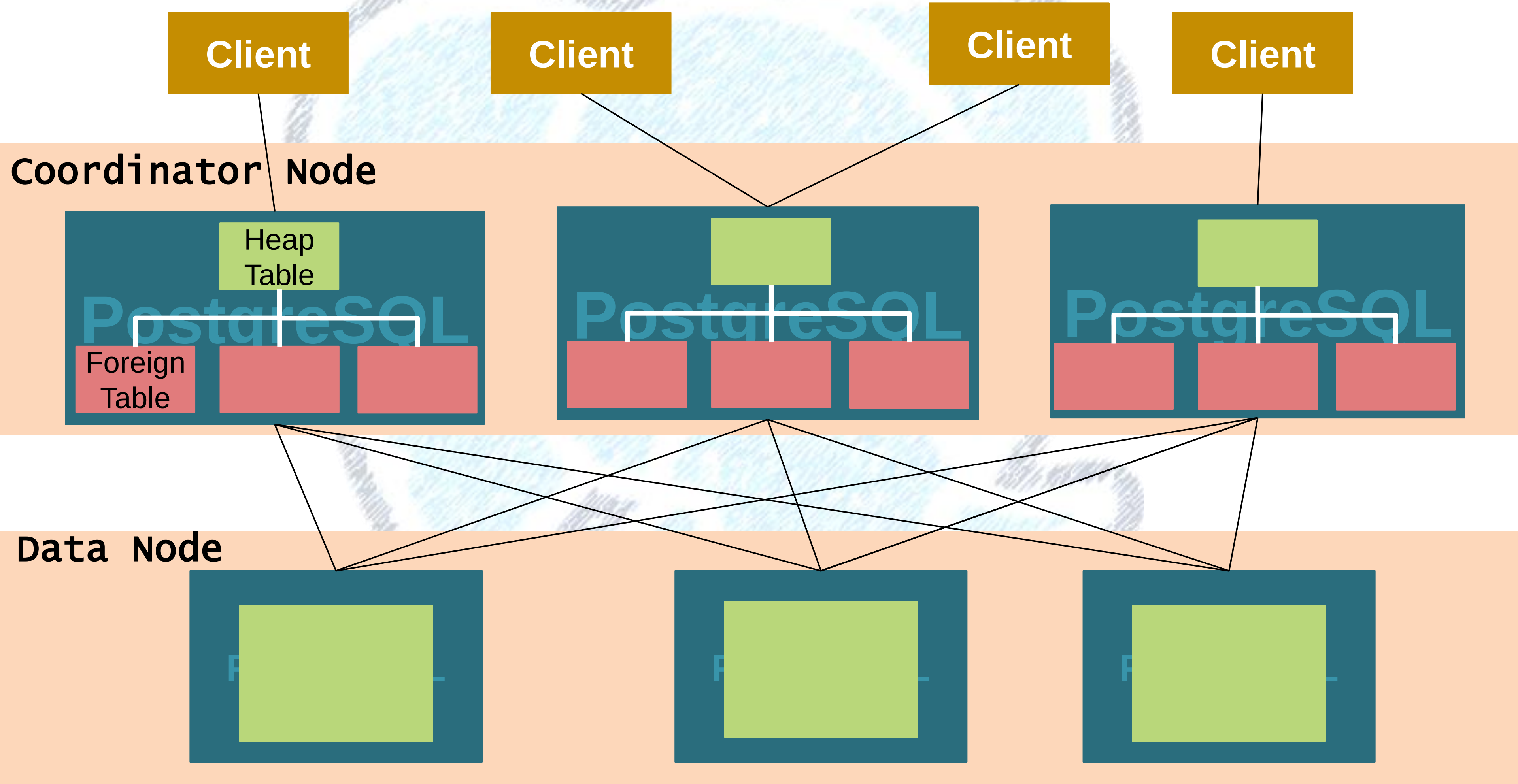
Basic Architecture (PG9.6)



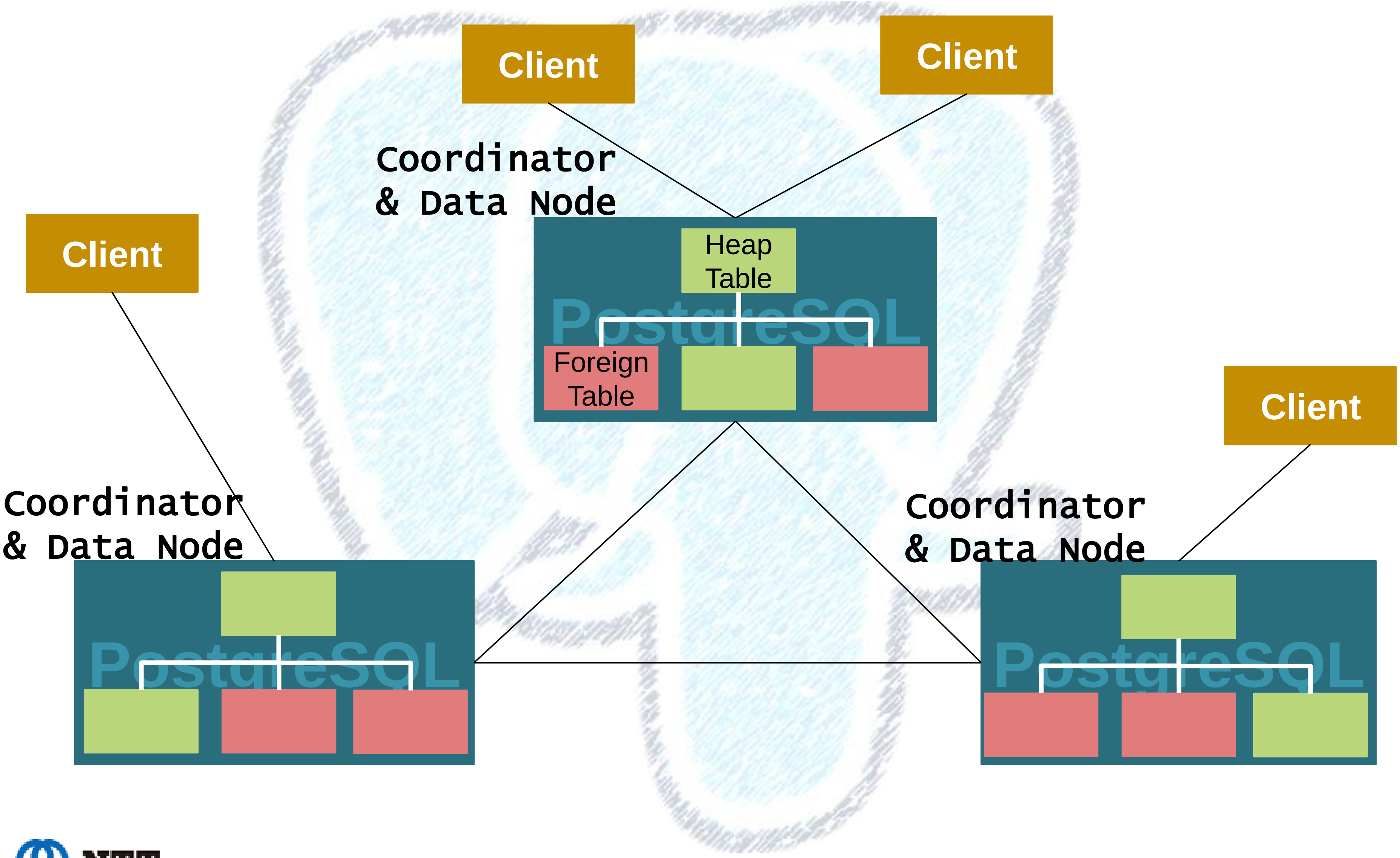
Basic Architecture (PG9.6)



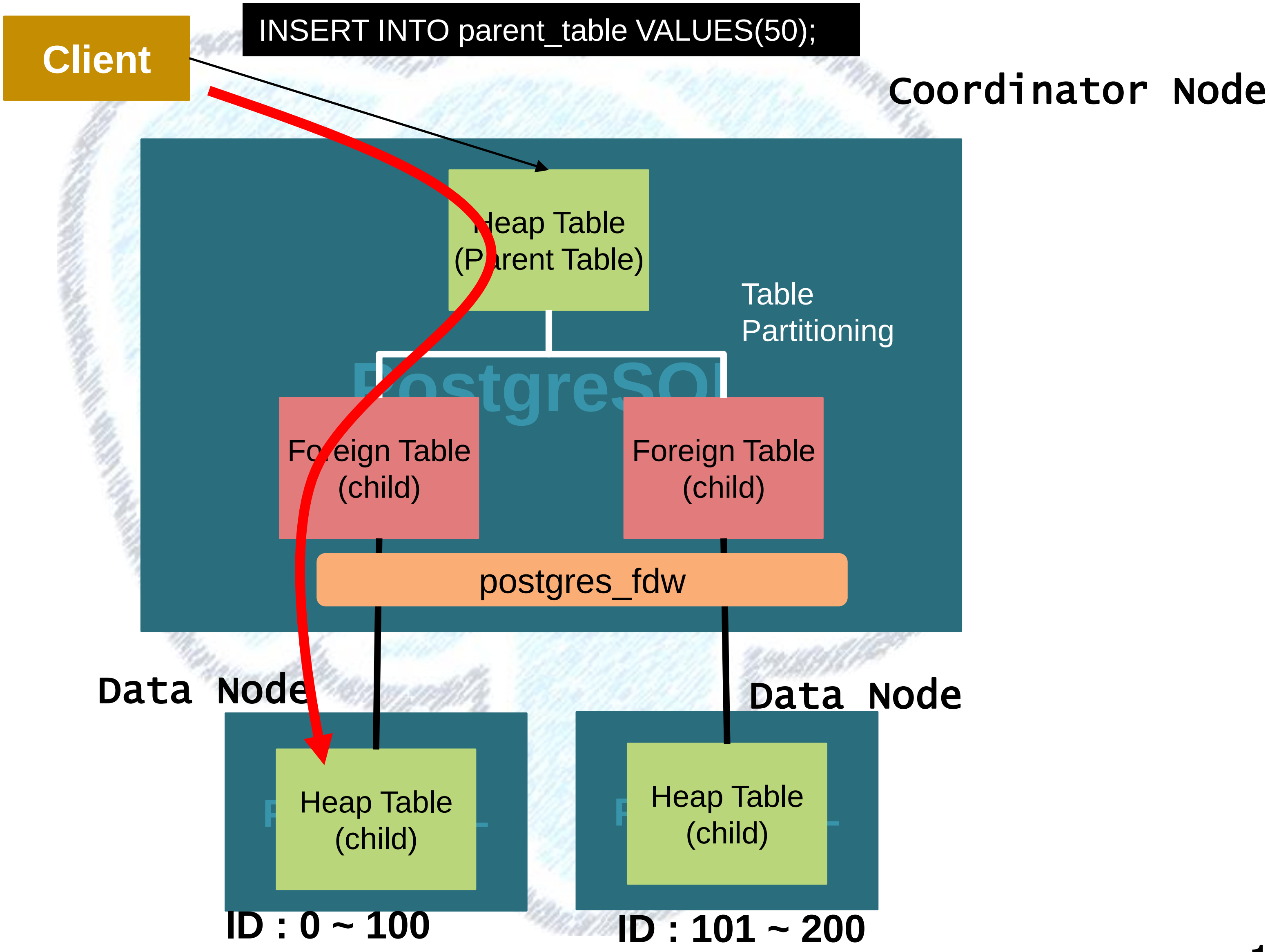
Multiple coordinator nodes (future)



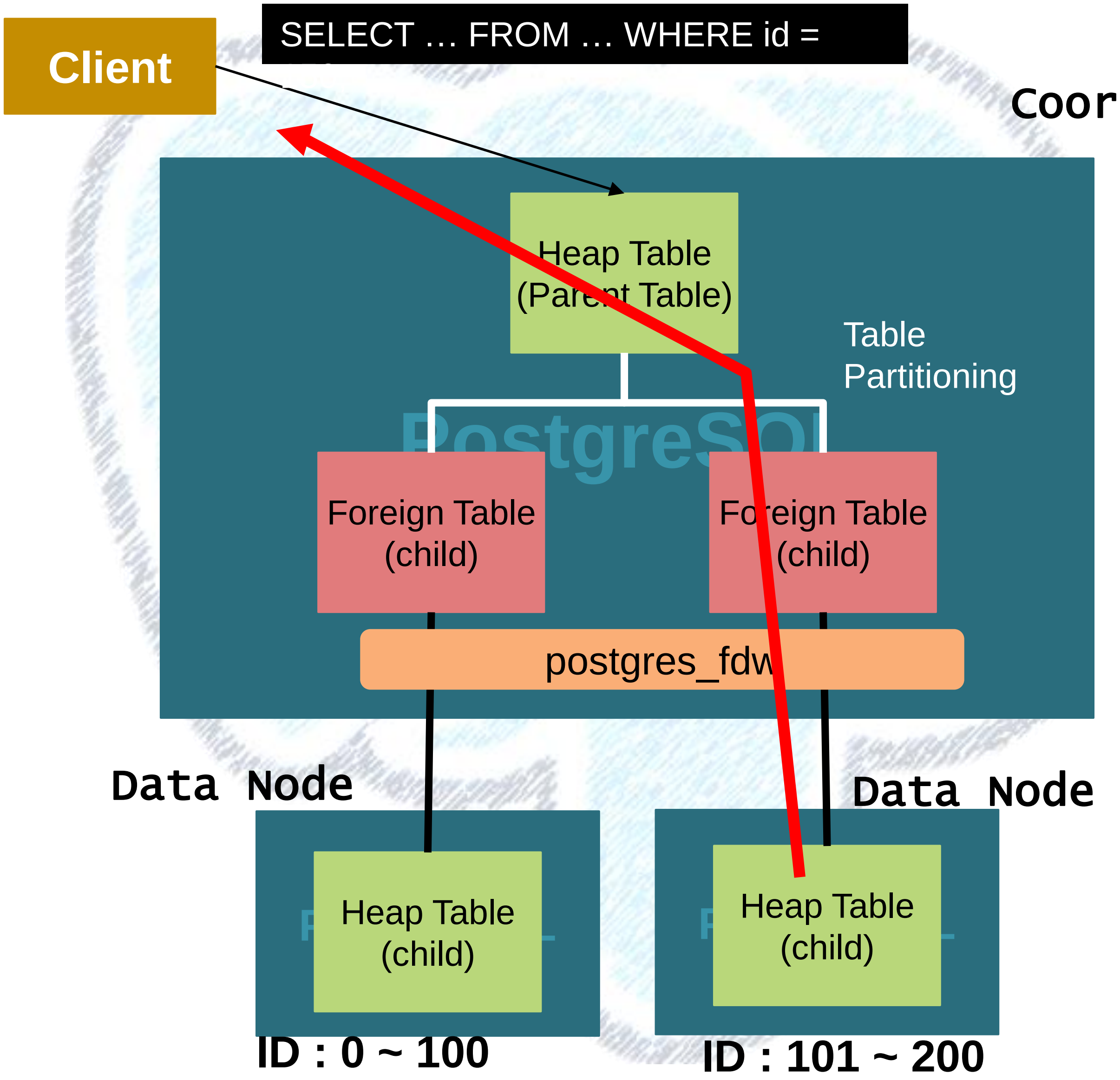
PostgreSQL server behaves both (future)



Insert data ID=50



Select data ID=150



Sort Push Down

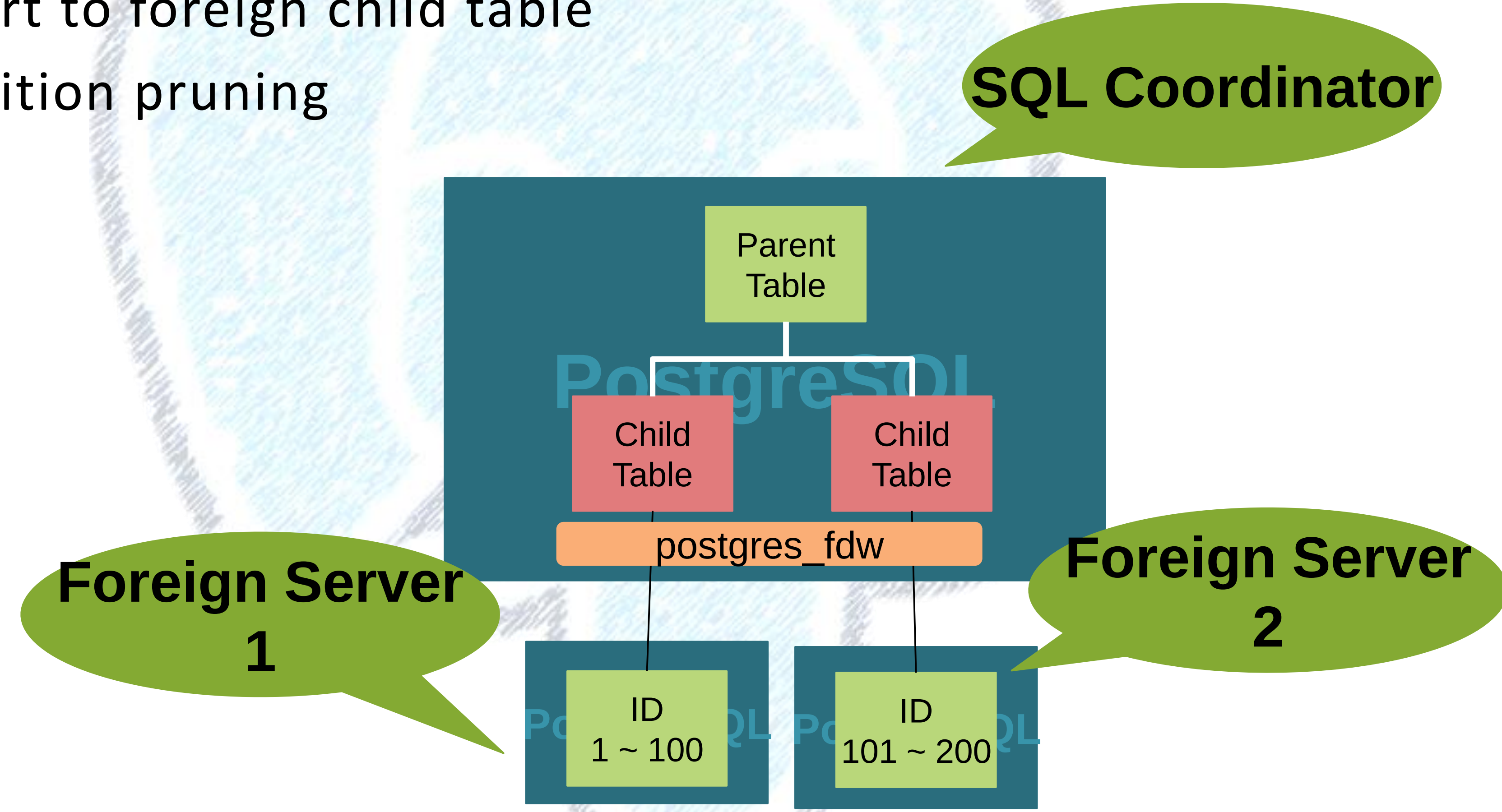
```
=# EXPLAIN (verbose on, costs off) SELECT * FROM p ORDER BY col;
```

```
-- 9.5
Sort
   Output: p.col
   Sort Key: p.col
   ->  Append
         ->  Seq Scan on public.p
               Output: p.col
         ->  Foreign Scan on public.s1
               Output: s1.col
               Remote SQL: SELECT col FROM public.s1
         ->  Foreign Scan on public.s2
               Output: s2.col
               Remote SQL: SELECT col FROM public.s2
```

```
-- 9.6
Merge Append
  Sort Key: p.col
  ->  Sort
        Output: p.col
        Sort Key: p.col
        ->  Seq Scan on public.p
              Output: p.col
  ->  Foreign Scan on public.s1
        Output: s1.col
        Remote SQL: SELECT col FROM public.s1 ORDER BY col ASC NULLS LAST
  ->  Foreign Scan on public.s2
        Output: s2.col
        Remote SQL: SELECT col FROM public.s2 ORDER BY col ASC NULLS LAST
```

Demonstration

- Using PostgreSQL 9.6.2
- Insert to foreign child table
- Partition pruning



FDW-based Sharding

- Transparent to the user
  - No need to modify application code
- No special DDLs for table management
  - same as local table partitioning
  - Can use multiple partitioning method; list, range (and hash)
- Horizontal partitioning
- Can support not only PostgreSQL shard node but also other source that corresponding FDW exists
- Coordinator node can be a shard node as well
- All features are Implemented as a generic feature
  - FDW features are useful on their own merit

Use cases

- PostgreSQL 9.6 can cover use cases where,
  - Frequent reads
  - The system requires write scale-out
  - Write single shard node in a transaction
    - If you don’t need transaction, you can do it with multiple server

Challenges and Key Techniques of FDW-based sharding

- More push down*
- Distributed query optimization*
- Asynchronous execution*
- Partitioning*
- Transaction support*
- Node registration
- High availability
- etc.

More push down

- Push-down makes distributed query execution more efficient
- What push down we can and can’t
  - Conditionals
  - data types, operators, function (including extension-provided)
  - Join, Sort, Aggregate(PG10+)
  - Grouping sets, window function aren’t yet
- Patches for PostgreSQL 10
  - “Push down more full joins in postgres_fdw” by Etsuro Fujita
  - “Push down more UPDATEs/DELETEs in postgres_fdw” by Etsuro Fujita
  - “postgres_fdw: support parameterized foreign joins” by Etsuro Fujita

postgres_fdw and distributed queries

Operation	PostgreSQL 9.5	PostgreSQL 9.6	PostgreSQL 10
SELECT	Foreign table pruning	Foreign table pruning	Foreign table pruning
Conditionals	Push down	Push down	Push down
Aggregations	Local	Local	**Push down**
Sorts	Local	Push down	Push down
Joins	Local	Push down (Left, Right, Full)	**Push down* (Left, Right, Full)**
UPDATE, DELETE	Tuple based using CURSOR	Directly execution	**Directly execution* (with joins)**
INSERT	INSERT to remote server using Prepare/Execute	INSERT to remote server using Prepare/Execute	INSERT to remote server using Prepare/Execute

Partitioning

- Need declarative partitioning
  - Committed basic infrastructure and syntax to PostgreSQL 10!

- Still missing building blocks
  - Tuple routing feature
    - doesn’t support insert foreign partitioned table so far
  - Executor improvement
  - Global unique index

Asynchronous Execution

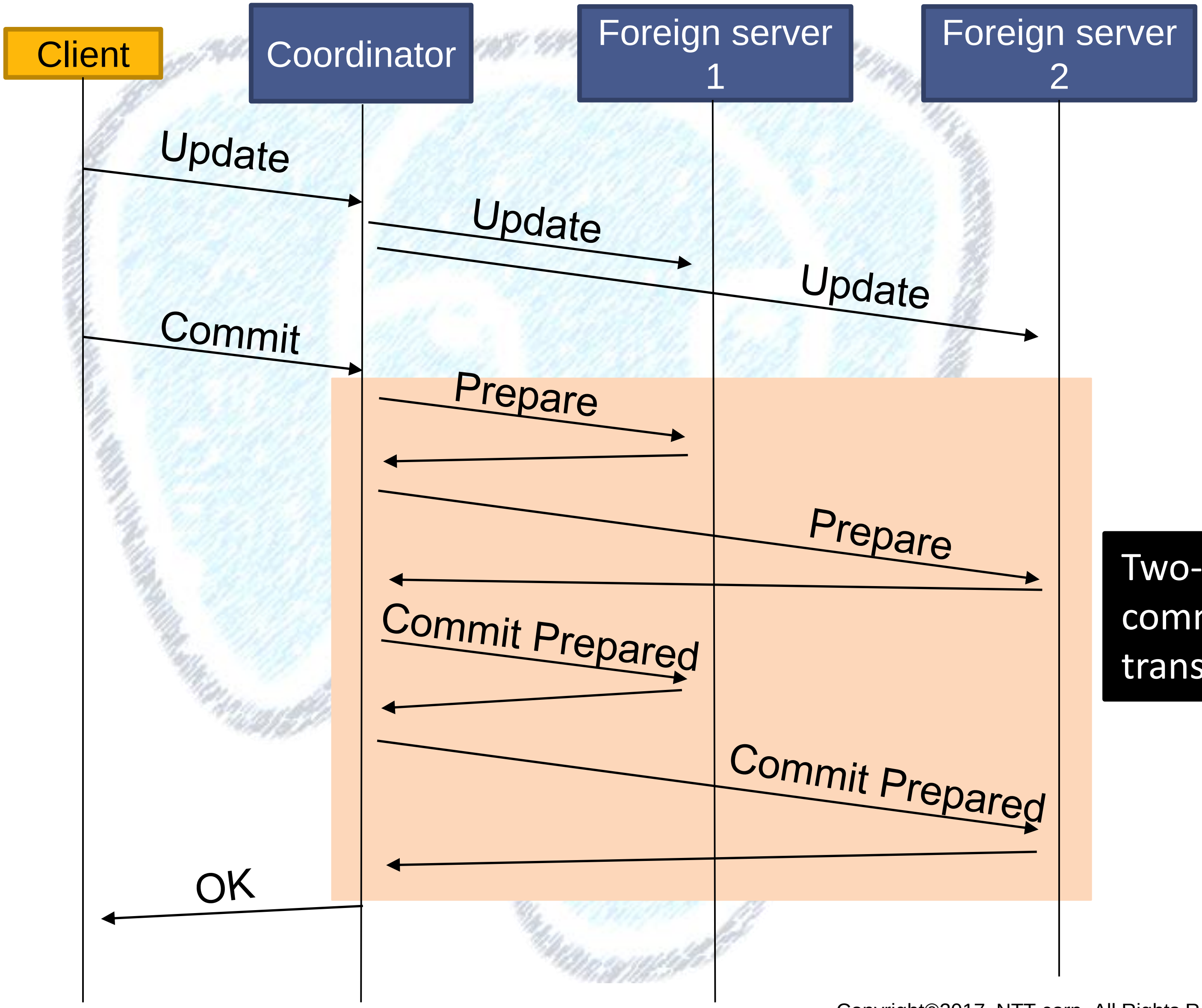
- Executor improvement
- Data fetching request to different site can be sent asynchronously
- Improves foreign table scanning performance
- Patch
  - Under discussion
  - “Asynchronous execution for postgres_fdw” by Kyotaro Horiguchi

Distributed Transaction Management

- Provide cluster-wide transaction (ACID)
  - Atomic commit

- Under reviewing
  - Transaction involving multiple foreign servers commits using two-phase-commit protocol
  - Patch
    - “Transactions involving multiple postgres foreign servers” by Masahiko Sawada, Ashutosh Bapat

Processing Sequence of 2PC on FDW



Two-phase commit is used transparently.

Use cases with PostgreSQL 10

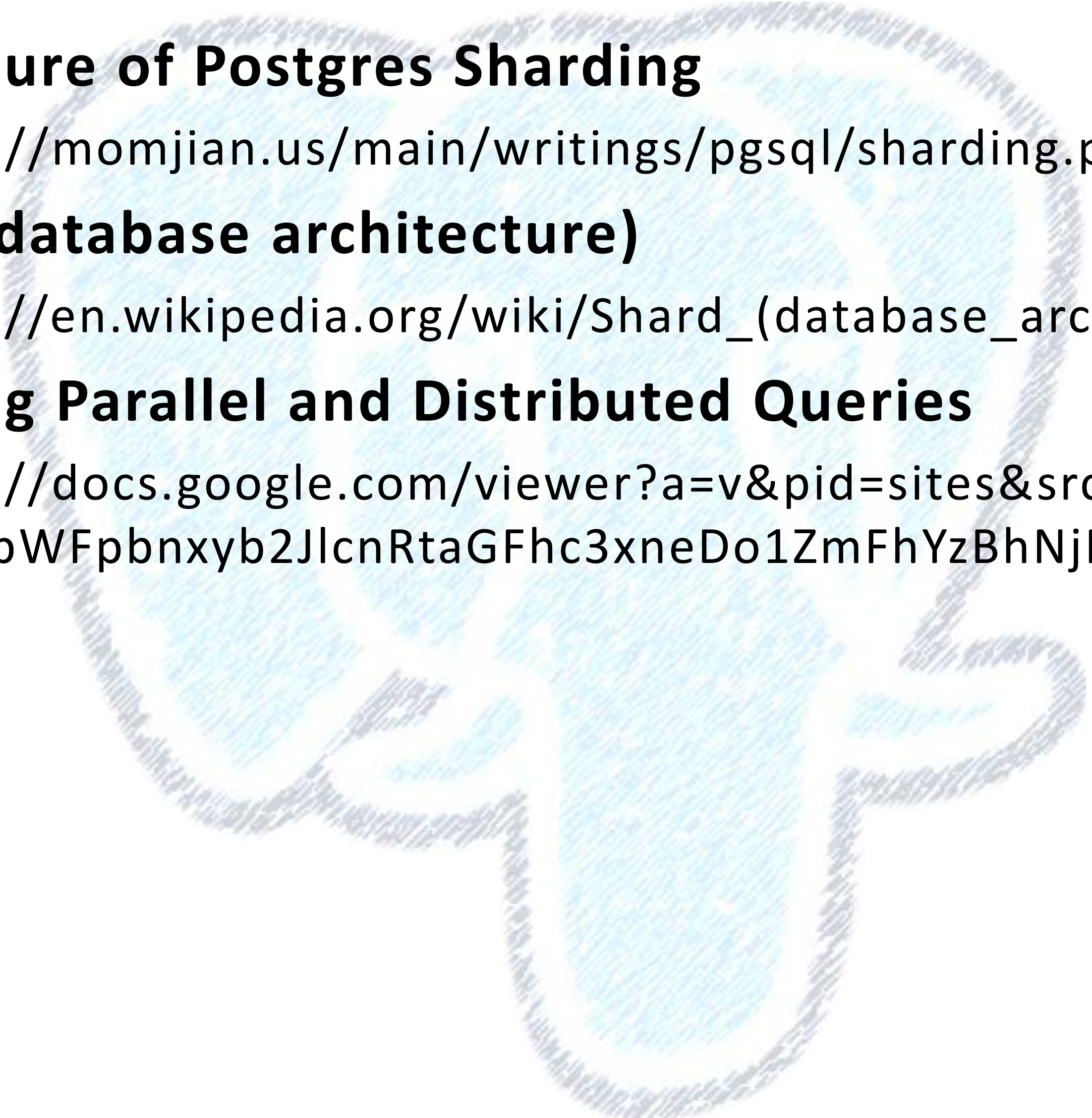
- Business report (complex query analyzing large data)
  - By aggregation pushdown, optimizer improvement
- Update partition key across nodes atomically
  - By atomic commit distributed transaction

Conclusion

Conclusion - Keep challenging -

- FDW-based sharding brings us a native PostgreSQL scale-out solution
- A lot of work in-progress building blocks
- Do we really need it?
  - To expand the applicability to more critical system
  - Each sharding feature improves PostgreSQL generically
- More detail of FDW-based sharding,
  - https://wiki.postgresql.org/wiki/Built-in_Sharding

References



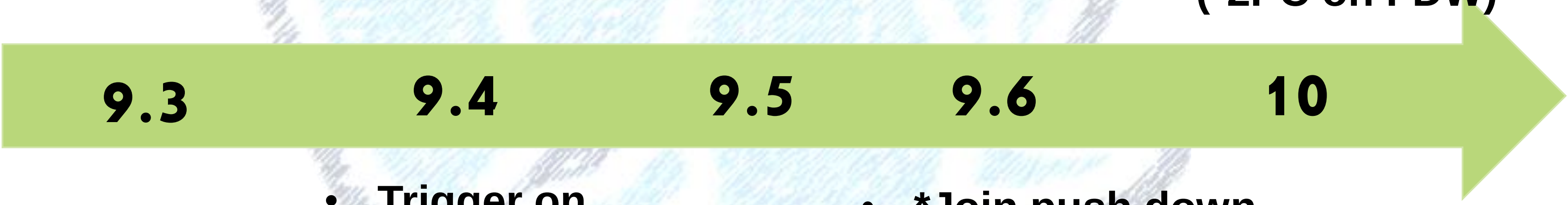
- **The Future of Postgres Sharding**
  - https://momjian.us/main/writings/pgsql/sharding.pdf
- **Shard (database architecture)**
  - https://en.wikipedia.org/wiki/Shard_(database_architecture)
- **Planning Parallel and Distributed Queries**
  - https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxyb2JlcnRtaGFhc3xneDo1ZmFhYzBhNjNhNzVhMDM0

Thank you
Спасибо

Masahiko Sawada
sawada.mshk@gmail.com

FDW features

- NTT has been developing feature related to FDW-based sharding since PostgreSQL 9.3, with the knowledge obtained through the development Postgres-XC.



- *Introduce postgres_fdw
- *Write via FDW

- Trigger on Foreign table

- *Foreign table inheritance

- *Join push down
- *Sort push down
- *Direct perform UPDATE and DELETE
- Extension-provided operator push down

- *Partitioning
- Aggregate push down
- (*Async execution)
- (*2PC on FDW)