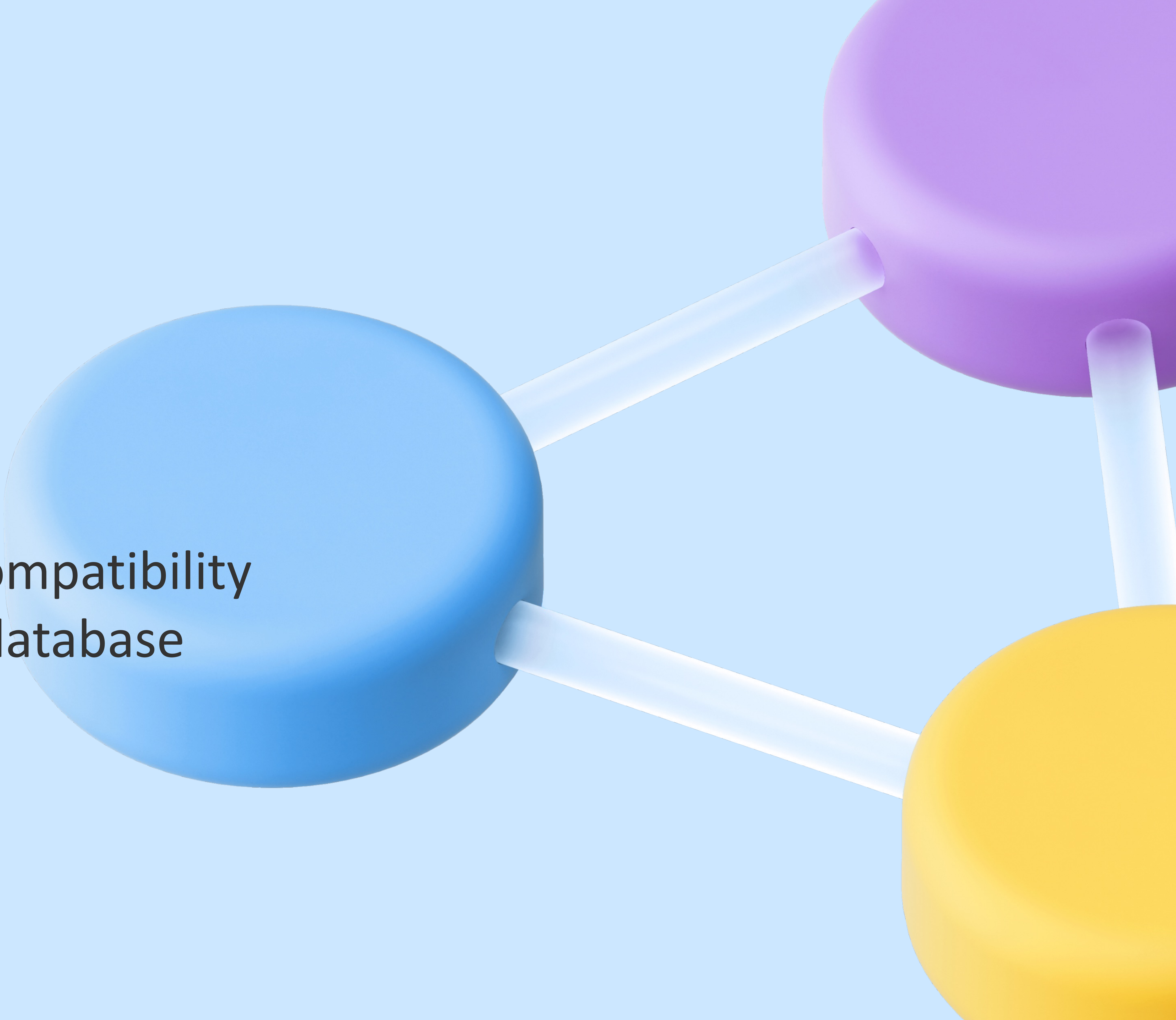
YDB



YDB

Adding PostgreSQL compatibility to a Distributed SQL database

Timofey Koolin, senior developer

Timofey Koolin

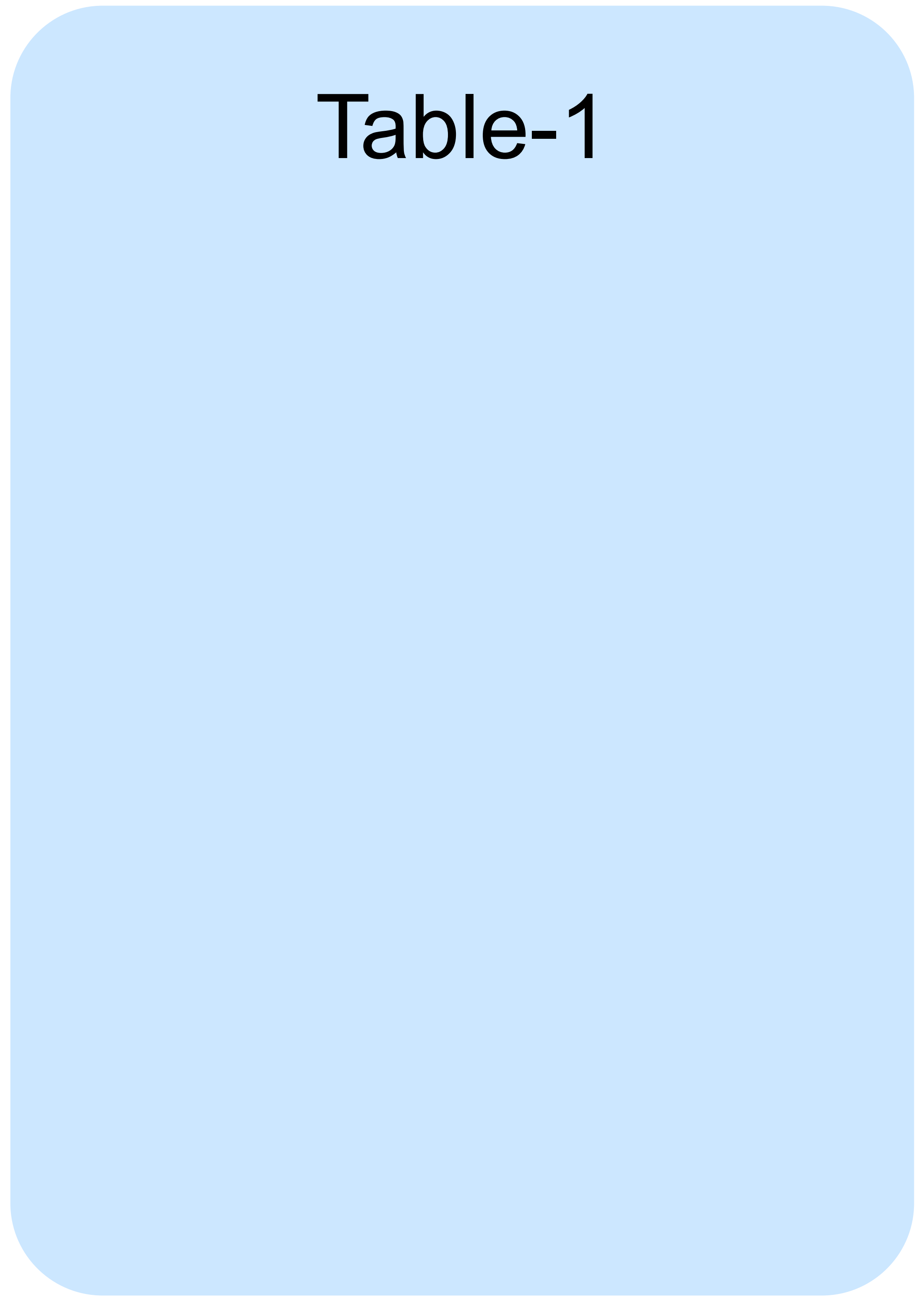
- Senior developer with >10 years experience
- PG compatibility researcher and YDB drivers developer

Contents

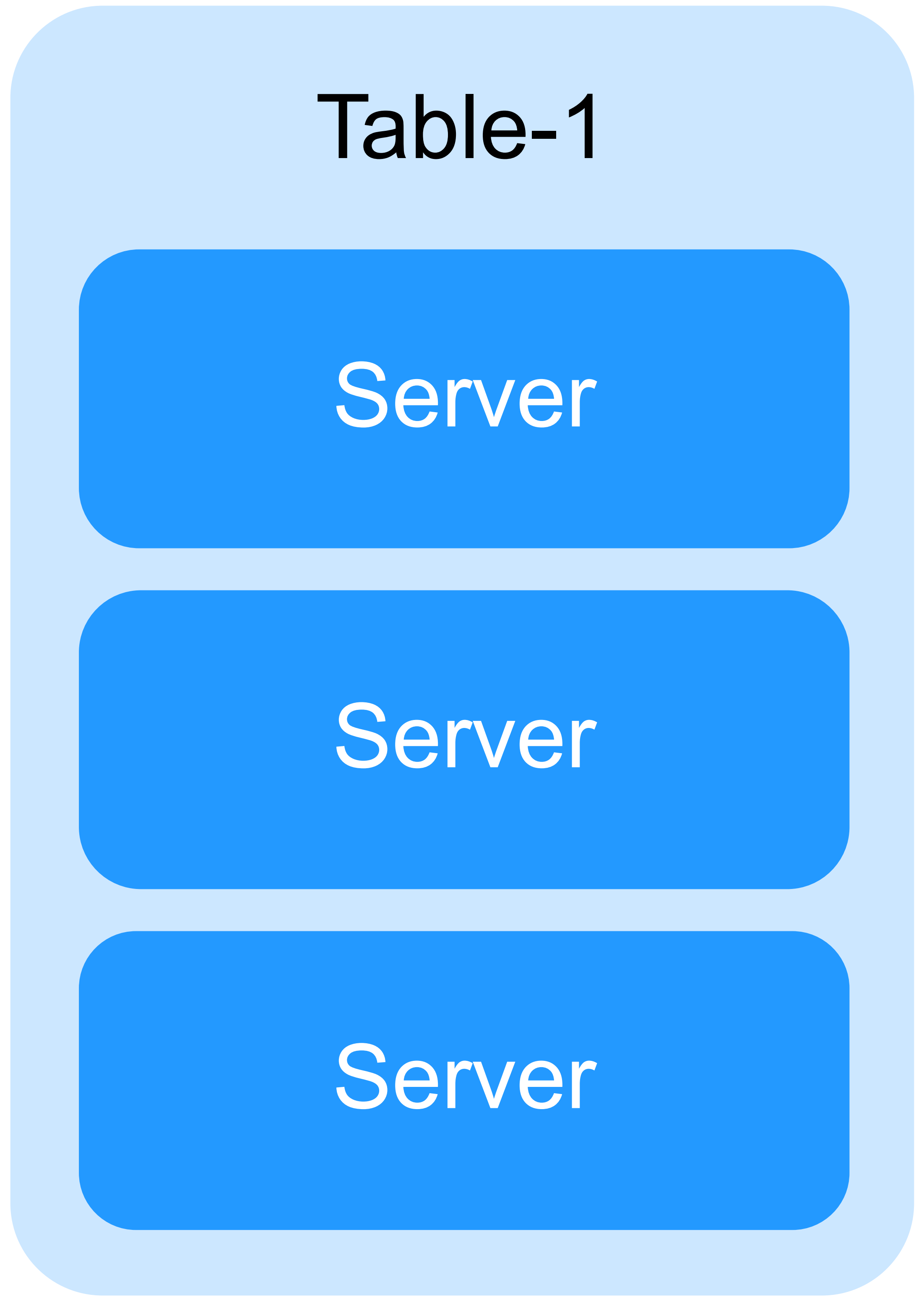
The short history of YDB

- 2014 - started as an inhouse infrastructure technology
- 2020 - provided as a managed service for a few regions
- 2022 - published to open source under Apache 2.0 license
- 2023 - started working on PG compatibility

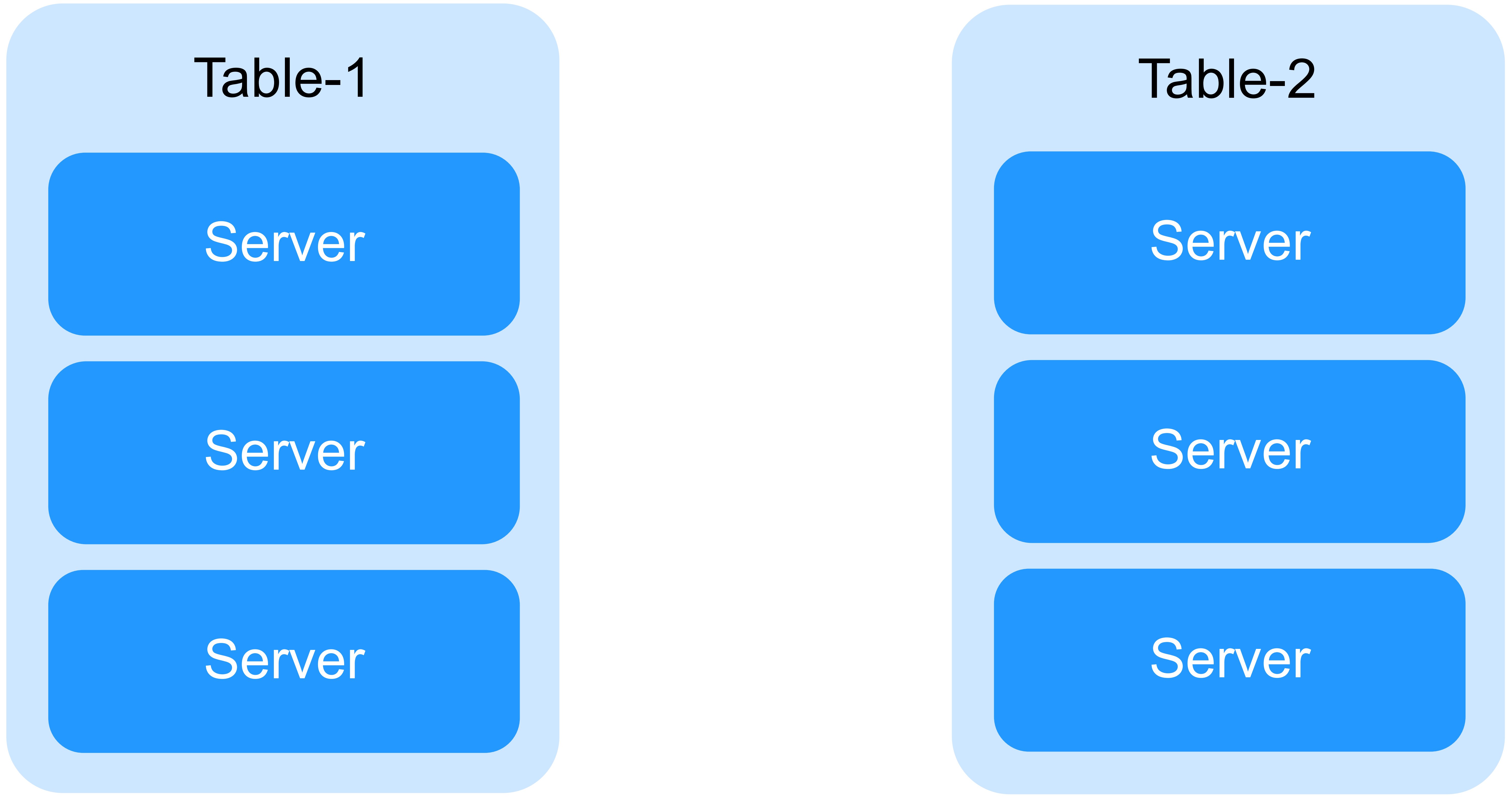
YDB is a distributed DB



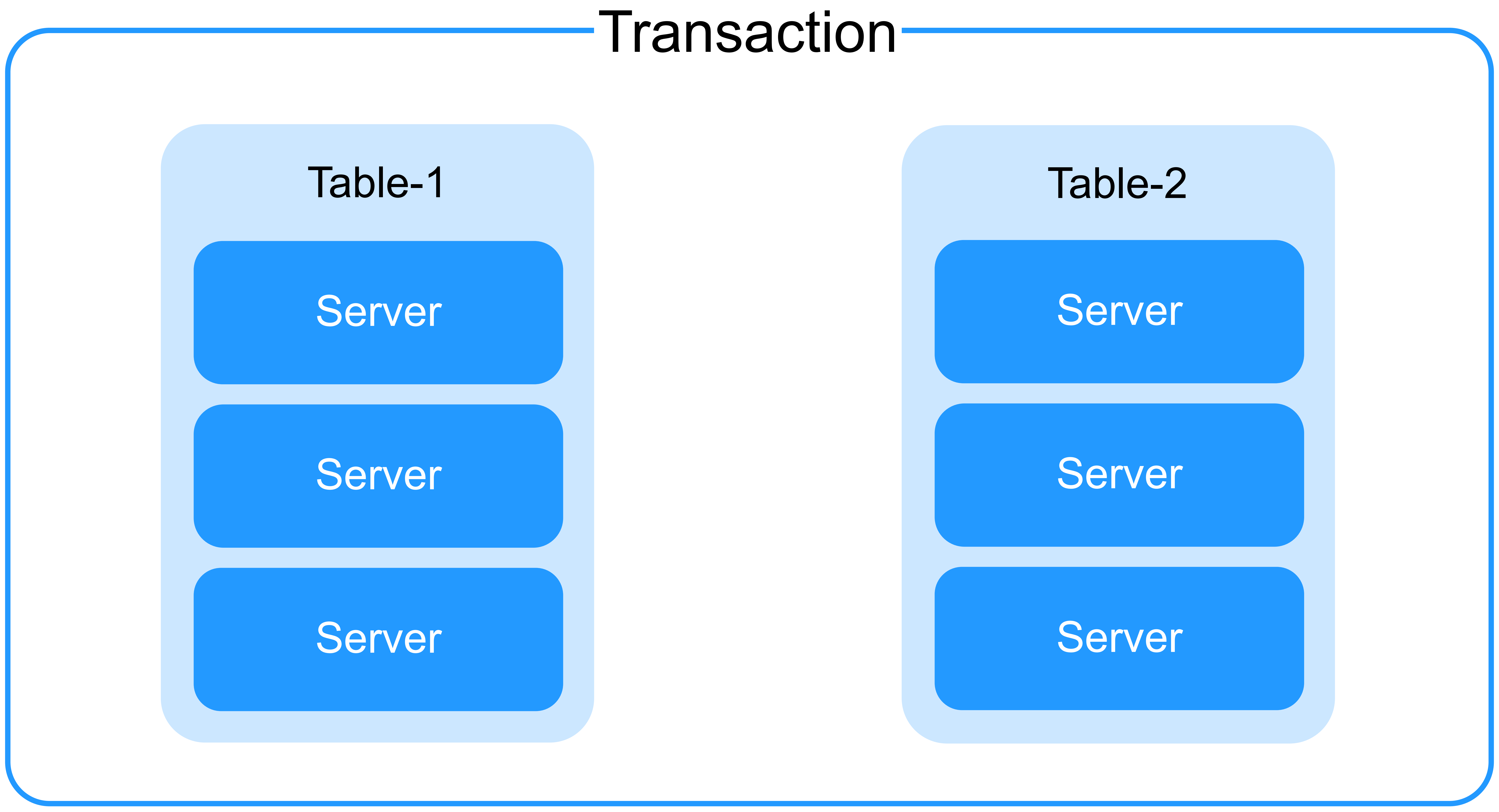
YDB is a distributed DB



YDB is a distributed DB



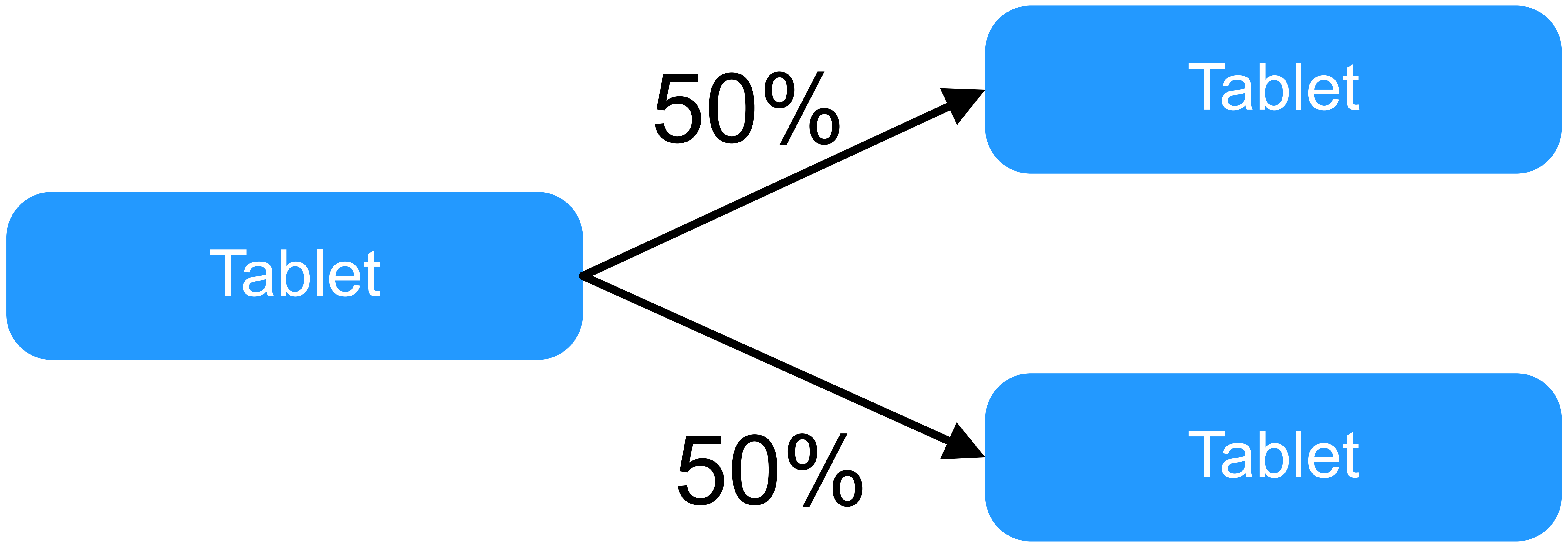
YDB is a distributed DB



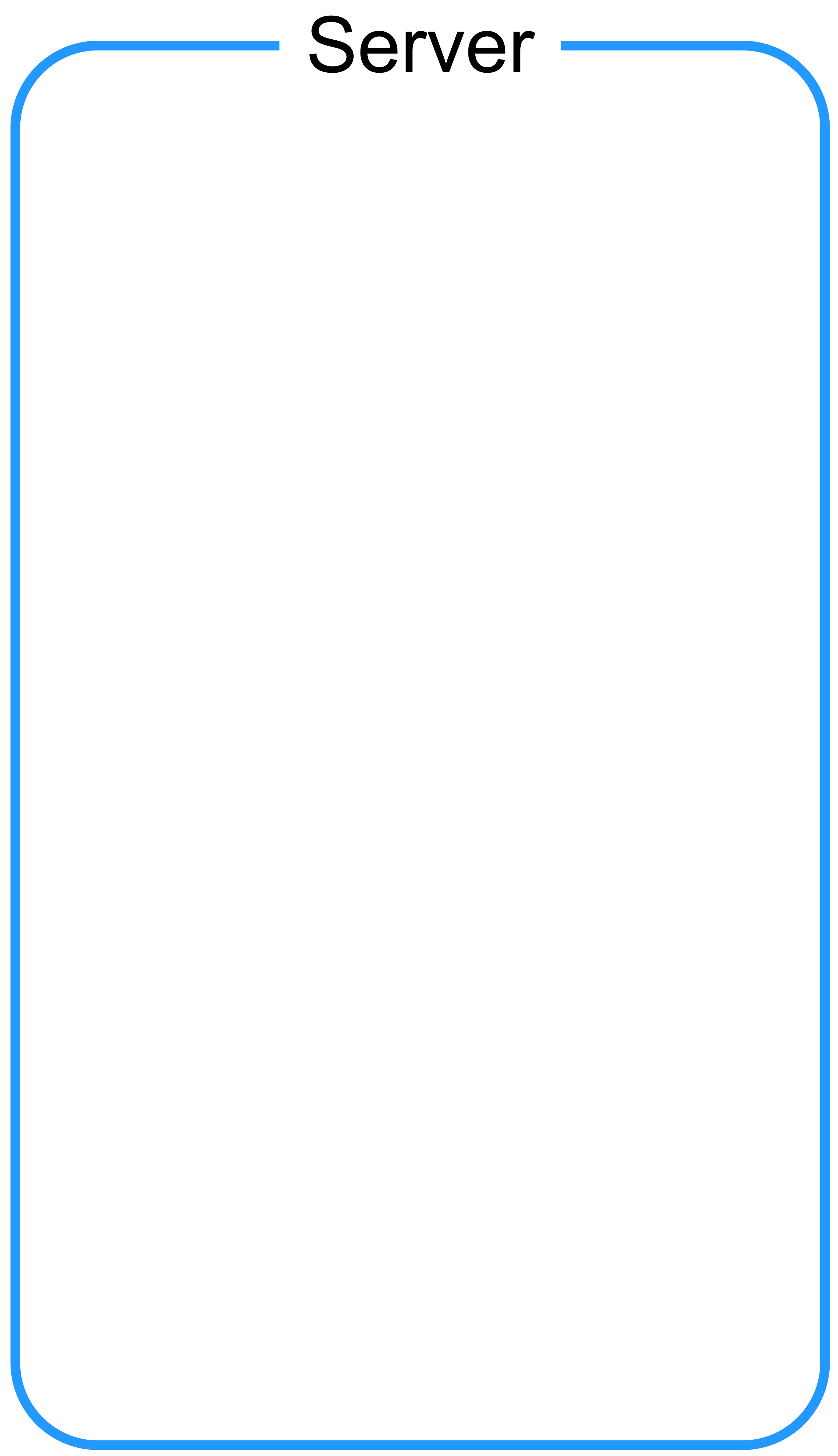
YDB Tablet

- Actor – lightweight thread
- Minimal scale/HA block
- Has own HA mini-database
- Handles a specific function
  - Row-oriented store
  - Column-oriented store
  - Message queue
  - …
- Handles a small piece of data and workload

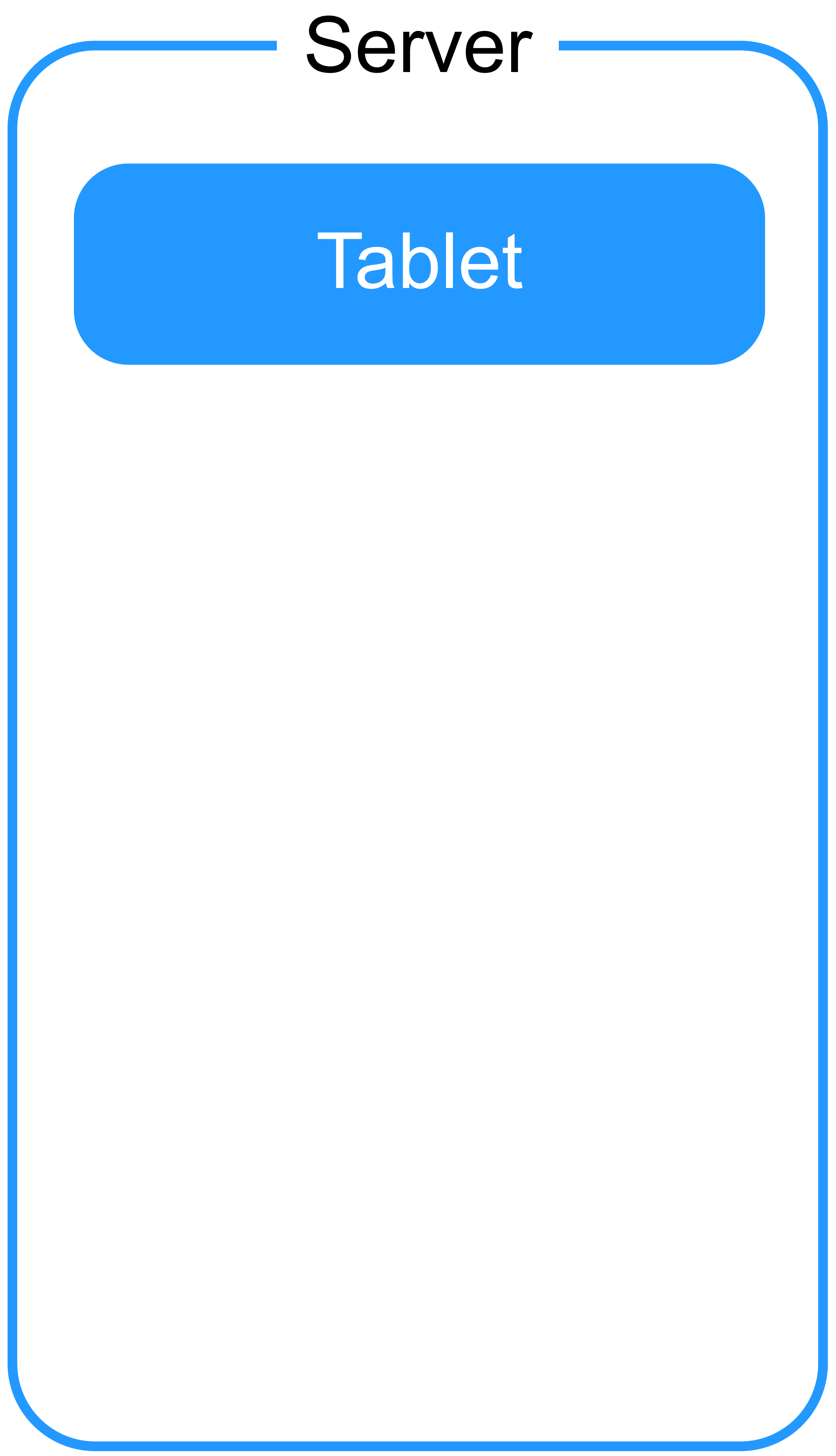
Tablet splitting by size or load



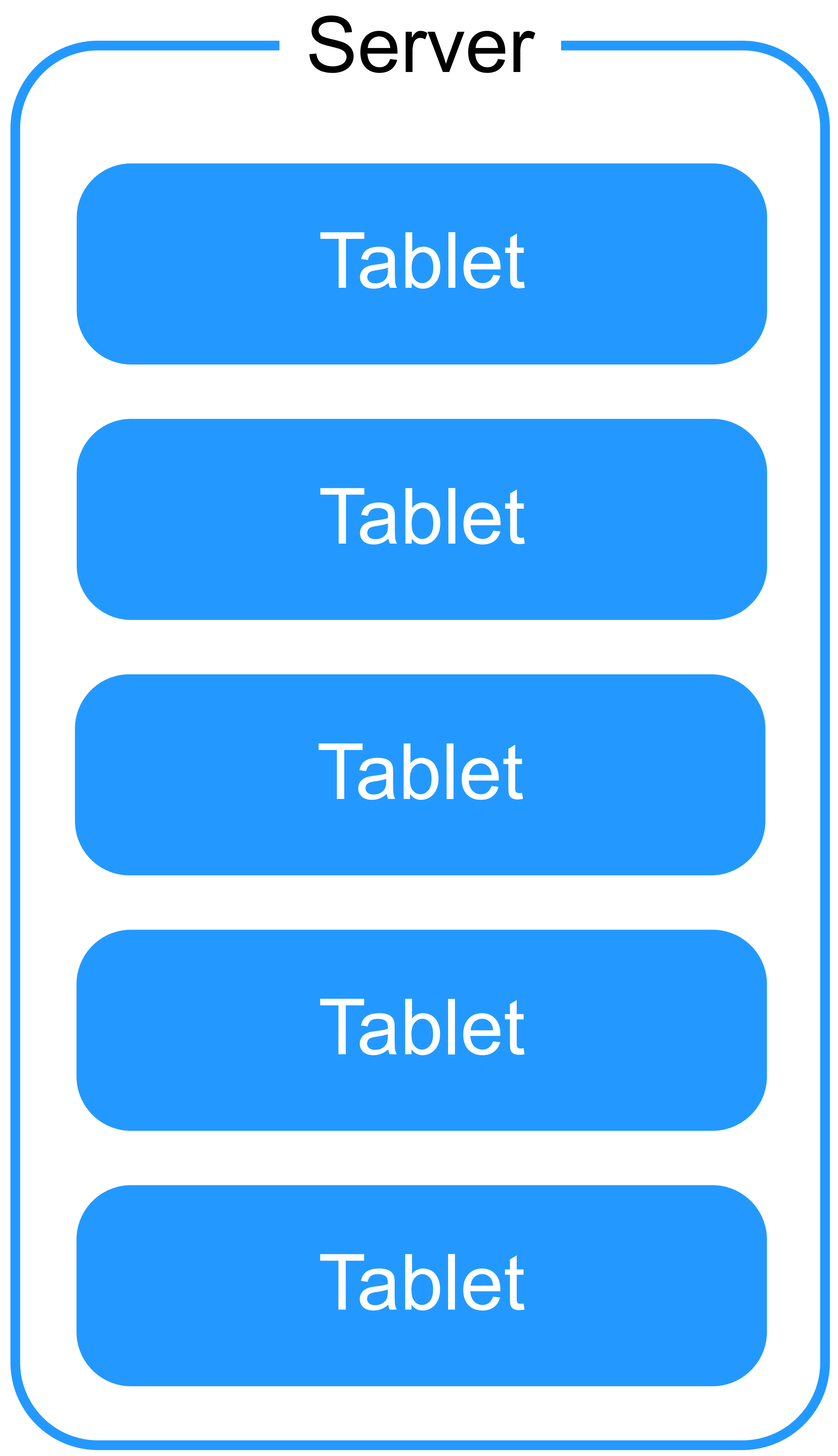
YDB architecture



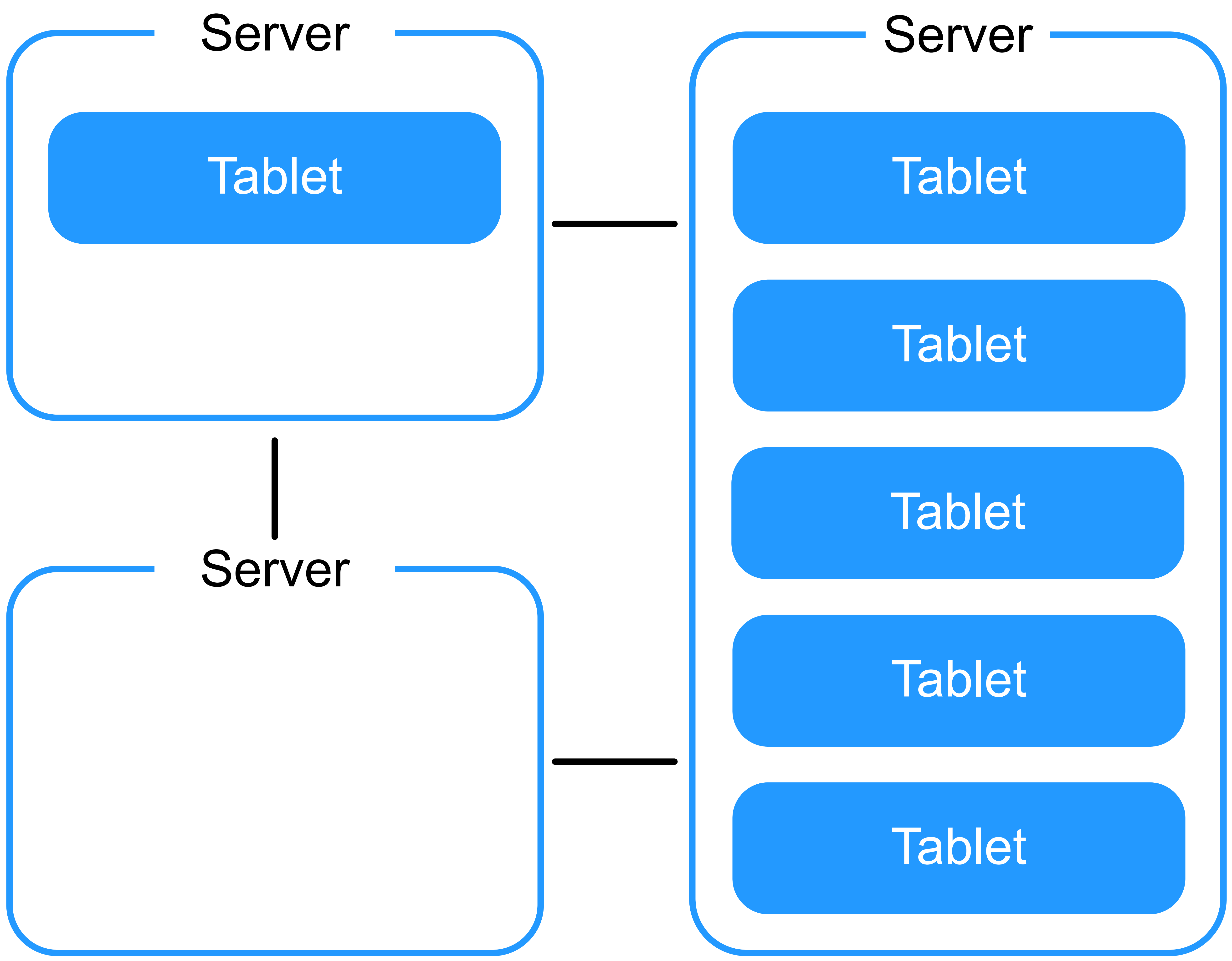
YDB architecture



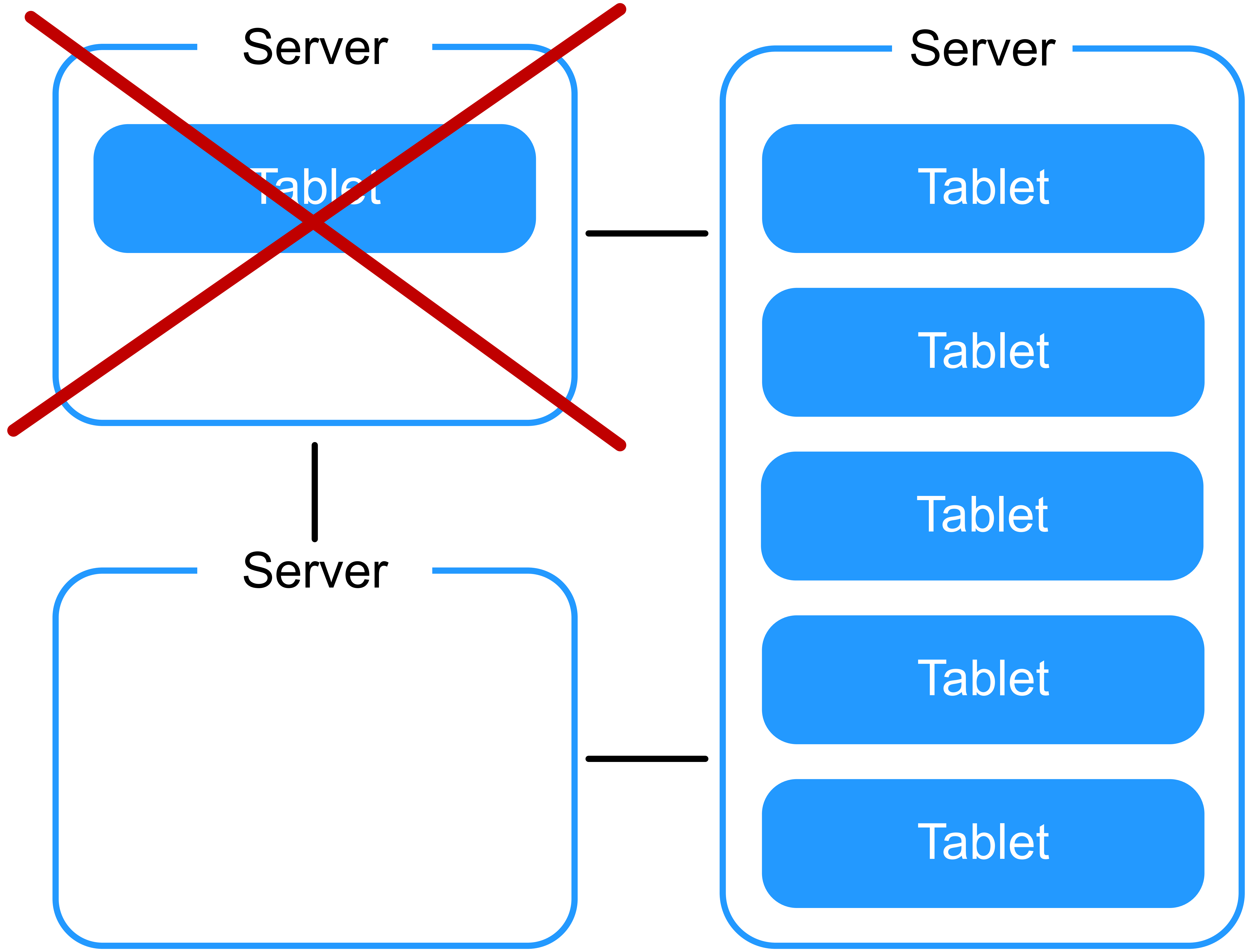
YDB architecture



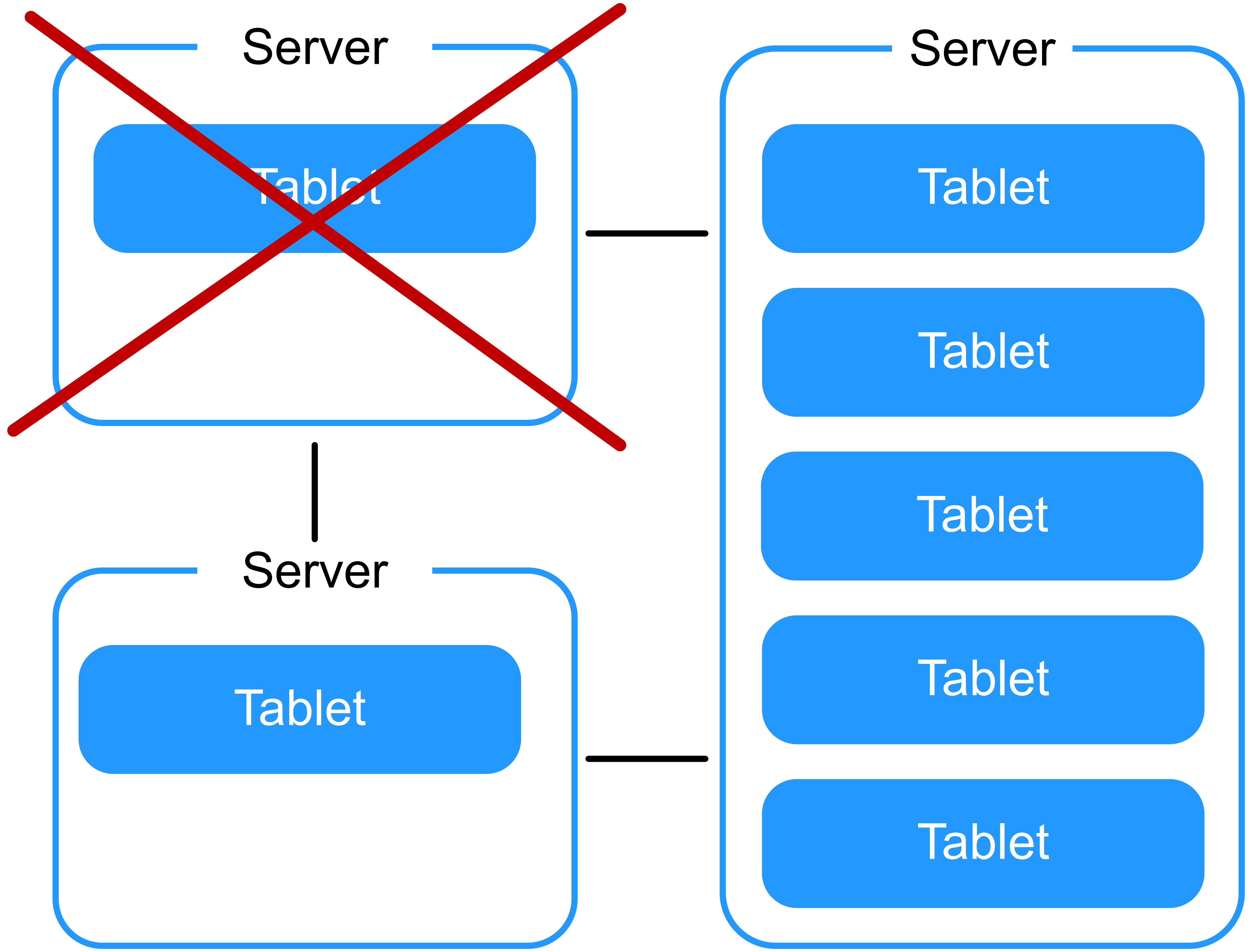
YDB architecture



YDB architecture



YDB architecture



Postgres compatibility goals

- Make it possible to use existing open source toolset
- Reuse distributed YDB query engine for:
  - High availability
  - Strong consistency
  - Scalability limited by budget
- Interoperability between YDB and Postgres layers

Implement everything from scratch

Advantages

- Design implementation for distributed environment
- Freedom to optimize algorithms

Disadvantages

- A lot of work to reimplement all features
- Need to reimplement all PG features for future PG releases
- Hard to mimic all corner cases

Use full Postgres runtime

Advantages

- Best runtime compatibility
- Relatively easy PG release upgrade
- Limited extension support

Disadvantages

- Limited by Postgres runtime capabilities
- Need to maintain a PG fork

Write a PG extension

Advantages

- Uses native PG extensions
- Easy to upgrade PG release

Disadvantages

- Limited extension points
- Limited by PG runtime capabilities

YDB way:
Best of two worlds

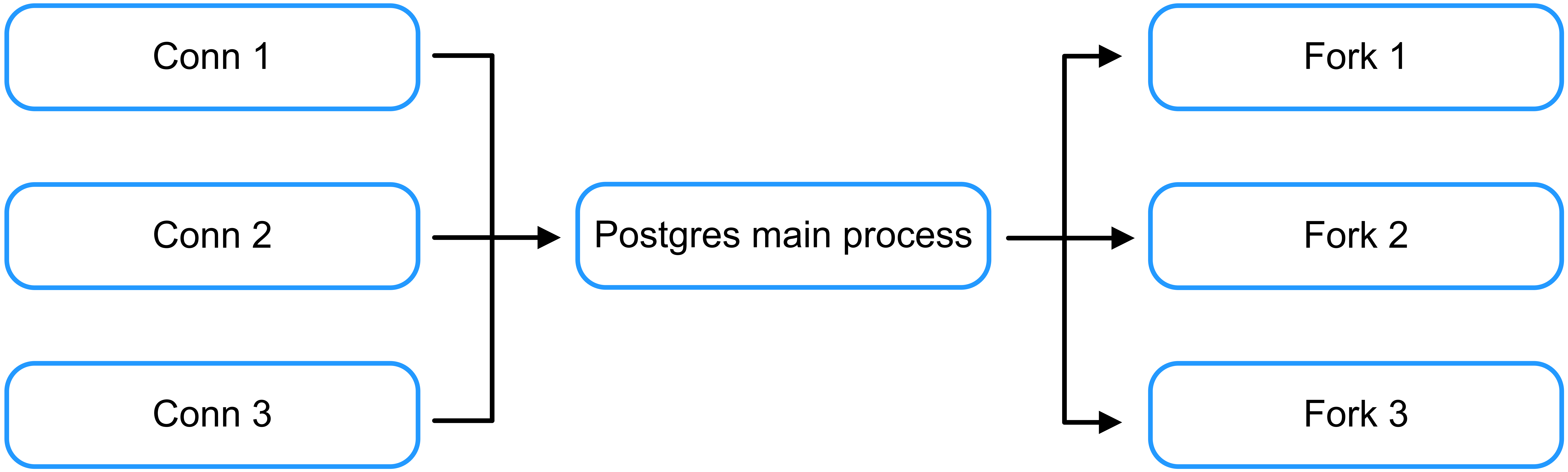
Advantages

- Reuse YDB distributed transactions, executor, optimizer
- Reuse PG query parser
- Allow to call native PG functions
- Interoperability of PG and YDB workloads

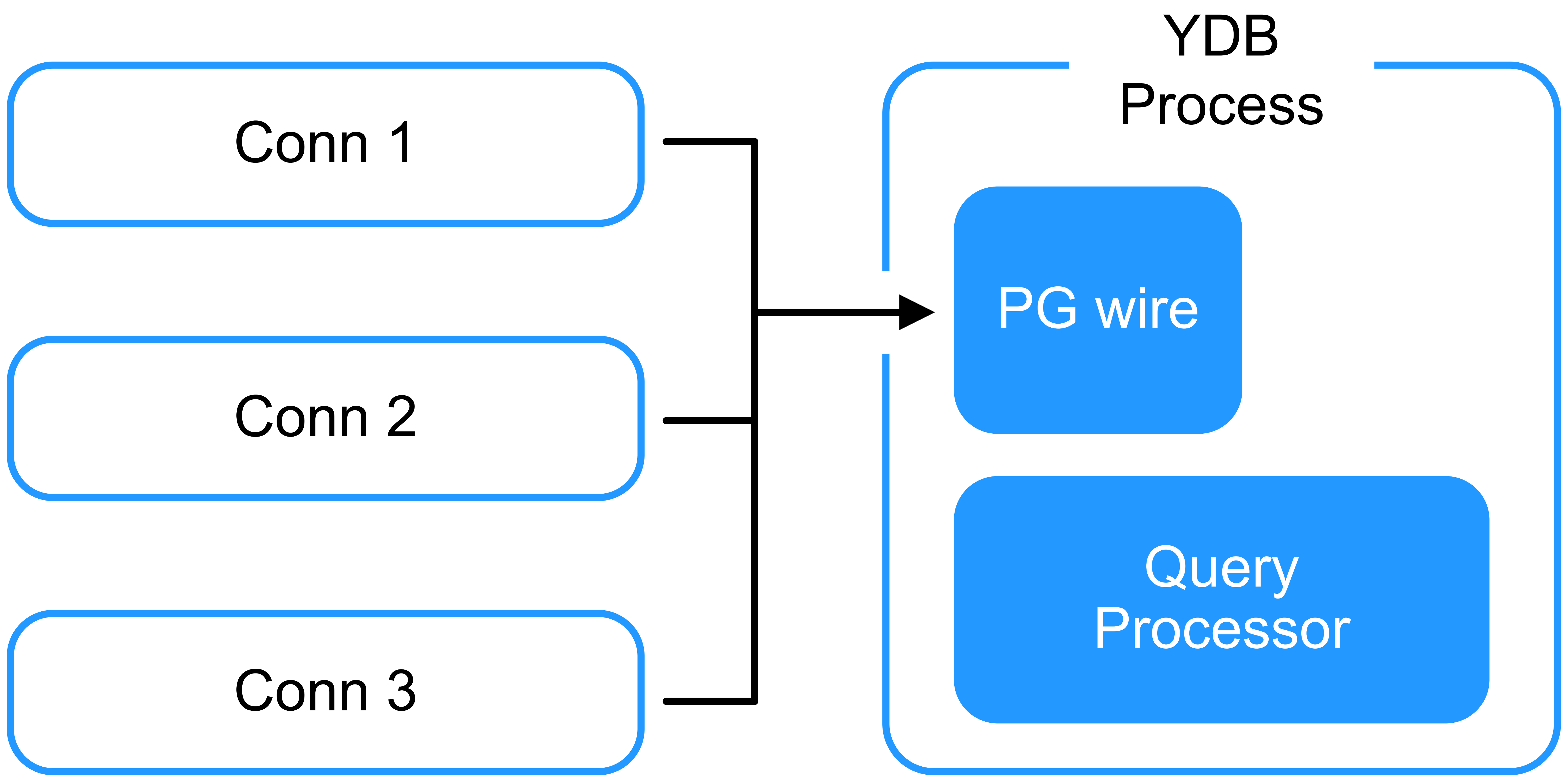
Disadvantages

- A lot of work for integration
- Moderate complexity of PG release upgrade

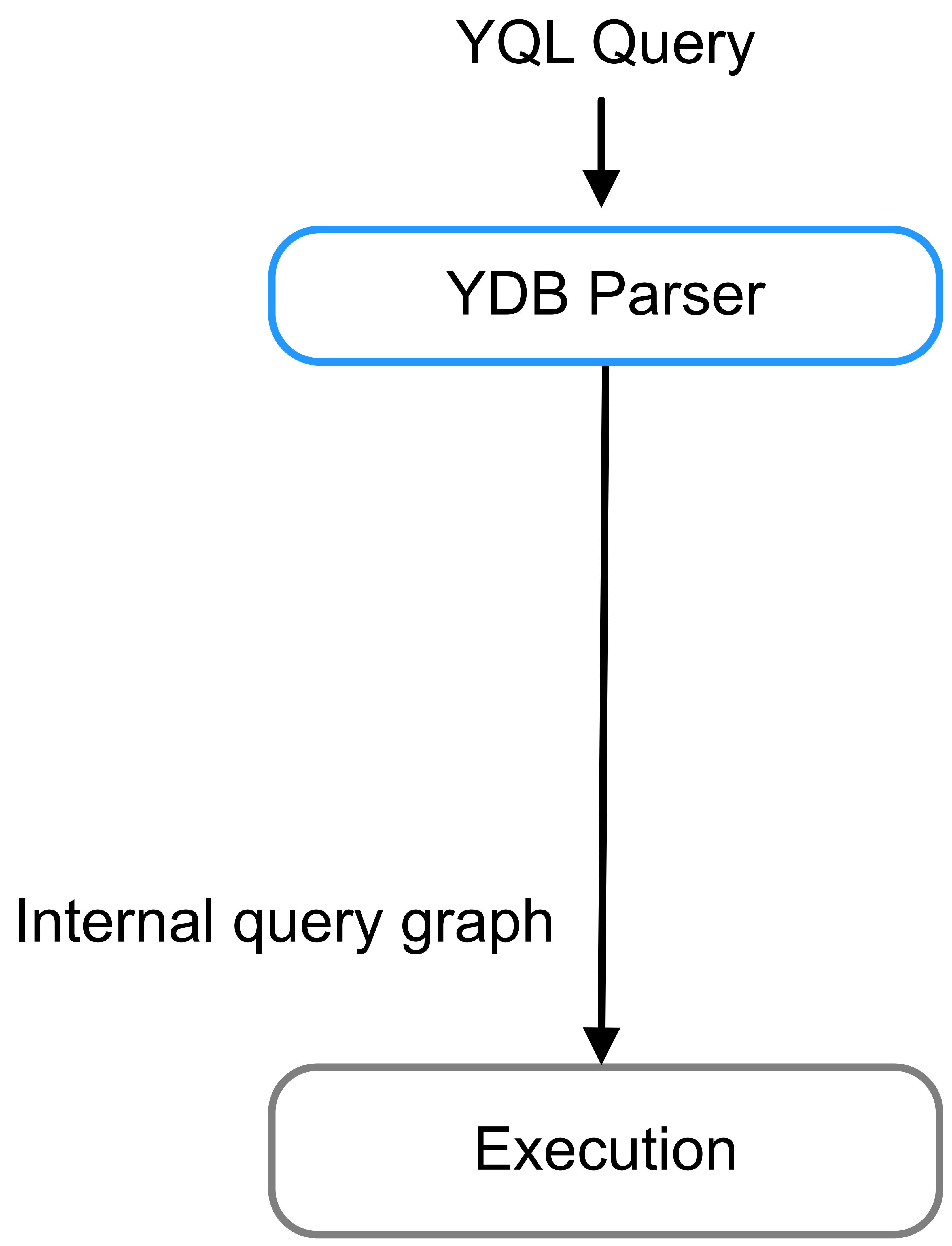
Postgres connection handling



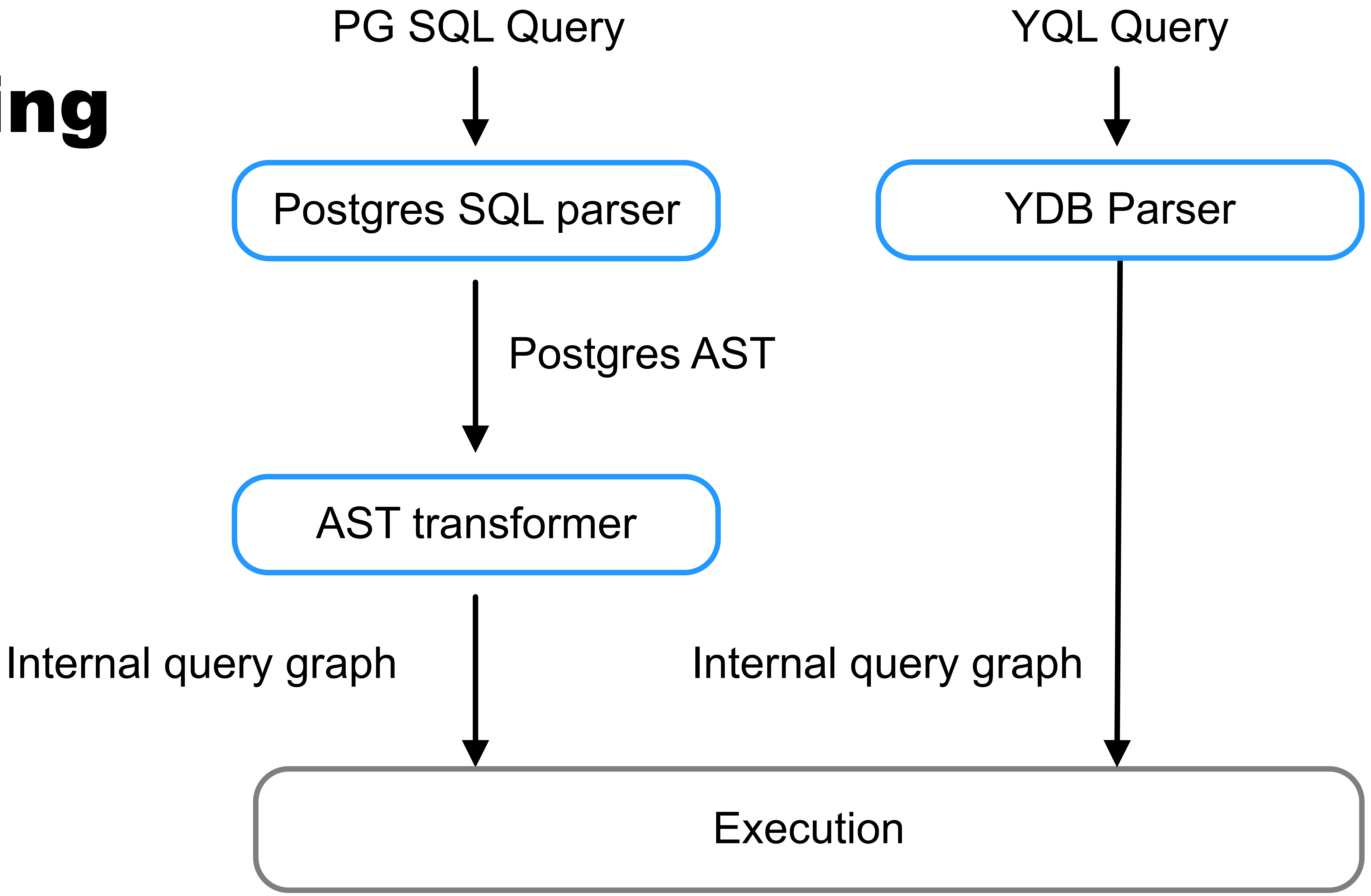
YDB PG Wire



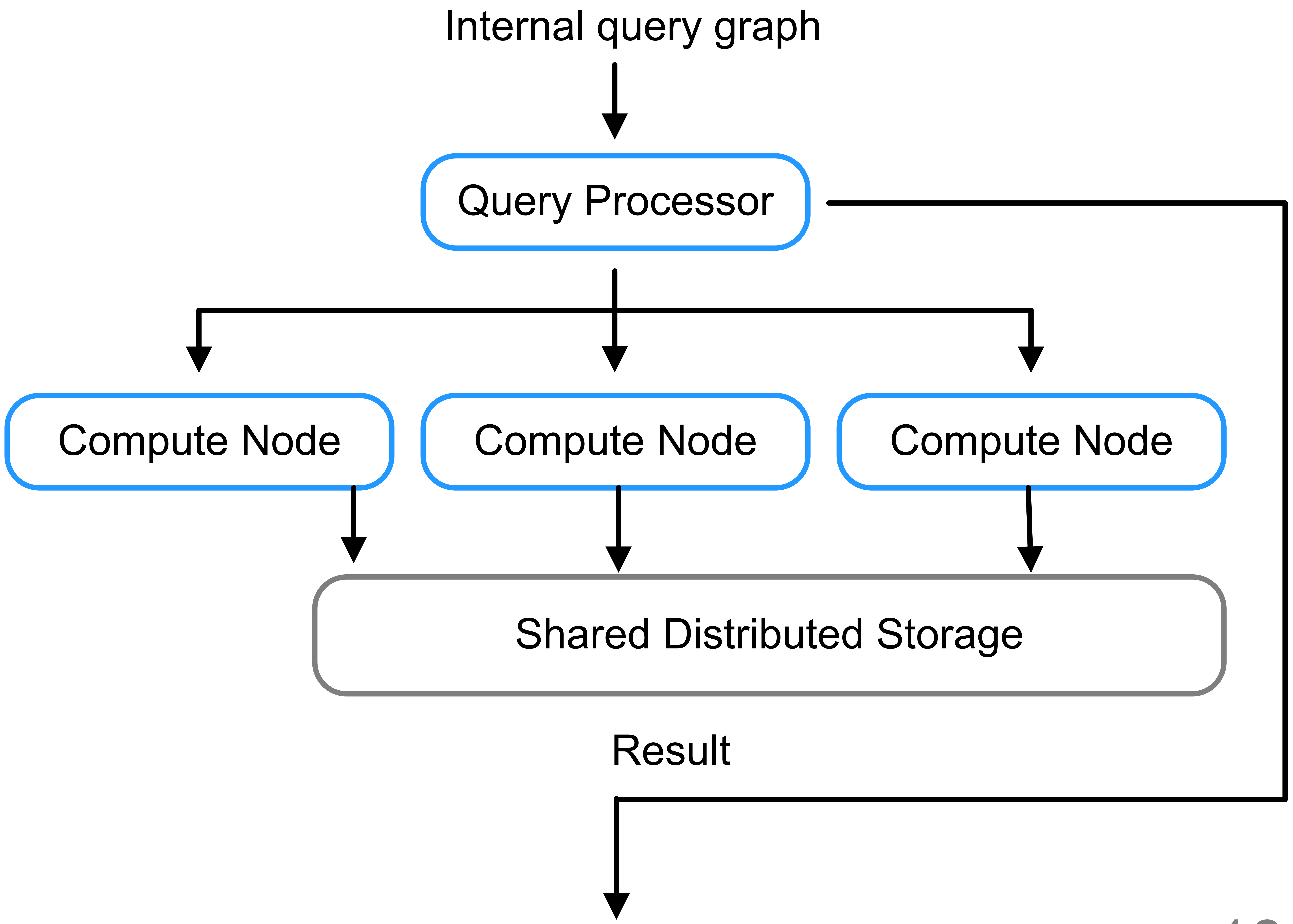
YQL query handling



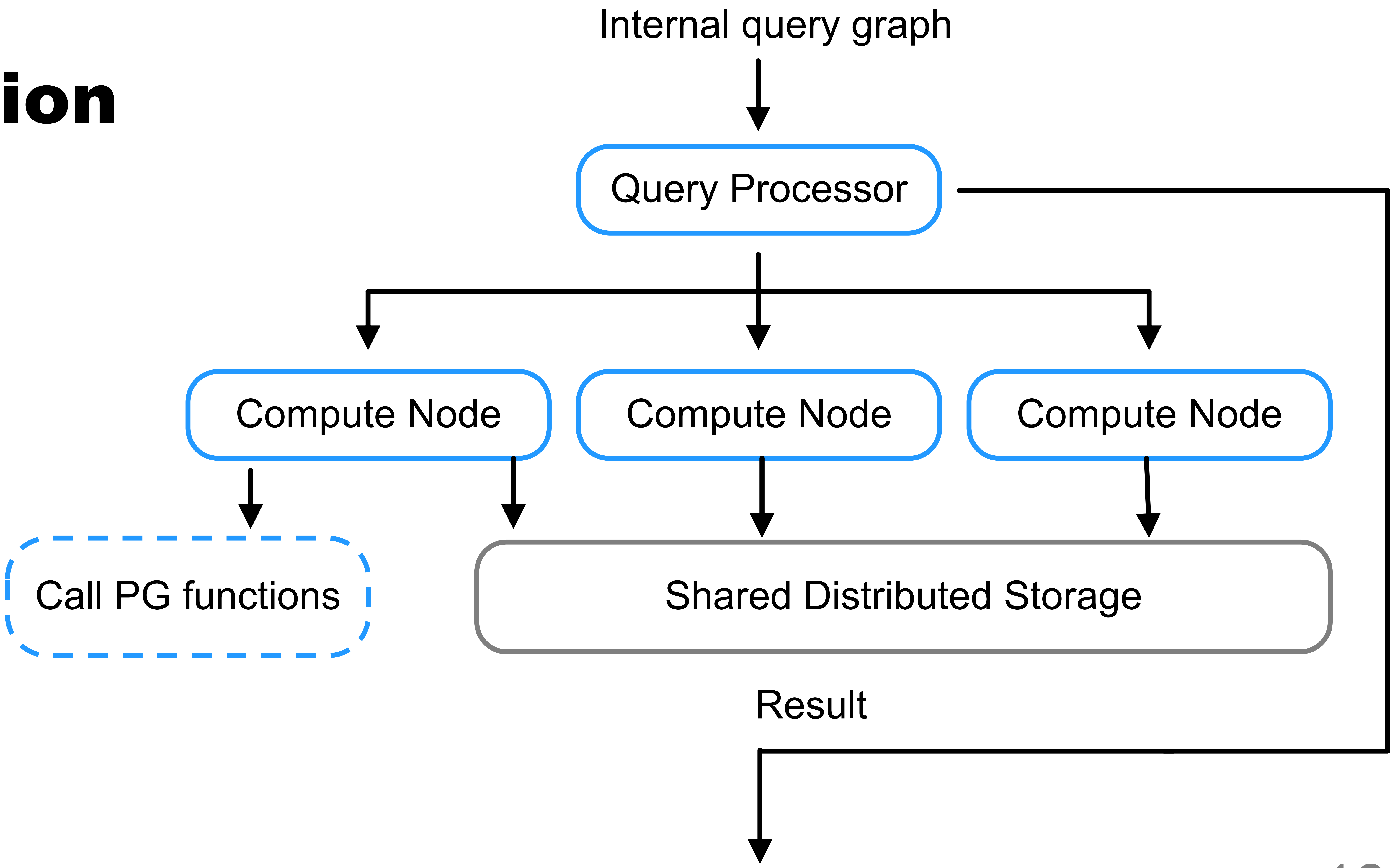
PG SQL query handling



Query execution



Query execution



Testing PG compatibility

Testing PG compatibility

- Postgres regression tests

Testing PG compatibility

- Postgres regression tests
- Documentation based tests

Testing PG compatibility

- Postgres regression tests
- Documentation based tests
- Drivers integration tests

Testing PG compatibility

- Postgres regression tests
- Documentation based tests
- Drivers integration tests
- Applications

YDB

Q&A

Timofey Koolin
Senior developer

bitly

https://ydb.tech

Testing PG compatibility

Current

- Postgres regression tests
- Documentation based tests
- Drivers integration tests
- Applications

Future

- Applications tests
- Benchmarks
- Fuzzing