2ndQuadrant®
PostgreSQL

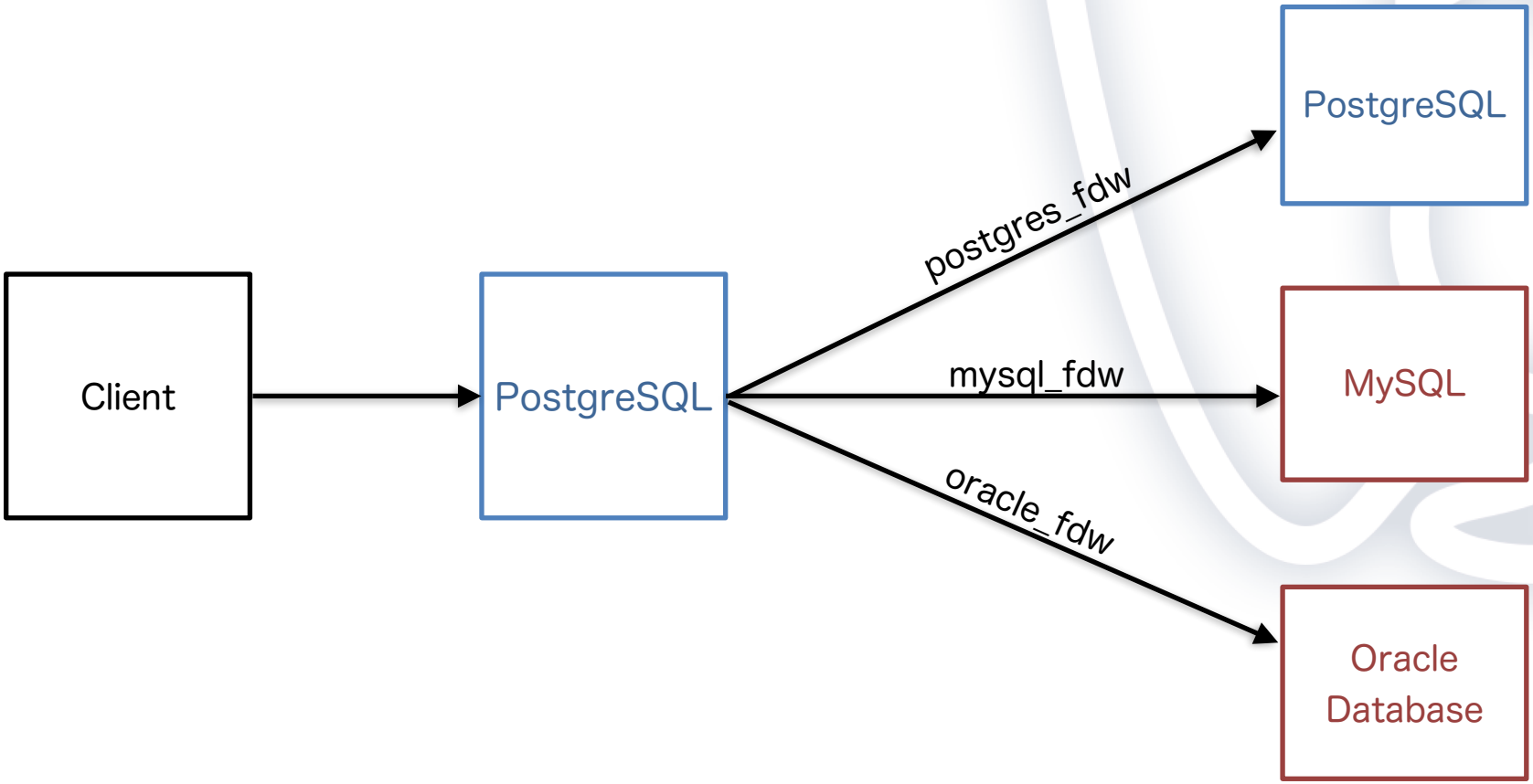
Toward full ACID distributed transaction support with Foreign Data Wrapper

Masahiko Sawada

What is Foreign Data Wrapper

- Implementation of the SQL/MED
- Access data that resides outside PostgreSQL using regular SQL queries
- Foreign tables
- Pluggable architecture
  - https://wiki.postgresql.org/wiki/Foreign_data_wrapper
- Writable at 9.3 or later

Foreign Data Wrapper



Current Status

- FDW plugin is responsible for transaction management on the remote nodes (foreign transaction)
  - begin, commit, rollback and savepoint
- Foreign transaction termination can use XactCallback, which is called BEFORE committing the local transaction

postgres_fdw's Transaction Managements

- Open a foreign transaction when FDW access the remote first time within the local transaction
- Foreign transaction uses SERIALIZABLE when the local transaction has SERIALIZABLE.
  - Otherwise use REPEATABLE READ
- This ensures that if a query performs multiple table scans on the remote server, it will get snapshot-consistent results for all the scans

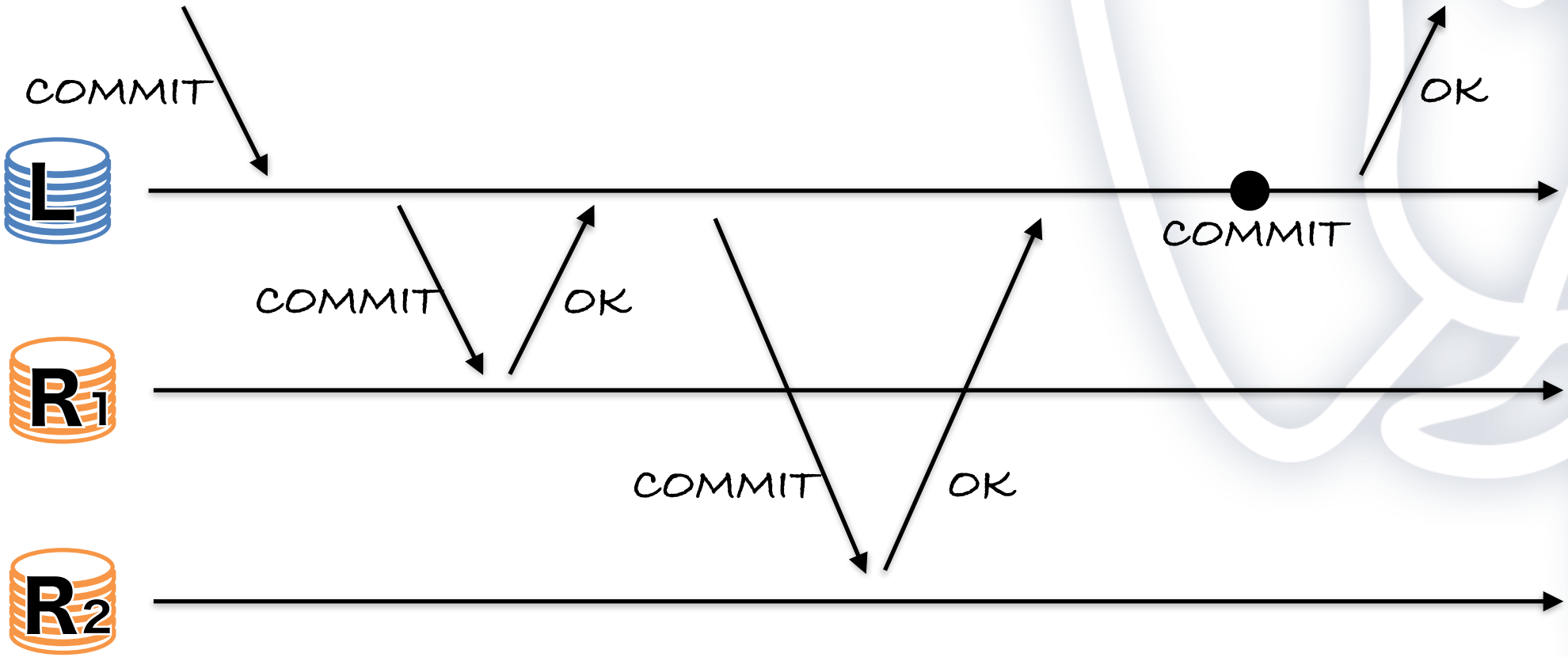
Two Major Limitations

- Atomic Commit
- Read Issues

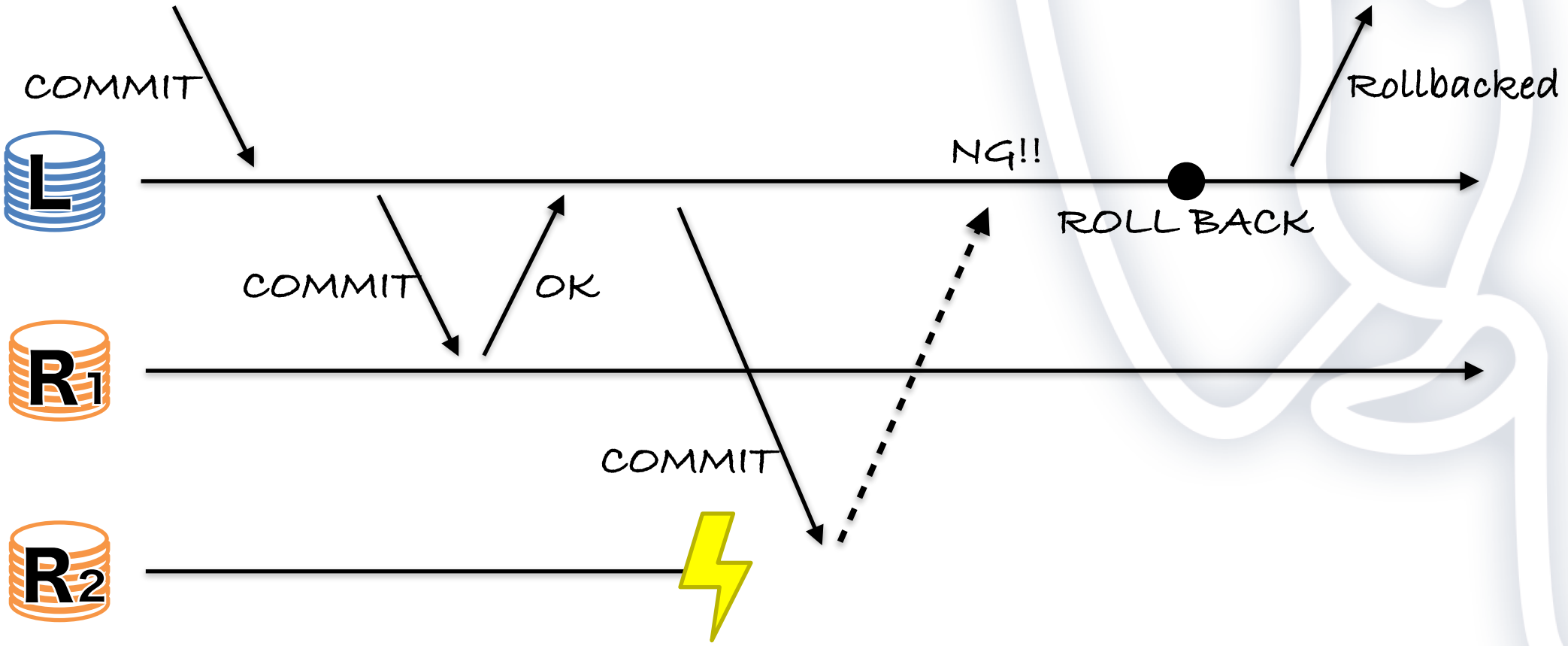
Atomic Commit Problem

- Foreign transactions are committed one by one before the local transaction commits
- If a remote server crashes during the commit, some transactions are committed whereas other are not (it's the same when the local node crashes)
- There is no guarantee that all servers (including local) are committed or rollbacked

Transaction Commit w/ FDW



Transaction Commit w/ FDW



Two-Phase Commit Protocol

- A type of atomic commitment protocol
- Blocking protocol
- Consists of two phases: prepare phase and commit phase

1. Coordinator sends PREPARE request to all participants
2. Coordinator sends COMMIT request to all participants if and only if all participants sent OK in prepare phase
3. Otherwise coordinator sends ROLLBACK request to all participants

2PC for FDW

- First proposal at 2015
  - “Transactions involving multiple postgres foreign servers, take 2” by Masahiko Sawada and Ashutosh Bapat
- The core manages remote transactions
- Introduces new FDW APIs for transaction managements
  - Commit, Rollback, Prepare and GetPrepareId

2PC for FDW - Commit

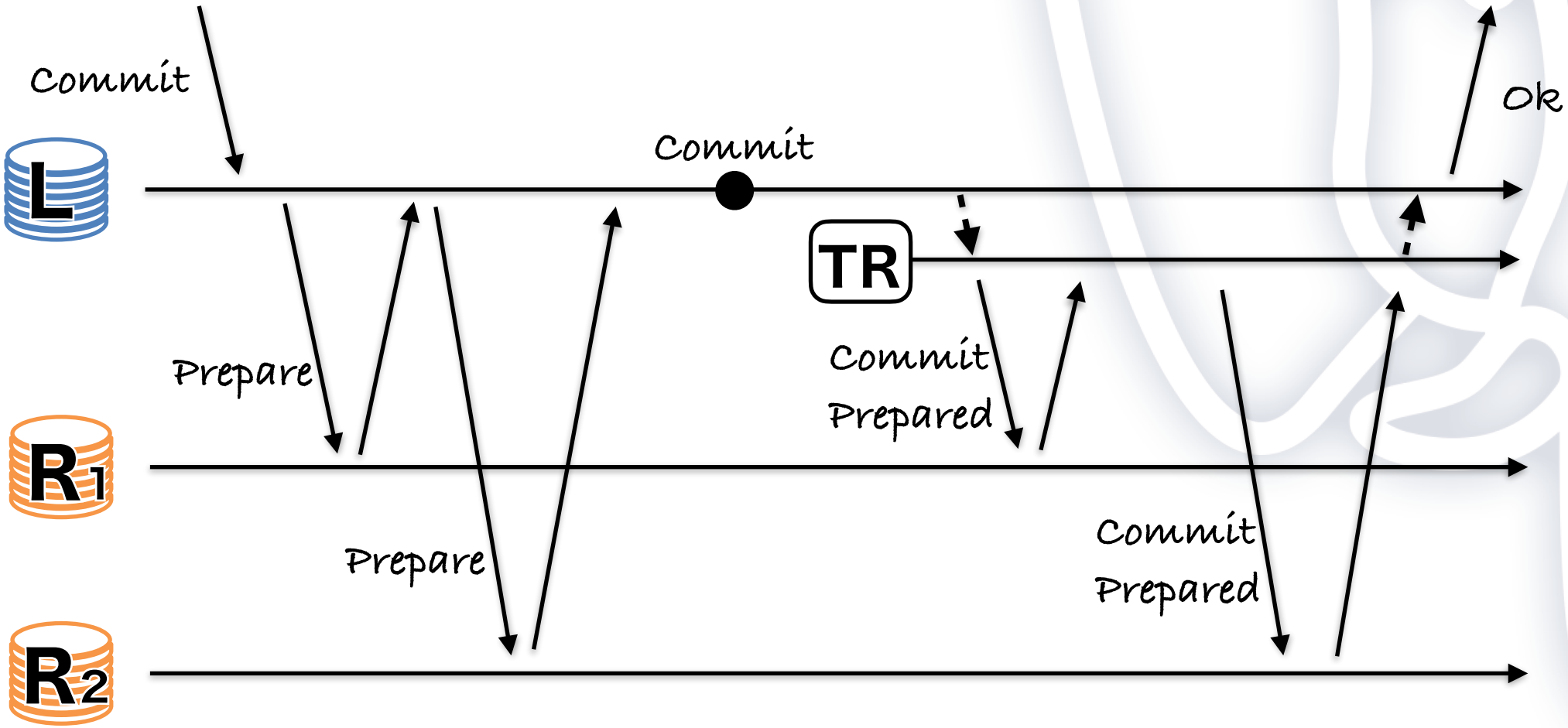
Transaction involving foreign transactions implements commit via following steps:

1. Prepare all foreign transactions
2. Commit locally
3. Commit all foreign transaction

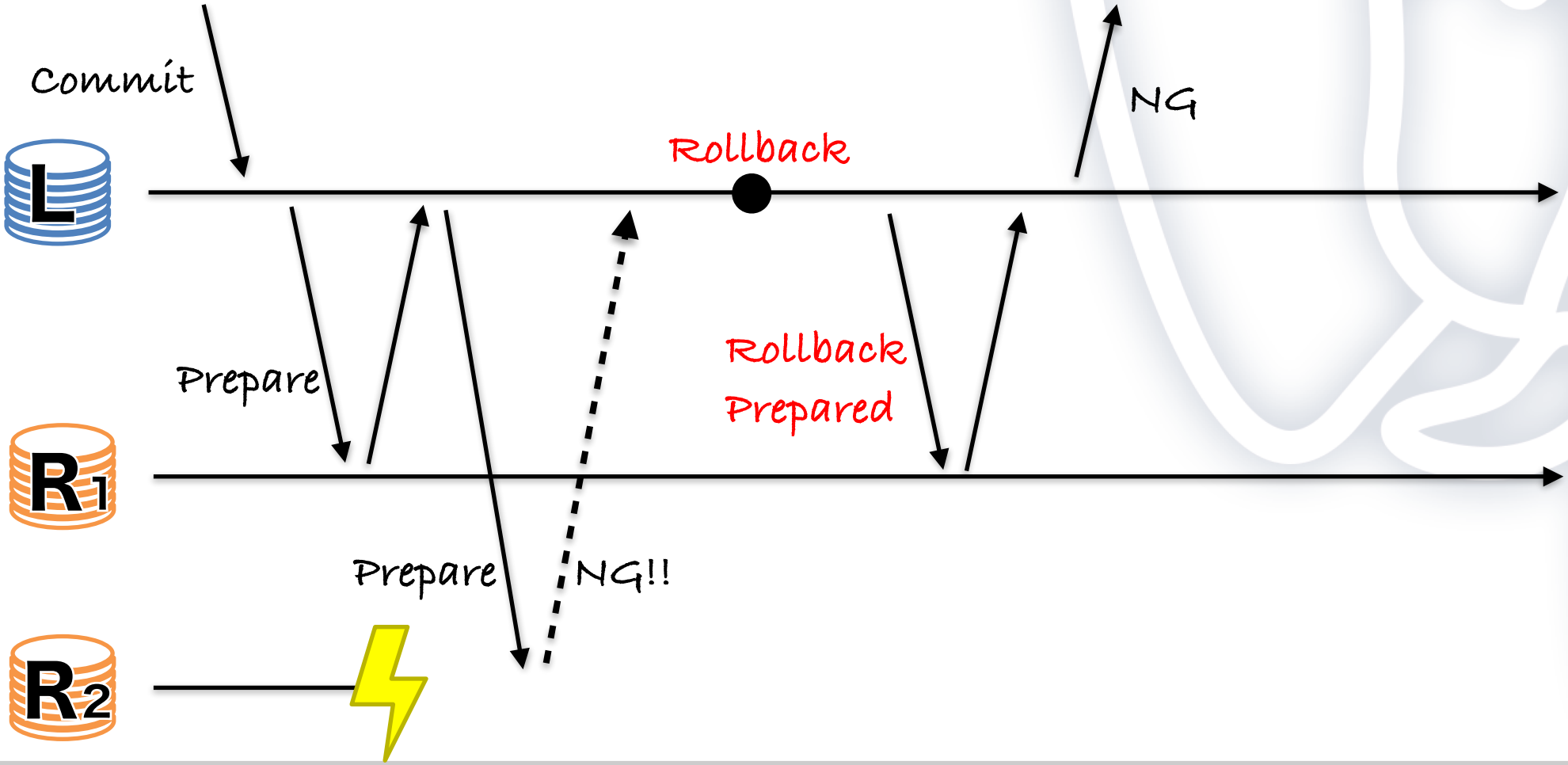
2PC for FDW

- The core persists information about foreign transactions to disk via WAL records so that these can be recovered after restart
- Introduce a new background process called Transaction Resolver
  - Executing COMMIT PREPARED in-progress foreign transactions
  - Resolving recovered or in-doubt foreign transactions

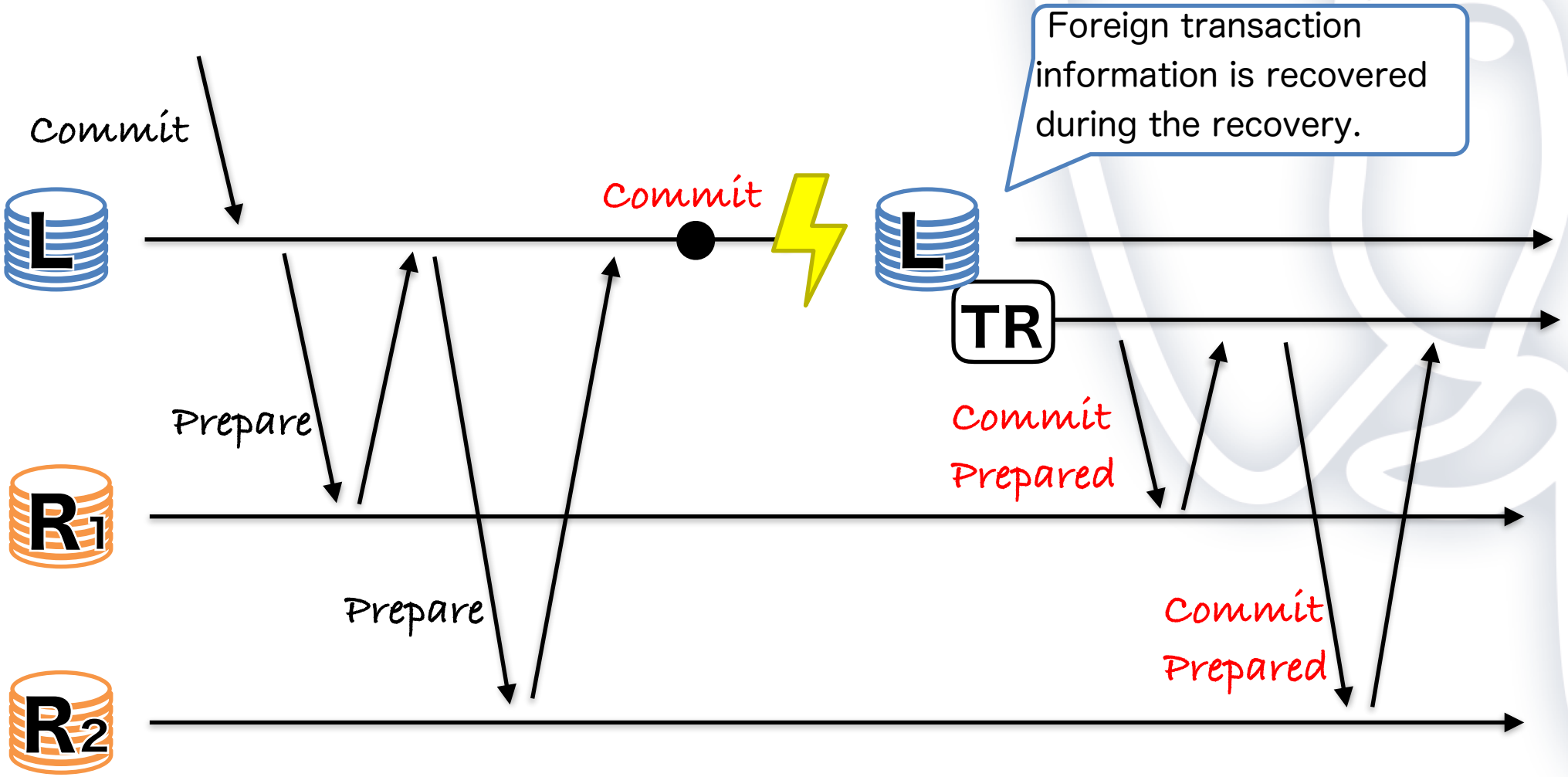
FDW transaction with 2PC



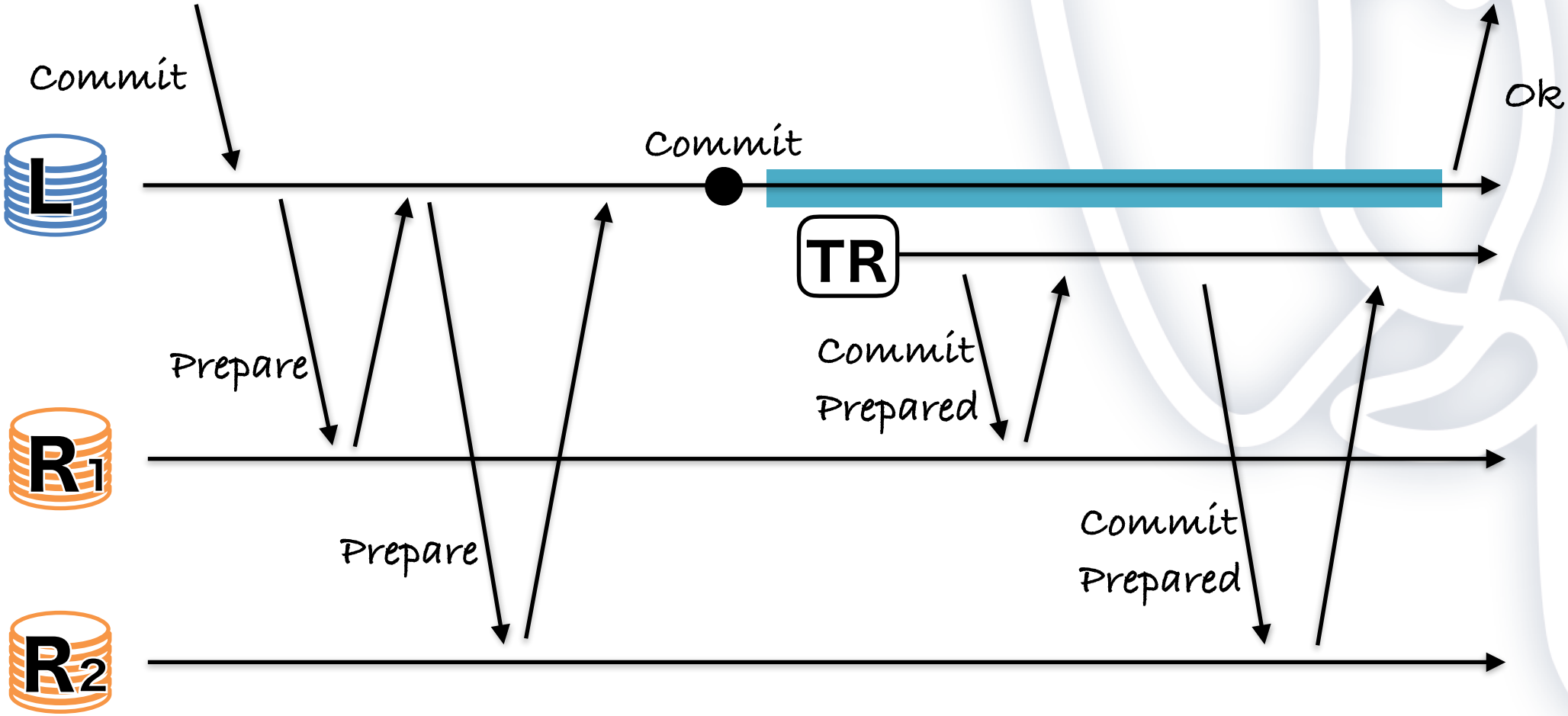
Scenario-1: A participant crashes before commit



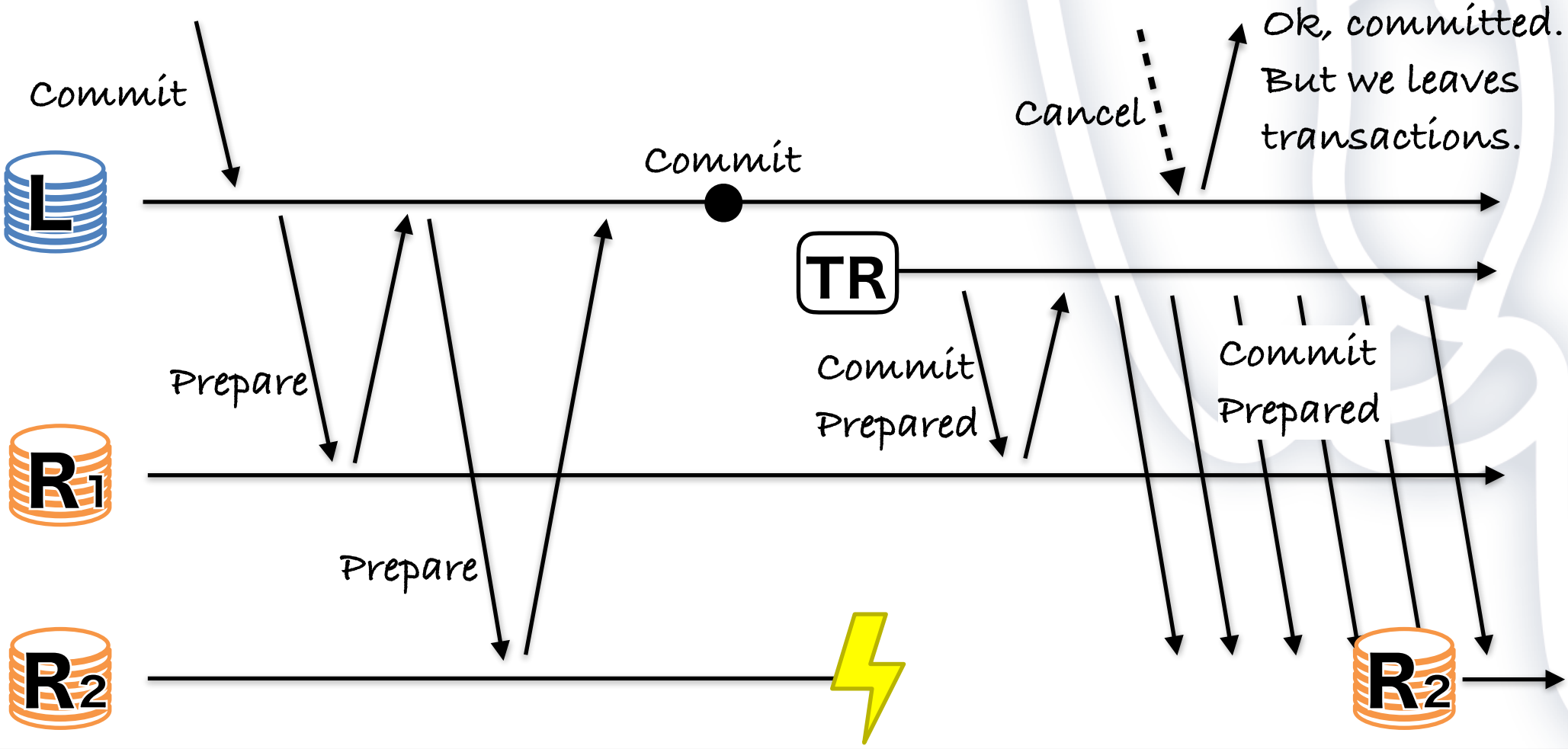
Scenario-2: Local node crashes after commit



Interruptions



Interruptions

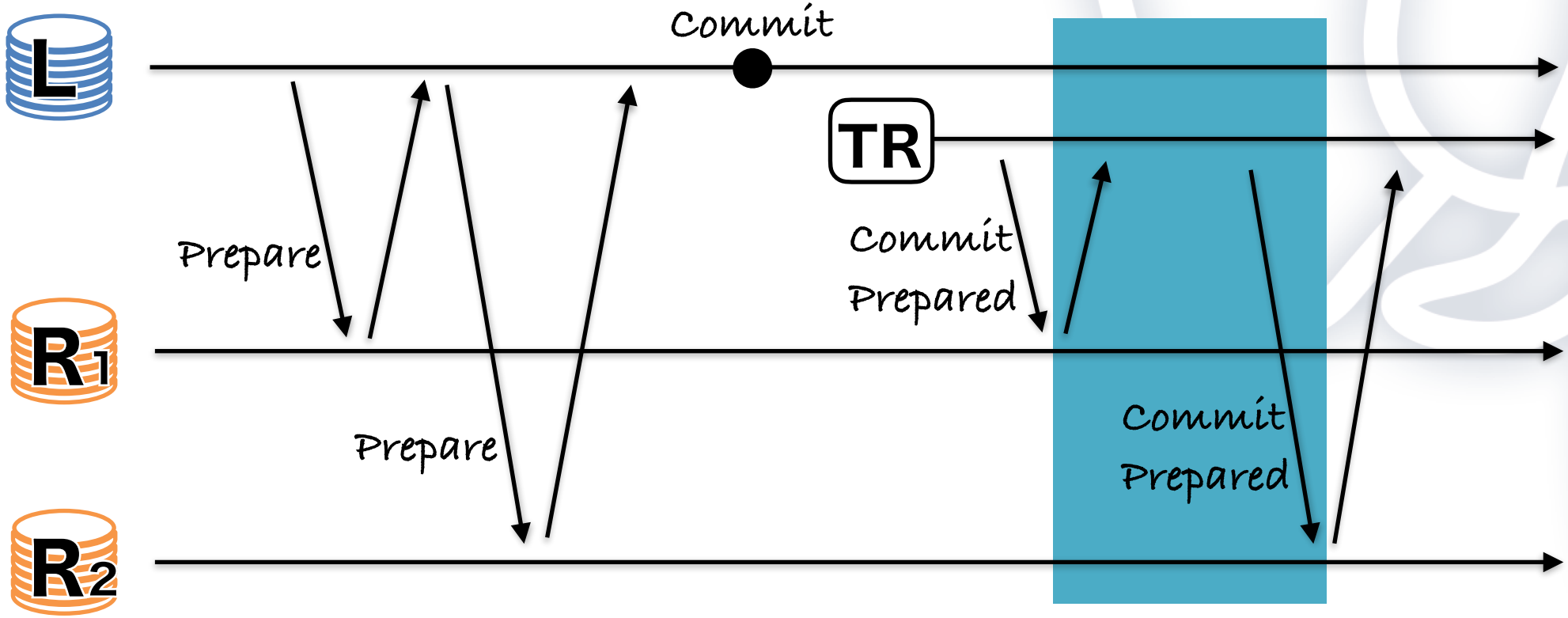


Atomic Visibility

Either all or none of each transaction's updates are observed by other transaction

Atomic Visibility - uncertain period

A transaction starts in this period will be able to see only one committed data



Read Issues

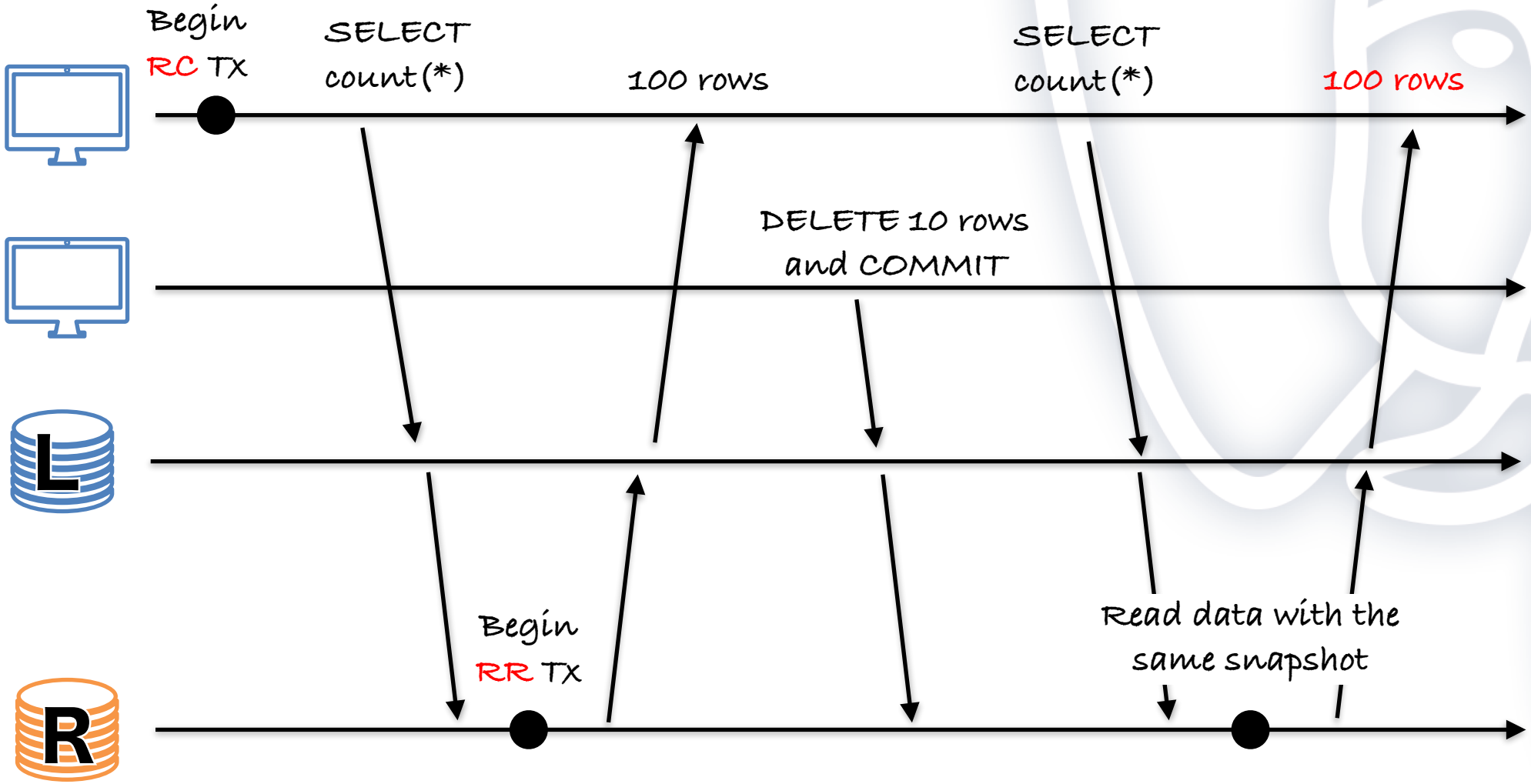
- One of the most important goals of FDW is that if the client uses PostgreSQL with the foreign server then it needs to function the same way as a single PostgreSQL server would do
- Unfortunately, current FDW doesn't work even if a transaction involves only one remote node

Transaction Isolation Levels (ANSI/ISO)

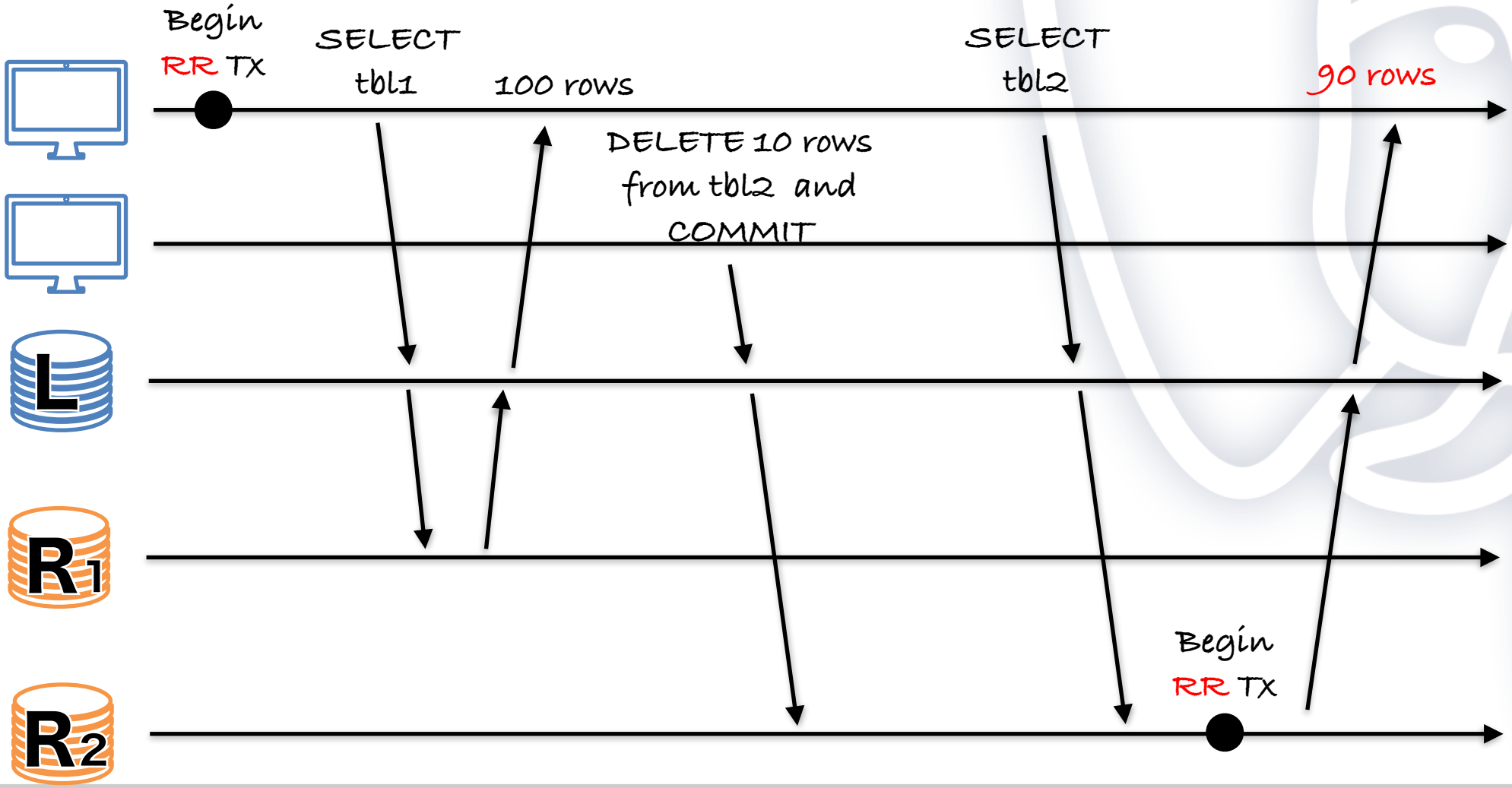
	Dirty read	Non-repeatable Read	Phantom Read
Read Uncommitted (*1)	May occur	May occur	May occur
Read Committed	Don't occur	May occur	May occur
Repeatable Read	Don't occur	Don't occur	May occur (*2)
Serializable	Don't occur	Don't occur	Don't occur

(*1) : PostgreSQL doesn't support Read Uncommitted transaction isolation level.
(*2) : Don't occur in PostgreSQL

Example 1: two clients access the same foreign table



Example 2: two clients access the different foreign table



Summary

- In both case, the client gets the different result than when using single PostgreSQL server
- There is not guarantee that the cluster return the consistent result among all foreign servers even when there is only one foreign table.

Global Snapshots

- Provide globally consistent snapshots
- Global snapshots could be PostgreSQL snapshots (xmin, xmax and in-progress xids), CSN, timestamp etc

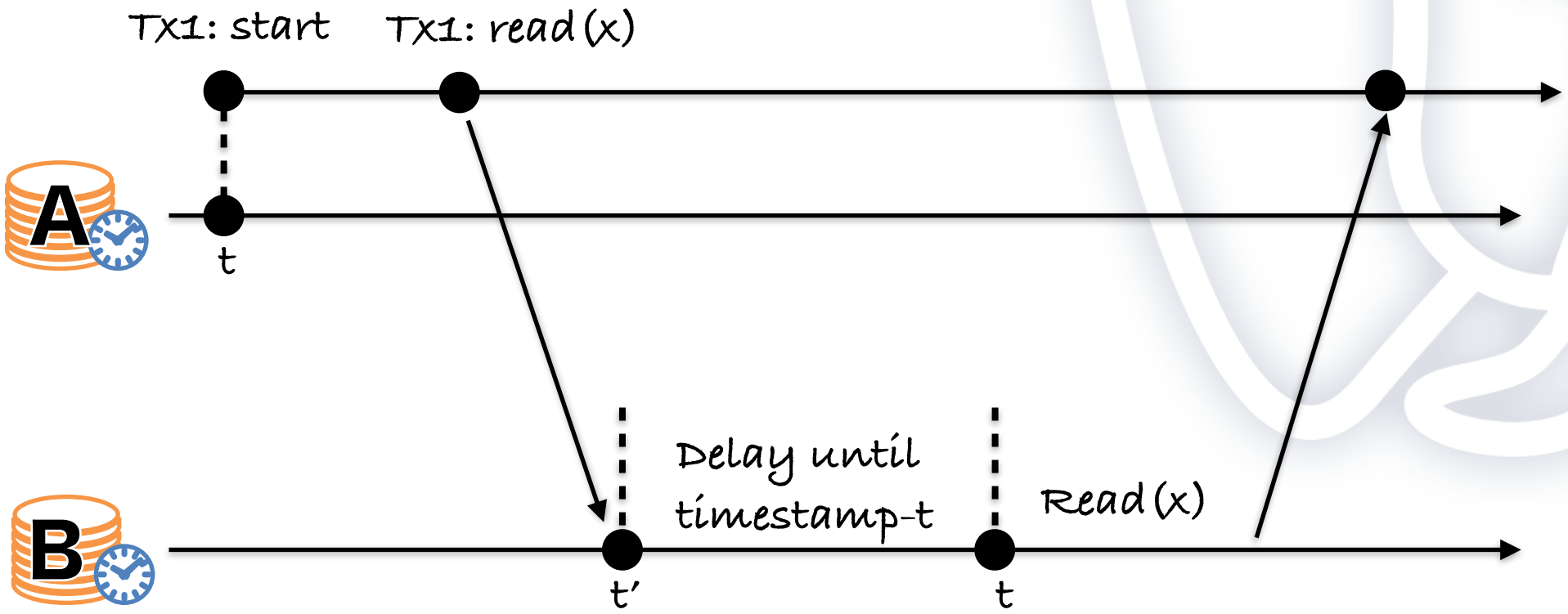
Central Transaction Manager

- Postgres-XL has a dedicated global transaction manager node (GTM node)
  - All coordinators have to access GTM to get a global consistent snapshot
  - GTM provides PostgreSQL snapshot consistent across all data nodes
- Google Percolator has similar concept: Timestamp Oracle
  - Which can produce timestamps in a strictly increasing order
  - Timestamps coming from the timestamp oracle are used as the time when read/write operation happens
- But.. SPOF issues with GTMs

Clock-SI

- Jiaqing Du et al., “Clock-SI: Snapshot Isolation for Partitioned Data Stores Using Loosely Synchronized Clocks”, 2013 IEEE 32nd International Symposium on Reliable Distributed Systems, 2013.
- Use physical time as CSN
- Address clock skew problem
- A patch which implements Clock-SI paper was proposed at 2018
  - “Global Snapshots” by Stas Kelvich

Clock Skew



Summary

- Foreign Data Wrapper is the powerful feature to access the distributed data across heterogenous data stores
- A big missing pieces is transaction management
- Several ideas are proposed
  - 2PC over FDW
  - Clock-SI

Thank you

masahiko.sawada@2ndquadrant.com