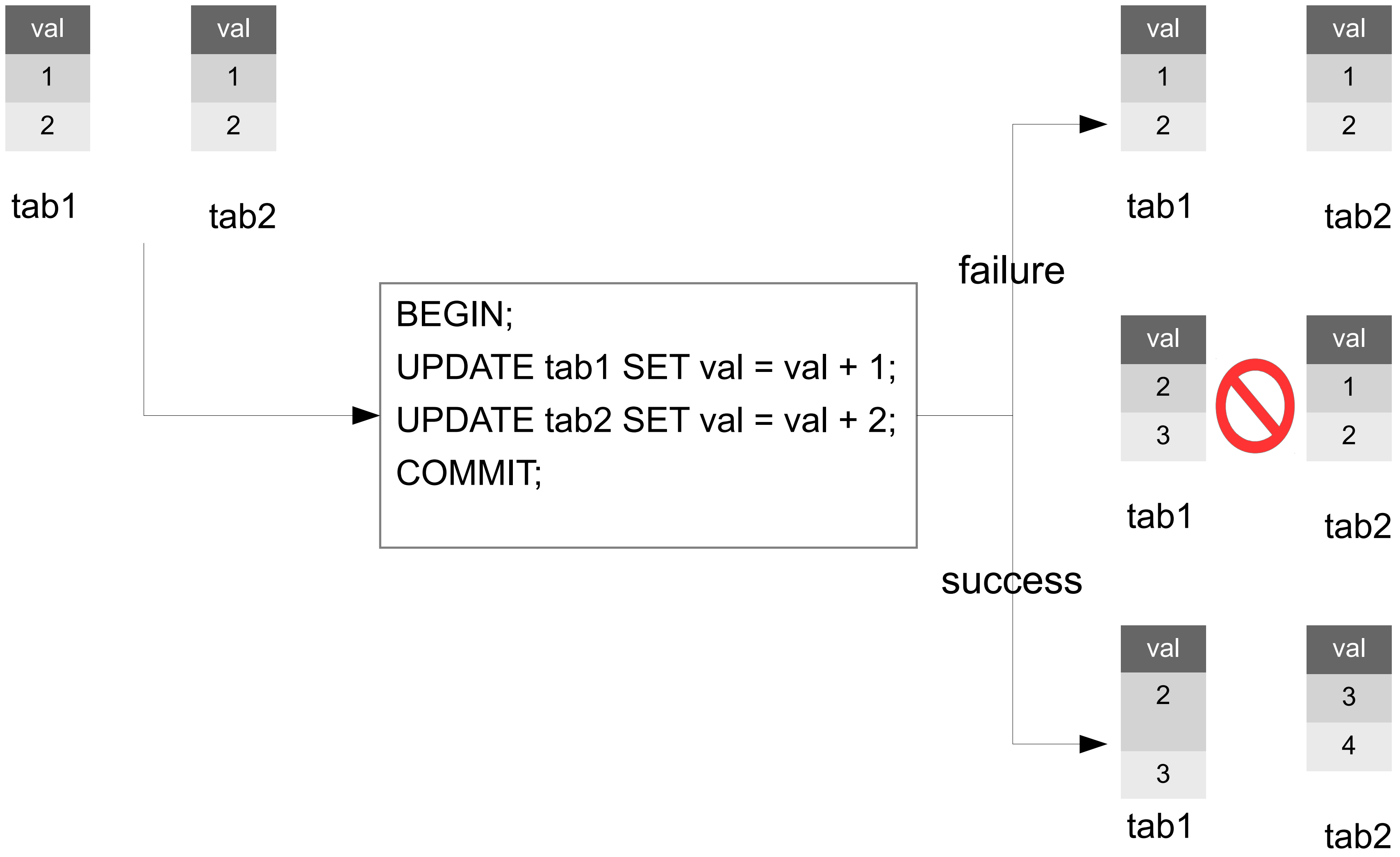
Transacting with foreign servers

- Ashutosh Bapat | 19th June 2015

Agenda

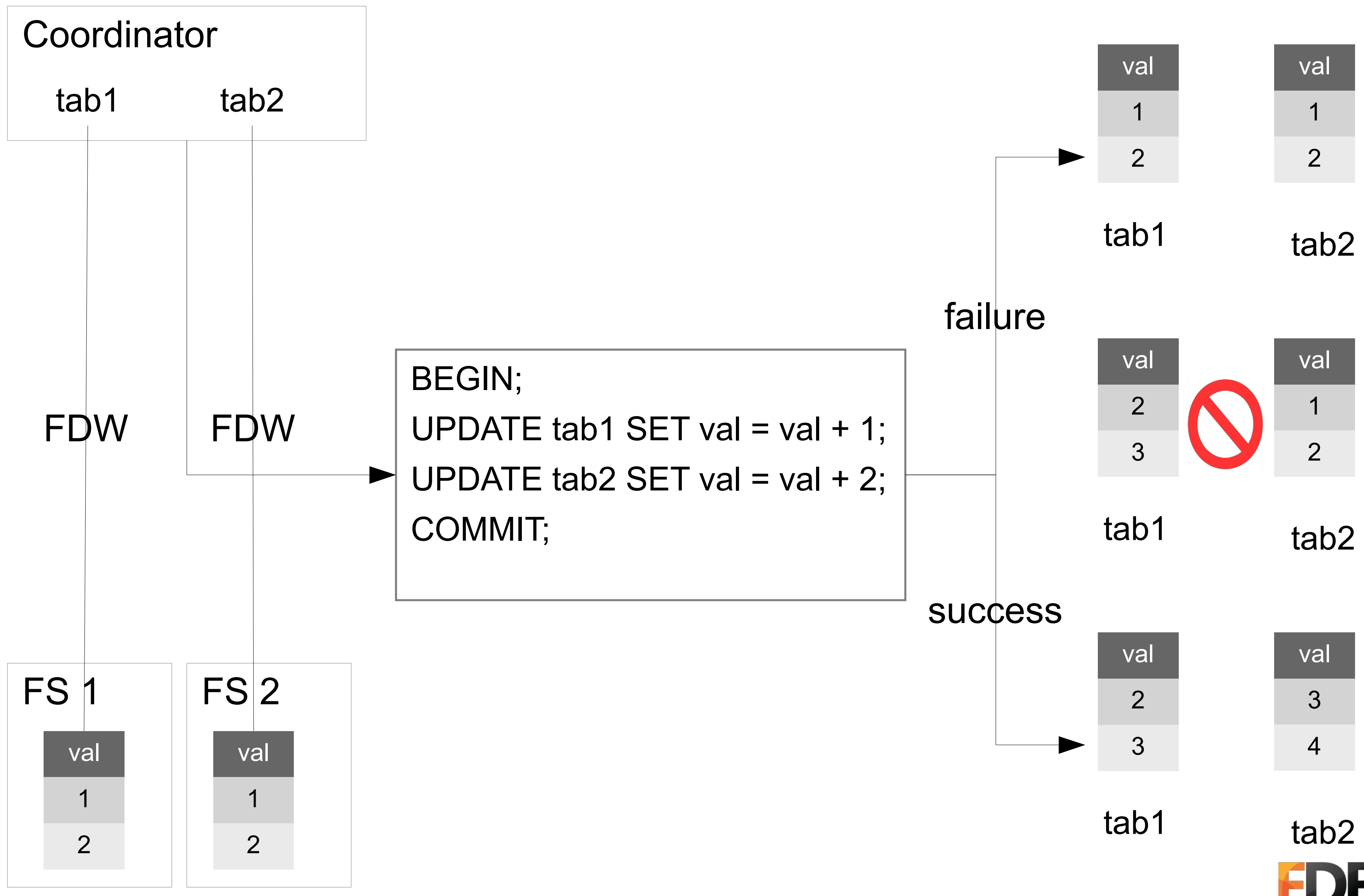
- Atomic commit for transactions involving foreign servers
  - Current status
  - Solution – two-phase commit protocol
  - Overview of implementation
  - Foreign servers without 2PC support
- Atomic visibility

Atomicity



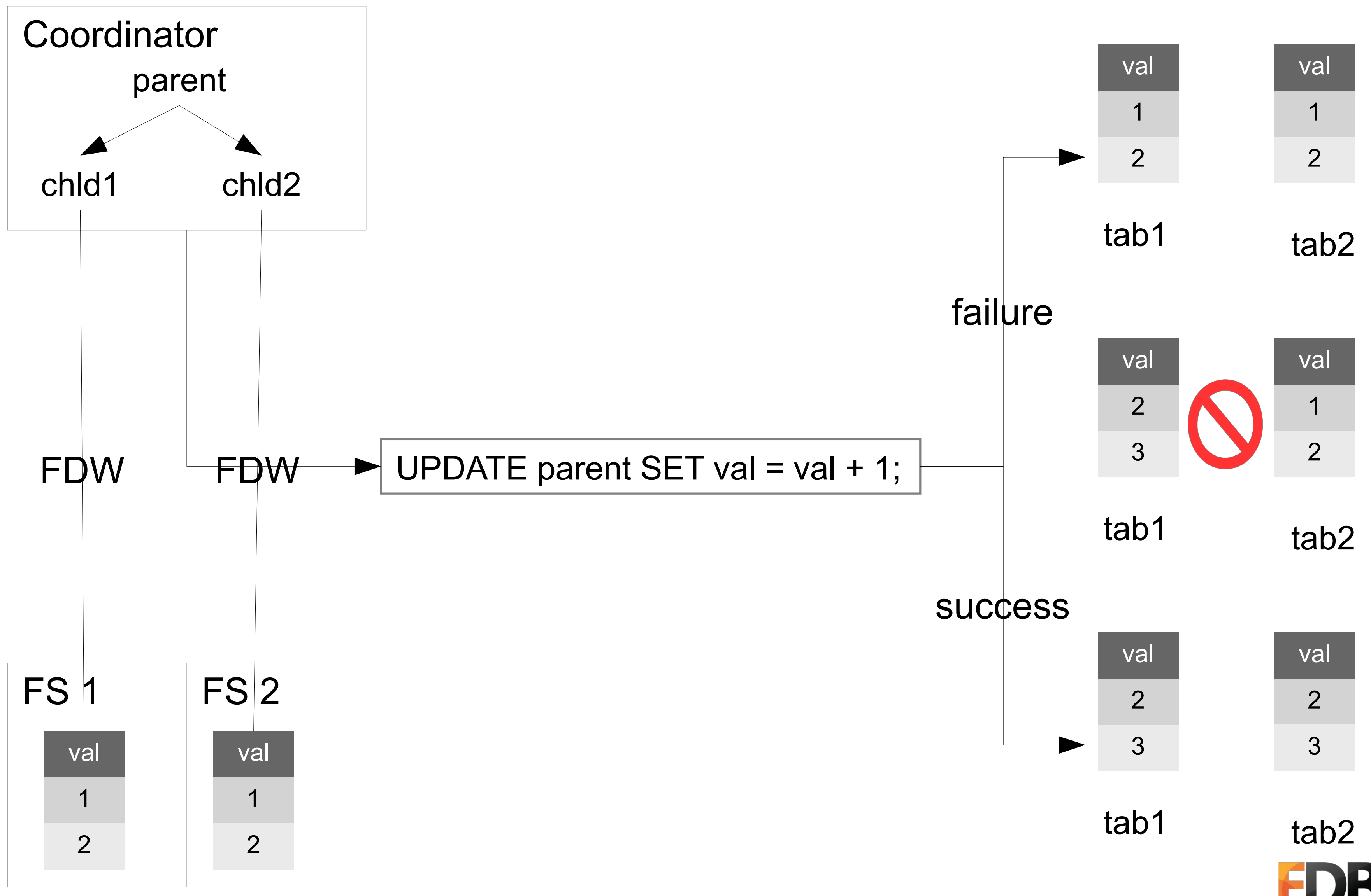
```
BEGIN;
UPDATE tab1 SET val = val + 1;
UPDATE tab2 SET val = val + 2;
COMMIT;
```

Distributed Atomicity



```
BEGIN;
UPDATE tab1 SET val = val + 1;
UPDATE tab2 SET val = val + 2;
COMMIT;
```

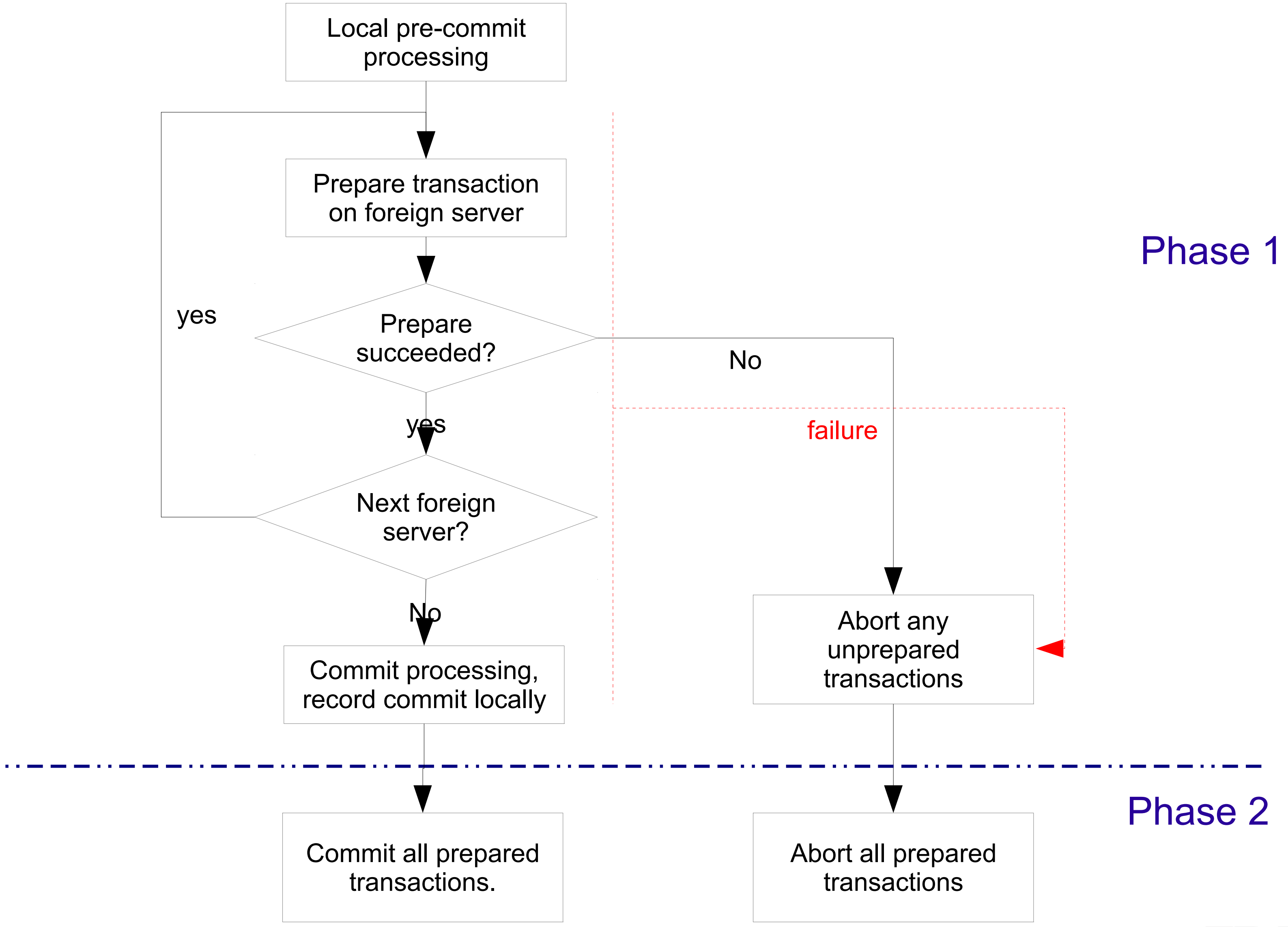
Atomicity - foreign table inheritance



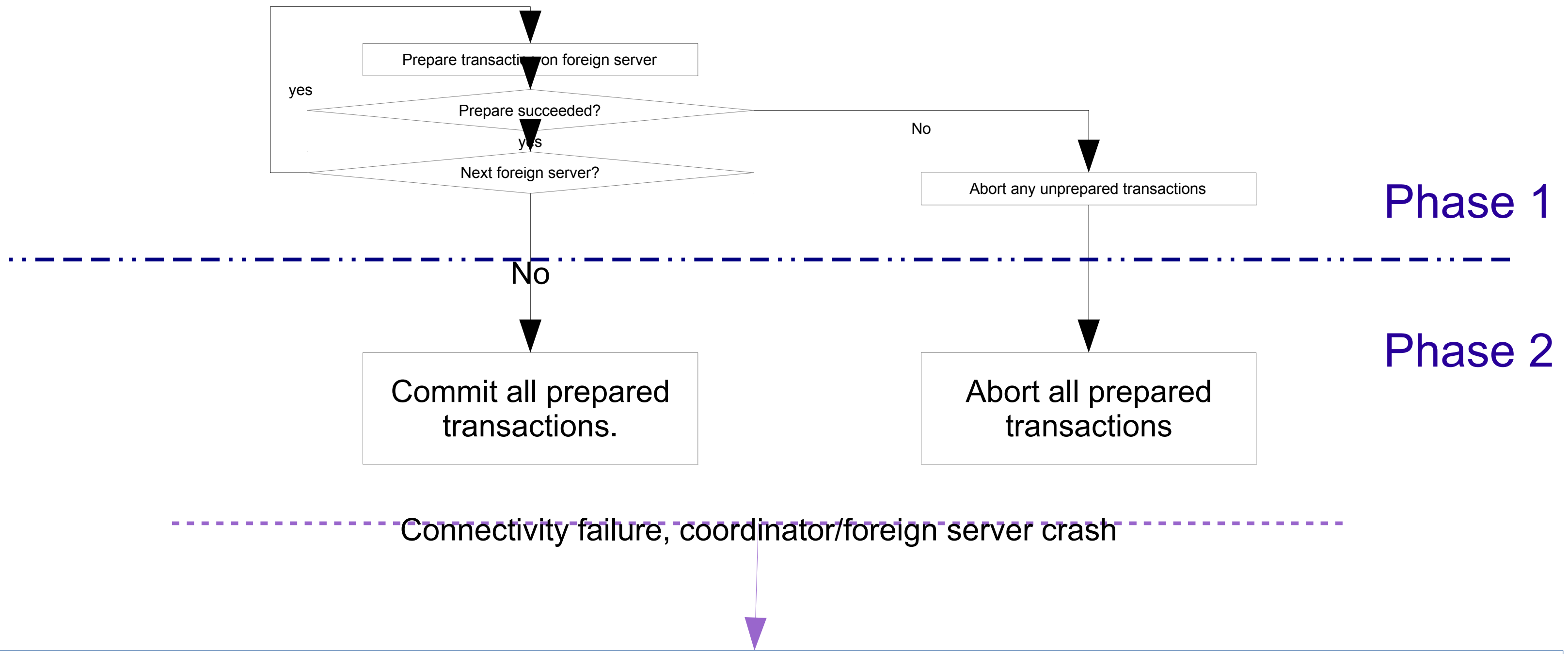
Current status

- Transaction management is entirely implemented by FDW
  - postgres_fdw uses one-phase commit
  - any failure (including connectivity) during commit processing can cause changes to some foreign servers committed and others aborted
- Atomicity is not guaranteed

Two-phase commit

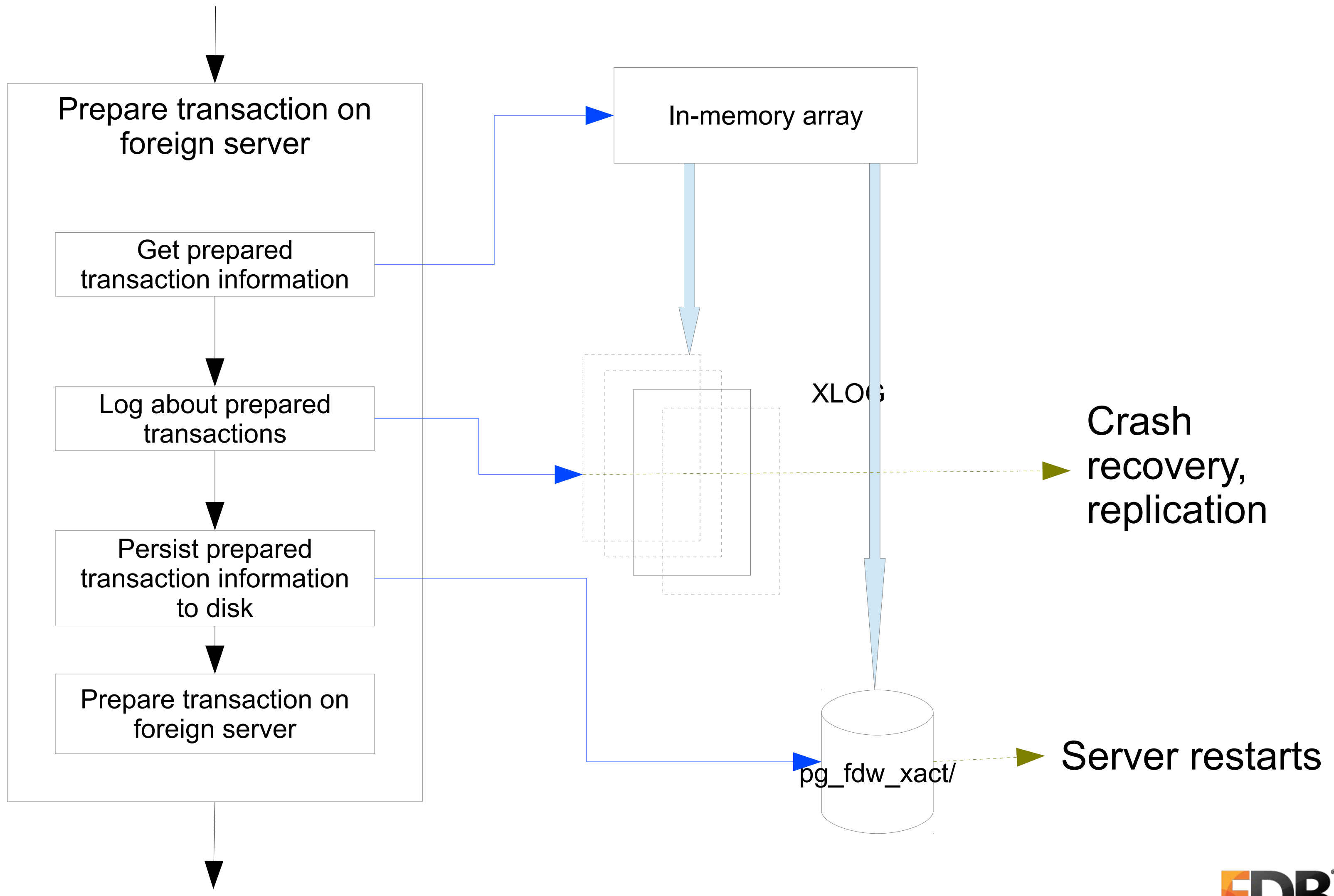

Local pre-commit processing
Prepare transaction on foreign server
Prepare succeeded?
yes
No
failure
yes
Next foreign server?
No
Commit processing, record commit locally
Abort any unprepared transactions
Phase 1
Phase 2
Commit all prepared transactions.
Abort all prepared transactions

Two-phase commit

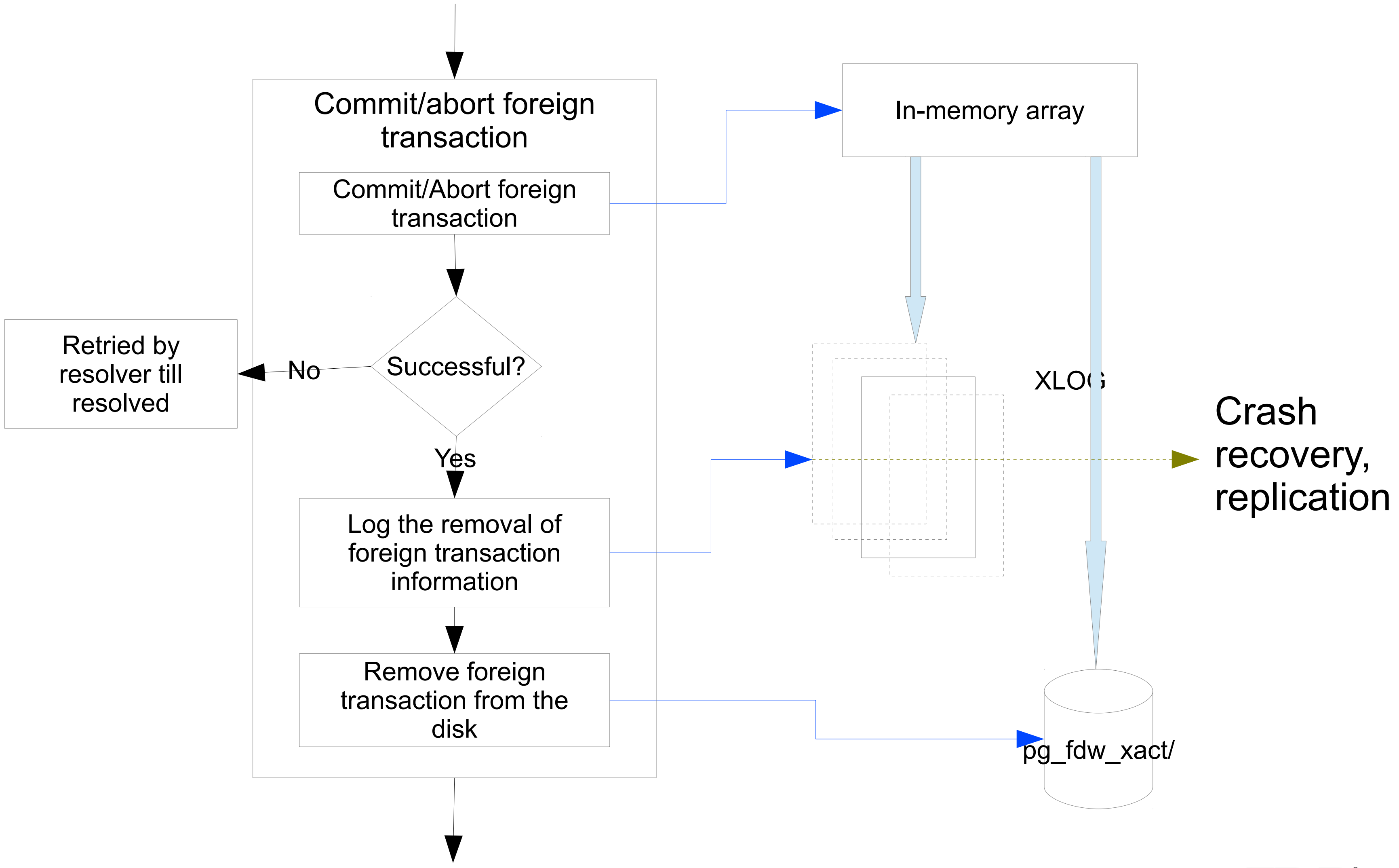

Prepare transaction on foreign server
yes
Prepare succeeded?
yes
Next foreign server?
No
Abort any unprepared transactions
Phase 1
No
Commit all prepared transactions.
Abort all prepared transactions
Phase 2
Connectivity failure, coordinator/foreign server crash

- all the foreign transactions and local transaction may not commit at the same time
  - Atomic commit – guaranteed
  - Atomic visibility – not guaranteed
- Unresolved prepared transactions – prepared transactions waiting to be committed/aborted
  - Blocked resources, degraded performance

Remembering foreign transactions


Prepare transaction on foreign server
Get prepared transaction information
Log about prepared transactions
Persist prepared transaction information to disk
Prepare transaction on foreign server
In-memory array
XLOG
Crash recovery, replication
pg_fdw_xact/
Server restarts

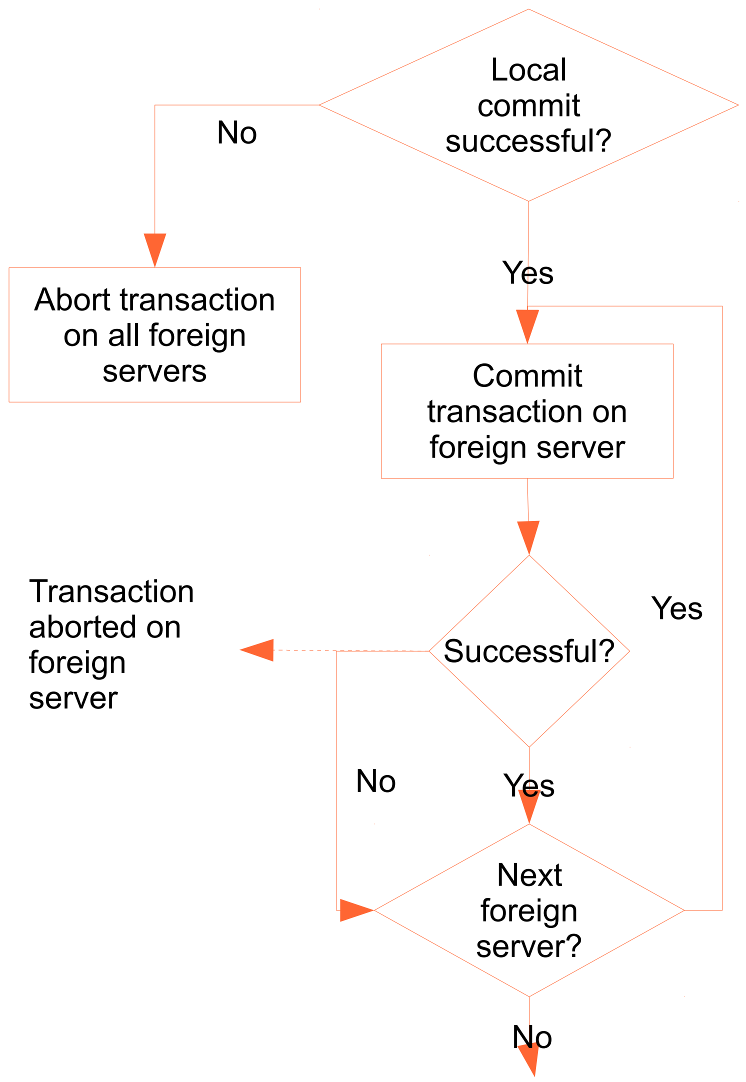
Resolving foreign transactions


Commit/abort foreign transaction
Commit/Abort foreign transaction
Successful?
No
Retried by resolver till resolved
Yes
Log the removal of foreign transaction information
Remove foreign transaction from the disk
In-memory array
XLOG
Crash recovery, replication
pg_fdw_xact/

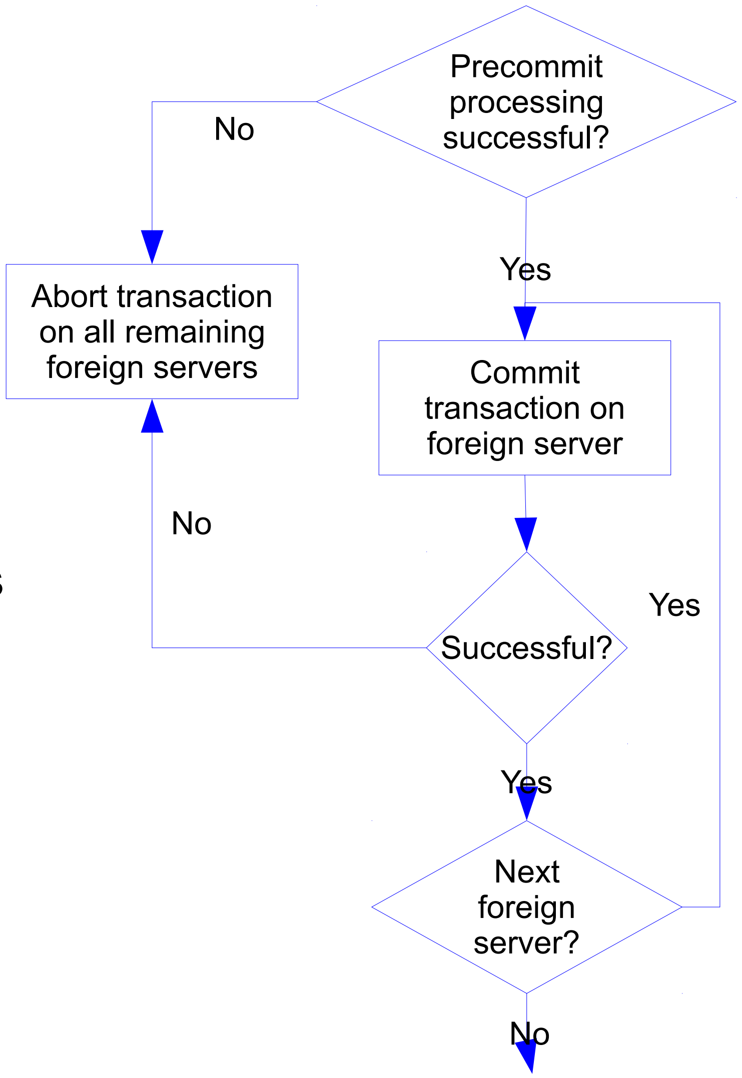
Foreign transaction information

- Transaction id: decides the fate of foreign transaction
- Database id: where to look for foreign server and user mapping
- Foreign server and user: where and who prepared the foreign transaction
- Foreign transaction id: identifies the prepared transaction on the foreign server
  - e.g. GID on PostgreSQL

Foreign servers without 2PC support


Local commit successful?
No
Abort transaction on all foreign servers
Yes
Commit transaction on foreign server
Yes
Transaction aborted on foreign server
Successful?
No
Yes
Next foreign server?
No

Vs


Precommit processing successful?
No
Abort transaction on all remaining foreign servers
Yes
Commit transaction on foreign server
No
Yes
Successful?
Yes
Next foreign server?
No

FDW hooks for transaction management

- GetPrepareId
  - function to obtain prepared transaction id for a given foreign server
  - Each FDW might have different rules for identifier
  - Persisted and WAL logged

FDW hooks for transaction management

- HandleForeignTransaction - function to end foreign transaction in following ways
  - Commit/Rollback running transaction
  - Prepare running transaction
  - Commit/Rollback prepared transaction with given identifier

GUCs

- atomic_foreign_transaction – when ON, two phase commit is used for transactions involving foreign servers.
  - When ON requires all participating servers to support 2PC
  - Can be set any time during the transaction.Value at the time of commit is used.
- max_foreign_xacts – maximum number of transactions prepared on foreign servers at a given time

Foreign transaction resolver

- Builtin function (pg_fdw_resolve())– resolves all the unresolved foreign transactions in the database where it's run
- Background worker process – fires the built-in function by connecting to various databases
  - Available at contrib/pg_fdw_xact_resolver
  - Install and add to 'shared_preload_libraries' (and restart)

Atomic visibility

Atomic visibility – rough ideas

- Problem: MVCC allows older versions to be read
  - Older version of data is read while transaction modifying it is prepared but not committed/resolved
- Solution: Wait for prepared transactions to get committed/resolved
  - Stronger locking – FOR SHARE, SERIALISABLE ISOLATION – performance affected
  - Resolve prepared transactions before data is accessed
    - You have connectivity now, right?
    - How to detect this situation?

Current status

- Hackers discussion thread with subject “Transactions involving multiple postgres foreign servers”
- First WIP patch on 17th Feb.
  - 2015-06 commitfest
  - Had several TODOs
- Improved patch to be submitted soon
  - Takes care of many TODOs
  - Addresses atomic commit
  - Does not address atomic visibility

merci
spasiba
kam ouen
gratzias
manana
mahalo
cheers
toda
grassie
thank you
danki
hvala
gracias
grazie
tak
kitos
welalin
THANK YOU
mahalo
danki
thanks
takk
gracias
talofa
miigwetch
domo arrigato
grattitude
merci
na gode
dankon
danke
thanks
mesi
modupe
kitos
dziekuje
takk