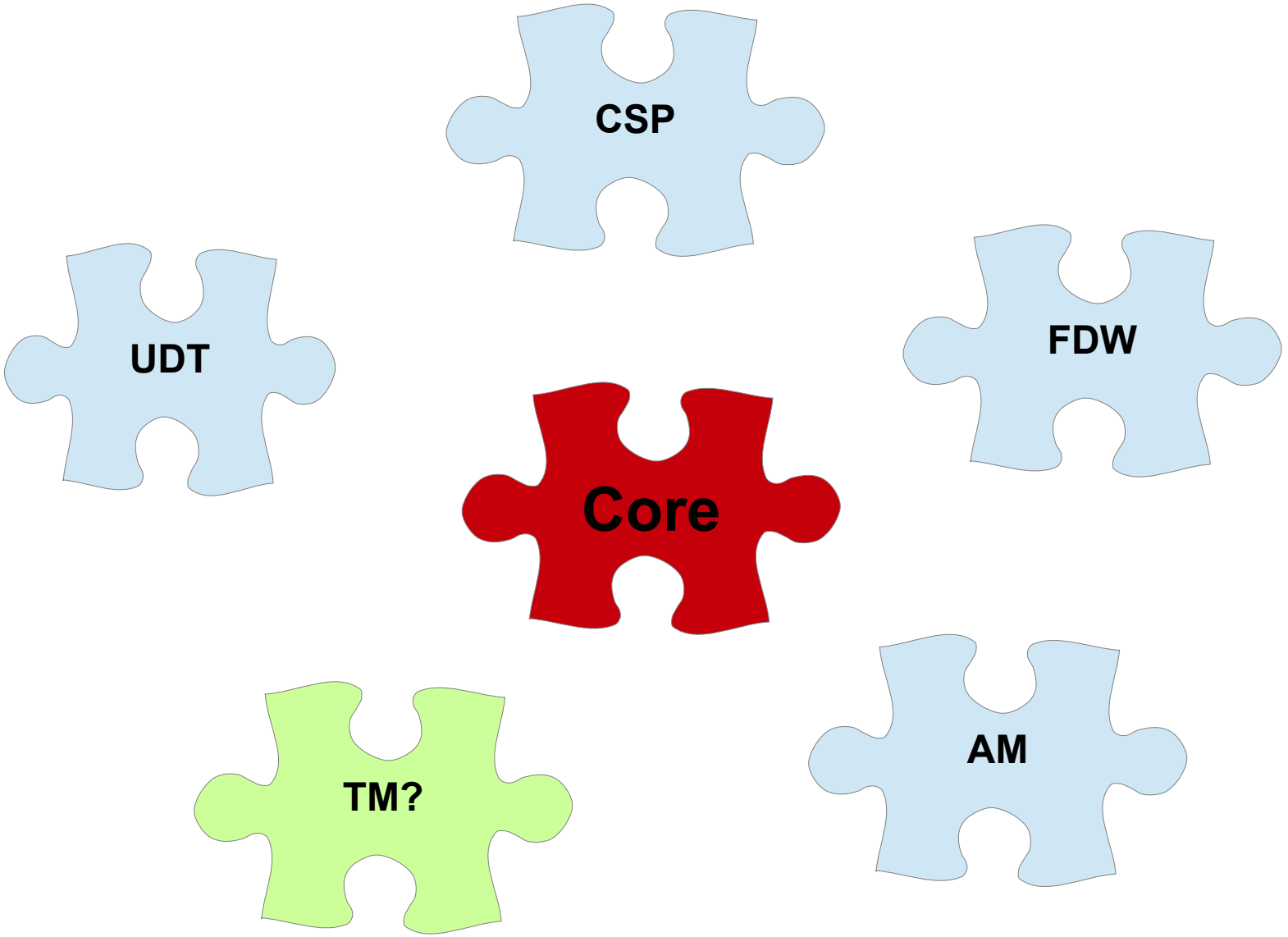
Distributed Transaction Manager

Postgres PROFESSIONAL

Pluggable transaction API



Postgres PROFESSIONAL

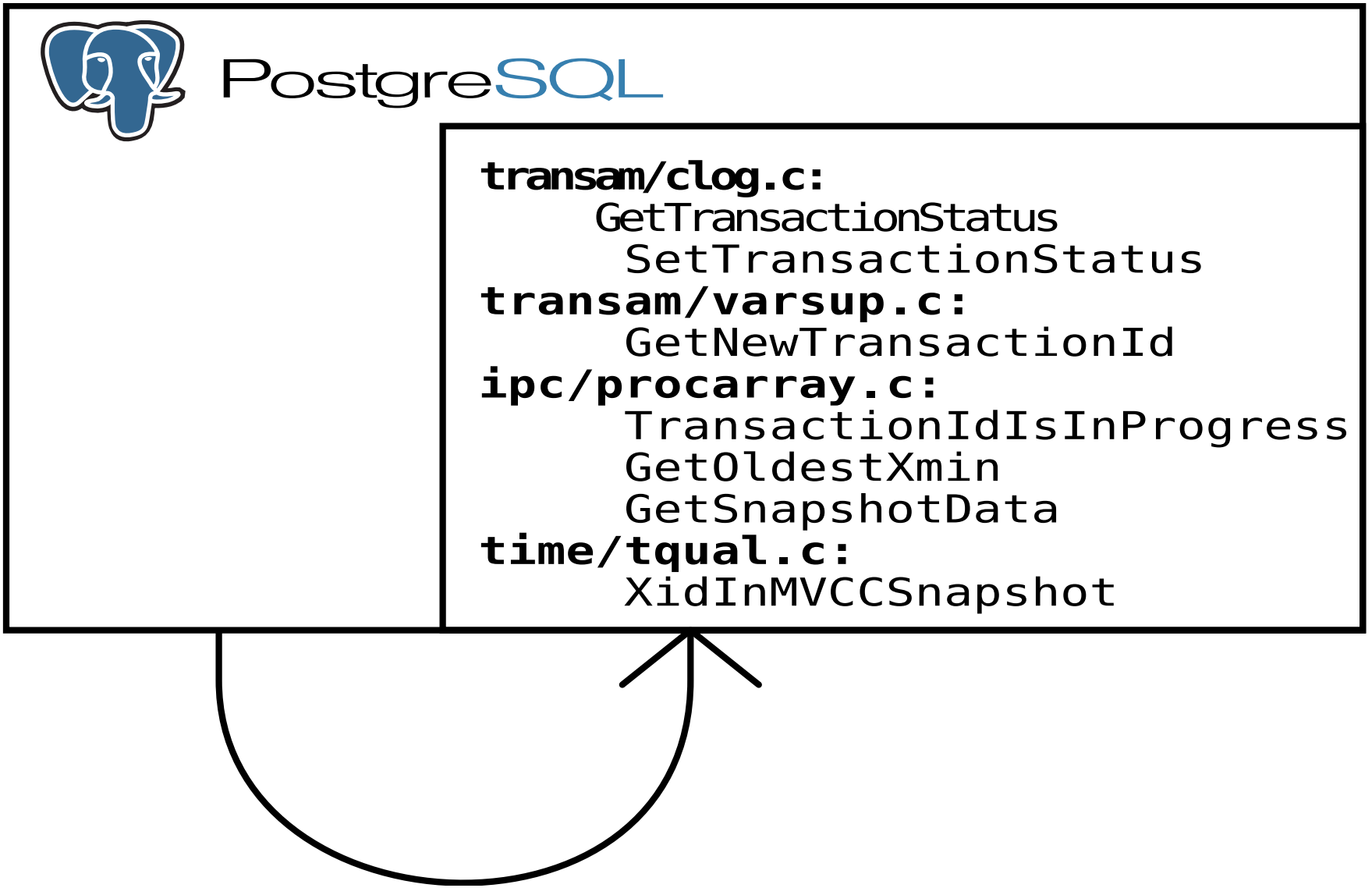
eXtensible Transaction API

- XidStatus (*GetTransactionStatus)(TransactionId xid, XLogRecPtr *lsn);
- void (*SetTransactionStatus)(TransactionId xid, int nsubxids, TransactionId *subxids, XidStatus status, XLogRecPtr lsn);
- Snapshot (*GetSnapshot)(Snapshot snapshot);
- TransactionId (*GetNewTransactionId)(bool isSubXact);
- TransactionId (*GetOldestXmin)(Relation rel, bool ignoreVacuum);
- bool (*IsInProgress)(TransactionId xid);
- TransactionId (*GetGlobalTransactionId)(void);
- bool (*IsInSnapshot)(TransactionId xid, Snapshot snapshot);

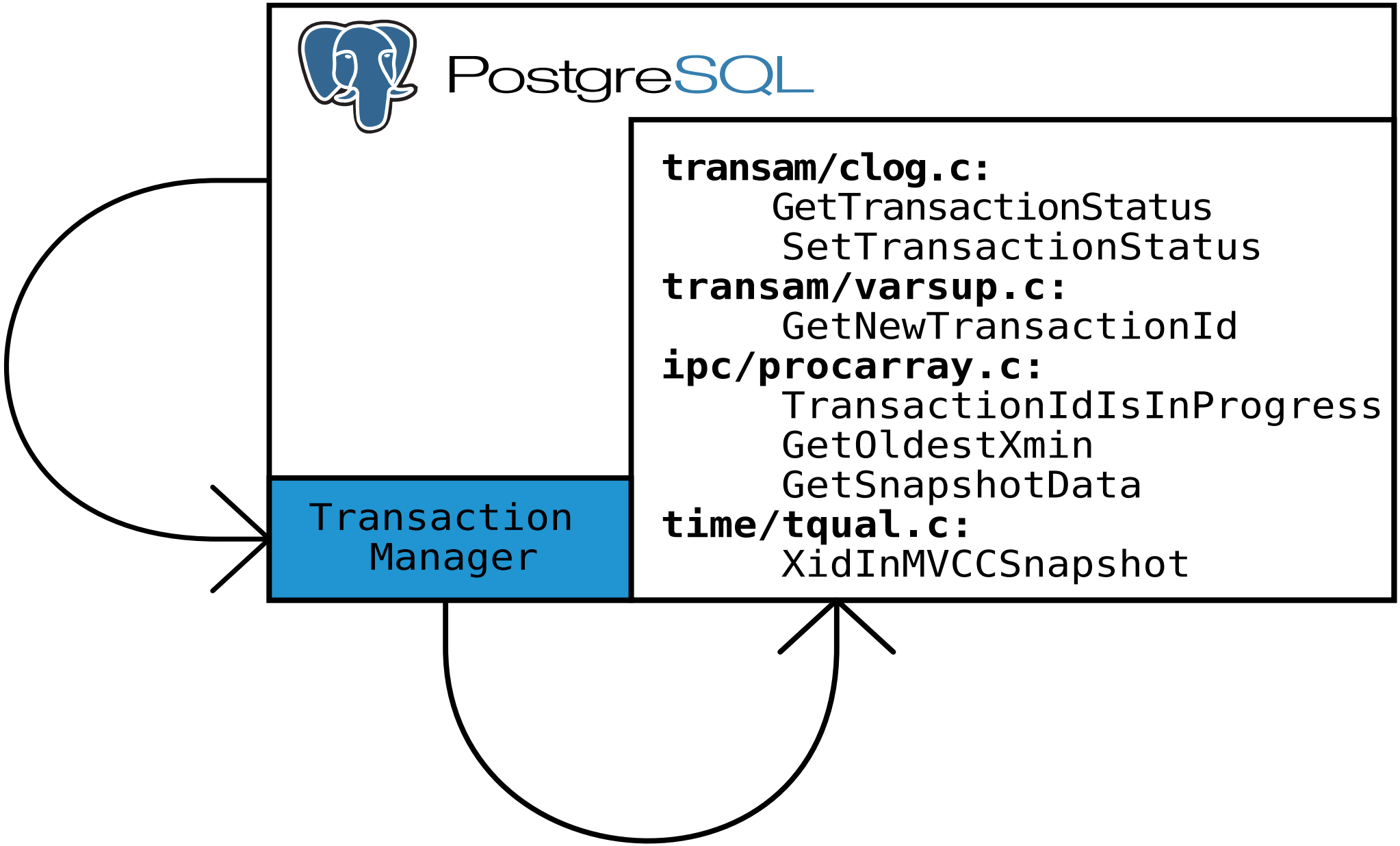
New commit callback events

- XACT_EVENT_START,
- XACT_EVENT_COMMIT,
- XACT_EVENT_PARALLEL_COMMIT,
- XACT_EVENT_ABORT,
- XACT_EVENT_PARALLEL_ABORT,
- XACT_EVENT_PREPARE,
- XACT_EVENT_PRE_COMMIT,
- XACT_EVENT_PARALLEL_PRE_COMMIT,
- XACT_EVENT_PRE_PREPARE,
- XACT_EVENT_COMMIT_PREPARED,
- XACT_EVENT_ABORT_PREPARED

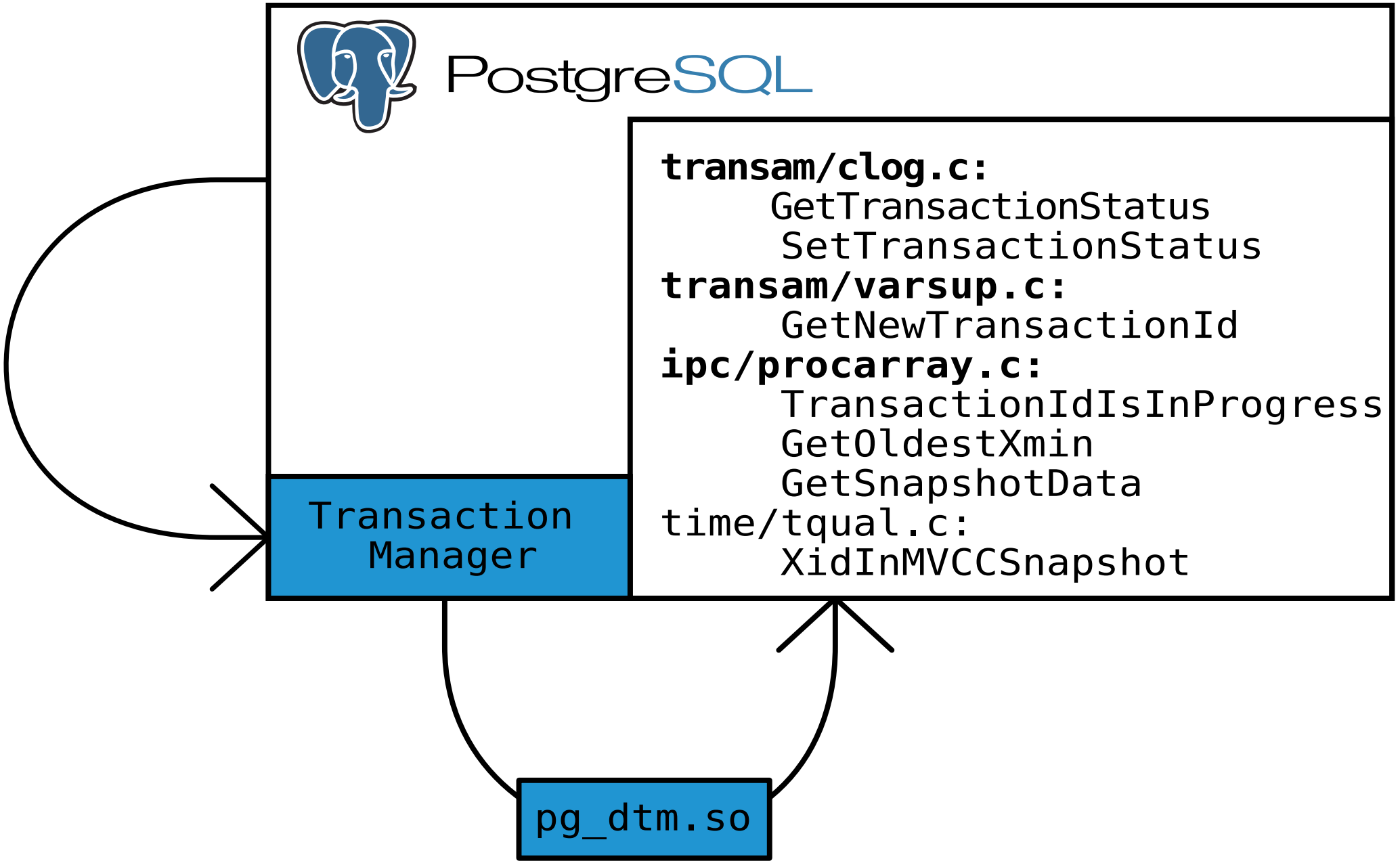
Transaction Manager before patch



Transaction Manager after patch



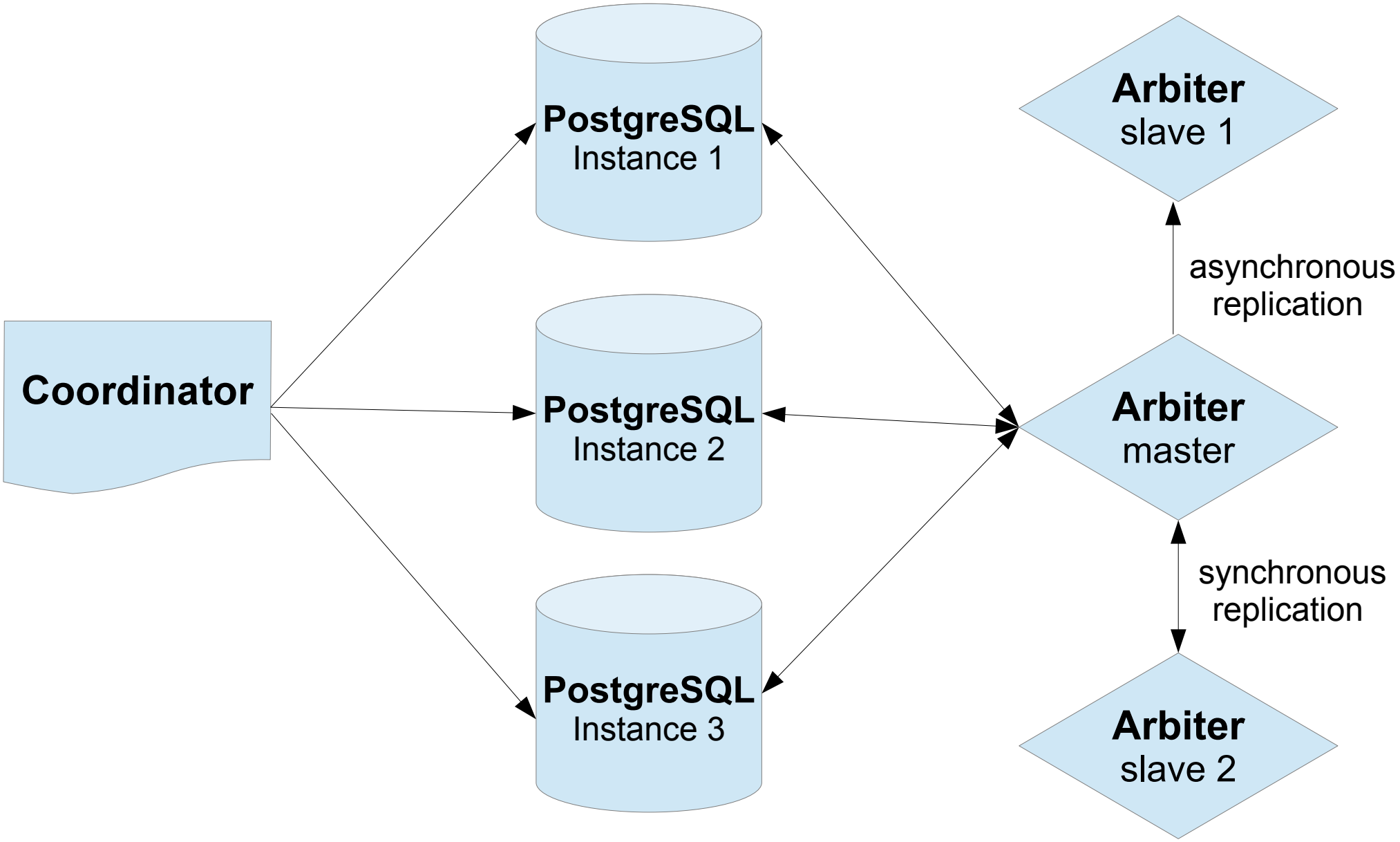
Distributed Transaction Manager



Different DTM implementations

	Local transactions	2PC	Arbiter	Examples
Snapshot sharing			✓	XL, DTM
Timestamp	✓	✓		Spanner, Cockroach, tsDTM
Incremental	✓		✓	SAP HANA

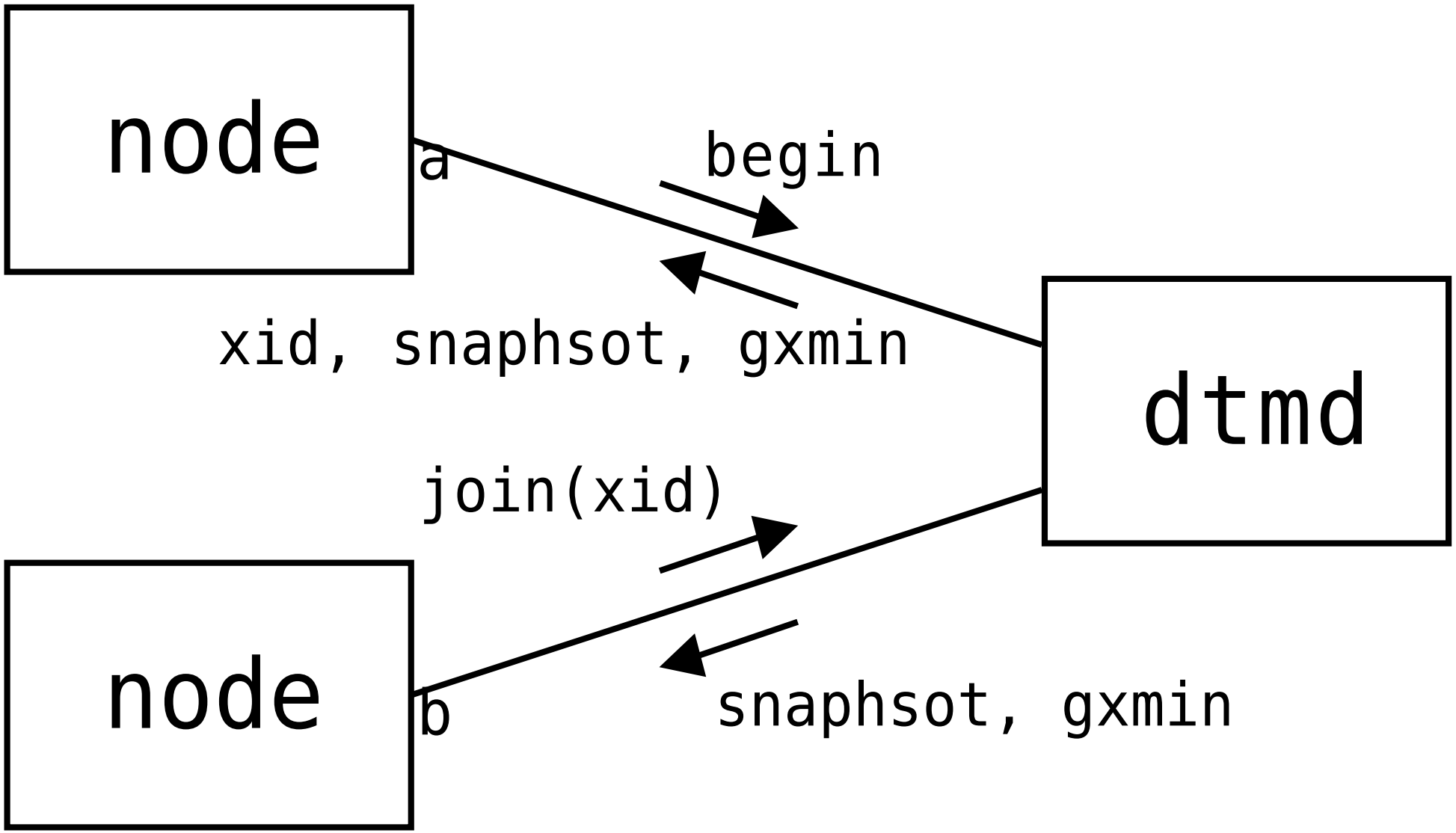
DTM architecture



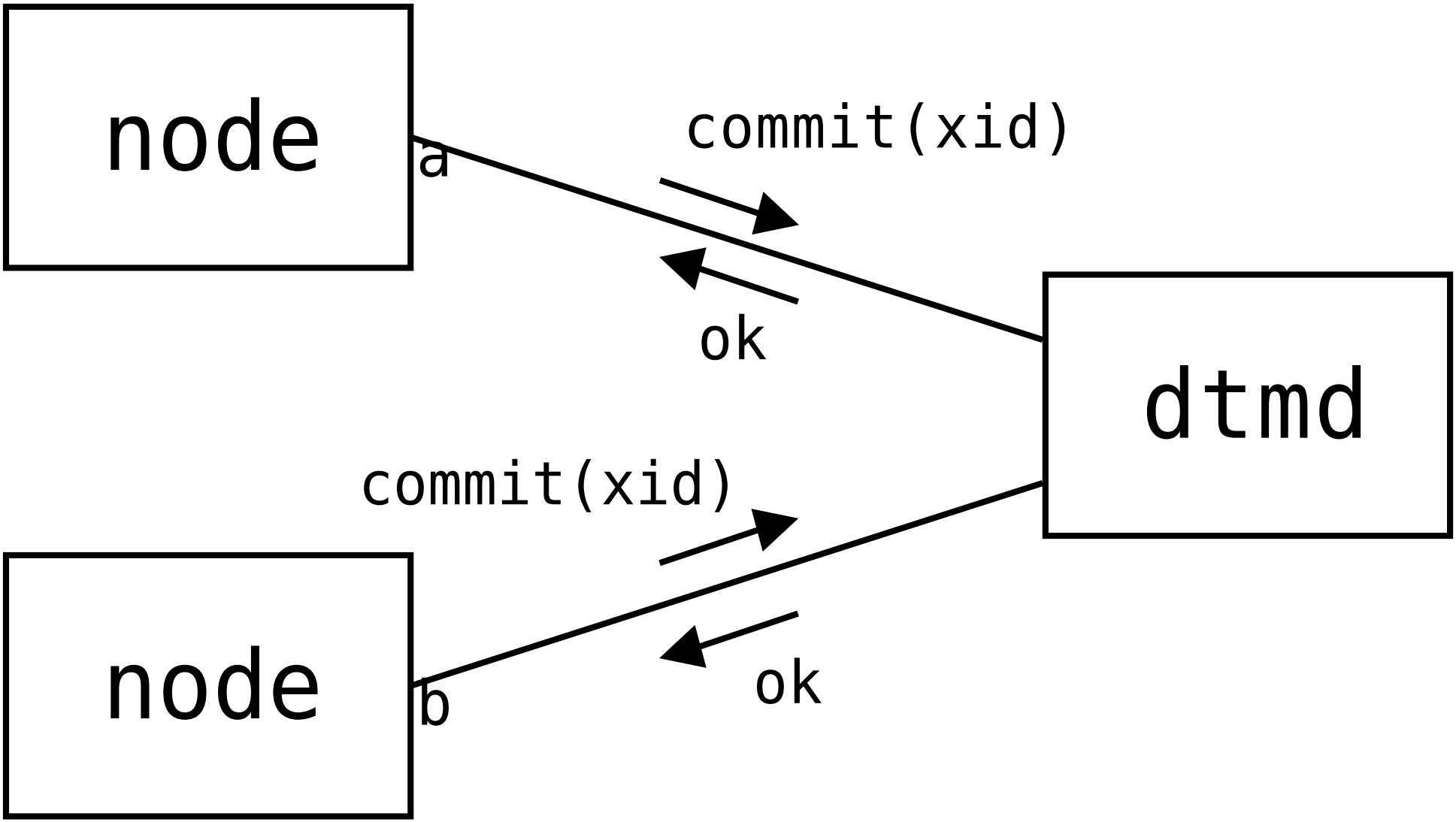
DTM from client's point of view

Primary server	Secondary server
create extension pg_dtm;	create extension pg_dtm;
select dtm_begin_transaction(); begin transaction; update...; commit;	select dtm_join_transaction(xid); begin transaction; update...; commit;

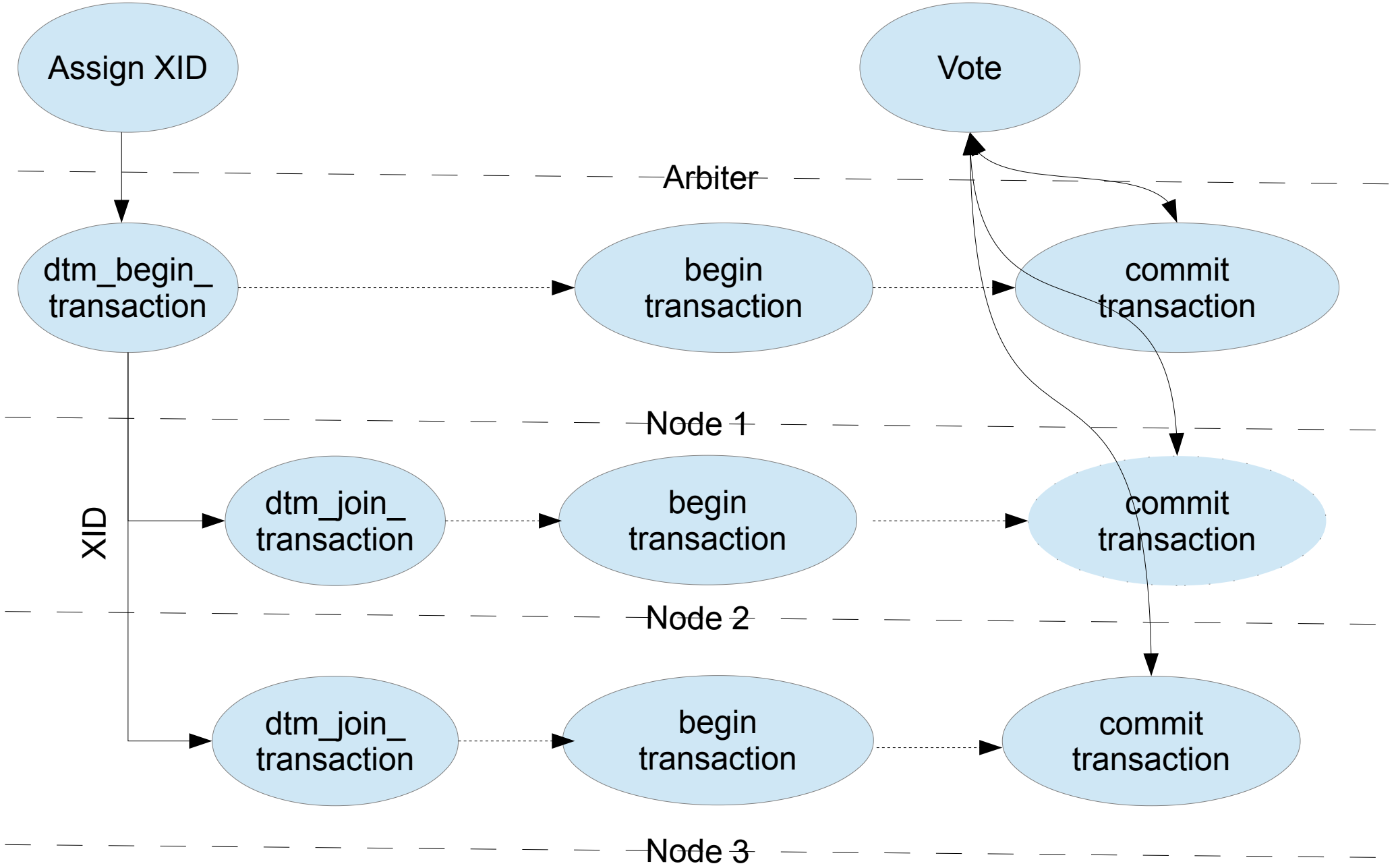
Arbiter protocol (begin)



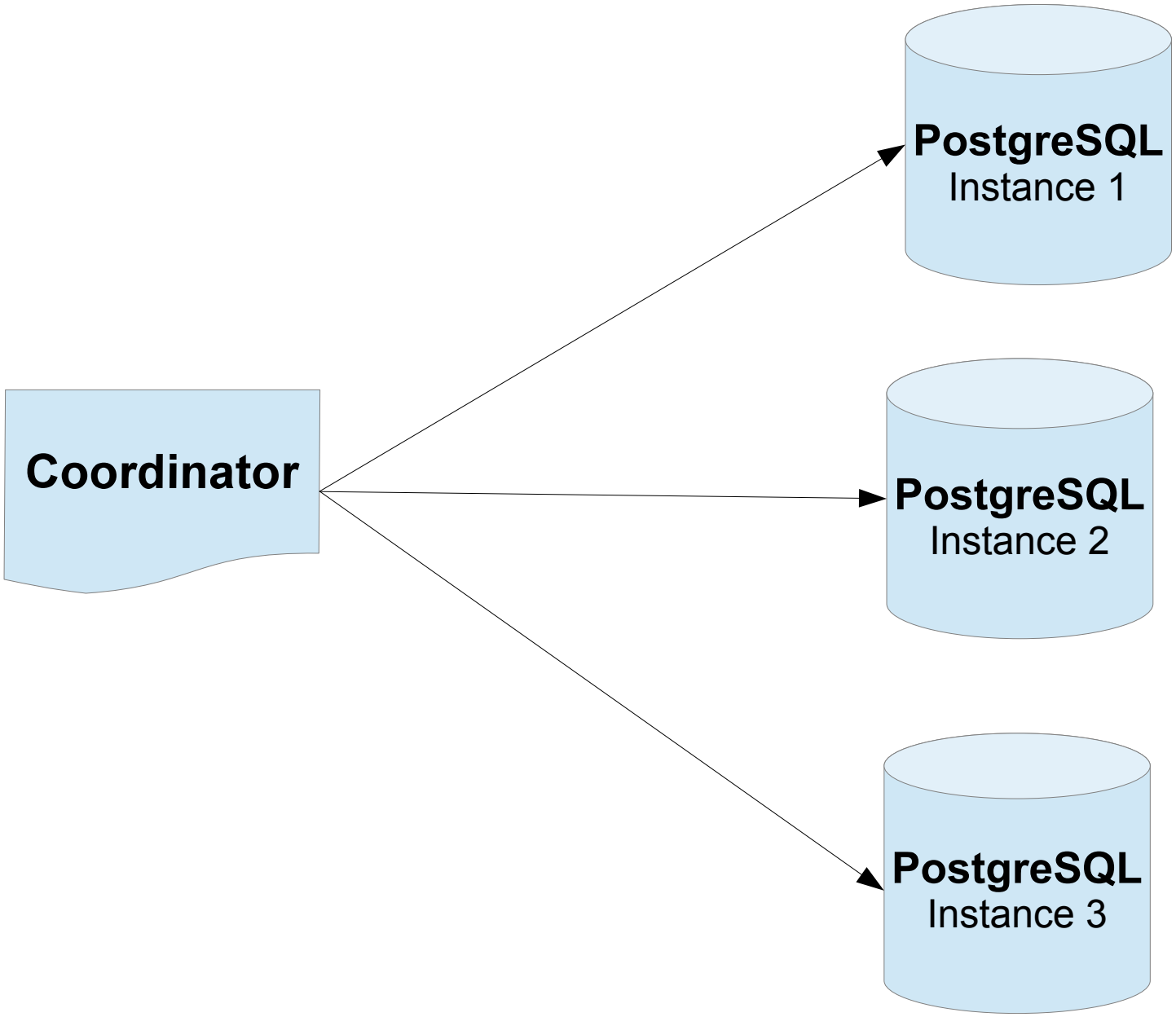
Arbiter protocol (end)



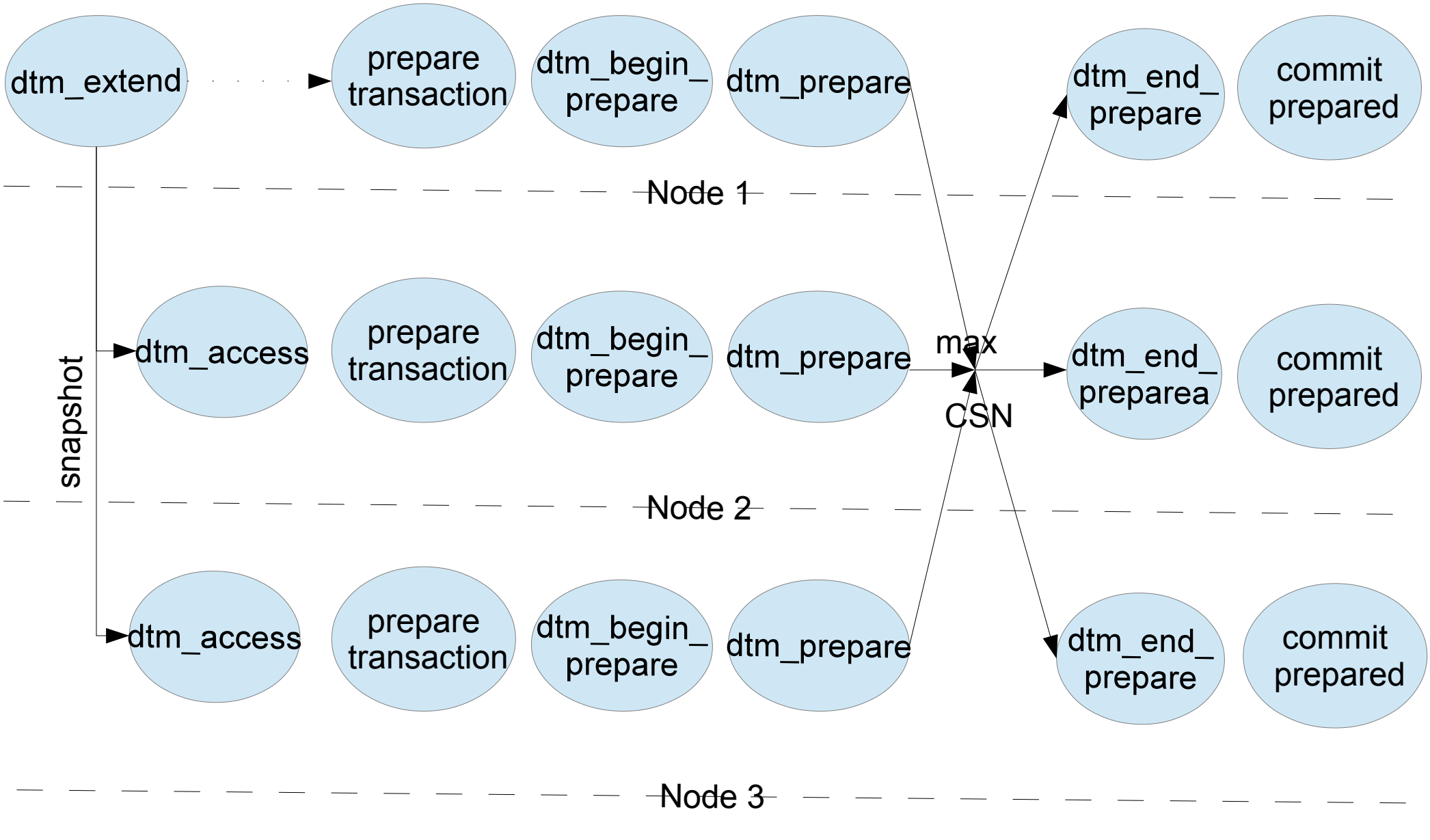
DTM transaction control flow



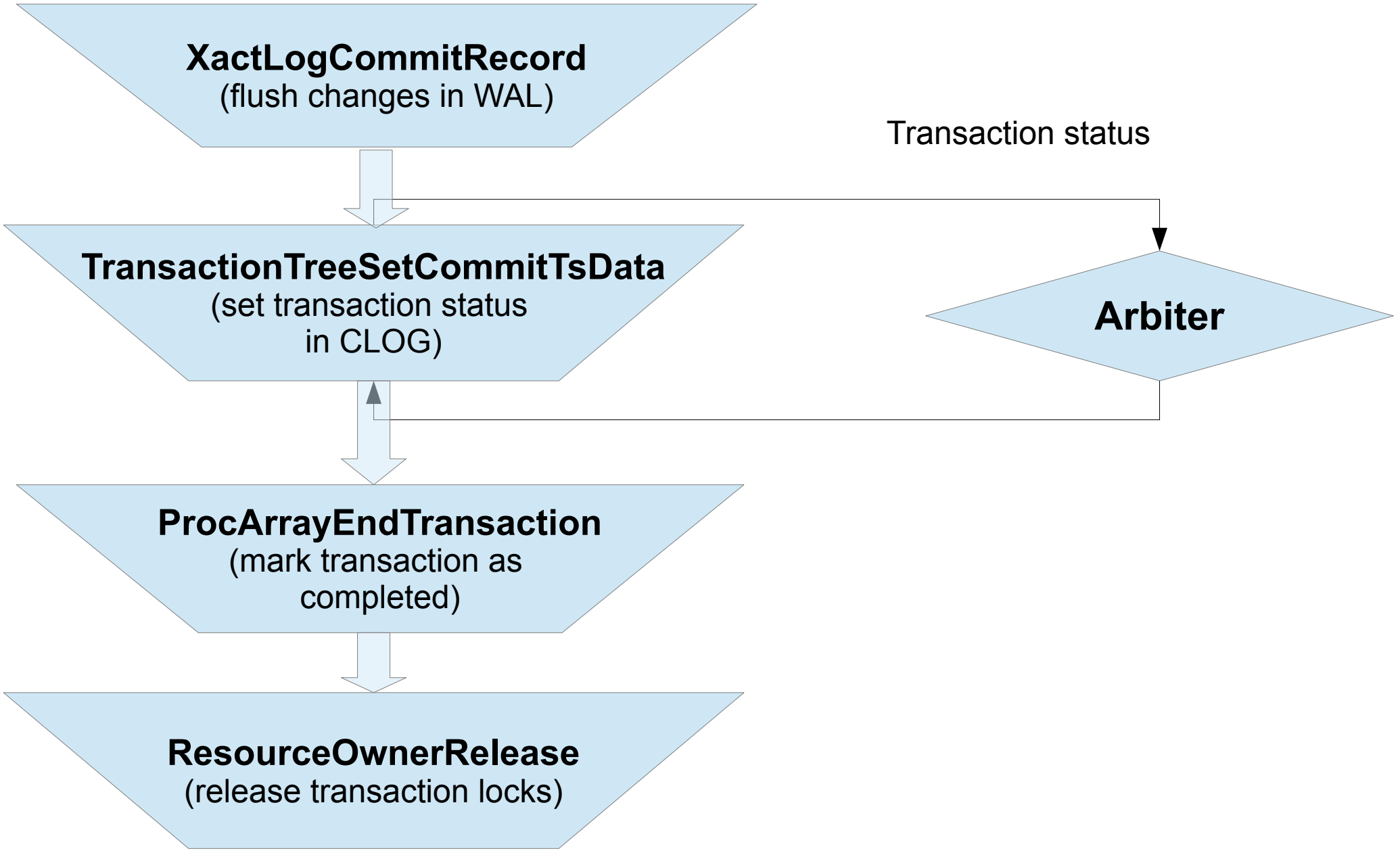
tsDTM architecture



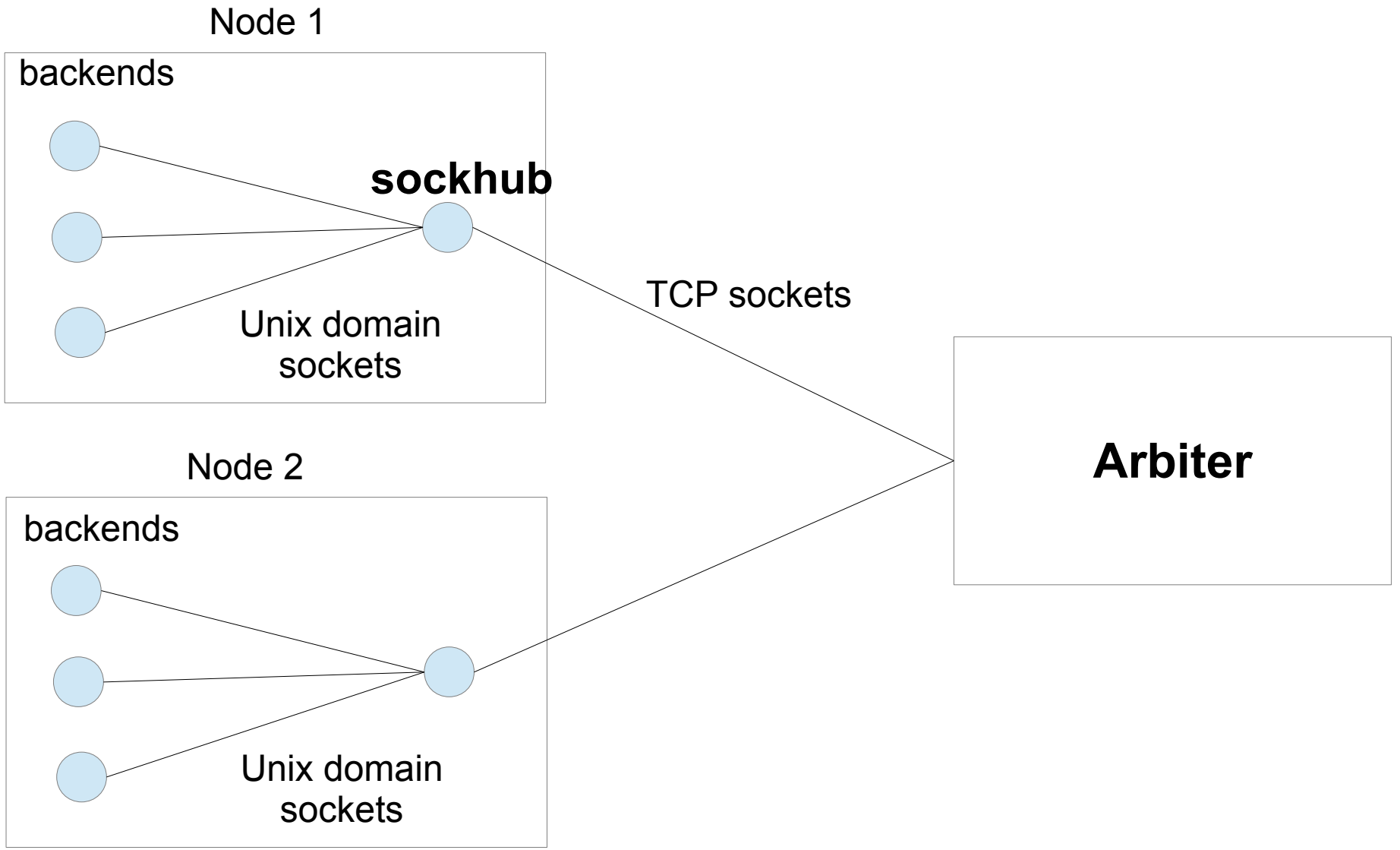
tsDTM transaction control flow



Lightweight two-phase commit



Multiplexing



Postgres PROFESSIONAL

Example of interaction with DTM

```
xid := execQuery(con1, "select dtm_begin_transaction()")
exec(con2, "select dtm_join_transaction($1)", xid)
exec(con1, "begin transaction")
exec(con2, "begin transaction")
exec(con1, "update t set v = v + $1 where u=$2", amount, account1)
exec(con2, "update t set v = v - $1 where u=$2", amount, account2)
var wg sync.WaitGroup
wg.Add(2)
asyncExec(con1, “commit”, &wg)
asyncExec(cnn2, “commit”, &wg)
wg.Wait()
```

Example of interaction with tsDTM

```
exec(con1, "begin transaction")
exec(con2, "begin transaction")
snapshot = execQuery(con1, "select dtm_extend($1)", gtid)
snapshot = execQuery(con2, "select dtm_access($1, $2)", snapshot, gtid)
exec(con1, "update t set v = v + $1 where u=$2", amount, account1)
exec(con2, "update t set v = v - $1 where u=$2", amount, account2)
exec(con1, "prepare transaction '" + gtid + "'")
exec(con2, "prepare transaction '" + gtid + "'")
exec(con1, "select dtm_begin_prepare($1)", gtid)
exec(con2, "select dtm_begin_prepare($1)", gtid)
csn = execQuery(con1, "select dtm_prepare($1, 0)", gtid)
csn = execQuery(con2, "select dtm_prepare($1, $2)", gtid, csn)
exec(con1, "select dtm_end_prepare($1, $2)", gtid, csn)
exec(con2, "select dtm_end_prepare($1, $2)", gtid, csn)
exec(con1, "commit prepared '" + gtid + "'")
exec(con2, "commit prepared '" + gtid + "'")
```

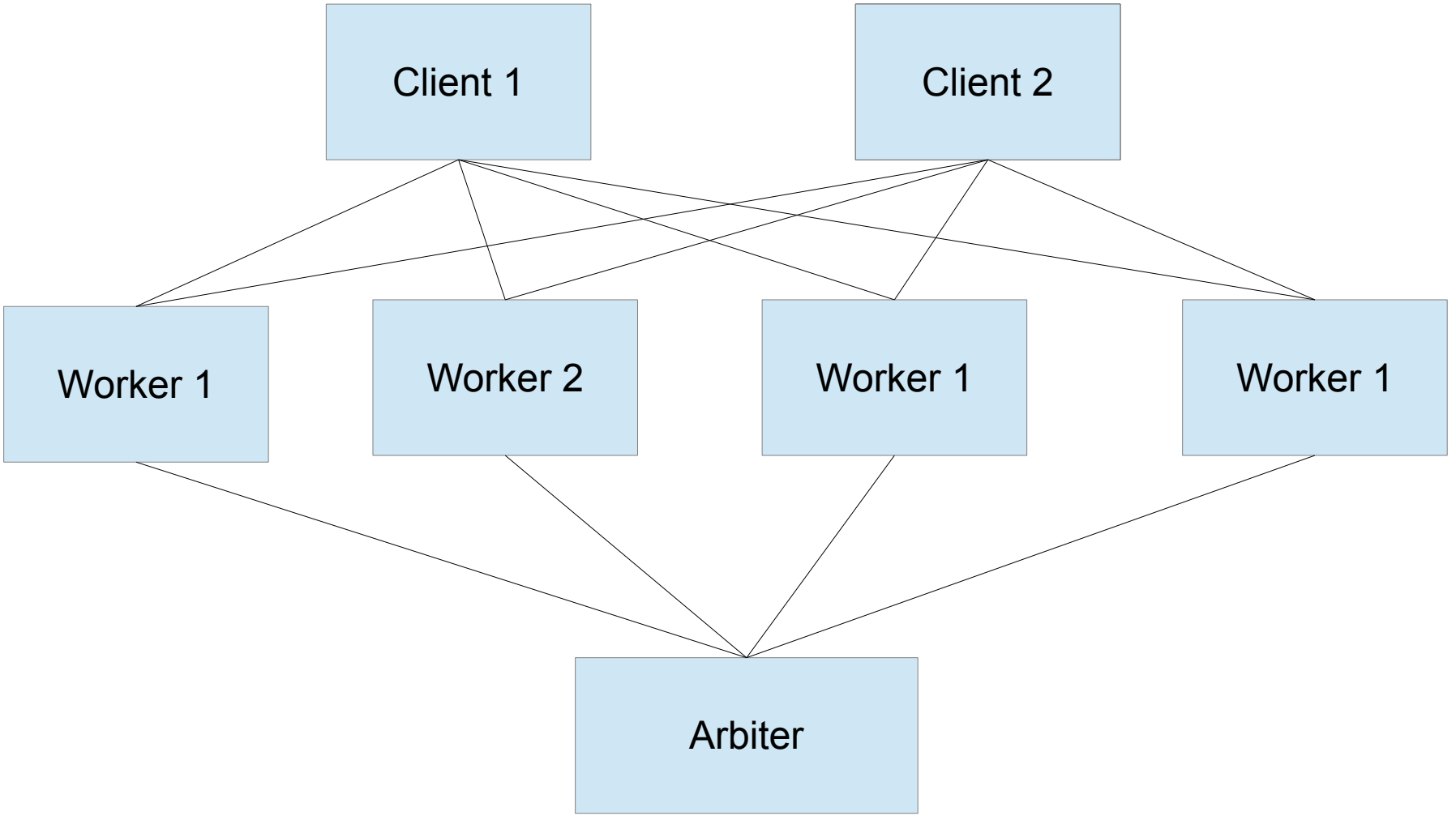
Example of using FDW

```
exec(con, "select dtm_begin_transaction()")
exec(con, "begin transaction")
exec(con, "update t set v = v + $1 where u=$2",
      amount, account1)
exec(con, "update t set v = v - $1 where u=$2",
      amount, account2)
exec(con, “commit”)
```

Example of using pg_shard

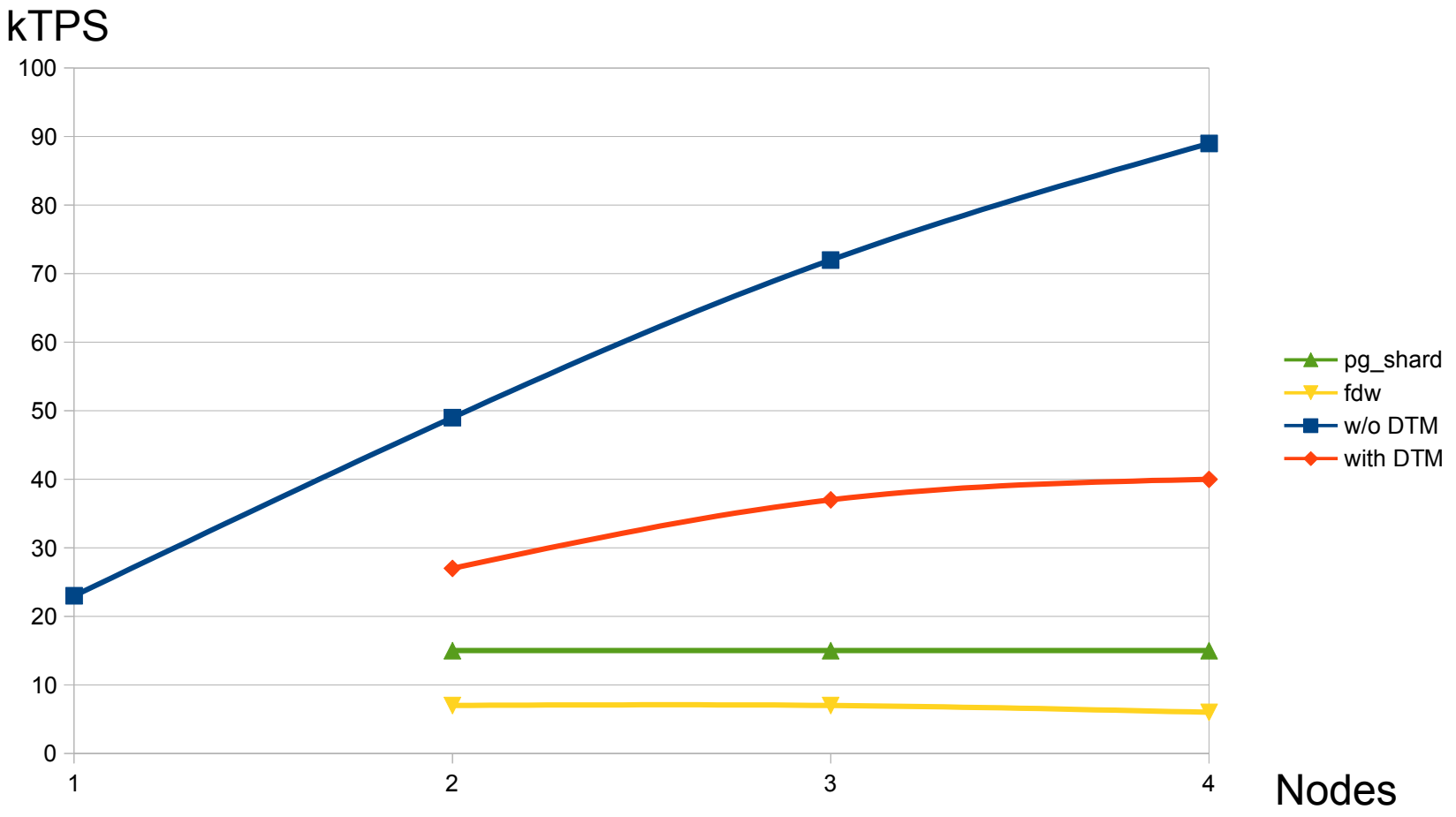
```
exec(con, "begin transaction")
exec(con, "update t set v = v + $1 where u=$2",
        amount, account1)
exec(con, "update t set v = v - $1 where u=$2",
        amount, account2)
exec(con, “commit”)
```

Test configuration



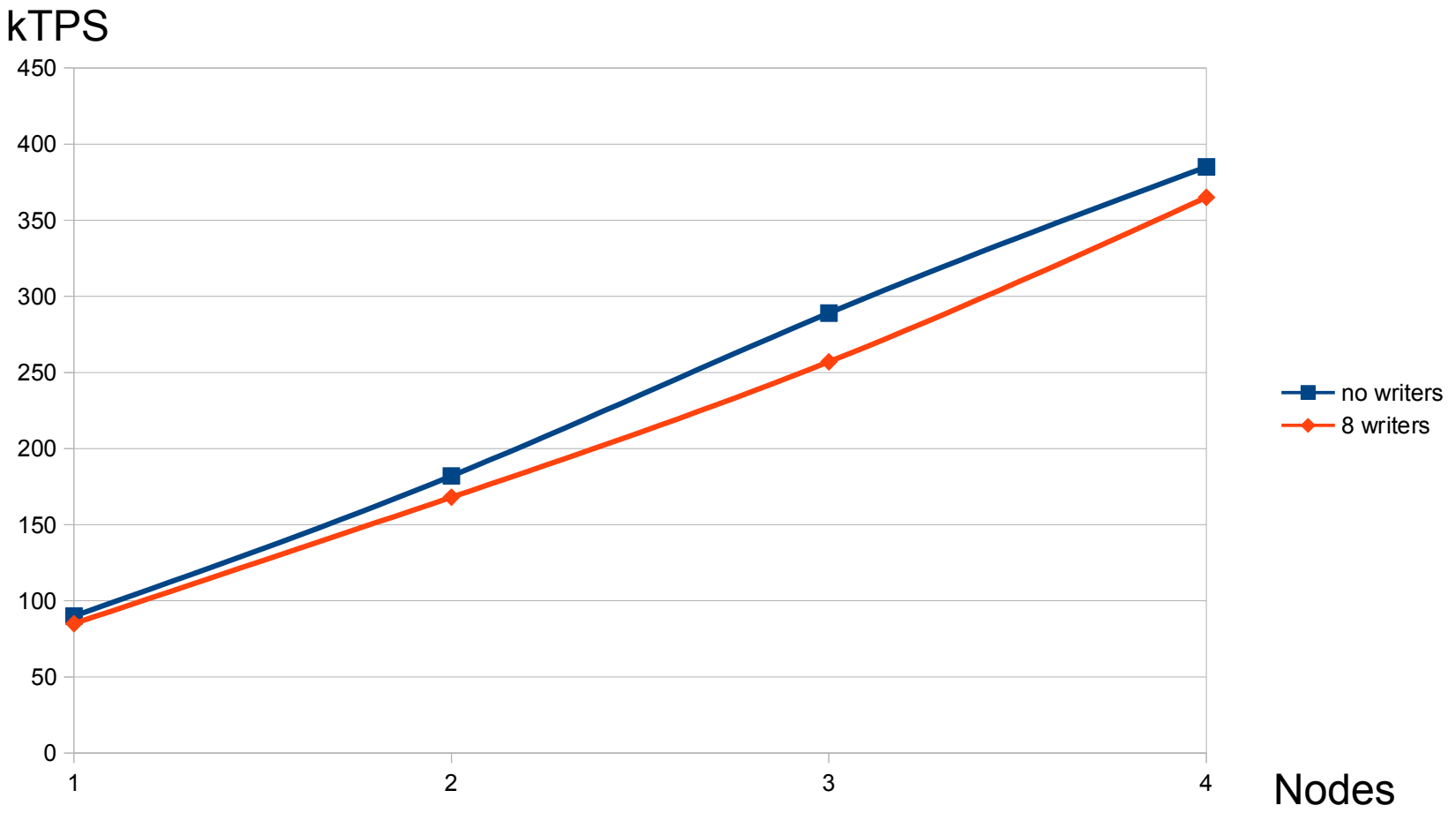
Performance measurement

Simple bank debit/credit benchmark (a-la TPC-A)
Two clients with 60 writers



Multimaster performance

Simple update/select queries
Three clients with 140 readers



Roadmap

- Add XTM patch to PostgreSQL 6
- Experiment with different DTM implementations
- Provide integration of DTM with different cluster solutions (pg_shard, FDW, XL,...)
- Implement multimaster on top of DTM