Deciphering 2phase commit

Ashwin Agrawal

Asim Praveen

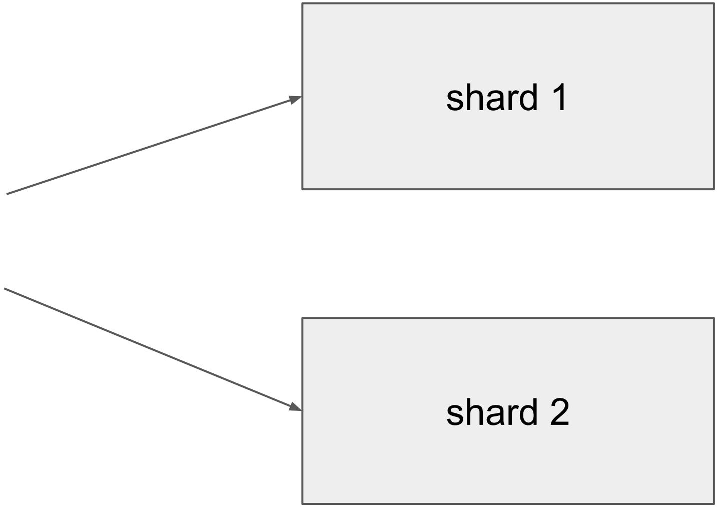
{aagrawal, apraveen}@pivotal.io

Scale out

- Single instance is limited
- Manual attempts at sharding PostgreSQL
- FDW based sharding
- MPP → distributed databases

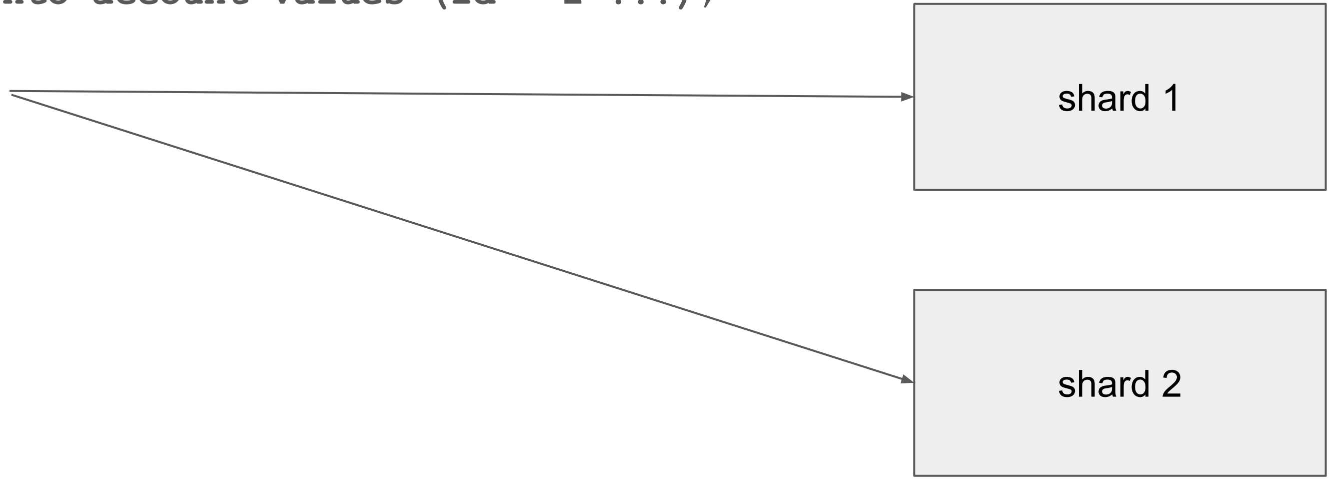
Challenge with atomicity

```
begin;

insert into account values (id = 1 ...);

insert into account values (id = 2 ...);

commit;
```



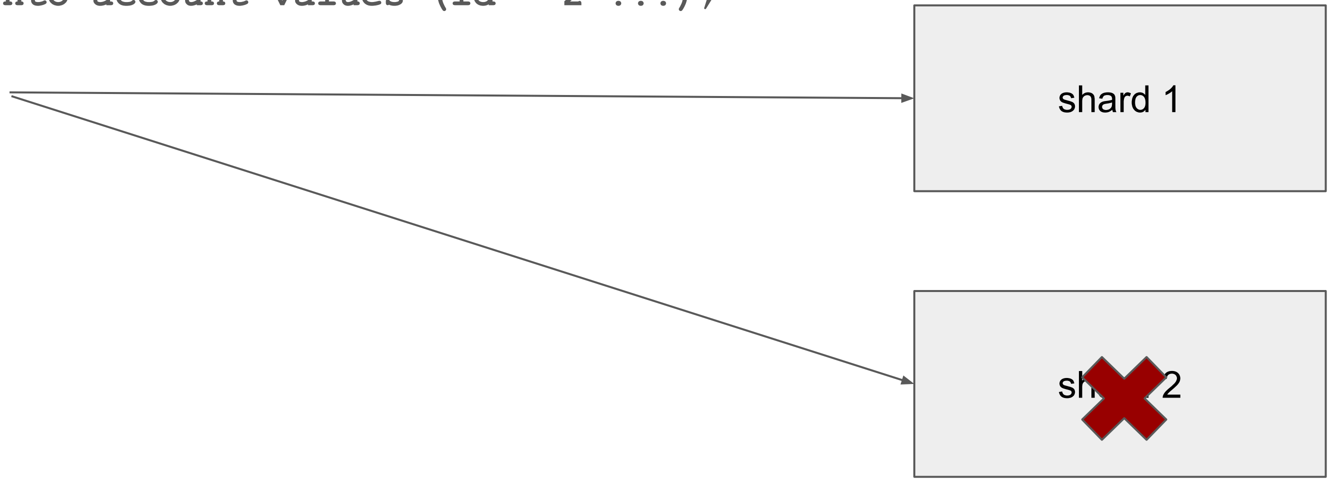
Challenge with atomicity

```
begin;

insert into account values (id = 1 ...);

insert into account values (id = 2 ...);

commit;
```

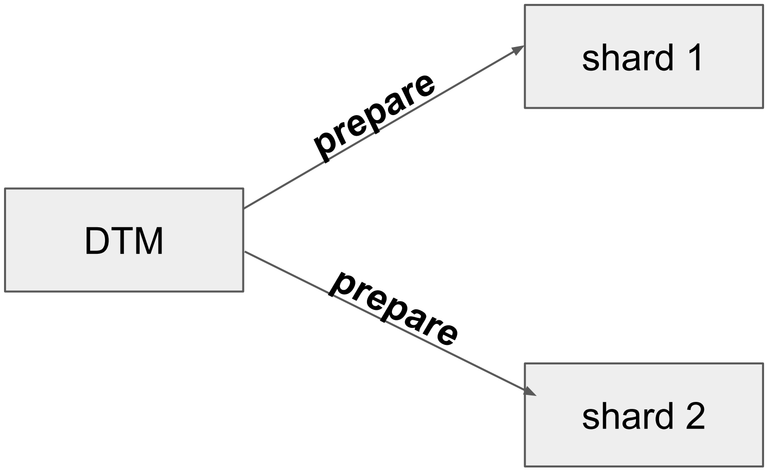


Challenge with atomicity

```
begin;

insert into account values (id = 1 ...);

insert into account values (id = 2 ...);

commit;
```

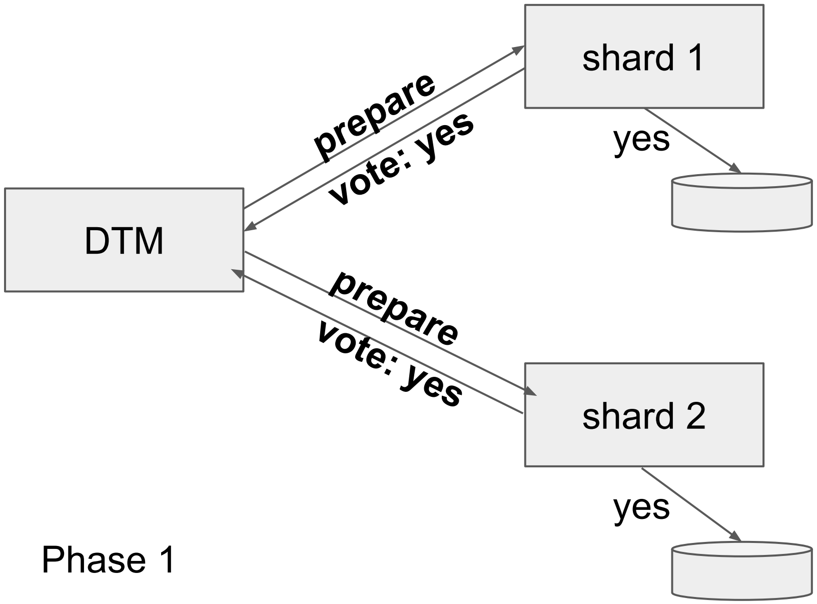


Two phase commit



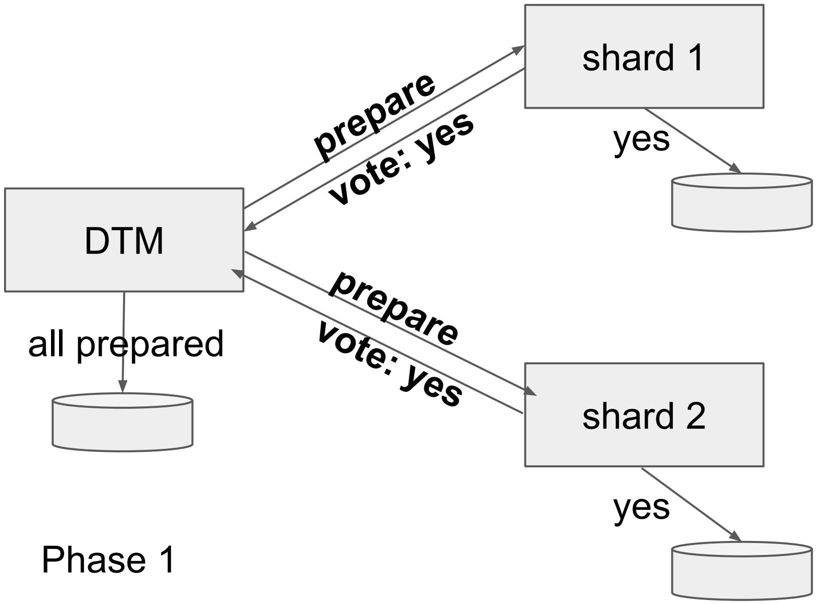
Phase 1

Two phase commit



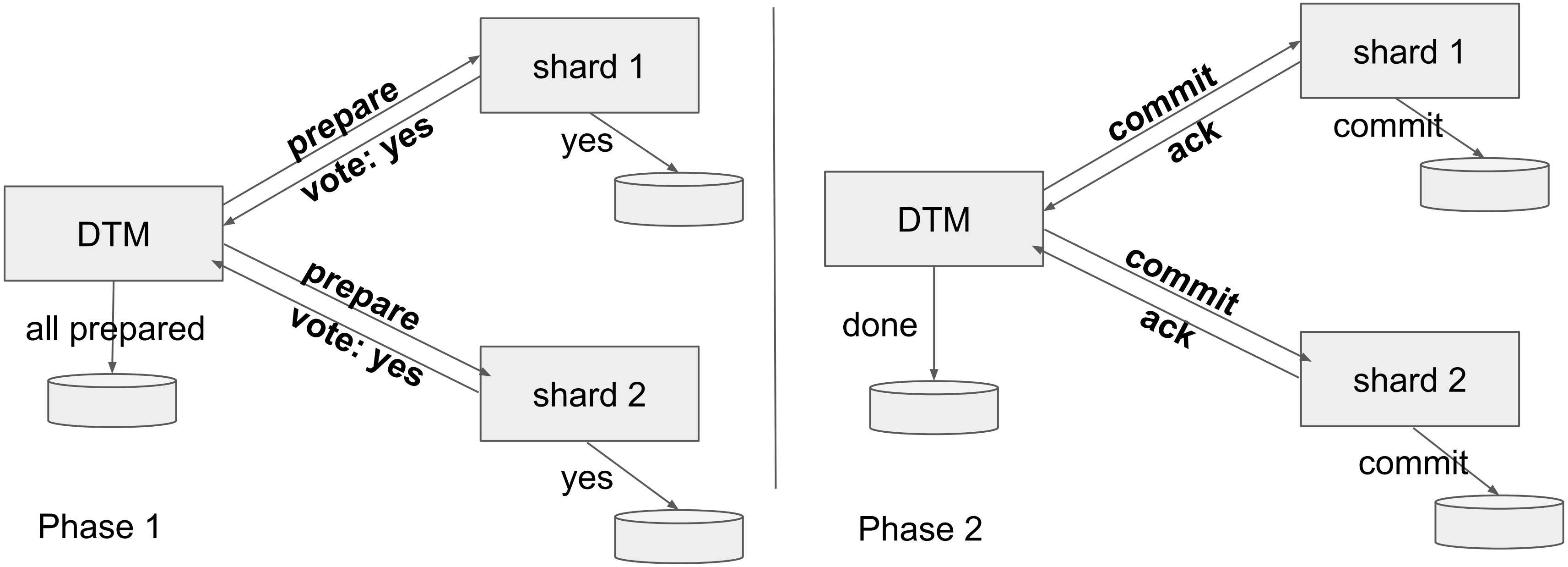
Phase 1

Two phase commit

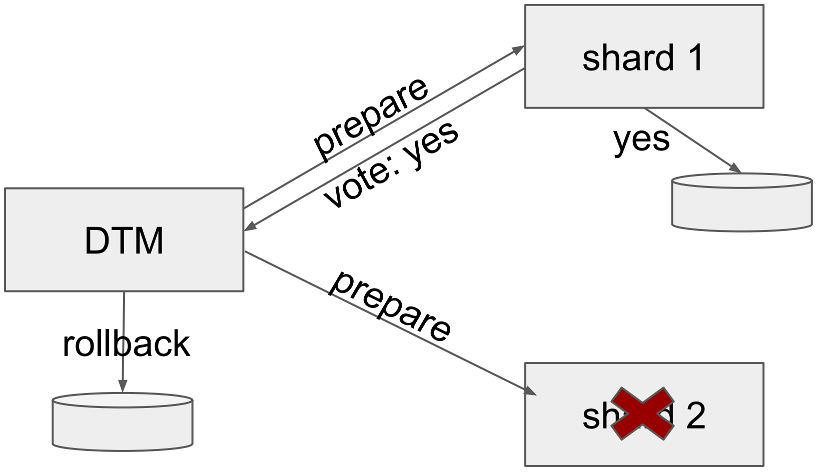


Phase 1

Two phase commit

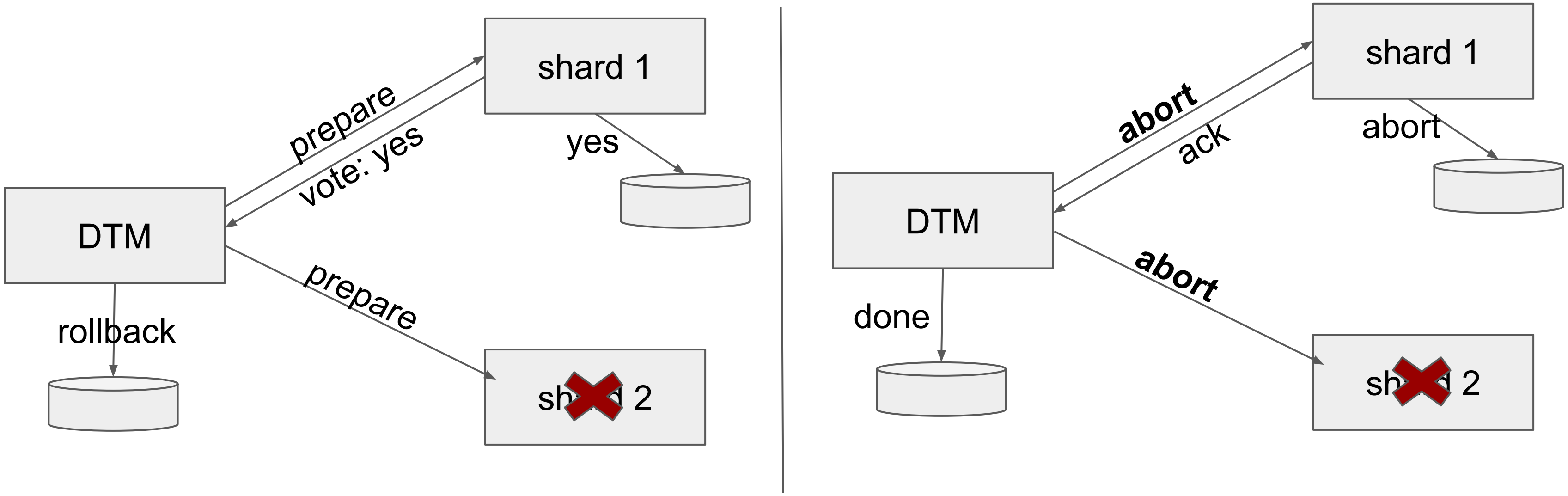


2PC: shard failure in phase one



Phase 1

2PC: shard failure in phase one



shard 1

prepare

vote: yes

yes

DTM

prepare

rollback

shard 2

shard 1

abort

ack

abort

DTM

abort

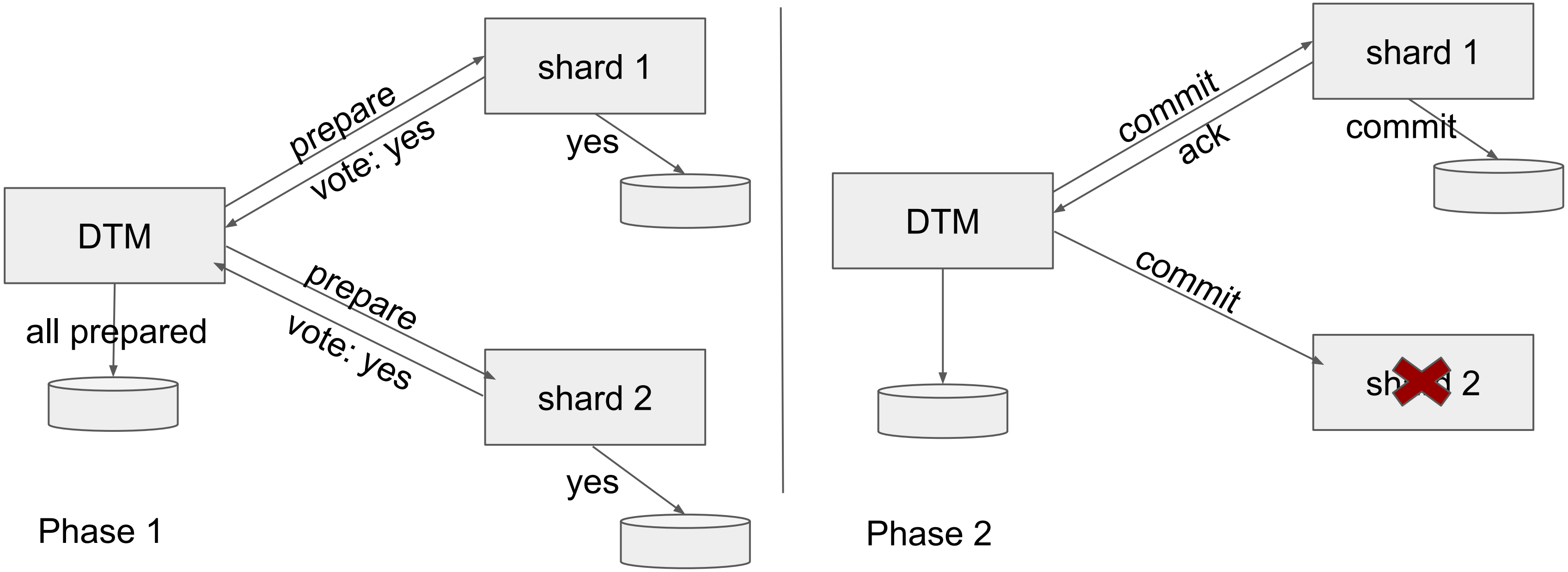
done

shard 2

Phase 1

Phase 2

2PC: shard failure in phase 2



Phase 1

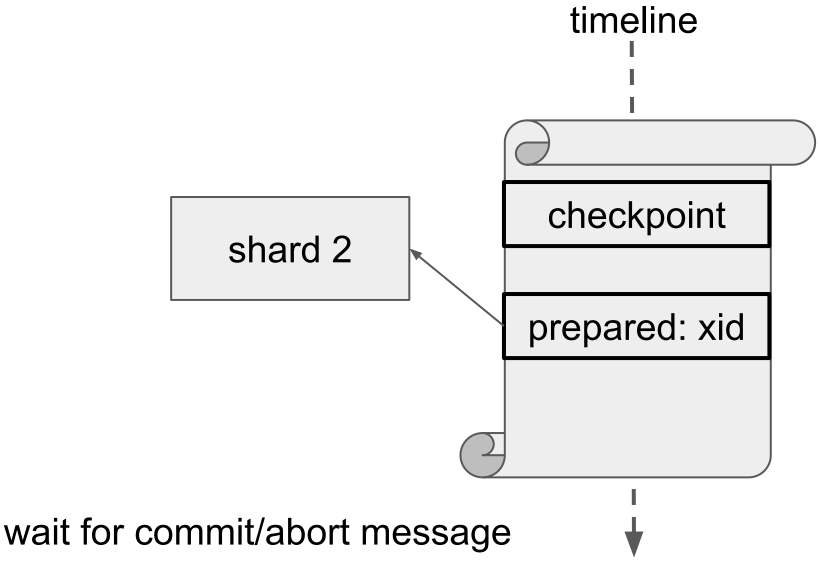
Phase 2

2PC: recovery of shard 2

DTM

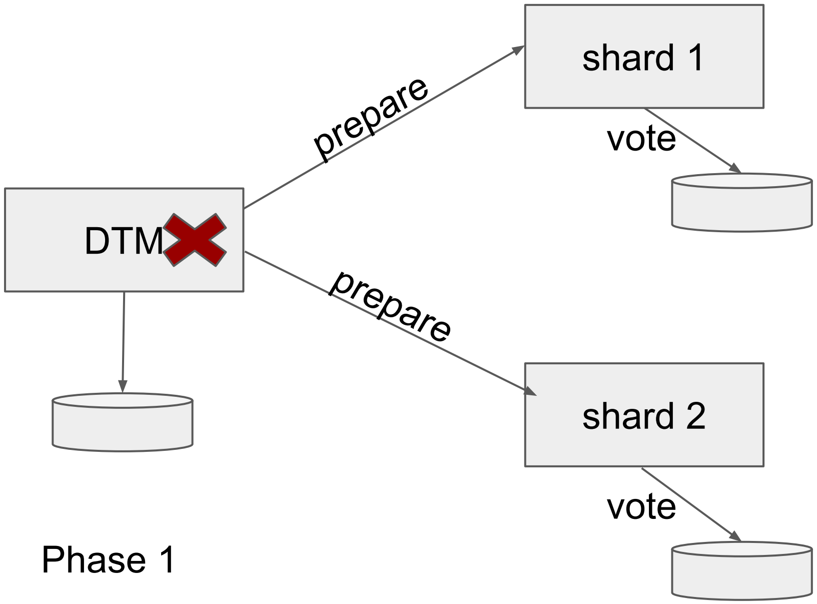
Ack not received from shard 2;

retry sending commit to shard 2



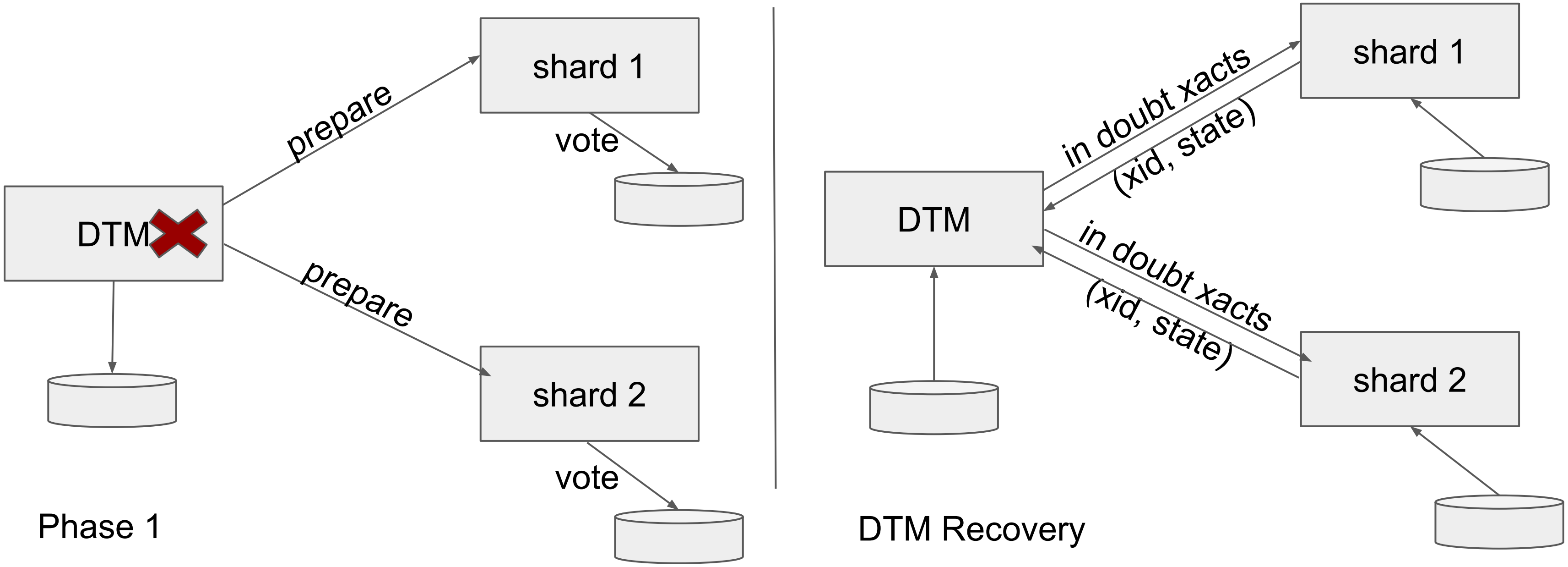
wait for commit/abort message

2PC: DTM crashed in phase 1

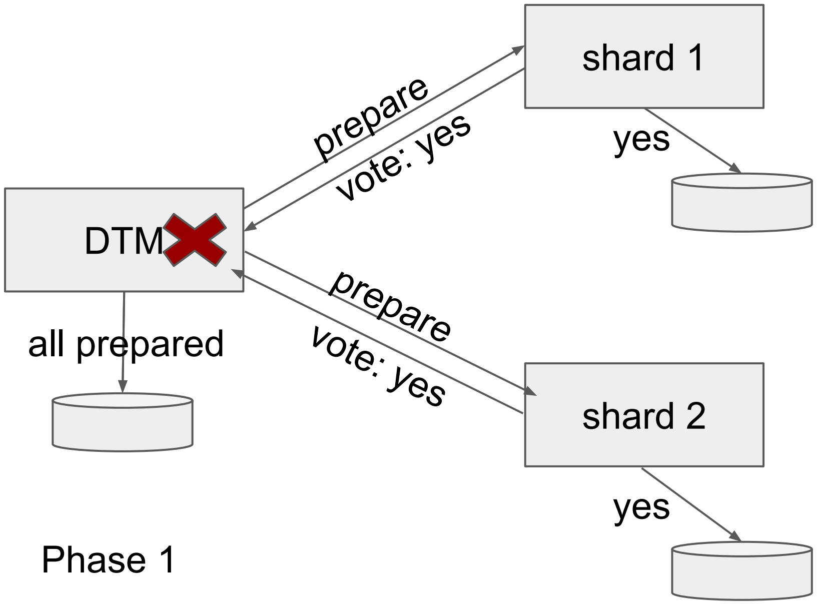


Phase 1

2PC: DTM crashed in phase 1

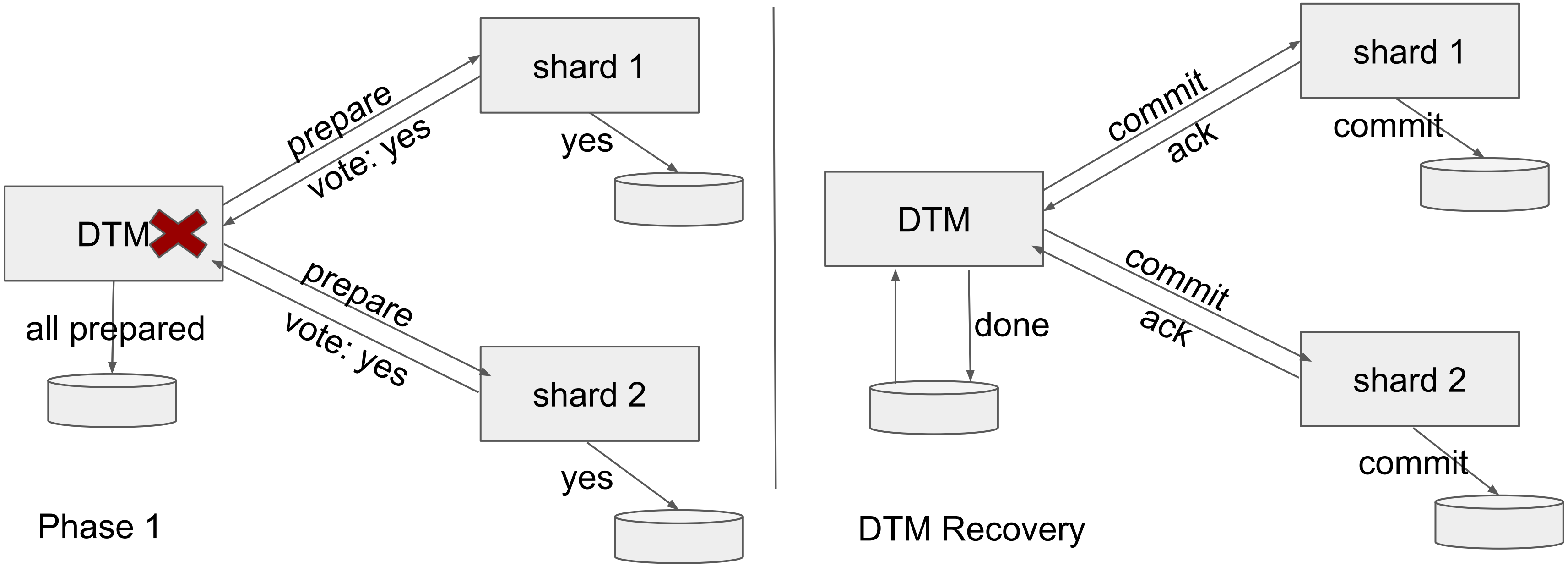


2PC: DTM crashed after phase 1



Phase 1

2PC: DTM crashed after phase 1



2PC vs 1PC

- Prepare phase
  - 1 network round trip
  - 1 disk flush
- Commit phase
  - 1 network round trip
  - 1 disk flush
- 2PC guarantees A and D of ACID

Single node snapshot isolation

Tuple headers contain:

- xmin: transaction ID of inserting transaction
- xmax: transaction ID of replacing/deleting transaction (initially NULL)

Basic idea: tuple is visible if xmin is **valid** and xmax is not. "**Valid**" means "**either committed or the current transaction**".

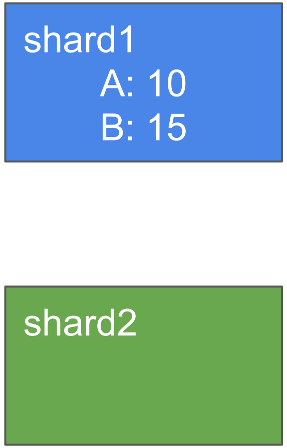
“Snapshot” filter away active transactions

Rules ensuring no transaction committing after the current transaction’s start be considered committed:

- Currently running transactions IDs never considered valid, even if shown committed in pg_clog.
- Transaction ID higher than the current transaction is not valid (future transaction).

Challenge with isolation

```
A: begin;
B: begin;

A: insert into acc values(id=1, ...);
B: insert into acc values(id=3, ...);
```



shard1

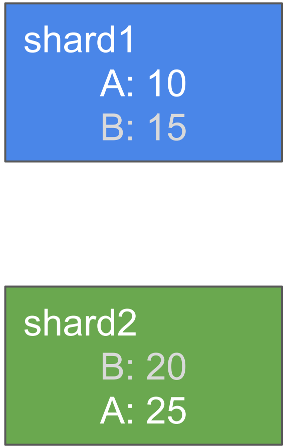
A: 10

B: 15

shard2

Challenge with isolation

```
A: begin;
B: begin;

A: insert into acc values(id=1, ...);
B: insert into acc values(id=3, ...);

B: insert into acc values(id=4, ...);
A: insert into acc values(id=2, ...);

B: commit;
```

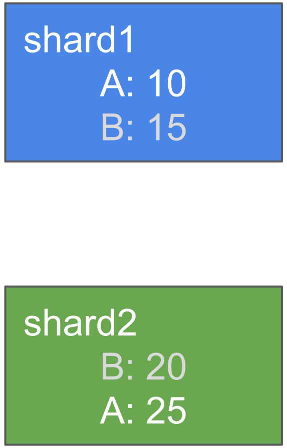


shard1
A: 10
B: 15

shard2
B: 20
A: 25

Challenge with isolation

```
A: begin;
B: begin;

A: insert into acc values(id=1, ...);
B: insert into acc values(id=3, ...);

B: insert into acc values(id=4, ...);
A: insert into acc values(id=2, ...);

B: commit;
A: select * from acc;
     1, 2, 4
```



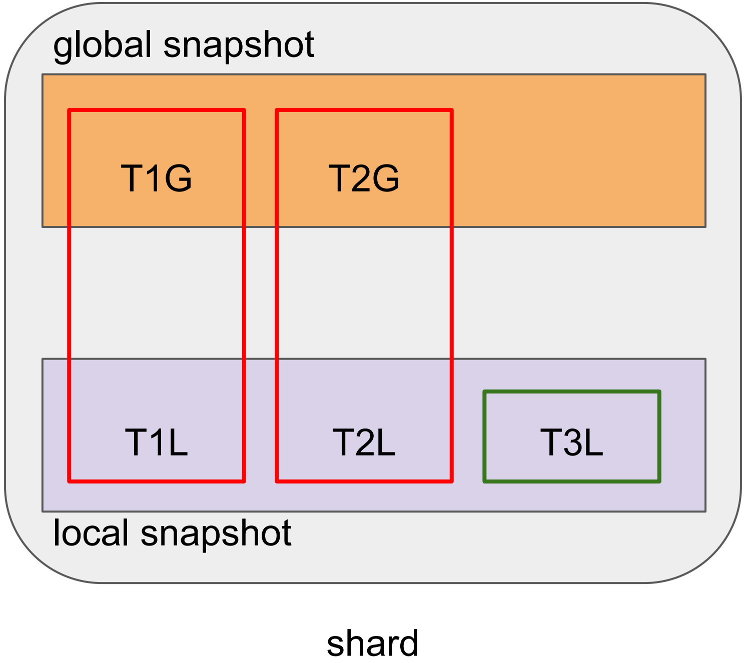
shard1
A: 10
B: 15

B is in future for A

shard2
B: 20
A: 25

B visible to A !!!

Global snapshot

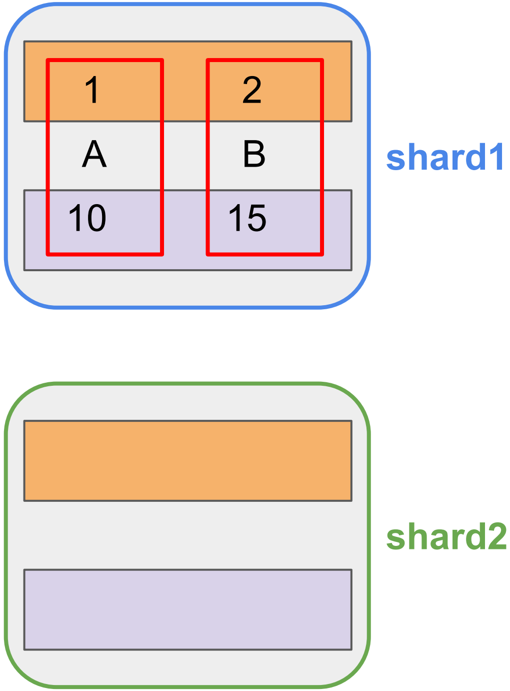


- Global xid and global snapshot provided by DTM
- Gxmin, Gxmax, gInProgress [ ]
- tuples contain local xmin/xmax

```
if (!XidInSnapshot(GS, xid))
{
    XidInSnapshot(LS, xid)
}
```

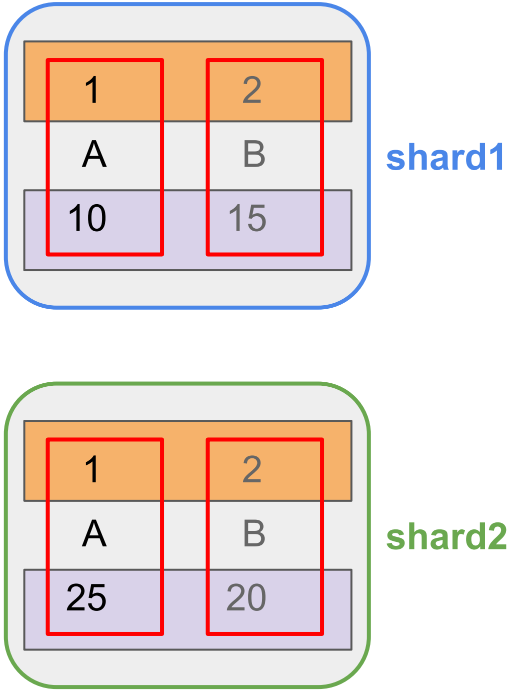
Global snapshot in action

```
A: begin; GXID: 1
B: begin; GXID: 2

A: insert into acc values(id=1, ...);
B: insert into acc values(id=3, ...);
```



Global snapshot in action

```
A: begin; GXID: 1
B: begin; GXID: 2

A: insert into acc values(id=1, ...);
B: insert into acc values(id=3, ...);

B: insert into acc values(id=4, ...);
A: insert into acc values(id=2, ...);

B: commit;
```



1	2
A	B
10	15

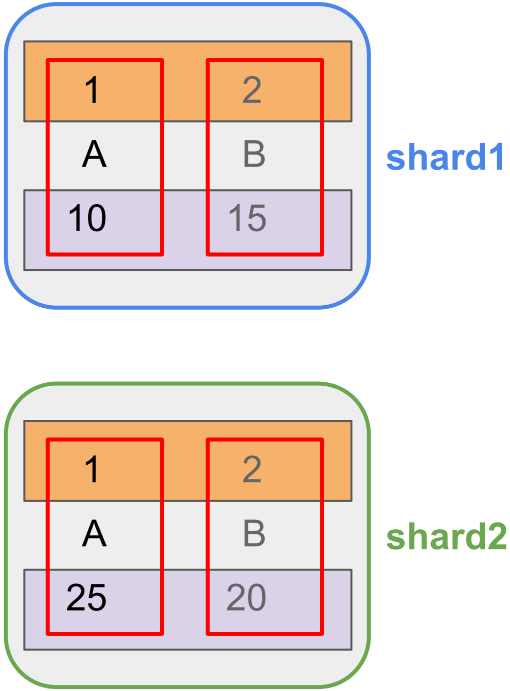
shard1

1	2
A	B
25	20

shard2

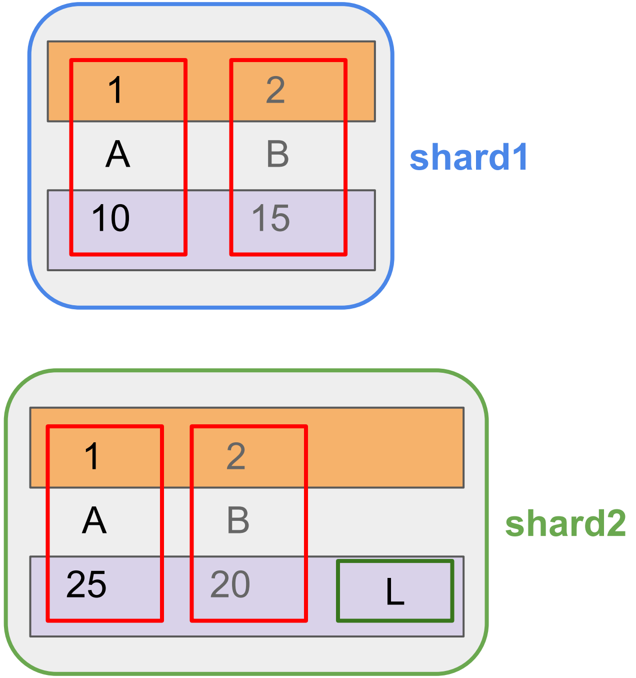
Global snapshot in action

```
A: begin; GXID: 1
B: begin; GXID: 2

A: insert into acc values(id=1, ...);
B: insert into acc values(id=3, ...);

B: insert into acc values(id=4, ...);
A: insert into acc values(id=2, ...);

B: commit;
A: select * from acc;
      1, 2
```



Global snapshot with local transaction

```
A: begin; GXID: 1
B: begin; GXID: 2

A: insert into acc values(id=1, ...);
B: insert into acc values(id=3, ...);

B: insert into acc values(id=4, ...);
A: insert into acc values(id=2, ...);
L: select * from acc;
     /* 0 rows */
B: commit;
L: select * from acc;
     4
```



Implementation options

- Model
  - Pull
    - DTM as a service
    - participants join a transaction
    - transaction can be initiated by any participant
  - Push
    - DTM initiates transaction and decides participants
    - MPP databases

Implementation options

- Transactions
  - Global and local
    - global and local transaction IDs, snapshots
    - mapping between global and local transaction IDs
  - Global only
    - only one xid and snapshot across the cluster

ACID distributed system

- Two phase commit ⇒ A and D
- Global snapshot ⇒ I

?? Spot the Problem ??

```
A: begin;
A: update acc set ... where id = 1
B: begin;
B: update acc set ... where id = 2
B: update acc set ... where id = 1
A: update acc set ... where id = 2
```

ACID distributed system

- **Two phase commit ⇒ A and D**
- **Global snapshot ⇒ I**

?? Spot the Problem ??

```
A: begin;
A: update acc set ... where id = 1
B: begin;
B: update acc set ... where id = 2
B: update acc set ... where id = 1
A: update acc set ... where id = 2
```

- Global lock manager ⇒ C

thank you
danke
рахмат
謝謝
ngiyabonga
teşekkür ederim
Баярлалаа
спасибо
faafetai lava
mersi
barka
kia ora
spas
welalin
tack
dank je
misaotra
matondo
paldies
grazzi
mahalo
tapadh leat
gracias
vinaka
спасибі
blagodaram
kiitos
dankie
nanni
nandri
dhanyavad
mauruuru
hvala
köszönöm
bedankt
bayarlalaa
gràcie
enkosi
dziękuję
akun
dankon
aciu
хвала
asante
manana
obrigada
tenki
murakoze
chokrane
djiere dieuf
tau
mochchakkeram
дякую
mamnun
chnorakaloutioun
gratias ago
gràcies
sulpáy
go raibh maith agat
sobodi
dĕkuji
obrigado
sukriya
sağolun
kop khun krap
taiku
arigatô
takk
dakujem
trugarez
mèsi
didi madloba
najis tuke
ありがとう
tanemirt
grazie
rahmet
terima kasih
kam sah hamnida
rahmat
ευχαριστώ
diolch
dhanyavadagalu
shukriya
merce
мерси
তোমাকে ধন্যবাদ
감사합니다
xièxie
merci