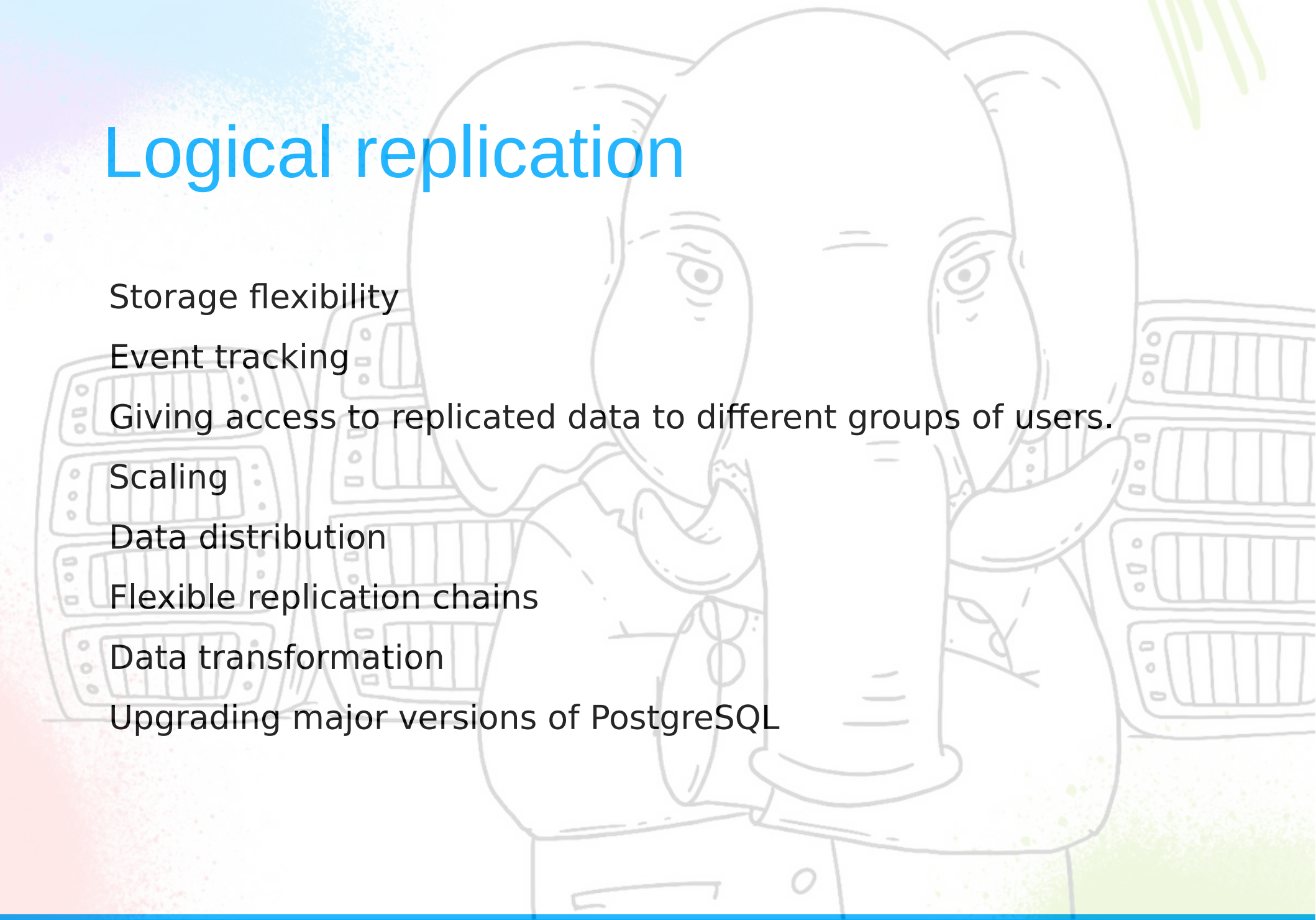
Recovery Use Cases for Logical Replication in PostgreSQL 10

Konstantin Evteev

Mikhail Tyurin

Ottawa 2018

Logical replication



Storage flexibility

Event tracking

Giving access to replicated data to different groups of users.

Scaling

Data distribution

Flexible replication chains

Data transformation

Upgrading major versions of PostgreSQL

Logical replication in Avito

Data streams in Avito (https://pgconf.ru/2016/89825)
Building data streams (https://pgday.ru/ru/2016/papers/79)
Logical replication in Avito (https://goo.gl/xSBXeT)

Dictionaries delivery

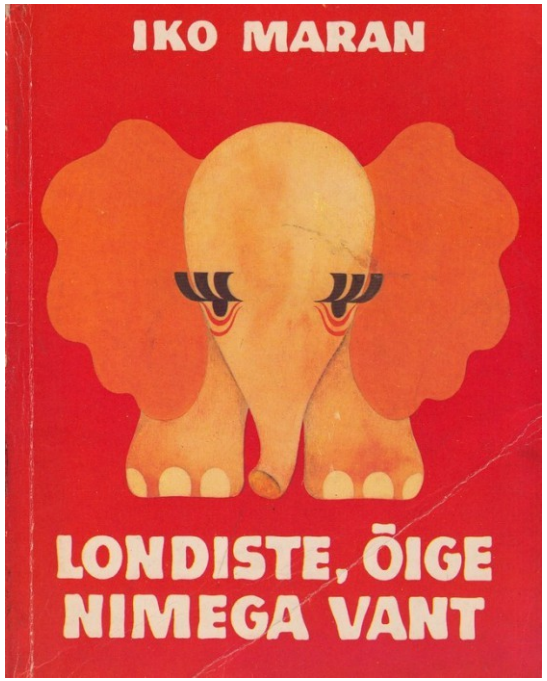
Load balancing

Partial replication to services

Data streaming to search systems

Persistent queue

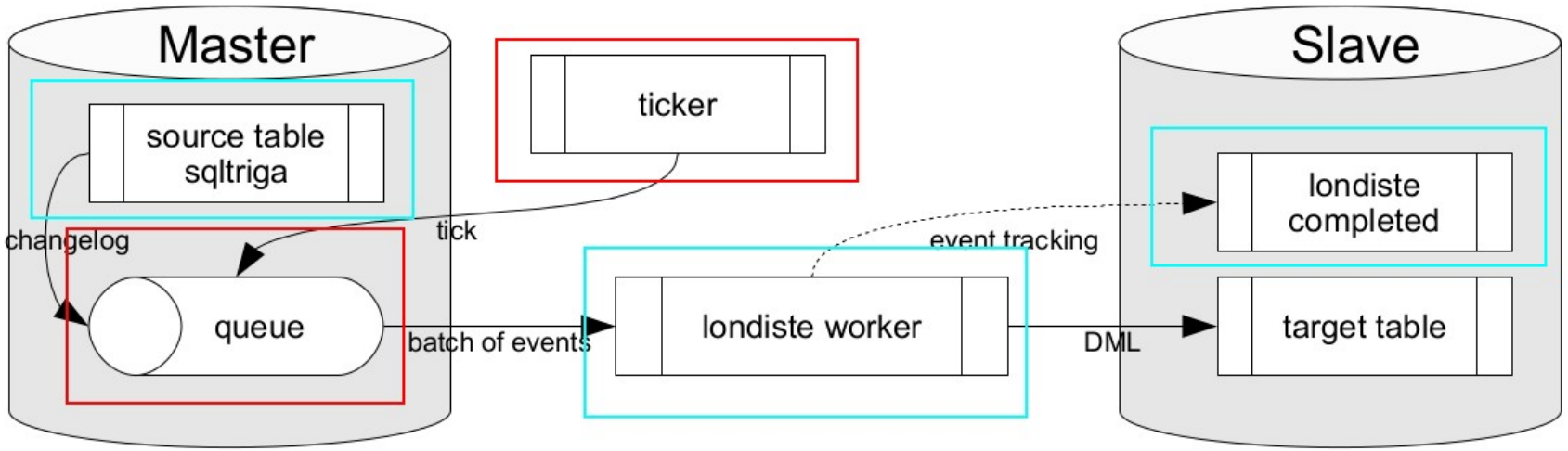
Interservice communication



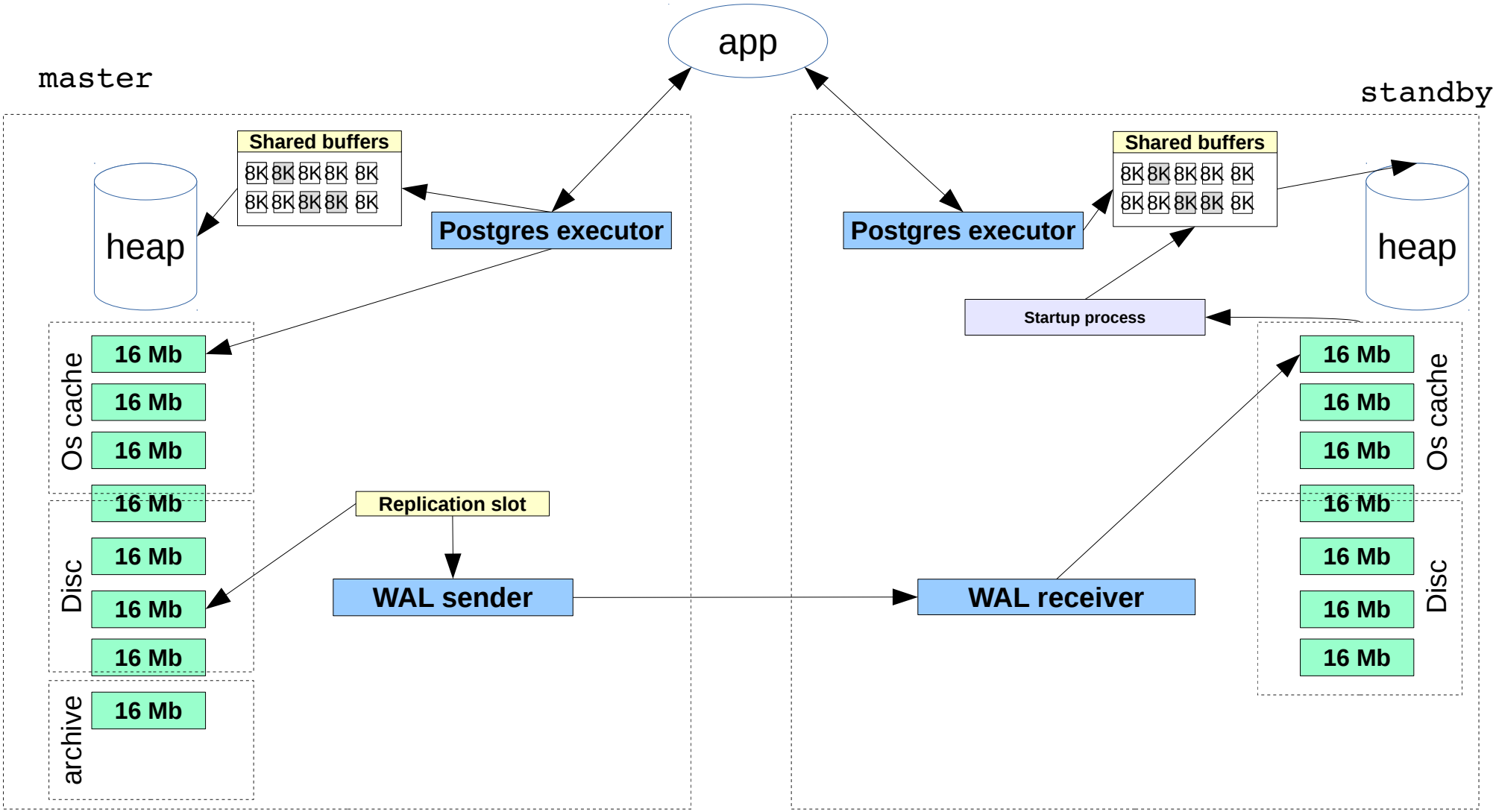
Londiste

https://www.pgcon.org/2009/schedule/attachments/91_pgq.pdf

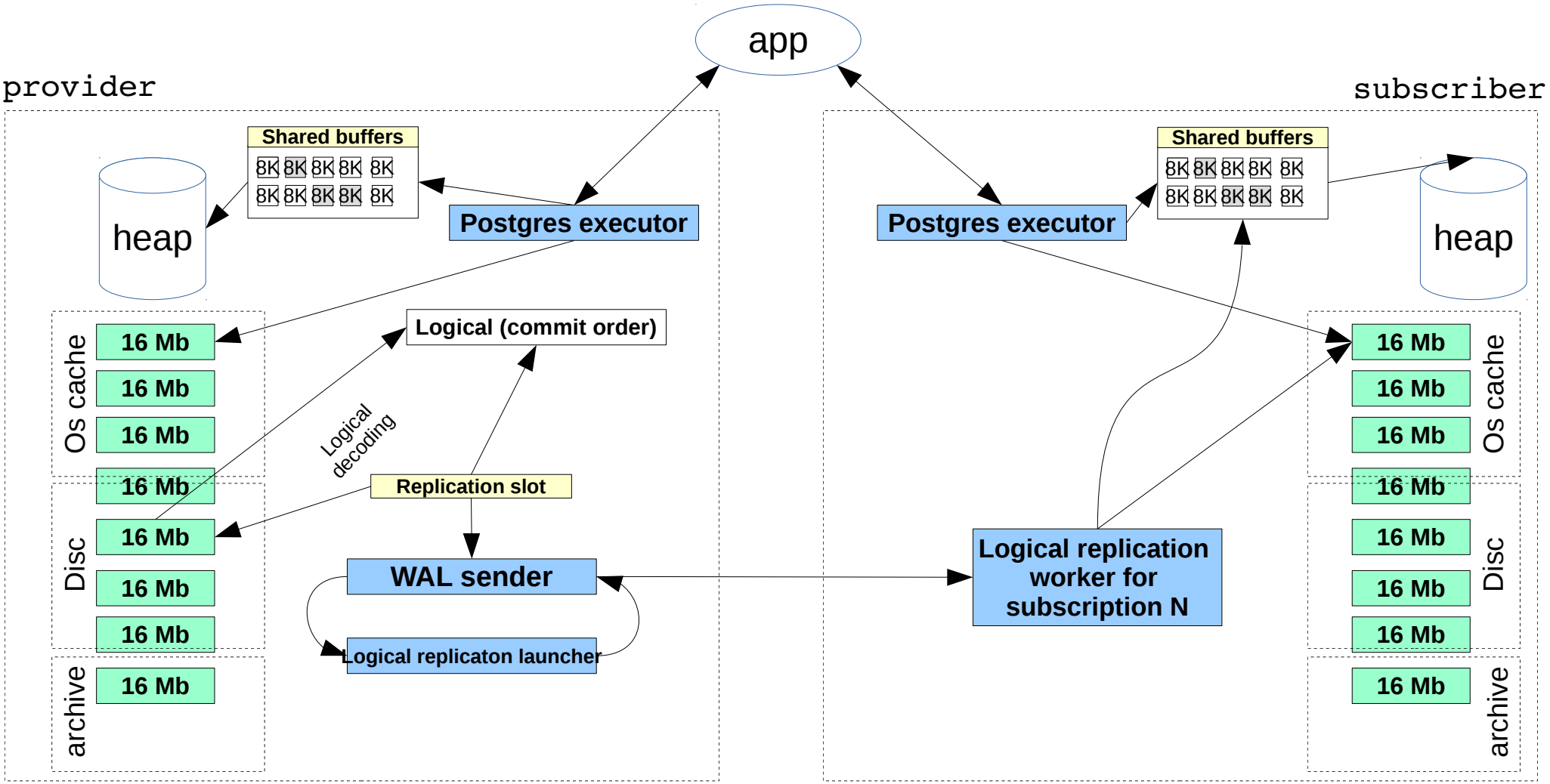
https://github.com/avito-tech/skytools



Streaming


app
master
standby
Shared buffers
8K 8K 8K 8K 8K
8K 8K 8K 8K 8K
heap
Postgres executor
Postgres executor
Shared buffers
8K 8K 8K 8K 8K
8K 8K 8K 8K 8K
heap
Startup process
Os cache
16 Mb
16 Mb
16 Mb
16 Mb
Disc
16 Mb
16 Mb
16 Mb
archive
16 Mb
Replication slot
WAL sender
WAL receiver
16 Mb
16 Mb
16 Mb
Os cache
16 Mb
16 Mb
16 Mb
Disc
16 Mb

Logical


app
provider
subscriber
Shared buffers
8K 8K 8K 8K 8K
8K 8K 8K 8K 8K
heap
Postgres executor
Logical (commit order)
Os cache
16 Mb
Logical decoding
Replication slot
Disc
WAL sender
Logical replicaton launcher
archive
Logical replication worker for subscription N

Recovery use cases

(1) Reinitializing subscriber from another subscriber

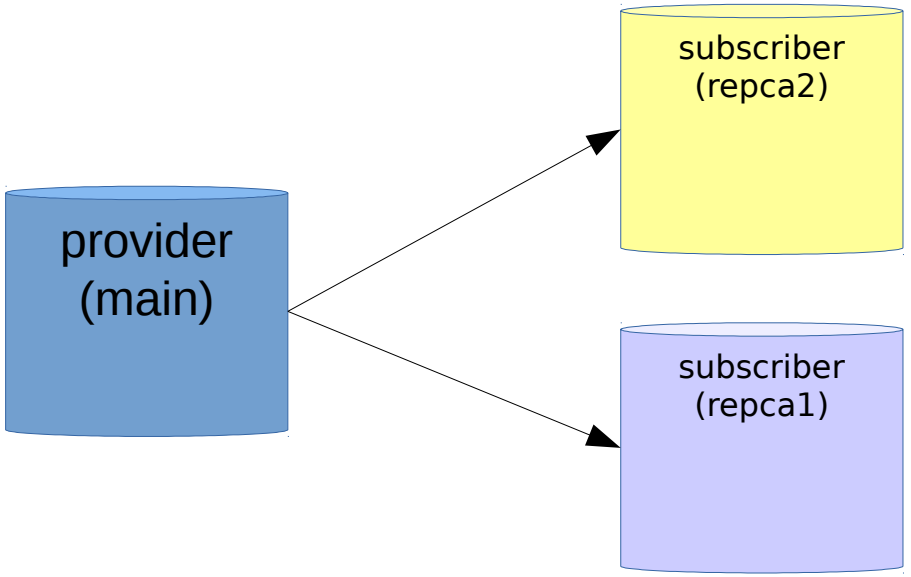
(2) UNDO recovery on the destination side

(3) REDO - reposition source (subscriber's crash)

(4) REDO 2 - on provider's side (provider's crash and switching to the provider's standby, subscriber is falling behind)

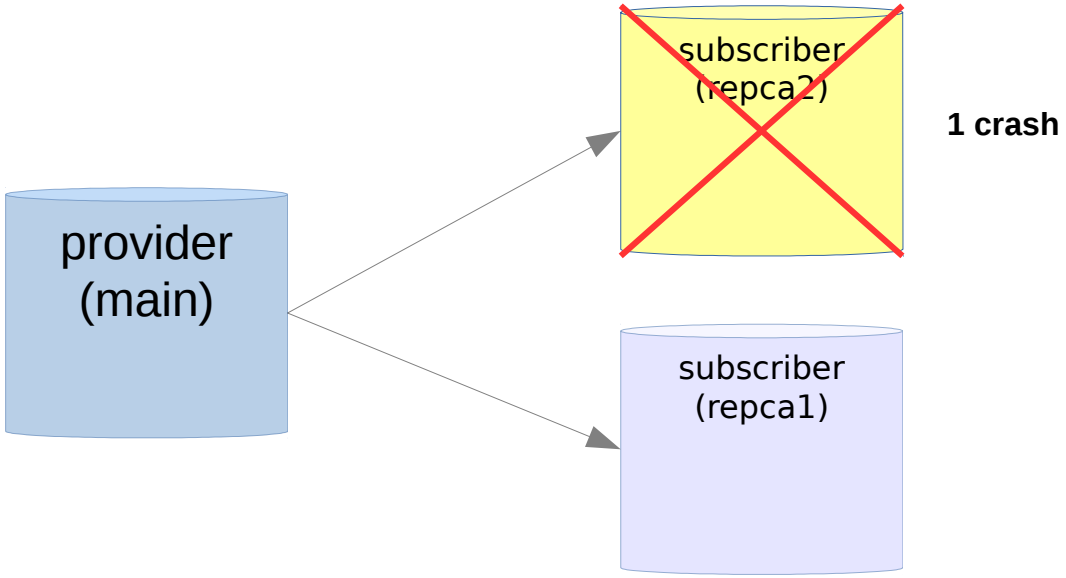
* https://github.com/avito-tech/dba-docs/blob/master/PGCon2018_Ottawa/Recovery_Use_Cases_for_Logical_Replication_in_PostgreSQL10.txt

(1) Reinitializing subscriber from another subscriber

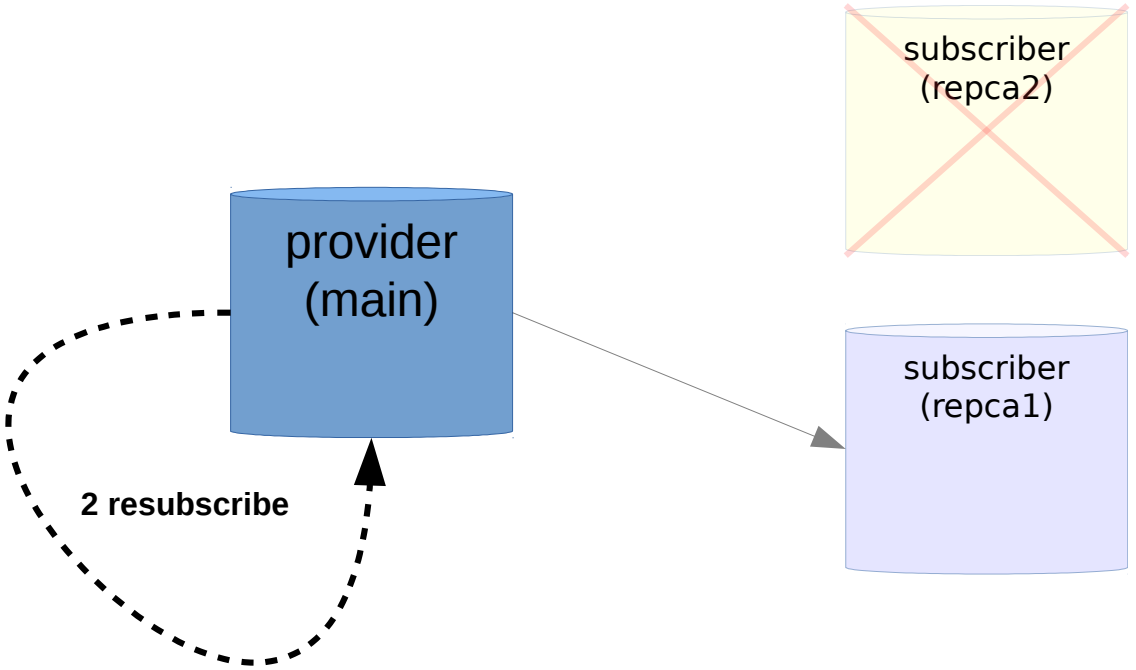


* There are two replicas for redundancy (Crashes can happen so if you want to have no downtime, reserve every node)

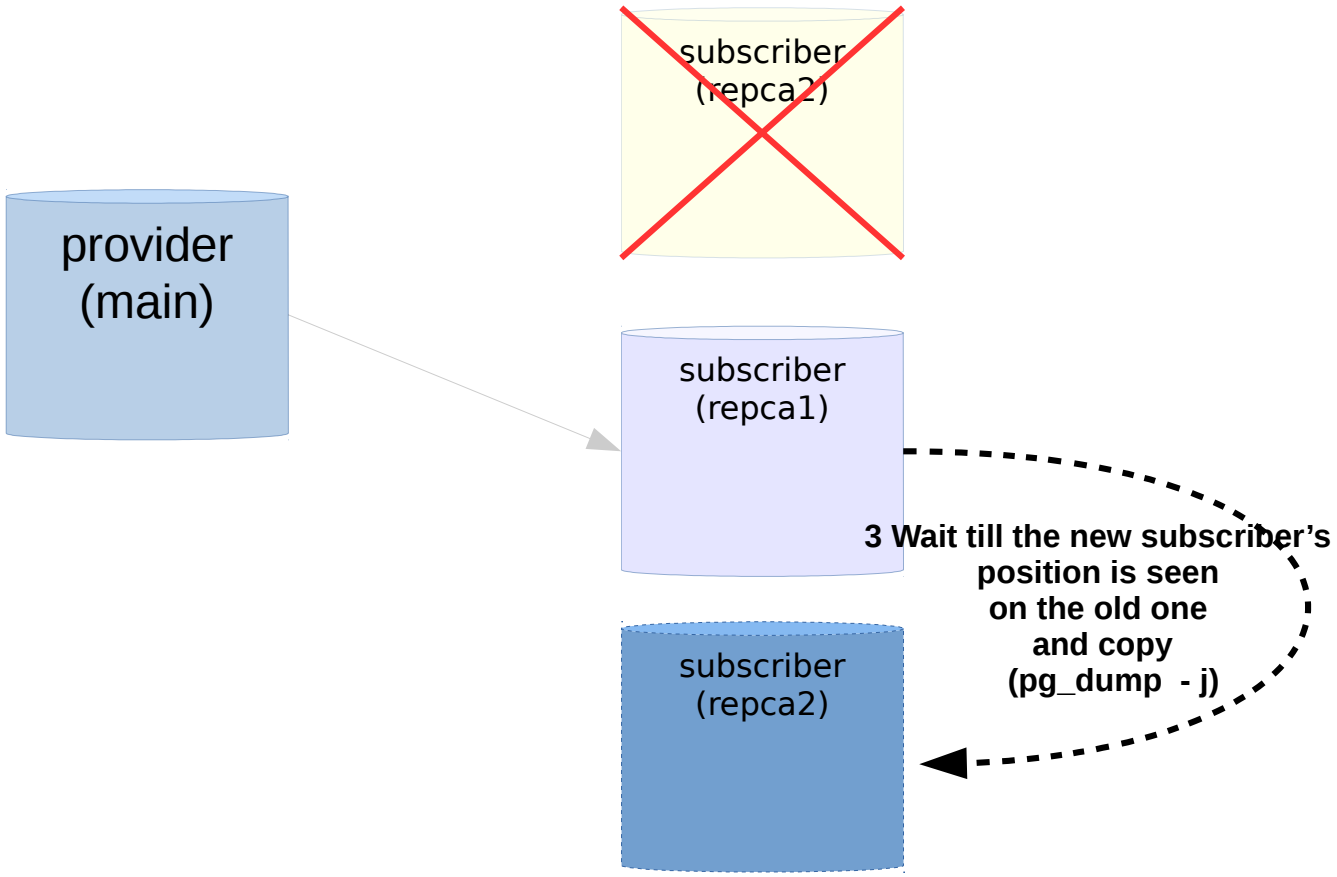
(1) Reinitializing subscriber from another subscriber



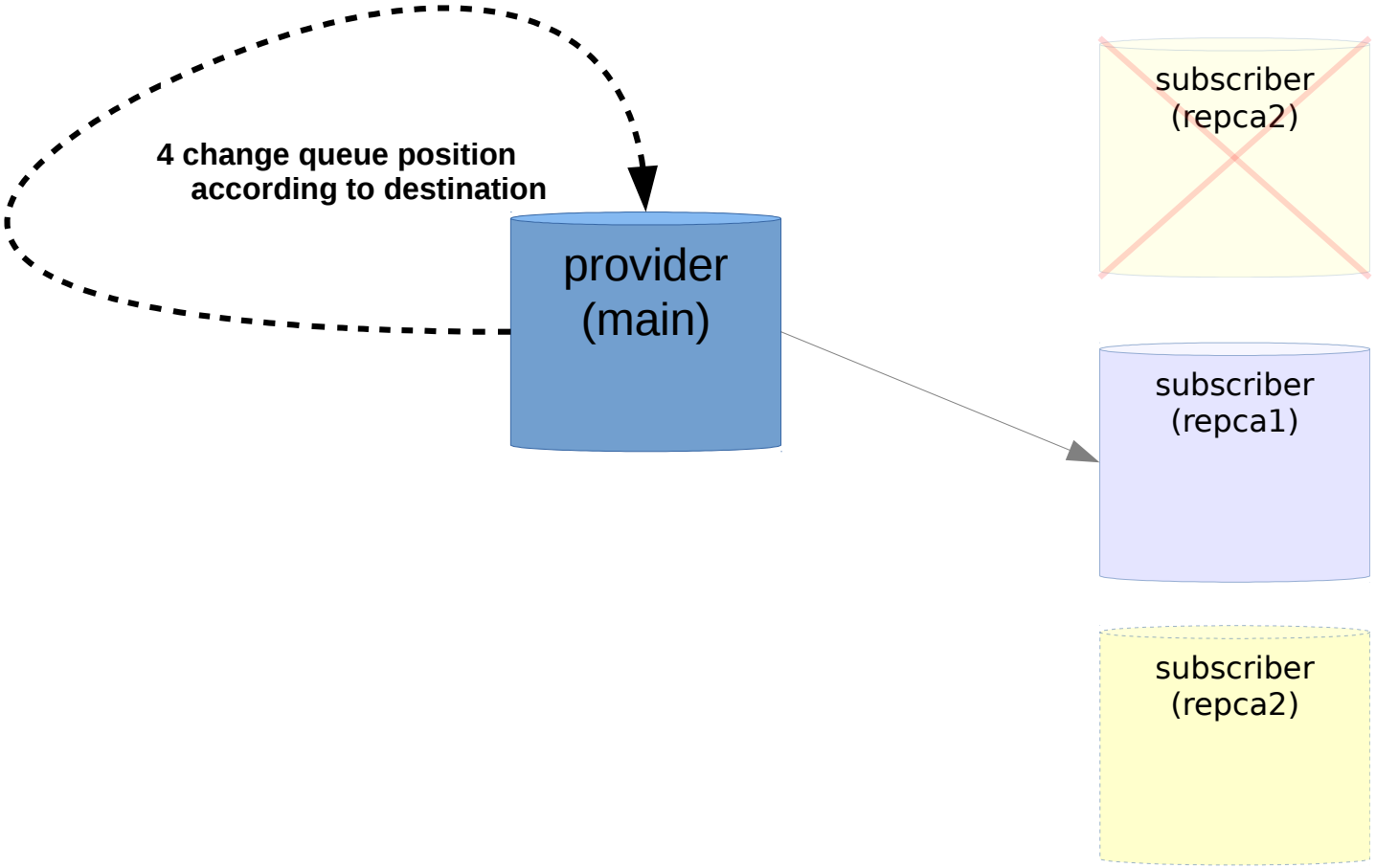
(1) Reinitializing subscriber from another subscriber



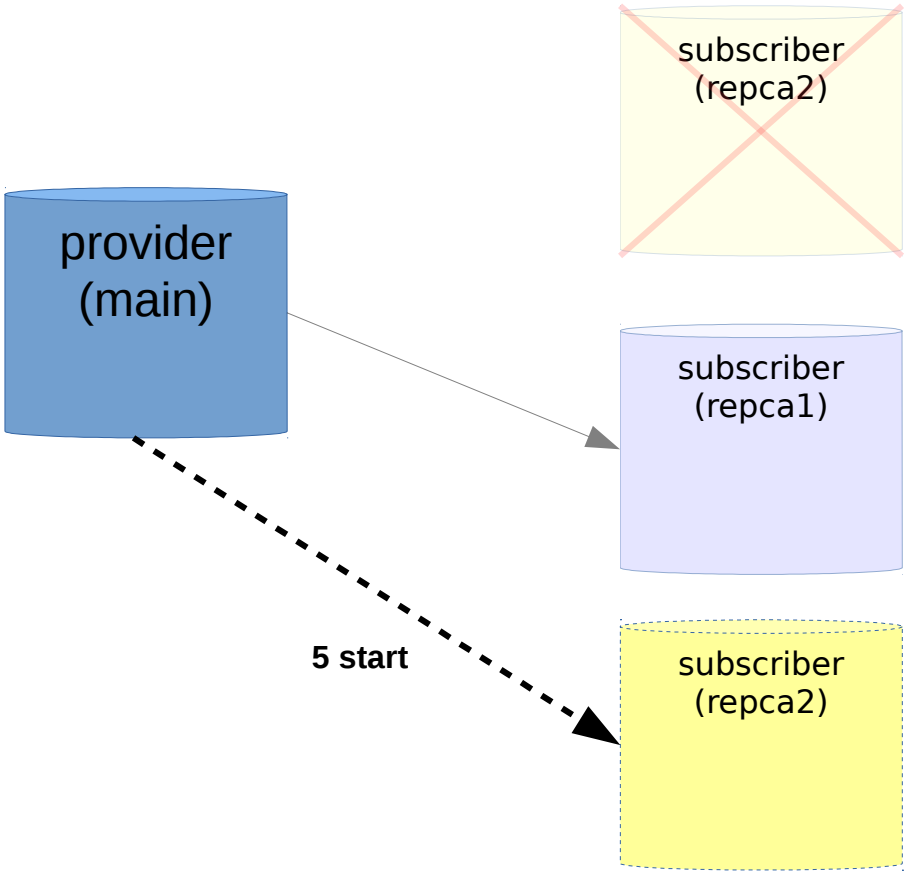
(1) Reinitializing subscriber from another subscriber



(1) Reinitializing subscriber from another subscriber



(1) Reinitializing subscriber from another subscriber



(1) Reinitializing subscriber from another subscriber

1. Creating a logical slot for a new subscriber:

```
pg_recvlogical -p 5432 -U postgres -d src --create-slot -S repca2 -P pgoutput

psql -p 5432 -U postgres -X -d src -c 'select slot_name, active  from pg_replication_slots'
```

```
 slot_name | active
-----------+--------
 standby   | t
 repca1    | t
 repca2    | f
(3 rows)
```

(1) Reinitializing subscriber from another subscriber

1. Creating a logical slot for a new subscriber:

```
pg_recvlogical -p 5432 -U postgres -d src --create-slot -S repca2 -P pgoutput
psql -p 5432 -U postgres -X -d src -c 'select slot_name, active  from pg_replication_slots'
```

```
 slot_name | active
-----------+--------
 standby   | t
 repca1    | t
 repca2    | f
(3 rows)
```

2. Disabling active subscription:

```
psql -p 5434 -U postgres -X -d dst -c 'alter subscription repca1 disable'
```

(1) Reinitializing subscriber from another subscriber

1. Creating a logical slot for a new subscriber:

```
pg_recvlogical -p 5432 -U postgres -d src --create-slot -S repca2 -P pgoutput
psql -p 5432 -U postgres -X -d src -c 'select slot_name, active  from pg_replication_slots'
```

```
 slot_name | active
-----------+--------
 standby   | t
 repca1    | t
 repca2    | f
(3 rows)
```

2. Disabling active subscription:

```
psql -p 5434 -U postgres -X -d dst -c 'alter subscription repca1 disable'
```

3. Logging current LSN:

```
psql -p 5434 -U postgres -X -d dst -c 'select remote_lsn from pg_replication_origin_status'
```

```
 remote_lsn
------------
 0/3039E78
```

(1) Reinitializing subscriber from another subscriber

4. `pg_dump -p 5434 -U postgres -Fc --serializable-deferrable **--no-subscriptions** -d dst \`
   `| pg_restore -p 5435 -U postgres -C -d postgres`

(1) Reinitializing subscriber from another subscriber

subscriber (repca1)

4. `pg_dump -p 5434 -U postgres -Fc --serializable-deferrable --no-subscriptions -d dst \`
   `| pg_restore -p 5435 -U postgres -C -d postgres`

subscriber (repca2)

5. At the same time:
   `psql -p 5434 -U postgres -X -d dst -c '`**`alter subscription repca1 enable`**`'`

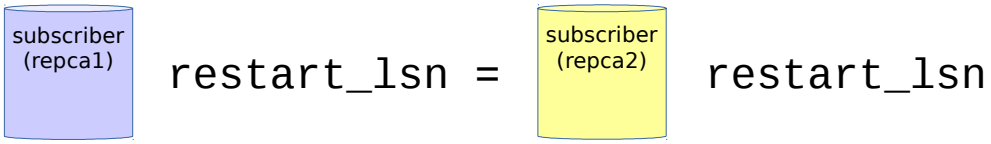
subscriber (repca1)

`psql -p 5432 -U postgres -X -d src`
`-c '`**`select slot_name, restart_lsn, confirmed_flush_lsn, active from pg_replication_slots`**`'`

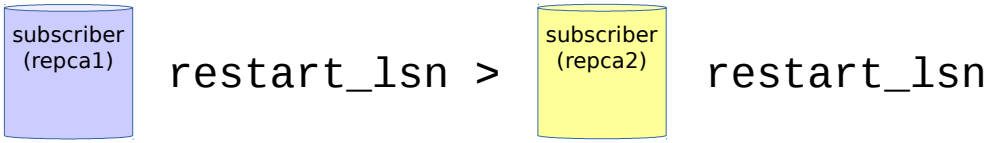
provider (main)

```
 slot_name | restart_lsn | confirmed_flush_lsn | active
-----------+-------------+---------------------+--------
 standby   | 0/303A6A0   |                     | t
 repca1    | 0/303A588   | 0/303A6A0           | t
 repca2    | 0/3039F90   | 0/3039FC8           | f
(3 rows)
```

(1) Reinitializing subscriber from another subscriber



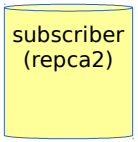
slot_name	plugin	slot_type	datoid	database	temporary	active	active_pid	xmin	catalog_xmin	restart_lsn	confirmed_flush_lsn
repca2	pgoutput	logical	16384	src	f	f			561	0/30409D0	0/3040A08
repca1	pgoutput	logical	16384	src	f	t	16117		561	0/30409D0	0/3040A08
standby		physical			f	t	15871			0/3040A08	



slot_name	plugin	slot_type	datoid	database	temporary	active	active_pid	xmin	catalog_xmin	restart_lsn	confirmed_flush_lsn
repca2	pgoutput	logical	16384	src	f	f			561	0/30409D0	0/3040A08
repca1	pgoutput	logical	16384	src	f	t	16727		562	0/30410A8	0/30411C0
standby		physical			f	t	15871			0/30411C0	

(1) Reinitializing subscriber from another subscriber

6. Creating subscription




```
psql -p 5435 -U postgres -X -d dst -f- <<'EOF'
create subscription repca2 connection 'port=5432 dbname=src'
  publication pub with (enabled = false, create_slot = false, copy_data = false);

select oid, * from pg_subscription;
EOF
```

```
CREATE SUBSCRIPTION
  oid  | subdbid | subname | subowner | subenabled |     subconninfo      | subslotname | subsynccommit | subpublications
-------+---------+---------+----------+------------+----------------------+-------------+---------------+-----------------
 16557 |   16384 | repca2  |       10 | f          | port=5432 dbname=src | repca2      | off           | {pub}
(1 row)
```

(1) Reinitializing subscriber from another subscriber

6. Creating subscription

subscriber (repca2)

```
psql -p 5435 -U postgres -X -d dst -f- <<'EOF'
create subscription repca2 connection 'port=5432 dbname=src'
  publication pub with (enabled = false, create_slot = false, copy_data = false);

select oid, * from pg_subscription;
EOF
```

```
CREATE SUBSCRIPTION
  oid  | subdbid | subname | subowner | subenabled |     subconninfo      | subslotname | subsynccommit | subpublications
-------+---------+---------+----------+------------+----------------------+-------------+---------------+-----------------
 16557 |   16384 | repca2  |       10 | f          | port=5432 dbname=src | repca2      | off           | {pub}
(1 row)
```

7. Moving LSN

subscriber (repca2)

```
psql -p 5435 -U postgres -X -d dst -c  "select pg_replication_origin_advance('pg_16557', '0/3039E78')"
```

```
# psql -p 5434 -U postgres -X -d dst -c 'select remote_lsn from pg_replication_origin_status'
```

subscriber (repca1)

```
 remote_lsn
------------
 0/3039E78
```

(1) Reinitializing subscriber from another subscriber

8. Checking the subscription state:


```
psql -p 5435 -U postgres -X -d dst -c 'select * from pg_replication_origin_status'
```

```
 local_id | external_id | remote_lsn | local_lsn
----------+-------------+------------+-----------
        1 | pg_16557    | 0/3039E78  | 0/0
```

(1) Reinitializing subscriber from another subscriber

8. Checking the subscription state:

subscriber (repca2)

```
psql -p 5435 -U postgres -X -d dst -c 'select * from pg_replication_origin_status'
```

```
 local_id | external_id | remote_lsn | local_lsn
----------+-------------+------------+-----------
        1 | pg_16557    | 0/3039E78  | 0/0
```

9. Enabling subscription:

subscriber (repca2)

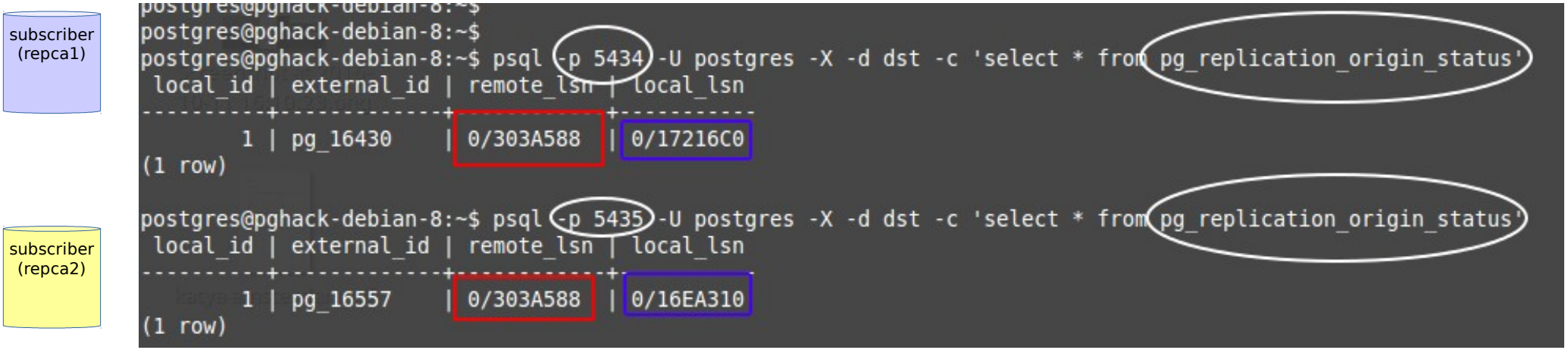
```
psql -p 5435 -U postgres -X -d dst -c 'alter subscription repca2 enable'
```

provider (main)

```
psql -p 5432 -U postgres -X -d src
            -c 'select slot_name, restart_lsn, confirmed_flush_lsn from pg_replication_slots'
```

```
 slot_name | restart_lsn | confirmed_flush_lsn
-----------+-------------+---------------------
 standby   | 0/303A6A0   |
 repca1    | 0/303A588   | 0/303A6A0
 repca2    | 0/303A588   | 0/303A6A0
(3 rows)
```

(1) Reinitializing subscriber from another subscriber



Recovery use cases

(1) Reinitializing subscriber from another subscriber

(2) UNDO recovery on the destination side

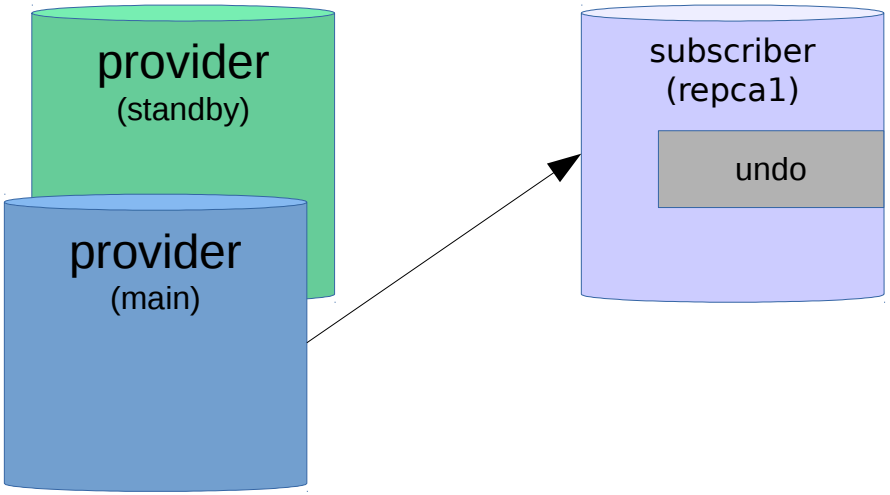
(3) REDO - reposition source (subscriber's crash)

(4) REDO 2 - on provider's side (provider's crash and switching to the provider's standby, subscriber is falling behind)

* https://github.com/avito-tech/dba-docs/blob/master/PGCon2018_Ottawa/Recovery_Use_Cases_for_Logical_Replication_in_PostgreSQL10.txt

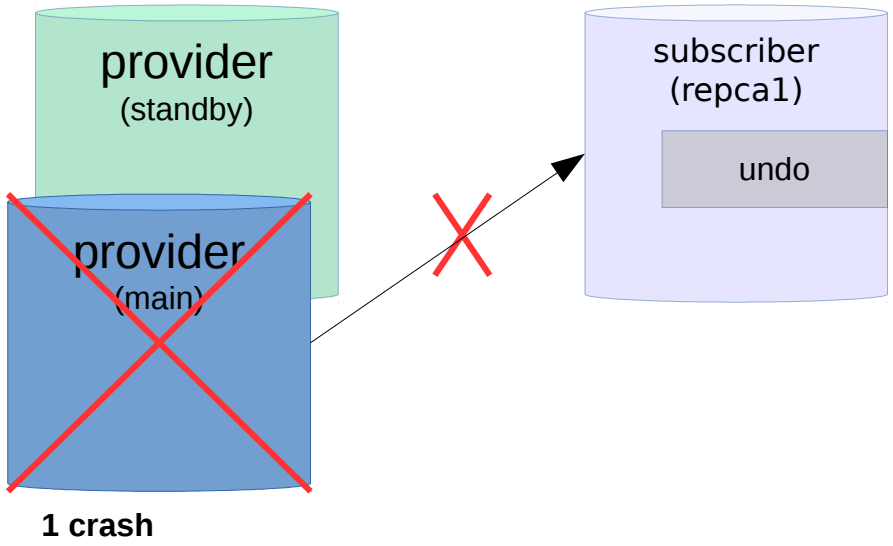
(2) UNDO

recovery on the destination side



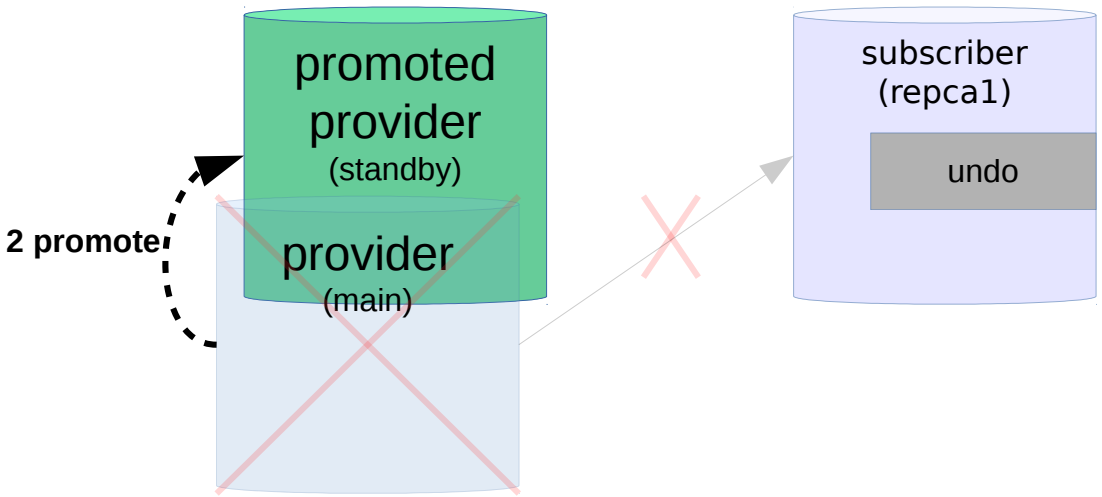
(2) UNDO

recovery on the destination side



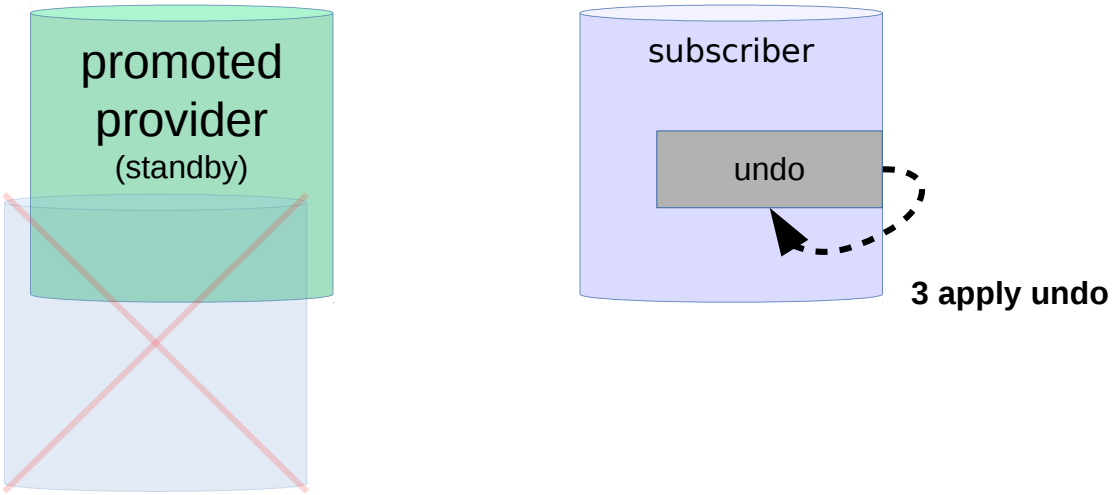
(2) UNDO

recovery on the destination side



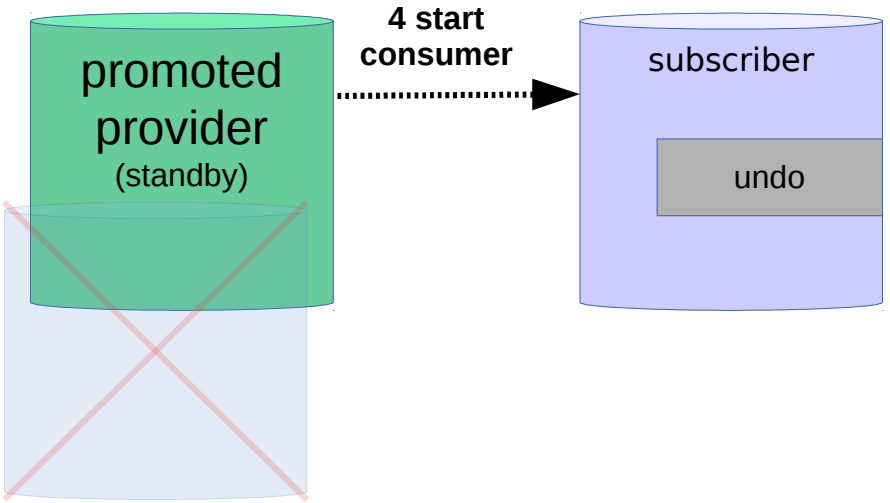
(2) UNDO

recovery on the destination side



(2) UNDO

recovery on the destination side



(2) UNDO

recovery on the destination side

```
create table undo_log (
     id bigserial primary key,
     txtime timestamptz default now(),
     lsn pg_lsn default pg_replication_origin_session_progress(false),
     dst_schema name, dst_table name,
     undo_cmd char, cmd_data hstore, cmd_pk hstore -- identity
);
```

```
select
     id, LSN, dst_schema, dst_table, undo_cmd, cmd_data, cmd_pk
from
     undo_log order by id
```

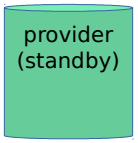
id	lsn	dst_schema	dst_table	undo_cmd	cmd_data	cmd_pk
1	0/30377D8	public	cats	D		"cat_id"=>"3"
2	0/30377D8	public	items	D		"item_id"=>"4"
6	0/3037E80	public	cats	D		"cat_id"=>"36"
7	0/3037E80	public	cats	D		"cat_id"=>"37"
8	0/3037E80	public	cats	D		"cat_id"=>"38"
9	0/3038590	public	cats	D		"cat_id"=>"39"
10	0/3038590	public	cats	D		"cat_id"=>"40"
11	0/3038590	public	cats	D		"cat_id"=>"41"

* https://github.com/avito-tech/dba-docs/blob/master/PGCon2018_Ottawa/Recovery_Use_Cases_for_Logical_Replication_in_PostgreSQL10.txt

(2) UNDO

recovery on the destination side

1. Write down the WAL replay LSN before promotion



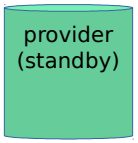

```
psql -p 5433 -U postgres -X -d src -c 'select pg_last_wal_replay_lsn()'
```

```
 pg_last_wal_replay_lsn
------------------------
 0/3037F60
(1 row)
```

(2) UNDO

recovery on the destination side

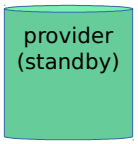
1. Write down the WAL replay LSN before promotion




```
psql -p 5433 -U postgres -X -d src -c 'select pg_last_wal_replay_lsn()'
```

```
 pg_last_wal_replay_lsn
------------------------
 0/3037F60
(1 row)
```

2. Logical Replication Slot isn't replicated to the standby, that's why you shouldn't turn on traffic immediately after promotion of standby




```
pg_ctl -D /var/lib/postgresql/10/standby  promote

pg_recvlogical -p 5433 -U postgres -d src --create-slot -S repca1 -P pgoutput
```

(2) UNDO

recovery on the destination side

1. Write down the WAL replay LSN before promotion

provider (standby)

```
psql -p 5433 -U postgres -X -d src -c 'select pg_last_wal_replay_lsn()'
```

```
 pg_last_wal_replay_lsn
------------------------
 0/3037F60
(1 row)
```

2. Logical Replication Slot isn't replicated to the standby, that's why you shouldn't turn on traffic immediately after promotion of standby

provider (standby)

```
pg_ctl -D /var/lib/postgresql/10/standby  promote

pg_recvlogical -p 5433 -U postgres -d src --create-slot -S repca1 -P pgoutput
```

3. There are some changes for Undo

subscriber (repca1)

```
select id, LSN, dst_schema, dst_table, undo_cmd, cmd_data, cmd_pk from undo_log
```

```
 id |    lsn    | dst_schema | dst_table | undo_cmd | cmd_data |      cmd_pk
----+-----------+------------+-----------+----------+----------+----------------
  1 | 0/30377D8 | public     | cats      | D        |          | "cat_id"=>"3"
  2 | 0/30377D8 | public     | items     | D        |          | "item_id"=>"4"
  3 | 0/3037E80 | public     | cats      | D        |          | "cat_id"=>"4"
  4 | 0/3037E80 | public     | cats      | D        |          | "cat_id"=>"5"
  5 | 0/3037E80 | public     | cats      | D        |          | "cat_id"=>"6"
(5 rows)
```

(2) UNDO

recovery on the destination side

4. Current subscriber's LSN


```
psql  -p 5434 -U postgres -X -d dst -c 'select * from pg_replication_origin_status'
```

```
 local_id | external_id | remote_lsn | local_lsn
----------+-------------+------------+-----------
        1 | pg_16430    | 0/3038480  | 0/16FA1D0
```

(2) UNDO

recovery on the destination side

4. Current subscriber's LSN

subscriber (repca1)

```
psql  -p 5434 -U postgres -X -d dst -c 'select * from pg_replication_origin_status'
```

```
 local_id | external_id | remote_lsn | local_lsn
----------+-------------+------------+-----------
        1 | pg_16430    | 0/3038480  | 0/16FA1D0
```

5. Applying Undo

subscriber (repca1)

```
postgres@pghack-debian-8:~$ psql  -p 5434 -U postgres -X -d dst -c "select run_undo('0/3037F60')"
NOTICE:  last applied LSN: 0/3038480
NOTICE:  undo events with LSN >= 0/3037E80
NOTICE:  undo (5 | 0/3037E80): DELETE FROM ONLY public.cats d  WHERE d.cat_id = '6'
NOTICE:  undo (4 | 0/3037E80): DELETE FROM ONLY public.cats d  WHERE d.cat_id = '5'
NOTICE:  undo (3 | 0/3037E80): DELETE FROM ONLY public.cats d  WHERE d.cat_id = '4'
NOTICE:  set current replayed LSN to 0/3037E80
 run_undo
----------
        3
(1 row)
```

(2) UNDO

recovery on the destination side

4. Current subscriber's LSN

subscriber (repca1)

```
psql  -p 5434 -U postgres -X -d dst -c 'select * from pg_replication_origin_status'
```

```
 local_id | external_id | remote_lsn | local_lsn
----------+-------------+------------+-----------
        1 | pg_16430    | 0/3038480  | 0/16FA1D0
```

5. Applying Undo

subscriber (repca1)

```
postgres@pghack-debian-8:~$ psql  -p 5434 -U postgres -X -d dst -c "select run_undo('0/3037F60')"
NOTICE:  last applied LSN: 0/3038480
NOTICE:  undo events with LSN >= 0/3037E80
NOTICE:  undo (5 | 0/3037E80): DELETE FROM ONLY public.cats d  WHERE d.cat_id = '6'
NOTICE:  undo (4 | 0/3037E80): DELETE FROM ONLY public.cats d  WHERE d.cat_id = '5'
NOTICE:  undo (3 | 0/3037E80): DELETE FROM ONLY public.cats d  WHERE d.cat_id = '4'
NOTICE:  set current replayed LSN to 0/3037E80
 run_undo
----------
        3
(1 row)
```

6. Enabling logical replication

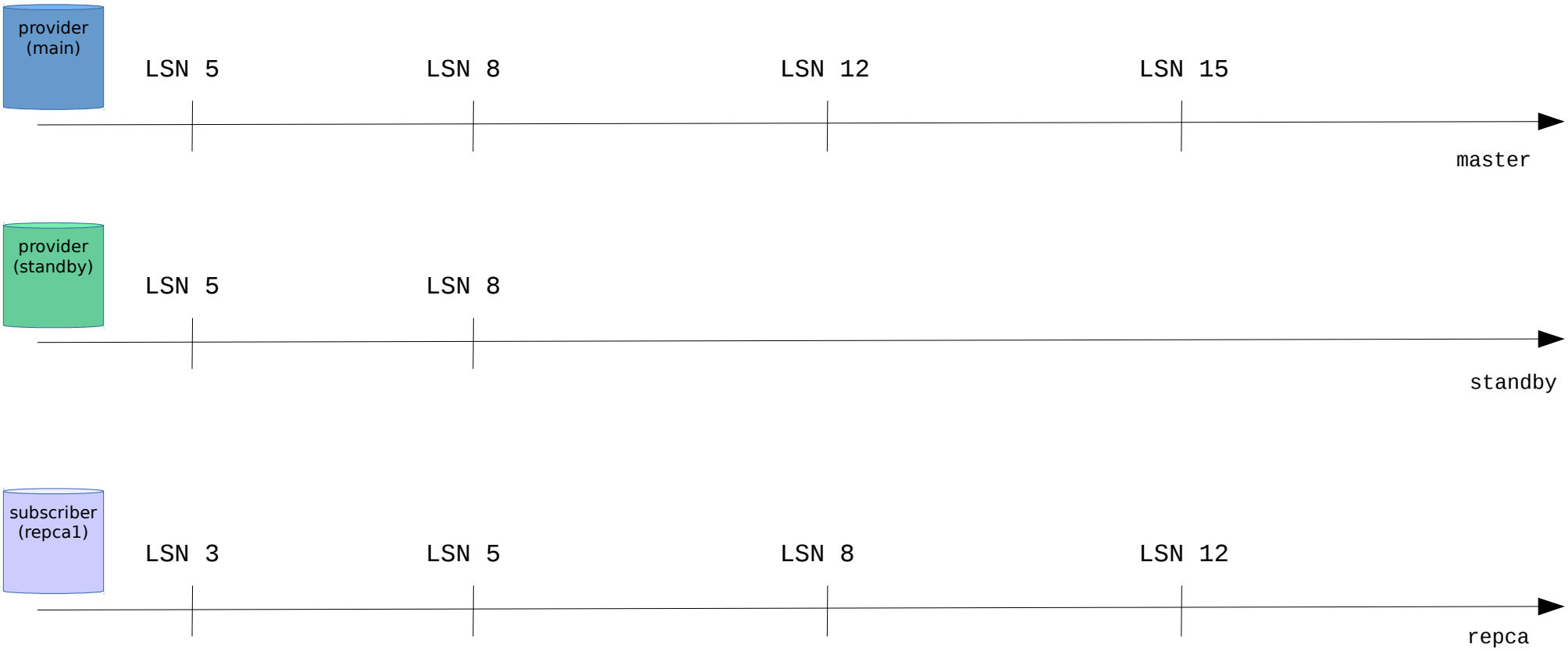
subscriber (repca1)

```
psql -p 5434 -U postgres -X -d dst -c 'alter subscription repca1 enable'
```

(2) UNDO

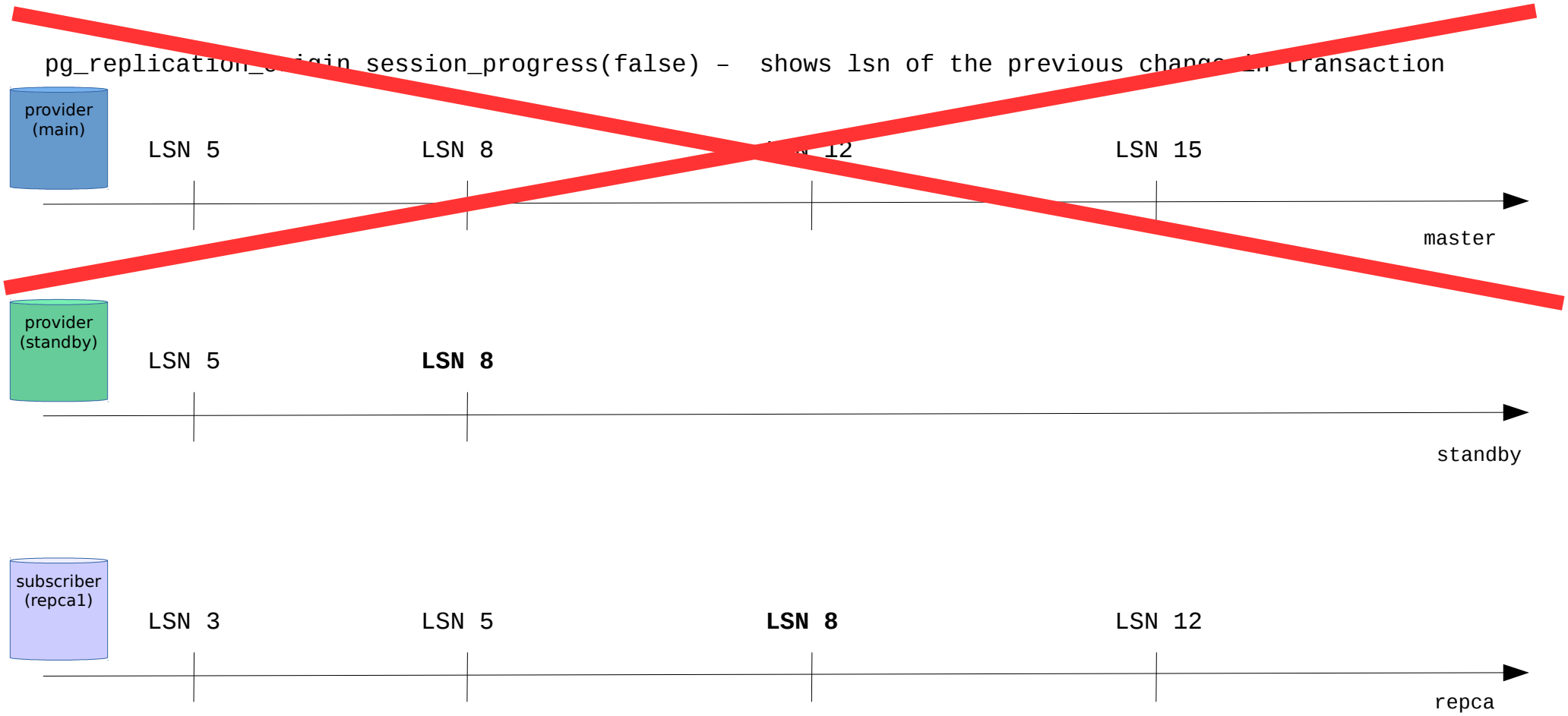
recovery on the destination side

pg_replication_origin_session_progress(false) – shows lsn of the previous change in transaction



(2) UNDO

recovery on the destination side



(2) UNDO

recovery on the destination side

Few sources, one subscription and Undo

- In the trigger get the subscriber's name and write it to the undo log with opposite actions

```
select subname
from
    pg_stat_subscription p
where
    p.pid = pg_backend_pid()
```

(2) UNDO

recovery on the destination side

Few sources, one subscription and Undo

- In the trigger get the subscriber's name and write it to the undo log with opposite actions

```
select subname
from
    pg_stat_subscription p
where
    p.pid = pg_backend_pid()
```

- On the publication side there is no possibility to find out who consumes the slot. As we don't know links between subscriber and publisher – we need to make an external list with logical consumers (for londiste we do the same thing), to apply undo if the source is crashed

(2) UNDO

recovery on the destination side

Few sources, one subscription and Undo

- In the trigger get the subscriber's name and write it to the undo log with opposite actions

```
select subname
from
    pg_stat_subscription p
where
    p.pid = pg_backend_pid()
```

- On the publication side there is no possibility to find out who consumes the slot. As we don't know links between subscriber and publisher – we need to make an external list with logical consumers (for londiste we do the same thing), to apply undo if the source is crashed

- So the consumer name has to be in special format. This will be useful to find out the link between publication and subscription

Recovery use cases

(1) Reinitializing subscriber from another subscriber

(2) UNDO recovery on the destination side

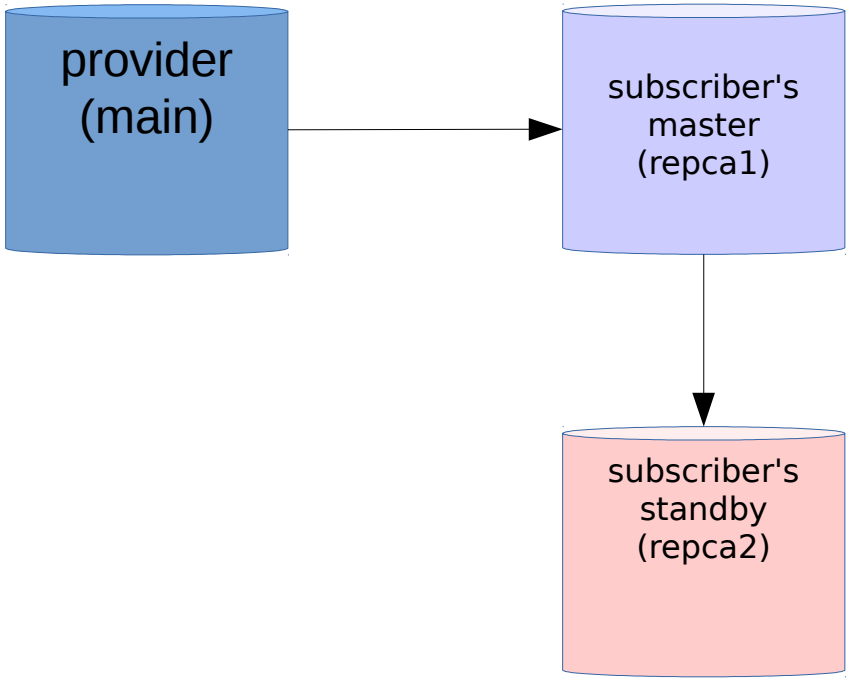
(3) REDO - reposition source (subscriber's crash)

(4) REDO 2 - on provider's side (provider's crash and switching to the provider's standby, subscriber is falling behind)

* https://github.com/avito-tech/dba-docs/blob/master/PGCon2018_Ottawa/Recovery_Use_Cases_for_Logical_Replication_in_PostgreSQL10.txt

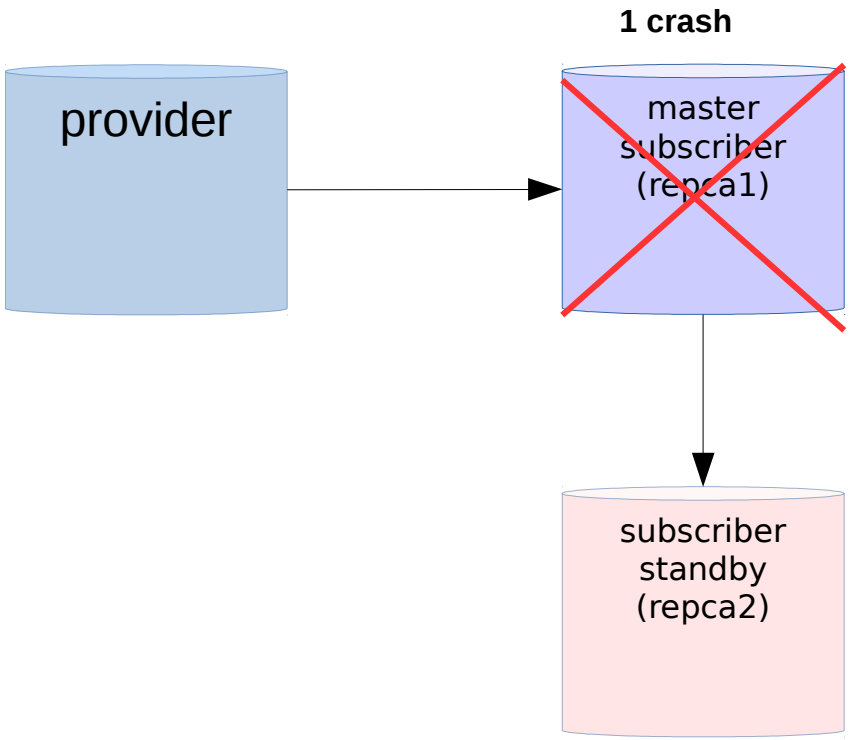
(3) REDO

reposition source (subscriber's crash)



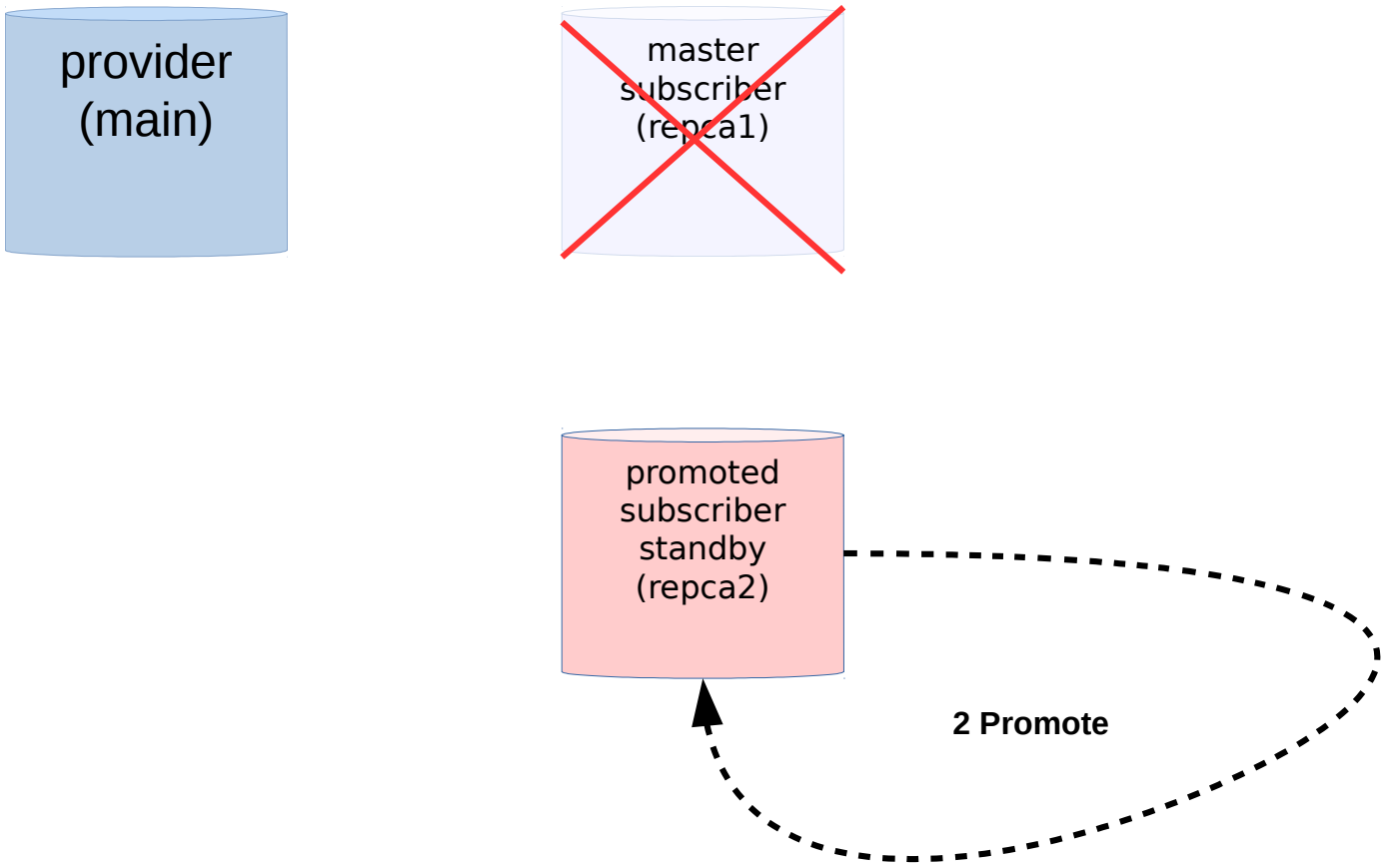
(3) REDO

reposition source (subscriber's crash)



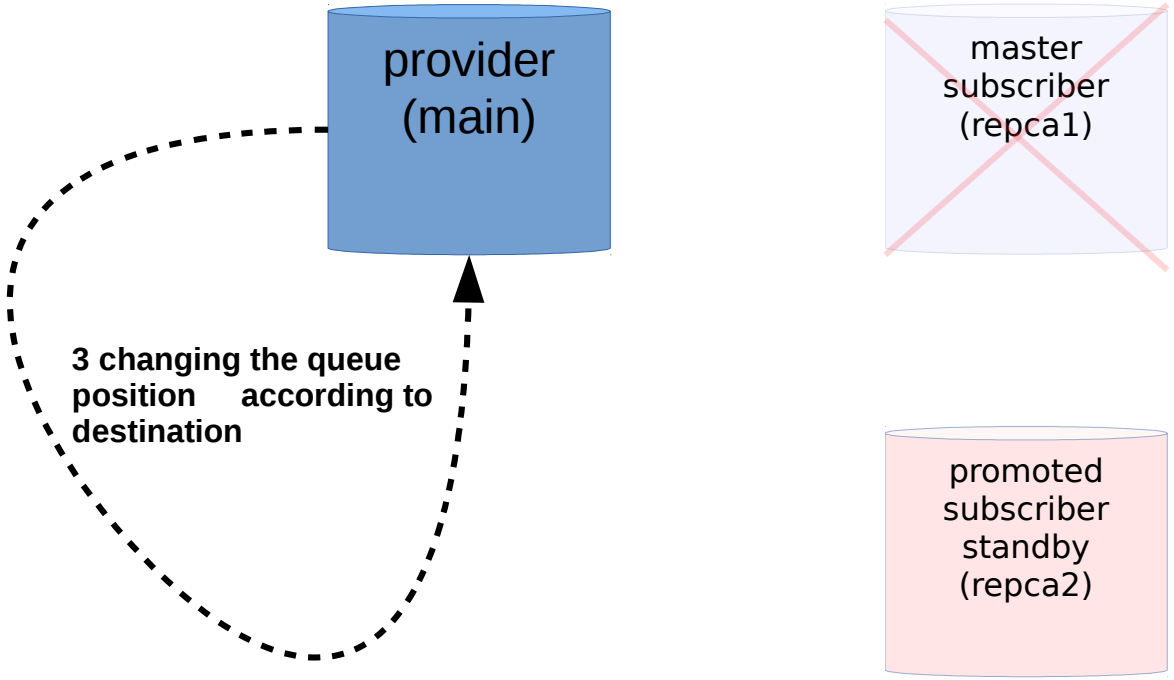
(3) REDO

reposition source (subscriber's crash)



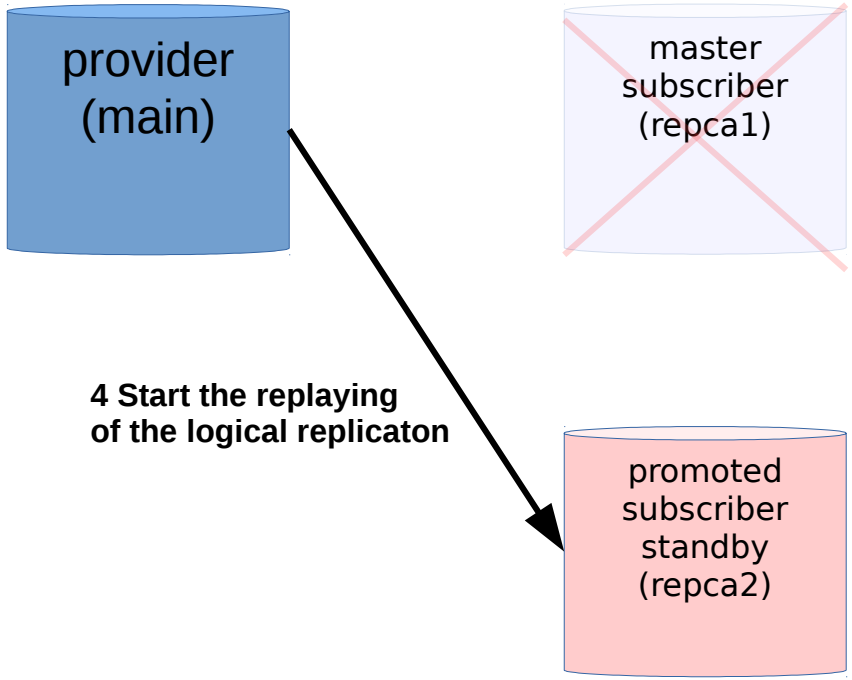
(3) REDO

reposition source (subscriber's crash)



(3) REDO

reposition source (subscriber's crash)



(3) REDO

reposition source (subscriber's crash)

The command for moving a replication slot. We create the slot manually for the promoted consumer's standby to prevent replicated queue (WAL) rotation on provider's side:

```
psql -p 5433 -U postgres -X -d src
-c "
select * from  pg_logical_slot_get_binary_changes(
    'repca2'::name,
    '0/38AFCC0'::pg_lsn,
    null::int,
    variadic array['proto_version', '1', 'publication_names',
'pub']
)"
```

(3) REDO

reposition source (subscriber's crash)

1. Creating a logical slot to prevent WAL's rotation on provider's side(these WAL files can be needed for promoted subscriber's standby)


```
pg_recvlogical -p 5432 -U postgres -d src --create-slot -S repca2 -P pgoutput
psql -p 5432  -U postgres -X -d src
     -c 'select slot_name, active, confirmed_flush_lsn from pg_replication_slots'
```

```
 slot_name | active | confirmed_flush_lsn
-----------+--------+---------------------
 repca1    | t      | 0/16631F8
 repca2    | f      | 0/16631F8
(2 rows)
```

(3) REDO

reposition source (subscriber's crash)

1. Creating a logical slot to prevent wal's rotation on provider's side(these WAL files can be nedeed for promoted subscriber's standby)

provider (main)

```
pg_recvlogical -p 5432 -U postgres -d src --create-slot -S repca2 -P pgoutput
psql -p 5432  -U postgres -X -d src
     -c 'select slot_name, active, confirmed_flush_lsn from pg_replication_slots'
```

```
 slot_name | active | confirmed_flush_lsn
-----------+--------+---------------------
 repca1    | t      | 0/16631F8
 repca2    | f      | 0/16631F8
(2 rows)
```

2. Adding new changes for our logical consumer:

provider (main)

```
psql -p 5432 -U postgres -1 -X -d src -f- <<'EOF'
insert into cats (cat_name) values ('category 3');
EOF
```

(3) REDO

reposition source (subscriber's crash)

1. Creating a logical slot to prevent wal's rotation on provider's side(these WAL files can be nedeed for promoted subscriber's standby)

provider (main)

```
pg_recvlogical -p 5432 -U postgres -d src --create-slot -S repca2 -P pgoutput
psql -p 5432  -U postgres -X -d src
     -c 'select slot_name, active, confirmed_flush_lsn from pg_replication_slots'
```

```
 slot_name | active | confirmed_flush_lsn
-----------+--------+---------------------
 repca1    | t      | 0/16631F8
 repca2    | f      | 0/16631F8
(2 rows)
```

2. Adding new changes for our logical consumer:

provider (main)

```
psql -p 5432 -U postgres -1 -X -d src -f- <<'EOF'
insert into cats (cat_name) values ('category 3');
EOF
```

3. Replication slot for our subscriber's standby is in the past

provider (main)

```
psql -p 5432 -U postgres -X -d src
          -c 'select slot_name, active, confirmed_flush_lsn from pg_replication_slots'
```

```
 slot_name | active | confirmed_flush_lsn
-----------+--------+---------------------
 repca1    | t      | 0/1663410
 repca2    | f      | 0/16631F8
(2 rows)
```

(3) REDO

reposition source (subscriber's crash)

4. Checking pg_replication_origin status on subscriber's side and subscriber's standby side:


```
psql -p 5434 -U postgres -X -d dst -c 'select local_lsn from pg_replication_origin_status'
```

```
 local_lsn
-----------
 0/30343B0
```


```
psql -p 5435 -U postgres -X -d dst -c 'select local_lsn from pg_replication_origin_status'
```

```
 local_lsn
-----------
 0/30343B0
```

(3) REDO

reposition source (subscriber's crash)

4. Checking pg_replication_origin status on subscriber's side and subscriber's standby side:

subscriber (repca1)

```
psql -p 5434 -U postgres -X -d dst -c 'select local_lsn from pg_replication_origin_status'
```

```
 local_lsn
-----------
 0/30343B0
```

subscriber standby (repca2)

```
psql -p 5435 -U postgres -X -d dst -c 'select local_lsn from pg_replication_origin_status'
```

```
 local_lsn
-----------
 0/30343B0
```

5. “Moving” replication slot with the help of SQL protocol

provider (main)

```
psql -p 5432 -U postgres -X -d src -c  "select * from
pg_logical_slot_get_binary_changes('repca2'::name, '0/1663410'::pg_lsn, null::int,
                  variadic array['proto_version', '1', 'publication_names', 'pub'])"
```

```
postgres@pghack-debian-8:~$ psql -p 5432 -U postgres -X -d src -c "select * from pg_logical_slot_get_binary_changes('repca2'::name, '0/1663410':
:pg_lsn, null::int, variadic array['proto_version', '1', 'publication_names', 'pub'])"
    lsn    | xid |                                                       data
-----------+-----+------------------------------------------------------------------------------------------------------------------
 0/16631F8 | 559 | \x4200000000016632c800020eea519f26e60000022f
 0/16631F8 | 559 | \x520000400370756 26c6963006361747300640002016361745f69640000000017ffffffff006361745f6e616d650000000019ffffffff
 0/16631F8 | 559 | \x49000040034e0002740000000133740000000a63617465676f72792033
 0/16632F8 | 559 | \x4300000000000016632c800000000016632f800020eea519f26e6
(4 rows)
```

(3) REDO
reposition source (subscriber's crash)

4. Checking pg_replication_origin status on subscriber's side and subscriber's standby side:

subscriber (repca1)

```
psql -p 5434 -U postgres -X -d dst -c 'select local_lsn from pg_replication_origin_status'
 local_lsn
-----------
 0/30343B0
```

subscriber standby (repca2)

```
psql -p 5435 -U postgres -X -d dst -c 'select local_lsn from pg_replication_origin_status'
 local_lsn
-----------
 0/30343B0
```

5. “Moving” replication slot with the help of SQL protocol

provider (main)

```
psql -p 5432 -U postgres -X -d src -c  "select * from
pg_logical_slot_get_binary_changes('repca2'::name, '0/1663410'::pg_lsn, null::int,
                  variadic array['proto_version', '1', 'publication_names', 'pub'])"
```

```
postgres@pghack-debian-8:~$ psql -p 5432 -U postgres -X -d src -c "select * from pg_logical_slot_get_binary_changes('repca2'::name, '0/1663410'::pg_lsn, null::int, variadic array['proto_version', '1', 'publication_names', 'pub'])"
    lsn    | xid |                                          data
-----------+-----+------------------------------------------------------------------------------------------------
 0/16631F8 | 559 | \x42000000000016632c800020eea519f26e60000022f
 0/16631F8 | 559 | \x520000400370756 26c696300636174730064000201636174 5f69640000000017ffffffff006361745f6e616d650000000019ffffffff
 0/16631F8 | 559 | \x49000040034e00027400000001337400000000a63617465676f72792033
 0/16632F8 | 559 | \x43000000000000016632c800000000016632f800020eea519f26e6
(4 rows)
```

6. LSN for both replication slots are equal

provider (main)

```
psql -p 5432 -U postgres -X -d src
          -c 'select slot_name, active, confirmed_flush_lsn from pg_replication_slots'
 slot_name | active | confirmed_flush_lsn
-----------+--------+---------------------
 repca1    | t      | 0/1663410
 repca2    | f      | 0/1663410
(2 rows)
```

(3) REDO
reposition source (subscriber's crash)

7. “Emulating delay of subscriber's standby replication”

1) sudo ***mcedit*** */var/lib/postgresql/10/repca1/****pg_hba.conf***

2) ***reload config***

*3) psql -p 5434 -U postgres -X -d dst -c "****select pg_terminate_backend(active_pid) from pg_replication_slots where slot_name = 'repca2****'"*

(3) REDO
reposition source (subscriber's crash)

7. “Emulating delay of subscriber's standby replication”

1) sudo mcedit /var/lib/postgresql/10/repca1/pg_hba.conf
2) reload config
3) psql -p 5434 -U postgres -X -d dst -c "select pg_terminate_backend(active_pid) from pg_replication_slots where slot_name = 'repca2'"

8. Adding one more record in the replicated table


```
postgres@pghack-debian-8:~$ psql -p 5432  -U postgres -1 -X -d src -f- <<'EOF'
> insert into cats (cat_name) values ('category 4');
> EOF
INSERT 0 1
```

(3) REDO
reposition source (subscriber's crash)

7. “Emulating delay of subscriber's standby replication”

subscriber (repca1)

1) sudo mcedit /var/lib/postgresql/10/repca1/pg_hba.conf
2) reload config
3) psql -p 5434 -U postgres -X -d dst -c "select pg_terminate_backend(active_pid) from pg_replication_slots where slot_name = 'repca2'"

8. Adding one more record in the replicated table

provider (main)

```
postgres@pghack-debian-8:~$ psql -p 5432  -U postgres -1 -X -d src -f- <<'EOF'
> insert into cats (cat_name) values ('category 4');
> EOF
INSERT 0 1
```

9. As expected subscriber's standby falls behind

subscriber (repca1)

```
postgres@pghack-debian-8:~$ psql -p 5434 -U postgres -X -d dst -c 'select * from cats'
 cat_id |  cat_name
--------+------------
      1 | category 1
      2 | category 2
      3 | category 3
      4 | category 4
(4 rows)
```

subscriber standby (repca2)

```
postgres@pghack-debian-8:~$ psql -p 5435 -U postgres -X -d dst -c 'select * from cats'
 cat_id |  cat_name
--------+------------
      1 | category 1
      2 | category 2
      3 | category 3
(3 rows)
```

(3) REDO

reposition source (subscriber's crash)

10. “Subscriber's crash”. Subscriber's standby is still behind


```
postgres@pghack-debian-8:~$ /usr/lib/postgresql/10/bin/pg_ctl -D /var/lib/postgresql/10/repca1 -m i -o "-p 5434" stop
waiting for server to shut down.... done
server stopped
postgres@pghack-debian-8:~$ psql -p 5435 -U postgres -X -d dst -c 'select * from cats'
 cat_id |  cat_name
--------+------------
      1 | category 1
      2 | category 2
      3 | category 3
(3 rows)
```

(3) REDO

reposition source (subscriber's crash)

10. “Subscriber's crash”. Subscriber's standby is still behind

subscriber (repca1)

subscriber standby (repca2)

```
postgres@pghack-debian-8:~$ /usr/lib/postgresql/10/bin/pg_ctl -D /var/lib/postgresql/10/repca1 -m i -o "-p 5434" stop
waiting for server to shut down.... done
server stopped
postgres@pghack-debian-8:~$ psql -p 5435 -U postgres -X -d dst -c 'select * from cats'
 cat_id |  cat_name
--------+------------
      1 | category 1
      2 | category 2
      3 | category 3
(3 rows)
```

11. Dropping slot which was consumed by crashed subscriber's primary

provider (main)

```
postgres@pghack-debian-8:~$ psql -p 5432 -U postgres -X -d src -c "select pg_drop_replication_slot('repca1')"
 pg_drop_replication_slot
--------------------------

(1 row)
```

provider (main)

```
postgres@pghack-debian-8:~$ psql -p 5432  -U postgres -X -d src -c 'select * from pg_replication_slots'
 slot_name |  plugin  | slot_type | datoid | database | temporary | active | active_pid | xmin | catalog_xmin | restart_lsn | confirmed_flush_lsn
-----------+----------+-----------+--------+----------+-----------+--------+------------+------+--------------+-------------+---------------------
 repca2    | pgoutput | logical   |  16384 | src      | f         | f      |            |      |          560 | 0/16633D8   | 0/1663410
(1 row)
```

(3) REDO

reposition source (subscriber's crash)

12. Checking pg_replication_origin_status


```
postgres@pghack-debian-8:~$ psql -p 5435 -U postgres -X -d dst -c 'select * from pg_replication_origin_status'
 local_id | external_id | remote_lsn | local_lsn
----------+-------------+------------+-----------
        1 | pg_16415    | 0/16632F8  | 0/30343B0
(1 row)
```

(3) REDO
reposition source (subscriber's crash)

12. Checking pg_replication_origin_status

subscriber standby (repca2)

```
postgres@pghack-debian-8:~$ psql -p 5435 -U postgres -X -d dst -c 'select * from pg_replication_origin_status'
 local_id | external_id | remote_lsn | local_lsn
----------+-------------+------------+-----------
        1 | pg_16415    | 0/16632F8  | 0/30343B0
(1 row)
```

13. Sync “subscriber's standby slot” actual with subscriber's standby pg_replication_origin

provider (main)

```
postgres@pghack-debian-8:~$ psql -p 5432 -U postgres -X -d src -c "select * from pg_logical_slot_get_binary_changes('repca2'::name, '0/16632F8'::pg_lsn,
\ lsn | xid | data
-----+-----+------
(0 rows)
```

(3) REDO
reposition source (subscriber's crash)

12. Checking pg_replication_origin_status

subscriber standby (repca2)

```
postgres@pghack-debian-8:~$ psql -p 5435 -U postgres -X -d dst -c 'select * from pg_replication_origin_status'
 local_id | external_id | remote_lsn | local_lsn
----------+-------------+------------+-----------
        1 | pg_16415    | 0/16632F8  | 0/30343B0
(1 row)
```

13. Sync “subscriber's standby slot” actual with subscriber's standby pg_replication_origin

provider (main)

```
postgres@pghack-debian-8:~$ psql -p 5432 -U postgres -X -d src -c "select * from pg_logical_slot_get_binary_changes('repca2'::name, '0/16632F8'::pg_lsn,
\ lsn | xid | data
-----+-----+------
(0 rows)
```

14. Subscriber hasn't had new changes yet

subscriber standby (repca2)

```
postgres@pghack-debian-8:~$ psql -p 5435 -U postgres -X -d dst -c 'select * from cats'
 cat_id |  cat_name
--------+------------
      1 | category 1
      2 | category 2
      3 | category 3
(3 rows)
```

(3) REDO
reposition source (subscriber's crash)

15. Promoting subscriber's standby

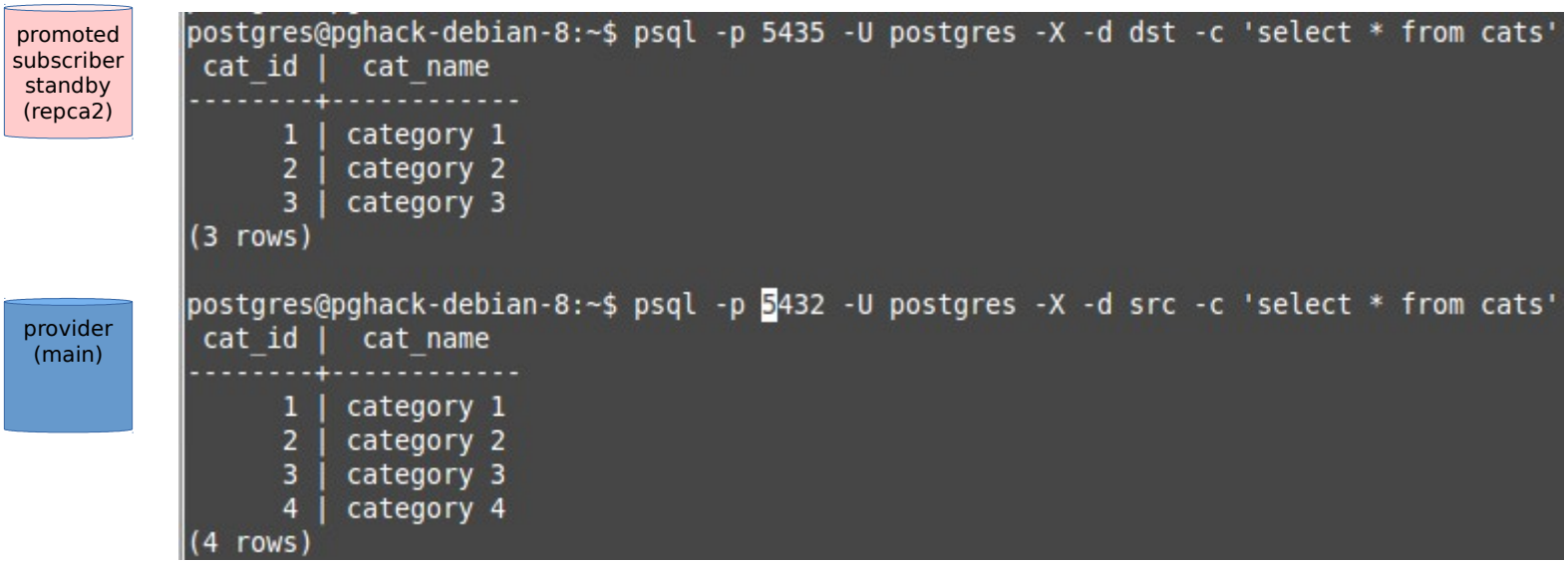

```
postgres@pghack-debian-8:~$ /usr/lib/postgresql/10/bin/pg_ctl -D /var/lib/postgresql/10/repca2 promote
waiting for server to promote.... done
server promoted
postgres@pghack-debian-8:~$ tail /var/log/postgresql/postgresql-10-repca2.log
```

```
2018-05-11 14:23:58.311 MSK [13852] LOG:  received promote request
2018-05-11 14:23:58.311 MSK [13852] LOG:  redo done at 0/3034490
2018-05-11 14:23:58.311 MSK [13852] LOG:  last completed transaction was at log time 2018-05-11 13:25:07.943351+03
2018-05-11 14:23:58.314 MSK [13852] LOG:  selected new timeline ID: 2
2018-05-11 14:23:58.362 MSK [13852] LOG:  archive recovery complete
2018-05-11 14:23:58.367 MSK [13851] LOG:  database system is ready to accept connections
2018-05-11 14:23:58.370 MSK [19305] LOG:  logical replication apply worker for subscription "repca1" has started
2018-05-11 14:23:58.372 MSK [19305] ERROR:  could not start WAL streaming: ERROR:  replication slot "repca1" does not exist
2018-05-11 14:23:58.373 MSK [13851] LOG:  worker process: logical replication worker for subscription 16415 (PID 19305) exited with exit code 1
2018-05-11 14:24:03.380 MSK [19314] LOG:  logical replication apply worker for subscription "repca1" has started
2018-05-11 14:24:03.382 MSK [19314] ERROR:  could not start WAL streaming: ERROR:  replication slot "repca1" does not exist
2018-05-11 14:24:03.383 MSK [13851] LOG:  worker process: logical replication worker for subscription 16415 (PID 19314) exited with exit code 1
2018-05-11 14:24:08.390 MSK [19321] LOG:  logical replication apply worker for subscription "repca1" has started
2018-05-11 14:24:08.393 MSK [19321] ERROR:  could not start WAL streaming: ERROR:  replication slot "repca1" does not exist
2018-05-11 14:24:08.393 MSK [13851] LOG:  worker process: logical replication worker for subscription 16415 (PID 19321) exited with exit code 1
```

(3) REDO

reposition source (subscriber's crash)

16. Promoted subscriber's standby is still behind



(3) REDO
reposition source (subscriber's crash)

16. Promoted subscriber's standby is still behind

promoted subscriber standby (repca2)

provider (main)

```
postgres@pghack-debian-8:~$ psql -p 5435 -U postgres -X -d dst -c 'select * from cats'
 cat_id |  cat_name
--------+------------
      1 | category 1
      2 | category 2
      3 | category 3
(3 rows)

postgres@pghack-debian-8:~$ psql -p 5432 -U postgres -X -d src -c 'select * from cats'
 cat_id |  cat_name
--------+------------
      1 | category 1
      2 | category 2
      3 | category 3
      4 | category 4
(4 rows)
```

17. Alter subscription: set actual slot, we prepared previously

promoted subscriber standby (repca2)

```
postgres@pghack-debian-8:~$ psql -p 5435 -U postgres -X -d dst -f- <<'EOF'
> alter subscription repca1 set (slot_name = repca2);
> EOF
ALTER SUBSCRIPTION
```

(3) REDO

reposition source (subscriber's crash)

18. Slot turned on and started being consumed by subscriber


```
postgres@pghack-debian-8:~$ psql -p 5432  -U postgres -X -d src -c 'select * from pg_replication_slots'
 slot_name |  plugin  | slot_type | datoid | database | temporary | active | active_pid | xmin | catalog_xmin | restart_lsn | confirmed_flush_lsn
-----------+----------+-----------+--------+----------+-----------+--------+------------+------+--------------+-------------+---------------------
 repca2    | pgoutput | logical   |  16384 | src      | f         | t      |      20298 |      |          561 | 0/1663740   | 0/1663858
(1 row)
```

(3) REDO
reposition source (subscriber's crash)

18. Slot turned on and started being consumed by subscriber

provider (main)

```
postgres@pghack-debian-8:~$ psql -p 5432  -U postgres -X -d src -c 'select * from pg_replication_slots'
 slot_name |  plugin  | slot_type | datoid | database | temporary | active | active_pid | xmin | catalog_xmin | restart_lsn | confirmed_flush_lsn
-----------+----------+-----------+--------+----------+-----------+--------+------------+------+--------------+-------------+---------------------
 repca2    | pgoutput | logical   |  16384 | src      | f         | t      |      20298 |      |          561 | 0/1663740   | 0/1663858
(1 row)
```

19. Subscriber replayed the changes

promoted subscriber standby (repca2)

```
postgres@pghack-debian-8:~$ psql -p 5435 -U postgres -X -d dst -c 'select * from pg_replication_origin_status'
 local_id | external_id | remote_lsn | local_lsn
----------+-------------+------------+-----------
        1 | pg_16415    | 0/1663740  | 0/3034B58
(1 row)
```

provider (main)

```
postgres@pghack-debian-8:~$ psql -p 5435 -U postgres -X -d dst -c 'select * from cats'
 cat_id |  cat_name
--------+------------
      1 | category 1
      2 | category 2
      3 | category 3
      4 | category 4
(4 rows)
```

(3) REDO

Algorithm for consuming a reserved slot

- Taking the LSN position for the subscriber's master's slot from pg_replication_slots


```
pghack@pghack-debian-8:~$ psql --cluster 10/main -U postgres -X -d src -c 'select * from pg_replication_slots'
 slot_name |  plugin  | slot_type | datoid | database | temporary | active | active_pid | xmin | catalog_xmin | restart_lsn | confirmed_flush_lsn
-----------+----------+-----------+--------+----------+-----------+--------+------------+------+--------------+-------------+---------------------
 repca2    | pgoutput | logical   |  16384 | src      | f         | f      |            |      |          559 | 0/3037800   | 0/3037B68
 repca1    | pgoutput | logical   |  16384 | src      | f         | t      |      27782 |      |          560 | 0/3037B68   | 0/3037BA0
```

(3) REDO

Algorithm for consuming a reserved slot

- Taking the LSN position for the subscriber's master's slot from pg_replication_slots

provider (main)

```
pghack@pghack-debian-8:~$ psql --cluster 10/main -U postgres -X -d src -c 'select * from pg_replication_slots'
 slot_name |  plugin  | slot_type | datoid | database | temporary | active | active_pid | xmin | catalog_xmin | restart_lsn | confirmed_flush_lsn
-----------+----------+-----------+--------+----------+-----------+--------+------------+------+--------------+-------------+---------------------
 repca2    | pgoutput | logical   |  16384 | src      | f         | f      |            |      |          559 | 0/3037800   | 0/3037B68
 repca1    | pgoutput | logical   |  16384 | src      | f         | t      |      27782 |      |          560 | 0/3037B68   | 0/3037BA0
```

subscriber (repca1)

- Check pg_replication_origin on the subscriber's master

(3) REDO

Algorithm for consuming a reserved slot

- Taking the LSN position for the subscriber's master's slot from pg_replication_slots

provider (main)

```
pghack@pghack-debian-8:~$ psql --cluster 10/main -U postgres -X -d src -c 'select * from pg_replication_slots'
 slot_name |  plugin  | slot_type | datoid | database | temporary | active | active_pid | xmin | catalog_xmin | restart_lsn | confirmed_flush_lsn
-----------+----------+-----------+--------+----------+-----------+--------+------------+------+--------------+-------------+---------------------
 repca2    | pgoutput | logical   |  16384 | src      | f         | f      |            |      |          559 | 0/3037800   | 0/3037B68
 repca1    | pgoutput | logical   |  16384 | src      | f         | t      |      27782 |      |          560 | 0/3037B68   | 0/3037BA0
```

subscriber (repca1)

- Check pg_replication_origin on the subscriber's master
- Wait until the subscriber's master's pg_replication_origin is seen on the subscriber's standby side

subscriber standby (repca2)

(3) REDO

Algorithm for consuming a reserved slot

- Taking the LSN position for the subscriber's master's slot from pg_replication_slots

provider (main)

```
pghack@pghack-debian-8:~$ psql --cluster 10/main -U postgres -X -d src -c 'select * from pg_replication_slots'
 slot_name |  plugin  | slot_type | datoid | database | temporary | active | active_pid | xmin | catalog_xmin | restart_lsn | confirmed_flush_lsn
-----------+----------+-----------+--------+----------+-----------+--------+------------+------+--------------+-------------+---------------------
 repca2    | pgoutput | logical   |  16384 | src      | f         | f      |            |      |          559 | 0/3037800   | 0/3037B68
 repca1    | pgoutput | logical   |  16384 | src      | f         | t      |      27782 |      |          560 | 0/3037B68   | 0/3037BA0
```

subscriber (repca1)

- Check pg_replication_origin on the subscriber's master
- Wait until the subscriber's master's pg_replication_origin is seen on the subscriber's standby side

subscriber standby (repca2)

- Consume the reserved slot (subscriber's standby slot)

provider (main)

```
select * from pg_logical_slot_get_binary_changes('repca2'::name,
'0/3037B68'::pg_lsn, null::int, variadic array['proto_version', '1',
'publication_names', 'pub'])"
```

Recovery use cases

(1) Reinitializing subscriber from another subscriber

(2) UNDO recovery on the destination side

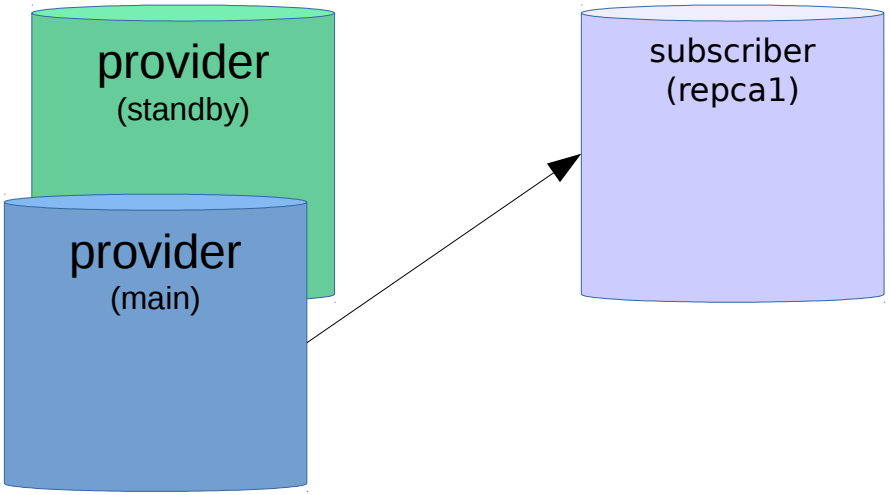
(3) REDO - reposition source (subscriber's crash)

(4) REDO 2 - on provider's side (provider's crash and switching to the provider's standby, subscriber is falling behind)

* https://github.com/avito-tech/dba-docs/blob/master/PGCon2018_Ottawa/Recovery_Use_Cases_for_Logical_Replication_in_PostgreSQL10.txt

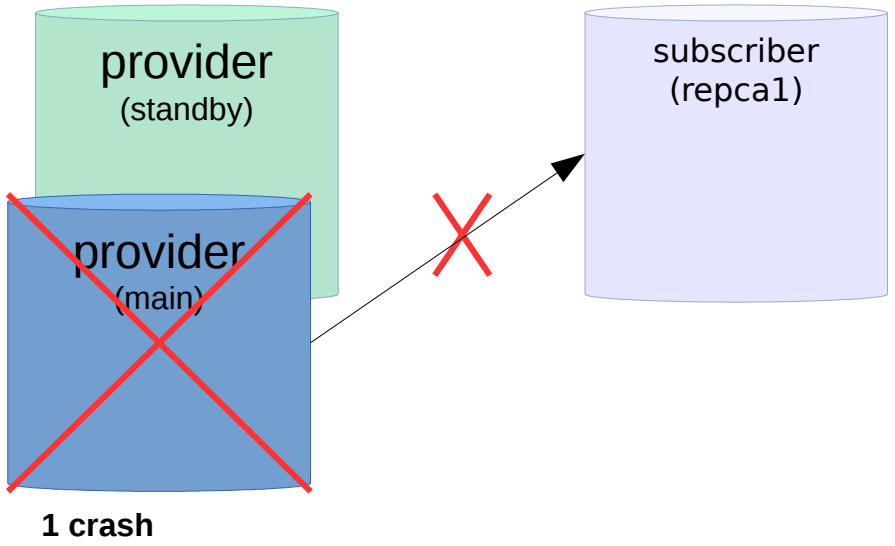
(4) Redo 2

on provider's side (provider's crash and switching to the provider's standby, subscriber is falling behind)



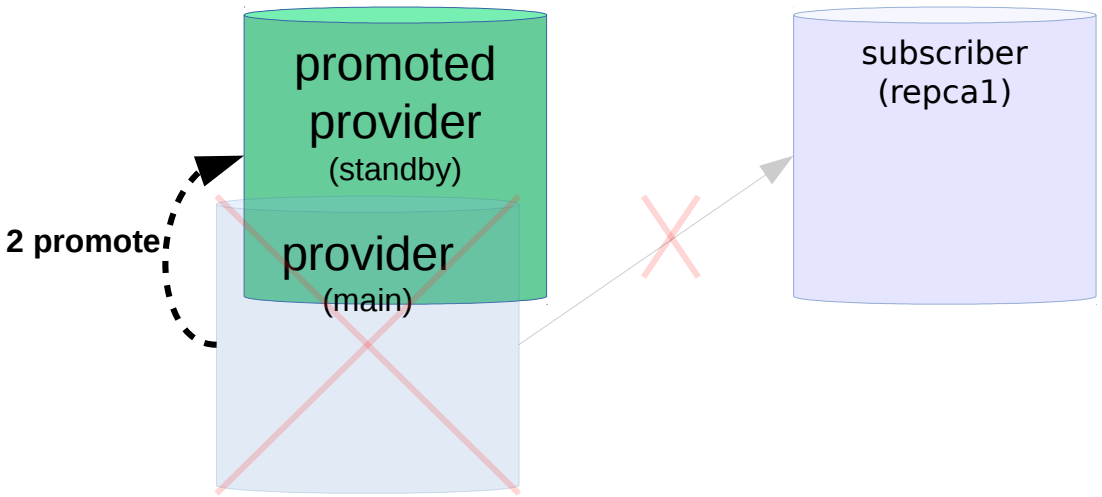
(4) Redo 2

on provider's side (provider's crash and switching to the provider's standby, subscriber is falling behind)



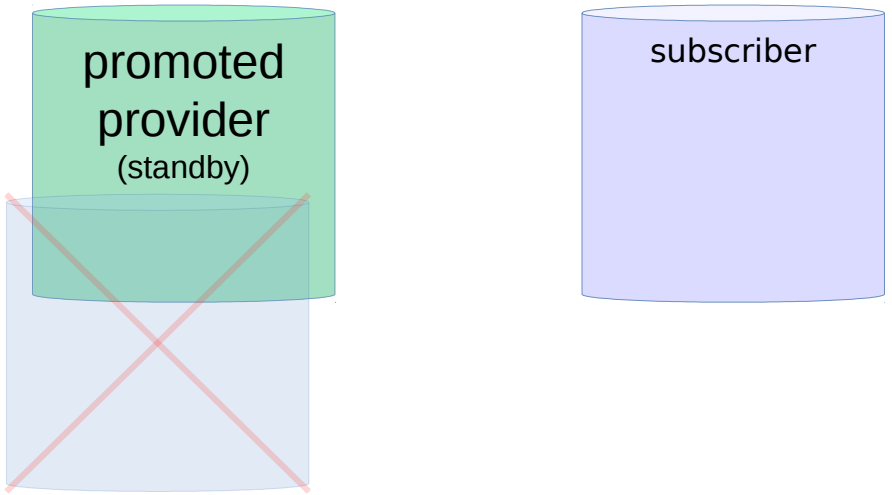
(4) Redo 2

on provider's side (provider's crash and switching to the provider's standby, subscriber is falling behind)



(4) Redo 2

on provider's side (provider's crash and switching to the provider's standby, subscriber is falling behind)

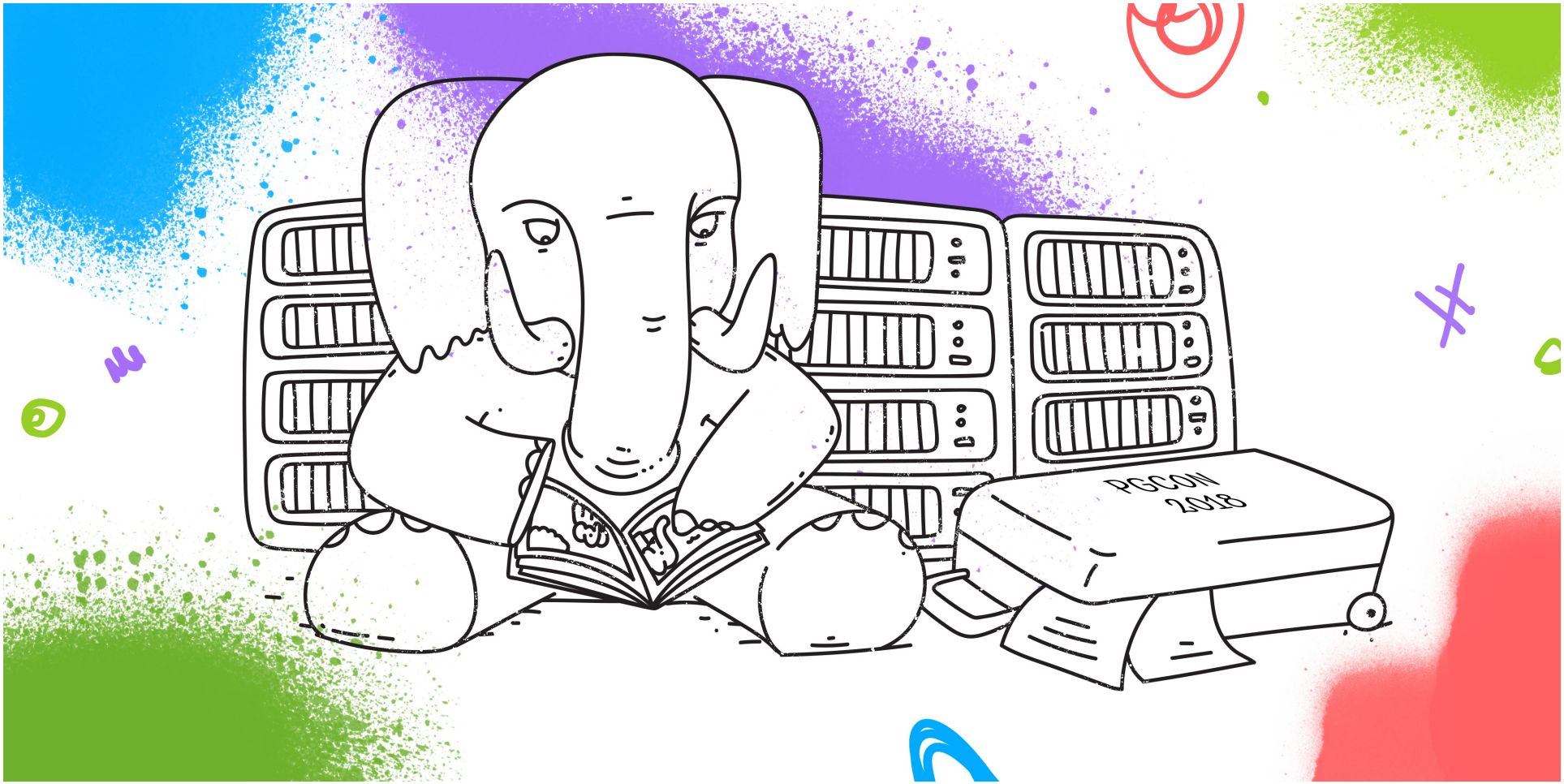


Data loss

* https://wiki.postgresql.org/wiki/Failover_slots

	SkyTools	PostgreSQL10
(1) Reinitializing subscriber from another subscriber	OK	OK
		To do 1: Make pg_replication_origin “transactional” ~ “pin to snapshot”
		To do 2: Dump the state of logical replication with pg_dump or option to enable it.
(2) UNDO recovery on the destination side		OK
		To do 3: Implement "logical UNDO" / or SQL API
(3) REDO reposition source (subscriber's crash)		OK
		To do 4: Track the progress of consuming a logical slot on the subscriber: “provider_restart_lsn” to make it easier to implement Redo case
		To do 5: Compare LSN between subscriber and provider (SerialConsumer). Raise error when provider state does not match subscriber
		To do 6: Function to move a slot on provider (waiting for PostgreSQL 11 pg_replication_slot_advance (slot_name name, upto_lsn pg_lsn))
(4) REDO 2 - on provider's side (provider's crash and switching to the provider's standby, subscriber is falling behind)		Blocker for production usage To do 7: Logical Slot on standby (the slot is not replicated to standby). There will be a gap in data after promotion.
		? To do 8: Binary provider standby must wait for a logical replica ~ “replication_target_cmd” to logical subscriber
*Event tracking		OK
		overhead on proxy table/ write custom decoder + pg_logical_emit_message
*Unlogged table		:-)
*Sequence		Not replicated

Thank you!



https://github.com/avito-tech

kevteev@avito.ru, tmihail@bk.ru

* https://github.com/avito-tech/dba-docs/blob/master/PGCon2018_Ottawa/Recovery_Use_Cases_for_Logical_Replication_in_PostgreSQL10.txt