Postgres 16 highlight: Logical decoding on standby

Bertrand Drouvot

Senior SDE at AWS
FOSDEM PGDay 2024

About me

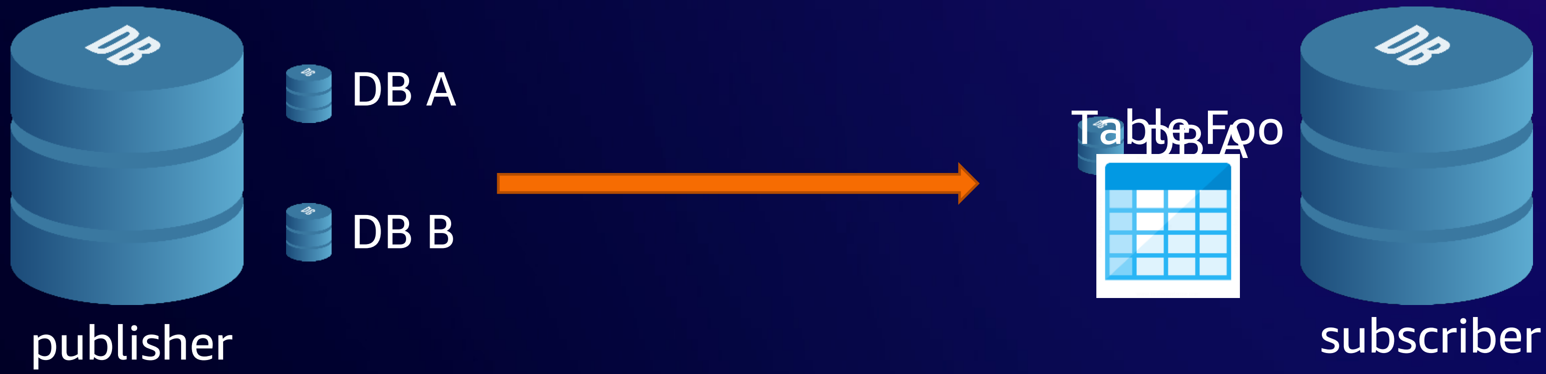
- Working in databases for about 25 years
- Oak table member
- ~~Oracle ACE~~
- Fell in love for PostgreSQL
- Working at AWS (RDS Open Source Databases)
- Twitter: @BertrandDrouvot
- https://bdrouvot.github.io/

Agenda

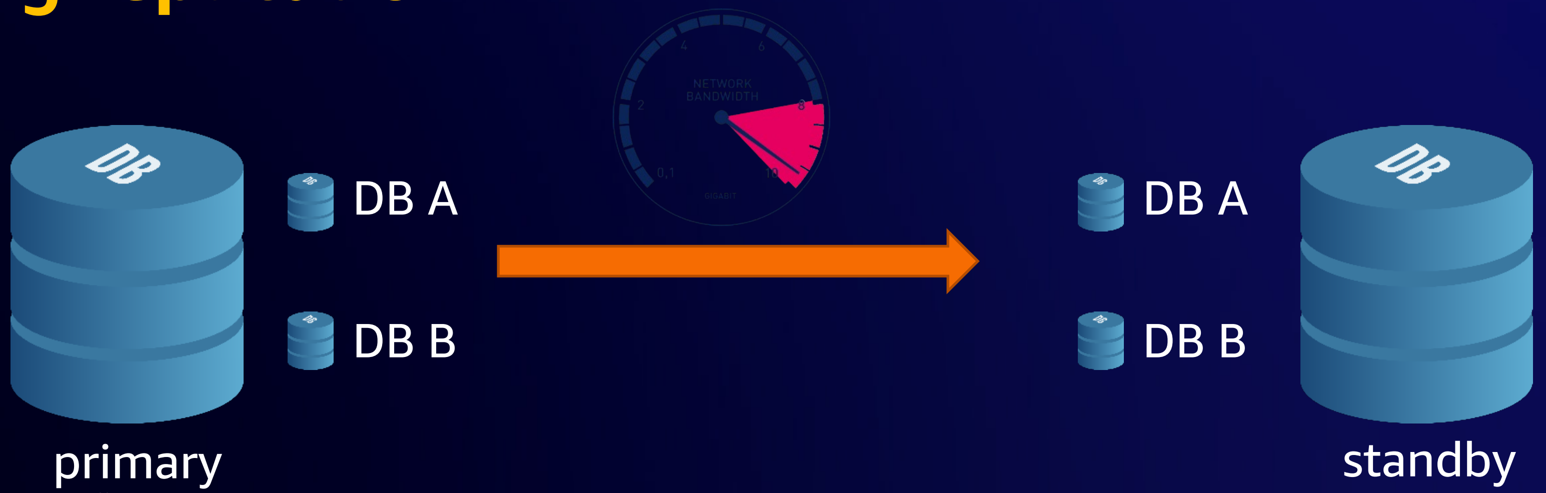
- Logical decoding?
- What is logical decoding on standby about?
- History of this effort
- Main challenges that have been faced during the implementation
- How the challenges have been addressed
- This new feature in action (live demo) with use cases
- "Transparent" logical replication slot failover from primary to standby

What is logical decoding?

Logical replication

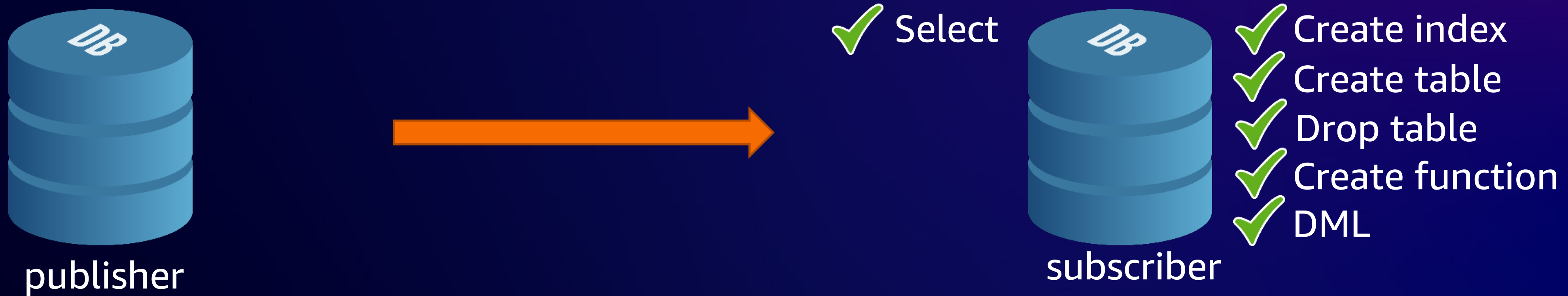


Streaming replication

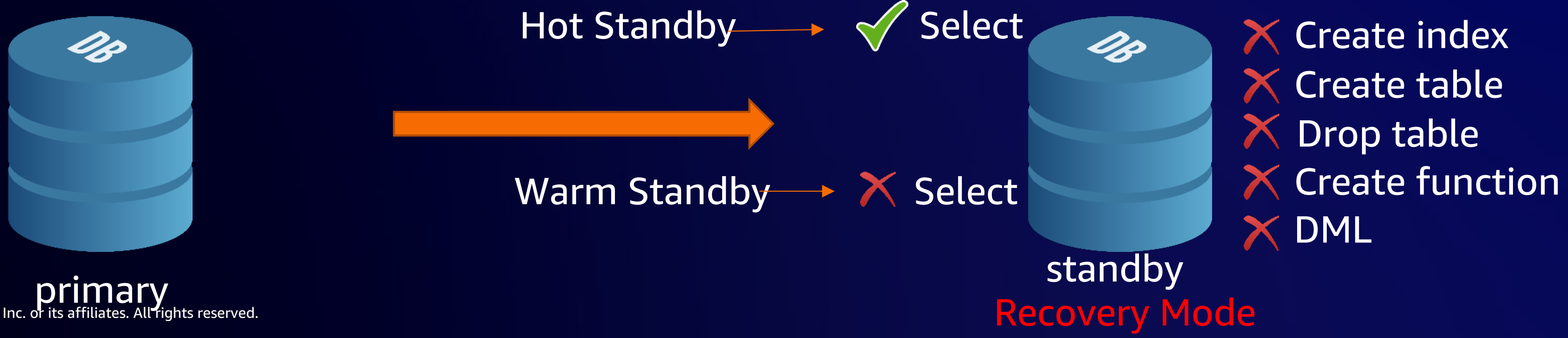


What is logical decoding?

Logical replication

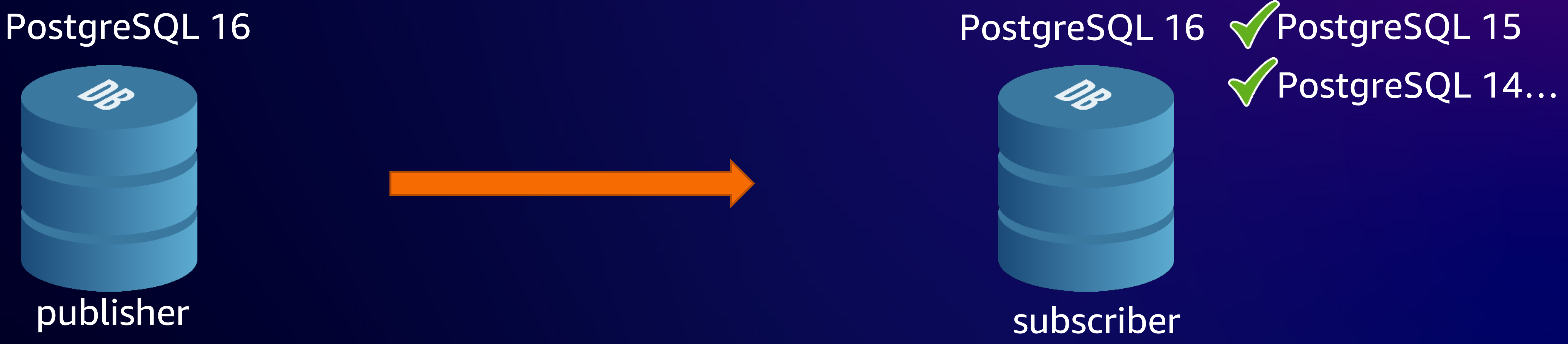


Streaming replication

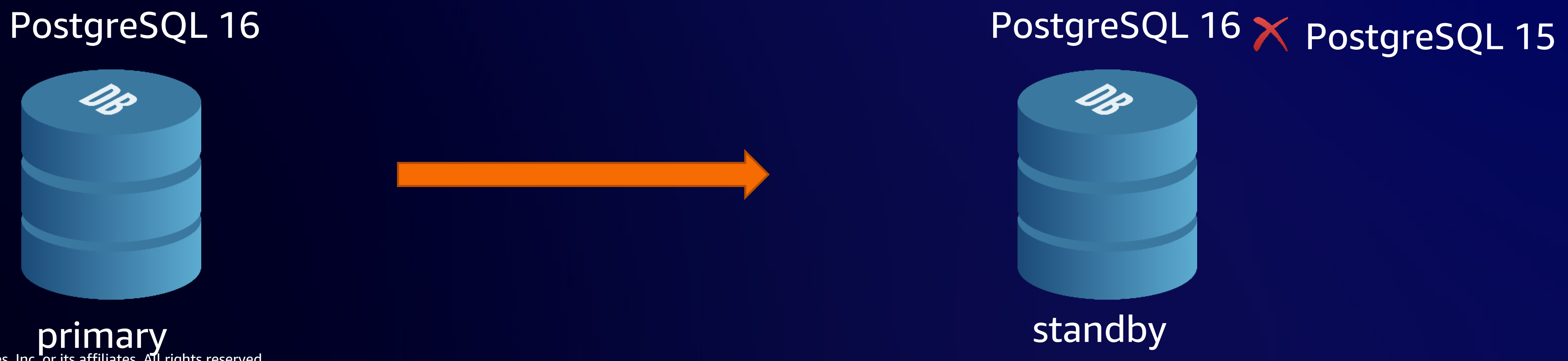


What is logical decoding?

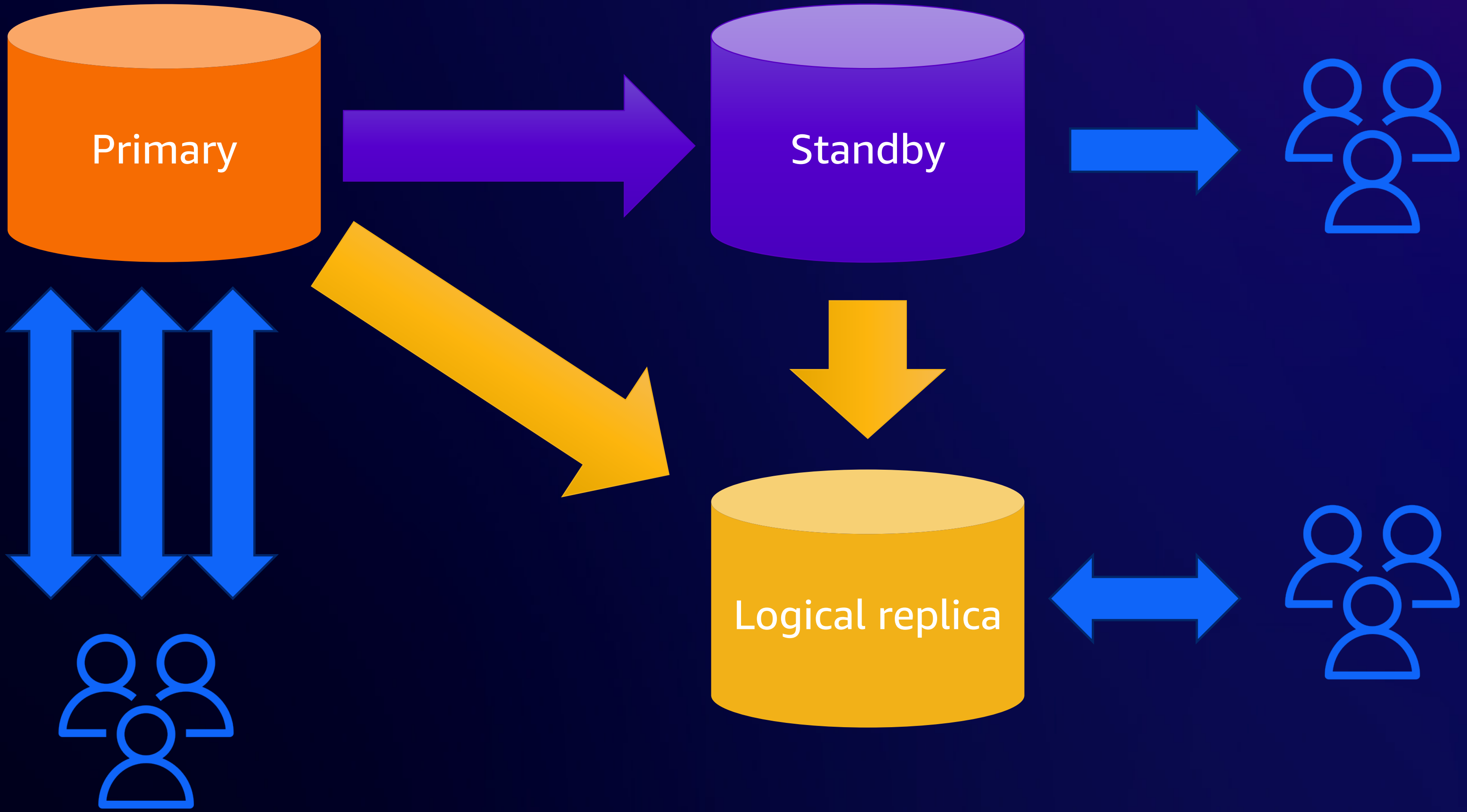
Logical replication



Streaming replication



What is logical decoding on standby about?



History of this effort

- Initial proposal in 2016
- Next major effort in 2018
- Efforts continue on and off for years
- Very challenging to design

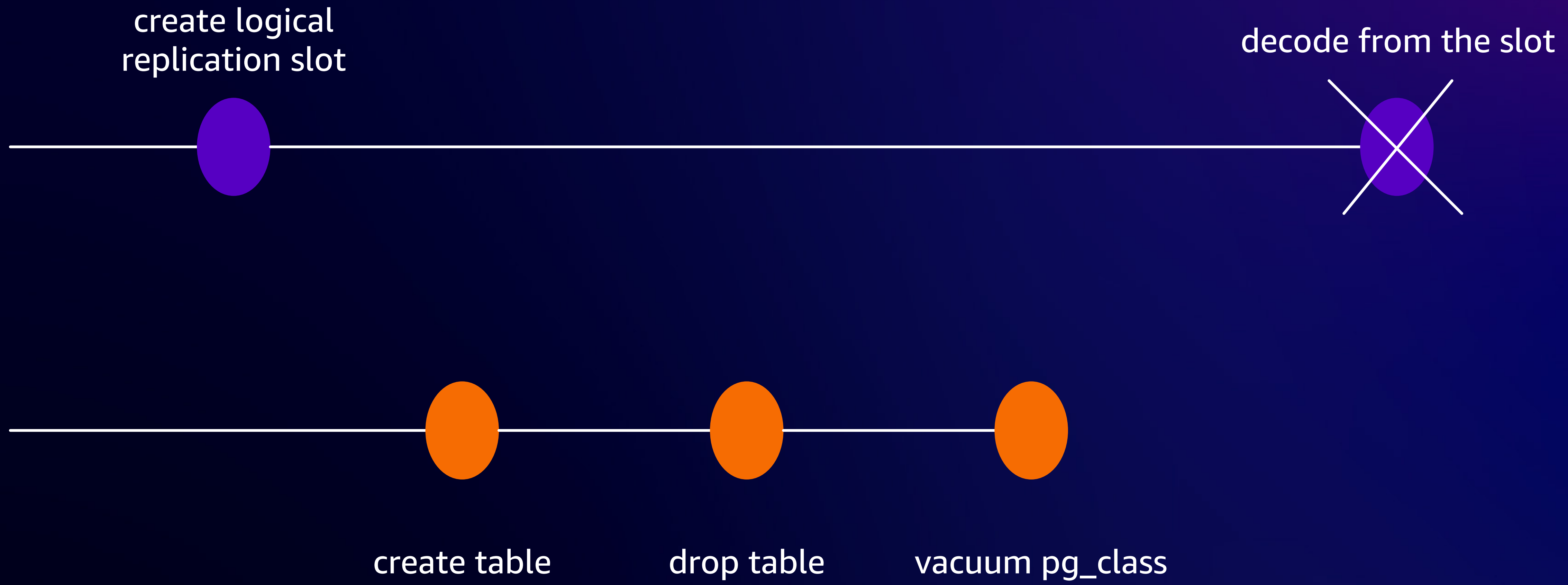
Main challenges that have been faced

- How to detect and handle conflicts
- How to handle promotion (timeline change)
- Logical walsenders awakened too early
- …

How to detect row removal conflicts

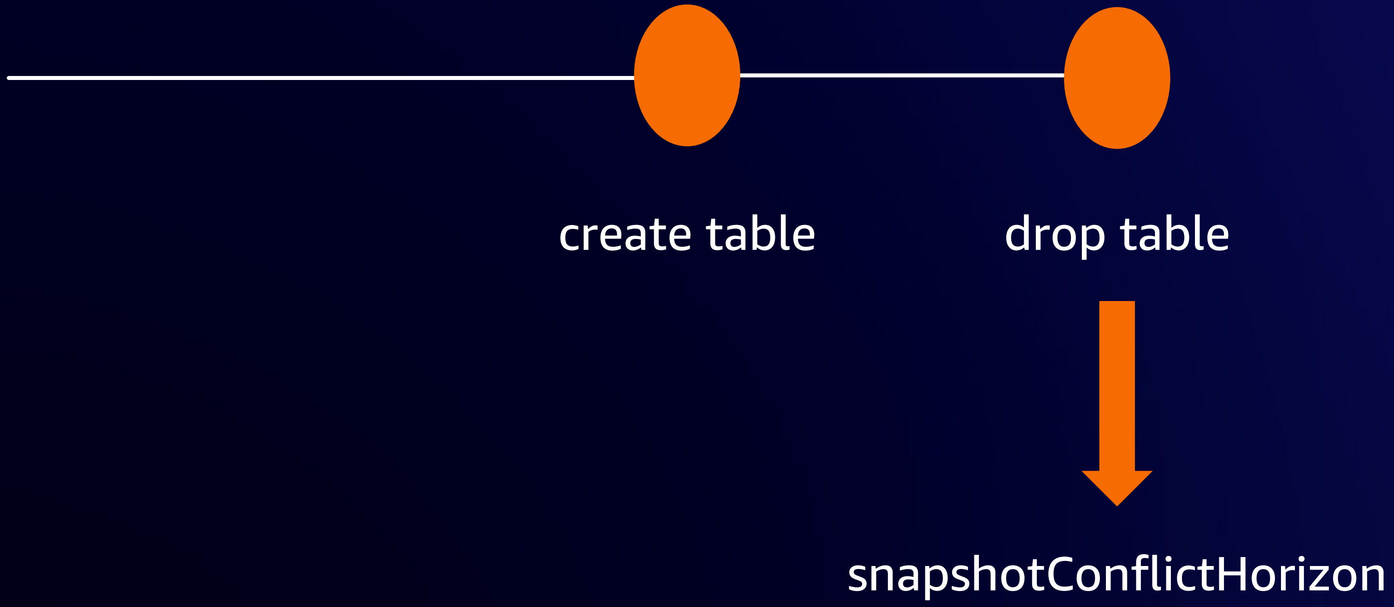
- Replay of WAL from the primary might remove data that is needed by logical decoding, causing error(s) on the standby

How to detect row removal conflicts



How to detect row removal conflicts

- Affects only catalog or user-catalog tables
- We need/have the snapshotConflictHorizon for each change, just as we do for physical replication conflicts

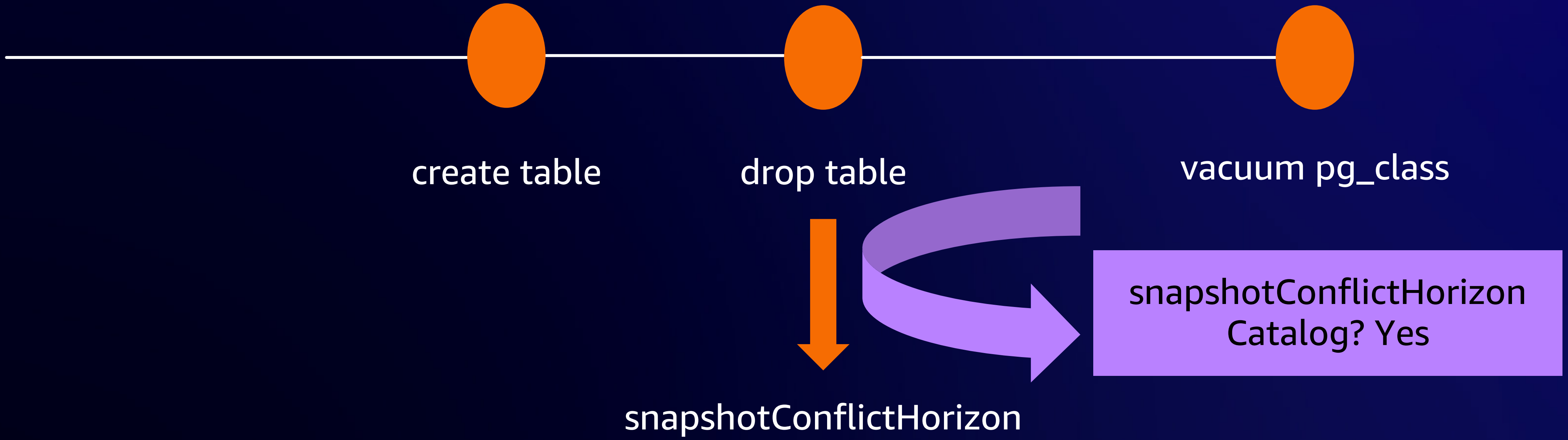


How to detect row removal conflicts

- Need to know if it affects a catalog or user-catalog table
- Problem: the startup process (doing the recovery) can't access catalog contents

How to detect row removal conflicts

- Every WAL record that potentially removes data from the index or heap must carry a flag indicating whether or not it is one that is related to catalog / user-catalog



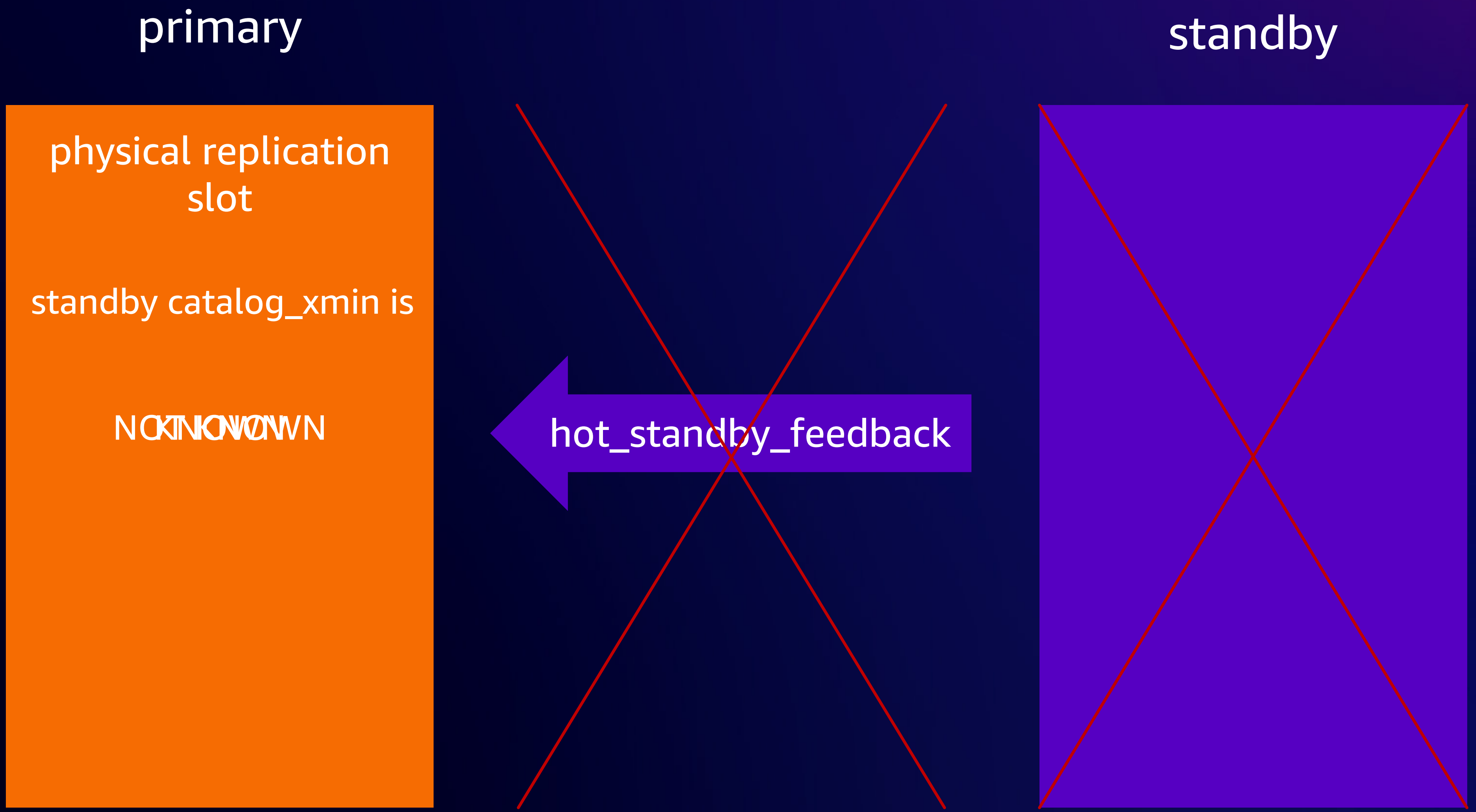
How to detect row removal conflicts

- Easy for tables but not for indexes
- 3 approaches have been tested!
    - Needs table_open() on the heap relation (from the index relation)
    - Adds dependency in pg_index
    - Pass down the heap relation to the functions linked to the WAL records of interest
- Choice: pass down the heap relation to the functions linked to the WAL records of interest

How to detect row removal conflicts

- But is that even needed if hot_standby_feedback is set to on?
  - Yes
  - No
  - It depends
- Depends if there is a physical replication slot (primary_slot_name) between the primary and the standby

How to detect row removal conflicts



How to detect row removal conflicts

- Did not want to mandate that configuration
- We can't ensure that the configuration is accurate all the time (while restoring from archives we can't rely on knowing that the slot still exists on the primary)
- Want to limit space use on the primary

How to detect wal_level conflict

- wal_level < logical on primary has to lead to conflict
- Not so hard as part of xl_parameter_change WAL record

How to deal with conflicts?

- Remove slots?
- Invalidate slots?

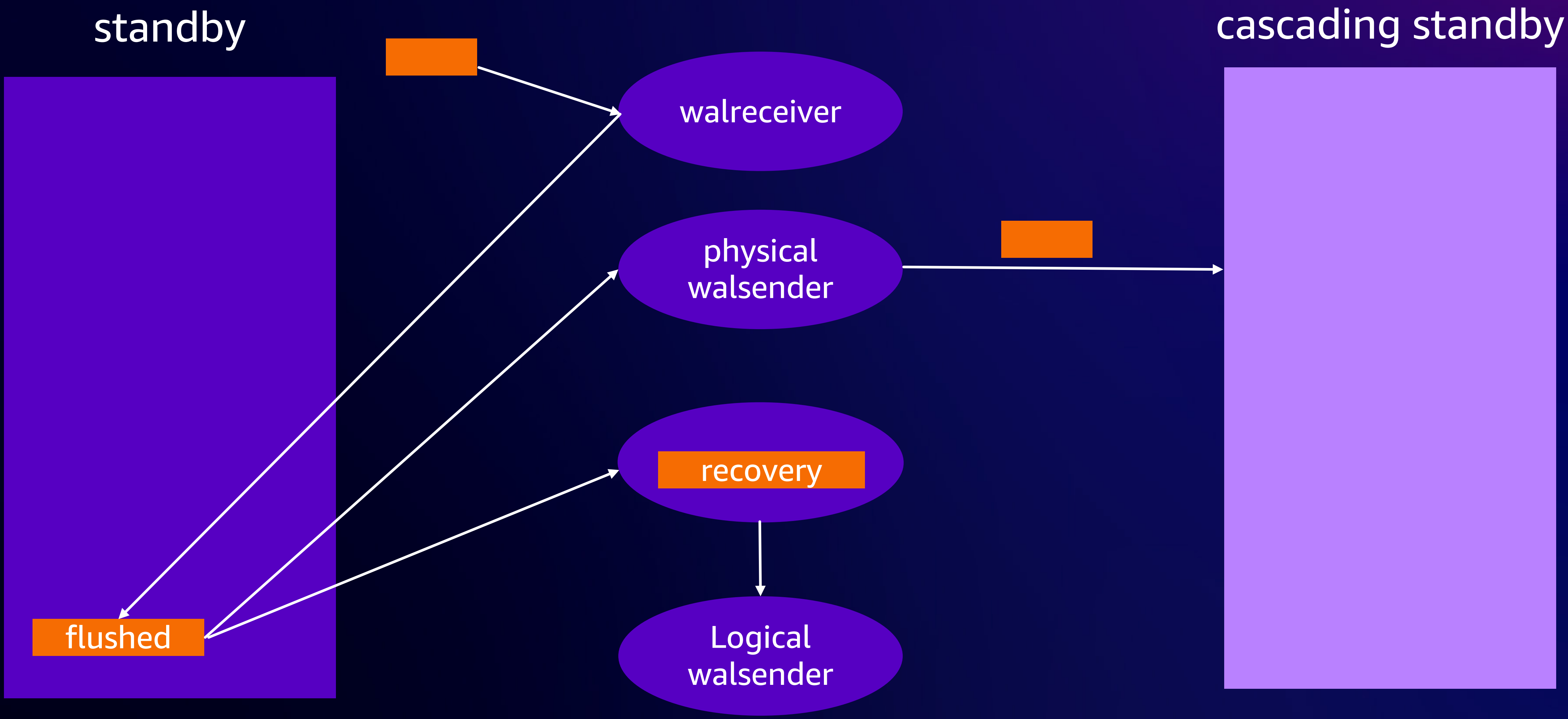
How to handle promotion?

- Should not break running decoding
- Have to use the right timeline
- Not such a big deal but took some time too
- Timeline related code changes during the development cycle

Logical walsenders awakened too early

- Previously, all walsenders (physical and logical) were awakened when the WAL was flushed
- Fine on primary but not on standby for logical walsenders (as might still not be replayed yet)
- Need to find a way to wake them up only when applied/replayed
- Build a new machinery for this based on walsender type

Logical walsenders awakened too early



And…

- Some WAL records alignment issues due the new bool in some WAL records
- Had to create pg_log_standby_snapshot() (to generate a xl_running_xacts WAL record without triggering an “expensive” checkpoint on the primary)
- Lot of tests (035_standby_logical_decoding.pl is the 2nd largest TAP test perl script regarding the number of lines)
- …

Finally

- Pass down table relation into more index relation functions
- Add info in WAL records in preparation for logical slot conflict handling (this commit message also explains the overall design)
- Replace replication slot’s invalidated_at LSN with an enum
- Prevent use of invalidated logical slot in CreateDecodingContext()
- Support invalidating replication slots due to horizon and wal_level
- Handle logical slot conflicts on standby
- For cascading replication, wake physical and logical walsenders separately
- Allow logical decoding on standbys

Demo

Thank you!

Bertrand Drouvot

bdrouvot@amazon.com

@BertrandDrouvot