Logical Decoding :

Replicate or do anything you want

- Amit Khandekar

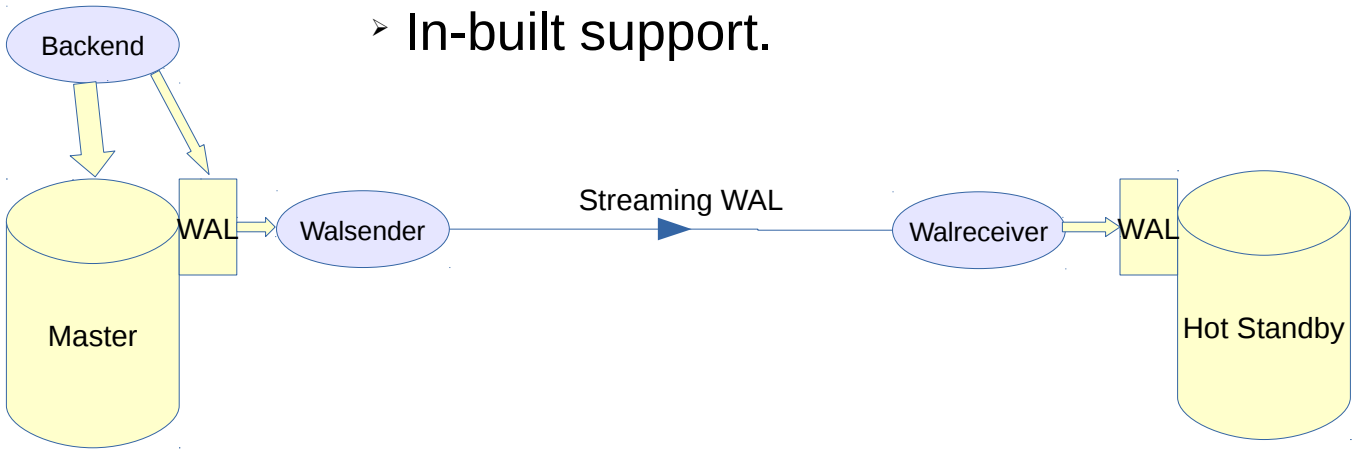
Agenda

- Background
- Logical decoding
  - Architecture
  - Configuration
- Use cases

© 2016 EnterpriseDB Corporation. All rights reserved.

Physical Replication

- Disk block level changes
- Low overhead
- WAL is already there, use it.
- In-built support.

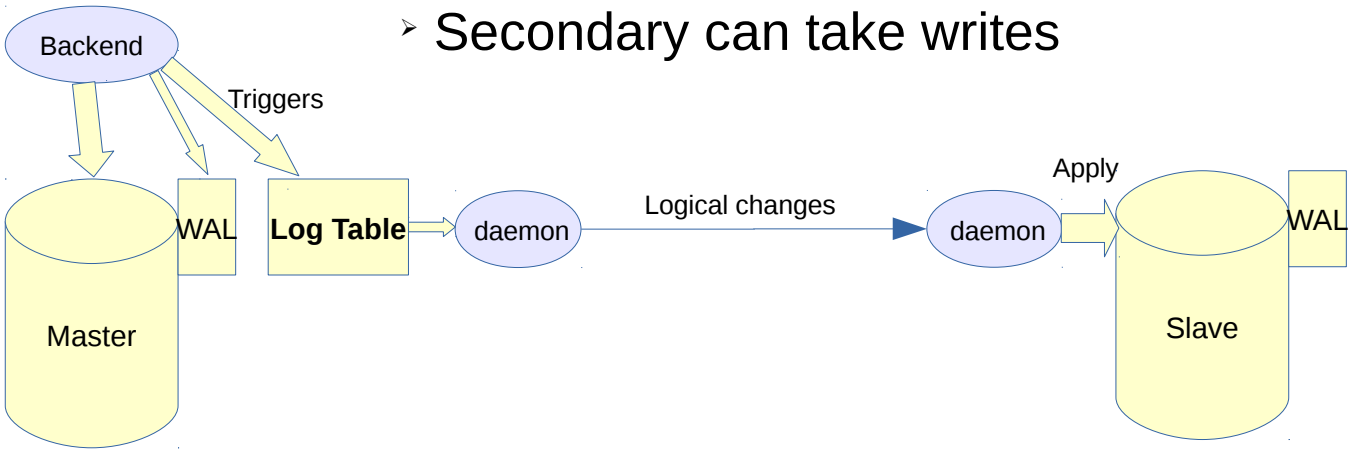


- Have to replicate everything
- Slave has to be PostgreSQL
- Slave cannot have a different PostgreSQL major version

© 2016 EnterpriseDB Corporation. All rights reserved.

Logical Replication : trigger based

- Logical changes (INSERT/DELETE etc)
- Selective replication
- Can replicate to different RDBMS
- Secondary can take writes

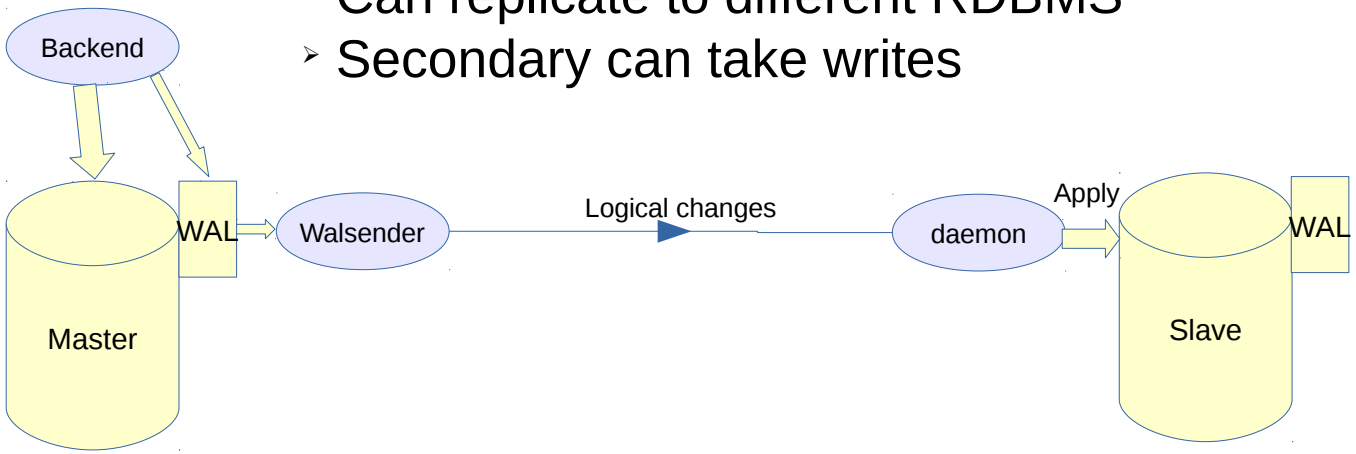


- **High overhead of writes : Additional writes to log table**
- Setup is outside PostgreSQL

© 2016 EnterpriseDB Corporation. All rights reserved.

Logical Replication : WAL based

- No separate log file.
- WAL file is already there. Re-use it.
- No external setup at master
- In-built support
- Selective replication
- Can replicate to different RDBMS
- Secondary can take writes



- Overhead of logical decoding, and APPLY step

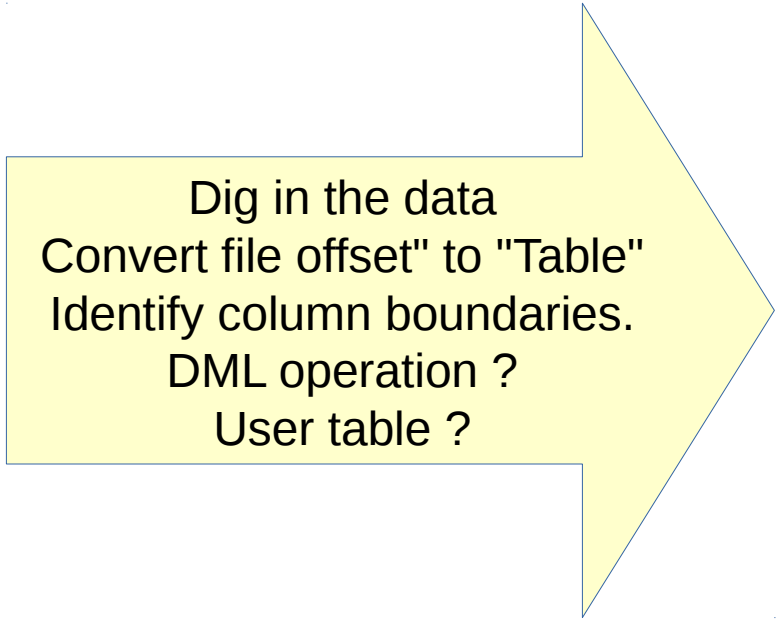
© 2016 EnterpriseDB Corporation. All rights reserved.

Logical decoder

WAL segment

Tablespace oid
File oid
Offset xxx on that file

Data ….



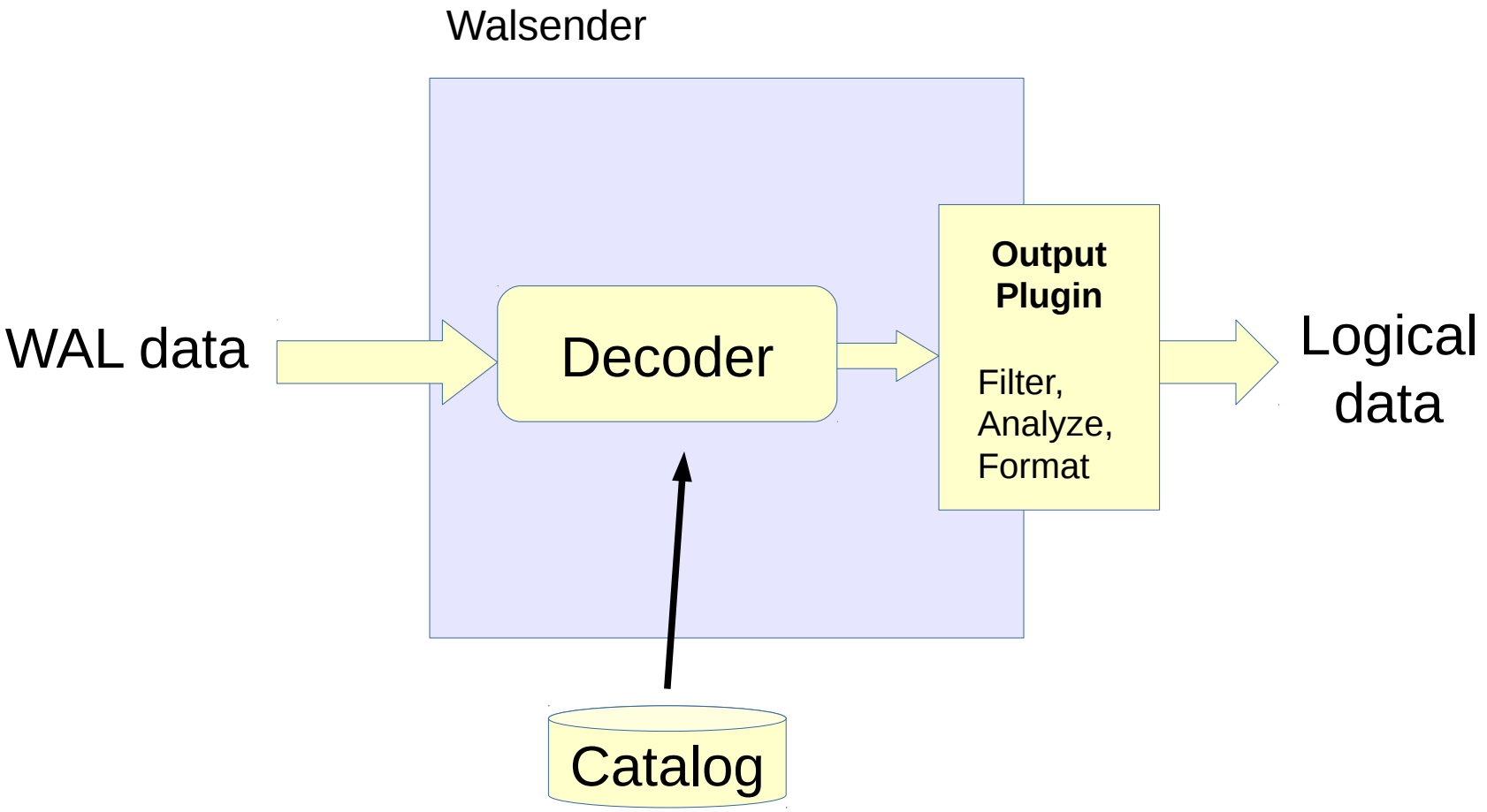
Logical Data

Table info

column values….

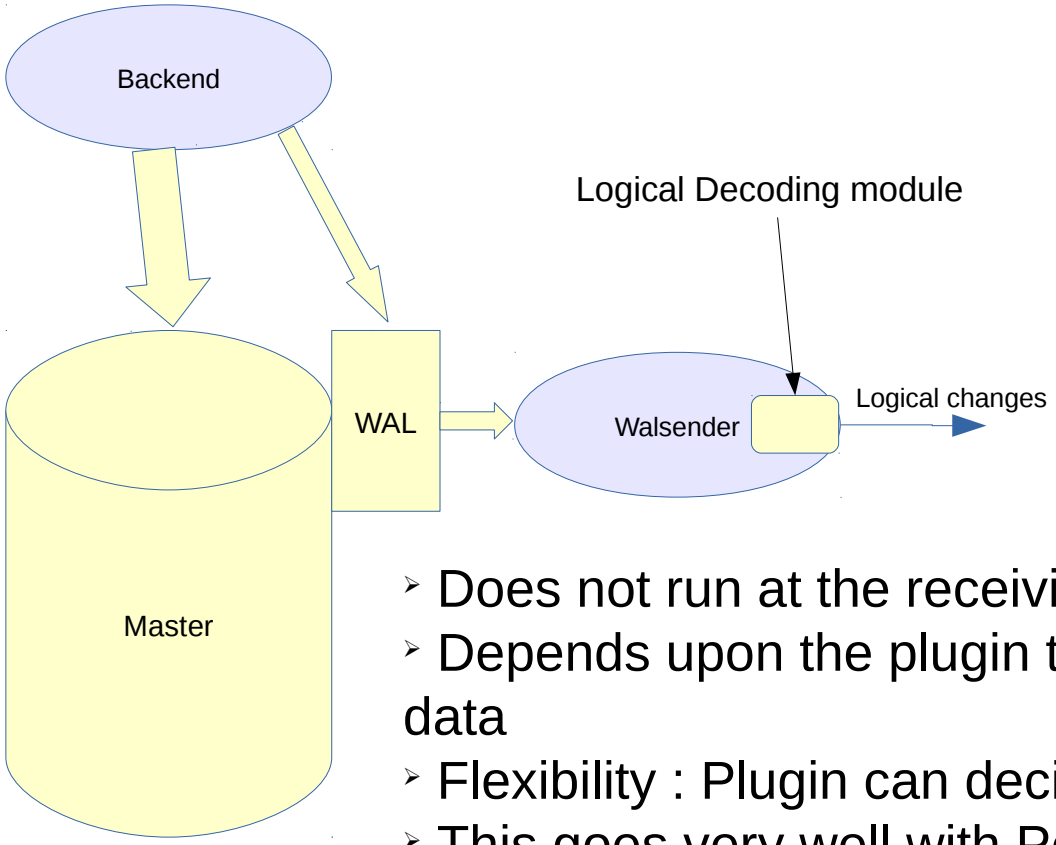
© 2016 EnterpriseDB Corporation. All rights reserved.

Logical decoding : Output plugin



© 2016 EnterpriseDB Corporation. All rights reserved.

Logical decoding module

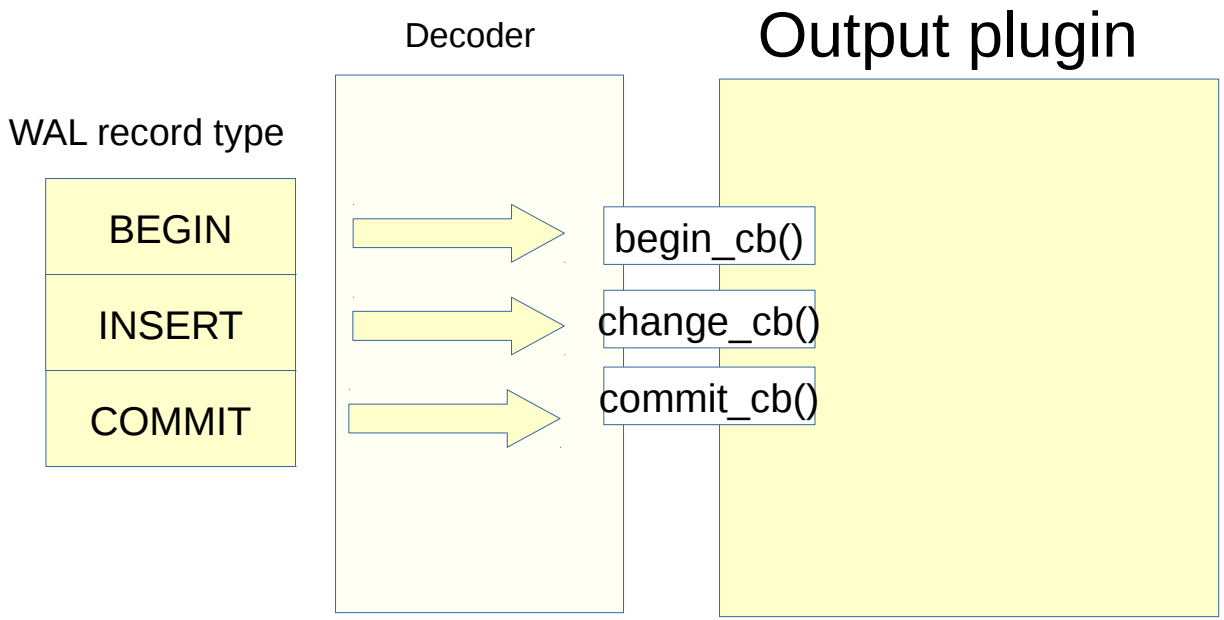


- Does not run at the receiving end.
- Depends upon the plugin to finally emit logical data
- Flexibility : Plugin can decide what to do with it.
- This goes very well with PostgreSQL extensibility.

© 2016 EnterpriseDB Corporation. All rights reserved.

Output Plugin

- Shared library written in C
- Has to set callback functions for specific events
- Sample plugin provided in contrib/test_decoding
- Plugin specified while initializing replication protocol



© 2016 EnterpriseDB Corporation. All rights reserved.

Tools for receiving data

```
postgres=# SELECT * from
pg_create_logical_replication_slot('my_slot1', 'test_decoding');
 slot_name | xlog_position
-----------+---------------
 my_slot1  | 0/23DFD78
```

© 2016 EnterpriseDB Corporation. All rights reserved.

Tools for receiving data

```
postgres=# create table tab (id int);
postgres=# insert into tab values (23), (24);
postgres=# update tab set id = 100 where id = 23;
```

```
$ pg_recvlogical -d postgres --slot='my_slot1' --start -f -
BEGIN 1153
COMMIT 1153
BEGIN 1154
table public.tab: INSERT: id[integer]:23
table public.tab: INSERT: id[integer]:24
COMMIT 1154
BEGIN 1155
table public.tab: UPDATE: id[integer]:100
COMMIT 1155
```

© 2016 EnterpriseDB Corporation. All rights reserved.

Tools for receiving data : SQL Interface

```
postgres=# SELECT * FROM
pg_logical_slot_get_changes('my_slot1', NULL, NULL);
 location  | xid  |                    data
-----------+------+-----------------------------------------
 0/23FE0B0 | 1149 | BEGIN 1149
 0/2400C08 | 1149 | COMMIT 1149
 0/2400C08 | 1150 | BEGIN 1150
 0/2400C08 | 1150 | table public.tab: INSERT: id[integer]:23
 0/2400C08 | 1150 | table public.tab: INSERT: id[integer]:24
 0/2400C88 | 1150 | COMMIT 1150
 0/2400CC0 | 1151 | BEGIN 1151
 0/2400CC0 | 1151 | table public.tab: UPDATE: id[integer]:100
 0/2400D38 | 1151 | COMMIT 1151
```

© 2016 EnterpriseDB Corporation. All rights reserved.

Replication slots

```
postgres=# SELECT * from pg_create_logical_replication_slot('my_slot1', 'test_decoding');
 slot_name | xlog_position
-----------+---------------
 my_slot1  | 0/23DFD78
(1 row)
```

- Introduced in PG 9.4 for LD
- Mandatory for LD
- Can be useful for physical replication as well.
- Can be thought as a FILE pointer which advances at each read.

© 2016 EnterpriseDB Corporation. All rights reserved.

Need for Replication slots

- physical streaming replication standby relies on continuous archiving.
- Logical replication cannot rely on WAL archives
- So logical replication has to have the master sending the decoded WAL segments.
- WAL segments should survive receiver crash.
- With replication slot, Master does not delete the WALs even after receiver crashes.
- **pg_xlog may consume all the disk space if it is not consumed by receiver** !

© 2016 EnterpriseDB Corporation. All rights reserved.

EDB ENTERPRISEDB

Configuration to allow logical replication

Physical replication	Logical replication

archive <= wal_level <= hot_standby
 	
max_wal_senders = 2
 	
# Not mandatory 	
max_replication_slots = 2
 	
wal_keep_segments = 32
archive_mode = on
archive_command = 'cp %p /path_to/archive/%f'
 	
# Following are required in case of 	
# streaming replication 	
listen_addresses = '192.168.0.0'
# replication user pg_hba.conf entry for streaming replication 	
 	
#Optional
synchronous_commit = on
	wal_level = logical
	
	max_wal_senders = 2
	
	# Should be at least 1
	max_replication_slots = 2
	
	# wal_segments and archive_mode
	# are not necessary, thanks to
	# replication_slots
	
	# Following are required if streaming
	# replication protocol is used
	listen_addresses = '192.168.0.0'
	# replication user pg_hba.conf entry for streaming replication
	
	#Optional
	synchronous_commit = on

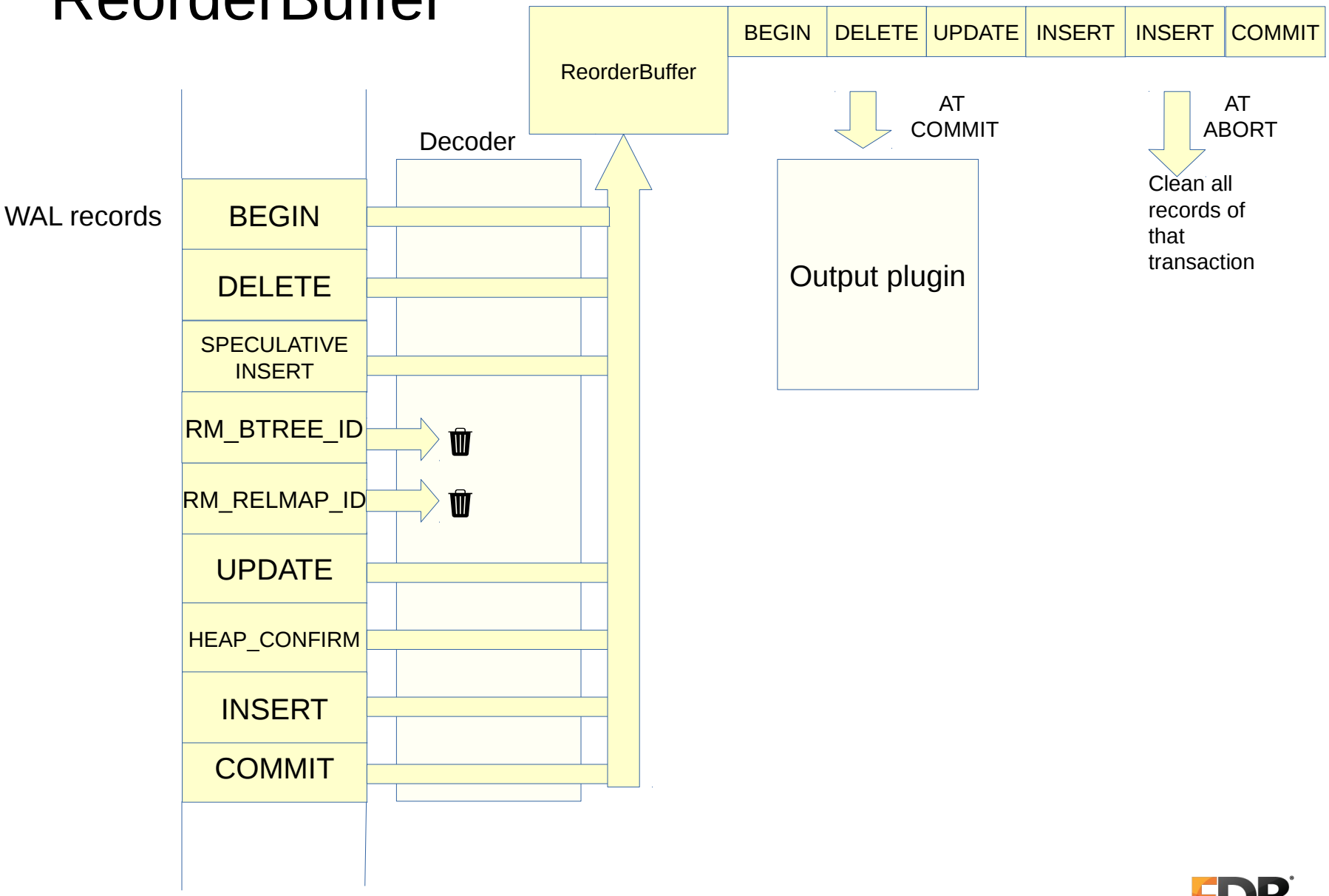
© 2016 EnterpriseDB Corporation. All rights reserved.

Properties

Physical replication	Logical replication
Replicates everything	Skips vacuum, index writes, etc
No temp tables on replica	Temp tables can be used on replica
DDL and DCL replicated	Only DMLs replicated !
Replicates whole instance	Replicates from single database
All transactions are replicated	Aborted transactions ignored.
Records are replicated on-the-fly.	Records are replicated only after a COMMIT record.

© 2016 EnterpriseDB Corporation. All rights reserved.

ReorderBuffer



© 2016 EnterpriseDB Corporation. All rights reserved.

Historical Snapshots

- Need catalog at the time WAL record inserted.
- Without that, can't deparse the row data.
- Hence, time travel snapshots required.
- Special snapshots used only for catalogs.

© 2016 EnterpriseDB Corporation. All rights reserved.

Plugins

- Wal2json
  - Outputs JSON object for each transaction
- test_decoding
  - Bundled with PostgreSQL.
  - Good starting point to build on.
- decoderbufs
    - Outputs in Protocol Buffer format.
      https://github.com/xstevens/decoderbufs
- pglogical_output
  - bundled with pglogical

© 2016 EnterpriseDB Corporation. All rights reserved.

REPLICA IDENTITY

(Used only for logical decoding).

ALTER TABLE table_name REPLICA IDENTITY
[NOTHING | USING INDEX | FULL | DEFAULT];

For OLD tuple , which column values will be included ?

DEFAULT : primary key columns, if any
NOTHING : No information about old tuple.
USING INDEX : index columns, if they are different in NEW tuple.
FULL : All.

table public.tab: UPDATE: **old-key**: id[integer]:222
new-tuple: id[integer]:333 v[character varying]:'abcd'

© 2016 EnterpriseDB Corporation. All rights reserved.

My own App

- Write output plugin and receiver.
- Receiver initiates streaming replication protocol connection :
  "dbname=<database_name> replication=database"
- Receiver creates slot and snapshot :
  CREATE_REPLICATION_SLOT slot_name LOGICAL <plugin>
  - Returns : Exported snapshot identifier, consistent point
- Use this snapshot to dump initial clone
- Start replication :
  START_REPLICATION SLOT slot_name LOGICAL ...

© 2016 EnterpriseDB Corporation. All rights reserved.

My own App (cont.)

- Walsender starts to stream WAL using CopyBoth to receiver
- Receiver receives data and applies changes, sends feedback.

© 2016 EnterpriseDB Corporation. All rights reserved.

Use cases : Logical Replication

- **Multi Master**
  - Bi-Directional replication
  - Not possible with physical replication
  - Ideal for geographically dispersed users
  - Requires conflict resolution
  - **Examples**
    - BDR
      http://2ndquadrant.com/en/resources/bdr
      - Currently uses modified PostgreSQL
    - EnterpriseDB xDB 6.0 Beta
      http://www.enterprisedb.com/docs/en/6.0/repguide/toc.html
      - Log-based MMR, new in 6.0

© 2016 EnterpriseDB Corporation. All rights reserved.

Use cases : Logical Replication

- **Single Master**
  - Uni-Directional replication
  - Can be used for :
    - Online upgrade
    - selective replication
    - Non-PostgreSQL replica
  - **Examples**
    - pglogical
      - Bypasses SQL layer during APPLY
      - Selective replication
      - Submitted as candidate for inclusion in PostgreSQL 9.6
      - Would be a game changer !

© 2016 EnterpriseDB Corporation. All rights reserved.

Use cases

- Auditing
- Real-time Data Analytics
- Event sourcing
  http://www.confluent.io/blog/bottled-water-real-time-integration-of-postgresql-and-kafka/
- Remote triggers
- Cache synchronization

© 2016 EnterpriseDB Corporation. All rights reserved.

Use cases

- In-memory database
- Table Replication for highly available cluster
- Incremental aggregation

© 2016 EnterpriseDB Corporation. All rights reserved.

Questions?

© 2016 EnterpriseDB Corporation. All rights reserved.

Thank You

© 2016 EnterpriseDB Corporation. All rights reserved.