THE FABULOUS DESTINY OF 000000020000000080000000BB

FOSDEM
2018-02-03

Patrick Francelle
Loxodata
WHO

Patrick Francelle

- PostgreSQL consultant and trainer
- First contact with PostgreSQL in 1999
- never stopped using it
- @pharrek
LOXODATA

Company built on 3 essential pillars

PostgreSQL  DevOps  Cloud
WHAT

The many possible lives of a WAL
SOON, I WILL BE A WAL FILE
WHAT AM I?

- transaction log
- REDO log
- Write Ahead Log
WAL?

- record data changes ASAP
- bring data consistency
- help restore data
- be the pillar of replication
SOME THEORY

ACID

- Atomicity
- Consistency
- Isolation
- Durability
Atomicity

Atomicity requires that each transaction be "all or nothing": if one part of the transaction fails, then the entire transaction fails, and the database state is left unchanged.
Consistency

The consistency property ensures that any transaction will bring the database from one valid state to another.
Isolation

The isolation property ensures that the concurrent execution of transactions results in a system state that would be obtained if transactions were executed sequentially, i.e., one after the other.
Durability

The durability property ensures that once a transaction has been committed, it will remain so, even in the event of power loss, crashes, or errors.

(All quotes from Wikipedia)
SOME THEORY
SOME THEORY
SOME THEORY
SOME THEORY
SOME THEORY
SOME THEORY

User

Server

Memory

Processes

WAL

Data files

DIPTY
SOME THEORY
MY VISION OF "TIME"

- not human time
- depends on activity
- "soon" : microseconds to years
MY LIFE WISHES

- reach the end of file
- travel!
- not be involved in a "disaster"
- not end up in "cryo chamber"
MY DESTINY

- no choice
- all my goals may be achievable
LIFE
"BIRTH"

- preallocated
- recycled
- allocated on demand
JOB

- record "events"
  - data changes
  - replication events
  - checkpoints
- writings in append-only mode
  - except for metadata
WORK TIME

- after switch from previous WAL file
- until switch to next WAL file
- continuation of other's work
THE SWITCH

- normal switch at EOF
- manual switch with *pg_wal_switch()*
- special PITR / promote
CRYONICS

- at the end of working period
- may be copied to another location
- this is called archiving
ARCHIVING

- external command in `archive_command`
- enable with `archive_mode = on`
- may retain WAL longer than expected
ARCHIVES

- many possible destinations
  - local or remote filesystem
  - tape band or permanent storage
- would likely never been used again
"DEATH"

- checkpointer process: deleted or recycled
- "death" may be delayed in some cases
- manual (human) action: ERROR
- (not) Schrödinger paradox
DEFROST

- only when recovering
- copied from archives
- fully read to REDO transactions
IDENTITY
MY NAME

00000002 00000008 000000BB

It's made of 3 parts, 8 digits each.

- TimeLine ID
  - starts at 1
- Logical file ID
  - starts at 0
- Physical file ID
  - from 00 to FF
- First of all WAL:
  0000000100000000000000000001
MY NAME

00000002 00000008 000000BB

• (unofficial) nickname: 8/BB
• the 0xBBth segment in the 0x8th logical file
• all my bytes have an address
• LSN: 8/BB3CB0D2 is my byte 3 977 426
Something horrible happened.
That's why the TLID is 2 and not 1
A part of the family was abandoned.
Some informations in file 00000002.history
I am tied to the PostgreSQL version
my internals may differ from one major version to another
SIZE

- default size: 16 MB
- divided into blocks, by default 8 kB each
- full size when allocated
WAL LEVEL

- 3 different levels available
- `wal_level` in configuration
- allows different life opportunities
- might change over time
LEVEL "MINIMAL"

- recovery, only from "crash"
- data consistency
- short life :'(
LEVEL "REPLICA"

- archiving
- physical replication (travel!)
- read-only queries on standby
- more informations stored
LEVEL "LOGICAL"

- logical decoding
- logical replication
- even more informations stored
MY PLACE

- pg_wal directory in $PGDATA
- or any directory symlinked as pg_wal
PASSPORT

- date of birth?
- date of issue/expiration?
- photo?

Not a human passport
DISASTER MANAGEMENT
• restore a backup
• replay work after that backup
• maybe stop at some point
• go back in production
AUTOMATIC RECOVERY

- after a brutal stop
- no need to restore a backup
- last checkpoint lookup
- transactions replay
DELIBERATE RECOVERY

- start from a physical backup
- write file `recovery.conf`
- `restore_command` to fetch WAL
- same as automatic recovery
- timeline change
PITR

- point-in-time recovery
- deliberate recovery
- specify end of recovery
- end of recovery action
TIMELINE CHANGE

- last WAL of recovery copied
- name differs by TLID
- content differs from recovery point
- subsequent WAL in old timeline abandoned
TRAVELLING
TRIP

- origin, source
  - primary, provider, publisher
- destinations
  - standby, subscriber
- transport method
  - via archives
  - streaming replication
CONCEPT

- continuously up-to-date clone of data
- copy data, then replay transactions
- who's the best at recording transactions?
PHYSICAL REPLICATION

- duplication of WAL file
- from one cluster to another
- streaming replication
LOGICAL REPLICATION

- decoded on publisher side
- information transformed
- sent to feed another WAL out there
- no travel
WAL SENDER

- gets replication connections
- runs replication protocol commands
- sends WAL content
WAL RECEIVER

- fetchs data
- permits REDO events
- sends feedback
PG_RECEIVEWAL

- special receiver process
- collects and stores (no REDO)
- streamed archive
REPLICATION SLOTS

- client dedicated resource
- stores replication status
- forbids deletion until replicated
TIME TO GET TO WORK
THANK YOU FOR YOUR ATTENTION