Our answer to Uber

Alexander Korotkov

Postgres Professional

April 7, 2017

Russian developers of PostgreSQL:

Alexander Korotkov, Teodor Sigaev, Oleg Bartunov

- Speakers at PGCon, PGConf: 20+ talks
- GSoC mentors
- PostgreSQL committers (1+1 in progress)
- Conference organizers
- 50+ years of expertship: development, audit, consulting
- Postgres Professional co-founders

PostgreSQL CORE

- Locale support
- PostgreSQL extendability: GiST(KNN), GIN, SP-GiST
- Full Text Search (FTS)
- NoSQL (hstore, jsonb)
- Indexed regexp search
- Create AM & Generic WAL
- Table engines (WIP)

Extensions

- intarray
- pg_trgm
- ltree
- hstore
- plantuner
- jsquery
- RUM
- imgsmlr

Disclaimer

- I'm NOT a MySQL expert. I didn't even touch MySQL since 2011...
- This talk express my own opinion, not PostgreSQL community position, not even Postgres Professional official position.
- Uber's guys knows better which database they should use.

What happened?

- Uber migrated from MySQL to PostgreSQL in 2012.
- Uber migrated from PostgreSQL to MySQL in 2016.
- PostgreSQL to MySQL migration made a log of buzz in PostgreSQL community.

Why did it happen?

- Uber migrated from MySQL to PostgreSQL for “a bunch of reasons, but one of the most important was availability of PostGIS” [1]
- Uber migrated from PostgreSQL to MySQL “some of the drawbacks they found with Postgres” [2]

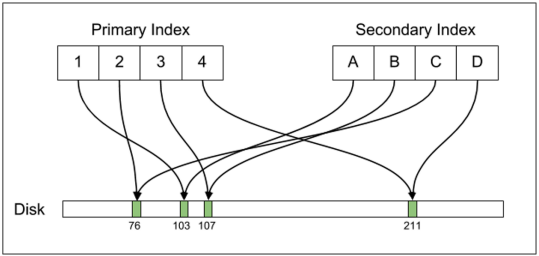
[1] https://www.yumpu.com/en/document/view/53683323/migrating-uber-from-mysql-to-postgresql
[2] https://eng.uber.com/mysql-migration/

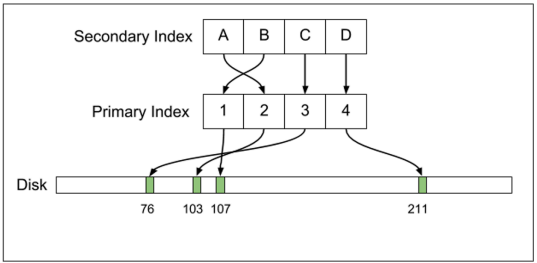
Uber's complaints to PostgreSQL

Uber claims following "PostgreSQL limitations":

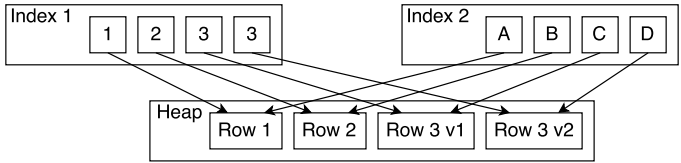
- Inefficient architecture for writes
- Inefficient data replication
- Issues with table corruption
- Poor replica MVCC support
- Difficulty upgrading to newer releases

PostgreSQL's vs. InnoDB's storage formats



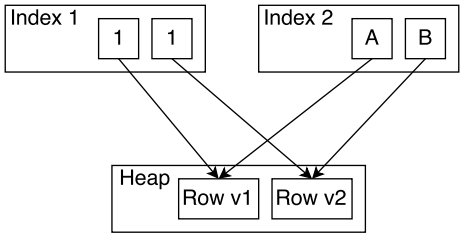


PostgreSQL storage format



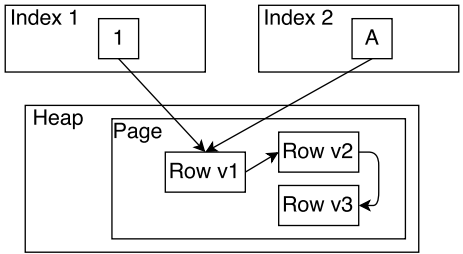
- Both primary and secondary indexes point to location (blkno, offset) of tuple (row version) in the heap.
- When tuple is moved to another location, all corresponding index tuples should be inserted to the indexes.
- Heap contains both live and dead tuples.
- VACUUM cleans up dead tuples and corresponding index tuples in a bulk manner.

Update in PostgreSQL



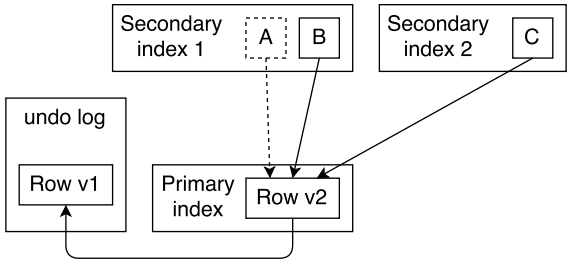
- New tuple is inserted to the heap, previous tuple is marked as deleted.
- Index tuples pointing to new tuple are inserted to all indexes.

Heap-Only-Tuple (HOT) PostgreSQL



- When no indexed columns are updated and new version of row can fit the same page, then HOT is used and only heap is updated.
- Microvacuum can be used to free required space in the page for HOT.

Update in MySQL



- Table rows are placed in the primary index itself. Updates are performed in-place. Old version of rows are placed to special segment (undo log).
- When secondary indexed column is updated, then new index tuple is inserted while previous index tuple is marked as deleted.

Updates: InnoDB in comparison with PostgreSQL

Pro:

- Update of few indexed columns is cheaper.
- Update, which don't touch indexed columns, doesn't depend on page free space in the page

Cons:

- Update of majority of indexed columns is more expensive.
- Secondary index scan is slower.
- Primary key update is disaster.

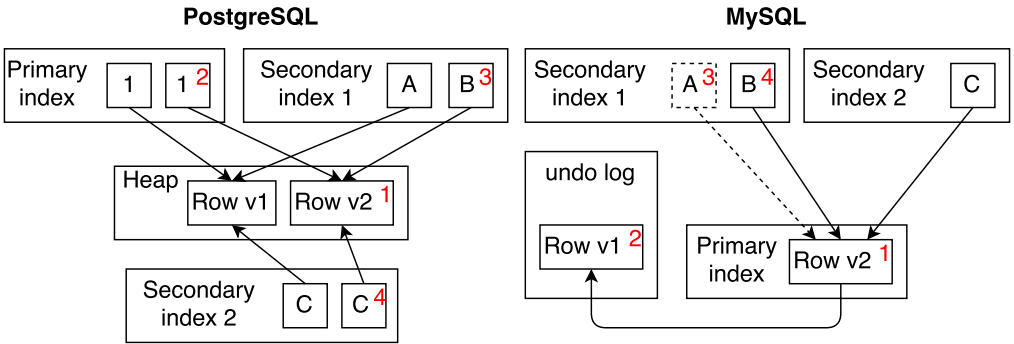
Uber example for write-amplification in PostgreSQL

```
CREATE TABLE users (id SERIAL PRIMARY KEY,
                    first TEXT,
                    last TEXT,
                    birth_year INTEGER);
CREATE INDEX ix_users_first_last ON users (first, last);
CREATE INDEX ix_users_birth_year ON users (birth_year);
```

```
UPDATE users SET birth_year = 1986 WHERE id = 1;
```

1. Write the new row tuple to the tablespace
2. Update the primary key index to add a record for the new tuple
3. Update the (first, last) index to add a record for the new tuple
4. Update the birth_year index to add a record for the new tuple
5. Previous actions are protected by WAL log.

Uber example for write-amplification: MySQL vs. PostgreSQL



Uber example for write-amplification: MySQL vs. PostgreSQL

PostgreSQL

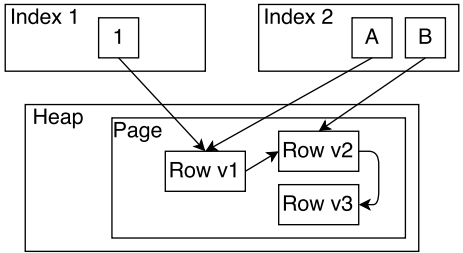
1. Write the new row tuple to the tablespace
2. Insert new tuple to primary key index
3. Insert new tuple to (first, last) index
4. Insert new tuple to birth_year index
5. Previous actions are protected by WAL log.

MySQL

1. Update row in-place
2. Write old version of row to the rollback segment
3. Insert new tuple to birth_year index
4. Mark old tuple of birth_year index as obsolete
5. Previous actions are protected by innodb log
6. Write update record to binary log [a]

[a] Assuming we have replication turned on

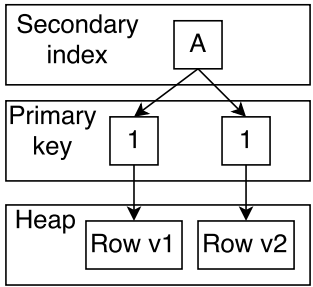
Pending patches: WARM (write-amplification reduction method)



- Behaves like HOT, but works also when some of index columns are updated.
- New index tuples are inserted only for updated index columns.

`https://www.postgresql.org/message-id/flat/20170110192442.ocws4pu5wjxcf45b%40alvherre.pgsql`

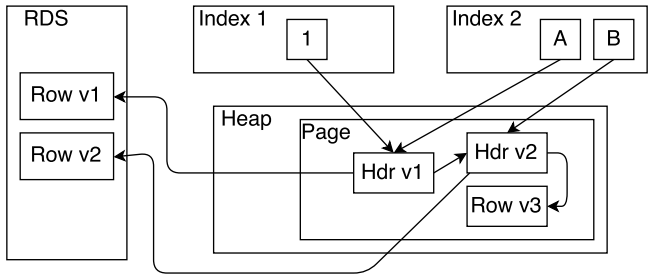
Pending patches: indirect indexes



- Indirect indexes are indexes which points to primary key value instead of pointer to heap.
- Indirect index is not updates until corresponding column is updated.

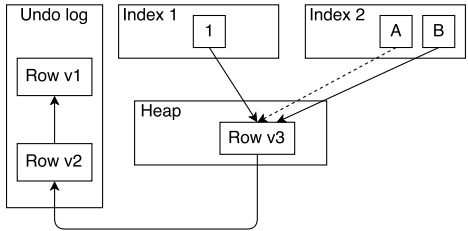
https://www.postgresql.org/message-id/20161018182843.xczrxsa2yd47pnru@alvherre.pgsql

Ideas: RDS (recently dead store)



- Recently dead tuples (deleted but visible for some transactions) are displaced into special storage: RDS.
- Heap tuple headers are left in the heap.

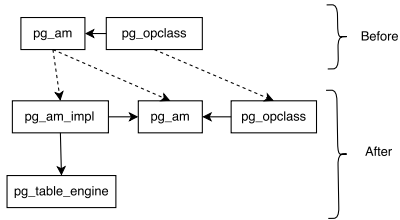
Idea: undo log



- Displace old version of rows to undo log.
- New index tuples are inserted only for updated index columns. Old index tuples are marked as expired.
- Move row to another page if new version doesn't fit the page.

https://www.postgresql.org/message-id/flat/CA%2BTgmoZS4_CvkaseW8dUcXwJuZmPhdcGBoE_GNZXWWn6xgKh9A%40mail.gmail.com

Idea: pluggable table engines



Owns

- Ways to scan and modify tables.
- Access methods implementations.

Shares

- Transactions, snapshots.
- WAL.

`https://www.pgcon.org/2016/schedule/events/920.en.html`

Types of replication

- Statement-level – stream writing queries to the slave.
- Row-level – stream updated rows to the slave.
- Block-level – stream blocks and/or block deltas to the slave.

Replication types in PostgreSQL vs. MySQL

Replication Type	MySQL	PostgreSQL
Statement-level	builtin	pgPool-II
Row-level	builtin	pgLogical Londiste Slony ...
Block-level	N/A	builtin

Uber's replication comparison

- Uber compares MySQL replication versus PostgreSQL replication.
- Actually, Uber compares MySQL row-level replication versus PostgreSQL block-level replication.
- That happened because that time PostgreSQL had builtin block-level replication, but didn't have builtin row-level replication. Simultaneously, MySQL had builtin row-level replication, but didn't have builtin block-level replication.

Uber’s complaints to PostgreSQL block-level replication

- Replication stream transfers all the changes at block-level including “write-amplification”. Thus, it requires very high-bandwidth channel. In turn, that makes geo-distributed replication harder.
- There are MVCC limitations for read-only requires on replica. Apply of VACUUM changes conflicts with read-only queries which could see the data VACUUM is going to delete.

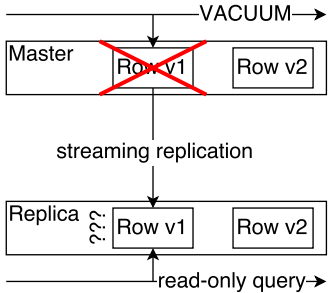
Is row-level replication superior over block-level replication?

Alibaba works on adding block-level replication to InnoDB. Zhai Weixiang, database developer from Alibaba considers following advantages of block-level replication: [3]

- Better performance: higher throughput and lower response time
  - Write less data (turn off binary log and gtid), and only one fsync to make transaction durable
  - Less recovery time
- Replication
  - Less replication latency
  - Ensure data consistency (most important for some sensitive clients)

[3] https://www.percona.com/live/data-performance-conference-2016/sessions/physical-replication-based-innodb

Replica read-only query MVCC conflict with VACUUM

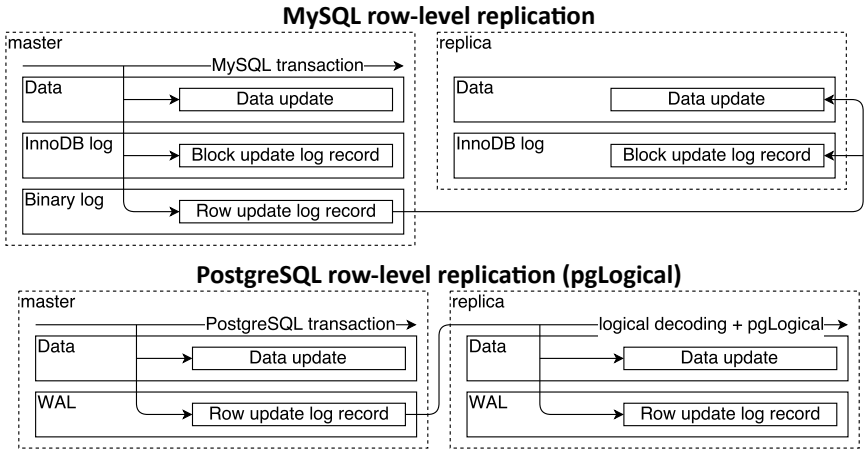


Possible options:

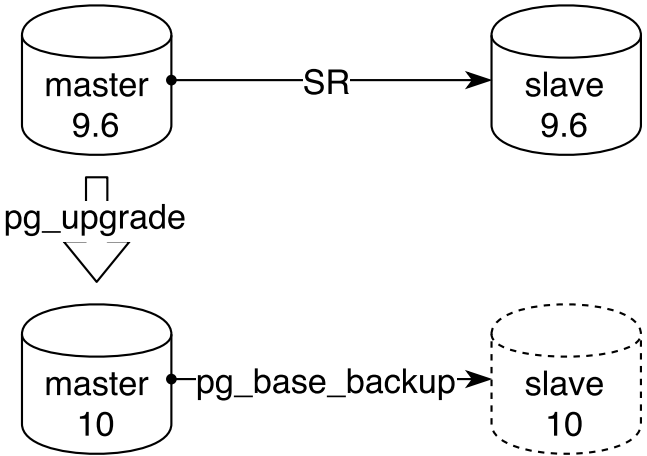
- Delay the replication,
- Cancel read-only query on replica,
- Provide a feedback to master about row versions which could be demanded.

Undo log would do better, we wouldn't have to choose...

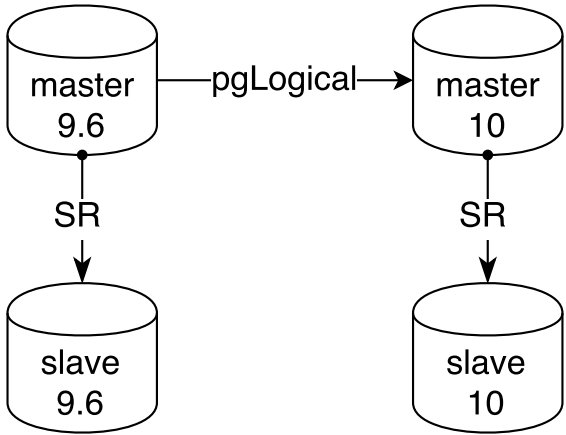
More about replication and write-amplification



Major version upgrade with pg_upgrade



Major version upgrade with pgLogical



https://www.depesz.com/2016/11/08/major-version-upgrading-with-minimal-downtime/

Other Uber notes

- PostgreSQL 9.2 had data corruption bug. *It was fixed long time ago. Since that time PostgreSQL automated tests system was significantly improved to evade such bugs in future.*
- pread is faster than seek + read. *Thats really gives 1.5% acceleration on read-only benchmark.* [4]
- PostgreSQL advises to setup relatively small shared_buffers and rely on OS cache, while "InnoDB storage engine implements its own LRU in something it calls the InnoDB buffer pool". *PostgreSQL also implements its own LRU in something it calls the shared buffers. And you can setup any shared buffers size.*
- PostgreSQL uses multiprocess model. So, connection is more expensive since unless you use pgBouncer or other external connection pool.

[4] https://www.postgresql.org/message-id/flat/a86bd200-ebbe-d829-e3ca-0c4474b2fcb7%40ohmu.fi

Postgres PROFESSIONAL

Thank you for attention!