The Silent Killer: How Undetected Database Performance Issues Can Cripple Your Apps & Business

A DEEP DIVE INTO POSTGRES TRANSACTION ID WRAPAROUND

Ankit Arora
Staff Cloud Platform Engineer@Gemini
18th May, 2024

DevOpsInside.com

$whoami

ankitarora

- STAFF CLOUD PLATFORM ENGINEER@GEMINI
- EX-ZETA, WINGIFY(VWO), ZOPPER
- LOVES MUSIC AND MEMES.

@DEVOPSINSIDE

ANKITARORA-DEVOPSINSIDE

A short story about the INC.

- Postgres Instances - (Primary + Secondary)
- DB size - 3TBs
- Number of DBs on Instance - 4 (Big mistake)
- Downtime - 6 Hours
- Data Loss?
- Application/Business Impact - 5% only.
- About Application - Internal Service for Customer support team.

When I got to know about some issue with DB.

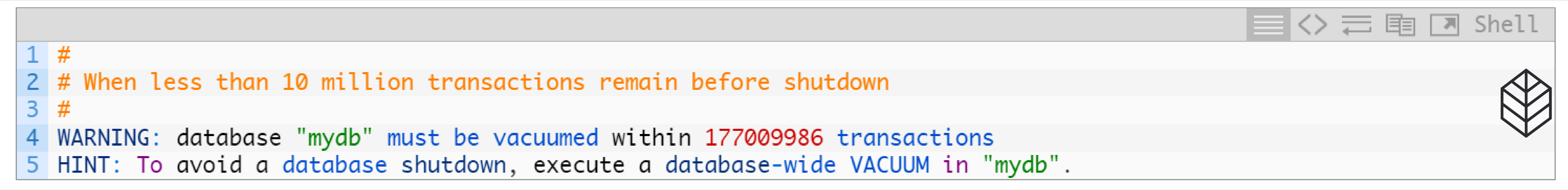
What all we tried to make it up?

- Did we try stopping and starting the DB again? - Yes
- Promoted the secondary to primary.
- Stopped replication.
- Tweaked and tuned DB configs and Params.
- Upgraded the Machine size.
- Started Vacumming again.

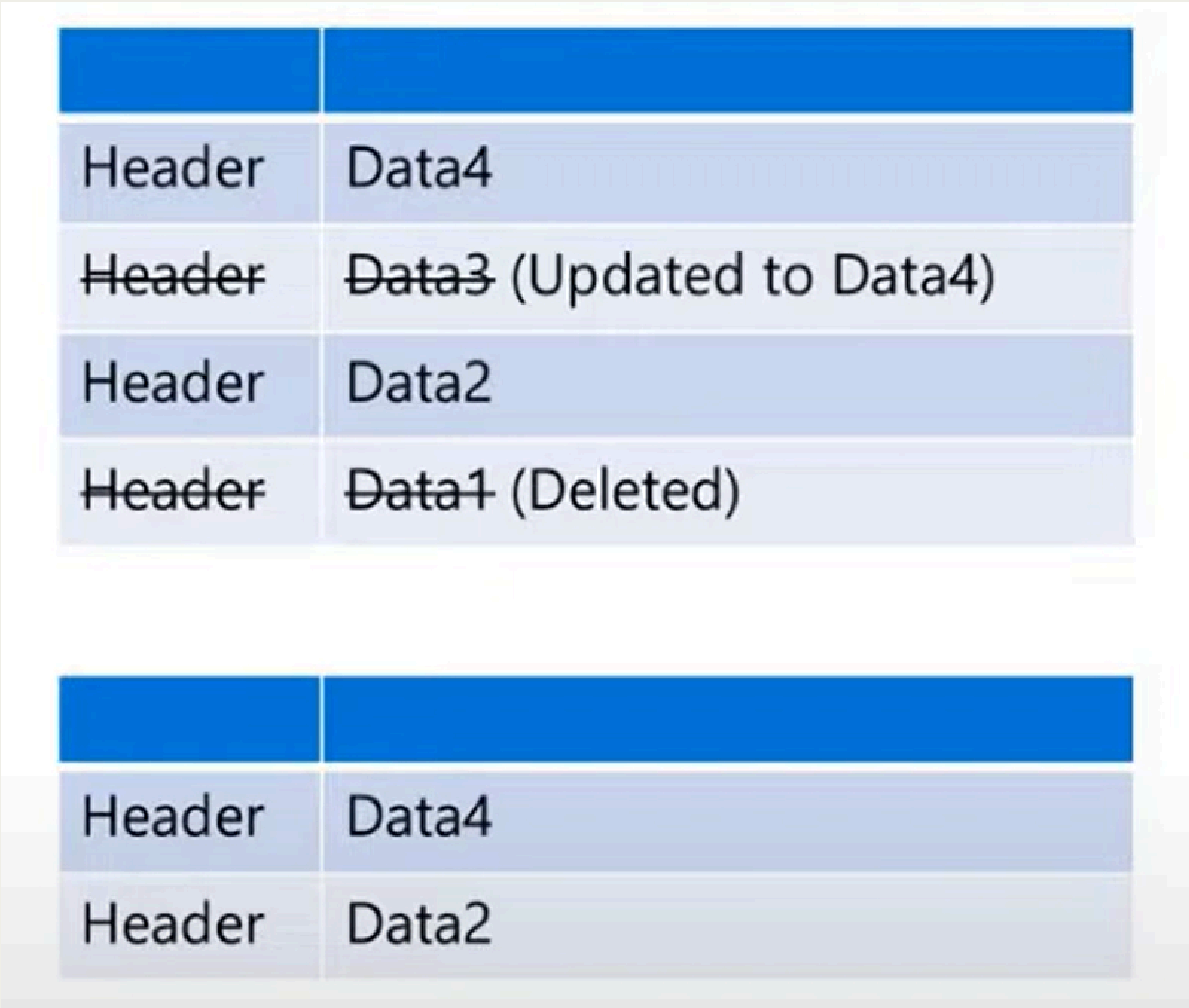
Me after realizing this is not what I was thinking

What is the Transaction ID Wraparound Issue?

- XID is a counter to assign unique IDs to transactions.
- XID maintains data consistency and isolation.
- XID utilization reaches 100% and goes beyond 2 billion transactions.
- XIDs are 32-bit integers.
- Shutdowns the DB in order to protect the data.



```
#
# When less than 10 million transactions remain before shutdown
#
WARNING: database "mydb" must be vacuumed within 177009986 transactions
HINT: To avoid a database shutdown, execute a database-wide VACUUM in "mydb".
```



Header	Data4
~~Header~~	~~Data3~~ (Updated to Data4)
Header	Data2
~~Header~~	~~Data1~~ (Deleted)

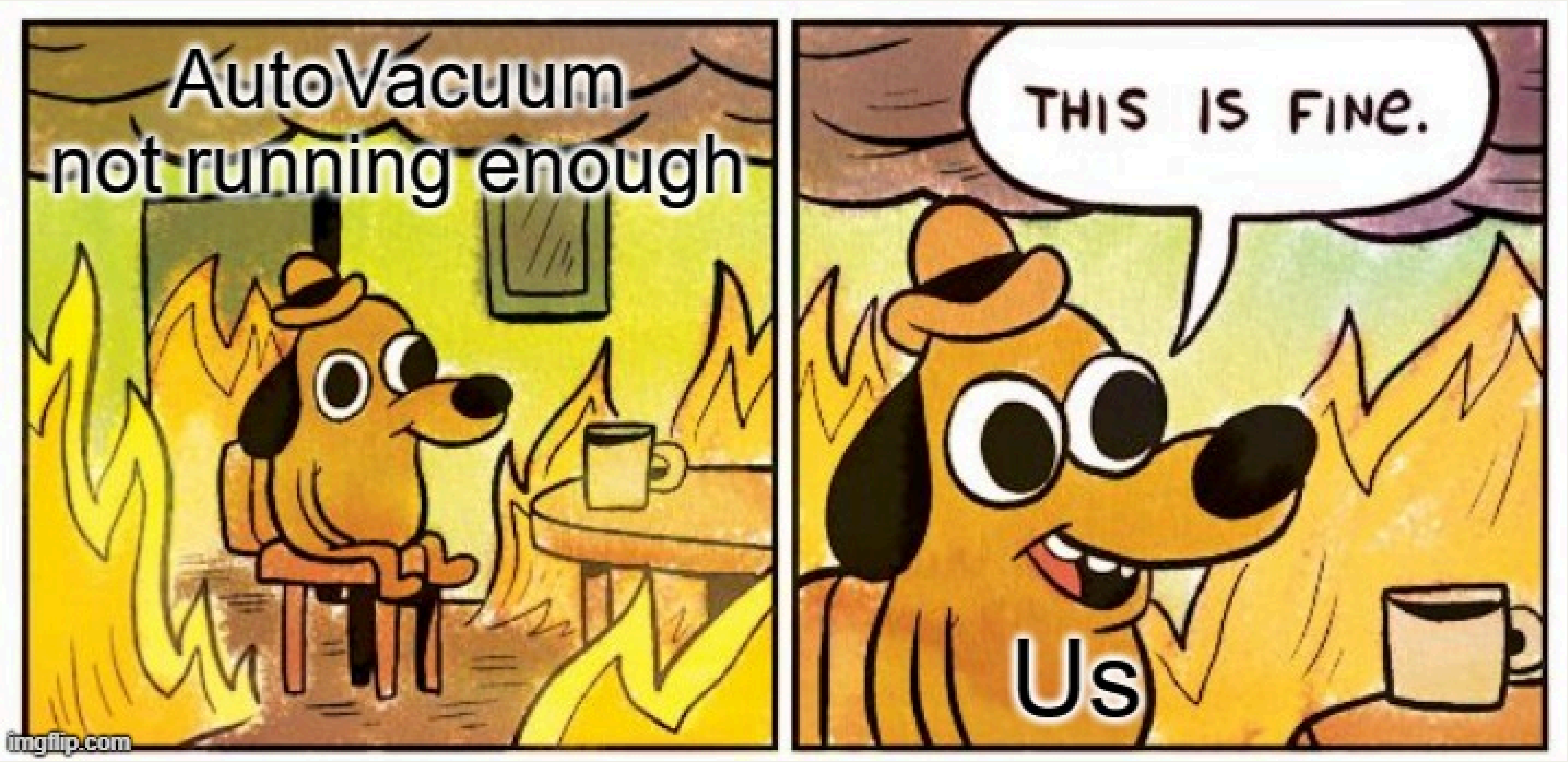
Header	Data4
Header	Data2

Reasons behind XID Wraparound?

Combination of one or more of the following circumstances:

- Autovacuum is turned off or running slow or not running enough.
- Long-lived transactions
- Database logical dumps (on a REPLICA using streaming replication)
- Many session connections with locks extending across large swaths of the data cluster.
- Intense DML operations(INSERT, UPDATE, DELETE) forcing the cancellation of autovacuum worker processes.

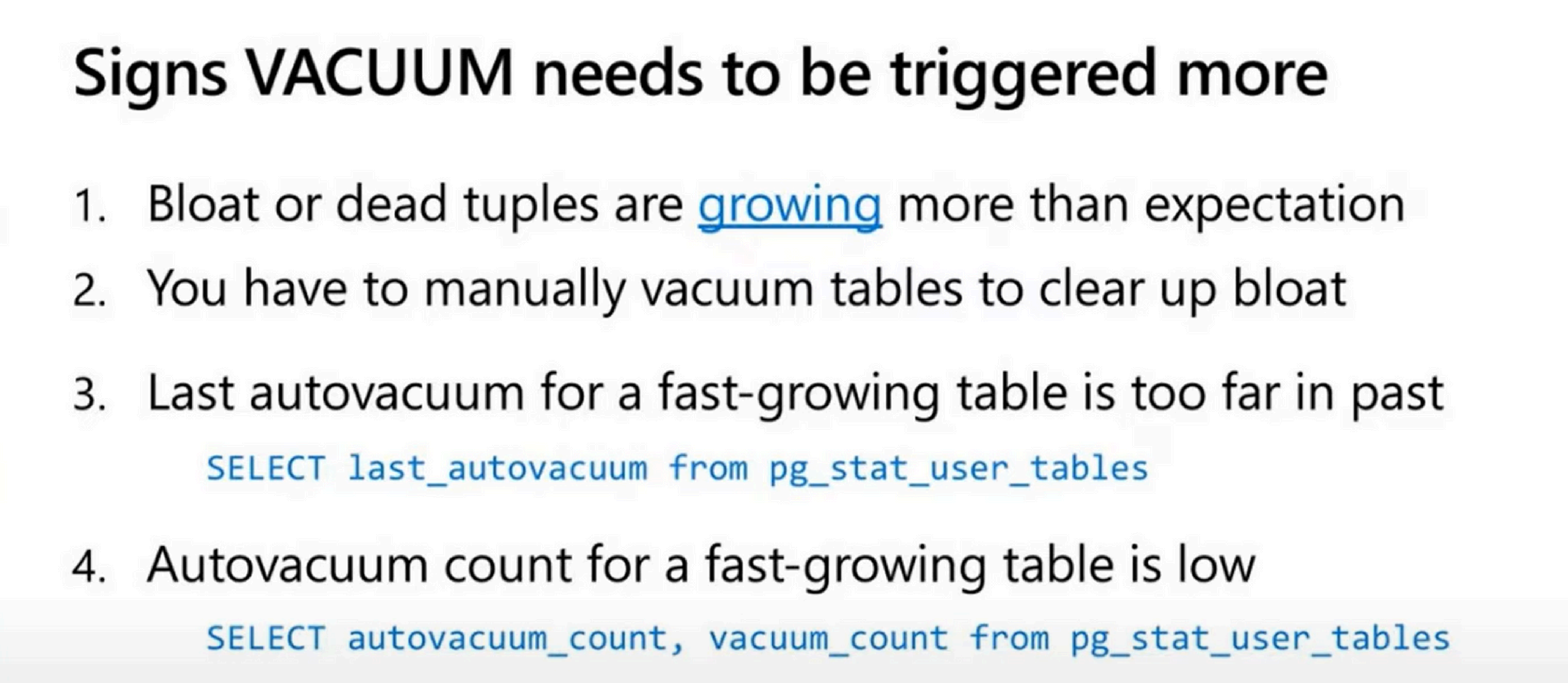
Culprit in our case?



Culprit in our case?

Combination of one or more of the following:

- Autovacuum was not running enough and not completing properly.
- Size of 1 table in 1DB was more than 1TB.
- Some signs we missed:



Signs VACUUM needs to be triggered more

1. Bloat or dead tuples are growing more than expectation
2. You have to manually vacuum tables to clear up bloat
3. Last autovacuum for a fast-growing table is too far in past

```
SELECT last_autovacuum from pg_stat_user_tables
```

4. Autovacuum count for a fast-growing table is low

```
SELECT autovacuum_count, vacuum_count from pg_stat_user_tables
```

After getting nothing about the issue on Google.

Me inside, trying to understand how to fix it.

Teams asking me to fix the issue asap.

Me after, the CTO joined the war room and helped me out.

Recovering from Disaster

- Took the latest backup dumps and restored them in new VMs.
- Upgraded the server to handle more load.
- Ran the vacuuming manually.
- Tuned the DBs.
- Didn’t start replication to avoid more load and transactions.

Lessons learned:

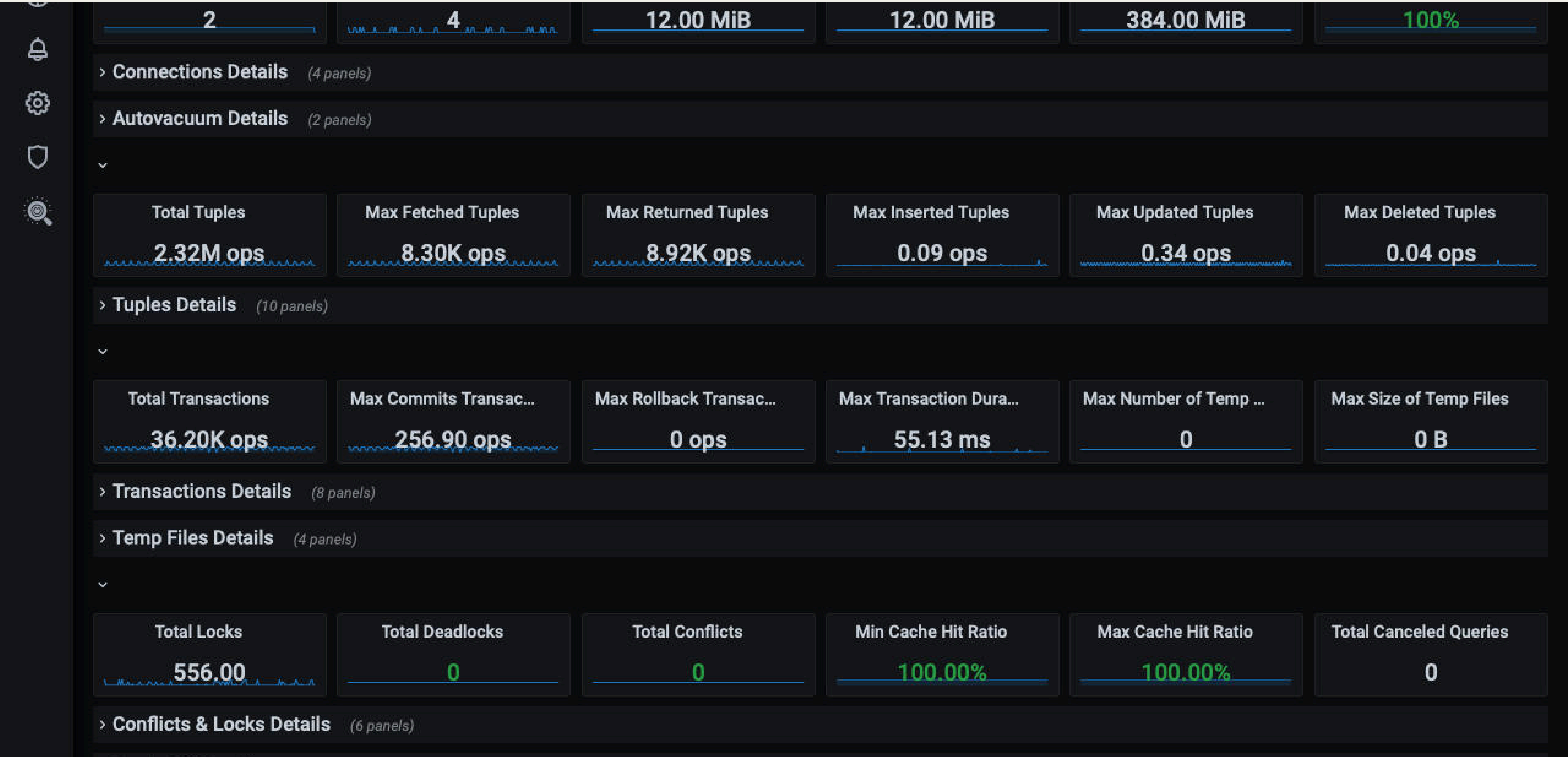
- Always monitor the performance of the DB as well, monitoring just resources(CPU, Memory, Network) of instance is not enough. (How?)
- Never put more than 1 DB in a single instance. If anything goes wrong, all of them will be impacted.
- Always keep an eye on DB logs. Try to understand what DB is trying to warn you about.
- Scale gives you healthy challenges but try to capture them before any hand. (More responsibility comes with more scale)
- Blessing in Disguise.

WHAT DO WE NEED NOW?
38
DB PERF MONITORING & ALERTING
imgflip.com

Moving Forward: Future-Proofing Strategies

- Started Monitoring the Performance of DB using PMM.
- Added alerts.
- Server with better config.
- Change the following params in the DB:
    - autovacuum_freeze_max_age = 500000000
    - autovacuum_max_workers = 6
    - autovacuum_naptime = '15s'
    - autovacuum_vacuum_cost_delay = 0
    - maintenance_work_mem = '5GB'
    - vacuum_freeze_min_age = 10000000

Percona Monitoring and Management Tool



2 | 4 | 12.00 MiB | 12.00 MiB | 384.00 MiB | 100%

› Connections Details (4 panels)

› Autovacuum Details (2 panels)

Total Tuples	Max Fetched Tuples	Max Returned Tuples	Max Inserted Tuples	Max Updated Tuples	Max Deleted Tuples
2.32M ops	8.30K ops	8.92K ops	0.09 ops	0.34 ops	0.04 ops

› Tuples Details (10 panels)

Total Transactions	Max Commits Transac...	Max Rollback Transac...	Max Transaction Dura...	Max Number of Temp ...	Max Size of Temp Files
36.20K ops	256.90 ops	0 ops	55.13 ms	0	0 B

› Transactions Details (8 panels)

› Temp Files Details (4 panels)

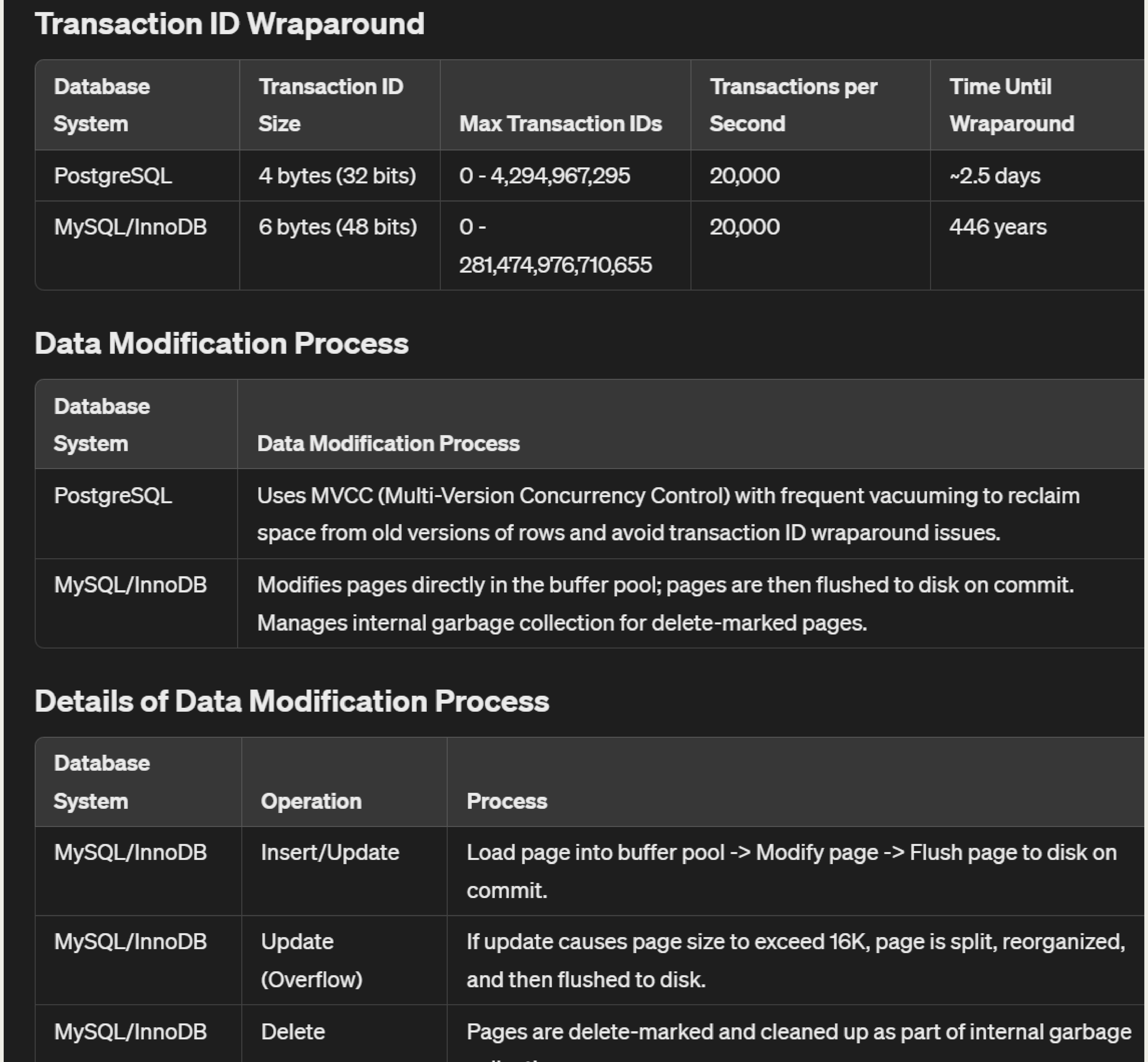
Total Locks	Total Deadlocks	Total Conflicts	Min Cache Hit Ratio	Max Cache Hit Ratio	Total Canceled Queries
556.00	0	0	100.00%	100.00%	0

› Conflicts & Locks Details (6 panels)

Me after fixing the Outage.

Bonus Slide:

Why Wraparound happens in PG not in MySQL?



Transaction ID Wraparound

Database System	Transaction ID Size	Max Transaction IDs	Transactions per Second	Time Until Wraparound
PostgreSQL	4 bytes (32 bits)	0 - 4,294,967,295	20,000	~2.5 days
MySQL/InnoDB	6 bytes (48 bits)	0 - 281,474,976,710,655	20,000	446 years

Data Modification Process

Database System	Data Modification Process
PostgreSQL	Uses MVCC (Multi-Version Concurrency Control) with frequent vacuuming to reclaim space from old versions of rows and avoid transaction ID wraparound issues.
MySQL/InnoDB	Modifies pages directly in the buffer pool; pages are then flushed to disk on commit. Manages internal garbage collection for delete-marked pages.

Details of Data Modification Process

Database System	Operation	Process
MySQL/InnoDB	Insert/Update	Load page into buffer pool -> Modify page -> Flush page to disk on commit.
MySQL/InnoDB	Update (Overflow)	If update causes page size to exceed 16K, page is split, reorganized, and then flushed to disk.
MySQL/InnoDB	Delete	Pages are delete-marked and cleaned up as part of internal garbage

REFERENCES:

- https://www.percona.com/blog/overcoming-vacuum-wraparound/
- https://mailchimp.com/what-we-learned-from-the-recent-mandrill-outage/
- https://www.tritondatacenter.com/blog/manta-postmortem-7-27-2015
- https://forums.percona.com/t/vacuum-why-transaction-wraparound-only-happens-in-postgresql-but-not-mysql-and-the-relation-of-dead-tuples-btree/24893
- https://twitter.com/samokhvalov/status/1722585894430105822
- https://docs.percona.com/percona-monitoring-and-management/details/dashboards/dashboard-postgresql-instances-overview.html

Questions?

Thank you!

DevOpsInside.com

@DEVOPSINSIDE

ANKITARORA-DEVOPSINSIDE