

Quinn Weaver <quinn@pgexperts.com>

POSTGRES SQL LOCKING ISSUES A TALK FOR DEVS AND DBAS

LICENSE

This work is licensed under the Creative Commons Attribution 4.0 International License.
To view a copy of this license, visit <http://creativecommons.org/licenses/by/4.0/>

<https://fairpath.com/locks.pdf>

ABOUT THIS TALK

- ▶ Beginner/intermediate.
- ▶ For developers and DBAs.
- ▶ 1. A little theory.
- ▶ 2. Lots of tips for common problems.

LOCKS 101

- ▶ PostgreSQL has its own locking system.
 - ▶ PostgreSQL locks
 - !=
filesystem-level locks
 - != `max_standby_*_delay`

TYPES OF LOCKS

- ▶ PostgreSQL has many exotic lock types:
 - ▶ virtualxid locks
 - ▶ extend locks
 - ▶ page locks...
- ▶ You seldom need to think about these.
- ▶ This talk covers table locks/row locks.

WHAT CAUSES LOCKS?

- ▶ Implicitly: DML statements, DDL statements, and (auto)vacuum.
- ▶ Explicitly: 'BEGIN; LOCK TABLE ... IN MODE ...'
- ▶ Can lock the whole table or just certain row(s).

LOCK MODES

- ▶ PostgreSQL has a bunch of "lock modes"
 - ▶ = Precedence levels
 - ▶ With confusing, historical names like "SHARE UPDATE EXCLUSIVE MODE"
- ▶ You don't need to memorize these; just refer to the docs:
<https://www.postgresql.org/docs/current/static/explicit-locking.html>

<https://fairpath.com/locks.pdf>

LOCK MODES

- ▶ PostgreSQL will always use the least annoying lock mode possible, for instance:
 - ▶ If you're running (most) DDL, it will still allow writes.
 - ▶ Ditto if autovacuum is running.
 - ▶ If you're writing row(s), PostgreSQL will still allow queries to read those rows.
 - ▶ If you're writing row(s), PostgreSQL will still allow queries to write other rows in the table.

<https://fairpath.com/locks.pdf>

ON TO THE EXAMPLES!

- ▶ So much for theory.
- ▶ On to practice!
- ▶ First, any questions?
 - ▶ (Clarification only, please; save discussion for afterward.)

COMMON LOCK PROBLEMS: VACUUM/DDDL ISSUES

- ▶ (auto)vacuum is preventing you from running DDL.
 - ▶ ... or vice versa!
- ▶ Anti-solution: disable autovacuum. Don't do this!
- ▶ One solution: set autovacuum to be more aggressive.
 - ▶ I have another talk for that!
<http://fairpath.com/vacuum-slides.pdf>
- ▶ Another solution: split up large tables: partition, or archive.

<https://fairpath.com/locks.pdf>

COMMON LOCK PROBLEMS: VACUUM/DDDL ISSUES

- ▶ Special case: autovacuum freeze (to prevent xid wraparound)
 - ▶ Can't be killed; will just start over.
 - ▶ Avoidable by adjusting vacuum parameters and/or doing explicit VACUUM FREEZE off-peak.
 - ▶ XIDs are 32 bits; easy to get near wraparound.
 - ▶ If you disable it, your DB will shut down. Don't!

COMMON LOCK PROBLEMS: 'SELECT ... FOR UPDATE' OVERUSE

- ▶ `SELECT ... FOR UPDATE;`
 - ▶ Blocks writes to those rows.
 - ▶ Plus instances of these statements conflict with each other!
 - ▶ Good way to kill parallelism; easy to overuse.
 - ▶ Hint: if you are implementing a queue in PostgreSQL, don't.

COMMON LOCK PROBLEMS: 'IDLE IN TRANSACTION'

- ▶ 'idle in transaction' transactions...
 - ▶ ... that hold locks...
 - ▶ forreeeevvvverr

COMMON LOCK PROBLEMS: 'IDLE IN TRANSACTION'

```
sandbox=# select * from pg_stat_activity where state = 'idle in transaction';
-[ RECORD 1 ]-----+-----
datid          | 24688
datname        | sandbox
pid            | 64152
usesysid       | 10
username       | quinn
application_name | psql
client_addr    | NULL
client_hostname | NULL
client_port    | -1
backend_start  | 2018-09-05 23:23:01.869974-07
xact_start     | 2018-09-05 23:23:06.135994-07
query_start    | 2018-09-05 23:23:27.745046-07
state_change   | 2018-09-05 23:23:27.745286-07
wait_event_type | Client
wait_event     | ClientRead
state          | idle in transaction
backend_xid    | NULL
backend_xmin   | NULL
query         | RELEASE pg_psql_temporary_savepoint
backend_type   | client backend
```

COMMON LOCK PROBLEMS: 'IDLE IN TRANSACTION'

- ▶ Note: IIT transactions are a problem even without locks, because they mean your writes aren't visible!
- ▶ They also prevent (auto)vacuuming of relevant tuples.
- ▶ The fix is, find the offending app code and make sure it calls COMMIT.

COMMON LOCK PROBLEMS: 'IDLE IN TRANSACTION'

- ▶ A mitigation: implement an 'idle in transaction' killer.
 - ▶ That should get your devs' attention. ;)
 - ▶ In PostgreSQL 9.6 and up, in postgresql.conf:
`idle_in_transaction_session_timeout = 2min`
 - ▶ Pre-9.6, write a cron job.
- ▶ Less drastic: log long IIT transaction info in your monitoring/alerting system.

COMMON LOCK PROBLEMS: LONG DDL LOCKS

- ▶ The scenario:
 - ▶ ALTER TABLE my_table
ADD COLUMN my_column text NOT NULL DEFAULT 'blah';
 - ▶ Ditto with altering a column.
- ▶ The problem:
 - ▶ This takes an AccessExclusive lock and *rewrites the whole table*.
 - ▶ Yikes, cancel! (You did run that in a transaction, right?)
- ▶ Fixed in PostgreSQL 11 (thanks, Andrew Dunstan!)

<https://fairpath.com/locks.pdf>

COMMON LOCK PROBLEMS: LONG DDL LOCKS

- ▶ Workaround pre-PostgreSQL 11:
- ▶ Create the column without constraints.
- ▶ Backfill all existing values, probs in batches.
 - ▶ While using triggers to set the value for new rows!
- ▶ Then...

COMMON LOCK PROBLEMS: LONG DDL LOCKS

- ▶ At the end:

```
ALTER TABLE my_table  
ALTER COLUMN my_column SET DEFAULT 'blah';
```

This applies to future rows only (won't rewrite table).

- ▶ Caveat: adding a NOT NULL constraint *does* check the whole table. No way around this.

SO MUCH FOR COMMON PITFALLS!

- ▶ Next we'll examine more-general diagnostics.
- ▶ Any questions?
- ▶ Again, clarification only – save discussion for the end.

IDENTIFYING LOCK PROBLEMS

- ▶ How do you know locks might be the problem?
- ▶ The classic symptom:
 - ▶ High query latency...
 - ▶ ... but low resource utilization – especially:
 - ▶ *Low I/O utilization.*

IDENTIFYING LOCK PROBLEMS

- ▶ Two basic options:
 - ▶ "Now Mode": interrogate the system in real time.
 - ▶ "Later Mode": collect data and analyze it later.

"NOW MODE": PROS AND CONS

▶ Pros

- ▶ Useful in emergencies.

▶ Cons

- ▶ Time-consuming.
- ▶ Requires knowledge of system tables, and of the PostgreSQL locking model.
- ▶ Locks may be too ephemeral to observe by hand.

<https://fairpath.com/locks.pdf>

TOOLS FOR "NOW MODE"

- ▶ pg_locks and pg_stat_activity

```
SELECT * FROM pg_stat_activity WHERE wait_event_type =  
'Lock';
```

```
SELECT * FROM pg_stat_activity JOIN pg_locks USING (pid)  
WHERE NOT pg_locks.granted;
```

- ▶ Can also do fancier JOINS, e.g.,
<http://hacksoclock.blogspot.com/2016/01/blocked-by-rdsadmin.html>
- ▶ Hey, I said it was difficult!

<https://fairpath.com/locks.pdf>

"LATER MODE": PROS AND CONS

▶ Pros:

- ▶ Much easier.
- ▶ Much more data.
- ▶ Nice reports.
- ▶ Catches ephemeral locks.

▶ Cons:

- ▶ Takes more time.
- ▶ That's it!

ADVICE

- ▶ Just use Later Mode. It's almost always better.
- ▶ If you must use Now Mode, first turn on the logging needed for Later Mode! :)

TOOLS FOR "LATER MODE": PGBADGER

- ▶ The go-to tool.
- ▶ You feed it an hour of two of logs.
- ▶ It gives you a nice HTML report.
 - ▶ Including many other subsections besides locks
 - ▶ Because locks might not be the problem!

USING PGBADGER: CAVEATS

- ▶ Make sure you have enough space for a few GB of logs.
- ▶ If you're on RDS, make sure you're not maxed out on IOPS.
 - ▶ (in RDS, logs and data go on the same device).
- ▶ If you use rsyslogd, make sure *not* to synchronously write each log line.
 - ▶ `local0.* -/var/log/postgres.log`
 - ▶ The minus sign is what says "Don't flush."

<https://fairpath.com/locks.pdf>

USING PGBADGER: SETTINGS

▶ Recommended log settings (in postgresql.conf):

```
# logging_collector: requires a restart to change.  
# You need it turned on to produce logs in the csvlog format.  
logging_collector = on
```

```
# deadlock_timeout: 1000 ms is the default, and recommended;  
# lower values will log shorter locks,  
# but will hurt performance.  
deadlock_timeout = 1000
```

```
client_min_messages = 'notice'  
log_autovacuum_min_duration = 0  
log_checkpoints = on  
log_connections = on  
log_destination = 'csvlog,stderr'  
log_disconnections = on  
log_duration = off  
log_error_verbosity = 'default'  
log_filename = 'pgbadger-%Y-%m-%d-%H'  
log_lock_waits = on  
log_min_duration_statement = 0  
log_rotation_age = 1h  
log_rotation_size = 1GB  
log_statement = 'none'  
log_temp_files = 0
```

<https://fairpath.com/locks.pdf>

USING PGBADGER: RUNNING

```
pgbadger \  
  -x html \  
  -o my_output_filename.html \  
  --nocomment \  
  --top 50 \  
  --exclude-query '^(SET|DISCARD|BEGIN|COMMIT)' \  
pgbadger-*.csv
```

USING PGBADGER: OUTPUT

- ▶ Let's explore an examples report:
<http://pgbadger.darold.net/samplev7.html>

Thank you!

<https://fairpath.com/locks.pdf>

Questions?

Quinn Weaver

Consultant at
PostgreSQL Experts, Inc.
<https://pgexperts.com/>

<https://hacksoclock.blogspot.com/>

<https://fairpath.com/locks.pdf>