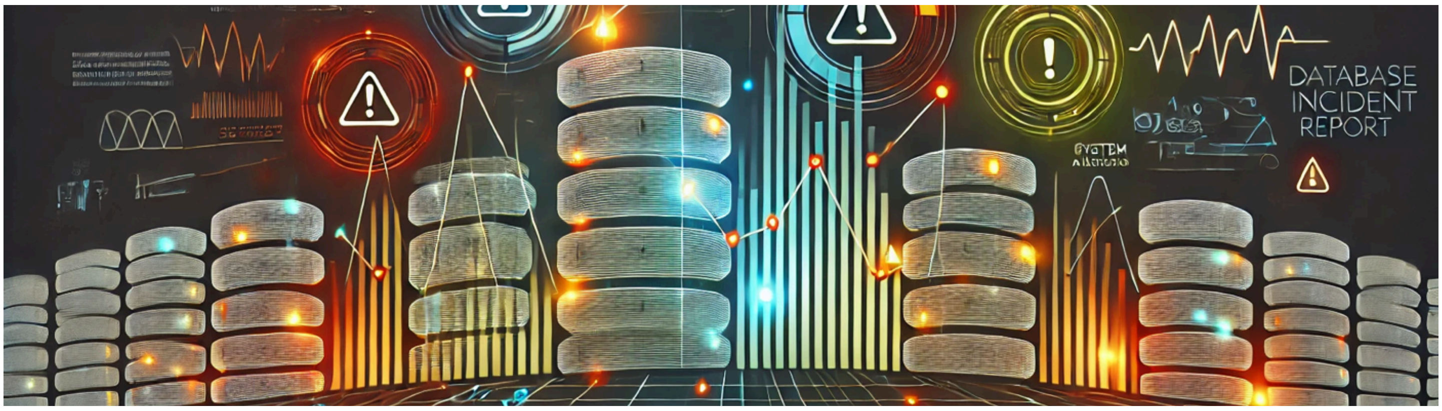
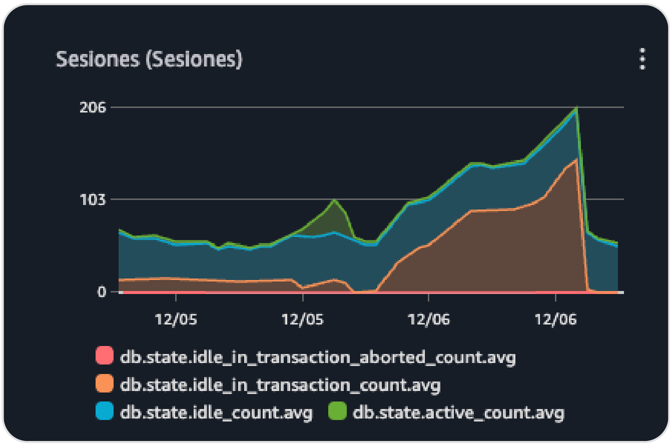


Idle in transaction Sessions

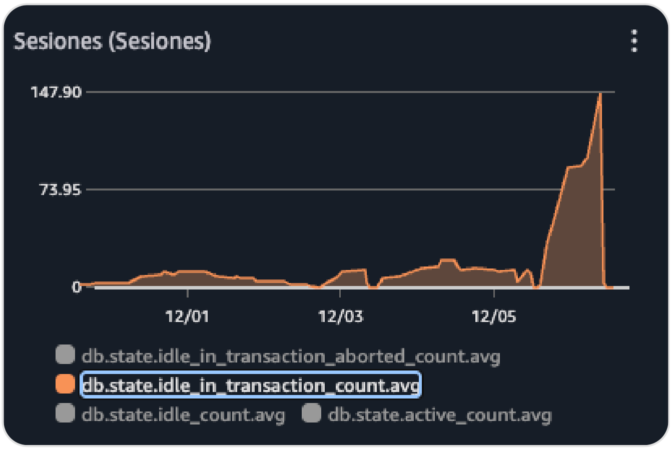
Author: Martín San Juan

Due to a rollout on December 12th, 2024, we started to see a increase on the idle in transaction sessions on the database as is shown in the image.



This behavior wasn't new, but the number of rolled-out subscription types and the lack of monitoring on these metrics kept it unnoticed until now.

This was caused for:



A code error

A wrong exception handling left a the transaction rollout outside the flow and the session remains active, but in idle state because the process that opened was finished.

A database configuration

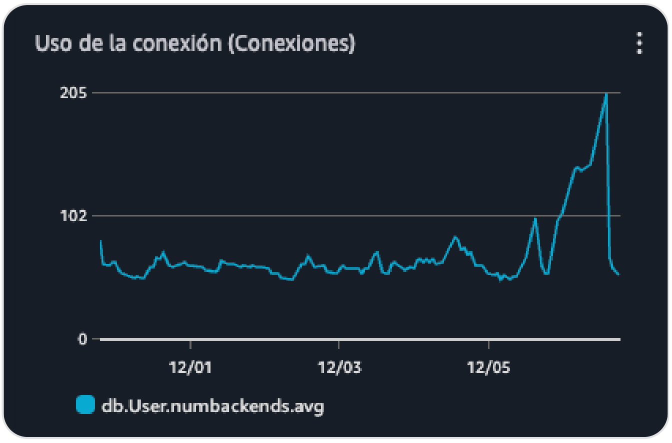
Postgres is allowed to close sessions if remains inactive for a period of time, by default, this configuration is set to one day, so we found sessions opened more than 20 hours ago in **idle in transaction** state.

Impact

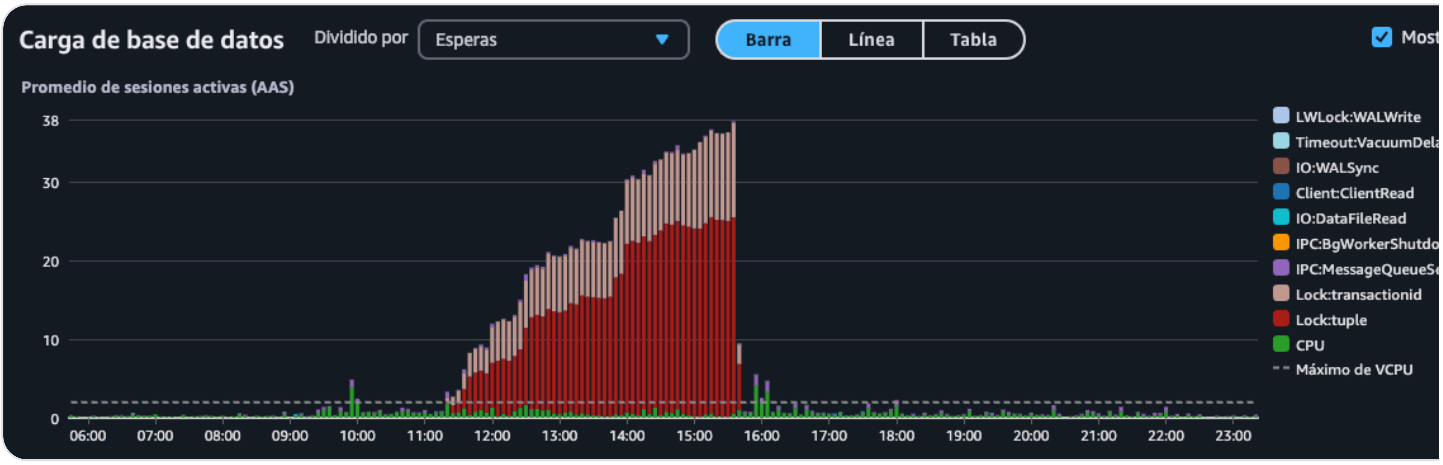
No impact was caused by this, but if left unmonitored, it could lead to several consequences, degrade certain tasks, or completely shut down some of them.

The increase of sessions in **idle in transaction** state implies the increase of connections in the database, and in the pool.

The GORM configuration on the subscriptions API allows only 10 connections in the pool. If all the connections are occupied, any new transactions will be blocked until those sessions are released.



This was the scenario that occurs on 📅 December 5th, 2024



The hypothesis was a lock generated by some concurrence problem on the same resource, but we couldn't prove it, but we rollbacked the deploy and the situation was normalized. The reason for that was that the deploy activate the migration for all the subscription types at once, and extend the problem, which at the moment was affecting but undetected in a small amount of cases.

We don't have logs to check the incident on 📅 November 28th, 2024 due to log retention), but considering that it was resolved by increasing the tasks on the subscriptions API from 3 to 20, and the number of sessions we had today was 145 (which means 145 open connections), reaching 200 open connections (10 per task), we can infer that it was the same scenario. A spike in traffic could generate blocks again if we continue keeping sessions open. However, given that the issue lies in actions triggered on subscription charges, and the number of sessions processed for subscriptions remains stable over the days, we reached a peak of sessions and freed them after a day without incurring further locks.

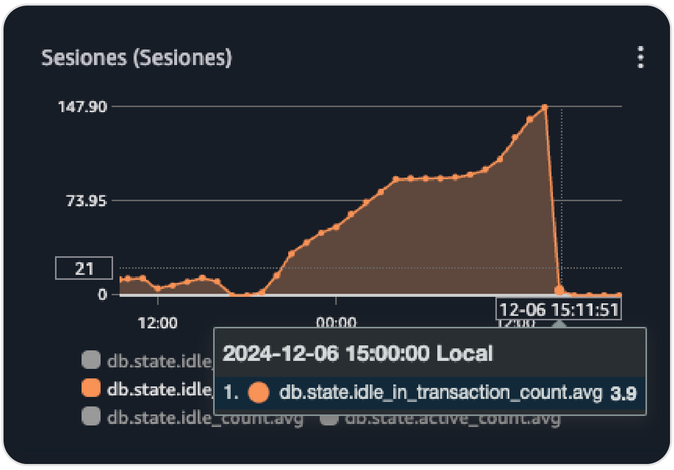
Actions Taken

Fix the issue | Type: Corrective

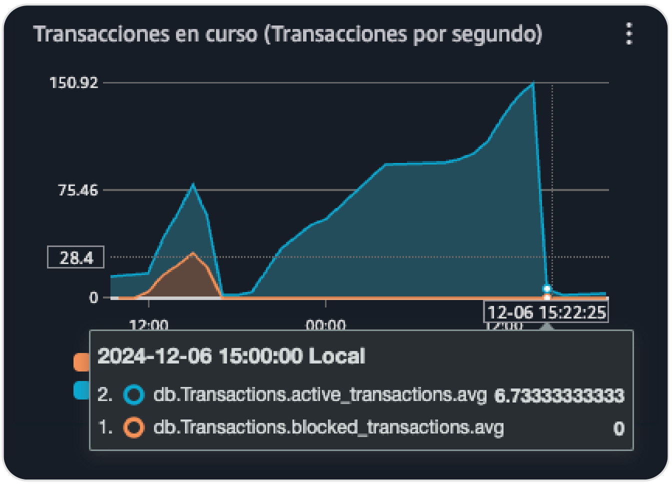
We fix the error in the PR rollbacking the transaction on errors.

Release dead sessions - Corrective

Once the fix was deployed, we manually close the connections in idle in transactions



2024-12-06 15:00:00 Local
1. db.state.idle_in_transaction_count.avg 3.9



2024-12-06 15:00:00 Local
2. db.Transactions.active_transactions.avg 6.73333333333
1. db.Transactions.blocked_transactions.avg 0

Database Configuration | Preventive

The change the default value of the `idle_in_transaction_session_timeout` from 1 day to 5 minutes in Postgres.

This will help us prevent lock events even if we encounter a similar error in the future because the sessions will be freed automatically five minutes after they start. This could make it harder to detect code problems like this one, but we will create alerts and monitors to check the number of idle-in-transaction sessions, even during smaller spikes.

This action will not completely prevent the issue in case of occurrences, but a much larger burst would be needed to replicate the same problem.

Datadog - Cloudwatch | Type: Preventive

We will work with the Platform team to create alerts of anomalies on sessions and connections, as well as database locks.