Closing client connection does not cancel query

Last updated by | Vitor Tomaz | Jun 8, 2022 at 5:34 AM PDT

Contents

Issue

If you disconnect the client connection, any active query might continue to run until it either times out or is ready to deliver the results to the application.

In a specific scenario, the customer is connecting through a JDBC driver to Azure SQL Database. In the application, they monitor how long each query is running. If it takes longer than expected, they kill the connection using the connection.close() command from JDBC.

They noticed that the connection.close() command does not come back with any success or failure message. The connection is closed but the SQL query keeps running at the database until it either finishes or times out. It continues to consume resources and is potentially blocking other workloads, which the customer had wanted to avoid by closing the connection.

This scenario includes queries that are still executing, or queries that have returned results but which haven't been fully retrieved by the application yet.

Investigation / Analysis

This is by design. Just closing a connection will not cancel a running or incomplete query execution. Cancelling a query and closing a connection are two separate operations; one does not imply the other.

If the client connection is closed on an active query execution, SQL Server will not immediately notice that the server-side session is now orphaned. SQL Server will usually only realize it later when it tries to write the results back to the client.

On the telemetry (MonLogin/ASC Troubleshooter), you might see one of the following errors when the disconnect is detected:

```
Error: 17900 - State 25
A network error occurred in the established connection; the connection has been closed.

Error: 7885
Network error 0x%lx occurred while sending data to the client on process ID %d batch ID %d.
A common cause for this error is if the client disconnected without reading the entire response from the serve
This connection will be terminated.
```

At the application side, if `close()` is called without cancelling the query first, it can result in the driver appearing to “hang” in the background, while it is waiting for SQL Server and trying to clean things up.

Mitigation

Client applications have to complete any running SQL requests before closing the connection.

If the user wants to immediately stop any running queries and close the connection, the correct course of action is to first cancel any running queries, or any not-fully-processed results in the case of data readers. This is done via SqlCommand.Cancel() in .NET or Statement.cancel() in JDBC. Call `close()` only after the `cancel()` has completed.

Public Doc Reference

cancel Method (SQLServerStatement)

> "When executing a statement that produces a single large forward-only, read-only result set, you might only be interested in some initial set of rows in the returned result set. In this case, the application might call the cancel method of the associated statement object before closing the result set in order to minimize the processing time needed to discard the remaining unnecessary rows. We recommend considering the tradeoff between the processing time that would be saved and the time and the additional round trip to the server needed to cancel the execution when deciding whether to use this technique or not."

SqlCommand.Cancel Method

High client usage when closing result set after huge query

How It Works: Attention, Attention or Should I say Cancel the Query and Be Sure to Process Your Results

How good have you found this content?