<https://dba.stackexchange.com/questions/312576/issue-with-idle-connections-open-transactions-from-python-to-sql-server/312609#312609>

Search on Database Administrators…

Database Administrators

Home

Questions

Unanswered

Tags

Users

Companies

Teams

Ask questions, find answers and collaborate at work with Stack Overflow for Teams.

Issue with idle connections/open transactions from Python to SQL Server

Asked 2 years, 10 months ago

Modified 2 years, 5 months ago

Viewed 2k times

2

I am having issue with Python code leaving idle connections to SQL Server while having open transactions in them.

Background

Recently, we had some major blockings on the SQL Server database due to Python connections. The login used has read-only access to SQL Server. The Python users are performing only reads on the database so they do not necessarily need to open transaction. However, it seems there is some default setting in Python that creates the open transaction. While users tend to leave their connections open, so we end up having resources locked sometimes from these idle connections.

Question: I manage SQL Server database and developers own Python code. Need suggestion on how developers can change their settings in Python to avoid transactions for read-only queries. One of the solution I have is to kill their connections when a particular job needs to run so as to avoid blocking but I do not prefer this approach.

sql-serversql-server-2019python

Share

Improve this question

Follow

asked May 25, 2022 at 16:53

The driver used for the database connection should have connection settings that govern transaction behavior. The pyodbc driver for python, for example, has the “autocommit” setting. With autocommit set to “True”, the SQL Server session will autocommit transactions - each statement that opens a transaction will autocommit that transaction as it completes. Using the pyodbc driver, if autocommit is set to “False” the session will have implicit\_transactions ON. A number of statements implicitly begin a transaction (unless the session is already within a transaction) in this mode.

Note that some SELECT statements may avoid starting a transaction if they do not specify a table. Implicit transactions do not commit automatically when the statement which began them completes. Implicit transactions require a commit or rollback.

More information about pyodbc here - but if using a different driver look for that driver’s documentation.

pyodbc Database Transaction Management

More information about implicit transactions here… implicit\_transactions