**About the code**

1. Runwrapper.py is a script which runs other scripts as well as individual SQL statements

2. It is called by another python script that keeps track of the return code from the execution of runwrapper.py

3. Parent script of runwrapper.py updates a table that tracks the return code from runwrapper.py

4. 2 arguments are passed as input to runwrapper.py

The first will be the job\_id and second an instance\_id.

Example:             runWrapper.py 3650 518

An instance\_id represents a batch of jobs being run.

Each job(job\_id) can have multiple parameters. A parameter can be a sql script or sql statement. The execution of one job will call the runWrapper script so the instance\_id and job\_id are constant for the entire execution of this script each time the script is called.

5. **The script will create a lock file to prevent the same job from running at the same time.  The script then logs into a Postgres database to retrieve some parameters from a table (same as the parameters in point 4 above).  The Postgres table can contain a name of a script to run or a SQL statement to run.  Once the script has completed its tasking, it will remove the lock file it created when it started.**

**Issue to be solved:**

Normally, the script runs fine but if someone adds bad SQL to the Postgress table or runs a bad SQL script then the lock file is not being removed from the system when it exits out.  This causes the next execution of this script to find the undeleted lock file then issue an exit(0) which tells the parent script that the job completed successfully when in actually didn’t do anything more than find the undeleted lock file.

So, the 2 problems to be solved:

1. To handle the condition of finding the previous lock file but still allow for the condition that the previous job may still be running.

Which implies: Checking for the lock file upon startup and if a lock file exists, use an exit(99) for this condition. The 99 is caught by the parent script as a ‘reset’ indicator so it sets the job back a state like it had not been run before. This allows the job to not pass nor fail.

2. Removing the lock file when an error occurs (Identify why the lock file is not being removed upon error)

(A little brief on “locking function” The functioning of the locking library is not material to this exercise.  Just know that it works correctly when called.  The name of the lock file will be constant for the same job\_id being run for the same instance\_id.  It is normal for runWrapper.py to run multiple times to run a sql statement to check a condition of data – like if a file has been recorded in a table.  If the condition succeeds then the job will be marked as completed and the next job will run at the next cycle (typically every minute).  If the condition is not met then the job will be reset so it can run again (using the exact same parameters) at the next cycle.  This is why it is so important that the lock file be removed when the script succeeds and any time it fails.  The issue is that it is not always being removed upon failure.)

**My proposed solution:**

For 1st problem statement:

Checking for the lock file upon startup and if a lock file exists, use an exit(99) for this condition. The 99 is caught by the parent script as a ‘reset’ indicator so it sets the job back a state like it had not been run before. This allows the job to not pass nor fail.

|  |  |
| --- | --- |
| Old code – runWrapper.py | New code runwrapper\_updated.py |
| Line 137  sys.exit(0) | (Line139-141)  logging.info(*'Previous job still under process'*) #for logging info  logging.info(*'Exiting script %s....'* %(scriptName)) #for logging info  sys.exit(99) |

For 2nd problem statement:

2. Removing the lock file when an error occurs:

Change 1:

|  |  |
| --- | --- |
| Old code – runWrapper.py | New code runwrapper\_updated.py |
| Line 552  sys.exit(1) | (Line 555-559)  # Removing the Lock  logging.info(*'Removing file lock %s '*,(lockFile))  locking.deleteLockingFile(lockFile)  sys.exit(1) |

Reason: The code was exiting without removing the lock file.

Change 2:

|  |  |
| --- | --- |
| Old code – runWrapper.py | New code runwrapper\_updated.py |
| Line 563  sys.exit(1) | (Line 570-573)  # Removing the Lock  logging.info(*'Removing file lock %s '*,(lockFile))  locking.deleteLockingFile(lockFile)  sys.exit(1) |

Reason: Again, the code was exiting without removing the lock file