jupyter modification Last Checkpoint 11 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

Out[3]:

	number	city	gender	age	income	illness
0	1	Dallas	Male	41	40367.0	No
1	2	Dallas	Male	54	45084.0	No
2	3	Dallas	Male	42	52483.0	No

In [4]:
```
df.columns
```

Out[4]:
```
Index(['number', 'city', 'gender', 'age', 'income', 'illness'], dtype='object')
```

In [5]:
```
import psycopg2
import numpy as np
# Since we want to benchmark the efficiency of each insert strategy
from timeit import default_timer as timer
```

In [6]:
```
param_dic = {
    "host"      : "localhost",
    "database"  : "toydata",
    "user"      : "myuser",
    "password"  : "Gowtham@123"
}
```

In [7]:
```
def connect(params_dic):
    """ Connect to the PostgreSQL database server """
    conn = None
    try:
        # connect to the PostgreSQL server
        print('Connecting to the PostgreSQL database...')
        conn = psycopg2.connect(**params_dic)
    except (Exception, psycopg2.DatabaseError) as error:
```

jupyter modification Last Checkpoint 11 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [1]:
```
import pandas as pd

csv_file = "toy_dataset.csv"
df = pd.read_csv(csv_file)
print("Total number of rows = %s" % len(df.index))
df.head(3)
```

```
Total number of rows = 150000
```

Out[1]:

	Number	City	Gender	Age	Income	Illness
0	1	Dallas	Male	41	40367.0	No
1	2	Dallas	Male	54	45084.0	No
2	3	Dallas	Male	42	52483.0	No

In [3]:
```
df = df.rename(columns={
    "Number": "number",
    "City": "city",
    "Gender": "gender",
     "Age": "age",
     "Income": "income",
    "Illness": "illness"
})


df.head(3)
```

Out[3]:

	number	city	gender	age	income	illness
0	1	Dallas	Male	41	40367.0	No

jupyter modification Last Checkpoint: 11 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [7]:
```
def connect(params_dic):
    """ Connect to the PostgreSQL database server """
    conn = None
    try:
        # connect to the PostgreSQL server
        print('Connecting to the PostgreSQL database...')
        conn = psycopg2.connect(**params_dic)
    except (Exception, psycopg2.DatabaseError) as error:
        print(error)
        sys.exit(1)
    print("Connection successful")
    return conn
```

In [8]:
```
conn = connect(param_dic)
```

```
Connecting to the PostgreSQL database...
Connection successful
```

In [9]:
```
#write the code to create table called (info) using the  above columns i gave before
def create_table(conn):
    """ Create a table called 'info' in the PostgreSQL database """
    create_table_query = '''
    CREATE TABLE IF NOT EXISTS info (
        number INT,
        city TEXT,
        gender TEXT,
        age INT,
        income FLOAT,
        illness BOOLEAN
    )
    '''
    try:
        # create a cursor object
        cursor = conn.cursor()
```

jupyter modification Last Checkpoint: 11 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

In [9]:
```
#write the code to create table called (info) using the  above columns i gave before
def create_table(conn):
    """ Create a table called 'info' in the PostgreSQL database """
    create_table_query = '''
    CREATE TABLE IF NOT EXISTS info (
        number INT,
        city TEXT,
        gender TEXT,
        age INT,
        income FLOAT,
        illness BOOLEAN
    )
    '''
    try:
        # create a cursor object
        cursor = conn.cursor()
        # execute the create table query
        cursor.execute(create_table_query)
        # commit the changes to the database
        conn.commit()
        print("Table 'info' created successfully.")
    except (Exception, psycopg2.DatabaseError) as error:
        print("Error creating table:", error)
        conn.rollback()

# Call the create_table function
create_table(conn)
```

```
Table 'info' created successfully.
```

In [10]:
```
def execute_query(conn, query):
    """ Execute a single query """
```

jupyter modification Last Checkpoint: 11 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

```
Table 'info' created successfully.
```

In [10]:

```
def execute_query(conn, query):
    """ Execute a single query """

    ret = 0 # Return value
    cursor = conn.cursor()
    try:
        cursor.execute(query)
        conn.commit()
    except (Exception, psycopg2.DatabaseError) as error:
        print("Error: %s" % error)
        conn.rollback()
        cursor.close()
        return 1

    # If this was a select query, return the result
    if 'select' in query.lower():
        ret = cursor.fetchall()
    cursor.close()
    return ret
```

In [11]:

```
def execute_many(conn, df, table):
    """
    Using cursor.executemany() to insert the dataframe
    """
    # Create a list of tupples from the dataframe values
    tuples = [tuple(x) for x in df.to_numpy()]
    # Comma-separated dataframe columns
    cols = ','.join(list(df.columns))
    # SQL query to execute multiple insertion
    query  = "INSERT INTO %s(%s) VALUES(%%s,%%s,%%s,%%s,%%s,%%s)" % (table, cols)
    cursor = conn.cursor()
```

jupyter modification Last Checkpoint: 11 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [11]:
```
def execute_many(conn, df, table):
    """
    Using cursor.executemany() to insert the dataframe
    """
    # Create a list of tupples from the dataframe values
    tuples = [tuple(x) for x in df.to_numpy()]
    # Comma-separated dataframe columns
    cols = ','.join(list(df.columns))
    # SQL query to execute multiple insertion
    query  = "INSERT INTO %s(%s) VALUES(%%s,%%s,%%s,%%s,%%s,%%s)" % (table, cols)
    cursor = conn.cursor()
    try:
        cursor.executemany(query, tuples)
        conn.commit()
    except (Exception, psycopg2.DatabaseError) as error:
        print("Error: %s" % error)
        conn.rollback()
        cursor.close()
        return 1
    print("execute_many() done")
    cursor.close()

# Run the execute_many strategy
execute_many(conn, df, 'info')
```

```
execute_many() done
```

In [12]:
```
execute_query(conn, "select count(*) from info;")
# Check that the values were indeed inserted
```

Out[12]:
```
[(150000,)]
```

In [13]:
```
# Clear the table
execute_query(conn, "delete from info where true;")
```

jupyter modification Last Checkpoint: 11 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

Out[12]:
```
[(150000,)]
```

In [13]:
```
# Clear the table
execute_query(conn, "delete from info where true;")
```

Out[13]:
```
0
```

In [14]:
```
import psycopg2.extras as extras

def execute_batch(conn, df, table, page_size=100):
    """
    Using psycopg2.extras.execute_batch() to insert the dataframe
    """
    # Create a list of tupples from the dataframe values
    tuples = [tuple(x) for x in df.to_numpy()]
    # Comma-separated dataframe columns
    cols = ','.join(list(df.columns))
    # SQL quert to execute
    query  = "INSERT INTO %s(%s) VALUES(%%s,%%s,%%s,%%s,%%s,%%s)" % (table, cols)
    cursor = conn.cursor()
    try:
        extras.execute_batch(cursor, query, tuples, page_size)
        conn.commit()
    except (Exception, psycopg2.DatabaseError) as error:
        print("Error: %s" % error)
        conn.rollback()
        cursor.close()
        return 1
    print("execute_batch() done")
    cursor.close()


# Run the execute_many strategy
```

jupyter modification Last Checkpoint: 11 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted

Python 3 (ipykernel)

Run Code

In [14]:
```
import psycopg2.extras as extras

def execute_batch(conn, df, table, page_size=100):
    """
    Using psycopg2.extras.execute_batch() to insert the dataframe
    """
    # Create a list of tupples from the dataframe values
    tuples = [tuple(x) for x in df.to_numpy()]
    # Comma-separated dataframe columns
    cols = ','.join(list(df.columns))
    # SQL quert to execute
    query  = "INSERT INTO %s(%s) VALUES(%%s,%%s,%%s,%%s,%%s,%%s)" % (table, cols)
    cursor = conn.cursor()
    try:
        extras.execute_batch(cursor, query, tuples, page_size)
        conn.commit()
    except (Exception, psycopg2.DatabaseError) as error:
        print("Error: %s" % error)
        conn.rollback()
        cursor.close()
        return 1
    print("execute_batch() done")
    cursor.close()


# Run the execute_many strategy
execute_batch(conn, df, 'info')
```

```
execute_batch() done
```

In [15]:
```
# Check that the values were indeed inserted
execute_query(conn, "select count(*) from info;")
```

jupyter modification Last Checkpoint 11 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

```
execute_batch() done
```

In [15]:
```
# Check that the values were indeed inserted
execute_query(conn, "select count(*) from info;")
```

Out[15]:
```
[(150000,)]
```

In [16]:
```
# Clear the table
execute_query(conn, "delete from info where true;")
```

Out[16]:
```
0
```

In [17]:
```
def execute_values(conn, df, table):
    """
    Using psycopg2.extras.execute_values() to insert the dataframe
    """
    # Create a list of tupples from the dataframe values
    tuples = [tuple(x) for x in df.to_numpy()]
    # Comma-separated dataframe columns
    cols = ','.join(list(df.columns))
    # SQL quert to execute
    query  = "INSERT INTO %s(%s) VALUES %%s" % (table, cols)
    cursor = conn.cursor()
    try:
        extras.execute_values(cursor, query, tuples)
        conn.commit()
    except (Exception, psycopg2.DatabaseError) as error:
        print("Error: %s" % error)
        conn.rollback()
        cursor.close()
        return 1
    print("execute_values() done")
    cursor.close()
```

jupyter modification Last Checkpoint: 11 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [17]:
```
def execute_values(conn, df, table):
    """
    Using psycopg2.extras.execute_values() to insert the dataframe
    """
    # Create a list of tupples from the dataframe values
    tuples = [tuple(x) for x in df.to_numpy()]
    # Comma-separated dataframe columns
    cols = ','.join(list(df.columns))
    # SQL quert to execute
    query  = "INSERT INTO %s(%s) VALUES %%s" % (table, cols)
    cursor = conn.cursor()
    try:
        extras.execute_values(cursor, query, tuples)
        conn.commit()
    except (Exception, psycopg2.DatabaseError) as error:
        print("Error: %s" % error)
        conn.rollback()
        cursor.close()
        return 1
    print("execute_values() done")
    cursor.close()
```

In [18]:
```
# Run the execute_many strategy
execute_values(conn, df, 'info')
```

```
execute_values() done
```

In [19]:
```
# Check that the values were indeed inserted
execute_query(conn, "select count(*) from info;")
```

Out[19]:
```
[(150000,)]
```

In [20]:
```
# Clear the table
```

jupyter modification Last Checkpoint: 12 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [18]:
```
# Run the execute_many strategy
execute_values(conn, df, 'info')
```

```
execute_values() done
```

In [19]:
```
# Check that the values were indeed inserted
execute_query(conn, "select count(*) from info;")
```

Out[19]:
```
[(150000,)]
```

In [20]:
```
# Clear the table
execute_query(conn, "delete from info where true;")
```

Out[20]:
```
0
```

In [21]:
```
def execute_mogrify(conn, df, table):
    """
    Using cursor.mogrify() to build the bulk insert query
    then cursor.execute() to execute the query
    """
    # Create a list of tupples from the dataframe values
    tuples = [tuple(x) for x in df.to_numpy()]
    # Comma-separated dataframe columns
    cols = ','.join(list(df.columns))
    # SQL quert to execute
    cursor = conn.cursor()
    values = [cursor.mogrify("(%s,%s,%s,%s,%s,%s,%s)", tup).decode('utf8') for tup in tuples]
    query  = "INSERT INTO %s(%s) VALUES " % (table, cols) + ",".join(values)

    try:
        cursor .execute(query, tuples)
        conn.commit()
    except (Exception, psycopg2.DatabaseError) as error:
```

jupyter modification Last Checkpoint: 12 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [21]:
```
def execute_mogrify(conn, df, table):
    """
    Using cursor.mogrify() to build the bulk insert query
    then cursor.execute() to execute the query
    """
    # Create a list of tupples from the dataframe values
    tuples = [tuple(x) for x in df.to_numpy()]
    # Comma-separated dataframe columns
    cols = ','.join(list(df.columns))
    # SQL quert to execute
    cursor = conn.cursor()
    values = [cursor.mogrify("(%s,%s,%s,%s,%s,%s)", tup).decode('utf8') for tup in tuples]
    query  = "INSERT INTO %s(%s) VALUES " % (table, cols) + ",".join(values)

    try:
        cursor .execute(query, tuples)
        conn.commit()
    except (Exception, psycopg2.DatabaseError) as error:
        print("Error: %s" % error)
        conn.rollback()
        cursor.close()
        return 1
    print("execute_mogrify() done")
    cursor.close()
```

In [22]:
```
# Run the execute_many strategy
execute_mogrify(conn, df, 'info')
```

```
execute_mogrify() done
```

In [23]:
```
# Check that the values were indeed inserted
execute_query(conn, "select count(*) from info;")
```

jupyter modification Last Checkpoint: 12 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

```
execute_mogrify() done
```

In [23]:
```
# Check that the values were indeed inserted
execute_query(conn, "select count(*) from info;")
```

Out[23]:
```
[(150000,)]
```

In [24]:
```
# Clear the table
execute_query(conn, "delete from info where true;")
```

Out[24]:
```
0
```

In [25]:
```
def single_inserts(conn, df, table):
    """
    For benchmarking purposes, let's also implement a strategy in
    which the inserts calls are done separately. Here we are
    creating and closing a new cursor at each time, so this is probably
    one of the slowest ways to achieve what we want.
    """
    for i in df.index:
        cols  = ','.join(list(df.columns))
        vals  = [df.at[i,col] for col in list(df.columns)]
        query = "INSERT INTO %s(%s) VALUES(%s,'%s','%s',%s,%s,'%s')" %  (table, cols, vals[0], vals[1], vals[2], vals[3], vals[4]
        execute_query(conn, query)
    print("single_inserts() done")


#----------------------------------------
# COMPARE THE SPEED OF EACH STRATRGY
#----------------------------------------
def benchmark_bulk_inserts(df):
    execute_query(conn, "delete from info where true;")
    functions = [
```

jupyter modification Last Checkpoint: 12 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [25]:
```
def single_inserts(conn, df, table):
    """
    For benchmarking purposes, let's also implement a strategy in
    which the inserts calls are done separately. Here we are
    creating and closing a new cursor at each time, so this is probably
    one of the slowest ways to achieve what we want.
    """
    for i in df.index:
        cols  = ','.join(list(df.columns))
        vals  = [df.at[i,col] for col in list(df.columns)]
        query = "INSERT INTO %s(%s) VALUES(%s,'%s','%s',%s,%s,'%s')" %  (table, cols, vals[0], vals[1], vals[2], vals[3], vals[4]
        execute_query(conn, query)
    print("single_inserts() done")


#----------------------------------------
# COMPARE THE SPEED OF EACH STRATRGY
#----------------------------------------
def benchmark_bulk_inserts(df):
    execute_query(conn, "delete from info where true;")
    functions = [
        single_inserts,
        execute_many,
        execute_batch,
        execute_values,
        execute_mogrify
     ]
    exec_times = pd.DataFrame(index=range(len(functions)), columns=['nrows','strategy','time'])

    # Psycop2's functions
    i = 0
    for strategy in functions:
        start = timer()
        strategy(conn, df, 'info')
```

jupyter modification Last Checkpoint: 12 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

```
#----------------------------------------
# COMPARE THE SPEED OF EACH STRATRGY
#----------------------------------------
def benchmark_bulk_inserts(df):
    execute_query(conn, "delete from info where true;")
    functions = [
        single_inserts,
        execute_many,
        execute_batch,
        execute_values,
        execute_mogrify
     ]
    exec_times = pd.DataFrame(index=range(len(functions)), columns=['nrows','strategy','time'])

    # Psycop2's functions
    i = 0
    for strategy in functions:
        start = timer()
        strategy(conn, df, 'info')
        end = timer()

        exec_times.at[i,'nrows'] = len(df.index)
        exec_times.at[i,'strategy'] = strategy.__name__
        exec_times.at[i,'time'] = end-start

        # Prepare for the next test
        execute_query(conn, "delete from info where true;")
        i = i + 1


    return exec_times

benchmark_bulk_inserts(df)
```

jupyter modification Last Checkpoint: 12 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

```
    ]
    exec_times = pd.DataFrame(index=range(len(functions)), columns=['nrows','strategy','time'])

    # Psycop2's functions
    i = 0
    for strategy in functions:
        start = timer()
        strategy(conn, df, 'info')
        end = timer()

        exec_times.at[i,'nrows'] = len(df.index)
        exec_times.at[i,'strategy'] = strategy.__name__
        exec_times.at[i,'time'] = end-start

        # Prepare for the next test
        execute_query(conn, "delete from info where true;")
        i = i + 1


    return exec_times

benchmark_bulk_inserts(df)
```

```
single_inserts() done
execute_many() done
execute_batch() done
execute_values() done
execute_mogrify() done
```

Out[25]:

	nrows	strategy	time
0	150000	single_inserts	132.111471

jupyter modification Last Checkpoint: 12 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

```
single_inserts() done
execute_many() done
execute_batch() done
execute_values() done
execute_mogrify() done
```

Out[25]:

	nrows	strategy	time
0	150000	single_inserts	132.111471
1	150000	execute_many	36.304083
2	150000	execute_batch	6.525919
3	150000	execute_values	3.264394
4	150000	execute_mogrify	3.212137

In [26]:
```
# Repeating our dataframe 30 times to get a large test dataframe
big_df = pd.concat([df]*31, ignore_index=True)
print(len(big_df.index))
```

```
4650000
```

In [40]:
```
df_lst = []

for nrows in [100000,500000,1000000,3000000]:
    print("Nrows = %s" % nrows)
    test_df = big_df[0:nrows]
    perf_df = benchmark_bulk_inserts(test_df)
    df_lst.append(perf_df)


performances = pd.concat(df_lst, axis=0).reset_index()
performances.head(3)
```

jupyter modification Last Checkpoint: 12 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [26]:
```
# Repeating our dataframe 30 times to get a large test dataframe
big_df = pd.concat([df]*31, ignore_index=True)
print(len(big_df.index))
```

```
4650000
```

In [40]:
```
df_lst = []

for nrows in [100000,500000,1000000,3000000]:
    print("Nrows = %s" % nrows)
    test_df = big_df[0:nrows]
    perf_df = benchmark_bulk_inserts(test_df)
    df_lst.append(perf_df)


performances = pd.concat(df_lst, axis=0).reset_index()
performances.head(3)
```

```
Nrows = 100000
single_inserts() done
execute_many() done
execute_batch() done
execute_values() done
execute_mogrify() done
Nrows = 500000
single_inserts() done
execute_many() done
execute_batch() done
execute_values() done
execute_mogrify() done
Nrows = 1000000
single_inserts() done
execute_many() done
execute_batch() done
execute_values() done
```

jupyter modification Last Checkpoint: 12 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

```
execute_mogrify() done
Nrows = 3000000
single_inserts() done
execute_many() done
execute_batch() done
execute_values() done
execute_mogrify() done
```

Out[40]:

	Index	nrows	strategy	time
0	0	100000	single_inserts	43.563458
1	1	100000	execute_many	10.750199
2	2	100000	execute_batch	2.138876

In [41]:

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(12,5))
for strategy in performances['strategy'].unique():
    subset = performances[performances['strategy'] == strategy]
    ax.plot(subset['nrows'], subset['time'], 'o--', label=strategy)

plt.xlabel('Number of rows in dataframe')
plt.ylabel('Execution time (sec)')
plt.title("COMPARING THE DIFFERENT BULK INSERT STRATEGIES", fontsize=14)
plt.legend()
plt.savefig("./all_insert_strategies.png", dpi=400)
plt.show()
```