Сидоренко Катерина, ФБ-11мп

Task 1 Оцінка вставки 100К значень в реляційну СКБД

Необхідно земулювати 100К приблизно одночасних запитів на вставку даних в реляційну СКБД, та оцінити час необхідний для цього

```
import mysql.connector as connector
import mysql.connector.pooling as pooling
from concurrent.futures import ThreadPoolExecutor
from concurrent.futures import wait
import time

CONFIG = {
  'user': 'client',
  'password': '<password>',
  'host': '127.0.0.1',
  'database': 'db',
}
```

```
TOTAL = 100_000
SQL = "INSERT INTO posts (id, text) VALUES (%s, %s)"
```

```
def measure_time(func, name):
    start = time.time()
    func()
    end = time.time()
    print(name, ":", end - start)
```

```
def drop_table(conn):
    cur = conn.cursor()
    cur.execute("DELETE FROM posts")
    conn.commit()
    cur.close()
```

```
def insert(conn, cursor, i):
    try:
        cursor.execute(SQL, (i, f"Post {i}"))
        conn.commit()
    except Exception as e:
        print(e)
```

```
def insert_and_close(conn, i):
    cursor = conn.cursor()
    insert(conn, cursor, i)
    cursor.close()
    conn.close()
```

```
## SYNC INSERT OF 100_000 entries
def run_sync_and_measure():
    conn = connector.connect(**CONFIG)
    drop_table(conn)
    measure_time(lambda: run_sync(conn), "sync")
    conn.close()
```

```
def run_sync(conn):
    cursor = conn.cursor()
    for i in range(TOTAL):
        insert(conn, cursor, i)
    cursor.close()
```

```python
## ASYNC INSERT OF 100_000 entries
def run_async_and_measure(n_threads):
    conn = connector.connect(**CONFIG)
    drop_table(conn)
    conn.close()
```

```python
    measure_time(lambda: run_async(n_threads), f"async {n_threads}")
```

```python
def run_async(n_threads):
    with ThreadPoolExecutor(max_workers=n_threads) as executor:
        for i in range(0, TOTAL, n_threads):
            futures = []
            for j in range(i, i+n_threads):
                conn = connector.connect(**CONFIG)
                futures.append(executor.submit(insert_and_close, conn, j))
            wait(futures)
```

```python
# ASYNC WITH CONNECTION POOL
def run_async_with_pool(n_threads):
    pool = pooling.MySQLConnectionPool(pool_size = n_threads,**CONFIG)
    with ThreadPoolExecutor(max_workers=n_threads) as executor:
        for i in range(0, TOTAL, n_threads):
            futures = []
            for j in range(i, i+n_threads):
                conn = pool.get_connection()
                futures.append(executor.submit(insert_and_close, conn, j))
            wait(futures)
```

```python
def run_async_with_pool_and_measure(n_threads):
    conn = connector.connect(**CONFIG)
    drop_table(conn)
    conn.close()
```

```python
    measure_time(lambda: run_async_with_pool(n_threads), f"async & pool {n_threads}")
```

```python
if __name__ == "__main__":
    run_sync_and_measure() # sync : 1129.7238166332245
```

```python
    run_async_and_measure(20) # async 20 : 1405.968431711197
    run_async_and_measure(50) # async 50 : 1377.8299469947815
    run_async_and_measure(100) # async 100 : 1366.4317593574524
```

```python
    run_async_with_pool_and_measure(20) # async & pool 20 : 1317.2488374710083
    run_async_with_pool_and_measure(32) # async & pool 32 : 1294.6909358501434
```

Сценарий	Время(с)
Последовательно	1129.72
Параллельно (20 потоков) без connection pool	1405.97
Параллельно (50 потоков) без connection pool	1377.83
Параллельно (100 потоков) без connection pool	1366.43
Параллельно (20 потоков) с connection pool на 20	1317.25
Параллельно (32 потока) с connection pool на 32	1294.69