# Database Project (SWE3033) (Fall 2024)

# Homework #2 (50pts, Due date: 09/24)

**Student ID**: ____2018310773____

**Student Name**: _____임승재____

1. **[5pts]** Write the system setup of your environment. Fill in the blanks below.

| Type | Specification |
|---|---|
| OS | Windows 10 |
| CPU | Intel core i7-7700HQ |
| Memory (RAM) | 16GB |
| Linux Kernel | ubuntu 22.04.3 LTS |

2. **[5pts]** Write the benchmark setup refer to the following command line. Fill in the blanks below.

```
./tpcc_start -h mysql -u root –p 1234 -d tpcc5 -w 7 \
-S /tmp/mysql.sock –c 6 –r 20 –l 150
```

| Type | Specification |
|---|---|
| Number of Warehouses | 7 |
| Database Name | root |
| Measure Time | 150 |
| Number of Connections | 6 |

3. **[25pts]** Run the benchmark that you have a database with a size of 10 warehouses. Check out the slides for detailed configurations. Present experimental results and analyze the results yourself. Explain the reason for the performance shift as the buffer size varies.

| Buffer pool size | TpmC | Hit ratio |
|---|---|---|
| 64M | 3017.2 | 927/1000 |
| 256M | 8073.0 | 986/1000 |
| 512M | 9978.6 | 996/1000 |
| 1G | 11340.4 | 1000/1000 |
| 2G | 12013.2 | 1000/1000 |

When the buffer size was initially increased from 64MB to 256MB, there was a significant performance improvement. However, the rate of improvement gradually decreased, and eventually, there was almost no improvement when the buffer size was increased from 1GB to 2GB. This is primarily related to the size of the database; the database consists of 10 warehouses, each occupying around 100MB, resulting in a total database size of approximately 1GB. Therefore, in theory, once the buffer size reaches 1GB or more, which is enough to hold

all the database data in memory, any further increase in buffer size becomes irrelevant to performance improvement. Consequently, the optimal buffer pool size for achieving the best performance with this database is 1GB. In fact, it was observed that the hit rate reached 100% when the buffer size was set to 1GB. Nevertheless, the reason why significant performance improvements were seen even before the buffer size reached 1GB is that the database buffer not only loads database data into memory but also uses algorithms to manage buffer frames efficiently, thereby minimizing access to the actual storage and increasing the hit rate.

1) **[15pts]** Fill in the performance table below as you change the buffer size. Attach screen shots showing the results for each buffer size. These screen shots should include `<Raw Results>`, `<TpmC>`, and `BUFFER POOL AND MEMORY` information.

*Hint: Use the tpcc_start and SHOW commands to monitor the innodb status while the benchmark is running.*

| Buffer Pool Size | TpmC | Hit Ratio |
|---|---|---|
| 64M | 3017.2 | 927/1000 |
| 256M | 8073.0 | 986/1000 |
| 512M | 9978.6 | 996/1000 |

1) 64M

```
<Raw Results>
  [0] sc:0 lt:15086  rt:0  fl:0 avg_rt: 81.5 (5)
  [1] sc:564 lt:14486  rt:0  fl:0 avg_rt: 40.2 (5)
  [2] sc:1032 lt:477  rt:0  fl:0 avg_rt: 16.8 (5)
  [3] sc:64 lt:1444  rt:0  fl:0 avg_rt: 363.4 (80)
  [4] sc:0 lt:1510  rt:0  fl:0 avg_rt: 149.1 (20)
 in 300 sec.

<TpmC>
                3017.200 TpmC
---------------------
BUFFER POOL AND MEMORY
---------------------
Total large memory allocated 68714496
Dictionary memory allocated 157272
Buffer pool size   4096
Free buffers       72
Database pages     3929
Old database pages 1436
Modified db pages  1963
Pending reads      0
Pending writes: LRU 121, flush list 0, single page 8
Pages made young 35049, not young 2787840
0.00 youngs/s, 0.00 non-youngs/s
Pages read 883881, created 63664, written 381787
0.00 reads/s, 0.00 creates/s, 0.00 writes/s
Buffer pool hit rate 925 / 1000, young-making rate 2 / 1000 not 209 / 1000
Pages read ahead 0.00/s, evicted without access 0.00/s, Random read ahead 0.00/s
LRU len: 3929, unzip_LRU len: 0
I/O sum[220729]:cur[1354], unzip sum[0]:cur[0]
```

2) 256M

```
<Raw Results>
  [0] sc:1 lt:40364  rt:0  fl:0 avg_rt: 29.6 (5)
  [1] sc:7257 lt:33052  rt:0  fl:0 avg_rt: 14.8 (5)
  [2] sc:3502 lt:535  rt:0  fl:0 avg_rt: 5.9 (5)
  [3] sc:2463 lt:1573  rt:0  fl:0 avg_rt: 131.4 (80)
  [4] sc:0 lt:4037  rt:0  fl:0 avg_rt: 70.8 (20)
 in 300 sec.

<TpmC>
                8073.000 TpmC
```

```
---------------------
BUFFER POOL AND MEMORY
---------------------
Total large memory allocated 274857984
Dictionary memory allocated 157272
Buffer pool size   16382
Free buffers       970
Database pages     14389
Old database pages 5299
Modified db pages  8513
Pending reads      0
Pending writes: LRU 0, flush list 0, single page 0
Pages made young 420911, not young 2285332
0.00 youngs/s, 0.00 non-youngs/s
Pages read 681208, created 69746, written 617197
0.00 reads/s, 0.00 creates/s, 0.00 writes/s
Buffer pool hit rate 986 / 1000, young-making rate 9 / 1000 not 48 / 1000
Pages read ahead 0.00/s, evicted without access 0.00/s, Random read ahead 0.00/s
LRU len: 14389, unzip_LRU len: 0
I/O sum[152487]:cur[2232], unzip sum[0]:cur[0]
```

3) 512M

```
<Raw Results>
  [0] sc:2 lt:49891  rt:0  fl:0 avg_rt: 23.5 (5)
  [1] sc:13360 lt:36477  rt:0  fl:0 avg_rt: 12.1 (5)
  [2] sc:4505 lt:485  rt:0  fl:0 avg_rt: 4.1 (5)
  [3] sc:3721 lt:1269  rt:0  fl:0 avg_rt: 112.7 (80)
  [4] sc:0 lt:4988  rt:0  fl:0 avg_rt: 55.7 (20)
 in 300 sec.

<TpmC>
                    9978.600 TpmC
---------------------
BUFFER POOL AND MEMORY
---------------------
Total large memory allocated 549715968
Dictionary memory allocated 157272
Buffer pool size   32764
Free buffers       1027
Database pages     29623
Old database pages 10915
Modified db pages  19843
Pending reads      0
Pending writes: LRU 0, flush list 121, single page 0
Pages made young 125501, not young 352911
0.00 youngs/s, 0.00 non-youngs/s
Pages read 80712, created 60410, written 187593
0.00 reads/s, 0.00 creates/s, 0.00 writes/s
Buffer pool hit rate 996 / 1000, young-making rate 9 / 1000 not 15 / 1000
Pages read ahead 0.00/s, evicted without access 0.00/s, Random read ahead 0.00/s
LRU len: 29623, unzip_LRU len: 0
I/O sum[75238]:cur[356], unzip sum[0]:cur[0]
```

2) **[10pts]** Write which buffer pool size performs best in the above experiment and explain how the buffer pool size affects the hit ratio and tpmC, respectively, and why.

Buffer pool size with 512M performs best in the above experiment. As I answered in question 2, for this database, the performance continues to improve up to a buffer size of 1GB. As the buffer size increases, the amount of database data pages that can be stored in the buffer also increases, resulting in a hit rate that continues to rise until the buffer size reaches 1GB. In the case of TpmC, which measures the number of New-Order transactions performed per minute, it decreases when the hit rate is low due to the high frequency of disk I/O operations needed to fetch data from the database. However, as the hit rate increases, the number of disk I/O operations decreases, causing TpmC to rise accordingly.

4. **[15pts]** Why is it difficult to store the entire database in memory in real world, and why do we need to use buffer pool in database management system?

In the real world, databases are extremely large in size. Memory is much more expensive compared to disk storage, and data stored in memory is lost when the device's power is turned off. Therefore, it is difficult to store the entire database in memory in real-world scenarios. Consequently, it is important to establish a buffer pool in the memory space that connects the disk storage and the central processing unit (CPU). By implementing an appropriate buffer policy, data stored in the buffer can be utilized when accessing the database, preventing direct disk I/O operations.