CSCD43 A1 Report

I. Group Information

- Student 1: Zhitao Xu, Student number: 1006668697
- Student 2: Lianting Wang, Student number: 1007452374

II. Preparation

1. SQL statements for Scan Queries and Index Scan Queries

- Scan Queries
- (Seq Scan) SELECT COUNT(*) FROM Data WHERE A < 45;
- (Seq Scan) SELECT COUNT(*) FROM Data WHERE B % 5 = 0;
- (Seq Scan) SELECT COUNT(*) FROM Data WHERE C BETWEEN 15 AND 25;
- (Seq Scan) SELECT COUNT(*) FROM Data WHERE A > 30 AND B > 45 AND C < 70;
- (Seq Scan) SELECT COUNT(*) FROM Data WHERE A % 4 = 0 AND C % 4 = 0;
- (Seq Scan) SELECT COUNT(*) FROM Data WHERE A < 25 OR B > 85;
- (Seq Scan) SELECT COUNT(*) FROM Data WHERE C > 85;
- (Seq Scan) SELECT COUNT(*) FROM Data WHERE A BETWEEN 45 AND 55;
- (Seq Scan) SELECT COUNT(*) FROM Data WHERE B < 30 OR C > 70;
- (Seq Scan) SELECT COUNT(*) FROM Data WHERE A < 15 OR A > 85;

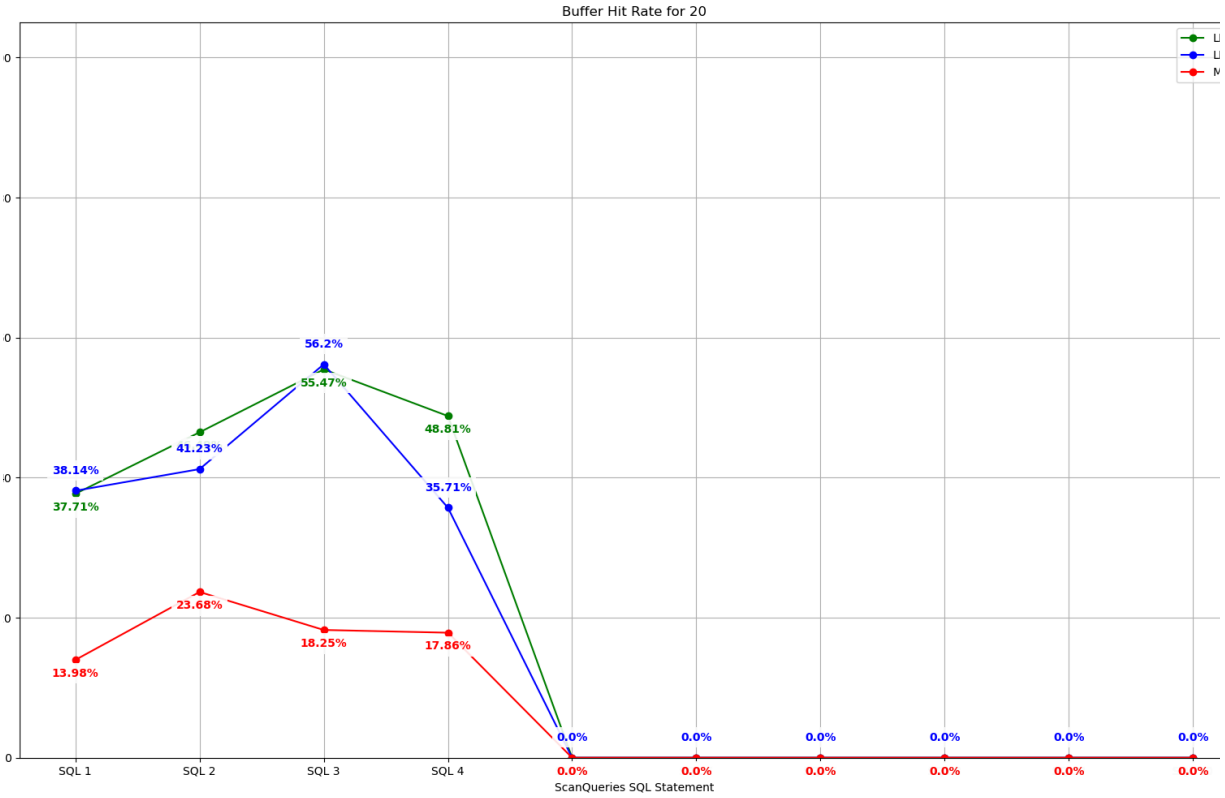
- IndexScanQueries
- (Index Scan) SELECT COUNT(*) FROM Data WHERE ID BETWEEN 10 AND 50;
- (Seq Scan) SELECT COUNT(*) FROM Data WHERE ID % 100 = 0;
- (Index Scan) SELECT COUNT(*) FROM Data WHERE ID > 2500 AND ID < 2600;
- (Index Scan) SELECT COUNT(*) FROM Data WHERE ID BETWEEN 1 AND 1000;
- (Seq Scan) SELECT COUNT(*) FROM Data WHERE ID % 2 = 0;
- (Seq Scan) SELECT COUNT(*) FROM Data WHERE ID < 500 OR ID > 4500;
- (Index Scan) SELECT COUNT(*) FROM Data WHERE ID BETWEEN 300 AND 600;
- (Index Scan) SELECT COUNT(*) FROM Data WHERE ID BETWEEN 100 AND 200;
- (Index Scan) SELECT COUNT(*) FROM Data WHERE ID BETWEEN 2150 AND 2250;
- (Index Scan) SELECT COUNT(*) FROM Data WHERE ID BETWEEN 4200 AND 4230;

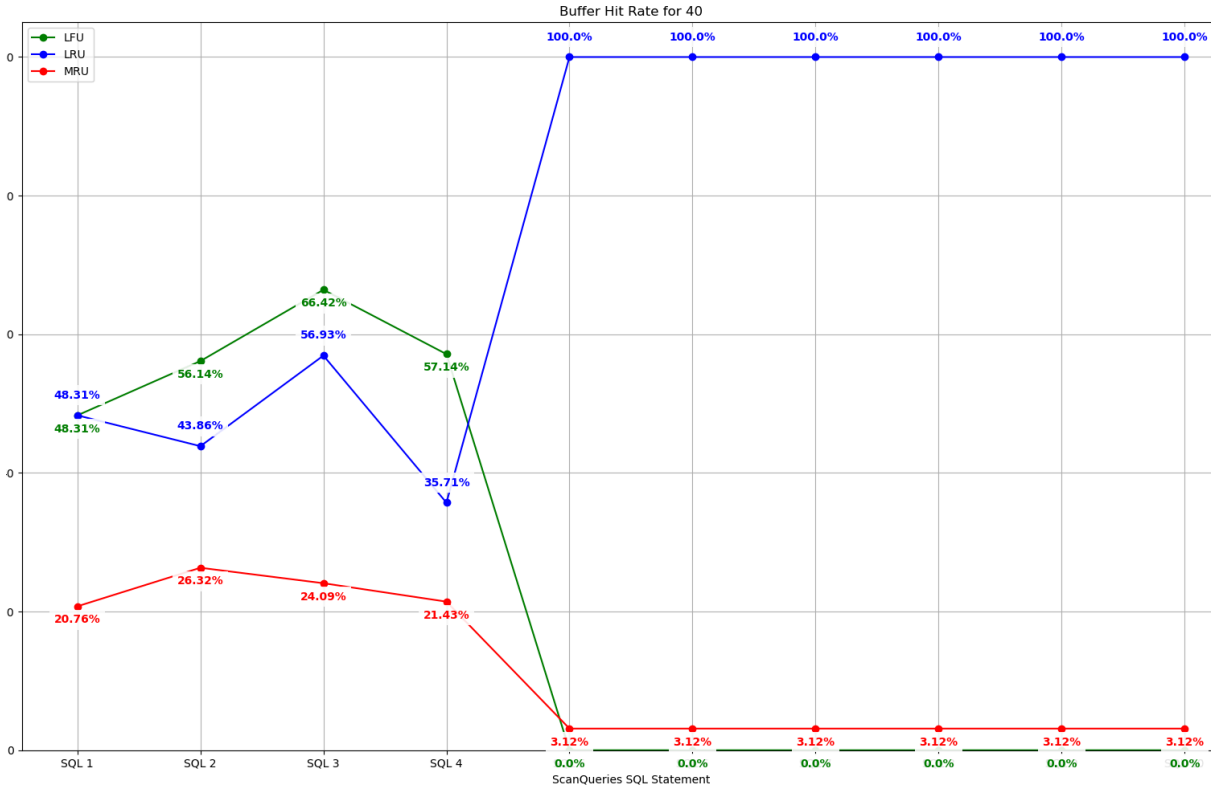
2. Apply " EXPLAIN" on the SQL statements

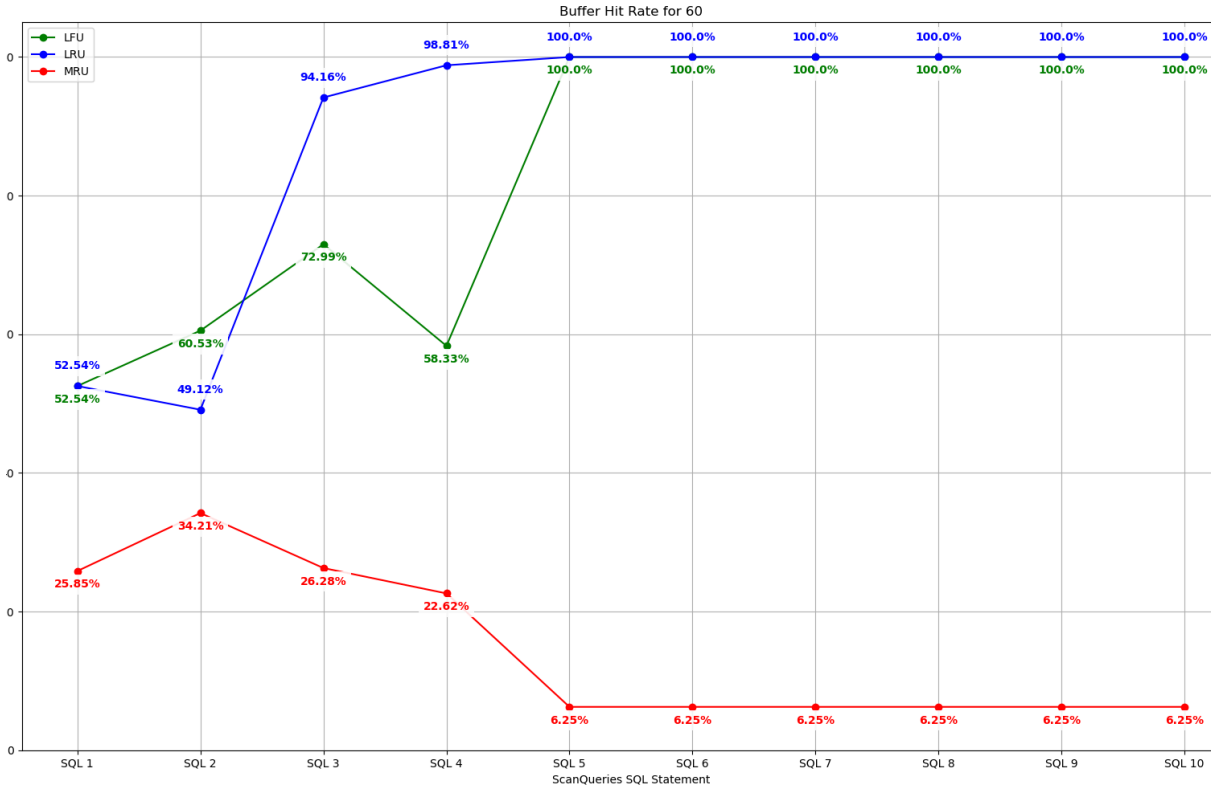
- Results are in the bracket before the Query
- We could see Postgres choose either Seq Scan on the query or Index Scan on the queries and later in the report, we will see how the buffer rate will be related.
    - For Index Scan Queries, statements 2, 5, 6 is Seq Scan and the result of IndexScanQueries are Index Scan

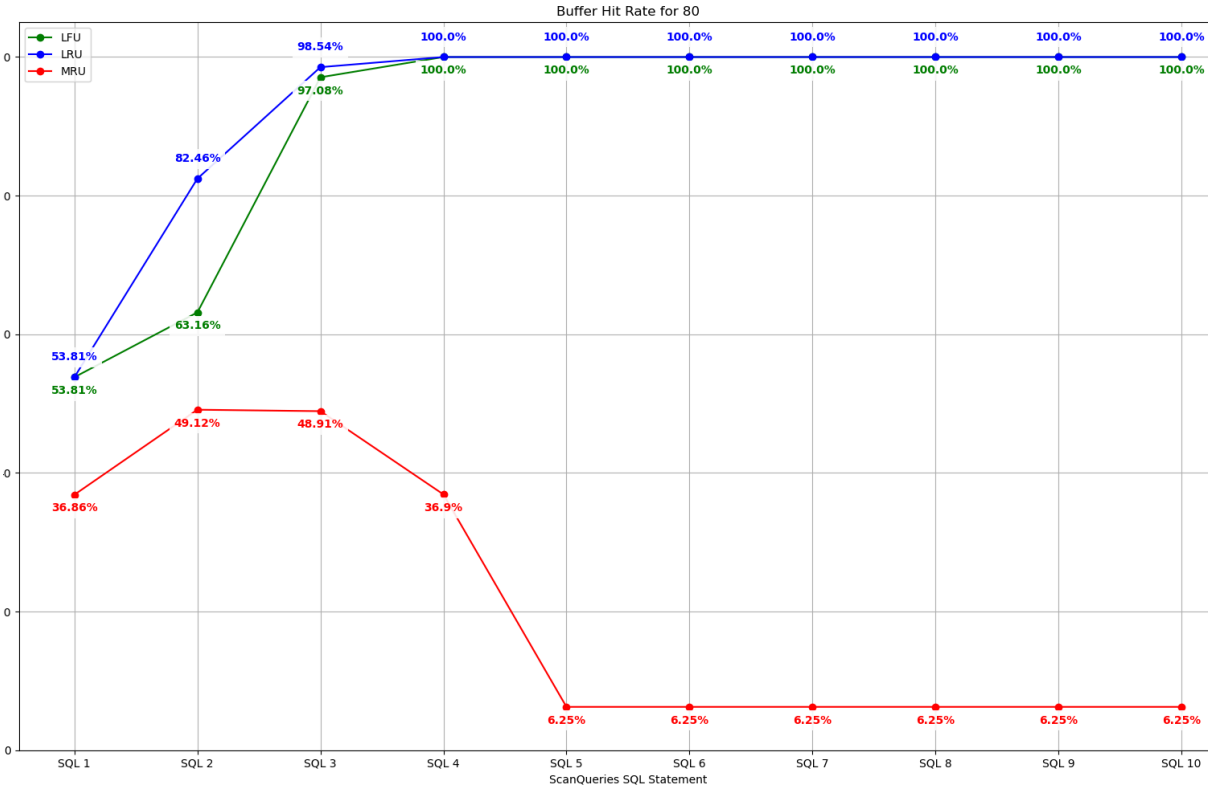
III. Generate the Data Sequentially

- By using the command get from handout
    - cat <queryfile.sql> | postgres -B <numbuffers> -D <datadir> -d <debug-level> -s <databasename>
- We can get a series of diagrams comparing the buffer hit rate in different buffer sizes for different SQL statements. The whole result is in the folder /result/sequential.png. Some partial diagrams (the most meaningful ones) are on the next page.
- ***For Sequential SQL Statement of Scan Queries:***
- For the diagram, we could see that for both LRU and LFU algo, the buffer hit rate becomes 0% after the fourth query when “num buffers” is less than 32. After the threshold of the number of buffers 32, LRU will recover quickly and have a 100% butter hit rate since the fifth query. For LFU, the buffer hit rate will start to catch up as the number of the buffers increases and over 50 number of buffers.
- Some Diagrams of the Scan Queries:

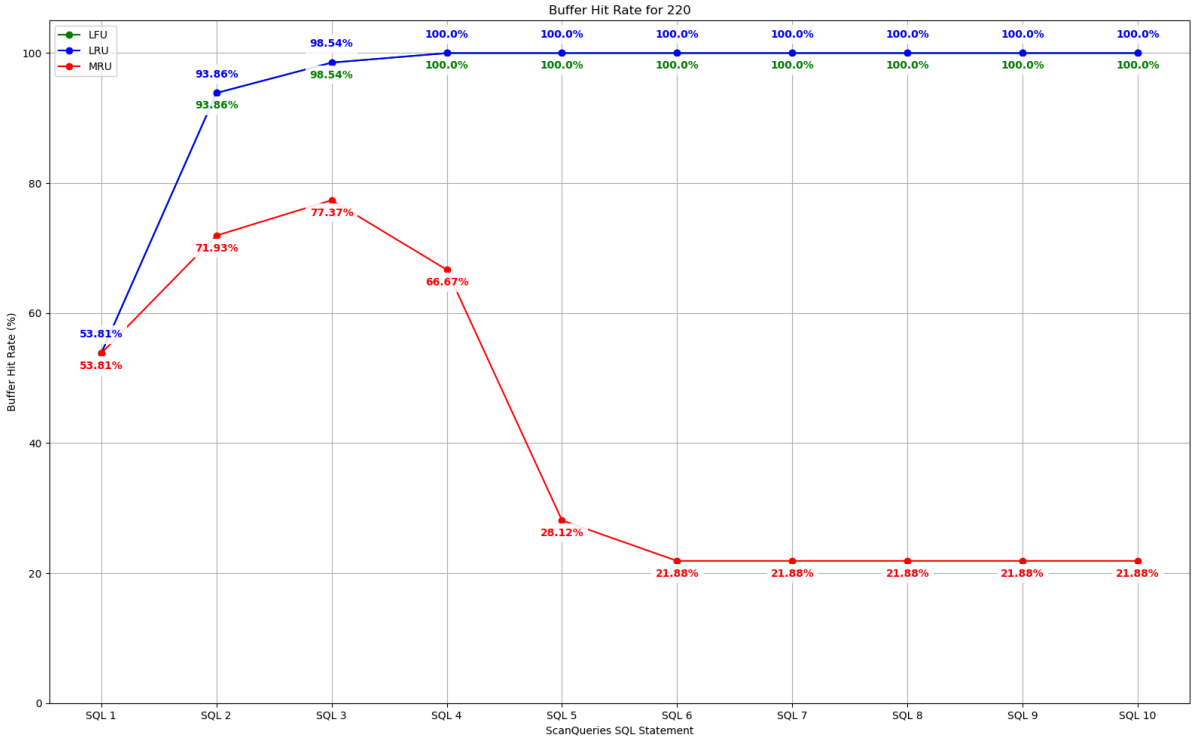


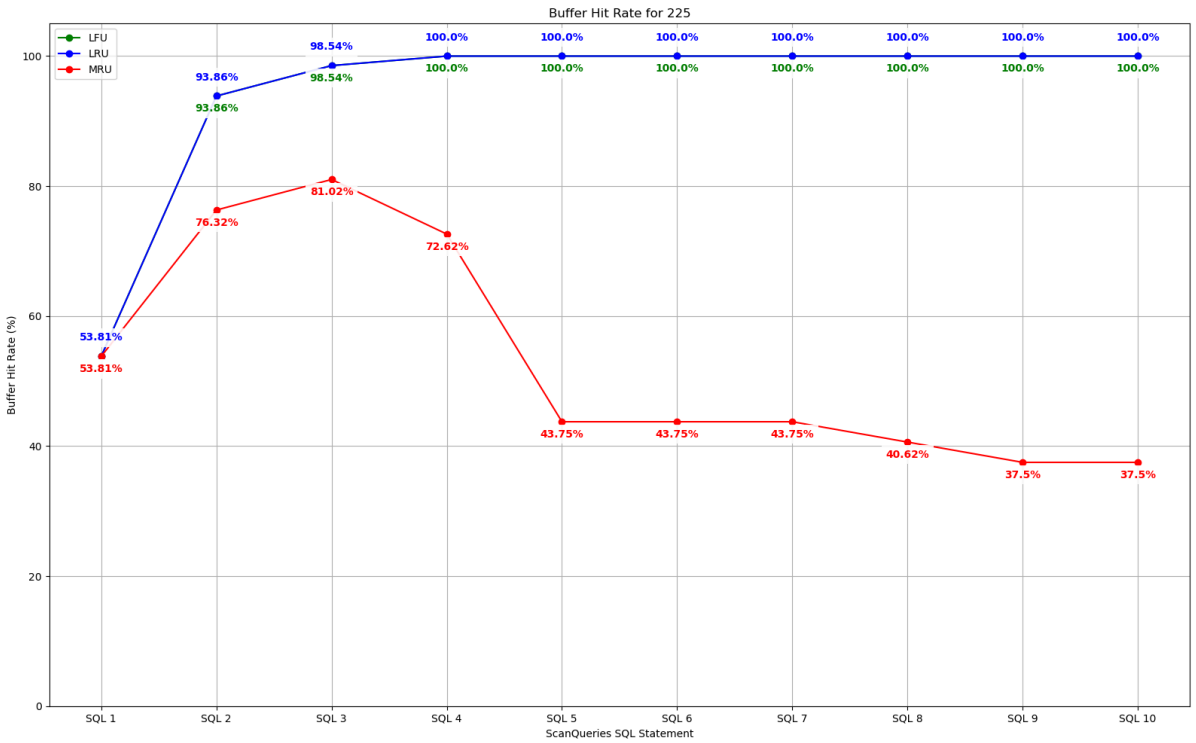

Buffer Hit Rate for 40
LFU
LRU
MRU
100.0%
48.31%
56.14%
66.42%
57.14%
48.31%
43.86%
56.93%
35.71%
26.32%
24.09%
20.76%
21.43%
3.12%
0.0%
SQL 1
SQL 2
SQL 3
SQL 4
ScanQueries SQL Statement

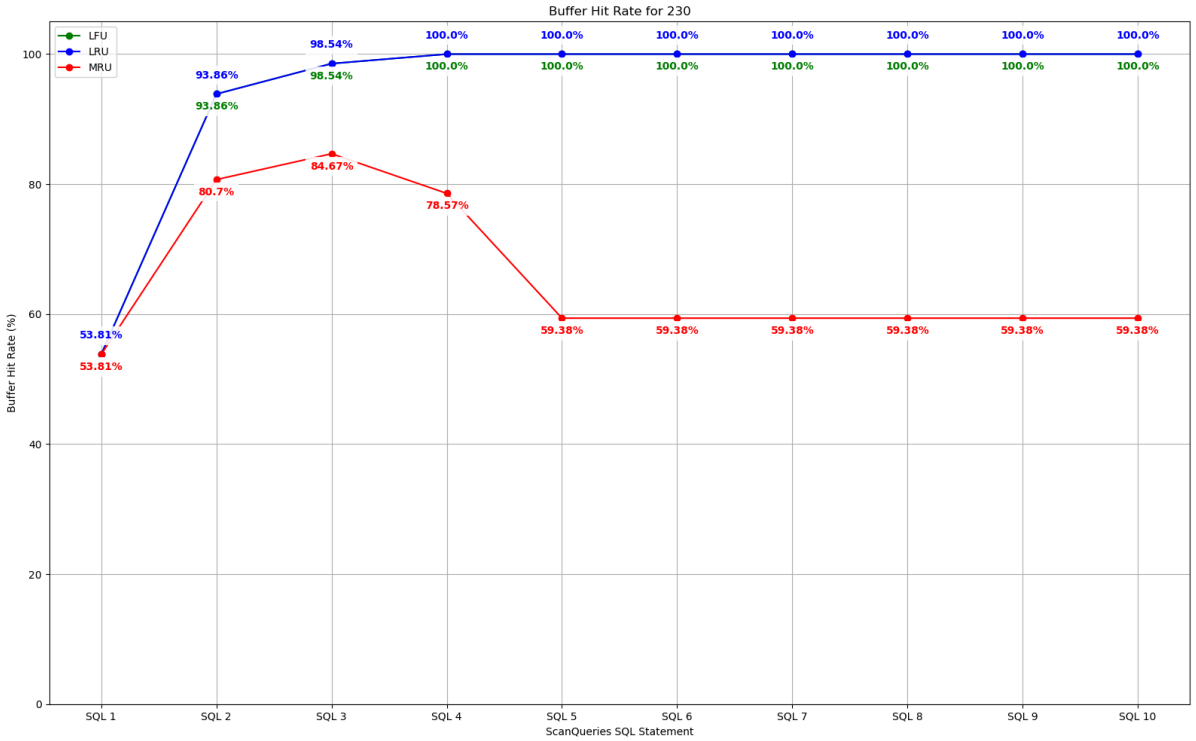

Buffer Hit Rate for 60
LFU
LRU
MRU
100.0%
98.81%
94.16%
52.54%
60.53%
72.99%
58.33%
49.12%
34.21%
25.85%
26.28%
22.62%
6.25%
SQL 1
SQL 2
SQL 3
SQL 4
SQL 5
SQL 6
SQL 7
SQL 8
SQL 9
SQL 10
ScanQueries SQL Statement

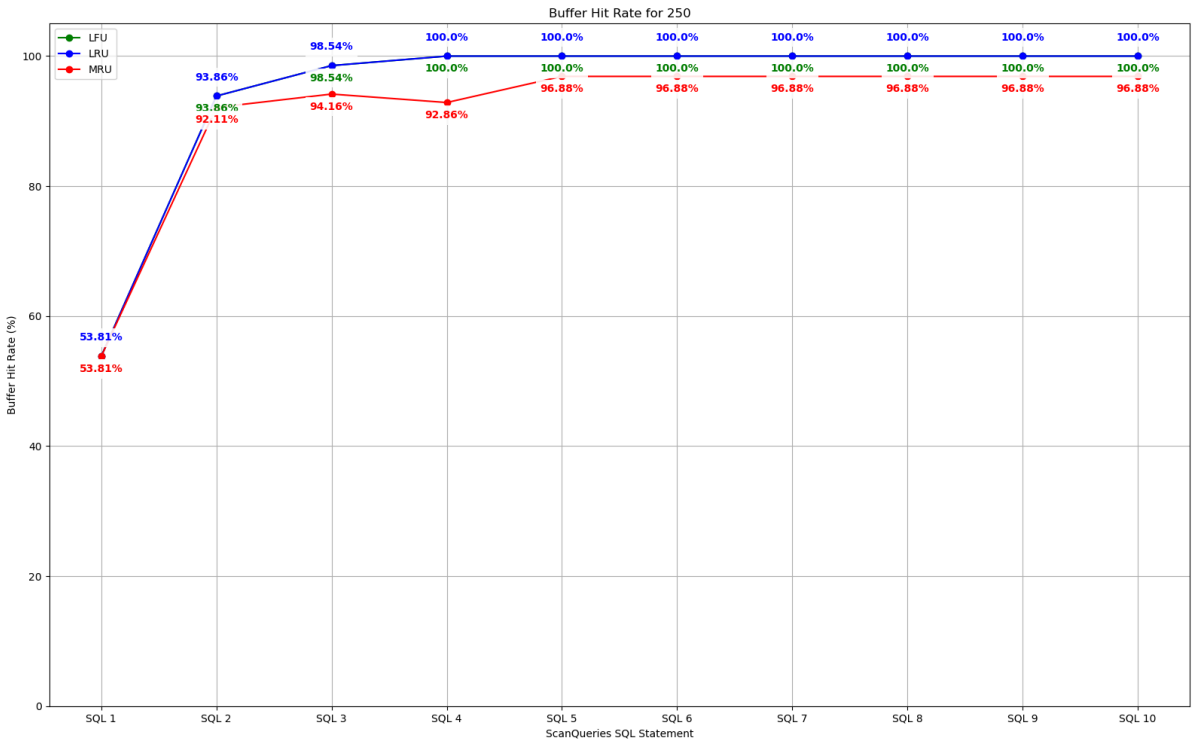


- For MRU, different from LRU and LFU, when the number of buffers reached around 221, the buffer hit rate for the fifth and rest queries started to rise from very low and back to normal.
- Our insight on why they have such a threshold is because a possible sequential flood might happen when the whole buffer pool is full and always cause a buffer hit missing.
- Buffer hit rate of buffer sizes 220, 225, 230 and 250:

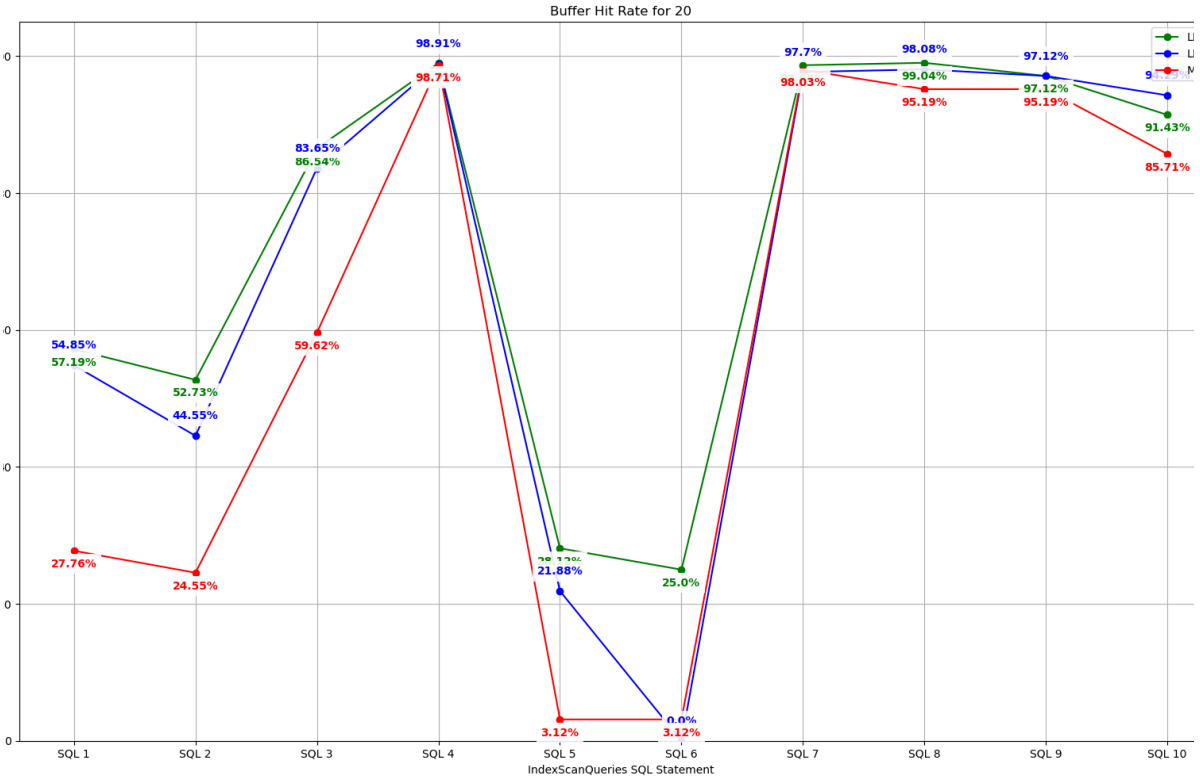


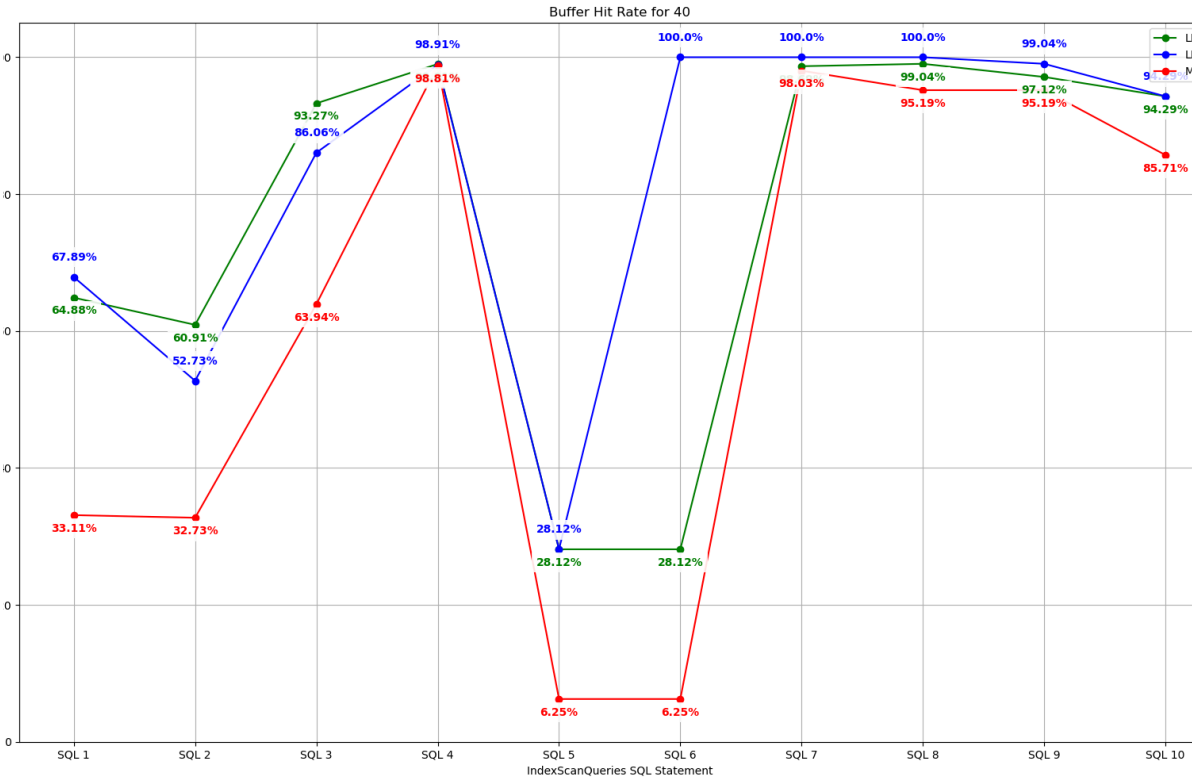

Buffer Hit Rate for 225
LFU
LRU
MRU
Buffer Hit Rate (%)
ScanQueries SQL Statement
SQL 1
SQL 2
SQL 3
SQL 4
SQL 5
SQL 6
SQL 7
SQL 8
SQL 9
SQL 10
53.81%
93.86%
98.54%
100.0%
76.32%
81.02%
72.62%
43.75%
40.62%
37.5%

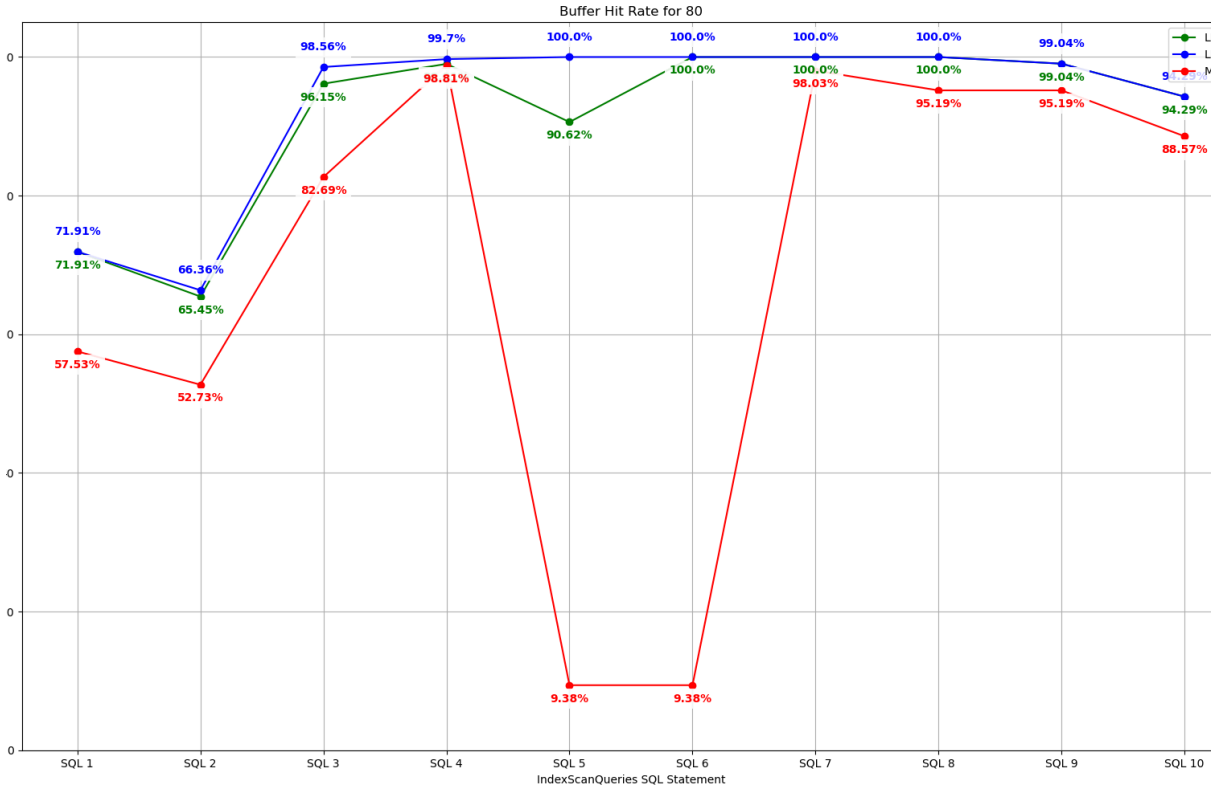

Buffer Hit Rate for 230
LFU
LRU
MRU
Buffer Hit Rate (%)
ScanQueries SQL Statement
SQL 1
SQL 2
SQL 3
SQL 4
SQL 5
SQL 6
SQL 7
SQL 8
SQL 9
SQL 10
53.81%
93.86%
98.54%
100.0%
80.7%
84.67%
78.57%
59.38%



- ***<u>For Sequential SQL Statement of Index Scan Queries:</u>***
- We could see that the buffer hit rate for all three algorithms increases when the number of buffers increases. When the number of buffers is low, there is no big difference between LRU and LFU and MRU is still lower than both of them most time. When the number of buffers reaches 100, the buffer hit rate for LRU and LFU becomes the same. And all buffer hit rate stops at some point.
- For MRU, the average buffer hit rate is still smaller than LRU and LFU but we could see in some queries with a decent buffer size, the buffer hit rate can also be pretty high as well.
- <u>Compared to Scan Queries</u>, the average buffer hit rate of Index Scan Queries is higher for the SQL that uses index scan, which matches our expectations.
- For SQL Statements 2, 5, and 6 which result in also using Seq Scan, the buffer hit rate is significantly lower than the others, as shown in the following graph.


Buffer Hit Rate for 20
54.85%
57.19%
52.73%
44.55%
83.65%
86.54%
98.91%
98.71%
27.76%
24.55%
59.62%
21.88%
3.12%
3.12%
0.0%
25.0%
97.7%
98.03%
98.08%
99.04%
95.19%
97.12%
97.12%
95.19%
91.43%
85.71%
SQL 1
SQL 2
SQL 3
SQL 4
SQL 5
SQL 6
SQL 7
SQL 8
SQL 9
SQL 10
IndexScanQueries SQL Statement

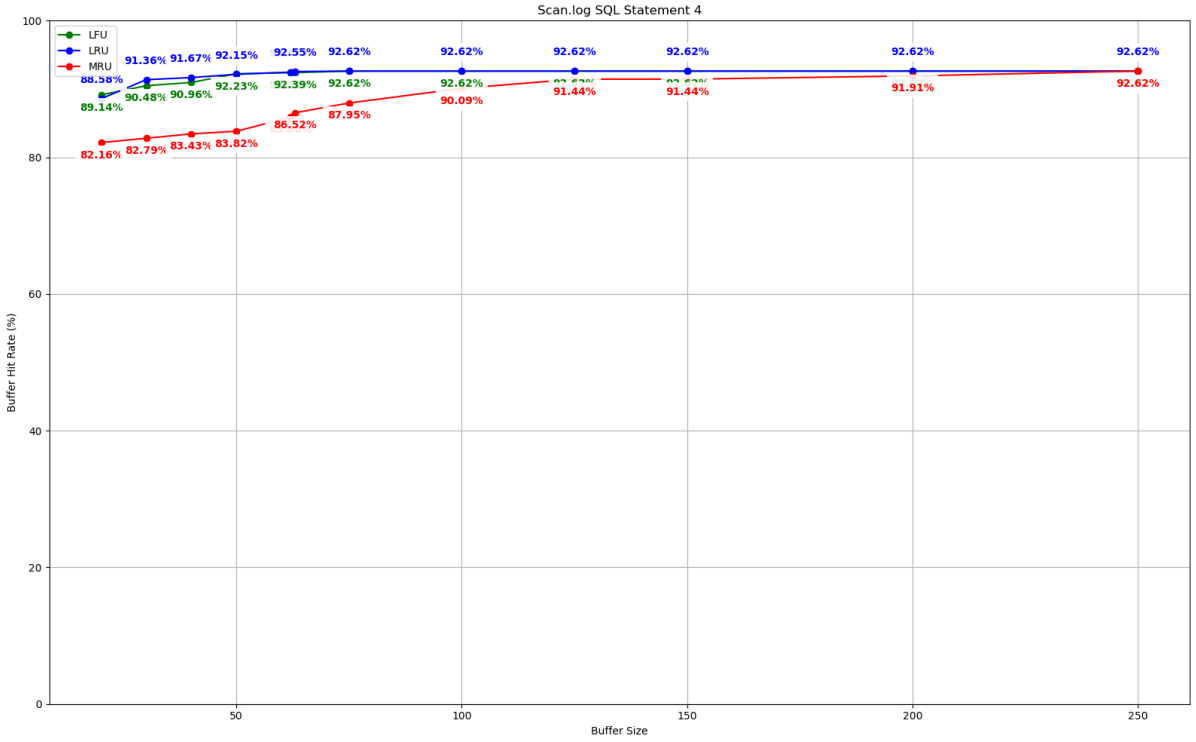

Buffer Hit Rate for 40
67.89%
64.88%
60.91%
52.73%
93.27%
86.06%
98.91%
98.81%
33.11%
32.73%
63.94%
28.12%
28.12%
6.25%
6.25%
28.12%
100.0%
100.0%
100.0%
98.03%
99.04%
95.19%
99.04%
97.12%
95.19%
94.29%
85.71%
SQL 1
SQL 2
SQL 3
SQL 4
SQL 5
SQL 6
SQL 7
SQL 8
SQL 9
SQL 10
IndexScanQueries SQL Statement



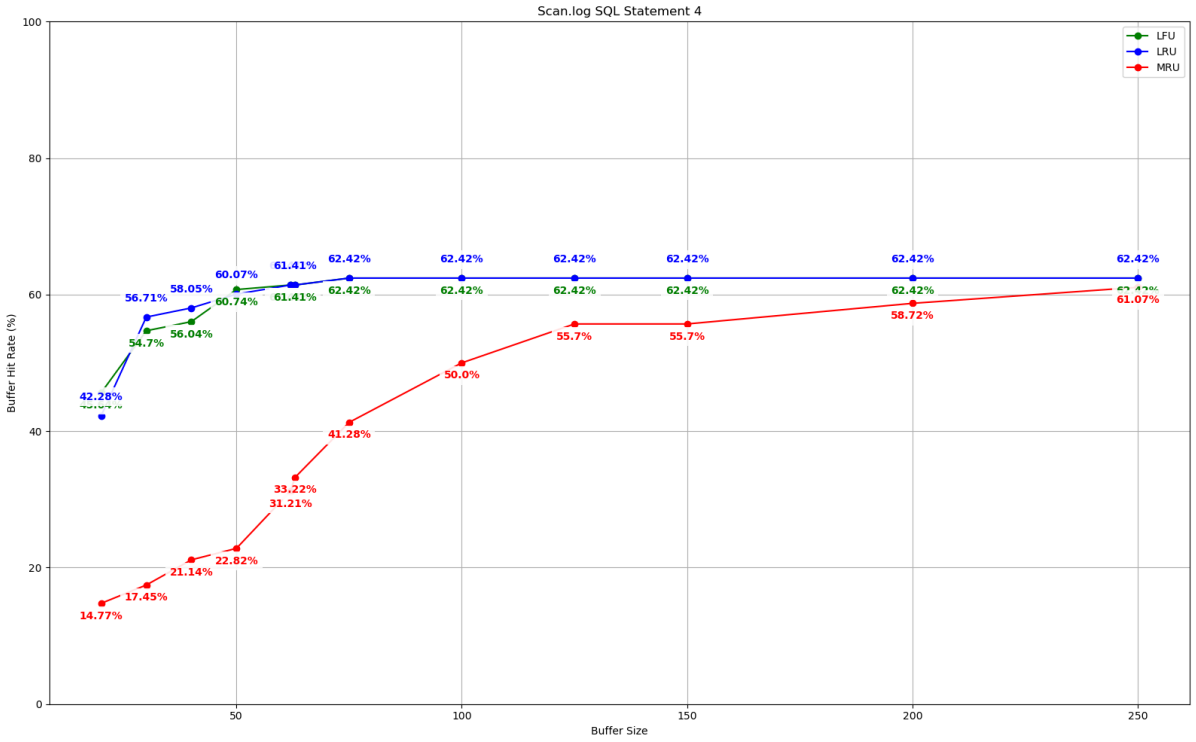
IV. Generate the Data separately

- We also have a set of diagrams to compare each SQL separately. (Run each query individually). The idea of having this diagram is to focus on how will the buffer size affect an SQL statement so each diagram will focus on individually running a SQL statement. The whole result is in the folder /result/individual.png.
- We have found that in the case of good SQL statements (Index Scan Queries SQL Statement 4), the data is generally high both at lower buffer sizes and higher buffer sizes, perhaps even approaching 100%. However, this is not a universal situation. For most of the data, the final result will be distributed between 60% and 80%.
- It is clear that as the buffer size increases, the hit rate of all algorithms increases. The LFU and LRU algorithms improve more significantly in the interval 20 to 40, while MRU grows faster from the interval 50 to 70, where there is always a steep slope between 63 and 64. But overall, MRU does not boost as fast as the LFU and LRU algorithms. It shows a much smoother change in the image.
- If we compare the LFU and LRU algorithms, we find that the hit rate of the LRU algorithm is generally larger than that of the LFU algorithm and that the difference can be observed only for relatively small buffer sizes (less than 50), with almost no difference after larger than this size.
- We also find that usually, the hit rate of IndexScanQueries is significantly higher than that of ScanQueries in all cases.

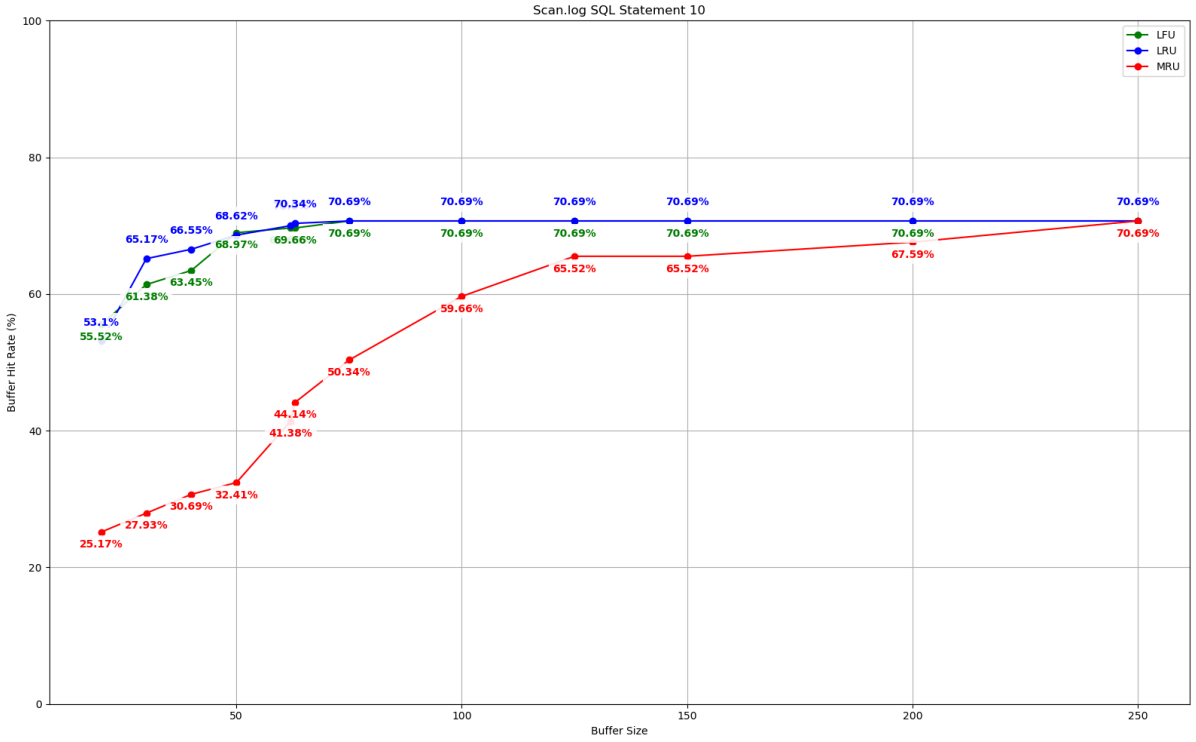
- IndexScanQueries SQL Statement 4



- ScanQueries SQL Statement 4



- IndexScanQueries SQL Statement 10



- ScanQueries SQL Statement 10