Database Systems Implementation

Project 3: "Joins" Report

Candidate number: _____

February 1, 2013

1 Experimental Setup

All experiments were performed on a quad-core Intel Xeon X3450 CPU (8M Cache, 2.66 GHz) with 32 GB RAM. A single execution thread was used (i.e. no parallelism was exploited in the experiments).

Each join for a particular combination of the buffer pool size and the number of records per relation was performed five times. The arithmetic averages of resulting execution times/page miss counts were used in the analysis.

1.1 Measures

For all join configurations, two types of data points were collected:

- elapsed wall-clock running time, and
- buffer page miss count.

Elapsed wall-clock time for an execution of the join operation for a particular configuration was measured using UNIX function `gettimeofday`, in the accuracy of microseconds.

The buffer page miss count was measured using a provided Minibase buffer manager's method `BufMgr::GetStat`.

2 Analysis

The performance of three different nested join algorithms:

- tuple-at-a-time,
- block, and
- index,

was analysed w.r.t. the measures defined above for a changing number of records in relations, and a changing buffer page pool size.

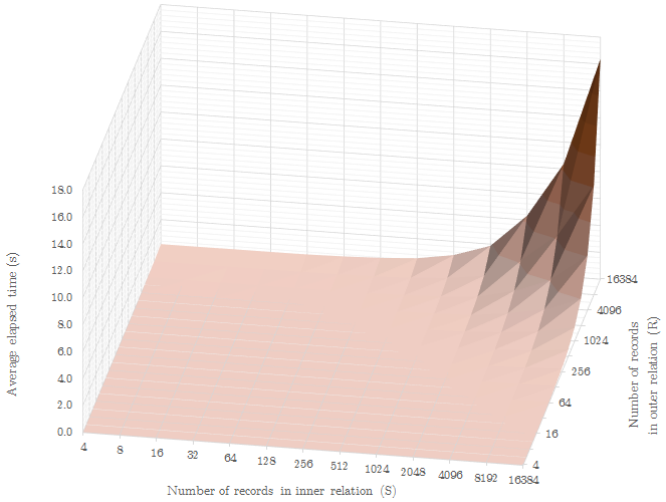
The results are summarized in the following subsections.

2.1 Variable number of records

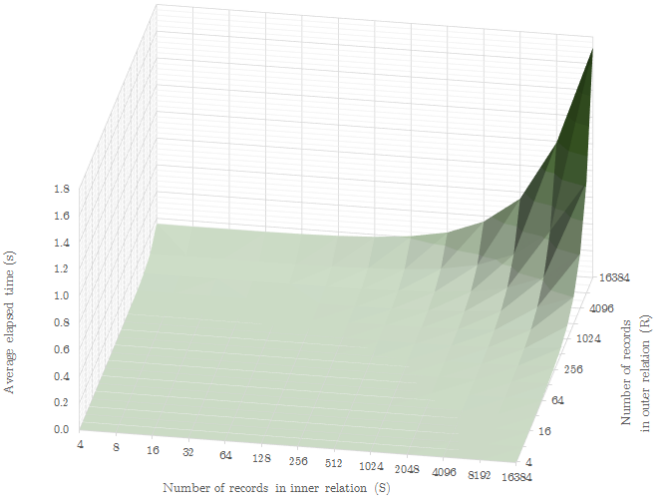
For all three algorithms, the buffer pool size was fixed to 50 pages[1].

The elapsed wall-clock running time for all three algorithms with the record counts changing from 4 to 16,384 entries per relation is shown in Figure 1.

a) Tuple-at-a-time nested join algorithm



b) Block nested join algorithm



In order to better facilitate the comparison between these three algorithms, Figure 3 shows the elapsed

[1] For the block nested loop join algorithm, the size of the simulated "block" was set to `(MINIBASE_BM->numOfUnpinnedBuffers()-3*3)*MINIBASE_PAGESIZE`.

c) Index nested join algorithm

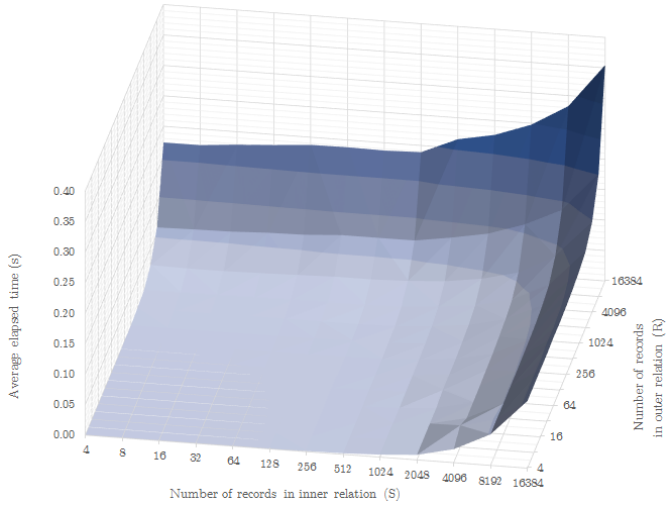


Figure 1: Elapsed wall-clock time for three join algorithms with the record counts changing from 4 to 16,384 entries per relation.

wall-clock running time for a special case when $|R| = 4 \times |S|$.

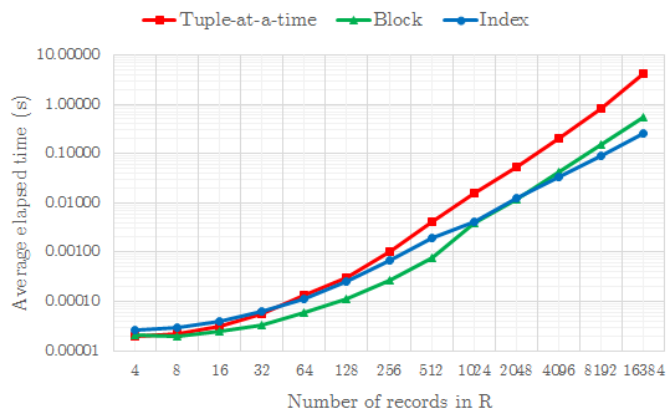


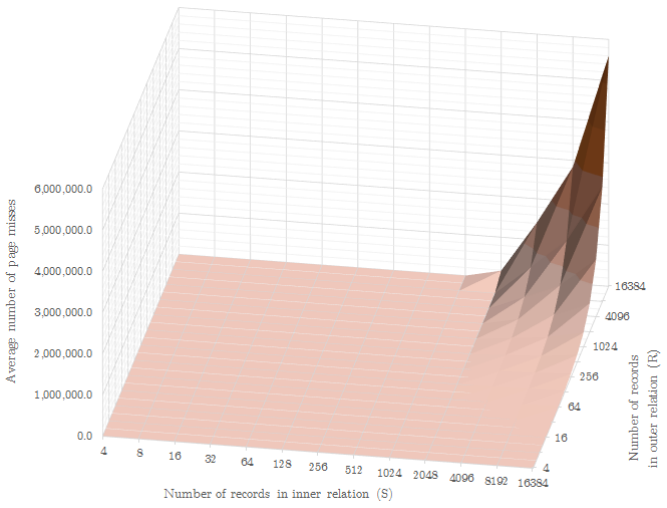
Figure 2: Elapsed wall-clock time for three join algorithms with the changing relation size, for a special case when $|R| = 4 \times |S|$.

Clearly, the *index* and *block* nested loop join algorithms outperform the *tuple-at-a-time* nested loop join algorithm.

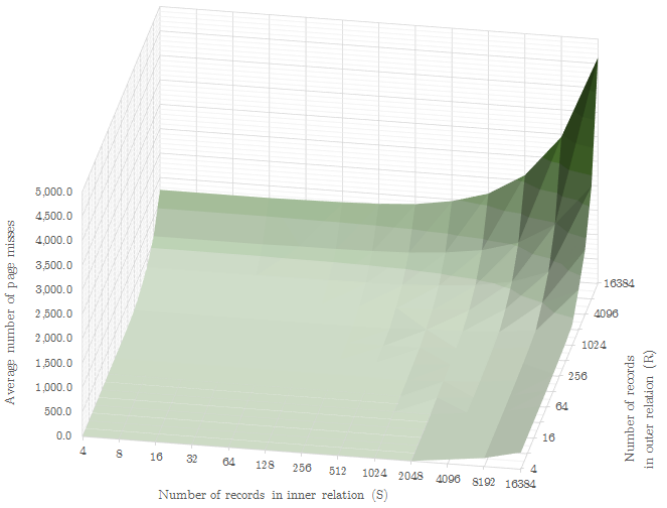
A similar comparison between all three algorithms for the average number of page misses is shown in Figures 3 and 4.

For this measure, *block* nested loop join algorithm outperforms both the *index* and *tuple-at-a-time* algorithms.

a) Tuple-at-a-time nested join algorithm



b) Block nested join algorithm



c) Index nested join algorithm

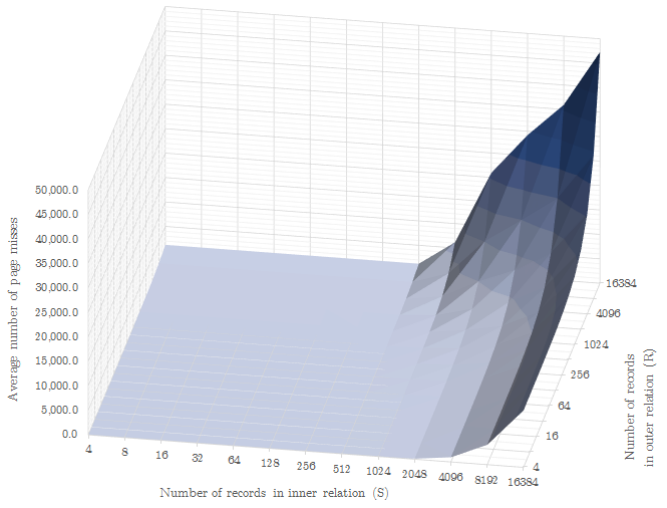


Figure 3: Average number of page misses for three join algorithms with the record sizes changing from 4 to 16,384 entries per relation.

2.2 Variable buffer pool size

The performance of all three algorithms w.r.t. a changing buffer pool size was measured while keeping R and S relation sizes constant at $|R| = 10,000$ and $|S| = 2,500$.

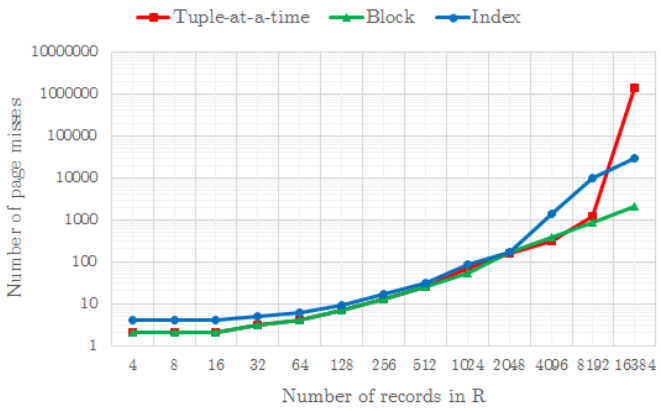


Figure 4: Average number of page misses for three join algorithms with the changing relation size, for a special case when $|R| = 4 \times |S|$.

As can be seen in Figure 5, the *index* and *block* nested join algorithms have clearly better elapsed running times than the *tuple-at-a-time* algorithm.

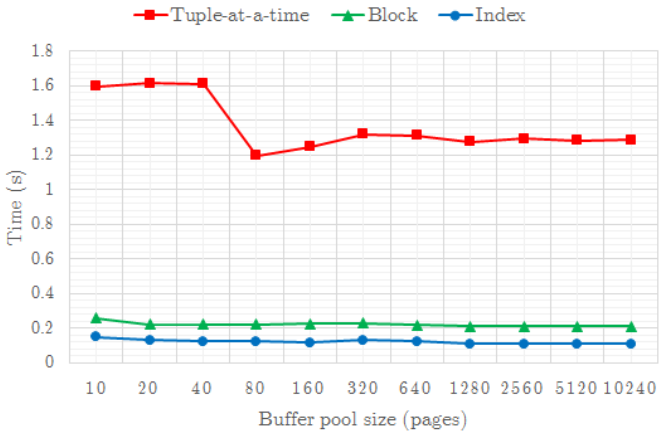


Figure 5: Elapsed wall-clock time for three join algorithms with the changing buffer pool size, while keeping $|R|$ and $|S|$ constant at $|R| = 10,000$, $|S| = 2,500$.

While *index* and *block* nested join algorithms deliver a comparable performance w.r.t. time, *block* algorithm outperforms *index* w.r.t. the number of a page misses in a number pool, as shown in Figure 6.

The performance of a *block* algorithm is further examined in the following subsection.

2.3 Changing block size for *block* algorithm

The performance of a block nested loop join algorithm is analysed by changing the block size, while keeping the buffer pool and relation sizes fixed as follows:

- $|R| = 10,000$,

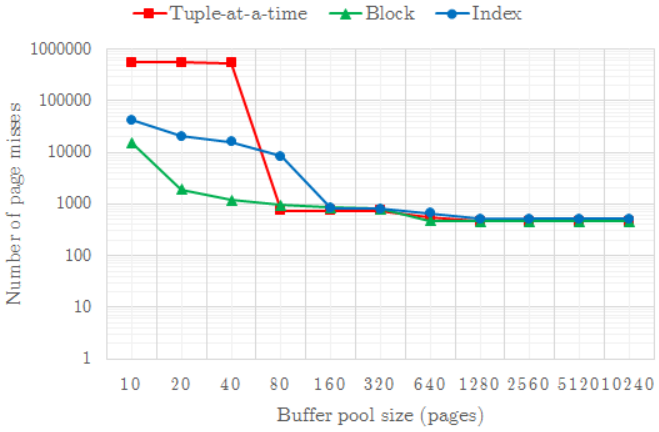


Figure 6: Average number of page misses for three join algorithms with the changing buffer pool size.

- $|S| = 2,500$, and
- buffer pool size = 50 pages.

Rather unsurprisingly, the elapsed running time of the *block* algorithm decreases with the increasing block size, as shown in Figure 7.

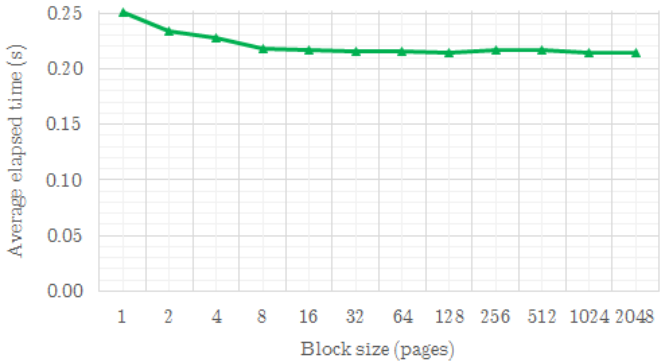


Figure 7: Elapsed wall-clock running time for *block* nested loop join algorithm with the changing block size, while keeping the buffer pool and relation sizes fixed.

Similarly, the number of page misses decreases with the increasing block size, as shown in Figure 8.

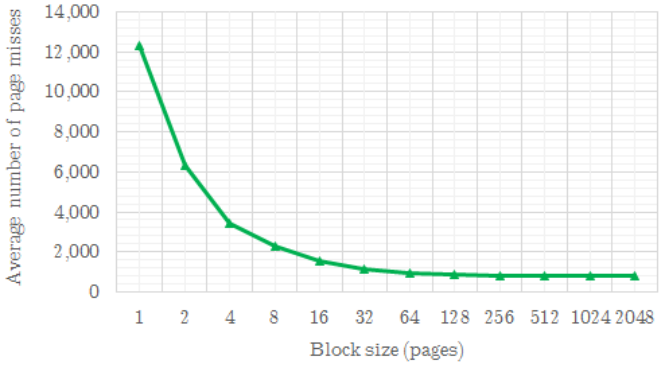


Figure 8: Average number of page misses for *block* algorithm with the changing block size.