WPI

Statistical Analysis of the Effect of Denormalization on Query Resource Consumption and Performance

Group 3: Edith Gomez, Ryan Killea, Agustina Maccio, Nathanael Mercaldo, and Todd Moore

01 Motivation

- The increase of data volumes in organizations demands for designing modern systems with high performance.
- Disk and memory usage, and query execution time have significant effects on the usability and applicability of a DBMS.
- Denormalization: tool in strategic database implementations to boost database performance and reduce query response times.
- Denormalization has not received much attention in academia ☐ lack of in-depth analysis to illustrate the conditions where they are most effective.

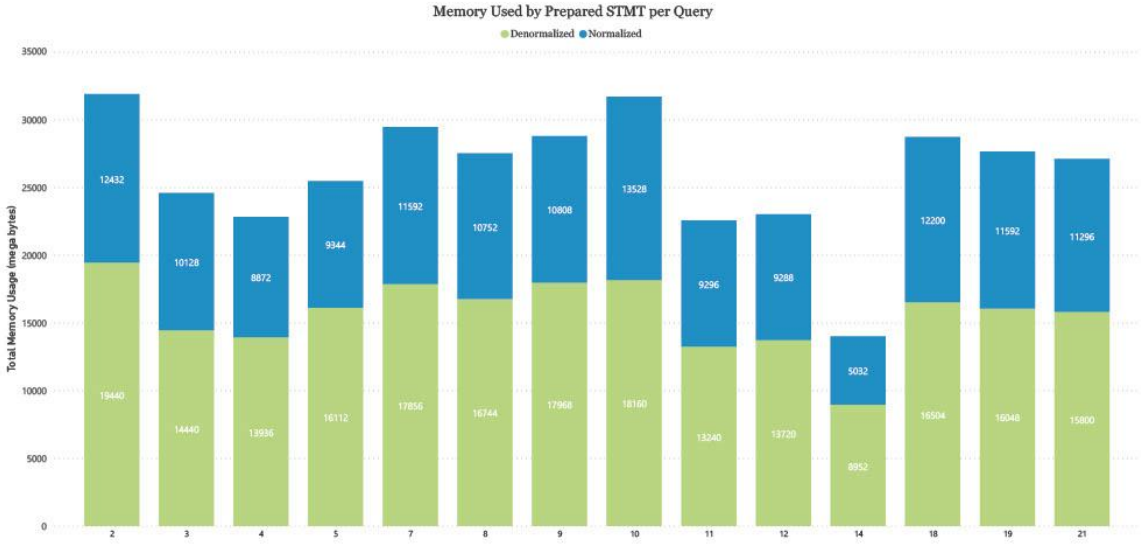
02 Background & Related work

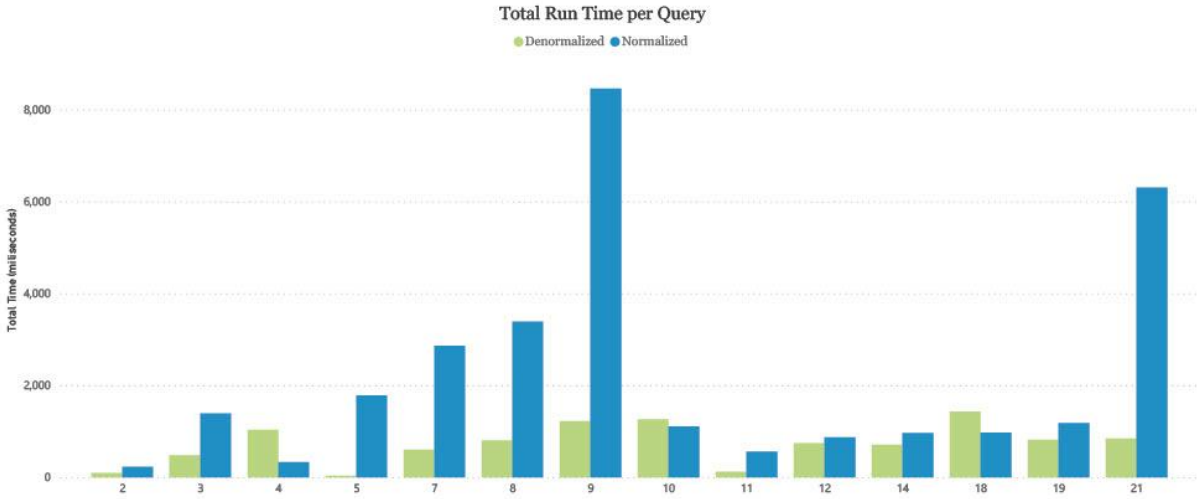
	Normalization	Denormalization
Implementaion	Decomposes data into different tables to reduce redundancy.	Combines data to improve the access time
Query Execution Speed	Speed of update, delete and write operations is higher.	Speed of read operations is higher, but that of update and write operations is slower.
Memory Consumption	Memory consumption is less as data redundancy is less.	Memory consumption is more as redundancy is introduced.
Number of Tables	Number of tables is more on account of decomposition of data.	Combines tables and hence number of tables are less.
Data Integrity	Data integrity is maintained.	Data integrity might not be maintained.

03 Approach

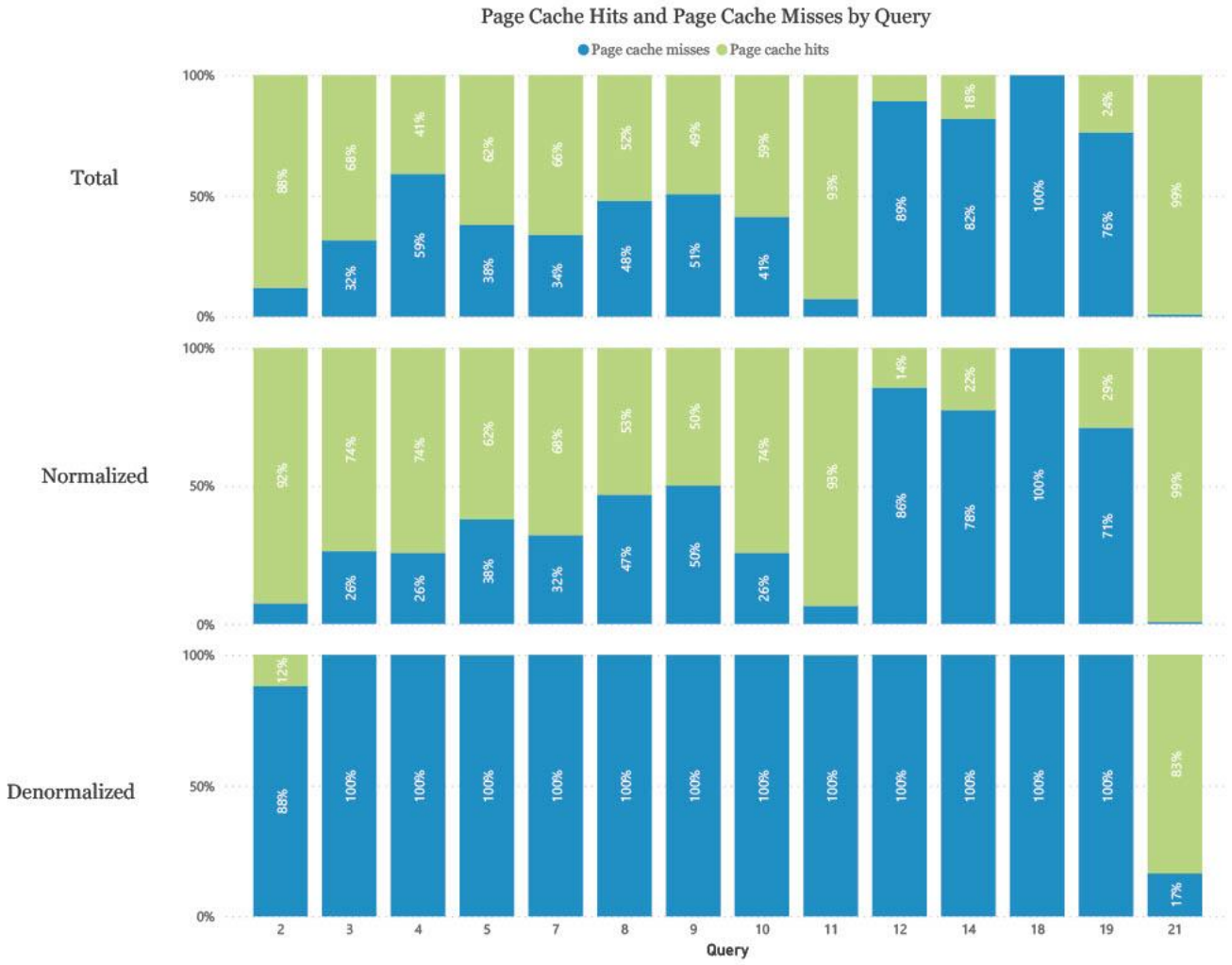
- Selected interesting TPC-H queries which appeared amenable to denormalization based on model technique
- Created denormalized relations and rewrote selected queries to take advantage of new relations - eliminated queries that did not fit model
- Collected query execution statistics using built in SQLite stats logging for both normalized and denormalized queries
- Analyzed statistics to discover interesting differences between denormalized and normalized query groups

04 Analysis


Memory Used by Prepared STMT per Query
Denormalized
Normalized
Total Memory Usage (mega bytes)
35000
30000
25000
20000
15000
10000
5000
0
12432
10128
8872
9344
11592
10752
10808
13528
9296
9288
5032
12200
11592
11296
19440
14440
13936
16112
17856
16744
17968
18160
13240
13720
8952
16504
16048
15800
2
3
4
5
7
8
9
10
11
12
14
18
19
21

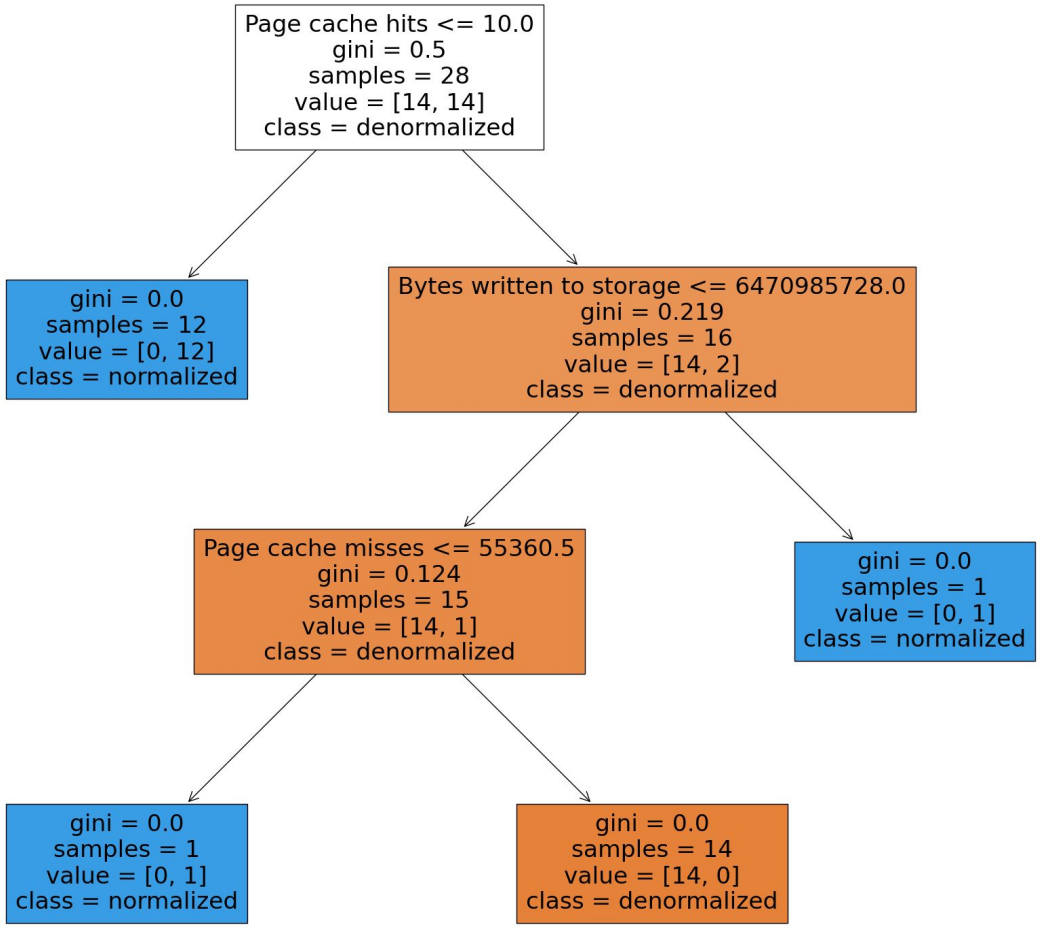

Total Run Time per Query
Denormalized
Normalized
Total Time (milliseconds)
8,000
6,000
4,000
2,000
0
2
3
4
5
7
8
9
10
11
12
14
18
19
21

04 Analysis


Page Cache Hits and Page Cache Misses by Query
Page cache misses
Page cache hits
Total
Normalized
Denormalized
100%
50%
0%
88%
68%
41%
62%
66%
52%
49%
59%
93%
89%
82%
100%
76%
99%
32%
59%
38%
34%
48%
51%
41%
18%
24%
92%
74%
74%
62%
68%
53%
50%
74%
93%
86%
78%
100%
71%
99%
26%
26%
38%
32%
47%
50%
26%
14%
22%
29%
12%
88%
100%
83%
17%
2
3
4
5
7
8
9
10
11
12
14
18
19
21
Query

04 Analysis

- Predict whether query is in denorm or norm class
- Decision tree helped discover more sophisticated relationships
- Page cache as primary class predictor?



05 Conclusions

- Denormalization can provide large performance improvements for queries making use of many joins
- However, there is no free lunch
- Denormalization increases memory and disk usage and may not always be faster. (more rows due to data duplication)
- Also increases chance of page cache misses thus driving up disk load

THANK YOU