Novel Selectivity Estimation Strategy for Modern DBMS
Jun Hyung Shin¹, Florin Rusu¹, Alex Şuhan²
University of California, Merced¹, MapD Technologies, Inc²
jshin33, frusu@ucmerced.edu; alex@mapd.com

Selectivity Estimation Problem

σ_{A=x AND B>y1 AND B<y2 AND (C=z1 OR C=z2)}(R)
Selectivity estimation requires database statistics, such as:
T(R) = # tuples of relation R
V(R, A) = # distinct values relation R has in attribute A

Estimated Selectivity:
T(\sigma_{...}(R)) = T(R) \cdot V(R, A) \cdot \left( \frac{1}{\sqrt{|V(R, C)|}} + \frac{1}{|V(R, C)|} \right)
Assumption: each predicate is independent + uniform data
• Not realistic in real world
• Estimation can be wrong by orders of magnitude
• Results in a bad query execution plan

Estimation is not accurate with complex predicates
Selectivity can be better estimated by other methods:
• Histograms & Sketches
  ▶ better estimation for a single attribute
  ▶ still bad on multiple attributes
• Samples
  ▶ number of attributes does not matter
  ▶ bad on skewed & sparse data

Modern Database Systems

I/O bottleneck when read from disk was critical in the past
Evolution of modern database systems:
• In-memory: much less I/O access
• GPU-accelerated: massively parallel architecture
  better at running aggregate queries

Main Idea

Focus on accuracy:
T(\sigma_{...}(R)) = \text{SELECT COUNT(*) FROM R WHERE A = x AND B < y1 AND B > y2 AND (C = z1 OR C = z2)};
Running this query will return exact selectivity and would not take much time in modern database systems

Effective for complex predicates
Does not require any database synopses and statistics
Consider the trade-off between accuracy and time
• Cost of time is still relatively expensive
• Hybrid approach for better optimization

Selectivity Estimation

MapD: In-memory, GPU-accelerated, column-oriented database system
MapD does not perform selection push-down
• high selectivity → huge materialization → system slows down
• requires accurate selectivity estimation to prevent such selection
Compute exact selectivity by running an extra plan for each selection
• avoids push-down on small relations & push-down irrelevant projection to read only necessary columns
• push-down only if selectivity < PUSH_DOWN_MAX_SELECTIVITY
• execute selection immediately and store the output as temporary table so that the system can reuse it while executing original query

Experiments

TPC-H Benchmark dataset on MapD
2 Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00GHz, 1 Tesla K80 GPU, 8 DDR4 memory 32GB @ 2400 MHZ

Overhead:

<table>
<thead>
<tr>
<th>SF</th>
<th>50</th>
<th>10</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>tables</td>
<td>O</td>
<td>PS</td>
<td>P</td>
</tr>
<tr>
<td>O</td>
<td>21</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>PS</td>
<td>20</td>
<td>23</td>
<td>28</td>
</tr>
<tr>
<td>P</td>
<td>21</td>
<td>20</td>
<td>22</td>
</tr>
<tr>
<td>S</td>
<td>21</td>
<td>22</td>
<td>22</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CPU</th>
<th>70</th>
<th>95</th>
<th>66</th>
</tr>
</thead>
<tbody>
<tr>
<td>57</td>
<td>49</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>21</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>21</td>
<td>23</td>
<td></td>
</tr>
</tbody>
</table>

Selection Push-Down Decision:

by scale factor

by # attributes

by selectivity

by # tables

by # attributes

by complexity

Jun Hyung Shin, Florin Rusu (UC Merced); Alex Şuhan (MapD Technologies, Inc)