Graph Processing on an “almost” relational database

Ramesh Subramonian
Oracle Labs
ramesh.subramonian@oracle.com

Work done while at LinkedIn
Context

• Not responsible for recovery/backup/...
• Data sets needed for specific problem small enough for 1 machine
• No need for fault tolerance
• Analyses performed in reflective mode, not reactive mode
• Problem definition is changing rapidly
Some Sample Problems

1. Second Degree Network
2. Incremental Path Navigation
3. Filtered Endorsements
4. People You Should Know (as opposed to PYMK)
Inspiration
Motivation

• We must not think of the things we could do with, but only of the things that we can’t do without.
• Let your boat of life be light, packed with only what you need.
• You will have time to think as well as to work.

Three Men in a Boat, Jerome K. Jerome
Motivation

• Tables are at a lower level of abstraction than relations
  – they give the impression that positional (array-type) addressing is applicable
    (which is not true of n-ary relations)
  – they fail to show that the information content of a table is independent of row order

• Nevertheless, even with these minor flaws, tables are the most important conceptual representation of relations, because they are universally understood
Inspiration

• Ease of expressing constructs arising in problems
• Suggestivity
• Ability to subordinate detail
• Economy
• Amenability to formal proofs

• *Debugging Support* – *test as you go*
**Second Degree Network – Data Structure**

<table>
<thead>
<tr>
<th>index</th>
<th>Member ID (mid)</th>
<th>TC(_{lb})</th>
<th>TC(_{ub})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>200</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>300</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>index</th>
<th>from</th>
<th>to</th>
<th>to_idx</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100</td>
<td>200</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>300</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>100</td>
<td>0</td>
</tr>
</tbody>
</table>

Bad programmers worry about code. Good programmers worry about data structures.
Second Degree Network – Algorithm

- Find $i$ such that $T_M[i].mid = m$ (fast because $T_M$ is sorted on mid)
  
  $i = \text{"q f_to_s TM mid \"op=[get_idx]:val=[\$m]\""}

- Find range of rows in $T_C$ that contain edges out of $m$
  
  $TC_{lb} = \text{"q f_to_s TM TC_{lb} \"op=[get_val]:idx=[\$i]\""}$
  
  $TC_{ub} = \text{"q f_to_s TM TC_{ub} \"op=[get_val]:idx=[\$i]\""}$

- Create temp table $TD_1$ by copying column $to_{idx}$ for above rows
  
  $q \ copy_fld_ranges TC to_{idx} \"\"$ $TC_{lb}$ $TC_{ub}$ $TD1$
Second Degree Network – Algorithm (contd)

• Repeat previous step for each row of TD$_1$ to create TD$_2$
  – By using field $to_{idx}$ and not field $to$, we avoid searching T$_M$ for each entry of TD$_1$
• Implemented as ``user-defined function''
• De-dupe members in TD$_2$ to create output T$_{out}$
• q mk_uq TD2 mid Tout mid
## Performance Numbers – Second Degree Network

<table>
<thead>
<tr>
<th>Size (1\textsuperscript{st} Degree)</th>
<th>Size (2\textsuperscript{nd} Degree)</th>
<th>Time (msec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>64349</td>
<td>8.070</td>
</tr>
<tr>
<td>263</td>
<td>112213</td>
<td>12.53</td>
</tr>
<tr>
<td>505</td>
<td>334246</td>
<td>41.51</td>
</tr>
<tr>
<td>1021</td>
<td>694644</td>
<td>80.13</td>
</tr>
<tr>
<td>2053</td>
<td>1166594</td>
<td>166.4</td>
</tr>
<tr>
<td>4091</td>
<td>1956817</td>
<td>259.4</td>
</tr>
<tr>
<td>8199</td>
<td>4069339</td>
<td>1363</td>
</tr>
<tr>
<td>16378</td>
<td>8319301</td>
<td>1516</td>
</tr>
</tbody>
</table>