Efficient and Effective Aggregate Keyword Search on Relational Databases

Luping Li, Baidu, Inc., Beijing, China
Stephen Petschulat, SAP Business Objects, Coquitlam, BC, Canada
Guanting Tang, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada
Jian Pei, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada
Wo-Shun Luk, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada

ABSTRACT

Keyword search on relational databases is useful and popular for many users without technical background. Recently, aggregate keyword search on relational databases was proposed and has attracted interest. However, two important problems still remain. First, aggregate keyword search can be very costly on large relational databases, partly due to the lack of efficient indexes. Second, finding the top-k answers to an aggregate keyword query has not been addressed systematically, including both the ranking model and the efficient evaluation methods. In this paper, the authors tackle these two problems to improve the efficiency and effectiveness of aggregate keyword search on large relational databases. They designed indexes efficient in both size and construction time. The authors propose a general ranking model and an efficient ranking algorithm. They also report a systematic performance evaluation using real data sets.

Keywords: Aggregate Keyword Search, Data Cube, Data Mining, Group-By, Relational Database

INTRODUCTION

More and more relational databases contain textual data and thus keyword search on relational databases becomes popular. Aggregate keyword search (Zhou & Pei, 2009) was recently proposed on relational databases: given a set of keywords, find a set of aggregates such that each aggregate is a group-by covering all query keywords.

Aggregate keyword search on relational databases has attracted a lot of attention (Chen, Wang, & Liu, 2011; Ding, Yu, Zhao, Lin, Han, & Zhai, 2010; Ding, Zhao, Lin, Han, & Zhai,
2010; Draper & Smith, 1981; Koren, Zhang, & Liu, 2008; Li, Xu, Lu, & Qian, 2010; Zhou & Pei, 2009). A few critical challenges have been identified, such as how to develop efficient approaches for finding all minimal group-bys (Zhou & Pei, 2009) or top-k relevant cells (Ding, et al., 2010) to a user given keyword query. To motivate, we revisit the example in Zhou and Pei (2009).

**Example 1 (Motivation) (Zhou & Pei, 2009):**

Table 1 shows a database of tourism event calendar. Such an event calendar is popular in many tourism web sites and travel agents’ databases (or data warehouses). To keep our discussion simple, in the field of description, a set of keywords are extracted. In general, this field can store text description of events. Scott, a customer planning his vacation, is interested in seeing space shuttles, riding motorcycle and experiencing American food. He can search the event calendar using the set of keywords {“space shuttle,” “motorcycle,” “American food”}. Unfortunately, the three keywords do not appear together in any single tuple, and thus the results returned by the existing keyword search methods may contain at most one keyword in a tuple.

However, Scott may find the aggregate group (December, Texas, *, *, *) interesting and useful, since he can have space shuttles, motorcycle, and American food all together if he visits Texas in December. The * signs on attributes city, event, and description mean that he will have multiple events in multiple cities with different description.

To make his vacation planning effective, Scott may want to have the aggregate as specific as possible – it should cover a small area (for example, Texas instead of the whole United States) and a short period (for example, December instead of year 2009).

In summary, the task of keyword search for Scott is to find minimal aggregates in the event calendar database such that for each of such aggregates, all keywords are contained by the union of the tuples in the aggregate.

Two problems still remain for aggregate keyword search. First, aggregate keyword search is still costly on large relational databases, partly due to the lack of efficient indexes. For example, the keyword graph index (Zhou & Pei, 2009) is used to generate all aggregate groups for a keyword query. However, it often takes a long time to construct the index on large database and has large space consumption, as demonstrated in Table 2 using some real data sets to be discussed in detail in the upcoming section.

The second problem is that finding the top-k answers to an aggregate keyword query has not been addressed systematically. Since aggregate keyword search on large relational databases may find a large number of answers, ranking the answers effectively becomes important. It is necessary to develop efficient top-k algorithm to find the top-k most relevant

<table>
<thead>
<tr>
<th>Month</th>
<th>State</th>
<th>City</th>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>December</td>
<td>Texas</td>
<td>Houston</td>
<td>Space Shuttle Experience</td>
<td>Rocket, Supersonic, Jet</td>
</tr>
<tr>
<td>December</td>
<td>Texas</td>
<td>Dallas</td>
<td>Cowboy’s Dream Run</td>
<td>Motorcycle, Culture, Beer</td>
</tr>
<tr>
<td>December</td>
<td>Texas</td>
<td>Austin</td>
<td>SPAM Museum Party</td>
<td>Classical American Hormel Foods</td>
</tr>
<tr>
<td>November</td>
<td>Arizona</td>
<td>Phoenix</td>
<td>Cowboy Culture Show</td>
<td>Rock Music</td>
</tr>
</tbody>
</table>
Rule-Based Data Mining Cache Replacement Strategy
[www.igi-global.com/article/rule-based-data-mining-cache/75615?camid=4v1a](www.igi-global.com/article/rule-based-data-mining-cache/75615?camid=4v1a)

Mining for Mutually Exclusive Items in Transaction Databases
[www.igi-global.com/article/mining-mutually-exclusive-items-transaction/1789?camid=4v1a](www.igi-global.com/article/mining-mutually-exclusive-items-transaction/1789?camid=4v1a)