Supporting Ranking and Clustering as Generalized Order-By and Group-By

Chengkai Li (UIUC)

joint work with
Min Wang     Lipyeow Lim     Haixun Wang (IBM)
Kevin Chang (UIUC)
Boolean database queries

Example 1:

```
SELECT SUPPNO, PRICE
FROM QUOTES
WHERE PARTNO = '010002'
AND MINQ <= 1000 AND MAXQ >= 1000;
```

(Relational Algebra and System R papers)
Data Retrieval

21st Century Document Management

SPAM
Example: What Boolean queries provide

```
SELECT * 
FROM Houses H 
WHERE 200K<price<400K AND #bedroom = 4
```
Example: What may be desirable

- < 500K is more acceptable
- but willing to pay more for big house
- close to the lake is a plus
- avoid locations near airport
- …

<table>
<thead>
<tr>
<th>query semantics</th>
<th>results organization</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Boolean query</strong></td>
<td></td>
</tr>
<tr>
<td>(True or False)</td>
<td></td>
</tr>
<tr>
<td><strong>fuzzy retrieval</strong></td>
<td></td>
</tr>
<tr>
<td>“soft” constraints</td>
<td></td>
</tr>
<tr>
<td>(preference,similarity,relevance,</td>
<td></td>
</tr>
<tr>
<td>…)</td>
<td></td>
</tr>
</tbody>
</table>

- a flat table
- too many (few) answers
- a ranked list
- a grouping of results
- etc.
“Retrieval” of DATA:
From Boolean query to fuzzy retrieval

retrieval engine

search engine

Google™  YAHOO!™  Ask.com™

data

text
Retrieval mechanisms: Learning from Web search

- Ranking
- Clustering
- Navigation Map
- Facets
- Categorization
Generalizing SQL constructs for data retrieval
From crispy ordering to fuzzy ranking

- Crispy ordering
  - Order By
    - Houses.size, Houses.price
  - by attribute values
  - equality of values
  - order ≠ desirability

- Fuzzy ranking
  - Order By
    - Houses.size - 4*Houses.price
  - by ranking function
  - combine matching criteria
  - order => desirability : top-k
From crispy grouping to fuzzy clustering

- **Crispy grouping**
  - Group By
    - Houses.size, Houses.price
  - by attribute values
  - equality partition
  - no limit on output size

- **Fuzzy clustering**
  - Group By
    - k-means(H.size, H.price)
  - by distance function
  - proximity of values
  - number of clusters
  - Into
    - 5
Need for combining ranking with clustering

- Clustering-only
  - A group can be big
    “too many answers” problem persists
  - How to compare things within each group?

- Ranking-only
  - Lack of global view
    top-k results may come from same underlying group
    (e.g., cheap and big houses come from a less nice area.)
  - Different groups may not be comparable
Contributions

- **Concepts**
  - generalize Group-By to fuzzy clustering, parallel to the generalization from Order-By to ranking
  - integrate ranking with clustering in database queries

- **Efficient processing framework**
  - summary-based approach
Related works

- Clustering
  - not on dynamic query results
  - use summary (grid with buckets)
    (e.g., STING [WangYM97] WaveCluster [SheikholeslamiCZ98])
- Ranking (top-k) in DB: many instances (e.g., [LiCIS05])
  - use summary (histogram) in top-k to range query translation [ChaudhuriG99]
- Categorization of query results [ChakrabartiCH04]
  - different from clustering
  - no integration with ranking
  - focus on reducing navigation overhead, not processing
- Web search and IR (e.g., [ZamirE99] [LeuskiA00])
Integrate the two generalizations

Boolean conditions

Ranking

Clustering

semantics

evaluation
**Query semantics: ClusterRank queries**

```
SELECT * 
FROM Houses 
WHERE area="Chicago" 
GROUP BY longitude, latitude 
ORDER BY size - 4*price 
INTO 5 
LIMIT 3
```

**Semantics: order-within-groups**
Return the top k tuples within each group (cluster).
Several notes

- **Non-deterministic semantics**
  - clustering is non-deterministic by nature
  - sacrificing the crispiness of SQL queries
  - worthy for exploring the fuzziness in data retrieval?

- **Language syntax isn’t our focus**
  - current SQL semantics: *order-among-groups*
    - Select… From… Where…Group By… Order By…(RankAgg[LiCl06])
  - OLAP function

- **Clustering function**
  - algorithm, distance measure hidden behind

- **Other semantics**
  - e.g., cluster the global top k
Query evaluation:
Straightforward Materialize-Cluster-Sort approach

```
SELECT * 
FROM Houses 
WHERE area="Chicago" 
GROUP BY longitude, latitude INTO 5 
ORDER BY size – 4*price LIMIT 3 
```
Query evaluation: Straightforward Materialize-Cluster-Sort approach

- **Overkill:**
  cluster and rank all, only top 10 in each cluster are requested

- **Inefficient:**
  - fully generate Boolean results
  - clustering large amount of results is expensive
  - sorting big group is costly
Query evaluation:
Summary-driven approach

- use summary to cluster
- use summary for pruning in ranking
- use bitmap-index
  - to construct query-dependant summary
  - to bring together Boolean, clustering, and ranking
Summary for clustering

- K-means on original tuples
- Weighted K-means on virtual tuples
ORDER BY size - 4 *price
LIMIT 3
Construct summary by bitmap-index

The advantages of using bitmap index:

- Small
- Bit operations (&, |, ~, count) are fast
- Easily integrate Boolean, clustering, and ranking

<table>
<thead>
<tr>
<th>TID</th>
<th>size(10,20]</th>
<th>-4*price[-40,-30]</th>
<th>&amp;</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>t2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>t3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>t4</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>t5</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

![Diagram showing TIDs and price values]
Integrating Boolean, clustering, and ranking

Vec(area="chicago")

00111111…

longitude

latitude

Vec(cluster1)

00101001…

longitude

latitude

Vec(cluster2)

00010110…

longitude

latitude

-4*price

size
Experiments

- **ClusterRank** (summary-driven approach) vs. **StraightFwd** (materialize-cluster-rank)
  - Processing efficiency: \( \text{ClusterRank} \gg \text{StraightFwd} \)
  - Clustering Quality: \( \text{ClusterRank} \approx \text{StraightFwd} \)

- synthetic data
- various configuration parameters
  (#tuples, #clusters, #clustering attr, #ranking attr, #paritions per attr, k)
Efficiency

- **ClusterRank** vs **StraightFwd**

**Execution Time (sec.)**

- **t**: #clusters
- **s**: #tuples

4M tuples, 5 clustering attr, 3 ranking attr, top 5

10 clusters, 5 clustering attr, 3 ranking attr, top 5
Clustering quality

close(res_SF, res_CR): closeness of results from StraightFwd and ClusterRank

close(res_SF, res_SF): closeness of results from different runs of StraightFwd

t: #clusters
4M tuples, 8 clustering attr

s: #tuples
10 clusters, 3 clustering attr
Conclusions

- Borrow innovative mechanisms from other areas to support data retrieval applications

- Ranking and clustering as generalized Order-By and Group-By, integrated in database queries

- Query semantics: ClusterRank queries

- Query evaluation: summary-driven approach vs. materialize-cluster-sort
  - evaluation efficiency: ClusterRank $>>$ StraightFwd
  - clustering quality: ClusterRank $\approx$ StraightFwd
Acknowledgement

- Rishi Rakesh Sinha: source code of bitmap index
- Jiawei Han: discussions regarding presentation
Alternative semantics?

- **global clustering / local ranking (focus of this paper)**
  - clustering: Boolean results
  - ranking: local top k in each cluster

- **local clustering / global ranking**
  - clustering: global top k
  - ranking: Boolean results

- **global clustering / global ranking**
  - clustering: Boolean results
  - ranking: in each cluster, return those belonging to global top k

- **rank the clusters? (by average of local top k?)**
Join queries

- Star-schema
  fact table, dimension tables, key and foreign key
- Bitmap join-index
  index the fact table by the attributes in dimension tables