Introduction

- Georgia Koutrika, Alkis Simitis, Yannis E. Ioannidis: Précis: The Essence of a Query Answer. ICDE 2006
- Kostas Stefanidis, Marina Drosou, Evaggelia Pitoura: PerK: Personalized Keyword Search in Relational Databases through Preferences. EDBT 2010
Introduction

- A lot of information is stored in relational databases.
- Many users need to access it.
- Many of them do not have knowledge about the database schema or complicated queries.
- They need to get results from these databases related to their goals and final intentions.

SELECT Customer.Name From ...
How these users can get results related to their final goals from a huge amount of data?

**PERSONALIZED KEYWORD SEARCH**
Keyword-Based Search

- It is very popular
  It allows users to discover information without knowing the database schema or complicated queries.

- Basic idea in relational databases
  In the database, locate tuples that can be joined together, that contain the **KEYWORDS**.

- Need for ranking results
  A huge volume of information may be returned. Just a part of it is useful for the users.
Keyword Search in RDB-Preliminaries

- Dependencies in the database schema
- R: A database with n relations $R_1, R_2, \ldots, R_n$.
- $G_D$: The schema graph capturing the foreign key relationships in the schema. $G_D$ has one node for each relation $R_i$.
- $R_i \rightarrow R_j$: an edge, if and only if, $R_i$ has a set of foreign key attributes referring to the primary key attributes of $R_j$. 

Answering keyword queries.
Keyword Search in RDB-Preliminaries

- $W$: the potentially infinite set of all keywords.

- $Q$: A keyword query that consists of a set of keywords ($Q \subseteq W$).

- JTT: Joining Tree of Tuples. It is a tree of tuples $T$, such that, for each pair of adjacent tuples $t_i, t_j$ in $T$, $t_i \in R_i$, $t_j \in R_j$, there is an edge $(R_i, R_j)$ in the graph.

- Total JTT: A JTT $T$ is total for a keyword query $Q$, if and only if, every keyword of $Q$ is contained in at least one tuple of $T$. 
Minimal JTT: A JTT T that is total for a keyword query Q is also minimal for Q, if and only if, we cannot remove a tuple from T and get a total JTT for Q.

The size of a JTT is equal to the number of its tuples, i.e. the number of nodes in the tree, which is one more than the number of joins.

QUERY RESULT: Given a keyword query Q, the result Res(Q) is the set of all JTTs that are both total and minimal for Q.
Q = \{Matrix, Tampere\}

**MOVER**

<table>
<thead>
<tr>
<th>mid</th>
<th>year</th>
<th>title</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>1999</td>
<td>Matrix</td>
</tr>
<tr>
<td>m2</td>
<td>2001</td>
<td>Beautiful Mind</td>
</tr>
<tr>
<td>m3</td>
<td>2003</td>
<td>Finding Nemo</td>
</tr>
<tr>
<td>m4</td>
<td>2010</td>
<td>The Black Swan</td>
</tr>
</tbody>
</table>

**PLAY**

<table>
<thead>
<tr>
<th>mid</th>
<th>tid</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>t1</td>
<td>23-11-2016</td>
</tr>
<tr>
<td>m1</td>
<td>t2</td>
<td>24-11-2016</td>
</tr>
<tr>
<td>m2</td>
<td>t3</td>
<td>27-11-2016</td>
</tr>
<tr>
<td>m3</td>
<td>t3</td>
<td>28-11-2016</td>
</tr>
<tr>
<td>m4</td>
<td>t4</td>
<td>29-11-2016</td>
</tr>
</tbody>
</table>

**THEATER**

<table>
<thead>
<tr>
<th>tid</th>
<th>region</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>Tampere</td>
<td>Cine Atlas</td>
</tr>
<tr>
<td>t2</td>
<td>Tampere</td>
<td>Plevna</td>
</tr>
<tr>
<td>t3</td>
<td>Helsinki</td>
<td>Tennispalatsi</td>
</tr>
<tr>
<td>t4</td>
<td>Pori</td>
<td>Promenadi</td>
</tr>
</tbody>
</table>
Results in keyword queries may include a large number of JTTs (a huge volume of data may be returned).

Some results may not be relevant for the users.

1. Rank JTTs based on their **relevance** to the query, based on
   - JTT size (the number of nodes in the tree).
   - The importance of the tuples of the JTT (assign scores to JTTs).

2. Rank JTTs based on **user preferences**.
Personalized Keyword Search

The result in keyword queries may include a large number of JTTs.

Incorporating user preferences to personalize results.

- Personalization of user information.
- Results more related to the interests of the user.
- Explicit and implicit information.
- Preferences saved into a user profile.
- User-specific or domain requirements.
Solutions

- **Algorithms for incorporating preferences and optimizing results**
  - Ranking results by relevance and preferences
  - Effectiveness evaluated with a set of experiments

- **Changes in framework and system architecture (Koutrika et al. 2006)**
  - Enables the extraction of structured data
  - Near natural-language representation of results

- **Result diversification (Stefanidis et al. 2010)**
  - k most representative results
  - Cover different preferences of the user
  - Have different content
Solutions

- Koutrika et al. 2006
  - Weights assigned to edges of the database graph.

- Stefanidis et al. 2010
  - Strict partial ordering based on pairwise preferential order.
  - Ordering keywords based on user preferences.
Koutrika et al. (2006)
Weights assigned to edges of the database graph

**Weights.** A weight, \( w \in [0,1] \), is assigned to each edge of the graph showing the significance of the bond between the corresponding nodes.

**Movie Database Schema:**

- **THEATRE** (tid, name, phone, region)
- **PLAY** (tid, mid, date)
- **MOVIE** (mid, title, year, did)
- **GENRE** (mid, genre)

**Movie Database Graph:**

- \( w = 1 \) strong relationship: if one node appears in an answer, then the other node appears as well.
- \( w = 0 \), occurrence of one node in an answer does not imply occurrence of the other one.
A directed join edge expresses the dependence of the left part of the join on the right part.

- Movies and genres are related but one may consider that genres are more dependent on movies than the other way around.
- For instance, the weight of the edge from GENRE to MOVIE is 1, while the weight of the edge from MOVIE to GENRE is 0.9.

---

Koutrika et al. (2006)

---

answer regarding a genre

always contain

related movies

answer regarding a movie

not necessary contain

related genre
Using different weights on graph’s edges allows constructing different answers to the same query.

Weights may be set by the user at query time using an appropriate user interface. Enables interactive exploration of the contents of a database.

Users may explore different regions of the database.

Sets of weights may be created by a designer targeting different groups of users.

For instance, reviewers and cinema fans access a movie database.

In-depth, prefer detailed answers  Prefer shorter answers
Multiple sets of weights corresponding to different user profiles may be stored in the system.

User-specific weights allow generating personalized answers.

different users may see different answers to the same query

User is extremely interested in the genre when he gets movies from the query answer. He prefers detailed answers.

User is extremely interested in the date when he gets movies from the query answer. He prefers shorter answers.

Koutrika et al. (2006)
Information about MOVIE, GENRE, PLAY, THEATER in his query answer. Incorporating implicit information.

Just Information about MOVIE and DATE.
Criteria for the quality of results
- Relevance and degree of preference (as traditional)
- Coverage of the set of results
  - Percentage of choice keywords appearing in the results
- Diversity of the set of results
  - Jaccard distance used to measure dissimilarity between two results
  - Diversity is defined as the average distance over all pairs of results

Generate results in order of indirect dominance
- Query is extended to include user-specific preferences
- For example, if Q = {western} and user has a preference ({western}, C. Eastwood ≻ J. Wayne), results prioritize Eastwood’s films over Wayne’s.
Stefanidis et al. (2010)
Ordering keywords based on user's preferences

- Ordering of preferences
  - Strict partial ordering
  - For example, consider preferences for $Q = \{\text{western}\}$:
    - $\text{cp1} = (\{\text{western}\}, \text{C. Eastwood} \succ \text{J. Wayne})$
    - $\text{cp2} = (\{\text{western}\}, \text{C. Eastwood} \succ \text{R. Redford})$
    - $\text{cp3} = (\{\text{western}\}, \text{E. Flynn} \succ \text{R. Redford})$
    - $\text{cp4} = (\{\text{western}\}, \text{G. Hackman} \succ \text{K. Costner})$
    - $\text{cp5} = (\{\text{western}\}, \text{J. Wayne} \succ \text{C. Lloyd})$
    - $\text{cp6} = (\{\text{western}\}, \text{R. Redford} \succ \text{C. Lloyd})$
Stefanidis et al. (2010)

- **Winnow operator**
  - Retrieves the most preferable choices at each level

- **Sharing-results algorithm**
  - Avoids redundant computation

- An algorithm that works with both winnow operator and sharing-results algorithm to produce the top-k representative results
Next week: Diving deeper and discussing drawbacks

Thank you!