Database Query Approaches to overcome the Tradeoff between Usability and Expressiveness

Simon Gottschalk
Wadim Ortlieb

10/07/2014
Problems and Motivation

Tradeoff between expressiveness and usability

- **Usability**
  - Easy to use
  - Complicated

- **Expressiveness**
  - Less expressive
  - More expressive

**Keyword search**
- possibly imprecise results
- BANKS, DBXPloerer, Discover ('02)

**Structured queries**
- language, schema
- SQL, SPARQL, XQuery
- OBE ('75), NLQ ('99)

**Goal:** Expressive AND Easy to use
Problems and Motivation

- Users don’t know
  - database schema
  - query language

- But users are familiar with
  - Google’s search field
  - forms
Approaches

(1) combine keyword search and form-based interfaces

(2) provide a form that the user can modify

(3) iteratively ask the users questions to find out what information he is searching for
Example database

- Flight
  - flightNumber
  - origin
  - destination
  - duration
  - distance
  - airlineName

- Seat
  - seatNumber
  - flightNumber
  - price
  - fareClass
  - available

- Airline
  - name
  - url
Keyword Search - Example

paris london seats

Which seats are in which flights

- Basic Form
- Aggregation

Flights with their airline

<table>
<thead>
<tr>
<th>Flight</th>
<th>Seat</th>
</tr>
</thead>
<tbody>
<tr>
<td>origin</td>
<td>price</td>
</tr>
<tr>
<td>destination</td>
<td>fareClass</td>
</tr>
<tr>
<td>duration</td>
<td>available</td>
</tr>
</tbody>
</table>
Skeleton Template Generation

Generate a set of *skeleton templates* with short descriptions

1) entities without foreign keys
   - “Details about an airline“
     \[ \sigma_{\text{name } op \text{ value } \land \text{url } op \text{ value}} \text{AIRLINE} \]

2) entities referencing other entities
   - “Which seats are in which flights“
     \[ \sigma_{\text{price } op \text{ value } \land \ldots \land \text{origin } op \text{ value } \land \ldots} (\text{SEAT } \bowtie \text{FLIGHT}) \]

3) non-foreign key equi-joins
Form Generation

- generate SQL queries from skeleton templates and map them to forms
- four query classes
  - Simple SELECT
  - Aggregation
  - GROUP BY (+ Aggregation)
  - UNION/INTERSECT
Example of Form Generation

**Skeleton template**

\[ \sigma_{\text{price } op \text{ value } \land \ldots \land \text{origin } op \text{ value } \land \ldots} (\text{SEAT } \bowtie \text{FLIGHT}) \]

**SQL template – query class „Aggregation“**

```
SELECT COUNT(*)
FROM Seat s
JOIN Flight f USING(flightNumber)
WHERE f.origin op value AND ...
AND s.price op value AND ...
```

```
<table>
<thead>
<tr>
<th>Flight</th>
<th>Seat</th>
</tr>
</thead>
<tbody>
<tr>
<td>origin</td>
<td>price</td>
</tr>
<tr>
<td>destination</td>
<td>fareClass</td>
</tr>
<tr>
<td>duration</td>
<td>available</td>
</tr>
</tbody>
</table>
```
Finding relevant Forms for Keyword Query I

- **keyword query terms:**
  - *schema term*: term is name of table
    - “flight” → table Flight
  - *data term*: term is a value in a table
    - “london” → value in Flight.origin

- **two inverted indexes**
  - DataIndex: term → <tuple-id, table>
    - “london” → <243, Flight>, <547, Flight>
  - FormIndex: term → form-ids
    - search within the form’s descriptions, schema terms, …
"berlin flight"
Ranking and Grouping

- **Rank forms**
  - interprete forms as „documents“ and use well-known TF/IDF scores

- **Group forms**
  - Problem: forms based on the same skeleton template get the same score
  - Solution: Group them together

- Which seats are in which flights
- Flights with their airline
  - Basic Form
    - Aggregate Form(s)
    - Group By Form(s)
    - Union/Intersect Form(s)
- Details about an Airline
Keyword Search - Result

- set of 700 forms
- 7 information needs
- 7 users

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>percentage of returned queries (average)</td>
<td>27.8%</td>
</tr>
<tr>
<td>ranks of correct form (medians per query)</td>
<td>1 or 2</td>
</tr>
<tr>
<td>interaction time (medians per query)</td>
<td>from 24.6 to 106.9 seconds</td>
</tr>
<tr>
<td>response time (average)</td>
<td>84.57 ms</td>
</tr>
</tbody>
</table>
Approaches

(1) combine keyword search and form-based interfaces

(2) provide a form that the user can modify

(3) iteratively ask the users questions to find out what information he is searching for
Form Definition

- **Form**
- **Form Element**
  - set of form controls
- **Form Group**
  - set of form-elements
Form Components

- Main components
  - Inputs
  - Outputs
  - Relationships

- Each component can be represented as a tree

- Form as 3-tuple: \(<IT, OT, RT>\)

- Modifying a form
  - using predefined set of form modification operators
  - leads to modification of the corresponding tree
Input Tree

- root
  - FLIGHT
    - Constraint (Origin)
    - Constraint (Destination)
    - Aggregate (Seat)
  - SEAT
    - Constraint (FareClass)
    - Constraint (Available)
  - AIRLINE
    - Constraint (Name)
    - Constraint (URL)

Criteria Pane

- Enter search criteria
  - Northwest
    - Flight
      - Origin
      - Destination
    - COUNT
      - Seat
    - Airline
      - Name
      - URL
    - Seat
      - FareClass
      - Available

If a field is missing, click here to add it.

Submit Reset
Output Tree

Result Pane

14/07/2014 Database Query Approaches to overcome the Tradeoff between Usability and Expressivity
Query Generation

- Translation procedure
  - dynamically formulate the query
  - in contrast to static forms and predefined queries

- Input Tree
  - WHERE

- Output Tree
  - SELECT
  - ORDER BY

- Relation Tree
  - FROM
  - JOIN
Form Customization - Result

- Usability studies
  - casual and expert users
  - given forms and a set of not fully supported queries
  - measured time taken to solve querying tasks using the interface

- Results
  - customizable forms are effective regardless of querying expertise

Results of expert user study
Approaches

(1) combine keyword search and form-based interfaces

(2) provide a form that the user can modify

(3) iteratively ask the user questions to find out what information he is searching for
Interactive Query Construction - Example

- Keyword query:  london paris 2 15

- Select the first correct option from the top:
  - ✗ Flight.origin: [london]
  - Flight.destination: [london]
Interactive Query Construction - Example

- Keyword query: london paris 2 15

- Select the first correct option from the top:
  - Departure.time: [2, 15]
  - Flight.duration: [2, 15] ✗
  - Flight.distance: [15]
  - Departure.date: [2, 15]
  - Seat.price: [2]
Comparison

- **Expressivity**
  - (2) only approach with result ordering
  - (3) no aggregation

- **Usability**
  - (1) problems with the overview of the retrieved forms
  - (2) fits best in the workflow (just one form)

- **Scalability**
  - (1) all the forms have to be created off-line
  - (2) best for mono-thematic databases
  - (3) well scalable using ontologies
Result

- no perfect solution

- decision criteria
  - topic
  - scale
  - wished expressivity

- combined approach
  - start with keyword query and retrieve customizable forms
Summary – Three Approaches

(1) combine keyword search and form-based interfaces
   - skeleton templates
   - query generation algorithm

(2) provide a form that the user can modify
   - form definition: panes, groups, elements
   - tree modification: Input, Output and Relation Tree

(3) iteratively ask the users questions to find out what information he is searching for
References

  - http://doi.acm.org/10.1145/1353343.1353395

  - http://doi.acm.org/10.1145/1559845.1559883

- Elena Demidova. 2014. Refinement of keyword queries over structured data with ontologies and users
  - Lecture Slides for “Advanced Methods of IR”, SS 14
  - Source of the image in slide 2