iTrails: Pay-as-you-go Information Integration in Dataspaces

Marcos Vaz Salles  Jens Dittrich  Shant Karakashian
Olivier Girard  Lukas Blunschi
ETH Zurich
VLDB 2007
Outline

- Motivation
- iTrails
- Experiments
- Conclusions and Future Work
Problem: Querying Several Sources

Data Sources
- Laptop
- Email Server
- Web Server
- DB Server

Query Systems

What is the impact of global warming in Zurich?
Solution 1: Use a Search Engine

Data Sources

Jobs in Climate Change: Earthworks PhD Student Mountain Alpinist ...

PHD Student Mountain Alpinist Sails and Global Warming, Zurich, A PHD position is open to an enthusiastic person interested in the research of high elevation ...

www.earthworks.jobs/climatefar2007.html - 7k - Cached - Similar pages

Impact of global dimming and brightening on global warming
Impact of global dimming and brightening on global warming. Martin Wild. Institute for Atmospheric and Climate Sciences, ETH Zurich, Zurich, Switzerland ...

www.agu.org/jobs/cross/ref/2005/G02/2005GO232651.shtml - 7k - Cached - Similar pages

swissinfo - swissinfo talks to Swiss scientist Konrad Steffen ...

Iceman keeps his cool despite global warming... set up the Swiss Camp in Greenland for the Federal Institute of Technology in Zurich in 1990 (swissinfo) ...

www.swissinfo.ch/eng/features/danielboershafter/ iceman_keeps_his_cool_despite_global_warming.html?pSect=10&as=... - 41k - Cached - Similar pages

SSPP: Uncertainty and Global Warming: An Option Pricing Approach to

Query global warming zurich

Graph IR Search Engine

Drawback: Query semantics are not precise!
Solution 2: Use an Information Integration System

Drawback: Too much effort to provide schema mappings!
Research Challenge: Is There an Integration Solution in-between These Two Extremes?

Pay-as-you-go Information Integration

Dataspace Vision by Franklin, Halevy, and Maier [SIGMOD Record 05]
Outline

- Motivation
- iTrails
- Experiments
- Conclusions and Future Work
iTrails Core Idea: Add Integration Hints Incrementally

- **Step 1**: Provide a search service over all the data
  - Use a general graph data model (see VLDB 2006)
  - Works for unstructured documents, XML, and relations

- **Step 2**: Add integration semantics via hints (trails) on top of the graph
  - Works across data sources, not only between sources

- **Step 3**: If more semantics needed, go back to step 2

- **Impact**: Smooth transition between search and data integration
  - Semantics added incrementally improve precision / recall
iTrails: Defining Trails

- **Basic Form of a Trail**

  ![Diagram](image)

  Queries: NEXI-like keyword and path expressions

  Attribute projections

- **Intuition:**
  
  When I query for $Q_L [\cdot C_L]$, you should also query for $Q_R [\cdot C_R]$
Trail Examples: Global Warming Zurich

- **Trail for Implicit Meaning:**
  “When I query for global warming, you should also query for Temperature data above 10 degrees”

- **Trail for an Entity:**
  “When I query for zurich, you should also query for references of zurich as a region”

### Temperatures

<table>
<thead>
<tr>
<th>date</th>
<th>city</th>
<th>region</th>
<th>celsius</th>
</tr>
</thead>
<tbody>
<tr>
<td>24-Sep</td>
<td>Bern</td>
<td>BE</td>
<td>20</td>
</tr>
<tr>
<td>24-Sep</td>
<td>Uster</td>
<td>ZH</td>
<td>15</td>
</tr>
<tr>
<td>25-Sep</td>
<td>Zurich</td>
<td>ZH</td>
<td>14</td>
</tr>
<tr>
<td>26-Sep</td>
<td>Zurich</td>
<td>ZH</td>
<td>9</td>
</tr>
</tbody>
</table>
Trail Example: Deep Web Bookmarks

- **Trail for a Bookmark**: “When I query for train home, you should also query for the TrainCompany’s website with origin at ETH Uni and destination at Seilbahn Rigiblick”

```
train home →
//trainCompany.com/*[origin=“ETH Uni” and dest =“Seilbahn Rigiblick”]
```
Trail Examples: Thesauri, Dictionaries, Language-agnostic Search

- **Trail for Thesauri:** “When I query for car, you should also query for auto”

- **Trails for Dictionary:**
  - “When I query for car, you should also query for carro and vice-versa”
Trail Examples: Schema Equivalences

- **Trail for schema match on names**: “When I query for `Employee.empName`, you should also query for `Person.name`”

  ```
  //Employee/* .tuple.empName → //Person/* .tuple.name
  ```

- **Trail for schema match on salaries**: “When I query for `Employee.salary`, you should also query for `Person.income`”

  ```
  //Employee/* .tuple.salary → //Person/* .tuple.income
  ```

<table>
<thead>
<tr>
<th>Employee</th>
<th>Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>empld</td>
<td>SSN</td>
</tr>
<tr>
<td>empName</td>
<td>name</td>
</tr>
<tr>
<td>salary</td>
<td>age</td>
</tr>
<tr>
<td></td>
<td>income</td>
</tr>
</tbody>
</table>
Outline

- Motivation
- iTrails
- Experiments
- Conclusion and Future Work

- Core Idea
- Trail Examples
- How are Trails Created?
- Uncertainty and Trails
- Rewriting Queries with Trails
- Recursive Matches
How are Trails Created?

- Given by the user
  - Explicitly
  - Via Relevance Feedback
- (Semi-)Automatically
  - Information extraction techniques
  - Automatic schema matching
  - Ontologies and thesauri (e.g., wordnet)
  - User communities (e.g., trails on gene data, bookmarks)
Uncertainty and Trails

- **Probabilistic Trails:**
  - model uncertain trails
  - probabilities used to rank trails

\[ Q_L [.C_L] \xrightarrow{p} Q_R [.C_R], \ 0 \leq p \leq 1 \]

- Example: car $\rightarrow$ auto
  - $p = 0.8$
Certainty and Trails

- **Scored Trails:**
  - give higher value to certain trails
  - scoring factors used to boost scores of query results obtained by the trail

\[ Q_L [.C_L] \xrightarrow{sf} Q_R [.C_R], \text{sf} > 1 \]

- **Examples:**
  - \( T_1: \text{weather} \rightarrow //\text{Temperatures/}^{*} \)
    \[ p = 0.9, \text{sf} = 2 \]
  - \( T_2: \text{yesterday} \rightarrow //\text{[date = today() - 1]}^{*} \)
    \[ p = 1, \text{sf} = 3 \]
Rewriting Queries with Trails

(1) Matching

$T_2: \text{yesterday} \rightarrow /*[\text{date} = \text{today}() - 1]$ 

(2) Transformation

Query

Trail

T_2: matches

yesterday

weather

$T_2$: 

yesterday

weather

$/*[\text{date} = \text{today}() - 1]$
Replacing Trails

- Trails that use replace instead of union semantics

(1) Matching

\[ T_2 \]

\[ \text{weather yesterday} \]

(2) Transformation

\[ \text{weather} \; //*[\text{date} = \text{today}() - 1] \]

(3) Merging

\[ \text{weather} \; //*[\text{date} = \text{today}() - 1] \]
Problem: Recursive Matches (1/2)

New query still matches $T_2$, so $T_2$ could be applied again

Infinite recursion
Problem: Recursive Matches (2/2)

Trails may be mutually recursive.

We again match $T_3$ and enter an infinite loop.
Solution: Multiple Match Coloring Algorithm

First Level

T_1 matches

T_2 matches

Second Level

T_3, T_4 match

T_1: weather → //Temperatures/*

T_2: yesterday → /*[date = today() – 1]

T_3: /*.tuple.date → /*.tuple.modified

T_4: /*.tuple.date → /*.tuple.received
Multiple Match Coloring Algorithm Analysis

- **Problem:** MMCA is exponential in number of levels
- **Solution:** Trail Pruning
  - Prune by number of levels
  - Prune by top-K trails matched in each level
  - Prune by both top-K trails and number of levels
Outline

- Motivation
- iTrails
- Experiments
- Conclusion and Future Work
iTrails Evaluation in iMeMex

- **iMeMex Dataspace System**: Open-source prototype available at [http://www.imemex.org](http://www.imemex.org)

- **Main Questions in Evaluation**
  - Quality: Top-K Precision and Recall
  - Performance: Use of Materialization
  - Scalability: Query-rewrite Time vs. Number of Trails
iTrails Evaluation in iMeMex

- **Scenario 1:** Few High-quality Trails
  - Closer to information integration use cases
  - Obtained real datasets and indexed them
  - 18 hand-crafted trails
  - 14 hand-crafted queries

- **Scenario 2:** Many Low-quality Trails
  - Closer to search use cases
  - Generated up to 10,000 trails
iTrails Evaluation in iMeMex: Scenario 1

- Configured iMeMex to act in three modes
  - **Baseline**: Graph / IR search engine
  - **iTrails**: Rewrite search queries with trails
  - **Perfect Query**: Semantics-aware query

- Data: shipped to central index

<table>
<thead>
<tr>
<th>Data Source</th>
<th>Desktop</th>
<th>Wiki4V</th>
<th>Enron</th>
<th>DBLP</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net Data size</td>
<td>1,230</td>
<td>26,392</td>
<td>111</td>
<td>713</td>
<td>28,446</td>
</tr>
</tbody>
</table>

sizes in MB
Quality: Top-K Precision and Recall

Scenario 1: few high-quality trails (18 trails)

Search Engine misses relevant results

Q3: pdf yesterday

Perfect Query always has precision and recall equal to 1

Search Query is partially semantics-aware

Q13: to = raimund.grube@enron.com
Performance: Use of Materialization

Scenario 1:
few high-quality trails
(18 trails)

Trail merging adds overhead to query execution

Trail Materialization provides interactive times for all queries

<table>
<thead>
<tr>
<th>Q. No.</th>
<th>iTrails with Basic Indexes</th>
<th>iTrails with Trail Mat.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.18</td>
<td>0.21</td>
</tr>
<tr>
<td>2</td>
<td>0.74</td>
<td>0.52</td>
</tr>
<tr>
<td>3</td>
<td>10.72</td>
<td>0.39</td>
</tr>
<tr>
<td>4</td>
<td>1.86</td>
<td>0.07</td>
</tr>
<tr>
<td>5</td>
<td>0.56</td>
<td>0.44</td>
</tr>
<tr>
<td>6</td>
<td>0.32</td>
<td>0.05</td>
</tr>
<tr>
<td>7</td>
<td>1.73</td>
<td>0.67</td>
</tr>
<tr>
<td>8</td>
<td>5.27</td>
<td>0.48</td>
</tr>
<tr>
<td>9</td>
<td>179.02</td>
<td>1.50</td>
</tr>
<tr>
<td>10</td>
<td>10.14</td>
<td>0.29</td>
</tr>
<tr>
<td>11</td>
<td>0.60</td>
<td>0.60</td>
</tr>
<tr>
<td>12</td>
<td>0.60</td>
<td>0.60</td>
</tr>
<tr>
<td>13</td>
<td>0.49</td>
<td>0.44</td>
</tr>
<tr>
<td>14</td>
<td>0.14</td>
<td>0.14</td>
</tr>
</tbody>
</table>

response times in sec.
Scalability: Query-rewrite Time vs. Number of Trails

Scenario 2: many low-quality trails

Query-rewrite time can be controlled with pruning
Conclusion: Pay-as-you-go Information Integration

- **Step 1:** Provide a search service over all the data
- **Step 2:** Add integration semantics via trails
- **Step 3:** If more semantics needed, go back to step 2

**Our Contributions**
- **iTrails:** generic method to model semantic relationships (e.g. implicit meaning, bookmarks, dictionaries, thesauri, attribute matches, ...)
- We propose a framework and algorithms for Pay-as-you-go Information Integration
- Smooth transition between search and data integration
Future Work

- Trail Creation
  - Use collections (ontologies, thesauri, wikipedia)
  - Work on automatic mining of trails from the dataspace

- Other types of trails
  - Associations
  - Lineage
Questions?

Thanks in advance for your feedback! 😊

marcos.vazsalles@inf.ethz.ch

http://www.imemex.org
Backup Slides
Problem: Global Warming in Zurich

- Query: “What is the impact of global warming in Zurich?”
- Search for: global warming zurich
- Meaning of keyword query
  - global warming should lead to query on Temperatures
  - zurich should lead to a query for a city
Problem: PDF Yesterday

- Query: “Retrieve all PDF documents added/modified yesterday”
- Search for: `pdf yesterday`
- Meaning of keywords `pdf` and `yesterday`
- Different sources, different schemas:
  - Laptop: modified
  - Email: received
  - DBMS: changed
Related Work: Search vs. Data Integration vs. Dataspaces

<table>
<thead>
<tr>
<th>Features</th>
<th>Search</th>
<th>Dataspaces</th>
<th>Data Integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration Effort</td>
<td>Low</td>
<td>Pay-as-you-go</td>
<td>High</td>
</tr>
<tr>
<td>Query Semantics</td>
<td>Precision / Recall</td>
<td>Precision / Recall</td>
<td>Precise</td>
</tr>
<tr>
<td>Need for Schema</td>
<td>Schema-never</td>
<td>Schema-later</td>
<td>Schema-first</td>
</tr>
</tbody>
</table>

Integration Solution
Personal Dataspaces Literature


iDM: iMeMex Data Model

- **Our approach**: get the data model closer to personal information – not the other way around

- **Supports**:
  - Unstructured, semi-structured and structured data, e.g., files&folders, XML, relations
  - Clearly separation of logical and physical representation of data
  - Arbitrary directed graph structures, e.g., section references in LaTeX documents, links in filesystems, etc
  - Lazily computed data, e.g., ActiveXML (Abiteboul et. al.)
  - Infinite data, e.g., media and data streams

See VLDB 2006
## Data Model Options

<table>
<thead>
<tr>
<th>Support for Personal Data</th>
<th>Non-schematic data</th>
<th>Serialization independent</th>
<th>Support for Graph data</th>
<th>Support for Lazy Computation</th>
<th>Support for Infinite data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Models</strong></td>
<td>Bag of Words</td>
<td>Relational</td>
<td>XML</td>
<td>iDM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>Extension: XLink/ XPointer</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>Extension: ActiveXML</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Extension: Document streams</td>
<td>Extension: Relational streams</td>
<td>Extension: XML streams</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Data Models for Personal Information

Abstraction Level

lower

Physical Level

Relational

Document / Bag of Words

XML

iDM

Personal Information

higher
Architectural Perspective of iMeMex

Complex operators (query algebra)

Indexes & Replicas access (warehousing)

Data source access (mediation)