SEEK: Scalable Extraction of Enterprise Knowledge

Joachim Hammer

Dept. of CISE
University of Florida

26-Feb-2002
Project Overview

Faculty
Joachim Hammer
Mark Schmalz
William O’Brien
Ray Issa
Joe Geunes
Sherman Bai

Students
Sangeetha Shekar
Nikhil Haldevnekar
Jae-Hyun Choi
Rodrigo Castro
Huanqing Lu
Bibo Yang
Bo Lu

- Sponsored by NSF
- Year 2 of 4-year effort
Outline

- Motivation
- SEEK Information Architecture
- Knowledge Extraction
  - Data Reverse Engineering
  - Code Mining
- Summary and Conclusion
Motivation

- Need for decision/negotiation support to improve performance and customization across extended business networks
- Hard ... participating firms have unique and often incompatible information systems, varying levels of sophistication
  - Current approaches require manual coding of connection software - Not scalable
- Development of a toolkit to facilitate integration of heterogeneous legacy data and knowledge
SEEK Environment & Context

extended enterprise

supplier

sub/supplier

SEEK

SEEK

SEEK

Secure Hosting Infrastructure

coordinator/lead

sub/supplier

analysis (e.g., E-ERP)
SEEK Information Architecture

Connection Toolkit

Legacy Source

SEEK Components

Secure, value-added extraction of firm knowledge

Analysis Module

Knowledge Extraction Module

Wrapper

AM: query analysis, knowledge composition (mediation)
W: source connection and translation
KEM: configuration of W and AM at build-time

Domain Expert

Legacy Data and Systems
Run-Time Interactions

- Queries, results, domain knowledge represented in XML
  - Different information contexts: Hub, Analysis Module, Source
  - Translator needed to convert between information contexts
  - Assume existence of translator between AM and Hub contexts

- Analysis module provides robust (value-added) mediation
  - Solution strategy based on information available in source
  - Capable of composing final answer out of multiple source results

- SEEK wrapper responsible for syntactic and semantic conversions
  - Formulates source queries based on capabilities of source
  - Restructures source results to conform to information context of AM
Run-Time Interactions

- $Q/R_C$: query/result expressed in information context $C$
  - Subscript $i$ denotes set of queries/results
  - E.g., single query from hub may generate one or more queries from AM
Knowledge Extraction

- Produces “description” of accessible knowledge in source
- Iterative process, quality and accuracy of extracted knowledge (and hence the wrapper) improves over time and with human input
- Assume legacy information is accessible via API, e.g., JDBC
- Access to application code improves quality of extraction knowledge further
3-Phases of Knowledge Extraction

Phase-1: Data Reverse Engineering (DRE)
- Legacy DB
- Structure

Phase-2: Semantic Analyzer (SA)
- Application Code

Phase-3: Semantic Matching (SM)
- Concepts
- Relshps
- Constraints
- Business Rules
- Semantics
- Information Model for AM

Feedback:
- Revise, validate
- Train, validate

Note: The diagram illustrates the process of knowledge extraction with phases focusing on data reverse engineering, semantic analysis, and semantic matching.
3-Phases

- **Data Reverse Engineering (DRE)**
  - Extract structural and schematic information from legacy source
  - E.g., entity names, relationships, data types, simple constraints
- **Semantic Analysis (SA)**
  - Validate results from DRE
  - Augment structural information with semantics
  - Infer business rules
- **Schema Matching (SM)**
  - Infer mappings between terms in the information context of AM with related terms in the context of the legacy source
- Phases are interleaved
- User input validates results
Data Reverse Engineering

- Start with DRE of relational schemas
  - Wealth of existing algorithms and approaches, none work perfectly
- Based on several approaches (Petit et al. 96, Signore 95, ...)
  - Modified to fit SEEK needs
  - Extend to discover schema info in semistructured sources (e.g., Web sources)
- Use database catalog to directly extract concepts and simple constraints
- Use database instances to infer relationships and constraints
### What is Extracted?

#### Table name

<table>
<thead>
<tr>
<th>Relation Name</th>
<th>Entity</th>
<th>Relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>T</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>P_Z</td>
<td>No</td>
<td>M-N Relationship between P and Z</td>
</tr>
<tr>
<td>Z</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

#### Attribute

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Table name</th>
<th>Data Type</th>
<th>Key</th>
<th>Foreign key</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_ID</td>
<td>P</td>
<td>Integer</td>
<td>yes</td>
<td>no</td>
<td>Not NULL</td>
</tr>
<tr>
<td>T_ID</td>
<td>P</td>
<td>Integer</td>
<td>No</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>T</td>
<td>Date</td>
<td>No</td>
<td>No</td>
<td>CHECK valid date</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Semantic Analysis

- Application code frequently contains semantic descriptions for schema items in underlying database
  - E.g., Trace database schema names back to output statements
- Identify patterns that encode useful information
  - E.g., business rules encoded in IF-THEN-ELSE statements
- Using technique called *code slicing* to reduce code to only those statements that are of interest to the analyzer (Horwitz, Reps 92)
- Apply pattern matcher to discover associations among variables, patterns, etc.
Example

/* sample program to perform project scheduling */
char *aValue;
char *cValue;

int bValue = 0;
/* more code … */
EXEC SQL SELECT A, C, INTO :aValue, :cValue FROM T
WHERE B = :bValue;
/* more code … */
int flag = 0;
IF (cValue <= aValue)
{
    flag = 1; /* exception handling */
}
/* more code … */
printf ("Task Start Date %d", aValue);
printf ("Task Finish Date %d", cValue);
/* more code … */
What is Extracted?

<table>
<thead>
<tr>
<th>Relation Name</th>
<th>Entity</th>
<th>Relationship</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>Yes</td>
<td>No</td>
<td>Task?</td>
</tr>
<tr>
<td>P_T</td>
<td>No</td>
<td>M-N Relationship between P and T</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attr</th>
<th>Table name</th>
<th>Data Type</th>
<th>Key</th>
<th>Foreign key</th>
<th>Constraint</th>
<th>Var Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_ID</td>
<td>P</td>
<td>Integer</td>
<td>yes</td>
<td>no</td>
<td>Not NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T_ID</td>
<td>P</td>
<td>Integer</td>
<td>No</td>
<td>Task</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>T</td>
<td>Date</td>
<td>No</td>
<td>No</td>
<td>A ≥ C → ERROR</td>
<td>aValue</td>
<td>Task Start Date</td>
</tr>
<tr>
<td>C</td>
<td>T</td>
<td>Date</td>
<td>No</td>
<td>No</td>
<td>A ≥ C → ERROR</td>
<td>cValue</td>
<td>Task Finish Date</td>
</tr>
</tbody>
</table>
Next Step

- How to relate extracted (source) concepts/schema to known schema of AM?
- *Schema matching*...
Current Status

- Validated SEEK approach to managing business networks in simple prototype
  - Hardcoded SEEK wrapper capable of processing three query scenarios (construction domain)
  - MS Project-based legacy source
- Implementing knowledge extraction framework consisting of DRE and SA
  - Researching approaches to schema matching
- Setting up testbed
  - Data collection in cooperation with partners from construction industry (manufacturing examples planned)
Summary and Conclusion

- SEEK is a structured approach to integrating domain-specific legacy sources
- Modular architecture provides several important capabilities
- (Semi)automatic knowledge extraction
  - DRE, semantic analysis, schema matching
- Important contributions to theory of knowledge capture and integration
- Requirement for building scalable sharing architectures