Constructing an Interactive Natural Language Interface for Relational Databases

Fei Li and H. V. Jagadish, VLDB14

Presented by Liat Peterfreund

Talk Outline

- Introduction
- Contribution
- System Overview
- Parse Tree Node Interpretation
- Parse Tree Structure Adjustment
- SQL Generation
- Experiments
- Conclusions

Introduction

- SQL is an expressive and powerful query language.
- However, it is too difficult for users without technical training.
  - Even for experts it poses challenges
- There is a real need in user-friendly query interface:
  - natural language interface.
- Not surprisingly, a natural language interface is regarded by many as the ultimate goal for a database query interface, and many natural language interfaces to databases (NLIDBs) have been built towards this goal.
- Despite these advantages, NLIDBs have not been adopted widely.
- The fundamental problem is that understanding natural language is hard.

Introduction

- Challenges in natural languages:
  - slang words, technical terms, and dialect-specific phrasing.
  - natural language is inherently ambiguous.
- Query-response cycle in real life.
- NaLIR – a query mechanism to facilitate collaboration between the system and the user in processing natural language queries.
  - First, the system explains how it interprets a query, from each ambiguous word/phrase to the meaning of the whole sentence. These explanations enable the user to verify the answer and to be aware where the system misinterprets her query.
  - Second, for each ambiguous part, it provides multiple likely interpretations for the user to choose from.
  - Does not burden the user too much.

Introduction

- How should a system represent and communicate its query interpretation to the user ?
  - SQL is too difficult for most non-technical humans.
  - A representation that is both
    - “human understandable” and
    - “RDBMS understandable”.
- Query Tree.
  - An intermediate between a linguistic parse tree and a SQL statement, a query tree is easier to explain to the user than an SQL statement.
  - Given a query tree verified by the user, the system will almost always be able to translate it into a correct SQL statement.

Introduction – cont'd

- Putting the above ideas together, NaLIR (Natural Language Interface to Relational databases) is an NLIDB comprising three main components:
  - a first component that transforms a natural language query to a query tree,
  - a second component that verifies the transformation interactively with the user, and
  - a third component that translates the query tree into a SQL statement.

Talk Outline

- Introduction
- Contribution
- System Overview
- Parse Tree Node Interpretation
- Parse Tree Structure Adjustment
- SQL Generation
- Experiments
- Conclusions

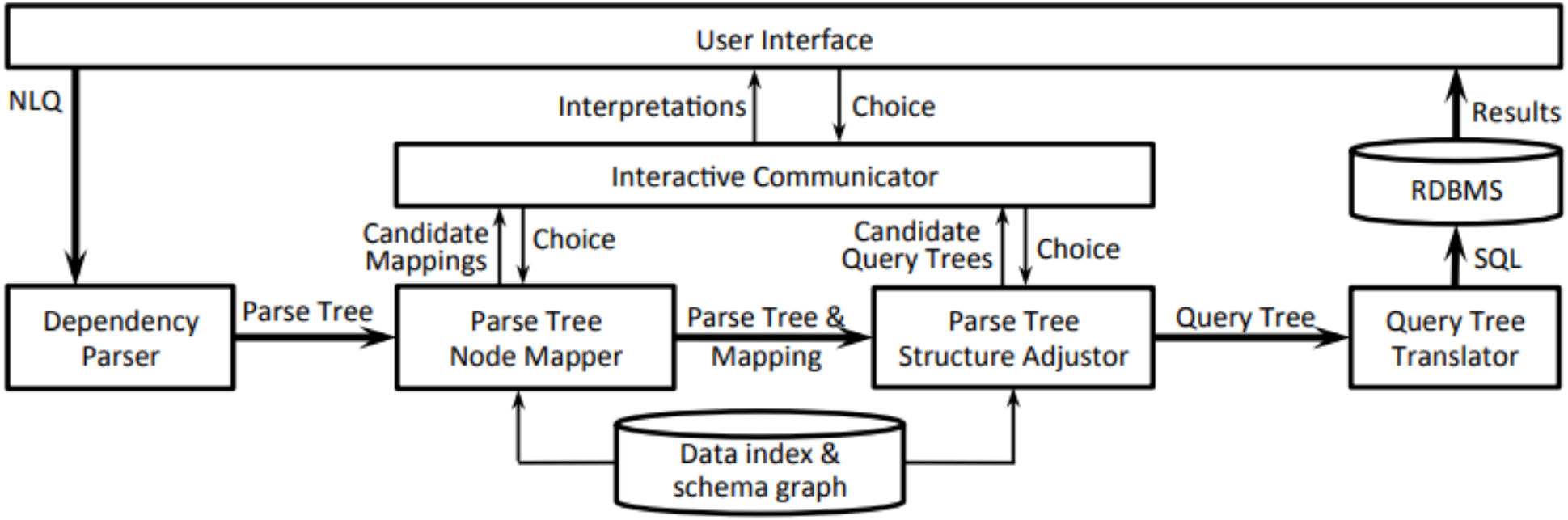
Contribution

1. Interactive Query Mechanism
   - with a little interaction help.
2. Query Tree.
   - A query tree can be explained to the user for verification, and once verified, will almost always be correctly translated to SQL.
3. System Architecture.
   - Modular architecture to support such a query mechanism, in which each component can be designed, and improved, independently.
4. User Study.
   - usable in practice even for naive users.

Talk Outline

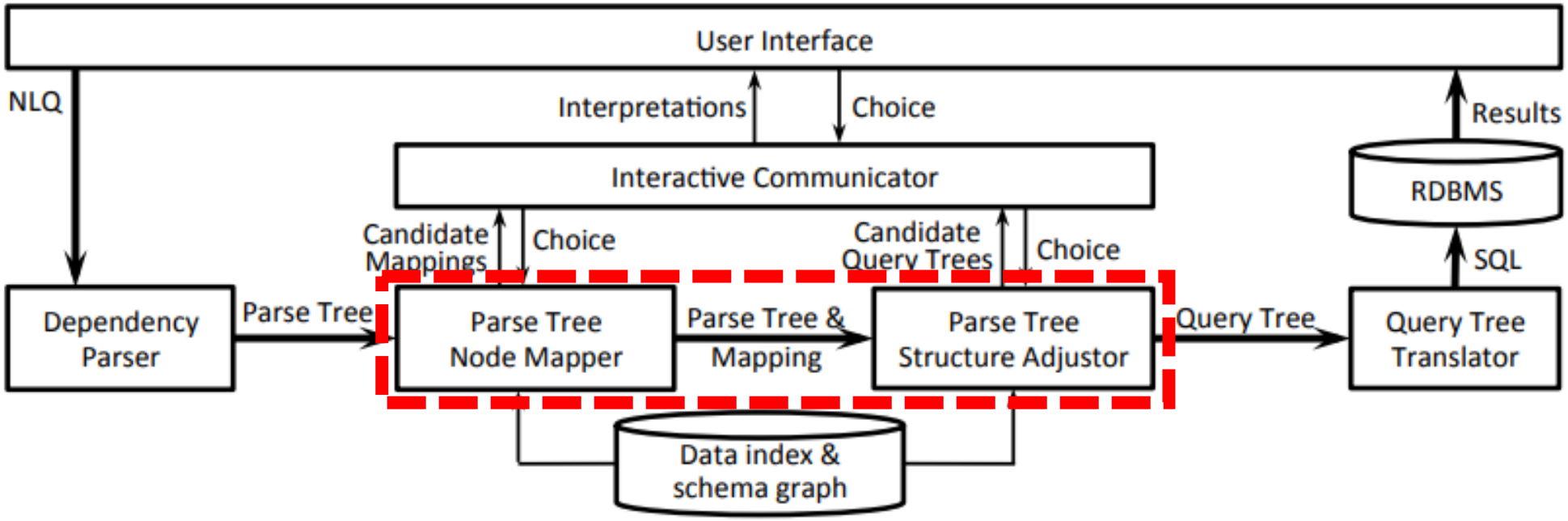
- Introduction
- Contribution
- System Overview
- Parse Tree Node Interpretation
- Parse Tree Structure Adjustment
- SQL Generation
- Experiments
- Conclusions

System Overview



System Overview

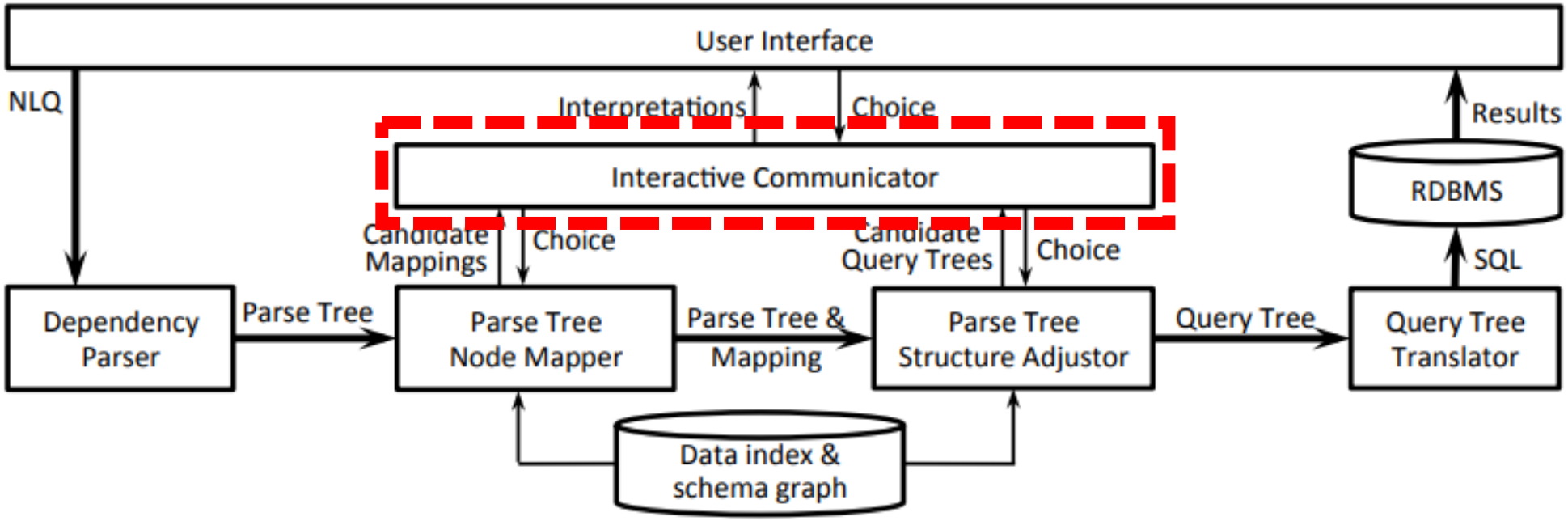
- The system consists of three main parts



- The query interpretation part –
  - responsible for translating the natural language query to a query tree

System Overview

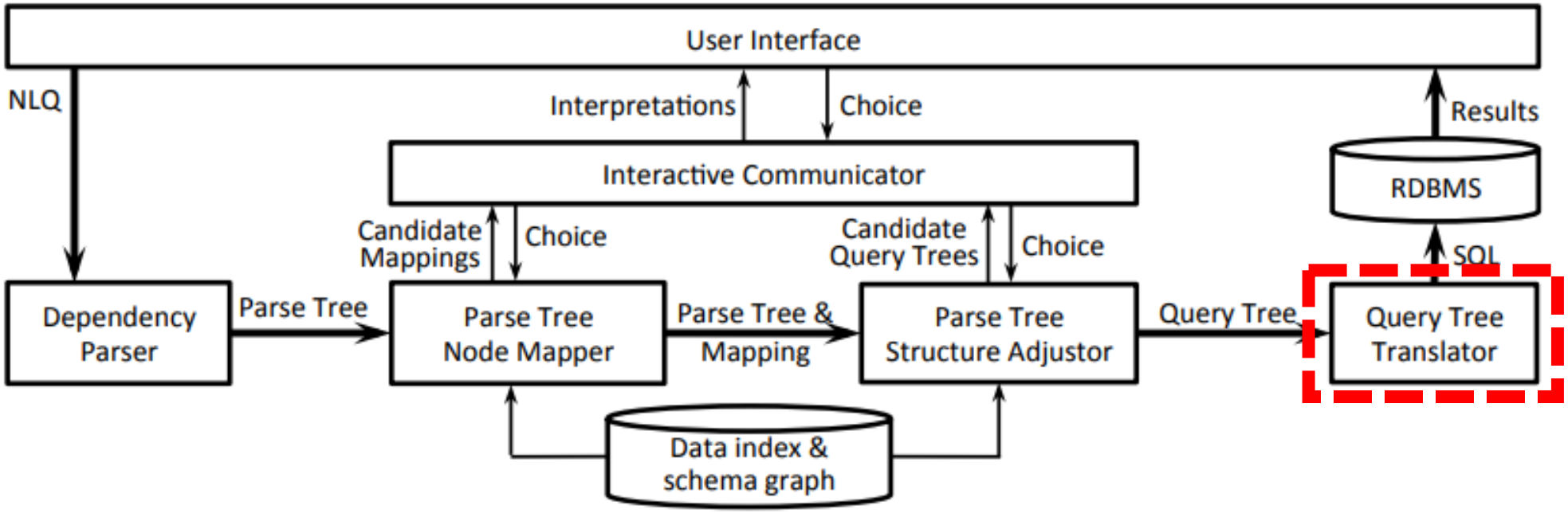
- The system consists of three main parts



- The interactive communicator–
  - responsible for communicating with the user to ensure the translation process is correct.

System Overview

- The system consists of three main parts

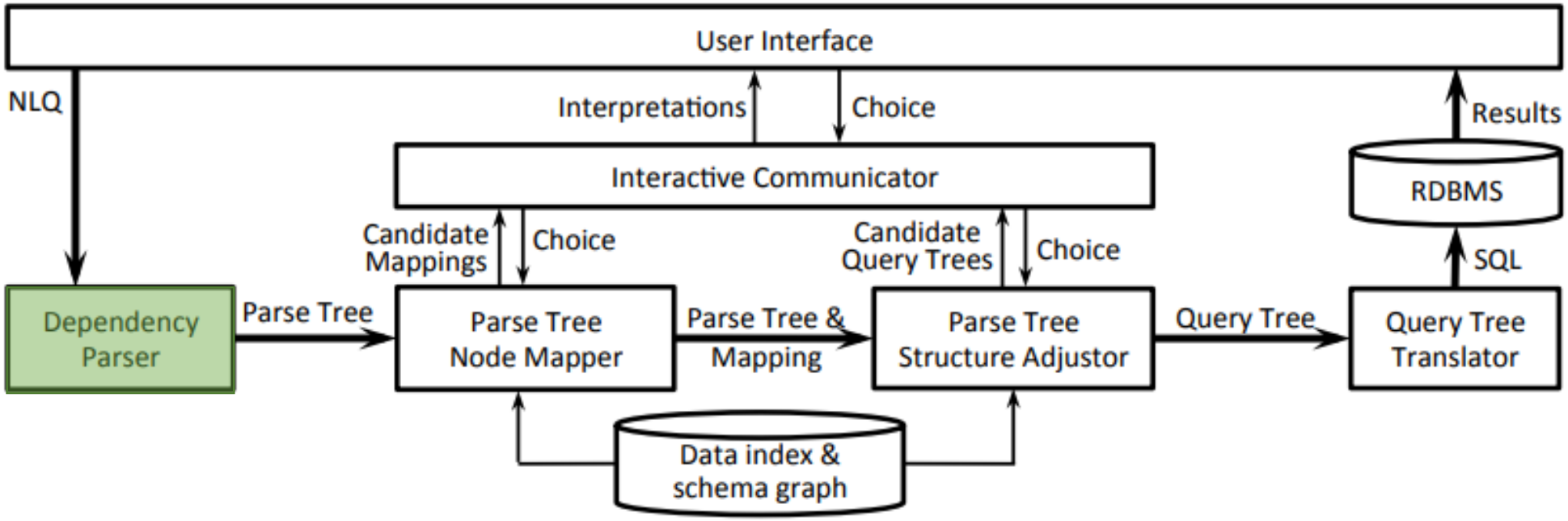


- The query tree translator–
  - responsible for translating the query tree into a SQL statement.

Talk Outline

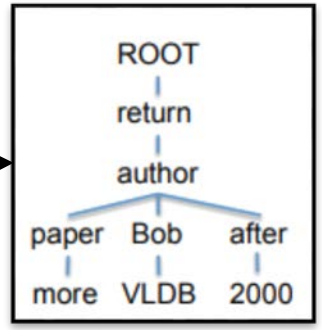
- Introduction
- Contribution
- System Overview
- System Components
- Parse Tree Node Interpretation
- Parse Tree Structure Adjustment
- SQL Generation
- Experiments
- Conclusions

System Components



Return authors who have more papers than Bob in VLDB after 2000.

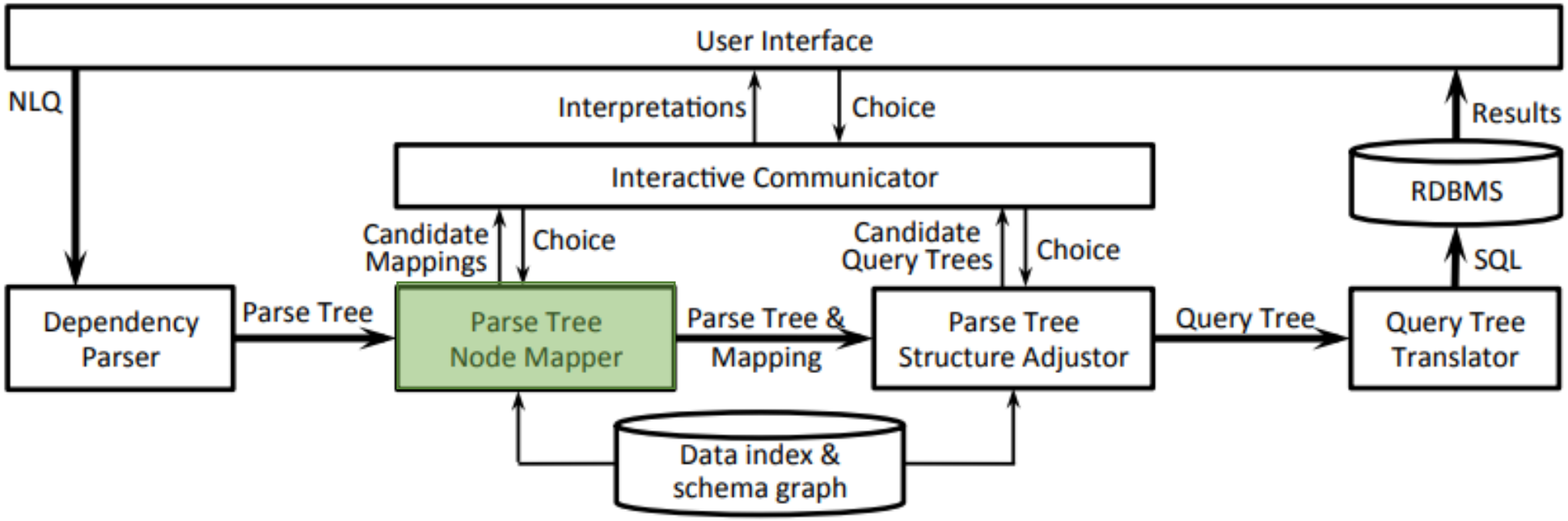
→ Dependency Parser →

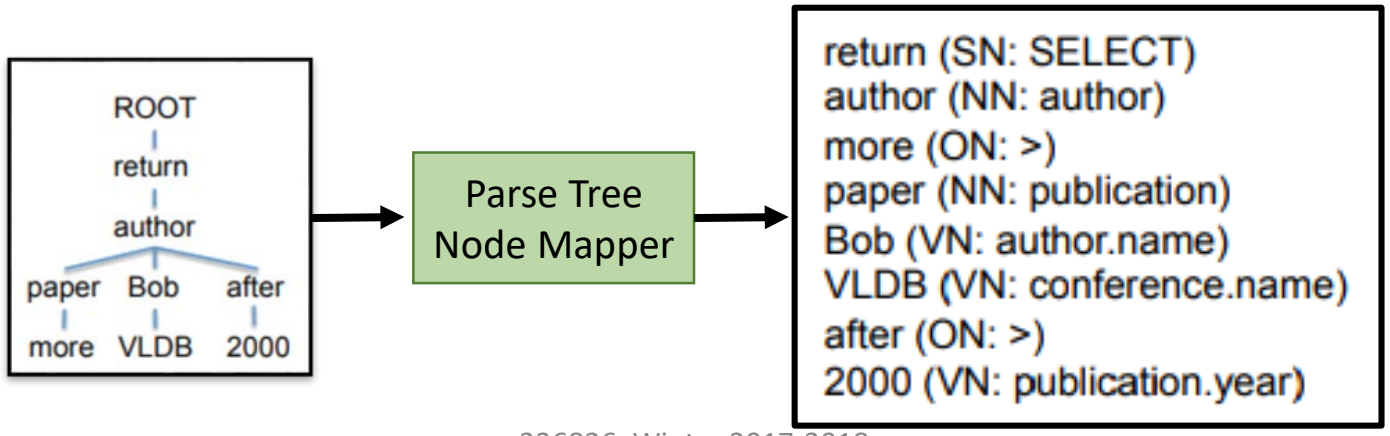


Talk Outline

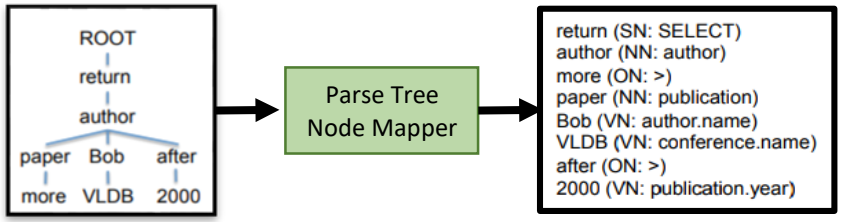
- Introduction
- Contribution
- System Overview
- System Components
- Parse Tree Node Interpretation
- Parse Tree Structure Adjustment
- SQL Generation
- Experiments
- Conclusions

System Components





Parse Tree Node Interpretation



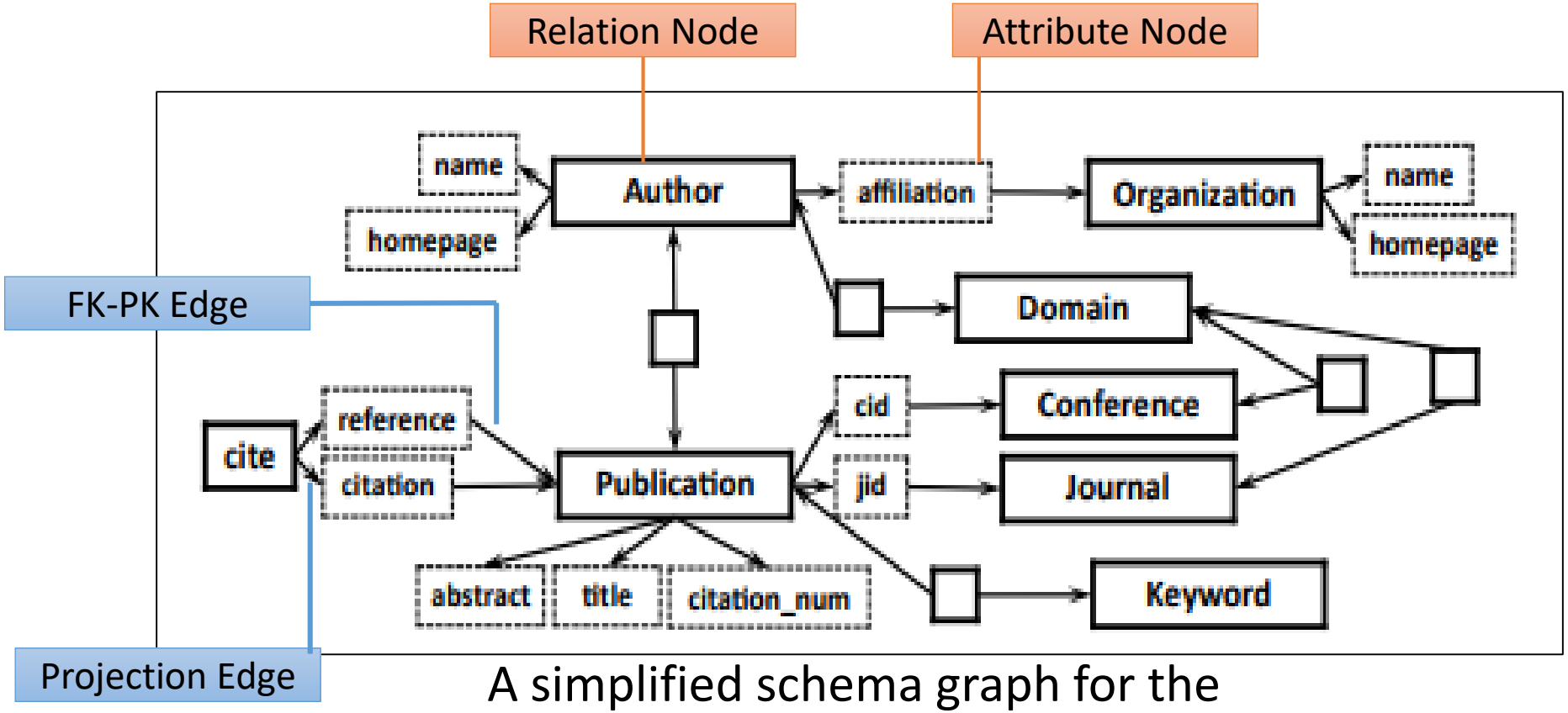
- Identify the parse tree nodes that can be mapped to SQL components.
  - Divide those into different types

  that are independent of the input database.
  - Name nodes and value nodes depend on the

  database being queried.
- This interpretation might be ambiguous.
- The user helps to resolve this ambiguity.

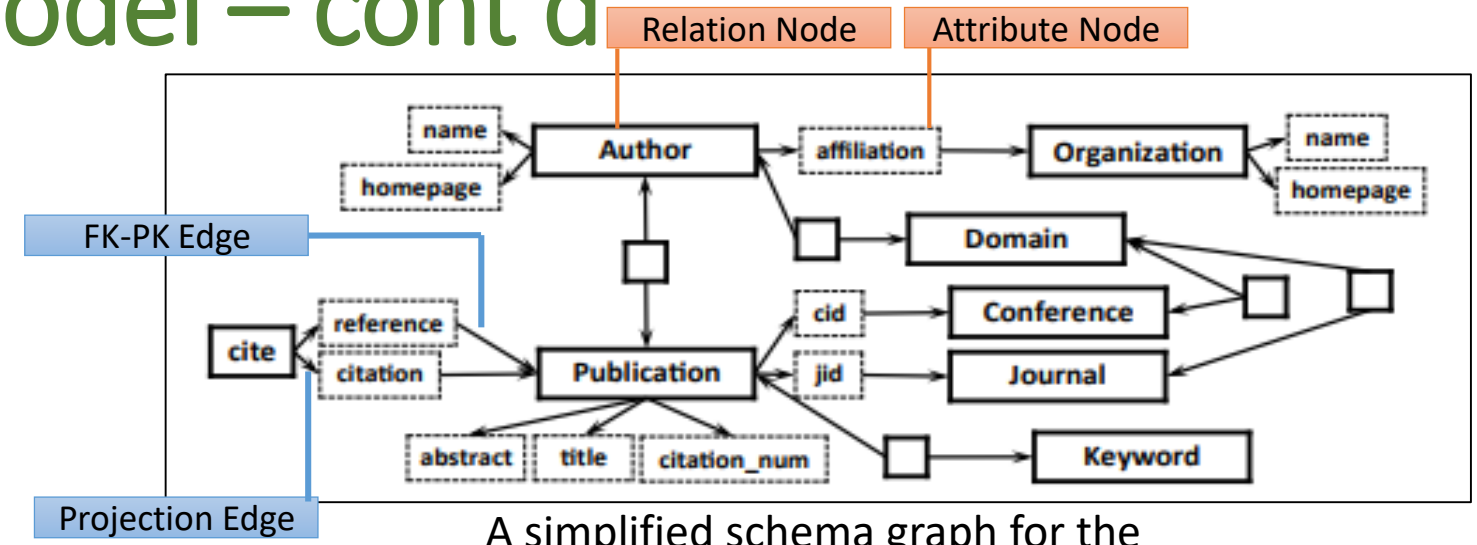
Node Type	Corresponding SQL Component
Select Node (SN)	SQL keyword: SELECT
Operator Node (ON)	an operator, e.g. =, <=, !=, contains
Function Node (FN)	an aggregation function, e.g., AVG
Name Node (NN)	a relation name or attribute name
Value Node (VN)	a value under an attribute
Quantifier Node (QN)	ALL, ANY, EACH
Logic Node (LN)	AND, OR, NOT

Parse Tree Node Interpretation
Data Model



A simplified schema graph for the Microsoft Academic Search Database

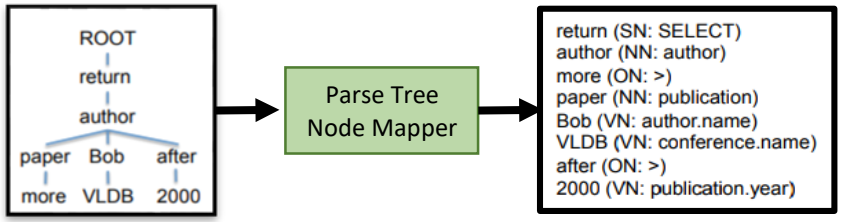
Parse Tree Node Interpretation Data Model – cont'd



A simplified schema graph for the Microsoft Academic Search Database

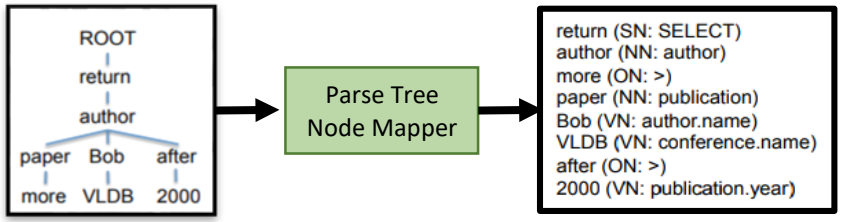
- A join path is a list of connected schema elements.
  - Each edge is assigned to a weight between 0 and 1
    - where a larger weight indicates a stronger connection.
  - The weight of the path is the production of weights of its edges.

Parse Tree Node Interpretation Candidate Mappings



- The system bases its mapping according to similarity score between nodes and schema elements.
- The similarity score is based on two factors:
  - The semantic similarities between the words (based on Wordnet) and
  - the spelling similarity.
- If this score is greater than a predefined threshold then it is a candidate mapping.
- There is often more than one candidate mapping and we deal with this kind of ambiguity interactively with the user.
- For example, the node “VLDB” may have multiple candidate mappings in the database of Microsoft Academic Search:
  - VLDB, VLDB workshops, VLDB PhD workshop, PVLDB and VLDB Journal.

Parse Tree Node Interpretation Default Mapping Strategy

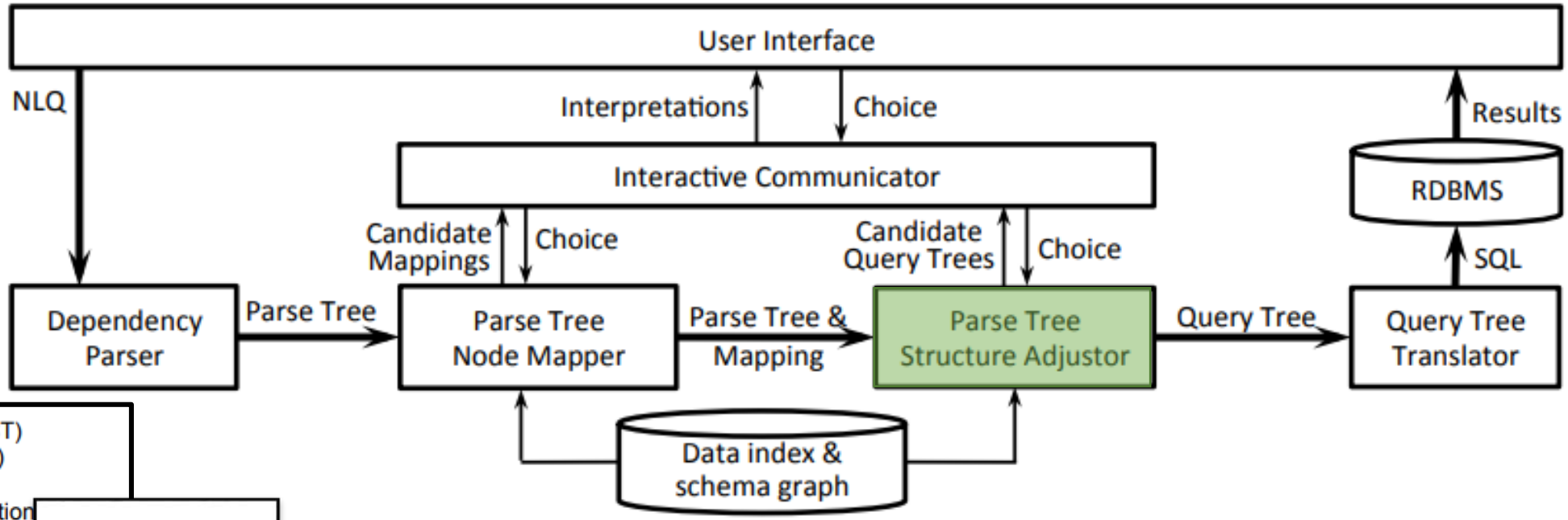


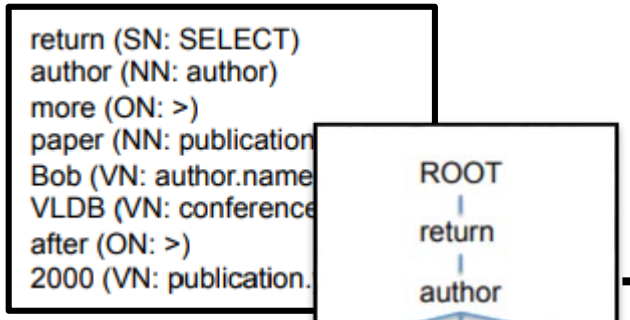
- To facilitate user choice, there is a default mapping.
- The default mapping is based both on
  - the similarity score, as well as
  - the mutual relevance between each pair of nodes.
- The relevance between schema elements is determined by the weight of path that connects them.
- Consider the query “return all conferences in the database area”.
  - The node “database” maps to both the value “database”
    - under Domain.name and
    - under Keyword.keyword.
  - The node “area” is the parent of the node “database” in the parse tree and maps to Domain with high similarity.
  - Therefore node “database” is more likely to refer to a domain name rather than a keyword.

Talk Outline

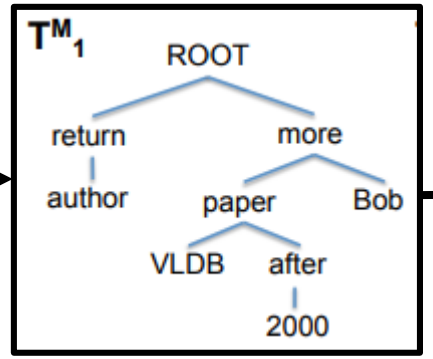
- Introduction
- Contribution
- System Overview
- System Components
- Parse Tree Node Interpretation
- Parse Tree Structure Adjustment
- SQL Generation
- Experiments
- Conclusions

System Components

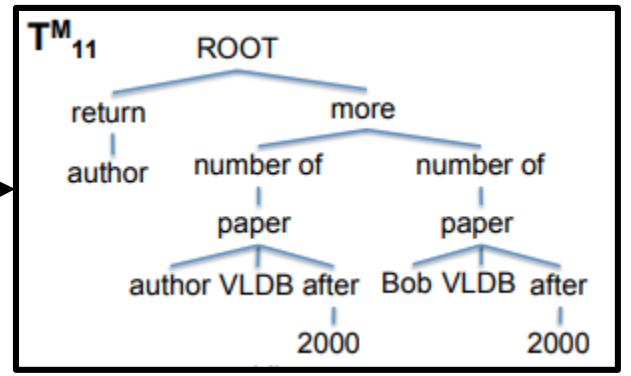




Parse Tree Structure Adjustor



Parse Tree Structure Adjustor

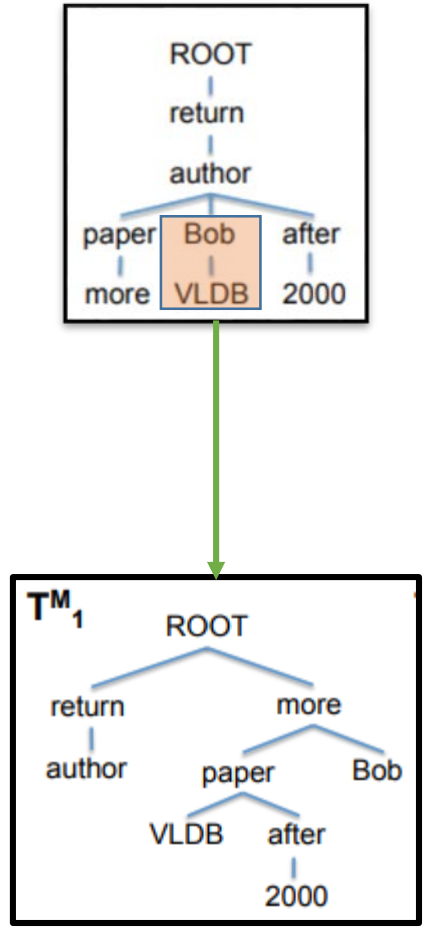


Parse Tree Structure Adjustment

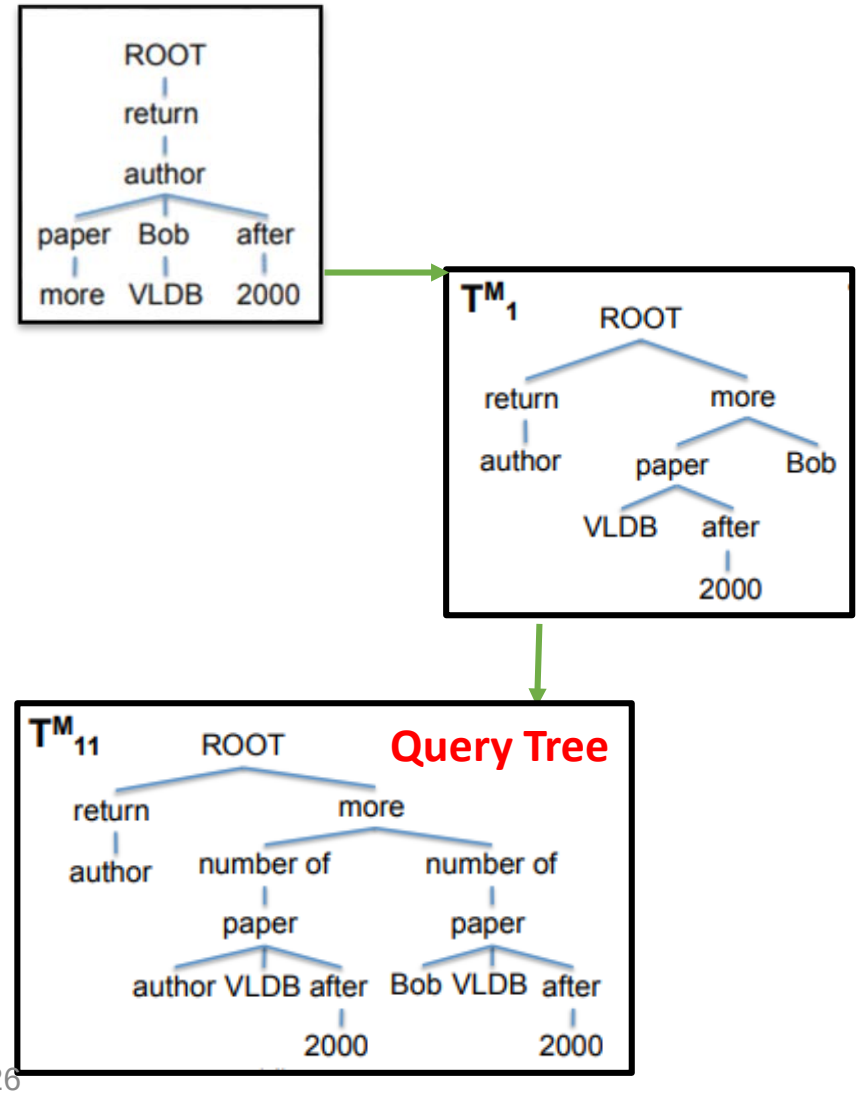
- Three obstacles lie in the way of understanding the query from the parse tree and the mapping obtained in previous step:
  - The linguistic parse tree generated may be incorrect.
  - The structure of the linguistic parse tree does not directly reflect the relationship between the nodes from the database's perspective.

    (a) "return author who has more papers than Bob", and

    (b) "return author whose papers are more than Bob's".
    - Different parse trees, yet similar semantic meanings.
  - Elliptical expressions - group of words with certain understood words omitted.
    - The sentence may be ambiguous before the elliptical part is completed.
- The Parse Tree Structure Adjustor deals with these issues.



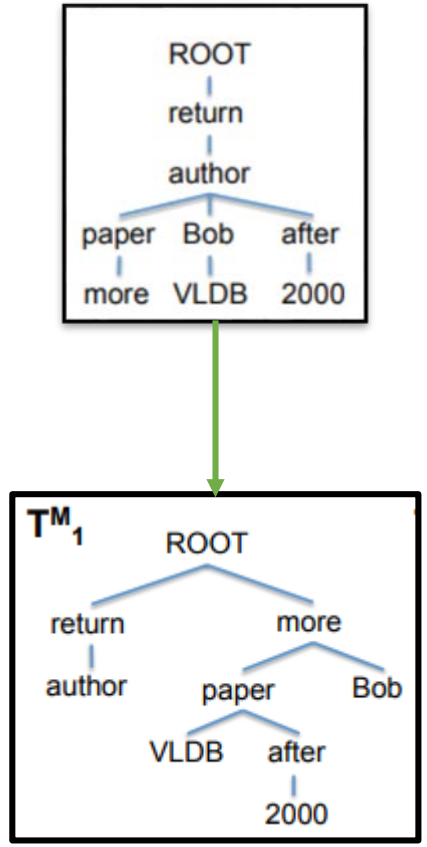
Parse Tree Structure Adjustment
Query Tree



- Query Trees is an intermediate step in the translation:
  - Lies between parse trees and their mappings to the SQL queries.
  - Are valid with respect to a given grammar.
- There is often a big gap between the linguistic parse tree and its corresponding query tree.
- The following two strategies make the mapping process accurate.
  - The system explains a query tree to the user in natural language.
  - The system generates multiple candidate query trees for the user to choose from.

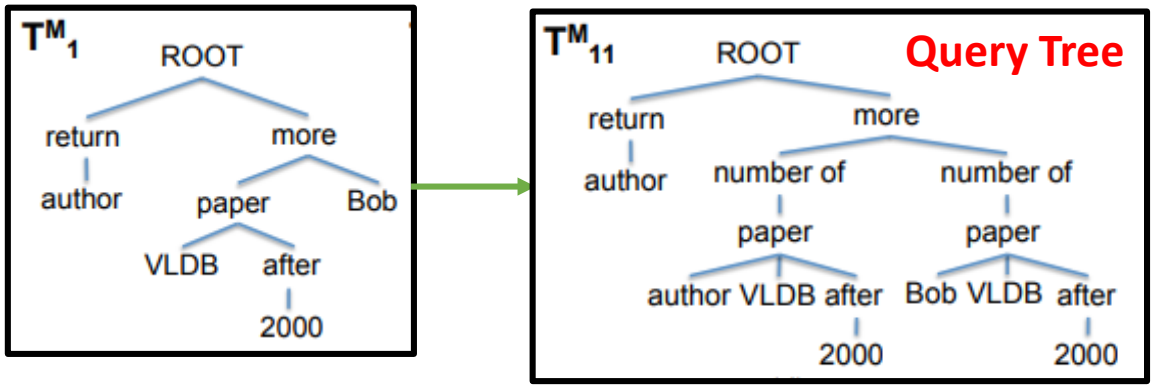
Parse Tree Structure Adjustment
Parse Tree Reformulation

- We use subtree move operations to edit the parse tree until it is syntactically valid according to a predefined grammar.
- To choose from a set of parse trees we take into consideration three aspects:
  1. Its resemblance to the grammar.
  2. How it corresponds to the mapping between the nodes and the schema elements.
  3. Its resemblance to the original linguistic parse tree.



Parse Tree Structure Adjustment
Implicit Nodes Insertion

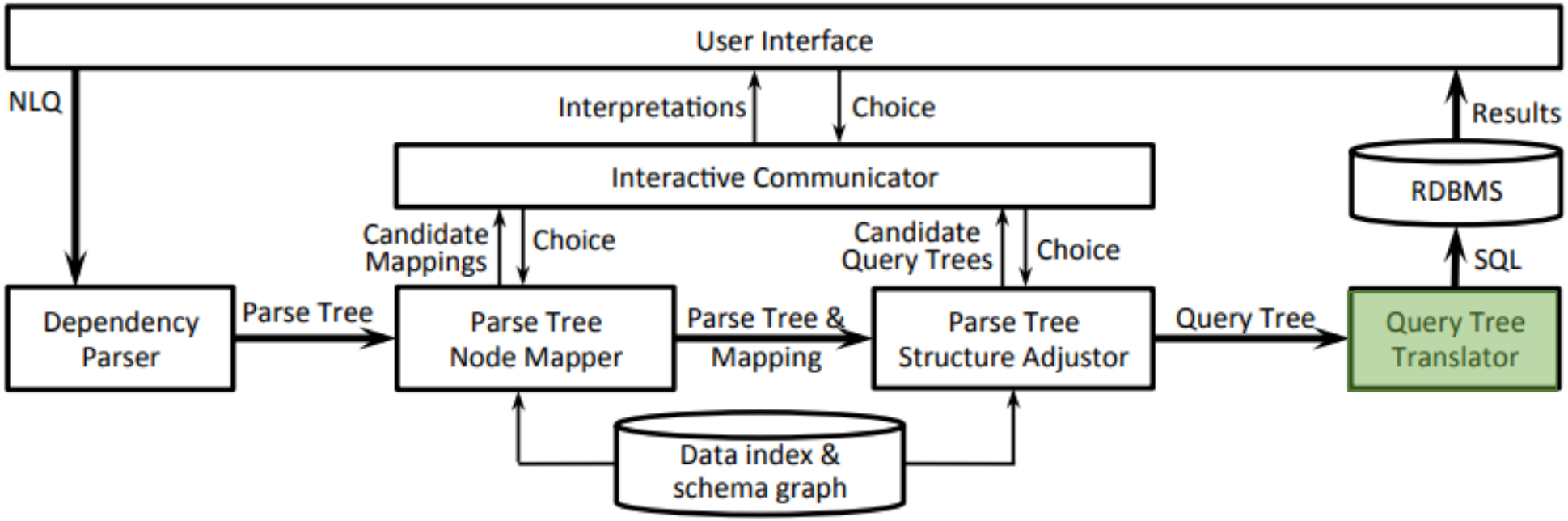
- Natural language sentences often contain elliptical expressions, which make some nodes in their parse trees implicit.
  - Therefore there is need to detect and insert implicit nodes.
- In a query that compares two things –
  - The query tree should be symmetric.

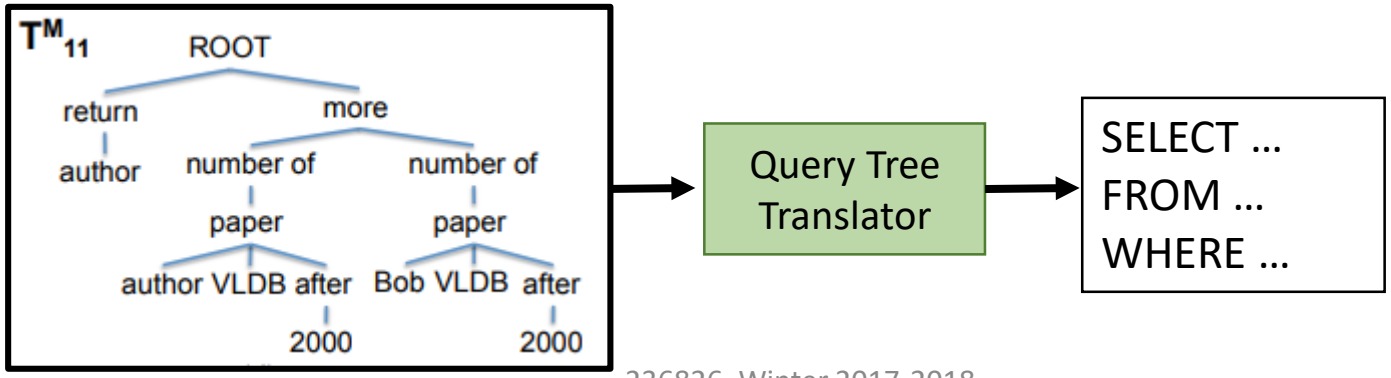


Talk Outline

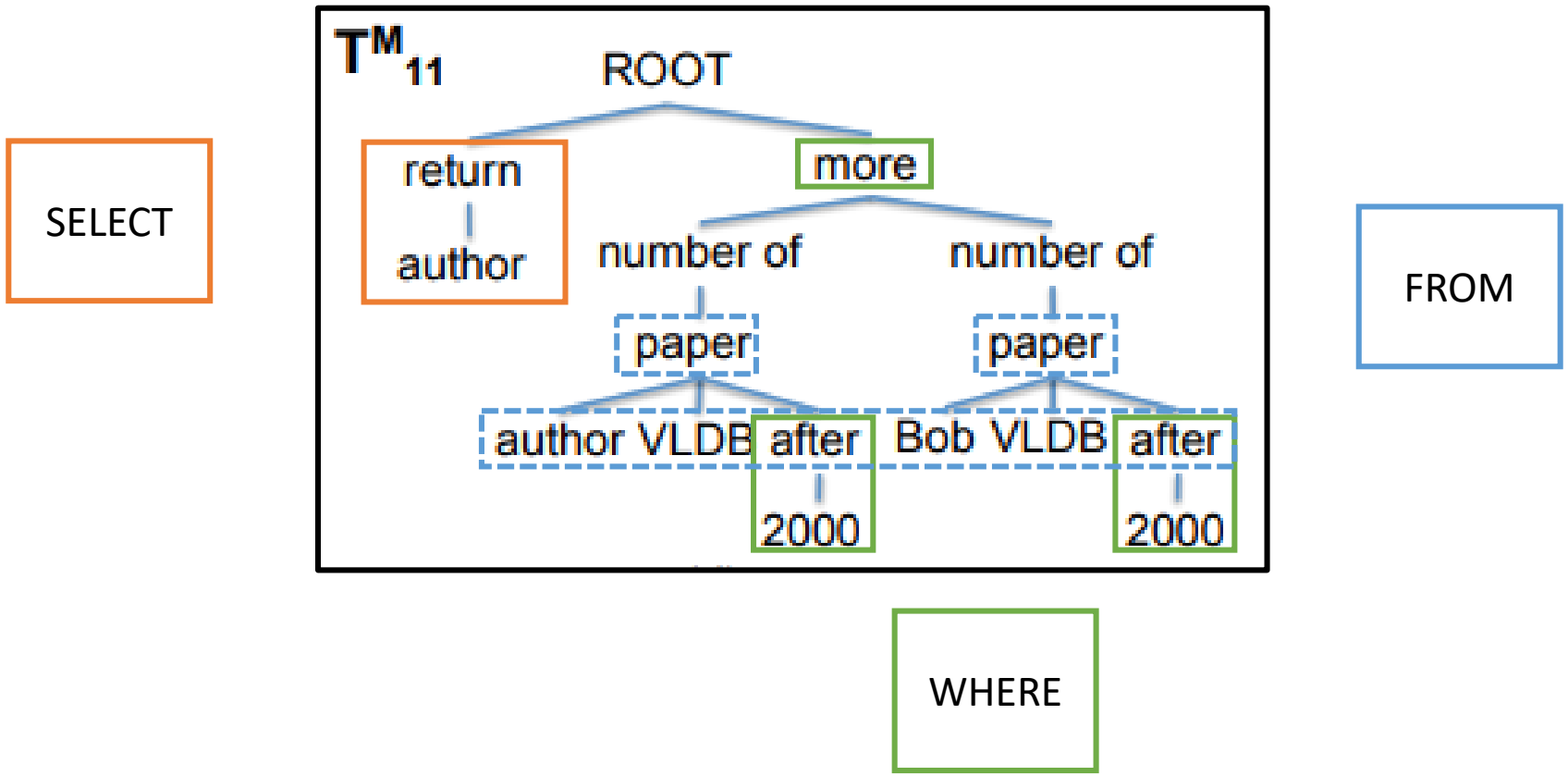
- Introduction
- Contribution
- System Overview
- System Components
- Parse Tree Node Interpretation
- Parse Tree Structure Adjustment
- SQL Generation
- Experiments
- Conclusions

System Components





SQL Generation
Basic Translation



Talk Outline

- Introduction
- Contribution
- System Overview
- System Components
- Parse Tree Node Interpretation
- Parse Tree Structure Adjustment
- SQL Generation
- Experiments
- Conclusions

Experiments

- NaLIR is a stand-alone interface that can work on top of any RDBMS.
  - MySQL serves as the RDBMS and
  - the Stanford Natural Language Parser as the dependency parser.
- There are two crucial aspects to evaluate:
  - the quality of the returned results (effectiveness) and
  - whether the system is easy to use for non-technical users (usability).
- Effectiveness.
  - effectiveness of NaLIR is the percentage of the queries that were perfectly answered.
- Usability.
  - Measuring the actual time taken by the participants.
  - Post-experiment questionnaire.

Results and analysis

	with Interaction	without Interaction	MAS
Simple:	34/34	26/32	20/33
Medium:	34/34	23/34	18/32
Hard:	20/30	15/32	18/33

Figure 11: Effectiveness.

- NaLIR is more effective than the query mechanism of MAS.
- When the interactive communicator was disabled, the effectiveness of the system decreased significantly,
  - especially as the query tasks became more complex.

Results and analysis

	Mapper	Reformulation	Insertion	Translation
w/o Interaction	15	19	0	0
with Interaction	0	10	0	0

Figure 12: Failures in each Component.

- NaLIR could always correctly detect and insert the implicit parse tree nodes, even without interactive communications with the user.
- When the query tree was correctly generated, NaLIR translated it to the correct SQL statement.
- When the interactive communicator was enabled, the accuracy in the parse tree node mapper improved significantly,
  - the participants were able to recognize the correct mapping from others.
- The accuracy in parse tree structure reformulation was also improved when the interactive communicator was enabled.

Results and analysis

	with Interaction	without Interaction	MAS
Simple:	48	34	49
Medium:	70	42	67
Hard:	103	51	74

Figure 13: Average Time Cost (s).

- When the interactive communicator was enabled it took longer to answer queries.
  - However, not much longer with respect to MAS.
  - Yet the results were more accurate.

Talk Outline

- Introduction
- Contribution
- System Overview
- System Components
- Parse Tree Node Interpretation
- Parse Tree Structure Adjustment
- SQL Generation
- Experiments
- Conclusions

Conclusion

- NaLIR is suitable for naïve users.
- Enables the users to accomplish logically complex query tasks
  - in which the target SQL statements include comparison predicates, conjunctions, quantifications, multi-level aggregations, nestings etc.
- Follow-up works:
  - Provenance for Natural Language Queries, Deutch et. Al, VLDB 2017

Questions?