ORA-Semantics based Keyword Search in XML and Relational Databases

Tok Wang Ling, Zhong Zeng, Thuy Ngoc Le, and Mong Li Lee
School of Computing
National University of Singapore
{lingtw, zengzh, ltngoc, leeml}@comp.nus.edu.sg

***Abstract*—Keyword search in XML and relational databases (RDB) has gained popularity as it provides a user-friendly way to explore structured data. Existing works on XML and RDB keyword search only rely on the structures of XML/RDB data and/or schemas, and this causes serious problems of returning incomplete answers, meaningless answers and overwhelming answers. In this paper, we identify the problems of existing keyword search methods and point out that the main reason of these problems is due to the unawareness of the Object-Relationship-Attribute (ORA) semantics in XML/RDB. We exploit the ORA semantics in XML and RDB, and capture these semantics by constructing the Object tree for XML, and the Object-Relationship-Mixed (ORM) data graph for RDB, respectively. Based on the Object tree and the ORM data graph, we propose an ORA-Semantics based keyword search in XML and RDB. Our semantic approach can avoid the problems of existing methods and improves the completeness and correctness of keyword search. In addition, we extend the keyword query language to include keywords that match the metadata, i.e., the names of tags in XML and the names of relations and attributes in RDB. These keywords reduce the ambiguities of queries and enable us to infer user' search intention more precisely. Finally, we incorporate aggregate functions and GROUPBY into keyword queries to retrieve statistical information from XML and RDB.**

I. Introduction

The success of web search engines has made keyword search the most popular search paradigm for ordinary users. As increasing amounts of data over the Internet are stored in XML and relational databases (RDB), the ability to support keyword search in XML and RDB has become important. In contrast to traditional structured queries such as XQuery and SQL, keyword queries enable users to query databases by simple keywords without knowing the database schemas or having to write complicated queries.

Keyword search in XML and RDB has been widely studied. Existing works in XML keyword search typically consider an XML document without ID references and model it as a tree [1], [2]. An answer to a keyword query is defined as an LCA (Least Common Ancestor) of keyword match nodes, or its variants such as SLCA [1] and ELCA [2]. Since these LCA-based approaches only rely the hierachical structure of the XML tree and ignore the semantics of objects, relationships and their attributes in the data, they suffer from serious problems such as meaningless answers, duplicated answers, missing answers, and so on.

Existing works in RDB keyword search can be classified into data graph approach [3], [4] and schema graph approach [5], [6]. In data graph approach, an RDB is represented as a graph where each node represents a tuple and each edge represents a foreign key-key reference. An answer to a keyword query is typically defined as a minimal connected subgraph (Steiner tree) which contains all the keywords. On the other hand, schema graph approach considers an RDB as a schema graph where each node represents a relation and each edge represents a foreign key-key constraint. Based on the schema graph, it translates a keyword query into a set of SQL statements, and leverages on RDBMSs to evaluate the statements and retrieve answers. Similarly, both data graph approach and schema graph approach rely on foreign key-key references of the RDB and do not consider the semantics of objects and relationships as well as their attributes in the data. As a result, they suffer from problems of returning incomplete answers, meaningless answers and overwhelming answers.

In this paper, we investigate existing works on XML and RDB keyword search and identify their serious problems of returning incomplete answers, meaningless answers and overwhelming answers. The main reason of these problems is due to the unawareness of the Object-Relationship-Attribute (ORA) semantics in XML/RDB data. To capture the ORA semantics, we construct the Object tree for XML and the Object-Relationship-Mixed (ORM) data graph for RDB. Keyword queries are processed via the Object tree and the ORM data graph to avoid the problems of existing works.

In addition, we extend the keyword query language to include keywords that match the metadata, that is, the names of tags in XML and the names of relations and attributes in RDB. These keywords provide the context of subsequent keywords and explicitly indicate the search target of the queries. Thus, the ambiguities of keyword queries are significantly reduced.

Finally, we incorporate aggregate functions and GROUPBY into keyword queries to retrieve statistical information from XML and RDB. Given a query with aggregates and GROUPBY, we utilize the ORA semantics to obtain the various query interpretations. Further, we distinguish objects with the same attribute value and detect duplications of objects and relationships to compute the aggregate functions correctly.

II. Problems of current approaches

A. *Problems of current XML keyword search*

We choose the LCA-based approach as a representative and use the XML data in Fig. 1 to illustrate the problems of current XML keyword search. More details can be found in [7].

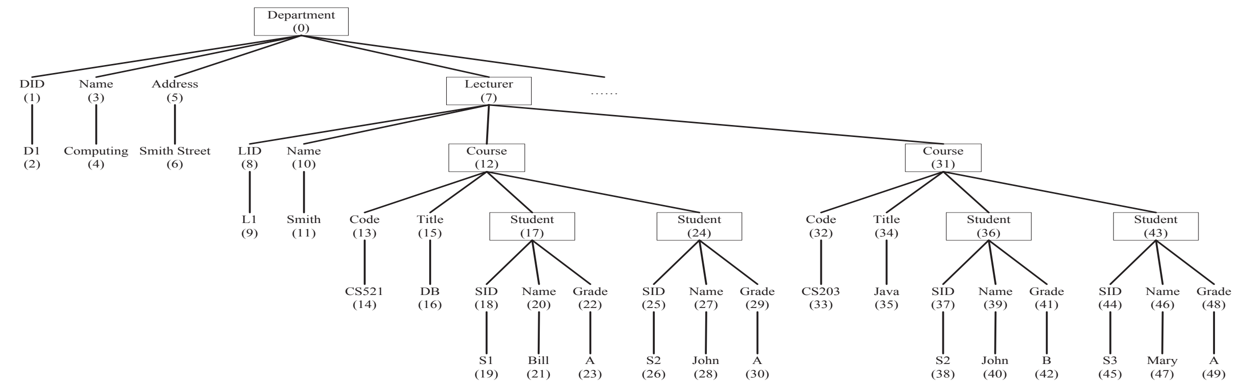

Fig. 1: University.xml

Meaningless answer. Consider query {`Bill`}. The LCA-based approach returns node `Bill_(21)`. However, this is not useful since it does not provide any relevant information about `Bill`. This happens when a returned node is a non-object node, e.g., an attribute or a value. The reason is that the LCA-based approach cannot differentiate object and non-object nodes. Returning object node is meaningful whereas returning non-object node is not. The expected answer should be `Student_(17)`, the object w.r.t. to `Bill_(21)` since it contain relevant information about `Bill`.

Missing answer. Consider query {`DB Java`}. The LCA-based approach only returns node `Lecturer_(7)`. However, it can never recognize that, `Student_(24)` and `Student_(36)` refer to the same object `Student S2`. This is the common student taking the `DB` and `Java` courses. The LCA-based approach should also return the common student taking these two courses, namely, `Student S2` appearing as `Student_(24)` or `Student_(36)` as an answer.

Duplicated answer. Consider query {`S2 John`}. Two answers `Student_(24)` and `Student_(36)` of this query are duplicated because the two nodes refer to the same object `Student S2`. This problem is caused by the unawareness of duplication of object having multiple occurrences. Users expect that either of `Student_(24)` or `Student_(36)` should be returned, but not both since they are different occurrences of the same object `Student S2`.

Problems related to relationships. Consider query {`Bill A`}. The LCA-based approach returns node `Student_(17)`. This answer is incomplete because `A` grade is not an attribute of a student, but it is grade of a student taking a course instead. In other words, `Grade` is a relationship attribute between `Student` and `Course`, not an object attribute. The LCA-based approach cannot distinguish between an object attribute and a relationship attribute under an object node. The proper answer should be the student `Bill` taking the `Course_(12)` and obtaining an `A` grade. To do that, the answer should be moved up to contain other objects (i.e., `Course_(12)`) participating in the relationship that `A` grade belongs to.

Inconsistent types of answers. Consider query {`S1 S2`} and query {`S1 S3`}. Both queries comprise of two student ids. However, the LCA-based approach returns answer {`Course_(12)`} for the first query and answer {`Lecturer_(7)`} for the second query. These two answers refer to objects of different classes and users may get confused as the queries are similar. The reason is that the LCA-based approach does not interpret users' search intention and blindly returns the LCAs of keyword match nodes.

Schema dependence. There may be several designs for the same data source. The XML data in Fig. 1 can be represented by another design where `Student` objects become the parents of `Course` objects. Since the LCA-based approach replies on the hierarchical structure of the XML data, it may return different answers for different designs even though these designs refer to exactly the same information and we are dealing with the same query.

B. *Problems of current RDB keyword search*

We choose data graph approach as a representative to illustrate the problems of current RDB keyword search. More details can be found in [8]. Note that schema graph approach suffers from similar problems. Let us consider the sample RDB in Fig. 2 and the corresponding data graph in Fig. 3.

Student

SID	Name
S1	Bill
S2	John
S3	Mary

Course

Code	Title	LID
CS301	IR	L2
CS521	DB	L1
CS203	Java	L1

Department

DID	Name	Address
D1	Computing	Smith Street
D2	Business	Queen Street

Lecturer

LID	Name	DID
L1	Smith	D1
L2	Smith	D2
L3	Steven	D1

Qualification

	DID	Degree	Major	University
Q1	L1	PhD	CS	NUS
Q2	L3	PhD	CS	SMU
Q3	L3	Master	EE	NTU

Enroll

	SID	Code	Grade
E1	S1	CS521	A
E2	S2	CS203	B
E3	S2	CS521	A
E4	S3	CS203	A
E5	S3	CS301	B

Fig. 2: University database

Incomplete object answer. Suppose a user issues the keyword query {`Steven`} to retrieve all the information about him. Existing works only return his id and name, that is, the tuple $L3$ in the `Lecturer` relation. However, information about the degrees, majors and universities of `Steven`, which are stored in the `Qualification` relation, are not retrieved.

Incomplete relationship answer. Suppose a user wants to know the information of the course where a student `Bill`

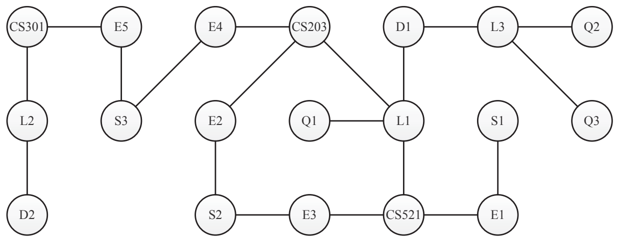

Fig. 3: The data graph for the RDB in Fig. 2

obtains grade `A`, and issues the keyword query {`Bill A`}. Existing works retrieve a Steiner tree which contains the tuples $S1$ and $E1$, as the two query keywords occur in these tuples respectively and there exists a foreign key reference between them. This answer is not informative as details such as the course id and title are not retrieved.

Meaningless answer. Suppose a user issues the keyword query {`S1 S3`}. Existing works returns two answers: (a) $S1-E1-CS521-L1-CS203-E4-S3$ and (b) $S1-E1-CS521-E3-S2-E2-CS203-E4-S3$. The first answer indicates that student `S1` is enrolled in the course `CS521` and student `S3` is enrolled in the course `CS203`. Both the courses are taught by the same lecturer `L1`. The second answer means that student `S2` is enrolled in the same course `CS521` as `S1`; `S2` is also enrolled in the same course `CS203` as `S3`. We observe that the second answer is most likely meaningless to the user.

Difficult to understand the meanings of answers. Given a query answer which is a Steiner tree, e.g., $S1-E1-CS521-L1-CS203-E4-S3$ for query {`S1 S3`}, the user may feel difficult to understand. This is because the answer may consists of many nodes that are connected in a complex structure.

Inconsistent types of answers. Similar to XML keyword search, existing RDB keyword search methods return inconsistent types of answers for similar queries. For example, existing works retrieve answer $S1-E1-CS521-E3-S2$ for query {`S1 S2`} and answer $S1-E1-CS521-L1-CS203-E4-S3$ for query {`S1 S3`}. Clearly, the second answer is far more complex than the first one and the user may get confused for the difference between answers to the two similar queries.

Schema dependence. Given the same data source, the relations in RDB are often denormalized to improve runtime performance. This denormalization leads to data duplication and affects the database schema. Existing works do not consider unnormalized relations in RDB, and thus may suffer from the problems of duplicated answers and missing answers.

For example, suppose we join the `Student`, `Enrol` and `Course` relations in Fig. 2 and obtain an unnormalized relation `Enrolment(SID, Name, Code, Title, LID, Grade)` to store information of students, courses and the many-to-many relationships between students and courses. Given the keyword query {`Bill`}, the existing works will retrieve duplicated answers as information of student `Bill` are duplicated in the `Enrolment` relation. On the other hand, the data graph of this relation has no edges because of no foreign key reference. The existing works will retrieve no answers for query {`S1 S3`}.

Summary. Existing approaches on XML and RDB keyword search highly depend on the database and their schemas, i.e., the hierarchical structure of XML and the foreign key-key references of RDB, and do not consider the ORA semantics in the data. As a result, they cannot interpret keyword queries and fail to retrieve answers that satisfy users' search intention.

III. Our ORA-Semantics based approach

In this section, we exploit the ORA semantics and utilize it to solve the problems of current XML and RDB keyword search in Sec.II. The concept of ORA semantics includes the identification of objects, relationships and attribute values in XML document and RDB, and includes the identification of object classes, relationship types and object/relationship attributes in XML schema and RDB schema.

A. ORA semantics in XML

The ORA semantics in XML are discovered in our work [9]. Based on the ORA semantics, we construct an Object tree for the XML data by keeping only object nodes and associating all non-object nodes to the corresponding object nodes. Compared to the original XML, the Object tree contains a much smaller number of nodes and every node represents an object. Fig. 4 shows the Object tree for the XML in Fig. 1.

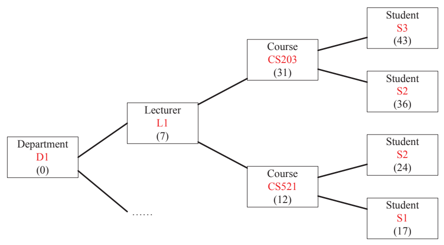

Fig. 4: The Object tree for the XML in Fig. 1

Given a keyword query, we search for LCOAs (lowest common object ancestor) over the Object tree. Since each LCOA contains all the information of an object, we can avoid returning meaningless answers. We also search for HCODs (highest common object descendant) to find answers that are missed by LCOAs and propose algorithms to filter duplicated answers on the fly. Finally, we propose CRs (common relative) to perform a schema independent keyword search. More details are given in [10], [11].

B. ORA semantics in RDB

Our work [8] captures the ORA semantics in RDB by classifying relations into object relations, relationship relations, mixed relations and component relations. An object (relationship resp.) relation captures the information of objects (relationships resp.). The multivalued attributes of an object class (relationship type) are captured in component relations. A mixed relation contains information of both objects and relationships, which occurs when we have a many-to-one relationship. Then we construct an Object-Relationship-Mixed (ORM) data graph that consists of object, relationship and mixed nodes. Each node includes some tuple in the corresponding relation. Tuples in component relations are attached to their corresponding object (relationship or mixed) type nodes. Two nodes are connected via an edge if there exists a foreign key-key reference between tuples in the nodes. Fig. 5 shows the ORM data graph for the RDB in Fig. 2.

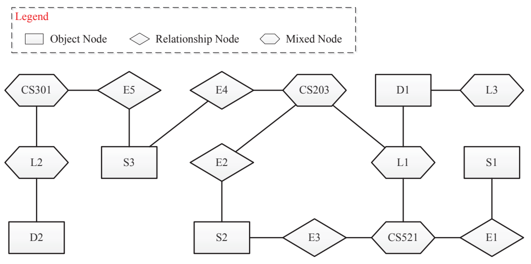

Fig. 5: The ORM data graph for the RDB in Fig. 2

We search over the ORM data graph and process keyword queries based on the types of keyword match nodes. The information of objects and relationships in the ORM data graph enable us to retrieve more complete and informative answers.

IV. Extended keyword query language

A keyword query may have multiple interpretations due to its inherent ambiguity. As existing works do not consider the interpretations of the query, they often retrieve overwhelming answers with various interpretations mixed together.

We observe that when a user issues a query, s/he must have some particular search intention in mind. This motivates us to extend the keyword query language so that users can explicitly indicate his/her search intention whenever possible. In particular, we include keywords that match the metadata, that is, the names of tags in XML and the names of relations and attributes in RDB. These keywords provide the context of subsequent query keywords and reduce the query ambiguities.

Consider query {`Lecturer Smith`} on the XML data in Fig.1. The keyword `Smith` refers to a lecturer name or a department address. However, since the keyword `Lecturer` matches a tag name that specifies a lecturer object, we deduce that the user is more likely to be interested in a lecturer called Smith than the address of a department.

To process an extended keyword query, we utilize the ORA semantics to determine the objects/relationships referred to by the keywords and discover the various ways these objects interact with each other via relationships. We infer users' possible search intentions and rank them. The top-k ranked search intentions are used to retrieve the answers from the database. Answers are grouped by query interpretations with English descriptions. More details are described in [12].

Our second extension to keyword queries is to incorporate the aggregate functions and GROUPBY. We call these aggregate queries. Aggregate queries provide a powerful mechanism to retrieve statistical information from XML and RDB. For instance, suppose we want to know the number of courses taught by the lecturer `Smith` in Fig. 2, we can issue the aggregate query {`Smith COUNT Course`}.

The work in [13] supports aggregate queries on RDB. However, it does not have concepts of objects and cannot distinguish objects with the same attribute value, e.g., two lecturers called `Smith`. Thus, it may return incorrect answers.

Given an aggregate query, we classify the query keywords into simple keywords and aggregates/GROUPBY. We use the simple keywords to interpret the various query interpretations based on the ORA semantics, and then apply the aggregates and GROUPBY on appropriate object/relationship attributes. For each query interpretation, we further utilize the ORA semantics to distinguish objects with the same attribute value and detect duplications of objects and relationships in order to compute the aggregate functions correctly. We show that without the ORA semantics, it is impossible to process aggregate queries correctly. More details are described in [14], [15].

V. Conclusion

We have investigated existing works on XML/RDB keyword search and identified their serious problems of returning incomplete answers, meaningless answers and overwhelming answers. The main reason of these problems is due to unawareness of the ORA semantics. We exploit the ORA semantics in XML/RDB and capture these semantics using the Object tree for XML and ORM data graph for RDB respectively. Keyword queries are processed via the semantic tree/graph to avoid the problems of existing works. In addition, we extend keyword queries to include keywords that match the metadata, i.e., the names of tags in XML and the names of relations and attributes in RDB to reduce query ambiguities. Finally, we incorporate aggregate functions and GROUPBY into keyword queries to retrieve statistical information from XML and RDB.

References

[1] Y. Xu and Y. Papakonstantinou, "Efficient keyword search for smallest LCAs in XML databases," in *SIGMOD*, 2005.
[2] R. Zhou, C. Liu, and J. Li, "Fast ELCA computation for keyword queries on XML data," in *EDBT*, 2010.
[3] A. Hulgeri and C. Nakhe, "Keyword searching and browsing in databases using BANKS," in *ICDE*, 2002.
[4] B. Ding, J. X. Yu, S. Wang, L. Qin, X. Zhang, and X. Lin, "Finding top-k min-cost connected trees in databases," in *ICDE*, 2007.
[5] V. Hristidis and Y. Papakonstantinou, "Discover: keyword search in relational databases," in *VLDB*, 2002.
[6] Y. Luo, X. Lin, W. Wang, and X. Zhou, "SPARK: top-k keyword query in relational databases," in *SIGMOD*, 2007.
[7] T. N. Le, H. Wu, T. W. Ling, L. Li, and J. Lu, "From structure-based to semantics-based: Effective XML keyword search," in *ER*, 2013.
[8] Z. Zeng, Z. Bao, M. L. Lee, and T. W. Ling, "A semantic approach to keyword search over relational databases," in *ER*, 2013.
[9] L. Li, T. N. Le, H. Wu, T. W. Ling, and S. Bressan, "Discovering semantics from data-centric XML," in *DEXA*, 2013.
[10] T. N. Le, W. T. Ling, H. V. Jagadish, and J. Lu, "Object semantics for xml keyword search," in *DASFAA*, 2014.
[11] T. N. Le, Z. Bao, and W. T. Ling, "Schema-independence in xml keyword search," in *ER*, 2014.
[12] Z. Zeng, Z. Bao, T. N. Le, M. L. Lee, and W. T. Ling, "ExpressQ: Identifying keyword context and search target in relational keyword queries," in *CIKM*, 2014.
[13] S. Tata and G. M. Lohman, "SQAK: Doing more with keywords," in *SIGMOD*, 2008.
[14] Z. Zeng, M. L. Lee, and W. T. Ling, "Answering keyword queries involving aggregates and groupby on relational databases," in *EDBT*, 2016.
[15] T. N. Le, Z. Bao, W. T. Ling, and G. Dobbie, "Group-by and aggregate functions in xml keyword search," in *DEXA*, 2014.