Survey on Keyword Search over XML Documents

Thuy Ngoc Le [1], Tok Wang Ling [2]
National University of Singapore
[1]ltngoc@u.nus.edu, [2]lingtw@comp.nus.edu.sg

ABSTRACT

Since XML has become a standard for information exchange over the Internet, more and more data are represented as XML. XML keyword search has been attracted a lot of interests because it provides a simple and user-friendly interface to query XML documents. This paper provides a survey on keyword search over XML document. We mainly focus on the topics of defining semantics for XML keyword search and the corresponding algorithms to find answers based on these semantics. We classify existing works for XML keyword search into three main types, which are tree-based approaches, graph-based approaches and semantics-based approaches. For each type of approaches, we further classify works into sub-classes and especially we summarize, make comparison and point out the relationships among sub-classes. In addition, for each type of approach, we point out the common problems they suffer.

1. INTRODUCTION

Since XML has become a standard format for data representation and data exchange over the Internet, it has wide applications such as electronic business, science, text databases, digital libraries, healthcare, finance, and even in the cloud [3]. As a result, XML has attracted a huge of interests in both research and industry with a wide range of topics such as XML storage, twig pattern query processing, query optimization, XML view, and XML keyword search. There have been several XML database systems such as Timber [15], Oracle XML DB[1], MarkLogic Server[2], and the Toronto XML Engine[3].

As XML has become more and more popular and the volume of XML data is increasing, keyword search in XML data has attracted a lot of research interests. Given a set of keywords in a keyword query, XML keyword search aims to find the most relevant information with the input keywords over the corresponding XML document. Approaches for XML keyword search can be classified into three types: *tree-based approaches* for XML documents with no IDREF (usually modeled as a tree), *graph-based approaches* for XML documents with IDREFs (usually modeled as a graph), and *semantics-based approaches* for both XML document with and with no IDREF.

For tree-based approaches, the typical solution is based on the LCA (Lowest Common Ancestor) semantics, which was first introduced in [11]. LCA-based approaches search for the lowest common ancestors of nodes matching keywords. Many subsequent works either enhance the efficiency [40, 6] or the effectiveness of the search by adding reasonable constraints to the LCA definition to filter less meaningful LCA results such as SLCA [36], ELCA [41], VLCA [24] and MLCA [27]. In Section 2, we will discuss in details these approaches. Moreover, we will make comparison and show relationships among these approaches. Additionally, we will point out problems these approaches commonly suffer and discuss the reasons behind.

For graph-based approaches, the search semantics are mainly based on Steiner tree/subgraph and can be classified into (1) directed tree, (2) bi-directed tree and (3) subgraph. Directed and bi-directed Steiner tree semantics are applied for directed graph [9, 12], while subgraph semantics are applied for undirected

[1]http://www.oracle.com/technetwork/database-features/xmldb/overview/index.html
[2]http://www.marklogic.com/
[3]http://www.cs.toronto.edu/tox/

graph [25, 17, 28, 8]. More details about these works will be reviewed in Section 3. Similar to the tree-based approaches, beside describing graph-based approaches, we make comparison, show relationships, and point out problems of these approaches.

For semantics-based approaches, researchers have exploited the semantics of Objects, Relationships among objects, Attributes of objects, and Attribute of relationships (referred to as *ORA-semantics*) to improve the effectiveness, the efficiency and the expressiveness of XML keyword search. The ORA-semantics is defined as the identifications of nodes in XML data and schema. More information about the semantics-based approaches will be studied in Section 4. We will also discuss on how exploiting semantics helps solve problems of the tree-based and graph-based approaches.

Although several surveys [34, 31, 35, 38, 5] have been done for XML keyword search, to the best of our knowledge, no survey can clearly show the relationships among existing approaches or discuss problems of each type of approaches. In this survey, we not only present existing works, but we also classify them, make comparison, show their relationships, and especially point out the problems they commonly suffer.

2. TREE-BASED APPROACHES FOR XML KEYWORD SEARCH

When XML documents do not contain IDREF, they can be modeled as trees. Approaches to handle such documents are called tree-based approaches because they are based on tree model. Inspired by the hierarchical structure of the tree model, most of existing tree-based approaches are based on the LCA (Lowest Common Ancestor) semantics, which returns the lowest common ancestors of matching nodes to keyword queries. There are many subsequent semantics to filter less meaningful answers. Existing works either improve the effectiveness by proposing a new semantics or improve the efficiency by proposing a new method for a certain semantics. The widely accepted LCA-based semantics include LCA itself, SLCA, VLCA, MLCA, ELCA, and etc, among which, SLCA and ELCA are the most popular semantics. We classify the existing research works into these semantics and the result of our classification is shown in Figure 1. Some research works study more than one semantics such as XRANK [11], Set-intersection [40], and Top-K [4]. In Section 2.7, we will summarize the discussed semantics, show their relationships, and use the same example to demonstrate them and their differences.

2.1 LCA Semantics

The LCA semantics for XML keyword search was first proposed in XRANK [11]. By the LCA semantics, for a set of matching nodes, each of which contains at least one query keyword and each query keyword matches at least one node in this set, the lowest common ancestor (LCA) of this set is a returned node. An answer is a subtree rooted as a returned node (i.e., an LCA node) or a path from a returned node to matching nodes. XRANK is extended from Googles Pagerank algorithm for ranking. It takes into account the proximity of the keywords and the references between attributes. XRANK implements a naive approach, and three optimized approaches afterwards to improve the search.

2.2 SLCA Semantics

The SLCA (Smallest LCA) semantics was first proposed in XKSearch [36]. The SLCA semantics defines an SLCA to be an LCA that does not have any other LCAs as its descendants. There are many works on finding the set of SLCAs for a keyword query.

XKSearch [36] proposes two efficient algorithms to compute SLCAs, namely Indexed Lookup Eager and Scan Eager. To find all SCLAs, there are two tasks, namely finding all LCAs and remove all ancestors among LCAs to get the SLCAs. It is costly to find all LCAs. When the number of keywords and the number of matching nodes for each keyword are increased, the number of combinations is huge. XKSearch optimizes as follows. Firstly, for each matching node u of the keyword which has the least number of matching nodes, XKSearch finds its left match and right match. Therefore, given two keywords k_1, k_2 and a node u that contains keyword k_1, one needs not inspect the whole node list of keyword k_2 in order to discover potential solutions. Instead, one only needs to find the left and right match of u in the list of k_2, where the left (right) match is the node with the greatest (least) Dewey ID (identifier) that is smaller (greater) than or equal to the Dewey ID of u.

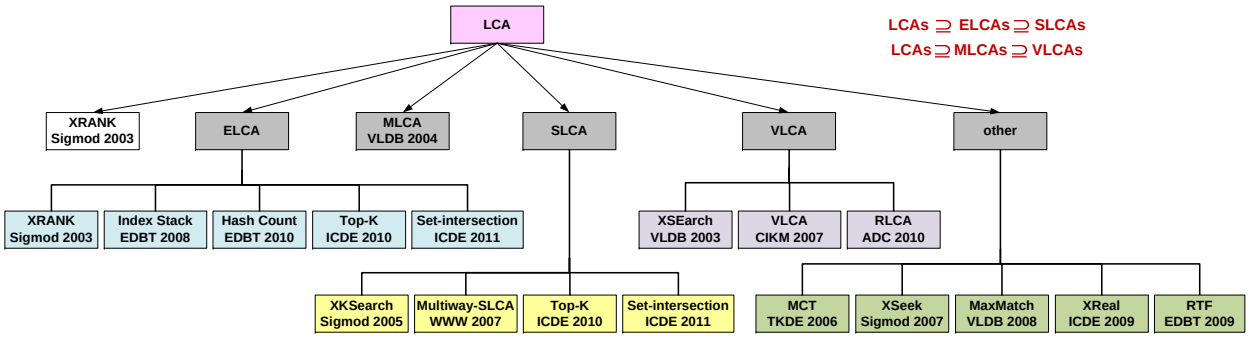


Figure 1: Our classification for tree-based approaches based on the semantics used

Multiway-SLCA [6] further optimizes the performance of XKSearch computation. The key motivation behind this approach is to avoid redundant steps of XKSearch where SLCAs are computed by computing many intermediate SLCA. Multi-way SLCAs approach computes each potential SLCA by taking one data node from each keyword list in a single step instead of breaking the SLCA computation into a series of intermediate SLCA computations.

Top-k [4] studies how to support efficient top-k XML keyword query processing based on the JDewey labeling scheme, where each component of a JDewey label is a unique identifier among all the nodes at the same depth. According to this property, the proposed Join-based algorithms perform set intersection operation on all lists of each tree depth from the leaf to the root.

Set-intersection [40] presents a novel method to find SLCAs. The basic idea is that common ancestors derived from any two keywords are the intersection of the two sets of nodes matching those keywords. After finding common ancestors, it creates a tree containing all common ancestors. Leaves of this tree are SLCAs.

2.3 ELCA Semantics

The ELCA (Exclusive LCA) semantics is also widely accepted. ELCAs is a superset of SLCAs, and it can find some relevant information that SLCA cannot find. An ELCA is an LCA with its own witnesses, i.e., matching nodes. In other words, consider a node u, if u contains matching nodes of all query keywords after removing all subtrees rooted at its descendant ELCAs, then u is an ELCA. This semantics is first introduced in XRANK [11] with the DeweyInvertedList algorithm, which reads match nodes in a preorder traversal, and uses a stack to simulate the postorder traversal. Many other algorithms are proposed to find ELCAs of a keyword query.

[37] proposes an Index Stack algorithm to find ELCAs more efficiently. The algorithm to find all the ELCAs can be decomposed into two steps: first find all ELCA candidates, and then find ELCAs in those candidates. The first step can be leveraged the algorithm IndexedLookupEager in XKSearch [36].

[41] presents an efficient algorithm to find ELCAs named HashCount. This algorithm can be divided into two subtasks: firstly, it finds out ELCA candidates; and then it verifies these candidates, discard the false positives and obtain the real results. Note that this framework is the same as the Indexed Stack algorithm in [37], but techniques used are different.

Besides proposing algorithms for finding SLCAs, Top-k [4] and Set-intersection [40] also presents algorithms for finding ELCAs with the similar methods with those of finding SLCAs.

2.4 VLCA Semantics

The VLCA (Valuable LCA) semantics is introduced in [24]. According to the VLCA semantics, any two matching nodes in an answer must be homogeneous, that is there are no two nodes of the same elementary type (i.e., label, tag) on the paths connecting the two matching nodes, except themselves. Two algorithms, the Brute-Force algorithm and the Stack-based algorithms are proposed in [24] to finds VLCAs for a keyword query. There are two variants of VLCA semantics, namely XSEarch [7] and RLCA (Relevant LCA) [32].

In XSEarch [7], the whole algorithm is based on a property, called interconnection. The intuition of such a property is that it differentiates the attributes that belongs to different entities. XSEarch try to find sets of match nodes, such that each set contains all keywords and every two keywords in a set is interconnected. XSEarch returns the path of each set as the search

result. However, the complexity is *NP-complete*. So XSEarch only requires that each node in one set should be interconnected with one node. This looser condition is called star-interconnected and makes it possible to find all the results in polynomial time.

RLCA [32] is similar to XSEarch. RLCA is different from XSearch into two aspects: (1) it accepts that two nodes with the same type can be meaningfully connected in a subtree, due to the fact that a user may be interested in finding more than one entity with the same type. (2) For queries related to only single entity, RLCA uses node types to detect the relevancy of fragments rather than simply uses node labels. Hence, it can detect that some nodes are still homogeneous although there are some nodes of the same types on the path connecting them, such as the two attributes of the same object type.

2.5 MLCA Semantics

Meaningful LCA (MLCA) [27] introduces the concept of meaningful relationship between two nodes. According to the MLCA semantics, two nodes are meaningfully related to each other if (1) they have the hierarchical relationship (ancestor-descendant relationship), or (2) the two nodes belong to the same types, or (3) the LCA of matching nodes in the data tree belongs to the LCA of their node types in the schema tree. Otherwise, the two nodes are not meaningful. An MLCA is an LCA in which any two matching nodes have a meaningful relationship.

Although the MLCA semantics is similar to the VLCA semantics, conditions of the MLCA semantics is looser than that of the VLCA semantics. They have two main differences. Firstly, for MLCA, two matching nodes of the same types always provide a meaningful answer, while for VLCA, the meaningful answer still depends on whether any nodes between them are of the same type. Secondly, for VLCA semantics, there must be no two nodes on the paths connecting matching nodes are of the same type, while for MLCA semantics, the nodes on the paths connecting matching nodes cannot be of the same type with matching nodes only.

2.6 Other Semantics

MCT [13] introduces MCT (minimum connecting tree) of a set of nodes to be a minimum subtree that connects all nodes of that set. The root of the subtree is an LCA. The advantage of MCT is to exclude irrelevant information which is not related to keywords.

XReal [1] applies idea from information retrieval. It exploits the statistics of underlying XML database to identify the search target nodes, keyword ambiguity and relevance oriented ranking. Firstly, it finds the node type which is most likely users is searching for. That *search for* nodes should contain all the keywords in subtrees and not to be deeply nested in the XML. Secondly, it determines the node type which is most likely to be the correspondent to each keyword. After that, it computes the similarity between an XML node and the query for ranking.

An answer of a keyword query has two parts: the returned node (defined by the semantics) and output presentation (which information should be returned with the returned node). XSeek [29] focuses on the second part. XSeek uses some heuristics to identify the appropriate data nodes to be returned after the connection between the matches is already established.

MAXMATCH [30] provides the first novel algorithm that satisfies four properties of data monotonicity, query monotonicity, and data consistency and query consistency. For data Monotonicity, if we add a new node to the data, then the data content becomes richer, therefore the number of query results should be (non-strictly) monotonically increasing. For query Monotonicity, if we add a keyword to the query, then the query becomes more restrictive, therefore the number of query results should be (non-strictly) monotonically decreasing. For data consistency, after a data insertion, each additional subtree that becomes (part of) a query result should contain the newly inserted node. For query consistency, if we add a new keyword to the query, then each additional subtree that becomes (part of) a query result should contain at least one match to this keyword.

RTF [19] introduces the concept of Relaxed Tightest Fragment (RTF) as the basic result type. Then it proposes a new filtering mechanism to overcome the two problems in MAXMATCH, which are the false positive problem (discarding interesting nodes) and the redundancy problem (keeping uninteresting nodes).

2.7 Relationship and Comparison on the LCA-based semantics

We classify the existing research works based on the

semantics they apply and the classification has been shown in Figure 1. In addition, we find that for the same query Q, the relationships among the set of answers by the LCA-based semantics are follows:

$$LCA(Q) \supseteq ELCA(Q) \supseteq SLCA(Q) \quad \text{and}$$
$$LCA(Q) \supseteq MLCA(Q) \supseteq VLCA(Q)$$

As can be seen, for the same query Q, the LCA semantics provides the most answers. However, many of them are contained by the other and are not really relevant. Therefore, the other semantics have constraints to filter out such answers. However, they may filter out meaningful answers as well. As a result, no semantics is the best and can beat all the others. Each has its own advantages and disadvantages. We summarize these semantics and the relationships among them in Table 1 and use the following example for illustration.

EXAMPLE 1. *Consider keyword query* $\{Q$ = *Clinton, Kennedy*$\}$ *issued against the XML data tree in Figure 2, in which we circle and label some nodes as (&o1), (&o2), (&o3), (&o4) and (&o5) for discussion. Two nodes (&o4) and (&o5) are LCAs, SLCAs, ELCAs, MLCAs, and VLCAs. LCAs of the query are nodes (&o1), (&o2), (&o3), (&o4) and (&o5). Among LCAs, only the two nodes (&o4) and (&o5) are SLCA nodes while the other do not because they are ancestors of either node (&o4) or node (&o5). Nodes (&o2) and (&o3) are not ELCA nodes either because they do not have their own witnesses. Although, node (&o1) is not an SLCA node, it is an ELCA node because after removing the two nodes (&o4) and (&o5), it still has* `Kennedy` *and* `Clinton` *as its descendants (under node (&o2) and node (&o3)). In this example, all LCA nodes are MLCA nodes. Among LCA nodes, node (&o1) is not a VLCA node because there exists nodes of the same types (student) on the path connecting matching nodes. The remaining nodes are VLCAs. As we can see,* $LCA(Q) \supseteq ELCA(Q) \supseteq SLCA(Q)$ *and* $LCA(Q) \supseteq MLCA(Q) \supseteq VLCA(Q)$. *Returned nodes of the semantics for query* Q *are also summarized in Table 1.*

2.8 Common Problems of the LCA-based Semantics

Although different LCA-based semantics (e.g., LCA, SLCA, ELCA, VLCA, etc) provide different answers,

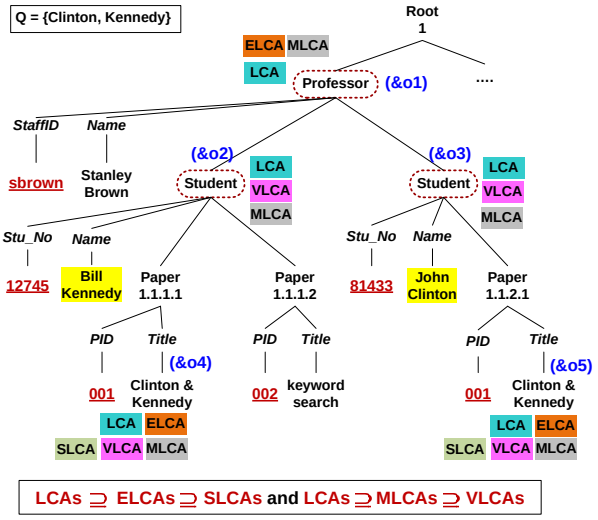


Figure 2: Returned nodes for {Clinton, Kennedy}

they all are based on the concept of LCA. Moreover, they all ignore the semantics of object, relationship, object attribute and relationship attribute (referred to as ORA-semantics). Therefore, we find that they suffer from several common problems. We will systematically point out the common problems of all the LCA-based semantics by comparing answers returned by the LCA-based approaches and answers expected by users. We use the XML data in Figure 3 for illustration. Note that `Course_(11)` and `Course_(35)` refer to the same object `<Course:CS5201>` despite of appearing as different nodes.

Problem 1. Useless answer. Consider Q_1 = {`Bill`}. The LCA-based approaches return node `Bill_(6)`. However, this is not useful since it does not provide any additional 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 approaches do not have the concept of object and attribute and thus cannot differentiate object and non-object nodes. Returning object node is useful whereas returning non-object node is not. The expected answer should be forced up to `Student_(1)`, the object w.r.t. to `Bill_(6)` since it contain additional information related to `Bill` such as `major` and `student_No`.

Problem 2. Missing Answer. Consider an XML keyword query Q_2 = {`Bill, John`} issued to the XML data in Figure 3, in which the query keywords match first name of two students. The LCA-based approaches return the document root as an answer for

Table 1: Our summary on the LCA-based semantics

Semantics	Definition	Existing algorithms	Returned nodes in Example 1
LCA	An LCA is a lowest common ancestor of a combination of matching nodes, i.e., each keyword corresponds to at least one matching node in the combination	XRANK Sigmod 2003	{ &o1, &o2, &o3, &o4, &o5 }
ELCA (Exclusive LCA)	*An ELCA is an LCA of a combination of matching nodes *An ELCA has its own witnesses, i.e., it does not share its matching nodes with its descendant ELCA nodes	*Index Stack EDBT 2008 *Hash Count EDBT 2010 *Top-K ICDE 2010 *Set-intersection ICDE 2011	{ &o1, &o4, &o5 }
SLCA (Smallest LCA)	*An SLCA is an LCA of a combination of matching nodes *There is no LCA node as its descendant	*XKSearch Sigmod 2005 *Multiway-SLCA WWW 2007 *Top-K ICDE 2010 *Set-intersection ICDE 2011	{ &o4, &o5 }
VLCA (Valuable LCA)	*A VLCA is an LCA of a combination of matching nodes *For each pair of matching nodes, all nodes in the path connecting them are of different types.	*XSEarch VLDB 2003 *VLCA CIKM 2007 *RLCA ADC 2010	{ &o2, &o3, &o4, &o5 }
MLCA (Meaningful LCA)	*An MLCA is an LCA of a combination of matching nodes *The LCA of matching nodes in the data tree belongs to the LCA of their node types in the schema tree	*MLCA VLDB 2004	{ &o1, &o2, &o3, &o4, &o5 }

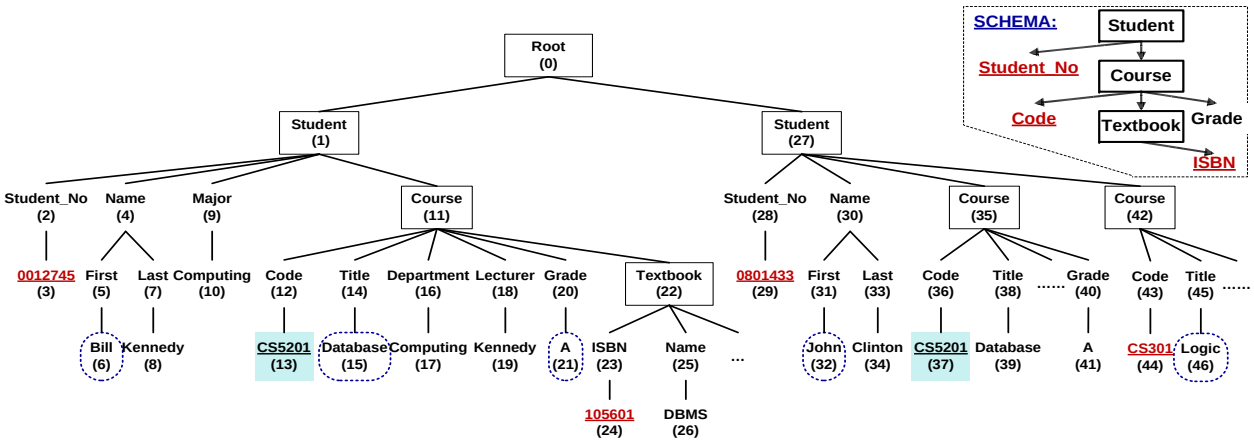


Figure 3: An XML data tree about student and course of a university

Q, which is intuitively meaningless for users because returning the root means returning the whole XML document. Note that two objects are the same if they belong to the same object class and have the same OID value. Then `Course_(11)` and `Course_(35)` refer to the same object `<Course:CS5201>`[4] because they belong to the same object class `Course` and have the same OID value `CS5201`. Therefore, `<Course:CS5201>` is the common course taken by both students `Bill` and `John` and should be an answer. However, the LCA-based approaches miss this answer because they are not aware of object, OID and the duplication of the same object. Thus, the common courses taken by both students are not found.

[4] `<Course:CS5201>` denotes an object which belongs to object class Course and has OID value CS5201.

Problem 3. Duplicated answer. Consider Q_3 = {`CS5201, Database`}, `Course_(11)` and `Course_(35)` are two duplicated answers because the two nodes refer to the same object `<Course CS5201>`. This problem is caused by the unawareness of duplication of object having multiple occurrences. Users expect that either of `Course_(11)` or `Course_(35)` should be returned, but not both since they are different occurrences of the same object `<Course CS5201>`. In reality, if the course has 300 students enrolled, then such answers are duplicated 300 times. This really overwhelms and annoys users.

Problem 4. Incorrect answer. Consider Q_4 = {`Database A`}. The LCA-based approaches return `Course_(11)` and `Course_(35)` as answers. These answers are incorrect because `'A'` grade is not an attribute of a course, but it is grade of a student taking

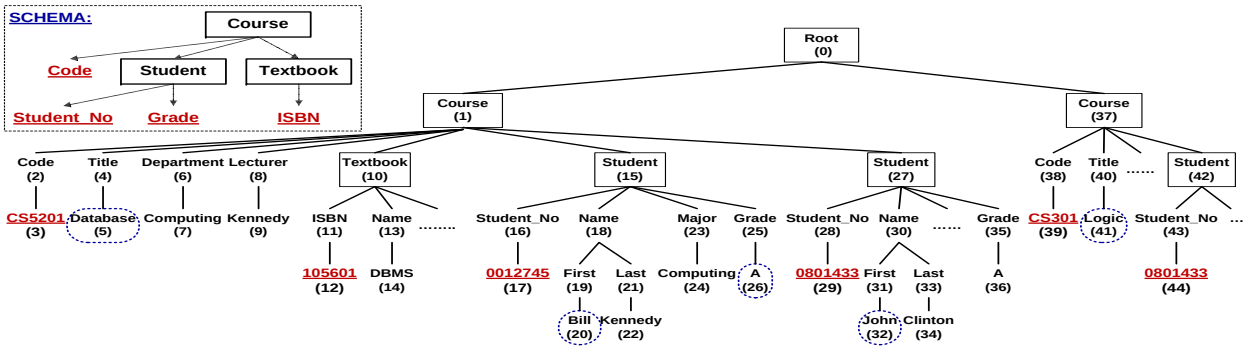


Figure 4: Another design for the university XML data in Figure 3

the course instead. On the other hand, `Grade` is a relationship attribute between `Student` and `Course`, not an object attribute. This is because the LCA-based approaches cannot distinguish between an object attribute and a relationship attribute under an object node. The proper answer should be all students taking course `Database` and getting an `'A'` grade. To do that, the answer should be moved up to contain other objects (e.g., students) participating in the relationship that `'A'` grade belongs to.

Problem 5. Schema-dependent answer. There may be several schema designs with different hierarchical structures for the same data content. The XML data in Figure 3 can also be represented by another design as in Figure 4 with different hierarchical structure among object classes, e.g., `Course` becomes the parent of `Student`. Consider Q_5 = {`Bill, Database`}. With the design in Figure 3, the LCA-based approaches return `Student_(1)`. With the design in Figure 4, `Course_(1)` is returned. As shown, answers for different designs are different though these designs refer to exactly the same information and we are dealing with the same query. This is because answers from the LCA-based semantics rely on the hierarchical structure of XML data. Different hierarchical structures may provide different answers for the same query. Users issue a keyword query without knowledge about the underlying structure of the data. Thus, their expectation about the answers is independent to the schema design. Therefore, the expected answers should also be semantically the same with all designs of the same data content.

Summary. The above problems and their reasons behind are summarized in Table 2. The main reasons of the above problems are the high dependence of answers returned by the LCA-based approaches on the hierarchical structure of XML data (e.g., Q_5), and the unawareness of ORA-semantics. Particularly, unawareness of objects causes *missing answers* (e.g., Q_2), and *duplicated answer* (e.g., Q_3) because the LCA-based approaches cannot discover the same object. Unawareness of object and attribute causes *useless answer* (e.g., Q_1) because it cannot differentiate XML elements (object vs. attribute). Unawareness of relationship causes *incorrect answers* (e.g., Q_4) because of it is unable to know the degree of a relationship type and not differentiate an object attribute and a relationship attribute.

Table 2: Summary of the discussed queries

Query	Keyword	Problem	Reason
Q_1	Bill	Useless answer	unawareness object and attribute, cannot differentiate XML elements
Q_2	Bill, John	Missing answer	unawareness object, cannot discover duplicated objects
Q_3	CS5201, Database	Duplicated answer	unawareness object, cannot discover duplicated objects
Q_4	Database, A	Incorrect answer	unawareness relationship, cannot distinguish relationship attribute and object attribute
Q_5	Bill, Database	Schema-dependent answer	depend on the hierarchy

3. GRAPH-BASED APPROACHES FOR XML KEYWORD SEARCH

ID/IDREF is an XML standard and is often used in XML documents. With IDREF, XML is modeled as a graph because it is no longer a tree. Existing graph techniques can be applied for XML graph-structured data such as [2, 8, 10, 12, 16, 33, 25, 17]. Semantics applied in the existing graph-based approaches can be classified into (1) subtree, (2) subgraph and (3) bi-directed tree.

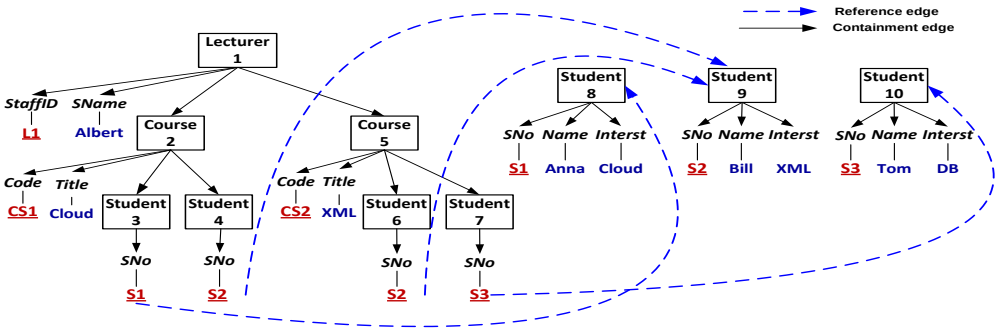
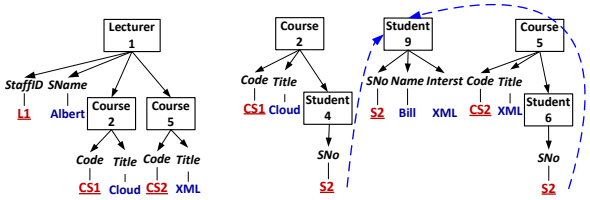


Figure 5: XML data graph



(a) Answer (b) Not answer

Figure 6: Illustration for query {CS1,CS2}

3.1 Subtree based Semantics for Directed Graphs

It is natural to model an XML document as a *directed graph* where *forward edges* (or edges in unambiguous contexts) are parent-child relationships or IDREFs (reference edges). Most approaches for this kind of data model find a *minimal rooted tree* containing all keywords, in which the path from the root to each content node is directed. This kind of semantics includes the *minimal Steiner tree* semantics [9] and the *distinct root* semantics [12]. Intuitively, these semantics are similar to the LCA semantics and they also suffer from the same *problem of missing answers* as the LCA semantics does (discussed in Section 2.8). Particularly, even with IDREF, the common object appearing as the child (or the descendant in general) of two nodes cannot be found by these semantics. This is because the directed tree based semantics only search backward (i.e., follow the reversed direction of the directed edges), but never search forward to find common information which related to all matching nodes.

For example, consider query {`CS1, CS2`} against the directed XML graph in Figure 5, where the keywords match the two objects `course 2` and `course 5`. Note that in this example, we match keywords with the whole object rather than a single value node. Both pieces of information in Figure 6(a) and in Figure 6(b) are meaningful to users. Intuitively, the first one (in Figure 6(a)) means the two courses are taught by Lecturer `Albert`, and the second one (in Figure 6(b)) means the two courses are both taken by Student named `Bill`. However, the directed tree based semantics only return the first one in Figure 6(a), but not able to return the other in Figure 6(b).

3.2 Subgraph based Semantics for Undirected Graphs

An XML document can also be modeled as an undirected graph by ignoring the direction of edges. For undirected graph, an answer is commonly either a subgraph such as the *r-radius* semantics [25] and the *r-clique* semantics [17]; or *minimum cost connected tree* [8][5]. These semantics can provide more answers than the directed tree based semantics do, including common descendants because they search for all directions, rather than just follow the reversed direction of edges as the subtree based semantics do. However, they may also provide answers which can be *hardly interpreted* (or even *meaningless*) because many answers contain matching nodes which are very far or even not related at all.

For example, suppose the XML document in Figure 5 is modeled as an undirected graph by ignoring the direction of edges. Consider keyword query = {`S1, S3`} where the keywords match two students. For this query, a use want to know all relationships between those two students, and their common information such as common lecturers teaching them or common courses taken by them. Figure 7 shows an answer[6] under the subgraph based semantics. This answer means the two

[5]It is actually acyclic subgraph.

[6]For ease of comprehension, we only show objects. Note that both `Student 4` and `Student 6` refer to object `<Student:S2>`.

students study two courses which are both taken by another student. Intuitively, the relationship of the two students is too weak and users do not expect such answer. Although several recent works [25, 17, 28] take the distance between each pair of (content) nodes into account, these works still return such answer because the relationship between the two nodes may still meaningless even the distance between them is not far.

Figure 7: A meaningless answer of the subgraph based semantics

3.3 Bi-directed Tree based Semantics for Directed Graphs

Some works such as [2, 16] model data as directed graph, but they create an *backward edge* corresponding to each forward edge with the reversed direction (probably with lower score for ranking in the backward edge). Thereby, the answer they return can be a subtree with forward edges, or a subtree with backward edges. Some works such as [18] even return a subtree with a mix of forward and backward edges. Such answer is actually a subgraph. Thus it may be meaningless as illustrated in Section 3.2. Edge direction for this work is mainly served in improving efficiency of the search.

3.4 Other Methods based on Graph

XKeyword [14] views an XML document as a directed graph of nodes. The result of a keyword query is the minimal total target object networks which are the minimal graphs involving all query keywords and in which each node is a target object. Since the XML document is stored in relational database, a target object in this paper corresponds to a tuple in relational database, which is not always correct as studied in [39]. This work exploits the properties of the schema of the database to facilitate the result presentation, to find target objects and to optimize the performance of the search system, e.g., reducing search space. XKeyword focuses on the presentation of the result and on techniques to provide fast response time. However, since the schema does not fully contain the ORA-semantics, XKeyword does not discover real relationships among objects, does not distinguish relationship attributes and object attributes, and does not always discover objects correctly.

3.5 Relationship and Comparison on Graph-based Approaches

We summarize existing graph-based approaches, their problems, and classify these approaches based on the semantics they apply in Figure 8. Note that trees are directed. However, some above works use the term undirected trees with the meaning of acyclic graph.

In brief, for the efficiency, the subtree based semantics over directed graph is generally faster than the others because in the directed graph, the search follows only one direction. For the effectiveness, the subtree based semantics may miss a lot of answers because they search for only one direction. The subgraph based semantics can provide more answers, including the missing answers of the subtree based semantics. However, many of the answers provided by the subgraph based semantics are meaningless because the matching nodes are not closely related, or even not related at all.

3.6 Common Problems of the Graph-based Approaches

Besides the problems of each semantics discussed above, in generally, all graph-based approaches suffer from the same problems of the LCA-based approaches (studied in Section 2.8) when not all objects in the XML data are under IDREF mechanism. When all objects are under IDREF mechanism, graph-based approaches can handle some but not all problems of the LCA-based approaches. Particularity, the *incorrect answer* (when handling relationship attributes) and *useless answer* (due to returning non-object nodes) cannot be solved while *missing answer*, *duplicated answer* and *schema-dependent answer* can be solved partly.

We use the XML data in Figure 9 which contains both objects with duplication and objects with IDREFs to illustrate problems of the graph-based search. We apply the widely accepted semantics *minimum Steiner tree* [8, 10] for illustrating the problems. In the XML data in Figure 9, Object `<Employee:HT08>` is duplicated with two occurrences `Employee_(6)` and `Employee_(26)`. Ternary relationship type among `Supplier`, `Project` and `Part` means suppliers supply parts to projects. `Quantity` is an attribute of this ternary relationship and represents the quantity of a part supplied to a project by a supplier. Besides, binary

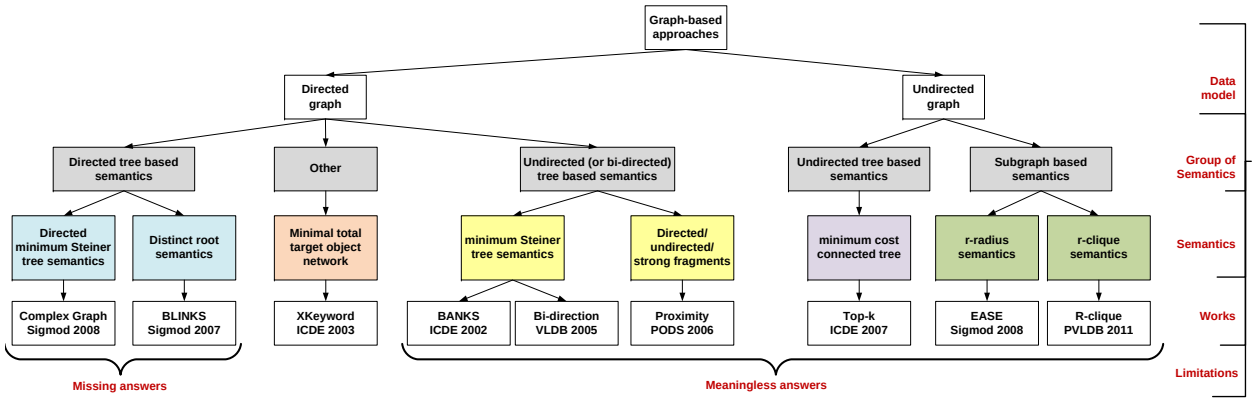


Figure 8: Relationship of Graph-based approaches and the semantics used

relationship between `Supplier` and `Part` has an attribute `Price` to represent the price of a part supplied by a supplier.

3.6.1 *Problems cannot be solved with IDREF*

IDREF mechanism is aware of semantics of object and object ID. However, the semantics of relationship and attribute is still not recognized and utilized which causes the problems of *useless answer*, and *incorrect answer*.

Useless answer. Not differentiating object and non-object nodes cause useless answer when the returned node is a non-object node. For example, for Q_1 = {`Amazon`}, the answer is only `Amazon_(45)` without any other information.

Incorrect answers. Without semantics of relationship, the graph-based search cannot distinguish object attribute and relationship attribute, and cannot recognize n-ary ($n \geq 3$) relationship. These cause problems related to relationship.

For example, for Q_2 = {`PARTA, 100`}, the subtree rooted at `Part_(46)` is an answer. However, this is not complete since price `100` is the price of a part named `PARTA` supplied by `Supplier_(41)`. It is not the price of `Part_(46)`. Thus, the answer should be moved up to `Supplier_(41)` to include `Supplier_(41)` as well.

3.6.2 *Problems can be partly solved with IDREF*

Recall that the problems of *missing answer*, *duplicated answer* and *schema-dependent answer* are caused by lack of semantics of object. Therefore, using IDREF can avoid these problems because IDREF mechanism is based on semantics of object and object ID. However, if IDREF mechanism is not totally applied for all objects, i.e., there exists some duplicated objects, e.g., object `<Employee:HT08>` in Figure 9, then the above problems are not totally solved.

For example, Q_3 = {`Bill, HT08`} has two duplicated answers, `Employee_(6)` and `Employee_(26)`. For Q_4 = {`Prj2012, Prj2013`}, only the subtree containing `Supplier_(41)` can be returned by the graph-based approaches whereas the subtree containing `<Employee: HT08>` is *missed*. If object class `Employee` is designed as the parent of object class `Project`, the missing answer of Q_4 are found. It shows that the graph-based search also *depends on the design of XML schema* in this case.

Summary. The graph-based search can avoid *missing answer*, *duplicated answer* and *schema-dependent answer* only if the IDREF completely covers all objects. Otherwise, the above limitations cannot avoid. The other problems including *useless answer* and *incorrect answer* are still unsolved no matter IDREFs are used or not because IDREF mechanism only considers semantics of object and OID but ignores semantics of relationship and attribute.

4. SEMANTICS-BASED APPROACHES FOR XML KEYWORD SEARCH

Recently, the semantics of Objects, Relationships among objects, Attributes of objects, and Attribute of relationships (referred to as *ORA-semantics*) has been exploited to improve the effectiveness, the efficiency and the expressiveness of XML keyword search. The ORA-semantics is defined as the identifications of nodes in XML data and schema. In XML schema, an internal node can be classified as object class, explicit relationship type, composite attribute and grouping

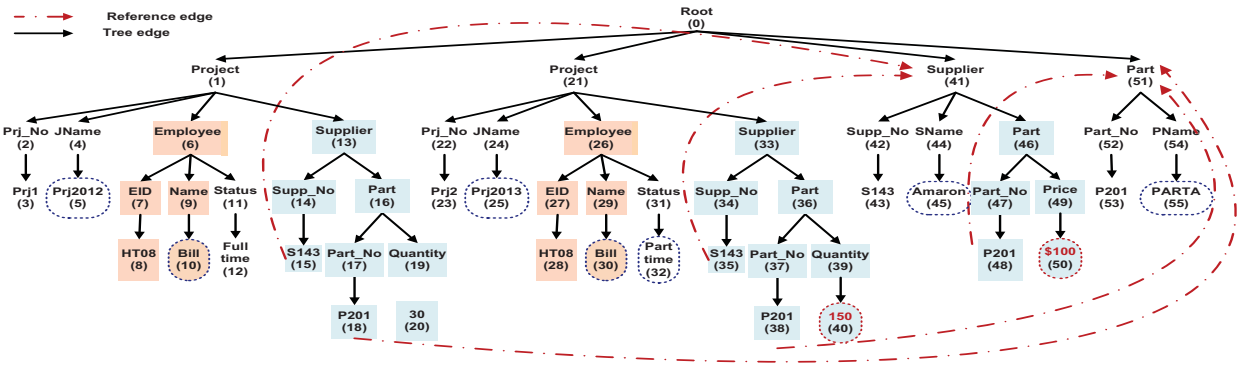


Figure 9: An XML document with both IDREFs and duplicated objects

node; and a leaf node can be classified as object identifier (OID), object attribute and relationship attribute. In XML data, a node can be an object node or a non-object node. More information about ORA-semantics and how to discover it is given in [26].

The ORA-semantics is hidden in XML and in the mind of database *designers* and *users*. For example, under ID/IDREF mechanism of XML, database *designers* must know object and object identifier (OID) to create reference edges. Otherwise, they cannot design an XML document with ID/IDREF. Based on ID/IDREF in XML, *users* also know object and OID.

Approaches for XML keyword search without using of the ORA-semantics return answers which may be useless, duplicated, incorrect, missing and schema-dependent answers as pointed out in Section 2 and Section 3. Recently based on the ORA-semantics, several approaches proposed to not only address the above problems but also to improve the usability of XML keyword search. These works can be briefly described as follows.

To solve the problems of the LCA-based approaches discussed in Section 2.8, based on the ORA-semantics, [22] introduces a novel search semantics, called Nearest Common Object Node (NCON), which includes not only common ancestors, but also common descendants of matching nodes to answer a keyword query. [22] also proposes an approach to find NCONs for a keyword query over XML tree. The approach uses the reversed data tree where the object paths from the root to each leaf nodes are reversed with those of the original data tree. Then, common descendants in the original data tree correspond to common ancestors in the reversed data tree. Therefore, the common ancestors from both the original and reversed data tree provide the set of NCONs for a keyword query.

Also based on the ORA-semantics, [23] models an XML document with IDREF as a so-called XML IDREF graph. [23] discovers that an XML IDREF graph still has hierarchical structure where a reference edge can be considered as a parent-child relationship, in which the parent is the referring node and the child is the referred node. This helps generalize efficient techniques of the LCA-based approaches for keyword search over XML IDREF graph. Thereby, it can achieve an efficient algorithm to find NCONs over XML IDREF graph.

Not only common ancestors and common descendants of the matching nodes provide meaningful answers to users, *common relatives* of the matching nodes, which are common ancestors in XML documents with some equivalent schemas, are also meaningful to users. This is because if a database is designed in the way that the mentioned common relative becomes a common ancestor of matching nodes in some equivalent schema, then that common relative is returned as an LCA node. Therefore, based on the ORA-semantics, [20] proposes the CR (Common Relative) semantics to include all together common ancestors, common descendants and common relatives as answers. This leads to another important advantage of the CR semantics is that it is independent from schema designs [20]. In contrast, existing approaches depend on schema designs because they may return different query answers for different hierarchical structures of the same data content. This advantage is important because when users issue a keyword query, they often have some intention in mind about what they want to search for regardless of the schema used. Hence, they expect the same answers from different designs of the same data content.

In [21] supports expressive keyword queries with

group-by and aggregate functions including *max, min, sum, avg, count* for XML keyword search. It faces with several challenges. The first challenge is how to handle ambiguity where a query has multiple interpretations in order not to mix the results of group-by and aggregate functions from different query interpretations together. The second challenge is how to handle object duplication and relationship duplication to calculate group-by and aggregate functions correctly. To overcome these challenges, the ORA-semantics is exploited again to identify interpretations of a query and to detect duplication.

5. CONCLUSION AND FUTURE WORK

XML keyword search has gained a lot of interests with many works done. This paper provides a survey for XML keyword search. We classified existing works into three types: tree-based approaches, graph-based approaches and semantics-based approaches. For each type of approaches, we summarized the main features, showed the relationships among works and especially pointed out the common problems that each type of approaches suffer.

From these problems, more broadly, this paper demonstrates the benefit of object orientation in XML. Without even requiring full-blown object orientation, merely by recognizing the concept of objects, object identifiers, and relationships among objects, researchers are able to add substantial semantics to XML represented data and showed how this small amount of additional annotation can greatly benefit keyword search. Therefore, in the future, exploring how other XML processing can similarly benefit is a promising topic.

6. REFERENCES

[1] Z. Bao, T. W. Ling, B. Chen, and J. Lu. Efficient XML keyword search with relevance oriented ranking. In *ICDE*, 2009.

[2] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using BANKS. In *ICDE*, 2002.

[3] J. Camacho-Rodriguez, D. Colazzo, and I. Manolescu. Building large XML stores in the amazon cloud. In *ICDEW*, 2012.

[4] L. J. Chen and Y. Papakonstantinou. Supporting top-k keyword search in XML databases. In *ICDE*, 2010.

[5] Y. Chen, W. Wang, Z. Liu, and X. Lin. Keyword search on structured and semi-structured data. In *SIGMOD*, 2009.

[6] S. Chong, C.-Y. Chan, and G. A. K. Multiway SLCA-based keyword search in XML data. In *WWW*, 2007.

[7] S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv. XSEarch: A semantic search engine for XML. In *VLDB*, 2003.

[8] B. Ding, J. X. Yu, S. Wang, L. Qin, X. Zhang, and X. Lin. Finding top-k min-cost connected trees in database. In *ICDE*, 2007.

[9] K. Golenberg, B. Kimelfeld, and Y. Sagiv. Keyword proximity search in complex data graphs. In *SIGMOD*, 2008.

[10] K. Golenberg, B. Kimelfeld, and Y. Sagiv. Keyword proximity search in complex data graphs. In *SIGMOD*, 2008.

[11] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. XRANK: Ranked keyword search over XML documents. In *SIGMOD*, 2003.

[12] H. He, H. Wang, J. Yang, and P. S. Yu. BLINKS: ranked keyword searches on graphs. In *SIGMOD*, 2007.

[13] V. Hristidis, N. Koudas, Y. Papakonstantinou, and D. Srivastava. Keyword proximity search in XML trees. *TKDE*, 2006.

[14] V. Hristidis, Y. Papakonstantinou, and A. Balmin. Keyword proximity search on XML graphs. In *ICDE*, 2003.

[15] H. V. Jagadish and S. AL-Khalifa. Timber: A native XML database. Technical report, University of Michigan, 2002.

[16] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, and R. D. Hrishikesh Karambelkar. Bidirectional expansion for keyword search on graph databases. In *VLDB*, 2005.

[17] M. Kargar and A. An. Keyword search in graphs: finding r-cliques. *PVLDB*, 2011.

[18] B. Kimelfeld and Y. Sagiv. Finding and approximating top-k answers in keyword proximity search. In *In PODS*, 2006.

[19] L. Kong, R. Gilleron, and A. L. Mostrare. Retrieving meaningful relaxed tightest fragments for xml keyword search. In *EDBT*, 2009.

[20] T. N. Le, Z. Bao, and T. W. Ling. Schema-independent XML keyword search. *ER*, 2014.

[21] T. N. Le, Z. Bao, T. W. Ling, and G. Dobbie. Group-by and aggregate functions in XML keyword search. In *DEXA*, 2014.

[22] T. N. Le, T. W. Ling, H. V. Jagadish, and J. Lu. Object semantics for XML keyword search. In *DASFAA*, 2014.

[23] T. N. Le, Z. Zeng, and T. W. Ling. Finding missing answers due to object duplication in XML keyword search. In *DEXA*, 2014.

[24] G. Li, J. Feng, J. Wang, and L. Zhou. Effective keyword search for valuable LCAs over XML documents. In *CIKM*, 2007.

[25] G. Li, B. C. Ooi, J. Feng, J. Wang, and L. Zhou. EASE: Efficient and adaptive keyword search on unstructured, semi-structured and structured data. In *SIGMOD*, 2008.

[26] L. Li, T. N. Le, H. Wu, T. W. Ling, and S. Bressan. Discovering semantics from data-centric XML. In *DEXA*, 2013.

[27] Y. Li, C. Yu, and H. V. Jagadish. Schema-free XQuery. In *VLDB*, 2004.

[28] X. Liu, C. Wan, and L. Chen. Returning clustered results for keyword search on XML documents. *TKDE*, 2011.

[29] Z. Liu and Y. Chen. Identifying meaningful return information for XML keyword search. In *SIGMOD*, 2007.

[30] Z. Liu and Y. Chen. Reasoning and identifying relevant matches for XML keyword search. In *PVLDB*, 2008.

[31] Z. Liu and Y. Chen. Processing keyword search on xml: A survey. *World Wide Web*, 2011.

[32] K. Nguyen and J. Cao. Exploit keyword query semantics and structure of data for effective xml keyword search. In *ADC*, 2010.

[33] L. Qin, J. X. Yu, L. Chang, and Y. Tao. Querying communities in relational databases. In *ICDE*, 2009.

[34] Z. Tian, J. Lu, and D. Li. A survey on XML keyword search. In *APWeb*, 2011.

[35] H. Wang and C. C. Aggarwal. A survey of algorithms for keyword search on graph data. In *Managing and Mining Graph Data*. 2010.

[36] Y. Xu and Y. Papakonstantinou. Efficient keyword search for smallest LCAs in XML databases. In *SIGMOD*, 2005.

[37] Y. Xu and Y. Papakonstantinou. Efficient LCA based keyword search in XML data. In *EDBT*, 2008.

[38] J. X. Yu, L. Qin, and L. Chang. *Keyword Search in Databases*. 2010.

[39] Z. Zeng, Z. Bao, M.-L. Lee, and T. W. Ling. A semantic approach to keyword search over relational databases. In *ER*, 2013.

[40] J. Zhou, Z. Bao, W. Wang, T. W. Ling, Z. Chen, X. Lin, and J. Guo. Fast SLCA and ELCA computation for XML keyword queries based on set intersection. In *ICDE*, 2012.

[41] R. Zhou, C. Liu, and J. Li. Fast ELCA computation for keyword queries on XML data. In *EDBT*, 2010.