Design Independent Query Interfaces

Arash Termehchy, Marianne Winslett, Yodsawalai Chodpathumwan, and Austin Gibbons

Abstract—Real-world databases often have extremely complex schemas. With thousands of entity types and relationships, each with a hundred or so attributes, it is extremely difficult for new users to explore the data and formulate queries. Schema free query interfaces (*SFQI*s) address this problem by allowing users with no knowledge of the schema to submit queries. We postulate that SFQIs should deliver the same answers when given alternative designs for the same underlying data set. In this paper, we introduce and formally define *design independence*, which captures this property for SFQIs. We establish a theoretical framework to measure the amount of design independence provided by an SFQI. We show that most current SFQIs provide a very limited degree of design independence. We also show that SFQIs based on the statistical properties of data can provide design independence when the changes in the schema do not introduce or remove redundancy in the data. We propose a novel XML SFQI called *Duplication Aware Coherency Ranking* (DA-CR) based on information-theoretic relationships among the data items in the database, and prove that DA-CR is design independent. Our extensive empirical study using three real-world data sets shows that the average case design independence of current SFQIs is considerably lower than that of DA-CR. We also show that the ranking quality of DA-CR is better than or equal to that of current SFQI methods.

Index Terms—Query Interface, Design Independence.

1 Introduction

Lack of understanding of the schema can be a significant obstacle for users who want to access the information in a database (DB). Enterprise DBs typically have complex schemas with hundreds of tables, columns, or XML elements. With little documentation available for the DB design, and even less motivation to study it, even computer-savvy users find it challenging to explore the DB and formulate appropriate queries. Many end-users are not even familiar with the concept of a schema [1], [2], [3], [4], [5], [6]. Further, as the schema evolves over time, even an expert user may not be able to pose the correct query for her needs.

Schema free query interfaces [2], [3] and keyword query interfaces [1], [4], [5], [6], [7] (both called SFQIs in this paper) have been proposed as solutions to these problems for XML DBs. With SFQIs, users do not need to know schema details or a query language. For example, suppose a user wants to find the papers that *John Lee* published about *XML* in the DB fragment in Fig. 1(a). The user submits query *John Lee XML*. The SFQI returns the answer, which is the paper at node 6.

DB administrators (DBAs) may revise the DB design over time to address issues such as redundancy, space overhead, performance, and usability. For instance, path *paper/booktitle* is repeated under every subtree of element *proceedings* in the DB excerpted on the right in Fig. 2. The DBA may normalize the schema as shown in the fragment on the left in Fig. 2 [8]. Or, as each paper has more than one author, the DBA may add a new node *authors* to the DB excerpted in Fig. 1(a) and create a new DB excerpted in Fig. 1(b) to make the DB more usable. A DBA may also remove nodes such as *authors* to save space.

Current DBMSs provide physical independence for users. Users do not have to modify their XQuery queries if the physical representation of the data changes. Similarly, an SFQI should have the property that its answers to a query stay the same when the DB schema changes. In other words, users should not have to change their queries to get the same answers over the new schema. For instance, an SFQI should return the *paper* at node 6 in Fig. 1(a) and the *inproceedings* at node 5 in Fig. 1(b) for query *John Lee XML*, as they both represent the entity in question. Otherwise, users would have to learn aspects of the new schema to formulate the new query, which contradicts the goal of having an SFQI. We call this property **design independence**. Design independence provides a metric to measure the degree of *logical data independence* sought by the architects of modern data models [9]. To the best of our knowledge, design independence has not previously been defined and explored for SFQIs.

Our contributions in this paper are as follows:

- We explore and formally define the similarity between answers to the same query across different DBs, from a user's perspective.
- We introduce and formally define the design independence property and explore its benefits for SFQI users. We introduce *VS preserving* DB transformations, which preserve schema information and the DB content. We prove that if an SFQI is built properly, it will be design independent under VS preserving transformations.
- We analyze the design independence property for current XML SFQIs and argue all except one, Coherency

- *Authors are with Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, 61801. E-mail: {termehch,winslett,ychodpa2,gibbons4}@illinois.edu*
- *This work is supported by NSF grant 0938071, 0716532, and 0938064.*

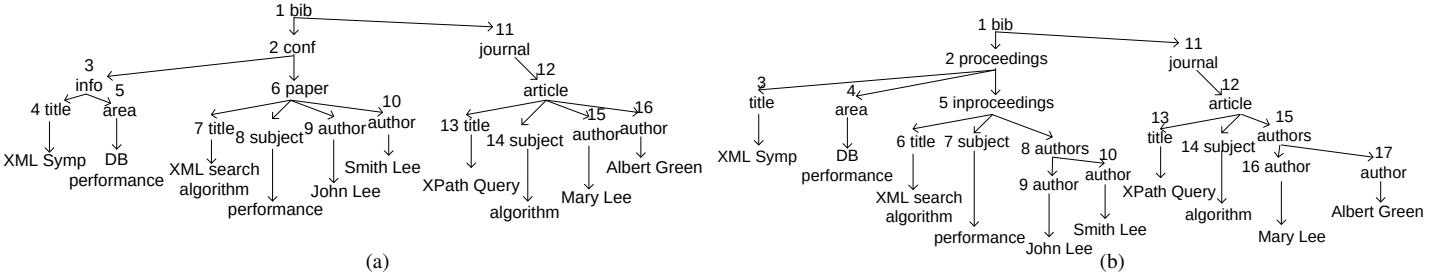

Fig. 1: Bibliographic databases

Ranking (CR for short) [7], are not design independent.

- Design independence requires the original and redesigned DBs to have the same content, so it does not cover DB redesigns that introduce or eliminate data redundancy. However, DB transformations such as normalization affect redundancy. We introduce a new property called *weak design independence* for the case where the new DB has more or less redundant data than the old DB.
- No current SFQI is weakly design independent. We develop a new SFQI called **Duplication Aware Coherency Ranking** (DA-CR) that is weakly design independent. DA-CR identifies desirable query answers by exploiting the information-theoretic relationships between the elements in the DB, combined with novel content scoring formula.
- We provide an extensive empirical study to evaluate the average case design independence of current and new SFQIs over three real-world data sets. We show that CR and DA-CR have considerably better average design independence than other SFQIs when the new DB design does not introduce or remove data redundancy. We also show that DA-CR has higher average design independence than other methods if the new designs create or eliminate data redundancy. Through an extensive user study, we show that DA-CR delivers the same or better ranking than other methods. Thus, its design independence does not reduce its effectiveness.

In the reminder of the paper, Section 2 presents basic definitions and related work. Section 3 defines design independence and explores its properties. Section 4 analyzes this property for current SFQIs. Section 5 defines and analyzes weak design independence and introduces DA-CR. Section 6 discusses experimental results for the average case design independence and weak design independence for all discussed methods, and analyzes the effectiveness of DA-CR using real world DBs. Section 7 concludes the paper.

2 Background

2.1 Basic Definitions

We model an XML DB as a tree $D = (r, V, E, L, C, A)$, where V is the set of nodes in the tree, $r \in V$ is the root, E is the set of parent-child edges between members of V, $C \subset V$ is a subset of the leaf nodes of the tree called *content nodes*, L assigns a label to each member of $V - C$, and A assigns a data value (e.g., a string) to each content node. We call the parent of a content node its *attribute*. We assume no node has both leaf and non-leaf children, and each node has at most one leaf child; other settings can easily be transformed to this one. We assume no order between the sibling nodes in the DB tree. Each *subtree* $S = (r_S, V_S, E_S, L_S, C_S, A_S)$ of T is a tree such that $V_S \subseteq V$, $E_S \subseteq E$, $L_S \subseteq L$, $C_S \subseteq C$, and $A_S \subseteq A$. Consider the depth-first traversal of a tree, where we visit the children of a node in the alphabetic order of their labels. Each time we visit a node, we output its label (or content); each time we move up one level in the tree, we output *-1*. The result is the unique *prefix string* for that tree [10]. For instance, the prefix string for the subtree rooted at node 3 in Fig. 1(a) is *info area "DB performance" -1 -1 title "XML Symp" -1 -1*. We refer to an XML DB simply as a DB.

Trees T_1 and T_2 are isomorphic iff there is a bijection f between sets of nodes in T_1 and T_2 that maps every node $n \in T_1$ with label l to node $f(n) \in T_2$ whose label is l and every edge $e : (n, m) \in T_1$ to one and only one edge $f(e) \in T_2$ such that we have $f(e) : (f(n), f(m))$. A **pattern** concisely represents a maximal set of isomorphic trees (its **instances**). The pattern can be obtained from the prefix string of any member of the set of instances, by removing the content. For example, pattern *paper author -1 title -1* has two instances in Fig. 1(a), both rooted at node 6. The **size** of a pattern is its number of leaves. A pattern is a **path** if it has only one leaf. Patterns p_1 and p_2 are isomorphic iff at least one instance of p_1 and one instance of p_2 are isomorphic. A **root-pattern** is a pattern whose root is the root of the DB. A **root-path** is a path that is a root-pattern. Except where otherwise noted, we consider only root-patterns in this paper. Pattern P_1 is a **subpattern** of pattern P_2 if each of P_1's instances is a subtree of one of P_2's instances. The **value** of a subtree (if it exists) is the list of content associated with its leaves. For example, in Fig. 1(a), the value of the subtree rooted at node 3 with pattern *info area -1 title -1* is ("XML Symp","DB performance"). Similarly, the value of a pattern instance is the list of content associated with the children of its leaves. The **values** of a pattern are given by the multiset containing the values of all of its instances.

We model a schema free query (*query* for short) as a bag of terms $Q = t_1, \ldots, t_q$, where each term t_i, $1 \leq i \leq q$, is the label of an attribute (label term) or a keyword (keyword term). The exact definition of a query

differs slightly for different SFQIs. In some SFQIs, users can specify the selection attributes, i.e., the attributes that contain keywords; and the projection attributes, i.e., the attributes whose values users want to see in the output [2], [3]. In others, such as keyword query interfaces [1], [2], [3], [4], [5], [6], the query interface has to figure out these pieces of information and return the values of selection and projection attributes. Our framework is orthogonal to the format of the query as long as it does not contain any information about the relationship between attributes in the schema.

A subtree S is a candidate answer (*CA*) to Q iff each of its content nodes either contains at least one instance of each keyword term in Q or the attribute of the content node contains one label term. Current keyword query interfaces over database systems consider only CAs that contain all terms of the input query [1], [2], [3], [4], [5], [6]. Hence, we consider only these CAs in this paper. Our approach could be extended for the cases where a CA contains at least one term of the input query. The root of a CA is the *lowest common ancestor* (LCA) of its content nodes. For example, the subtree with LCA 3 in Fig. 1 (a) is a CA for query Q_1: *title performance*. We consider the path from the LCA of a CA to the root of the DB as part of the CA, as it simplifies our analysis. All the CAs of a query form a multiset.

The baseline SFQI returns all CAs to a query. However, as schema free queries (SFQs) cannot be neatly mapped to XQuery or XPath, many SFQ CAs are unhelpful. For instance, consider query Q_2: *algorithm Lee* on the DB fragment shown in Fig. 1(a). CA a_1: *1 2 6 7 "XML search algorithm" -1 -1 -1 -1 11 12 15 "Mary Lee" -1 -1 -1 -1* and CA a_2: *1 2 6 9 "John Lee" -1 -1 -1 -1 11 12 14 "algorithm" -1 -1 -1 -1* are unhelpful answers for Q_2. The only relationship their attributes have is that they occur in the same DB, which is not very interesting. Researchers have proposed SFQIs that filter out CAs they deem unhelpful [3], [4], [5], [11], or return a *ranked list of CAs* (list for short) The *rank* of a CA in a list is its position in the list. For instance, one approach filters out every CA whose root is an ancestor of the root of another CA [4], [5]; the LCAs of the remaining CAs are called the *smallest* LCAs (SLCAs). The SLCA approach relies on the intuitively appealing heuristic that far-apart nodes are not as tightly related as nodes that are closer together. SLCA removes a_1 and a_2, as their roots (node 1) are ancestors of node 6, which is the root of another CA, a_3: *6 7 "XML search algorithm" -1 -1 9 "John Lee' -1 -1*. If an SFQI only filters out irrelevant CAs, it outputs a *multiset of CAs* (multiset for short) for a given query.

One can develop a domain-dependent SFQI and then leverage additional domain- and DB-specific knowledge to help identify desirable answers. For example, at the Internet Movie Database (IMDB) (*www.imdb.com*), the majority of users may be more interested in new releases than older movies. In this paper we focus on domain-independent techniques, to minimize the need for manual addition of domain-specific heuristics [1], [2], [3], [4], [5], [6], [7].

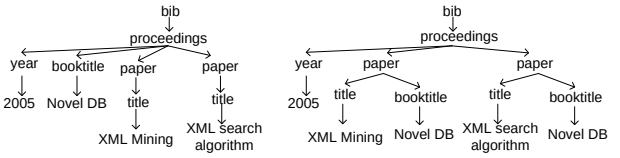

Fig. 2: Normalization of a bibliographic DB

2.2 Related Work

SFQIs trade precision and recall for logical data independence. To the best of our knowledge, the issue of design independence has not been explored for SFQIs. Information preservation and query preservation properties of XML and relational schema mappings have been studied in the context of data exchange and data integration [12], [13]. The goal there is to identify schema mappings where every SQL or XQuery query over the source schema can be *manually translated to a new query* over the target schema. Our goal is to identify mappings where SFQIs can return the same answers *without changing the query*, and we propose a new SFQI that does this. Researchers have analyzed the modifications of text-centric XML files where the content of the output file is a subsequence of the content of the input file [14]. However, these modifications may not preserve content or structure of the XML file. Researchers have tried to provide logical data independence by representing the DB as an instance of one single "universal" relation (UR) [15], so that each DB query maps to a query over the UR. UR users do not need to know how to join relations to build the UR, but they must know a query language, attributes' names, and which information is stored in which attribute. Thus a UR is harder to use than an SFQI. We do not assume a common representation for all DBs; instead, we develop an SFQI that returns the same answers over different designs of the same DB content. This approach provides more logical data independence. For instance, if the DB designer changes attributes' labels, our SFQI returns the same query answers over the old and new DB. UR users would have to rewrite their queries for the new DB.

In previous work [16], we defined the property of design independence, analytically determined the worst case design independence for current SFQIs, and introduced SFQI DA-CR. In this paper, we present a much more extensive study of average case design independence for SFQIs, over larger data sets and workloads than our previous work. We also explore the information preservation characteristics of weak value structure preserving transformations in this paper, and explore transformations that slightly change DB values. Further, we fully justify DA-CR's ranking approach, and compare SFQIs to the UR approach.

3 Design Independence

A **transformation** T over DB D is a function that modifies D to generate a new DB $T(D)$ [12], [13]. We assume that any SFQI operating over the transformed DB is unaware of the original DB or the transformation.

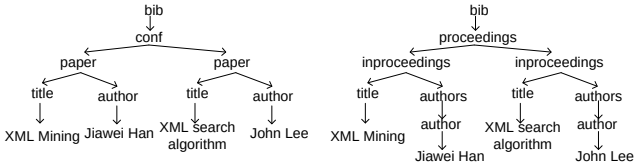

Fig. 3: Answers to Q_4 over original and transformed DBs

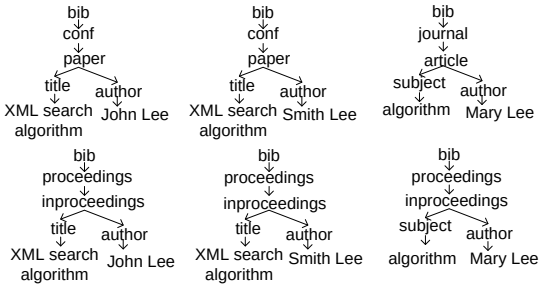

Fig. 4: Q_2 answers over original and transformed DBs

If two CAs are isomorphic and their corresponding leaf nodes contain the same content, they are *label-content isomorphic*. Label-content isomorphic answers represent the same information. Hence, we intuitively consider an SFQI to be design independent if it returns the same list (multiset) of (label-content isomorphic) answers for every query over the original and modified DBs. However, when a DB is modified, an SFQI may not be able to return exactly the same list of CAs for a query over the old and new DBs. For instance, Fig. 1(b) is the result of a transformation of the fragment shown in Fig. 1(a). Consider Q_3: *Query Green* over Fig. 1(a), where its CA is the subtree whose LCA is node 12 and contains nodes 13 and 16, and Fig. 1(b), where its CA is the subtree whose LCA is node 12 and contains nodes 13 and 17. Since the CA for Q_3 over Fig. 1(b) contains a new node *authors*, the CAs of Q_3 over Fig. 1(a) and Fig. 1(b) are not isomorphic. However, they both represent the authorship relation between *Albert Green* and *XPath Query*. Thus, we argue that users gain the same information from two CAs that express *equivalent relationships* between data items that have *the same content*. Therefore, we also consider an SFQI to be design independent if it returns essentially the same information for every query over the original and transformed DBs. We will precisely define what it means for answers over the original and transformed DBs to convey the same information to users.

3.1 Preserving Content

Answers that convey the same information must contain the same content. For instance, if the content of node 13 in Figs. 1(a) and (b) were different, users would consider the CAs to Q_3 over these data fragments to be different. Consider a transformation over the data fragment in Fig. 1(a) that removes *author* and *title* nodes that are children of the same *paper* node and creates a single new child node for each *paper* node, called *paperInfo*, that contains the merged content of the eliminated *author* and *title* nodes. The CAs for Q_3 over the original and transformed DBs have similar content. Nevertheless, the CA of the original DB represents the author and the title of the paper in separate nodes and the CA of the transformed DB shows the author and the title of the paper in one content node. By looking at the CA of the transformed DB, some users may not be able to distinguish the tokens that represent the title from the tokens that represent the author.

Definition 1: A bijective mapping M from pattern instance i_1 to pattern instance i_2 is **value preserving** iff it maps each member of the value of i_1 to an equal member of the value of i_2.
We consider two value members equal if they are lexicographically equal: they have the same length and contain the same characters in the same positions. We define value preserving mapping between candidate answers similarly.

Definition 2: Pattern instances i_1 and i_2 are **value equivalent** iff there is a value preserving mapping from i_1 to i_2.

We can similarly define value equivalent candidate answers. Since users observe all CAs that an SFQI returns for a query, we must define value equality between the lists or multisets for the same query over different DBs.

Definition 3: A bijective mapping M from list l_1 to list l_2 is **value preserving** iff it maps each CA $s \in l_1$ to a value equivalent CA $M(s) \in l_2$, where the rank of s in l_1 is equal to the rank of $M(s)$ in l_2.

Definition 4: Lists l_1 and l_2 are **value equivalent** iff there is a value preserving mapping from l_1 to l_2.
Definition 4 requires the value equivalent lists to have the same number of CAs. Similarly, a bijective mapping M from multiset u_1 to multiset u_2 is **value preserving** iff it maps each CA $s \in u_1$ to a value equivalent CA $M(s) \in u_2$. Hence, two multisets are value equivalent if we can define a value preserving mapping between them. There may be more than one value preserving mapping for value equivalent multisets.

If a transformation manipulates the value of a content node, no SFQI will be able to return value equivalent lists (multisets) for queries whose CAs contain that content node. For instance, if a transformation updates the value of node 9 in Fig. 1(a) to "Kate Lee", there is no value equivalent mapping between the CAs of Q_2 over the original and transformed DBs. Since SFQIs do not know about transformations, they cannot return value equivalent lists (multisets) to Q_2. Thus, we must find the properties of transformations that allow SFQIs to be design independent.

Definition 5: A bijective mapping M from pattern p_1 to pattern p_2 is **value preserving** iff it maps each instance of p_1 to a value equivalent instance of p_2.
Patterns p_1 and p_2 are **value equivalent** iff there is a value preserving mapping from p_1 to p_2.

Definition 6: A transformation T over DB D is **value preserving** iff it maps each pattern p in D to exactly one value equivalent pattern $T(p)$ in DB $T(D)$, where each pattern $T(p)$ is mapped to by only one pattern p in D.
Given a query, an SFQI filters or ranks the CAs for the query. Hence, a transformation must map all possible CAs

```
<!ELEMENT bib (conf*, journal*)>
<!ELEMENT conf (info, paper*)>
<!ELEMENT info (title, area)>
<!ELEMENT paper (title, subject, author*)>
<!ELEMENT journal (article*)>
<!ELEMENT article (title, subject, author*)>
```

Fig. 5: Schema of the bibliographic DB in Fig. 1(a)

```
<!ELEMENT bib (proceedings*, journal*)>
<!ELEMENT proceedings (title, area, inproceedings*)>
<!ELEMENT inproceedings (title, subject, authors)>
<!ELEMENT authors (author*)>
<!ELEMENT journal (article*)>
<!ELEMENT article (title, subject, authors)>
```

Fig. 6: Schema of the bibliographic DB in Fig. 1(b)

for every query over original and transformed DBs value equivalent to allow an SFQI to be design independent.

Proposition 1: Given a value preserving transformation T over DB D, the multiset of CAs to every query q over D and $T(D)$ are value equivalent.

As noted, if a transformation introduces new terms to a DB or eliminates some terms from the DB, it causes every query interface to return different answers over the original and transformed DBs for all queries that contain the added or removed terms. One may argue that it may be possible to develop an SFQI that returns similar results under a transformation that modifies only the number of instances of some terms in the content nodes of a DB. Researchers have considered a similar problem in the context of retrieval over noisy document collections [17]. They have shown that even slight modification in the number of instances of query terms inside the documents of a collection results in considerably different rankings for some queries for using any reasonably effective retrieval method. We have to consider the properties of the content nodes in the retrieval formula in order to develop a reasonably effective SFQI. Since each content node is a small document, it will not be possible to have such SFQI that returns similar results under this type of transformations.

3.2 Preserving Structure

Non-content nodes represent structural relationships between content nodes. If a transformation renames or removes schema elements or introduces new schema elements, it may change the structural relationship between content nodes. Hence, SFQIs cannot always deliver answers with exactly the same structure over the original and new DB. Assume users can mentally translate the structure of the old answers to the structure of the new answers. Then, the SFQI returns structurally similar answers over both DBs. Since an SFQI uses schema information to rank and/or filter CAs, we must only consider transformations that do not lose any information of the schema of the original DB. (We do not consider complex consistency or integrity constraints as part of the schema information.)

Let S be the schema of XML DB D (e.g., its DTD) and let $I(S)$ be the set of all DBs with schema S. Given schemas S_1 and S_2, if we can find an invertible function T: $I(S_1) \rightarrow I(S_2)$, then S_2 carries at least as much information as S_1 [12], [13]. In other words, we can reconstruct the information available in any DB over the original schema S_1, given the transformed DB. This definition is suitable for data exchange and schema mapping, where the target schema may contain more information than the original schema. Therefore, it allows T to be a partial function. Since we would like to have similar answers over the original and transformed DBs, we need T to cover all DBs of a given XML schema and not add any new information to the original schema. Given XML schemas S_1 and S_2, transformation $T : I(S_1) \rightarrow I(S_2)$ is **bijective** if it is a bijective mapping.

Figs. 5 and 6 show the schema of the DBs excerpted in Figs. 1(a) and (b), respectively. There is a bijective transformation between these schemas, with each DB under the schema of Fig. 5 mapped to one and only one DB under the schema of Fig. 6. Assume a transformation U over the schema of Fig. 5 that removes *journal*, *article*, *info*, *conf*, and *paper* and connects all attribute nodes to the *bib* node. Since we defined an XML DB as an unordered tree, U maps more than one DB of the schema shown in Fig. 5 to one DB of the new schema. Thus, U is not bijective and loses schema information.

We must also consider the limitations of domain-independent SFQIs. If two DB nodes have the same label, domain-independent query interfaces, and SFQIs in particular, consider them to be instances of the same entity type. Consider a version of the data fragment in Fig. 1(a) that contains additional *article*s written by different authors. Assume a transformation removes node 15 from this DB and maps node 12 to *Mary-Lee-article*. Users may recognize that the resulting CA for Q_3 carries the same information as the CA over the original DB. Nevertheless, we do not expect SFQIs to draw such a conclusion, because values and labels play distinct roles in query interfaces, even in SFQIs [1], [2], [6], [7], [11]. Thus as far as they are concerned, the new schema introduces a new entity type that is different from *article* entities and must be treated differently. Moreover, domain independent SFQIs treat values as uninterpreted bags of words (or objects) [1], [2], [3], [4], [5], [6], [7], [11]. Therefore, they cannot comprehend that the information added to the label carries the removed content information.

As another example, consider a transformation that groups all *article*s about the same *subject* under a new node labeled *article-subject* that is a child of *journal* in the original DB whose fragments are shown in Fig. 1(a). An SFQI may reasonably rank CAs containing articles under the same *article-subject* node higher than CAs containing articles under different *article-subject* nodes. This is because the first group of CAs represents a more interesting relationship between articles than the second group. However, since SFQIs consider the values as uninterpreted objects and different from labels, they return the same ranking for CAs in both groups over the original DB.

Users expect CAs with equal content nodes to also have similar structural information. Fig. 3 shows a CA for Q_4:

search Han mining John from a DB whose schema is in Fig. 5. Fig. 3 also shows the value equivalent answer from a DB whose schema is in Fig. 6. *John Lee* and *Jiawei Han* have the same path in the CA on the left. This indicates that they have similar structural roles in the original and transformed DBs. Thus, users expect their equivalent content nodes in the other answer to have similar structural roles. The path of a content node encodes its structural role in a DB. Hence, if two content nodes have the same path in every CA a, their mapped content nodes in the value equivalent CA of a must have the same paths. In addition to the same structural role for each content node, users should see similar structural relationships between mapped content nodes in CAs. For instance, the authorship of a paper whose title is *XML Mining* by *Jiawei Han* is represented using *bib conf paper title -1 author -1 -1 -1* in the left CA in Fig. 3. This pattern also represents the relationship between *XML search algorithm* and *John Lee* in the same CA. Since the patterns of content nodes mapped to {*XML Mining*, *Jiawei Han*} and {*XML search algorithm*, *John Lee*} in the right CA in Fig. 3 are isomorphic, the CAs convey similar structural information about the relationships of these content nodes.

Definition 7: A bijective mapping M from CA s_1 to CA s_2 is **value structure preserving** (VS preserving for brevity) iff for all subpattern instances i_1 and i_2 of s_1:

- i_1 and $M(i_1)$ are value equivalent.
- The patterns of i_1 and i_2 are isomorphic iff the patterns of $M(i_1)$ and $M(i_2)$ are isomorphic.

Definition 8: CAs s_1 and s_2 are **value structurally equivalent** (VS equivalent) iff there is a VS preserving mapping from s_1 to s_2.

According to the first constraint in Definition 7, a VS preserving mapping is value preserving. Hence, VS equivalent CAs are also value equivalent.

Similar to the case for value equivalence, users consider all CAs in the list (multiset) of CAs to a query when judging the similarity of two lists (multisets). Fig. 4 shows some CAs for Q_2: *algorithm Lee* over Fig. 1(a) in the first row. It also shows some CAs over a transformation of Fig. 1(a) in the second row, each of which is value equivalent to the one above it in the first row. The first two CAs in the first row have the same pattern and represent the same structural relationship between *XML search algorithm* and *John Lee* and *XML search algorithm* and *Smith Lee*, respectively. Thus, users will expect the first two CAs in the second row to have the same pattern, which they do. Since *Mary Lee* is the author of an article and *Smith Lee* is the author of a paper, they have different paths in the list of CAs for Q_2 over Fig. 1(a). Hence, users expect them to have different paths in the list of CAs for Q_2 over the transformed DB, but they do not. Thus, we consider the CA lists in the first and second row of Fig. 4 to be structurally dissimilar. We define the subpattern instances of a list (multiset) of CAs as subpattern instances of its CAs.

Definition 9: A bijective mapping M from list l_1 to list l_2 is **VS preserving** iff for all subpattern instances i_1 and i_2 of l_1:

- i_1 and $M(i_1)$ are value equivalent.
- The CA that contains i_1 and the CA that contains $M(i_1)$ have equal ranks in l_1 and l_2, respectively.
- The patterns of i_1 and i_2 are isomorphic iff the patterns of $M(i_1)$ and $M(i_2)$ are isomorphic.

Definition 10: A list of CAs l_1 is **VS equivalent** to a list of CAs l_2 iff there is a VS preserving mapping from l_1 to l_2.

Based on the first two conditions in Definition 9, a VS preserving mapping between lists of CAs is also value preserving. Thus, two VS equivalent lists of CAs are also value equivalent. VS equivalence is defined similarly for two multisets of CAs. There may be more than one VS equivalent mapping for two multisets of CAs. VS equivalent lists (multisets) of answers for a query *preserve* the contents of structural relationships between the content nodes; therefore, users get essentially the same information on the relationships between content nodes in the original and new DBs.

If the CAs for every query q over a DB are VS equivalent to the CAs for q over the transformed DB, an SFQI can provide VS equivalent answers for q over the old and new DBs. Thus, we define the required conditions for such transformations.

Definition 11: A bijective mapping M from pattern p_1 to pattern p_2 is **VS preserving** iff for all subpattern instances s_1 and s_2 of p_1:

- s_1 and $M(s_1)$ are value equivalent.
- The patterns of s_1 and s_2 are isomorphic iff the patterns of $M(s_1)$ and $M(s_2)$ are isomorphic.

Definition 12: Patterns p_1 and p_2 are **VS equivalent** iff there is a VS preserving mapping from p_1 to p_2.

Similar to value preserving transformations, we have:

Definition 13: A transformation T over DB D is **VS preserving** iff it maps each pattern p in D to exactly one VS equivalent pattern $T(p)$ in DB $T(D)$, and each pattern $T(p)$ is mapped to by only one pattern p in D.

For instance, the mapping from the fragment in Fig. 1(a) to the fragment in Fig. 1(b) is VS preserving.

Theorem 3.1: If T is VS preserving for D, then any query's CAs for D and $T(D)$ are VS equivalent.

Definition 14: An SFQI is **design independent** if for each DB D, query q, and VS transformation T for D, the answer to q from D and from $T(D)$ are VS equivalent.

An SFQI may use some information in the DB that may not be in the CAs to rank and/or filter the CAs. Thus, a VS preserving transformation must not lose any schema information of the original DB, so that SFQIs will be able to use this information. Patterns of a schema are shared between all instances of the schema. Hence, we can define VS transformations over all DBs of a given schema. Given XML schemas S_1 and S_2, we define a VS preserving transformation as a mapping from $I(S_1)$ to $I(S_2)$.

Theorem 3.2: Given XML schemas S_1 and S_2, every VS preserving transformation $T : I(S_1) \rightarrow I(S_2)$ is bijective.

We can, however, find a bijective transformation that is not VS preserving. For instance, assume the data fragment shown in Fig. 1(a) has more than one article in each journal and each article has only one *subject*. Assume that transformation T groups all the *article*s in a journal that have the same subject under a new parent node *article-subject* that is the child of *journal*. This transformation is bijective, as it provides a bijective mapping over the instances of the two schemas. Nevertheless, it does not provide a bijective mapping over the patterns of the original and transformed DBs.

Since a VS preserving transformation may change the labels of the leaf nodes, a query that contains label terms might have different numbers of CAs in the original and transformed DBs. If an SFQI allows its queries to contain label terms, we restrict VS preserving transformations so that they preserve the labels of the leaf nodes in the original DB.

4 Design Independence of SFQIs

The baseline technique for answering queries, called the LCA method, returns all CAs. Based on Theorem 3.1, the LCA method is design independent. However, as mentioned in Section 2, this approach returns all the non-relevant CAs. Hence, it has the lowest precision. Moreover, it does not rank the CAs. Thus other methods use the properties of CAs to improve its effectiveness. They filter out irrelevant CAs via *filtering methods* [1], [2], [3], [4], [11], and/or rank the answers, via *ranking methods* [2], [6], [7]. There are two categories of filtering techniques: distance based and label based. Distance based methods deem a CA to be irrelevant if its subtree is relatively large. ELCA [1] and MLCA [3] assume that only the closest nodes are meaningfully related. If candidate subtree t_1 shares a leaf node with another CA t_2 and the LCA of t_1 is an ancestor of the LCA of t_2, they filter out t_1. For instance, for query Q_5: *DB XML* over Fig. 1(a), we have two CAs: a_1^5 whose LCA is node 2 and contains nodes 4 and 7 and a_2^5 whose LCA is node 3 and contains nodes 4 and 5. ELCA and MLCA filter a_1^5 as it shares a leaf node with a_2^5 and its LCA is an ancestor of the LCA of a_2^5. Q_5 has also two CAs over Fig. 1(b): a_3^5, whose LCA is node 2 and contains nodes 4 and 6; and a_4^5, whose LCA is node 2 and contains nodes 3 and 4. Since the LCAs of these CAs are the same node, ELCA and MLCA return both CAs. Thus, ELCA and MLCA return different results for these two fragments.

Proposition 2: The ELCA and the MLCA methods are not design independent.

Generally, VS preserving transformations that change the structure of a DB by adding or removing non-leaf nodes change the results of ELCA and MLCA. These changes are quite frequent in XML DB design. For instance, one designer may decide to put *title* and *area* of a conference into a new node to increase readability of the DB and answers, while another designer may decide to remove these nodes to save space and increase performance.

As mentioned in Section 2, SLCA method [4], [5] filters out every candidate subtree whose LCA is an ancestor of the LCA of another candidate subtree. SLCA returns two CAs for query Q_6: *performance XML* over Fig. 1(a): the one whose LCA is node 3 and contains nodes 4 and 5 and the one whose LCA is node 6 and contains nodes 7 and 8. However, SLCA returns only one CA for Q_6 over Fig. 1(b), the one whose LCA is node 5 and contains nodes 6 and 7. The CA whose LCA is node 2 is filtered, as node 2 is an ancestor of node 5. Thus, SLCA-based methods return different results under VS preserving transformations. Similar to ELCA and MLCA, VS preserving transformations that add or remove non-leaf nodes change the results of SLCA.

Proposition 3: The SLCA method is not design independent.

Label based techniques such as XSearch [2] and CVLCA [11] remove every candidate subtree having two non-attribute nodes with the same label. The idea is that non-leaf nodes are instances of the same entity type if they have duplicate labels, and there is no interesting relationship between entities of the same type. For example, Q_7: *Lee XML* over the leftmost data fragment in Fig. 3 has two CAs: a_1^7, whose LCA is node *bib*; and a_2^7, whose LCA is node *paper*. These methods filter out a_1^7 because it contains duplicate labels (*paper*). They return three CAs for Q_8: *Green Lee* over Fig. 1(a): two whose LCAs are node 1 and one whose LCA is node 12. However, they return only one CA to the same query over Fig. 1(b). They filter out the CAs whose LCAs are node 1 as they contain duplicate label *authors*.

Proposition 4: The XSearch and the CVLCA methods are not design independent.

In general, label based methods are sensitive to the changes in the schema by the transformation that group similar nodes under a new node. For instance, in Fig. 1(b) a designer has decided to group all authors under a new grouping node *authors* to make the DB more usable. Their results are also sensitive to renaming schema elements.

Ranking techniques include using depth and the number of nodes in candidate subtrees, extensions of the PageRank technique, and statistical information about candidate patterns. XSearch ranks the candidate subtrees according to the number of nodes in the candidate subtree [2]. However, as we mentioned, VS preserving transformations can add or remove nodes from a DB. Thus, they can change the number of nodes for different CAs if they are not instances of the same pattern. This will change the rank of the CAs over original and transformed DBs. XReal uses the depths of the LCAs and attribute nodes of candidate subtrees to find and rank the CAs [6]. Since VS transformations can add or remove nodes from the DB, they can change the depth of LCA and attribute nodes. Again, if the CAs do not belong to the same pattern, VS transformations can change the depth of their LCA and attributes unequally, resulting in a different ranking for the original and transformed DBs.

XRank extends the PageRank formula to XML DBs, where nodes replace web pages and edges replace the links in the original PageRank technique. Thus, the importance of a node is proportional to the number of edges connected to it. The VS transformation can change the number of

edges connected to a node through removing its children or grouping its children under new nodes. For instance, the number of edges connected to node *conf* in Fig. 1(a) is different from node *proceedings* in Fig. 1(b). Changes to authoritative nodes such as *conf* alter the overall ranking of the nodes considerably [18].

CR [7] and NTPC [19] rank the CAs according to the correlations of their patterns. They use values of candidate patterns to compute correlations. The more correlated the values of a pattern are, the stronger their relationship is [7]. Consider the original DB excerpted in Fig. 1(a), where each conference has many papers. Knowing the title of a paper gives a considerable amount of information about the author of the paper in the DB. In other words, given a value of *title* in *bib conf paper title -1 author -1 -1 -1*, one can determine the value of *author* with high probability, as each paper does not have many authors. However, knowing *title* in *bib conf info title -1 -1 paper author -1 -1 -1* does not narrow down *author* very much, as each conference has many paper *author*s. Statistics-based approaches exploit this fact, and return a CA if its pattern's correlation exceeds a given threshold ϵ. They rank the answers in descending order of their patterns' correlations. CR and NTPC use an extended version of mutual information to compute patterns' correlations. Since we deal with data-centric XML, we focus on CR here, which is for this type of data; the design independent properties of NTPC can be shown similarly.

Before analyzing the design independence property of CR, we briefly review the concepts associated with entropy in XML. The probability of value a in pattern p is $P(a) = \frac{1}{n}count(a)$, where $count(a)$ is the number of instances of p with value a and n is the total number of instances of p in the DB. Intuitively, the entropy of a random variable indicates how predictable it is. The *entropy* of a pattern p having values $a_1, \ldots, a_n$ with probabilities $P(a_1), \ldots, P(a_n)$ respectively is $H(p) = \sum_{1 \leq j \leq n} P(a_j) \lg (1/P(a_j))$. Normalized total correlation (NTC) measures the correlation of a pattern; its value for a pattern t with root-paths $p_1, \ldots, p_n$, $n > 1$ is: $NTC(q) = 1 - \frac{H(q)}{\sum_{1 \leq i \leq n} H(p_i)}$. If the size of a pattern is one, CR sets the value of $NTC(q)$ to $H(q)$.

Proposition 5: Given VS transformation T that maps DB D to $T(D)$, the NTCs of each pattern $p \in D$ and $T(p) \in T(D)$ are equal.

CR, similar to XSearch [2] and XReal [6], considers each candidate subtree as a small document and uses variations of TF-IDF formulas to leverage the values of CAs in delivering the final ranking. Each term k_i in the input query has a document frequency (DF) that is the number of attribute nodes of the same label in the DB that contain k_i. The larger this number is, the less important the term becomes. If a candidate subtree contains more instances of important terms in a query, it has higher term frequency (TF) and gets higher rank. Since the content of CAs are the same over VS transformed DBs, TF remains the same under VS transformation.

Assume that a VS transformation does not change the label of the attributes. Since VS transformations do not change the number of instances of each path of a DB, they do not change the number of instances for attribute nodes with the same label. Thus, DF will be equal over the original and transformed DBs. If a VS transformation changes the name of attributes, DF may not be equal for the original and VS transformed DBs. In this case, we can measure DF over paths instead of attributes and make the TF-IDF formula design independent. CR computes a linear combination over NTC and TF-IDF scores to deliver the final ranking. Since both parts of CR are design independent, we have:

Corollary 1: The CR method is design independent.

We can prove that NTPC is design independent similarly.

5 Weak Design Independence

The definition of VS transformation delivers the same number and similar answers for the same query over the original and transformed schemas. Therefore, it does not include transformations that preserve schema information but are not able to deliver exactly the same number of similar answers to users. In particular, it requires both DBs to have the same number of equal values. Fig. 2 depicts a non-VS transformation over a bibliographic DB fragment that moves *booktitle* nodes from *proceedings* and duplicates them under all *paper*s of the same *proceedings*. The transformation denormalizes the data fragment and creates redundancy [8]. Since it maps each DB from the schema of the DB whose fragment is shown on the left to one DB from the schema of the DB whose fragment is shown on the right, it is bijective. However, Q_9: *Novel 2005* has one CA over the data fragment on the left and two CAs on the data fragment on the right. The CAs over the right data fragment are VS equivalent. Thus, we may consider them duplicates. If users have the ability to identify duplicate CAs, we can consider both CAs as a single CA. Thus, the results of Q_9 over both DBs will be VS equivalent. It is not unrealistic to expect users of a search system to recognize duplicate answers [20]. Consider Q_{10}: *Novel DB*. It has one CA over the left fragment a_1^{10} whose LCA is *booktitle*, but three CAs on the right data fragment: a_2^{10} and a_3^{10}, whose LCAs are *booktitle* nodes; and a_4^{10}, whose LCA is node *proceedings*. a_1^{10} and a_2^{10} are VS equivalent. a_4^{10} is not VS equivalent to any CAs for the right data fragment, but conveys the same information as the others about the query and we can consider it a duplicate.

Definition 15: Pattern instance S_1 is **almost equal** to pattern instance S_2 iff there is an onto relation R between every path p of S_1 and every path $p' \in R(p)$ of S_2 such that p and p' are isomorphic and have equal values.

For instance, CA a_2^{10} and a_4^{10} are almost equal. In this section, we consider the values of a pattern to be a set.

Definition 16: Pattern p_1 is a **duplicate** of pattern p_2 iff there is an onto relation R between every instance i of p_1 and every instance $i' \in R(i)$ of p_2, such that i and i' are almost equal.

For instance, pattern b_1: *bib proceedings paper title -1 -1 paper booktitle -1 -1 -1* is a duplicate of pattern b_2: *bib*

proceedings paper title -1 booktitle -1 -1 -1 in the right data fragment in Fig. 2. CAs are *duplicate*s of each other if they are almost equal and their patterns are duplicates of each other. For example, the CAs of Q_{11}: *Mining DB* over the right data fragment in Fig. 2 are duplicates of each other.

We define the LCA of a pattern similarly to the LCA of its instances. Since the duplicate relationship between patterns is symmetric and reflexive, patterns can be divided into equivalence classes based on this relationship. For each equivalence class, we choose the pattern with the fewest paths to be its representative element. If more than one pattern qualifies, we choose the first one in lexicographic order. The representative pattern does not have two distinct isomorphic paths.

Definition 17: A transformation T over DB D is **weak VS (WVS) preserving** if it maps each equivalence class of duplicate patterns g to exactly one equivalence class of duplicate patterns $T(g)$, such that the representative pattern of g is VS equivalent to the representative pattern of $T(g)$ and each equivalence class $T(g)$ is mapped to by only one equivalence class g.

The transformation shown in Fig. 2 is WVS preserving. Each VS preserving transformation is WVS preserving, but there are some WVS preserving transformations that are not VS preserving, such as the transformation shown in Fig. 2. We relax the condition on VS equivalence to define WVS equivalence for lists or multisets of CAs. We consider the output of an SFQI as a set, where all duplicate CAs in the same list or multiset are considered as equal. Also, we consider the position of each class of duplicate CAs in a list as the rank of the CA in the class with the lowest rank.

Proposition 6: Given a WVS preserving transformation T over DB D, the CAs for every query q over D and $T(D)$ are WVS equivalent.

As discussed in Section 3.2, SFQIs may use some schema information that is not in the candidate answers of input queries. Thus, we must make sure that WVS transformations do not add or lose any schema information. In other words, they must be bijective over the instances of original and transformed schemata as defined in Section 3.2.

Theorem 5.1: Given XML schemas S_1 and S_2, every WVS preserving transformation $T : I(S_1) \rightarrow I(S_2)$ is bijective.

Should an ideal SFQI return similar results under *all* bijective transformations that preserve DB content? For example, suppose we map *article*s written by *Mary Lee* in Fig. 1(a) to *Mary-Lee-article*. As noted in Section 3.2, SFQIs treat the labels of structural nodes and values of content nodes differently. It is not reasonable to expect SFQIs to return the same answers for this transformation, or for any others that use the *values* of a pattern instance in D to decide how to transform D's structure.

What about the remaining bijective transformations T that do not modify DB content and use only D as input? All remaining transformations use the labels and the ancestor-descendant relationships between non-leaf nodes of D in their predicates. Hence, they modify the instances of isomorphic sub-patterns similarly. According to Definition 17, these transformations are WVS preserving. We call a method **weakly design independent** if it returns a WVS equivalent list or a set of CAs over a WVS transformation.

5.1 Duplicate Aware CR

Since VS transformations are also WVS preserving, methods that are not design independent will not be weakly design independent. With a WVS transformation, VS equivalent patterns may have different numbers of almost equal instances. Thus, if we consider the values of a pattern to be a multiset, their NTC values can be different in the old and new DBs. Hence, CR does not provide design independence over WVS transformations. This means that patterns' correlations should be measured by statistical properties of their instances that will provide effective ranking *and* do not change under WVS transformations. Researchers have shown that the more distinct values an attribute has in the DB, the more important the information usually is that it stores [21]. As WVS transformations do not change the number of distinct values of a pattern in a DB, NTC simply needs to be changed to ignore duplicates.

Definition 18: Given pattern t with paths $p_1, \ldots, p_n$, $n \geq 1$, p's **normalized set total correlation** (NSTC) is the NTC of its set of pattern values.

As we will show in Section 6, patterns' NSTC and NTC tend to be very close in practice, so both estimate pattern correlation very well. From a more analytical point of view, their similarity follows from Zipf's law, which ensures that DB attributes often have many rare values. For example, approximately 65% of DBLP (*dblp.uni-trier.de*) authors have published only one paper [22]. Zipf's law is not the only source of rare attribute values. Entities typically have keys and often have semi-keys, such as *title* in Fig. 1. These attributes' values are highly selective, i.e., rare. Further, a pattern will be highly selective if it has at least one highly selective path. For example, *bib/conf/title* is not highly selective, but almost all values of *bib conf title -1 paper title -1 -1 -1* occur only once in the DB excerpted in Fig. 1(a). Hence NSTC and NTC tend to be close for a pattern with many instances. We compute the values of NSTC as for NTC [7], in a preprocessing stage.

As mentioned in Section 4, SFQIs adapt TF-IDF methods for IR-style ranking. Assume a WVS transformation maps path p in D to $T(p)$ in $T(D)$. Since there could be different numbers of instances of p and $T(p)$, the DF of the terms in the values of p and $T(p)$ will be different. Hence, current TF-IDF methods are not design independent under WVS transformations. Thus *we redefine the DF of a term w in pattern p to be the number of distinct values of p that contain w*. With the redefined DF, we extend the pivoted normalization method [20] to determine the contextual rank $ir(t, Q)$ of a CA t with pattern p for query q:

$$ir(t,Q) = \sum_{w \in q,t} \frac{1 + \ln(1 + \ln(tf(w)))}{(1-s) + s(el_t/avel_p)} \times qtf(w) \times \ln\left(\frac{N_p + 1}{df_p}\right). \quad (1)$$

Here, $tf(w)$ and $qtf(w)$ are the number of occurrences of w in t and the qs, respectively. el_t is the total length of the

content of t, and $avel_p$ is the average length of the distinct values of p. N_p is the count of distinct values of p, and df_p is the number of distinct values of p that contain w. s is a constant; the IR community has found that 0.2 is the best value for s [20]. We combine ir and NSTC on a sliding scale as:

$$r(t, Q) = \alpha NSTC(p) + (1 - \alpha) ir(t, Q), \qquad (2)$$

where p is t's pattern and α is a constant that determines the relative weight given to structural versus contextual information. We determined the best value of α empirically.

The proposed ranking scheme has two problems. First, the user has to scan through many duplicate patterns in the list of answers. Second, the IR formula may rank larger patterns in the same equivalence class higher, as they have more paths and therefore may contain more query terms. To address these problems, we group patterns with equal values for NSTC and the same set of paths before query time. After finding the CAs at query time, we find the equivalence class of the pattern of each CA and consider only CA(s) with the smallest patterns in each class. If there is more than one such pattern, we break the ties arbitrarily. We call the new approach *Duplicate Aware CR (DA-CR)*.

Theorem 5.2: DA-CR is weakly design independent.

To compute ir in Formula (2), for every term w and pattern p up to a given size, we must compute $df_p(w)$, the number of distinct values of p that contain w; and N_p, the number of distinct values of pattern p. We also must compute $avel_p$, the average length of all distinct values in p. Since the number of patterns and terms in a large DB is huge, exact computations of these values are prohibitively expensive. Furthermore, this information occupies a lot of space on disk. Thus, we estimate them by assuming that the terms occur in the paths of a pattern independently. Assume that p contains paths $q_1, \ldots, q_n$ and $p_w(q_i)$ shows the probability of term w appearing in distinct values of q_i:

$$\frac{df_p(w)}{N_p + 1} \approx \frac{df_p(w)}{N_p} \approx 1 - \prod 1 \leq i \leq n(1 - p_w(q_i)). \quad (3)$$

Similarly, we estimate $avel_p$ as $\sum_{1 \leq i \leq n} avel_{q_i}$. Spark [23] used the same idea to estimate IR-style statistics for relational data, and found the average error rate to be around 30%. We computed exact values for $avel_p$ and $\frac{df_p(w)}{N_p+1}$ for all patterns up to size 3 for DBLP and got an average error rate of 32%, which subsequent experiments show to be acceptable for ranking purposes.

6 EXPERIMENTS

We performed an extensive empirical study to measure the average case design independence of the SFQIs discussed in Section 4 on three real world databases: IMDB (205MB, 35 schema elements, max depth 7), DBLP (102MB, 31 schema elements, max depth 7), and Mondial's geographical information about countries (*dbis.informatik.uni-goettingen.de/Mondial*) (1.5 MB, 53 schema elements, max depth 5). Since the structure of DBLP was originally relatively flat and similar to Mondial, we transformed it by putting papers and articles under their associated journals and proceedings to get a more nested structure. Parts of the schemata for IMDB, DBLP, and Mondial can be found in appendix. Then we asked undergraduate CS majors who were not conducting this research and were familiar with the concepts and issues of DB design and XML to create new designs for each DB. We explained to them the properties of a WVS transformation, so that they created a new DB that was a WVS transformation of the original DB. They were required to provide an acceptable objective for each redesign. For instance, if they created a new entity, they had to explain how the new entity helps users understand the data and answers better (usability). For instance, one designer decided to create a new entity called *crew-info* that contains the information on *actor*, *actress*, *director*, *writer*, and other movie crew members. He also created another new entity *production-info* that contained all information on the production and distribution process of a movie, such as *location*, *production-company*, and *distributor*. Inside this new entity, the business information such as the box office of a movie was grouped under another entity called *overall-business*. The objective of this design was to increase the usability of the data because IMDB has more than 40 attributes for each movie and most attributes can have over 30 instances. Thus, it is very hard for users to find the information they want among all these attributes. Similarly, if designers merged some entities or denormalized the DB, they had to explain how it would improve query processing performance. For instance, one designer moved *year* and *booktitle* from *proceedings* or *journals* to *papers* in DBLP. This way the system needs to perform fewer structural joins between these nodes and other nodes from a *paper*, such as *title* and *author* [3].

We collected three redesigns for each DB. Since Mondial has a relatively simple and flat schema, our designers could not find a WVS transformation that involves duplication. Furthermore, we also collected a different design for each dataset from a real database in the same domain. We have used the design of SIGMOD Record at *www.dia.uniroma3.it/Araneus/Sigmod* as an alternative design for DBLP, the design of the movie data set from Metacritic at *www.metacritic.com* as an alternative design for IMDB, and the design of the geographic database from CIA World Factbook at *www.cia.gov/library/publications/the-world-factbook* as an alternative design for Mondial. Since some of the schema elements in our data are not presented in their schemata, we added some new schema elements to these designs in order to make their transformations VS preserving. Parts of the transformed schemata for all data sets can be also found in appendix. The designers wrote one XSL script for each new design to transform the DB instance. We name each design by its objective. We collected 80 keyword queries for DBLP, 79 keyword queries for IMDB, and 60 keyword queries for Mondial, submitted by 16 users. The query length ranged from 2 to 5. A complete list of the queries can be found in the Appendix. Descriptions of the modified schema elements can also be found in the appendix.

DB	CVLCA	XSearch	XReal
DBLP	11	11	1
IMDB	0	0	0
Mondial	26	26	33

TABLE 1: Average number of queries whose answers' LCA is the DB root

DB	CVLCA	XSearch	XReal	Op.CVLCA	Op.XSearch	Op.XReal
DBLP	3	3	3	14	14	4
IMDB	0	0	11	0	0	11
Mondial	0	0	5	26	26	6

TABLE 2: Average number of queries returning no results

As mentioned in Section 4, SFQIs filter out CAs and/or rank CAs. We implemented filtering methods SLCA, CVLCA, XSearch, XReal, and ELCA, and ranking methods CR, XSearch, XReal, and XRank. Accordingly, we used two metrics to compare the result lists or multisets delivered by a method over different designs of a DB. For filtering methods, we adapted the Jaccard index to measure the similarity between their results, by counting the number of different CAs in the multiset results, and normalizing it by dividing by the total number of CAs from both results together. If the results have exactly the same CAs, the value is 1; if they do not overlap, the value is 0. For rank-based comparison, we used Kendall's tau metric [24], which penalizes a list if a CA occurs in a position different from its position in the other list. We normalized Kendall's tau by dividing it by its maximum value, $n(n-1)/2$, where n is the total number of elements in the list. If the list of results from two DBs is identical, then the measurement is 1, and if the ranked result from one DB is the reverse of the other's, then the measurement is 0. Kendall's tau is designed to compare lists with the same number of elements. As ranking methods such as XSearch and XReal do both ranking and filtering, they can produce ranked lists of different lengths for the same query over different DBs. Hence, we do multiset comparisons of these methods to find the difference between the multisets of elements in both lists. Then, we assume that the missing elements from a list are ranked at the end of the list, and compute Kendall's tau to find the difference between the rankings in the lists. Normally, ranking methods return too many CAs for an input keyword query, but users focus more on the top ranked results. Hence, we also compute the average design independence for ranking methods over only the top 20 CAs for each query.

6.1 Average Design Independence

The first five bars (solid colors) for each dataset in Figure 7 show the design independence of the filtering methods discussed in Section 4 over DBLP, IMDB, and Mondial, averaged over the different designs. Since IMDB's schema is more complex than DBLP's and Mondial's, all methods except ELCA have their lowest average design independence for IMDB. IMDB has more paths and patterns than DBLP and Mondial, making its CAs more diverse. ELCA has its lowest average design independence for Mondial because ELCA is a distance based method, and all transformations over Mondial change the distance between leaf nodes. Since Mondial was originally very flat , a small change in distances between nodes makes a relatively large difference in query results.

The first four bars (solid colors) for each dataset in Figure 8 show the design independence of the ranking methods in Section 4 for designs of DBLP, IMDB, and Mondial, respectively. Similarly, Figure 9 shows the average design independence of these methods considering only the top 20 answers. Interestingly, the resulting values for top 20 answers are quite close to the values computed over all ranked answers, except for a noticeable difference in IMDB, which we discuss later. DA-CR delivers perfect design independence over all three data sets (not shown in the figures). XSearch and XReal deliver their lowest average design independence over IMDB. CR and XRank, however, show their lowest design independence for DBLP and Mondial, respectively, for reasons explained later. All ranking methods except DA-CR deliver their lowest average design independence over IMDB when considering only the top 20 CAs as shown in Figure 9. Since Mondial has the simplest schema, all filtering and ranking methods except for DA-CR generally have their highest average design independence for Mondial. Since all the transformations over Mondial are VS preserving, CR provides perfect average design independence over Mondial. The other filtering and ranking methods do not provide perfect average design independence even over such a relatively simple DB.

We have proved that if a method is weakly design independent, it delivers perfect design independence over all WVS transformations of a DB. Also, we have proved that if a method is design independent but not weakly design independent, it will show its lowest design independence over WVS preserving transformations. Our experiments confirm these theoretical results. DA-CR delivers perfect design independence for all transformations. CR shows its lowest design independence for *Performance* transformations over DBLP and IMDB, which are WVS preserving. As DA-CR provides perfect design independence over all transformations and DBs, we ignore it in the rest of our analysis of average design independence. Since other filtering and ranking methods are not (weakly) design independent, there is no specific type of transformation where their design independence is worst. Generally, method M has lower average design independence for transformation T, if T modifies the information used by M to filter or rank CAs the most. Hence, methods are less design independent for transformations that extensively change the DB structure, either by introducing new nodes or modifying the nodes' positions. As discussed in Section 4, if a method is not design independent, it returns dissimilar results under *at least one* VS preserving transformation. Interestingly, none of the design dependent methods provides perfect average design independence over *any* VS preserving transformation in our experiments. This observation underlines the need for design independent query interfaces in real-world settings.

For DBLP, all ranking methods except XSearch deliver their lowest average design independence with the

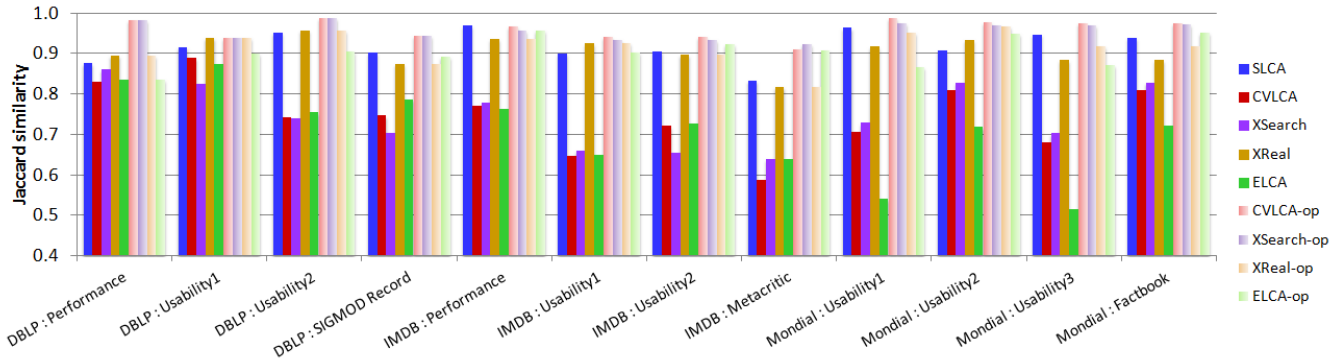

Fig. 7: Design independence of filtering methods

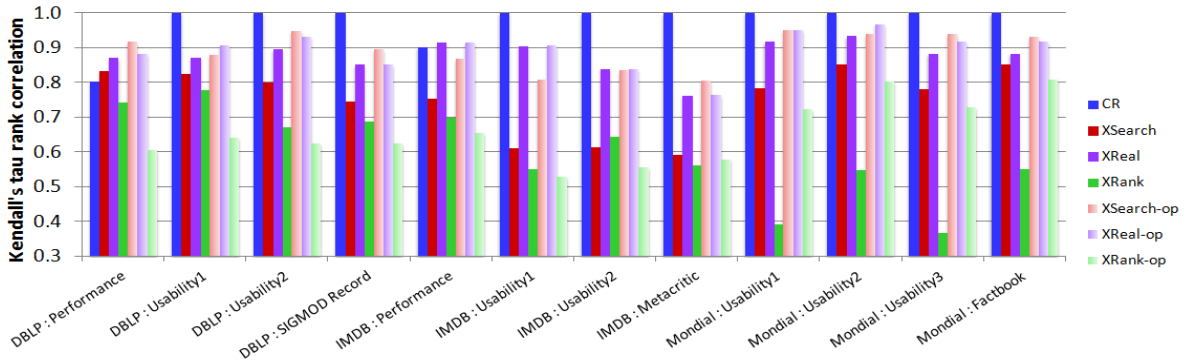

Fig. 8: Design independence of ranking methods

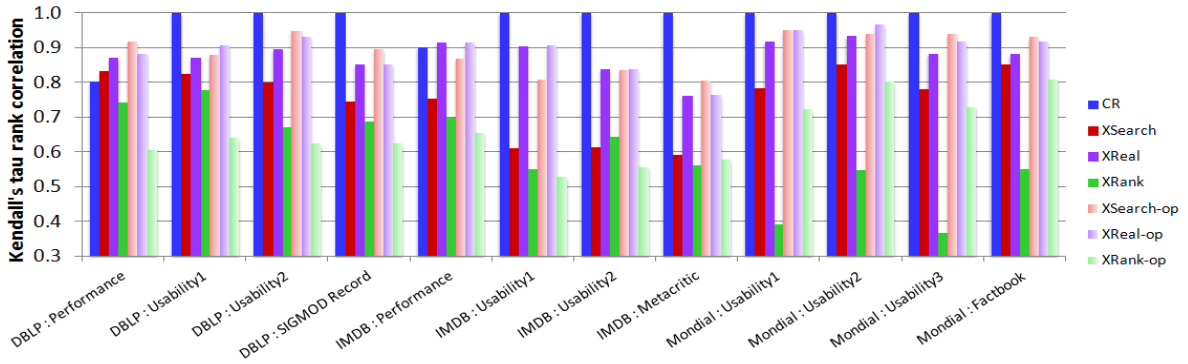

Fig. 9: Design independence of ranking methods for top 20 answers

Performance transformation, because this mapping adds many new and duplicate nodes. As discussed in Section 4, XSearch ranks CAs by their sizes. This CA property is changed the most by *Usability2* and in *SIGMOD Record* transformations. Thus, XSearch delivers its lowest average design independence for these redesigns of DBLP. We observe a similar trend for filtering methods. XReal and SLCA deliver their lowest average design independence with *Performance*. They use the depth of the LCAs to filter the CAs, and this property is modified the most by *Performance*. *Usability2* adds more duplicate labels and changes the distance between leaf nodes the most. As other methods consider duplicate labels and the distance between leaf nodes in CAs, they show their lowest average design independence under this transformation.

Except for CR, in IMDB the average design independence of filtering and ranking methods for *Usability1* and *Usability2* is lower than for *Performance*. This is because only a few queries match the parts of IMDB changed by *Performance*. *Usability1* and *Usability2* extensively change the DB structure at popular query nodes such as *actor*, *actress*, and *title*. The average design independence for *Usability1* and *Usability2* over IMDB is very close for the different methods. All redesigns of Mondial are VS preserving. All methods except CR have their lowest average design independence with *Usability3* and their highest with *Usability2*. This is because *Usability3* adds more new nodes and increases the depth of the paths the most, whereas *Usability1* slightly changes the DB design.

Figures 7 shows that the structural filtering technique of SLCA and XReal provides better design independence than label based techniques such as CVLCA and XSearch. However, their higher degree of design independence is mostly because they filter out more CAs than other filtering methods. For instance, XReal returns on average fewer than 1/5th of the CAs returned by XSearch, XRank, or CVLCA over IMDB. Similarly, SLCA returns on average fewer than half of the CAs returned by XSearch, XRank, and CVLCA over IMDB. If an SFQI returns more CAs for a query, it is likely to have lower average design independence, assuming all other conditions remain the same.

All queries in our experiments are guaranteed to have at least one CA whose LCA is not the root of the DB. Table 2 shows the number of queries that return no answer at all, using filtering and ranking methods, averaged over the DB designs. XReal does not return any CA for many queries over all three DBs. It has been shown that SLCA and XReal have lower recall than other filtering methods [7]. Hence, the relatively high design independence of these methods comes at the cost of missing relevant answers, which is not a desirable property of an SFQI. Every filtering and ranking method but SLCA, CR, and DA-CR returns some CAs whose LCA is the root of the DB. This type of CA is not relevant in a large DB [7]. Table 1 shows the number of queries where the LCA of every returned CA is the DB root, for CVLCA, XSearch, and XReal, averaged over all DB designs. The recall of these methods for such queries is zero.

After DA-CR, CR delivers the highest overall average design independence among the ranking methods. Although CR is not weakly design independent, its design independence is better than all ranking methods except for DA-CR and XReal for *Performance* with IMDB and DBLP. This shows that if a method does not rely on schema details, it generally provides higher average design independence. XReal has the second best average design independence for *Performance* over DBLP and IMDB after DA-CR. However, XReal returns on average fewer than 1/6th of the CAs retuned by CR over DBLP with *Performance*. Similar results hold for XReal answers over IMDB. XReal has considerably lower recall than CR [7], as confirmed in Tables 1 and 2. A method that never returns any answers will have perfect design independence, so it is not surprising that XReal is slightly more design independent than CR over DBLP and IMDB for *Performance*, but it misses many relevant answers. XRank's rankings depend very much on the shape of the DB. Since all transformations make significant changes to DB shape, XRank has the lowest average design independent among all ranking methods.

The ranking algorithms of SLCA, CR, and DA-CR are designed so that they do not return any CA whose LCA is the root of the DB. To be fair to other methods, we changed the ranking algorithms of other methods so that they discard such CAs. The last four bars in brighter colors in Figures 7 show average design independence after this optimization. Generally, the design independence of all filtering methods improves dramatically after this optimization, but no method delivers perfect average design independence. Though CVLCA and XSearch improve significantly, as shown in Table 2, they do not return any CA for a considerable number of queries. As discussed above, returning much lower number of answers tends to increase design independence, but severely penalizes recall. The average design independence of XReal does not change much. ELCA improves, but is still far from delivering perfect average design independence.

The last three bars with brighter colors in Figures 8 show average design independence of ranking methods over different DBs after optimization through removal of CAs whose LCA is the DB root. The average design independence of XSearch and XReal increase significantly, but the optimized versions of XSearch and XReal do not return any CA for a considerable number of queries. The average design independence of XRank decreases after optimization, which shows that its ranking method is quite unstable. Overall, the average design independence of most ranking methods increases after optimization. However, they still do not deliver a perfect design independence as DA-CR does on every transformation, or as CR does on VS transformations.

	DA-CR	CR
IMDB	0.721	0.714
DBLP	0.837	0.832
Mondial	0.881	0.881

TABLE 3: Mean average precision for DBLP and IMDB queries

6.2 Effectiveness

As discussed above, higher design independence does not necessarily mean better effectiveness, i.e., better recall and precision. However, our previous work has already shown that CR delivers better ranking quality than the previously proposed methods discussed in this paper [7]. To show that DA-CR is at least as effective than CR, we compared the ranking quality of DA-CR and CR over three real-world DBs. We asked 15 users who were not conducting this research to provide up to 5 keyword queries for IMDB and DBLP. We collected 25 queries for DBLP, 40 queries for IMDB, and 35 queries for Mondial. There are fewer queries for DBLP because some of the users are not CS researchers, and could not provide meaningful DBLP queries. A complete list of queries can be found in [7]. We developed a query processing prototype based on the baseline algorithm that returns every CA for each query. Our users submitted their queries to this system and judged each CA as relevant or irrelevant by selecting a checkbox in front of each CA. We compared the ranking quality of CR and DA-CR using mean average precision [20]. We use the NSTC and NTC values generated by $EV = 3$ and $L = 5$ for all DBs, which suffices for our queries. We set α to 0.84 for all DBs, which provides the best empirical results. Table 3 shows that DA-CR has almost the same MAP as CR over Mondial and DBLP, and a better MAP for IMDB. The new TF-IDF formula in DA-CR delivers better

ranking quality for IMDB, because it has a more nested structure. Each query over IMDB returns more candidate patterns than for the DBLP and Mondial queries.

7 Conclusions

We introduced a highly desirable property for schema free query interfaces, called design independence. We formalized this property and analyzed and compared the design independence of current keyword search and schema free query interfaces. We differentiated between design independence and weak design independence; the latter allows for data duplication and denormalization. We showed analytically and empirically that among current methods, only CR is design independent. Since CR is not weakly design independent, we provided a novel weakly design independent method, DA-CR. Our empirical studies showed that the average case design independence of other methods is lower than CR's, which in turn is lower than DA-CR's when the new DB design affects data redundancy. Finally, we found that the ranking quality of DA-CR is generally at least as good as that of CR.

References

[1] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram, "XRANK: Ranked Keyword Search over XML Documents," in *SIGMOD*, 2003.

[2] S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv, "XSearch: A Semantic Search Engine for XML," in *VLDB*, 2003.

[3] Y. Li, C. Yu, and H. V. Jagadish, "Schema-Free XQuery," in *VLDB*, 2004.

[4] Y. Xu and Y. Papakonstantinou, "Efficient Keyword Search for Smallest LCAs in XML Databases," in *SIGMOD*, 2005.

[5] Z. Liu and Y. Chen, "Reasoning and Identifying Relevant Matches for XML Keyword Search," in *VLDB*, 2008.

[6] Z. Bao, T. W. Ling, B. Chen, and J. Lu, "Effective XML Keyword Search with Relevance Oriented Ranking," in *ICDE*, 2009.

[7] A. Termehchy and M. Winslett, "Using Structural Information in XML Keyword Search Effectively," *TODS*, vol. 36, no. 1, 2011.

[8] M. Arenas and L. Libkin, "A Normal Form for XML Documents," *TODS*, vol. 29, no. 1, pp. 195–232, 2004.

[9] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," *CACM*, vol. 13, no. 6, pp. 377–387, 1970.

[10] M. Zaki, "Efficiently Mining Frequent Trees in a Forest," *TKDE*, vol. 17, no. 8, pp. 1021–1035, 2005.

[11] G. Li, J. Feng, J. Wang, and L. Zhou, "Effective Keyword Search for Valuable LCAs over XML Documents," in *CIKM*, 2007.

[12] R. Hull, "Relative Information Capacity of Simple Relational Database Schemata," *SICOMP*, vol. 15, no. 3, 1986.

[13] W. Fan and P. Bohannon, "Information Preserving XML Schema Embedding," *TODS*, vol. 33, no. 1, 2008.

[14] T. Antonopoulos, W. Martens, and F. Neven, "The Complexity of Text-Preserving XML Transformations," in *PODS*, 2011.

[15] M. Vardi, "The universal-relation data model for logical independence," *IEEE Software*, vol. 5, pp. 80–85, 1988.

[16] A. Termehchy, M. Winslett, and Y. Chodpathumwan, "How Schema Independent Are Schema Free Query Interfaces?" in *ICDE*, 2011.

[17] Y. Zhou and B. Croft, "Ranking Robustness: A Novel Framework to Predict Query Performance," in *CIKM*, 2006.

[18] D. Fetterly, M. Manasse, M. Najork, and J. L. Wiener, "A Large-Scale Study of the Evolution of Web Pages," in *WWW*, 2003.

[19] A. Termehchy and M. Winslett, "Keyword Search for Data-Centric XML Collections with Long Text Fields," in *EDBT*, 2010.

[20] C. Manning, P. Raghavan, and H. Schutze, *An Introduction to Information Retrieval*. Cambridge University Press, 2008.

[21] C. Yu and H. V. Jagadish, "Schema Summarization," in *VLDB*, 2006.

[22] E. Elmacioglu and D. Lee, "On Six Degrees of Separation in DBLP-DB and More," *SIGMOD Record*, 2005.

[23] Y. Luo, X. Lin, W. Wang, and X. Zhou, "SPARK: Top-k Keyword Query in Relational Databases," in *SIGMOD 2007*.

[24] M. Kendall and J. D. Gibbons, *Rank Correlation Methods*. Edvard Arnold, London, 1990.

Arash Termehchy is a PhD student in Computer Science at University of Illinois at Urbana-Champaign. His research interests are in the area of large scale data management, data mining, human centric data management, and information retrieval. He is the recipient of the best student paper award of ICDE 2011 and the Yahoo! Key Scientific Challenges Award in 2011-12. He holds a Master's degree from Sharif University of Technology and a Bachelor's degree from Iran University of Science and Technology both in computer science.

Marianne Winslett is a professor in the Department of Computer Science at the University of Illinois. Her research interests lie in information security and in the management of scientific data. She is currently on the editorial boards of ACM Transactions on the Web and ACM Transactions on Information and System Security. She is an ACM Fellow, and received a Presidential Young Investigator Award from the National Science Foundation in 1989. She has received multiple best paper awards from data management and security conferences such as VLDB, ICDE, and USNIX Security. She received her PhD in Computer Science from Stanford University in 1987.

Yodsawalai Chodpathumwan is a PhD student in Computer Science at University of Illinois at Urbana-Champaign. Her current research interests are in data and information management, information retrieval, and semantic Web. She graduated with a Bachelor's of Engineering in Computer Science from University of Illinois at Urbana-Champaign, including Mathematics minor and Software Engineering certificate in 2011. She is the recipient of the best student paper award of ICDE 2011 and a Royal Thai Government scholar.

Austin Gibbons is a Master's Candidate in Computer Science at Stanford University. He graduated with a Bachelor's of Engineering in Computer Science from the University of Illinois at Urbana-Champaign in 2011. His research interests lie in distributed and parallel systems, machine learning applications, and information retrieval.