How Schema Independent Are Schema Free Query Interfaces?

Arash Termehchy #1, Marianne Winslett #1, Yodsawalai Chodpathumwan #1

#*Department of Computer Science, University of Illinois at Urbana-Champaign*
Urbana, IL 61801
1`termehch,winslett,ychodpa2@uiuc.edu`

***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 (*SFQIs*) 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 but equivalent schemas for the same underlying information. 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.**

I. Introduction

Schemas are crucial for finding desirable query answers, and lack of understanding of the schema can be a significant obstacle for users who want to access the information in a database. 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 read it), even computer-savvy users find it challenging to explore the DB and formulate appropriate queries. Of course, ordinary 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 [7].

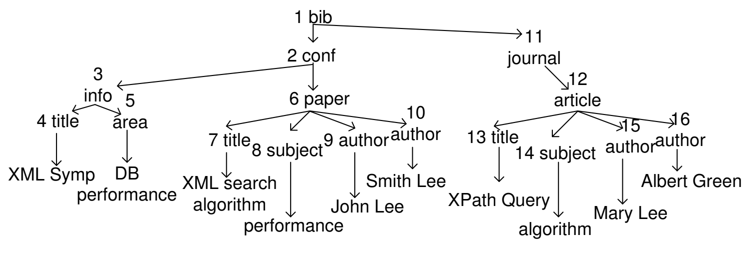
Schema free query interfaces [2], [3] and keyword query interfaces [1], [4], [5], [6], [8] (both are called SFQIs in this paper) have been proposed as solutions to these problems for XML DBs. With SFQIs, users do not need to know schema details or a query language. For example, suppose a user wants to 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 may revise the DB design over time to address issues such as redundancy, space overhead, performance, and usability [7], [9]. 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 [10]. 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 schema more usable. A DBA may also remove nodes such as *authors* to save space.

Current DBMSs provide physical independence for their users [11]. Thus, 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 after a change in the schema of the DB. In other words, users should not have to change their keyword or schema free queries in order to receive 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 the query *John Lee XML*, as they both represent the entity in question. Otherwise, users would have to learn aspects of the new schema in order to formulate the right 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 model database models [11]. 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 the users' perspective.
- We introduce and formally define the design independence property and explore its benefits for users of SFQIs. We introduce a group of transformations over DBs called *VS preserving* transformations that preserve the schema information and the content of the DB. We prove that if


(a)

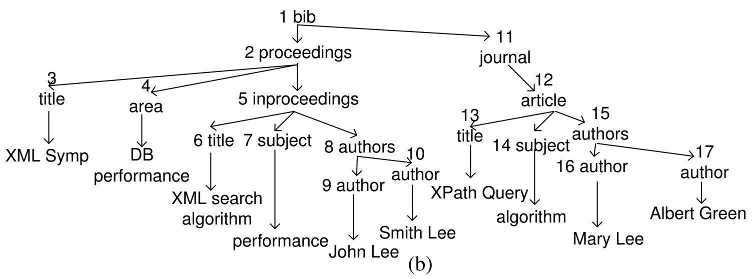

(b)

Fig. 1. Bibliographic databases

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 prove all except for one, CR [8], [12], are not design independent.
- Design independence requires the original and redesigned DBs to have the same content, so it does not cover the redesigns of a DB that introduce or eliminate data redundancy. However, many DB transformations, such as normalization, affect redundancy. We introduce a new property called *weak design independence* that addresses cases where the transformed DB contains more or less data redundancy than the original DB.
- No current method 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 TF-IDF statistics.
- We provide an extensive empirical study to evaluate the average case design independence of current and newly proposed methods using three real-world data sets. Our results 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. We have also performed an extensive user study and shown that DA-CR delivers almost the same or better ranking than other methods. Thus, its desired design independence property does not come at the cost of losing effectiveness.

Although our focus in this paper is on XML databases, the design independence property can be defined for relational and graph databases (e.g., RDF, OWL) similarly.

In the reminder of the paper, Section II presents basic definitions and related work. Section III defines design independence and explores its properties. Section IV analyzes this property for current SFQIs. Sections V and VI define and analyze weak design independence and introduce DA-CR. Section VII discusses experimental results comparing 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 VIII concludes the paper.

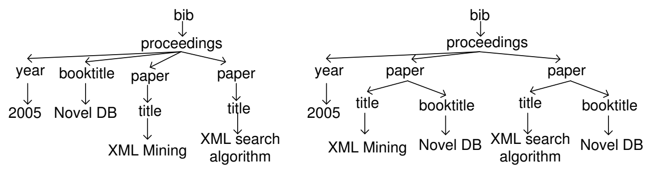


Fig. 2. Normalization of a bibliographic DB

II. Background

A. 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 an *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 node can be identified by its *path* from the root; e.g., node 3 in Fig. 1(a) has path */bib/conf/info*. 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 [13]. For instance, the prefix string for the subtree rooted at node 3 in Fig. 1(a) is *info title "XML Symp" -1 -1 area "DB performance" -1 -1*. We refer to an XML DB simply as a DB.

Trees T_1 and T_2 are **(label) isomorphic** if the nodes of T_1 can be mapped to the nodes of T_2 in such a way that node labels are preserved and the edges of T_1 are mapped to the edges of T_2. 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 are 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. 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 title -1 area -1* is ("XML Symp","DB performance"). 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 is a bit different between different SFQIs. In some SFQIs, users are able to 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* for short) to Q iff each of its content nodes contains at least one instance of each keyword term in Q and/or the attribute of the content node contains one label term. 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 to query Q_1 : *title performance*. We consider the path from the LCA of a CA to the root of the database as part of the CA, as it makes our analysis simpler. Also, methods such as XReal [6] use the depth of the LCA in their ranking scheme, which is captured by this path. All the CAs of a query form a multiset.

The baseline SFQI returns all CAs to a query [14]. However, as SFQs cannot be neatly mapped to XQuery or XPath, many CAs for SFQs are not helpful. 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* do not provide helpful answers for Q_2. The only relationship their attributes have is that they occur in the same DB, which is not interesting for the user. Researchers have proposed several SFQIs that filter the CAs they deem unhelpful [3], [4], [5], [14], [15] or rank the CAs so that the CAs they find interesting are placed at the start of the list of answers [1], [2], [6], [8]. For instance, one approach eliminates 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 for another candidate subtree answer, a_3 : *6 7 "XML search algorithm" -1 -1 9 "John Lee' -1 -1*. If an SFQI ranks its CAs, it returns a *ranked list of CAs* (list for short) for an input query. The *rank* of a CA in a list is its position in the list. If an SFQI only filters irrelevant CAs, it outputs a *multiset of CAs* (multiset for short) for a given query.

The IR community has been developing retrieval techniques for *text-centric* XML [16], where structures are simple. Extracting metadata (e.g., title, author, conference) from text-centric content produces data-centric XML, the focus of this paper, where structure plays an important role.

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 IMDB internet movie DB (*www.imdb.com*), users are more interested in new releases than older movies. In this paper we focus on domain-independent techniques, where the need for manual addition of domain-specific heuristics is minimized [1], [2], [3], [4], [5], [6], [8].

B. 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 [17], [18]. The goal of these works is to identify the schema mappings where every SQL or XQuery query over the source schema can be *manually translated to a new query* over the target schema. However, our work is to identify the mappings where SFQIs will be able to return the same answers *without changing the query*. Our work proposes a new SFQI that is able to complete this task.

Schema evolution has been studied in contexts such as data integration and maintenance [9], [7]. The goal is to adapt the current system to the new schema with minimal effort. Work in this area does not explore the design independence property, as these systems must have full knowledge of the schema. In our previous work [8], [12], we have used examples to illustrate the benefits of design independence, and the potential of statistical based methods to have such a property. However, we did not introduce the property and did not provide any framework to define and explore the design independence property. We also did not explore it over SFQIs, either analytically or experimentally. We also explore the information preservation properties of design independence in this paper. More importantly, we show that there are some cases where the statistics-based method proposed in [8], [12] is not, in fact, design independent. We propose an extension to this method that is design independent. We also introduce weak design independence and propose a novel SFQI that has this property.

III. Design Independence

A **transformation** T over DB D is a function that modifies D and generates a new DB $T(D)$ [17], [18]. We assume that any SFQI that operates over the transformed DB does not know anything about the original DB or the transformation.

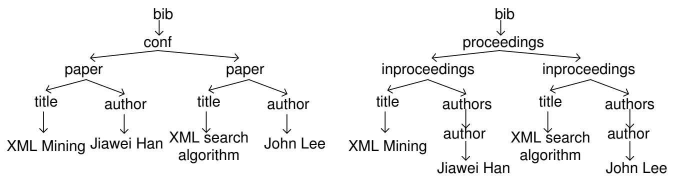


Fig. 3. Answers to Q_4 over original and transformed bibliographic DBs

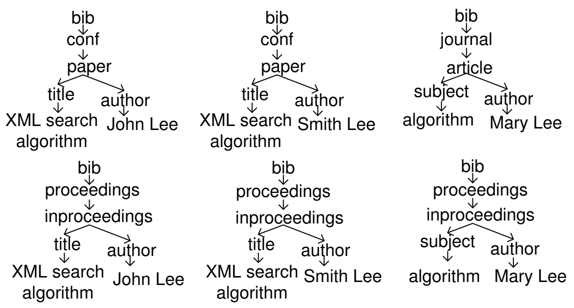


Fig. 4. Answers to Q_2 over original and transformed bibliographic DBs

Answers to a query represent the information sought by the users who submit the queries. If two answers 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 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 gets modified, an SFQI may not be able to return exactly the same list (multiset) of candidate answers for a query over the old and new DBs. For instance, Fig. 1(b) is the result of a transformation over 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 can assume that users receive essentially the same information from these CAs. Hence, we argue that users gain the same information from two answers that express *equivalent relationships* between data items that have *the same content*. Therefore, we consider an SFQI to be design independent if it returns essentially the same information for every query over the original and transformed DBs. However, we must precisely define what it means for answers over the original and transformed DBs to convey the same information to users.

A. Preserving Content

Answers that convey the same information must contain the same content. For instance, if the content of node 13 in Fig. 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 new child node for the same *paper* node called *paperInfo* that contains the content of the eliminated *author* and *title* nodes. The CAs for Q_3 over the original and transformed DBs have the same content. Nevertheless, the CA of the original DB represents the author and the title of the paper in different nodes and the CA of the transformed DB shows the author and the title of the paper in the same content node. By looking at the CA of the transformed DB, some users may not be able to distinguish the token(s) that represent the title and the token(s) that represent the author of the paper.

Definition 1: CA s_1 is **value equivalent** to CA s_2 iff there is a bijective mapping that maps each content node of s_1 to an equal content node of s_2.
We consider two content nodes equal if they are lexicographically equal. 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 2: List l_1 is **value equivalent** to list l_2 iff there is a bijective mapping M that 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 2 requires the value equivalent lists to have the same number of CAs. Value equivalency is defined for multisets similarly. There may be more than one value equivalent 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 the 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", it will not be possible to define a value equivalent mapping between the CAs of Q_2 over the original and transformed DBs. Since SFQIs and transformations do not know about each other, SFQIs 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 3: A transformation T over DB D is **value preserving** if it maps each pattern p in D to exactly one pattern $T(p)$ in DB $T(DB)$, such that there exists a bijective mapping that maps each instance of p to a value equivalent instance of $T(p)$.

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

Proof: According to Definition 3, if q does not have any CA over D, it will not have any CA over $T(D)$. Hence, the lists of answers are value equivalent. Consider the case where q has at least one CA over D. Each CA is an instance of a pattern. Since all pattern instances of D and $T(D)$ are value equivalent, each CA of q over D is value equivalent with one and only one CA of q over $T(D)$. Thus, the multisets for q over D and $T(D)$ are value equivalent. ■

Since a value preserving transformation may change the labels of the attributes, 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

```
<!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)

restrict value preserving transformations so that they preserve the labels of the attributes in the original DB.

B. 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, an SFQI is not able to deliver answers with exactly the same structure over the original and new DB for every query. However, if users can translate the information represented by the structures of the answers of an SFQI to query q over the original DB to the structures of the answers of q over the transformed DB, we can assume that the SFQI returns structurally similar answers to q over both DBs. Since an SFQI uses schema information to rank and/or filter CAs, we must consider only the 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 the same amount of information as S_1 [17], [18]. In other words, we can reconstruct the information available in each DB of the original schema S_1 from its transformed DB. This definition is mainly 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.

Fig. 5 and Fig. 6 depict the schema of the DBs excerpted in Fig. 1(a) and (b), respectively. The transformation between these schemas is bijective, as each DB following the schema of Fig. 5 is mapped to one and only one DB following 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 attributes nodes to the*bib* node. Since we consider 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, the transformation is not bijective and loses schema information.

First we prove that it is *possible* to translate the CAs to a query over a DB to its bijective transformation.

Proposition 2: Given a bijective transformation T, there exists a computable function that maps the multiset of query q over DB D to the multiset of q over $T(D)$.

Proof: We can write each possible CA for q over D as an XQuery expression (or in any query language that supports composition). The multiset of CAs to a query is the union of these expressions. Therefore, according to [18], if the transformation is bijective, there exists a computable function f for each query that maps the CAs over the original DB to the CAs over the transformed DB. ■

We must also consider the limitations of domain-independent SFQIs. If two nodes in DB have the same label, domain-independent query interfaces in general, 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 more *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 identify 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 different roles in query interfaces including SFQIs [1], [2], [6], [8], [15]. 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], [8], [15]. 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 *a-subject* that is a child of *journal* in Fig. 1(a). It is reasonable to assume that an SFQI may rank CAs containing articles under the same *a-subject* node higher than the CAs containing articles under different *a-subject* nodes. This is because the first group of CAs represent a more interesting relationship between articles than the second group of CAs. 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 the CAs that have 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 shown in Fig. 5. It also shows its value equivalent answer from a DB whose schema is depicted 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 4: Patterns p_1 and p_2 are **structurally equivalent** iff there is a bijective mapping M between all subpatterns s and r of p_1 and subpatterns $M(s)$ and $M(r)$ of p_2, such that s and r are isomorphic iff $M(s)$ and $M(r)$ are isomorphic.

CAs that are instances of structurally equivalent patterns are **structurally equivalent**. According to Definition 4, all value equivalent CAs whose patterns are paths are structurally 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 (multiset). 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 that are value equivalent with the ones in the first row. The value equivalent CAs are vertically aligned with each other. The leftmost CA and the CA in the middle of 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 expect the CAs in the second row that are value equivalent to these two CAs to have the same pattern. This property is met in this example. 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. Instead, they have the same paths as each other in the list of CAs over the new DB. Thus, we consider the two lists of CAs in Fig. 4 to be structurally dissimilar. We define the subpatterns of a list (multiset) of CAs as subpatterns of the patterns of its CAs.

Definition 5: A list of answers l_1 is **structurally equivalent** to a list of answers l_2 iff there is a bijective mapping M between all subpatterns s and r of l_1 and subpatterns $M(s)$ and $M(r)$ of l_2, such that:

- s and r are subpatterns of the same CA in l_1 iff $M(s)$ and $M(r)$ are subpatterns of the same CA in l_2.
- s and r are isomorphic iff $M(s)$ and $M(r)$ are isomorphic.

The first condition makes sure that the mapping keeps each CA structurally equivalent to its associated CA in the other list. Structural equivalence is defined similarly for multisets of CAs. There may be more than one structurally equivalent mapping for the multisets of CAs. If two lists (multisets) of answers are both value and structurally equivalent, we call them **VS equivalent**. 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 databases. If a transformation removes *journal*, *article*, *proceedings*, and *inproceedings* and connects all attribute nodes to the *bib* node in Fig. 1(b), no SFQI will be able to preserve the dissimilarities between different paths in the original DB.

Definition 6: A transformation T from DB D to $T(D)$ is **structure preserving** iff it maps each pattern p in D to exactly one structurally equivalent pattern $T(p)$ in $T(D)$.

If a transformation is both value and structure preserving, it is **VS preserving**. For instance, the transformation that maps the fragment shown in Fig. 1(a) to the fragment in Fig. 1(b) is VS preserving. 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 structurally equivalent answers for q over the old and new DBs.

Theorem 3.1: Given a VS preserving transformation T over DB D, the CAs for every query q over D and $T(D)$ are VS equivalent.

Proof: According to Proposition 1, CAs over D and $T(D)$ are value equivalent. Since each subpattern of the pattern of a CA is a pattern in the original DB, according to Definition 6, the multiset of CAs for q over D and $T(D)$ are VS equivalent. ■

Definition 7: An SFQI is **design independent** if it returns a VS equivalent list (multiset) of CAs for every query over each DB D and all its VS preserving transformations $T(D)$.

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. Also, structure preserving transformations treat content nodes as uninterpreted objects. 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.

Proof: Assume that there are two DBs D_1 and E_1 with schema S_1 that map to the same DB D_2 of S_2 using T. According to Definition 3, we can define a bijective mapping that maps each value node in D_1 to an equal value node in E_1. Similarly, based on Definition 6, we can define a a bijective mapping that maps each pattern from D_1 to an isomorphic pattern from E_1. Thus, D_1 and E_1 represent XML trees with equal patterns and values. ■

However, we can 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.

IV. Design Independence of Current SFQIs

The baseline technique for answering queries, called the LCA method, returns all CAs [14]. Based on Theorem 3.1, the LCA method is design independent. However, as mentioned in Section II, 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], [15], and/or rank the answers, via *ranking methods* [2], [6], [8].

A. Filtering Methods

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 3: SFQIs that use ELCA or 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 database design. For instance, one designer may decide to put *title* and *area* of a conference into a new node to increase readability of the data set and answers, while another designer may decide to remove these nodes to save space and increase performance.

As mentioned in Section II, SLCA-based methods [4], [5] filter 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 4: SFQIs based on the SLCA heuristic are not design independent.

Label based techniques such as XSearch [2] and CVLCA [15] 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 5: SFQIs that use the label based heuristic 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 data set more usable. Their results are also sensitive to renaming schema elements.

B. Ranking Methods

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 [19].

CR [8], [12] and NTPC [20] 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 [8], [12]. 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 database. 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: $NTC(q) = 1 - \frac{H(q)}{\sum_{1 \leq i \leq n} H(p_i)}$ [8], [12].

Proposition 6: 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.

Proof: T maps every pattern p in D to exactly one pattern $T(p)$ in $T(D)$. Since T is VS preserving, it maps each instance s of p to exactly one instance $T(s)$ of $T(p)$, where s and $T(s)$ are value equivalent. Hence, p and $T(p)$ will have the same values. Thus, they have the same NTC. ■

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.

V. 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 [10]. 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 [16]. 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 8: 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 9: 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 10: A transformation T over DB D is **weak VS (WVS) preserving** if it maps each equivalence class of duplicate patterns g to one and only one equivalence class of duplicate patterns $T(g)$, such that the representative pattern of g is VS equivalent to the representative pattern of $T(g)$.

For instance, 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 7: Given a WVS preserving transformation T over DB D, the CAs for every query q over D and $T(D)$ are WVS equivalent.

Proof: Since the CAs of the representative patterns in both DBs are VS equivalent, the CAs are WVS equivalent. ■

We call a method **weakly design independent** if it returns a WVS equivalent list or a set of CAs over a WVS transformation.

VI. Weakly Design Independent SFQIs

Since WVS preserving transformations are also VS preserving transformations, methods that are not design independent will not be weakly design independent. VS equivalent patterns over a WVS preserving transformation may have different numbers of equal values. Thus, if we consider the values of a pattern to be a multiset, their NTC values will not be equal. Hence, CR does not provide design independence over WVS transformations. We fix this problem by considering only distinct data items in a pattern when computing its correlation for CR.

Definition 11: Given pattern t with paths $p_1, \ldots, p_n$, $n > 1$, its **normalized set total correlation** (NSTC) is the NTC of its set of pattern values.

We argue that patterns' NSTC and NTC are usually very close, and therefore both estimate pattern correlation very well. Because of Zipf's law, many DB attributes have several rare values. For example, approximately 65% of DBLP (*dblp.uni-trier.de*) authors have published only one paper [21]. Zipf's law is not the only source of rare attribute values. Entities typically contain 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 the NSTC and NTC tend to be close for a pattern with many instances. Our empirical study in Section VII confirms this argument. We compute the values of NSTC similarly as for NTC [8], in a preprocessing stage.

As mentioned in Section IV, SFQIs adapt TF-IDF methods for IR-style ranking. Assume a WVS transformation maps path p in DB D to $T(p)$ in DB $T(D)$. Since there could be different numbers of instances of p and $T(p)$, the DF of the terms contained in the values of p and $T(p)$ will be different. Hence, current adaptations of TF-IDF methods over WVS transformation are not design independent. 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 [16] to determine the contextual rank $ir(t, Q)$ of a CA t with pattern p for query q:

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

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 [16]. We combine ir and NSTC on a sliding scale as:

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

where p is t's pattern and α is a constant that determines the relative weight given to structural versus contextual information. We determine the 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 6.1: DA-CR is weakly design independent.

Proof: All patterns in the same equivalence class have equal NSTC. As the mapped representative patterns in a WVS transformation are VS equivalent, they will also have the same NSTC. Since WVS transformations do not merge content nodes, the CAs whose patterns are the smallest in the mapped equivalence classes have the same size. As they are value

IMDB	DBLP	Mondial
Beautiful Mind	Madden TinyDB	Greek Orthodox
Edward Norton	William Yurcik	China Beijing
Artificial Intelligence	Radu Sion	Rostock Schwerin
True Dreams	Barbara Liskov	Czech Republic
Jackie Chan 2007	Hoarding	Kuala Lumpur
Basic Instinct	Authenticated Dictionary	Tyrol Styria Graz
Crime the Godfather	Language Based Security	Reggio Di Calabria
Peter Sellers Blake Edwards	CCS 2008	Berlin Erlangen Furth
Belgian Detective	Closed Frequent Pattern	Great Slave Lake
Slumdog Millionaire 2008	Social Network Evaluation	Andorra La Vella

TABLE I

SOME OF THE COLLECTED QUERIES OVER DBLP, IMDB, AND MONDIAL

Database Design	SLCA	CVLCA	XSearch	XReal	ELCA
Performance	0.8175	0.7592	0.7535	0.8186	0.7681
Usability1	0.8779	0.7696	0.7635	0.8474	0.7531
Usability2	0.9044	0.8588	0.8139	0.8866	0.8218

TABLE II

MULTISET COMPARISON FOR DBLP

Database Design	CR	XSearch	XReal	XRank
Performance	0.8218	0.7997	0.8083	0.8250
Usability1	1.0	0.8894	0.6908	0.7519
Usability2	1.0	0.9122	0.8400	0.8095

TABLE III

RANKED COMPARISON FOR DBLP

Database Design	SLCA	CVLCA	XSearch	XReal	ELCA
Performance	0.6813	0.6509	0.6151	0.7544	0.6670
Usability1	0.8783	0.6609	0.6563	0.7996	0.6464
Usability2	0.6227	0.6371	0.5620	0.7313	0.6326

TABLE IV

MULTISET COMPARISON FOR IMDB

Database Design	CR	XSearch	XReal	XRank
Performance	0.7182	0.7058	0.6955	0.5933
Usability1	1.0	0.6201	0.7875	0.5769
Usability2	1.0	0.5276	0.6955	0.5933

TABLE V

RANKED COMPARISON FOR IMDB

Database Design	SLCA	CVLCA	XSearch	XReal	ELCA
Usability1	0.9354	0.8274	0.8481	0.9399	0.5457
Usability2	0.9355	0.8762	0.8949	0.9410	0.7020
Usability3	0.9187	0.8013	0.8222	0.8139	0.5212

TABLE VI

MULTISET COMPARISON FOR MONDIAL

Database Design	CR	XSearch	XReal	XRank
Usability1	1.0	0.7843	0.9294	0.4143
Usability2	1.0	0.8347	0.9292	0.5483
Usability3	1.0	0.7636	0.7864	0.3926

TABLE VII

RANKED COMPARISON FOR MONDIAL

equivalent, they will have equal $tf(w)$ and el_t, and their patterns will have equal $avel_p$ and N_p. Thus, they will have the same score. As the two DBs have the same number of CAs, the results of DA-CR over the DBs are WVS equivalent. ■

To compute ir in Equation 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 data set 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 can estimate $avel_p$ as $\sum_{1 \leq i \leq n} avel_{q_i}$. Spark [22] 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 is acceptable for ranking purposes.

VII. EXPERIMENTS

A. Design Independence

We performed extensive empirical studies to measure the average case design independence of the methods mentioned in Section IV on three real world data sets: IMDB (198MB, max depth 7), DBLP (78MB, max depth 7), and Mondial's geographical information about countries (*dbis.informatik.uni-goettingen.de/Mondial*) (1.5 MB, max depth 5). Since the structure of the original DBLP data set was relatively flat and similar to Mondial, we converted it and put papers and articles under their associated journals and proceedings to get a more nested structure. Then we asked undergraduate computer science majors who were not conducting this research and were familiar with the concepts and issues of database design and XML to create new designs for each data set. 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 of the database. 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 the IMDB data set has more than 40 attributes for each movie and most attributes could have more than 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, a designer moved *year* and *booktitle* from the *proceeding*s or *journal*s to *paper*s in DBLP. This way an XML query processing system needs to perform fewer structural join operations for queries that contain information about these nodes and other nodes from a *paper* such as *title* and *author* [3].

We collected three redesigns for each data set. Since Mondial has a relatively simple and flat schema, our designers could not find a WVS transformation that involves duplication. The designers wrote one XSL script for each new design to convert the data from the original DB to the new DB. We named each design by its objective in the reported tables. We collected 80 keyword queries for DBLP and IMDB and 50 keyword queries for Mondial, submitted by 16 users. The query length ranged from 2 to 5.

As mentioned in Section IV, there are two types of SFQIs: those that filter out CAs (filtering methods) and those that rank CAs (ranking methods). Thus, we used two different metrics to compare the list and multisets of CAs delivered by the same method over different designs of the same data set. For filtering methods, we adapted the Jaccard index to measure the similarity between their results [23]: we counted the number of answers that were different in the multisets of answers, and normalized it by dividing by the total number of answers from both results together. If the answers from the datasets are the same, the value is 1; if the answers from the datasets are completely different, then the measurement is 0.

For rank-based comparison, we used Kendall's tau metric [24]. This metric penalizes a list when an answer occurs in a position different from its position in another list. Thus, if two lists have almost the same ordering, they have a low Kendall's tau distance. 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 answers from two datasets are ranked the same, then the measurement is 1, and if the ranked result from one dataset 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. However, ranking methods such as XSearch and XReal do both ranking and filtering, we may have ranked lists of different lengths for the same query over different data sets. 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.

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

TABLE VIII

MEAN AVERAGE PRECISION (MAP) FOR DBLP AND IMDB QUERIES

Tables II, IV, and VI show the similarities between the results of filtering methods discussed in Section IV for different designs over DBLP, IMDB, and Mondial, respectively. All methods deliver their worst design independence over IMDB, because the schema of IMDB is more complex. It has more paths and patterns than DBLP and Mondial, which results in more diverse answers to the queries. The results also show that the structural filtering techniques of SLCA and XReal provide better design independence than label based techniques such as CVLCA and XSearch. Interestingly, our designers introduced some duplicate labels in their new designs which created this issue. For instance, in *Usability1* in DBLP, all *author*s are grouped under a new node *authors* and all *editor*s are grouped under a new node *editors*, which introduces duplicate labels in the patterns that contain *author* and *editor* nodes. Since *Performance* over DBLP involves denormalization of the DB and duplicate paths, generally all methods show their lowest design independence for this design. Interestingly, these methods also show relatively low design independence for *Usability2* over IMDB. Since IMDB has a more complex structure, this transformation introduces many new entity types, which makes it hard for methods to return similar answers for the same query. The multiset based similarity for CR and DA-CR is 1.0, as they do not prune any CA (not shown in the tables).

Tables III, V, and VII show the similarity between answers for ranking methods. Similar to filtering methods, ranking methods deliver their lowest design independence over IMDB. CR delivers perfect design independence over the transformations that do not involve duplication, such as *Usability1* and *Usability2* over IMDB. For transformations that involve duplication, such as *Performance* over IMDB and *Performance* over DBLP, CR still shows good similarity compared to other methods. Overall, XReal is the second most design independent ranking method, as it does not rely on schema details as XSearch and XRank do. Nevertheless, as multiset based comparisons illustrate, ranking methods other than CR miss or add some answers over different designs of the same data set. DA-CR delivers perfect design independence over all designs of all data sets (not shown in the tables).

B. Effectiveness

We have shown that DA-CR is more design independent than CR. However, it does not necessarily follow that DA-CR is more effective than CR. Thus, we have empirically compared the ranking quality of DA-CR and CR over three real-world data sets. 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 computer science researchers, and could not provide meaningful DBLP queries. Some of the queries on data sets are shown in Table I. We developed a query processing prototype based on the baseline algorithm that returns every candidate answer for each query [14]. Our users submitted their queries to this system and judged each answer as relevant or irrelevant by selecting a checkbox in front of each returned answer. We compared the ranking quality of CR and DA-CR using mean average precision [16]. We use the NSTC and NTC values generated by $EV = 3$ and $L = 5$ for all data sets, which suffices for our queries. We set α to 0.84 for both data sets. Table VIII compares DA-CR with CR. DA-CR shows essentially 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. It has been shown that CR has superior ranking quality than other discussed methods [8], [12].

VIII. Conclusions

We have 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 identified two classes of design independent SFQIs: design independent and weakly design independent, which considers the effect of data duplication and denormalization. We showed 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. We presented extensive empirical studies that 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 better than or equal to the ranking quality of CR.

Acknowledgment

We thank Vagelis Hristidis for helpful comments and Austin Gibbons for useful feedbacks. This work is supported by NSF grant number 0938071.

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] C. A. Curino, H. J. Moon, and C. Zaniolo, "Graceful Database Schema Evolution: The Prism Workbench," in *VLDB*, 2008.
[8] A. Termehchy and M. Winslett, "Effective, Design-Independent XML Keyword Search," in *CIKM*, 2009.
[9] Y. Velegrakis, R. J. Miller, and L. Popa, "Mapping Adaptation under Evolving Schemas," in *VLDB*, 2003.
[10] M. Arenas and L. Libkin, "A Normal Form for XML Documents," *TODS*, vol. 29, no. 1, pp. 195–232, 2004.
[11] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," *Communications of the ACM*, vol. 13, no. 6, pp. 377–387, 1970.
[12] A. Termehchy and M. Winslett, "Using Structural Information in XML Keyword Search Effectively," *TODS*, vol. 36, no. 1, 2011.
[13] M. Zaki, "Efficiently Mining Frequent Trees in a Forest," *TKDE*, vol. 17, no. 8, pp. 1021–1035, 2005.
[14] A. Schmidt, M. Kersten, and M. Windhouwer, "Querying XML Documents Made Easy: Nearest Concept Queries," in *ICDE*, 2001.
[15] G. Li, J. Feng, J. Wang, and L. Zhou, "Effective Keyword Search for Valuable LCAs over XML Documents," in *CIKM*, 2007.
[16] C. Manning, P. Raghavan, and H. Schtze, *An Introduction to Information Retrieval*. Cambridge University Press, 2008.
[17] R. Hull, "Relative Information Capacity of Simple Relational Database Schemata," *SIAM Journal on Computing*, vol. 15, no. 3, 1986.
[18] W. Fan and P. Bohannon, "Information Preserving XML Schema Embedding," *TODS*, vol. 33, no. 1, 2008.
[19] D. Fetterly, M. Manasse, M. Najork, and J. L. Wiener, "A Large-Scale Study of the Evolution of Web Pages," in *WWW*, 2003.
[20] A. Termehchy and M. Winslett, "Keyword Search for Data-Centric XML Collections with Long Text Fields," in *EDBT*, 2010.
[21] E. Elmacioglu and D. Lee, "On Six Degrees of Separation in DBLP-DB and More," *SIGMOD Record*, 2005.
[22] Y. Luo, X. Lin, W. Wang, and X. Zhou, "SPARK: Top-k Keyword Query in Relational Databases," in *SIGMOD 2007*.
[23] P. Jaccard, "Nouvelles Recherches Sur la Distribution Florale," *Bull. Soc. Vaudoise Sci. Nat.*, vol. 44, 1908.
[24] M. Kendall and J. D. Gibbons, *Rank Correlation Methods*. Edvard Arnold, London, 1990.