Algorithms for Minimal Dependency Set and Membership Based on XML Functional Dependency and Multi-valued Dependency

Zhongping Zhang[1,2], Chunzhen Fang[1]
[1]The School of Information Science and Engineering, Yanshan University,
Qinhuangdao, Hebei, 066004, China
[2]The Key Laboratory for Computer Virtual Technology and System Integration of Hebei Province,
Qinhuangdao, Hebei, 066004, China
Email: zpzhang@ysu.edu.cn, fcz_chunzhen@163.com

***Abstract*—As the XML functional dependency and multi-valued dependency impact on the normalization design of semi-structured data, definitions of XML functional dependency, XML multi-valued dependency, path dependency base and the minimal dependency set are given in this paper. Algorithms for minimal dependency set and membership with path expression based on the coexistence of XFD and XMVD are then proposed. Finally, the correctness and termination of these algorithms are proved, and their time complexities are analyzed as well.**

***Index Terms*—XML functional dependency, XML multi-valued dependency, path dependency base, membership, minimal dependence set**

I. Introduction

XML (eXtensible Markup Language) inherits the powerful function of SGML(Standard Generalized Markup Language)[1] and makes up the deficiencies of HTML. It is a standard which is used for data expression and data exchange on the Internet, providing an effective means for data description and data exchange on the Internet applications. It is widely used in many fields [2-5]. XML schema is an important concept in the field of XML and it is the first step to build database applications. Currently, DTD develops well, and it is widely used in the practical applications of the XML document. However, because of some unusual data dependencies in XML database, it may result in data redundancies and abnormal operations due to design flaws for DTD [6].

In recent years, scholars have done a lot of exploration and research on the normalization of XML database: the normalization based on functional dependency [7-12] and multi-valued dependency [13-15]. These research literatures analyze the effect on XML data and free redundancy from a single point of view such as XML functional dependency or XML multi-valued dependency. However, they do not consider the effect on XML data and XML database normalization on the condition of coexistence of XML functional dependency and multi-valued dependency. Therefore, according to the inference rules based on the coexistence of XML functional dependency and XML multi-valued dependency [16], we propose algorithms for minimal dependency set (DEP-MINIMIZE algorithm) and membership (DEP-MEMBERSHIP algorithm) with path expression based on coexistence of XML functional dependency and multi-valued dependency. It simplifies the dependency set and ensures the simplification on analysis and calculation.

II. Preliminary Definitions And Notations

In this section, we present some preliminary definitions and notations that we need.

Definition 1: (XML Tree) An XML tree is defined as T=(V,lab,ele,att,val,root), it is said to conform to a DTD[9] D=(E_1,E_2,A,P,R, r), denoted by T⊨D[17], where

(i) T is the XML tree's name;

(ii) V is a finite set of nodes in T;

(iii) lab is a function from V to $E_1 \cup E_2 \cup A$,which assigns a identifier to each node in V. A node v in V is called a complex element node if $lab(v) \in E_1$; a simple element node if $lab(v) \in E_2$, and an attribute node if $lab(v) \in A$;

(iv) ele is a function from V to a sequence of V nodes ,so that for any $v \in V$, if $lab(v) \in E_1$, $ele(v)$ is a set of some children of v. The node in $ele(v)$ is element node, and if $ele(v)=\{v_1,...,v_m\}$, then $\{lab(v_1),...,lab(v_m)\} \in P(lab(v))$;

(v) att is a function from V to A. If $att(v, l)=v_1$,then $lab(v) \in E_1$ and $lab(v_1)=l$; if $att(v)=\{v_1,...,v_n\}$, then $\{lab(v_1),...,lab(v_n)\} \in R(lab(v))$, where $v \in V$, $l \in A$;

Manuscript received December, 1, 2013; revised March 24, 2014; accepted April 21,2014.

doi:10.4304/jsw.9.11.2837-2843

(vi) val is a function that assigns a value to each node. If a node v is a leaf node or a simple element node of T, val(v) is a string value which is either the content of a text element or the content of an attribute; otherwise val(v) is the node's identifier of v.

(vii) root is the unique root node labeled with complex element name r.

Example 1: Describe a college's relationship among three entities Course, Student and Teacher and store data information in Relational database. Relational schema R of database is designed as following:

Course(Cno, Cname) Student(Sno, Sname)
Teacher(Tno, Tname)
Courses(Cno,Sno, Tno,Cname, Sname, Tname)

If the relational database need to be stored as an XML document, this document's DTD D is expressed as:

<!ELEMENT Courses (Course*)>
<!ELEMENT Course (Cname,Student*)>
<!ATTLIST Course
Cno CDATA #REQUIRED>
<!ELEMENT Cname (#PCDATA)>
<!ELEMENT Student (Sname,Teacher)>
<!ATTLIST Student
Sno CDATA #REQUIRED>
<!ELEMENT Sname (#PCDATA)>
<!ELEMENT Teacher (Tname)>
<!ATTLIST Teacher
Tno CDATA #REQUIRED>
<!ELEMENT Tname (#PCDATA)>

There are some data instances in TABLE I:

TABLE I.
A PART OF STUDENTS' COURSE RECORDS OF ONELEGE COL

Cno	Cname	Sno	Sname	Tno	Tname
C01	XML	S001	Amily	T001	Mary
C01	XML	S002	Stephen	T001	Mary
C01	XML	S001	Amily	T002	Anne
C01	XML	S002	Stephen	T002	Anne

According to this course records, we can obtain an XML tree T which based on the DTD D. It is shown in Figure 1.

Definition2: (Path Instance on XML Tree) Let T be an XML tree that satisfied the given DTD D. A path instance [7] over an XML tree T is a sequence, $v_1 = v_r$ and for every v_i, $2 \le i \le n$, $v_i \in V$ and v_i is a child of v_{i-1}. A path $v_1.v_2. \ldots .v_{n-1}.v_n$ is defined as one path instance based on the path $p_1.p_2. \ldots .p_{n-1}.p_n$ if for all v_i, $1 \le i \le n$, $lab(v_i)=p_i$. All path instances based on a path p over a tree T form a set which denoted by Paths(p). All Path sets on D are defined as the Paths (D).

Example 2: As Figure 1 shows that, $v_1.v_2.v_4$ is a path instance of path Courses.Course.Cno, Paths(Courses.Course.Cno)={$v_1.v_2.v_4$, $v_1.v_3.v_6$}.

Definition 3:(XFD) Let T be an XML tree that satisfied the given DTD D. An XFD is a statement of the form $p_1, \ldots, p_k \rightarrow q_1, \ldots, q_m$, $k \ge 1, m \ge 1$. P={$p_1, \ldots, p_k$} and Q={$q_1, \ldots, q_m$} are subsets in Paths(D).There are arbitrary two distinct path instances $V=\{v_1^1.v_2^1.\ldots.v_{l-1}^1.v_l^1, \ldots, v_1^m.v_2^m.\ldots.v_{n-1}^m.v_n^m\}$

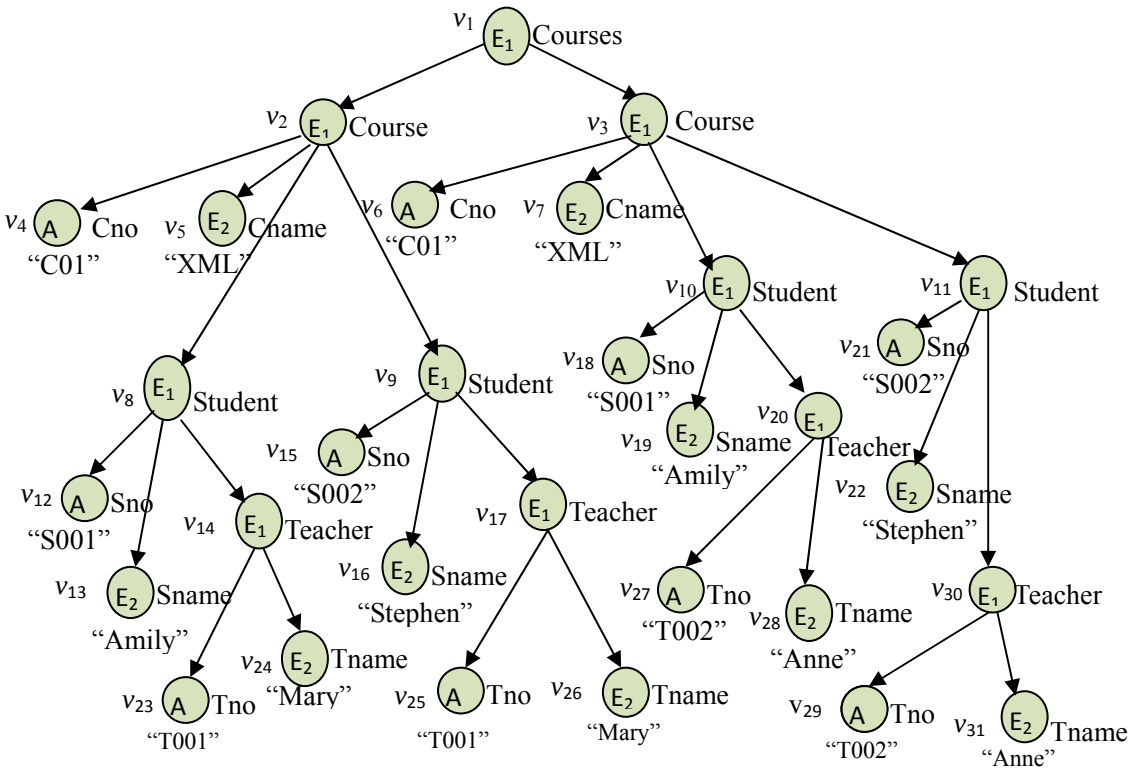

Figure 1. An XML tree formed from a part of students' course records

and W={ $w_1^1 . w_2^1 . \ldots . w_{l-1}^1 . w_l^1$, … , $w_1^m . w_2^m . \ldots . w_{n-1}^m . w_n^m$ } in Paths(Q), n≥1, l≥2, T satisfies the XFD: $p_1 ,. . .,p_k \rightarrow q_1, . . ., q_m$,where

(i) Q⊂P; or

(ii)For any two distinct path instances $v_1^i . v_2^i . \ldots . v_{t-1}^i . v_t^i$ and $w_1^i . w_2^i . \ldots . w_{t-1}^i . w_t^i$ in Paths(q_i), if Last(p_j)[16] ∈E_1 and x_{ij} =y_{ij} , or Last(p_j) ∉ E_1 and val(Nodes(x_{ij} , p_j)[16])∩val(Nodes(y_{ij}, p_j)) ≠∅ then val(v_t^i)=val(w_t^i); where x_{ij}={v|v∈{ v_1^i , …, v_t^i } ∧v∈N(p_j∩q_i)},y_{ij}={v|v∈{ w_1^i , …, w_t^i } ∧v∈N(p_j∩q_i)},1≤j≤k,1≤i≤m,t≥1.

We note that the path p_j ∩q_i is a prefix of q_i. There exists only one node in the set {v_1 , … ,v_t } also in N(p_j ∩q_i) , therefore x_{ij} contains only one node. y_{ij} .is similar to x_{ij} and it also contains only one node.

Definition 4: (XMVD) Let T be an XML tree that conforms to a DTD D. An XMVD is a statement of the form p_1 , … ,$p_k \rightarrow\rightarrow q_1$, … ,q_m |r_1 , … ,r_s , 1≤k,1≤m,1≤s. P={p_1 ,· · · ,p_k },Q={q_1 ,· · · ,q_m } and R={r_1 , …, r_s} are subsets in Paths(D), {{p_1, ..., p_k}∪{q_1, ..., q_m}∪{r_1, ..., r_s}}⊂Paths(D). The tree T satisfies the XMVD if there exists a q_i, 1≤i≤m, and two distinct path instances $v_1^i . v_2^i . \ldots . v_{t-1}^i . v_t^i$ and $w_1^i . w_2^i . \ldots . w_{t-1}^i . w_t^i$ in Paths(q_i),1≤t, where

(i) val(v_t^i) ≠val(w_t^i);

(ii)There exists a r_j∈R,1≤j≤s, and two nodes z_1, z_2 ,where z_1 ∈Nodes(x_{ij} ,r_j) and z_2 ∈Nodes(y_{ij} ,r_j), such that val(z_1) ≠val(z_2);

(iii)For all p_h ,1≤h≤k, there exists two nodes z_3 and z_4 ,where z_3 ∈Nodes(x_{ijh} ,p_h)and z_4 ∈Nodes(y_{ijh} ,p_h), such that val(z_3)=val(z_4);

(iv)There exists a path instance $v'^i_1 . v'^i_2 . \ldots . v'^i_{t-1} . v'^i_t$ in Paths(q_i), we have val(v'^i_t)=val(v_t^i), and there is a node z'_1 in Nodes(x'_{ij} , r_j) , such that val(z'_1)=val(z_2). There exists a node z'_3 in Nodes(x'_{ijh}, p_h)such that val(z'_3)=val(z_3);

(v)There is a path instance $w'^i_1 . w'^i_2 . \ldots . w'^i_{t-1} . w'^i_t$ in Paths(q_i) making val(w'^i_t)=val(w_t^i), and there exists a node z'_2 in Nodes(y'_{ij}, r_j), we have val(z'_2)=val(z_1) and there exists a node z'_4 in Nodes(y'_{ijh}, p_h)such that val(z'_4)=val(z_4).

Where, x_{ij} ={v|v∈{ v_1^i , ... , v_t^i } and v∈N(r_j∩q_i)}, y_{ij}={v|v∈{ w_1^i , ... , w_t^i } and v∈N(r_j∩q_i)}, x_{ijh} ={v|v∈{ v_1^i , ... , v_t^i } and v∈N(p_h∩r_j∩q_i)}, y_{ijh}={v|v∈{ w_1^i , ... , w_t^i } and v∈N(p_h∩r_j∩q_i)}; x'_{ij} ={v|v∈{ v'^i_1 , ... , v'^i_t } and v∈N(r_j∩q_i)}, y'_{ij} ={v|v∈{ w'^i_1 , ... , w'^i_t } and v∈N(r_j∩q_i)}, x'_{ijh} ={v|v∈{ v'^i_1 , ... , v'^i_t } and v∈N(p_h∩r_j∩q_i)}, y'_{ijh} ={v|v∈{ w'^i_1 , ... , w'^i_t } and v∈N(p_h∩r_j∩q_i)}.

We note that the path r_j ∩q_i is a prefix of q_i, there exists only one node in set { v_1^i , ... , v_t^i } also in N(r_j∩q_i), therefore x_{ij} contains only one node. y_{ij}, x_{ijh}, y_{ijh}, x'_{ij}, y'_{ij}, x'_{ijh}, y'_{ijh} are similar to x_{ij}. The XMVD is symmetrical, i.e. the XMVD: p_1, ... , $p_k \rightarrow\rightarrow q_1$, …, q_m|r_1, …, r_s holds iff the XMVD: p_1 , … ,$p_k \rightarrow\rightarrow r_1$, … ,r_s |q_1 , … ,q_m.

Definition 5: (Minimal Base) Let T be an XML tree that satisfied the given DTD D, Path set is P={P_1,...,P_k}. We assume that Paths(D)= P_1∪...∪P_k. The minimal base of P is denoted by MB(P) and it is a partition of Paths(D), S_1,...,S_q ,where

(i) Each P_i is a set of S_j by union operation;

(ii) There exists no partition that satisfies the condition (i) and the number of partition is less than q, where 1≤i≤k, 1≤j≤q.

Definition 6: (Dependency Base) Let ∑ be the set of XFD and XMVD over the complete instance document XML tree T which satisfied DTD D, P⊆Paths (D), the minimal base MB(P^+) of P^+ [16] is a path dependency base relative to ∑, denoted by DEP(P).

Definition7: (Logical Implication) Let Paths(D) be the path set of DTD D, and ∑ is the data dependency set of D. If each XML tree T of D satisfies ∑ and P→→Q|R or P→Q, we call P→→Q|R or P→Q implicated logically by ∑, and denotes ∑⊨P→→Q|R or ∑⊨P→Q.

III. Membership Algorithm On The Condition Of Coexistence Of XML Functional Dependency And XML Multi-Valued Dependency

A.1. Path Dependency Base Algorithm

Dependency base is a partition of attribute set of relational data schema in the relational data theory. The path dependency base can gain the logical implication of multi-valued dependency directly; in other words, when the dependency base based on given multi-valued dependency is confirmed, we can get all multi-valued dependencies implicated logically by some attribute set.

Lemma 1: Let T be an XML tree that satisfied the given DTD D. p_h, r_j, q_i∈D,1≤h≤k,1≤i≤m,1≤j≤s. If XML tree T satisfies XFD: p_1, … , $p_k \rightarrow q_1$, …, q_m , T will satisfy XMVD: p_1, …, $p_k \rightarrow\rightarrow q_1$, …, q_m|r_1, …, r_s, where R={r_1, ..., r_s}∈D.

B.1. Algorithm Description

We note that if there is one dependency: XMVD P→→Q(XFD P→Q), and dependency set ∑ implicates logically this dependency, so Q is the union set of some paths among DEP(P) according to the definition 6 and 7. The algorithm for path dependency base is given before solving the membership on the condition of coexistence of XML functional dependency and XML multi-valued dependency:

First, extend all the XFDs among dependency set ∑ to XMVD, then we can get a transformation from ∑ to ∑′ which only includes XMVD; the initial value of BASIS

consists of all the single paths from path set P and Paths(D)-P. The initial value of change-flag is set as T, then execute "while" loop: initialize the value of change-flag as F, and then do the following operations for each MVD in $\sum'$: let P′ be the non-empty set, all of its elements from the intersection of BASIS and P, Q′=Q-P′. At the first time, W is empty. For the g_i in BASIS, add g_i into W if $g_i \subset Q'$. If Q′≠∅and Q′≠W, set change-flag as T, then add Q′ into BASIS by the definition 5. Executing the "while" loop repeatedly before Q′=∅ or Q′ has been the union set of some elements from BASIS.

Algorithm 1: DEP-BASE(Path Dependency Base).

INPUT: mixed set $\sum$ consisting of XFD and XMVD, Paths(D), P={p_1, … , p_k};

OUTPUT: DEP(P).

DEP-BASE($\sum$,Paths(D),P)

Begin

(1) To transform $\sum$ into $\sum'$including XMVD only.

(2) BASIS:={{ p_i}| $p_i \in P$}∪{Paths(D)-P};

(3)change-flag:= 'T';

(4)while change-flag do

(5) {change-flag:='F';

(6) for every MVD $P \to\to Q \in \sum'$ do

(7) {P′:=∪{R|R∈BASIS and R∩P≠∅};

(8) Q′:=Q-P′;

(9) W:=∅;

(10) for $g_i \in$ BASIS do

(11) {if $g_i \subset Q'$ then

(12) {W:=W∪g_i;}

(13) if Q′≠∅ and Q′≠W then do

(14) {change-flag:= 'T';

(15) BASIS:= MB{BASIS∪Q′};}}}

(16)return(BASIS)

End

C.1. Analysis of Algorithm

C.1.1. Termination of the Algorithm

After the initialization of step (2), step (3) and the step (4)'s loop operation, BASIS becomes a partition of Paths(D), because every subset after partitioning is non-empty, and the number of sets attained by partitioning the Paths(D) is at most the number of all the paths in Paths(D). After every time (except the last time) for executing step (4), the size of BASIS will always increase. We note that the size of BASIS is at most the number of all the Paths in Paths(D). Because the XMVD in $\sum'$ is finite, so that the "for" loop in step (6) is terminable. In conclusion, algorithm 1 is terminable.

C.1.2. Correctness of the Algorithm

Correctness of the algorithm is to prove that BASIS is equal to DEP(P) when the algorithm terminates. The proof process includes two parts. The first part is to prove BASIS⊂DEP(P), and the second part is to prove DEP(P) ⊂BASIS.

First, to prove BASIS⊂DEP(P). When the algorithm terminates, BASIS={{P_1},...,{P_m},{p_1},...,{p_k}},where {P_1},...,{P_m} is a partition of Paths(D)-P, and single paths of P form a set only including one path. According to the reflexivity inference rules, we note that $\sum$ implicates logically XMVD $P \to\to p_i$. The induction method is used to prove $P \to\to P_j \in \sum^+$, for each {P_j}∈BASIS, where 1≤j≤m as following.

According to the reflexivity inference rules, we have $P \to\to Q \in \sum+$, so that $P \to\to$VPaths(D)-P-Q$\in \sum+$ holds. After executing the step(2), all elements in BASIS depend on P by multi-value, that is to say $P \to\to Pj \in \sum+$ hold , for each {Pj}∈BASIS, where 1≤j≤m.

After t(t≥0) times loop, we assume $P \to\to P_j \in \sum^+$ hold, for each {P_j}∈BASIS, where 1≤j≤m. For the $(t+1)^{th}$ loop: if $P \to\to Q \in \sum$ is an XMVD during the $(t+1)^{th}$ loop, BASIS value will change(if BASIS values don't change, it will be the final value). Let P′ be the non-empty set whose elements are from the intersection of BASIS and P. $P' \to\to Q$ holds in accordance with the augmentation inference rules, because BASIS is a partition of Paths(D) , $P \subset P'$. $P \to\to P_j \in \sum^+$ holds, for each {P_j}∈BASIS, where 1≤j≤m before the $(t+1)^{th}$ loop, then $P \to\to P'$ holds under the union rules. At the same time, we note that $P \to\to Q\text{-}P' \in \sum^+$ holds according to transitivity rules. Q′=Q-P′ is set. If Q′ is empty or Q′ is a union set of some elements among BASIS, Q′ is added into BASIS. Then solve the minimal base of BASIS. BASIS value doesn't change according to definition 6. We need to make a corresponding modification for BASIS when Q′ is not empty and Q′ isn't a union set of some elements among BASIS. Now, according to the difference rules, $P \to\to P_j$ holds after modifying, for each {P_j}∈BASIS. In conclusion, $P \to\to P_j \in \sum^+$holds, for every {P_j}∈BASIS, 1≤j≤m during the $(t+1)^{th}$ loop, so that BASIS⊂DEP(P) holds in accordance with the definition 6.

Second, to prove DEP(P)⊂BASIS. Constructing a XML tree T that conforms to DTD D, where

(i) Each XMVD on T from $\sum$ is legitimate;

(ii) The necessary and sufficient condition of one XMVD $P \to\to Q$ on T holds is that Q′ is a union set of some elements among BASIS.

Constructing an XML document tree T: there are 2^m tree tuples, and every tuple in $Tuple_T(D)$[16] has a group of corresponding sequence {a_1,...,a_m}, where $a_i \in [0,1]$. {P_1},...,{P_m} is a partition of Paths(D)-P. The corresponding values of all paths in P^+ for every tuple in $Tuple_T(D)$ are 1 ,and the value of every path in P_i is a_i .

Properties of the XML document tree T:

Property 1: Each XMVD is valid when on the right of T is P_i.

Property 2: Some XMVD holds on T when on the right of P_i is a non-empty subset iff the left of XMVD has intersection with P_i.

Each multi-valued dependency on the T will be proved correct as following:

$P \to\to Q \in \Sigma$ holds, and P′ is the set whose elements attained from the intersection of BASIS and P. We note that Q-P′ is an empty set or the union set of some elements among BASIS in accordance with termination of the algorithm, so that $P \to\to Q\text{-}P'$ holds on T. According to Property 2, $P \to\to Q \cap P'$ holds on T, then $W \to\to R$ holds on T based on union rules among

deriving.

We will prove P→→Q hold on T iff Q is the union set of some elements in BASIS.

We note from definition 5: if Q is the union set of some elements in BASIS, P→→Q holds on T, that is to say the sufficiency condition holds.

If P→→Q holds on T, P→→Q∩P_i holds on T, for each i, 1≤i≤m. Because the intersection of P and P_i is empty, Q∩P_i is empty or is P_i based on Property 1 and Property 2. Therefore, Q is the union set of some elements among BASIS.

In conclusion, BASIS=DEP(P), algorithm 1 is correct.

C.1.3. Time Complexity of the Algorithm

"m" expresses the total number of paths from Paths(D), and "n" expresses the number of dependencies from ∑.

Because XFD is a special case of XMVD, so we note that every XFD can be extended to the corresponding XMVD.

Partition the path set into g subsets, using matrix g×m for expressing. Each row with m digits expresses a set. Each column of the matrix has only one 1, other positions are 0. The $(ij)^{th}$ position of matrix is 1 iff path p_j belongs to the i^{th} set. Obviously, p_j and p_i are in the same path set iff the i^{th} column and the j^{th} column of the matrix are the same; in other words, when the set is on the h^{th} row, the values of h_i and h_j both are 1, then other positions of the i^{th} column and the j^{th} column of the matrix are 0. Therefore, to find the minimal path dependency base, we partition the same columns into one set whose row values are 1 in the same columns. Arbitrary two of all the columns are compared for g times. We note that the time cost of which solving the minimal path dependency base is O(g×m).

This algorithm initializes the path dependency base at first. Every single paths from path P set form a set respectively and all sets constitute one set, then path set Paths(D)-P forms one set. Since constituting two rows of the matrix after initialization, we get the time cost of the operation is O(m).

The loop in step (4) executes at most m times. In every loop, for the given XMVD P→→Q, our goal is to find the union set of all the elements attained from the intersection of BASIS and Q. Since the size of BASIS is at most m, the time cost of the operation is $O(m^2)$. The time complexity of the "for" loop is $O(m^2)$ of step (10). In step (4), it is possible to examine whether n dependencies change the BASIS value or not. Therefore, the loop continuous until the BASIS value change, and it need to execute $O(n\times m^2)$ times. The time cost of solving minimal path dependency base is $O(m^2)$ and the time cost of executing step(4) once is $O(n\times m^2)$. After executing step (4) completely, the total time cost is $O(n\times m^3)$.

In conclusion, the total time complexity of this algorithm is $O(n\times m^3)$.

A.2. Membership Algorithm

The membership is to solve whether some dependency implicated logically by dependency set or not. On the basis of DEP-BASE algorithm, we provide the membership algorithm in the following.

B.2. Algorithm Description

First, we get the path dependency base, and the initial value of Q′ is set as empty. If the dependency is a multi-valued dependency, the following operations is executed: if the element in DEP(P) belongs to the right path set of this multi-valued dependency, then add this element into Q′. Whether Q′=Q hold or not examined. If holds, this dependency is implicated logically by Σ. If the dependency is functional dependency, and need to examine whether the right path set Q∈P^+ of this dependency hold or not; if holds, this dependency is implicated logically by Σ.

Algorithm 2: DEP-MEMBERSHIP(XFD and XMVD Membership).

INPUT: dependency set Σ, path set Paths(D), a dependency g:P→→Q(P→Q).

OUTPUT: if this dependency is implicated logically by Σ, the output is True; otherwise, the output is False.

```
DEP-MEMBERSHIP(∑, g)
Begin
(1)DEP(P):=DEP-BASE(Σ,Paths(D), P);
(2)Q′=∅;
(3)if P→→Q
(4)    for every Pi in DEP(P)do
(5)        if Pi⊆Q then
(6)            Q′:= Q′∪ Pi;
(7)if (P→→Q and Q′=Q) or (P→Q and Q∈P+) then
(8)    return(True);
(9)else
(10)    return(False);
End
```

C.2. Analysis of Algorithm

C.2.1. Termination Of the Algorithm

The step (1) of algorithm 2 calls the algorithm 1 to get the DEP(P), so the step(1) is terminable by algorithm 1. Because the number of elements from DEP(P) is finite and at most the number of paths in Paths(D), the "for" loop is terminable in step (4) of algorithm 2. Therefore, the algorithm 2 is terminable.

C.2.2. Correctness of the Algorithm

If the dependency needed to be examined is XMVD, it can be transformed from judging whether ∑ implicates logically the XMVD P→→Q to judging whether the Q is the union set of some elements from DEP(P) by definition 6. The step (1) of algorithm 2 calls the algorithm 1 to attain the DEP(P), and we note that the step (1) is correct. The "for" loop in step(4) is to examine one by one whether every set belongs to Q, then Q′ is used for expressing the sets whose elements in DEP(P) also in Q. Finally, compare Q′ with Q. If they are same, Q is the union set of some elements from DEP(P), now return True, otherwise , return False. If the dependency needed to be examined is XFD, we note that algorithm 2 is correct by the definition of closure.

C.2.3. Time Complexity of the Algorithm

The number of paths from Paths(D) is expressed as m, and n expresses the number of dependencies from Σ. The time complexity of the step (1) in this algorithm is $O(n \times m^3)$. The "for" loop in step (4) mainly make a comparing among sets, and the number of elements from DEP(P) is at most m, so its time complexity is $O(m^2)$. Therefore, the total time cost of algorithm 2 is $O(n \times m^3)$.

IV. MINIMAL DEPENDENCY SET ALGORITHM

Definition 8: Let Σ be the set of XFD and XMVD, if Σ_{min} is a minimal dependency set of Σ, the Σ_{min} should meet the following conditions:

(i) There is no redundancy path in Σ_{min}, that is to say there is no $P \rightarrow\rightarrow Q$(or $P \rightarrow Q$) that makes

$\Sigma_{min}=\Sigma_{min}-\{P \rightarrow\rightarrow Q\}$(or $\Sigma_{min}=\Sigma_{min}-\{ P \rightarrow Q \}$).

(ii) There is no redundancy on the left of every dependency, namely for every XMVD $P \rightarrow\rightarrow Q$(or XFD $P \rightarrow Q) \in \Sigma_{min}$, there not exist XMVD $P' \rightarrow\rightarrow Q$ (or XFD $P' \rightarrow Q) \in \Sigma^+$ that makes $P' \subset P$ hold.

(iii) There is no redundancy on the right of every dependency, and the right of every dependency is single path, namely for every XMVD $P \rightarrow\rightarrow Q$(or XFD $P \rightarrow Q) \in \Sigma_{min}$, there not exist XMVD $P \rightarrow\rightarrow Q'$(or XFD $P \rightarrow Q') \in \Sigma^+$ that makes $\varnothing \subset Q' \subset Q$ hold.

A. Algorithm Description

Data dependency plays an important role in the normalization design of database [18]. Designing an XML database that can avoid data redundancy and abnormal operation is an important research subject in the XML field [19].

Some data dependency can be implicated logically by other data dependencies. Removing the redundancy dependency and redundancy path is to simplify the given dependency set and make sure it is simple during the analysis and computation, on the condition that the data dependency set closure doesn't change. This problem is about minimal dependency set.

First, we set change-flag as T, then execute "while" loop, let change-flag be F, and initialize the value of Σ as Σ'. To perform the following operations for every d in Σ: judging whether the dependency is redundancy using DEP-MEMBERSHIP (algorithm 2) at first. If it is redundant, then remove it from Σ. If it is not redundant then to examine whether the left and right of this dependency exist the redundancy paths respectively, and if there is a redundancy path , it will be removed. After executing the operations, we need to examine whether $\Sigma \neq \Sigma'$ hold or not, if it holds, Σ is not the minimal dependency set. Then set the change-flag as T and re-execute the "while" loop. If $\Sigma=\Sigma'$, Σ is a minimal dependency set.

Algorithm3: DEP-MINIMIZE(Minimize Dependency).

INPUT: A set Σ including XFD and XMVD.

OUTPUT: the minimal dependency set Σ_{min} of Σ.

DEP-MINIMIZE(Σ)

Begin

(1)change-flag:='T';
(2)while change-flag do
(3)change-flag:= 'F';
(4)Σ':= Σ;
(5)for (every dependency d in Σ) do
(6) if DEP_MEMBERSHIP(Σ-d, d) then
(7) Σ:=Σ-d;
(8) if (exist redundancy path on the left of d)
(9) for every path p$\in$ P do
(10) if d is a XFD, then $P \rightarrow Q$ do
(11) if DEP_MEMBERSHIP(Σ,{P-p}$\rightarrow$Q) then
(12) Σ:=(Σ-{ $P \rightarrow Q$ }$\cup${P-p}$\rightarrow$Q));
(13) if d is a XMVD, then $P \rightarrow\rightarrow Q$ do
(14) if DEP_MEMBERSHIP(Σ,{P-p}$\rightarrow\rightarrow$Q) then
(15) Σ:=(Σ-{ $P \rightarrow\rightarrow Q$ }$\cup${P}-{p}$\rightarrow\rightarrow$Q));
(16) if (exist redundancy attribute on the right of d)
(17) for every path q$\in$ Q do
(18) if d is a XFD, then $P \rightarrow Q$ do
(19) Σ:=(Σ-{ $P \rightarrow Q$ })$\cup${$P \rightarrow q_1$, ..., $P \rightarrow q_m$};
(20) if d is a XMVD, then $P \rightarrow\rightarrow Q$ do
(21) if DEP_MEMBERSHIP(Σ,$P \rightarrow\rightarrow$ {Q}-{q}) then
(22) Σ:=(Σ-{ $P \rightarrow\rightarrow Q$ })$\cup${ $P \rightarrow\rightarrow Q'$, $P \rightarrow\rightarrow$ {Q}-{q}}
(23)if $\Sigma \neq \Sigma'$
(24) change-flag:= 'T';
(25)return(Σ).
End

B. Analysis of Algorithm

B.1. Termination of the Algorithm

The number of elements from preliminary Σ and from Σ^+ is finite. In addition, the size of Σ is at most the size of Σ^+ after executing the step (5). The size of Σ^+ is finite which results in the "for" loop in step (5) is terminable. The left and right paths of every dependency are finite. We note that the "for" loop is terminable in step (9) and step (17) of algorithm 3. In conclusion, the algorithm 3 is terminable.

B.2. Correctness of the Algorithm

Examine whether every dependency conforms to the definition of minimal dependency set or not in "for" loop of step (5) after executing the step (1)~step (4) of this algorithm. step (6) and step(7) of algorithm calls DEP_MEMBERSHIP algorithm to remove the redundancy dependency; step(8)~step(15) calls DEP_MEMBERSHIP algorithm to remove the left redundancy path; step(18) and step (19) transforms the right path of XFD to single path; step (8)~step(15) calls DEP_MEMBERSHIP algorithm to remove the right redundancy path for XMVD. We note that algorithm 3 is correct in that the dependency meets the definition of minimal dependency set.

B.3. Time Complexity of the Algorithm

"m" expresses the total number of paths from Paths(D), and "n" expresses the number of dependencies from Σ.

The time cost of step (6) is $O(n \times m^3)$ based on calling the DEP_MEMBERSHIP algorithm; the time cost of step (8)~step(15) is equal to the time cost of the "for" loop, and it is $O(n \times m^4)$. The time complexity of step(16)~step(22) also depends on the "for" loop whose time complexity is $O(n \times m^4)$. In addition, the number of dependencies $\sum$ is n. Therefore, the time complexity of executing the step(5) once is $O(n^2 \times m^4)$. In conclusion, the time complexity of algorithm 3 is $O(n^2 \times m^4)$.

V. CONCLUSION

The design of database schema is the first step of the database application. If the database schema is well designed, abnormal data dependencies, data redundancies and abnormal operations can be avoided. This paper studies the membership problem from the view of the condition of coexistence of XML functional dependency and XML multi-valued dependency. At the same time, we propose DEP-BASE algorithm, DEP-MEMBERSHIP algorithm and DEP-MINIMIZE algorithm on the membership problem, not only proving the termination and correctness of these algorithms, but also analyzing their time complexities. They laid a foundation for designing a normal form which level of normalization is higher than others.

ACKNOWLEDGMENT

This work was financially supported by Hebei Provincial Natural Science Foundation of China (F2012203087), National Natural Science Foundation of China (61272124) and National Natural Science Foundation of China (61073060).

REFERENCES

[1] Erik Naggam, "Standard Generalized Markup Language"[EB/OL].(1996-03-01)[2013-11-26] http://www.w3.org/MarkUp/SGML.
[2] Arash Termehchy, Marianne Winslett, "Using Structural Information in XML Keyword Search Effectively". ACM Transactions on Database Systems, 2011, vol.36, no.1, pp.4.
[3] Haiping Xu, Abhinay Reddyreddy, Daniel F. Fitch, "Defending Against XML-Based Attacks Using State-Based XML Firewall". Journal of Computers, 2011, vol. 6, no. 11, pp.2395-2407.
[4] Ren Li, Jianhua Luo, Dan Yang, Haibo Hu, Ling Chen, "A Scalable XSLT Processing Framework based on MapReduce". Journal of Computers, 2013, vol. 8, no. 9, pp.2175-2181.
[5] Shihan Yang, Jinzhao Wu, Anping He, Yunbo Rao, "Derivation of OWL Ontology from XML Documents by Formal Semantic Modeling". Journal of Computers, 2013, vol. 8, no. 2, pp. 372-379.
[6] Haiyan Huang, Ronghua Shi, Gaoshi Li, "The normalization of the algorithm on XML multi- valued dependency". Journal of hunan institute of science and technology, 2008, vol.29, no.12, pp. 126-129.
[7] Millist W.Vincent, JIXUE LIU, CHENGFEI LIU, "Strong Functional Dependencies and Their Application to Normal Forms in XML". ACM Transactions on Database Systems, 2004, vol.29, no.23, pp. 445-462.
[8] E. F. Codd, "Recent investigations in relational data base systems"[C]//Proceedings of IFIP Congress 74, Stockholm, Sweden, 1974, pp.1017-1021.
[9] Kamsuriah Ahmad, Ali Mamat, Hamidah lbrahim, Shahrul Azman Mohd Noah, "Defining Functional Dependency for XML". Journal of Information Systems, Research & Practices, 2008, vol.1, no.1, pp. 26-34.
[10] Xiangguo Zhao, Junchang Xin, Ende Zhang, "XML Functional Dependency and Schema Normalization"[C]//Proceedings of the 9th International Conference on Hybrid Intelligent Systems, Shenyang, China, 2009, pp.307-312.
[11] Marcelo Arenas, "Normalization Theory for XML". SIGMOD Record, 2006, vol.35, no.4, pp. 57-64.
[12] Tadeusz Pankowski, Tomasz Pilka, "Transformation of XML Data into XML Normal Form". Informatica (Slovenia). 2009, vol.33, no.4, pp.417-430.
[13] CATRIEL BEERI, "ON the Membership Problem for Functional and Multivalued Dependencies in Relational Database". ACM Transactions on Database Systems. 1980, vol.5, no.3, pp.241-259.
[14] Zhongping Zhang, "The logical implication algorithm research of XML multi-valued dependency". Computer Science, 2006, vol.33, no. 11(Supplement), pp.353-354.
[15] Wei Qiu, Lichen Zhang, "Study of Normalization Existing MVD in XML DTD". Computer Science, 2007, vol.34, no.2, pp. 149-185.
[16] Zhixiao Liu, "XML normalization research based on functional dependency and multi-valued dependency", yanshan university, qinhuangdao, MA, 2012.
[17] Millist W.Vincent, Jixue Liu, Chengfei Liu, Mukesh Mohania, "Mutivalued Dependencies and a 4NF for XML". CAiSE 2003, pp. 14-29.
[18] Jixue Liu, Jiuyong Li, Chengfei Liu, Yongfeng Chen., "Discover Dependencies from Data—A Review". IEEE Transactions on Knowledge and Data Engineering, 2012, vol. 24, no.2, pp.251-264.
[19] M. W. Vincent, J. Liu, M. Mohania, "The implication problem for 'closest node' functional dependencies in complete XML documents". Journal of Computer and System Sciences, 2012, vol.78, no. 4, pp. 1045-1098.

Zhongping Zhang, Male, Born in 1972, professor, Ph.D., post-doctoral, CCF Senior Member (E20-0006458S).His main research interests are the grid computing, data mining and semi-structured data etc. He has undertaken 1 project of provincial level and has participated 2 projects funded by national natural science foundation of China. He rewarded the provincial scientific and technological progress second-class Award. On the domestic and international academic conferences and journals, He published more than 80 papers, 15 of them were cited by EI.

Chunzhen Fang, Female, Born in 1987, Postgraduate student, the main research interest is the outlier detection in data mining.