Study on Paradigm and Its Normalization with Coexistence of XML Functional Dependence and Multivalues Dependence

Lijun Cao
Hebei Normal University of Science & Technology

Xiyin Liu, Jing Cao, Aiyong Liu, Zhongping Zhang
Hebei Normal University of Science & Technology, China
College of Information Science and Engineering Yanshan University, China

Abstract—This paper proposes the paradigm and the corresponding normalization algorithms with coexistence of eXtensible Markup Language (XML) functional dependence and XML values dependence. The concepts of redundancy and keys are firstly described in this paper, while the paradigm with coexistence of XML functional dependence and XML values dependence is defined later. Then, the non-redundancy judgment theorem is also also presented. Furthermore, the XML document tree normalization algorithm is introduced, while the capability to be terminated, validity, and computation complexity are analyzed and proved. Finally, the effectiveness of the algorithm is demonstrated by the experiments.

Index Terms—normalization; redundancy; paradigm; functional dependence; multi-value dependence; normalization rules

I. INTRODUCTION

While XML is widely used in more and more fields, a well-performed eXtensible Markup Language (XML) document with more Document Type Definitions (DTDs) is necessary to reduce the data redundancy. Model normalization theory is the theoretical basis to design a well-performed database model. Similar to relation databases, if the DTD design pattern is not suitable enough, abnormal data operation will take place. To fix this problem, normalization study is a good method to eliminate data redundancy and abnormal operation. Therefore, the paradigm theory is one of the hot topics in the research of database [1].

For the cases with coexistence of XML functional dependence and multivalues dependence, this paper proposes a paradigm and its normalization algorithm based on their logical implication and membership. After studying the relation database normalization theorem and characteristics of XML database, we also define 3XNF(XML Normal Form) and 4XNF, and further give normalization rules and a theorem on non-redundancy judgment. The normalization algorithm is finally designed while its effectiveness is then proved by the experiments.

II PARADIGM IN XML FUNCTIONAL DEPENDENCE

In relation databases, the usage of paradigm can reduce data redundancy and decrease the number of abnormal operations. The concept of paradigm is also utilized in XML. People can discuss whether a DTD is well-designed using a paradigm, and then apply normalization operation on the DTDs which do not meet the paradigm.

A. Effective Change and Redundancy

While XML is widely used nowadays, the design of a good model is becoming an important research topic [2]. Data redundancy not only wastes a lot of space for storage, but also increases the cost of data transfer and data operation. Therefore, a model with a good design pattern is needed to reduce data redundancy.

In relation databases, data redundancy means the repeated saving of the same data for many times. In a relation mode which satisfies dependence set $\Sigma$, if an attribute value can be obtained according to the dependence set as well as other attribute values, this attribute value is believed to be redundant. Particularly, there is a relationship instance based on the model, which satisfies the dependence set $\Sigma$. If an instance will not satisfy some dependencies in set $\Sigma$ after a value in this instance has been changed., the relation model is called to be redundant.

When borrowing the concept of redundancy theory in relation database, we present the definitions of effective change and redundancy as follows.

Definition 1 (Effective change) For a given DTD: D, T is a complete tree satisfying D. Let v be an arbitrary node in T. v is changed into v', where the identifier v' is different from other identifiers in T, and val(v) $\neq$ val(v'), a new tree T' is generated after this changing. If T' still satisfies D, then T is called as an effective change.

Definition 2 (Redundancy) For a given DTD: D, $\Sigma$ is a collection set of XFD and XMVD (XML Multi-valued Dependence) and $\Sigma'$ is another collection set. If $\Sigma'$ satisfies the same dependencies as $\Sigma$, then $\Sigma'$ is equivalent to $\Sigma$.

For the cases with coexistence of XML functional dependence and multivalues dependence, this paper proposes a paradigm and its normalization algorithm based on their logical implication and membership. After studying the relation database normalization theorem and characteristics of XML database, we also define 3XNF(XML Normal Form) and 4XNF, and further give normalization rules and a theorem on non-redundancy judgment. The normalization algorithm is finally designed while its effectiveness is then proved by the experiments.
Dependency), where the paths in XFD and XMVD are all collected in the set Paths(D). In a complete tree T satisfying D and \( \sum \), a node \( v \) is changed into \( v' \). If this changing will generate a new tree \( T' \), which violates \( \sum \), D is said to be redundant.

### B. The Third Paradigm in XML (3XNF)

In relation databases, BCNF is a paradigm with the highest normalization, which only considers the cases with functional dependence [3], while 3NF is a paradigm with the highest normalization to keep the functional dependence. Borrowing the related theory in relation data, we present the formal definition of the third paradigm in XML (3XNF) as below.

Definition 3 (The third paradigm in XML) For a DTD: D, \( \sum \) is the smallest dependance set in XFD. If for each abnormal XFD with \( p_1, \ldots, p_k \rightarrow q \), there exits XFD with \( p_1, \ldots, q \rightarrow pparent(q) \), where, last(q)e E2UA, then D is called to be the third paradigm in XML.

The definition of XML abnormal functional dependence which generates redundancy is given as follows.

Definition 4 (XML abnormal functional dependence) Assume there is a given DTD: D and a smallest dependance set containing XFD. For each abnormal XFD with \( p_1, \ldots, p_k \rightarrow q \), if this XFD does not satisfy the definition of XNF, i.e. \( p_1, \ldots, p_k \rightarrow q \) is established in D, but \( p_1, \ldots, p_k \rightarrow pparent(q) \) is not, where last(q)e E2UA, then XFD:p1, ..., q is called to be abnormal functional dependance.

Theorem 1: Assume in a given DTD D, T is a complete tree satisfying D, \( \sum \) is the XFD collection set, and \( T \) satisfies \( \sum \). If D satisfies the 3XNF model, then T is called to be with non-redundancy.

Proof: This theorem is proved by reduction to absurdity. That is: If T is redundant, then D does not satisfy 3XNF mode.

Assume there is redundancy in T, following the definition of 3XNF. We have: there exists a node \( v \) in T with \( \nu \in N(q) \), and abnormal XFD: \( p \rightarrow q \) is not established when \( v \) turns to \( v' \) with effective change.. According to definition 3.1, there are two different route instances \( q_1:v_1, \ldots, v_n(v_n=v) \) and \( q_2:w_1, \ldots, w_n(w_n=v) \) in Paths(q).

1. When \( pparent(q_1) \neq pparent(q_2) \), we have below conclusionse have conclusions as given below, because of \( \nu(v_n)=\nu(v_n) \) and \( pparent(q_1) \neq pparent(q_2) \). \( P \rightarrow pparent(q) \) is not established according to the definition of XFD, and thus it does not satisfy the definition of 3XNF.

2. When \( pparent(q_1) = pparent(q_2) \), below discussion can be established. Because \( last(q) \in E2UA, q_1 \) and \( q_2 \) are the same route according to definition 2.2. This conflicts with the assumption that \( q_1 \) and \( q_2 \) are different.

In conclusion, if D satisfies the 3XNF model, then T is called to be with non-redundancy.
(2) Normalization rule 2 Generation of element node

If the complex element vparent (last (q)) is embedded in a complex element last(P), attributes or elements are the simple attributes of vparent(last(q)) in the tree. Child elements are the simple attributes of vparent(last(q)) in the original tree. DTD D′ has been constructed as below figure to reduce the data redundancy.

Given a DTD D, Σ is a minimum function dependence set on XFD in D. Assume there exits an abnormal functional dependence p1, … ,pk→q in D, where (last(q))∈E2∪A. Let p1∩…∩pk=p, ancestor(q)∩p=s, if there exists last(pi)∈E2∪A, we move last(q) to make it be a child element of last(s) i.e.vparent(last(q))=last(s).

In the left route set P={p1, … ,pk}, let all routes in last(pi)∈E2∪A be added in set Vpath(P), i.e.Vpath(P)={pg, … ,ph}, let all routes in last(pi)∈E1 be added in set Epath(P), i.e.Epath(P)={pe, … ,pf}.

After DTD D has been normalized, paradigm can be represented as: D′ =(E′ ,A′ ,P′ ,R′ ,r), where E′ =E∪{lab(vnew)}; A′ =A; P′ (last(s))=P(last(s))∪{lab(vnew)}; P′ (last(vnew))={last(pg), … ,last(pf),last(q)}; P′(vparent(last(q)))=P(vparent(last(q)))-{last(q)}; other definitions of P′ are same to that of P. R′(lab(vnew))={vi|vi=last(pi)∈A∧last(pi)∈P (last(vnew))}; R′(vparent(last(q)))=R(vparent(last(q)))-{last(q)|last(q)∈A}; other definition of P′ and R′ are same with that in D. Let the functional dependence set in D′ be Σ′, this kind of functional dependence in Σ is turned into the formation of s. vnew. last(pg), … ,s. vnew. last(ph)→s. vnew. last(q), i.e.Σ′ =Σ−p1, … ,pk−q∪s. vnew. last(pg), … ,s. vnew. last(ph)→s. vnew. last(q), k≥1.

First of all, D′ is initialized as D, and Σ′ is initialized as Σ. We first need to find out whether D′ satisfies 3XNF. If yes, it directly returns D′; if not, it indicates that there exists abnormal functional dependence, and operation given below should be done. If the right tail node of the abnormal functional dependence is a simple element or attribute node, wherein the left path of this dependence is with public prefix of p and we find that Σ containing logically p1, … ,pk→p according to the membership algorithm, then normalization can be done with moving elements and following normalization rules. Otherwise, normalization will be done by generating element node normalization rules.

Algorithm 1: Algorithm of converting DTD D into DTD D′, which satisfies 3XNF (DTD-to-3XNF).
Input: DTD D and functional dependence set Σ;
Output: DTD D′ satisfying 3XNF
DTD-to-3XNF(D, Σ)
Begin
(1) Initialization:D′:=D, Σ′:=Σ;
(2) If D′ satisfies 3XNF, turn to (6);
(3) While each abnormal XFD∈F do
(4) Case1: if p1, … ,pk→q∈F and p⊂q, where last(q)∈E2∪A,p1∩…∩pk=p and DEP_MEMBERSHIP(Σ,p1, … ,pk→p) then use normalization 1;

(5) Case 2: if \( p_1, \ldots, p_k \rightarrow q \) and \( \text{last}(p_i) \in E_2 \cup A \), where \( \text{last}(q) \in E_2 \cup A \), \( q = \text{ancestor}(q) \cap p \) and \( \text{DEP\_MEMBERSHIP}(\Sigma, p_1, \ldots, p_k \rightarrow p) \) then use normalization 2;
(6) Return \((D')\).

End

The analysis of algorithm 1 is as below.

(1) The correctness of the algorithm

The algorithm presents the corresponding solutions according to the different situations. It makes XFD satisfying its definition using not only the method of moving the attributes and simple element, but also the method of creating and copying element nodes. If the definition of XNF is satisfied, apparently the correctness of the algorithm is established.

(2) The capability to be terminated

This capability is determined by the number of the abnormal functional dependence because this number is limited, and can be terminated. Therefore, the whole algorithm can be terminated.

(3) The time complexity of the algorithm

Let \( B \) be the number of abnormal XFD in \( \Sigma \). The algorithm's time complexity depends on the time complexity of \( \text{DEP\_MEMBERSHIP} \) algorithm and the number of execution of \( \text{DEP\_MEMBERSHIP} \), while the \( \text{DEP\_MEMBERSHIP} \) algorithm's time complexity is \( O((\text{mn})^3) \). \( \text{DEP\_MEMBERSHIP} \) execution times are determined by the number of abnormal XFD b. Therefore, the algorithm's time complexity is \( O((\text{mn})^3) \).

III PARADIGM WITH COEXISTENCE OF XML FUNCTIONAL DEPENDENCE AND XML VALUES DEPENDENCE

A The Fourth Paradigm

In relation databases, the fourth paradigm is with the highest paradigm with coexistence of XML functional dependence and XML values dependence [4]. Borrowing the related theories in relation databases, the formal definition of the fourth paradigm (4XNF) is given as below.

Key is an important part in semantic constraints [5]. Key will be set as an important data integrity constraint [6] in the definition of XML mode. The formal definition of key is given below.

Definition 5 (XML key) Given DTD \( D \), \( T \) is a complete tree satisfying \( D \), \( p \in \text{Paths}(D) \). If two arbitrary different nodes \( v_1, v_2 \), \( v_1 \) and \( v_2 \) belong to \( N(p) \), satisfies \( \text{val}(v_1) \neq \text{val}(v_2) \), \( p \) is called as key, and recorded as key(p).

Definition 6 (XML the fourth paradigm) Given DTD \( D \), \( T \) is a complete tree satisfying \( D \), \( \Sigma \) is a XMVD set. If each XMVD in \( \Sigma \), \( p_1, \ldots, p_k \rightarrow q_1, \ldots, qm|\rightarrow|, \ldots, rs \), satisfies any one of below conditions, then DTD \( D \) is the fourth paradigm of XML.

1. \( \text{key}(qi), \text{key}(rqj) \);
2. \( \text{key}(ph) \) and \( qi \cap rqj \neq ph \);
3. \( \text{key}(ph) \) and \( qi \cap rqj \) are the strict prefix of ph;
4. \( qi \cap rqj \) are root.

(5) \( \text{key}(ph) \) and \( rqj \cap qj \neq ph \);
where, \( ph = p_1 \cap \ldots \cap p \cap q_1 \cap \ldots \cap qf \cap rqj = r_1 \cap \ldots \cap re \).

Lemma 1: Given DTD \( D \), \( \Sigma \) is a XMVD set, and all routes in it are in \( \text{Paths}(D) \). Assume that \( \Sigma \) causes redundancy, then there exists a multi-value dependence \( p_1, \ldots, p_k \rightarrow q_1, \ldots, qm|\rightarrow|, \ldots, rs \), satisfying:

1. \( \forall \text{val}(q) \) or \( \forall \text{val}(rj) \), \( 1 \leq i \leq m, 1 \leq n \leq s \);
2. There exists two different instances \( v_1, v_2, \ldots, v_{t-1}, v_t \) \( v_{t-1}, w \in \text{Path}(q), 1 \leq t \), which makes \( \text{val}(v_t) = \text{val}(w) \);
3. There exists two instances \( z_1, z_2, z_1 \in \text{Nodes}(xij, rj), z_2 \in \text{Nodes}(yij, rj) \), making \( \text{val}(z_1) \neq \text{val}(z_2) \);
4. There exists two instances \( z_3, z_4, z_3 \in \text{Nodes}(xij, ph), z_4 \in \text{Nodes}(yij, ph) \), making \( \text{val}(z_3) = \text{val}(z_4), 1 \leq n \leq s \).

Proof: (1) we first prove that if \( \Sigma \) causes redundancy, then there exists a multi-value dependence XMVD \( p_1, \ldots, p_k \rightarrow q_1, \ldots, qm|\rightarrow|, \ldots, rs \), making it satisfy condition (1) in lemma 1.

If the situation is not established, there are two possibilities. The first one is as below. If \( \forall \text{val}(q), \forall \text{val}(rj) \), \( v \neq ph \), because \( v \) does not belong to any node in a route of \( \Sigma \), when node \( v \) turns to \( v \), the generated new tree \( T \) does not violate \( \Sigma \), and conflicts with \( \Sigma \) to cause redundancy contradiction. The other possibility assumes \( \forall \text{val}(ph) \), in which conditions, if \( \text{lab}(v) \neq \text{val}(ph) \), according definition 1, the changed \( v \) does not belong to any node in \( T \). Same as \( v \) in \( T \), \( T \) satisfies \( \Sigma \) according to definition 3. Therefore, \( \Sigma \) cannot cause redundancy or conflict with \( \Sigma \). If \( \text{lab}(v) \neq \text{val}(ph) \), after effective changes, node \( v \) makes \( \text{val}(v') \) not appear in \( T \). In the new tree \( T \) after the changes, XMVD: \( p_1, \ldots, p_k \rightarrow q_1, \ldots, qm|\rightarrow|, \ldots, rs \) still holds, and causes redundancy contradiction with \( \Sigma \). Thus, condition (1) in lemma 1 is established.

(2) Next we want to prove that if \( \Sigma \) causes redundancy and \( \forall \text{val}(q) \), i.e. satisfying condition (1) in lemma 1, there exists a XMVD \( p_1, \ldots, p_k \rightarrow q_1, \ldots, qm|\rightarrow|, \ldots, rs \), making itself satisfy condition (2) in lemma 1.

Assume this case is not established, then there are two conditions. The first one is: if there is only one route instance in \( \text{Paths}(q) \), \( \text{Paths}(q) \) in the new generated \( T' \) also has only one route instance after node \( v \), experiences an effective change, then XMVD \( p_1, \ldots, p_k \rightarrow q_1, \ldots, qm|\rightarrow|, \ldots, rs \) is known to be established in \( T' \), and conflicts with \( \Sigma \); if there are more than one routes in \( \text{Paths}(q) \) with different tail node values, let \( v_1, v_2, \ldots, v_{t-1}, w \in \text{Paths}(q), v \neq v' \). If in \( \text{Paths}(q) \), there exist route instances \( w_1, w_2, \ldots, w_{t-1}, w_t \), where \( v \neq w_t \), and makes conditions (2) and (3) in definition 7 satisfied, then according to definition 7, it’s known that \( v_1, v_2, \ldots, v_t \) are same to \( v_1, v_2, \ldots, v_{t-1}, w_t \). For the same reason, \( w_1, w_2, \ldots, w_{t-1}, w_t \) are same to \( w_1, w_2, \ldots, w_{t-1}, w_t \) according to condition (5) in definition 7, then after effective changes in node \( v \) and

© 2014 ACADEMY PUBLISHER
generation of new tree T', XMVD:p1,...,pk→q1,...,qm|r1,...,rs is still established, and conflicts with Σ. The other conditions is: if in Paths(qi) there is no route instances v1,w2, ... ,wt-1,wt, which satisfy condition (2) and (3) in definition 7, then it's known that after effective changes in node v and generation of new tree T', XMVD:p1,...,pk→q1,...,qm|r1,...,rs is still established, and conflicts with Σ. Therefore, condition (2) in lemma 1 is established.

(3) Then we prove that if Σ causes redundancy, there exists a multi-value dependence XMVD p1,...,pk→q1,...,qm|r1,...,rs, which satisfy condition (3) in lemma 1.

Assume this case is not established. Then there do not exist two nodes z1, z2, where z1∈Nodes(xij,rj), z2∈Nodes(yij,rj), which make val(z1)≠val(z2). It's known that route rj only has one route instance. From definition 7, XMVD:p1,...,pk→q1,...,qm|r1,...,rs still holds after effective changes in node v and generation of new tree T', XMVD:p1,...,pk→q1,...,qm|r1,...,rs still holds, and it conflicts with Σ. Thus condition (3) in lemma 1 is established.

(4) Finally, we prove that if Σ causes redundancy, there exists a multi-value dependence XMVD p1,...,pk→q1,...,qm|r1,...,rs, which satisfies condition (4) in lemma 1.

Assume this case is not established. Then there do not exist two nodes z3, z4, where z3∈Nodes(xijh,ph), z4∈Nodes(yijh,ph), which make val(z3)=val(z4), 1≤h≤k. It's known that the tail nodes of the instances in route ph are different. From definition 7, after effective changes in node v and generation of new tree T' XMVD:p1,...,pk→q1,...,qm|r1,...,rs still holds, and conflicts with Σ. Therefore, condition (4) of lemma 1 is established.

Due to the symmetry property, the cases when v∈N(rj) keeps the same conclusions as above. Here xij,yij,xijh,yijh,x'ij,y'ij,x'ijh,y'ijh holds the same meaning with that in definition3.

The non-redundancy judgment theorem can be obtained by following definition 6 and lemma 1.

Theorem 2: Given DTD D,T is a tree satisfying D, Σ is a XMVD set, T satisfies Σ. If D is with 4XNF mode, then Σ has no redundancy in T.

Proof: Statements given below prove that the cases of v∈N(rj) hold the same results with that in v∈N(qi) due to the symmetry property.

(1) Assume that condition (1) in definition 6 holds. That is when qi,rj are both keys, Σ causes redundancy.

According to lemma 1, after node v's effective changes into v', there exists v1∈N(qi),v2∈N(qi), and val(v1)=val(v2). Thus it's known that it conflicts with qi being the key, as well as the above assumption.

(2) Assume that condition (2) in definition 6 holds. That is when ph is a key and qi∩rj = ph, Σ causes redundancy.

According to lemma 1, if Σ causes redundancy, that is there exists a complete tree T satisfying D and Σ, if there is a node v in T, after effective changes into v', a new tree is generated, which leads that XMVD p1,...,pk→q1,...,qm|r1,...,rs does not hold, as well as v∈N(qi). Because XMVD p1,...,pk→q1,...,qm|r1,...,rs is not established in T', then Paths(qi) in T' keep two route instances v1.v2, ... .vt-1,vt (vt=v') and w1.w2, ... ,wt-1,wt, satisfying condition (1),(2),(3), but not (4) or (5) in definition 7. Because ph is a key, i.e. the tail node values in instance ph are different, from definition (3) in definition 7, i.e. val(z3)=val(z4), as well as val(z3)=val(z4) in (5), it's known that xijh=yijh; from condition (4) in definition 7, i.e. val(z3)=val(z3), i.e. val(z3)=val(z4) in (5), it's known that xijh=yijh, yijh,y'ijh. Because qi∩rj=ph, then we have N(qi∩rj)=N(qi∩rj∩ph)=N(ph), and xijh=yijh,x'ijh=y'ijh=xij=xij=x'ij=x'ij. Therefore, if T' does not satisfy XMVD p1,...,pk→q1,...,qm|r1,...,rs, then T does not satisfy XMVD p1,...,pk→q1,...,qm|r1,...,rs, which conflicts with the assumption.

(3) Assume that condition (3) in definition 6 holds.

That is when ph is a key and ph is a strict prefix, Σ causes redundancy.

According to lemma 1, if Σ causes redundancy, that is there exists a complete tree T satisfying D and Σ, if there is a node v in T, after effective changes into v', a new tree T' is generated, which leads that XMVD p1,...,pk→q1,...,qm|r1,...,rs does not hold, as well as v∈N(qi). Because XMVD p1,...,pk→q1,...,qm|r1,...,rs is not established in T', then Paths(qi) in T' keep two route instances v1.v2, ... .vt-1,vt (vt=v') and w1.w2, ... ,wt-1,wt, satisfying condition (1),(2),(3), but not (4) or (5) in definition 7. Because ph is a key, i.e. the tail node values in instance ph are different. Further because qi∩rj is the strict prefix of ph, then we have qi∩rj=qi∩rj∩ph. From condition (2),(3) in definition 7, it's known that xij=yij=x'ij=y'ijh. From condition (4),(5) in definition 7, it's known that xij=x'ijh,yijh=x'ijh,y'ijh. Therefore, if T' does not satisfy XMVD p1,...,pk→q1,...,qm|r1,...,rs, then T does not satisfy XMVD p1,...,pk→q1,...,qm|r1,...,rs, which conflicts with the assumption.

Assume that condition (4) in definition 6 holds. That is when qi∩rj=ph, Σ causes redundancy.

According to lemma 1, if Σ causes redundancy, that is means that there exists a complete tree T satisfying D and Σ, if there is a node v in T, after effective changes into v', a new tree T' is generated, which leads that XMVD p1,...,pk→q1,...,qm|r1,...,rs does not hold, as well as v∈N(qi). Because XMVD p1,...,pk→q1,...,qm|r1,...,rs is not established in T', then Paths(qi) in T' keep two route instances v1.v2, ... .vt-1,vt (vt=v') and w1.w2, ... ,wt-1,wt, satisfying condition (1),(2),(3), but not (4) or (5) in definition 7. Because qi∩rj=ph, then xij=yij,x'ijh=x'ijh=x'ijh,y'ijh, and they are all root nodes. Following consition (3), Σ can be proved to
cause no redundancy, which conflicts with the assumption.

(5) Assume that condition (5) in definition 6 holds, i.e. key(p) and rj\sqcap qi or rj\sqcap ph, \Sigma causes redundancy. Because ph is a key, i.e. the tail node values are different. Then from condition (3) in definition 7, it’s known that val(z)=val(z) and xij\sqcap yij From val(z)=val(z) in (4) and val(z)=val(z) in (5), it’s known that xij\sqcap yij\sqcap xij\sqcap yij. When qi\sqcap rj\sqcap qi=rj, and there exists a node v after effective change to v, and generating a new tree T, which does not affect (tj). From (4) and (5) in definition 7, it’s known that XMVD p1, ..., pk \rightarrow \Sigma, r be the public prefix of all routes in Q, r be the set in Q is represented as \{Q1, ..., Qn\}, and all responding instance set R be the public prefix of all routes in P, r be the set in R is represented as \{R1, ..., Rn\}, and all responding instance set Q be the public prefix of all routes in P, r be the set in Q is represented as \{Q1, ..., Qf\}, and all responding instance set R be the public prefix of all routes in P, r be the set in R is represented as \{R1, ..., Re\}. Therefore, \Sigma does not cause any redundancy, which conflicts with the assumption.

where xij, yij, xij, yij, xij, yij have the same meaning to that in definition 3.

B Design and Algorithm of XML Multi-value Dependence Normalization

Algorithm of XML multi-value dependence normalization is as follows.

Assume that complete XML document tree T is a XMLMD satisfying 3XNF and minimum dependence set \Sigma. P\rightarrow Q|R is assumed to be XMVD in \Sigma. Let \{P1, ..., Pn\} represent instance set of all P in T satisfying P\rightarrow Q|R. All responding instance set in Q is represented as \{Q1, ..., Qm\}, and all responding instance set in R is represented as \{R1, ..., Rn\}.

First of all, for P\rightarrow Q|R, let p be the left path public prefix, q be the public prefix of all routes in Q, r be the public prefix of all routes in R. For route instance set Phe Paths(P), where ph, qh, rh, are the public prefixes of all route instance sets Ph, Qh, Rh. If the tail node of ph is a complex element node, then the longer route between qh and rh is chose, which makes the parent node of the route’s tail node be last(ph); if the tail node of ph is a simple element node or attribute node, then the longer route between qh and rh is chose, which makes the parent node of the route’s tail node be a brother node of last(ph).

For route instance Pt in P of different instance Ph, when Pt=Ph, we first find out whether last(qh) and last(qt) have the same value. If not, and the value of last(qt) does not equal to the tail node of any route, which have parent(qh) as their parent node, then modify route qt and make it be the brother node of last(qh); when last(qh) has same value as last(qt), then delete route qt. For the same reason, find out whether the value of last(qh) equals to that of last(rt), if not, the value of last(rt) does not equal to the tail node value of any route with parent node being parent(rt) in T.\rightarrow, then modify route rt to be brother node of last(rt); if last(rt) has the same value as last(rt), then delete route rt. At last, delete route instance set Pt. Above operations make p be key route.

After above operations, condition (2) and (3) in definition of 4XNF are satisfied.

Algorithm 2: the algorithm to make DTD D be changed into D', which satisfies 4XFN.

Input: the minimum dependence set \Sigma of XFD and XMVD, the complete tree T satisfying \Sigma.

Output: XML document tree T' satisfying XMVD paradigm

DTD-to-4XNF(\Sigma, T)

Begin

(1) T:=T, \Sigma:=\Sigma;
(2) If T satisfies 4XNF, then turn to (4);
(3) For each XMVD: P\rightarrow Q|R do
(4) Paths(P):=\{P1, ..., Pn\};
(5) P:=\{p1, ..., pk\};
(6) p=p1 \sqcap ... \sqcap pk;
(7) Paths(Q):=\{Q1, ..., Qf\};
(8) Q:=\{q1, ..., qf\};
(9) q=q1 \sqcap ... \sqcap qf;
(10) Paths(R):=\{R1, ..., Re\};
(11) R:=\{r1, ..., re\};
(12) r=r1 \sqcap ... \sqcap re;
(13) For each Phe Paths(P) do
(14) if any condition in definition of 4XNF is not satisfied, then
(15) if last(ph) \sqcap E1
(16) if length(qh) \sqcap length(rh) then
(17) \{vparent(vparent(last(rh)))=last(ph);
(18) Modify rh in T' to modified rh, and modify rh in \Sigma' to modified rh;\}
(19) else
(20) \{vparent(vparent(last(qh)))=last(ph); Modify qh in T' to modified qh, and modify qh in \Sigma' to modified qh;\}
(20) else
(21) if length(qh) \sqcap length(rh) then
(22) \{vparent(vparent(last(rh)))=vparent(last(ph));
(23) Modify rh in T' to modified rh, and modify rh in \Sigma' to modified rh;\}
(23) else
(24) \{vparent(vparent(last(qh)))=vparent(last(ph));
(25) Modify qh in T' to modified qh, and modify qh in \Sigma' to modified qh;\}
(26) For route instance Pt of P different from Ph do
(27) If (Pt=Ph) then
(28) if val(last(qh))=val(last(qt)) and val(last(qt)) is not equal with the tail node value of any route node with parent node being parent(qh) then
(29) \{x:=last(qt);
(30) vparent(x):=vparent(last(qh));
(31) Modify qt in \Sigma' to modified qt;\}
(32) else if val(last(qh))=val(last(qt)) and vparent(last(qh))\neq vparent(last(qt)) then
(33) Paths(Q):=Paths(Q)- qt;
(34) if val(last(rh))=val(last(rt)) and val(last(rt)) is not equal with the tail node value of any route node with parent node being parent(rt) then
(35) \{y:=last(rt);
(36) vparent(y):=vparent(last(rt)); Modify rt in \Sigma' to modified rt;\}
(37) else if val(last(rh))=val(last(rt)) and vparent(last(rh))\neq vparent(last(rt)) then
(38) Paths(R):=Paths(R)- rt;
(39) Paths(P):=Paths(P)- P;
(40) Return(T');

© 2014 ACADEMY PUBLISHER
Analysis of the algorithm is as follows.

1. The correctness of the algorithm

Initialization is firstly done to make $T'$ be $T$, $\Sigma'$ be $\Sigma$. For each multi-value dependence in $\Sigma$, $P \rightarrow \SigmaQR$, let $p$ be the public prefix of all route in $P$, $q$ be the public prefix of all route in $Q$, $r$ the public prefix of all route in $R$. Then do the following operations: firstly find out whether this multi-value dependence satisfies the conditions in definition of 4XFN, if not, find whether $\text{last}(p) \in E1$ is established, if it’s established, then judge length($q$) is length($rh$) is satisfied or not. If it is satisfied, move node in $T'$ to realize $\text{vparent}(\text{vparent}(\text{last}(rh))) = \text{last}(p)$, if it is not satisfied, move node in $T'$ to realize $\text{vparent}(\text{vparent}(\text{last}(rh))) = \text{last}(p)$. The aim of above operations is to make route $q$ be a child node of $p$. If $\text{last}(p) \in E1$ is not established, then it’s known that this multi-value dependency has only one route in its left, and the tail node is a simple element node or an attribute node, then find out whether length($q$) is length($rh$) is established or not. If yes, move node in $T'$ to realize $\text{vparent}(\text{vparent}(\text{last}(rh))) = \text{vparent}(\text{last}(p))$; if not, move node in $T'$ to realize $\text{vparent}(\text{vparent}(\text{last}(rh))) = \text{vparent}(\text{last}(p))$. The aim of above operations is to make route $q$ be a brother node of $p$.

The next step is to compare $Pt$ with $Ph$. If they are equivalent, then find out whether $\text{val}(\text{last}(qh)) \neq \text{val}(\text{last}(qt))$ is established or not. If yes, and $\text{val}(\text{last}(qt))$ does not equal to the tail node value of any route node with parent node being $\text{parent}(qh)$ in $T'$, then copy tail node of $q$ to be a brother node of tail node of $qh$. If $\text{val}(\text{last}(qh)) = \text{val}(\text{last}(qt))$ holds, then delete route $q$ in $T'$.

Moreover, find whether $\text{val}(\text{last}(lh)) \neq \text{val}(\text{last}(rt))$ is established or not. If yes, and $\text{val}(\text{last}(qt))$ does not equal to the tail node value of any route node with parent node being $\text{parent}(qh)$ in $T'$, then copy tail node of $rt$ to be a brother node of tail node of $rh$. If $\text{val}(\text{last}(rh)) = \text{val}(\text{last}(rt))$ holds, then delete route $rt$ in $T'$. Repeat above operations, and then combine $q$ and $r$ in the same $P$ of $T'$ to make routes satisfy $q$ and $r$ be all strong route. Then condition (1) in definition 6 is satisfied. For any two different routes $Pt$ and $Ph$ in $P$, if they are equivalent, delete route set $Pt$ in $T'$. All routes that satisfy $P$ in $T'$ by this algorithm are strong. Then condition in definition 6 is satisfied.

When the outermost of the cycle is utilized, each XMVD in $\Sigma'$ satisfies the conditions of the definition 6. Thus $T'$ obtained by this algorithm meets 4XNF.

2. The capability to be terminated

The algorithm is constructed by three level cycles. The outermost cycle is determined by the number of XMVD, while the two internal cycles are determined by whether complete XML tree is limited. Therefore, the whole algorithm can be terminated because the number of XMVD and XML complete tree are both limited.

3. The time complexity of the algorithm

The algorithm is constructed by three level cycles. The outermost cycle is determined by the number of XMVD, while the number of the loop of the second cycle is determined by the maximum route instance set. Assume that the maximum route instance set contains $d$ instances, ($d$ is a multiple of 4). At best, in the first operation of the second cycle, the innermost cycle operates for $d-1$ times. In the second operation of the second cycle, the innermost cycle operates for $d-5$ times. From this trend, in the last operation of the second cycle, the innermost cycle operates for $3$ times. The second cycle operates for $d/4$ times at best, it’s known that the total times is $m(d^2+2d)/8$. At least, for the similar computing process, the total operation times is $m(d-2)/d$. Therefore, the time complexity is $O(m\times d^2)$.

IV. EXPERIMENTAL ANALYSIS

A Experiment Settings

Experimental environment settings are as follows. All algorithm realizations in this experiment are as follows.

1. Hardware, CPU: Intel 2.20GHz, Memory: 2.00GB, Hard Disk: 250G.
2. Operation system: Microsoft Windows XP.

Data set settings are as given below. This experiment used combined data set Course and real data set reed to verify the correction of the paradigm of the algorithm.

B Experiment Result and Performance Analysis

According to the three kinds of DTDs: D1, D2, D3, given by this paper, and the corresponding document trees T1, T2, T3, effectiveness of the normalization algorithm is verified in storage and query efficiency.

Along with the gradual increase of nodes in original document trees T1, T2 are generated after normalization by algorithm 1, and then T3 is generated by algorithm 2. The response time needed for searching a particular node is shown as Figure 3. The experimental results indicate that XML document query response time which satisfies paradigm with higher normalization degree is much lower than that with lower normalization degree.

![a) The relationship between response time of teacher and the number of nodes in initial document](image-url)
b) The relationship between response time of student and the number of nodes in initial document

c) The relationship between response time of Tname and the number of nodes in initial document

d) The relationship between response time of Sname and the number of nodes in initial document

Along with the gradual increase of nodes in original document trees T1, T2 are generated after normalization by algorithm 1, and then T3 is generated by algorithm 2. The storage needed for the three document trees is as shown in Figure 4. The experimental results indicate that XML document storage which satisfies paradigm with higher normalization degree is much lower than that with lower normalization degree.

The real data base uses the XML data base of course information of Reed College. T2 is generated based on original document tree T1 after normalization by algorithm 1, and then T3 is generated by algorithm 2. The query response time of the three document trees is shown as Figure 5. The experimental results indicate that XML document query response time which satisfies paradigm with higher normalization degree is much lower than that with lower normalization degree.

T2 is generated based on T1 after normalization by algorithm 1, and then T3 is generated from T2 by algorithm 2. The storage needed of the three documents is as shown in Figure 6. The experimental results show that XML document storage which satisfies paradigm with higher normalization degree is much lower than that with lower normalization degree.

V CONCLUSIONS
This paper describes the redundancy and key in XML, and presents the formal definition of 3XNF and 4XNF. Normalization rules and non-redundancy judgement theorem are then proposed. Furthermore, normalization algorithm based on XML document is also given which is followed by corresponding analysis and proof with respect to not only the capability to be terminated, but also the correctness, as well as the computation complexity. Finally, it has been shown that the effectiveness of the algorithm can be demonstrated by the experiments.

ACKNOWLEDGEMENT

This work has been financially supported by the Project of Major Reform to the Network Engineering Programme in Hebei Normal University of Science and Technology, which was assigned by Hebei Education Department in 2012. This work has been also sponsored by Qinhuangdao science and technology research (201101A038/201101A185/2012021A133)

REFERENCES