XML 文件的多值相依性與第四正規化

林穎聰
國立中興大學
資訊科學系
s9156029@cs.nchu.edu.tw

廖宜恩
國立中興大學
資訊科學系
ieliao@nchu.edu.tw

摘要

可延伸標示語言(eXtensible Markup Language)在最近幾年來迅速成為網際網路上，資料表示以及資料交換的標準格式，目前全球已有許多公司行號及單位組織，已經採用 XML 為電子商務的解決方案。隨著企業間及公司間的交易愈來愈頻繁，而伴隨產生極大量的 XML 文件，此時為了儲存以及管理龐大的 XML 文件，有愈來愈多的應用程式直接採用原生型 XML 資料庫(Native XML database)儲存 XML 文件。

XML Normal Form(XNF)已被提出能有效的在 XML 文件中縮減資料重複性以及減少資料的更新異常，本篇論文的動機是延伸 XML 正規化(XNF)的概念以及關聯式資料庫(relational database,RDB)的多值相依性(Multi-Valued Dependency)，來定義一個屬於 XML 的多值相依性(XML Multi-Valued Dependency, XMVD)，並且利用 XMVD 的語意概念來定義出符合 XML 第四正規化(4XNF)的文件，最後提出有一個有效的演算法來將不是 XML 第四正規化的文件，在不影響資訊內容下正規化成更節省空間、降低資料重複、消除更新異常的 4XNF 文件。

關鍵詞： XML、DTD、功能相依性、多值相依性、正規化

一、前言

網路的時代已經來臨，現在我們在全球資訊網(World Wide Web)上瀏覽來自世界各地的數位資訊，主要就是透過 HTML (Hyper Text Markup Language)[10]這一種超文字標記語言。HTML 發展至今，已成為全球資訊網的基礎，它提供標準化的方法將資訊格式化，並經由網際網路傳送給所有的使用者。但是 HTML 在開發之初主要是被設計用來顯示資料，而不是用來處理資訊的內容與資訊的結構，今日的全球資訊網已經不是僅僅用來提供使用者瀏覽資料而已，它已發展到了不論是對於企業或者是對於個人，只要有關資料的互動、處理與傳遞等等，都可以透過全球資訊網(WWW)進行，而這樣的網路演變就是我們需要 XML (eXtensible Markup Language)[8]的原因。

可延伸標示語言(eXtensible Markup Language)規格是由『全球資訊網標準製定組織』(W3C)[9] 制定，在最近幾年來迅速成為網際網路上，資料的表示以及資料交換的標準格式[2]。雖然說大部分的 XML 文件都可以被視為是關聯式的資料，而將這些資料存放在 RDB 中，但卻有愈來愈多的應用程式為了追求效率而直接取用原生型 XML 資料庫(Native XML database)儲存 XML 文件。XML 雖然提供了彈性的語法，但是 XML 的語意表達方面上，卻缺乏比較好的支援，一些學術上的文獻試著在這議題上研究，讓 XML 的表達能力更好，大部分這些研究都藉由在 XML 中定義 intergrity constraints [4,5] 來增強 XML 資料的表達能力。RDB 中的 functional dependencies(FDs)和 multivalued dependencies (MVDs)不僅提供資料間的語意概念，也在資料庫設計上扮演了重要角色。近來在 XML 上研究相依性[3,6,7]的議題已逐漸受到重視，當我們用 XML 來表達資料時，由於 XML 表達是彈性的，允許相同資料以許多不同的形式來表現，所以同樣一筆 XML 資料，資料表示方法不同，效益上可能會有不同的，例如需要較多或較少的儲存空間(storage)，或者是否有資料的重複性(redundancy)，本論文即是討論 XML 文件正規化的相關議題。

二、基本定義

在我們討論 XML 多值相依性以及 XML 第四正規化前，先在此節為後面會用到的符號，預先提出定義並說明。假設有四個不同的集合 *El*、*Att*、*Str*、*Vert*， *El* 是元素集合，*Att* 是屬性集合，*Str* 是所有可能的值，*Vert* 是所有節點構成的集合。S 表示#PCDATA，⊥表示 null，S 和⊥皆是保留的符號，並不屬於上述的任何集合裡。

定義 2.1 (文件型別定義,DTD) 一個文件型態定義(*Document Type Definition,DTD*)被定義為 $D=(E,A,P,R,r)$，其中 $E\subseteq El$ 是一個有限的元素型態集合，$A\subseteq Att$ 是一個有限的屬性集合，P 是一個函數(function)將 E 中的元素對應到元素型態定義。若是 τ 屬於 E，則 $P(\tau)$=S 或是 $P(\tau)$是一個常規表示式(regular expression)α:

$$\alpha ::= \varepsilon \mid a' \mid \alpha|\alpha \mid \alpha,\alpha \mid \alpha^*$$

其中 ε 代表空的元素集合，a' 代表屬於 E 的元素，“|”，“,”以及 “*” 分別代表 union，concatenation 以及 Kleene closure。R 是找出每一個在 E 中的元素的屬性集合(powerset of A)。r 是 DTD 的根元素，不失一般性下，令 $r\neq P(\tau)$當 $\tau\in E$，符

號 ε 表示元素型態是 EMPTY，S 表示#PCDATA。

例子 2.1 若有一個描述購物訂單的 DTD，如圖 2-1 所示。某家商店用這個 DTD 用來標準化訂單的格式。則 $D=(E,A,P,R,r)$，E、A、P、R、r 分別說明如下：

E 是此 DTD 裡所有的元素型態(element type)集合，E={PurchaseOrder,ItemsBought,Payments, Item,Payment}。

A 是 DTD 裡所有屬性集合，A= { @BuyerName,@Date,PartId,@Cost, @CreditCard , @ChargeAmt}，P(ItemsBought)=Item*，P(Item)= ε，R(ItemsBought) = ε，R(Item)={@PartId ,@Cost}，r 為樹根節點 PurchaseOrder。

```
<!ELEMENT PurchaseOrder (ItemsBought,
                                  Payments)>
      <!ATTLIST PurchaseOrder
            BuyerName   CDATA #REQUIRED
                 Date CDATA #REQUIRED>
<!ELEMENT ItemsBought (Item)*>
<!ELEMENT Item EMPTY>
     <!ATTLIST Item
                PartId CDATA #REQUIRED
                 Cost CDATA #REQUIRED>

<!ELEMENT Payments (Payment)* >
<!ELEMENT Payment EMPTY>
     <!ATTLIST Payment
            CreditCard CDATA #REQUIRED
            ChargeAmt CDATA #REQUIRED>
```

圖 2-1 purchaseorder.dtd

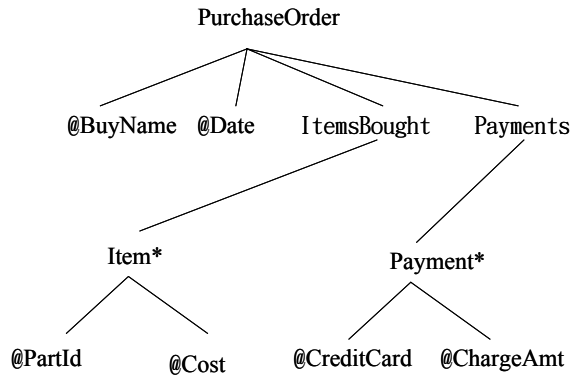


圖 2-2 purchaseorder.dtd 的樹狀圖

定義 2.2 (路徑,path) DTD 上的 path 是一種路徑的描述方式，在 $D=(E,A,P,R,r)$中，若有一節點到節點的路徑 $l_1.....l_n$，則 $l_i \in P(l_{i-1})$，此處 $i \in [2,n-1]$，若是 i=1,則 l_1 為 XML 樹的根節點。$l_n \in P(l_{n-1})$或是 l_n=@A，當然這時@A 會屬於 $R(l_{n-1})$。用 *paths*(*D*) 表示 *D* 上的所有 path 所形成的集合。*Epaths*(*D*)表示所有元素路徑(element path)所形成的集合，也就是 *Epaths*(*D*)裡的 paths 不可以是以字串(S)或是屬性(@)為終點，以符號表示的話，*Epaths*(*D*) = $\{p \in paths(D) | last(p) \in E\}$，*last*(*p*)表示路徑 *p* 的終點元素。一個 DTD 若有遞迴(*recursive*)的現象，則 *paths*(*D*)將會是無限的。*Leafpaths*(*D*)表示所有 leaf path 所形成的集合，
Leafpaths(*D*)= $\{p \in paths(D) | last(p) \notin E\}$。

例子 2.2 在圖 2-2 purchaseorder.dtd 裡頭，PurchaseOrder.@Date 和 PurchaseOrder.ItemsBought.Item.@PartId 是 *paths(D)* 裡頭的其中兩個 path，且皆是 *Leafpaths(D)*裡的路徑。PurchaseOrder.ItemsBought.Item 是 element path，屬於 *Epaths*(*D*)。
last(PurchaseOrder.ItemsBought.Item)=Item。

定義 2.3 (前序路徑,prefix path) 假如 p 是路徑 $l_1 \ldots l_n$，q 是路徑 $q_1 \ldots q_m$，若 p 是 q 的 prefix path，則記做 $p \subseteq q$，其中 $n \leqq m$，$l_1=q_1$，…，$l_n=q_n$，當 n=m 時，則 p=q。如果 p 是 q 的 prefix path，但是 $p \neq q$，則稱 p 為嚴格的前序路徑(strict prefix path)。令 q 是一個 path: $q_1 \ldots q_m$，定義函數 *parent*(q)= $q_1 \ldots q_{m-1}$，意即 *parent* 找出 q 的最大 strict prefix path。

例子 2.3 在圖 2-2 中，PurchaseOrder.ItemsBought 和 PurchaseOrder.ItemsBought.Item 都是 PurchaseOrder.ItemsBought.Item.@PartId 的 prefix path，*parent*(PurchaseOrder.ItemsBought.Item. @PartId)=PurchaseOrder.ItemsBought.Item。

定義 2.4 (部分路徑,partial path) 假如 p 是路徑 $l_i \ldots l_n$，若有一路徑 q:$l_1 \ldots l_m$，其中 $i \neq 1$，$n \leqq m$，則稱 p 是 q 的 partial path。

例子 2.4 圖 2-2 中，ItemsBought 和 ItemsBought.Item.@PartId 都算是 PurchaseOrder.ItemsBought.Item.@PartId 的 partial path。

定義 2.5 (路徑差集,path difference) 路徑差集是一個函數，所謂路徑差集就是在給定兩個路徑 p_1、p_2，找出在 p_1 裡的最大 partial path，但 partial path 上的元素不可以是 p_2 裡的元素，若有 2 個 path p_1、p_2，則用 $p_1 - p_2$ 來表示 p_1 跟 p_2 的 path difference，或以函式 *difference*(p_1,p_2)表示。

例子 2.5
difference(PurchaseOrder.ItemsBought.Item.@PartId, PurchaseOrder.ItemsBought)= Item.@PartId。
difference(PurchaseOrder.ItemsBought.Item.@PartId, PurchaseOrder.Payments.payment)
= ItemsBought.Item.@PartId。

定義 2.6 (路徑交集,path intersection) 路徑交集是一個函數，所謂路徑交集就是在給定兩個 path

情形下，找出這兩個 path 最大的共同 prefix path。若有 2 個路徑 p_1、p_2，則用 $p_1 \cap p_2$ 來表示 p_1 跟 p_2 的 intersection，或以函式 *intersection*(p_1,p_2)表示。

例子 2.6 *intersection*(
PurchaseOrder.ItemsBought.Item.@PartId,
PurchaseOrder.ItemsBought)=
PurchaseOrder.ItemsBought。
intersection(
PurchaseOrder.ItemsBought.Item.@PartId,
PurchaseOrder.Payments.payment)= PurchaseOrder。

定義 2.7 (XML tree) 一個 XML tree T 為一棵樹，用符號*(V,lab,ele,att,root)*來表示，其中 *V* 是有限的點集合，*lab* 是一個函數:$V \rightarrow El$，將 *V* 中的點(node)對應到 *E* 中的元素。*ele* 是一個函數:$V \rightarrow Str \cup V^*$找出 *V* 中的點的子節點。*att* 是一個 partial function:$V \times Att \rightarrow Str$。*root* 是 T 的根節點。

```
<PurchaseOrder BuyerName="Car Corporation"
               Date="1 Jan 2000">
    <ItemsBought>
        <Item PartId="1" Cost= "3000" />
        <Item PartId= 2" Cost="6000" />
    </ItemsBought>
    <Payments>
        <Payment CreditCard="8342398432"
                 ChargeAmt="8000.00" />
        <Payment CreditCard="3474324934"
                 ChargeAmt="2000.00" />
    </Payments>
</PurchaseOrder>
```

圖 2-3 purchaseorder.xml

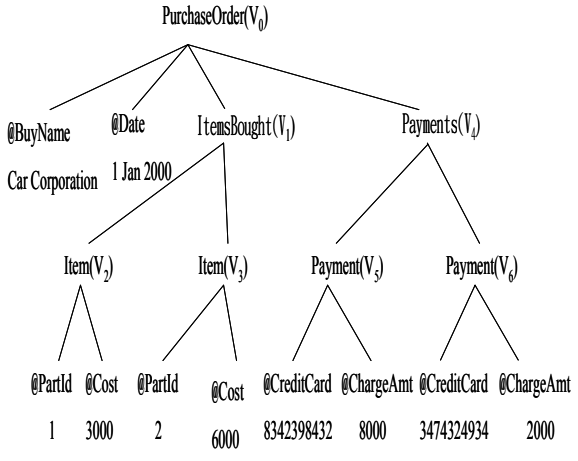


圖 2-4 purchaseorder.xml 的樹狀圖

例子 2.7 圖 2-3 是一個遵守圖 2-1 purchaseorder.dtd 的 XML 文件，purchaseorder.xml 用(*V,lab,ele,att,root*)來表示:
lab(V_0)=PuchaseOrder，*lab*(V_2)=Item，*ele*(V_1)={V_2,V_3}，*ele*(V_2)={1,3000}，*att*(V_5,@CreditCard)=8342398432，*root* 是 V_0 是 purchaseorder.xml 這個 XML 文件的根節點，如圖 2-4 所示。

定義 2.8 若有一個 DTD *D=(E,A,P,R,r)*和一個 XML tree *T = (V,lab,ele,att,root)*，若有下面的情形，則稱此 XML tree *T* 遵守 *D*。*lab* 是 *V* 對應到 *E* 的函數，*att* 是一個 partial function，任何屬於 *V* 集合裡的 v 和屬於 *A* 集合的@l，*att*(v, @l)存在若且為若 @l 屬於 *R*(*lab*(v))。*lab*(*root*)=*r* ，每一個在 *V* 集合裡的 v，如果 *P*(*lab*(v))=S，則 *ele*(v)=S。否則 *ele*(v)=[v_1,…,v_n]，*lab*(v_1),…,*lab*(v_n)屬於 *P*(*lab*(v))。

定義 2.9 (tree tuple) 給定一個 DTD *D* = (*E, A, P, R, r*), *D* 裡頭的每一個 tree tuple *t* 都是一個函數，*t: paths*(*D*) → *Vert* ∪ *Str* ∪ {⊥} 使得:

1. 若 p∈*EPaths*(*D*)，則 *t*(p)∈*Vert* ∪ {⊥}，但 *t*(*r*) ≠⊥(NULL).
2. 若 p∈*paths*(*D*) - *EPaths*(*D*)，*t*(p)∈Str∪{⊥}.
3. 若 *t*(p_1) = *t*(p_2) 而且 *t*(p_1)∈*Vert*, 那麼 p_1 會等於 p_2.
4. 若 *t*(p_1)=⊥ 而且 p_1 是 p_2 的 prefix path, 那麼 *t*(p_2)=⊥(NULL).
5. 若{p∈*paths(D)* | *t*(p)≠⊥}，則 p 會是一個有限的路徑.

用 *T(D)*表示 *D* 上的所有的 tree tuple 所形成的集合。

例子 2.8 若有一個 DTD *D* courses.dtd，如圖 3-1 所示。圖 3-2 courses.xml 是遵守 *D* 的 tree，在這個 tree 裡存在一個 tree tuple t，使得

t(courses) = v_0
t(courses.course) = v_1
t(courses.course.name) = v_2
t(courses.course.teacher) = v_3
t(courses.course.text) = v_4
t(courses.course.name.S) ="DB"
t(courses.course.teacher.S) ="JOE"
t(courses.course.text.S) ="Text A"

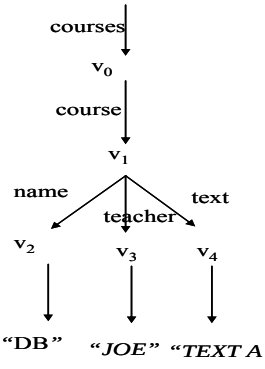


圖 2-5 tree tupple *t* 示意圖

用圖 2-5 可以更清楚的明白 tree tuple t 的對應關係，雖然某些 tree tuples 可能將 path 對應到空值

(null,⊥)，例如 $D = (E, A, P, R, r)$， $E = \{r, a, b\}$，$A = \phi$，$P(r) = (a|b)$，$P(a) = \varepsilon$ and $P(b) = \varepsilon$，$paths(D) = \{r, r.a, r.b\}$，路徑 $r.a$ 和路徑 $r.b$ 不會同時出現，所以一定會有 $t(r.a)=null$ 或是 $t(r.b)=null$ 的情形發生。在這篇論文裡頭，我們只討論每一個 XML tree 都有抵達葉節點，並且不去考量 null 的情形。

定義 2.10 (等價) 若存在兩個 XML tree T_1、T_2，如果 $tuples_D(T_1)$和 $tuples_D(T_2)$隱含相同的訊息則稱 T_1 和 T_2 具有等價關係，以 $T_1 \equiv T_2$ 來表示。

三、XML 文件的多值相依性

在這節裡，我們延伸文獻[3]所提出的 XML Functional Dependency(XFD)概念以及關聯式資料的多值相依性(MVD)，來定義一個新的 XML Multivalued Dependency (XMVD)，並在第四節說明如何利用 XML 的樹狀結構來達到減少資料重複的目的，但卻不會失去 XMVD 的語意概念。在這篇論文裡依舊使用 tree tuple 來定義 XML 上的 multivalued dependency 。

定義 3.1 (XML 多值相依性,XMVD) 在任何一個 DTD D 上，我們使用 tree tuple 來定義 XMVD，XML 多值相依性(XMVD)是指在 D 上具有 $p_1 \rightarrow\rightarrow p_2 | p_3$ 形式的一種相依性，這裡的 p_1、p_2、p_3 是 *paths(D)* 裡有限的、非空的子集合，所有在 D 上的 XMVD 用 ***XMVDS(D)*** 來表示。

任兩個在 *T(D)* 裡的 tree tuple t_1，t_2 ,若 $t_1(p_1)=t_2(p_1)$ 則會存在 t_3, t_4 在 $tuples_D(T)$裡,使得

$t_1(p_1)=t_2(p_1)=t_3(p_1)=t_4(p_1)$
$t_3(p_2)=t_1(p_2)$
$t_3(p_3)=t_2(p_3)$
$t_4(p_2)=t_2(p_2)$
$t_4(p_3)=t_1(p_3)$

直覺來看，一個 XML multivalued dependency: $p_1 \rightarrow\rightarrow p_2 | p_3$ 是表達出 p_1 跟 p_2 的關係獨立於 p_1 跟 p_3 的關係。

例子 3.1 圖 3-1 courses.dtd 是描述學校課程資訊的 dtd，若存在一個 XMVD:
courses.course.name.S→→ courses.course.teacher.S | courses.course.text.S (XMVD 1)

```
<!DOCTYPE courses [
    <!ELEMENT courses(course*)>
    <!ELEMENT course(name,teacher,text)>
    <!ELEMENT name(#PCDATA)>
    <!ELEMENT teacher(#PCDATA)>
    <!ELEMENT text(#PCDATA)>
]>
```

圖 3-1 courses.dtd

若圖 3.2 是一個遵守 courses.dtd 的文件，我們用 tree tuple 來表達圖 3-2 的文件。courses.xml 共由 4 個不同的 tree tuple 組成，見圖 3-3。

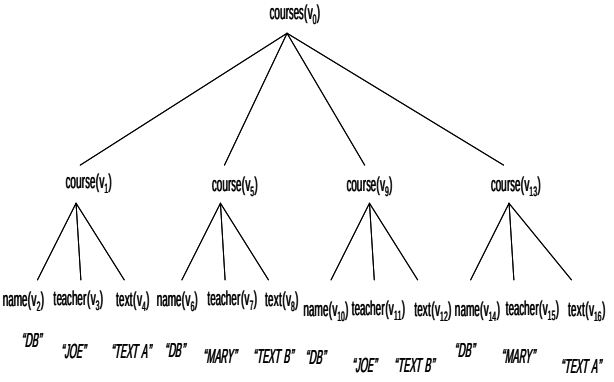


圖 3-2 courses.xml

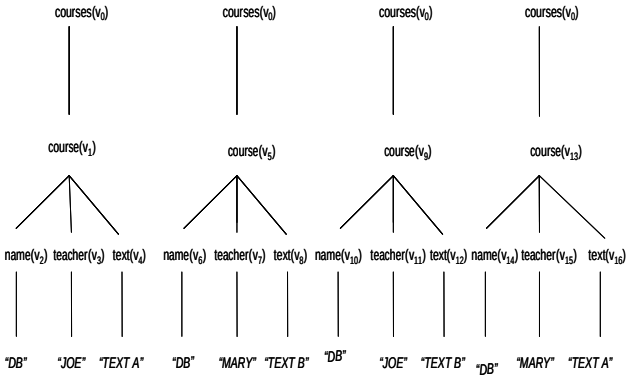


圖 3-3 四個不同的 tree tuple 示意圖

將 4 個 tree tuple 轉換成關聯式資料表的 4 個 tuple，如表 3-1 所示，觀察會得知 courses.xml 有類似關聯式的多值相依性。意即由表 3-1 我們會知道 courses.xml 滿足 XML 多值相依性(XMVD 1)。

表 3-1 圖 3-3 對應的關聯式資料表

t1	V_0	V_1	V_2	V_3	V_4	DB	JOE	TEXT A
t2	V_0	V_5	V_6	V_7	V_8	DB	MARY	TEXT B
t3	V_0	V_9	V_{10}	V_{11}	V_{12}	DB	JOE	TEXT B
t4	V_0	V_{13}	V_{14}	V_{15}	V_{16}	DB	MARY	TEXT A

如同關聯式資料庫一般，並不是所有的多值相依性都必須被消除，唯有那些會導致資料重複的 XMVD 才需被消除，那些會導致資料重複的 XMVD，我們稱它們為不規則的 XMVD (anomalous XMVDs)。另外必須定義出 trivial XMVD，好在正規化步驟裡略過 trivial XMVD，因為 trivial XMVDs 總是存在於任何 DTD 裡。在 RDB

裡，trivial MVD 是具有 A→→B 形式的多值相依性，其中 A∪B=R 或是 B⊆A。要在 XML 裡定義出 trivial XMVD 是更加複雜的一件事，已知 DTD D 和 XMVDs 集合 Σ，在此 DTD 和 Σ 條件下衍生出來的 XMVD 稱 φ，用 $(D, \Sigma) \vdash \varphi$ 來表示，任何遵守 (D,Σ) 的 XML 文件也會遵守 φ，所有被 (D,Σ) 衍生出的 XMVDs 用集合 $(D,\Sigma)^+$ 來表示，我們說一個 XMVD φ 是 trivial 的意思是指 $(D, \emptyset) \vdash \varphi$，舉例說明:在 XML 文件裡若 *p* 屬於 *Epaths*(*D*)，且存在 p' 是 *p* 的前序路徑(prefix path)則 $(D, \emptyset) \vdash p \rightarrow\rightarrow p'$ 會存在。另外若 *p*、*p@l* 屬於 *paths*(*D*)，則 $(D, \emptyset) \vdash p \rightarrow\rightarrow p@l$ 會存在。

四、XML 第四正規化

一個符合 XMVD 的 XML 文件可能會有什麼樣的問題，一個遵守圖 3-1 courses.dtd 的 XML 文件，並滿足(XMVD 1)的 XML 文件，如圖 3-2 所示的 courses.xml。圖 3-2 courses.xml 被使用來儲存學校裡授課的資訊，此文件儲存著每一門課程的名稱以及可以教授此課程的老師，以及可以用來使用的教材。假如我們要將 peter 老師加入 DB 可授課老師行列裡，就會產生異常情形，因為 courses.dtd 的結構問題，我們必須要同時加入 2 筆資料才不會因為加入 peter 老師而導致違背(XMVD 1)限制。同樣的若是一位老師退休的話也勢必會有異常現象，這也是因為要從 xml 文件裡刪除 2 筆資料，才不會使文件違背 XMVD。

接下來看另一個例子，若是有一個遵守圖 4-1 dtd 的 XML 文件，並且滿足(XMVD 2)的 XML 文件，如圖 4-2 newcourses.xml。在 newcourses.xml 裡，我們將 peter 老師加入 DB 的可授課老師行列裡，就不會有異常情形，因為只需在 teachers 節點下加入 peter 老師的訊息即可。若是一位老師退休的話也不會有刪除異常的現象了。這是因為所刪除和所加入的資料並不會導致 XML tree 違背 (XMVD 2)。

```
<!DOCTYPE newcourses [
    <!ELEMENT courses(course*)>
    <!ELEMENT course(name,teachers,texts)>
    <!ELEMENT name(#PCDATA)>
    <!ELEMENT teachers(teacher*)>
    <!ELEMENT teacher(#PCDATA)>
    <!ELEMENT texts(text*)>
    <!ELEMENT text(#PCDATA)>
]>
```

圖 4-1　newcourses.dtd

courses.course.name.S→→
courses.course.teacher.teacher.S |
courses.course.texts.text.S
(XMVD 2)

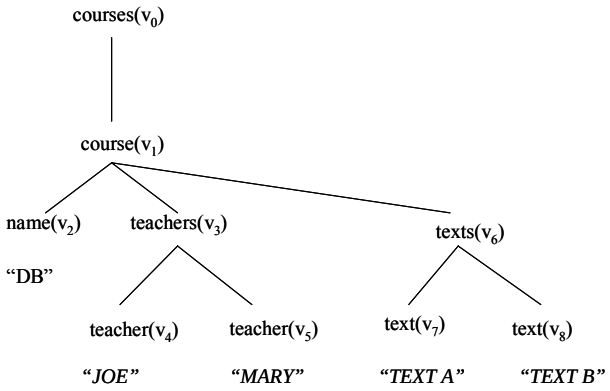


圖 4-2 newcourses.xml 包含重複的資訊

觀察上述的兩個 XML 文件，我們可以輕易的發現上面這兩個 XML 文件的目的是一樣的，所儲存的資訊也是一樣的，都是為了存放學校課程資訊而被設計的，也都各自遵循所該遵循的 DTD 和 XMVD，只是結果不同，其中一個會導致更新的異常，而另外一個卻不會。經由上面例子的描述，我們可以很容易的相信明白，當我們若要設計一份具有 XMVD 語意的 XML 文件時，有可能會設計出隱含有更新異常問題的文件或是無更新異常問題的文件。當然我們會希望設計出的文件是沒有更新異常且是盡量佔用較少儲存空間的文件。為了能更進一步的說明接下來的內容，我們將一份遵循 XMVDs 的文件且此文件沒有更新異常現象，稱為是滿足 4XNF 的文件。

根據先前的敘述，我們提出一個正式定義來描述何謂是 4XNF 的文件。

定義 4.1 (XML 第四正規化, 4XNF)

若有一個 DTD D，存在一個 XMVD 及 XFD 的集合 Σ，(D,Σ) 若是符合 XML 第四正規化 (4XNF) 若且為若每一個具有 $S_1 \rightarrow\rightarrow S_2 \mid S_3$ 的形式的 nontrivial XMVD $\varphi \in (D,\Sigma)^+$，其中 $S_1=\{q,x_1.@x_1,...,x_m.@x_m\}$，$S_2=\{y.@l\}$，$S_3=\{z_1.@z_1,...,z_n.@z_n\}$，$m \geqq 0$，$n \geqq 1$，$S_1 \cup S_2 \cup S_3 = \mathit{Leafpaths}(D) \cup q$，$S_i \cap S_j = \phi$，for $i \neq j$ 且任何 $(a,b) \in S_2 \times S_3$，p=intersection(a,b)，存在 $S_1 \rightarrow p \in (D,\Sigma)^+$。

4.2 有效的 XML 第四正規化演算法

接下來將解釋如何將任意一個滿足 DTD *D*、相依性集合 Σ，但不符合 4XNF 的文件，轉換成符合第四正規化的文件。若 $S_1 \rightarrow\rightarrow S_2 \mid S_3 \in (D,\Sigma)^+$ 是一個異常的 XMVD (anomalous XMVD)，且 $S_2=\{y.@l\}$，則稱 y.@l 是 anomalous path，用集合 $AP(D,\Sigma)$ 來表示 *D* 隱含的所有 anomalous path。若針對 $S_1 \rightarrow\rightarrow S_2 \mid S_3$ 這個 anomalous XMVD 做正規化，則會得到新的 DTD D'，新的相依性集合 Σ'。

我們假設所有的 DTD 都是非遞迴的 (non-recursive)，所有的 anomalous XMVD 都是 $S_1 \rightarrow\rightarrow S_2 \mid S_3$ 的形式，其中 $S_1=\{q,x_1.@x_1,...,x_m.@x_m\}$，$S_2=\{y.@l\}$，

$S_3=\{z_1.@z_1,...,z_n.@z_n\}$，m≧0，n≧1，$S_1 \cup S_2 \cup S_3 = Leafpaths(D) \cup q$，$S_i \cap S_j = \phi$，for i≠j 且 $q \in EPaths(D)$。最多只有一個 element path 在左邊，這是因為若有兩個以上的 element path 發生在左半部，那麼這個 XMVD 在語意表達上會相當的不自然，然而即使真的存在這樣的情形，也可以輕易的利用增加一個屬性@l 來轉變，例如將 $\{q, q'\} \cup S \rightarrow\rightarrow S_2 \mid S_3$ 縮減成 $q'.@l \rightarrow q'$ 和 $\{q, q'.@l\} \cup S \rightarrow\rightarrow S_2 \mid S_3$。此外我們假設並沒有以 S(#PCDATA)做為終點的路徑(path)，這是因為在這個演算法表達方法裡，p.S 可以用@ p 來表達。

我們依 XMVD 的形式，將演算法分成三種情形，每一種情形適合某條件成立的 XMVD，依據 XMVD 的條件，使用其中之一來正規化 XML 文件。

(一) case 1:

$D=(E,A,P,R,r)$ 是一個 DTD，令 anomalous XMVD : $S_1 \rightarrow\rightarrow S_2 \mid S_3$ ，其中 $S_1=\{q,x_1.@x_1,...,x_m.@x_m\}$，$S_2=\{y.@l\}$，$S_3=\{z_1.@z_1,...,z_n.@z_n\}$，m≧0，n≧1，$S_1 \cup S_2 \cup S_3 = Leafpaths(D) \cup q$，$S_i \cap S_j = \phi$，for i≠j 且，$q \in EPaths(D)$，若 $q \rightarrow S_1$，要消除 anomalous XMVD，則在 q 底下建立一個新的點(element) τ，移動屬性@l 到 τ 的屬性集合裡，使得屬性@l 為 τ 的屬性，生成新的 DTDD[y.@l:=q.τ.@l]，如圖 4-3 所示。

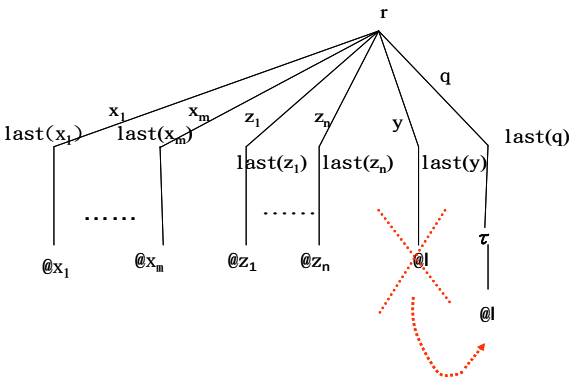


圖 4-3 case 1：DTD 轉換示意圖

令新生成的 DTD D[y.@l:=q.τ.@l]用 (E', A, P', R', r) 來表示，$E'=E \cup \tau$，且

1.如果 $P(last(q))$是一個 regular expression，則

$P'(last(q)) = (P(last(q)), \tau^*)$，

$P'(\tau) = \varepsilon$。

$P'(e) = P(e)$ for each $e \in E-\{last(q)\}$。

2. $R'(\tau) = @l$，$R'(last(y)) = R(last(y))-\{@l\}$，

$R'(e) = R(e)$ for each $e \in E-\{last(y)\}$。

將 DTD 做轉換後，得到新的 DTD $D'=D$[y.@l:=q.τ.@l]，由於轉換動作已經改變了 DTD 的結構，所以連帶的產生了新的相依性 $\sum' = \sum$ [y.@l:=q.τ.@l]，$\sum'$ 包含 $\sum$ 裡的所有的相依性，但是將 y.@l 改為 q.τ.@l 。

例子 4.1 若是存在一個 DTD 如圖 4-4 department.dtd 是一個描述某學院的系所上課地點和課程名稱的 dtd， 若此 dtd 具有(XMVD 3)這個多值相依性的話，意思是說同一個系所開的任何課程可以使用系所分配到的任何教室上課，若有一文件符合此 DTD 且滿足(XMVD 3)如圖 4-5。圖 4-5 並不符合 4XNF，因為存在一個路徑 college.department.class_info，此路徑是 college.department.class_info.c_name.S 和 college.department.class_info.location.S 做 intersection 得來.且 college.department 無法決定(→) college.department.class_info，所以這樣的文件不符合 4XNF。我們依據演算法，將它變成符合 4XNF 的文件如圖 4-6。我們在元素 department 底下建立一個叫 class 的 element，並移動 c_name.S 到 class 底下，把 c_name.S 當成是 class 的的屬性，得到新的 DTD，完成正規化。經正規化後的文件明顯的少了會引起資料重複的(XMVD 3)，而以不會引起資料重複的 XMVD :

college.department →→ college.department.class.c_name.S | {college.department.class_info.location.S , college.department.@name}取代會引起異常的(XMVD 3)。

```
<!DOCTYPE department [
    <!ELEMENT college(department*)>
    <!ELEMENT department(class_inf*)>
    <!ATTLIST department
        name CDATA #REQUIRED >
    <!ELEMENT class_inf(c_name,location)>
    <!ELEMENT c_name(#PCDATA)>
    <!ELEMENT location(#PCDATA)>
]>
```

圖 4-4 department.dtd

college.department →→ college.department.class_info.c_name.S | {college.department.class_info.location.S , college.department.@name} (XMVD 3)

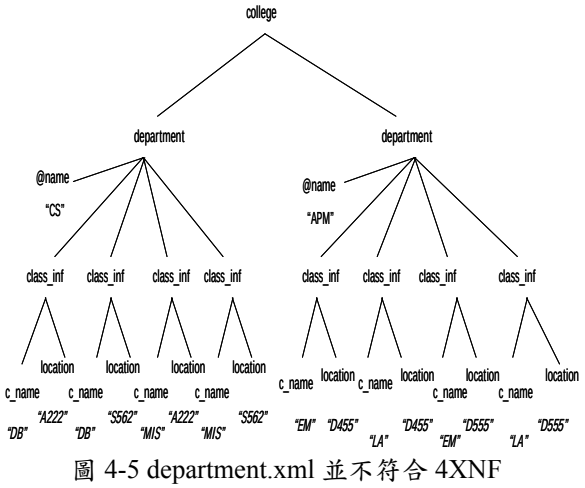


圖 4-5 department.xml 並不符合 4XNF

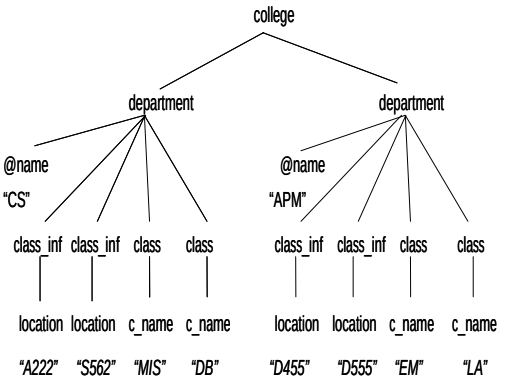


圖 4-6 department.xml 經正規化後的文件

(二) case 2:

令 $D=(E,A,P,R,r)$ 是一個 DTD，若存在 anomalous XMVD: $S_1 \rightarrow\rightarrow S_2 \mid S_3$ ，其中 $S_1=\{q,x_1.@x_1,...,x_m.@x_m\}$，$S_2=\{y.@l\}$，$S_3=\{z_1.@z_1,...,z_n.@z_n\}$，$S_1 \cup S_2 \cup S_3 = Leafpaths(D) \cup q$，$S_i \cap S_j = \phi$，for $i \neq j$，$q \in EPaths(D)$，$m \geqq 1$，$n \geqq 1$，，若是同時具有以下情形時:

1.任何$(a,b) \in S_2 \times S_3$，$intersection(a,b)=y$

2.若 S_1 裡最長的路徑是 p，

$intersection(y,p)=parent(p)$

在 y 底下建立兩個新的子節點(element) τ，τ'，移動@l 到 τ' 的屬性集合裡。原在 S_3，以 y 為 root 的根節點的路徑，全部轉變成以 τ 為根節點。形成成新的 DTD $D[y@l:=y.\tau'.@l, [z_1.@z_1,...,z_n.@z_n]:= y.\tau[(z_1\text{-}y).@z_1,\ldots,(z_n\text{-}y).@z_n]]$。如圖 4-7 所示，用 (E', A, P', R', r) 來表示。

$E'=E \cup \{\tau, \tau'\}$,且

1. $P'(\tau')=\varepsilon$， $P'(\tau)=P(\text{last(y)})$，
   $P'(\text{last(y)})=(\tau^*, \tau'^*)$，
   $P'(e)=P(e)$ for $e \in E-\{\text{last(y)}\}$。
2. $R'(\tau')=@l$。
   $R'(\tau)=R(\text{last(y)})-\{@x_1,...,@x_m,@l\}$。
   $R'(\text{last(y)})=R(\text{last(y)})-\{@z_n,...,@z_n,@l\}$。
   $R'(e)=R(e)$for each $e \in E-\{\text{last(y)}\}$。

$\sum' = \sum [y@l:=y.\tau'.@l, [z_1.@z_1,...,z_n.@z_n]:= y.\tau.[(z_1\text{-}y).@z_1,\ldots,(z_n\text{-}y).@z_n]]$，$\sum'$ 包含 $\sum$ 裡的所有的相依性，但是將 y.@l 改為 $y.\tau'.@l$， $z_i.@z_i$ 變成 $y.\tau.(z_i\text{-}y).@z_i$，$z_i$ 變成 $y.\tau.(z_i\text{-}y)$。

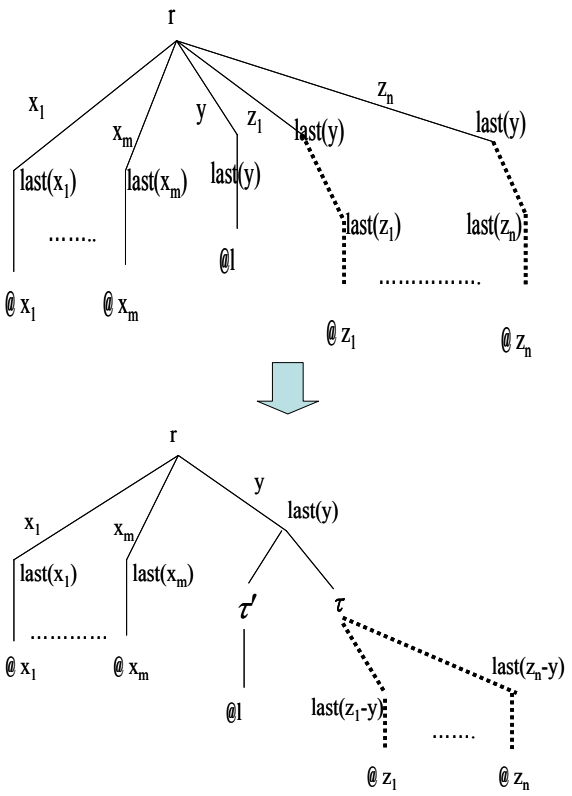


圖 4-7 case 2：DTD 轉換示意圖

例子 4.2 圖 3-2 是遵守圖 3-1 courses.dtd 的 XML 文件，且具有多值相依性(XMVD 1)。因為圖 3-2 不是 4XNF 的文件，所以將文件轉換成圖 4-8 格式。將(XMVD 1)視為 **courses.course.@name→→ courses.course.@teacher | courses.course.@text。** courses.course.@teacher∩courses.course. @text= courses.course。courses.course 是 S_1 裡最長路徑 courses.course.@name 的 parent path，courses.course = *parnet*(courses.course.@name)。所以在 courses.course 底下建立兩個子點 teachers 和 texts，移動 teacher.S 到 teachers 底下，移動 text.S 值到 texts 底下。會得到圖 4-8 的 xml tree。

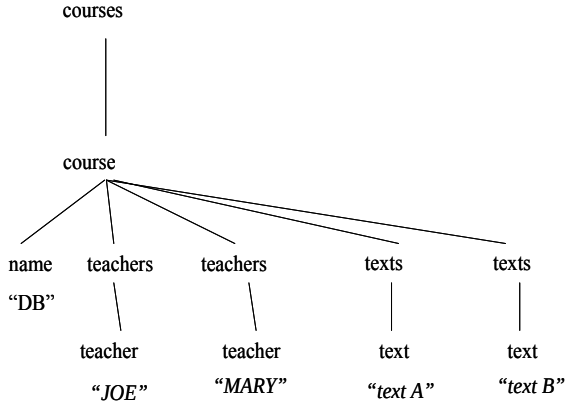


圖 4-8 courses.xml 經正規化後的文件

(三) case 3:

令 $D=(E,A,P,R,r)$是一個 DTD，若存在

anomalous XMVD: $S_1 \rightarrow\rightarrow S_2 \mid S_3$ ，其中 $S_1=\{q,x_1.@x_1,...,x_m.@x_m\}$，$S_2=\{y.@l\}$，$S_3=\{z_1.@z_1,...,z_n.@z_n\}$，$S_1 \cup S_2 \cup S_3 = Leafpaths(D) \cup q$，$S_i \cap S_j = \phi$，for $i \neq j$，$q \in EPaths(D)$，$m \geq 1$，$n \geq 1$，若 XMVD 不是上述的兩種 case 狀況的話，則建立一個 element τ，使 τ 為 *last*(q)的子節點，並在 τ 底下建立 m+1 個子節點 τ_1，…，τ_m，τ'。分別移動屬性$@x_1$，...，$@x_m$ 到 τ_1，…，τ_m 的屬性集合裡，並且移動屬性@l 到 τ' 的屬性集合，但是不將原本屬於 $last(x_1),\ldots,last(x_m)$ 的$@x_1,\ldots,@x_m$ 移除，這個步驟形成新的 DTD $D[y@l:=q.\tau[\tau_1@x_1,\ldots,\tau_m@x_m,\tau'.@l]]$，如圖 4-9 所示。

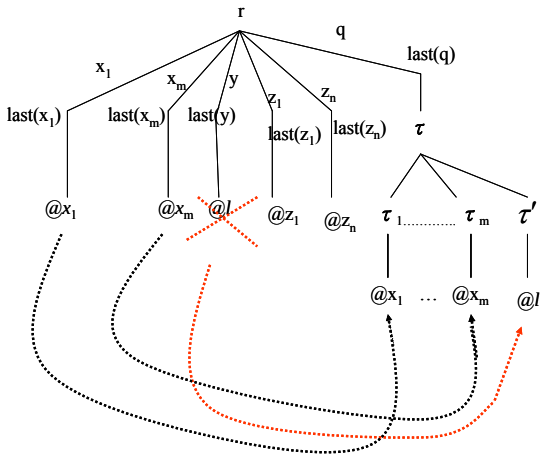


圖 4-9 case 3：DTD 轉換示意圖

DTD $D[y@l:=q.\tau[\tau_1.@x_1,\ldots,\tau_m.@x_m,\tau'.@l]]$ 用 (E', A, P', R', r) 來表示。

$E'=E \cup \{\tau,\tau',\tau_1,\ldots,\tau_m\}$

1. 如果 $P(last(q))$ 是一個 regular expression，用 s 來表示這樣的 regular expression，
   則 $P'(last(q))=(s, \tau^*)$，
   $P'(\tau)=(\tau_1^*,\ldots,\tau_n^*,\tau'^*)$。
   $P'(\tau_i)=\varepsilon$ for each $i \in [1,m]$。
   $P'(\tau')=\varepsilon$。
   $P'(e)=P(e)$ for $e \in E-\{last(q)\}$。
2. $R'(\tau)=\varepsilon$。
   $R'(\tau_i)=@x_i$ for each $i \in [1,m]$。
   $R'(\tau')=@l$。
   $R'(e)=R(e)$for each $e \in E-\{last(y)\}$。
   $R'(last(y))=R(last(y))-\{@l\}$。

$\sum'=\sum$ $[y@l:=q.\tau[\tau_1.@x_1,\ldots,\tau_m.@x_m,\tau'.@l]]$，由底下(1)、(2)、(3)構成：

(1)所有在 $\sum$ 裡的相依性，將 y.@l 改為 $q.\tau.\tau'.@l$。

(2)如果 $S_1 \cup S_2 \subseteq \{q,x_1,\ldots,x_m,x_1.@x_1,\ldots,x_m.@x_m, y.@l\}$ 且 $S_3 \subseteq paths(D)- S_1-S_2$，$S_1 \rightarrow\rightarrow S_2 \mid S_3 \in (D,\sum)^+$，則將 x_i 改為 $q.\tau.\tau_i$，$x_i.@x_i$ 改為 $q.\tau.\tau_i.@x_i$，y.@l 改為 $q.\tau.\tau'.@l$。

(3)$\{q, q.\tau.\tau_1.@x_1,..., q.\tau.\tau_m.@x_m\} \rightarrow q.\tau$ 及$\{q.\tau, q.\tau.\tau_i.@x_i\} \rightarrow q.\tau.\tau_i$。

當使用這個 case 轉換時，被轉換的 anomalous XMVD 需要是 minimal 的，亦即我們不希望轉換後，取代的 XMVD 還包含有 anomalous 元素。如果沒有 anomalous XMVD：

$S' \rightarrow\rightarrow x_i.@x_i \mid Leafpaths(D)- S'-\{x_i.@x_i\} \in (D,\sum)^+$，其中 $i \in [0,n]$且 $S' \subset \{q,x_0,\ldots,x_m,x_0.@x_0,...,x_m.@x_m\}$，$|S'| \leq n$，則我們稱$\{q,x_1.@x_1,...,x_m.@x_m\} \rightarrow\rightarrow x_0.@x_0 \mid Leafpaths(D)-\{q,x_1.@x_1,...,x_m.@x_m, x_0.@x_0\}$ 是 minimal anomalous XMVD。

例子 4.3 若存一個遵守圖 4-10 course2.dtd 的 XML tree 稱 course2.xml，如圖 4-11 所示，且 course2.xml 滿足(XMVD 4)

```
<!DOCTYPE course2 [
  <!ELEMENT courses(course*)>
      <!ATTLIST courses
        department CDATA #REQUIRED >
      <!ELEMENT course(name,teacher,text)>
  <!ELEMENT name(#PCDATA)>
  <!ELEMENT teacher(#PCDATA)>
  <!ELEMENT text(#PCDATA)>
]>
```

圖 4-10 course2.dtd

courses.course.name.S→→ courses.course.teacher.S |{courses.course.text.S , courses.@department}

(XMVD 4)

因為有存在一個 path 是 courses.course.teacher.S 和 courses.course.text.S 的對大 prefix path，這個最大 prefix path 是 courses.course，使得路徑 courses.course.name.S 無法決定 courses.course ，所以 course2.xml 並不符合 4XNF。依據演算法，可以將 course2.xml 轉會成符合 4XNF 的文件，見圖 4.12 在這裡 q 是 courses，在 courses 底下建立一個子節點 info 用來存放 S_1、S_2 的資訊，在 info 底下建立兩個子節點分別是 course_joint 和 teachers，移動 name.S 到 course_joint 底下，移動 teacher.S 到 teachers 底下，注意我們不可刪除原本在 course 底下的 name.S，因為這個值存在的目的是要做 joint 來還原訊息的。

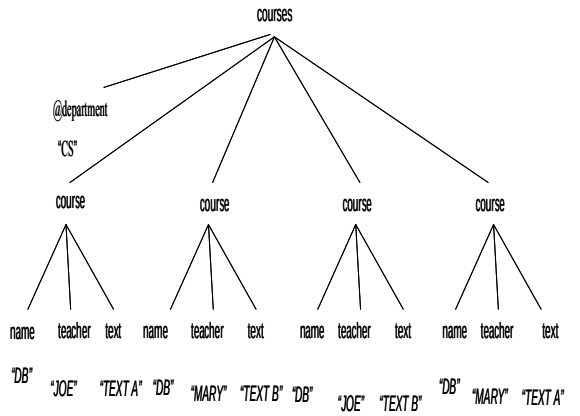


圖 4-11 course2.xml

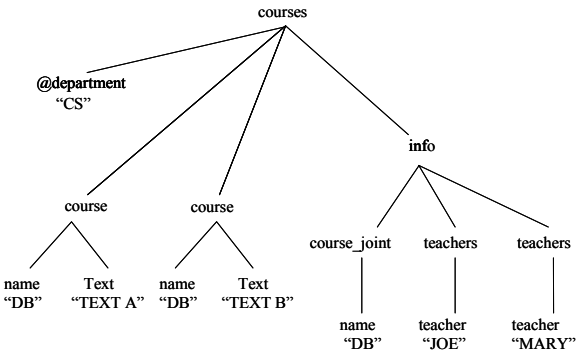


圖 4-12 course2.xml 經正規化後的文件

綜合以上 3 種 case，我們將 XML 第四正規化演算法整理成如圖 4-13 所示。

(1) If (D,Σ) is in 4XNF then return (D,Σ), otherwise go to step (2).
(2) If there is an anomalous xmvd $S_1 \rightarrow\rightarrow S_2 \mid S_2$ where $S_1=\{q,x_1.@x_1,...,x_m.@x_m\}$,$S_2=\{y.@l\}$, $S_3=\{z_1.@z_1,...,z_n.@z_n\}$,$m \geqq 0, n \geqq 1$, $S_1 \cup S_2 \cup S_3 = Leafpaths(D) \cup q$, $S_i \cap S_j = \phi$ for $i \neq j$, $q \in EPaths(D)$,若 $q \rightarrow S_1$,then:
(2.1) Create a fresh element type τ
(2.2) D := D[y.@l:=q.τ.@l]
(2.3) $\Sigma := \Sigma$ [y.@l:=q.τ.@l]
(2.4) Go to step(1)
(3) Else if there is an anomalous xmvd $S_1 \rightarrow\rightarrow S_2 \mid S_3$ where $S_1=\{q,x_1.@x_1,...,x_m.@x_m\}$,$S_2=\{y.@l\}$, $S_3=\{z_1.@z_1,...,z_n.@z_n\}$,$m \geqq 1, n \geqq 1$, $S_1 \cup S_2 \cup S_3 = Leafpaths(D) \cup q$, $S_i \cap S_j = \phi$ for $i \neq j$, $q \in EPaths(D)$, $\forall (a,b) \in S_2 \times S_3$, $intersection(a,b)=y$, $intersection(y,p)=parent(p)$,where p is the longest path in S_1,then
(3.1) Create fresh element types τ ,τ'
(3.2) D:=D[y@l:=y.τ'.@l , $[z_1.@z_1,...,z_n.@z_n] := y.\tau[(z_1\text{-}y).@z_1,\ldots,(z_n\text{-}y).@z_n]]$
(3.3) $\Sigma := \Sigma$ [y@l:=y.τ'.@l , $[z_1.@z_1,...,z_n.@z_n] := y.\tau[(z_1\text{-}y).@z_1,\ldots,(z_n\text{-}y).@z_n]]$
(3.4) Go to step(1)
(4) Else there is a minimal anomalous XMVD: $S_1 \rightarrow\rightarrow S_2 \mid S_3$ where $S_1=\{q,x_1.@x_1,...,x_m.@x_m\}$,$S_2=\{y.@l\}$, $S_3=\{z_1.@z_1,...,z_n.@z_n\}$,$m \geqq 1, n \geqq 1$, $S_1 \cup S_2 \cup S_3 = Leafpaths(D) \cup q$, $S_i \cap S_j = \phi$ for $i \neq j$, $q \in EPaths(D)$,then
(4.1) Create fresh element types $\tau, \tau', \tau_1,\ldots,\tau_m$
(4.2) D:=D[y@l:=q.τ[τ_1@x_1,….,τ_m@x_m,τ'.@l]]
(4.3) $\Sigma := \Sigma$ [y@l:=q.τ[τ_1@x_1,…,τ_m@x_m,τ'.@l]]
(4.4) Go to step(1)

圖 4-13 4XNF 重組演算法

由於 XML 查詢語言尚未標準化，所以文獻[3]採用函數對應的方式來證明正規化的不失真，這裡我們延續文獻的方法來證明我們的演算法是不失真的重組
，所謂不失真重組是指文件經正規化重組動作後，資訊能完整被保留，資訊不會因為正規化動作而有遺失。
換句話說，就是正規化前的文件上訊息，可由正規化後的文件中取得。如果有一個函數 f 存在，如圖 4-14，使得 D' 裡的 paths 能對應到 D 裡頭的 paths，每個遵守 (D,Σ) 的 xml tree T，若存在一個 xml tree T' 遵守 (D',Σ') 使得 D、D' 上的 paths 皆滿足函數 f，我們稱 (D',Σ') 是不失真的重組結果。

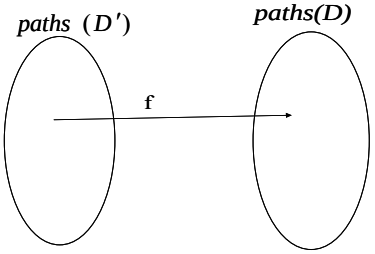


圖 4-14 f (*paths*(D'))=*paths*(*D*)

上述的 3 種轉換 cases，我們進一步證明每一個 case 都是不失真的重組，意即轉換後的文件並不會有資訊內容的遺失。詳細證明，請參閱附錄或[1]。

定義 4.2： 假設 D 是一個 DTD，Σ 是 D 上的相依性集合，若 $S_1 \rightarrow\rightarrow S_2 \mid S_3$ 是 Σ 裡的一個 anomalous XMVD，針對 $S_1 \rightarrow\rightarrow S_2 \mid S_3$ 做正規化所得到 (D',Σ')，若是存在一個函數 f 可以將 D' 裡的 path 對應到 D 裡的 path，使得每一個 T 遵守 (D,Σ) 會存在一個 T' 遵守 (D',Σ') 使得 $T \equiv_f T'$，我們稱 (D',Σ') 是 (D,Σ) 的不失真重組，用 $(D,\Sigma) \leq_{\text{lossless}} (D',\Sigma')$ 表示。

定理 4.1 (D,Σ) 經由我們所提出的正規化演算法轉變成 (D',Σ')，每一個 case 都是不失真的重組: $(D,\Sigma) \le_{\text{lossless}} (D',\Sigma')$。

經每一步驟的轉換，我們可以證明轉換後的 XML 文件上的不規則路徑會小於轉換之前。詳細證明，請參閱附錄或[1]。

定理 4.2 假設 D 是一個 DTD，Σ 是 D 上的相依性集合，若 $S_1 \rightarrow\rightarrow S_2 \mid S_3$ 是 Σ 裡的一個 anomalous XMVD，針對 $S_1 \rightarrow\rightarrow S_2 \mid S_3$ 做正規化所得到 (D',Σ')，會有 $|\text{AP}(D',\Sigma')| < |\text{AP}(D,\Sigma)|$ 的性質。

定理 4.3 4XNF 重組演算法動作終止時，亦即 Σ 中已無需正規化的 anomalous XMVD 時，則此時的 (D,Σ) 是 4XNF 形式。

4.3 4XNF 與 XFD、XMVD 的關聯性

在 RDB 的 MVD 理論裡，並不是所有 MVD 都是危險的，若存在一個 MVD: X→→Y 而同時具有 FD : X→Y 的話，此 MVD 並不會導致文件違反 4NF，trivial MVDs 也不違反 4NF，真正在 RDB 裡做第四正規化必須消除的 MVD 是所謂 true MVDs，true MVD 是具有 X→→Y 形式的多值相依性，但是 X↛Y，見圖 4-15，灰色的外圈所包含的多值相依性是必須被消除的。

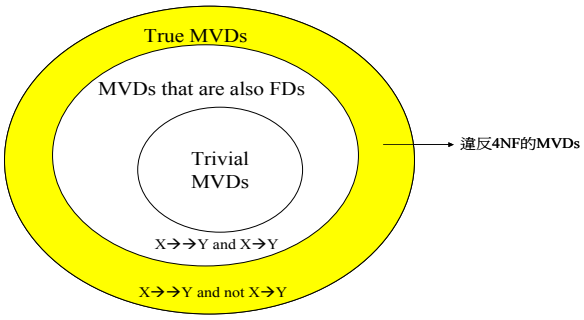


圖 4-15 RDB 理論的 FD、MVD 與 4NF 的關係圖

文獻[3]所提到需正規化的功能相依性是具有 $S_1 \rightarrow S_2$ 形式的 nontrivial XFD，其中 $S_1=\{q,x_1.@x_1,...,x_m.@x_m\}$，$S_2=\{y.@l\}$，$m \ge 0$，本篇論文所定義需正規化的多值相依性是具有 $S_1 \rightarrow\rightarrow S_2 \mid S_3$ 形式的 nontrivial XMVD，其中 $S_1=\{q,x_1.@x_1,...,x_m.@x_m\}$，$S_2=\{y.@l\}$，$S_3=\{z_1.@z_1,...,z_n.@z_n\}$，$n \ge 0$，$j \ge 1$，$S_1 \cup S_2 \cup S_3 = \textit{Leafpaths}(D) \cup q$，$S_i \cap S_j = \phi$ for $i \ne j$，$q \in \textit{EPaths}(D)$。在 XML tree 裡，若是 XFD: $\{q,x_1.@x_1,...,x_m.@x_m\} \rightarrow \{y.@l\}$ 存在，$m \ge 0$，那麼 $S_1 \rightarrow\rightarrow S_2 \mid S_3$ 形式的多值相依性也會存在，其中 $S_1=\{q,x_1.@x_1,...,x_m.@x_m\}$，$S_2=\{y.@l\}$，$S_3=\{z_1.@z_1,...,z_n.@z_n\}$，$m \ge 0$，$n \ge 1$，$S_1 \cup S_2 \cup S_3 = \textit{Leafpaths}(D) \cup q$，$S_i \cap S_j = \phi$ for $i \ne j$，$q \in \textit{EPaths}(D)$。見圖 4-16，不同於 RDB 裡的相依性關係，會使 XML 文件違背 4XNF 的 XMVD 可能發生在相依性 $S_1 \rightarrow\rightarrow S_2 \mid S_3$ 與 $S_1 \rightarrow S_2$ 同時存在或不同時存在時。

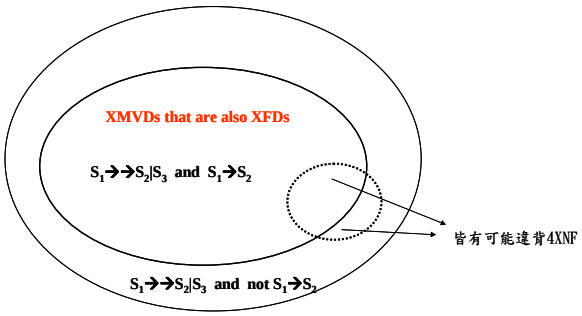


圖 4-16 狹義的的 XFD、XMVD 關係圖

圖 3-2 course.xml 和圖 4-2 newcourses.xml 同樣都具有 $S_1 \rightarrow\rightarrow S_2 \mid S_3$ 形式的多值相依性且皆無 $S_1 \rightarrow S_2$ 形式的相依性，其中 course.xml 違背 4XNF，但 newcourses.xml 符合 4XNF，可得知 $S_1 \rightarrow\rightarrow S_2 \mid S_3$ 會不會導致 XML 文件違背 4XNF 跟是否具有 $S_1 \rightarrow S_2$ 形式的相依性無關。見圖 4.17 DBLP.xml，DBLP.xml 若滿足功能相依性: DBLP.conf.issue→DBLP.conf.issue.article.@year，則此 tree 同時也會滿足多值相依性: DBLP.conf.issue→→DBLP.conf.issue.article.@year | *Leafpaths*(D)- {DBLP.conf.issue, DBLP.conf.issue.article.@year}，但是 DBLP.xml 並不符合 4XNF。會有以上異於 RDB 理論的結果，是因為 XML 第四正規化符合條件並不是由 $S_1 \rightarrow S_2$ 來決定。

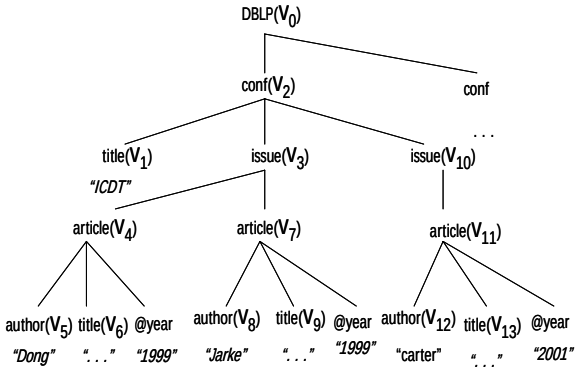
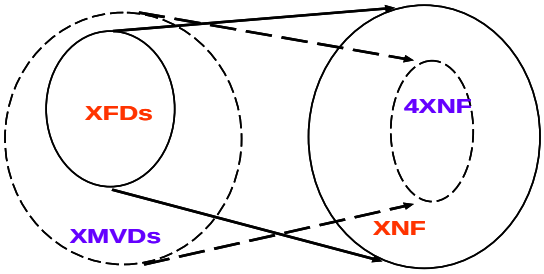


圖 4-17 DBLP.xml

最後我們說明 XNF 跟 4XNF 的關係，我們知道 RDB 中的 4NF schema 同時也是符合 BCNF，同樣的情形也會發生在 XML 上，每一個符合 4XNF 的 XML 文件也會符合 XNF。以下面說明這樣的事實，如果說存在某個 DTD *D* 不符合 XNF 的話，則存在某個 XFD $S_1 \rightarrow y.@l \in (D,\Sigma)^+$，但 $S_1 \rightarrow y \notin (D,\Sigma)^+$，因為 $S_1 \rightarrow y.@l \in (D,\Sigma)^+$ 會導出 XMVD $S_1 \rightarrow\rightarrow y.@l \mid \textit{Leafpaths}(D) - S_1 - \{y.@l\}$，path k 是 y.@l 和集合 $\textit{Leafpaths}(D) - S_1 - \{y.@l\}$ 裡任何 path 做 intersection 得來，k 一定會是 y.@l 的 strict prefix path，所以當 k=y 時，存在 $S_1 \rightarrow k \notin (D,\Sigma)^+$，使得 *D* 不符合 4XNF。換言之，一個 XML 文件若

不符合 XNF，那麼此文件也不可能是是 4XNF 的文件。見圖 4-18 表示 XNF 和 4XNF 的關係以及 XFD 和 XMVD 的關係，XFDs 導出 XNF，XMVDs 導出 4XNF。



4-18 XFDs、XMVDs、4XNF、XNF 關係圖

五、結論與未來展望

這一篇論文裡，我們延續了文獻[3]提到的 XFD 及正規化方法來更進一步定義 XML 的多值相依性，並且提出 XML tree 上的第四正規化定義，最後提出一個有效的演算法來縮減資料重複的文件。XML 不同於關聯式資料表的結構，雖然無法像關聯式資料表正規化一般來將 XML 文件切割為兩個以上的獨立 tree，但是我們卻可以藉由樹狀結構重組來做到類似的結果，XML 的文件結構是樹狀的結構，嚴格來說，在 XML 文件上的正規化動作是一種樹狀結構的重新組合，在不失去資料的前題下，我們可以任意組合 DTD 樹狀結構，以獲得能保留原來訊息且更無資料重複特性的新生樹狀結構。不過為了使演算法的效率較好，並不是任意的組合樹狀圖，而是要儘量在不變動樹狀結構情形下，轉換成新的結構。我們證明若是有一個 anomalous XMVD 會使資料違背 4XNF，經由正規化後，這一個 anomalous XMVD 將會不存在，且正規化後的文件會比正規化前擁有較少的 anomalous path。此外我們把焦點鎖在 non- trivial XMVD，而不是 trivial XMVD，是因為 trivial XMVD 是永遠發生在任何 DTD 上的，難以在 trivial XMVD 探討資訊重複，同此情形發生在 RDB 的正規化一般。

在研究的過程中，我們發現有許多未來足供發展的空間。我們將未來可以改進及研究之方向以下三點說明：

一、我們提出 4XNF 重組演算法除了可以避開異常的更新外，最直接的效益就是減少原生型 XML 文件的複雜分支(NODE 減少，語意不變)更省空間，當然也是有可能有更好的方法來節省空間，這是未來可研究的。我們節省空間的觀點是以"樹狀"結構的觀點出發(這是直覺的觀點)。有可能在節省空間方面，可跳脫"樹狀"的觀點去研究，例如實體儲存媒體觀點，有否存在一種特別技術能利用少量的儲存資訊還原大量資訊，而沒失真問題。此外 XML 正規化演算法是在不失去資訊意含內容下重組樹狀結構，但是重組的選擇並不唯一，是否有更好的組合方式來節省節點是未來可以努力的方向。

二、目前 XML 文件設計有多議題，但將 RDB 正規化過程觀點搬移到 XML 文件上探討，目前只探討到 XNF (相當 RDB 的 BCNF)，另一個是 4XNF(相當於 RDB 的 4NF)。是否有更進一步正規化的可能是可討論的。XML 文件上是否需要有合併相依性(join dependency)語意的 constraints 存在，若是需要的，那麼如何重新組合樹狀結構來使文件符合相當於 RDB 中的 PJNF 是可以努力的方向。

三、系統的實際上應用:以下有幾點是必須的

1.一個免費的 XML 原生型資料庫(這部份有很多的 open source)。

2.利用本文導出的結論以及文獻的 XNF 能幫助我們設計 XML 正規化的機器，下列為機器大致功能：

(1)已知的第四正規化語意，以及已知的 DTD、XML 文件，若文件不符合正規化，將它調整成正規化文件(XNF,4XNF)。

(2)未知 DTD 、未知 XML 文件，在設計之初，根據我們需要的語意，配合我們的結論直接設計好的 DTD 文件格式，直接效益是空間的大幅減少，更新無異常。

六、參考文獻

[1] 林穎聰,"XML 文件的多值相依性與第四正規化",中興大學資訊科學研究所,碩士論文, 2005.

[2] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web*. Morgan Kaufmann, San Francisco, CA, 2000.

[3] M. Arenas and L. Libkin. A Normal Form for XML Documents. *ACM Transactions on Databases Systems (TODS)*,29(1): pp.195 -232, 2004.

[4] P. Buneman, W. Fan, J. Simeon, and S. Weinstein. Constraints for semi-structured data and XML. *SIGMOD Record*, 30(1): pp.47-54, 2001.

[5] P. Buneman, S. Davidson, W. Fan, and C. Hara. Reasoning about keys for xml.In *International Worksshop on Database Programming Languages*, 2001.

[6] M. W. Vincent and J. Liu, Multivalued dependencies and a 4NF for XML, *in 'CAiSE 2003'* : pp.14–29.

[7] M. W. Vincent , J. Liu . and C. Liu., Strong functional dependencies and their application to normal forms in XML, *ACM Transactions on Database Systems* 29(3): pp.445–462, 2004

[8] Extensible Markup Language. http://www.w3.org/TR/REC-xml.

[9] W3C and XML. http://www.w3c.org/xml

[10] HTML 4.01 Specification. http://www.w3.org/TR/REC-html40/

附錄

定理 4.1:

$(D,\sum)$ 經由我們所提出的正規化演算法轉變成 $(D',\sum')$，則 $(D,\sum) \le_{\text{lossless}} (D',\sum')$。

證明:

[1]case1:

假設使用第一種轉換方法將 $(D,\sum)$ 轉換成 $(D',\sum')$，D' = D[y.@l:=q.τ.@l]，[y.@l:=q.τ.@l]，S_1→→ $S_2 \mid S_3$ 是一個不規則的 XMVD 。在這個例子裡，我們可以定義出一個函式 f，將 *paths*(D') 裡的 path 對應回 *paths*(D)裡的 path，f 定義如下: f(q.τ)=y，f(q.τ.@l)=y@l，令其他在 *paths*(D')裡的 path 為 p' ，則 f(p')= p' 。

[2] case2:

假設使用第二種轉換方法將 $(D,\sum)$ 轉換成 $(D',\sum')$， D' = *D*[y@l:=y. τ' .@l , $[z_1.@z_1,...,z_n.@z_n]$:= y.τ[$(z_1$-y).@z_1,… ,(z_n-y).@z_n]] $\sum'$ = $\sum$ [y@l:=y. τ' .@l , $[z_1.@z_1,...,z_n.@z_n]$:= y.τ[(z_1-y).@z_1,… ,(z_n-y).@z_n]]。

定義的函數 f : f(y. τ')= y，f(y. τ' .@l)= y.@l ，f(y.τ)=y，f(y.τ.(z_i-y))=z_i，f(y.τ.(z_i-y).@z_i)=z_i.@z_i ，f(y.τ.p)= y.p ，p 是任一路徑 z_i-y 裡的任一 prefix path，i∈[1,n]，其他在 paths(D')裡的 path 為 p' ，f(p')= p' 。

[3]case 3:

假設使用第三種轉換方法將 $(D,\sum)$ 轉換成 $(D',\sum')$，

D' =*D*[y.@l:=q.τ[τ_1.@x_1,….,τ_m.@x_m, τ' .@l]]，

$\sum'$ = $\sum$[y.@l:=q.τ[τ_1.@x_1,….,τ_m.@x_m, τ' .@l]]。定義的函數 f : f(q.τ)= $x_1 \cap x_2 \cap ... \cap x_m \cap y$，f(q.τ.$\tau_i$.@$x_i$)=$x_i$.@$x_i$，f(q.τ.$\tau_i$)=$x_i$，f(q.τ. τ' .@l)= y.@l，f(q.τ. τ')= y，其他在 paths(D')裡的 path 為 p' ， f(p')= p' 。

定理 4.2 證明:

分別證明 4XNF 重組演算法三種 case

[1]Case 1:

若是 *D* 是一個 DTD，$\sum$ 是 *D* 上的相依性集合，若是 S_1→→ $S_2 \mid S_3$ 是 $\sum$ 裡的一個 anomalous XMVD，其中 S_1={q,x_1.@x_1,...,x_m.@x_m}，S_2 ={y.@l}，S_3={z_1.@z_1,...,z_n.@z_n}，m≧0，n≧1，$S_1 \cup S_2 \cup S_3$=*Leafpaths*(D)∪q，$S_i \cap S_j = \phi$ for i≠j，，q∈*EPaths*(*D*)， q→S_1，對 S_1→→ $S_2 \mid S_3$ 正規化後，D' =*D*[y.@l:=q.τ.@l]， $\sum'$ = $\sum$ [y.@l:=q.τ.@l]，會有|AP(D' , $\sum'$)|＜|AP(*D*, $\sum$)|的性質。 □

證明: 若針對 anomalous xmvd:S_1→→y.@l | S_3 做正規化，得到新的 xmvd: S_1→→q.τ .@l | S_3，原本的 anomalous path 是 y.@l 經正規化後此 path 變成 q.τ .@l，首先證明 q.τ .@l 不會是 anomalous path (q.τ .@l∉ AP(D′,Σ′))。我們分 5 種情形討論。

(1)若存在 $S_1 \cup P$→→ q.τ .@l | S_3-P∈ $(D',\Sigma')^+$是一個 nontrivial xmvd，令 P⊂S_3，k 是 q.τ .@l 和 S_3-P 集合裡任何 path 的路徑交集 (k=*intersection*(q.τ .@l ,b),where b∈S_3-P)，因為 S_1→q，且 q→k (trivial fd)，所以 S_1→k，並得 $S_1 \cup P$→k，因此 q.τ .@l∉ AP(D′,Σ′)。

(2)另外若 S_1-Q→→ q.τ .@l | $S_3 \cup Q$∈ $(D',\Sigma')^+$是一個 nontrivial xmvd，令 Q⊂S_1，若 q.τ .@l 是 anomalous path，則存在 k 是 q.τ .@l 和 $S_3 \cup Q$ 集合裡某 path 的路徑交集 (k=*intersection*(q.τ .@l ,b), where b∈$S_3 \cup Q$)，使得 S_1-Q→k∉ $(D',\Sigma')^+$，因為 S_1→q，q→k(trivial fd)，所以 S_1→k，且因假設 S_1-Q→k∉ $(D',\Sigma')^+$，可知必定 Q→k，$S_3 \cup Q$→k。存在一個 XML tree $T' \models (D',\Sigma')$， S_1-Q→k∉ $(D',\Sigma')^+$和 $S_3 \cup Q$→k∈ (D',Σ') 的條件同時存在，會使 T' 不滿足 S_1-Q→→ q.τ .@l | $S_3 \cup Q$，這造成了矛盾，故這種型情下 q.τ .@l∉ AP(D′,Σ′)。

(3)若 S_3→→q.τ .@l | S_1∈ $(D',\Sigma')^+$是 anomalous xmvd，存在 k 是 q.τ .@l 和 S_1 集 合裡某個 path 的路徑交集(k=*intersection*(q.τ .@l ,b),where b∈ S_1)，且 S_3→k∉ $(D',\Sigma')^+$，因為 S_1→q，q→k(trivial fd)，所以 S_1→k。S_3→k∉ $(D',\Sigma')^+$ 以及 S_1→k∈ $(D',\Sigma')^+$同時存在會與 S_3→→q.τ .@l | S_1∈ $(D',\Sigma')^+$矛盾，故 S_3→→q.τ .@l | S_1∈ $(D',\Sigma')^+$不是 anomalous xmvd，q.τ .@l ∉ AP(D′,Σ′)。

(4)已知(1) $S_1 \cup P$→→ q.τ .@l | S_3-P 不是 anomalous xmvd，P⊂S_3，同(3)證法，可證 S_3-P→→ q.τ .@l | $S_1 \cup P$ 也不是 anomalous xmvd，q.τ .@l∉ AP(D′,Σ′)。

(5)已知(2) S_1-Q→→ q.τ .@l | $S_3 \cup Q$ 不是 anomalous xmvd，Q⊂S_1，同(3)證法，可證 $S_3 \cup Q$→→ q.τ .@l | S_1-Q 也不是 anomalous xmvd，q.τ .@l∉ AP(D′,Σ′)。

其次我們證明|AP(D' , $\sum'$)|≦|AP(*D*, $\sum$)|:

假設 $S_1 \cup S_2 \subseteq$ *paths*(D')-{q.τ .@l}，由 Σ' 定義得知

若 $(D,\Sigma) \vdash S_1$→→$S_2 \mid$ *Leafpaths*(*D*)- S_1-S_2

則 $(D',\Sigma') \vdash S_1$→→$S_2 \mid$ *Leafpaths*(D')-S_1-S_2

假設 $(D,\Sigma) \not\vdash S_1$→→$S_2 \mid$ *Leafpaths*(*D*)- S_1-S_2，則存在一個 XML tree T ⊨ (D,Σ)，但是 T⊭ S_1→→$S_2 \mid$ *Leafpaths*(*D*)- S_1-S_2，定義一個 XML tree $T' \models (D',\Sigma')$ 然而 $T' \not\models S_1$→→$S_2 \mid$ *Leafpaths*(D')-S_1-S_2，因此 $(D',\Sigma') \not\vdash S_1$→→$S_2 \mid$ *Leafpaths*(D')-S_1-S_2

以上得證|AP(D' , $\sum'$)|≦|AP(*D*, $\sum$)|，但我們知道 y.@l∈AP(*D*, $\sum$)且 y.@l 和 q.τ .@l∉ AP(D′,Σ′)，所以總結|AP(D' , $\sum'$)|<|AP(*D*, $\sum$)|。

[2]Case 2:

若是 D 是一個 DTD，$\sum$ 是 D 上的相依性集合，若是 S_1→→ $S_2 \mid S_3$ 是 $\sum$ 裡的一個 anomalous XMVD，其中 S_1={q,x_1.@x_1,...,x_m.@x_m}，S_2 ={y.@l}，S_3={z_1.@z_1,...,z_n.@z_n}，*m*≧*1*，*n*≧*1*，*q*∈*EPaths*(D)，若是具有以下情形時:

1. 任何(a,b)∈S_2×S_3，*intersection*(a,b)=y
2. 若 S_1 裡最長的路徑是 p，*intersection*(y,p)=*parent*(p)

在 y 底下建立兩個新的子節點(element)τ，τ'，移動@l 到τ'的屬性集合裡。原在 S_3，以 y 為 root 的根節點的路徑，全部轉變成以 τ 為根節點。經正規化後

D' = D[y@l:=y.τ'.@l , [z_1.@z_1,...,z_n.@z_n] := y.τ[(z_1-y).@z_1,… ,(z_n-y).@z_n]]，

$\sum'$ = $\sum$ [y@l:=y.τ'.@l , [z_1.@z_1,...,z_n.@z_n] := y.τ[(z_1-y).@z_1,… ,(z_n-y).@z_n]]，會有 |AP(D',$\sum'$)| <|AP(D,$\sum$)| 的性質。 □

證明： 若針對 anomalous xmvd:S_1→→y.@l | S_3 做正規化，得到新的 xmvd: S_1→→y.τ'.@l | S3′，S3′ 是 S_3 裡的 paths 依據 D' 轉換，原本的 anomalous path 是 y.@l 經正規化後此 path 變成 y.τ'.@l，首先證明 y.τ'.@l 不會是 anomalous path (y.τ'.@l∉ AP(D′,Σ′))。

分 5 種情形討論。

(1) 若存在 S_1∪P→→ y.τ'.@l | S3′-P∈ $(D',\Sigma')^+$ 是一個 nontrivial xmvd，令 P⊂ S3′，k 是 y.τ'.@l 和 S3′-P 集合裡任何 path 的路徑交集(k=*intersection*(y.τ'.@l ,b),where b∈S3′-P)，因為 S_1→y，且 y→k (trivial fd)，所以 S_1→k，並得知S_1∪P→k，因此y.τ'.@l∉ AP(D′,Σ′)。

(2)另外若 S_1-Q→→ y.τ'.@l | S3′ ∪Q∈ $(D',\Sigma')^+$ 是一個 nontrivial xmvd，Q⊂S_1，k 是 y.τ'.@l 和 S3′ ∪Q 集合裡某 path 的路徑交集(k=*intersection*(y.τ'.@l ,b), where b∈S3′ ∪Q)，若 y.τ'.@l 是 anomalous path，則 S_1-P→k∉ $(D',\Sigma')^+$，因為 S_1→y，y→k(trivial fd)，所以 S_1→k，且因 S_1-P→k∉ $(D',\Sigma')^+$，可知必定 P→k，S3′ ∪P→k。存在一個 XML tree $T' \models (D',\Sigma')$，因為 S_1-Q→k∉ $(D',\Sigma')^+$同時 S3′ ∪Q→k 也存在的事實會使T'不滿足 S_1-Q→→ y.τ'.@l | S3′ ∪Q，這造成了矛盾，故這種型情下，y.τ'.@l ∉ AP(D′,Σ′)。

(3) 假設 S3′ →→y.τ'.@l | S_1∈ $(D',\Sigma')^+$是 anomalous xmvd，若 S_3→→q.τ .@l | S_1 ∈ $(D',\Sigma')^+$是 anomalous xmvd，存在 k 是 y.τ'.@l 和 S_1 集合裡某個 path 的路徑交集(k=*intersection*(y.τ'.@l ,b),where b∈ S_1)，且 S3′ →k∉ $(D',\Sigma')^+$，因為 S_1→y，y→k(trivial fd)，所以 S_1→k。S3′ →k∉ $(D',\Sigma')^+$以及 S_1→k∈ $(D',\Sigma')^+$同時存在會與 S3′ →→ y.τ'.@l | S_1∈ $(D',\Sigma')^+$矛盾，故 S3′ →→ y.τ'.@l | S_1∈ $(D',\Sigma')^+$不是 anomalous xmvd，y.τ'.@l∉ AP(D′,Σ′)。

(4) 同理可證 S3′ -P →→ y.τ'.@l | S_1∪P ∈ $(D',\Sigma')^+$不是 anomalous xmvd where P⊂ S3′ ，y.τ'.@l∉ AP(D′,Σ′)

(5) 同理可證 S3′ ∪Q →→ y.τ'.@l |S_1-Q ∈ $(D',\Sigma')^+$不是 anomalous xmvd where Q⊂S_1，y.τ'.@l∉ AP(D′,Σ′)

其次我們證明|AP(D',$\sum'$)|≦|AP(D,$\sum$)|:

由Σ'定義可得知，$\sum'$ 包含所有在$\sum$ 中的相依性，但$\sum$ 中的 y@l 變成 y.τ'.@l，z_i.@z_i 變成 y.τ(z_i-y)@z_i，z_i 變成 y.τ.(z_i-y)。假設 S_1∪S_2⊆*paths*(D)-{y.@l}，

若 $(D,\Sigma) \vdash S_1$→→S_2 |*Leafpaths*(D)-S_1-S_2，則 $(D',\Sigma') \vdash$ S1′→→S2′ | *Leafpaths*(D')-S1′-S2′ ，此處 S1′、S2′ 是由 S_1、S_2 轉變而來(y.@l 變成 y.τ'.@l，z_i.@z_i 變成 y.τ.(z_i-y).@z_i，z_i 變成 y.τ.(z_i-y))。

假設 $(D,\Sigma) \nvdash$ S_1→→S_2 | *Leafpaths*(D)-S_1-S_2，則存在一個 XML tree T $\models (D,\Sigma)$ 但是 T⊭S_1→→S_2 | *Leafpaths*(D)-S_1-S_2，定義一個 XML tree $T' \models (D',\Sigma')$，

然而$T' \not\models$ S1′→→S2′ | *Leafpaths*(D')-S1′-S2′ ，

因此 $(D',\Sigma') \nvdash$ S1′→→S2′ | *Leafpaths*(D')-S1′-S2′ 。

以上得證|AP(D',$\sum'$)|≦|AP(D,$\sum$)|，我們知道 y.@l∈AP(D,$\sum$)，但是 y.@l 和 y.τ'.@l∉ AP(D′,Σ′)，所以總結 |AP(D',$\sum'$)|<|AP(D,$\sum$)|。

[3]Case 3:

若是 D 是一個 DTD，$\sum$ 是 D 上的相依性集合，若 S_1→→ S_2 | S_3 是$\sum$ 裡的一個 minimal anomalous XMVD，其中 S_1={q,x_1.@x_1,...,x_m.@x_m}，S_2={y.@l}，S_3={z_1.@z_1,...,z_n.@z_n}，m≧1，n≧1，S_1,S_2,S_3 ⊂*paths*(**D**)，

S_1∪S_2∪S_3=*Leafpaths*(D)∪q，S_i∩S_j=ϕ for i≠j，q∈*EPaths*(D)，經正規化後

D' =D[y.@l:=q.τ[τ_1.@x_1,…. ,τ_m.@x_m, τ'.@l]]，

$\sum'$ = $\sum$ [y.@l:=q.τ[τ_1.@x_1,…. ,τ_m.@x_m, τ'.@l]]，會有 |AP(D',$\sum'$)| <|AP(D,$\sum$)| 的性質。 □

證明： 首先證明 q.τ.τ_i.@x_i 不會是 anomalous path，其中 i∈[1,m]，若存在一個 xmvd：

S'→→q.τ.τ_i.@x_i | *Leafpaths*(D')-S'-{q.τ.τ_i.@x_i} 屬於$(D',\Sigma')^+$，$S' \subset$*paths*(D)，令 k 是 q.τ.τ_i.@x_i 和 *Leafpaths*(D')-S'-{q.τ.τ_i@x_i}集合裡任何 path 的路徑交集(k=*intersection*(q.τ.τ_i.@x_i ,b), where b∈*Leafpaths*(D')-S'-{q.τ.τ_i.@x_i}，令 S'=S_1∪S_2 ，where

(1) S_1∩({q, q.τ.τ'.@l}∪{ q.τ.τ_j | j∈[1,m]}∪ {q.τ.τ_j.@x_j | j∈[1,m] and j≠i})=ϕ

(2) S_2⊆ {q, q.τ.τ'.@l}∪{ q.τ.τ_j| j∈[1,m]}∪ {q.τ.τ_j.@x_j | j∈[1,m] and j≠i}。

根據Σ'的定義以及{q,x_1.@x_1,...,x_m.@x_m}→→y.@l | {z_1.@z_1,...,z_n.@z_n}是 minimal，其中 $m \geqq 1, n \geqq 1$ 。下面三個的其中之一是成立的:

(1) S_2→→ q.τ.τ_i.@x_i 不是 anomalous XMVD

(2){q, q.τ.τ_1.@x_1,..., q.τ.τ_m.@x_m, q.τ.τ'.@l}= S_2∪{q.τ.τ_i.@x_i}

(3){q.τ.τ_i, q.τ.τ_1.@x_1,..., q.τ.τ_m.@x_m, q.τ.τ'.@l} =S_2∪{q.τ.τ_i.@x_i}

在(1)裡，q.τ.τ_i.@x_i∉AP(D',Σ') 。

在(2)(3)裡，因為{q, q.τ.τ_1.@x_1,..., q.τ.τ_m.@x_m}→ q.τ ，所以 S_2∪{q.τ.τ_i.@x_i}→ q.τ，q.τ→k (trivial fd)，得到 S_2∪{q.τ.τ_i.@x_i}→k。

若是 $q.\tau.\tau_i.@x_i$ 不是 anomalous path 的話
則 $S_1 \cup S_2 \rightarrow k$ 要存在，只要能証 $S_2 \rightarrow k$ 存在即可。
假設 $q.\tau.\tau_i.@x_i \rightarrow k \in (D', \Sigma')^+$，存在一個 XML tree $T' \models (D', \Sigma')$，$t_1, t_2 \in T(D')$，
$t_1(q.\tau.\tau_i.@x_i) = t_2(q.\tau.\tau_i.@x_i)$則 $t_1(k) = t_2(k)$，因為在 Σ' 不含 $q.\tau.\tau_i.@x_i \rightarrow S'$，所以令 $t_1(S') \neq t_2(S')$，此會與 $S' \rightarrow\rightarrow q.\tau.\tau_i.@x_i \mid leafpaths(D') - S' - \{q.\tau.\tau_i.@x_i\}$ 矛盾，故 $q.\tau.\tau_i.@x_i \rightarrow k \notin (D', \Sigma')^+$。
因已證明 $S_2 \cup \{q.\tau.\tau_i.@x_i\} \rightarrow k$，且 $q.\tau.\tau_i.@x_i \rightarrow k \notin (D', \Sigma')^+$，所以 $S_2 \rightarrow k \in (D', \Sigma')^+$，
$S_1 \cup S_2 \rightarrow k \in (D', \Sigma')^+$，得證 $q.\tau.\tau_i.@x_i \notin AP(D', \Sigma')$
利用類似證法可得證 $q.\tau.\tau'.@l \notin AP(D', \Sigma')$
其次我們證明$|AP(D', \Sigma')| \leq |AP(D, \Sigma)|$:
$S_3 \cup S_4 \subseteq paths(D) - \{y.@l\}$
由 Σ' 定義得知:
若 $(D, \Sigma) \vdash S_3 \rightarrow\rightarrow S_4 \mid Leafpaths(D) - S_3 - S_4$
則 $(D', \Sigma') \vdash S_3 \rightarrow\rightarrow S_4 \mid Leafpaths(D') - S_3 - S_4$
假設 $(D, \Sigma) \nvdash S_3 \rightarrow\rightarrow S_4 \mid Leafpaths(D) - S_3 - S_4$，則存在一個 XML tree $T \models (D, \Sigma)$ 但是 $T \not\models S_3 \rightarrow\rightarrow S_4 \mid Leafpaths(D) - S_3 - S_4$，定義一個 XML tree $T' \models (D', \Sigma')$ 然而
$T' \not\models S_3 \rightarrow\rightarrow S_4 \mid Leafpaths(D') - S_3 - S_4$，
因此 $(D', \Sigma') \nvdash S_3 \rightarrow\rightarrow S_4 \mid Leafpaths(D') - S_3 - S_4$
以上得證$|AP(D', \Sigma')| \leq |AP(D, \Sigma)|$，但我們知道 $y.@l \in AP(D, \Sigma)$且 $y.@l \notin AP(D', \Sigma')$，所以總結 $|AP(D', \Sigma')| < |AP(D, \Sigma)|$。