2009 First Asian Conference on Intelligent Information and Database Systems

Deriving Conceptual Schema from XML Databases

Oviliani Yenty Yuliana[1], Suphamit Chittayasothorn[2]
[1]*Department of Informatics Engineering, Petra Christian University, Surabaya, Indonesia*
[2]*Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Thailand*
ovi@peter.petra.ac.id, suphamit@kmitl.ac.th

Abstract

In this paper, two concepts from different research areas are addressed together, namely functional dependency (FD) and multidimensional association rule (MAR). FD is a class of integrity constraints that have gained fundamental importance in relational database design. MAR is a class of patterns which has been studied rigorously in data mining. We employ MAR to mine the interesting rules from XML Databases. The mined interesting rules are considered as candidate FDs whose all confidence itemsets are 100%. To prune the weak rules, we pay attention to support and correlation itemsets. The final strong rules are used to generate an Object-Role Model conceptual schema diagram.

1. Introduction

In recent years, XML [1] has emerged as the dominant standard for representing and exchanging data over the Internet. However, there are many existing XML Databases which have been implemented without schemas and consistency checking. As a result, XML Databases may be inconsistent, incomplete or difficult to maintain. Such systems are also normally poor documented. These problems can be overcome by having XML Schemas which can be created using conceptual modeling techniques.

XML Schemas design using the Object-Role Modeling (ORM) as its conceptual schema are conducted by [2,3,4]. In addition, [4] captured all of ORM constraints that are still not defined in [1] and used XQueries to detect invalid constraints in XML Databases. Furthermore, [5] proposed reengineering the existing XML Databases using a conceptual schema approach.

At present, more and more scholars conduct research on association rules mining and many methods are based on the Apriori Algorithm that was proposed by [6]. Single Association Rule Mining from XML Data was conducted by [7]. Moreover, [8] studied extracting Association Rules from XML Documents using XQuery [9]

In this paper, we employ Multidimensional Association Rule (MAR) to improve the ORM reverse engineering which was proposed by [5]. So far, deriving conceptual schemas from XML Databases using MAR has not been addressed. MAR is used for mining the interesting rules, i.e. candidate FDs, which all confidence itemsets are 100%. Usually FDs are based on superkey [10,11]. XQueries are applied for calculating confidence, support, and correlation itemsets. To prune the weak rules support and correlation itemsets are considered.

2. Basic concepts and notation

2.1. Functional dependencies

Let R be a relation schema. A subset K of R is a superkey of R if, in any legal relation $r(R)$, for all pairs t_1 and t_2 of tuples in r such that $t_1 \neq t_2$, then $t_1[K] \neq t_2[K]$. That is, no two tuples in any legal relation $r(R)$ may have the same value on attribute set K.

The notation of functional dependency generalizes the notion of superkey [10]. Let $\alpha \subseteq R$ and $\beta \subseteq R$. The functional dependency $\alpha \rightarrow \beta$ holds on R if, in any relation $r(R)$, for all pairs of tuples t_1 and t_2 in r such that $t_1[\alpha]=t_2[\alpha]$. It is also the case that $t_1[\beta]=t_2[\beta]$. Using the functional-dependency notation, K is a superkey of R if $K \rightarrow R$. That is, K is a superkey if, whenever $t_1[K]=t_2[K]$, it is also the case that $t_1[R]=t_2[R]$, that is $t_1=t_2$. In this paper, we will mine FDs using the elementary facts concept which was proposed by [12].

2.2. Object-role modeling

ORM is called "fact-oriented modeling" because it expresses the information in terms of simple/ elementary facts. An elementary fact is defined by [12]

978-0-7695-3580-7/09 $25.00 © 2009 IEEE
DOI 10.1109/ACIIDS.2009.13

as an assertion that an object has a property, or that one or more objects participate in a relationship, where the fact cannot be split into simpler facts with the same object types without information loss.

In order to reverse engineering XML Databases into ORM conceptual schema, our concern is the second step of the conceptual schema design procedure, i.e. draw the fact types and apply a population check. In this paper, fact types are populated with fact instances from XML Databases using XQuery. For population checking, we propose an application that implements the MAR concept.

2.3. Multidimensional association rules

Let $J=\{i_1, i_2, \ldots, i_m\}$ be a set of items. Let D, the task-relevant data, be a set of database transactions where each transaction T is a set of items such that $T \subseteq J$. Let A be a set of items. A transaction T is said to contain A if and only if $A \subseteq T$. [13] defined an association rule as an implication of the form $A \Rightarrow B$, that is, $A_1 \wedge \ldots \wedge A_m \rightarrow B_1 \wedge \ldots \wedge B_n$, where $A \subseteq J$, $B \subseteq J$, and $A \cap B = \emptyset$. Association rules that involve two or more dimensions or predicates can be referred to as Multidimensional Association Rules, for instance: age(A,"20…29")∧income(A,"20K…29K")⇒buys(X," CD player").

The rule $A \Rightarrow B$ holds in the transaction set D with support s, where s is the percentage of transactions in D that contain $A \cup B$, i.e. both A and B. This is taken to be the probability, $P(A \cup B)$. The rule $A \Rightarrow B$ has confidence c in the transaction set D if c is the percentage of transactions in D containing A that also contain B. This is taken to be the conditional probability, $P(B|A)$.

$$\text{Support } (A \Rightarrow B) = P(A \cup B) = \frac{P(A \cup B)}{P(U)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

$$\text{Confidence } (A \Rightarrow B) = P(B|A) = \frac{P(A \cup B)}{P(A)} \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

Association Rules mined using a support-confidence framework are useful for many application. However, the support-confidence framework can be misleading in that it may identify a rule $A \Rightarrow B$ as interesting when the occurrence of A does not imply the occurrence of B. The occurrence of itemset A is independent of the occurrence of itemset B if $P(A \cup B)=P(A)P(B)$; otherwise itemsets A and B are dependent and correlated as events.

$$\text{Correlation A,B} = \frac{P(A \cup B)}{P(A)P(B)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

If the resulting value of (3) is less than 1, then the occurrence of A is negatively correlated with the occurrence of B. If the resulting value is greater than 1, then A and B are positively correlated, meaning the occurrence of one implies the occurrence of the other. If the resulting value is equal to 1, then A and B are independent and there is no correlation between them.

In this paper, a confidence value is used for mining the rule $A \Rightarrow B$ as the interesting FDs. If and only if all confidences in itemsets are 100% (see FD definition on section 2.1) then the rules $A \Rightarrow B$ is an interesting FDs. For FD purpose, we also mine the rule $B \Rightarrow A$. In addition, support and correlation are used to prune the interesting rules when more than one determinant determines the same dependant.

3. Mining multidimensional association rules from XML databases

A framework for deriving conceptual schema from XML Databases is shown in Figure 1. There are three main processes, i.e. modify XML List, mine interesting rules, and prune weak rules. The first process is to create XML List from XML Databases then generate itemsets to be mined by the second process. Create the rules (FDs) which are to be pruned in the third process and get inputs from the second and the third processes to update or delete the XML List elements.

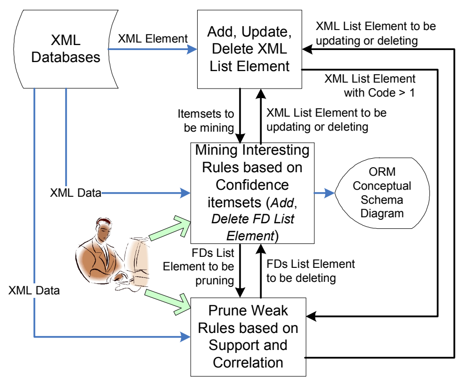

XML Databases
XML Element
Add, Update, Delete XML List Element
XML List Element to be updating or deleting
XML List Element with Code > 1
Itemsets to be mining
XML List Element to be updating or deleting
Mining Interesting Rules based on Confidence itemsets (Add, Delete FD List Element)
ORM Conceptual Schema Diagram
XML Data
FDs List Element to be pruning
FDs List Element to be deleting
XML Data
Prune Weak Rules based on Support and Correlation

Figure1. The deriving conceptual schema from XML databases framework

The second process is to mine interesting rules by joining and calculating confidence itemsets using XQuery. Figure 2(a) and Figure 2(b) show XQueries for 2-itemsets and 3-itemsets respectively. The XQuery can be extended for further itemsets. The XQueries implement the confidence equation (2). The itemset is an FD, if and only if all confidences of the itemset are 100% then save the itemset into an FD List.

The last process, prune weak rules (FDs) when XML List elements are refered more than one with considering average support and correlation itemsets.

Support equation (1) and correlation equation (3) are implemented on XQuery in Figure 2(c). Prune the lowest support FDs with the correlation FDs greater than or lower than one. The final FD List is used to generate an ORM conceptual schema diagram.

```
declare variable $Tbl:=
  doc("SPJ.xml")/SPJ/SupplierPartProject;
<ItemSet-2>
 {for $C1 in distinct-values($Tbl/ Ⓐ)
  let $C1C:= count($Tbl [Ⓐ =$C1])
  for $C2 in distinct-values($Tbl/ Ⓑ)
  let $p:=$Tbl [Ⓐ =$C1 and  Ⓑ =$C2]
  order by $C1,$C2
  return
  if (exists($p)) then
    <Frequent>
      <A>{$C1}</A>
      <B>{$C2}</B>
      <Sup.Count>{count($p)}</Sup.Count>
      <Confidence>{count($p) div $C1C *
        100}%</Confidence>
    </Frequent>
  else ()}
</ItemSet-2>
```

(a) Confidence A⇒B

```
declare variable $Tbl :=
  doc("SPJ.xml")/SPJ/SupplierPartProject;
<ItemSet-3>
 {for $C1 in distinct-values($Tbl/ Ⓐ)
  for $C2 in distinct-values($Tbl/ Ⓑ)
  let $C1C:= count($Tbl [Ⓐ =$C1 and  Ⓑ =$C2])
  for $C3 in distinct-values($Tbl/ Ⓒ)
  let $p:=$Tbl [Ⓐ =$C1 and Ⓑ =$C2 and
   Ⓒ =$C3]
  order by $C1, $C2, $C3
  return
  if (exists($p)) then
    <Frequent>
      <A>{$C1}</A>  <B>{$C2}</B>  <C>{$C3}</C>
      <Sup.Count>{count($p)} </Sup.Count>
      <Confidence>{count($p)div$C1C*100}%</Confidence>
    </Frequent>
  else ()}
</ItemSet-3>
```

(b) Confidence A^B⇒C

```
declare variable $Tbl :=
doc("SPJ.xml")/SPJ/SupplierPartProject;
<SupCor>
 {let $CC:=count($Tbl )
  for $C1 in distinct-values($Tbl/ Ⓐ)
  let $C1C:=count($Tbl [Ⓐ =$C1])
  for $C2 in distinct-values($Tbl/ Ⓑ)
  let $C2C:=count($Tbl [Ⓑ =$C2])
  let $p:=$Tbl [Ⓐ =$C1 and Ⓑ =$C2]
  order by $C1, $C2
  return
  if (exists($p)) then
    <Frequent>
      <A>{$C1}</A>
      <B>{$C2}</B>
      <Support>{count($p) div $CC*100}%</Support>
      <Correlation>{count($p) div ($C1C*$C2C)}  </Correlation>
    </Frequent>
  else ()}
</SupCor>
```

(c) Support & Correlation rule A⇒B

Figure 2. XQueries for the calculation of confidence, support, and correlation itemsets

Three struct data types in Figure 3, i.e. EXML, EDT, and EDP are proposed to support the framework. EXML is used in XML List for storing the third level XML elements in XML Databases. There are three possibilities value in XML→Code, i.e. -1 (the element is referred to as determinant), 0 (the element is not referred), >=1 (the element is referred at least one). EDT and EDP are used for constructing a FD List, i.e. DT List and DP List. DP List is used for storing dependent items. DP→next is used for linking between dependents. Set the last DP→next to Nil. DT List is used for storing determinant items that can be single or composite item. In case composite determinant, DT→nextT is used for linking between determinants. DT→nextB is used for linking DT elements with DP elements. In addition DT→top and DT→bottom is used for linking between DT elements. In case, no element in the top or bottom DT element than set DT→top or DT→ bottom to Nil.

```
Struct EXML{
      string Name;
      int Code;
      EXML* next;
}
```

```
Struct EDT{
      EDT* top;
      String Name;
      EDT* nextT;
      EDT* nextB;
      EDT* bottom;
}
```

```
Struct EDP{
      string Name;
      EDP* next;
}
```

Figure 3. The proposed struct data type

An algorithm for mining and pruning the interesting FDs is as follows:

Step 1: Create an XML List. For every element on the third level element in XML Databases, create an element in XML List. The XML List is a guide for generating itemsets, start from 2-itemsets, 3-itemsets, etc. until all XML→Code≠0. For generating n-itemsets, n array of pointers are needed, namely XML[i] for i=0, …, n-1. Set pointer XML[0] to the first element XML List and set pointer XML[1] to XML[0]→next. Go to step 3.

Step 2: Move pointer XML[0]. Set pointer PXML to XML[0]→next. If PXML→next≠Nil and still have XML→Code=0 then move pointer XML[0] to PXML. Set pointer XML[1] to XML[0]→next. Otherwise go to step 8 for pruning the weak rules.

Step 3: Mine interesting rules 2-itemsets. Set false to Result1 and Result2. Pass XML[0]→Name and XML[1]→Name into the XQuery in Figure 2(a) for calculating confidence. If all confidences are 100% then set true to Result1. In addition, calculate confidence by passing XML[1]→Name and XML[0]→Name into the XQuery. If all confidences are 100% then set true to Result2.

Step 4: If both Result1 and Result2 are true, it means XML[0] element is a determinant of XML[1] element or the XML[1] element is also a determinant of XML[0]. Ask the user to choose the right one. If the user chooses XML[1] element is a determinant of XML[0] element then go to step 7. Otherwise, check if XML[0]→Code=0 then create a DT element and fill DT→Name with XML[0]→Name. Create a DP element and fill DP→Name with XML[1]→Name. Moreover, set XML[0]→Code with -1. If XML[1]→next≠nil then set PXML to XML[1] and move XML[1] to PXML→next. Furthermore, delete the element PXML and go to step 3.

Step 5: If both Result1 and Result2 are false, If XML[1]→next≠nil then set pointer PXML to XML[1], move pointer XML[1] to PXML→Next, and go to step

3. Otherwise go to step 2.

Step 6: If Result1 is true, it means XML[0] element is a determinant of XML[1] element. If XML[0]→Code=0, it means that XML[0] is not in the DT List then create a DT element and fill DT→Name with XML[0]→Name. Create a DP element and fill the DP→name with XML[1]→Name. Furthermore, set XML[0]→Code with -1 and increment XML[1]→Code. If XML[1]→next≠nil then set pointer PXML to XML[1], move pointer XML[1] to PXML→Next, and go to step 3. Otherwise go to step 2.

Step 7: If only Result2 is true, it means the XML[1] element is a determinant of XML[0] element. If XML[1]→Code=0 then create a DT element and fill DT→Name with XML[1]→Name. Create a DP element and fill DP→Name with XML[0]→Name. Set XML[1]→Code with -1. If XML[0]→Code≠-1 then increment XML[0]→Code else set 1 to XML[0]→Code. Go to step 2.

Step 8: Prune the weak rules. This step is used for deleting XML List elements witch codes are equal or greater to 1. If XML→Code is greater than 1 then find all elements in DP List witch DP→Name are the same with XML→Name. Calculate support and correlation for the itemsets using XQuery in Figure 2(c). Find the highest support and correlation approximately to one for keeping the itemsets in FD List and delete the others. In case there are several itemsets with the same support and correlation, than ask the user to choose only one itemsets. If no XML→Code=0 then go to step 10.

Step 9: Mine interesting rules for itemsets more than 2 items. This step is used for mining interesting rules with composite determinant, start from 3-itemsets. It means n-1 combination XML List elements with code -1 as a determinant and one element with code 0 as dependent. For calculating confidence all 3-itemsets use the XQuery in Figure 2(b) go to step 8.

Step 10: Generate ORM conceptual schema from FD List using the algorithm that proposed by [5].

4. A case study

We use a Suppliers-Parts-Projects case study that is used by [10] for demonstrating the normalization technique. In our work, an input XML document is SPJ.XML that is shown in the left side of Figure 4. The document is too long to fit in the paper (includes 24 SupplierPartProject elements and every element include 13 other elements), so we visualize the document as a table in the right side of Figure 4.

To reverse XML Databases into ORM conceptual schema, we demonstrate the proposed algorithm. It is guaranteed that relational schemas based on the created ORM will become in the fifth normal form. The created XML List from SPJ.XML by step 1 is shown in Figure 5(a). Every EXML→Code=0.

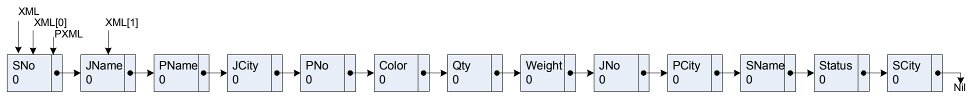
```
<?xml version="1.0" encoding="utf-8" ?>
<SPJ> → the first level element
  <SupplierPartProject> → the second level element
    <SNo>S1</SNo> → the third level element
    <JName>Sorter</JName>
    <PName>Nut</PName>
    <JCity>Paris</JCity>
    <PNo>P1</PNo>
    <Color>Red</Color>
    <Qty>200</Qty>
    <Weight>12</Weight>
    <JNo>J1</JNo>
    <PCity>London</PCity>
    <SName>Smith</SName>
    <Status>20</Status>
    <SCity>London</SCity>
  </SupplierPartProject>
. . . etc
</SPJ>
```

SNo	JName	PName	JCity	PNo	Color	Qty	Weight	JNo	PCity	SName	Status	SCity
S1	Sorter	Nut	Paris	P1	Red	200	12	J1	London	Smith	20	London
S1	Console	Nut	Athens	P1	Red	700	12	J4	London	Smith	20	London
S2	Sorter	Screw	Paris	P3	Blue	400	17	J1	Rome	Jones	10	Paris
S2	Display	Screw	Rome	P3	Blue	200	17	J2	Rome	Jones	10	Paris
S2	OCR	Screw	Athens	P3	Blue	200	17	J3	Rome	Jones	10	Paris
S2	Console	Screw	Athens	P3	Blue	500	17	J4	Rome	Jones	10	Paris
S2	RAID	Screw	London	P3	Blue	600	17	J5	Rome	Jones	10	Paris
S2	EDS	Screw	Oslo	P3	Blue	400	17	J6	Rome	Jones	10	Paris
S2	Tape	Screw	London	P3	Blue	800	17	J7	Rome	Jones	10	Paris
S2	Display	Cam	Rome	P5	Blue	100	12	J2	Paris	Jones	10	Paris
S3	Sorter	Screw	Paris	P3	Blue	200	17	J1	Rome	Blake	30	Paris
S3	Display	Screw	Rome	P4	Red	500	14	J2	London	Blake	30	Paris
S4	OCR	Cog	Athens	P6	Red	300	19	J3	London	Clark	20	London
S4	Tape	Cog	London	P6	Red	300	19	J7	London	Clark	20	London
S5	Display	Bolt	Rome	P2	Green	200	17	J2	Paris	Adams	30	Athens
S5	Console	Bolt	Athens	P2	Green	100	17	J4	Paris	Adams	30	Athens
S5	RAID	Cam	London	P5	Blue	500	12	J5	Paris	Adams	30	Athens
S5	Tape	Cam	London	P5	Blue	100	12	J7	Paris	Adams	30	Athens
S5	Display	Cog	Rome	P6	Red	200	19	J2	London	Adams	30	Athens
S5	Console	Nut	Athens	P1	Red	100	12	J4	London	Adams	30	Athens
S5	Console	Screw	Athens	P3	Blue	200	17	J4	Rome	Adams	30	Athens
S5	Console	Screw	Athens	P4	Red	800	14	J4	London	Adams	30	Athens
S5	Console	Cam	Athens	P5	Blue	400	12	J4	Paris	Adams	30	Athens
S5	Console	Cog	Athens	P6	Red	500	19	J4	London	Adams	30	Athens

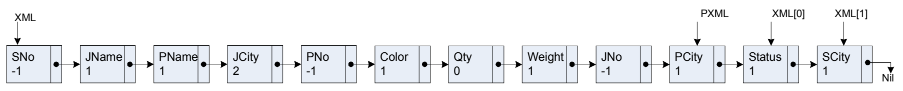
Figure 4. Part of SPJ XML database and visualize SPJ XML database as a table

Mine 2-itemsets ($A \Rightarrow B$) by passing 2 items {SNo, JName}, {SNo, PName}, {SNo, JCity}, {SNo, PNo}, {SNo, Color}, {SNo, Weigh}, {SNo, JNo}, {SNo, PCity}, {SNo, SName}, {SNo, Status}, {SNo, SCity}, {JName, PName}, {JName, JCity}, …, {JName, SCity}, …, {Status, SCity} to the XQuery in Figure 2(a) in sequence for calculating confidence. For every sequence, also calculate confidence by exchanging the item in the itemsets, for instance {*SNo*, *SCity*} and {*SCity*, *SNo*}. The confidence itemsets {*SNo*, *SCity*} and {*SCity*, *SNo*} are shown in Figure 6. All confidences for itemsets {*SNo*, *SCity*} are 100% so assign true to Result1. It means rule *SNo*⇒*SCity* is an interesting FD rule. However, all confidences for itemsets {*SCity*, *SNo*} are not 100% so assign false Result2. If only Result1 is true, than store item SNo into DT List and store item SCity into DP List. An example for only Result2 is true, when mining a rule

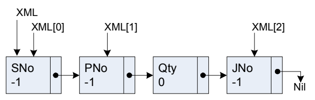
PName⇒PNo and a rule *PNo⇒PName*. Only all confidences for itemsets {*PNo*, *PName*} are 100%. Thereforfe, store PNo into DT List and PName into DP List. An example for both Result1 and Result2 are true, when mining rules *JName⇒JNo* and *JNo⇒JName*. In this case, ask the user to choose only one rule. The XML List after processed 2-itemsets is shown in Figure 5(b) and The FD List is shown in Figure 8(a).



(a) XML list initial



(b) XML list after processed 2-itemsets



(c) XML list after pruned and ready for processing next itemsets

Figure 5. The XML list

```
< ItemSet-2>
  <Frequent>
    <A>S1</A>
    <B>London</B>
    <Sup.Count>2</Sup.Count>
    <Confidence>100%</Confidence>
  </Frequent>
. . . etc
</ItemSet-2>
```

SNo	SCity	Sup.count	Confidence
S1	London	2	100%
S2	Paris	8	100%
S3	Paris	2	100%
S4	London	2	100%
S5	Athens	10	100%

SCity	SNo	Sup.count	Confidence
Athens	S5	10	100%
London	S1	2	50%
London	S4	2	50%
Paris	S2	8	80%
Paris	S3	2	20%

Figure 6. The confidence itemsets {SNo, SCity} and {SCity, SNo}

To prune the weak FD rules use XML List as a guide line, i.e. XML→Code>=1. If XML→Code>1 then FD should be pruned according to the support and the correlation those are calculated by XQuery in Figure 2(c). For example, in Figure 5(b) element Name JCity, the Code is 2, i.e. itemsets {JName, JCity} and {JNo, JCity}. The calculation support and correlation itemsets is show in Figure 7. The average support and the average correlation for rule *JName⇒JCity* and *JNo⇒JCity* are same, i.e. 14% and 0.30 respectively. Therefore ask the user to choose the rule to be pruned. In this study case, we choose to prune the rule *JName⇒JCity*. Furthermore, delete all XML elements in XML List with Code are greater or equal one for preparing to next process mining. As a result the XML List is shown in Figure 5(c). There is still one XML List element with Code is 0.

```
<SupCor>
  <Frequent>
    <A>Console</A>
    <B>Athens</B>
    <Support>33.33%</Support>
    <Correlation>0.1</Correlation>
  </Frequent>
. . .etc
</SupCor>
```

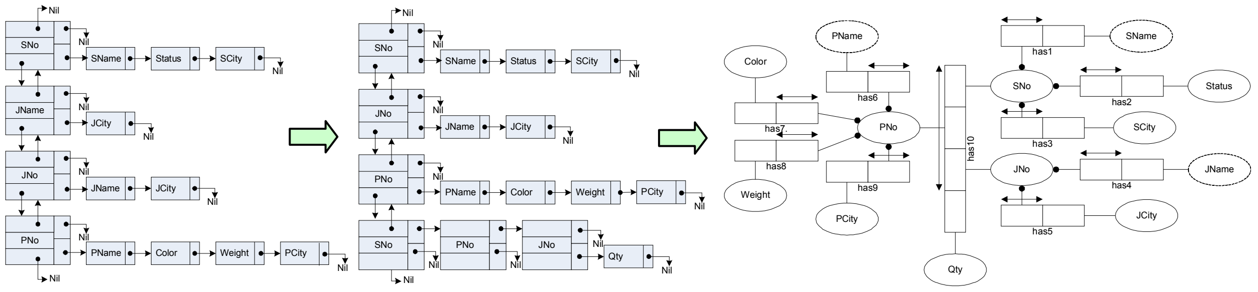
JName	JCity	Support	Correlation
Console	Athens	33.33%	0.10
Display	Rome	20.83%	0.20
EDS	Oslo	4.17%	1.00
OCR	Athens	8.33%	0.10
RAID	London	8.33%	0.20
Sorter	Paris	12.50%	0.33
Tape	London	12.50%	0.20
	Average	14.29%	0.30

JNo	JCity	Support	Correlation
J1	Paris	12.50%	0.33
J2	Rome	20.83%	0.20
J3	Athens	8.33%	0.10
J4	Athens	33.33%	0.10
J5	London	8.33%	0.20
J6	Oslo	4.17%	1.00
J7	London	12.50%	0.20
	Average	14.29%	0.30

Figure 7. The support and correlation itemsets {JName, JCity} and {JNo, JCity}

Mine 3-itemsets (*A^B⇒C*) by combination two items A-B with Code -1 and one item C with Code at least 0, i.e. {SNo, PNo, Qty}, {SNo, JNo, Qty}, {PNo, JNo, Qty}. Pass 3-itemsets into the XQuery in Figure 2(B). However, no all confidence for every 3-itemsets is 100%. Therefore, mine 4-itemset (*A^B^C ⇒D*), i.e. {SNo, PNo, JNo, Qty}. All confidences in the itemsets are 100%. As a result store the rule in FD List and increment the Qty →Code by one. If all XML→Code is not 0 then stop the mining process.

As a result FD List is shown Figure 8(b). The last step is used for generating an ORM conceptual schema in Figure 8(c), for detail algorithm refer to [5].



(a) FD list initial (b) FD list final (c) ORM diagram

Figure 8. The FD list and the generated ORM diagram

5. Conclusion

This paper presents a conceptual schema reverse engineering technique and shows that MAR can be used for mining FDs in XML Databases. If XML Databases are not big enough, it is possible to produce interesting rules which depend on difference determinants. The support and the correlation itemsets can be employed to prune the weak FD rules. It is concluded that MAR techniques can improve our proposed conceptual schema reverse engineering technique [5] and can be used generally to reverse engineer FDs from XML Databases.

6. References

[1] World Wide Web Consortium. XML 1.0 (Second Edition) W3C Recommendation. http://www.w3.org/XML.

[2] Linda, Bird, Andrew Goodchild, and Terry Halpin. "Object Role Modeling and XML-Schema", *Proc. of ER 2000*, USA, October 2000, pp. 1-14.

[3] Narudol C. and Suphamit C., "An Object and XML Database Schemas Design Tool", *Proc. of ITCC 2004*, USA, April 2004, pp. 421-424.

[4] Yuliana, Oviliani Y. and Chittayasothorn S., "A Conceptual Schema Based XML Schema with Integrity Constraints", *Proc. of ICHIT 2008*, Korea, August 28-29 2008, pp. 19-24.

[5] Yuliana, Oviliani Y. and Chittayasothorn S., "XML Schema Re-Engineering Using a Conceptual Schema Approach", *Proc. of ITCC 2005*, USA, April 4-6 2005, pp. 255-260.

[6] R. Agrawal and Srikant R, "Fast Algorithm for Mining Association Rules", Proc. of the 20th International Conference on Very Large Databases, Chile, 1994, pp. 487-499.

[7] Myint Myint Khaing and Nilar Thein, "An Efficient Association Rule Mining for XML Data", *Proc. of SICE-ISACE 2006*, Korea, October 18-21 2006, pp. 5782-5786.

[8] Jacky W.W. Wan and Gillian Dobbie, "Extracting Association Rules from XML documents Using XQuery", Proc of WIDM, USA, November 7-8 2003, pp. 94-97.

[9] D. Chamberlin, "XQuery: An XML query language", *IBM Systems Journal Vol. 41 No. 4*, 2002, pp. 597-615.

[10] Date, C. J., *An Introduction to Database Systems the 7th edition*, Addison Wesley Longman, Inc., USA, 2000.

[11] Pedro Sousa, Lurdes Pedro-de-Jesus et. al, "Clustering Relations into Abstract ER Schemas for Database Reverse Engineering", *Prof. of the 3th European conference on Software Maintenance and Reengineering*, Netherlands, March 03-05 1999, pp.169-176.

[12] Halpin, Trerry, *Conceptual Schema & Relational Database Design the 2nd edition*, WytLytPub, USA, 1999.

[13] Han, Jiawei and Micheline Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, USA, 2001.