A New Type of BOM Model and Its Application

Yao Wang, Weiguo Wang, Song Mao
Beijing Institute of Aerospace Information
Beijing, China
wangvito@gmail.com

***Abstract*—To exactly represent the graph data structure of BOM in actual design and production, BOM structure was defined as an independent entity in the storage model and the relations of Father_Part-Structure and Structure-Son_Part were separated and described mathematically respectively. A novel storage model combining XML data model and relational data model was built, and the tree structured XML model was designed as the cache of BOM query results to avoid the inefficiency of transforming BOM data from relational model to tree structured object model. The entity relationship diagram of the novel model and the XML schema of the cache were present, and the methods of creating and querying the cache in XML type was introduced in detail. Running in a real system, the novel model can greatly improve efficiency of BOM data querying comparing to traditional models.**

Keywords- Bill of Material; Storage model; XML; Cache

I. INTRODUCTION (HEADING 1)

High mix/low volume production has become an important production mode of modern manufacturing enterprises because of its high flexibility and [1]. In this mode of production, there are more shared parts and components and engineering changes are more relatively frequent. Although one component can own only one Bill Of Material(BOM) assembly structure in memory, yet there may be several versions or different valid periods in storage model, and in different phases of product life cycle there should be EBOM(Engineering BOM) view, PBOM(Process BOM) view and MBOM(Manufacturing BOM) view in various functional department[2]. Therefore the BOM storage model is different from the tree structured BOM object in memory; it is actually a graph structured model. However, the information of assembly structure is generally coupled with part or component information in traditional BOM models, leading to high redundancy of the data of general parts and shared parts, difficulties in expressing graph structured BOM and automatic implementation of BOM engineering changes. Xie's[3] new BOM data structure based on concatenated code improved the operating efficiency of BOM, however it's not flexible enough while expressing shared BOM structure and there exists high redundancy in data storage. With a view to balancing the efficiency of BOM storing and building, the BOM structures could be assembled using a reusable component lib based on semi-structured data model in [4], yet structures in this model was still coupled with components, not suitable for representing many-to-many relation between structures and components.

In paper [5], the assembly relation between parts is defined as a triple of SR=<c1, c2, x>, which indicates that the assembly amount of part "c2" in assembly "c1" is x. However, the triple restricts the relation between part and assembly to simply "Father_Part-Son_Part" relation. There are redundancies when expressing different assemble structure and reference relation. Take the BOM of product "p" in Fig.1 for an example, one product "p" is assembled by two "s" and one "t", and one "s" is assembled by two "a" and two "b", and so on. According to the triple, the BOM of product p can be expressed as SP(s) ={ RA(p),RA(s),RA(t), RA(b)1, R(b)2, R(c)}; RA(p) = {<p, s,2>, <p,t,1>}; RA(s) = {<s,p,-2>, <s,a,2>, <s,b,2>}; RA(t) = {<t, p, -1>,<t, b, 3>, <t,c,2>}; RA(b) 1 = {<b,s,-2>, <b,e,2>, <b,f,4>}; RA(b)2 = {<b,t,-3>, <b,e,2>, <b,f,4>} , R(c)={<c,t,-2> , <c,g,2>,<c,h,3>}. It is obviously that the relation of <b,e,2>, <b,f,4> is expressed repeatedly in RA(b) 1 and RA(b) 2. Actually, <p, s, 2> in RA(p) and <s,p,-2> in RA(s) repeat the same relation.

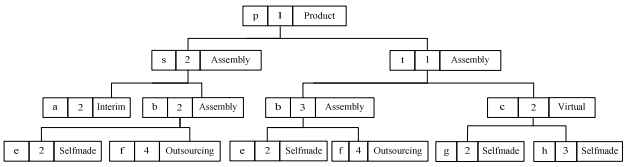

Figure 1. An instinct of product BOM in tree style

II. DEFINITION OF MBOM ASSEMBLY STRUCTURE

There are mainly two types of MBOM information. One type of MBOM information is a simple one-dimensional data table, composed of attrib fields related to the part(or material) itself, such as name, code, version, et al. Another type of MBOM information is used for expression of assembly relation or assembly structure. Assembly structure is generally described as a set of "Father-Son" relation and amount of son-parts. In actual engineering application, BOM assembly structure is graph data structure.

"Father_Part" is a component assembled by a set of "Son_Part". As in Fig.2a, component a is assembled by a set of part (c1,c2,…,c10), then component a is Father_Part of these parts and each individual part is Son-Part of p. "Assembly Structure"(or simply as structure) is a set of assembly relations between Father_Part and Son_Part. In

traditional models, assembly relations are generally expressed by "Father_Part-Son_Part". There are too many redundancies in these models and components with multi-structure are difficult to describe. As in Fig.2a, there are obvious redundancies in describing a and g, which have a same set of Son-Part (c1,c2,…,c10). And different structures of "a-(c1,c2,…,c10)"and "a-(e,f)" are hard to be distinguished.

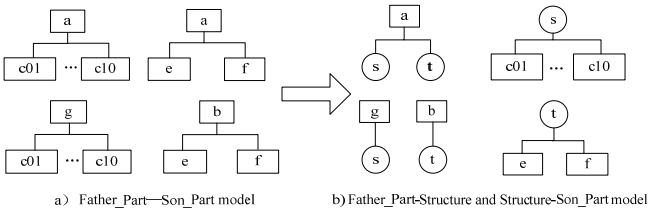

Figure 2. BOM structure comparison of traditional model and new model

"Father_Part-Structure" indicates that "Father_Part" has a certain structure. As in Fig.2b, "a-s" indicates that component named a has a structure named s(A character with a circle indicate structure.). And "Structure-Son_Part" such as "s-c01" in Fig2, indicates that a certain structure has a certain part. In this way, relation types of "Father_Part-Structure" and "Structure-Son_Part" are divided by the independence of assembly structure in BOM model. So that the new BOM model is more suitable for describing complex structures and avoiding redundancies. As in Fig.2b, the appearance of structure s brings the independence of "a-s" and "s-c01", avoiding repeated expression of the same structure s. At the same time, two different structures of s and t can be more specifically described. In order to describe the new BOM model more exactly, a formally mathematic expression is used as follows.

Assign the set of part x as X = {x1, x2, ⋯}, and the set of structure s as S = {s1, s2,⋯}, in which s=<v, w, tf, tu>, v refers to version, w refers to view, tf refers to begin of the valid period, tu refers to the end of the valid period. In this storage model, there could be several structures in any component. However, if the version, view and time of a BOM instance are all specific, there can only be one valid tree structure(If there are several structures, then the one with the max version is valid). Assign the valid structure as svalid(or sv), then $sv=|S|^{\max(v)\wedge w_v(t_f\le t_{now})\wedge(t_{now}<t_u)}$.

Assign the relation of Father_Part-Structure as a two-tuple of F(x)s1=<x,s1>, the relation set of Father_Part-Structure as F(x)={<x,s1>,<x,s2>,⋯}. Assign the relation of Structure-Son_Part as a triple C(s)x1=<s,x1,n1>, assembly amount as Q[C(s)x1]=n1, the relation set of Structure-Son_Part as C(s)= {<s,x1,n1>,<s,x2,n2>,⋯}.

As in Fig.1, the BOM instance can described as X={xp,xs,xt,xa,xb,xc,xe,xf,xg,xh}, S={s1,s2,s3,s4,s5}, F={<xp,s1>,<xs,s2>,<xt,s3>,<xb,s4>,<xc,s5>}, C={<s1,xs,2>,<s1,xt,1>,<s2,xa,2>,<s2,xb,2>,<s3,xb,3>, <s3,xc,2>, <s4,xe,2>, <s4,xf,4>,<s5,xg,2>,<s5,xh,3>}. Comparing to the triple of SR=<c1, c2, x>, repeatedly expression of component b is avoid.

III. NEW STORAGE MODEL BASED ON XML

Presently, BOM is always stored in the form of relational model. However, BOM in the memory is always a tree structure of object model. BOM object in the memory is generally constructed from the stored BOM by two methods. The first method, which is simply to implement but slow, is to construct the BOM object successively by a recursion query of the BOM hierarchy. The second method is to insert BOM information in a temporary table in database by a recursion query of SQL script, then to construct a tree type BOM object[6] in memory by the temporary table. The speed of the second method is relatively increased, but it still cannot fulfill the requirement of complex product structures. BOM data should be transformed from relational model to tree type model with each query of the two methods, which is the main reason of inefficient BOM reading. Therefore, a new method is put forward in this paper, which stores the query result directly in the database in the form of tree structure. So that when a same query is implemented, the tree structure in the database can be mapped into the memory directly, avoiding inefficient query of the same relational mode afterward. Since the query result stored in the database effects as cache data of application systems, it is called "BOM cache" in this paper.

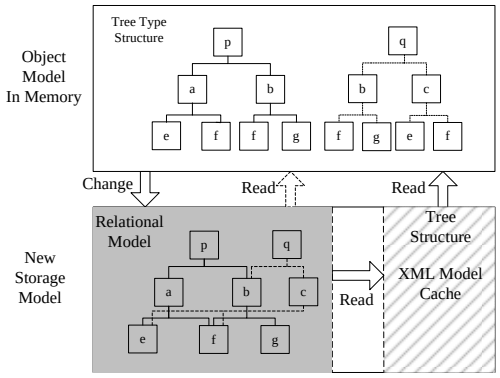

Figure 3. New storage model based on XML and its implement

XML is a kind of semi-structured data with ability of self-description. The natural tree type structure of XML, which has a same data structure with BOM object in the memory, is quite suitable for expressing BOM hierarchy. At present, XML data type is generally supported by leading commercial databases. And the data query language of XQuery has become a recommend standard of W3C. However, as XML standard of data updating is not mature, and most application system supports only relational data model, so that a new storage model combined relational model and XML model is presented. In this model, high efficiency of data reading is obtained based on BOM cache based on XML type, and interface of reading and updating relational data is reserved to keep compatibility.

In Fig.4, the novel BOM storage model above is presented. There are five data tables of Product, Items,PartBomLink, Structure and PartsInstructrue in this model. PK indicates the prime key of each data table, FK1 and FK2 indicate the foreign key and U1 indicateds the unique key. The ItemType field in Items is to storage item types of 'Product', 'Assembly', 'Selfmade', 'Outsourcing', 'Purchasing','Virtual', 'Interim' and 'Blank'. In the table of PartBomLink, a XML type filed of StructureInXml is designed as the BOM cache.

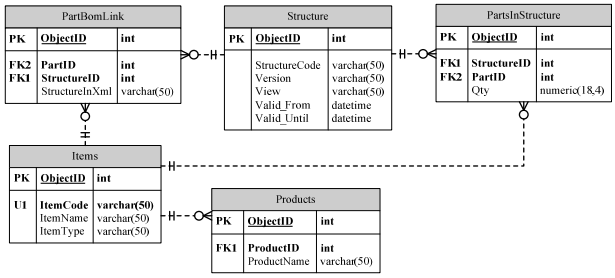

Figure 4. Entity-Relation chart of new storage model

The XML data storaged in the StructureInXml field, whose XML schema could be discribed simply as in Fig.5('ParentPart' element as the root node), defines whole BOM structure of an assembly in detail. To describe the assembly relation exactly, 'Part' element contains attributes of PartID and Qty at least (some attributes is omitted in Fig.5) and has a self nested structure.

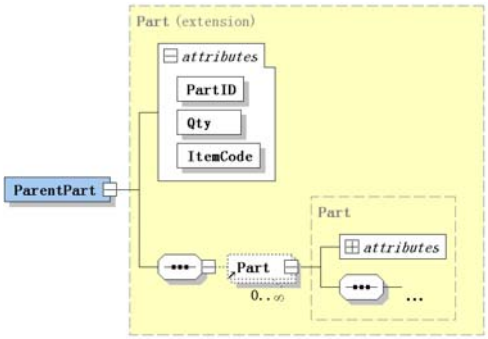

Figure 5. XML schema of the StruectueInXml field

Based on the XML schema in Fig.5, the XML code of BOM example in Fig.1 could be described as follows.

```
<ParentPart PartID="1" ItemCode="p" Qty="1.0">
    <Part PartID="2" ItemCode="s" Qty="2.0">
        <Part PartID="4" ItemCode="a" Qty="2.0"/>
        <Part PartID="5" ItemCode="b" Qty="2.0">
            <Part PartID="6" ItemCode="e" Qty="2.0"/>
            <Part PartID="7" ItemCode="f" Qty="4.0"/>
        </Part>
    </Part>
    <Part PartID="3" ItemCode="t" Qty="1.0">
        <Part PartID="5" ItemCode="b" Qty="3.0">
            <Part PartID="6" ItemCode="e" Qty="2.0"/>
            <Part PartID="7" ItemCode="f" Qty="4.0"/>
        </Part>
        <Part PartID="8" ItemCode="c" Qty="2.0">
            <Part PartID="9" ItemCode="g" Qty="2.0"/>
            <Part PartID="10" ItemCode="h" Qty="3.0"/>
        </Part>
    </Part>
</ParentPart>
```

IV. CONSTRUCTION AND QUERY OF BOM CACHE

BOM cache is generally constructed while the first BOM structure query of a product or assembly. A temp table is constructed while the first query is implemented with a given ID of 'productID'(corresponding to ObjectID in the Items table), then converts to tree-type BOM data based on XML. The specific steps are as follows.

Step1. Construct a temp table named TempBOM with field of ParentPartID, ChildPartID, ActualQty and Level. Then construct a temp stack table named TempParts and with fields of ChildPartID and Level. Then construct another temp table named TempStructureXml with a filed of ParentPartID and a XML type field of PartSturecture. Then construct a temp viarable level =0 and insert a new record of (productID, level) to TempParts.

Step2. If level>=0, then assign the value of ParentPartID in the first record of TempParts to parentPartID, and deletle the first record of TempParts. Then go to Step3. If level<0, then go to Step6.

Step3. Find the corresponeding record in PartBomLink by parentPartID. If value of StructureInXml is not null, then insert the value of parentPartID and StructureXml into TempStructureXml and execute decrement of level by 1. And then go to Step2. Else if StructureInXml is null and

there are n fields of StructureID which is not null, then go to Step4.

Step4. Find all Structure sets corresponding to these n fields of StructureID, get the valid structure of sv, and go to Step5 with the ObjectID of sv. If there is no valid structure, then go to Step2 with the decrement of level by 1.

Step5. Increment level by 1. Get all corresponding records of PartID and Qty from PartsInStructure by StructureID from Step4. Then assign PartID, Qty, parentPartID from Step3 and level to the fields of ChildPartID, AcutualQty, ParentPartID and Level in TempBOM. Then assign PartID and level to the fields of ChildPartID and Level in TempParts. Then go to Step2.

Step6. Get BOM data based on XML type by recursion of TempBOM and PartStructure. Then update the corresponding field of StructureInXml in PartBomLink with productID. Then the BOM cache based on XML is constructed or modified.

The convertion of BOM data from relational type to tree type is achieved by the above steps. For traditional storage model without BOM cache, a similar low-speed convertion is exectued in each construction of a tree type BOM data from relational data. And for the novel storage model with a XML type cache, this convertion is exectued only in the frist BOM query operation. When the BOM cache is constructed, the BOM structure of assembly s in Fig.1 could by retrieved with the following SQL statement in Microsoft SQL Server 2005.

```
SELECT StructureInXml.query('
  for $x in /ParentPart//Part
  where $x/@ItemCode = "s"
  return $x')  FROM PartsInStructure
```

The BOM data result of the query is as follows:

```
<Part PartID="2" ItemCode="s" Qty="2.0">
  <Part PartID="4" ItemCode="a" Qty="2.0" />
  <Part PartID="5" ItemCode="b" Qty="2.0">
    <Part PartID="6" ItemCode="e" Qty="2.0" />
    <Part PartID="7" ItemCode="f" Qty="4.0" />
  </Part>
</Part>
```

With the tree type structure in XML, the BOM object in the memory could be constructed easily with the convertion of each Part element to a Part object.

V. CONCLUSION

Manufacturing BOM is constantly used in a manufacturing management system. And it remains a difficulty to improve the efficiency of BOM storage model. It is proved that the effect of efficiency improvement of this novel storage model is more obvious while there are more assembly nodes and levels. Taking a BOM of 1065 nodes and 11 levels as an example, 10 tests are carried out to compare the novel storage model with the traditional model. It is proved the first construction process of BOM tree takes 6.173s with the traditional storage model and it takes 6.270s with the novel storage model. For the 2 to 10 times of BOM tree construction process, it takes 6.156s with the traditional model and 0.078s with the new model.

References:

[1] Benbo LI. Research on Process Capability and Control Chart for Mass Varieties and Small Batches Production[D]. Chongqing: Chongqing University, 2007.(In Chinese)

[2] Zhiqiang Wei, Xiankui Wang, Dan Wu, et al. BOM multi-view mapping of product based on a single data source[J]. Journal of Tsinghua University(Science and Technology). 2002,42(6):802-805. (In Chinese)

[3] Guiliang Xie，Junqiang Wang，Shudong Sun. BOM information system based on component and new data structure[J]. Chinese Journal of Mechanical Engineering. 2004, 40(5): 118-120.(In Chinese)

[4] Zheng DONG，Xiao-fei XU，De-chen ZHAN. Research on New Product Structure Model Based on Semi-structured Data[J]. Computer Integrated Manufacturing Systems. 2003, 9(1): 15-19.(In Chinese)

[5] Xuewen HUANG，Yushun Fan. RESEARCH ON BOM VIEWS AND BOM VIEW MAPPING MODEL[J]. Chinese Journal of Mechanical Engineering. 2005, 41(04): 97-102.(In Chinese)

[6] Xiaopan Zhang, Yi Lei, Hongfeng Liu, etal. Customization Approach for CAXA-BOM Generation[J]. Journal of Computer Aided Design & Computer Graphics. 2002, 14(5): 483-485.(In Chinese)