在 XML 資料倉儲中群組之研究

邱紹豐　　余政憲
大葉大學資訊工程學系
R9206028@mail.dyu.edu.tw

摘要

資料倉儲在目前已成為企業有效進行資料分析與查詢的平台，而在眾多的軟體供應商(ASP)也開發出許多相關的軟體來提供企業建構資料倉儲。在現今 XML 的興起，也漸漸廣泛的被企業採用，但是在目前的環境中，卻沒有有效的工具來提供企業對 XML 資料進行整合與分析。在本篇研究中我們根據 XCube 架構來開發一套以 XML 資料為基礎的資料倉儲，並針對 XML 資料群組化時因頻繁掃描 XML 文件而造成系統 I/O 負擔問題進行改善。

關鍵詞：資料立方體，線上分析系統，群組與聚合。

1. 前言

有了功能齊全的資料倉儲系統，使用者還需要不同分析工具的輔助，OLAP(Online Analytical Process)便是可是讓使用者根據決策需求(如切割、剖析…等)來瀏覽資料，而資料倉儲與 OLAP 工具是基於多維度的模型，將該模型資料看作資料立方體(Data Cube)。資料立方體允許以多重維度對資料建立模型與觀察，它由維度(dimension)與事實表(fact)來定義。事實表是數值的度量(numeric measures)在關聯式資料庫系統中常見的方法就是群組(group by)與聚合(aggregate)的運算。

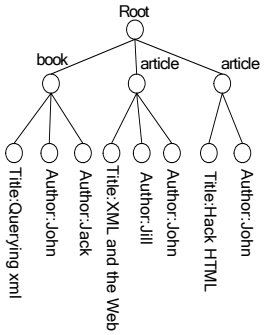
在現今 XML 資料中雖然有一些個別的工具能針對 XML 資料進行查詢，但是卻無法有效整合這些資訊，尤其利用 OLAP 來分析以 XML 為基礎的資料倉儲更顯得額外困難。另一方面對 XML 資料進行群組與聚合計算時在建構一個 Cube 時最少需要掃描原始 XML 一次，但是當要產生多個 Cubes 時，如何將掃描 XML 文件次數降低來提升效率減少系統的負擔。

在本篇研究中採用 XCube 特殊結構的資料倉儲並利用 OLAP 分析工具來快速找到符合條件的資料並產生結果。本文在實體化(materialize)多個 Cubes 方面，藉由分析每個 Cube 維度的 XPath 來找出交集的祖先節點，透過該節點來擷取原始 XML 資料中符合該節點的子樹，每當出現符合該節點的子樹時，並分析該子樹的所有節點集合中是否有我們要實體化 Cube 的維度，來達到只要掃描原始 XML 資料一次即可實體化所有的 Cubes。

本論文第二章介紹 XML 背景與利用 XCube 來實現 OLAP 相關技術，第三章說明系統架構與群組最佳化問題，第四章則對實作系統說明，第五章歸納出結論與未來研究方向。

2. 相關研究

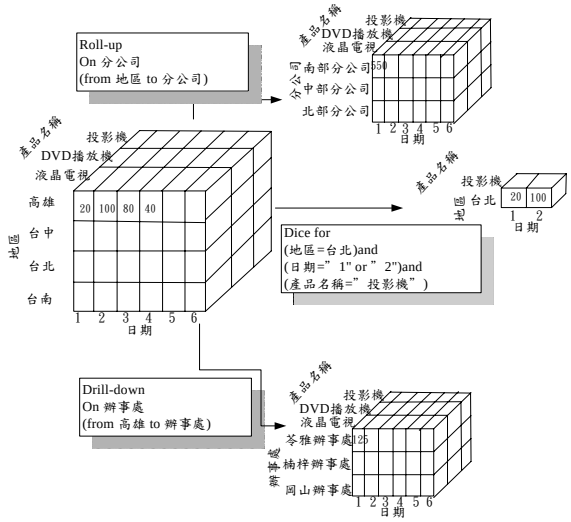
XML 是由 W3C 組織所提出的在 Internet 上描述資料與交換的新標準，可以利用自訂標準的資料格式，來達到跨平台的電子交換。XML 資料並沒有固定的架構或規則性，因此常稱為半結構性資料(Semi-structured Data)[8]，而以樹狀結構描述的 DOM(Document Object Model)是最常被使用來表示半結構性資料[6]，如圖一所示。



圖一　以 DOM 表示 XML 的範例

在資料組織可視成為一種多維度的架構，透過這些維度由概念階層定義成多個抽象層，經由 OLAP 的操作 (如上捲(roll-up)、下鑽(drill-down)、切片 (slice) 等) 來產生這些不同的視域(views)[4,5,10]。圖二為某公司分設辦事處的層級(level)關係，$\preceq$ 表示進行聚合資料，使用"辦事處 $\preceq$ 地區 $\preceq$ 分公司 $\preceq$ 總公司"來表示整個階層的關係，以辦事處 $\preceq$ 地區為例代表將辦事處依所在的地區進行資料的分群。在圖三中可視為一個三維度的資料立方體，分別進行 OLAP 的上捲、下鑽與切片操作，上捲操作是透過維度的概念階層向上攀升在 Cube 上進行聚集運算來產生新的視域，下鑽操作是上捲的逆向運作，透過維度的概念階層向下或引入新的維度操作，切片操作是在給定的資料立方體中進行特定維度值的選擇來產生子資料立方體(Sub-Data Cube)。

圖二 資料庫屬性階層結構

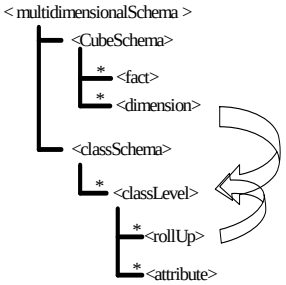


圖三 多維度資料上OLAP操作範例

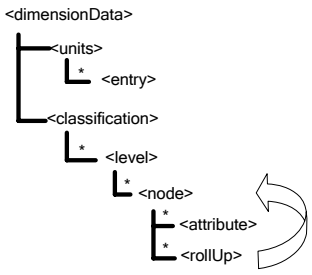
XCube 是由 Wolfgang Hümmer 等學者提出針對 XML 建構資料倉儲的架構[12]，在 XCube 的成員中包含 XCubeSchema、XCubeDimension 和 XCubeFact。XCubeSchema 主要描述多維度結構的資料立方體，包含每個資料立方體的維度與事實表。XCubeDimension 主要描述每個維度的層級、分類(classification)與等級制度(hierarchies)。XCubeFact 描述資料立方體中所包含的單元(cell)，每個 cells 包含相關的維度與事實表值(fact value)。

XCubeSchema 是主要核心部份來描述 Data Cube 的多維度資料，包含每個 Cube 的維度與事實表，在圖四中粗線代表父子關係，星號表示出現次數(包含零次)，每個 CubeSchema 區塊(block)描述每個 Cube 的維度與事實表，而 Cube 裡的維度可以參考到 classSchema 區塊中每個維度的等級關係，在 classSchema 區塊中描述每個維度上捲到更高的層級與每個層級中維度的屬性。

雖然在 XCubeSchema 中描述每個 Cube 的維度與層級，但是缺少了每個維度層級中包含哪些節點(node)，在 XCubeDimension 主要功用便是詳細定義這些層級中節點的分類。這整份 XCubeDimension 的架構在圖五，dimensionData 是整個文件的進入點(root)包含兩個子節點，units 是描述當維度牽涉到某些度量單位(如歐元、公尺等)，classification 定義每個維度的分類、屬性與層級關係。XCubeSchema 與 XCubeDimension 是息息相關的，在 XCubeSchema 中的每個維度必須完整描述與定義在 XCubeDimension 文件中。



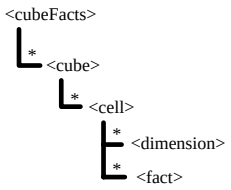
圖四 XCubeSchema



圖五 XCubeDimension

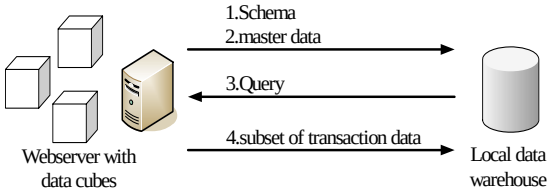
XCubeFact 為描述每個 Data Cube 中的 cells，這部份必須牽涉到資料的群組與聚合函數運算，在第 3 節中我們針對這問題進行最佳化。

在圖六中定義每個 Cube 包含哪些 cells，每個 cells 中包含兩個部份 dimension 與 fact，dimension 表示多維度資料中相同的類別(coordinates)，且必須符合 XCubeDimension 節點的分類和本身相關的事實表值，(每個多維度資料中可能儲存許多 fact 值，因此在每個 cell 中也必須 hold 這些 facts)。



圖六 XCubeFact

由於資料立方體的資料量相當龐大，因此在可能的情況下能由伺服端(Server)去進行資料分析，回傳客戶端(Client)所要查詢的資料。如圖七所示，步驟一由伺服端下載 XML 架構(schema)與 Master data(如 XCubeDimension 等)。步驟二 Client 端根據 Server 端所提供的 Schema 來下達查詢(Query)所需要的資料。步驟三 Server 端分析資料並將結果回傳給 Client 端。



圖七 Client對Server進行資料查詢之流程

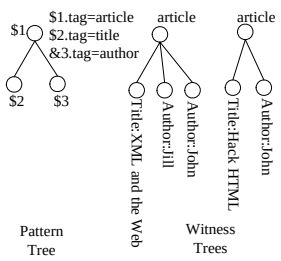
在建立資料立方體時，需對原始資料進行資料分群[1,2,5,7,11]。但是在進行 XML 資料分群時，因為不像傳統關聯式資料庫有固定架構的限制，因此相對在進行資料分群時顯得額外困難。圖八中是由 W3C 所提供的 XQuery 語法與 XPath 來分群“author”，並輸出每位作者出版過的文章名稱(title of article) [3,9]。對於 XQuery 一些作者也提出相關的見解與改善[1,2,11]，分別提出下列二點的議題：

1、 使用XQuery需要收集一些相關的集合資料，但是這些集合中可能包含一些無效的資訊。
2、 當收集$a與$b集合後，在需要分析兩集合資料中是否存在a binding b時則回傳c的條件下，若有一子樹只存在a與b時，則無法有效回傳c。

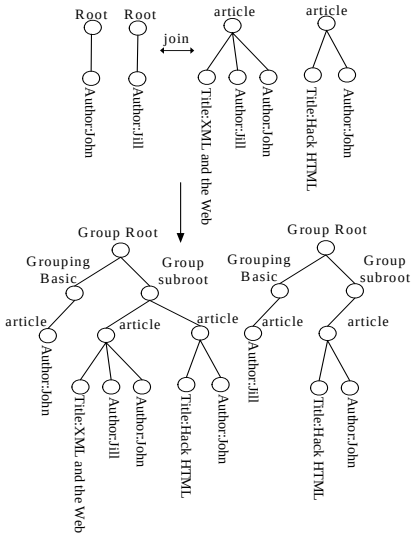
```
01 FOR $a IN distinct-values(document("bib.xml")//author)
02 RETURN
03    <authorpubs>
04    { $a }
05    {
06       FOR $b IN document("bib.xml")//article
07       WHERE $a = $b/author
08       RETURN $b/title
09     }
10    </authorpubs>
```

圖八 XML Query

針對這二點的議題，Stelios Paparizos 等學者提出一些的改善方法[2,11]。圖八的語法與圖一的資料中可以發現所要的資訊都在以 article 為 root 的子樹，以此條件建立一個 Pattern Tree，根據此 Pattern Tree 掃描 XML 資料產生符合條件的 Witness Trees 集合，如圖九所示。以 Witness Trees 為資料樹來 group by author 來產生圖十中左上部分以 author 為主的子樹，最後將這 author 子樹與 Witness Trees 進行合併(Join)運算產生下半部的結果。



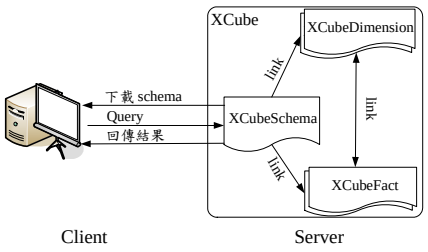
圖九 Pattern Tree 與 Witness Trees



圖十 Author tree Join Witness Trees

3. 系統架構與建構群組

資料倉儲是典型多維度的資料模式，為了有效在多維度的資料間做資料整合，本文採用 XCube 架構來描述 XML 資料。如圖十一所示，在 Server 端中 XCube 架構包含三份文件，XCubeSchema 主要描述每個 Cube 所包含的維度資料、事實表與每個維度能夠上捲的層級。XCubeDimension 描述每個維度階層的分類。XCubeFact 為描述 Cube 中的 cells。Client 端可下載 Server 端的 XCubeSchema 文件，根據該文件資訊讓使用者挑選所要查詢的 Cube 相關資訊，當 Server 端接收到 Client 端 Query，透過 Query 條件分析 XCubeSchema 來鏈結(link)符合的 XCubeFact、XCubeDimension 文件進行聚合等相關運算，並將結果回傳給 Client 端。



圖十一 系統架構圖

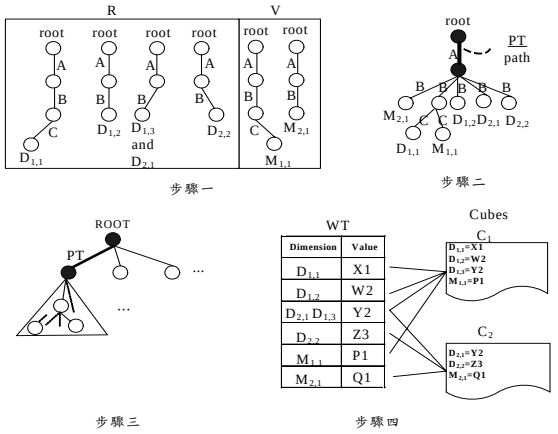
在產生 XCubeFact 文件時，須對原始資料進行分群與聚合運算，以 XQuery 語法而言，每當要計算一個 Cube 必須掃描(scan)XML 文件一次，這部份探討如何在需要產生 N 個 Cube 前提下，將掃描 XML 文件次數減到最少。

當有 N 個 Cube 必須要實體化時，令每個 Cube 表示為 $c_i=(D_i,F_i)$，$1 \leqq i \leqq N$，D_i 與 F_i 分別代表在第 c_i 個 Cube 中所包含的維度路徑與度量(measure)路

徑集合。$D_i=\{D_{i,1},D_{i,2}....D_{i,K}\}$，$D_{i,K}$ 表示在第 i 個 Cube 中第 K 個維度路徑。$F_i=\{M_{i,1},M_{i,2}...M_{i,L}\}$，$M_{i,L}$ 表示為在第 i 個 Cube 中第 L 的度量路徑。設 $C=\{c_1, c_2...c_N\}$，C 為 Cubes 的集合，N 為 Cube 的個數。

設 R 為 C 中每個 Cube 維度的不同路徑之集合，V 為 C 中每個 Cubes 的事實表中所要度量的不同屬性路徑之集合。分析步驟如下(圖十二)：

1. 將每個 Cubes 的 D_i 與 F_i，去除重複路徑，分別放入 R 與 V 集合。
2. 分析 R 與 V 的路徑，找出每個路徑共同交集的子路徑，將其設為 PT。
3. 掃描 XML 文件篩選符合 PT 路徑的子樹。並收集該子樹中符合 R 集合中的節點，將其放入 WT 集合中。
4. 如果 WT∈ C，則比對 WT 在 C 中是否已有此維度值，如果存在則聚合 V，反之則新增每個 Cube 中的所需的 R 與 V。



圖十二 群組化步驟

4. 系統實做

我們採用一份業務 XML 檔案作為我們的資料檔，資料格式如圖十三所示：

```
<dataroot >
<cust>
<姓名>陳小姐</姓名>
<辦事處>苓雅辦事處</辦事處>
<產品名稱>投影機</產品名稱>
<月份>3</月份>
<出貨量>39</出貨量>
<業績>1170</業績>
</cust>
<cust>
……
</cust>……
</dataroot >
```

圖十三 XML 資料範例

4.1 定義 XCubeFact 文件

我們將此資料檔建立二個多維度資料，分別為 <cubeschema id="辦事處,月份,產品名稱">計算出貨量總和與平均與計算業績的平均值。其餘 Cube 資訊如圖十四所示。圖十五為我們進行群組化與聚合函數的系統介面，由於須實做兩個 Cube，所以分別產生兩份 XCubeFact 檔案，檔名分別為該 Cube 的 Cube ID。

```
Cube id= 辦事處,月份,產品名稱
Dimension =辦事處
Dimension =月份
Dimension =產品名稱
Fact=出貨量
    aggregation operator "sum" and "avg"
Fact=業績
    aggregation operator "avg"
```

```
Cube id= 月份,產品名稱
Dimension =月份
Dimension =產品名稱
Fact=出貨量
    aggregation operator "count" and "sum" and " max"
Fact=業績
    aggregation operator "max"
```

圖十四 建構 Cube 資訊

```
請輸入有幾個cube :2

請輸入第1個cube的第1個維度 ://辦事處
請輸入第1個cube的第2個維度 ://月份
請輸入第1個cube的第3個維度 ://產品名稱
請輸入第1個cube的第4個維度 :
請輸入此cube要計算多少的aggregation :3
ggregation計算功能爲 :<  sum , avg , count , max , min >
請輸入目前的Cube第1個要進行aggregation計算功能 :sum
請輸入要計算的維度名稱 ://出貨量
請輸入目前的Cube第2個要進行aggregation計算功能 :avg
請輸入要計算的維度名稱 ://出貨量
請輸入目前的Cube第3個要進行aggregation計算功能 :avg
請輸入要計算的維度名稱 ://業績

請輸入第2個cube的第1個維度 ://月份
請輸入第2個cube的第2個維度 ://產品名稱
請輸入第2個cube的第3個維度 :
請輸入此cube要計算多少的aggregation :4
ggregation計算功能爲 :<  sum , avg , count , max , min >
請輸入目前的Cube第1個要進行aggregation計算功能 :count
請輸入要計算的維度名稱 ://出貨量
請輸入目前的Cube第2個要進行aggregation計算功能 :sum
請輸入要計算的維度名稱 ://出貨量
請輸入目前的Cube第3個要進行aggregation計算功能 :max
請輸入要計算的維度名稱 ://出貨量
請輸入目前的Cube第4個要進行aggregation計算功能 :max
請輸入要計算的維度名稱 ://業績
請輸入xml檔案名稱 :example.xml
```

圖十五 進行群組化與聚合函數系統介面

4.2 產生 XCubeSchema 文件

當產生這些 XCubeFact 文件後，我們必須定義 XCubeSchema 文件，來描述這些 Cubes 的特性，每個 Cube 所包含的事實表和維度與該事實表牽涉到的聚合操作和維度間的階層關係，如圖十六所示。

4.3 定義 XCubeDimension 文件

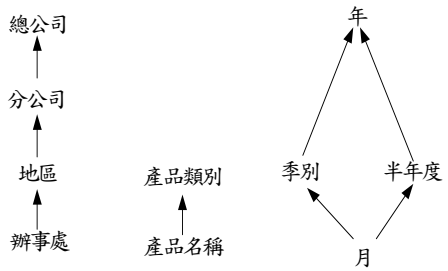
在我們定義好 XCubeSchema 文件來描述每個 Cube 的包含的維度、事實表與維度階層關係後，我們還需要 XCubeDimension 文件來完整描述每個維度在某一個階層中所有的分類與每個分類中的節點上捲的關係。圖十七為設定每個維度的階層關係圖。

圖十八為根據圖十七來定義每個維度的使用者介面，左上為根據原始的 XML 資料中顯示該資料檔所有的標籤，圖中我們選擇維度“產品名稱”，

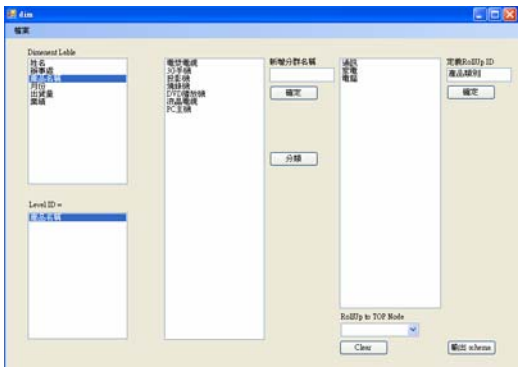
中間視窗為所有維度為產品名稱的 distinct-values，在新增分群名稱的文字方塊中由使用者定義上捲的分類，該圖是以通訊、家電跟電腦分類，當分類完畢後右邊文字方塊來定義上捲階層的 ID。

```
<multidimensionschema>
  <multiCubeSchema>
      <cubeschema id="辦事處,月份, 產品名稱">
        <fact id="出貨量">
          <defaultAggregate>
             <aggregation operator="sum"/>
             <aggregation operator="avg"/>
          </defaultAggregate>
        </fact>
        <fact id="業績">
          <defaultAggregate>
             <aggregation operator="avg"/>
        </defaultAggregate>
        </fact>
        <dimension id="辦事處"/>
        <dimension id="月份"/>
        <dimension id="產品名稱"/>
      </cubeschema>

      <cubeschema id="月份, 產品名稱">
      ……
      </<cubeschema>

    <classSchema>
      <classLevel id="月份">
          <rollUp toLevel="季別"/>
          <rollUp toLevel="半年度"/>
      </classLevel>
    <classLevel id="季別">
    ……
    </classSchema>
  </multiCubeSchema>
</multidimensionschema>
```

圖十六 XCubeSchema 文件範例



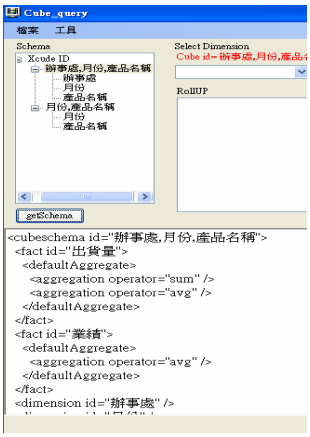
圖十七 每個維度階層圖



圖十八 定義 XCubeDimension 文件操作介面

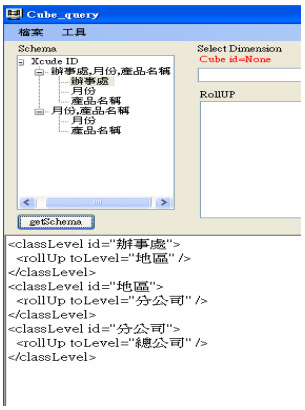
4.4 查詢 Schema

當我們都定義好這些 Schema 後，便可以開始進行 OLAP 功能。由於 Data Cube 的資料量相當龐大，因此必須要能顯示出在資料倉儲中包含哪些 Cube，哪些 Cube 能回答使用者所要的問題與每個維度能夠上捲的層級。在圖十九中左上視窗的樹狀結構顯示 XCubeSchema 文件中包含哪些 Cube 資訊，圖中顯示有兩個 Cube 分別為{(辦事處，月份，產品名稱)(月份，產品名稱)}，下部視窗為顯示使用者挑選的該 Cube 詳細資訊，圖中使用者為挑選 Cube 為{辦事處，月份，產品名稱}，下面視窗為該 Cube 的資訊，如事實表、維度、聚合函數操作等。



圖十九 查詢 Cube_Schema 系統介面

當使用者挑選好 Cube 後，還需要給使用者該 Cube 裡所有維度能上捲的程度，圖二十為對維度“辦事處”上捲資訊。

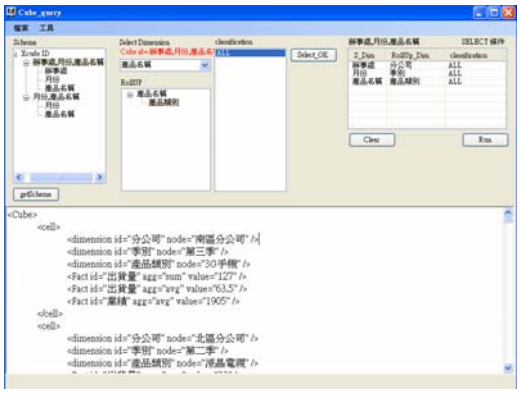


圖二十 查詢維度層級系統介面

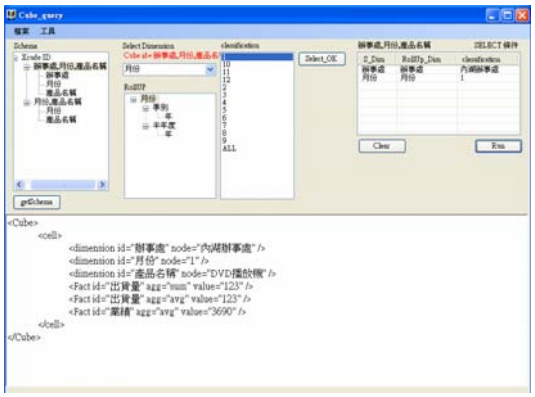
4.5 上捲與切片選擇操作

瞭解 Cube 資訊後，我們可以開始對維度做階層的上捲與對維度上的切片選擇操作，假設在該{辦事處，月份，產品名稱}Cube 中分別進行這些操作。(1)上捲操作：指定維度辦事處 $\preceq$ 分公司、月份

≼季別、產品名稱≼產品類別，在圖二十一中，右上視窗表示對上捲的操作，下半部表示上捲出的結果。(2)切片操作：我們需維度月份一月與辦事處為內湖辦事處的資料如圖二十二所示。



圖二十一 上捲操作



圖二十二 切片操作

5. 結論與未來研究方向

在本論文中，我們利用 XCube 架構來建構以 XML 資料為基礎的資料倉儲，並透過我們開發的 OLAP 工具來分析資料。在實體化 Cube 方面，我們改善當實體化 N 個 Cube 時所需掃描 XML 次數，來減少系統 I/O 的負擔。

在進行上捲與切片操作時，必需掃描相關的 XCubeFact 檔案來挑選符合條件的 cells 做聚合函數運算，未來我們考慮在 cells 中建立索引機制來提升對 OLAP 的效率問題。

參考文獻

[1] H. V. Jagadish, Shurug Al-Khalifa, Adriane Chapman, Laks V. S. Lakshmanan, Andrew Nierman, Stelios Paparizos, Jignesh M. Patel, Divesh Srivastava, Nuwee Wiwatwattana, Yuqing Wu and Cong Yu, ”TIMBER: A native XML database,” VLDB J. 11(4): 274-291 (2002).

[2] H. V. Jagadish, Laks V. S. Lakshmanan, Divesh Srivastava and Keith Thompson, ”TAX: A Tree Algebra for XML,” DBPL 2001: 149-164.

[3] J. Clark and Steve DeRose , ”XML path language (XPath),”http://www.w3.org/TR/xpath.

[4] Jiawci Han & Micheline Kamber. “Data Mining Concepts and Techniques,” Morgan Kaufmann Publishers, 2000.

[5] Jim Gray, Adam Bosworth, Andrew Layman and Hamid Pirahesh, ”Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total,” ICDE 1996: 152-159

[6] Philippe Le Hégaret, Ray Whitmer, and Lauren Wood, “Document Object Model (DOM),” W3C DOM IG, January 19, 2005.

[7] Rajesh R. Bordawekar and Christian A. Lang, ” Analytical processing of XML documents : opportunities and challenges, ”ACM SIGMOD Record Volume 34 , Issue 2 (June 2005).

[8] S. Abiteboul, P. Buneman and D. Suciu, “Data on the Web,” Morgan Kaufmann Publishers, 2000.

[9] S. Boag, D. Chamberlin, M. Fernandez, D. Florescu, J. Robie, J. Simeon, and M. Stefanescu, ” XQuery: A Query Language for XML,” W3C Working Draft. Available from http://www.w3.org/TR/xquery

[10] S. Chaudhuri and U. Dayal, “An overview of data warehousing and OLAP technology,” SIGMOD Record, 26(1):65-74, Mar. 1997.

[11] S.Paparizos, S. Al-Khalifa, H.V. Jagadish, L. Lakshmanan, A. Nierman, D.Srivastava, and Y. Wu,“Grouping in XML,” In EDBT Workshop on XML Data Management (XMLDM'02), 2002.

[12] Wolfgang Hümmer, Andreas Bauer and Gunnar Harde, “XCube: XML for data warehouses,” Proceedings of the 6th ACM international workshop on Data warehousing and OLAP (DOLAP’03), 33-40.