# 設計一個簡潔的行動資料庫處理語言

呂瑞麟*
中興大學資訊管理學系
jllu@nchu.edu.tw

盧廷貴
朝陽科技大學資訊管理學系
s9214620@mail.cyut.edu.tw

## 摘要

近年來由於無線通訊技術逐漸發展成熟，促成行動商務的發展與應用。連帶使得利用手機上網的人數也會愈來愈多，所以使用者對於手機資料處理功能的需求也越來越高，於是有了行動資料庫的需求。目前的研究中，Lu 和 Cheng[4]開發了一個手機(Java Phones)的小型行動資料庫，它是使用 XSLT 來設計新增、刪除、修改、查詢等處理規則。雖然這個行動資料庫非常小也容易安裝與使用，然而最需要加強地方就是，利用 XSLT 來做為處理規則的缺點是 XSLT 檔案較大會造成使用較多的頻寬，以及佔用過多的記憶體空間而導致處理效能較差。為了改善此缺點，透過本研究提出，XQT (XML-based Query Translation expression)來取代 XSLT 做為新的處理規則語言，其優點是它比 XSLT 的語法更為精簡，所撰寫出的處理規則檔案比原本使用 XSLT 來的更小。因此不會佔用過多的傳輸頻寬與儲存空間，所以它能改善使用 XSLT 所產生處理規則檔案較大的缺點。同時 XQT 也是一份符合 XML 規範的 XML 文件，所以也和 XSLT 有相同資料處理跨平台性的優點。

**關鍵詞**：XML、XSLT、Query translation、行動資料庫。

## 1. 緒論

近年來由於無線通訊技術逐漸發展成熟，一般社會大眾可以輕易的使用筆記型電腦、手機或其他行動設備連上網際網路。使用者可以不受傳統實體網路的限制，直接使用網路上豐富的各種資源。這也連帶加速促成行動商務的發展與應用，其相關的應用有:行動購物與付款、行動銀行、行動辦公室等[20]。完整的行動商務系統主要包含有後端的軟硬體設備（如內容與服務的提供者、無線寬頻業者、中介軟體以及硬體設備）所組成，由於內容與服務的提供者整合網路上的現有資源，所以能夠提供豐富的內容給使用者。當使用者有需要時，就可以透過無線網路（Mobile Ad Hoc Networks）來直接下載資料到行動設備上，因此可以讓他們在離線時，繼續處理或者閱讀之前下載的資料，等到重新連線後，可以跟伺服器端更新資料。除此之外，行動設備使用者也可以在離線時，事先建立他們個人的資料，之後再將個人的資料藉由無線網路而傳送給其它的使用者，透過資料的傳遞與交換來達到資料分享的目的[12,20]。例如當使用者在手機上收到某些實用資訊(如最近上映電影或某百貨公司的折扣消息)，但是當這些資訊多了之後，便會有管理這些資料的需求。例如:能夠立刻搜尋出所有折扣資訊，而不是去檢視每一筆資料找尋是否有相關的折扣資訊。為了要讓這些下載到行動設備的資料能夠便於使用者運用及處理，因此產生了行動資料庫[15,20]的需求。

由於目前利用手機上網的人數有逐漸增加的趨勢[2,10]，也有不少的製造商推出支援 Java 功能的手機[12]，讓使用者下載 Java 的應用程式並可以直接在手機中執行，因此利用 Java 來開發在手機平台的行動資料庫，能達到運作在不同手機上的跨平台性。在 Lu 和 Cheng 的研究[4]，曾經開發出在 Java 手機上運作的行動資料庫。這個行動資料庫是使用 XML (Extensible Makeup Language)來描述資料庫，因為 XML 允許使用者利用標籤來描述資料，同時透過應用程式來對 XML 文件進行相關資料的搜尋、存取、與交換，所以能夠提升應用程式資料處理的能力，也可以在不同品牌的手機間相互共享與使用資料。此外，還具有簡易的新增、刪除、修改、查詢等功能，而每一項資料處理功能都是透過資料處理規則來完成，因此在進行相關的資料處理前，就需事先產生資料處理規則。由於這個行動資料庫是利用 XSLT (Extensible Stylesheet Language Transformation)來撰寫資料處理規則，XSLT 本身也是 XML 文件，所以此行動資料庫具備有資料與資料處理規則都能共享與重複使用的特質。

雖然 Lu 和 Cheng[4]所實作的行動資料庫具有簡單的新增、刪除、修改、查詢等功能，但是它還有值得進一步加強的地方。其中目前我們認為最需要加強地方就是，利用 XSLT 來做為處理規則，雖然可以提升資料的處理與跨平台性，但是它的缺點是撰寫利用 XSLT 來描述處理規則時，會需要用到較多的語法，使得檔案變得較大。所造成的影響是需要使用較多的頻寬來傳輸檔案，及佔用過多的記憶體空間而導致處理效能較差，還有需要較大的儲存空間來儲存檔案。這些對於傳輸頻寬及儲存空間有限的手機硬體而言，的確是個負擔。所以為了改善此缺點，本研究，提出 XQT (XML-based Query Translation expression)來取代 XSLT 做為新的處理

---

1 This research was partially supported by the National Science Council, Taiwan, R.O.C., under contract number: NSC93-2213-E-005-035

規則語言，其優點是它比 XSLT 的語法更為精簡，所撰寫出的處理規則檔案比原本使用 XSLT 來的更小。所以不會佔用過多的傳輸頻寬與儲存空間，因此它可以改善使用 XSLT 撰寫處理規則檔案較大所產生的問題。為了讓 XQT 和 XSLT 有相同的資料處理跨平台性的優點，所以這個新的語法會符合 XML 的標準以達到資料處理的跨平台性。要達成上述目標，相關的領域就是在 XML 語言轉換(Query translation)的這部分研究上，因此我們將針對 XML 語言轉換的研究，做一個完整的介紹。

本篇論文的架構如下，第二節將對 XML 語言轉換的相關文獻進行探討，第三節將說明 XQT (XML-based Query Translation expression)的設計，第四節為分析，第五節為結論與未來工作。

## 2. 文獻探討

關於 XML 語言轉換(Query translation)的研究動機是當利用關聯式資料庫來儲存 XML 文件時，由於 SQL 不是 XML 的查詢語言所以無法直接查詢資料庫中的 XML 文件。為了能夠查詢在資料庫中的 XML 文件，學者利用將 XML 的查詢語言(如 XQuery、XML-QL、XQL、XSLT 等)轉換成 SQL 的方式，對關聯式資料庫來查詢相關的資料 XML 文件。因此關於這部分的研究[3,5,6,7,8,13,16,17]是著重在如何將 XML 的查詢語言，如 XQuery、XML-QL、XQL、XSLT 等，轉換成 SQL 語法。目前最常見的作法是透過中介程式（Middleware）來轉換並產生新的 SQL 語法，並透過新的 SQL 來查詢關聯式資料庫中的 XML 文件。在這些轉換的過程中，通常會使用的一些簡單的符號(expression)來表達原來的查詢語言，所以我們認為這是具有簡化語言的功能。因此我們試圖從目前現有的研究中找尋符合目標的研究。目前 XML 查詢語言主要有 XQuery、XML-QL、XQL、XSLT 等，因此可以依據不同的 XML 查詢語言分為以下 3 種類型。1. XML-QL to SQL，2. XQuery to SQL，3.XSLT to SQL。我們的目的是在於如何簡化 XSLT 因此就直接探討第三類的研究。

Suciu 等學者[17]所提出利用 XSLT 來轉成 SQL，然後來查詢關聯式資料庫是利用 XSLT 所包含的語義來進行 SQL 語法的轉換。缺點是由於每次 Query 所包含的語義，並不像 XSLT 語法是固定的，因此難有一個標準化的程序，所以難用電腦自動化的方式來處理。因此 Liu 等學者[8]提出使用 XSLT 語法來轉成 SQL，他們是依據 XSLT 語法的特性，將 XSLT 的指令大致上分成 2 種類型。第一類 輸出資料的指令，稱為 O-statement，如 xsl:value-of、xsl:copy-of、xsl:variable 等。第二類流程控制與範本(template)，如 xsl:for-each、xsl:if、xsl:sort、xsl:template 等。由於 XSLT 的語法結構是由範本(template)所構成，每個範本都是由一連串的流程控制和輸出資料的指令所組成。因此利用這點他們為上述這些指令，透過所設計的 T-expression 符號來表示。但是這種方式的缺點是，如果範本的相似程度愈高時，T-expression 重覆性也就越高，因此這個方法並不適用於 XSLT 的簡化上。由於 Lu 和 Cheng[4]所開發的手機(Java Phones)行動資料庫，是利用 XSLT 做為處理規則。其缺點是利用 XSLT 來描述處理規則時，會需要用到較多的語法，使得檔案變得較大。因此會造成使用較多的頻寬，以及佔用過多的記憶體空間而導致處理效能較差。所以我們希望透過設計一個更精簡的語法來取代 XSLT，成為行動資料庫新的處理規則，同時還能保有原本使用 XSLT 的優點並進一步提升儲存的空間。

## 3. XQT (XML-based Query Translation expression )

我們依據 Lu 和 Cheng[4]設計的行動資料庫，使用的 9 種 XSLT 的指令(如圖 1)，並利用這些 XSLT 的指令，來設計新的語言取代 XSLT，成為行動資料庫新的處理規則。

| | |
|---|---|
| **xsl:stylesheet** | **xsl:element** |
| **xsl:variable** | **xsl:copy-of** |
| **xsl:template** | **xsl:value-of** |
| **xsl:text** | **xsl:if** |
| **xsl:apply-template** | |

**圖 1 Lu和 Cheng[4]行動資料庫所使用的XSLT**

依據 W3C 所制定的 XSLT 1.0[1]規範中，XSLT 是一份 well-formed XML 的文件，以 xsl:stylesheet 為根節點，其子元素稱為頂層元素(top-level element)，主要目的是讓使用者可以用來宣告範本、變數等。範本是包含處理規則，這些處理規則有負責輸出 xsl:value-of、建立文字節點 xsl:text、條件判斷 xsl:if、變數宣告 xsl:variable 等指令。XSLT 就是利用這些範本內所包含的處理規則來進行資料的處理。因此我們依據這樣的特性，將上述 XSLT 的指令，依用途大致分為兩類。第一類是定義元素，第二類是處理規則，接下來我們將詳細介紹這兩類指令。

### 3.1 定義元素

定義元素主要為變數的宣告與定義範本的指令，如 xsl:stylesheet、xsl:variable、xsl:template。

#### 3.1.1 xsl:stylesheet

依據 XSLT 的語法規範，xsl:stylesheet 的內容會以下列形式出現。

```
<xsl:stylesheet version="1.0" xmlns:
  xsl="http://www.w3.org/1999/XSL/Transform">
        ……
</xsl:stylesheet>
```

這裡簡單說明上述語法所代表的資訊，version 是代表 XSLT 的版本，目前為 1.0 版 xmlns:xsl="http://www.w3.org/1999/XSL/Transform 則是 XSLT 的名稱空間(name space)。由於因此我們將其內容省略，以 <XQT> 來代表原來的 <xsl:stylesheet>。以圖 4 的第 1 行為範例，因此可以改為<XQT> ……</XQT>。

### 3.1.2 xsl:variable

xsl:variable 為設定變數的指令，其語法如範例圖 4 第 2 行<xsl:variable name="Input_query_name" select="cheng-rex"/> 所示。這個範例的語義為，宣告一個名稱為 Input_query_name，參數值為 "cheng-rex" 的函數。這裡我們以 $( )函數代表 xsl:variable，這裡"$"用來表示函數名稱，括號內容則是代表函數的參數值。因此範例可以改為 $Input_query_name ("cheng-rex")。

### 3.1.3 xsl:template

xsl:template 是建立範本的指令，語法如範例圖 4 第 3 行 <xsl:template match ="/">，這個範本的意義為建立一個符合根節點的範本。這裡我們以 T( ) 函數來表示 xsl:template，括號內容則是代表函數的參數值，因此範例<xsl: template match = "/">，可以改為 T ("/")。

## 3.2 處理規則

處理規則為實際進行資料輸出運算處理的指令，如 xsl:text、xsl:element、xsl:copy-of、xsl:value-of、xsl:if、xsl:apply-template。

### 3.2.1 xsl:text 和 xsl:element

xsl:text 所代表的意義為 xsl:text 建立文字節點，其語法如範例<xsl:text>Person</xsl:text>所示，這裡我們分別以 t( )函數代表 xsl:text，括號內容代表函數的參數值，這裡的參數值為<xsl:text>Person </xsl:text>的元素內容 Person，因此範例可改為 t("Person" )。

xsl:element 所代表的意義為 xsl:element 建立 XML 元素。其語法如範例圖 4 第 7 行<xsl:element name="PhoneBook">所示，這裡我們分別以 e( )函數代表 xsl:element，括號內容代表函數的參數值，因此範例可改為 e("PhoneBook")。

### 3.2.2 xsl:copy-of 和 xsl:value-of

xsl:copy-of 所代表的意義為複製節點，語法如範例圖 4 第 14 行為<xsl: copy-of select="*">。這裡我們以 c( )函數代表 xsl:copy-of，括號內容代表函數的參數值，因此範例可改為 c("*")。

xsl:value-of 所代表的意義為取出選擇 XML 元素或屬性內容，語法如範例<xsl:value-of select= "address">。這裡我們 v( )函數代表 xsl:value-of，括號內容代表函數的參數值，因此範例可改為 v("address")。

### 3.2.3 xsl:if

xsl: if 這個指令是屬於條件判斷，其語法如範例圖 4 第 12 行為 <xsl:if test="name=$Input_query_name" >。所代表意義為判斷是否符合"name=$Input_query_name"的條件，若符合則執行繼續下面其他指令，若符合則不執行。這裡我們將 xsl:if 用大括號來取代，並將判斷條件 "name=$Input_query_name"放入大括號中。因此範例就可改成{"name=$Input_query_name"}。

### 3.2.4 xsl:apply-template

xsl:apply-template 的意義，是執行所有符合條件的範本規則，其語法如範例圖 4 第 4 行為 <xsl:apply-template />。我們將 xsl:apply-template 以 a( )函數來取代，括號內容為參數值。這裡範例並無指定參數值所以範例就可改成 a( )。

### 3.2.5 階層符號 "/"

由於 XSLT 也是一份符合 XML 規範的文件，因此也具有和 XML 相同的階層式資料結構，為了表達這樣的關係。因此我們設計了階層符號"／"符號來表達父子關係，如<A><B></B></A>，A 是 B 的父節點，因此 A/B 就可把表示出這樣的階層關係。接下來我們將上述 XQT 做一個整理，如圖二所示。

| XSLT | XQT |
|---|---|
| xsl:stylesheet | <XQT> |
| xsl:variable | $ |
| xsl:template | T( ) |
| xsl:text | t ( ) |
| xsl:element | e ( ) |
| xsl:copy-of | c ( ) |
| xsl:value-of | v ( ) |
| xsl:if | { } |
| xsl:apply-template | a ( ) |

圖二、XSLT與XQT的對照表

## 3.3 範例

我們以 Lu 和 Cheng[4]所設計的查詢處理規則做為範例說明，圖 3 為一份 XML 電話簿的範例，記錄的資料有姓名、電話、年齡、性別、地址及電子郵件。透過圖 4 的處理規則，會從電話簿找出有符合為 cheng-rex 的條件並輸出其姓名、電話、年齡、性別、地址及電子郵件。接著我們進一步的說明，如何將 XSLT 轉換成 XQT。

```
1  <?xml version="1.0" ?>
2  <PhoneBook>
3    <Person>
4      <name>cheng</name>
5      <number>0935000001</number>
6      <age>31</age>
7      <sex>male</sex>
8      <address>Taipei</address>
9      <email>cheng@mail.cyut.edu.tw</email>
10   </Person>
11   <Person>
12     <name>iron</name>
13     <number>0935000002</number>
14     <age>24</age>
15     <sex>male</sex>
16     <address>Taipei</address>
17     <email>iron@mail.cyut.edu.tw</email>
18   </Person>
19   <Person>
20     <name>cheng-rex</name>
21     <number>0935000003</number>
22     <age>25</age>
23     <sex>male</sex>
24     <address>Taichung</address>
25     <email>cheng-rex@mail.cyut.edu.tw</email>
26   </Person>
27 </PhoneBook>
```

圖 3 XML電話簿的範例

圖 4 的範例是由 3 個範本所組成，依據 XQT 的設計，首先將第 1 行的根節點 xsl:stylesheet 轉換成<XQT>標籤。繼續將第 2 行的 xsl:variable，設定函數名稱為 Input_query_name，參數值為 cheng-rex 的$Input_query_name("cheng-rex")函數，如圖 5 的第 2 行所示。繼續將第 3 行的 xsl:template 設定參數值為"/" 的 T("/") 函數以代表 xsl:template，因為 xsl:template 下有子元素 xsl:apply-template，所以用階層符號 / 來區隔父子關係，得到 T("/" / )。第 4 行的 xsl:apply-template 沒有設定參數值，因此用 a( ) 函數來代表。所以第一個範本(3-5 行)的執行結果為 T("/" / a( ))，如圖 5 的第 3 行所示。

第二個範本的結構(6-10 行)，只比第一個範本多了第 7 行<xsl:element>指令，因此 xsl:element 以參數值為"PhoneBook"的 e("PhoneBook")函數來表示。如同第一個範本的做法，可以得到第二個範本(6-10 行)的執行結果為 T("PhoneBook"/e("PhoneBook"/a( )) )，如圖 5 的第 4 行所示。第 11 行的 xsl:template 以 T("Person") 函數來代表，因為 xsl:template 有子元素 xsl:if，所以用階層符號 / 來區隔父子關係，得到 T("Person"/ )。第 12 行的 xsl:if 則以大括號來表示，而大括號內容代表判斷條件 "name=$Input_query_name"，因此得到 T("Person"/{"name=$Input_query_name"})。第 13 行 xsl:element 為 xsl:if 的子元素，所以得到 T("Person/{"name=$Input_query_name"/e("Person")}) 。第 14 行 xsl:copy-of 以參數值為"*"的函數 c("*")來表示，同時 xsl:copy-of 為 xsl:element 的子元素，因此得到第三個範本(11-17 行)的執行結果為 T("Person"/{"name=$Input_query_name"/e("Person"/c("*")) } ) ，如圖 5 的第 5 行所示。

```
1  <xsl:stylesheet version="1.0" xmlns:xsl=
     "http://www.w3.org/1999/XSL/Transform">
2  <xsl:variable name="Input_query_name" select="cheng-rex"/>
3  <xsl:template match="/" >
4    <xsl:apply-templates/>
5  </xsl:template>
6  <xsl:template match="PhoneBook">
7    <xsl:element name="PhoneBook">
8      <xsl:apply-templates/>
9    </xsl:element>
10 </xsl:template>
11 <xsl:template match="Person">
12   <xsl:if test="name=$Input_query_name" >
13     <xsl:element name="Person">
14       <xsl:copy-of select="*"/>
15     </xsl:element>
16   </xsl:if>
17 </xsl:template>
18 </xsl:stylesheet>
```

圖 4 XSLT – 查詢(Query)處理規則

```
1  <XQT>
2  $Input_query_name("cheng-rex")
3  T("/" / a() )
4  T("PhoneBook"/e("PhoneBook"/a()) )
5  T("Person"/{"name=$Input_query_name"/
     e("Person"/c("*")) } )
6  </XQT>
```

圖 5 XQT – 查詢(Query)處理規則

## 4. 分析

經過上述範例的說明，XQT 可以縮減 XSLT 所撰寫的處理規則。因此我們再進一步的分析，針對 Lu 和 Cheng[4]所設計的四種新增、刪除、修改、查詢處理規則，與我們所設計 XQT 來做分析比較。

首先是查詢處理規則如圖 4 所示，這份處理規則它會從電話簿(如圖 3)查詢，找出有符合為 cheng-rex 的條件並輸出其姓名、電話、年齡、性別、地址及電子郵件，結果如圖 5 所示。新增處理規則，它是在電話簿中進行新增姓名、電話、年齡、性別、住址及電子郵件等資料。刪除處理規則，它是從電話簿中將姓名符合為 cheng 的資料，將其姓名、電話、年齡、性別、住址及電子郵件等個人資料刪除。修改處理規則，它是修改電話簿中，姓名、電話及年齡等個人資料。經由 XQT 的處理後，這四種新增、刪除、修改、查詢處理規則其字元數分別為新增從 1048 字元降到 338 字元，刪除從 390 字元降到 118 字元，修改從 587 字元降到 211 字元，查詢從 413 字元降到 141 字元。透過圖 6，我們可以看見 XQT 確實可以大幅縮減 XSLT 所撰寫的處理規則。就上述的字元數來說，使用 XQT 所撰寫的處理規則只有原來 XSLT 的三分之一左右的大小，因此使用 XQT 來撰寫處理規則，確實可以有效縮小檔案。

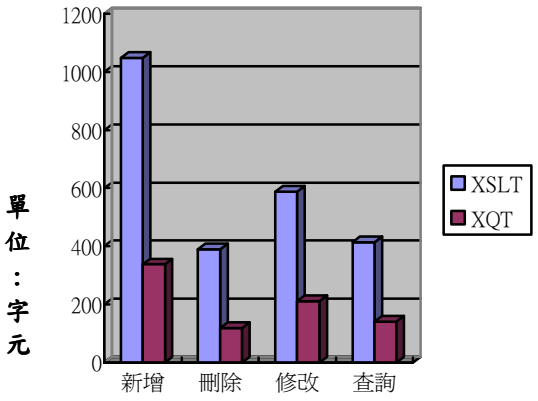


圖 6 XSLT 與 XQT 的比較

## 5. 結論與未來工作

經由分析結果得知，使用本研究所提出的 XQT 來撰寫處理規則的檔案，確實會比用 XSLT 來的小。同時 XQT 也符合 XML 的規範，因此也有 XML 跨平台的優點。所以使用 XQT 將能夠解決 XSLT 來做為處理規則所產生佔用記憶體空間及使用過多頻寬的問題。雖然 XQT 具有上述的優點，但是對一般人而言，要利用 XQT 來撰寫處理規則是一件不容易的事。因此我們未來的工作將提供一個處理規則產生器，它是一種圖形化的介面，可以讓使用者輕鬆的撰寫處理規則以進一步提昇可用性, 並完成 XQT 的資料處理器以進一步做效能評估。

## 參考文獻


[1] W3C XSLT version 1.0,1999 . http://www.w3.org/TR/xslt

[2] Barnes,S.J.,“The mobile commerce value chain: analysis and future developments,”International Journal of Information Management, Vol. 22, pp. 91-108, 2002.

[3] D. Florescu and D. Kossmann , “Storing and Querying XML data using an RDBMS,” IEEE Data Engineering Bulletin, Vol. 22, No. 3, pp. 27-34, 1999.

[4] Eric Jui-Lin Lu, and Yung-Yuan Cheng, “Design and Implementation of a Mobile Database for Java Phones,”Computer Standards and Interfaces, Standards and Interfaces, Vol. 26, No. 5, pp. 401-410, September. 2004.

[5] J. Shanmugasundaram, K. Tufte, G. He, C. Zhang, D. De-Witt, and J. Naughton, “Relational databases for querying XML documents: limitations and opportunities,”In Proceedings of VLDB, Edinburgh, UK, pp.

302-314, September 1999.

[6] J. Shanmugasundaram, J. Kiernan, E. J. Shekita, C. Fan, and J. Funderburk,"Querying XML Views of Relational Data," In Proceedings of VLDB, 2001.

[7] J. Shanmugasundaram, E. Shekita, R. Barr, M. Carey, B. Lindsay, H. Pirahesh, and B. Reinwald, "Efficiently Publishing Relational Data as XML Documents,"In VLDB, 2000.

[8] Jixue Liu, Millist Vincent, "Querying relational databases through XSLT," Data & Knowledge Engineering, Volume: 48, Issue: 1, pp.103-128, January 2004.

[9] Karlsson, J. S., Lal, A., Leung, C., and Pham, T., "IBM DB2 everyplace: a small footprint relational database system," Proceedings of 17th International Conference on Data Engineering, Heidelberg, Germany, pp. 230-232, April 2001.

[10] Kinoshita, M., "DoCoMo's vision on mobile commerce," Proceedings of the 2002 Symposium on Applications and the Internet, Nara, Japan, pp. 39-40, January 2002.

[11] Kuramitsu, K., and Sakamura, K.,"Towards ubiquitous database in mobile commerce," Proceedings of the Second ACM International Workshop on Data Engineering for Wireless and Mobile Access, Santa Barbara, California, USA, May 2001.

[12] Lawton, G., "Moving java into mobile phones, " Computer, Vol. 35, pp. 17-20, 2002.

[13] M. Fern_andez, D. Suciu, and W.C. Tan, "SilkRoute: Trading Between Relations and XML,"In WWW9, 2000.

[14] Madria, S. K., Mohania, M., Bhowmick, S. S., and Bargava, B., "Mobile data and transaction management," Information Sciences, Vol. 141, pp. 279-309, 2002.

[15] Ortiz, S., "Embedded databases come out of hiding," Computer, Vol. 33, pp. 16-19, 2000.

[16] Ronald Bourret, "Mapping DTDs to Databases, "http://www.xml.com/lpt/a/2001/05/09/dtdtodbs.html.

[17] S. Jain, R. Mahajan, and D. Suciu, "Translating XSLT Programs to Efficient SQL Queries," In Proceedings of WWW, 2002.

[18] Shepard, S. J., "Embedded databases can power emerging world of information to go," IT Professional, Vol. 2, pp. 10-13, 2000.

[19] Tsalgatidou, A., and Pitoura, E., "Business models and transactions in mobile electronic commerce: requirements and properties, " Computer Networks, Vol. 37, pp. 221-236, 2001.

[20] Varshney, V., and Vetter, R., "Mobile commerce: framework, applications and networking support," Mobile Networks and Applications, Vol. 7, pp. 185-198, 2002.