

A Relational-XML data warehouse for data aggregation with SQL and XQuery

Joseph Fong, Herbert Shiu and Davy Cheung
Computer Science Department, City University of Hong Kong, Hong Kong
Email: csjfong@cityu.edu.hk

Abstract:

Integration of multiple data sources is becoming increasingly important for enterprises that cooperate closely with their partners for e-commerce. OLAP enables analysts and decision makers fast access to various materialized views from data warehouses. However, many corporations have internal business applications deployed on different platforms. No standard solution for integration exists, except to develop an autonomous system for large enterprises. To enable business intelligence query activities, a corporation needs to build a data warehouse on top of a middleware to aggregate summarized data from various heterogeneous database systems. Since XML documents are a common data representation standard on the Internet — and relational tables are traditional production data — OLAP must handle both relational and XML data on the World Wide Web. SQL and XQuery can be used to process the materialized relational and XML data cubes of aggregated data. This paper aims to facilitate handling these two data cubes from a Relational-XML data warehouse with ETL (Extract, Transformation and Loading).

Keywords: data warehouse, XML data cube, Relational data cube, Aggregation data, XQuery, SQL, schema integration, data integration, metadata

1 Introduction

To acquire business intelligence and analyses, a corporation must build a data warehouse to aggregate data (compute total) from a variety of heterogeneous database systems and sources. The traditional approach has been to build a relational data warehouse by using a full set of data warehouse tools of middleware, messaging and ETL tools. This paper offers a methodology to develop a heterogeneous relational-XML data warehouse by using OLAP in SQL and XQuery.

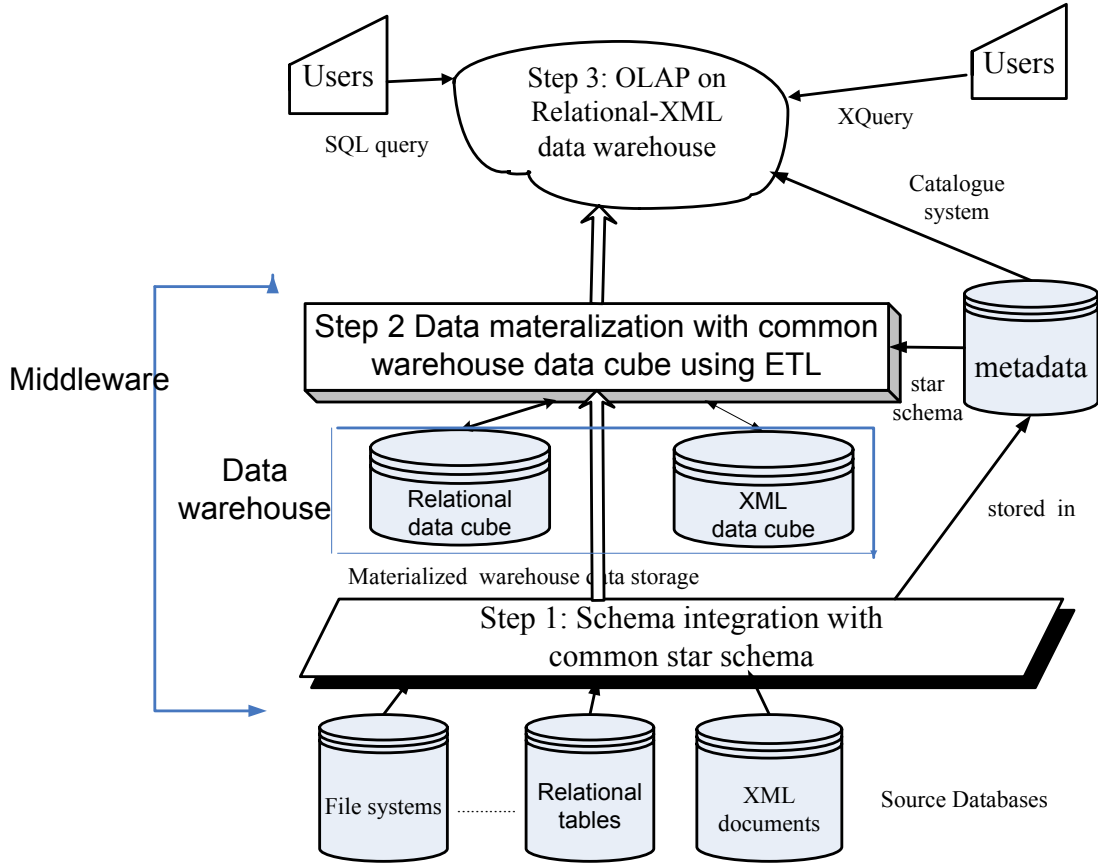
An ETL methodology is provided to summarize information from heterogeneous database systems using relational and XML data. The approach is to integrate a company's business processes with its subsidiaries, partners, customers and suppliers. Thus, the system exposes data extraction, data integration and data materialization on the middleware to facilitate the following:

- To evaluate the use of schema translation between relational and XML databases
- To integrate Relational tables and XML documents for information aggregation
- To access the use of Relational-XML as a data warehouse
- To represent materialized views of XML data using XQuery[18], and of relational tables using SQL.

To tackle these issues, a data warehouse on top of a middleware is a solution for system developers for such situations. Middleware integrates all the business applications to perform on-demand data integration, and stores the integrated data into a relational and an XML database. Two approaches exist to handle data integration on the middleware. The first approach is direct database access, which acquires direct access and knowledge of the schema of the targeted database systems. The second is service call, through which each physically-separated business application communicates with the middleware by web service. These individual applications maintain their information sharing based on their own criteria, which is therefore a costly solution. Yet, it solves the case of flat or hierarchical structures of some legacy database systems as well as security issues enforced for political reasons.

Both approaches have different strengths and weaknesses. Hence, this paper proposes an optimized solution through a hybrid implementation. It can balance the strengths and weaknesses of different approaches so that data integration is feasible, in which the system resolves the conflicts between entities and relationships from the metadata using predefined rules. The several XML documents and relational tables are merged together conceptually into a relational or an XML data cube based on the relations in the metadata.

Information aggregation is a technology that gathers relevant information from a variety of data sources to help corporations effectively analyze the aggregated information using Internet technologies. The desired structure of information aggregation is a Relational-XML data warehouse that can provide both Relational and XML data cubes to the users as shown in Figure 1.



Note: a data cube is a denormalized database for data warehouse

Figure 1 Architecture of a Relational-XML Data Warehouse

2 Related work

Much research has been done in the information aggregation area. Zhu[1] categorized information aggregation into three categories based on a survey of over 100 existing and emerging web aggregators that provided information aggregation services:

1. comparison aggregation,
2. relationship aggregation, and
3. intra-organization and inter-organization aggregation

Madnick and Siegel[2] discussed each aggregation category with examples and the after-aggregation analysis; that is, how to make use of the aggregated information to improve the business. However, they did not explain how to extract the information being integrated together from different resources — especially from the mixed data sources of relational data.

Website[3] claimed that data integration and aggregation are important in financial services. Financial service providers (FSP) rely on the aggregated information to make decisions and drive their financial products and services. A major cost for FSP lies in data integration. With XML technology, data integration is simplified[24]. Also, Website[3] suggested that having an XML-based mid-tier operational data server is a flexible data aggregation solution, and proposed an approach towards such a data server and explained why a RDBMS was not the choice. However, no explanation was provided on how to integrate the relational data into an XML-based mid-tier data server.

Website[4] demonstrated how Stylus Studio queries the real-world data from a stock-quote Web service and combines the historical data stored in a relational database for information aggregation processing. They showed how to make an application to transform the relational schema into an XML schema so that relational data are mapped into an XML document. However, this application is not an open source and so, from the academic point of view, it cannot be studied (though it did provide operational experience).

Website[5] forecast that mid-tier aggregation will continue to grow — from \$44.7 million in 2004 to a projected \$1.8 billion by 2009 — whereas this paper discusses how such mid-tier aggregation can be developed to handle the real-time data aggregation of heterogeneous database systems. The data sources can be relational and XML database systems.

Chaudhri et al[20] gave a detailed description on how to design and manage the XML data warehouse, and developed a system called DAWAX. One problem in DAWAX is that it stores all XML documents as relational data. Chaudhri et al did not explain how to ensure that the original XML document is recovered.

Xyleme[21] is an XML data warehouse system that stores all data on the web as XML documents. A special-purpose DBMS called NATIX is used to store all XML data. In order to have a unified view of the data in the data warehouse, Xyleme provides the domain ontology model as abstract DTD and defines mapping rules between the DTD of the stored documents and the abstract DTD. Since Xyleme stores all XML documents into a domain, the storage space is a challenge.

Pedersen[22] introduced a federated query language SQL_{XM} that allows XML data to be used directly in an OLAP query to furnish multidimensional cubes with external XML data, and to group and select cube data based on XML data values. Park[23] proposed a new multidimensional expression language for XML cubes, XML-MDX, which used XQuery expressions to specify the measured data, axis dimensions and slicer. Madnick[24] used COIN (COntext INterchange) to capture context knowledge and improve data quality by reconciling semantic differences between the sources and the receivers. Kimball[25] conformed heterogeneous data from multiple sources into standardized dimension tables and fact tables.

Besides Fong[14], who proposed an object-relational data warehousing, most research work has focused on developing an XML data warehouse. This paper contributes by adding a relational view in addition to an XML view of a data warehouse.

3 Methodology

In general, middleware[6] applies many technologies to handle information aggregation[7] in heterogeneous database system environments, and hosts a global database schema that represents the schemas of multiple heterogeneous database systems in distributed environments. The global schema stores the detailed information of the database systems of host locations and database types. Middleware makes use of this information to perform user queries by distributing the translated sub-queries to the corresponding database system.

In the case of a legacy system with proprietary database or third party systems, queries can be handled by the web service. Once the sub-queries are performed, in cases of an XML data cube, all of the results are translated into XML documents that can be aggregated into an XML document. In cases of a relational data cube, all of the results are translated into relational tables using an aggregation. Finally, the aggregated set of XML documents or relational tables is represented in the web database.

The following is a stepwise semi-automated middleware process shown as a methodology:

Step 1: Map and integrate source schemas into Metadata

- Map XML schemas into a metadata
- Map relational schema into a metadata
- Integrate schemas between source database schemas
- Map source metadata schemas into an integrated metadata schema

Step 2: Implement star schema[25] and Data Cubes for Aggregation

- Create dimension table by mapping table and attributes to a global metadata schema
- Create fact table by mapping table and attributes to a global metadata schema

- Create star schema by relating the dimensions and fact tables
- Convert and load relational data into a relational data cube
- Convert and load XML documents into an XML data cube

Step 3: Aggregate Data

- Aggregate data in a materialized Relational or XML data cube using OLAP[15]

Metadata is data about data. The structure of middleware metadata includes three sections: source database metadata, global schema metadata, and star schema metadata. The source database schema is recognized by the middleware, including database type, tables, attributes, relations and keys. Since the middleware integrates multiple heterogeneous database systems, a separate set of metadata is needed to describe all the source database schemas in order to integrate them. A wizard was prepared to guide users to complete source metadata creation — which requires good technical knowledge — so a database administrator or super user with knowledge of database structures would probably accomplish the metadata setup better. Relational metadata is applied to develop a Relational-XML data warehouse for data catalog and internal directory usage. XML schemas and documents are applied to build the data cube for the data warehouse.

For example, a metadata can be a data dictionary, a data catalog, or a data directory for a database. The metadata schema stores the source database schemas as shown in figure 2.

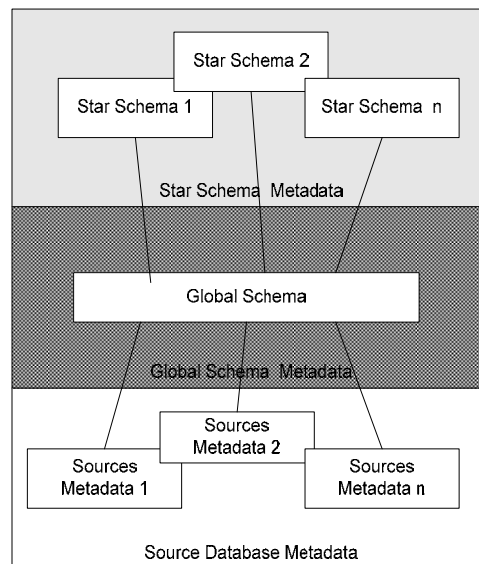


Figure 2 The three types of metadata

Step 1 Map source schemas into a metadata

The methodology is a bottom-up approach that integrates existing relational data and XML documents into a metadata for building the Relational-XML data cube. The XML schema is mapped into a metadata[9] to provide a global view of the heterogeneous database systems. To map an XML schema into a metadata, an Extended Entity Relationship model is extracted from the XML schema.

Mapping XML schema into source metadata:

The following are the rules on mapping an XML schema into a relational schema[8]:

- ❑ Mapping Elements to Tables
 - All of the XML elements found in XML documents are mapped to metadata tables.
- ❑ Mapping Attributes to Columns
 - The attributes of the XML elements are mapped to the columns of the metadata tables.
- ❑ Mapping XML Elements with Relationships to relational tables
 - The parent element and its child elements are mapped into a parent relational table and child table in the metadata tables.

Mapping Relational schema into source metadata

Similarly, a relational schema can be mapped into the metadata table.

As a result, both the XML schema and relational schema can be mapped into a metadata schema as shown in Figure 3. Primary keys are underlined in a metadata schema; foreign keys have “*” as their prefix, and types of operation include the insert, update and delete procedures of an operation.

Relation Meta_database (database_id, database_name, database_type, drive_string, jdbc_url, jdbc_username, jdbc_password)

Relation Meta_table (table_id, *database_id, parent_id, table_name, primary_key, operation_type)

Relation Meta_attribute (*table_id, attribute_name, attribute_type, *method_name, cardinality)

Relation Meta_method (method_name, parameters, web_service, description)

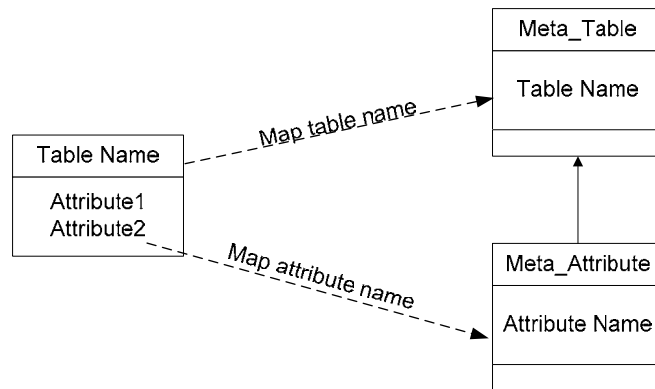


Figure 3 Mapping a relational table name and attribute to metadata

Case Study

A case study shows three different applications in which each one connects to a different backend database system. They are; a customer relationship management system (CRM); a warehouse management system (WMS) and an order management system (OMS) with databases of relational, XPath and XML DB[17] types, respectively. Figure 4 shows the overview of the system. In this case study, a star schema is created to find the total sales of orders along with items and customers.

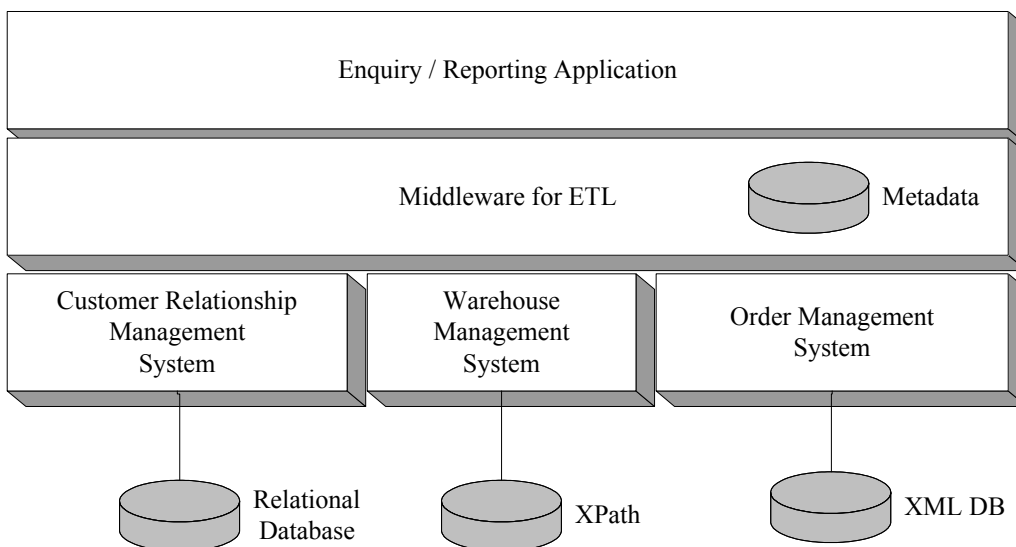


Figure 4 System overview of the case study

Step 1: Map and integrate source schemas into Metadata

- Metadata of the systems

Table 1 shows the data dictionary of different database schemas in the case study:

Table 1 Metadata

<i>Application</i>	<i>Database Type</i>	<i>Columns</i>	<i>Particulars</i>
CRM	Relational	ACCOUNT_ID	Customer account ID
CRM	Relational	AMOUNT_OWED	Amount owed
CRM	Relational	AVAILABLE_CREDIT	Credit limit
WMS	XML	ITEM_ID	Product ID
WMS	XML	ITEM_DESC	Product description
WMS	XML	QUANTITY	Available quantity
WMS	XML	UNIT_PRICE	Product unit price
OMS	XML	OrderID	Sales order ID
OMS	XML	CustomerID	Customer ID
OMS	XML	Ordered Date	Order date
OMS	XML	ShipmentDate	Shipment date
OMS	XML	ShipmentLocation	Shipment destination
OMS	XML	TotalAmount	Total order amount
OMS	XML	DeliveryMethod	Delivery method
OMS	XML	Priority	Priority
OMS	XML	OrderStatus	Order status
OMS	XML	ItemID	Product ID
OMS	XML	Quantity	Ordered product quantity
OMS	XML	ItemStatus	Ordered product status

To setup the metadata of the source databases, the user must understand the remote databases' schema structures and tables' relations. The simplest way to set up this configuration is to follow the wizard walkthrough to enter the required content step-by-step.

Metadata Database

Database id	Database name	Database type	Jdbc url
ID_CRM	CRM	RELATIONAL	jdbc:mysql://localhost/crm
ID_OMS	OMS	XML	XMLdb:xindice://localhost/db/oms
ID_WMS	WMS	XML	XMLdb:xindice://localhost/db/wms

Metadata TABLE

TABLE ID	PRIMARY KEY	TABLE NAME	DATABASE ID	PARENT ID
ID_CUSTOMER	ACCOUNT_ID	CUSTOMER	ID_CRM	
ID_SALES_ORDER	ORDER_ID	SALES_ORDER	ID_OMS	
ID_ORDER_DETAIL	ORDER_ID	ORDER_DETAIL	ID_OMS	ID_SALES_ORDER
ID_INVENTORY	ITEM_ID	INVENTORY_ITEM	ID_WMS	

Metadata ATTRIBUTE

ATTRIBUTE_NAME	TABLE ID	ATTRIBUTE_TYPE	Cardinality
ACCOUNT_ID	ID_CUSTOMER	TEXT	1
AMOUNT_OWED	ID_CUSTOMER	TEXT	1
AVAILABLE_CREDIT	ID_CUSTOMER	NUMERIC	1
ITEM_ID	ID_INVENTORY	NUMERIC	1
ITEM_DESC	ID_INVENTORY	TEXT	1
QUANTITY	ID_INVENTORY	NUMERIC	1
UNIT_PRICE	ID_INVENTORY	NUMERIC	1
OrderID	ID_SALES_ORDER	NUMERIC	1
CustomerID	ID_SALES_ORDER	NUMERIC	1

Ordered Date	ID SALES ORDER	TEXT	1
ShipmentDate	ID SALES ORDER	TEXT	1
ShipmentLocation	ID SALES ORDER	TEXT	1
TotalAmount	ID SALES ORDER	NUMERIC	1
DeliveryMethod	ID SALES ORDER	TEXT	1
Priority	ID SALES ORDER	TEXT	1
OrderStatus	ID SALES ORDER	TEXT	1
OrderID	ID ORDER DETAIL	NUMERIC	1
ItemID	ID ORDER DETAIL	NUMERIC	1
Quantity	ID ORDER DETAIL	NUMERIC	1
ItemStatus	ID ORDER DETAIL	TEXT	1

Map metadata into a global metadata schema

The source metadata is used as input to create the global metadata schema of the middleware. The integration of the global schema resolves conflicts in the database tables. The conflict resolution rules are stored in the global metadata for data integration. The setup of global metadata is the same as the source metadata setup. To smooth the process, a wizard with question and answer walkthrough is employed.

Different source database schemas might contain the same tables for the representation of their own system. The following are guidelines on how to perform schema integration between two different database schemas.

Schema Integration between Source Database Schemas[16]

To map the relational database schema into metadata, several procedures are available in the mapping process for recovering data semantics[9]. The integrated schema is a global view of heterogeneous databases, consisting of multiple enterprise applications of a company. Each local database system is treated as a unit object for integration. The objective of schema integration is to ensure no duplicated data exist. The identified conflicts are resolved by storing them in a global metadata schema.

□ *Resolve semantic conflicts among source schemas*

This resolves the definition-related conflicts of inconsistency in key or synonyms and homonyms. This is a kind of Enterprise Application Integration (EAI) process among different applications. The data type conflicts among schemas are captured in metadata as relationship mapping.

□ *Merge entities and relationships*

The merging of entities can be automated using union operators if their domains are the same. By identifying the same keys with the same entity name in different database schemas, the entities are merged by union as follows:

- Merge relationship by cardinality
- Merge entities by subtype relationship
- Merge entities by generalization
- Merge entities by aggregation

The integrated metadata schema is mapped with heterogeneous source schemas and presented as a simple database schema in the middleware. It is useful for analytical tasks with an integrated database schema. The integrated table defines the integrated view information, and the integrated field defines mapping of integrated fields and source fields, and the mapping rules describe the method(s) for handling data type conflicts in source fields mapping.

Integrated Metadata schema Setup Wizard:

```

Begin
  For each metadata table do
    Begin
      Create global metadata table with table name and key attributes;
      Select available tables list from source Metadata;
      Map available attributes from source Metadata to integrated schema;
    End
  End

```

```

For each attribute do
  Add conflict resolve rules based on available selections;
End
End

```

The integrated schema is:

Relation meta_join_key (join_key_id, table, attribute, key_type, *mapping_rule)

Relation meta_gloabl_table (global_table, *join_key_id, table, attribute)

Relation meta_global_field (*global_table, global_field, local_table, local_field, field_type, *mapping_rule)

Relation meta_mapping_rule (mapping_rule, rule_name, type, rule)

Global schema extracted from an integrated schema

The global schema is mapped from the distributed data sources of the source metadata. To accomplish this metadata setup, user supervision — with knowledge of the source schema of the metadata — is needed.

GLOBAL TABLE

GLOBAL TABLE	GLOBAL FIELD NAME	JOIN FIELD ID	KEY TYPE
CUSTOMER	ACCOUNT_ID	ACCOUNT_ID	PRIMARY
SALES ORDER	CUSTOMER_ID	CUSTOMER_ID	FOREIGN
SALES ORDER	ORDER_ID	ORDER_ID	PRIMARY
ORDER_DETAIL	ORDER_ID	ORDER_ID	PRIMARY
ORDER_DETAIL	ITEM_ID	ITEM_ID	FOREIGN
INVENTORY	ITEM_ID	ITEM_ID	PRIMARY

GLOBAL FIELD

GLOBAL FIELD	LOCAL TABLE	LOCAL FIELD	FIELD TYPE	RULE
CUSTOMER_ID	CUSTOMER	ACCOUNT_ID	NUMERIC	SYNONYM
AMOUNT_OWED	CUSTOMER	AMOUNT_OWED	TEXT	
AVAILABLE_CREDIT	CUSTOMER	AVAILABLE_CREDIT	NUMERIC	
ITEM_ID	INVENTORY	ITEM_ID	NUMERIC	
ITEM_DESC	INVENTORY	ITEM_DESC	TEXT	
QUANTITY	INVENTORY	QUANTITY	NUMERIC	
UNIT_PRICE	INVENTORY	UNIT_PRICE	NUMERIC	
ORDER_ID	SALES ORDER	ORDERID	NUMERIC	SYNONYM
CUSTOMER_ID	SALES ORDER	CUSTOMERID	NUMERIC	SYNONYM
ORDER_DATE	SALES ORDER	ORDERDATE	TEXT	SYNONYM
SHIPMENT_DATE	SALES ORDER	SHIPMENTDATE	TEXT	SYNONYM
SHIPMENT_LOCATION	SALES ORDER	SHIPMENTLOCATION	TEXT	SYNONYM
TOTAL_AMOUNT	SALES ORDER	TOTALAMOUNT	NUMERIC	SYNONYM
DELIVERY_METHOD	SALES ORDER	DELIVERYMETHOD	TEXT	SYNONYM
PRIORITY	SALES ORDER	PRIORITY	TEXT	
ORDER_STATUS	SALES ORDER	ORDERSTATUS	TEXT	SYNONYM
ORDER_ID	ORDER_DETAIL	ORDERID	NUMERIC	SYNONYM
ITEM_ID	ORDER_DETAIL	ITEMID	NUMERIC	SYNONYM
QUANTITY	ORDER_DETAIL	QUANTITY	NUMERIC	
ITEM_STATUS	ORDER_DETAIL	ITEMSTATUS	TEXT	SYNONYM

XML Schema Definition (XSD) of OMS system

```

<?XML version = "1.0" encoding = "UTF-8"?>
<xs:schema XMLNs:xs = "http://www.w3.org/2001/XMLSchema">
  <!-- SalesOrder definition -->
  <xs:element name = "SalesOrder">
    <xs:complexType>
      <xs:attribute name = "CustomerID" type = "xs:string"/>
      <xs:attribute name = "OrderDate" type = "xs:string"/>
      <xs:attribute name = "TotalAmount" type = "xs:decimal"/>
      <xs:attribute name = "OrderStatus" type = "xs:string"/>
      <xs:attribute name = "OrderID" type = "xs:string"/>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

Convert element to table name

```

<xs:attribute name = "ShipmentDate" type = "xs:string"/>
<xs:attribute name = "ShipmentLocation" type = "xs:string"/>
<xs:attribute name = "DeliveryMethod" type = "xs:integer"/>
<xs:attribute name = "Priority" type = "xs:string"/>
</xs:complexType>
</xs:element>

<!-- item definition-->
<xs:element name = "LineItem">
  <xs:complexType>
    <xs:attribute name = "Item" type = "xs:string"/>
    <xs:attribute name = "Quantity" type = "xs:integer"/>
    <xs:attribute name = "UnitPrice" type = "xs:decimal"/>
    <xs:attribute name = "ItemStatus" type = "xs:string"/>
  </xs:complexType>
</xs:element>
</xs:schema>

```

Convert attribute into column name

Document Type Definition (DTD) of WMS system

```

<!DOCTYPE INVENTORY_ITEM [
<!ELEMENT INVENTORY_ITEM (ITEM+)>
<!ELEMENT ITEM (ITEMID,ITEM_DESC?,QUANTITY,UNIT_PRICE)>
<!ELEMENT ITEMID (#PCDATA)>
<!ELEMENT ITEM_DESC (#PCDATA)>
<!ELEMENT QUANTITY (#PCDATA)>
<!ELEMENT UNIT_PRICE (#PCDATA)>
]>

```

Convert document type to table name

Convert elements into columns

Step 2 Implement the star schema and data cubes for Aggregation

Once a global metadata is built, data cubes are developed from the derived integrated database metadata. Schema integration [9] provides a global view of multiple schemas of different database systems. The global schema design of this middleware employs a bottom-up approach to integrate existing database schemas into a global database schema. A metadata structure of a star schema is shown in Figure 5. A wizard with walkthrough approach guides the user during the creation process. The metadata of a star schema contains a star schema of each OLAP cube. Each star schema contains a fact table element and related dimension tables. These fact table and dimension tables are used in OLAP cube formation.

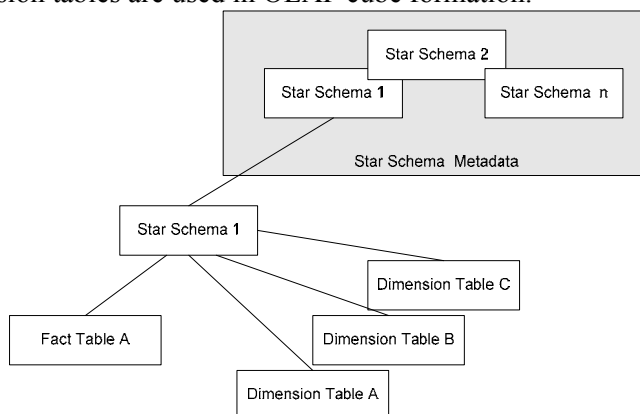


Figure 5 Metadata structure of a star schema

Walkthrough Process:

```
Begin
  For each cube do
    Begin
      Select available tables from global metadata;
      Create fact table by mapping tables and attributes in global metadata schema by the user
      defining the measurement fields;
      Select available tables from global metadata related to fact table;
      For each dimension do
        Begin
          Create dimension table by mapping tables and attributes in global schema by user defining
          the dimension;
          Map the dimensions into fact table and store the relations in star schema table;
        End
      end
    end
  end
```

The heterogeneous database systems are integrated in the form of a source database schema and stored in global metadata. Once the schemas are integrated, there will be no duplicate data in the global schema. The information concerning databases, tables, attributes, and methods is stored in the source metadata. The information is mapped into the star schema.

The Star schema metadata is as follows:

```
Relation meta_fact_table      (star_schema_id, star_name, fact_name)
Relation meta_fact_table_dtl  (*star_schema_id, global_table, global_field, data_field, measure_name)
Relation meta_dimension_table (dimension_id, *star_schema_id, dimension_name)
Relation meta_dimension_table_dtl (*dimension_id, global_table, global_field, data_field, level_desc,
                                  level_no)
```

Creation of star schema

To enable multi-dimensional queries, a star schema is applied in this system. The relational schema shows the star schema metadata of the system. The star schema metadata includes tables of fact table and dimension tables. The star schema is designed to aggregate the information from multiple data sources. Since the star schema selecting data is based on user requirements, user supervision is involved in the star schema design.

Creation of data cube [10]

To enable multi-dimensional queries, OLAP query language – which focuses on data aggregation instead of on data analysis – is required.

Star Schema for Aggregation

In this case study, three different database schemas are mapped into a metadata — the relational schema of CRM, XML Schema of OMS, and DTD of WMS.

Translate XML schema definition into relational schema

XML schema definition is translated into relational schema and then stored into metadata.

Map XML schemas OMS and WMS into Relation Sales_order and Relations Inventory_Item.

Map relational database schema into metadata

Once all schemas are translated into relational schemas, the schemas are stored in metadata.

Map metadata into star schema

After the schema integration, the global view of all database schemas is represented in figure 6

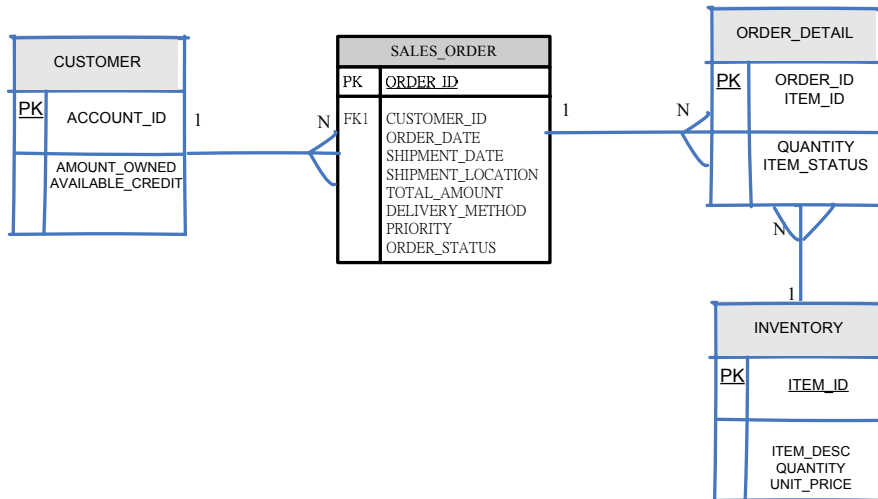


Figure 6 The global view of integrated schemas

Based on user requirements, the star schema is created in figure 7.

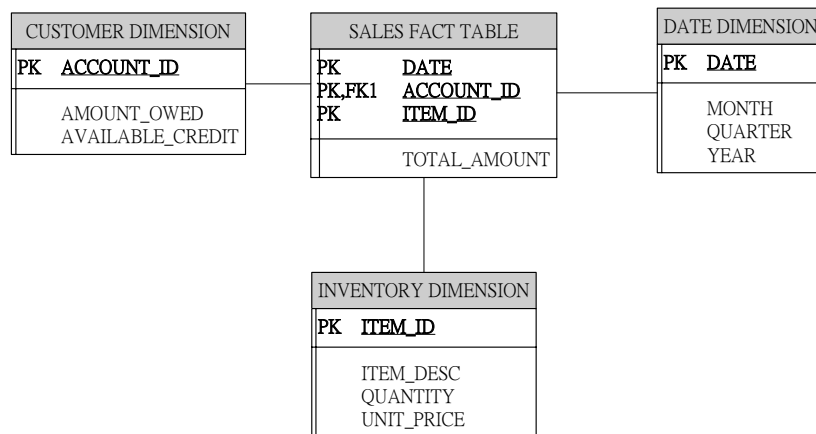


Figure 7 A star schema of the case study

The schema is designed to query sales facts of customers. It contains customer dimension, inventory dimension, date dimension and an aggregated sales fact table that combines sales order and order detail tables. This information is stored in the metadata tables of the star schema as shown below.

Dimension table metadata in Meta Dimension Table

SCHEMA	DIMENSION	DIMENSION FIELD	GLOBAL TABLE	GLOBAL FIELD
Sales	Inventory	ITEM_DESC	INVENTORY_ITEM	ITEM_DESC
Sales	Inventory	QUANTITY	INVENTORY_ITEM	QUANTITY
Sales	Inventory	UNIT_PRICE	INVENTORY_ITEM	UNIT_PRICE
Sales	Customer	AMOUNT_OWED	CUSTOMER_ACCOUNT	AMOUNT_OWED
Sales	Customer	AVAILABLE_CREDIT	CUSTOMER_ACCOUNT	AVAILABLE_CREDIT

Fact table metadata of the Star schema

SCHEMA	DIMENSION	GLOBAL TABLE	GLOBAL FIELD	MEASURE
Sales	Inventory	SALES_ORDER	ITEM_ID	N
Sales	Customer	INVENTORY	ACCOUNT_ID	N
Sales	Sales	SALES_ORDER	TOTAL_AMOUNT	Y

Loading data into the Relational-XML Data Warehouse

After building a star schema, data are loaded into a data cube according to the derived star schema. The system aims to aggregate data information from heterogeneous databases or applications. Once the metadata of global schema and star schema are completed, the information is used to load data from multiple data

sources. Since the data loading consists of heterogeneous database systems, several processes are performed to build the XML data warehouse automatically.

(i) XML data cube of the Relational-XML Data Warehouse

□ Data Centric Schema in an XML Data Warehouse

The schema of this XML data warehouse has a data centric design. The Relational-XML data warehouse is used to store integrated data for OLAP processing. XQuery is used to retrieve data from XML documents.

□ Retrieve fact table and dimension tables

Algorithm for loading XML document into XML data cube:

Fact table creation Algorithm

Begin

Retrieve fact table from star schema metadata;

Retrieve global tables, attributes and mapping rules from global metadata;

Retrieve source tables, attributes and mapping rules from sources metadata;

For each source database do

Begin Select the targeted tables in the database

For each targeted table do

Begin Retrieve required attributes' data by generating local queries to the database

Transform the queried results into XML documents

End

Integrate the XML documents using XQuery

End

Integrate the XML documents using XQuery of different databases;

If fact table exists

Then append full integrated XML into XML databases into fact table

Else create full integrated XML into XML databases into fact table

End

Dimension tables creation

Begin

Retrieve dimension from star schema metadata;

Retrieve global tables, attributes and mapping rules from global metadata;

Retrieve source tables, attributes and mapping rules from sources metadata;

For each source database do

Begin Select the targeted tables in the database

For each targeted table do

Begin Retrieve required attributes' data by generating local queries to the database;

Transform the queried results into XML documents;

End

Integrate the XML documents of different tables using XQuery;

End

Integrate the XML documents of different databases using XQuery;

Store the full integrated into XML databases into dimension directory;

End

□ Data retrieval

The middleware retrieves the source database information using the metadata database. Based on the metadata, the system traces back the original database, table and attributes from the metadata of the star schema, global schema and source database schema.

The loading of relational data is a projection based on primary key matching. If there are duplicate attributes, one of them will be ignored. If there are new attributes, they will be inserted into the element(s) that they match as shown in Figure 8.

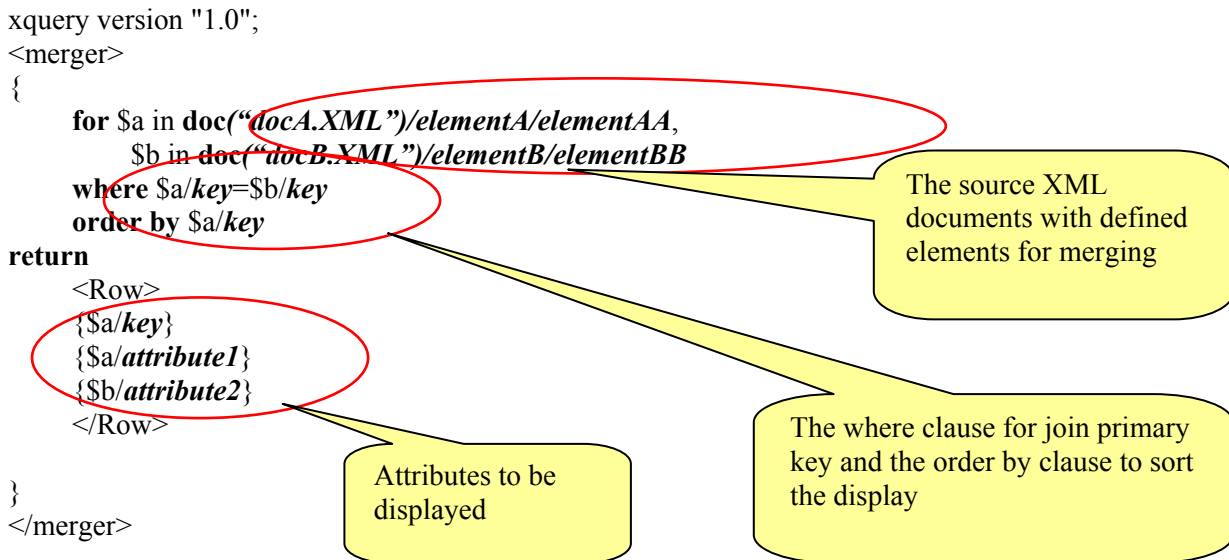


Figure 8 XQuery structure of XML Documents merger

❑ **Data loading into XML DB**

After the data integration process has been accomplished, the retrieved data from heterogeneous data sources are loaded into an XML data warehouse. The XML DB schema is a tree structure.

XML Schema for Data Warehouse

```

<?XML version="1.0" encoding="UTF-8"?>
<xs:schema XMLNs:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="element_fact">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="Row"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Row">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="attribute1"/>
        <xs:element ref="attribute2"/>
        <xs:element ref="attributen"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="attribute1" type="xs:integer"/>
  <xs:element name="attribute2" type="xs:integer"/>
  <xs:element name="attributen" type="xs:string"/>
</xs:schema>

```

(ii) Relational data cube of the Relational-XML Data Warehouse

❑ **Retrieve fact table and dimension table**

The algorithm for loading relational data into a relational data cube is similar to the algorithm for loading XML documents into an XML data cube by replacing XML document and XQuery with relational tables and SQL.

Retrieve target data fields from source relational tables:

A SQL SELECT statement can be issued to merge two relational databases tables A and B into a metadata as shown in Figure 9.

```
Select Ra_A1, Ra_A2, Rb_A3 from Ra, Rb where Ra_A1=Rb_A1
Insert Rx (A1, A2, A3) Value (a11, a21, null)
Insert Rx (A1, A2, A3) Value (a12, a22, null)
Insert Rx (A1, A2, A3) Value (a13, null, a31)
Insert Rx (A1, A2, A3) Value (a14, null, a32)
```

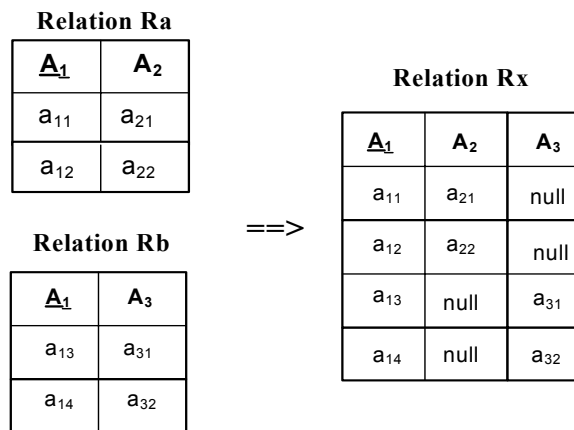


Figure 9 SQL structure of Relational tables merger

Case study: Loading data according to XML or Relational view

As the global database schema and star schema are created, the mapped information in the metadata is used to perform the data aggregation. The database data for the case study appear below.

□ Data from CRM system

CUSTOMER

ACCOUNT_ID	AMOUNT_OWED	AVAILABLE_CREDIT
100001	1000.00	4000.00
100002	2000.00	3000.00
100003	5000.00	0

□ Data from WMS system

INVENTORY ITEM

ITEM ID	ITEM DESC	QUANTITY	UNIT PRICE
1	A01	30	10.00
2	B01	20	20.00
3	C01	10	20.00
4	C02	15	25.00
5	C03	15	20.00

□ Data from OMS system

SALES ORDER

ORDER ID	CUSTOMER ID	TOTAL_AMOUNT	... (OMITTED)
1001	100001	2000.00	...
1002	100001	2000.00	...
1003	100002	1000.00	...

ORDER DETAIL

ORDER ID	ITEM ID	QUANTITY	... (OMITTED)
1001	2	10	...
1002	2	10	...
1003	3	10	...

❑ Construct local database queries

The middleware constructs local database queries according to the target database system. In this case study, three different types of queries were constructed: relational query, XML query and web database.

Retrieve target tables fields from metadata

Retrieve dimension field information from metadata

```
-- Tables and fields from dimension table
SELECT
D.DIMENSION_ID, D.DIMENSION_NAME, D.STAR_SCHEMA_ID,
DT.GLOBAL_TABLE, DT.GLOBAL_FIELD, DT.LEVEL_DESC, DT.PRIMARY_FIELD, DT.LEVEL_NO,
GF.LOCAL_TABLE, GF.LOCAL_FIELD, GF.FIELD_TYPE, GF.MAPPING_RULE,
AT.TABLE_ID, AT.ATTRIBUTE_TYPE, AT.METHOD_NAME, AT.CARDINALITY, TB.PRIMARY_KEY,
TB.OPERATION_TYPE, TB.PARENT_ID, DB.DATABASE_ID, DB.DATABASE_NAME, DB.DRIVER_STRING,
DB.JDBC_URL, DB.JDBC_USERNAME, DB.JDBC_PASSWORD, DB.DATABASE_TYPE

FROM
META_DIMENSION_TABLE D
INNER JOIN META_DIMENSION_TABLE_DTL DT ON D.DIMENSION_ID = DT.DIMENSION_ID
INNER JOIN META_GLOBAL_FIELD GF ON DT.GLOBAL_TABLE = GF.GLOBAL_TABLE
AND DT.GLOBAL_FIELD = GF.GLOBAL_FIELD
INNER JOIN META_ATTRIBUTE AT ON GF.LOCAL_FIELD = AT.ATTRIBUTE_NAME
INNER JOIN META_TABLE TB ON AT.TABLE_ID = TB.TABLE_ID AND GF.LOCAL_TABLE = TB.TABLE_NAME
INNER JOIN META_DATABASE DB ON TB.DATABASE_ID = DB.DATABASE_ID

WHERE D.STAR_SCHEMA_ID = 'SALES'
AND D.DIMENSION_ID = 'INVENTORY'
```

Retrieve fact information from star schema

```
-- Tables and fields from fact table
SELECT
F.MEASURE_NAME, F.FACT_NAME, F.STAR_SCHEMA_ID, F.GLOBAL_TABLE, F.GLOBAL_FIELD,
F.LEVEL_DESC, F.PRIMARY_FIELD,
GF.LOCAL_TABLE, GF.LOCAL_FIELD, GF.FIELD_TYPE, GF.MAPPING_RULE, AT.TABLE_ID,
AT.ATTRIBUTE_TYPE, AT.METHOD_NAME, AT.CARDINALITY,
TB.PRIMARY_KEY, TB.OPERATION_TYPE, TB.PARENT_ID,
DB.DATABASE_ID, DB.DATABASE_NAME, DB.DRIVER_STRING, DB.JDBC_URL, DB.JDBC_USERNAME,
DB.JDBC_PASSWORD, DB.DATABASE_TYPE

FROM
META_FACT_TABLE F
INNER JOIN META_FACT_TABLE_DTL DT ON F.STAR_SCHEMA_ID = DT.STAR_SCHEMA_ID
INNER JOIN META_GLOBAL_FIELD GF ON DT.GLOBAL_TABLE = GF.GLOBAL_TABLE
AND DT.GLOBAL_FIELD = GF.GLOBAL_FIELD
INNER JOIN META_ATTRIBUTE AT ON GF.LOCAL_FIELD = AT.ATTRIBUTE_NAME
INNER JOIN META_TABLE TB ON AT.TABLE_ID = TB.TABLE_ID
AND GF.LOCAL_TABLE = TB.TABLE_NAME
INNER JOIN META_DATABASE DB ON TB.DATABASE_ID = DB.DATABASE_ID

WHERE D.STAR_SCHEMA_ID = 'SALES'
```

Construct queries for target database system

After the target tables and fields' information have been retrieved from metadata, the middleware constructs queries according to database target and database type.

SQL for CRM

```
SELECT ACCOUNT_ID, AMOUNT_OWED, AVAILABLE_CREDIT
FROM CUSTOMER_ACCOUNT
```

XQuery for OMS

```
xquery version "1.0";
<ORDER_FACT>
{
  for $i in doc("salesorder.XML")//sales_order
  let $b := doc("orderdetail.XML")//order_detail[order_id = $i/order_id]
  order by $i/order_id
  return
    <ROW>
      <TIME>{current-dateTime()}</TIME >
      { $i/order_id }
      { $i/customer_id }
      { $b/item_id }
      { $i/total_amount }
    </ROW>
}
</ORDER_FACT>
```

XQuery for WMS

```
xquery version "1.0";
<INVENTORY>
{
  let $d := /inventory /Row
  return
    <ROW>
      {$d/ITEM_ID}
      {$d/ITEM_DESC}
      {$d/QUANTITY}
      {$d/UNIT_PRICE}
    </ROW>
}
</INVENTORY>
```

Data integration of queried results

The queried results are stored in the data warehouse. The data from the relational database are inserted into the data warehouse, while the XML documents from the XML database are transformed into Data Object Module (DOM) objects and then loaded into the data warehouse.

(i) Query on XML data cube:

Data are transformed into XML documents by mapping each relational table into each XML document with each tuple into each row instance.

Translated XML document from RELATIONAL

```
<CUSTOMER>
  <ROW>
    <ACCOUNT_ID>100001</ACCOUNT_ID>
    <AMOUNT_OWED>1000</AMOUNT_OWED>
    <AVAILABLE_CREDIT>4000</AVAILABLE_CREDIT>
  </ROW>
  <ROW>
    <ACCOUNT_ID>100002</ACCOUNT_ID>
    <AMOUNT_OWED>2000</AMOUNT_OWED>
    <AVAILABLE_CREDIT>3000</AVAILABLE_CREDIT>
  </ROW>
  <ROW>
    <ACCOUNT_ID>100003</ACCOUNT_ID>
```

```
<AMOUNT_OWED>5000</AMOUNT_OWED>
<AVAILABLE_CREDIT>0</AVAILABLE_CREDIT>
</ROW>
</CUSTOMER>
```

- ❑ Data from WMS system (retrieve data from Fact table for later OLAP operation)

INVENTORY DIMENSION

```
<INVENTORY>
  <ROW>
    <ITEM_ID>1</ITEM_ID>
    <ITEM_DESC>A01</ITEM_DESC>
    <QUANTITY>30</QUANTITY>
    <UNIT_PRICE>10</UNIT_PRICE>
  </ROW>
  <ROW>
    <ITEM_ID>2</ITEM_ID>
    <ITEM_DESC>B01</ITEM_DESC>
    <QUANTITY>20</QUANTITY>
    <UNIT_PRICE>20</UNIT_PRICE>
  </ROW>
  <ROW>
    <ITEM_ID>3</ITEM_ID>
    <ITEM_DESC>C01</ITEM_DESC>
    <QUANTITY>10</QUANTITY>
    <UNIT_PRICE>20</UNIT_PRICE>
  </ROW>
  <ROW>
    <ITEM_ID>4</ITEM_ID>
    <ITEM_DESC>C02</ITEM_DESC>
    <QUANTITY>15</QUANTITY>
    <UNIT_PRICE>25</UNIT_PRICE>
  </ROW>
  <ROW>
    <ITEM_ID>5</ITEM_ID>
    <ITEM_DESC>C03</ITEM_DESC>
    <QUANTITY>15</QUANTITY>
    <UNIT_PRICE>20</UNIT_PRICE>
  </ROW>
</INVENTORY>
```

- ❑ Data from OMS system

SALES ORDER FACT

```
<ORDER_FACT>
  <ROW>
    <TIME>2006-04-23T17:49:52.359+08:00</TIME>
    <ORDER_ID>1001</ORDER_ID>
    <CUSTOMER_ID>100001</CUSTOMER_ID>
    <ITEM_ID>2</ITEM_ID>
    <TOTAL_AMOUNT>2000.00000000</TOTAL_AMOUNT>
  </ROW>
  <ROW>
    <TIME>2006-04-23T17:49:52.359+08:00</TIME>
    <ORDER_ID>1002</ORDER_ID>
    <CUSTOMER_ID>100001</CUSTOMER_ID>
    <ITEM_ID>2</ITEM_ID>
    <TOTAL_AMOUNT>2000.00000000</TOTAL_AMOUNT>
  </ROW>
  <ROW>
    <TIME>2006-04-24T17:49:52.359+08:00</TIME>
    <ORDER_ID>1003</ORDER_ID>
    <CUSTOMER_ID>100002</CUSTOMER_ID>
```

```

<ITEM_ID>3</ITEM_ID>
<TOTAL_AMOUNT>1000.00000000</TOTAL_AMOUNT>
</ROW>
</ORDER_FACT>

```

(ii) Query on Relational view:

❑ Data from CRM relational table

CUSTOMER DIMENSION

ACCOUNT_ID	AMOUNT_OWNED	AVAILABLE_CREDIT
100001	1000.00	4000.00
100002	2000.00	3000.00
100003	5000.00	0

❑ Transform RELATIONAL from WMS XML document by mapping each XML document into each Relational table with each row instance into each tuple.

INVENTORY DIMENSION

ITEM_ID	ITEM_DESC	QUANTITY	UNIT_PRICE
1	A01	30	10
2	B01	20	20
3	C01	10	20
4	C02	15	25
5	C03	15	20

❑ Transform RELATIONAL from OMS XML document by mapping each XML document into each Relational table with each row instance into each tuple.

SALES_ORDER FACT

TIME	ORDER_ID	CUSTOMER_ID	ITEM_ID	TOTAL_AMOUNT
2006-04-23T17:49:52.359+08:00	1001	100001	2	2000.00000000
2006-04-23T17:49:52.359+08:00	1002	100001	2	2000.00000000
2006-04-24T17:49:52.359+08:00	1003	100002	3	1000.00000000

Step 3 Aggregate data

After the data cube has been built, aggregated data can be extracted from the data cube. The middleware stores aggregated data from heterogeneous sources into a relational or an XML database. The schema design is data centric since it minimizes the programming complexity during data integration, and is beneficial to data materialization when it performs SQL or XQuery for the aggregation of relational or XML data. Figure 10 illustrates how the OLAP cube retrieves an XML document or relational data for data materialization.

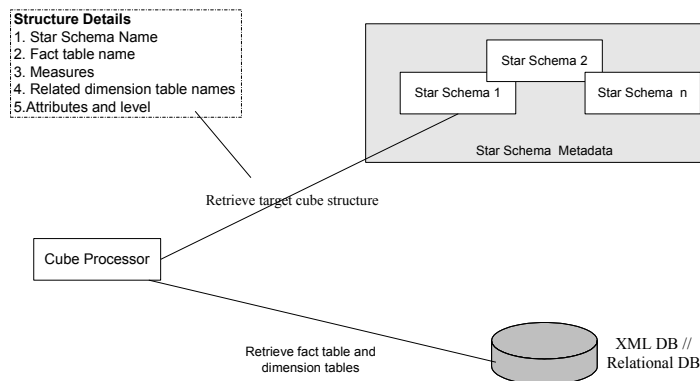


Figure 10 Cube Data Retrieval and Materialization

(i) Data materialization of XML data cube aggregation data using XQuery

Cube formation Algorithm

```

Begin
  Retrieve OLAP schema from star schema of metadata;
  For each dimension do

```

```

    Load XML contents from XML DB to DOM object;
    Load fact table content from XML DB into DOM object;
    Retrieve cube default setting;
    Load slice dimension parameters for filter;
    For each aggregate dimension row do
        Begin Build XQuery with slice and aggregate dimensions parameters;
            Aggregate and filter to summarize measures using XQuery to XML document;
        End
    Integrate all aggregate dimension rows XML into one XML document;
    Transform the serialized XML content into HTML using XSLT;
End

```

□ Use OLAP to retrieve aggregate data [11]

The middleware retrieves the fact table, dimension tables and XML documents based on the metadata configuration. The retrieved XML documents are aggregated and transformed into HTML using an XSLT engine.

□ Transforming XML cube to visual representation

The created cube is transformed to HTML using XSLT transformation. The transformed HTML is displayed in the Internet browser.

□ Interacting on the materialized OLAP cube

The formed OLAP cube is transformed into various materialized views with the changed dimension values. The selection of OLAP report by the user triggers the OLAP cube to retrieve data from the metadata of star schema. The OLAP cube uses the metadata of star schema to build the materialized view of the report. It can be implemented by ETL.

Dimension table retrieval:

```

1: Retrieve star schema dimensions from star schema metadata
2: Retrieve dimensions XML documents from XML data warehouse.
3: Store the dimensions XML documents to the OLAP cube object.

```

Fact table retrieval:

```

1: Retrieve star schema facts from star schema metadata
2: Retrieve fact XML documents from XML data warehouse.
3: Store the fact XML documents to the OLAP cube object.

```

Data materialization with aggregation and slice:

Algorithm

```

Begin
    Receive user input of dimensions;
    Process user input with dimensions XML documents in OLAP cube object;
    For each slice dimension do
        Retrieve XML contents using XQuery;
    For each aggregate dimensions do
        Aggregate XML contents using XQuery;
        Transform materialized XML contents into HTML using XSLT engine;
End

```

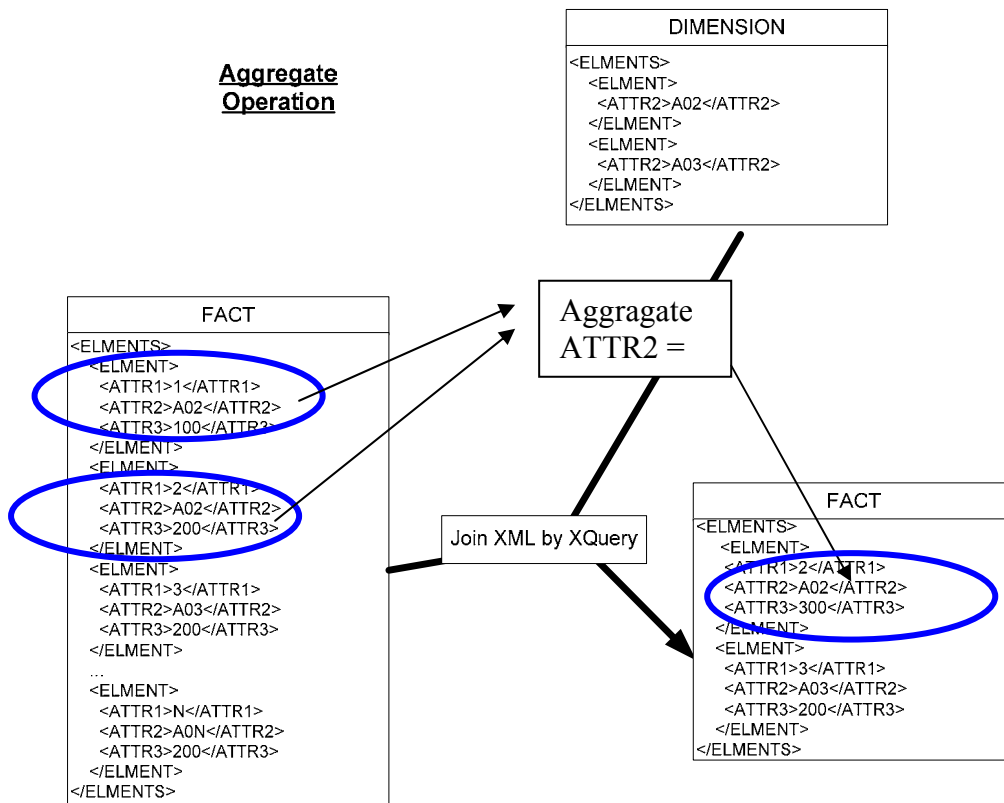


Figure 11 Aggregation done by XQuery

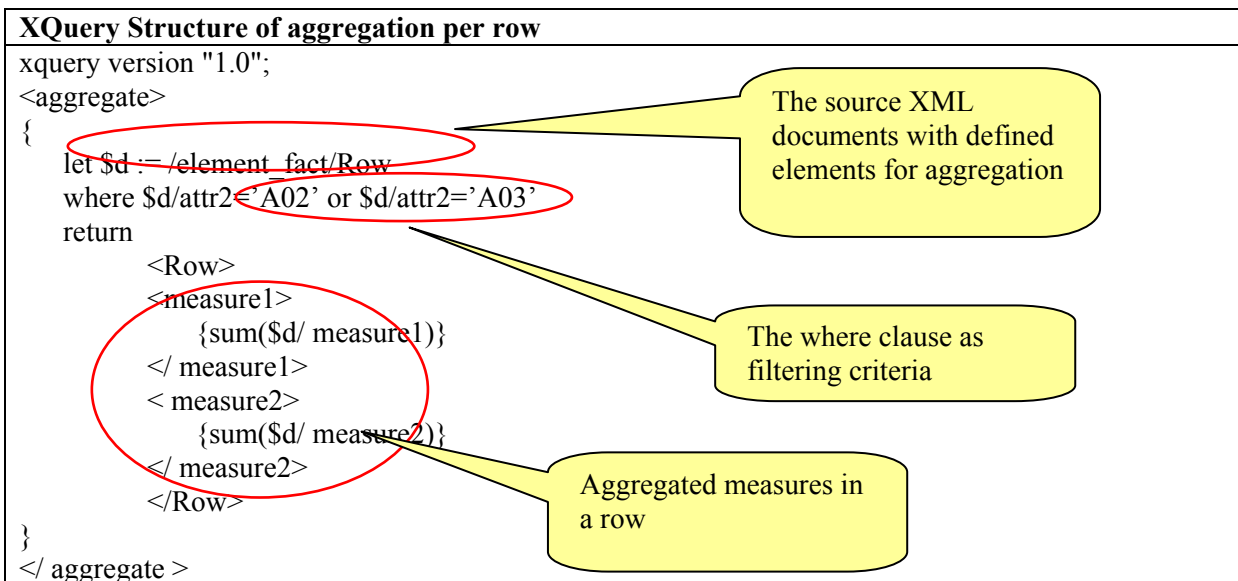


Figure 12 XQuery data aggregation for measures

(ii) Data materialization of Relational data cube aggregation data using SQL

Cube formation Algorithm:

```

Begin
  Retrieve OLAP schema from star schema of metadata;
  For each dimension do
    Load relational data from Relational database;
    Load fact table content from Relational database;
  Retrieve cube default setting
  
```

```

Load slice dimension parameters for filter;
For each aggregate dimension row do
    Begin Build SQL with slice and aggregate dimensions parameters;
        Aggregate and filter to summarize measures using SQL to relational data;
    End
Integrate all aggregate dimension rows relational data into one relational table;
End

```

Dimension table retrieval:

```

1: Retrieve star schema dimensions and levels from star schema metadata
2: Retrieve dimensions relational tables from relational database.
3: Store the dimensions relational data to the OLAP cube object.

```

Fact table retrieval:

```

1: Retrieve star schema facts and levels from star schema metadata
2: Retrieve fact relational tables from relational database
3: Store the fact relational tables to the OLAP cube object.

```

Data materialization with aggregation and slice:

Algorithm

```

Begin
Receive user input of dimensions;
Process user input with dimensions relational tables in OLAP cube object;
For each slice dimensions do
Retrieve relational contents using SQL;
For each aggregate dimensions do
Aggregate relational contents using SQL;
End

```

Aggregate Operation

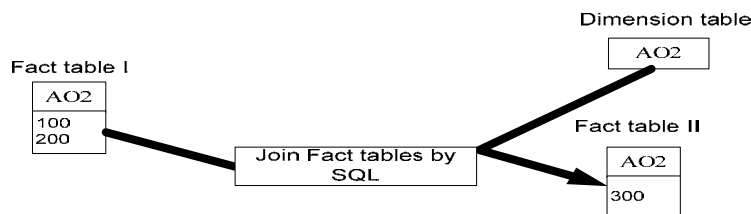


Figure 13 Aggregation done by SQL

```

Select Sum(A02) from Fact_Table_1 where Dimension_table_1_A02=Fact_table_1_A02
Insert Fact_table_II (A02) value (Sum(A02))

```

Data Materialization of Aggregated XML Data

Finally, the integrated XML data in the data warehouse are retrieved and materialized by the OLAP middleware engine. When a user browses an OLAP report page in the browser, the middleware retrieves the required fact table and dimension tables based on the information in the star schema metadata. The middleware performs slice and aggregation of the XML contents with expand mode and transforms the queried contents to HTML.

Rollup and drill down on XML data cube

The middleware captures the user input and aggregates the XML contents using XQuery. Figure 14 shows the result of rollup where item B01 is in Order 1001 and 1002, and which aggregate to a total amount of 4000 by referring back the Sales_Order_Fact Table, and Order 1001, 1002 and 1003 are summed up to a total amount of 5000.

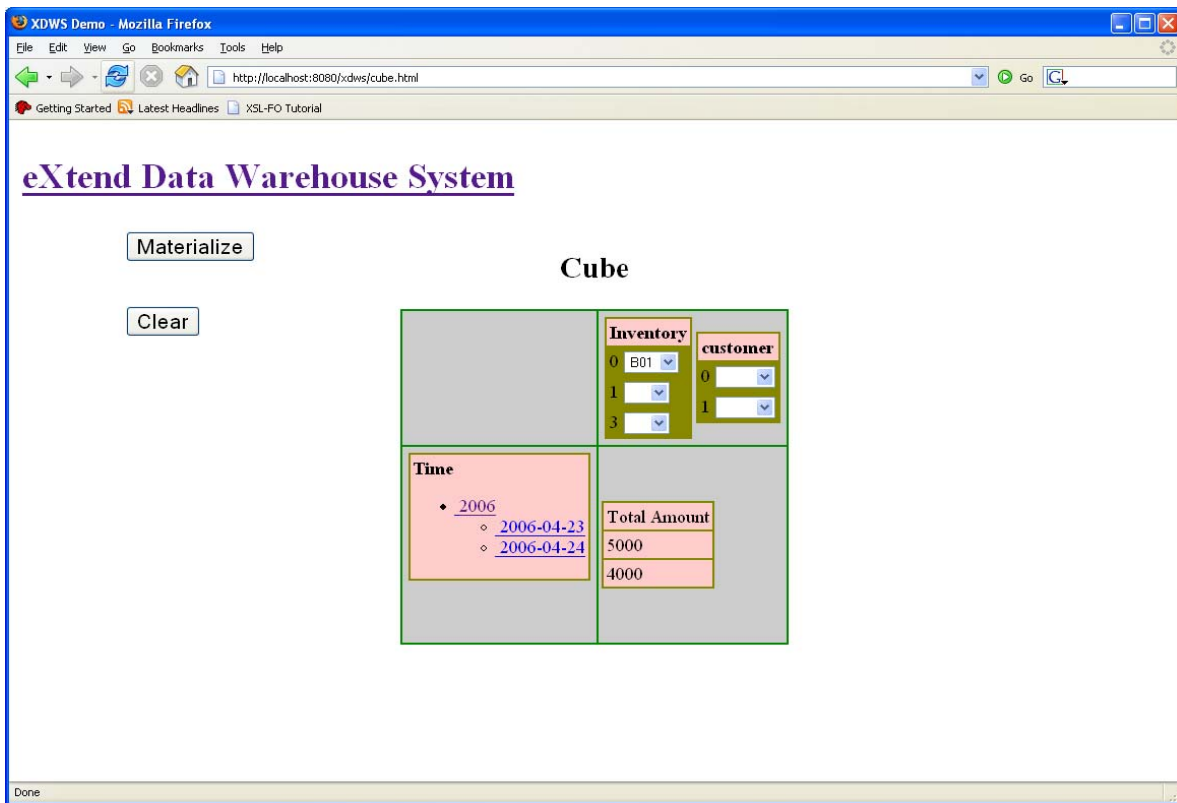


Figure 14 Screen capture of an OLAP report in the case study

Performance analysis

The objective performance analysis of RXDW (Relational XML Data Warehouse) system was to test the actual performance of the system using sets of testing data. RXDW is a system capable of distributed computing. In order to mitigate the deviation in performance affected by a network environment, the whole testing environment was setup and configured in a standalone server. Since RXDW required excess memory usage, the program logic was changed by partitioning the XML during merging and the Java heap size should be increased on the application server. The performance result was acceptable except during the partial aggregation when it merged the data with tuples. The operation was the actual merge of two or more XML documents. It is CPU bounded. Though the actual merge time of 100,000 would take 8 minutes time in total, the web application will display a few hundred rows on one page and the merge process is partitioned to many pages with a processing time of 2-3 seconds each on average.

The hardware and software configuration are shown below.

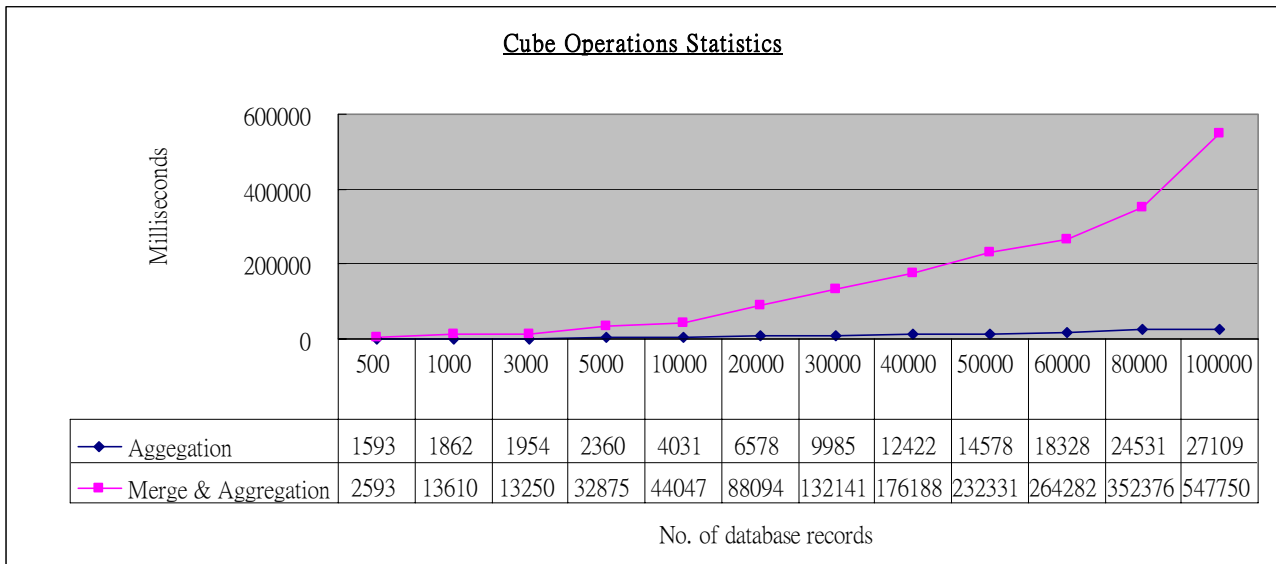
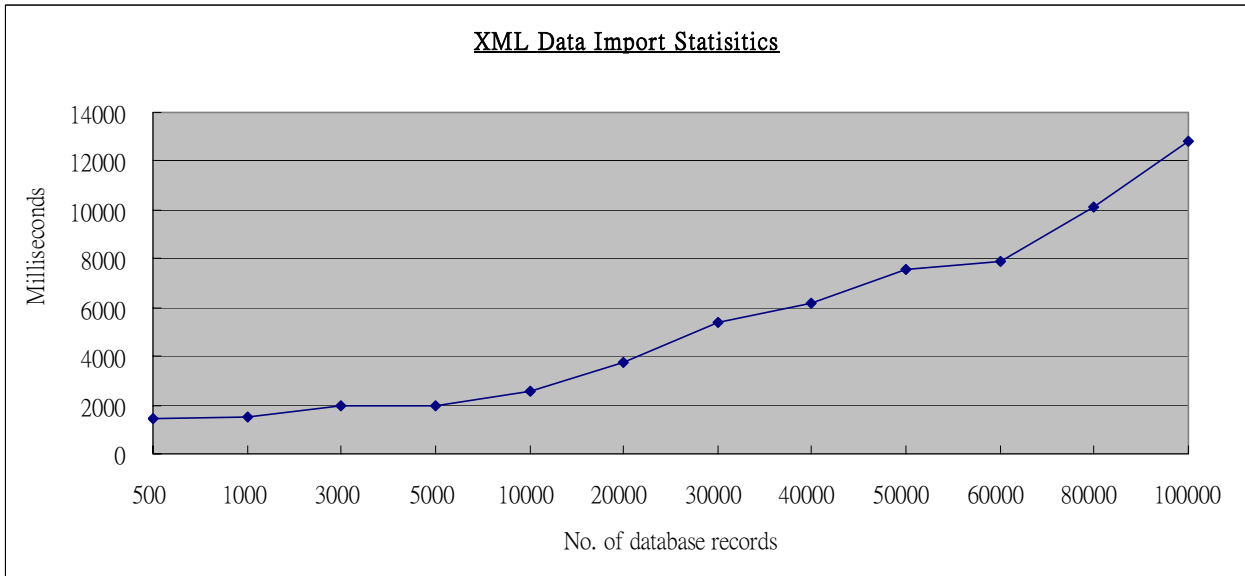
- Pentium M Intel 1.86 GHz CPU
- 1.5GB RAM
- 100 GB 7200rpm Hard disk
- Windows XP Professional service pack 2
- MySQL 5.0
- Java Development Kit 1.5.0
- Xindice 1.1 – XML Database
- Tomcat 5.5 (Xms 256 Xmx 768)

Statistical Result:

Table 2 shows the test results of RXDW

Table 2

Data Volume (No. of Records)		Import to XML database (Milliseconds)		Cube Retrieve and Merge XML (Milliseconds)	
Master Data	Transaction Data	Aggregate Dimension	Aggregate Fact	w/ aggregate only	w/ merge operation & aggregation
3000	500	1282	1437	1593	2593
3000	1000	1282	1515	1862	7610
3000	3000	1282	1984	1954	13250
3000	5000	1282	1991	2360	32875
3000	10000	1282	2531	4031	44047
3000	20000	1282	3766	6578	88094
3000	30000	1282	5406	9985	132141
3000	40000	1282	6172	12422	176188
3000	50000	1282	7531	14578	352376
3000	60000	1282	7906	18328	264282
3000	80000	1282	10125	24531	352376
3000	100000	1282	12828	27109	547750



4 Conclusion

XML plays an important role in today's EAI[13]. The critical performance issue is how the middleware aggregates data from multiple data sources. In fact, the main bottleneck of the middleware is in the data translation and data loading processes. This issue can be solved by distributing the data integration process on the remote server with SOA[19], which is the preprocessed result before it sends it back to the middleware host. After using XQuery and XSLT during the integration process, the data integration process becomes easier and increases performance amazingly.

The main contribution of this paper shows that it is feasible to provide a relational data cube as well as an XML data cube of a Relational-XML data warehouse wherein the data are extracted from relational tables and XML documents. The benefit is that the user is able to select a familiar OLAP language — either SQL or XQuery — to retrieve aggregation data from a Relational-XML data warehouse.

Acknowledgement: This paper is funded by Strategic Research Grant 7002140 of City University of Hong Kong

References:

- [1] Zhu, Hongwei, Michael D. Siegel, and Stuart E. Madnick; “Information Aggregation - A Value-added E-Service”, <http://ebusiness.mit.edu/research/papers/106%20Madnick,%20Siegel%20Information%20Aggregation.pdf>, Working Paper #106, 2001
- [2] Stuart E. Madnick, and Michael D. Siegel; “Seizing the Opportunity: Exploiting Web Aggregation”, <http://ebusiness.mit.edu/research/papers/144%20Madnick,Aggregator.pdf>, Working Paper #144, 2001
- [3] High-Performance Information Aggregation Using XML-Based Operational Data Servers, <http://www.zapthink.com/report.html?id=WP-0114>
- [4] Building XQuery Based Web Service Aggregation and Reporting Applications, http://www.stylusstudio.com/xquery_tutorial.html
- [5] Mid Tier XML Market Opportunities, Strategies, and Forecasts, 2004 to 2009, http://www.giichinese.com.tw/chinese/wg17637_XML_market_toc.html
- [6] IEEE Society Distinguish the type of middleware
http://dsonline.computer.org/portal/site/dsonline/menuitem.20d6846e1c7ed783fla516106bbe36ec/index.jsp?&pName=dso_level1_home&path=dsonline/topics/middleware&file=index.XML&xsl=generic.xsl&
- [7] Aggregate Data with Xquery from Oracle Technical Network
<http://www.oracle.com/technology/oramag/oracle/05-mar/o25XML.html>
- [8] D. Lee and W. W. Chu, “CPI: Constraints-Preserving Inlining Algorithm for Mapping XML DTD to Relational Schema,” *Data Knowledge Engineering*, Vol. 39, No. 1, 2001, 3-25.
- [9] “Information Systems Reengineering and Integration” by Joseph Fong, published by Springer Verlag, 2006, pp.21-22, pp.160-198
- [10] Building a data warehouse for decision support;” second edition, by Vidette Poe, Patricia Klanuer and Stephen Brost, Prentice Hall, 1998, chapter 9 Designing the database for a data warehouse
- [11] “Data Mining: Concepts and Techniques” by Han and Kamber, Morgan Kaufmann publishers, 2001, chapter 2, pp. 39-103
- [12] “Enterprise Application Integration with XML and Java” by JP Morgenthal, Bill La Forge, published by Prentice Hall PTR, 2001, pp. 131-146, pp.291-373, pp.719-853.
- [13] “XML Primer Plus” by Nicholas Chase, published by SAMS, 2003, pp.47-113, pp.291-373.
- [14] “Universal Data Warehousing Based on a Meta-Data Modeling Approach” by J. Fong, Q. Li and S.M. Huang, *International Journal of Cooperative Information System – World Scientific*, Volume 12, Number 3, September 2003.
- [15] Jiawei Han and Micheline Kamber, “Data Mining: Concepts and Techniques”, Morgan Kaufmann Publishers’, 2001, pp. 58-70.
- [16] “Schema Integration Methodology and its Verification by use of Information Capacity” by Irene Kwan and Joseph Fong,, *Information Systems*, Volume 24, Number 5, 1999, pp.355-376.
- [17] “XML How to program” by Deitel, Deitel, Nieto, Lin and Sadhu, Prentice Hall, 2001, pp.297-318.
- [18] “XQuery from the Experts: A Guide to the W3C XML Query Languages”, by Chamberlin, Darper, Fernandez, Kay, Robie, Rys, Simeon, Tivy and Wadler, edited by Howard Katz, published by Addison-Wesley, 2004, pp.187-234, 355.
- [19] <http://www.w3org/2003/Talks/1211-XML2003-wssoa>
- [20] Akmal B. Chaudhri, Awais Rashid, and Roberto Zicari. *XML Data Management: Native XML and XML-Enabled Database Systems*. 1st ed. Addison Wesley Professional, 2003.
- [21] Lucie Xyleme. *Xyleme, a Dynamic Warehouse for XML Data of the Web*. Database Engineering & Applications, 2001 International Symposium on, 2001 pp. 3-8.
- [22] Pedersen, D., Riis, K. and Pedersen, T. B.(2002), “XML-Extended OLAP Querying”, Proceedings of the 14th International Conference on Scientific and Statistical Database Management (SSDBM’02).
- [23] Park, B., Han, H. and Song, I.(2005), “XML-OLAP: A Multidimensional Analysis Framework for XML Warehouses”, Proceedings of DAWAK’05, LNCS 3589, pp.32-42.
- [24] Madnick, S., and Zhu, H.,(2006) “Improving data quality through effective use of data semantics”, *Data and Knowledge Engineering*, November, 59, pp.460-475.
- [25] Kimball, R. and Caserta, J., (2004), “The Data Warehouse ETL Toolkit”, Wiley Publishing, Inc., ISBN: 0-764-56757-8.