A TOOL FOR HEALTHCARE INFORMATION INTEGRATION

Sellappan Palaniappan and Ng Yih Huey

Department of Information Technology
Malaysia University of Science and Technology
Kelana Square, 47301 Petaling Jaya, Malaysia

{sell, yhng}@must.edu.my

ABSTRACT

Currently, information collected by the various healthcare providers (hospitals and clinics) in Malaysia is not integrated. Each collects information for its own internal use. The information is not aggregated and analyzed at the state or national level to provide useful information for decision-making (e.g. by the Ministry of Health). Extracting and aggregating information from diverse data sources (databases) has always been a challenge for the healthcare industry. This paper presents a tool for integrating healthcare information from several relational databases into a single data warehouse. It is simple, cost-effective and yet solves the information integration problem. It consists of a schema mapping process which is partially automated and an Extraction, Transformation and Loading (ETL) process which is fully automated. To demonstrate its viability, a data warehouse containing materialized data is built from three databases, namely, Access, SQL Server, and Oracle. It is implemented using the Microsoft .NET Framework.

Keywords: Data Warehouse, Healthcare Information Integration, Extraction Transformation and Loading (ETL), Metamodel, Schema Mapping.

1.0 MOTIVATION

Currently, healthcare providers (i.e., hospitals and clinics) in Malaysia use conventional transaction processing systems to record their daily clinical and financial data. However, as people's lifestyles become more complex and technologies become more advanced, users demand better quality or valued-added services. Healthcare providers are expected to increase the value of their transaction processing systems - they need to turn data into actionable information (Phipps, 2002).

Healthcare providers generate voluminous data, but they are not leveraged for generating valuable information. Extracting, aggregating, and analyzing healthcare information from distributed and heterogeneous data sources (relational databases) has always been a challenge for the healthcare industry. Aggregating and integrating healthcare information is important for understanding the health of a community because the quality of services provided depends on the availability of accurate, relevant, and timely information. This paper presents a tool for efficiently integrating healthcare information from several data sources into a single data warehouse. Such a data warehouse can be used to build healthcare decision support systems that can enhance the quality of healthcare services provided. For example, it can support OnLine Analytical Processing (OLAP) and data mining to generate more focused healthcare information (Poole & Mellor, 2001).

Most healthcare providers do not use data warehouse technologies because they are very costly. This paper presents an integration tool that is effective and yet inexpensive compared to most other tools available in the market. The tool solves the data integration problem by (1) semi-automating the schema mapping process and (2) fully-automating the Extraction, Transformation and Loading (ETL) process.

2.0 TOOL ARCHITECTURE DESIGN

The tool is based on the client-server architecture (Fig. 1). It comprises two main components: Agent and Web Server. The Agent is installed on each client computer. Data extracted from the clients are uploaded to the Web Server for integration via the Internet.

The Web Server consists of a Data Warehouse Manager (**DWM**) and an ETL engine (Fig. 2). The DWM manages the data warehouse and its metadata. The ETL engine manages the data integration process. A Web application developed using ASP.NET facilitates the administrative tasks. It allows the various

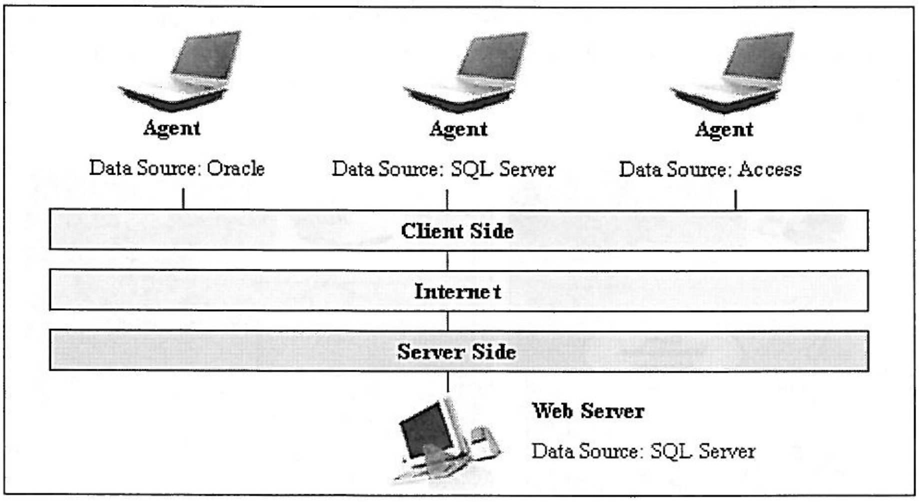


Fig. 1: High level view of architecture design

stakeholders, e.g. healthcare providers, administrators, Malaysia Medical Association, Ministry of Health, and researchers, to access the integrated healthcare information. The healthcare providers can upload their XML data files extracted automatically by the Agent via the Web application.

The Agent performs the schema mapping and data extraction (Fig. 2). The schema mapping is semi-automated; the engine first extracts the client's database metadata and the data warehouse metadata. The user then performs the mapping semi-automatically using the client database structure and the data warehouse structure. The mapping generates a schema mapping metadata for data integration. The data extraction process, on the other hand, is fully automated. The ETL engine extracts the data from the client database using the schema mapping metadata and then converts them to XML data files. The files are uploaded to the Web server for data integration.

The Web services are implemented as a set of Remote Procedure Calls (RPCs). The Universal Description, Discovery and Integration (UDDI) is used to publish the services while the Web Services Description Language (WSDL) describes the rules for interfacing and interacting. The clients i.e. Agent and Web server request services and the server responds to these requests. The information is exchanged as XML documents using Simple Object Access Protocol (SOAP) over HTTP. The Web services implemented include database web services, data warehouse management web services, metadata management web services, and ETL web services. These Web services are used to integrate information into the data warehouse.

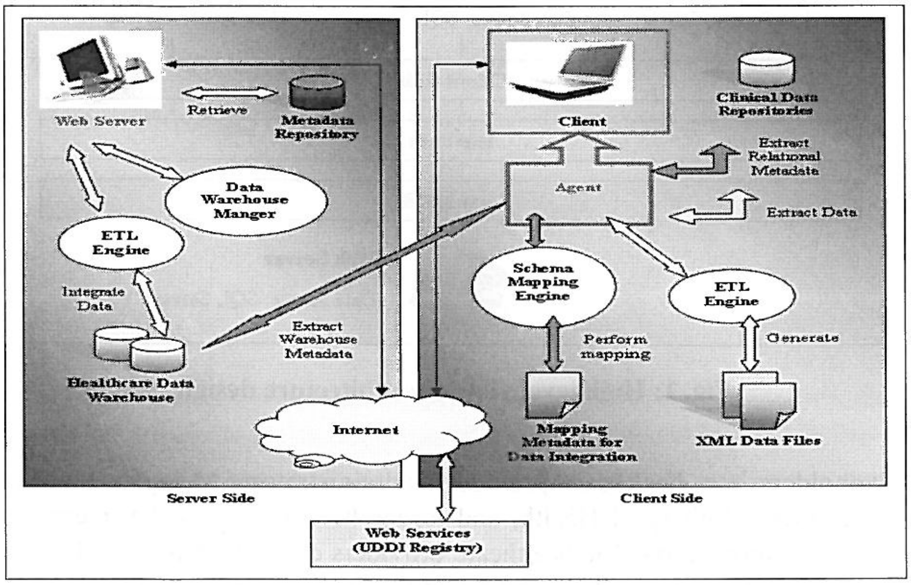


Fig. 2: Low level view of architecture design

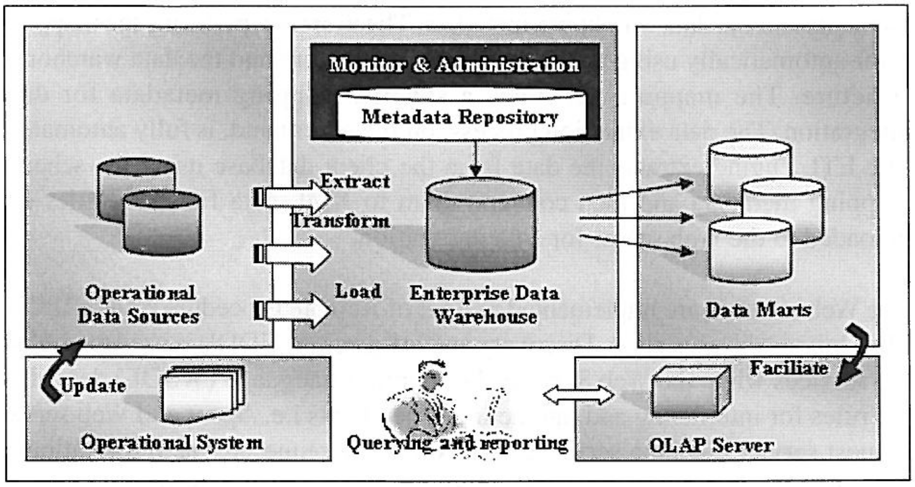


Fig. 3: Data warehouse architecture (Zhengxin, 2002)

3.0 DATA WAREHOUSE ARCHITECTURE DESIGN

The data warehouse architecture consists of several components: database sources, enterprise data warehouse storage, data marts, back end systems and metadata repository (Fig. 3).

The design is based on the three tier architecture: the bottom tier is a data warehouse server, the middle tier is an OLAP server, and the top tier is a client with query and reporting facilities. This research focuses only on the bottom tier. The middle and the top tiers are areas for future work.

4.0 DATA WAREHOUSE METAMODEL DESIGN

The metamodel is an important element in the data warehouse environment as it represents the structures and semantics of the different data models within the tool (Tan & Zaslavsky, 2003). It is used to integrate the heterogeneous data sources. Fig. 4 shows the design of the metamodel.

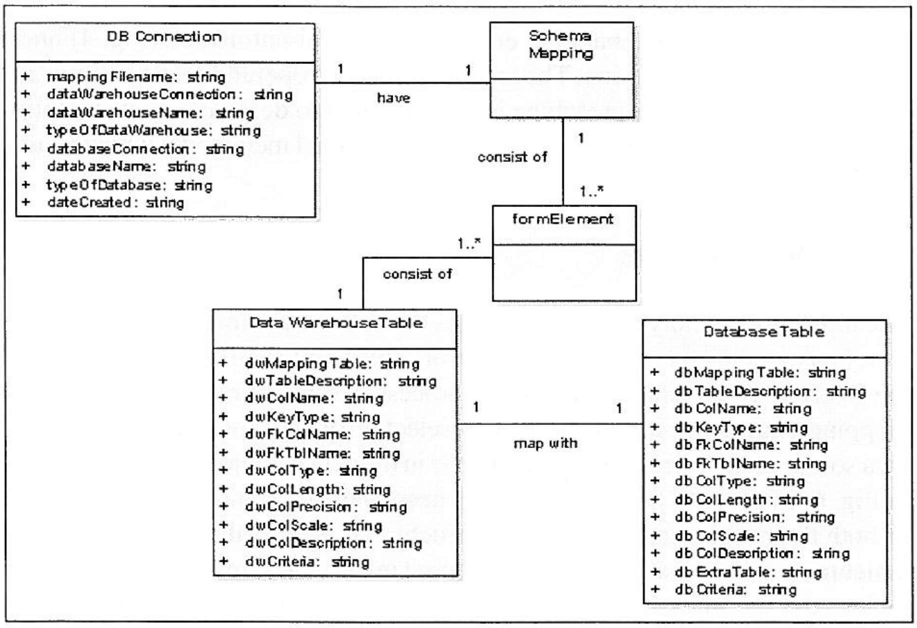


Fig. 4: Data warehouse metamodel design

5.0 DATA WAREHOUSE SCHEMA DESIGN

The conceptual data warehouse schema can be modelled using the dimensional model or the tabular model. Both models have their own advantages. This research uses the tabular model because it captures both detailed and summarized data. Thus, it can provide greater value to stakeholders in the future as these can be optimized for numeric and textual analysis. Fig. 5 shows the conceptual data warehouse schema design. The schema serves to demonstrate the feasibility of the information integration tool. The schema can be extended to include other information. The tool is quite generic, so it can be adapted to other data warehouse schema design.

Although this research uses the tabular model to structure the data warehouse schema, we can also easily derive the dimensional model should there be a need, for example, to support OLAP functions which require retrieving tables and their attributes. Not that the tabular model can still support analysis functions using Relational OLAP (ROLAP) tools.

6.0 IMPLEMENTATION

The tool uses a schema mapping engine that is semi-automated (Fig. 4) and a fully automated ETL engine. The tool provides interoperability, flexibility, and convenience to users. A prototype is implemented to demonstrate its viability. The schema mapping metadata adheres to the defined metamodel which is used to facilitate the ETL process.

6.1 Schema Mapping

The tool automatically extracts and loads the data source metadata and the data warehouse metadata. It displays a list of tables and their attributes in a tree view format to facilitate the mapping process. Users interact with the schema mapping engine through the Agent to select attributes for mapping from the data source to the corresponding attributes in the data warehouse tables as shown in Fig. 6. In order to perform mapping, users must have a good understanding of both their organizations' database structure as well as the data warehouse's structure (such as what information is stored in the tables, relationships between tables, and the integrity constraints). In addition, users must also have basic knowledge of constructing SQL queries. This is because only they know their organization's business flow and only they can do the schema mapping. The

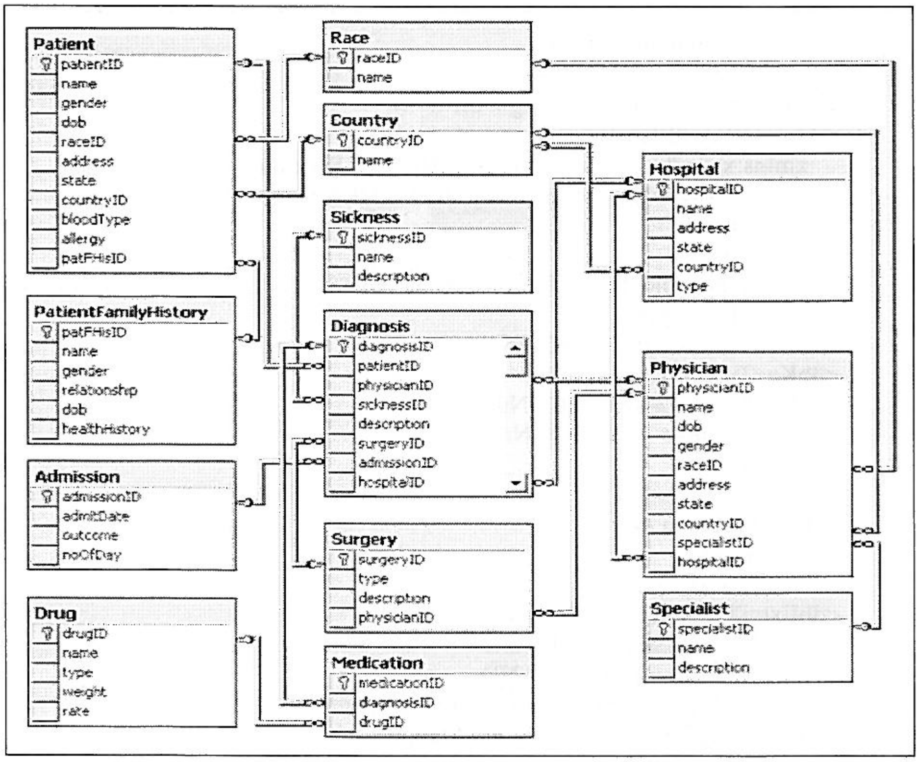
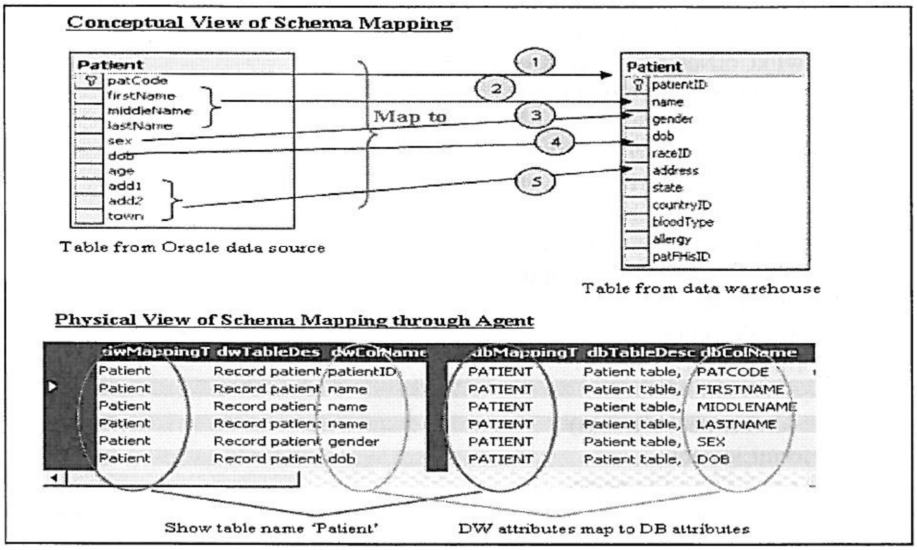


Fig. 5: Data warehouse schema design


Conceptual View of Schema Mapping
Patient
patCode
firstName
middleName
lastName
sex
dob
age
add1
add2
town
Map to
1
2
3
4
5
Patient
patientID
name
gender
dob
raceID
address
state
countryID
bloodType
allergy
patFHisID
Table from Oracle data source
Table from data warehouse
Physical View of Schema Mapping through Agent
Show table name 'Patient'
DW attributes map to DB attributes

dwMappingT	dwTableDes	dwColName	dbMappingT	dbTableDesc	dbColName
Patient	Record patient	patientID	PATIENT	Patient table,	PATCODE
Patient	Record patient	name	PATIENT	Patient table,	FIRSTNAME
Patient	Record patient	name	PATIENT	Patient table,	MIDDLENAME
Patient	Record patient	name	PATIENT	Patient table,	LASTNAME
Patient	Record patient	gender	PATIENT	Patient table,	SEX
Patient	Record patient	dob	PATIENT	Patient table,	DOB

Fig. 6: Schema mapping

db1-mappSschema.mapp

```
<?xml version=”1.0" encoding=”utf-8"?>
<ArrayOfFormElement
   xmlns:xsd=”http://www.w3.org/2001/XMLSchema”
    xmlns:xsi=”http://www.w3.org/2001/XMLSchema-
    instance”>
<formElement>
<dbMappingTable>tblStaff</dbMappingTable>
<dbTableDescription />
<dbColName>staffID</dbColName>
<dbKeyType>PK</dbKeyType>
<dbFkColName>-</dbFkColName>
<dbFkTblName>-</dbFkTblName>
 <dbColType />
<dbColLength />
<dbColPrecision />
<dbColScale />
<dbColDescription />
<dbExtraTable>tblStaffType</dbExtraTable>
 <dbCriteria>tblStaff.staffTypeID =
   tblStaffType.staffTypeID AND
    tblStaffType.description = Doctor’</dbCriteria>
<dwMappingTable>Physician</dwMappingTable>
<dwTableDescription>Record physician personal
   data</dwTableDescription>
<dwColName>physicianID</dwColName>
<dwKeyType>PK</dwKeyType>
 <dwFkColName>-</dwFkColName>
<dwFkTblName>-</dwFkTblName>
<dwColType>bigint</dwColType>
<dwColLength>8</dwColLength>
<dwColPrecision>19</dwColPrecision>
<dwColScale>0</dwColScale>
<dwColDescription>Unique
   identifier</dwColDescription>
 <dwCriteria />
</formElement>
<formElement>
   :
   : (define next element)
 </formElement>
</ArrayOfFormElement>
```

Fig. 7: Sample of schema mapping metadata specifying mapping information

organizations may have different business flows even if the nature of their business is the same. That means, only the users can ensure the correctness of the mapping. This step is crucial because only correct mapping will lead to correct data integration. The schema mapping metadata (Fig. 7) is generated automatically according to the specified metamodel (Fig. 4).

The sample schema mapping metadata shows that the field 'staffID' from table 'tblStaff' of the data source is mapped to the field 'physicianID' from table 'Physician' of the data warehouse. The data is structured in XML format. This mapping is only needed for the first time when data from a data source is to be integrated into the data in the data warehouse. From then on, the metadata is loaded from the system whenever users update the structure of their data sources or the structure of the data warehouse.

6.2 Extraction, Transformation, and Loading (ETL) Process

The schema mapping metadata created from the schema mapping process is used for expressing the ETL process. Algorithms are developed to automate the ETL process. Data extraction involves extracting data from several data sources such as SQL Server, Oracle, and Access. This invariably requires resolving semantic conflicts and handling multiple data types and data structures. Data are extracted based on specified criteria in the schema mapping metadata and structured to the data warehouse structure. This step simplifies data integration and loading. Data transformation involves handling inconsistent data, checking for null values, data concatenation and type conversion, and eliminating duplicate records. All these steps improve the quality of the data before they are loaded into the data warehouse. Data loading also includes integrity constraints and surrogate key management and integration.

The ETL engine in the Agent is responsible for automating the data extraction and data transformation processes whereas the ETL engine in the Web server is responsible for automating the data transformation and data loading processes. The materialized data integration is achieved by loading the actual data into the data warehouse.

6.2.1 Integrity Constraints Management

In order to resolve integrity constraints imposed by the data warehouse during the uploading of data into the corresponding tables, all the independent tables are processed first, followed by the dependent tables. An independent table is one that does not have any foreign keys. A dependent table is one which has one or more foreign keys. The order of uploading tables is important in order to take care of integrity constraints. The rule is: upload tables with the least number of foreign keys. The tables referenced by the foreign keys must be uploaded first before uploading the tables containing these foreign keys.

6.2.2 Surrogate Key Management

Primary key conflicts arise because different data sources may use different formats such as numeric, text, or both. A uniform surrogate key is assigned to handle this problem. This avoids having inconsistent or poor quality data. Index tables are created to assign the surrogate key so that it stores both the primary key that is generated automatically by the system and the actual key used to reference the data. Each data source has a set of index tables to reference the individual data warehouse tables. All the index tables are maintained throughout the life of the data warehouse. For example, the data source dbHospital has a set of index tables (e.g. dbHospital_Patient_Index, dbHospital_Physician_Index, dbHospital_Diagnosis_Index) and the data source dbHealth has its own set of index tables (e.g. dbHealth_Patient_Index, dbHealthl_Physician_Index, dbHealth_Diagnosis_Index). The reason for having a separate set of index tables for the different data sources is to avoid having the same key to refer to different data records which can cause problems, e.g. data inconsistency, when the data are uploaded.

The system updates the index table every time it uploads data into the independent table (also called primary table). It searches for the actual key in the index table in order to retrieve the corresponding surrogate key which is then used (instead of the actual key) in the foreign key column when uploading data into the dependent table. Fig. 8 shows how this is done.

7.0 EVALUATION

7.1 Tool Generalization

As the tool presented here is quite generic, it can be easily adapted to other application domains such as education, finance, and e-government. Obviously, some modifications, e.g. user interfaces, are needed to port it to other domains.

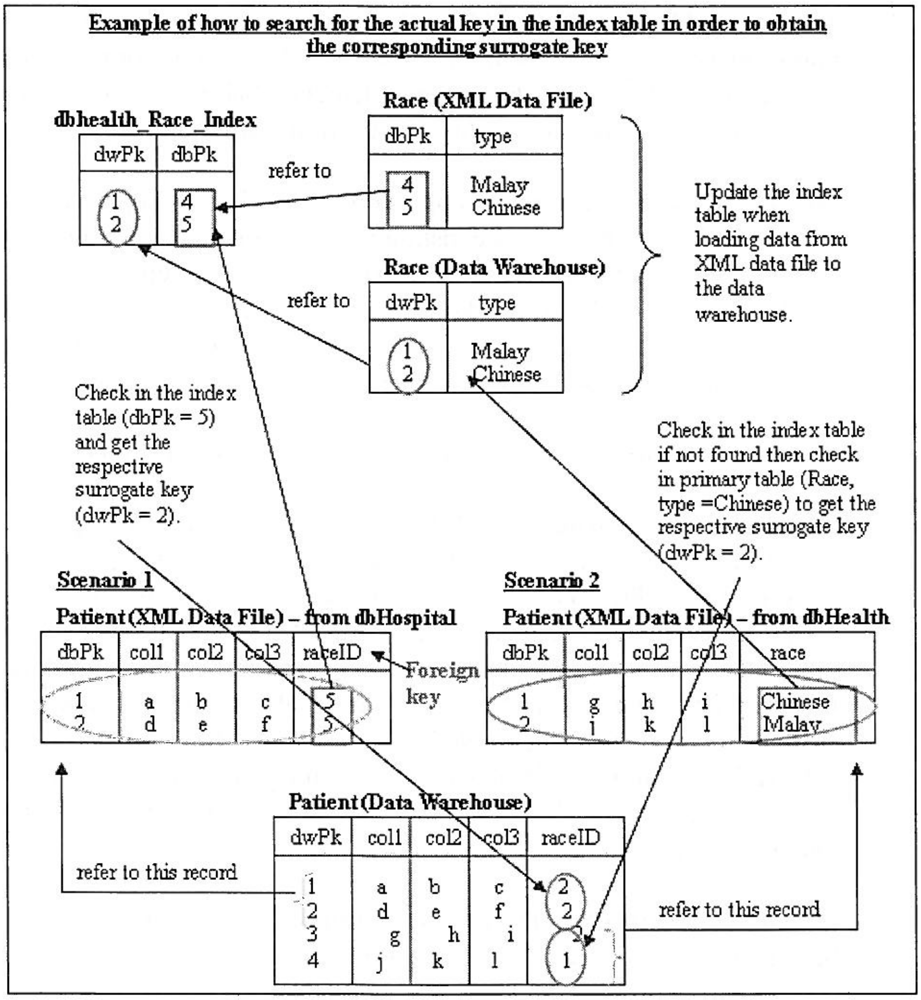

Example of how to search for the actual key in the index table in order to obtain the corresponding surrogate key
dbhealth_Race_Index
dwPk dbPk
1 2 | 4 5
refer to
Race (XML Data File)
dbPk type
4 Malay
5 Chinese
Race (Data Warehouse)
dwPk type
1 Malay
2 Chinese
refer to
Update the index table when loading data from XML data file to the data warehouse.
Check in the index table (dbPk = 5) and get the respective surrogate key (dwPk = 2).
Check in the index table if not found then check in primary table (Race, type =Chinese) to get the respective surrogate key (dwPk = 2).
Scenario 1
Patient (XML Data File) – from dbHospital
dbPk col1 col2 col3 raceID
1 a b c 5
2 d e f 5
Foreign key
Scenario 2
Patient (XML Data File) – from dbHealth
dbPk col1 col2 col3 race
1 g h i Chinese
2 j k l Malay
Patient (Data Warehouse)
dwPk col1 col2 col3 raceID
1 a b c 2
2 d e f 2
3 g h i 2
4 j k l 1
refer to this record
refer to this record

Fig. 8: Solution for handling primary key conflicts

7.2 Feasibility of the Tool

The tool supports mapping of heterogeneous data sources which include:

- ***Mapping Tables with One-to-One Relationship*** – Tables with one-to-one relationship are the most common type. The attributes of a table in the data source are directly mapped to the attributes in the corresponding table in the data warehouse.

- ***Mapping Tables with One-to-Many Relationship*** – The schema mapping allows mapping of tables with one-to-many relationship. For example, for the tables Diagnosis, Patient and Medicine, one patient may have several diagnoses and one diagnosis may require several drugs.

- ***Mapping Tables with Integrity Constraints*** – The schema mapping allows mapping tables with integrity constraints. For example, the foreign key attribute of a table in the data source can be mapped to the corresponding attribute of the table in the data warehouse. Specifying additional conditions may or may not be required depending on how the tables in the data source are structured in relation to the tables in the data warehouse.

The tool provides flexibility to users to perform the mapping which include:

- ***Flexible Mapping Sequence*** – When users perform the mapping, they do not have to map according to the sequence of the column specified by the ORDINAL_POSITION of the data warehouse. That means, users are not constrained to map all the attributes of a table before they map other attributes from other tables. They are free to map any attribute from any table and in any sequence. Additionally, users can delete any attribute mapped or re-map a deleted attribute at any time. The mapping is flexible as it allows users to easily add extra attributes to the schema mapping metadata without needing to create an entire new schema mapping metadata.

- ***Concatenation of Data*** – Data extracted from one or more attributes can be concatenated to a single attribute. For example, data extracted from the attributes firstName, middleName and lastName, can be concatenated to a single attribute, say, Name.

- ***Easy Handling of Structures and Semantics Conflicts*** – Integrating data from different data sources with different data structures and semantics can lead to structure and semantic (e.g., homonyms and synonyms) conflicts. Homonym conflicts occur when different data sources use the same name for different attributes while synonyms conflicts occur when different names are used for the same attribute. However, users need not worry about the structure and semantic conflicts occurring among data sources and data warehouse as these can be handled easily by specifying conditions while mapping the attributes. The tool handles these conflicts during the integration process using the schema mapping metadata which captures the schema mapping details defined by the user.

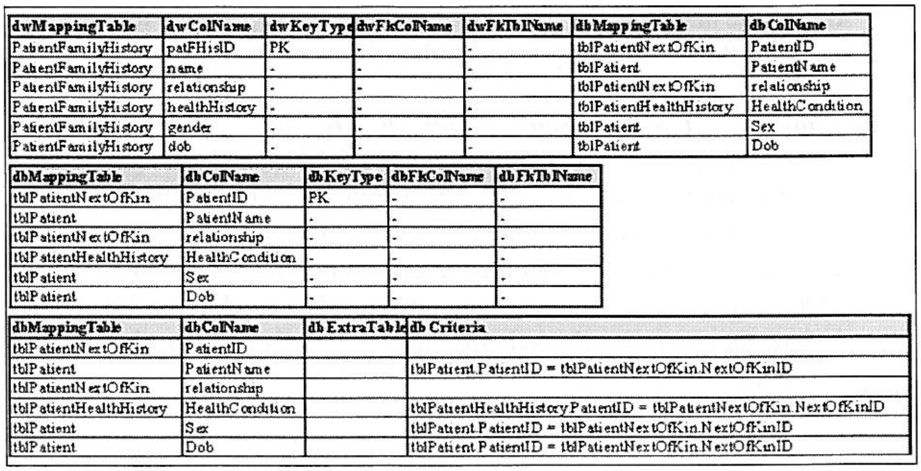
- *Allows Easy Update of Schema Mapping Metadata* – The user can easily update the schema mapping metadata to accommodate changes made to structures in the data sources or data warehouse. Also, the users need not convert their existing data models to other formats in order to use this schema mapping engine unlike the one proposed by Tan and Zaslavsky (2002), where users have to convert their data model to XML format before they can use the schema mapping engine. Thus, this schema mapping is more flexible.

- *Allows Users to Specify Additional Conditions for Easy Mapping* – The schema mapping allows users to specify conditions for mapping attributes of data sources to the attributes of the data warehouse. As organizations may use different data structures, specifying conditions is important so that different data source structures can map to the data warehouse structure. The condition specified is a simple WHERE clause used in an SQL query.

To illustrate, here is a simple example that extracts the patient's family history data from the data source and integrates it to the PatientFamilyHistory table of a data warehouse. To do this, we need to specify additional conditions because of structure conflicts between the tables in the data source and the data warehouse (Fig. 10). There are also integrity constraints imposed on the tables in the data source and data warehouse. The result of the mapping is shown in Fig. 9. In the data source, three tables are affected for extracting the patient's family history data, which are tblPatient, tblNextOfKin, and tblPatientHealthHistory. This is because the tables are constructed such that information of the patient and the patient's next of kin are both stored in the same table, tblPatient. The table tblPatientNextOfKin thus acts as the reference table where the attribute NextOfKinID is a foreign key that refers to the primary table tblPatient for the patient's next of kin's information.

8.0 FURTHER RESEARCH

Currently, the tool supports only three types of databases: Oracle, SQL Server, and MS Access. It can be extended easily to support other databases such as



dwMappingTable	dwColName	dwKeyType	dwFkColName	dwFkTblName	dbMappingTable	dbColName
PatientFamilyHistory	patFHisID	PK	-	-	tblPatientNextOfKin	PatientID
PatientFamilyHistory	name	-	-	-	tblPatient	PatientName
PatientFamilyHistory	relationship	-	-	-	tblPatientNextOfKin	relationship
PatientFamilyHistory	healthHistory	-	-	-	tblPatientHealthHistory	HealthCondition
PatientFamilyHistory	gender	-	-	-	tblPatient	Sex
PatientFamilyHistory	dob	-	-	-	tblPatient	Dob

dbMappingTable	dbColName	dbKeyType	dbFkColName	dbFkTblName
tblPatientNextOfKin	PatientID	PK	-	-
tblPatient	PatientName	-	-	-
tblPatientNextOfKin	relationship	-	-	-
tblPatientHealthHistory	HealthCondition	-	-	-
tblPatient	Sex	-	-	-
tblPatient	Dob	-	-	-

dbMappingTable	dbColName	db ExtraTable	db Criteria
tblPatientNextOfKin	PatientID		
tblPatient	PatientName		tblPatient.PatientID = tblPatientNextOfKin.NextOfKinID
tblPatientNextOfKin	relationship		
tblPatientHealthHistory	HealthCondition		tblPatientHealthHistory.PatientID = tblPatientNextOfKin.NextOfKinID
tblPatient	Sex		tblPatient.PatientID = tblPatientNextOfKin.NextOfKinID
tblPatient	Dob		tblPatient.PatientID = tblPatientNextOfKin.NextOfKinID

Fig. 9: Schema mapping metadata for tables that are mapped by specifying additional conditions

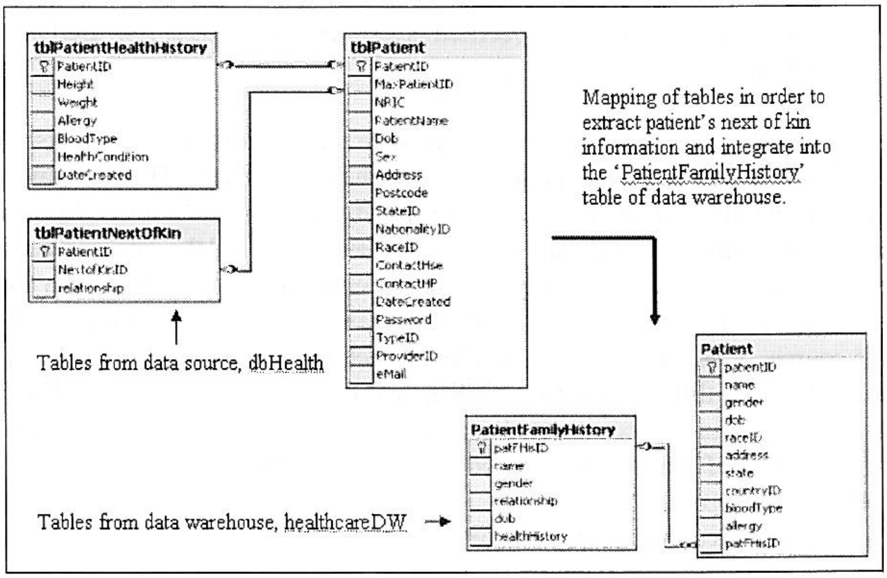


Fig. 10: Example of mapping tables by specifying additional conditions

MySQL, dbBase and Interbase. The data warehouse used is the SQL Server and the algorithms for automating data uploading are based on the data warehouse schema design. Any changes to this schema design, e.g. adding new tables or attributes, will require updating the data uploading functions as these are implemented by calling insert stored procedures. Further work can be done to automate the data loading process so that it supports other data warehouse schema designs. Algorithms can also be designed so that they are independent of the data warehouse schema design. Changes made to data structures can automatically create new or update existing insert stored procedures. This can reduce development time and make maintenance easier.

The tool can also be extended to include validation of schema mapping. When mapping attributes in a data source to the corresponding attributes in the data warehouse, most of the related metadata information such as data types, table and column descriptions, and integrity constraints (primary key or foreign key) are loaded automatically. Users may want to enter additional tables and conditions. This can create problems such as schema mapping becoming more error prone. For example, users may enter table names, conditions, or SQL syntax that are invalid. This can affect the data extraction process since the schema mapping metadata is used for expressing the ETL process. Validation of schema mapping can ensure that the mapping is done correctly.

The tool provides limited data cleaning to improve the data quality such as detecting and removing errors and inconsistencies, checking for null values, eliminating duplicate records, and resolving structure and semantic conflicts. This is done before loading the transformed data into the data warehouse. Currently, the tool only deals with attribute semantic conflicts, but not with the semantic conflicts in the actual data. The latter occurs when data are extracted from different data sources where different formats are integrated, e.g. date based on UK format (DD/MM/YYYY) and date based on US format (MM/DD/YYYY). A similar situation arises when integrating currency and metric data. To handle semantic conflicts in data, generic functions can be implemented to convert data source data to the corresponding data warehouse data. During the schema mapping, other information such as the unit used in the data source, can be captured so that the system can automatically perform the conversion function during the data transformation process.

Other enhancements include updating records of the data warehouse based on changes made to the data source, developing an update scheduler, developing OLAP with data mining to support the decision-making process, and developing algorithms to auto-generate the dimensional tables to facilitate OLAP and data mining functions. Issues such as scalability, maintainability, and security can also be explored.

REFERENCES

Phipps, C. (2002). *Migrating an operational database schema to data warehouse schemas.* Unpublished PhD dissertation. University of Cincinnati.

Poole, J. & Mellor, D. (2001*). Common warehouse metamodel: An introduction to the standard for data warehouse integration (1st ed.).* New York: John Wiley & Sons.

Tan, J. & Zaslavsky, A. (2003). Domain-specific metamodels for heterogeneous information systems. *Proceedings of the 36th Hawaii International Conference on System Sciences, HICSS'03.*

Zhengxin, C., (2002). *Intelligent data warehousing: From data preparation to data mining.* New York: John Wiley & Sons.