# A proposed model for data warehouse ETL processes

**3 authors:**

Shaker El-Sappagh
Benha University

**22** PUBLICATIONS   **137** CITATIONS

SEE PROFILE

Abdeltawab M. Ahmed Hendawi
University of Virginia

**41** PUBLICATIONS   **161** CITATIONS

SEE PROFILE

Ali El-Bastawissy
Modern Sciences and Arts University

**62** PUBLICATIONS   **149** CITATIONS

SEE PROFILE

**Some of the authors of this publication are also working on these related projects:**

Project   record linkage View project

Project   Master of Science in DW Modeling View project

King Saud University

**Journal of King Saud University –
Computer and Information Sciences**

www.ksu.edu.sa
www.sciencedirect.com

# ORIGINAL ARTICLE

# A proposed model for data warehouse ETL processes

**Shaker H. Ali El-Sappagh [a],\*, Abdeltawab M. Ahmed Hendawi [b],
Ali Hamed El Bastawissy [b]**

[a] Mathematics Department, College of Science, King Saud University, Saudi Arabia
[b] Information Systems Department, Faculty of Computers and Information, Cairo University, Cairo, Egypt

**Abstract**  Extraction–transformation–loading (ETL) tools are pieces of software responsible for the extraction of data from several sources, its cleansing, customization, reformatting, integration, and insertion into a data warehouse. Building the ETL process is potentially one of the biggest tasks of building a warehouse; it is complex, time consuming, and consume most of data warehouse project's implementation efforts, costs, and resources. Building a data warehouse requires focusing closely on understanding three main areas: the source area, the destination area, and the mapping area (ETL processes). The source area has standard models such as entity relationship diagram, and the destination area has standard models such as star schema, but the mapping area has not a standard model till now. In spite of the importance of ETL processes, little research has been done in this area due to its complexity. There is a clear lack of a standard model that can be used to represent the ETL scenarios. In this paper we will try to navigate through the efforts done to conceptualize

Production and hosting by Elsevier

the ETL processes. Research in the field of modeling ETL processes can be categorized into three main approaches: Modeling based on mapping expressions and guidelines, modeling based on conceptual constructs, and modeling based on UML environment. These projects try to represent the main mapping activities at the conceptual level. Due to the variation and differences between the proposed solutions for the conceptual design of ETL processes and due to their limitations, this paper also will propose a model for conceptual design of ETL processes. The proposed model is built upon the enhancement of the models in the previous models to support some missing mapping features.

## 1. Introduction

A data warehouse (DW) is a collection of technologies aimed at enabling the decision maker to make better and faster decisions. Data warehouses differ from operational databases in that they are subject oriented, integrated, time variant, non volatile, summarized, larger, not normalized, and perform OLAP. The generic data warehouse architecture consists of three layers (data sources, DSA, and primary data warehouse) (Inmon, 2002; Vassiliadis, 2000). Although ETL processes area is very important, it has little research. This is because of its difficulty and lack of formal model for representing ETL activities that map the incoming data from different DSs to be in a suitable format for loading to the target DW or DM (Kimball and Caserta, 2004; Demarest, 1997; Oracle Corp., 2001; Inmon, 1997). To build a DW we must run the ETL tool which has three tasks: (1) data is extracted from different data sources, (2) propagated to the data staging area where it is transformed and cleansed, and then (3) loaded to the data warehouse. ETL tools are a category of specialized tools with the task of dealing with data warehouse homogeneity, cleaning, transforming, and loading problems (Shilakes and Tylman, 1998). This research will try to find a formal representation model for capturing the ETL processes that map the incoming data from different DSs to be in a suitable format for loading to the target DW or DM. Many research projects try to represent the main mapping activities at the conceptual level. Our objective is to propose conceptual model to be used in modeling various ETL processes and cover the limitations of the previous research projects. The proposed model will be used to design ETL scenarios, and document, customize, and simplify the tracing of the mapping between the data source attributes and its corresponding in data warehouse. The proposed model has the following characteristics:

- *Simple*: to be understood by the DW designer.
- *Complete*: to represent all activities of the ETL processes.
- *Customizable*: to be used in different DW environments.

We call the proposed model entity mapping diagram (EMD). Also, the paper will make a survey of the previous work done in this area. The paper will be organized as follows: Section 2 will discuss the ETL modeling concepts. The ETL processes related or previous work is discussed in Section 3. We will discuss the proposed framework in Section 4. The comparison between the previous model and proposed one is discussed in Section 5. Next, other related works will be shown in Section 6. Finally, Section 7 shows the conclusion and future work.

## 2. ETL modeling concepts

The general framework for ETL processes is shown in Fig. 1. Data is *extracted* from different data sources, and then propagated to the DSA where it is *transformed* and cleansed before being *loaded* to the data warehouse. Source, staging area, and target environments may have many different data structure formats as flat files, XML data sets, relational tables, non-relational sources, web log sources, legacy systems, and spreadsheets.

### 2.1. The ETL phases

During the ETL process, data is extracted from an OLTP databases, transformed to match the data warehouse schema, and loaded into the data warehouse database (Berson and Smith, 1997; Moss, 2005). Many data warehouses also incorporate data from non-OLTP systems, such as text files, legacy systems, and spreadsheets. ETL is often a complex combination of process and technology that consumes a significant portion of the data warehouse development efforts and requires the skills of business analysts, database designers, and application developers. The ETL process is not a one-time event. As data sources change the data warehouse will periodically updated. Also, as business changes the DW system needs to change – in order to maintain its value as a tool for decision makers, as a result of that the ETL also changes and evolves. The ETL processes must be designed for ease of modification. A solid, well-designed, and documented ETL system is necessary for the success of a data warehouse project.

An ETL system consists of three consecutive functional steps: extraction, transformation, and loading:

#### 2.1.1. Extraction
The first step in any ETL scenario is data extraction. The ETL extraction step is responsible for extracting data from the source systems. Each data source has its distinct set of characteristics that need to be managed in order to effectively extract data for the ETL process. The process needs to effectively integrate systems that have different platforms, such as different
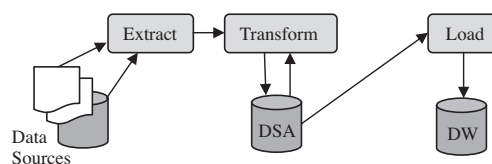


**Figure 1**   A general framework for ETL processes.

database management systems, different operating systems, and different communications protocols.

During extracting data from different data sources, the ETL team should be aware of (a) using ODBC\JDBC drivers connect to database sources, (b) understand the data structure of sources, and (c) know how to handle the sources with different nature such as mainframes. The extraction process consists of two phases, initial extraction, and changed data extraction. In *the initial extraction* (Kimball et al., 1998), it is the first time to get the data from the different operational sources to be loaded into the data warehouse. This process is done only one time after building the DW to populate it with a huge amount of data from source systems. *The incremental extraction* is called changed data capture (CDC) where the ETL processes refresh the DW with the modified and added data in the source systems since the last extraction. This process is periodic according to the refresh cycle and business needs. It also captures only changed data since the last extraction by using many techniques as audit columns, database log, system date, or delta technique.

### 2.1.2. Transformation
The second step in any ETL scenario is data transformation. The transformation step tends to make some cleaning and conforming on the incoming data to gain accurate data which is correct, complete, consistent, and unambiguous. This process includes data cleaning, transformation, and integration. It defines the granularity of fact tables, the dimension tables, DW schema (stare or snowflake), derived facts, slowly changing dimensions, factless fact tables. All transformation rules and the resulting schemas are described in the metadata repository.

### 2.1.3. Loading
Loading data to the target multidimensional structure is the final ETL step. In this step, extracted and transformed data is written into the dimensional structures actually accessed by the end users and application systems. Loading step includes both loading dimension tables and loading fact tables.

## 3. Models of ETL processes

This section will navigate through the efforts done to conceptualize the ETL processes. Although the ETL processes are critical in building and maintaining the DW systems, there is a clear lack of a standard model that can be used to represent the ETL scenarios. After we build our model, we will make a comparison between this model and models discussed in this section. Research in the field of modeling ETL processes can be categorized into three main approaches:

1. Modeling based on mapping expressions and guidelines.
2. Modeling based on conceptual constructs.
3. Modeling based on UML environment.

In the following, a brief description of each approach is presented.

### 3.1. Modeling ETL process using mapping expressions

Rifaieh and Benharkat (2002) have defined a model covering different types of mapping expressions. They used this model to create an active ETL tool. In their approach, queries are used to achieve the warehousing process. Queries will be used to represent the mapping between the source and the target data; thus, allowing DBMS to play an expanded role as a data transformation engine as well as a data store. This approach enables a complete interaction between mapping metadata and the warehousing tool. In addition, it addresses the efficiency of a query-based data warehousing ETL tool without suggesting any graphical models. It describes a query generator for reusable and more efficient data warehouse (DW) processing.

### 3.1.1. Mapping guideline
Mapping guideline means the set of information defined by the developers in order to achieve the mapping between the attributes of two schemas. Actually, different kinds of mapping guidelines are used for many applications. Traditionally, these guidelines are defined manually during the system implementation. In the best case, they are saved as paper documents. These guidelines are used as references each time there is a need to understand how an attribute of a target schema has been generated from the sources attributes. This method is very weak in the maintenance and evolution of the system. To keep updating these guidelines is a very hard task, especially with different versions of guidelines. To update the mapping of an attribute in the system, one should include an update for the paper document guideline as well. Thus, it is extremely difficult to maintain such tasks especially with simultaneous updates by different users.

### 3.1.2. Mapping expressions
Mapping expression of an attribute is the information needed to recognize how a target attribute is created from the sources attributes. Examples of the applications where mapping expressions are used are listed as follows:

- *Schema mapping* (Madhavan et al., 2001): for database schema mapping, the mapping expression is needed to define the correspondence between matched elements.
- *Data warehousing tool* (*ETL*) (Staudt et al., 1999): includes a transformation process where the correspondence between the sources data and the target DW data is defined.
- *EDI message mapping*: the need of a complex message translation is required for EDI, where data must be transformed from one EDI message format into another.
- *EAI* (*enterprise application integration*): the integration of information systems and applications needs a middleware to manage this process (Stonebraker and Hellerstein, 2001). It includes management rules of an enterprise's applications, data spread rules for concerned applications, and data conversion rules. Indeed, data conversion rules define the mapping expression of integrated data.

### 3.1.3. Mapping expression examples
Some examples of the mapping expressions identified from different type of applications are shown as follows:

- *Break-down/concatenation*: in this example the value of a field is established by breaking down the value of a source and by concatenating it with another value, as shown in Fig. 2.
- *Conditional mapping*: sometimes the value of a target attribute depends on the value of another attribute. In the example, if X = 1 then Y = A else Y = B, as shown in Fig. 3. More about mapping expression rules and notation are found in Jarke et al. (2003) and Miller et al. (2000).
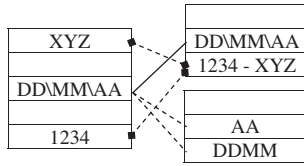
**Figure 2** Example 1: Break-down/concatenation (Jarke et al., 2003).

### 3.2. Modeling ETL processes using conceptual constructs

In Vassiliadis et al. (2002a, 2003, 2005) the authors attempt to provide a first model towards the conceptual modeling of the data warehousing ETL processes. They introduce a framework for the modeling of ETL activities. Their framework contains three layers, as shown in Fig. 4.

*The lower layer* namely; schema layer, involves a specific ETL scenario. All the entities of the schema layer are *instances* of the classes data type, function type, elementary activity, recordset and relationship.

*The higher layer* namely; metamodel layer involves the aforementioned classes. The linkage between the metamodel and the schema layers is achieved through instantiation ("instanceOf") relationships. The metamodel layer implements the aforementioned generality: the five classes which are involved in the metamodel layer are generic enough to model any ETL scenario, through the appropriate instantiation.

*The middle layer* is the template layer. The constructs in the template layer are also meta-classes, but they are quite custom-
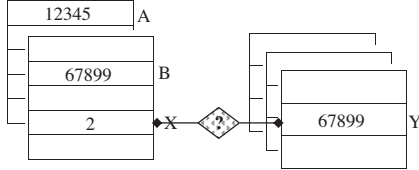
ized for the regular cases of ETL processes. Thus, the classes of the template layer represent specializations (i.e., subclasses) of the generic classes of the metamodel layer (depicted as "IsA" relationships). After defining the previous framework, the authors present the graphical notation and the metamodel of their proposed graphical model as shown in Fig. 5. Then, they detail and formally define all the entities of the metamodel:

– *Data types.* Each data type T is characterized by a name and a domain which is a countable set of values. The values of the domains are also referred to as constants.
– *Recordsets.* A recordset is characterized by its name, its logical schema (structure of the recordset) and its physical extension (i.e., a finite set of records under the recordset schema) which is the actual records values. Any data structure can be treated as a "record set" provided that there are the means to logically restructure it into a flat, typed record schema. The two most popular types of recordsets are namely relational tables and record files.
– *Functions.* A function type comprises a name, a finite list of parameter data types, and a single return data type. A function is an instance of a function type.
– *Elementary activities.* Activities are logical abstractions representing parts, or full modules of code. An abstraction of the source code of an activity is employed, in the form of a LDL (logic-programming, declarative language) statement, in order to avoid dealing with the peculiarities of a particular programming language (Naqvi and Tsur, 1989).



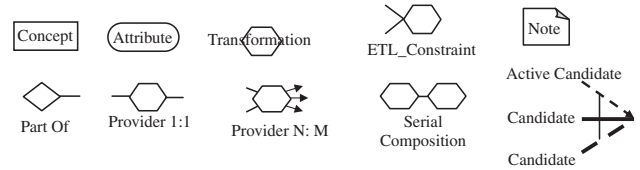**Figure 3** Example 2: Conditional mapping (Jarke et al., 2003).



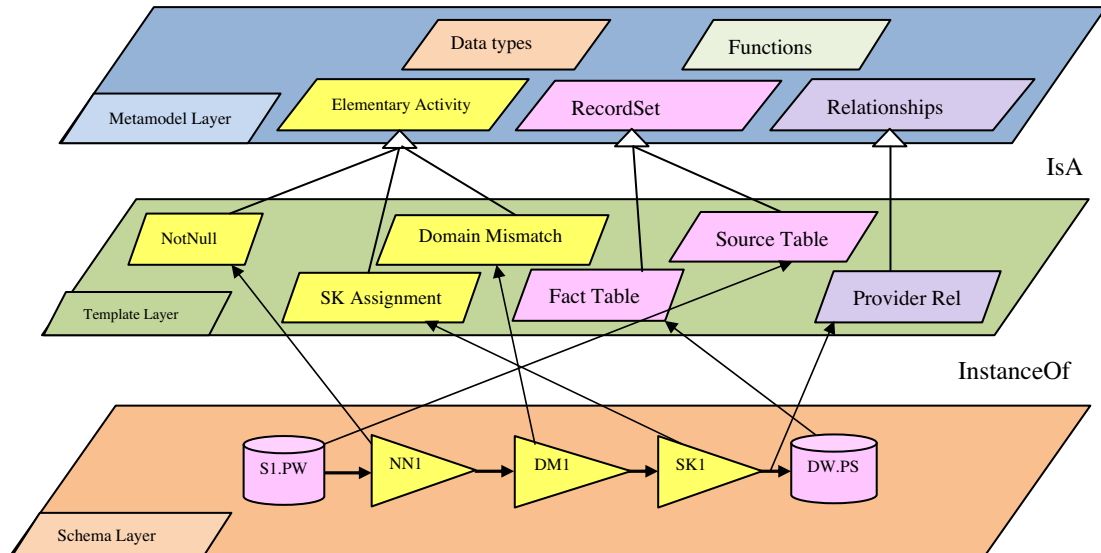**Figure 5** Notations for the conceptual modeling of ETL activities (Vassiliadis et al., 2002a).



**Figure 4** The metamodel for the logical entities of the ETL environment (Vassiliadis et al., 2003).

– *Relationships*. Depict the follow of data from the sources to the target.

Then the authors use their graphical model to represent ETL processes in a motivating example. As shown in Fig. 6, two data sources (S1.partsupp and S2.partsupp) are used to build the target data warehouse (DW.partsupp). The conceptual model of Vassiliadis et al. (2002a) is complemented in Vassiliadis et al. (2002b, 2003) and Simitsis (2003) with the logical design of ETL processes as data-centric workflows. In Vassiliadis et al. (2003) the authors describe a framework for the declarative specification of ETL scenarios. They discuss the implementation issues and they present a graphical tool 'ARKTOS II' that facilitates the design of ETL scenarios, based on their model. In Vassiliadis et al. (2002b) the authors model an ETL scenario as a graph which they call architectural graph and they introduce some notations for this graph. They introduce importance metrics to measure the degree to which entities are bound to each other. In Simitsis (2003) the author focuses on the optimization of the ETL processes, in order to minimize the execution time of an ETL process. Regarding data mapping, in Dobre et al. (2003) authors discuss issues related to the data mapping in the integration of data, and a set of mapping operators is introduced and a classification of possible mapping cases is presented, as shown in Fig. 7. However, no graphical representation of data mapping scenarios is provided, hence, it is difficult to be used in real world projects. In Bernstein and Rahm (2000) a framework for mapping between models (objects) is proposed.

Models are manipulated by a role of high-level operations including:

- Match – create a mapping between two models.
- Apply Function – apply a given function to all objects in a model.

| | |
|---|---|
| Union (U) | -Add (+), Subtract (-), Divide (/), Multiply (*) |
| Intersection (∩) | -Rename (Ren), Concatenate (Con), Split (Sp) |
| Join ( ) | - Data type conversion. |
| Difference (Δ) | ☐ ToNumeric (TN) |
| Division (/) | ☐ ToString (TS) |
| Multiply (*) | ☐ ToFloat (TF) |
| Rename (RN) | - Data format conversion |
| Duplicate Elimination (DE) | ☐ ToUpperCase (TUC) |
| | ☐ ToLowerCase (TLC) |
| | ☐ ToAmericanDate (TAD) |

**Figure 7** Sample mapping operators.

- Union, Intersection, Difference – applied to a set of objects.
- Delete – delete all objects in a model.
- Insert, Update – applied to individual objects in models.

### 3.3. Modeling based on UML environment

In Lujan-Mora et al. (2004) the authors introduce their model that is based on the UML (unified modeling language) notations. It is known that UML does not contain a direct relationship between attributes in different classes, but the relationship is established between the classes itself, so the authors extend UML to model attributes as first-class citizens. In their attempt to provide complementary views of the design artifacts in different levels of detail, the framework is based on a principled approach in the usage of UML packages, to allow zooming in and out the design of a scenario.

### 3.3.1. Framework
The architecture of a data warehouse is usually depicted as various layers of data in which data from one layer is derived from the data of the previous layer (Lujan-Mora and Trujillo, 2003). Following this consideration, the development of a DW can be structured into an integrated framework with five stages
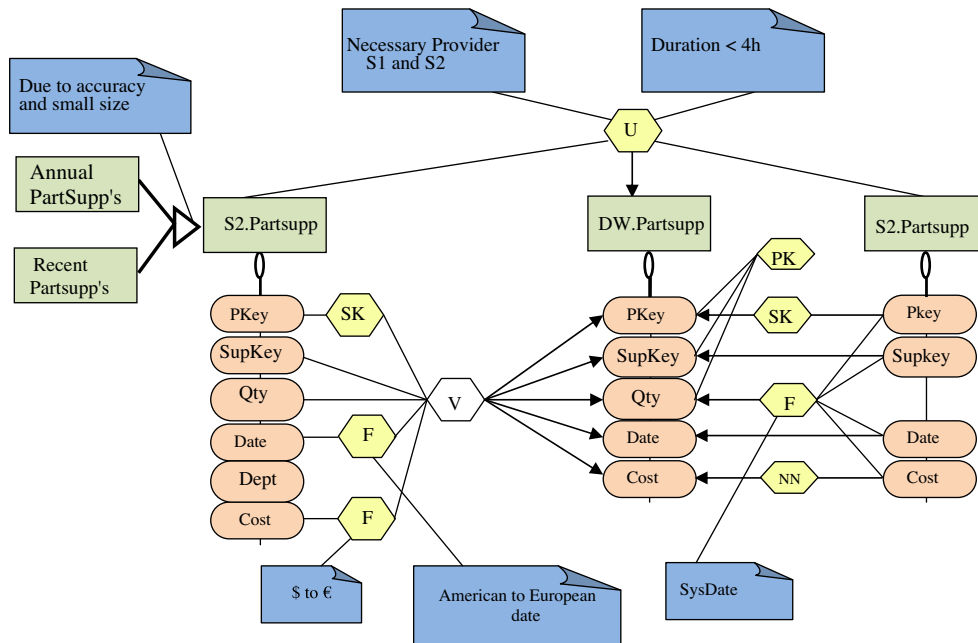


**Figure 6** Motivating example for conceptual model in Vassiliadis et al. (2002a).

and three levels that define different diagrams for the DW model, as explained below:

– **Phases**: there are five stages in the definition of a DW:
● *Source*: it defines the data sources of the DW, such as OLTP systems, external data sources.
● *Integration*: it defines the mapping between the data sources and the data warehouse.
● *Data warehouse*: it defines the structure of the data warehouse.
● *Customization*: it defines the mapping between the data warehouse and the clients' structures.
● *Client*: it defines special structures that are used by the clients to access the data warehouse, such as data marts or OLAP applications.
– **Levels**: each stage can be analyzed at three levels or perspectives:
● *Conceptual*: it defines the data warehouse from a conceptual point of view.
● *Logical*: it addresses logical aspects of the DW design, as the definition of the ETL processes.
● *Physical*: it defines physical aspects of the DW, such as the storage of the logical structures in different disks, or the configuration of the database servers that support the DW.

### 3.3.2. Attributes as first-class modeling elements (FCME)

Both in ERD model and in UML, attributes are embedded in the definition of their comprising "element" (an entity in the ER or a class in UML), and it is not possible to create a relationship between two attributes. In order to allow attributes to play the same role in certain cases, the authors propose the representation of attributes as FCME in UML. In a UML class diagram, two kinds of modeling elements are treated as FCME. Classes, as abstract representations of real-world entities are naturally found in the center of the modeling effort. Being FCME, classes acting as attribute containers. The relationships between classes are captured by associations. Associations can also be FCME, called association classes. An association class can contain attributes or can be connected to other classes. However, the same is not possible with attributes. They refer to the class that contains the attributes as the container class and the class that represents an attribute as the

attribute class. The authors formally define attribute/class diagrams, along with the new stereotypes, ⟨⟨Attribute⟩⟩ and ⟨⟨Contain⟩⟩, defined as follows:

*Attribute classes* are materializations of the ⟨⟨Attribute⟩⟩ stereotype, introduced specifically for representing the attributes of a class. The following constraints apply for the correct definition of an attribute class as a materialization of an ⟨⟨Attribute⟩⟩ stereotype:

– *Naming convention:* the name of the attribute class is the name of the corresponding container class, followed by a dot and the name of the attribute.
– *Features*: an attribute class can contain neither attributes nor methods.

*A contain relationship* is a composite aggregation between a container class and its corresponding attribute classes, originated at the end near the container class and highlighted with the ⟨⟨Contain⟩⟩ stereotype.

*An attribute/class diagram* is a regular UML class diagram extended with ⟨⟨Attribute⟩⟩ classes and ⟨⟨Contain⟩⟩ relationships.

In the data warehouse context, the relationship, involves three logical parties: (a) the provider entity (schema, table, or attribute), responsible for generating the data to be further propagated, (b) the consumer, that receives the data from the provider and (c) their intermediate matching that involves the way the mapping is done, along with any transformation and filtering. Since a mapping diagram can be very complex, this approach offers the possibility to organize it in different levels thanks to the use of UML packages.

Their layered proposal consists of four levels as shown in Fig. 8:

– *Database level (or level 0)*. In this level, each schema of the DW environment (e.g., data sources at the conceptual level in the SCS 'source conceptual schema', conceptual schema of the DW in the DWCS 'data warehouse conceptual schema', etc.) is represented as a package (Lujan-Mora and Trujillo, 2003; Trujillo and Lujan-Mora, 2003). The mappings among the different schemata are modeled in a single mapping package, encapsulating all the lower-level mappings among different schemata.
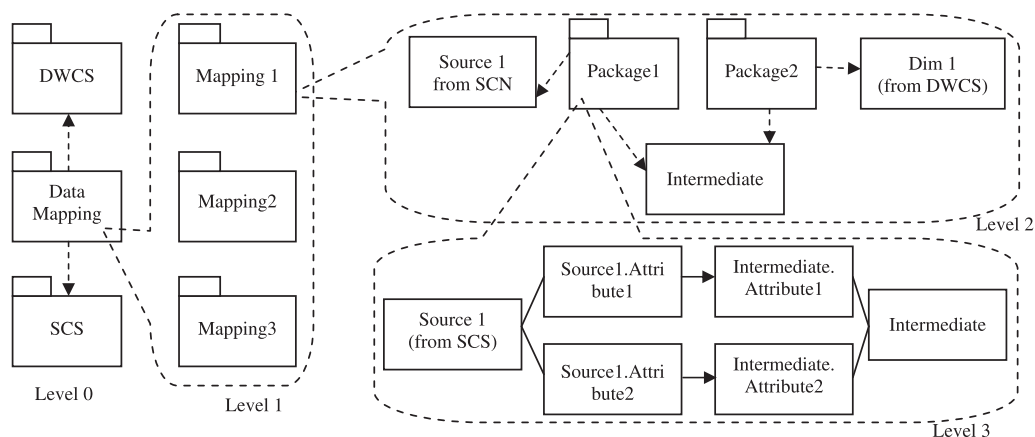


**Figure 8**   Data mapping levels (Lujan-Mora et al., 2004).

– *Dataflow level (or level 1)*. This level describes the data relationship among the individual source tables of the involved schemata towards the respective targets in the DW. Practically, a mapping diagram at the database level is zoomed-into a set of more detailed mapping diagrams, each capturing how a target table is related to source tables in terms of data.

– *Table level (or level 2)*. Whereas the mapping diagram of the dataflow level describes the data relationships among sources and targets using a single package, the mapping diagram at the table level, details all the intermediate transformations and checks that take place during this flow. Practically, if a mapping is simple, a single package that represents the mapping can be used at this level; otherwise, a set of packages is used to segment complex data mappings in sequential steps.

– *Attribute level (or level 3)*. In this level, the mapping diagram involves the capturing of inter-attribute mappings. Practically, this means that the diagram of the table is zoomed-in and the mapping of provider to consumer attributes is traced, along with any intermediate transformation and cleaning.

At the leftmost part of Fig. 8, a simple relationship among the DWCS and the SCS exists: this is captured by a single data mapping package and these three design elements constitute the data mapping diagram of the database level (or level 0). Assuming that there are three particular tables in the DW that we would like to populate, this particular data mapping package abstracts the fact that there are three main scenarios for the population of the DW, one for each of these tables. In the dataflow level (or level 1) of our framework, the data relationships among the sources and the targets in the context of each of the scenarios, is practically modeled by the respective package. If we zoom in one of these scenarios, e.g., mapping 1, we can observe its particularities in terms of data transformation and cleaning: the data of source 1 are transformed in two steps (i.e., they have undergone two different transformations), as shown in Fig. 8. Observe also that there is an intermediate data store employed, to hold the output of the first transformation (Step 1), before passed onto the second one (Step 2). Finally, at the right lower part of Fig. 8, the way the attributes are mapped to each other for the data stores source 1 and intermediate is depicted. Let us point out that in case we are modeling a complex and huge data warehouse, the attribute transformation modeled at level 3 is hidden within a package definition.

## 4. The proposed ETL processes model (EMD)

To conceptualize the ETL processes used to map data from sources to the target data warehouse schema, we studied the previous research projects, made some integration, and add some extensions to the approaches mentioned above. We propose entity mapping diagram (EMD) as a new conceptual model for modeling ETL processes scenarios. Our proposed model mainly follows the approach of modeling based on conceptual constructs. The proposed model will fulfill six requirements (El Bastawesy et al., 2005; Maier, 2004; Arya et al., 2006):

1. Supports the integration of multiple data sources.
2. Is robust in view of changing data sources.
3. Supports flexible transformations.
4. Can be easily deployed in a suitable implementation environment.
5. Is complete enough to handle the various extraction, transformation, and loading operations.
6. Is simple in creating and maintaining.

In this section, we will describe EMD framework, EMD metamodel, primitives of EMD constructs, and finally we will provide a demonstration example. A comparison and evaluation of the previous approaches against our proposed model will be presented in Section 5.

### 4.1. EMD framework

Fig. 9 shows the general framework of the proposed entity mapping diagram.

– In the data source(s) part: the participated data sources are drawn. The data sources may be structured databases or non-structured sources. In case of structured sources; the participated databases and their participated tables and attributes are used directly as the base source, and in case of non-structured sources; a conversion step should be applied first to convert the non-structured source into structured one (tables and its attributes). From the design view, there is one conversion construct that can convert any non-structured source into structured (relational) database, but from the implementation view, each type of non-structured source will have its own conversion module which is called wrapper. Wrappers are specialized program routines that automatically extract data from different data sources with
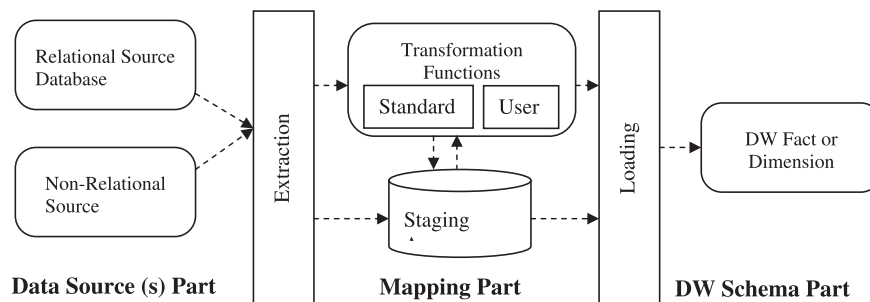


**Figure 9** A general framework of EMD.

different formats and convert the information into a structured format. The typical tasks of a wrapper are: (a) fetching data from a remote resource, (b) searching for, recognizing and extracting specified data, and (c) saving this data in a suitable structured format to enable further manipulation (Vassiliadis et al., 2005).

– Extraction: during the extraction process some temporary tables may be created to hold the result of converting non-structured sources into databases. The extraction process includes initial extraction and refresh. The initial extraction takes place when the ETL scenario executed for the first time while there is no data in the destination data warehouse. The refresh extraction takes place to capture the delta data (difference between old data in the DW and updated data in the data sources). It is preferred to separate the ETL scenario with initial extraction from the ETL scenario with refresh extraction. *This means that the user may need to build two EMD models for the same ETL scenario*; one for the initial extraction, and the other for the refresh extraction using the old data in the temp tables found in the staging area.

– In the DW schema part: the data warehouse schema table (fact or dimension) is drawn. In spite of that the fact table and the dimension table are clearly different in their functionalities and features but all of them are data containers. Basically the data warehouse is stored as relational structure not as multidimensional structure. The multidimensionality occurs in the online analytical processing (OLAP) engines.

– In the mapping part: the required transformation functions are drawn. The transformation operations take place on the incoming data from both the base source and/or the temporary source in the staging area. Some transformation operations lead to temporary results which are saved in temporary tables in the staging area.

– The staging area: a physical container that contains all temporary tables created during the extraction process or resulted from the applied transformation functions.

– Loading: as the data reaches the final appropriate format, it is loaded to the corresponding data element in the destination DW schema. The data may be loaded directly as a result of certain transformation function or captured from the desired temporary tables in the staging area.

Notice that both data sources and data warehouse schemas should be defined clearly before starting to draw EMD. Also the arrows' directions show that first, the data sources are drawn, after that a set of transformation are applied, and then the data are loaded to the destination data warehouse schema.

### 4.2. EMD metamodel

EMD is a proposed conceptual model for modeling the ETL processes which are needed to map data from sources to the target data warehouse schema. Fig. 10 shows the metamodel architecture for the proposed conceptual model EMD. The metamodel of the proposed EMD is composed of two layers; the first layer is abstraction layer in which five objects (function, data container, entity, relationship, and attribute) are clearly defined. The objects in the abstraction layer are a high level view of the parts or objects that can be used to draw an EMD scenario.

The second layer is the template layer which is an expansion to the abstraction layer.

The link between the abstraction layer and the template layer may be considered as an aggregation relationship. A function may be an attribute transformation, an entity transformation, a UDF (user defined function), or convert into structure (relation). Fig. 11 shows the types of transformation functions that can be applied to sources in the proposed EMD.

An entity transformation is a function that can be applied to a source table (e.g. duplicate elimination, union, etc.). An attribute transformation function can be applied to a source attribute (e.g. to upper case, to String, etc.). A user defined function (UDF) is any function that may be added by the user who is the creator of the ETL scenario (e.g. unification between different types of units). Convert into structure is a function that can be applied to the non-structured (semi-structured and unstructured) data sources so that it can be converted into structured source to enable the other transformation functions
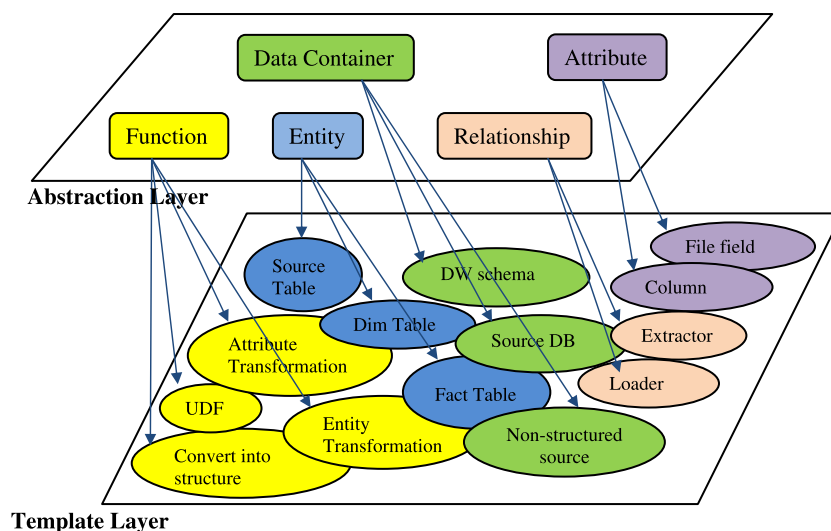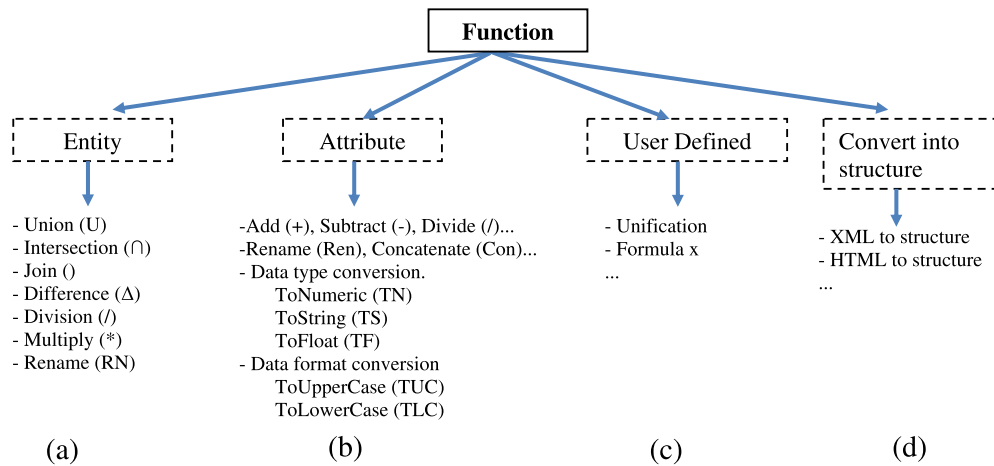


**Figure 10**    EMD metamodel.

**Function**

Entity

- Union (U)
- Intersection (∩)
- Join ()
- Difference (Δ)
- Division (/)
- Multiply (*)
- Rename (RN)

(a)

Attribute

-Add (+), Subtract (-), Divide (/)...
-Rename (Ren), Concatenate (Con)...
- Data type conversion.
    ToNumeric (TN)
    ToString (TS)
    ToFloat (TF)
- Data format conversion
    ToUpperCase (TUC)
    ToLowerCase (TLC)

(b)

User Defined

- Unification
- Formula x
...

(c)

Convert into structure

- XML to structure
- HTML to structure
...

(d)

**Figure 11**    Types of transformations in EMD.

to be applied on it. A data container may be a source database, a target data warehouse or data mart, or non-structured source. An entity may be a source table, a dimension table, or a fact table. A relationship may be an extractor or a loader. The extractor expresses the data extraction process from the source and the loader expresses the data loading process to the final destination. The attribute may be a table column or a non-structured file field.

The metamodel can be expanded to include any extra objects that may be required in the future. The user can use instances of the template layer to create his model to build the desired ETL scenario. It should be mentioned here that the user of EMD is the data warehouse designer or the ETL designer; this means that some primitive rules, limitations, and constrains are kept in his mind during the usage of different parts of EMD, i.e., union operation will be applied successfully when the participated tables have the same number of attributes with the same data types for the corresponding attributes.

### 4.3. Primitives of EMD constructs

The basic set of constructs that is used in the proposed entity mapping diagram are shown in Fig. 12. In this section, some explanation about the usage of the constructs of the proposed entity mapping diagram will be given, as follows:

- *Loader relationship*: is used when the data are moved directly from the last source element (the actual source or the temporary one) to the target data element. The actual source; is the base source from which the data are extracted, on the other hand, the temporary source; is the one that is resulted during the transformation operations.
- *Optional loader relationship*: is used to show that the loaded data to the output attribute could be extracted from candidate source element x or candidate source element y.
- *Convert into structure*: represents the conversion operations required to restructure the non-structured base source into structured one (relations as tables and attributes). The

| Mapping Construct | | To Represent |
|---|---|---|
| Name | Shape | |
| Cylinder | | Schema |
| Rectangle | | Entity |
| Oval | | Attribute |
| Diamond with rounded arrow | | Convert into structure |
| Solid arrow | | Loader Relationship |
| Connected arrows | | Optional Loader Relationship |
| Square with rounded edge | | Attribute Transformation |
| Square with triangle edge | | User Defined Function (UDF) |
| Hexagon | | Entity Transformation operation |
| Document | | Non-structured source |
| Rectangle with folded corner | | User Note |

**Figure 12**    Graphical constructs for the proposed EMD.

conversion operation saves its result into temporary tables, so the transformation operation can be applied to the new temporary source.

- *Entity transformation operation*: this kind of transformations usually results in a temporary entity. There are standard operators that are used inside this construct, Fig. 11(a) shows some of these operators.
- *Attribute transformation operation*: standard operations are used with this construct, Fig. 11(b) shows sample of these operators.
- *User defined function* (*UDF*) *as a transformation operation*: user can use his defined operations, so any kind of transformation can be added, such as currency conversion func-

tions, packages (units) conversions, and so on, as shown in Fig. 11(c).
- *Non-structured source*: represents any source that is not in the relational structure. The non-structured source may be semi-structured or unstructured source such as XML files, web logs, excel workbook, object oriented database, etc., as shown in Fig. 11(d).

Notice that a symbol or shorthand of the operation is put inside the entity or the attribute transformations construct. The transformation functions that take place in the proposed model EMD are classified into built-in or standard functions, such as join, union, and rename, and user defined functions as mentioned above, like any formula defined by the user. Another classification for the transformation functions according to the level of transformation is entity transformations functions, and attribute transformations functions.

### 4.4. Demonstration example

To illustrate the usage of our proposed graphical model, we introduce a simple example. A company wants to build a data warehouse for monitoring the sales processes in its two branches. It has a relational data source described by schema DS1 for selling books, shown in Fig. 13, another relational data source described by schema DS2 for selling general products, shown in Fig. 14. A relational data warehouse is designed to capture sales data from the two predefined data sources. The star schema in Fig. 15 shows the design of the proposed data warehouse which consists of one fact table and four dimensions tables.

Fig. 16 depicts the entity mapping diagram for building the products dimension from the desired data sources, passing through the required ETL activities. The explanation of this diagram is as follows:
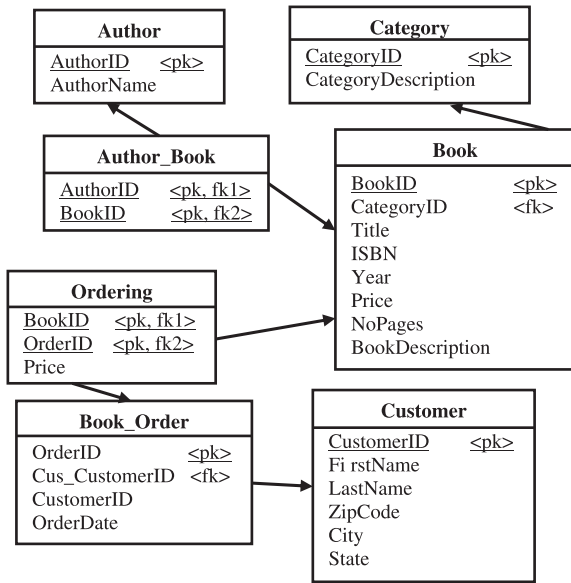
- DS1: refers to the first data source (books-orders database).



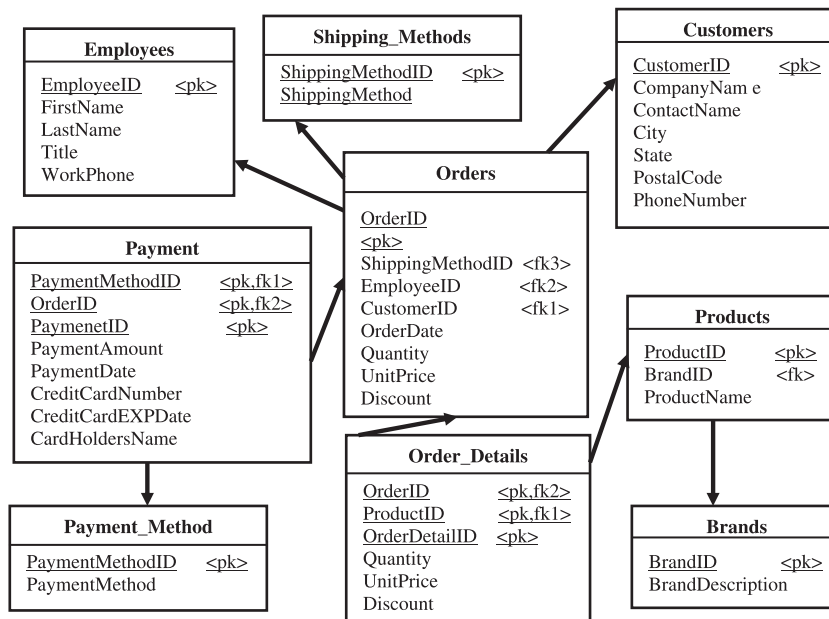**Figure 13**  Relational schema DS1 for books-orders database.



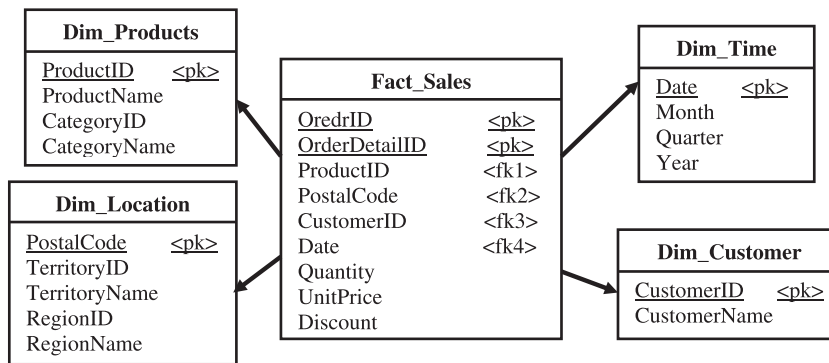**Figure 14**  Relational schema DS2 for products-orders database.

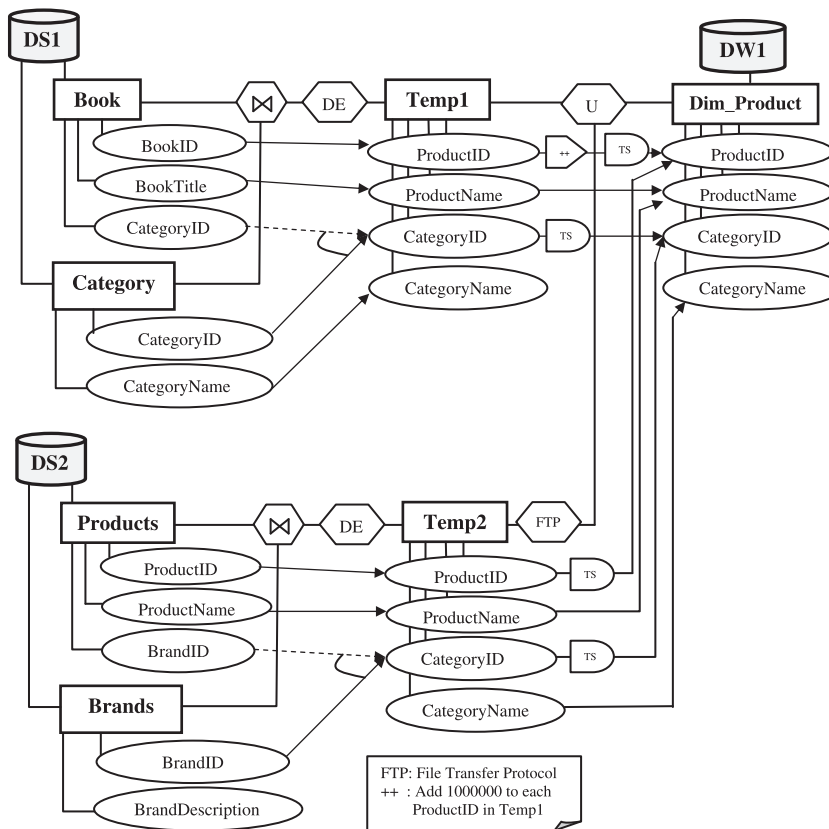**Figure 15**    Star schema for the proposed data warehouse.



**Figure 16**    EMD scenario for building products dimension.

- DS2: refers to the second data source (products-orders database).

There are two entities from each data source that participate in this diagram: Book (BookID, BookTitle, CategoryID) and Category (CategoryID, CategoryName) from the first data source, and Products (ProductID, ProductName, BrandID) and Brands (BrandID, CategoryName) from the second data source.

DW1: refers to the data warehouse schema to which the data will be moved, we may have one or more DW schemas, one or more data mart (DM) schemas, or a combination of DW and DM. Dim_Products is a dimension entity found in DW1. In the middle of the diagram, mapping processes are represented using a set of transformation steps; starting with join operation between Book and Category tables, then removing the redundant records by applying the duplicate elimination operation.

Temporary entity (Temp1) is created to capture the intermediate data that result from the previous operations. Notice that data of attribute Temp1.CategoryID could be loaded optionally from DS1.Book.CategoryID or DS1.Category.CategoryID. The same activities take place in the other site that contains DS2 to result Temp2 table.

After that, some attribute transformation operations take place before loading data to the target data warehouse, some of them are described as follows: ( + + ) is a user defined transformation operation applied to Temp1.ProductID to add

10,00,000 to each product code number as a user requirement. ProductID and CategoryID data types are transformed to string data type by using ToString (TS) operation. Temp2 table is transferred to the site of DS1 using file transfer protocol (FTP) operation, then a union operation (U) runs to combine the two tables. The loader relationships connected to Product-Name and CategoryName attributes mean that data is loaded from these two attributes to their corresponding attributes in the DW without any transformation.

We can now develop a prototype tool (named EMD Builder) to achieve the following tasks:

– Introducing a tool for drawing the entity mapping diagram scenarios using a pallet of graphical controls.
– Implementing a set of transformation operations.
– Transforming the graphical model to a code by generating SQL script.
– Generating the mapping document according to Kimball's standards (Kimball and Caserta, 2004).
– Executing the EMD scenario on the data sources to apply the extraction, and transformation operations, then loading data to the target DW schema.
– The code of may be written in C# or JAVA object-oriented programming languages and a rational database management system as Oracle or Microsoft SQL Server.

We propose the architecture in Fig. 17 for the model, and in the future work we will implement and test this model.

The first module checks the connection to the database management system installed on the machine on which the source databases exist. If the connection failed, an error mes-

sage will appear to alert the user and the application will halt. If the connection succeeded, new database "ETL" will be created. "ETL" plays the role of repository in which the metadata about the EMD scenarios will be stored. The metadata in the repository will be used to generate the mapping document. After creating "ETL" database the user may either create new EMD scenario or open existing one to complete it. In case of creating new scenario, new building area will appear to enable the user to draw and build his new model, and in case of opening an existing EMD scenario, two files will be read, the first one is ".etl" file from which the old scenario will be loaded to the drawing area to enable the user to complete it, and the second file is ".sql" in which the SQL script of the old part of the existing scenario were written and will be complete as the user completes his model. The next module loads both the metadata about the databases found on the database management system and "EMD Builder" interface icons. The metadata includes the databases names, tables, attributes, and so on. The interface icons will be loaded from our icon gallery, the interface elements will be shown in next sections. The next module facilitates the drawing process by which the user can use our palette of controls to draw and build his EMD scenario. By using the execution module, the EMD model will be translated into SQL script then executed on the incoming data from the source databases, so the extraction, transformation, and loading processes can be applied and the desired records will be transferred to the target DW schema in the required format. The last module is responsible for saving the user's EMD model. During the save operation, three files are generates; the first one contains the user EMD model in a binary format, so the user can open it at any time to update in its drawing, the second contains the generated SQL script, and the third generated file is the mapping document which is considered as dictionary and catalog for the ETL operations found in the user EMD scenario. The user can specify the folder in which the generated files will be saved. The generated files can be transferred from one machine to be used on another one that contains the same data sources and the same target data warehouse schema; this means that the generated files from our tool are machine independent, however they are data source and destination schema dependent. It is clear that the destination is a whole schema (data warehouse or data mart), but each part of this schema (fact or dimension) is handled as a standalone destination in a single EMD scenario.

## 5. Models evaluation and comparison

Table 1 contains the matrix that is used to compare the different ETL modeling approaches and evaluates our proposed model against the other models. The letter P in the matrix means that this model had partially supported the corresponding criteria.

## 6. Other related work

The ETL process, in data warehouse, is a hot point of research because of its importance and cost in data warehouse project building and maintenance. The method of systematic review to identify, extract and analyze the main proposals on modeling conceptual ETL processes for DWs (Muñoz et al., 2010a). Generating ETL processes for incremental loading (Jörg and
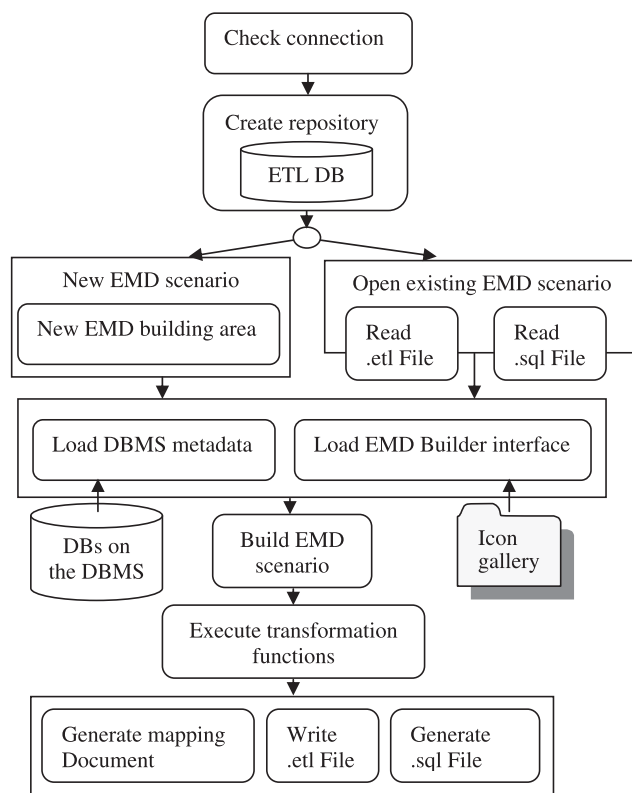


**Figure 17**    Basic modules of "EMD Builder".

**Table 1** Comparison and evaluation matrix.

| Criteria | Model | | | |
|---|---|---|---|---|
| | Mapping expressions | Conceptual constructs | UML environment | EMD |
| *Design aspects* | | | | |
| Complete graphical model | No | Yes | Yes | Yes |
| New constructs | No | Yes | No | Yes |
| (OO) concept independent | Yes | P | No | Yes |
| DBMS independent | Yes | Yes | Yes | Yes |
| Mapping operations | Yes | Yes | Yes | Yes |
| User defined transformation | No | No | No | Yes |
| Mapping relationship | Yes | Yes | Yes | Yes |
| Source independent (non-relational) | No | No | No | Yes |
| Source converting | No | No | No | Yes |
| Flat model | Yes | Yes | No | Yes |
| | | | | |
| *Implementation aspects* | | | | |
| Develop a tool | Yes | Yes | No | Yes |
| Generate SQL | Yes | No | No | Yes |
| Generate mapping document | No | No | No | Yes |
| Non-relational handling | No | No | No | No |
| Evaluation | 7 | 7.5 | 4 | 13 |

Yes = 1; No = 0; P: partial = 0.5; total = 14.

Deßloch, 2008). A simulation model for secure data extraction in ETL processes (Mrunalini et al., 2009). A set of measures with which to evaluate the structural complexity of ETL process models at the conceptual level discussed in Muñoz et al. (2010b). In Simitsis and Vassiliadis (2008) the author discusses the mapping of the conceptual model to the logical model. Generating an incremental ETL process automatically from the full ETL process is discussed in Zhang et al. (2008). In Simitsis et al. (2008) the author discusses the application of natural language generation techniques to the ETL environment. Measures the ETL processes models in data warehouses are discussed in Muñoz et al. (2009).

## 7. Conclusion and future work

ETL processes are very important problem in the current research of data warehousing. In this paper, we have investigated a very important problem in the current research of data warehousing. This problem represents a real need to find a standard conceptual model for representing in simplified way the extraction, transformation, and loading (ETL) processes. Some approaches have been introduced to handle this problem. We have classified these approaches into three categories; fist, is modeling based on mapping expressions and guidelines, second, is modeling based on conceptual constructs, and the final category, is modeling based on UML environment. We have explained each model in some detail.

What is more, we proposed a novel conceptual model entity mapping diagram (EMD) as a simplified model for representing extraction, transformation, and loading processes in data warehousing projects. In order to explain our proposed model; we defined a metamodel for the entity mapping diagram. In the metamodel we defined two layers; the first is the abstraction layer in which five objects (function, data container, entity, relationship, and attribute) are clearly defined. The objects in the abstraction layer are a high level view of the parts or objects that can be used to draw an EMD scenario. The second

is the template layer which is an expansion to the abstraction layer. The user can add his own layer in which the ETL designer draws his EMD scenario. We also set a framework for using this model. The framework consists of data sources part, data warehouse schema part, and mapping part. Both data sources and data warehouse schemas should be defined clearly before starting to draw EMD scenario. By comparing the proposed model to the previous research projects using the evaluation matrix, the proposed model handle may weak points that appear in the previous work. In the future work to this paper, we will develop and test a prototype tool call it 'EMD Builder' to achieve the following tasks: introducing a tool for drawing the entity mapping diagram scenarios using a pallet of graphical constructs, implementing a set of transformation operations, transforming the graphical model to a code by generating SQL script, and generating the mapping document according to Kimball's standards.

## References

Arya, P., Slany, W., Schindler, C., 2006. Enhancing Wrapper Usability through Ontology Sharing and Large Scale Cooperation. <www.ru5.cti.gr/HT05/files/andreas_rath.ppt> (accessed 2006).

Bernstein, P., Rahm, E., 2000. Data warehouse scenarios for model management. In: Proceedings of the 19th International Conference on Conceptual Modeling (ER'00), LNCS, vol. 1920, Salt Lake City, USA, pp. 1–15.

Berson, A., Smith, S.J., 1997. Data Warehousing, Data Mining, and OLAP. McGraw-Hill.

Demarest, M., 1997. The Politics of Data Warehousing. <http://www.hevanet.com/demarest/marc/dwpol.html>.

Dobre, A., Hakimpour, F., Dittrich, K.R., 2003. Operators and classification for data mapping in semantic integration. In: Proceedings of the 22nd International Conference on Conceptual Modeling (ER'03), LNCS, vol. 2813, Chicago, USA, pp. 534–547.

El Bastawesy, A., Boshra, M., Hendawi, A., 2005. Entity mapping diagram for modeling ETL processes. In: Proceedings of the Third International Conference on Informatics and Systems (INFOS), Cairo.

Inmon, B., 1997. The Data Warehouse Budget. DM Review Magazine, January 1997. < www.dmreview.com/master.cfm?NavID = 55& EdID = 1315 > .

Inmon, W.H., 2002. Building the Data Warehouse, third ed. John Wiley and Sons, USA.

Jarke, M., Lenzerini, M., Vassiliou, Y., Vassiliadis, P., 2003. Fundamentals of Data Warehouses, second ed. Springer-Verlag.

Jörg, Thomas, Deßloch, Stefan, 2008. Towards generating ETL processes for incremental loading. In: ACM Proceedings of the 2008 International Symposium on Database Engineering and Applications.

Kimball, R., Caserta, J., 2004. The Data Warehouse ETL Toolkit. Practical Techniques for Extracting, Cleaning, Conforming and Delivering Data. Wiley.

Kimball, R., Reeves, L., Ross, M., Thornthwaite, W., 1998. The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouses. John Wiley and Sons.

Lujan-Mora, S., Trujillo, J., 2003. A comprehensive method for data warehouse design. In: Proceedings of the Fifth International Workshop on Design and Management of Data Warehouses (DMDW'03), Berlin, Germany.

Lujan-Mora, S., Vassiliadis, P., Trujillo, J., 2004. Data mapping diagram for data warehouse design with UML. In: International Conference on Conceptual Modeling, Shanghai, China, November 2004.

Madhavan, J., Bernstein, P.A., Rahm, E., 2001. Generic schema matching with cupid. In: Proceedings of the 27th International Conferences on Very Large Databases, pp. 49–58.

Maier, T., 2004. A formal model of the ETL process for OLAP-based web usage analysis. In: Proceedings of the Sixth WEBKDD Workshop: Webmining and Web Usage Analysis (WEBKDD'04), in conjunction with the 10th ACM SIGKDD Conference (KDD'04), Seattle, Washington, USA, August 22, 2004 (accessed 2006).

Miller, R.J., Haas, L.M., Hernandez, M.A., 2000. Schema mapping as query discovery. In: Proceedings of the 26th VLDB Conference, Cairo.

Moss, L.T., 2005. Moving Your ETL Process into Primetime. < http://www.businessintelligence.com//ex/asp/code.44/xe/article.htm > (visited June 2005).

Mrunalini, M., Kumar, T.V.S., Kanth, K.R., 2009. Simulating secure data extraction in extraction transformation loading (ETL) processes. In: IEEE Computer Modeling and Simulation Conference. EMS'09. Third UKSim European Symposium, November 2009, pp. 142–147. ISBN: 978-1-4244-5345-0.

Muñoz, Lilia, Mazón, Jose-Norberto, Trujillo, Juan, 2009. Measures for ETL processes models in data warehouses. In: ACM Proceeding of the First International Workshop on Model Driven Service Engineering and Data Quality and Security, November 2009.

Muñoz, Lilia, Mazon, Jose-Norberto, Trujillo, Juan, 2010. Systematic review and comparison of modeling ETL processes in data warehouse. In: Proceedings of the Fifth Iberian Conference on IEEE Information Systems and Technologies (CISTI), August 2010, pp. 1–6. ISBN: 978-1-4244-7227-7.

Muñoz, Lilia, Mazónand, Jose-Norberto, Trujillo, Juan, 2010b. A family of experiments to validate measures for UML activity diagrams of ETL processes in data warehouses. Information and Software Technology 52 (11), 1188–1203.

Naqvi, S., Tsur, S., 1989. A Logical Language for Data and Knowledge Bases. Computer Science Press.

Oracle Corp., 2001. Oracle9i™ Warehouse Builder User's Guide, Release 9.0.2, November 2001. < http://www.otn.oracle.com/products/warehouse/content.html > .

Rifaieh, R., Benharkat, N.A., 2002. Query-based data warehousing tool. In: Proceedings of the Fifth ACM International Workshop on Data Warehousing and OLAP, November 2002.

Shilakes, C., Tylman, J., 1998. Enterprise Information Portals. Enterprise Software Team. < http://www.sagemaker.com/company/downloads/eip/indepth.pdf > .

Simitsis, A., 2003. Modeling and Managing ETL Processes. VLDB Ph.D. Workshop.

Simitsis, Alkis, Vassiliadis, Panos, 2008. A method for the mapping of conceptual designs to logical blueprints for ETL processes. Decision Support Systems, Data Warehousing and OLAP 45 (1), 22–40.

Simitsis, Alkis, Skoutas, Dimitrios, Castellanos, Malú, 2008. Natural language reporting for ETL processes. In: Proceeding of the ACM 11th International Workshop on Data Warehousing and OLAP, pp. 65–72. ISBN: 978-1-60558-250-4.

Staudt, M., Vaduva, A., Vetterli, T., 1999. Metadata Management and Data Warehousing. Technical Report, The Department of Information Technology (IFI) at the University of Zurich.

Stonebraker, M., Hellerstein, J., 2001. Content integration for e-business. In: Proceedings of the ACM SIGMOD/PODS 2001, Santa Barbara, CA, May 21–24, 2001.

Trujillo, J., Lujan-Mora, S., 2003. A UML based approach for modeling ETL processes in data warehouses. In: Proceedings of the 22nd International Conference on Conceptual Modeling. LNCS, Chicago, USA.

Vassiliadis, P., 2000. Data Warehouse Modeling and Quality Issues. Ph.D. Thesis, Department of Electrical and Computer Engineering, National Technical University of Athens (Greece).

Vassiliadis, P., Simitsis, A., Skiadopoulos, S., 2002. Conceptual modeling for ETL processes. In: Proceedings of the Fifth ACM International Workshop on Data Warehousing and OLAP, pp. 14–21.

Vassiliadis, P., Simitsis, A., Skiadopoulos, S., 2002. Modeling ETL activities as graphs. In: Proceedings of the Fourth International Workshop on the Design and Management of Data Warehouses (DMDW'02), Toronto, Canada, pp. 52–61.

Vassiliadis, P., Simitsis, A., Georgantas, P., Terrovitis, M., 2003. A framework for the design of ETL scenarios. In: Proceedings of the 15th CAiSE, Velden, Austria, June 16, 2003.

Vassiliadis, P., Simitsis, A., Georgantas, P., Terrovitis, M., Skiadopoulos, S., 2005. A generic and customizable framework for the design of ETL scenarios. Information Systems Journal.

Zhang, Xufeng, Sun, Weiwei, Wang, Wei, Feng, Yahui, Shi, Baile, 2008. Generating incremental ETL processes automatically. In: IEEE Computer and Computational Sciences, pp. 516–521.