Conceptual modeling for ETL processes

Author: Dhananjay Patil
Organization: Evaltech, Inc.
Evaltech Research Group,
Data Warehousing Practice.
Date: 08/26/04
Email: erg@evaltech.com

Abstract:
Due to the complexity and the long learning curve of ETL tools, many organizations prefer
to turn to in-house development to perform ETL and data cleaning tasks. Thus, it is
important that the design, development and deployment of ETL processes, which is
currently performed in an ad-hoc, in-house fashion, needs modeling, design and
methodological foundations. This white paper describes the problem of the definition of
ETL activities and provides formal foundations for their conceptual representation. The
proposed conceptual model is customized, enriched and constructed.

Intellectual Property / Copyright Material

All text and graphics found in this article are the property of the Evaltech, Inc.
and cannot be used or duplicated without the express written permission of the
corporation through the Office of Evaltech, Inc.
Summary

Due to the complexity and the long learning curve of ETL tools, many organizations prefer to turn to in-house development to perform ETL and data cleaning tasks. Thus, it is important that the design, development and deployment of ETL processes, which is currently performed in an ad-hoc, in house fashion, needs modeling, design and methodological foundations. This white paper describes the problem of the definition of ETL activities and provides formal foundations for their conceptual representation. The proposed conceptual model is customized, enriched and constructed.

This white paper includes following topics,

- Introduction to ETL processes
- Related work in the field of conceptual modeling
- Conceptual model
- Instantiation and Specialization layers
- Conclusion

Introduction

The proposed conceptual model is customized, enriched and constructed in the following manner.

- Customized for the tracing of inter-attribute relationships and the respective ETL activities in the early stages of a data warehouse project;
- Enriched with a 'palette' of a set of frequently used ETL activities, like the assignment of surrogate keys, the check for null values, etc; and
- Constructed in a customizable and extensible manner, so that the designer can enrich it with his own re-occurring patterns for ETL activities.

ETL tools are a category of specialized tools with the task of dealing with data warehouse homogeneity, cleaning and loading problems.

Following figure describe the general framework for ETL processes.
In the above figure bottom layer is a data store that are involved in the overall process. The data from the sources are extracted by extraction routines, which provide either complete snapshots or differentials of the data sources. Then, these data are propagated to the Data Staging Area (DSA) where they are transformed and cleaned before being loaded to the data warehouse. The data warehouse comprises the target data stores, i.e., fact tables and dimension tables. Eventually, the loading of the central warehouse is performed through the loading activities.

During the Data Warehouse design period, the data warehouse designer is concerned with two tasks, which are practically executed in parallel. The first of these tasks involves the collection of requirements from the part of the users. The second task, which is of equal importance for the success of the data warehousing project, involves the analysis of the structure and content of the existing data sources and their intentional mapping to the common data warehouse model.

Crucial task is the mapping of the attributes of the data sources to the attributes of the data warehouse tables.

Thus it is important that a simple conceptual model is employed in order to (a) facilitate the smooth redefinition and revision efforts and (b) serve as the means of communication with the rest of the involved parties.

### Related work in the field of conceptual modeling

- **Conceptual models for data warehouses**

  The front end of the data warehouse has monopolized the research on the conceptual part of data warehouse modeling. In fact, most of the work on conceptual modeling, in the field of data warehousing, has been dedicated to the capturing of the conceptual characteristics of the star schema of the warehouse and the subsequent data marts and aggregations.

  Major trends, includes: (a) dimensional modeling (b) standard E/R modeling   (c) UML modeling and (d) sui-generis models without a clear winner.

- **Conceptual models for ETL**

  As a first attempt to clearly separate the data warehouse refreshment process from its traditional treatment as a view maintenance or bulk loading process. However, any transformation is de-referenced for the logical model where a couple of generic operators are employed to perform this task.

- **ETL logical and physical aspects**

  Apart from the commercial ETL tools there also exist research efforts.
Conceptual modeling for ETL processes

**Conceptual Model**

The conceptual model for ETL activities is to specify the high level, user-oriented entities which are used to capture the semantics of the ETL process.

Following diagram shows the conceptual modeling for ETL activities and the different entities of the proposed model.
Following diagram shows the proposed meta model as UML diagram.
Consider that model contains two source databases S1 and S2 as well as a central data warehouse DW. The available concepts for these databases are listed in following Fig., along with their attributes. The scenario involves the propagation of data from the concept PARTSUPP of source S1 as well as from the concept PARTSUPP of source S2 to the data warehouse. Table DW.PARTSUPP stores daily (DATE) information for the available quantity (QTY) and cost (COST) of parts (PKEY) per supplier (SUPPKEY). We assume that the first supplier is European and the second is American, thus the data coming from the second source need to be converted to European values and formats.

<table>
<thead>
<tr>
<th>Concept</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1.PARTSUPP</td>
<td>PKEY, SUPPKEY, QTY, COST</td>
</tr>
<tr>
<td>S2.PARTSUPP</td>
<td>PKEY, SUPPKEY, DEPARTMENT, DATE, QTY, COST</td>
</tr>
<tr>
<td>DW.PARTSUPP</td>
<td>PKEY, SUPPKEY, DATE, QTY, COST</td>
</tr>
</tbody>
</table>

- **Concepts and Attributes**

  - **Attributes**: A granular module of information. The role of attributes is the same as in the standard ER/dimensional models.

  - **Concepts**: A concept represents an entity in the source databases or in the data warehouse. Concept instances are the files in the source databases, the data warehouse fact and dimension tables and so on. A concept is formally defined by a name and a finite set of attributes. In terms of the ER model, a concept is a generalization of entities and relationships; depending on the employed model (dimensional model or ER extension) all entities composed of a set of attributes are generally instances of class Concept.

  We can treat several physical storage structures as finite lists of fields, including relational databases, COBOL or simple ASCII files, multidimensional cubes and dimensions. Concepts are fully capable of modeling this kind of structures, possibly through a generalization (ISA) mechanism.

  Let us take OLAP structures as an example. Employing simply concepts is sufficient for the problem of ETL modeling. In the example one can observe several concepts. The concepts S1.PARTSUPP, S2.PARTSUPP and DW.PARTSUPP are shown in above Fig., along with their respective attributes.

- **Transformations, Constraints and Notes**

  - **Transformations**. Transformations are abstractions that represent parts, or full modules of code, executing a single task. Two large categories of transformations include (a) filtering or data cleaning operations, like the check for primary or foreign key violations and (b) transformation operations, during which the schema of the incoming data is transformed (e.g., aggregation). Formally, a transformation is defined by (a) a finite set of input attributes; (b) a finite set of output attributes and (c) a symbol that graphically characterizes the nature of the transformation. A transformation is graphically depicted as a hexagon tagged with its corresponding symbol. In the example, can observe several transformations. Note the ones pertinent to the mapping of S1.PARTSUPP to DW.PARTSUPP. Also can observe a surrogate key assignment transformation (SK), a function application calculating the system date (f) and a Not Null (NN) check for attribute Cost.
ETL Constraints. There are several occasions when the designer wants to express the fact that the data of a certain concept fulfill several requirements. For example, the designer might wish to impose a primary key or null value constraint over a (set of) attribute(s). This is achieved through the application of ETL constraints, which are formally defined as follows: (a) a finite set of attributes, over which the constraint is imposed and (b) a single transformation, which implements the enforcement of the constraint. Note, that despite the similarity in the name, ETL constraints are different modeling elements from the well-known UML constraints. An ETL constraint is graphically depicted as a set of solid edges starting from the involved attributes and targeting the facilitator transformation. In the example, observe that we apply a Primary Key ETL constraint to DW.PARTSUPP for the attributes PKey, SuppKey, Date.

Notes. Exactly as in UML modeling, notes are informal tags to capture extra comments that the designer wishes to make during the design phase or render UML constraints attached to an element or set of elements. As in UML, notes are depicted as rectangles with a dog-eared corner. In our framework, notes are used for:

- Simple comments explaining design decisions.
- Explanations of the semantics of the applied transformations. For example, in the case of relational selections/joins this involves the specification of the respective selection/join condition, whereas in the case of functions this would involve the specification of the function signatures.
- Tracing of runtime constraints that range over different aspects of the ETL process, such as the time/event based scheduling, monitoring, logging, error handling, crash recovery etc. For example, in the upper part of above Fig., can observe a runtime constraint specifying that the overall execution time for the loading of DW.PARTSUPP (that involves the loading of S1.PARTSUPP and S2.PARTSUPP) cannot take longer than 4 hours.

Part-Of and Candidate Relationships

Part-of Relationships. The fact, that a concept is composed of a set of attributes. In general, standard ER modeling does not treat this kind of relationship as a first-class citizen of the model; UML modeling on the other hand, hides attributes inside classes and treats part-of relationships with a much broader meaning. We do not wish to redefine UML part-of relationships, but rather to emphasize the relationship of a concept with its attributes (since we need attributes as first class citizens in the inter-attribute mappings). Naturally, we do not preclude the usage of the part-of relationship for other purposes, as in standard UML modeling. As usually, a part-of relationship is denoted by an edge with a small diamond at the side of the container object.

Candidate relationships. In the case of data warehousing, it is most common a phenomenon, especially in the early stages of the project, to have more than one candidate source files/tables that could populate a target, data warehouse table. Thus, a set of candidate relationships captures the fact that a certain data warehouse concept can be populated by more than one candidate source concepts. Formally, a candidate relationship comprises (a) a single candidate concept and (b) a single target concept. Candidate relationships are depicted with bold dotted lines between the candidates and the target concepts. Whenever exactly one of them can be selected, we annotate the set of candidate relationships for the same concept with a UML {XOR} constraint.

Active candidate relationships. An active candidate relationship denotes the fact that out of a set of candidates, a certain one has been selected for the population of the target concept. Thus, an active candidate relationship is a specialization of candidate relationships, with the same structure and refined semantics. We denote an active candidate relationship with a directed bold
dotted arrow from the provider towards the target concept. For the purpose of our motivating example, let us assume that source S2 has more than one production systems (e.g., COBOL files), which are candidates for S2.PARTSUPP. Assume that the available candidates (depicted in the left upper part of Fig. 5) are:

A concept AnnualPartSupp’s (practically representing a file F1), that contains the full annual history about part suppliers; it is used basically for reporting purposes and contains a superset of fields than the ones required for the purpose of the data warehouse.

A concept RecentPartSupp’s (practically representing a file F2), containing only the data of the last month; it used on-line by end-users for the insertion or update of data as well as for some reporting applications. The diagram also shows that RecentPartSupp’s was eventually selected as the active candidate; a note captures the details of this design choice.

### Provider Relationships and Serial Composition of Transformations

**Provider relationships.** A provider relationship maps a set of input attributes to a set of output attributes through a relevant transformation. In the simple 1:1 case, provider relationships capture the fact that an input attribute in the source side populates an output attribute in the data warehouse side.

If the attributes are semantically and physically compatible, no transformation is required. If this is not the case though, we pass this mapping through the appropriate transformation (e.g., European to American data format, not null check, etc.). In general, it is possible that some form of schema restructuring takes place; thus, the formal definition of provider relationships comprises:

(a) a finite set of input attributes;
(b) a finite set of output attributes;
(c) an appropriate transformation (i.e., one whose input and output attributes can be mapped one to one to the respective attributes of the relationship).

In the case of N:M relationships the graphical representation obscures the linkage between provider and target attributes. To compensate for this shortcoming, we annotate the link of a provider relationship with each of the involved attributes with a tag, so that there is no dis-ambiguity for the actual provider of a target attribute. In the 1:1 case, a provider relationship is depicted with a solid bold directed arrow from the input towards the output attribute, tagged with the participating transformation. In the general N:M case, a provider relationship is graphically depicted as a set of solid arrows starting from the providers and targeting the consumers, all passing through the facilitator transformation.

Let us examine now, the relationship between the attributes of concepts S1.PARTSUPP, S2.PARTSUPP and DW.PARTSUPP, as depicted in above Fig.

- **Attribute PKey** is directly populated from its homonymous attribute in S1 and S2, through a surrogate key (SK) transformation. Surrogate Key assignment is common tactics in data warehousing, employed in order to replace the keys of the production systems with a uniform key. For example, it could be the case that the part ‘Steering Wheel’ has PKEY=30 for source S1, PKEY=40 for source S2, while at the same time source S2 has PKEY=30 for part ‘Automobile Door’. These conflicts can be easily resolved by a global replacement mechanism through the assignment of a uniform surrogate key.

- **Attribute SuppKey** is populated from the homonymous attributes in the sources.

- **Attribute Date** is directly populated from its homonymous attribute in S2, through an American-To-European Date transformation function. At the same time, the date for the rows coming from source S1, is determined through the application of a Sysdate() function (since S1.PARTSUPP does not have a corresponding attribute). Observe the
function applied for the rows coming from source S1: it uses as input all the attributes of S1.PARTSUPP (in order to determine that the produced value is a new attribute of the row), passes through a function application transformation calculating the system date, which, in turns, is directed to attribute DW.Date.

- Attribute Qty is directly populated from its homonymous attributes in the two sources, without the need for any transformation.

Note also that there can be input attributes, like for example S2.PARTSUPP.Department, which are ignored during the ETL process.

**Transformation Serial Composition.** It is possible that to combine several transformations in a single provider relationship.

For example, to group incoming data with respect to a set of attributes, having ensured at the same time that no null values are involved in this operation. In this case, there is need to perform a not null check for each of the attributes and propagate only the correct rows to the aggregation. In order to model this setting, we employ a serial composition of the involved transformations. The problem would be the requirement that a transformation has a set of attributes as inputs and attributes; thus, simply connecting two transformations would be inconsistent. To compensate this shortcoming, we employ a serial transformation composition. Formally, a serial transformation composition comprises (a) a single initiating transformation and (b) a single subsequent transformation. Serial transformation compositions are depicted as solid bold lines connecting the two involved transformations. A rather complex part of the motivating example is the aggregation that takes place for the rows of source S2. Practically, source S2 captures information for part suppliers according to the particular department of the supplier organization. Loading this data to the data warehouse that ignores this kind of detail requires the aggregation of data per PKey, SuppKey, Date and the summation of Cost and Qty. This is performed from the aggregation transformation. Note also the tags on the output of the aggregation transformation, determining their providers (e.g., S2.PARTSUPP.PKey for DW.PARTSUPP.PKey and SUM[S2.PARTSUPP Qty] for DW.PARTSUPP Qty).

### Instantiation and specialization layers

The key issue in the conceptual representation of ETL activities lies (a) on the identification of a small set of generic constructs that are powerful enough to capture all cases and (b) on an extensibility mechanism that allows the construction of a 'palette' of frequently used types (e.g., for data stores and activities).

The Meta model layer implements the aforementioned generosity desideratum: the five classes, which are involved in the Meta model layer, are generic enough to model any ETL scenario, through the appropriate instantiation.

The simple provision of a meta- and an instance layer, to make Meta model truly useful for practical cases of ETL processes with a set of ETL-specific constructs, which constitute a subset of the larger meta model layer, namely the Template Layer. The constructs in the Template layer are also meta-classes, but they are quite customized for the regular cases of ETL processes. Thus, the classes of the Template layer as specializations (i.e., subclasses) of the generic classes of the Met model layer (depicted as “IsA” relationships in the Fig). Through this customization mechanism, the designer can pick the instances of the Schema layer from a much richer palette of constructs; in this setting, the entities of the Schema layer are instantiations, not only of the respective classes of the Meta model layer, but also of their subclasses in the Template layer.
The framework for the modeling of ETL activities in the example of Fig the concept DW.PARTSUPP must be populated from a certain source S2. Several operations must intervene during the propagation: for example, a surrogate key assignment and an aggregation take place in the scenario.

Moreover, there are two candidates suitable for the concept S2.PARTSUPP; out of them, exactly one (Candidate2) is eventually selected for the task. As one can observe, the concepts that take part in this scenario are instances of class Concept (belonging to the meta model layer) and specifically of its subclass ER Entity (assuming that we adopt an ER model extension). Instances and encompassing classes are related through links of type instanceOf. The same mechanism applies to all the transformations of the scenario, which are (a) instances of class Transformation and (b) instances of one of its subclasses. Observe the how the provider links from the concept S2.PARTSUPP towards the concept DW.PARTSUPP are related to class Provider Relationship through the appropriate instanceOf links. As far as the class Concept is concerned, in the Template layer.

## Conclusion

Extraction-Transformation-Loading (ETL) tools are pieces of software responsible for the extraction of data from several sources, their cleansing, customization and insertion into a data warehouse.

The first objective is the linkage of the proposed conceptual model to its logical and physical counterparts, with particular focus (a) on the relationship of the ETL activities to the underlying data stores; (b) the capturing of the composite workflow of the ETL scenario and (c) the optimization of its execution.