

A UML Based Approach for Modeling ETL Processes in Data Warehouses

Juan Trujillo, Sergio Luján-Mora
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante
{slujan, jtrujillo}@dlsi.ua.es

Abstract

Data warehouses (DWs) are complex computer systems whose main goal is to facilitate the decision making process of knowledge workers. ETL (Extraction-Transformation-Loading) processes are responsible for the *extraction* of data from heterogeneous operational data sources, their *transformation* (conversion, cleaning, normalization, etc.) and their *loading* into DWs. ETL processes are a key component of DWs because incorrect or misleading data will produce wrong business decisions, and therefore, a correct design of these processes at early stages of a DW project is absolutely necessary to improve data quality. However, not much research has dealt with the modeling of ETL processes. In this paper, we present our approach, based on the Unified Modeling Language (UML), which allows us to accomplish the conceptual modeling of these ETL processes together with the conceptual schema of the target DW in an integrated manner. We provide the necessary mechanisms for an easy and quick specification of the common operations defined in these ETL processes such as, the integration of different data sources, the transformation between source and target attributes, the generation of surrogate keys and so on. Moreover, our approach allows the designer a comprehensive tracking and documentation of entire ETL processes, which enormously facilitates the maintenance of these processes. Another advantage of our proposal is the use of the UML (standardization, ease-of-use and functionality) and the seamless integration of the design of the ETL processes with the DW conceptual schema. Finally, we show how to use our integrated approach by using a well-known modeling tool such as Rational Rose.

Keywords: ETL processes, Data warehouses, conceptual modeling, UML

1 Introduction

In the early nineties, Inmon [Inmon92] coined the term “data warehouse” (DW): “A data warehouse is a subject-oriented, integrated, time-variant, nonvolatile collection of data in support of management’s decisions”. A DW is “integrated” because data are gathered into the DW from a variety of sources and merged into a coherent whole. ETL (Extraction-Transformation-Loading) processes are responsible for the extraction of data from heterogeneous operational data sources, their transformation (conversion, cleaning, normalization, etc.) and their loading into DWs.

It is highly recognized that the design and maintenance of these ETL processes is a key factor of success in DW projects for several reasons [SQL02, Strange02]. DWs are usually populated by data from different and heterogeneous operational data sources such as legacy systems, relational databases, COBOL files, Internet (XML, web logs) and so on. Therefore, different routines have to be defined and configured for accessing these heterogeneous systems and loading the correct data into the common DW repository.

Moreover, data from the operational systems are usually specified in different schemas and have to be extracted and transformed to collect them into a common DW repository [Rahm00]. Some of the more common technical tasks that have to be accomplished with these data are as follows. Data coming from different sources have to be joined into a unique target in the DW. Data usually have to be aggregated in order to facilitate the definition of the queries and improve the performance of the DW. Data are usually in different types and formats and they need to be converted into a common format. Data in the operational systems are usually managed by different primary keys representing for example, product or store codes and so on. In DWs we usually use surrogate keys, and therefore, we need an efficient mechanism to assign surrogate keys to the operational data in the DW repository. Furthermore, as data are coming from different sources, we

usually need to check the different primary and foreign keys to assure a high quality of data. Moreover, we also need a high number of filters to verify the right data to be uploaded in the DW and many more problems.

Due to the high difficulty in designing and managing these ETL processes, there has lately been a proliferation in the number of available ETL tools that try to simplify this task [Friedman03, Greenfield03]. During 2001, the ETL market grew to about \$667 million [Agosta02]. Currently, companies expend more than thirty percent out of the total budget for DW projects in expensive ETL tools, but “It’s not unusual for the ETL effort to occupy 60 percent to 80 percent of a data warehouse’s implementation effort” [Strange02]. Nevertheless, it is widely recognized that the design and maintenance of these ETL processes has not yet been solved [Agosta02].

Therefore, we argue that a model and methodology is needed to help the design and maintenance of these ETL processes from the early stages of a DW project; as Kimball states, “Our job as data warehouse designers is to start with existing sources of used data” [Kimball96]. Unfortunately, little effort has been dedicated to propose a conceptual model that allows us to formally define these ETL processes.

In this paper, we present a conceptual model based on the Unified Modeling Language (UML) [OMG01] for the design of ETL processes which deals with the more common technical problems above-presented. As the UML has been widely accepted as the standard for object-oriented analysis and design, we believe that our approach will minimize the efforts of developers in learning new diagrams or methodologies for modeling ETL processes. Furthermore, as we accomplish the conceptual modeling of the target DW schema following our multidimensional modeling approach, also based in the UML [Trujillo01, Luján02a, Luján02b], the conceptual modeling of these ETL processes is totally integrated in a global approach. Therefore, our approach reduces the development time of a DW, facilitates managing data repositories, DW administration, and allows the designer to perform dependency analysis (i.e. to estimate the impact of a change in the data sources in the global DW schema).

The rest of the paper is organized as follows. Section 2 summarizes our DW modeling approach. Section 3 provides an overview of ETL processes and their surrounding data quality problems. Section 4 describes in detail how to accomplish the conceptual modeling of ETL processes using our proposal. Section 5 discusses two general examples that gather all the interesting ETL process problems presented throughout the paper and how they are easily covered and solved applying our approach. Section 6 shows how to use our ETL modeling approach in Rational Rose. Section 7 presents the related work. Finally, Section 8 presents the main conclusions and future work.

2 A global and integrated approach for data warehouse design

The architecture of a DW is usually depicted as various layers of data in which data from one layer is derived from data of the previous layer [Jarke03]. Following this consideration, we consider that the development of a DW can be structured into an integrated model with four different schemas as seen in Figure 1:

- Operational Data Schema (ODS): it defines the structure of the operational data sources.
- Data Warehouse Conceptual Schema (DWCS): it defines the conceptual schema of the DW.
- Data Warehouse Storage Schema (DWSS): it defines the physical storage of the DW.
- Business Model (BM): it defines the different ways or views of accessing the DW from the final users’ point of view. It is composed of different subsets of the DWCS.

From our point of view, two mapping schemas are also needed in order to obtain a global and integrated DW design approach that covers the necessary schemas:

- ETL Process: it defines the mapping between the ODS and the DWCS.
- Exportation Process: it defines the mapping between the DWCS and the DWSS.

Following our integrated approach, we design a DW as follows. We use different UML extensions to model the ODS according to the source: Rational’s UML Profile for Database Design [Naiburg01], a UML profile for data modeling [Ambler02], etc. We have proposed a conceptual model based on the UML to design the DWCS following the multidimensional paradigm [Trujillo01, Luján02a, Luján02b]. Then, from the ODS, we design the ETL processes that will be

responsible for the gathering, transforming and uploading data into the DW. Furthermore, we define different BM as different views from the DWCS for the different type of users that will access the DW. Finally, once the target platform has been chosen (ROLAP, MOLAP, etc.), an exportation process is defined to automatically map the conceptual modeling constructors used in the DWCS into the DWSS, which corresponds to the physical structures of the target platform [Trujillo01]. Therefore, the best advantage of our global approach is that we always use the same notation (based on the UML) for designing the different DW schemas and the corresponding transformations in an integrated manner.

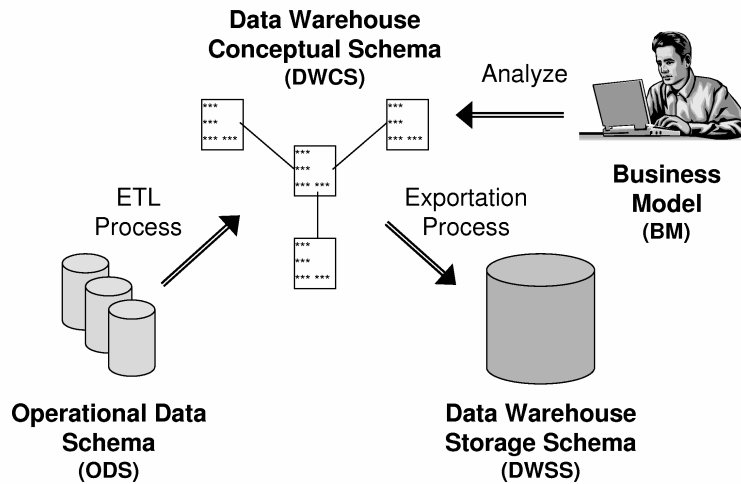


Figure 1. Different schemas for designing data warehouses

3 ETL

In an ETL process, the data extracted from a source system pass through a sequence of transformations before they are loaded into a DW. The repertoire of source systems that contribute data to a DW is likely to vary from standalone spreadsheets to mainframe-based systems many decades old. Complex transformations are usually implemented in procedural programs, either outside the database (in C, Java, Pascal, etc.) or inside the database (by using any 4GL). The design of an ETL process is usually composed of six tasks:

1. Select the sources for extraction: the data sources to be used in the ETL process are defined. It is very common in an ETL process to access different heterogeneous data sources.
2. Transform the sources: once the data have been extracted from the data sources, they can be transformed or new data can be derived. Some of the common tasks of this step are: filtering data, converting codes, performing table lookups, calculating derived values, transforming between different data formats, automatic generation of sequence numbers (surrogate keys), etc.
3. Join the sources: different sources can be joined in order to load together the data in a unique target.
4. Select the target to load: the target (or targets) to be loaded is selected.
5. Map source attributes to target attributes: the attributes (fields) to be extracted from the data sources are mapped to the corresponding target attributes.
6. Load the data: the target is populated with the transformed data from the sources.

The transformation step of the ETL processes can also perform data cleaning tasks, although ETL tools typically have little built-in data cleaning capabilities. Data cleaning deals with detecting and removing errors and inconsistencies from data in order to improve the data quality [Rahm00]. Data quality problems are very significant: it has been estimated that poor quality customer data cost U.S. businesses \$611 billion a year in postage, printing, and staff overhead [Eckerson02].

The manual creation and maintenance of ETL processes increases the cost of development, deployment, running, and maintenance of a DW. That is why the conceptual modeling of ETL processes can be of a crucial help.

4 Modeling ETL processes

In this section we present our ETL modeling proposal that is integrated in our global DW modeling approach presented in Section 2. Our approach allows the designer to decompose a complex ETL process into a set of simple processes. This approach helps the designer to easily design and maintain ETL processes. Moreover, our approach allows the DW designer to tackle the design of ETL processes from different detail levels: (i) the designer can define a general overview of the process and let the database programmer to specify them or, (ii) the designer can provide a detailed description of each one of the attribute transformations.

Based on our personal experience, we have defined a reduced and yet highly powerful set of ETL mechanisms. We have decided to reduce the number of mechanisms in order to reduce the complexity of our proposal. We have summarized these mechanisms in Table 1. We consider that these mechanisms process data in the form of records composed of attributes¹. Therefore, we provide the *Wrapper* mechanism to transform any source into a record based source.

In our approach, an ETL process is composed of UML packages, which allow the user to decompose the design of an ETL process into different logical units. Every particular ETL mechanism is represented by means of a stereotyped class². Moreover, we have defined a different icon for each ETL mechanism (Table 1). This icon can be used in a UML model instead of the standard representation of a class.

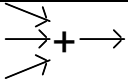
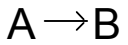

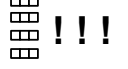
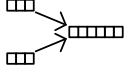
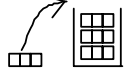

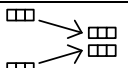
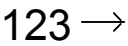
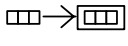
| ETL Mechanism (Stereotype) | Description | Icon |
|----------------------------|---------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| Aggregation | Aggregates data based on some criteria |  |
| Conversion | Changes data type and format or derives new data from existing data |  |
| Filter | Filters and verifies data |  |
| Incorrect | Reroutes incorrect data |  |
| Join | Joins two data sources related to each other with some attributes |  |
| Loader | Loads data into the target of an ETL process |  |
| Log | Logs activity of an ETL mechanism |  |
| Merge | Integrates two or more data sources with compatible attributes |  |
| Surrogate | Generates unique surrogate keys |  |
| Wrapper | Transforms a native data source into a record based data source |  |

Table 1. ETL mechanisms and icons

The ETL mechanisms are related to each other by means of UML dependencies. A dependency in the UML is represented as a dashed line with an arrowhead. The model element at the tail of the arrow (the client) depends on the model element at the arrowhead (the supplier). A dependency states that the implementation or functioning of one or more elements requires the

¹ In our approach, the concept of attribute is similar to the concepts of column, property or field.

² Due to the lack of space, we do not include the formal definition of each stereotype with their corresponding OCL constraints.

presence of one or more other elements. This implies that if the source is somehow modified, the dependents must be probably modified.

A UML note can be attached to every ETL mechanism to (i) explain the functioning of the mechanism and, (ii) define the mappings between source and target attributes of the ETL mechanisms³. These mappings conform to the following syntax: `target_attribute = source_attribute`. To avoid overloading the diagram with long notes, when source and target attributes' names match, the corresponding mappings can be omitted. Furthermore, when some kind of ambiguity may exist (e.g., two attributes with the same name in different sources), the name of the source can be indicated together with name of the attribute (e.g., `Customers.Name` and `Suppliers.Name`).

We do not impose any restriction on the content of these notes, in order to allow the designer the greatest flexibility, but we highly recommend a particular content for each mechanism. The designer can use the notes to define ETL processes at the desired detail level. For example, the description can be general, specified by means of a natural language, or very detailed, specified by means of a programming language.

In the following, we provide a deeper description of each one of the ETL mechanisms presented in Table 1 together with the more appropriated contents for the corresponding attached notes.

4.1 Aggregation

The Aggregation mechanism aggregates data based on some criteria. This mechanism is useful to increase the aggregation level of a data source⁴. The designer can define the grouping criteria and the aggregation functions employed (SUM, AVG, MAX/MIN, COUNT, STDDEV, VARIANCE and so on) in the attached note to this mechanism.

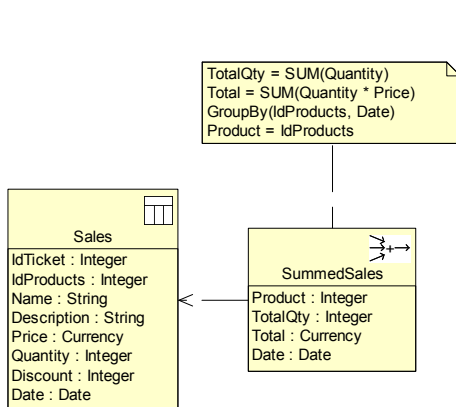


Figure 2. Aggregation example by using standard UML class notation

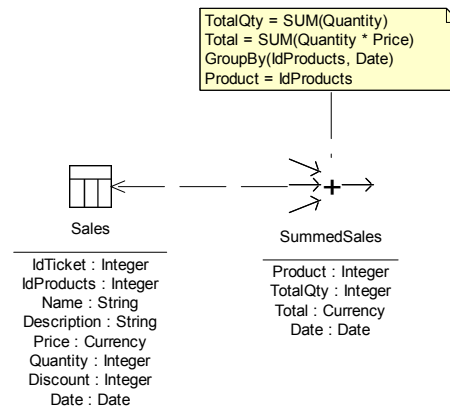


Figure 3. Aggregation example by using the defined stereotype icons

For example, in Figure 2, we have represented a portion of a loading process in a DW⁵ by using standard UML notation in which the stereotype icons are placed in the upper right-hand corner of the corresponding class. It may also be observed that the icon used in the `Sales` class corresponds to the Table stereotype icon defined in the UML profile for database design [Naiburg01]. As the grain of `Sales` is ticket line, we need the daily total sales in the DW. Therefore, `Sales` are grouped and summed up by product and date in `SummedSales`. We have decided to specify these aggregation tasks in the corresponding attached note. Figure 3 represents the same

³ The connection between a note and the element it applies to is shown by a dashed line without an arrowhead as this is not a dependency [OMG01].

⁴ Partial summarization of data under different criteria is a very common technique used in DWs to facilitate complex analysis. Summarization helps to reduce the size of the resulting DW and increase the query performance [Kimball96].

⁵ From now on, partial examples are used to describe each ETL mechanism.

ETL process using our ETL icons. From now on, we will use this representation throughout the paper.

4.2 Conversion

The Conversion mechanism is used to change data types and formats or to calculate and derive new data from existing data. The conversions are defined in the attached note by means of conversion functions applied to source attributes. The syntax of these conversions is `target_attribute = Function(source_attributes)`, where Function is any kind of function that can be applied to source attributes.

Based on our experience, we have defined conversion functions for the most common situations. However, this is not a closed set as the designer can define their own user-defined functions for more particular and complex situations:

- Data type conversions: convert data from a data type into another data type. For example: `Price = StringToCurrency(Price)`.
- Arithmetic conversions: perform arithmetic operations (add, multiply, etc.) with data. For example: `Total = Quantity * Price`.
- Format conversions: convert a value (currency, date, length, etc.) from one format into another one. For example: `Price = DollarToEuro(Price)`.
- String conversions: transform strings (upper and lower-case, concatenate, replace, substring, etc.). For example: `Name = Concatenate(FirstName, " ", Surname)`.
- Split conversions: break a value into different elements. For example, the following expression breaks a name ("John Doe") into first name ("John") and surname ("Doe"): `FName = FirstName(Name); SName = Surname(Name)`.
- Standardization conversions: standardize attributes to contain identical values for equivalent data elements. We can use a set of rules or look-up tables to search for valid values. For example, the following expression substitutes "Jan." or "1" with "January": `Month = StdMonth(Month)`.
- Value generator: generates a constant value or a variable value from a function. The new value does not depend on any source attribute. For example: `Date = Timestamp()`.
- Default value: when a value is missing (null, empty string, etc.), it is possible to define a default value. The syntax of this option is `target_attribute ?= value`. For example: `Type ?= "unknown"`.

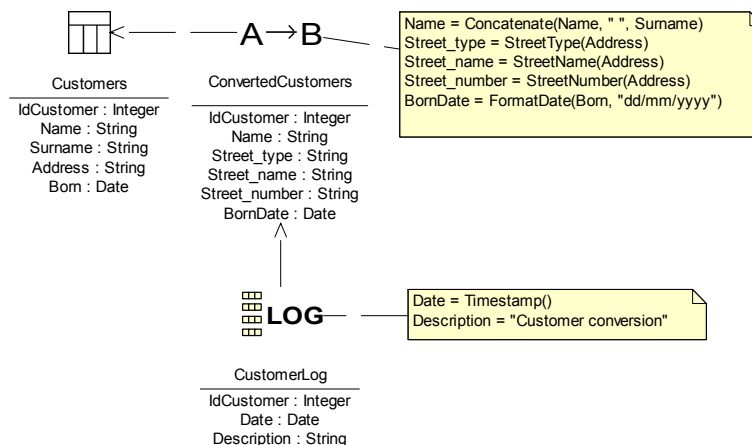


Figure 4. An example of Conversion and Log processes

Figure 4 presents an example in which different conversions are applied through the `ConvertedCustomers` stereotype. As it can be easily seen from the attached note, `Name` and `Surname` are concatenated; `Address` is split into street type, name and number; and `Born` is converted using a date format. Furthermore, all the activity is audited by `CustomerLog` (see next section).

4.3 Log

The `Log` mechanism can be connected to any ETL mechanism as it controls the activity of another ETL mechanism. This mechanism is useful to audit and produce operational reports for each transformation. The designer can add any kind of additional information in the note attached to this mechanism.

For example, in Figure 4, the activity of a Conversion mechanism is controlled by the Log mechanism called `CustomerLog`.

4.4 Filter

The `Filter` mechanism filters unwanted data and verifies the correctness of data based on constraints. In this way, this mechanism allows the designer to load only the required data or the data that meet an established quality level in the DW. The verification process is defined in the attached note by means of a set of Boolean expressions that must be satisfied. The Boolean expressions can be expressed by means of a set of rules or by means of look-up tables that contain the correct data. Some common tasks for this mechanism are checks for null values, missing values, values out of range, and so on. The data that do not satisfy the verification can be rerouted to an `Incorrect` mechanism (see Section 4.7).

For example, in Figure 5, `Customers` are filtered and only those that were born before 1950 are accepted for a subsequent processing.

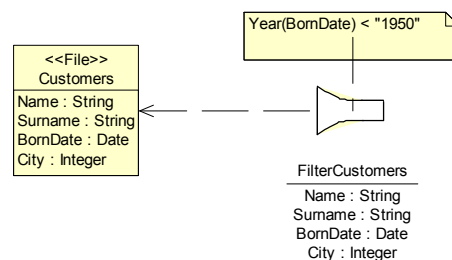


Figure 5. An example of Filter process

4.5 Join

The `Join` mechanism is used to join two data sources related to each other with some attributes (defined by means of a restrictive condition). The designer can define the following information in the attached note:

- The type of join: `Join(conditional_expression)`, where `Join` can be `InnerJoin` (includes only the records that match the conditional expression), `LeftJoin` (includes all of the records from the first (left) of the two data sources, even if there are no matching values for records in the second (right) data source), `RightJoin` (includes all of the records from the second (right) of the two data sources, even if there are no matching values for records in the first (left) data source), and `FullJoin` (includes all of the records from both data sources, even if there are no matching values for records in the other data source).
- If `LeftJoin`, `RightJoin` or `FullJoin` are used, then the designer can define the values that substitute the non-existing values. The syntax of this option is `target_attribute ?= value`.

In Figure 6, we have represented an ETL process that joins three data sources. Due to the fact that `Join` can only be applied to two sources, and in this particular case we are dealing with three sources, two `Join` mechanisms are needed. In the `CitiesStates` join, a `LeftJoin` is performed to join cities' names and states' names. When it is not possible to join a city with a state (because

Cities.State is missing or is incorrect), the corresponding state name is replaced by “unknown”. Finally, in the CompleteCustomers join, the result of the previous join is joined with Customers.

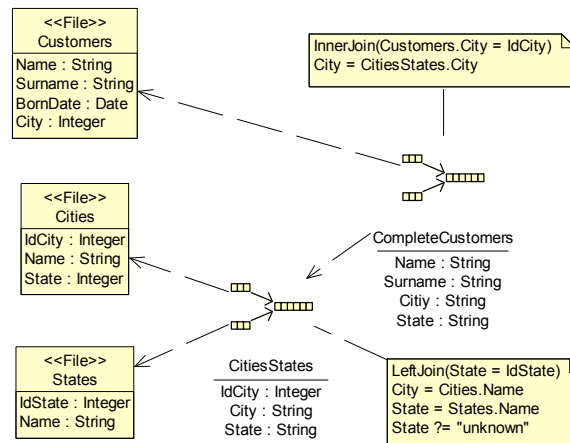


Figure 6. An example of Join process

4.6 Loader

The **Loader** mechanism loads data into the target of an ETL process such as a dimension or a fact in a DW. Every ETL process should have at least one **Loader** mechanism. Two operation modes are supported in the **Loader**:

- Free loading: the **Loader** mechanism does not verify any constraint as the target applies its own constraints to the new data.
- Constrained loading: the designer can apply primary and foreign key constraints during the loading process. Moreover, the designer can also define how to manage existing data in the target. The following information can be attached to the **Loader** mechanism:
 - **PK(source_attributes)**: defines the attributes of the source data that define a unique record in the target. This information is used for constraining the loading process and it is also used for detecting the old data that should be updated.
 - **FK(target_attributes; source_attributes)**: defines the attributes of the source data that should previously exist in a target.
 - **Append**: the target need not be empty before loading the new data; new data are loaded and old data are updated.
 - **Delete**: the target need be empty before loading the data.
 - **Insert**: only new data are loaded in the target; old data are not loaded again or updated, although they have changed,
 - **Update**: only existing data in the target are updated with the corresponding data, but new data are not loaded.

Append, Delete, Insert, and Update are mutually exclusive: only one of them can be used in a **Loader** mechanism.

For example, in Figure 7, **CustomerLoader** updates existing data in **CustomersDim** with data coming from **Customers**. Furthermore, due to the high probability of errors when making the loading process, those records that do not satisfy the constraints are rerouted to **DiscardedCustomers**. **CustomersDim** represents a dimension in our DW schema; the icon corresponds to the Dimension stereotype defined in our multidimensional modeling approach [Luján02a].

4.7 Incorrect

The **Incorrect** mechanism is used to reroute bad or discarded records and exceptions to a separate target. In this way, the DW designer can track different errors. This mechanism can only be used with the **Filter, Loader, and Wrapper**, because these mechanisms constrain the data they

process. The designer can add additional information in the note attached to this mechanism, such as a description of the error or a timestamp of the event.

For example, in Figure 7, the records that do not satisfy the constraints of **CustomerLoader** (primary key constraint on **IdCustomer** and only update existing data in the target) are rerouted to **DiscardedCustomers**, which collects the erroneous data and adds the **Date** attribute that is a timestamp of the event.

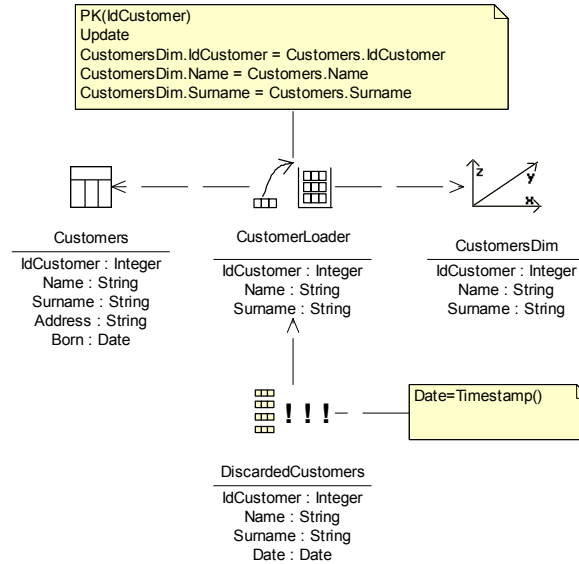


Figure 7. An example of Loader and Incorrect processes

4.8 Merge

The **Merge** mechanism integrates two or more data sources with compatible attributes. Two data sources are compatible as long as both of them contain a subset of the attributes defined in the target: the attributes used in the integration must have the same names in all the data sources. If the attributes do not have the same names, the **Conversion** mechanism can be previously applied in order to standardize them. The attached note to this mechanism is used to define the mapping between the data sources and the target.

For example, in Figure 8, **MergedCustomers** is used to integrate data coming from a file and from a database table. Firstly, **WrappedCustomers** is used to transform a file into a record based source (see next section). Then, **ConvertedCustomers** changes the names of the attributes (**CName** and **CSurname**) and adds a new attribute (**BornDate**) with a default value.

4.9 Wrapper

The **Wrapper** mechanism allows us to define the required transformation from a native data source into a record based data source. Different native sources are possible in an ETL process: fixed and variable format sources, COBOL files (line sequential, record sequential, relative files, and indexed files), multiline sources, XML documents, and so on. The needed code to implement the **Wrapper** is not relevant as we are at the conceptual level, although the designer can define in the attached note all the information that considers relevant to help the programmer at the implementation phase.

In Figure 8, **WrappedCustomers** is used to transform data from a fixed format file. Some information about the format of the file is included in the attached note.

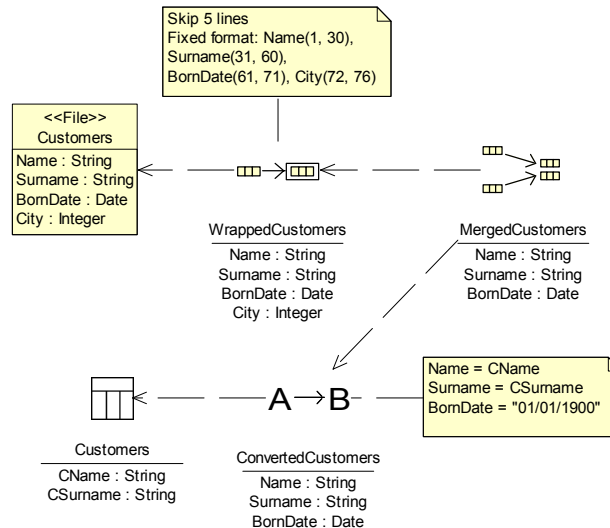


Figure 8. An example of Merge and Wrapper processes

4.10 Surrogate

The Surrogate mechanism generates unique surrogate keys. Surrogate key assignment is a common process in DWs, employed in order to replace the original keys of the data sources with a uniform key. The attached note to this mechanism is used to define the source attributes used to define the surrogate key. Surrogate keys could have been defined in the Conversion mechanism, however, due to the importance that surrogate keys represent in DWs, we have decided to define an own mechanism.

For example, in Figure 9, SurrogatedCities adds a surrogate key (IdCity) based on the attributes Name, State, and Country before loading the data into the DW.

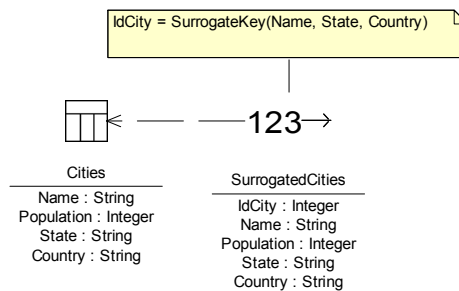


Figure 9. An example of Surrogate process

5 ETL examples

In previous sections, we have shown short and specific examples to explain how we modeled the particular ETL mechanisms. In this section, we present two interesting ETL examples. The first one is a complete example in which we show how to easily model different ETL mechanisms keeping a high grade of simplicity and yet a very powerful ETL process model. In the second example, we show how to merge two different data sources and deal with a multi-target loading. These examples show the expressiveness power of our ETL modeling approach.

5.1 A complete ETL process

In Figure 10, we have represented a loading process into a DW. The grain level of Sales (the data source) is ticket line, but we need the daily total sales in the DW (Sales fact table at the

right hand side of Figure 10). Therefore, Sales are grouped and summed up by product and date in SummedSales. Then, SurrogatedSales adds a surrogate key (Time) based on the Date attribute before loading the data into the DW. This surrogate key is used in the DW to establish a relation between Sales fact and Time dimension. Finally, SalesLoader loads summarized sales into Sales fact table, ProductsLoader loads the product list into Products dimension, and TimeLoader loads time data into Time dimension.

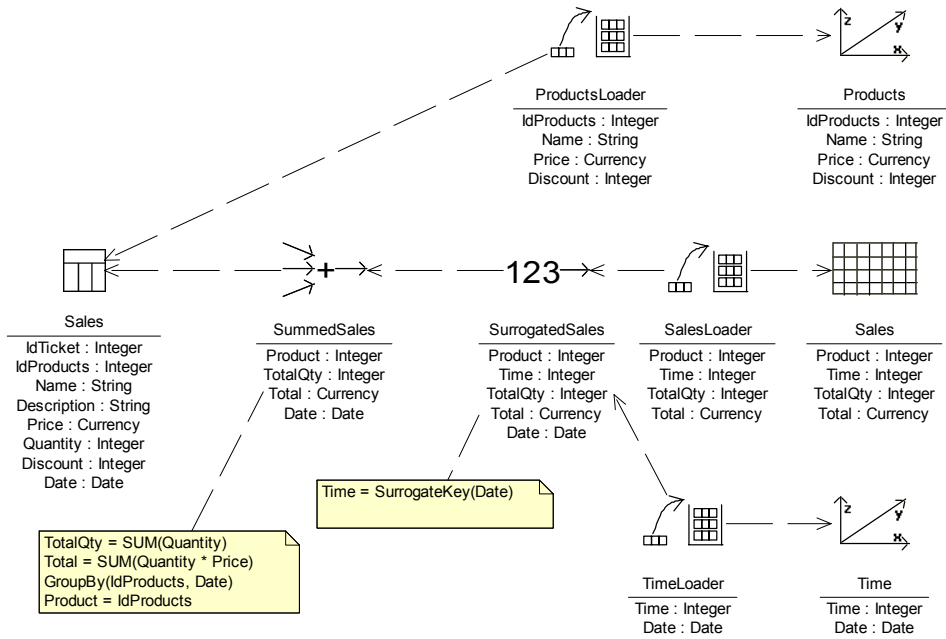


Figure 10. An example of a complete ETL process

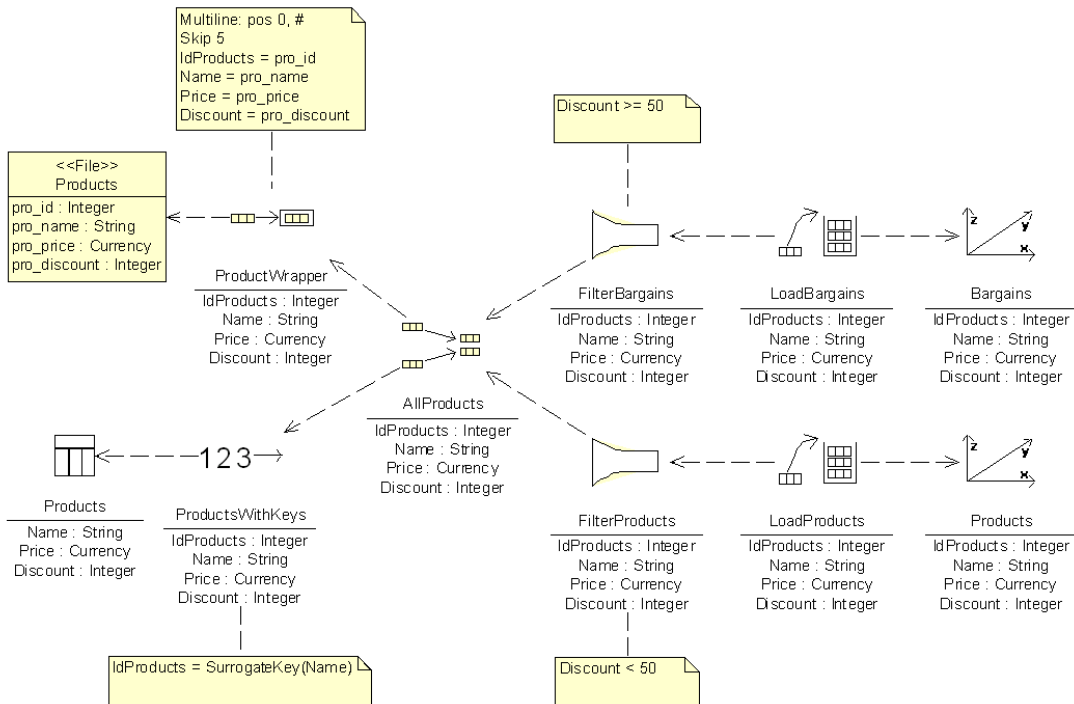


Figure 11. Merging two different data sources and accomplishing multi-target loading

5.2 Merging two different data sources and multi-target loading

In Figure 11, we deal with the problem of merging two incompatible data sources as they have different attribute number and the common attributes have different names. Regarding the first data source, *ProductWrapper* is used to transform a multiline file into a record based data source and rename the attributes' names. With respect the second one, *ProductsWithKeys* (Surrogate) generates a surrogate key from the *Name* attribute. Then, *AllProducts* (Merge) is used to integrate the two sources. Finally, all the products have to be loaded into two targets depending on the discount of each product: the products with a discount lower than 50 percent have to be loaded into the *Products* dimension, whereas the products with a discount equal or greater than 50 percent have to be loaded into the *Bargains* dimension. *FilterProducts* and *FilterBargains* are used to filter the products.

6 ETL modeling in Rational Rose

Rational Rose (RR) is one of the most well-known visual modeling tools. As RR supports the UML, it is becoming the common modeling tool for OO modeling. RR is extensible by means of add-ins, which allows the user to group together customizations and automation of several RR features through the Rose Extensibility Interface (REI) [Rational01] into one component. An add-in allows the user to customize main menu items, data types, stereotypes, etc.

We have developed a RR add-in that allows us to use the stereotypes we have defined for modeling ETL processes. This add-in complements our previous add-in [Luján02b] that allows the DW designer to accomplish the conceptual modeling of a DW in RR.

In Figure 12, we can observe the same ETL process presented in Section 5.2 but modeled in RR. On the left hand side, a hierarchy list of icons that represent different modeling elements is displayed: at the top, the DWCS is shown with the multidimensional modeling elements (*Bargains*, *CustomersDim*, *Sales*, etc.); in the middle, the ODS (*Source1*, *Source2*, etc.) are represented; at the bottom, different UML packages (*Aggregation*, *Conversion*, *Customers*, etc.) that contain ETL processes are shown. Furthermore, the vertical toolbar in the middle of Figure 12 contains ten new buttons that correspond to each one of our ETL stereotypes. Finally, on the right hand side, the ETL process is shown.

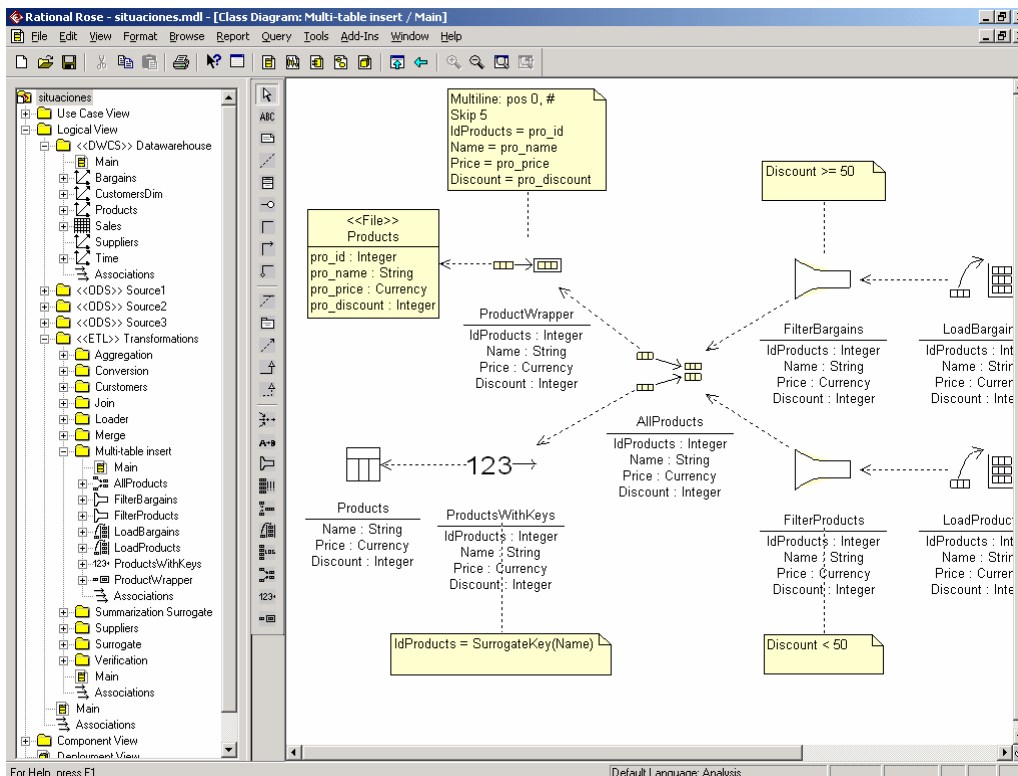


Figure 12. Modeling ETL processes using our approach in Rational Rose

7 Related work

Little effort has been dedicated to propose a conceptual model that allows the DW designer to formally define ETL processes. Various approaches for the conceptual design of DWs have been proposed in the last few years [Golfarelli98, Sapia98, Tryfona99, Husemann00, Abelló02]. However, none of them has addressed the modeling of ETL processes.

To the best of our knowledge, the best advance in this research line has been accomplished by the Knowledge and Database Systems Laboratory from NTUA [KDBS03]. In particular, they have proposed a conceptual model that provides its own graphical notation that allows the designer to formally define most of the usual technical problems regarding ETL processes [Vassiliadis02]. Furthermore, this approach is accompanied by an ETL tool called ARKTOS as an easy framework for the design and maintenance of these ETL processes [Vassiliadis01].

The main differences, and advantages, of our approach in contrast to Vassiliadis' are the use of a standard modeling language (UML), the use of a grouping mechanism (UML packages) that facilitates the creation and maintenance of complex ETL processes, and the integration of the design of ETL processes in a global and integrated approach for DW design. Moreover, Vassiliadis *et al.* do not employ standard UML notation because they need to treat attributes as "first class citizens" of their model, what we believe complicates the resulting ETL models: a DW usually contains hundreds of attributes, and therefore, an ETL model can become exceedingly complex if every attribute is individually represented as a model element.

8 Conclusions

In this paper, we have presented the modeling of ETL processes as part of an integrated and global approach for DW design. Thanks to the use of the UML, we can seamlessly model different aspects of a DW architecture such as operational data sources, conceptual schema and ETL processes in an integrated manner. In this way, it is very easy to detect inconsistencies between the different schemas of a DW architecture and it helps the designer to estimate the feasibility of the development.

Our approach for modeling ETL processes defines a set of UML stereotypes that represents the most common ETL tasks such as the integration of different data sources, the transformation between source and target attributes, the generation of surrogate keys, and so on. Furthermore, thanks to the use of the UML package mechanism, large ETL processes can be easily modeled in different ETL packages obtaining a very simple but yet powerful approach. Thanks to its simplicity, our approach facilitates the design and subsequent maintenance of ETL processes at any modeling phase. Finally, we have implemented our approach in Rational Rose 2002 through the Rose Extensibility Interface (REI) [Rational01].

Regarding future work, we are currently working on the automatic transformation of the ETL models into a target platform. For this task, we need specific information about the target platform to be able to transform the content of the attached notes to the defined stereotypes into valid code. In this way, our approach will help the programmer with the required code to solve the most common ETL problems.

References

- [Abelló02] Abelló, A., Samos, J., and Saltor, F. "YAM2 (Yet Another Multidimensional Model): An Extension of UML". In *International Database Engineering & Applications Symposium (IDEAS 2002)*, p. 172-181, Edmonton (Canada), 2002.
- [Agosta02] Agosta, L. "Market Overview Update: ETL". Giga Information Group, Technical Report RPA-032002-00021, March 2002.
- [Ambler02] Ambler, S. "A UML Profile for Data Modeling". Internet: <http://www.agiledata.org/essays/umlDataModelingProfile.html>, 2002.
- [Eckerson02] Eckerson, W. W. "Data Quality and the Bottom Line". The Data Warehousing Institute, Technical Report, 2002.
- [Friedman03] Friedman, T. "ETL Magic Quadrant Update: Market Pressure Increases". Gartner, Technical Report M-19-1108, January 2003.

- [Golfarelli98] Golfarelli, M., and Rizzi, S. "A Methodological Framework for Data Warehouse Design". In *ACM 1st International Workshop on Data warehousing and OLAP (DOLAP 1998)*, p. 3-9, Washington D.C. (USA), 1998.
- [Greenfield03] Greenfield, L. "Data Extraction, Transforming, Loading (ETL) Tools". The Data Warehousing Information Center. Internet: <http://www.dwinfocenter.org/clean.html>, 2003.
- [Husemann00] Husemann, B., Lechtenborger, J., and Vossen, G. "Conceptual Data Warehouse Design". In *2nd International Workshop on Design and Management of Data Warehouses (DMDW 2000)*, p. 3-9, Stockholm (Sweden), 2000.
- [Inmon92] Inmon, W. H. *Building the Data Warehouse*. QED Press/John Wiley, 1992. (Last edition: 3rd edition, John Wiley & Sons, 2002)
- [Jarke03] Jarke, M., Lenzerini, M., Vassiliou, Y., and Vassiliadis, P. *Fundamentals of Data Warehouses*. 2nd edition, Springer-Verlag, 2003.
- [KDBS03] Knowledge and Database Systems Laboratory from National Technical University of Athens (Greece). Internet: <http://www.dblab.ntua.gr/>, 2003.
- [Kimball96] Kimball, R. *The Data Warehouse Toolkit*. John Wiley & Sons, 1996. (Last edition: 2nd edition, John Wiley & Sons, 2002)
- [Luján02a] Luján-Mora, S., Trujillo, J., and Song, I. "Extending UML for Multidimensional Modeling". In *5th International Conference on the Unified Modeling Language (UML 2002)*, p. 290-304: Lecture Notes in Computer Science 2460, Dresden (Germany), September 30 - October 4 2002.
- [Luján02b] Luján-Mora, S., Trujillo, J., and Song, I. "Multidimensional Modeling with UML Package Diagrams". In *21st International Conference on Conceptual Modeling (ER 2002)*, p. 199-213: Lecture Notes in Computer Science 2503, Tampere (Finland), October 7-11 2002.
- [Naiburg01] Naiburg, E., and Maksimchuk, R. A. *UML for Database Design*. Addison-Wesley, 2001.
- [OMG01] Object Management Group (OMG). "Unified Modeling Language Specification 1.4". Object Management Group (OMG). Internet: <http://www.omg.org/cgi-bin/doc?formal/01-09-67>, 2001.
- [Rahm00] Rahm, E., and Do, H. "Data Cleaning: Problems and Current Approaches". *IEEE Bulletin of the Technical Committee on Data Engineering*, Vol 23 No. 4, p. 3-13, December 2000.
- [Rational01] Rational Software Corporation. "Using the Rose Extensibility Interface". Rational Software Corporation, Technical Report, 2001.
- [Sapia98] Sapia, C., Blaschka, M., Höfling, G., and Dinter, B. "Extending the E/R Model for the Multidimensional Paradigm". In *1st International Workshop on Data Warehouse and Data Mining (DWDM 1998)*, p. 105-116: Lecture Notes in Computer Science 1552, Singapore, 1998.
- [SQL02] SQL Power Group. "How do I ensure the success of my DW?" SQL Power Group, 2002. Internet: http://www.sqlpower.ca/page/dw_best_practices.
- [Strange02] Strange, K. "ETL Was the Key to this Data Warehouse's Success". Gartner, Technical Report CS-15-3143, March 2002.
- [Trujillo01] Trujillo, Juan, Palomar, Manuel, Gómez, Jaime, and Song, Il-Yeol. "Designing Data Warehouses with OO Conceptual Models". *IEEE Computer, special issue on Data Warehouses*, No. 34, p. 66-75, 2001.
- [Tryfona99] Tryfona, N., Busborg, F., and Christiansen, J. "starER: A Conceptual Model for Data Warehouse Design". In *ACM 2nd International Workshop on Data warehousing and OLAP (DOLAP 1999)*, Kansas City, Missouri, USA, 1999.
- [Vassiliadis01] Vassiliadis, P., Vagena, Z., Skiadopoulos, S., Karayannidis, N., and Sellis, T. "ARKTOS: towards the modeling, design, control and execution of ETL processes". *Information Systems*, No. 26, p. 537-561, 2001.
- [Vassiliadis02] Vassiliadis, P., Simitsis, A., and Skiadopoulos, S. "Conceptual Modeling for ETL Processes". In *5th ACM International Workshop on Data Warehousing and OLAP (DOLAP 2002)*, p. 14-21, McLean (USA), November 8 2002.