Modeling and managing ETL processes

Alkis Simitsis

National Technical University of Athens,
Dept. of Electrical and Computer Eng., Computer Science Division,
Iroon Polytechniou 9, Zografou 15773,
Athens, Greece
asimi@dbnet.ece.ntua.gr

Abstract

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 design, development and deployment of ETL processes, which is currently, performed in an ad-hoc, in house fashion, needs modeling, design and methodological foundations. Unfortunately, the research community has a lot of work to do to confront this shortcoming. Our research explores a coherent framework for the conceptual, the logical, and the physical design of ETL processes. We delve into the modeling of ETL activities and provide a conceptual and a logical abstraction for the representation of these processes. Moreover, we focus on the optimization of the ETL processes, in order to minimize the execution time of an ETL process.

1. Introduction

In order to facilitate and manage the data warehouse operational processes, specialized tools are already available in the market, under the general title *Extraction-Transformation-Loading* (ETL) tools. To give a general idea of the functionality of these tools we mention their most prominent tasks, which include: (a) the identification of relevant information at the source side; (b) the extraction of this information; (c) the customization and integration of the information coming from multiple sources into a common format; (d) the cleaning of the resulting data set, on the basis of database and business rules, and (e) the propagation of the data to the data warehouse and/or data marts.

To probe into the aforementioned issues, we have to clarify how the ETL processes fit in the data warehouse lifecycle. The lifecycle of a data warehouse begins with an initial *Reverse Engineering* and *Requirements Collection* phase where the data sources are analyzed in order to comprehend their structure and contents. At the same time, any requirements from the part of the users (normally a few power users) are also collected. The deliverable of this stage is a conceptual model for the data stores and the activities. In a second stage, namely the *Logical Design* of the warehouse, the logical schema for the warehouse and the activities is constructed. Third, the logical design of the schema and processes is refined to the choice of specific physical structures in the warehouse (e.g., indexes) and environment-specific execution parameters for the operational processes. This stage is called *Tuning* and its deliverable, the *physical model* of the environment. In a fourth stage, *Software Construction*, the software is constructed, tested, evaluated and a first version of the warehouse is deployed. This process is guided through specific software metrics. Then, the cycle starts again, since data sources, user requirements and the data warehouse state are under continuous evolution. An extra feature that comes into the scene after the deployment of the warehouse is the *Administration* task, which also needs specific metrics for the maintenance and monitoring of the data warehouse.

In our research, we provide a complete framework for the modeling of ETL process and we focus on the modeling and the optimization of ETL scenarios. The uttermost goal of our research is to facilitate, manage and optimize the design and implementation of the ETL processes both during the initial design and deployment stage and during the continuous evolution of the data warehouse. Briefly, our main contributions are:

- The provision of a novel *conceptual* and a novel *logical* model for the representation of ETL processes with two characteristics: genericity and customization.
- The presentation of a palette of several templates, representing frequently used ETL activities along with their semantics and their interconnection. In this way, construction of ETL scenarios, as a flow of these activities, is facilitated.
- The introduction of a set of algorithms to find the optimal ETL scenario according to a cost model.

- To complement the aforementioned issues, we have prototypically implemented a graphical tool, named ARKTOS II, with the goal of facilitating the design and the (re-)use of ETL scenarios, based on our model.

This paper is organized as follows. In section 2, we present related work. In section 3, we describe the conceptual model for ETL processes. In section 4, we focus on the logical design of ETL processes. In section 5, we delve into the optimization of ETL processes. Finally, in section 6, we conclude our results with a prospect to the future.

2. Related Work

In this section we discuss the state of art and practice for research efforts, commercial tools and standards in the field of ETL tools, along with any related technologies. For lack of space, we refer the interested reader to [VaSS02, VaSS02a] for an extended discussion of the issues that we briefly present in this section.

Conceptual models for data warehouses. The front end of the data warehouse has monopolized the research on the conceptual part of data warehouse modeling. Research efforts can be grouped in the following trends: (a) *dimensional modeling* [KKRT98]; (b) *(extensions of) standard E/R modeling* [SBHD98, CDL+99, HuLV00, TrBC99] (c) *UML modeling* [TrPG00] and (d) *sui-generis models* [GoMR98] without a clear winner. We must stress that our model is orthogonal to these efforts.

Conceptual models for ETL. There are few attempts around the specific problem of this work, although we are not aware of any other approach that concretely deals with the specifics of ETL activities in a conceptual setting [BoFM99, CDL+99]. In terms of industrial approaches, the model that stems from [KRRT98] would be an informal documentation of the overall ETL process.

Related work on ETL logical and physical aspects. There is a variety of ETL tools in the market; we mention a recent review [Gart03] and several commercial tools [Arde02, Data02, ETI02, IBM02, Info03, Micr02, Orac02]. There also exist research efforts including [Sara00, VVS+01, LWGG00]. Research prototypes include the AJAX data cleaning tool [GFSS00] and the Potter's Wheel system [RaHe00]. These two prototypes are based on algebras, which we find specifically tailored for the case of homogenizing web data.

In the context of this research, we develop the design tool ARKTOS II. Compared to the design capabilities of the aforementioned approaches, our technique contributes (a) by offering an extensible framework through a uniform extensibility mechanism, and (b) by providing formal foundations to allow the reasoning over the constructed ETL scenarios. It is also interesting to note that results in the field of data cleaning [Sara00] could be treated as special-purpose templates of ARKTOS II. In general, techniques provided by research in the field can be plugged-in orthogonally to our template library.

3. Conceptual Model

In this section, we focus on the conceptual part of the definition of the ETL process. For a detailed presentation of our conceptual model and formal foundations for the representation of ETL processes, we refer the interested reader to [VaSS02a, SiVa03].

More specifically, we are dealing with the earliest stages of the data warehouse design. During this period, the data warehouse designer is concerned with two tasks which are practically executed in parallel: (a) *the collection of requirements* from the part of the users; (b) *the analysis of the structure and content of the existing data sources* and their *intentional mapping to the common data warehouse model*. Related literature [KRRT98, Vass00] and personal experience suggest that the design of an ETL process aims towards the production of a crucial deliverable: *the mapping of the attributes of the data sources to the attributes of the data warehouse tables*. The production of this deliverable involves several interviews that result in the revision and redefinition of original assumptions and mappings; thus it is imperative that a simple conceptual model is employed in order to facilitate the smooth redefinition and revision efforts and to serve as the means of communication with the rest of the involved parties.

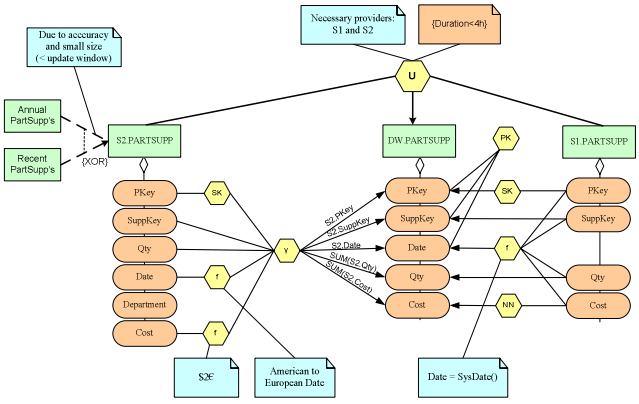

Figure 1: Conceptual design of our example

To motivate the discussion we introduce an example involving two source databases S_1 and S_2 as well as a central data warehouse DW. The scenario involves the propagation of data from the concept PARTSUPP(PKEY, SUPPKEY, QTY, COST) of source S_1 as well as from the concept PARTSUPP(PKEY, DEPARTMENT, SUPPKEY, QTY, DATE, COST) of source S_2 to the data warehouse. In the data warehouse, DW.PARTSUPP(PKEY, SUPPKEY, DATE, QTY, COST) 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. Throughout all the paper, we will clarify the introduced concepts through their application to this example.

In Figure 1, we depict the full fledged diagram of the example, in terms of our conceptual model.

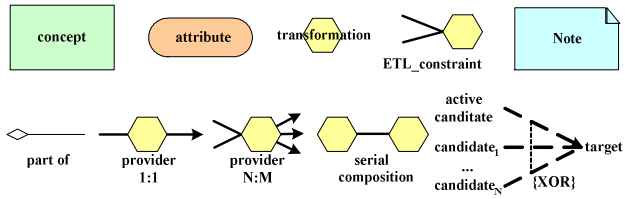

Figure 2: Notation for the conceptual model

In Figure 2, we graphically depict the different entities of the proposed model. We do not employ standard UML notation for concepts and attributes, for the simple reason that we need to treat attributes as first class citizens of our model. The main entities of our model are the following.

Attributes. A granular module of information. The role of attributes is the same as in the standard ER/dimensional models (e.g., PKEY, DATE, SUPPKEY, etc.).

Concepts. A concept represents an entity in the source databases or in the data warehouse (e.g., S1.PARTSUPP, S2.PARTSUPP, DW.PARTSUPP).

Transformations. Transformations are abstractions that represent parts, or full modules of code, executing a single task and include two large categories: (a) filtering or data cleaning operations; and (b) transformation operations, during which the schema of the incoming data is transformed (surrogate key assignment transformation (SK), function application (f), not null (NN) check, etc.).

ETL Constraints. They are used in several occasions when the designer wants to express the fact that the data of a certain concept fulfill several requirements (e.g., to impose a PK 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 [BoJR98] (e.g., a runtime constraint specifying that the overall execution time for the loading of DW.PARTSUPP cannot take longer than 4 hours).

Part-of Relationships. We bring up part-of relationships, not to redefine UML part-of relationships, but rather to emphasize the fact that a concept is composed of a set of attributes, since we need attributes as first class citizens in the inter-attribute mappings.

Candidate relationships. A set of candidate relationships captures the fact that a certain data warehouse concept (S_1) can be populated by more than one candidate source concepts (AnnualPartSupp, RecentPartSupp).

Active candidate relationships. An active candidate relationship denotes the fact that out of a set of candidates, a certain one (RecentPartSupp) has been selected for the population of the target concept.

Provider relationships. A 1:1 (N:M) provider relationship maps a (set of) input attribute(s) to a (set of) output attribute(s) through a relevant transformation.

Transformation Serial Composition. It is used when we need to combine several transformations in a single provider relationship (e.g., the combination of SK and γ).

The proposed model is constructed in a customizable and extensible manner, so that the designer can enrich it with his own re-occurring patterns for ETL activities, while, at the same time, we also offer a 'palette' of a set of frequently used ETL activities, like the assignment of surrogate keys, the check for null values, etc..

4. Logical Model

The ETL conceptual model is constructed in the early stages of the data warehouse project during which, the time constraints of the project require a quick documentation of the involved data stores and their relationships, rather than an in-depth description of a composite workflow. In this section we present a logical model for the activities of an ETL environment that concentrates on the flow of data from the sources towards the data warehouse through the composition of activities and data stores. Moreover, we provide a technical solution for the implementation of the overall process. For lack of space we present a condensed version of the model; the full-blown version and the formal representation of the model can be found in [VSGT03, VaSS02].

The full layout of an ETL scenario, involving activities, recordsets and functions can be deployed along a graph in an execution sequence that can be linearly serialized. We call this graph, the *Architecture Graph*. The design of this graph can be performed by the graphical tool ARKTOS II, provided that some construction rules are obeyed. The basic entities of our model are the following.

Attributes and part-of relationships. The first thing to incorporate in the architecture graph are the structured entities (activities and recordsets) along with all the attributes of their schemata. Then, we incorporate the functions along with their respective parameters and the part-of relationships among the former and the latter.

Data types and instance-of relationships. Next, we incorporate data and function types.

Parameters and regulator relationships. Afterwards, it is time to establish the regulator relationships of the scenario. In this case, we link the parameters of the activities to the terms (attributes or constants) that populate them.

Provider relationships. The last thing to add in the architecture graph is the provider relationships that capture the data flow from the sources towards the target recordsets in the data warehouse.

Derived provider relationships. There are certain output attributes that are computed through the composition of input attributes and parameters. A *derived provider relationship* is another form of provider relationship that models the situation where the activity computes a new attribute in its output. In this case, the produced output depends on all the attributes that populate the parameters of the activity, resulting in the definition of the corresponding derived relationship.

The graphical notation for the Architecture Graph is depicted in Figure 3.

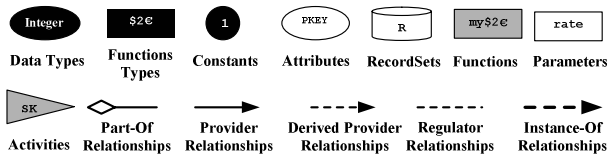

Figure 3: Notation for the Architecture Graph

One of the major contributions that our graph-modeling approach offers is the ability to treat the scenario as the skeleton of the overall environment. If we treat the problem from its software engineering perspective, the interesting problem is how to design the scenario in order to achieve effectiveness, efficiency and tolerance of the impacts of evolution. Therefore, we assign simple *importance metrics* to the nodes of the graph, in order to measure how crucial their existence is for the successful execution of the scenario. We measure the importance and vulnerability of the nodes of the graph through specific *importance metrics*, namely *dependence* and *responsibility*. Dependence stands for the degree to which an entity is bound to other entities that provide it with data and responsibility measures the degree up to which other nodes of the graph depend on the node under consideration. Other interesting usages of the aforementioned measures include: (a) detection of inconsistencies for attribute population; (b) detection of important data stores; (c) detection of useless (source) attributes; and (d) observations after zooming out the whole scenario. Moreover, we provide several simple algorithms (e.g., *Zoom-In*, *Zoom-Out*, *Major-Flow*, etc.) that reduce the complexity of the graph.

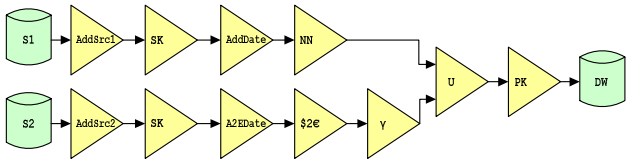

Figure 4: Logical design of our example

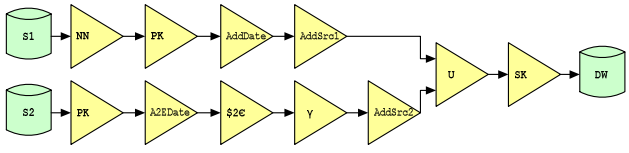

Figure 5: Optimal design of our example

In Figure 4, we depict a simplified (on account of the limited space) diagram of our motivating example, in terms of our logical model.

Similarly to the templates of the conceptual model, we offer a palette of templates to the logical model too [VSGT03].

5. Optimizing ETL scenarios

In this section, we focus on the optimization of ETL scenarios. We consider each ETL scenario as a state and we fabricate the state space. Thus, we model the ETL processes optimization problem as a state search problem. We are working on algorithms to find the optimal scenario according to a cost model. Moreover, we present our experimental results in order to validate our methods.

Intuitively, a state is a set of activities, deployed along a graph in an execution sequence that can be linearly serialized. Formally, a state consists of: *Id*, a unique identifier for the state; *Activities*, a finite list of activities (note that by employing a list instead of e.g., a set of activities, we impose a total ordering on the state); *Attributes*, a finite set of attributes; *Recordsets*, a finite set of recordsets; *Provider* relationships, a finite list of provider relationships among attributes of activities and recordsets of the state; *Part-Of* relationships, these relationships involve attributes and relate them to the respective activity or recordset to which they belong.

Being a graph, a state comprises nodes and edges. The involved recordsets and activities of an ETL process along with their respective attributes constitute the nodes of the graph. We model the provider and the part-of relationships as the edges of the graph.

We introduce a finite set of transitions that we can employ on a state. The application of a transition to a state results in another state. In the following table, we list this set of transitions, with their notation and meaning. We produce the state space as follows. Starting from the first state (i.e., user's scenario), we apply a greedy algorithm to produce a new graph: the *state space*. This algorithm implements a sequence of steps to handle the afore-mentioned transitions in an efficient manner. The optimal state is chosen according to our cost model's criteria, in order to minimize the execution time of an ETL scenario.

Name	Notation	Meaning
Swap	SWA(a_i,a_j) a_i,a_j : unary	Swap activities a_i,a_j *special case*: SWA(a_i,a_{i+1})
Merge	MER($a_{i;j}$,a_i,a_j)	Merge a_i,a_j and create $a_{i;j}$
Split	SPL($a_{i;j}$,a_i,a_j)	Split $a_{i;j}$ to two new a_i,a_j
Order	ORD(a_i)	Sort a set of tuples
Replace	REP(a_i,a_i')	Replace activity a_i with a_i'
Factorize	FAC(a_i,a_j) a_i,a_j : binary/unary	Factorization of unary a_j from binary a_i
Distribute	DIS(a_j,a_i) a_i,a_j : binary/unary	Distributivity of unary a_j with respect to binary a_i

In Figure 5, we present a simplified view of CSS's results, considering the case of our motivating example and taking for first state the scenario of Figure 4.

6. Conclusions and Future Work

In our research, we have proposed a novel conceptual and a novel logical model for the representation of ETL processes with two main characteristics: genericity and customization. Also, we have presented an extensible palette of several templates, representing frequently used ETL activities along with their semantics and their interconnection. Thus, we can use these activities to design and implement ETL scenarios. Moreover, we are working on a method to optimize the execution plan of an ETL scenario. To complement the aforementioned issues, we have prototypically implemented a graphical tool, named ARKTOS II, with the goal of facilitating the design and the (re-)use of ETL scenarios, based on our model.

Clearly, a lot of work remains to be done for the completion of our research approach. The main challenge is the practical application of this disciplined approach in real world cases and its further tuning to accommodate extra practical problems.

7. Acknowledgements

I would like to thank Panos Vassiliadis and Timos Sellis. Their helpful comments and suggestions improved not only the presentation of this paper, but allowed me to clarify the idea of my PhD thesis.

8. Bibliography

[Arde02] Ardent Software. DataStage Suite. Available at http://www.ardentsoftware.com/

[BoFM99] M. Bouzeghoub, F. Fabret, M. Matulovic. Modeling Data Warehouse Refreshment Process as a Workflow Application. In Proc. Intl. Workshop DMDW'99, Heidelberg, Germany, (1999).

[BoJR98] G. Booch, I. Jacobson, J. Rumbaugh. The Unified Modeling Language User Guide. Addison-Wesley Pub Co. (1998)

[CDL+99] D. Calvanese et al. A principled approach to data integration and reconciliation in data warehousing. Proc. of DMDW'99, Heidelberg, Germany, (1999).

[Data02] DataMirror Corporation. Transformation Server. Available at http://www.datamirror.com

[ETI02] Evolutionary Technologies Intl. ETI EXTRACT. Available at http://www.eti.com/

[Gart03] Gartner. ETL Magic Quadrant Update: Market Pressure Increases. Available at http://www.gartner.com/reprints/informatica/112769.html

[GFSS00] H. Galhardas, D. Florescu, D. Shasha and E. Simon. Ajax: An Extensible Data Cleaning Tool. In Proc. ACM SIGMOD, pp. 590, Texas, 2000.

[GoMR98] M. Golfarelli, D. Maio, S. Rizzi. The Dimensional Fact Model: a Conceptual Model for Data Warehouses. Invited Paper, International Journal of Cooperative Information Systems, vol.7(2&3)1998

[HuLV00] B. Husemann, J. Lechtenborger, G. Vossen. Conceptual data warehouse modeling. In Proc. of 2nd DMDW, pp. 6.1 –6.11, Sweden (2000).

[IBM02] IBM. IBM Data Warehouse Manager. Available at http://www-3.ibm.com/software/data/db2/datawarehouse

[Info03] Incormatica. PowerCenter 6. Available at: http://www.informatica.com/products/data+integration/powercenter/default.htm

[KRRT98] R. Kimbal, L. Reeves, M. Ross, W. Thornthwaite. The Data Warehouse Lifecycle Toolkit. John Wiley & Sons, February 1998.

[LWGG00] W. Labio et al. Efficient Resumption of Interrupted Warehouse Loads. In Proceedings of the 2000 ACM SIGMOD, pp. 46-57, Texas, 2000.

[Micr02] Microsoft Corp. MS Data Transformation Services. Available at www.microsoft.com/sq

[Orac02] Oracle Corp. Oracle9i™ Warehouse Builder User's Guide, Release 9.0.2. November 2001.

[RaHe00] V. Raman, J. Hellerstein. Potters Wheel: An Interactive Framework for Data Cleaning and Transformation. TR University of California at Berkeley, 2000. Available at http://www.cs.berkeley.edu/~rshankar/papers/pwheel.pdf

[Sara00] Sunita Sarawagi (editor). Special Issue on Data Cleaning. IEEE Data Engineering Bulletin, Vol. 23, No. 4, December 2000.

[SiVa03] A. Simitsis, P. Vassiliadis. A Methodology for the Conceptual Modeling of ETL Processes. In Proc. of DSE'03, Velden, Austria, June 17, 2003.

[SBHD98] C. Sapia, M. Blaschka, G. Höfling, B. Dinter: Extending the E/R Model for the Multidimensional Paradigm. In ER Workshops 1998, pp. 105-116. Lect. Notes in Comp. Science 1552 Springer 1999

[TrBC99] N. Tryfona, F. Busborg, J.G.B. Christiansen. starER: A Conceptual Model for Data Warehouse Design. In DOLAP, pp. 3-8, Missouri, USA, 1999.

[TrPG00] J.C. Trujillo, M. Palomar, J. Gómez: Applying Object-Oriented Conceptual Modeling Techniques to the Design of Multidimensional Databases and OLAP Applications. Proc. of WAIM-00, pp. 83-94, Shanghai, China, June 2000.

[Vass00] P. Vassiliadis. Gulliver in the land of data warehousing: practical experiences and observations of a researcher. In Proc. of DMDW, pp. 12.1 –12.16, Stockholm, Sweden, 2000.

[VaSS02] P. Vassiliadis, A. Simitsis, S. Skiadopoulos. Modeling ETL activities as graphs. In Proc. of DMDW'2002, pp. 52-61, Toronto, Canada, 2002. [Long Vers.: http://www.dbnet.ece.ntua.gr/~asimi]

[VaSS02a] P. Vassiliadis, A. Simitsis, S. Skiadopoulos. Conceptual Modeling for ETL processes. In Proc. of DOLAP, McLean, USA, November 8, 2002. [Long Vers.: http://www.dbnet.ece.ntua.gr/~asimi]

[VSGT03] Vassiliadis, P., Simitsis, A., Georgantas, P., and Terrovitis, M., A Framework for the Design of ETL Scenarios. In the Proceedings of the 15th CAiSE, Velden, Austria, June 16, 2003.

[VVS+01] P. Vassiliadis et al. Arktos: Towards the modeling, design, control and execution of ETL processes. Information Systems, 26(8), pp. 537-561, December 2001, Elsevier Science Ltd.