Managing ETL Processes

Alexander Albrecht Felix Naumann
Hasso-Plattner-Institute at the University of Potsdam, Germany
{albrecht,naumann}@hpi.uni-potsdam.de

ABSTRACT

ETL tools allow the definition of sometimes complex processes to extract, transform, and load heterogeneous data into a data warehouse or to perform other data migration tasks. In larger organizations many ETL processes of different data integration projects are accumulated. Such processes can encompass common sub-processes, shared data sources and targets, and same or similar operations. However, there is no common method or approach to systematically manage such ETL processes. We propose the high-level management of such processes as a generic approach to enable their flexible re-use, optimization, and rapid development. To this end we introduce a set of basic operators on ETL processes, such as MERGE or INVERT, and motivate their use in several scenarios.

1. ETL MANAGEMENT

Consolidating data into a single physical store has proven to be the most effective approach to provide fast, highly available, and integrated access to relevant information. ETL processes, i.e., extraction, transformation, and loading (ETL), are used to migrate heterogeneous data from one or more data sources into a target system to form data repositories, data marts, or data warehouses. The extraction phase largely deals with the technical heterogeneity of the different sources and imports relevant data into a staging area. The transformation phase is the heart of an ETL process. Here, syntactical and semantical heterogeneities are overcome using specialized components, often called stages: All data is brought into a common data model and schema using mapping technology; data scrubbing and cleansing techniques standardize the data; simple components reflect operators of the relational algebra for instance to aggregate or combine data sets; duplicate detection algorithms search for common representations of same real world objects, which are in turn consolidated to a single representation. Such stages are strung together to often complex ETL graphs. Finally, the load phase loads the integrated, consolidated, and cleaned data from the staging area into the appropriate databases or warehouses.

In large companies many data sources from different organizational areas (finances, customer relationships, human resources, etc.) are involved in a variety of data integration projects using ETL tools. Over time, there are many ETL processes, which may encompass shared data sources, same data targets, common sub-processes and stages configured in an equal or similar way. However, we are not aware of a common method, approach, or framework to uniformly manage entire ETL processes. We discuss several projects that cover certain sub-aspects, such as ETL optimization or ETL and schema mapping, later in Sec. 4. Apart from providing a structured overview of existing processes and the ability to search within the repository of all created processes, systematic ETL management promises more efficient development of new processes through reuse, better use of ETL results by channeling them to different targets and even back to the sources, and more efficient execution of existing processes through shared or rearranged stages.

Our approach is inspired by the model management research, which defines a set of operators for manipulating models and mappings [1, 5]. In analogy, the main idea of an ETL management platform is to reduce the amount of programming needed to develop or maintain ETL processes.

An ETL management platform should comprise all company-wide developed ETL processes in a common repository. In practice, there is not only one ETL tool [12] in use and often ETL is performed without any tooling but simply as a sequence of queries, algorithms, and short scripts: For instance, within a single company the marketing department may design their data integration projects with a tool different from the ETL tool used in the sales department, and the IT department might have home-grown processes performing similar tasks. Therefore, some degree of interoperability should be supported, and can be achieved by importing tool-specific process specifications in a tool-independent representation. Most commercial and open source ETL tools provide process specification in some proprietary XML format, which can be directly wrapped in an internal platform specific process description.

To establish ETL management we introduce a set of basic operators as first class citizens. Due to space constraints we omit the presentation of formal semantics. We want to expose the benefits of ETL Management operators in general. Examples include:

- SEARCH – retrieves all ETL processes that contain the specified search terms. Search terms may comprise dif-

VLDB '08, August 24-30, 2008, Auckland, New Zealand

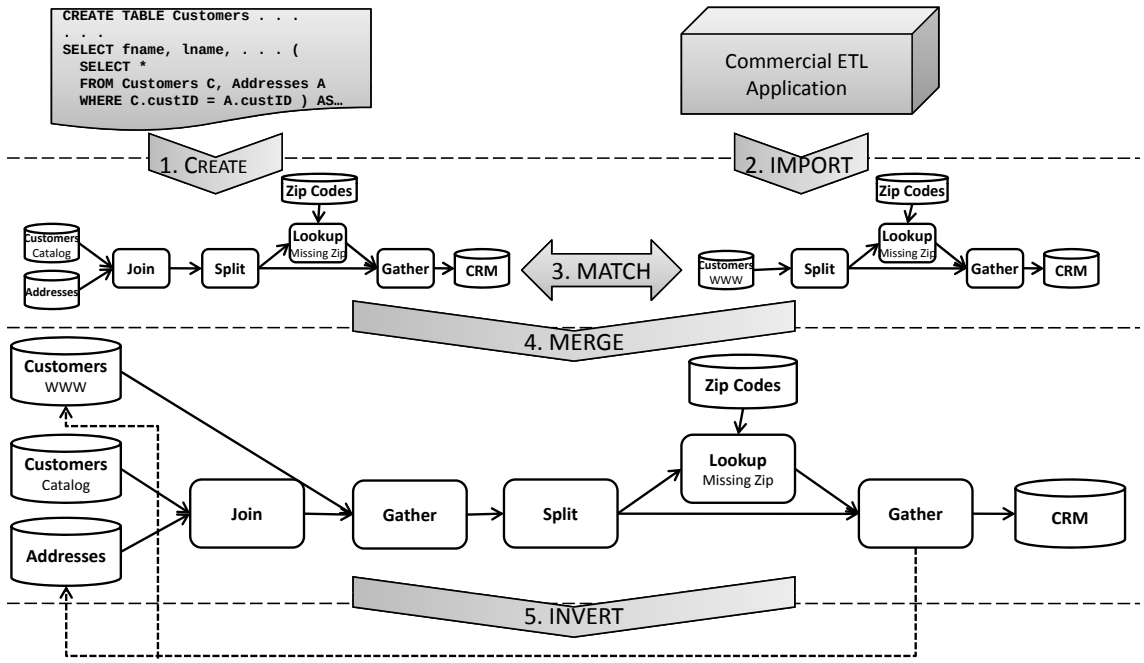

Figure 1: Sample ETL scenario: Marketing database with a single view of customers

ferent aspects of an ETL process, such as stage names, database schemata, or server addresses.

- MATCH – finds for a given ETL process all corresponding ETL processes that extract, transforms, or load common data in a similar way.
- CREATE – generates an ETL process from (a sequence of) non-ETL data transformation steps, such as mappings between data sources or SQL scripts.
- IMPORT – creates a tool-independent representation for a product-specific ETL process to support ETL management in a tool-independent manner.
- MERGE – takes one or more, often independently developed ETL processes as input and returns a merged ETL process.
- REWRITE – eliminates design mistakes made during ETL development by restructuring stages within an ETL process
- INVERT – feeds back the output of one ETL process to the sources in order to share the benefit of any data cleansing and integration stage.
- DEPLOY – generates from a tool-independent ETL representation an executable ETL process for a specific deployment platform [12].

Section 3 elaborates these and lists more useful operators. Before, in the following Sec. 2, we provide an exemplary scenario showing the benefits of high-level ETL management. Related work is surveyed in Sec 4 and finally, Sec. 5 concludes and outlines next steps.

2. SAMPLE SCENARIO

The scenario in Fig. 1 is a simplified example of loading data into a customer relationship management marketing database (marked as CRM), which provides a single view of customers from information stored in different sources. In this scenario, address information of catalog customers is stored in a separate table, whereas all information about internet customers resides in a single table. First, the underlying ETL process uses the stage Gather, which represents the union operator, to combine customer data from catalog sales and the online shop. Gather expects that both incoming data flows use the same schema, which is then propagated to the output.

Subsequently, a simplified address correction is performed: The data flow is split into two streams of tuples – one stream with missing zip codes, the other with existing zip codes. A tuple with a missing zip is assigned a postal code using an address lookup. Finally, the two split data streams are combined into one, which is in turn loaded into the CRM database.

Let us assume that the ETL process in Fig. 1 is the result of a MERGE operation on two independently developed ETL processes, one that integrates customer data from an online shop and the other loads customer data from catalog selling, whereas both corresponding ETL processes were found by the MATCH operator. It is imaginable that one of the processes was derived by applying the CREATE operator using existing SQL scripts. MERGE yields to a better utilization of system resources, compared to a separate execution of both single processes, because the address lookup is done for all customer data once. Furthermore, the combined ETL process provides an uniform view of all customer.

In addition to executing the MERGE operator, it makes sense to apply the INVERT operator to the merged ETL process, to replace the dirty data in the original sources with the consolidated, corrected, and enhanced address information.

Thus, applications working on the original sources may benefit from address data cleansing performed within the ETL process.

3. OPERATORS FOR ETL MANAGEMENT

In this section we describe the semantics of selected operators in detail. The underlying idea of all operators for ETL management is the reduction of complexity and the raise of abstraction to enable and simplify reuse, development, optimization, and maintenance of ETL processes.

3.1 Match

Given an ETL process, the MACTH operator allows to find similar ETL processes and therefore provides a simple access to a possibly large number of ETL processes stored within the repository. Similarity is yet to be defined, but can be interpreted as containing operators with same or similar functionality or as accessing same or similar data sources and/or targets. In analogy to information retrieval systems, the operator should determine a numeric similarity measure on how well each ETL process in the repository matches the given process, and rank the result according to this value. In general, it is hard to define a suitable similarity measure for ETL processes, because of the variety of ETL features and the presence of semantic or syntactic heterogeneity. Note that the given ETL process, used as input to MATCH, can not only be an entire complex process but also a single ETL stage or data source.

3.2 Merge

The intention of the MERGE operator is to combine two or more ETL processes into one. In order to merge ETL processes, it is necessary to first identify corresponding subprocesses between similar ETL processes. A common subprocess consists of an equal sequence of equivalent or sufficiently similar stages. Such sub-processes may be derived by rewriting the considered ETL processes. The smallest possible sub-process is a common ETL stage. A common sub-process may be located at the beginning, in the middle or at the end of the merged ETL process. One major difficulty in combining ETL processes is to find within a set of common sub-processes those corresponding sub-processes that are most qualified for a merge.

In general, company-wide developed ETL processes share common sources and targets and are performed within several ETL tools. Applying MERGE in such scenarios, may achieve a better utilization of shared resources, along with latency improvement and a reduced amount of data transmission. Applying MERGE to a set of similar ETL processes also promises an enhancement of performance compared to performing all processes in a separate run. This assumption is motivated by the expectation that in a merged ETL process common data may be processed in an equal way. In addition to optimization, a merged ETL process provides a single view of all information that was originally processed separately.

Apart from performance benefits, consolidating large sets of ETL processes promises reduced overhead and maintenance.

3.3 Rewrite

The purpose of the REWRITE operator is to restructure an existing ETL process in a (semi-)automatic way. In particular, REWRITE allows optimization and maintenance of ETL processes in order to eliminate design mistakes made during the ETL development. A sample rewrite may place late running standardization stages at the beginning of the process to ensure consistency of data for the following processing steps. In addition to effectiveness, poor design of an ETL process may have a negative impact on its efficiency too. In analogy to traditional query optimization, late filtering of data is a prominent example how ETL performance is negatively affected by performing expensive and time-consuming transformations for data that is later filtered out. Another common mistake in ETL design is the late placement of stages that discard attributes that are not involved in the population of the data warehouse. Row size has a significant impact on the throughput rate of ETL processes. Thus one should avoid costly processing of large rows by placing stages that reduce the number of attributes as soon as possible. Problems that arise in the context of reorganizing ETL processes are described for instance in [9].

3.4 Invert

The INVERT operator inverts an ETL (sub-)process in such a way that cleansed data from the staging area of an ETL process is moved backwards to the sources. This is accomplished by creating an inverted ETL process that starts at a predefined stage within the staging area and loads the cleansed data into a predefined source table. Based on an invertible ETL process, major challenges are restoring the data model and schema of the source table and inverting individual ETL stages, which for instance might discard attributes or delete tuples.

The idea of sending back improved data to the sources is mentioned in passing as *backflow of cleaned data* in [7], but this idea still remains to be worked out: Within an ETL process specific data quality problems are solved, and the consolidated and cleaned data should also be fed back to the sources in order to ensure data quality for applications working on the original sources. A backflow also promises an improvement of future ETL projects in order to avoid multiple fixing of data quality problems of the source systems.

Since master data management (MDM) became an important issue in most companies, the idea of replacing dirty master data with cleansed master data from ETL processes in the source systems is a promising approach. Master data describes business objects, such as products, business partners or customers. In particular, for ETL-based migration of dirty master data from sources to MDM server, the INVERT operator opens up an interesting approach to send back cleansed and consolidated master data to the appropriate sources in a (semi-)automatic manner. The inverted ETL process should then be regarded as an MDM process depending on its purpose.

An improved inversion of an ETL process may be achieved by a sequence of REWRITE operations, that place late running data quality stages at the beginning of the ETL process, near to the predefined source table, to overcome difficulties in inverting other ETL stages. This manual intervention should only be performed when appropriate.

3.5 Other basic operators

We conclude this section with a short examination of operators that support interoperability, including CREATE, IM-

PORT, and DEPLOY. This approach assumes for now that supported ETL or data integration tools exchange their data integration projects at a tool-independent, logical level.

Besides ETL, there are many ways to move data from sources to targets, such as setting up mappings between the data sources or performing SQL scripts. In order to benefit from data integration work that was already done, we suggest the CREATE operator, which converts non-ETL data transformation steps, into a tool-independent ETL representation. Such an ETL process could be enhanced within an specific ETL tool or managed with the introduced ETL management operators in order to make further use of it. In [4] Wisnesky et al. introduce Orchid – a prototype system that creates ETL processes from declarative mapping specifications and vice versa.

The IMPORT operator supports interoperability by converting tool-specific ETL process specifications into a tool-independent representation and by importing it into the respective ETL management platform. Subsequent ETL management operations are perform on the imported ETL process, which may be deploy afterwards into multiple data integration platforms using the DEPLOY operator. One major problem with deployment of tool-independent ETL processes is due to the fact that ETL tools have a different scope of stages and in general ETL processes are more expressive than other data integration approaches, such as mappings or SQL queries.

4. RELATED WORK

Although the practical importance of ETL in data integration projects is significant [11], only some work on ETL at a meta-level has been performed in the database research community. Most related research results are about ETL process modeling, whereas the Orchid approach in [4] uses the work of [8] to convert real, IBM WebSphere DataStage ETL processes into their abstract OHM model. Moreover, the Orchid approach provides a solution for creating ETL processes out of declarative schema mapping and vice versa. Furthermore, there is a body of research on ETL process rewriting and optimization by Vassiliadis et al. [9]. There are other approaches in modeling ETL processes [3, 10]; both approaches do not support the description of schema mappings between ETL stages. Thus, none of the existing ETL modeling approaches work on the full set of ETL mechanisms or support fundamental ETL aspects, such as fuzzy operators or data lineage.

In the context of ETL we also want to refer the research of [2] in the area of data lineage, in particular in this work aspects of data that undergoes a sequence of transformations are studied. An introduction to model management, with its inspiring principles of providing generic algebraic operations, can be found in [5].

5. CONCLUSIONS AND NEXT STEPS

In this paper we presented the idea of a generic approach for ETL management, meaning that all introduced high-level operators, such as SEARCH, MERGE, or INVERT, are applicable to different kinds of ETL processes from different platforms. The generic approach shall be achieved by treating ETL processes in a tool-independent representation. We motivated the usefulness of our approach by an example and described the behavior of some operators in more detail.

Our initial prototype is based on the open-source ETL tool Clover [6]. Clover is an Eclipse plugin, which provides ETL process specifications in a proprietary XML format. In addition, it allows the integration of ETL management operators based on the available java sources.

Our next steps include the formal definition of a number of the proposed operators, their implementation in our prototype, and subsequent case studies. Additionally we plan to build wrappers for the most common ETL tools, so that product-specific ETL processes can be imported, and rewritten processes can be deployed in different formats. In this context, exploring the different semantics of similar ETL stages in different tools and describing their expressiveness will be a further direction.

6. REFERENCES

[1] Philip A. Bernstein. Generic model management: A database infrastructure for schema manipulation. In *Proceedings of the International Conference on Cooperative Information Systems (CoopIS)*, pages 1–6, Trento, Italy, 2001.

[2] Yingwei Cui and Jennifer Widom. Lineage tracing for general data warehouse transformations. *VLDB Journal*, 12(1):41–58, 2003.

[3] Object Management Group. Common Warehouse Metamodel (CWM), 2003. `http://www.omg.org/technology/documents/formal/cwm.htm`.

[4] Ryan Wisnesky Ahmed Radwan Mauricio A. Hernandez, Stefan Dessloch and Jindan Zhou. Orchid: Integrating schema mapping and ETL. In *Proceedings of the International Conference on Data Engineering (ICDE)*, Cancun, Mexico., 2008.

[5] Sergey Melnik. *Generic Model Management: Concepts and Algorithms*. LNCS 2967. Springer Verlag, Berlin – Heidelberg – New York, 2004.

[6] David Pavlis. clover.etl, 2008. `www.cloveretl.org`.

[7] Erhard Rahm and Hong Hai Do. Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin*, 23(4):3–13, 2000.

[8] Alkis Simitsis. Mapping conceptual to logical models for ETL processes. In *DOLAP '05: Proceedings of the 8th ACM international workshop on Data warehousing and OLAP*, pages 67–76, New York, NY, USA, 2005. ACM.

[9] Alkis Simitsis, Panos Vassiliadis, and Timos Sellis. Optimizing ETL processes in data warehouses. In *Proceedings of the International Conference on Data Engineering (ICDE)*, Tokyo, Japan., 2005.

[10] Juan Trujillo and Sergio Luján-Mora. A UML based approach for modeling ETL processes in data warehouses. In *Proceedings of the International Conference on Conceptual Modeling (ER)*, Chicago, IL, 2003.

[11] Panos Vassiliadis, Anastasios Karagiannis, Vasiliki Tziovara, and Alkis Simitsis. Towards a benchmark for ETL workflows. In *Proceedings of the 5th International Workshop on Quality in Databases (QDB)*, Vienna, Austria., 2007.

[12] Wikipedia. ETL tools, 2008. `http://en.wikipedia.org/wiki/Category:ETL_tools`.