CEPIS UPGRADE is the European Journal for the Informatics Professional, published bi-monthly at <http://cepis.org/upgrade>

Publisher
CEPIS UPGRADE is published by CEPIS (Council of European Professional Informatics Societies, <http://www.cepis.org/>), in cooperation with the Spanish CEPIS society ATI (*Asociación de Técnicos de Informática,* <http://www.ati.es/>) and its journal Novática

CEPIS UPGRADE monographs are published jointly with Novática, that publishes them in Spanish (full version printed; summary, abstracts and some articles online)

CEPIS UPGRADE was created in October 2000 by CEPIS and was first published by Novática and INFORMATIK/INFORMATIQUE, bimonthly journal of SVI/FSI (Swiss Federation of Professional Informatics Societies)

CEPIS UPGRADE is the anchor point for UPENET (UPGRADE European NETwork), the network of CEPIS member societies' publications, that currently includes the following ones:
- inforewiew, magazine from the Serbian CEPIS society JISA
- Informatica, journal from the Slovenian CEPIS society SDI
- Informatik-Spektrum, journal published by Springer Verlag on behalf of the CEPIS societies GI, Germany, and SI, Switzerland
- ITNOW, magazine published by Oxford University Press on behalf of the British CEPIS society BCS
- Mondo Digitale, digital journal from the Italian CEPIS society AICA
- Novática, journal from the Spanish CEPIS society ATI
- OCG Journal, journal from the Austrian CEPIS society OCG
- Pliroforiki, journal from the Cyprus CEPIS society CCS
- Tölvumál, journal from the Icelandic CEPIS society ISIP

Editorial correspondence: Llorenç Pagés-Casas <pages@ati.es>
Advertising correspondence: <info@cepis.org>

Subscriptions
If you wish to subscribe to CEPIS UPGRADE please send an email to info@cepis.org with 'Subscribe to UPGRADE' as the subject of the email or follow the link 'Subscribe to UPGRADE' at <http://www.cepis.org/upgrade>

ISSN 1684-5285

Monograph of next issue (October 2011)
"Green ICT"

(The full schedule of **CEPIS UP**GRADE is available at our website)

CEPIS UPGRADE
The European Journal for the Informatics Professional
http://cepis.org/upgrade
Vol. XII, issue No. 3, July 2011

Monograph
Business Intelligence
(published jointly with Novática*)
Guest Editors: *Jorge Fernández-González and Mouhib Alnoukari*

UPENET (UPGRADE European NETwork)

CEPIS NEWS

* This monograph will be also published in Spanish (full version printed; summary, abstracts, and some articles online) by **Novática**, journal of the Spanish CEPIS society ATI (*Asociación de Técnicos de Informática*) at <http://www.ati.es/novatica/>.

Designing Data Integration: The ETL Pattern Approach

Veit Köppen, Björn Brüggemann, and Bettina Berendt

The process of ETL (Extract-Transform-Load) is important for data warehousing. Besides data gathering from heterogeneous sources, quality aspects play an important role. However, tool and methodology support are often insufficient. Due to the similarities between ETL processes and software design, a pattern approach is suitable to reduce effort and increase understanding of these processes. We propose a general design-pattern structure for ETL, and describe three example patterns.

Keywords: Business Intelligence, Data Integration, Data Warehousing, Design Patterns, ETL, Process.

1 Introduction

Business Intelligence (BI) methods are built on high-dimensional data, and management decisions are often based upon data warehouses. Such a system represents internal and external data from heterogeneous sources in a global schema. Sources can be operational data bases, files, or information from the Web. An essential success factor for Business Data Warehousing is therefore the integration of heterogeneous data into the Data Warehouse. The process of transferring the data into the Data Warehouse is called Extract-Transform-Load (ETL).

Although the ETL process can be performed in any individually programmed application, commercial ETL tools are often used [1]. Such tools are popular because interfaces are available for most popular databases, and because visualizations, integrated tools, and documentation of ETL process steps are provided. However, a tool does not guarantee successful data integration. In fact, the ETL expert has to cope with several issues. Many of the challenges are recurrent. Therefore, we believe that a support for ETL processes is possible and can reduce design effort. We propose the use of the pattern approach from software engineering because similarities exist between the ETL process and the software design process.

Software patterns are used in object-oriented design as best practices for recurring challenges in software engineering. They are general, re-usable solutions: not finished designs that can be transformed directly into code, but descriptions of how to solve a problem. These patterns are described in templates and often included in a catalogue. Consequently, a software developer can access these templates and implement best practices easily. The idea of design patterns has been adapted to different domains including ontology design [2], usage-interface design [3], and information visualization [4].

In the domain of enterprise system integration, the pattern approach is adapted by [5]. [6] develops patterns for the design of service-oriented architectures. In this paper, we present patterns for the design and implementation of ETL processes.

The paper is organized as follows: in Section 2, the ETL process is described, and in Section 3 we present the ETL pattern approach with three example patterns. A brief evaluation of these patterns is presented in Section 4, and in Section 5 we summarize our work.

Authors

Veit Köppen received his MSc degree in Economics from *Humboldt-Universität zu Berlin*, Germany, in 2003. From 2003 until 2008, he worked as a Research Assistant in the Institute of Production, Information Systems and Operation Research, *Freie Universität Berlin*, Germany. He received a PhD (Dr. rer. pol.) in 2008 from *Freie Universität Berlin*. He is now a member of the Database Group at the Otto-von-Guericke University Magdeburg, Germany. Currently, he is the project coordinator in the project funded by the German Ministry of Education and Research. His research interests include Business Intelligence, data quality, interoperability aspects of embedded devices, and process management. More information at <http://wwwiti.cs.uni-magdeburg.de/~vkoeppen>. <veit.koeppen@iti.cs.uni-magdeburg.de>

Björn Brüggemann studied Computer Science at Otto-von-Guericke-University Magdeburg, Germany, and received his Masters Degree in 2010. In his Masters Thesis, he focused on Data Warehousing and the ETL process in the context of Data Quality. Since 2010, he has been working at Capgemini, Berlin, Germany, in Business Intelligence and Data Warehouse projects. More information at <http://www.xing.com/profile/Bjoern_Brueggemann3>. <Brueggemann.Bjoern@gmx.de>

Bettina Berendt is a Professor in the Artificial Intelligence and Declarative Languages Group at the Department of Computer Science of K.U. Leuven, Belgium. She obtained her PhD in Computer Science/Cognitive Science from the University of Hamburg, Germany, and her Habilitation postdoctoral degree in Information Systems from Humboldt University Berlin, Germany. Her research interests include Web and text mining, semantic technologies and information visualization and their applications, especially for information literacy and privacy. More information at <http://people.cs.kuleuven.be/~bettina.berendt>. <Bettina.Berendt@cs.kuleuven.be>

2 The ETL Process

Data Warehouses (DW) are often described as an architecture where heterogeneous data sources, providing data for business analysis, are integrated into a global data schema. Besides the basis database, where data is stored at

> “Business Intelligence methods are built on high-dimensional data, and management decisions are often based upon data warehouses”

a fine-grained level, data marts for domain-specific analyses are stored, containing more coarse-grained information. Furthermore, management tools such as data-warehouse managers and metadata managers are included in the architecture. A DW reference architecture is given in [7].

The process of data integration is performed in the *staging area* in the architecture. Here, heterogeneous data are extracted from their origins. Adapters and interfaces can be used to extract data from different sources such as operational (OLTP) databases, XML files, plain files, or the Web. This extraction is followed by transformation into the DW schema. This schema depends on the DW architecture and the domain or application scenarios. In practice, relational data warehouse are used and star or snowflake schema are applied as relational On-Line Analytical Processing (ROLAP) technologies, see for instance [8]. In addition, transformations according to data formats and aggregations as well as tasks related to data quality such as the identification of duplicates are performed during this step. Finally, the data is loaded from the staging area into the basis database within the DW. Based on this, a cube or different data marts can be built, data mining algorithms applied, reports generated, and analyses performed. In Figure 1, we present the ETL process in its generic steps.

A *monitor* observes a data source for changes. This is necessary to load updated data into the DW. The monitoring strategy is defined depending on the data source. Two main strategies exist: either all changes are processed to the monitor and the delta of all changes can be computed, or the monitor can only identify that changes occurred. We distinguish the following mechanisms:

- Reactions are selected according to the event-condition-action rules for defined situations.
- Relevant data or changes are stored in an additional data store, therefore the data is replicated.
- Logs can be parsed and used, which are otherwise used for recovery.
- Applications that update data can be monitored via time stamp methods or snapshots.

The *extraction* operation is responsible for loading data from the source into the staging area. This operation depends upon monitoring the method and the data source. For example, it is possible that the monitor identifies a change, but the extraction process happens later, at a time predefined by the extraction operation. There exist different strategies for the extraction operation:

- ***Periodically***, where data is extracted continuously and recurrently at a given time interval.Tthis interval depends on requirements on timeliness as well as dynamics in the source.
- ***Query-based***, where the extraction is started when an explicit query is performed instantly. Where all changes are directly propagated into the dw.
- ***Event-based***, where a time-, external- or system-relevant event starts the extraction operation.

The *transformation* within the staging area fulfils the tasks of data integration and data fusion. All data are integrated and transformed into the DW schema, and at the same time, data quality aspects are addressed, such as duplicate identification and data cleaning. Different transformations exist and can be categorized as follows:

- ***Key handling****:* since not all database keys can be included into the dw schema, surrogates are used.
- ***Data-type harmonization***, where data are loaded from heterogeneous data sources.
- ***Conversion of encodings*** of the same domain at-

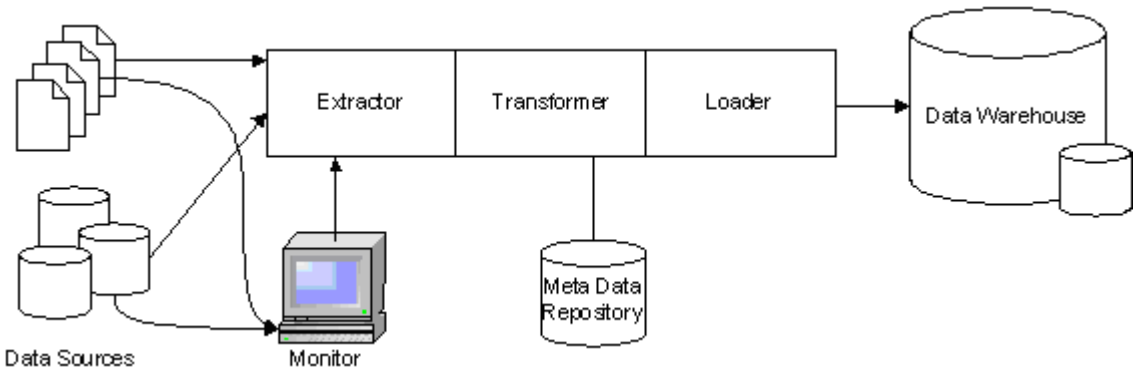

Figure 1: The ETL Process.

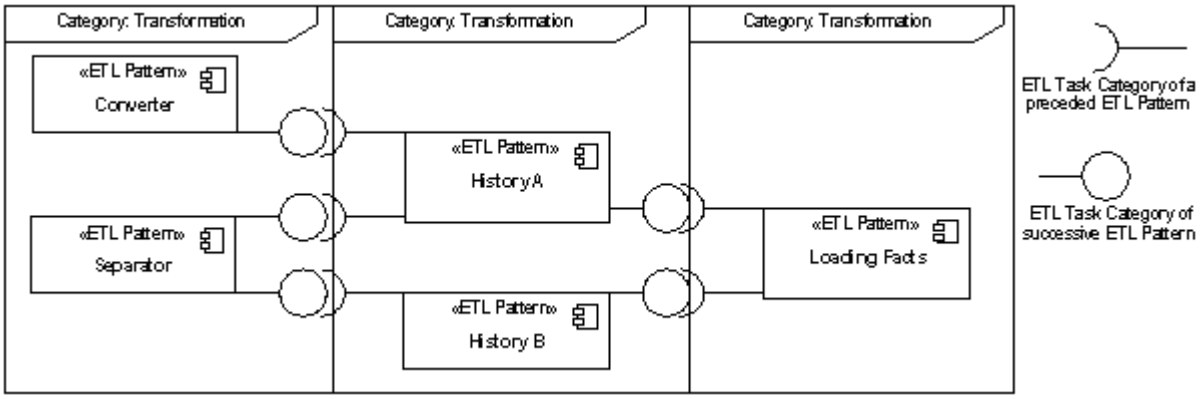

Figure 2: ETL Process with Patterns from Different Categories.

Element	META-DESCRIPTION	Mandatory?
Name	This name identifies the pattern in the catalogue.	Yes
Intention	A concise description at which use the pattern aims.	Yes
Classification	A reference to elementary or composite task with an optional refinement on the ETL steps.	Yes
Context	This describes the situation where the problem occurs.	Yes
Problem	A detailed description of the problem.	Yes
Solution	A concise description of the solution.	Yes
Resulting Context	This describes the outcome and the advantages and disadvantages of using this pattern.	No
Data Quality	Which data quality issues are addressed and which data quality dimension/s is/are improved.	No
Variants	A reference to similar and adapted patterns.	No
Alternative Naming	Other commonly used names of the pattern.	No
Composite Property	Only composite patterns use this description and state the composition property of the pattern.	No
Used in	This element describes briefly where the pattern is applied. this helps in the understanding and decision whether a pattern should be used.	No
Implementation	For various ETL tools, the solution is put into practice differently, therefore different implementations are referenced here.	No
Demonstration	A reference to an exemplary implementation of this pattern.	No

Table 1: ETL Pattern Structure.

> “The process of transferring the data into the Data Warehouse is called Extract-Transform-Load (ETL)”

tribute value to a common encoding (e.g., 0/1 and m/f for gender are mapped to m/f).

- ***Unification of strings***, because the same objects can be represented differently (e.g., conversion to lower case).
- ***Unification of date format:*** although databases handle different data formats, some other sources such as files can only provide a fixed data format.
- ***Conversion of scales and scale units***, such as currency conversions.
- ***Combination or separation of attributes***, depending on the attribute level of the heterogeneous sources and the DW.
- ***Computation and imputation***, in the case that values can be derived but are not given in the source systems.

The *loading* of the extracted and transformed data into the DW (either into the basis database or into data marts) can occur in online or offline mode. If the DW is or should be accessed while the loading takes place, an online strategy is necessary. This should be used for incremental updates, where the amount of loading is small. In the first (initial) loading of a DW, the loading is high and the DW is run in an offline mode for the users. At this time, the loading operation has exclusive access to all DW tables. Another task for the load operation is the historicization of data; old data is not deleted in a DW but should be marked as deprecated.

The ETL process can be refined into several *ETL steps*, where each step consists of an initialization, a task execution, and a completion. These steps enable ETL designers to structure their work. The following steps can be necessary in an ETL process: extraction, harmonization and plausibility checks, transformations, loading into DW dimensions, loading into DW fact tables, and updating. We use this categorization for our template approach in the next section.

3 ETL Patterns

The term "*pattern*" was first described in the meaning used here in the domain of architecture [9]. A pattern is described as a three-part rule consisting of the relations between context, problem, and solution. A pattern is used for recurrent problems and describes the core solution of this problem. For pattern users, it is necessary to identify problem, context, and solution in an easy way. Therefore, templates should be used to structure all patterns uniformly.

We derive our pattern structure from software engineering patterns because of the similarities between Software Design and ETL processes. A *template* consists of different elements such as name and description. For examples of templates in object-oriented software design see [10], for software architecture design patterns see [11], and for the domain of data movement see [12]. They all have in common that some elements are mandatory and others are optional. Mandatory elements are the name of the pattern, context, problem description, and core solution.

We see two levels of tasks in an ETL process: *elementary* and *composite tasks*. An elementary task inside an ETL process is often represented by an operator in the tools. A decomposition is not useful, although there might exist an application that allows a decomposition. We present the Aggregator Pattern as an example pattern for solving an elementary ETL task in Section 3.1.

Elementary tasks can be used in a composite task. A composite task is the sequence of several tasks or operators and therefore more complex. We can classify the composite tasks according to the ETL steps described in Section 2. Apart from the loading into the DW dimensions, all categories and consequently all ETL patterns are independent of the DW schema. We support the design of composite tasks in the ETL process by including composition properties. These *composition properties* describe categories of tasks that are executed before or after the composite task. Figure 3 depicts this composition property for the History Pattern described in Section 3.2. Before the History Task is performed, loading into the DW dimensions and transformations may be performed. After the completion of the History Task, a loading into DW fact tables or into DW dimensions is possible. Note that all elements are optional in this example.

Providing this information, a sequence structure can be defined and visualized as we present in Figure 2. In this way, the complete design of the ETL process can be given at an abstract level and customization of the ETL process can easily implemented.

We structure our ETL patterns according to the template shown in Table 1.

In the following, we present three ETL patterns as examples. In our first example, an elementary ETL task is presented, the aggregator pattern. In the other two examples, we present composite ETL tasks: the history pattern, where data is stored in the DW according to changes in DW dimensions, and the duplicates pattern for the detection of duplicates.

> “Although the ETL process can be performed in any individually programmed application, commercial ETL tools are often used”

> “A pattern is described as a three-part rule consisting of the relations between context, problem, and solution”

3.1 The Aggregator Pattern

Name: Aggregator Pattern

Intention: Data sets should be aggregated via this pattern within ETL processes.

Classification: Elementary task

Context: From a database or file data on a fine-grained level are loaded into the DW.

Problem: The DW data model does not require data at a fine-grained level. If data from the operational system is not needed at a fine-grained level, two problems may occur: more storage is required in the DW, and performance decreases due to more data having to be processed.

Solution: An ETL operator is used that collects data from the sources and transforms them into the desired granularity.

Resulting Context: A performance increase can be obtained, in the DW system as well as in the ETL process, through the reduction of data. Furthermore, the required storage space is reduced. However, one disadvantage is that there exists no inverse operation, so the inference to original data is not possible. If data granularity changes, information loss may result.

3.2 The History Pattern

Name: History Pattern

Intention: Data sets in the dimension tables should be marked and cataloged.

Classification: Composite task in the category of dimension loading for star schema.

Context: Product, Location, and Time are dimension in the DW that can change over time. Analyses in the context of master data can be done according to the dimensions.

Problem: Master data changes only occasionally, but they do sometimes change (such as the last name of a person). These changes should be taken into account in the dimension tables. However, challenges occur due to the use of domain keys that change over time, thus they cannot be use as primary keys. This is in contrast to the modeling of dimension tables in the star schema. Another problem is the use of domain keys if redundancy is required.

Solution: An important challenge is to store old and new data in the DW system. Furthermore, a relation of fact table and dimension data is necessary. For this purpose, the dimensional table has to be extended by additional attributes. In a first step, a virtual primary key is added, together with one or more attribute/s storing current or up-to-date information. The attributes *valid_from* and *valid to* are used to store the information about when the data was valid. This is described differently in the literature, for example as changes of type II dimensions [10] or as snapshot history [13]. For every data set, a decision has to be made: either it is a new dataset, an updated one, or a data set that already existed in the dimension tables of the DW. For this comparison, a key should be used that is persistent in time, such as the domain key. Every source data set is mapped with this key to dimensions. If this is not possible, a new entry is identified. If all attributes are equal for the source data set compared to a data set in the DW, an existing one is identified. Otherwise an updated data set is detected. A new data set has to be stored in the dimension tables and the attributes *valid_from* and *valid_to* as well as the virtual key have to be generated and *timeliness* set to true. For an update, the *timeliness* and *valid_to* information of the already existing dataset have to set before the source dataset can be entered into the DW.

Resulting Context: All data are historicized, however this influences performance due to the increase of the data amount in the dimension tables. The domain key has to be unique; otherwise, duplicate detection has to be performed first.

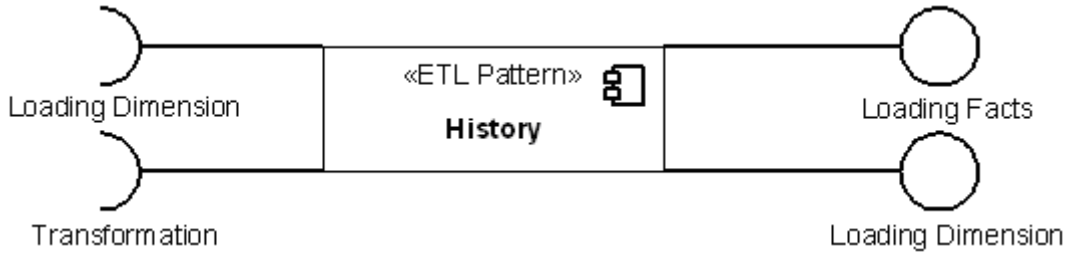

Figure 3: Composite Properties for History Pattern.

> “We derive our pattern structure from software engineering patterns because of the similarities between Software Design and ETL processes”

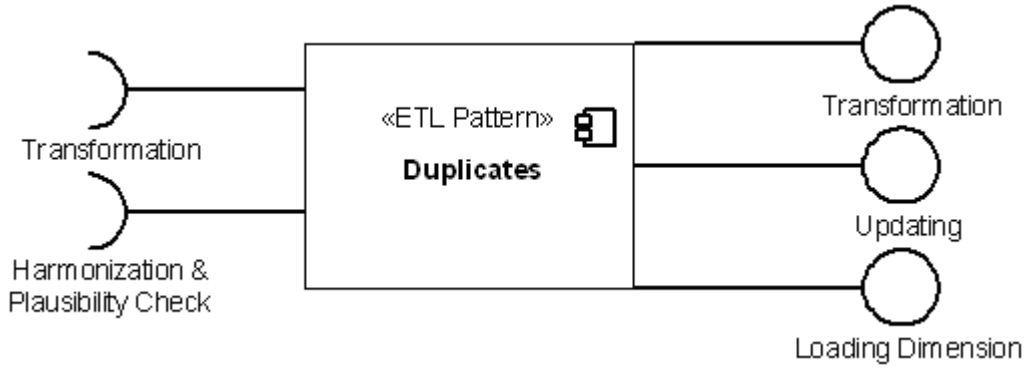

Figure 4: Composite Properties of the Duplicates Pattern.

Data Quality: All available information (*data completeness* for dimensions) is accessible for analysis with the help of the history pattern. Data *timeliness* is another advantage for data quality issues, as long as the loading is performed at short, regular time intervals.

Composite: Before an ETL task from the History Pattern is performed, patterns from the categories Loading Dimension and Transformation may be applied. The History Pattern can be followed by patterns from the Loading Facts and Loading Dimension categories.

3.3 The Duplicates Pattern

Duplicate detection is a common but complex task in ETL processes. With our pattern template, we briefly describe the solution, although in practice this should be described more comprehensively, see [14][15][16] for more details.

Name: Duplicates Pattern

Intention: This pattern reduces redundancy in the DW data; in the best case, it eliminates redundancy completely.

Classification: Composite task in the category transformation.

Context: Data from heterogeneous sources (e.g., applications, databases, files) have to be loaded into the DW.

Problem: A data hub for the integration of data is not always available, therefore master data redundancy occurs in different business applications. A duplicate are two or more data sets that describe the same real-world object. Data in the DW should give a consolidated view and must be free of duplicates.

> “The creation of complex ETL processes is often a challenging task for ETL designers. This complexity is comparable to software engineering”

Solution: Duplicates have to be identified and deleted. As a first step, data have to be homogenized. This includes conversions, encodings, and separations of all comparative attributes. Partitioning of data reduces comparison effort, but must be chosen with caution in order not to miss duplicates. The comparison is based on similarity measures that help to identify duplicates. There exist different methods and measures based on the data context.

A data fusion of identified duplicates has to be carried out. Aspects of uncertainty and inconsistencies have to be considered in this context. Inconsistency means that semantically identical attributes have different values. Uncertainty occurs if only null values are available. *Data conflict avoidance* can be carried out via the survivor strategy [17], where a predefined source entry is favored against all others, or via set-based merge [9], where the disjunction of all values is stored. In contrast, *data conflict resolution* can be carried out via a decision strategy, where an entry is determined from the sources, or a mediation strategy, where new values can be computed.

Resulting Context: Duplicates are only partially detected. Due to complexity of the duplicate detection, the ETL designer has to carefully consider data context and appropriate methods for measuring similarities or partitioning strategy.

Data Quality: The data quality issue *non-redundancy* is supported with this pattern.

Composite: The Duplicates Pattern can be preceded by patterns from the Transformation category as well as from the category Harmonization & Plausibility Check. The categories Transformation, Updating, and Loading Dimension include patterns that can be used for subsequent tasks, see Figure 4.

4 Conclusion and Future Work

The creation of complex ETL processes is often a challenging task for ETL designers. This complexity is comparable to software engineering, where patterns are used to structure the required work and support software architects and developers. We propose ETL patterns for the support of ETL designers. This provides an adequate structure for

> “We plan to create an ETL pattern catalogue with descriptions of most common ETL tasks and the corresponding challenges”

performing recurring tasks and allows developers to apply solutions more easily. In this paper we have presented a template for the general description of ETL patterns. Furthermore, we have presented three examples.

As future work, we plan to create an ETL pattern catalogue with descriptions of most common ETL tasks and the corresponding challenges. This includes an evaluation of the pattern catalogue as well as the application to different ETL tools.

References

[1] R. Schütte, Thomas Rotthowe, and Roland Holten, editors. Data Warehouse Managementhandbuch. Springer-Verlag, Berlin et al., 2001.

[2] OntologyDesignPatterns.org. <http://ontology design patterns.org>.

[3] S.A. Laakso. Collection of User Interface Design Patterns. University of Helsinki, Dept. of Computer Science. <http://www.cs.helsinki.fi/u/salaakso/patterns/index.html. 2003> [accessed July 20, 2011].

[4] J. Heer and M. Agrawala. Software Design Patterns for Information Visualization. IEEE Transactions on Visualization and Computer Graphics, 12 (5): 853, 2006.

[5] G. Hohpe and B. Woolf. Enterprise integration patterns. Designing, building, and deploying messaging solutions. Addison-Wesley, Boston, 2004.

[6] T. Erl. SOA Design Patterns. Prentice Hall PTR, Boston, 2009.

[7] A. Bauer and H. Günzel. Data-Warehouse-Systeme. Architektur, Entwicklung, Anwendung. dpunkt Verlag, Heidelberg, 2009.

[8] E.F. Codd, S.B. Codd, and C.T. Salley. Providing OLAP to user-analysts: An IT mandate. Technical report, Codd & Associates, 1993.

[9] D. Apel, W. Behme, R. Eberlein, and C. Merighi. Datenqualität erfolgreich steuern. Praxislösungen für Business-Intelligence-Projekte. Carl Hanser Verlag, 2009.

[10] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, 1995.

[11] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. Pattern-Oriented Software Architecture. A System of Patterns. Volume 1. Wiley, 1996.

[12] P. Teale. Data Patterns. Patterns and Practices. Microsoft Corporation, 2003.

[13 H.-G. Kemper, W. Mehanna, and C. Unger. Business Intelligence - Grundlagen und praktische Anwendungen. Eine Einführung in die IT-basierte Management unterstützung. Vieweg Verlag, Wiesbaden, 2006.

[14] I. P. Fellegi and A.B. Sunter. A Theory for Record Linkage. Journal of the American Statistical Association, 64(328):1183–1210, 1969.

[15] A.K. Elmagarmid, P.G. Ipeirotis, and V.S. Verykios. Duplicate record detection: A survey. IEEE Transactions on Knowledge and Data Engineering, 19:1–16, 2007.

[16] C. Batini and M. Scannapieco. Data Quality: Concepts, Methodologies and Techniques. Springer, 2006.

[17] R. Hollmann and S. Helmis. Webbasierte Datenintegration. Ansätze zur Messung und Sicherung der Informationsqualität in heterogenen Datenbeständen unter Verwendung eines vollständig webbasierten Werkzeuges. Vieweg Verlag, 2009.