Madhikerrni, Manik; Främling, Kary

Data discovery method for Extract- Transform-Load

Published in:
2019 IEEE 10th International Conference on Mechanical and Intelligent Manufacturing Technologies, ICMIMT 2019

DOI:
10.1109/ICMIMT.2019.8712027

Published: 09/05/2019

Document Version
Peer reviewed version

Please cite the original version:
Madhikerrni, M., & Främling, K. (2019). Data discovery method for Extract- Transform-Load. In *2019 IEEE 10th International Conference on Mechanical and Intelligent Manufacturing Technologies, ICMIMT 2019* (pp. 174-181). [8712027] IEEE. https://doi.org/10.1109/ICMIMT.2019.8712027

Data discovery method for Extract-Transform-Load

Manik Madhikermi, Kary Främling
School of Science, Aalto University
P.O. Box 15400, FI-00076 Aalto, Espoo, Finland
Email: firstname.lastname@aalto.fi

***Abstract*—Information Systems (ISs) are fundamental to streamline operations and support processes of any modern enterprise. Being able to perform analytics over the data managed in various enterprise ISs is becoming increasingly important for organisational growth. Extract, Transform, and Load (ETL) are the necessary pre-processing steps of any data mining activity. Due to the complexity of modern IS, extracting data is becoming increasingly complicated and time-consuming. In order to ease the process, this paper proposes a methodology and a pilot implementation, that aims to simplify data extraction process by leveraging the end-users' knowledge and understanding of the specific IS. This paper first provides a brief introduction and the current state of the art regarding existing ETL process and techniques. Then, it explains in details the proposed methodology. Finally, test results of typical data-extraction tasks from four commercial ISs are reported.**

***Keywords*-ETL; Database; Trigger; Reverse Engineering; Data Warehouse; Information System; Information Retrieval; Process Mapping; Data Discovery**

I. Introduction

Data-driven Business Intelligence (BI) is emerging as a crucial tool for supporting decision making, providing insight, and ultimately bolstering organisational performance [1]. Its effectiveness lies on the enterprise capability to collect and leverage its data assets. In order to manage organisational processes and streamline their operations, modern enterprises rely heavily on specialised information systems such as Product Lifecycle Management system (PLM), Enterprise Resource Planning system (ERP), Assets Management System (AMS) and Content Management System (CMS) etc. Often different units/departments of an enterprise have dedicated IS. The integration of these heterogeneous systems into a coherent enterprise architecture is notoriously complicated and expensive, limiting the ability to develop an effective BI system. Integrating data from multiple heterogeneous sources to support analytical reporting is commonly referred to as data warehousing. In order to build a Data Warehouse (DW), the necessary pre-processing steps are commonly known as Extract, Transform, and Load (ETL). ETL is a data integration framework that involves extracting data from several data management systems, cleaning it, transforming it according to business needs, and finally loading it into a single aggregated database.

Before starting any ETL process, DW designers must be familiar with structure and semantic of data to be imported from source ISs. However, DW designers are usually not experts in the specific domain, data and processes supported by the given IS [2]. Given the limited understanding of the domain, it is quite hard to locate the right set of data from commercial ISs consisting large number of tables and relationships. Therefore, the DW designer must understand, at least partially, the data model and the processes of the ISs he/she is planning to import. Given the limited understanding of the domain,it is quite hard to locate the right set of data. This is especially true for commercial ISs consisting of a large number of tables and relationships. In some cases, data extraction becomes almost impossible as some vendors deliberately obfuscate their data model in order to prevent reverse engineering [3]. In addition, relationships between tables are not always explicit as they are often encapsulated within the application logic [4].

The main objective of this paper is to develop an effective and generally-applicable technique to simplify the extraction step of any ETL process. The proposed technique guides ETL experts in the identification of the desired data from a pool of tables of an arbitrarily complex database. This technique is applicable to any Relational Database Management System (RDBMS) supporting Structured Query Language (SQL) and triggers. The vast majority of ISs implements their data model using RDBMS. Therefore, the proposed method covers the majority of the ISs currently available on the market. Before outlining the proposed methodology, this paper first provides a state of the art review of ETL processes and practices. Then, the proposed method and its implementation details are presented together with the testing results derived from extracting data out of four commercial ISs. Finally, discussion and conclusion are presented.

II. Current state of ETL process

The ETL is an end to end process taking data from one or more system(s), load it into a single unified system after processing and transformation [5]. Often the end result of an ETL is the creation of what is commonly known as DW. In general, DW projects are quite complex and are considered strategic in any data-driven enterprise. Given the industrial/commercial relevance of the problem, significant research has been carried out. In particular, the majority of the research works of this field is focuses on two main aspects:

- Conceptual modelling of ETL process.
- Mapping data-items from various systems into the DW.

A. ETL Conceptual Modeling Literature

One of the first graphical formalism for modelling ETL processes was introduced by Golfarelli et al. in [6] under the name of Dimensional Fact model (DFM). The conceptual model representation of DFM is called dimensional scheme. This model consists of a set of fact schemes whose basic elements are facts, dimensions and hierarchies. In this work, Golfarelli et al. also propose a semi-automated methodology to develop conceptual model based on pre-existing database schemas describing the enterprise relational database(s). Similarly, Vassiliadis et al. [7] propose a conceptual model dedicated to the design and documentation of the DW refreshment process, which includes activities such as monitoring, extracting transforming, integrating, cleaning data sources, and finally loading them into the DW. Their method relies on specifications stored in an object-oriented metadata repository. Trujillo et al. [8] propose an object-oriented approach for designing DW conceptual models, leveraging on the Unified Modeling Language (UML) as formalism for their description. This approach introduces a set of minimal constraints and extensions to UML for representing multidimensional properties. Later in 2003, Lujan-Mora and Trujillo in [9] extended their research and proposed a framework for the design of the DW back-stage based on the key observation that ETL steps fundamentally involves dealing with the specificities of information at very low levels of granularity including transformation rules at the attribute level. This model extends again UML to model attributes as first-class citizens. Liu et al. in [10] present a scalable dimensional ETL framework, ETLMR, based on MapReduce. ETLMR has built in support for DW-specific operations, such as, star schemas, snowflake schemas and slowly changing dimensions. This framework enables ETL developers to construct scalable MapReduce-based ETL flows efficiently and with very few lines of code. Caruccio et al. in [11] present the Conceptual Data Integration Language (CoDIL), a visual language providing conceptual level visual mechanisms to manipulate and integrate data sources. It provides a visual annotation technique to select concepts or sub-schemas to be compared, and a set of icon operators that can be applied to them in order to specify how they are mapped onto constructs of a reconciled schema.

B. Mapping Literature

Mapping is another widely researched topic in ETL. Several techniques have been proposed for mapping source schemas of the imported databases to the unified destination schema of the DW. Madhavan et al. in [12] back in 2001, proposed a schema integration algorithm called Cupid that discovers mappings between schema elements based on their names, data types, constraints, and schema structure. Similarly, Fagin et al. in [13] proposed an algorithm to address the needs of two major information integration problems: data integration and data exchange. The proposed algorithm discovers queries over the source, queries over the target, and a present specification of their relationship between them. In addition, the algorithm also generates query for data exchange between the source and target schemas. Later in 2002, Rifaieh and Benharkat in [14] developed a model based on mapping expressions and proposed mapping guidelines, in which queries are used to represent the mapping between source and target system. Jiang et al. in [15] and Touma et al. in [19] applied the concept of domain ontology in order to guide the mapping process. Jiang et al. leverage domain ontology for discovering data sources, defining data transformation, and eliminating heterogeneity between different database schemas. In this approach, the domain ontology is embedded in the metadata of the DW. Hence, data records in the relational database can be directly mapped into ontology classes using the OWL (Web Ontology Language). Whereas, Touma et al. in [16] proposed a semi-automatic ontology construction technique which can be used to support data integration while eliminating the requirement of a predefined target schema. This technique integrates existing ontology extraction, matching and merging techniques into a single end-to-end system. This approach extends common functionality of ontology extraction techniques with the generation of mappings between the extracted ontologies and the sources.

The integration techniques presented are semi-automatic methods and based on declarative mapping, ontologies, data types or schema structures. Most of these techniques rely on the fact that DW designers have intimate knowledge of the data and schemas they are importing which in many cases is not true. This is obviously paramount for a correct mapping in the DW destination schema. In order to understand these source schemas, in some of these techniques, data engineers had to typically rely on UML or Entity Relation (ER). Unfortunately, UML diagrams do not provide detailed understanding of data models, while ER diagrams have a serious practical limitation in their inability to cope with complexity [17], [18]. In addition, ER model lacks explicit abstraction mechanisms for managing the size and complexity of real world data models [19]. With large numbers of entities, complexity quickly becomes overwhelming and as a result, data models become very difficult for people, particularly non-technical users to understand [20], [18]. As it can be seen from literature, most of these techniques provides mechanism to explore data model via UML, ER diagram, schema but they lack a method to understand the source data models that need to be mapped. This paper focus exactly on this neglected aspect. Guiding data engineers in learning/understanding the structure and semantics of the data, they are importing into their DW.

III. Proposed Methodology

The main objective of this paper is to develop a generally applicable technique to locate and extract desired data from any IS. In particular, the proposed technique is applicable to any RDBMS that supports triggers. The vast majority of ISs implement their data model using an RDBMS. Therefore, the proposed method covers the majority of the ISs currently available on the market. This section outlines the proposed methodology and its rationale.

A. Rationale

The rationale behind the development of the proposed methodology is based on the fact that the most complex task of any ETL process is understanding the structure and semantics of data to be imported. This step is often given as pre-requisite in literature, but in reality the personnel performing the ETL process have, in many cases, a very limited knowledge regarding the data they manipulating. Grasping even partially the data model of a commercial IS can be very frustrating and it is quite easy to get lost in hundreds of tables with hundreds of columns. The typical setup of an ETL process is often far from ideal and the following conditions generally occur:

- ETL experts usually are not aware which data and processes are supported by the IS from which they are extracting the data.
- Domain experts, such as the IS end-users, can provide insight when it comes to data and processes, but they are seldom technically competent to understand the underlying IS architecture and database schema.
- Some information systems deliberately obfuscate their data model in order to prevent reverse engineering [3].
- The sheer size of the database, in terms of the number of tables and relationships, often makes it difficult to browse and determine where certain data is stored [21].
- Relationships between tables are not always explicit as they are encapsulated within the application logic [4].

Given the considerations above, the proposed methodology leverage on the end-users knowledge about the specific IS for guiding the ETL expert in discovery of the relevant pool of tables from the underlying IS data model. In other words, the proposed technique maps the processes supported by IS with its underlying database schema. We named the proposed methodology Data Model Logger (DML).

B. Method

In order to bridge the knowledge gap between ISs end users and ETL experts, the methodology proposes to exploit from the component of the IS that both understands, the graphical user interface. The developed system record click-by-click actions performed in the user interface and associate them with related changes happening in the RDBMS. The main steps of this process are displayed in Figure 1.

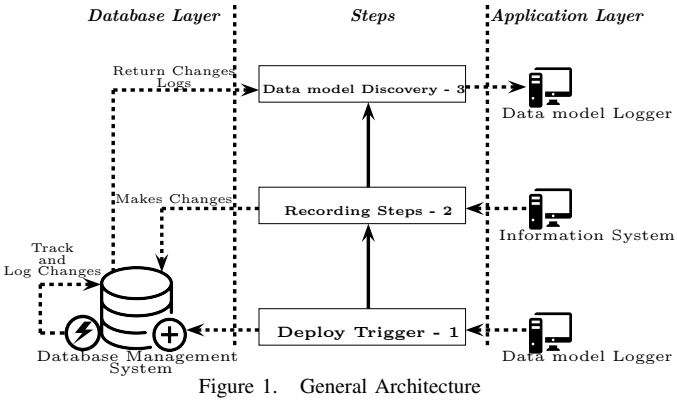

Figure 1. General Architecture

1) **Deploy Trigger**: Triggers are procedure that are automatically executed in response to certain events such as INSERT, UPDATE, or DELETE on a particular table or view in a database. Triggers are mostly used for maintaining the integrity of the information in the database. However, in the proposed solution, triggers are used to track changes made by IS to its data model. For each table, three types of triggers are deployed (one for each operation: Insert, Update, and Delete) according to the steps reported in Algorithm 1.

2) **Recording Graphical User Interface (GUI) Interaction**: In this step, the end-user performs a task on the IS'GUI, involving inputting or reading data which is relevant for the creation of the final DW. Assuming that the final DW wants to aggregate all work-orders for a given duty; the typical tasks that could be performed in the GUI are "creating a work-order", "updating work-order details", etc. The developed system keeps track of every clicks on the GUI and puts them in relationship with the changes happening in the RDBMS of the IS.

3) **Data model discovery**: Once a task has been completed, the developed system outputs an HTML report with screenshot of GUI for every action that had occurred in IS with the list of associated tables and columns that has been changed in underlying database. The ETL expert can review the report to understand/locate where data is stored in the database. In this phase, the knowledge gap between IS end user and ETL expert is finally bridged and the information described in the HTML report will be crucial for performing the extraction phase of the ETL process.

IV. Implementation and Case Studies

This section presents an implementation of the proposed methodology. First, design choices and implementation details are presented. Then, the application and its underlying methodology are tested in four different use cases, concern-

Algorithm 1: Trigger creation Procedure

Input : Database Credentials
Output: Successful creation of triggers and temporary tables to track changes in deployed database schema

$TableList \leftarrow get\_TableLists(IsTemp = False)$
/* Return list of tables for creating triggers */
foreach $Table \in TableList$ **do**
  $ColumnList \leftarrow ColumnLists(TableName)$
  /* Return list of columns for particular table */
  $CreateTemporaryTables(TableName, ColumnList)$
  /* Creates temporary table to log changes. */
  foreach $Action \in (Insert, Update, Delete)$ **do**
    $Query \leftarrow generateTriggerQuery(Table, ColumnList, DatabaseType, Action)$ /* Return SQL query for creating trigger based on action and databasetype */
    $DeployQuery(Query)$
    /* Deploy trigger in the database for logging track changes */
  end
end

ing data extraction from four different commercial ISs.

A. Implementation

As already mentioned, the core concept of the proposed approach is based on tracking database changes while performing operations in the front-end, typically a GUI of an IS. Tracking changes helps to understand how the data model is manipulated by IS, providing valuable insights regarding the underlying database schema.

A natural choice to track events/changes within a RDBMS is to use triggers. Triggers are procedural code that are automatically executed in response to certain events such as INSERT, UPDATE, or DELETE on a particular table or view in a database. Triggers are generally used for maintaining the integrity of the information in the database. However, in the proposed solution, triggers are used to track changes made by the IS to its data model.

The developed application guides the user, which in this case is the data engineer, in three main steps following the conceptual workflow presented in Figure 1. The software has been developed in Java, plus JavaFX for the GUI and Java database connectivity (JDBC) for communicating with the database. Our current implementation supports four major RDBMS specifically, Oracle, MSSQL, MYSQL and PostgresSQL.

1) Step 1 - Deployment Step: The first step deploys database triggers and temporary tables in the RDBMS of IS that will be used to track the changes made by the Controller of the IS. In this step, the user must provide valid database credentials (with permission to create and drop triggers and tables) of underlying database (see Figure 2). Once the user click on the Deploy Database Logger button, the solution dynamically generates necessary scripts to create temporary tables and triggers on tables and deploys them in database. For each table, three types of triggers are created, specifically once of each operation SQL (INSERT, UPDATE, DELETE). This procedure automatically creates triggers on existing tables based on the table information available on the database's internal views, such as DBA_TAB_COLUMNS for Oracle (other databases have different internal system views with the same purpose but with different names).

2) Step 2 - Recording Step: After successfully deploying the trigger, the user will provide a task name describing the operation (e.g. create maintenance request, create work order, publish a post, etc.) that the IS end user is about to perform on the IS. At this point the data engineer will click on the START button (see Figure 3) and the IS end user will then start performing the planned task by operating on the GUI of an IS. Every change made by the IS Controller on its database is logged in our temporary tables.

3) Step 3 - Changes Log Step: Once the IS end user complete the task, the data engineer can click on the STOP button (see Figure 4). At this point in the Results tab (see Figure 4), it is possible to retrieve the list of updated table, their columns and the specific row data that has been created or updated. In addition, when executed on Windows operating system, the developed solution lunches Process Step Recorder (PSR) at the beginning of each task. This utility software takes screenshot of every click performed and their timestamp, documenting click-by-click a specific task.

B. Case Studies and Results

The case studies presented in this section aim at validating the feasibility and general applicability of our methodology. For this reason, we selected four different ISs designed for different purposes and implemented their data model

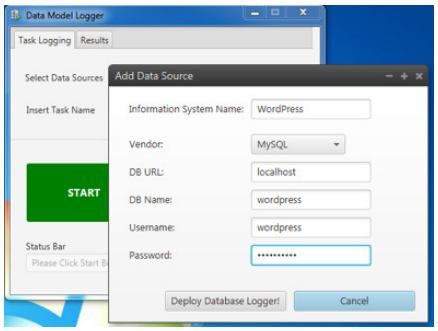

Figure 2. Deployment Step

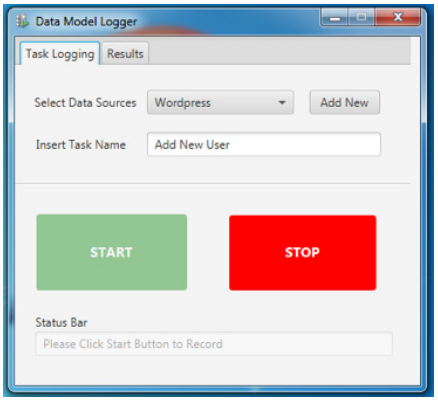

Figure 3. Recording Step

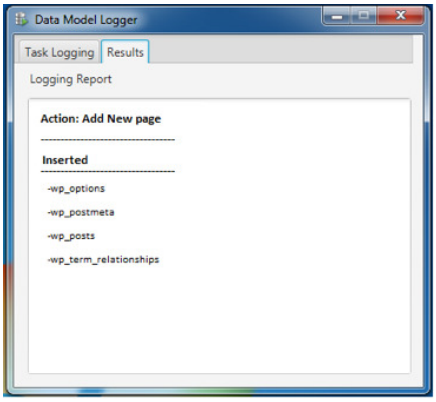

Figure 4. Changes Log Step

using different RBDMSs. The list of selected ISs and their underlying databases is shown in Table I. In these case studies, we omitted description of each task in studied ISs; here the main objective is to highlight feasibility and applicability of our purposed methodology to find relevant table and column in database associate with each task. The ISs are as follows:

Information System	Database	Number of tables
openMAINT	PostgresSQL	581
OpenCms	Oracle	40
SharePoint	MSSQL	2827
WordPress	MYSQL	12

Table I
LIST OF IS AND THEIR CORRESPONDING DATABASE

1) Wordpress: WordPress is a free open-source CMS based on PHP and MySQL. It is one of the most popular CMS, used by more than 27.5% of the top 10 million websites as of February 2017 [22]. Typical tasks in WordPress include create and publish posts, manage users, install plugins etc. The full list of tested operations (Tasks) and the related changes in the underlying database is reported in Table II.

2) openMAINT:: openMAINT is an asset management application for managing buildings, installations, and maintenance activities. It is a complex system with more than 581 database tables deployed in PostgreSQL. Common openMAINT tasks and corresponding database changes are listed in the Table III.

3) SharePoint:: SharePoint is the most widely used enterprise collaboration and document management platform. Developed by Microsoft, naturally leverage on MSSQL Server database. SharePoint is one of the most advanced general purpose CMS, providing sophisticated Managed Metadata Service (MMS) for improving filtering while performing searches. The full list of tested operations (Tasks) and the related changes in the underlying database is reported in Table IV.

4) OpenCMS:: OpenCMS is a powerful web based CMS designed and developed to create and maintain website quickly and efficiently. OpenCMS can be deployed with different database systems such as MySQL, and Oracle. In this case study, OpenCMS was deployed with Oracle 11 Express edition. The full list of tested operations (Tasks) and the related changes in the underlying database is reported in Table V.

V. DISCUSSION AND CONCLUSION

Understanding complex data models of commercial ISs is generally a complicated and time-consuming process. In many commercial systems, getting data out is a cumbersome task as some ISs deliberately obfuscate data model relationships and tables/columns name. In addition, only a small portion of the IS database might contain relevant data to be imported in a hypothetical DW. For instance, openMAINT is an asset management system supporting complex workflows for streamlining maintenance activities of building, equipment, heating systems etc. Its data model counts 581 tables, but during our tests while performing typical openMAINT tasks, only 20 tables were actually modified. Understanding a pool of 20 tables with a clear relationship between performed operation and modified tables/columns reduces dramatically the complexity. Often ISs provide APIs for accessing programmatically the data they manage. However, the complexity of these APIs is generally directly proportional with the complexity of the ISs and by reflection its database schema. Therefore, using these APIs correctly might be an overwhelming task as much as understanding the entire data model. In addition, the main benefit of the proposed methodology is the ability to leverage the knowledge of the IS end users, for identifying and performing a set of tasks (directly on the GUI of

Tasks	SQL Insert	SQL Update	
	Tables Name	Tables Name	Columns Name
Add New User	wp_usermeta wp_users	- - -	- - -
Add New page	wp_options wp_postmeta wp_posts wp_term_relationships	- - - -	- - - -
Install Plugin(Buddy Press)	wp_options	wp_options	option_value
Add New post	wp_options wp_postmeta wp_posts wp_term_relationships	wp_posts	Guid post_name post_date post_title post_status post_modified post_date_gmt post_modified_gmt
		wp_term_taxonomy	Count
		wp_options	option_value
		wp_postmeta	meta_value
post a comment	wp_comments	-	-
Approve Comment		wp_comments	comment_approved
		wp_posts	comment_count

Table II
CHANGES LOG FOR WORDPRESS

Tasks	SQL Insert	SQL Update	
	Tables Name	Tables Name	Columns Name
Create Maintenance Activity	maintenanceactivity map_maintenancegroupmaintenanceactivity map_servicecategorymaintenanceactivity map_servicemaintenanceactivity map_suggestedassigneeactivity map_teammaintenanceactivity	maintenanceactivity	BeginDate
Update Maintenance Activity (ActivityStatus)	maintenanceactivity maintenanceactivity_history map_maintenancegroupmaintenanceactivity map_servicecategorymaintenanceactivity map_servicemaintenanceactivity map_suggestedassigneeactivity map_teammaintenanceactivity	MaintenanceActivity - -	ActivityStatus User
Add Rent Contract	map_mainpartycontract rent		
Update Rent Contract (StartingDate)		rent	BeginDate StartingDate
Create Mainte-nance Request	shkactivities shkactivitydata shkassignmentstable shkprocessdata shkprocesses shkprocessrequesters	objectid	next
		shkcounters	Objectversion the_number
		shkprocesses	Objectversion State started laststatetime
		shkactivities	Objectversion State laststatetime
		MaintenanceRequest	Description BeginDate Type Details
		shkprocessdata	Objectversion variablevalue

Table III
CHANGE LOG FOR OPENMAINT

IS) involving the data items that should be imported in the DW. The proposed method allows data engineers to pinpoint exact tables and columns of database that have been updated/inserted, while understanding the context (action

Tasks	SQL Insert	SQL Update	
	Tables Name	Tables Name	Columns Name
Create New Group	ECMChangeLog ECMGroup	- - -	- - -
Create New TermSet	ECMChangeLog ECMTermSet	- - -	- - -
Create Term	ECMChangeLog ECMTerm ECMTermLabel ECMTermSetMembership	- - - -	- - - -
Create New Group	ECMChangeLog ECMGroup	- - -	- - -
Update Group	ECMChangeLog	ECMGroup	Name Description LastModifiedTime
Update Term	ECMChangeLog ECMTermLabel	ECMGroup	LastModifiedTime UseCount Owner IsDeprecated

Table IV
CHANGES LOG FOR SHAREPOINT

and data semantics) in which that data was created/modified.

In conclusion, the proposed methodology simplifies the extraction and mapping of complex data sources into a DW. This technique is applicable to any IS using a RDBMS and it can be easily integrated with any existing ETL process.

VI. FUTURE WORK

From the provided case studies, we can conclude that DML is capable of mapping IS activities with the changes that are made by application functions in its underlying database. Currently, the solution supports four databases, namely, Oracle, MSSQL, PostgreSQL, and MySQL. In near future, we are planning to add support for additional database, such as DB2 and SQLite. Another area that we want to improve is results report. Currently, our report displays a list of changed tables and related affected columns. In next iteration, we plan to improve it by suggesting relationships among tables which are based on changes made on tables by IS activities. Likewise, we also plan to add SQL query to extract relevant data from database.

ACKNOWLEDGMENT

The research leading to this publication is supported by the Digital, Internet, Materials & Engineering Co-Creation (DIMECC) S4Fleet program and bIoTope Project funded by the Horizon 2020 program (Call ICT30).

REFERENCES

[1] C. Vercellis, *Business intelligence: data mining and optimization for decision making*. John Wiley & Sons, 2011.

[2] S. Viaene, "Data scientists aren't domain experts," *IT Professional*, vol. 15, no. 6, pp. 0012–17, 2013.

[3] MarcL. (2010) What is plm data obfuscation and i why should i care? [Online]. Available: http://www.aras.com/Community/blogs/aras_corporate_blog/archive/2010/08/05/what-is-plm-data-obfuscation-and-i-why-should-i-care.aspx

[4] H. Pieterse and M. Olivier, "Data hiding techniques for database environments," in *Advances in Digital Forensics VIII*. Springer, 2012, pp. 289–301.

[5] H. Agrawal, G. Chafle, S. Goyal, S. Mittal, and S. Mukherjea, "An enhanced extract-transform-load system for migrating data in telecom billing," in *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*. IEEE, 2008, pp. 1277–1286.

[6] M. Golfarelli, D. Maio, and S. Rizzi, "The dimensional fact model: a conceptual model for data warehouses," *International Journal of Cooperative Information Systems*, vol. 7, no. 02n03, pp. 215–247, 1998.

[7] P. Vassiliadis, A. Simitsis, and S. Skiadopoulos, "Conceptual modeling for etl processes," in *Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*. ACM, 2002, pp. 14–21.

[8] J. Trujillo, M. Palomar, J. Gomez, and I.-Y. Song, "Designing data warehouses with oo conceptual models," *Computer*, vol. 34, no. 12, pp. 66–75, 2001.

[9] S. Luján-Mora and J. Trujillo, "A comprehensive method for data warehouse design," 2003.

[10] X. Liu, C. Thomsen, and T. B. Pedersen, "Etlmr: a highly scalable dimensional etl framework based on mapreduce," in *Transactions on Large-Scale Data-and Knowledge-Centered Systems VIII*. Springer, 2013, pp. 1–31.

[11] L. Caruccio, V. Deufemia, and G. Polese, "Visual data integration based on description logic reasoning," in *Proceedings of the 18th International Database Engineering & Applications Symposium*. ACM, 2014, pp. 19–28.

[12] J. Madhavan, P. A. Bernstein, and E. Rahm, "Generic schema matching with cupid," in *VLDB*, vol. 1, 2001, pp. 49–58.

[13] R. Fagin, L. M. Haas, M. Hernández, R. J. Miller, L. Popa, and Y. Velegrakis, "Clio: Schema mapping creation and data exchange," in *Conceptual Modeling: Foundations and Applications*. Springer, 2009, pp. 198–236.

Tasks	SQL Insert	SQL Update	
	Tables Name	Tables Name	Columns Name
Create a Page	cms_offline_structure cms_offline_resources cms_offline_properties cms_offline_contents_relationships	cms_offline_resources	project_lastmodified resource_type resource_flags resource_size date_content sibling_count date_lastmodified user_lastmodified
		cms_offline_properties	property_value
		cms_offline_structure	resource_id parent_id resource_path rstructure_stat resource_path date_released date_expired
		cms_userdata	data_type
Publish a Page	ccms_online_structure cms_history_projects cms_history_resources cms_online_resources cms_history_properties cms_history_structure cms_online_properties cms_publish_history cms_publish_jobs cms_contents cms_user_publish_list	cms_contents	online_flag
		cms_publish_jobs	project_id project_name user_id publish_locale publish_flags resource_count enqueue_time start_time finish_time
		cms_userdata	data_type
		cms_users	user_firstname user_lastname user_email user_lastlogin user_flags
		cms_online_structure	resource_id parent_id resource_path structure_state date_released date_expired

Table V
CHANGES LOG FOR OPENCMS

[14] R. Rifaieh and N. A. Benharkat, "Query-based data warehousing tool," in *Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*. ACM, 2002, pp. 35–42.

[15] L. Jiang, H. Cai, and B. Xu, "A domain ontology approach in the etl process of data warehousing," in *e-Business Engineering (ICEBE), 2010 IEEE 7th International Conference on*. IEEE, 2010, pp. 30–35.

[16] R. Touma, O. Romero, and P. Jovanovic, "Supporting data integration tasks with semi-automatic ontology construction," in *Proceedings of the ACM Eighteenth International Workshop on Data Warehousing and OLAP*. ACM, 2015, pp. 89–98.

[17] P. Feldman and D. Miller, "Entity model clustering: Structuring a data model by abstraction," *The Computer Journal*, vol. 29, no. 4, pp. 348–360, 1986.

[18] R. Kimball and M. Ross, "The data warehousing toolkit," *John Wiley& Sons, New York*, 1996.

[19] R. Weber *et al.*, *Ontological foundations of information systems*. Coopers & Lybrand and the Accounting Association of Australia and New Zealand Melbourne, 1997.

[20] D. Moody, "A practical methodology for the representation of large data models," in *Proceedings of the Australian Database and Information Systems Conference*, 1991.

[21] Y.-G. Kim and G. C. Everest, "Building an is architecture: Collective wisdom from the field," *Information & Management*, vol. 26, no. 1, pp. 1–11, 1994.

[22] (2017) W3techs. [Online]. Available: https://w3techs.com/technologies/overview/content_management/all/