Reverse Engineering in Data Integration Software

Vlad DIACONITA
The Bucharest Academy of Economic Studies
diaconita.vlad@ie.ase.ro

Integrated applications are complex solutions that help build better consolidated and standardized systems from existing (usually transactional) systems.
Integrated applications are complex solutions, whose complexity are determined by the economic processes they implement, the amount of data employed (millions of records grouped in hundreds of tables, databases, hundreds of GB) and the number of users [11].
Oracle, once mainly known for his database and e-business solutions has been constantly expanding its product portfolio, providing solutions for SOA, BPA, Warehousing, Big Data and Cloud Computing. In this article I will review the facilities and the power of using a dedicated integration tool in an environment with multiple data sources and a target data mart.
***Keywords**: ODI, reverse engineering, SOA, data mart*

1 Data integration software

There are many software packages that can aid to system integration. One of them is the Oracle Data Integrator (ODI) that has its roots in the acquisition of Sunopsis by Oracle in 2006. The primary use of ODI is to move and transform data from one place to another so there are many other Oracle products that are benefiting and thus creating the need of using this tool in many organizations.
Like shown in [7] data Warehouses and data marts (a subset of the data warehouse that is usually oriented to a specific department) are data-intensive systems that are used for analytical tasks in businesses such as analyzing sales/profits statistics, cost/benefit relation statistics, customer preferences statistics, etc. The idea of a data warehouse is to extract data from operational databases and to store them separately. The justification for this approach is that OLAP largely deals with condensed data, thus does not depend on the latest updates by transactions. Furthermore, OLAP requires only read-access to the data, so the separation of the data for OLAP from OLTP allows time-consuming transaction management to be dispensed with.
The need for data warehouse usually comes from the fact that the operational systems cannot be overload with the additional queries required by the business intelligence needs. Integrating data from multiple, usually heterogeneous sources, is a problem addressed by ODI.
Linking SOA with ODI and Data Warehousing (DW) can be beneficial. From a business perspective the key word in dealing with SOA is definitely flexibility. Companies must be able to keep pace with the rapid changing conditions of the business environment. In the same time the trend in IT architectures leads toward an integrated model by building business processes that span multiple operational systems and by enabling interoperability between legacy systems and newly developed systems [3].
As shown in [4] SOA has the merit to introduce a new kind of technological "democracy" where the application systems are considered a federation per se, thus opening the doors to a new kind of logical distributed computing approach where the technological platforms are downgraded to the implementation or physical level.
Most SOA efforts have centered on transaction systems, but data warehousing can benefit from SOA with the ability to join various actions (services) from different areas of the DW to create

composite applications or common services. The services that are part of a data warehouse such data extraction, transformation, loading, querying or updating should be part of the SOA from the start. This should make more comprehensive business intelligence possible, and could assist in the development of fully integrated SOA [2]. The usage of SOA makes data location of little importance to its users so using it in conjunction with ODI processes and transformations makes lot of sense. SOA can enable an abstract layer that makes data available inside an enterprise using homogeneous services.

ODI can also have a role in normalizing data definition so different applications reference the same data. It can also have a role in processing Big Data by delegating and distributing processing.

2. ETL and ELT

As an alternative to ETL, ODI proposes ELT which extracts data from the source, loads it into the target and processes there by using SQL the needed transformations.

This approach exploits database optimizers as opposed to transformation that is performed in-flight or requiring a separate intermediary. The ELT approach directly impacts performance and has proven to make data loading fast, efficient and incredibly reliable [9].

An ELT has the ability to manage a staging area, generate code and execute instructions on target systems but also on source systems, such systems that are being managed by any DBMS.

The components of ODI architecture are the repository, the studio, the graphical interface of the software, the agents and the console. In SOA, repositories are used to manage services and support service discovery at runtime. Usually, there is a process needs a data access service (DAS) to execute a query usually against a database. DAS are variations of the ordinary service concept; they are more data-intensive and designed to expose data as a service [5]. In an ELT solution the data is not stored in the repository; it is moved directly to the target. The repository is usually stored in a schema of an existing database and is composed of the master repository for the sensitive data and the work repository for the data needed by the developers. In a production environment an execution repository is also present. It stores only the operational metadata. The exchange of data can be done through versioning or by importing or exporting XML files. There are two types of agents: the standalone agent that can be installed on any platform and the JEE agent that runs on a Weblogic server. Usually the JEE agents has the role to distribute execution requests and balance load across different other agents which are usually standalone agents. A strategy using only standalone agents is also possible.

The key elements of ODI are Execution Contexts, Knowledge Modules, Models, Interfaces, Packages and Load Plans.

For assuring independence from the physical location of the data, logical schemas can be used, at execution this are translated into physical ones so the maintenance of the connection parameters, location of the databases, and schema names is entirely independent of the code itself.

Metadata can be imported using knowledge modules from applications, where objects usually are representation of the data or from databases using models. After it's imported, metadata can be enhanced in ODI (for example by adding constraints). Also new metadata can be created.

Another key element in ODI is the interfaces where the transformations are built. An interface contains among others description, mappings and flows.

Packages put together elements such as interfaces, variables and procedures. They are compiled into scenarios which execution can be organized with load plans.

3. Reverse-engineering the model metadata

Like shown in [1], to use ODI, first we declare a new data server in the ODI Physical Architecture and then a reference to a Physical Schema located on that server that holds the business data. We can also construct a work schema for every physical schema to store temporary data.

A Physical Schema definition needs to be associated with a Logical Schema name which will be exclusively used to build ODI models. The models are abstracted, independent of the data source but seem homogeneous to the developer. Usually they are built by reverse-engineering the structural data, that coming from databases, flat files, XML files or different ERP systems. Different other elements, such as structural integrity data can be added to the captured metadata.

Like shown in [6], the **Reverse-Engineering Knowledge Modules** role is to perform customized reverse engineering for a model. It connects to the application or metadata provider then transforming and writing the resulting metadata into Oracle Data Integrator's repository. The metadata is written temporarily into the SNP_REV_xx tables. The RKM then calls the Oracle Data Integrator API to read from these tables and write to Oracle Data Integrator's metadata tables of the work repository in incremental update mode (figure 1).

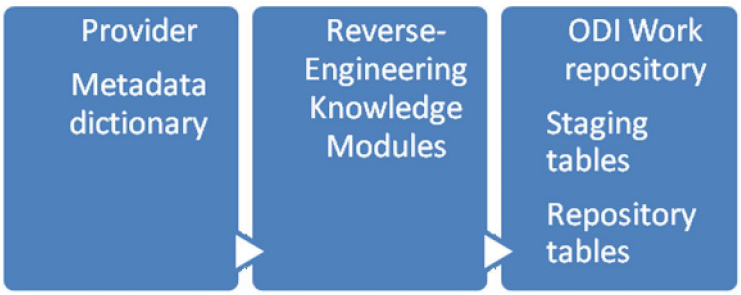

Fig. 1. RKM

As shown in [7], Oracle Data Integrator implements five different types of KMs. Each of them covers one phase in the transformation process from source to target. The three most important types of modules are the *integration knowledge module* (**IKM**), the *loading knowledge module* (**LKM**), and the *check knowledge module* (**CKM**). As explained in [1] when a Knowledge Module has been previously imported into the parent project and applied to the interface target and the interface is subsequently executed, it is the steps within the **IKM** that determine the what, how, and when data is moved into the target data store. **LKMs** load data into the staging area from other servers. If the source data is in the same server as the staging area then **LKM** is not needed. **CKMs** are used to check and enforce data integrity through testing conformance to constraints and references, either statically on data tables on source or target systems, or dynamically during the process of a data flow defined in an ODI interface.

To reverse engineer, after defining the topology, we create the model like shown in figure 2.

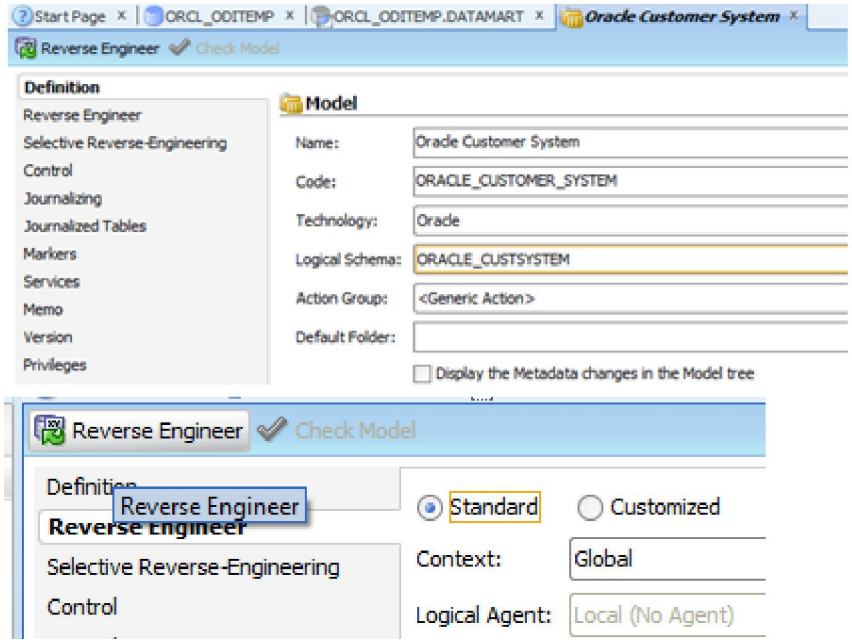

Fig. 2. Create the model

After configured the ODI representations of our data objects an interface can be built to move and transform the data into the data mart, like shown in figure 3.We can use the Automatic Mapping and also do manual mappings and transformations using SQL code and apply some timestamps useful for audit purposes. The transformed data will be loaded from an Oracle source into a target Oracle data mart.

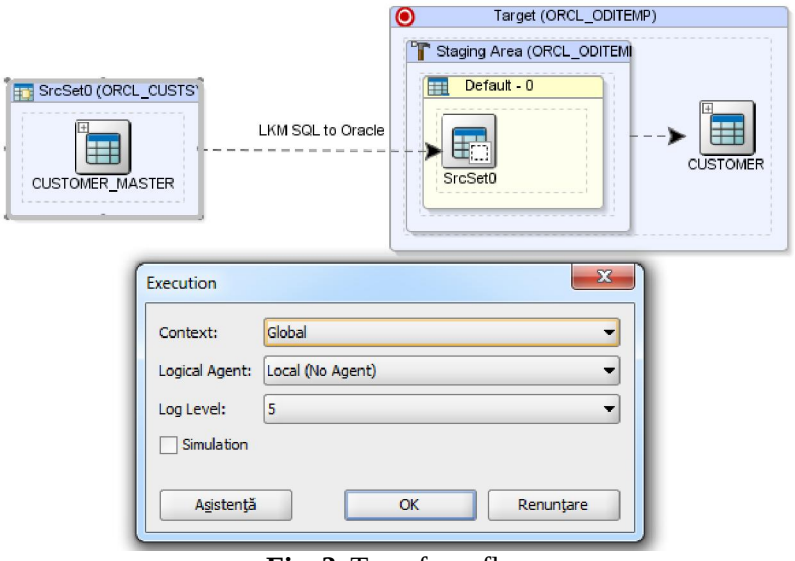

Fig. 3. Transform flow

We can add some additional complexity by introducing joins and lookups, heterogeneous data sources and data aggregation. And we can do this with multiple source databases usually using different JDBC connectors (figure 4).

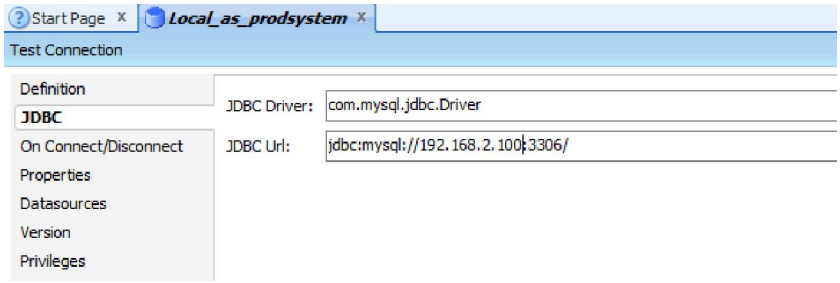

Fig. 4. MySql JDBC Driver

In the example shown in figure 5 we join three sources (actually we have 2 instances of the same data source table) and link them to the target data mart.

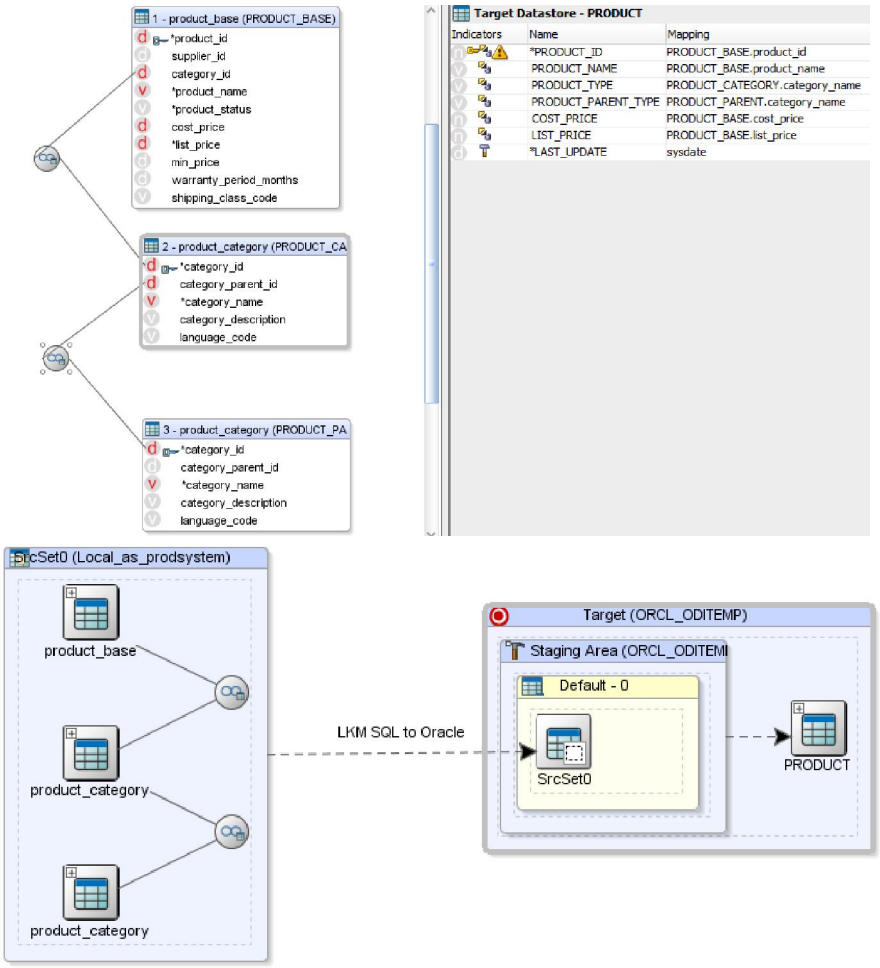

Fig. 5. Join transform flow

Like shown in figure 6, an interface can be built to move data using JDBC from a third-party DBMS to an Oracle data mart.

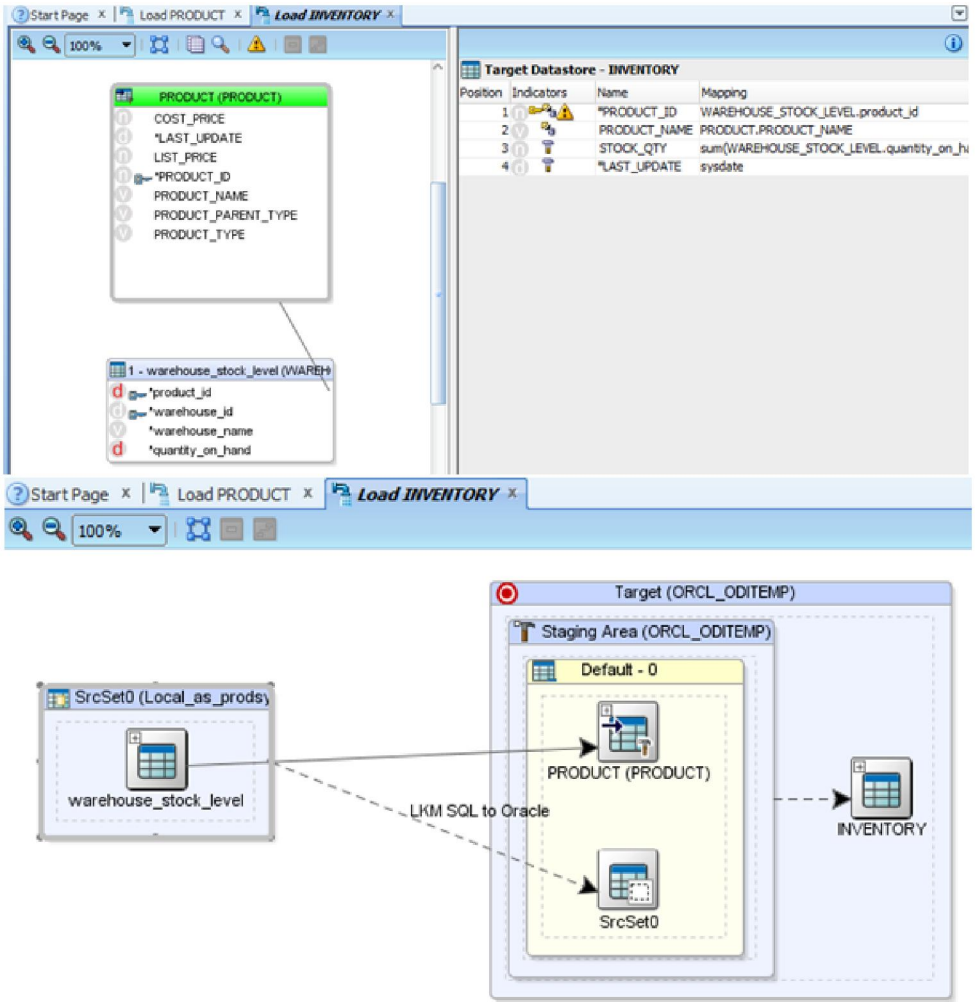

Fig. 6. Lookup transform flow

As shown in [10] eXtensible Markup Language (XML) is a platform-independent format for representing data and was designed as a standard for information exchange over the Internet. XML enables easy exchange of information, which allows interoperability between applications due to data encapsulation with metadata. To use XML inside ODI a JDBC driver is required which is available out-of-the-box (figure 7).

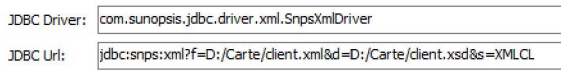

Fig. 7. XML JDBC

Let's look at the following XML file:

```
<CLIENT>
  <ID_CLIENT>10</ID_CLIENT>
  <FIRST_NAME>Jan</ FIRST_NAME >
  <LAST_NAME>Roberts</ LAST_NAME >
  <CREDIT>600</ CREDIT >

<EMAIL>Ishwarya.Roberts@LAPWING.COM</E
MAIL >
  <DATE_OF_BIRTH>21-MAR-44</
DATE_OF_BIRTH >
  <CIVIL_STATUS>single</ CIVIL_STATUS
>
  <SEX>F</SEX>
  <INCOME >G: 130,000 - 149,999</
INCOME >
 </CLIENT>
```

This would reverse-engineer to a table called CLIENT that has the following columns: ID_CLIENT, FIRST_NAME, LAST_NAME etc. The mapping is shown in figure 8.

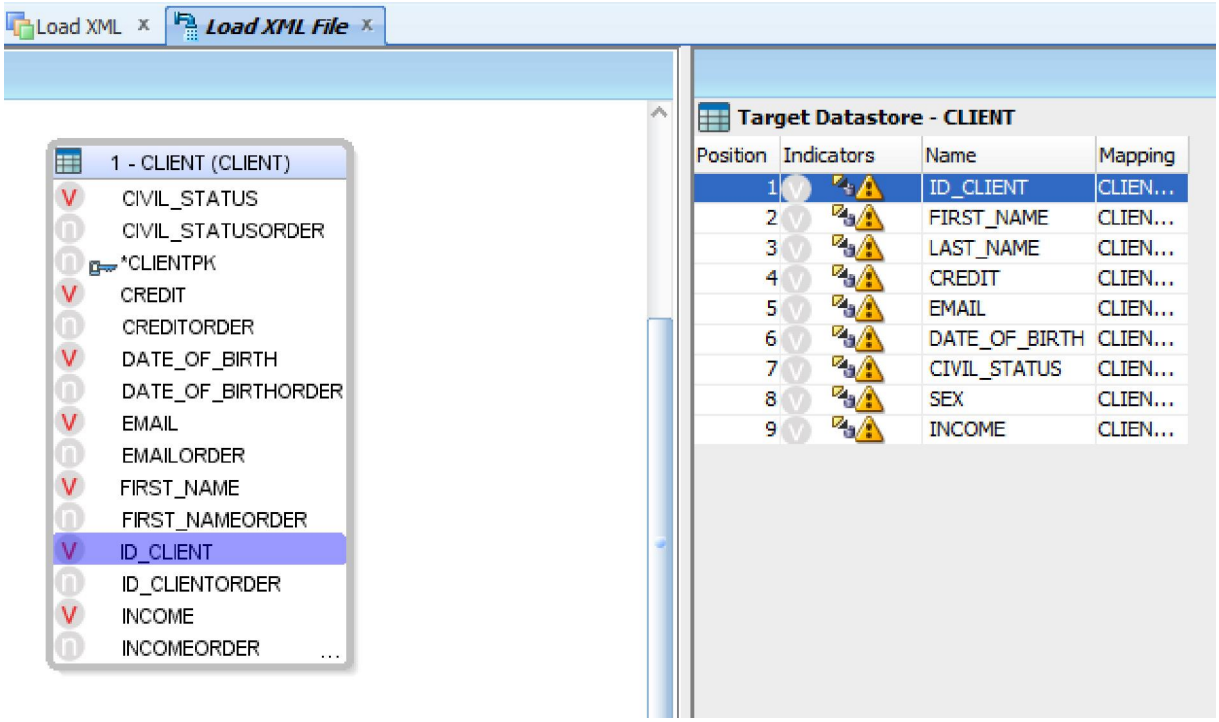

Fig. 8. XML-Relational Mapping

Conclusions

Data integration is very often a necessity in bigger projects. Using an integrated tool can be much more powerful and useful in projects that imply using data from heterogeneous sources, targets, and applications. Such products provide great aid in integrating databases, ERPs, CRMs, B2B systems, flat files, XML data, LDAP, JDBC or ODBC. It also can help in cutting hardware costs through improved utilization and high-performance data integration. Using external services for data integration and by deploying data services and transformation services that can be integrated within an SOA infrastructure Also, SOA business processes can assign large data operations to Oracle Data Integrator by using web services.

References

[1] Peter C. Boyd-Bowman, Christophe Dupupet, Denis Gray, David Hecksel, Julien Testut, Bernard Wheeler, *Getting Started with Oracle Data Integrator 11g: A Hands-On Tutorial*, May 2012

[2]http://www.information-management.com/news/1065308-1.html

[3] Alexandra Florea, Anca Andreescu, Vlad Diaconita, Adina Uta, "Approaches Regarding Business Logic Modeling in Service Oriented Architecture", *Informatica Economică* vol. 15, no. 3/2011

[4] Cătălin Strîmbei, "Smart Data Web Services", *Informatica Economică* vol. 16, no. 4/2012

[5] M. Turner, D. Budgen, P. Brereton, "Turning software into a service", *Computer*, 36 (2003), pp. 38–44

[6]http://docs.oracle.com/cd/E15586_01/integrate.1111/e12645/intro.htm

[7] Uli Bethke, *Developing a Knowledge Module in Oracle Data Integrator*, 2009

[8] Jane Zhao, Hui Ma, "ASM-based design of data warehouses and on-line analytical processing systems", *Journal of Systems and Software*, Volume 79, Issue 5, May 2006, Pages 613–629

[9]http://www.oracle.com/technetwork/middleware/data-integrator/learnmore/odi-for-soa-wp-1555852.pdf

[10] Iuliana Botha, "Managing XML Data to optimize Performance into Object-

Relational Databases", *Database Systems Journal* vol. II, no. 2/2011

[11] Vlad Diaconita, "Hybrid Solution for Integrated Trading", *Informatica Economică* vol. 14, no. 2/2010

Vlad DIACONIŢA is a lecturer at the Department of Economic Informatics and Cybernetics, within the Faculty of Economic Cybernetics, Statistics from the Bucharest Academy of Economic Studies. He has graduated the faculty at which he is now teaching in 2005 and since 2010 holds a PhD in the field of Cybernetics and Statistics. He is the co-author of 3 books in the domain of economic informatics, and of more than 25 papers in journals and conference proceedings. He participated as team member in 3 research projects that have been financed from national research programs. He is a member of the IEEE Computer and INFOREC professional associations. Domains of competence: Database systems, Data warehouses, OLAP and Business Intelligence, Integrated Systems, SOA.