Abstract: This paper presents some results of a research project regarding the possibilities of integrating various data sources from multiple measuring systems existing in the Wind Power Plants in Romania. We consider XML as a format for receiving data, object-relational model with spatial capabilities for data management and representation and JDBC for data access. Thus, a data level’s architecture can be developed in order to query data through data mining, multidimensional or spatial analysis.

Key-Words: wind power plants, data model, data integration, JDBC, object-relational database, Geographic Information System

1 Introduction
Our research project’s objective is to develop a decision support system that can be used in the National Power Grid Company for a real time analyses and forecast of the energy produced by the wind power plants (WPP) installed in the country. There are three main areas that concentrate the WPP units: Banat, Moldova and Dobrogea. The total amount of power installed is about 500 Pi [MW], but the problem is that the energy produced fluctuates, as it’s shown in the figure 1, due to the principal meteorological factors: wind speed and direction.

Fig. 1 – Fluctuation of the WPP’s production between august 2010 and february 2011
The decision support system’s prototype will have a forecast component that will contribute to a better management, planning and scaling the energy sources (thermo, hydro and nuclear power) in case of the WPP’s variations. Some previous experiments presented in [1], [2] will be used for the developing of the forecast algorithms. But in order to apply these algorithms and also the DSS analyses, we must consider to organize and manage data in a client – server architecture with a database management system (DBMS) environment. After the analysis of the data sources gathered from the different WPP units we found heterogeneity of the data to be provided by Transelectrica National Power Grid Company (the national TSO - Transmission and System Operator). The heterogeneity of the sources is due to the diversity of wind turbines, measuring instruments, and computer applications used to process data existing in different WPPs. Thus it is essential to use techniques of data migration and data integration for the data to be loaded in a consistent manner in a centralized operational database which can be later used for prediction, simulation and analysis.

The first step for building the prototype is to develop the data level which will consist of operational data sources from the WPP, as well as spatial data and the data warehouse to be built to facilitate multi-dimensional analysis. A general architecture was presented in [3]. Some of the data collected on operational activity will be stored as data in a spatial DBMS (Database Management System Data), in order to represent them on maps for interactive monitoring of energy and resources.

The data warehouse will be a centralized repository, organizational, composed of data marts for each type of activity which is analyzed: operational (energy production) and financial. To load data warehouse from data sources is necessary to undertake a process of Extract, Transform and Load (ETL). This process will automatically run at regular intervals depending on the technical requirements of implementation.

2 Integration architecture

At the TSO, data sources from the monitoring devices or from the WPP units came in various formats like text, data-sheets or even through e-mail. In order to build the data level of the prototype, some data integration techniques must be applied. We proposed that the data will be received in XML format and will be stored in object-relational databases, in order to realize further analysis, necessary for decision making. Also, since data integration from various sources is mandatory, we will use XML as a standard for integration and data warehouses for centralize data from heterogeneous data sources.

2.1 The data model

The data model consists in a set of formal data description elements that can be used to generate and implement the database’s structures and define the system’s metadata. These elements have to offer a high level description of data in order to make possible the implementation of the requirements within any type of database: relational, object-oriented or hierarchical. To ensure such flexibility and also an open standard for data description an XML extension can be used. The extension is known as XML Schema language, also known as XSD (XML Schema Definition) and it can be used to express a set of rules to which an XML document must conform in order to be considered ‘valid’ according to that schema. It provides a formal description that can contribute to create a description that is precise and follows a prescribed set of rules in order to generate the set of entities and associated features of the data model. According to [1], the XSD specification acknowledges the influence of DTDs and other early XML schemas (DDML, SOX, XML-Data, and XDR) and it has adopted features from each of these proposals. The features offered in XSD that are not available in XML’s native Document Type Definitions (DTDs) are namespace awareness, and data types, that is, the ability to define element and attribute content as containing values such as integers and dates rather than arbitrary text.

The main benefits of using the XML schema definition or XSD according to [5] are:

- offers an open platform to data modeling that support a wide range of programming languages and platforms;
- is object oriented and provides a well-specified framework for object oriented development;
provides a wide range of pre-defined, built-in simple types (attributes and elements with text only content) that constrains the data model to conform to multiple platform requirements;

- offers a formal definition of the model that allows developers to take advantage of validation functionality abstracted into schema-aware XML parsers. Most information processing frameworks in use today now have schema-aware XML parsers built in and readily available for access by application developers.

To define a schema we can say that is an abstract collection of metadata, consisting of a set of schema components: element and attribute declarations and complex and simple type definitions. These components are usually created by processing a collection of schema documents, which contain the source language definitions of these components which are organized by namespace: all the named schema components belong to a target namespace, and the target namespace is a property of the schema document as a whole.

According to the definition in [6], a schema component is the generic term for the building blocks that comprise the abstract data model of the schema. The $\text{xs:schema}$ symbol defines the root element of a schema that contains representations for a collection of components (type definitions and element declarations), which have a common target namespace.

For example, in order to specify the structure of the table WPP_unit (unit_id number primary key, unit_name varchar(29), unit_area varchar(30)), the XSD schema can be written as follows:

```xml
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="WPP_UNIT" targetNamespace="WPP_UNIT">
  <xs:element name="WPP_UNIT">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="UNIT_NAME">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:maxLength value="29"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="UNIT_AREA">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:maxLength value="30"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:attribute name="UNIT_ID" type="xs:ID" use="required"/>
      </xs:complexType>
    </xs:element>
  </xs:schema>
```

In order to implement the model into a relational database, such as Oracle Database, the XSD documents can be used to generate code for Data Definition Language (DDL). This capability is referred to as XML Data Binding. This facility adds another advantage over the XML schema definition which can be considered as the best solution for data modeling. For each of these entities a XSD schema is created. After the schemas’ completion, the DDL (Data Definition Language) code is generated for each table and the ER Diagrams (Entity-Relationship Diagram) are built.

### 2.2 Database access through Java technologies

We propose a client-server architecture in which the databases reside on a separate server then the application or the server. We use JDBC that allows SQL instructions to be processed in an independent and consistent manner. JDBC technology offers a consistent interface for manipulating data, regardless of the format in which the data is stored [7]. Using a high level of abstracting represented by the JDBC API, the situation of the programmer to handle different SQL calls to a certain DBMS (Database Management System) is avoided (figure 2).
In order to be able to connect to a certain DBMS we only need to switch the driver, operation that can be done dynamically, even when the application runs, without the recompilation of the application.

The way those drivers are built is standardized through the JDBC specifications which describes the standard interfaces that are to be implemented. JDBC provides object-oriented access to databases through the definition of classes and interfaces that cover several abstract concepts. Also, the JDBC standard defines a series of interfaces that are to be implemented by the drivers’ creators in order to give the developers information about the queried database, the DBMS used and so on.

Through JDBC programming we can cover many aspects: client-server communication, drivers, APIs, data types and SQL instructions. The JDBC API relieves much of the burden needed to create applications with databases. It comprises many simple and intuitive components that can work for the programmer.

For implementation, the first step is to obtain, install and configure the JDBC driver. Afterwards the needed component can be utilized in all the JDBC applications. After that compiling takes place, the next steps are running the application and solving the eventual issues.

JDBC is an API that encapsulates calls on two levels needed to access database data and interacts through a common interface. JDK (Java Development Kit) and JRE (Java Runtime Environment) both contain the standard API, the interfaces and classes being contained in two major packages: java.sql and javax.sql. The first package includes standard components, whilst the second includes enterprise level components.

The entire communication with the database occurs through the JDBC drivers. This driver converts the SQL instructions into a database server comprehensible format, using the correct networking protocols [8]. JDBC abstracts the specific communication with the database.

The Java SDK includes the ODBC-JDBC driver, thus allowing the access to ODBC drivers to database. Instead of accessing directly the database, JDBC “talks” to the ODBC drivers, which, in its turn communicates with the database.

First of all, the application should be able to communicate with the database. Afterwards, the application has to be able to establish connections with the database to create a communication channel in order to send SQL commands and retrieve results. Finally, the application has to have a mechanism to deal with the errors. In order to accomplish all these things, the JDBC API provides the following interfaces and classes:

- Driver – this interface controls the communication with the database server. Rarely one should need to interact with objects of the Driver type. Given this, the DriverManager objects can be used instead. These have an abstract representation of the details associated to the work with Driver objects.
- Connection – instantiating objects of this interface represents the physical connection to the database. The result set and transactions can be controlled using Connection objects.
- Statement – objects created with this interface in order to send SQL commands to the database. Some derived interfaces accept supplemental parameters in order to execute stored procedures.
ResultSet – these objects contain the retrieved data from the database after the query has been performed using Statement object. These objects allow the browsing of the data like an auto incrementation.

SQLException – a class that traps any error that is encountered in the application.

Any Java application that uses databases works directly or indirectly with those four components described earlier.

3 Data representation and management
The data model will be represented in an object-relational database with spatial extensions in order to use the facilities of a Geographic Information System.

3.1. Object-relational representation
The reason for using object – relational model is that we can benefit from the scalability and complex data types (large objects, multimedia data, XML data, spatial data, user defined object types, etc.) Object-relational databases can by-pass the necessity of using mapping techniques between relational database and application programs. Also, we can use a unique data model, recognized in both database and programming language (which is most certainly an object-oriented programming language).

The advantage of using an object-relational database is that we can get the benefits of both relational and object-oriented technologies, while the disadvantage translates into lower performance due to XML data mapping to the relational data, which can produce a database schema with many relations [9].

3.2. GIS representation
Some of the data collected on operational activity will be stored as spatial data in order to representing them on interactive maps monitoring the production of energy, and other data like multimedia data and LOB will use specific operations.

Whether spatial data is processed on a server or not, one thing is clear, the database must be located on the server. Through this method, the minimum requirements, for the system to be considered in SOA - Service Oriented Architecture, are gained. Thus, services, that provide access to the database, are loosely coupled, reusable, and autonomous, do not have a state, are composite, standardized and communicate through metadata.

The processing of spatial data will be made on the server by using spatial operations in a PL/SQL block (for Oracle DBMS) or special functions of spatial object classes in programming languages such as Java (using JGeometry class which handles spatial data and can be used under oracle.spatial.geometry package).

A preview of the first step of the data level development in a GIS environment can be observed in figure 3.

Fig. 3 – The WPP’s locations represented in ArcGIS Desktop
The data from spatial database tables can be transmitted through web services as it is or processed. Such processing can be performed either in high level programming languages (e.g. Java, C#), either directly via database stored functions or procedures transmitted through web services as it is or processed.

The application level of the prototype is the next step of our project and it will be discussed in a future work.

4 Conclusion

Using XML as a format for receiving data from Wind Power Plants measuring system will allow us to gather in real time the amount or energy produced by each unit and aggregate the values in a central database. In order to use the XML data, a XSD schema must reflect the data model to generate and implement the database’s structures and define the system’s metadata. By using the XSD schema we create an open data model that support a variety of DBMS (Database Management Systems) platforms with spatial and object-relational representations and take advantage of validation functionality abstracted into schema-aware XML parsers. For data access, considering the object-relational model, we proposed the JDBC driver API as a middleware for integration.

Acknowledgments

This paper is a result of the research project PN II, TE Program, Code 332: “Informatics Solutions for decision making support in the uncertain and unpredictable environments in order to integrate them within a grid network”, financed within the framework of People research program. This paper is also a result of the project POSDRU/6/1.5/S/11 „Doctoral Program and PhD Students in the education research and innovation triangle”. This project is co funded by European Social Fund through The Sectorial Operational Program for Human Resources Development 2007-2013, coordinated by The Bucharest Academy of Economic Studies.

References: