Towards Standardization and Interoperability of Database Backups

Namedin Pereira Teles Júnior

Federal University of Amazonas (UFAM), Brazil

Roberto Souto Maior de Barros

Federal University of Pernambuco (UFPE), Brazil

Summary
The current business model of modern companies is increasingly dependent on information technology. Thus, it is necessary to keep security copies of the data (backup) which may be used to restore them after a data loss event. On the other hand, this dependence encourages strong competition among suppliers and this leads to new releases of computing commodities, seeking competitive advantages. In this context, it is usual for corporations to deploy more than one DBMS to store their data, creating heterogeneous environments, and creating difficulties for data interoperability. Current DBMSs use proprietary backup formats and this means that other tools cannot access their data. This work describes the development of BKPML, an XML-based language to standardize backup files. It is aimed at removing this dependency of the original environment of the data, allowing the manipulation of such files by any DBMS or tool, making systems integration and data interoperability easier. A prototype tool, called BKPML Manager, was also developed to support the use of BKPML.

Key words:
Interoperability, standardization, data backup, XML, DBMS,.

1. Introduction

Information systems are becoming the key element of the innovation process capable of decreasing, clustering and removing stages, and pushing client interaction to higher levels of quality, service and standardization, as well as of accurately detecting new trends. The current needs of corporations to adopt systems may occur according to their needs or according to the increasing demand for new products supported by modern technologies with innumerable competitive market advantages.

This scenario of increasing technological evolution and business needs shows how corporations are adopting several information systems to serve sectors such as management, finance, and others. Often, different systems are supported by different Database Management Systems (DBMS), developed by different suppliers, and using different technologies. Files created by such systems usually block information, avoiding data access by other systems, hindering and even making the interoperability of the data impossible, leading to chaotic heterogeneous environments [14].

This work presents the Backup Markup Language (BKPML), an open structure for DBMS data backup, based on XML [23], proposed as an alternative to face difficulties related to data interoperability of different DBMS and to proprietary format issues. This proposed language aims to overcome the dependency on the original environment of objects and data stored in specific DBMSs; it can be handled by any software, not necessarily a DBMS, and can also be easily converted to secondary formats such as CSV [6], JSON [7], XML, YAML Ain't Markup Language (YAML) [5], and XLS.

To support this proposal, we also developed BKPML Manager, a prototype tool that makes it easier the setting, generation, restoring, and migration of data and object backups using the BKPML format, to and from several DBMSs. To avoid incompatibility among systems, the BKPLM Manager tool was developed to be accessed online, using a browser such as Internet Explorer [9] or Mozilla Firefox [11]. Tests in a real environment were carried out to observe tool functionalities and performance, including backup, restore and data migration using BKPML, as well as its acceptability by possible users.

The rest of this paper is organized as follows: Section 2 describes the BKPML language as well as its taxonomy; Section 3 presents the results of the experiments using the BKPML Manager tool in a real environment; and, finally, Section 4 summarizes our conclusions and proposes future work.

2. BKPML

The Backup Markup language (BKPML) is an electronic language based on XML designed to eliminate the dependency on proprietary formats used in database

Manuscript received August 5, 2012
Manuscript revised August 20, 2012

backups. BKPML, together with the BKPML Manager tool, makes it possible that backup files are handled by any software, including DBMSs, as well as permitting their conversion to other formats. Thus, BKPML eases data handling according to the users' needs.

Providers of DBMSs such as Oracle [13], SQL Server [10], MySQL [12] and others implement data storage and backups in their own platforms, using proprietary formats. Therefore, data transferred to backup files can only be restored to their original environments. The BKPML file format enables independence of application, making the manipulation of backup files in other environments possible.

Restoring or migrating data from backup files to several DBMSs contributes to cost reduction, because the format standardization makes it possible to restore them directly to the target DBMS or application. In addition, it is important to emphasize that previous proposals for data standardization (for example [14]) are restricted to objects of type table. On the other hand, in addition to the table objects, BKPML supports many other types of objects: view, index, grant, trigger, and method.

2.1 BKPML Taxonomy

The BKPML taxonomy defines the rules used to form BKPML documents, the required elements and data used to represent objects in this structure, as well as the relationships between each element resulting in a final object.

This taxonomy is composed by a dictionary, which provides standard definitions to represent the database objects, and data, following a hierarchical structure.

The BKPML taxonomy was written using XML Schema [24], which is another XML-based language that permits defining rules for document validation.

The development of the general structure of the language and the relationships between each one of the objects were based on the organization model of the DBMS. The first element of this structure, the root element, was named BKPML. This element keeps general information of BKPML backup files such as file name and date of generation.

SchemaDB is the second element of BKPML and is used to save information related to the source DBMS and the schema of recoverable objects. The information or attributes that compose this element are the schema name and the original DBMS name. The element Objects is responsible for keeping all objects supported by BKPML, which are Table, Index, View, Grant, Trigger and Method. Figure 1 presents the hierarchical structure of BKPML.

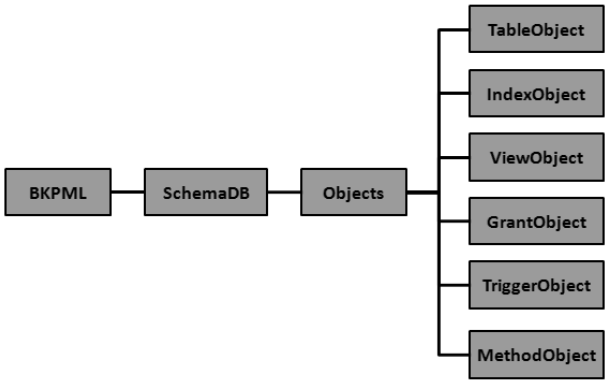

Fig. 1 BKPML Structure.

The *TableObject* element is responsible for keeping data information and metadata of tables. This element is composed by one attribute and two main elements. The *name* attribute stores the name of the table. The element Columns is responsible for keeping information related to the table structure such as*: column_names, data_types, column_sizes*, etc. The element *Records* is responsible for storing the data from a copied table. Each tuple (row) of the copied table is represented as a JSON object, which supports easy and fast information recovering and processing.

The *IndexObject* element is responsible for keeping basic data that composes the standard structure of this object. The syntax of this object may vary from one DBMS to another. So, the elements of that structure were based on a basic syntax, common to any DBMS. This element is composed by two attributes and three elements. The attributes *kind* and *type* describe the kind of index to be created (index, primary, unique) and the indexing method (Btree, Harsh, Rtree), respectively. The elements *index_name*, *index_table* and *index_field* keep basic information of this object.

The *ViewObject* element adopts a very simple syntax and is composed of two elements: the name element represents the name of the copied view and the query element stores the query that will produce the view results when executed.

The *GrantObject* element defines privileges of object access for a user or group of users. The structure of this object is composed of three elements: *privilege, object* and *user*. The *privilege* element stores the privileges and supports more than one privilege. The *object* element is the object associated to the privilege. The element *user* keeps the name of the user or group who receives the privilege.

The *TriggerObject* element keeps basic data needed to create a trigger, associated to a table, to be executed before or after an event. The structure of this object has five elements: *name* is the name of the trigger; *time* is the time of the trigger execution; *event* is the event associated to this trigger; *object* is the table associated to the trigger; and *body* is the command or block of commands that will be executed by the trigger.

The *MethodObject* element refers to two structures: function and procedure. It is composed of four elements: *name*, *parameter*, *result*, and *body*. The *name* element is the name of the object to be kept by the structure; the *parameter* element is related to the parameters of the object; the *result* element refers to the result of the method; and the *body* element refers to the commands that implement the method. In addition to these elements, this structure has the attribute *type* which records the object type: function or procedure.

The main goal of BKPML is to be capable of restoring and migrating backup files to any DBMS, irrespective of origin. Migrating or restoring backup data for heterogeneous DBMSs helps to reduce the costs and time involved in these processes. It will not be necessary to manipulate these files in their original DBMS because they can be handled directly by any relational DBMS thanks to the standardization of these files. An example of the BKPML document format is presented below.

```
<?xml version="1.0" encoding="UTF-8"?>
<BKPML date_created="2012-MAR-01" file_name="TEST">
<SchemaDB source="MYSQL5.2.35" name="BKPMLTest">
<Objects>
<TableObject table_name="names"> <!-- TableObject's Structure -->
   <Columns>
      <Column_Data column_name="ID" type="int" not_null="NO"
      primary_key="YES"/>
      <Column_Data column_name="NAME" type="varchar"
      max_length="45" not_null="NO"/>
   </Columns>
   <Records>
      <record>{"ID":"1","NAME":"ACTOR"}</record>
   </Records>
</TableObject>
<ViewObject order="1"> <!-- ViewObject's Structure -->
   <name>ViewNames</name>
   <query>
      SELECT ID, NAME, DESC
      FROM BKPMLTest.names
   </query>
</ViewObject>
</Objects>
<SchemaDB>
</BKPML>
```

Fig. 2 Example of BKPML.

The main advantages regarding the use of BKPML are:

- Independence and portability: data do not belong to any specific vendor or technology, they can be handled by any DBMS or software;
- Reduction of time and costs: BKPML proposes reduction of time and costs during these processes through the elimination of one phase of the process, which is the data restoration in its original environment; and
- Standardization: since BKPML is an XML based-language, there is no dependence on any technologies besides keeping an open structure shared by other users and also adaptable to their needs.

Further information on the development of the BKPML taxonomy as well as on all its forming objects are available [20]. Additionally, the complete code of the BKPML taxonomy can also be retrieved [21].

3. BKPML Manager

In order to better evaluate the advantages of BKPML, we developed a prototype tool to manage the main functionalities supported by the language, which are: generating backups using the BKPML structure, restoring or migrating these files directly to many DBMSs, and converting these files to other popular formats. In addition, it is important to emphasize that this tool eases the use of these functionalities through a simple user interface.

To use these functionalities, we implemented a module to maintain a simple database with basic information needed to correctly generate the BKPML structure and its objects. The main tables required to use the functionalities are: DBMSs, storage, and, for each DBMS, mapping, types and objects.

The DBMS table contains the list of all DBMSs that can be used in the tool. The storage registration contains the list of all repositories that can be used to store BKPML files and each of them is classified as local or cloud. For the local type, the files are stored in local servers. The cloud type uses the Simple Storage Service (S3) of Amazon Web services [1] to store the files. The mapping registration is used to inform the tool where to search for the object metadata of each DBMS. The registration of types and objects of each DBMS are used to validate data in BKPML files.

3.1 Architecture

The BKPML Manager tool was built using a web architecture composed by a web graphics client responsible for sending requests to the application server

(Servlet Container) Tomcat 6.0 [2] and communicating with a MYSQL 5.0 database using the hibernate 3.0 [8] framework, responsible for managing the access requests of BKPML Manager data.

As is usual in web applications, we decided to use the Model View Controller pattern (MVC) [3] to organize the architecture of the tool in layers, facilitating its use and maintenance. The model layer manages all object models (bean classes) used by BKPML. As BKPML Manager uses the hibernate framework, these models are known as entity classes.

The view layer is responsible for keeping and managing input and output screens data. This layer uses Java Server Page (JSP) [17] technology and Hypertext Markup Language (HTML) [22] to format and present data, respectively, and JavaServer Pages Standard Tag Library (JSTL) [18] in the interface between the control and view layers.

The control layer is responsible for implementing the functionalities of backup, restoration, transformation, and data migration. In the development of this layer we used the project patterns Decorator [4], Strategy [4], and Data Access Object (DAO) [19] to organize processes and reuse functionalities when necessary. Decorator was adopted to make the BKPML backup file generation process easier to use; Strategy was used in the process of data transformation for secondary files; and DAO was used to standardize the data access. The BKPML Manager is organized in three layers as presented in Figure 3.

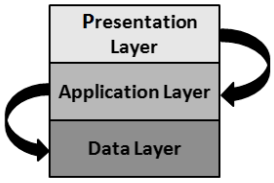

Fig. 3 BKPML manager architecture

The application layer contains the main methods responsible for the execution of processes for backup files generation and data restoration to DBMS or secondary files. Each process has its life cycle, which are the steps needed to conclude an activity.

To carry out a backup, the architecture executes the following steps: Request data (DBMS), Generate BKPML, Encrypt, Compress, Generate Hash, and Store. The restoration process executes the following steps: Search for file, Validate Hash, Unpack, Decrypt, and Restoration. Notice that Restoration can be done to any DBMS or to secondary files.

The Request Data step begins with a request from a DBMS and contains methods that permit executing the query, insert, update and delete commands, as well as connection and data transactions for several DBMSs.

The Generate BKPML Step contains methods for generation, validation and manipulation of BKPML files. The methods *Encrypt* and *Decrypt* belong to the *Cipher* process: these methods use the Triple-DES algorithm [16] to carry out the encryption step. The methods *Compress* and *Unpack* belong to the Compress step: these methods were developed using the java.util.zip package of Java.

The methods *GenerateHash* and *ValidateHash* belong to the *Hash* process, which is responsible for the extraction and validation of hash in BKPML files. Finally, the Storage process has methods to carry out data storage as defined by the user and is prepared to store and search files in local machines, in a network, and in clouds.

3.2 Registration and Execution of Data Backups

The first step needed to start a data backup process is to register all backup actions for the given objects. Then, the execution screen for data backup must be filled in with a list of all available backups to be executed. All settings of a data backup are carried out in the action registration. In other words, an action is a backup not executed, since it has all the information needed for that. The action registration requires information on the DBMS and objects to be copied, data repository, date and time of execution, and the frequency of execution of the backup. In the case of table objects, it is also necessary to provide a query for data extraction. After that, data registration can be confirmed. Figure 4 presents the action registration screen.

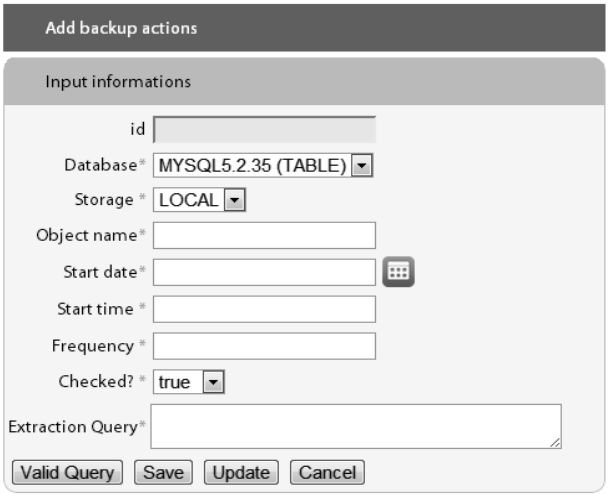

Fig. 4 Backup action registration screen

The action registration screen is not responsible for the execution of the backup, only for its registration. When the registration of an action is completed, it is sent to a list of backups that can be executed by the data backup functionality. The execution screen contains a list of actions that can be executed. The execution starts when an execution key available for each list element is selected. When these actions are carried out they are submitted to the processing cycle of backup according to the next steps. The generate BKPML process requires data information and object metadata and then generates a BKPML structure instance according to the given taxonomy. After that, encrypt and data compression processes are executed. Finally, the hash of the final file is extracted and the file is stored.

3.3 Restoration and Transformation of BKPML Files

The process of data restoration is responsible for retrieving the data of an existing BKPML file for some DBMS or converting them into secondary files. It is important to notice that this functionality is designed to manipulate BKPML files of table objects only; the other types of objects are handled by the complex data migration functionality. The screen used in this task contains the list of all previous backups carried out by the tool.

To carry out the data restoration for a DBMS, the user must choose a specific DBMS and select the transfer option in the format menu. After that, the execution button of the item must be pressed to start the process. The transformation process follows the same steps; the only difference is the chosen format. The data restoration and transformation screen is presented in Figure 5.

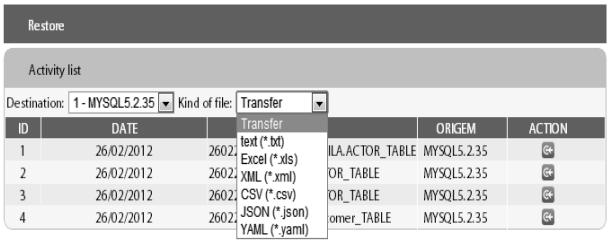

Fig. 5 Data restoration and transformation screen

3.4 Complex Data Migration

The functionality of complex data migration is used to deal with complex objects. In the current implementation of the tool, the complex objects require visual analysis and manual structure editing before they can be restored. The tool considers all objects to be complex, except table. Thus, the following objects are complex: view, index, permission, trigger, and method.

The migration screen works as an editor that permits adjusting the structure of the complex objects to the target DBMS. This screen implements the carry objects functionality, responsible for transforming objects kept in the BKPML files to the standard SQL syntax, except objects of the type method and trigger. For these objects, their metadata will be converted to a specific syntax hardcoded in the tool.

3.5 Results

The tests carried out using the BKPML Manager tool were set in a real environment provided by the Knowhow Consulting Company in the city of Manaus, Brazil.

This first phase of tests was aimed at finding and correcting errors in the basic registration module functionalities, as well as measuring the processing time taken by these functionalities using data provided by the company. Based on the results of this first phase of tests, the basic registration functionalities were considered successful and the processing time was considered adequate.

Because the first phase of tests used a comparatively small dataset, it was necessary to test the main functionalities of the tool with larger datasets. For that purpose, the company provided a machine with the following configuration: Intel Dual Core 2.1 GHz processor, with 2 GB of RAM and a 200 GB HD. The environment to execute the tool included a Postgres 8.4 DBMS [15]. After that, we wrote a program to insert random data in the test table. The BKPML Manager tool was executed to generate backup files of the stored data after each execution of this program.

Based on the results of these tests, we concluded that the BKPML backup file generation was comparatively fast, i.e. the performance of the tool in the backup of large files was good.

However, the time taken in the restoration process was much slower, but still acceptable. Based on the results, it was possible to estimate that, for 1 Gigabyte of information (approximately 24,000,000 records), the BKPML Manager tool would take about 17 hours to process data restoration, which is not very efficient. Nevertheless, it is important to notice that the results were obtained in a low-end configuration computer; using a more powerful configuration would probably lead to much better results.

4. Conclusions and Future Work

This work proposed BKPML, an open format standard based on XML, which was developed to support independence of platform in database backups, allowing for its manipulation by several DBMSs. More specifically, this format supports data migration projects.

After developing this language, a prototype tool to validate BKPML files, named BKPML Manager, was implemented. BKPML Manager was developed to generate and validate backup files written in the BKPML format as well to restore, transform and migrate data to different DBMS platforms or to secondary files. Additionally, this tool allows data storage in the clouds using Amazon's S3 service to store data.

After the BKPML Manager development, tests were carried out to measure the performance of the tool in many of its functionalities. Even though this is the first prototype developed, and despite the fact that there is certainly much room for improvements, its main functionalities, namely backup, transformation, restoration and migration of complex objects, presented successful results.

The main contributions of this work are:

- The proposal of an open format to standardize database backups;
- The development of a prototype tool to support the BKPML format and provide backup projects and data migration with greater flexibility regarding data manipulation in different platforms;
- To make portability and data manipulation easier by importing and exporting backup data using secondary files; and
- To provide an XML structure and a tool which are not restricted to objects of type table.

Although the results of this work are encouraging, further research is needed. Possible future work includes:

- Increasing the scope of BKPML to deal with more database objects such as *roles* and *packages*;
- Further developing the XML structure to store mapping, syntax, objects, and DBMS object properties, reducing the number of registrations needed to use it in practice;
- Adding a functionality to break very large BKPML files into smaller files, to make reading operations faster and avoid buffer overflow. XLink [25] could possibly be used to implement such functionality.

References

[1] Amazon inc. (2012). Amazon simple storage service (Amazon S3). http://aws.amazon.com/pt/s3, accessed August/2012.

[2] Apache inc. (2010). Tomcat website. http://tomcat.apache.org, accessed August/2012.

[3] Downey, T. (2007). Web development with java, using hibernate, JSPs and Servlets. British Library, Miami-FL, USA.

[4] Gamma, E., Johnson, R., Vlissides, J., and Helm, R. (1995). Design patterns elements of reusable object-oriented software. Addison Wesley.

[5] IETF org. (2001). Internet message format. http://www.ietf.org/rfc/rfc2822.txt, accessed August /2012.

[6] IETF org. (2005). Common format and MIME type for comma-separated values (CSV) files. http://tools.ietf.org/html/rfc4180, accessed August/2012.

[7] IETF org. (2006). The application json media type for javascript object notation (JSON). http://tools.ietf.org/html/rfc4627, accessed August/2012.

[8] Jboss inc. (2012). Relational persistence for java and .NET. http://www.hibernate.org, accessed August/2012.

[9] Microsoft inc. (2010). Internet explorer official website. http://ww.microsoft.com/brasil/windows/internet-explorer, accessed August/2012.

[10] Microsoft inc. (2011). SQLServer website. http://msdn.microsoft.com/pt-br/sqlserver/default, accessed August/2012.

[11] Mozilla org. (2010). Firefox reference and download. http://br.mozdev.org, accessed August/2012.

[12] Oracle inc. (2011). MYSQL official website. http://www.mysql.com, accessed August/2012.

[13] Oracle inc. (2011). Oracle official website. http://www.oracle.com/br/index.html, accessed August/2012.

[14] Oumtanaga, S., Lambert, K. T., Tiemoman, K., Pierre, T., and Florent, D. N. (2007). Use XML format like a model of data backup. International journal of computer and information engineering, vol.33, pp.170-175.

[15] PostgreSQL org. (2011). PostgreSQL official website. http://www.postgresql.org, accessed August/2012.

[16] RSA lab. (2012). What is triple-DES?, RSA laboratories. http://www.rsa.com/rsalabs/node.asp?id=2231, accessed August/2012.

[17] Sun inc. (2010). Java server pages website. http://java.sun.com/products/jsp, accessed August/2012.

[18] Sun inc. (2010). Javaserver pages standard tag lib website. http://java.sun.com/products/jsp/jstl/reference/docs/index.html, accessed August/2012.

[19] Sun inc. (2011). DAO reference. http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html, accessed August/2012.

[20] Teles Jr., N. P. (2011). Backup markup language (BKPML): uma proposta para padronização e interoperabilidade de backup de dados. Master's thesis, Universidade Federal de Pernambuco, Recife, Pernambuco, Brasil. *In Portuguese*. http://www.scribd.com/doc/48468544, accessed August/2012.

[21] Teles Jr., N. P. (2011). XML schema of BKPML. http://www.scribd.com/doc/47836213, accessed August/2012.
[22] W3C org. (2010). Hipertext markup language website. http://www.w3.org/MarkUp, accessed August/2012.
[23] W3C org. (2011). Extensible markup language. http://www.w3.org/XML, accessed August/2012.
[24] W3C org. (2011). XML schema. http://www.w3.org/XML/Schema, accessed August/2012.
[25] W3C org. (2012). XLink reference. http://www.w3.org/TR/xlink, accessed August/2012.

Namedin T. Pereira Jr. received his B.Sc. degree in Systems Analysis from Fundação Centro de Análise Pesquisa e Inovação Tecnológica (FUCAPI) in 2006, his Database Project and Management Specialization from Universidade do Norte (UNINORTE) in 2009, his M.Sc. degree in Computer Science from Universidade Federal de Pernambuco (UFPE) in 2011, and he is now a Ph.D. student at Universidade Federal do Amazonas (UFAM), all in Brazil. He also holds Systems Analyst and Professor positions at FUCAPI since 2004. His main research areas are databases, interoperability, cloud computing and distributed software development.

Roberto S. M. Barros received his B.Sc. and M.Sc. degrees in Computer Science from Universidade Federal de Pernambuco (UFPE), Brazil, in 1985 and 1988, respectively, and his Ph.D. degree in Computing Science from The University of Glasgow, Scotland (UK) in 1994. From 1985 to 1995 he worked as systems analyst at UFPE and since 1995 he is a full time Professor and Researcher, also at UFPE. His main research areas are software engineering, databases, programming languages, XML, and pattern recognition.