Storage and Access Control Policies for XML Documents

George Pallis
Aristotle University of Thessaloniki, Greece

Konstantina Stoupa
Aristotle University of Thessaloniki, Greece

Athena Vakali
Aristotle University of Thessaloniki, Greece

INTRODUCTION

The Internet (and networks overall) are currently the core media for data and knowledge exchange. XML is currently the most popular standardization for Web document representation and is rapidly becoming a standard for data representation and exchange over the Internet. One of the main issues is XML documents and in particular, storage and accessing. Among data management issues, storage and security techniques have a particular importance, since the performance of the overall XML-based Web information system relies on them. Storage issues mainly rely on the usage of typical database management systems (DBMSs), whereas XML documents can also be stored in other storage environments (such as file systems and LDAP directories) (Amer-Yahia & Fernandez, 2002; Kanne & Moerkotte, 2000; Silberschatz, Korth & Sudarshan, 2002). Additionally, in order to guarantee the security of the XML data, which are located in a variety of the above storage topologies, the majority of implementations also provide an appropriate access control. Most storage systems cooperate with access control modules implementing various models (Joshi, Aref, Ghafoor & Spafford, 2001), whereas there are few commercial access control products available. However, there are some standardized XML-based access control languages that can be adopted by most tools.

This article focuses on presenting an overview related to both storing and securing XML documents. Such an overview will contribute to identifying the most important policies for storage and access in Web-based information systems (which extensively use the XML as the data representation format). In addition, the most well-known related implementations are presented. A more integrated survey on these issues is presented in Pallis, Stoupa, and Vakali (2004).

BACKGROUND

Storage Policies

Several solutions for storing XML data have been proposed both in the scientific literature and in commercial products. In particular, storage approaches can be classified with respect to the type of system on which they rely and on the used XML document representation model. In this framework, they can be categorized as follows:

- Relational DBMS: Uses a collection of tables to represent both data and relationships among these data. More specifically, in order to represent XML data by using tables, it is necessary to break down the XML documents into rows and columns. The tree-like structure of XML facilitates both their decomposition and storage in relational tables. However, this process is expected to cause some performance overhead mainly due to the continuous translation of trees to tables (and vice versa). Due to its popularity, several models have been proposed to store XML documents in relational DBMSs (e.g., Bohannon, Freire, Roy & Simeon, 2002; Khan & Rao, 2001; Schmidt, Kersten, Windhouwer & Waas, 2000; Tian, DeWitt, Chen & Zhang, 2000; Zhu & Lu, 2001).
- Object-Oriented (O-O) DBMS: XML documents are stored as collections of object instances, using relationships based on the O-O idea (McHugh, Abiteboul, Goldman, Quass & Widom, 1997). O-O DBMSs have been designed to work well with object programming languages such as C++, C#, and Java. Inheritance and object-identity are their basic characteristics. However, O-O DBMSs cannot easily handle data with a dynamic structure since a new

class definition for a new XML document is needed, and the use of O-O DBMSs for XML document storage is not as efficient and flexible.

- Object-Relational (O-R) DBMS: The XML documents are stored in a nested table, in which each tag name in DTD (or XML schema) corresponds to an attribute name in the nested table. Currently, researchers have showed a steadily increasing interest in O-R DBMSs, since they combine both the benefit of the relational maturity and the richness of O-O modeling (Pardede, Rahayu & Taniar, 2004). In O-R DBMSs, the procedure for storing XML data to relation mapping is modeled by an O-R model. In this context, a number of transformation steps from the XML schema to an O-R model are presented (Widjaya, Taniar & Rahayu, 2004). More specifically, each nested XML element is mapped into an object reference of the appropriate type. Then, several mapping rules are indirectly embedded in the underlying model.
- Native XML DBMS: In this case, the XML document is the fundamental unit of storage (Kanne & Moerkotte, 2000). Therefore, a native XML database defines a (logical) model for an XML document, and stores and retrieves documents according to that model. In particular, two basic steps are involved for storing the XML documents on a native XML DBMS: (1) the data are described by its structure (DTD or XML schema), and (2) a native database XML schema (or a data map) is defined. However, the native XML DBMSs have not yet become very popular, since these systems must be built from scratch.
- LDAP Directories: XML documents are stored in LDAP directories which can be considered as a specialized database (Marron & Lausen, 2001). Therefore, the internal storage model of this database system is defined in terms of LDAP classes and attributes. More details about the architecture of the LDAP model and protocol are discussed by Howes, Smith, and Good (1999). Comparing the LDAP directories with the typical DBMSs, they are more widely distributed, more easily extended, and replicated on a higher scale.
- File Systems: Since an XML document is a file, a typical storage approach is to store it simply as a flat file. In particular, this approach uses a typical file-processing system, supported by a conventional operating system (as a basis for database applications). The wide availability of XML tools for data files results in a relatively easy accessing and querying of XML data (which are stored in files). By using a flat file for XML data, we have faster storing (or retrieving) of whole documents. However, this storage format has many disadvantages, such as difficulty in accessing and updating data, since the only way to accomplish this is to overwrite the whole file (Silberschatz et al., 2002).

Access Control Policies

In order to protect XML files, we need to design authorizations defining which client (subject) can access which protected resource (object) and under which mode. Since XML files are organized according to DTDs or XML schemas, protected resources can be both XML files, DTDs, or schemas, or parts of them, such as a specific element or even attribute.

The basic access control models are:

- Discretionary Access Control (DAC): Every subject and object is enumerated, and there are authorizations connecting each subject and object. The owner is responsible for defining policies in order to protect his/her resources, and (s)he can also define who is going to be assigned some access privileges. It is the most flexible and simple access control model. However, it does not provide high levels of protection, and it cannot be used in case multiple security levels are required.
- Mandatory Access Control (MAC): This is based on the existence of one central administrator responsible for the definition of policies and authorizations. It is expressed with the use of security labels associated with both subjects and objects. Every object has a classification label defining its sensitivity, and each subject is assigned a clearance label defining its trustworthiness. This model is more secure than DAC, but it cannot be used in wide distributed Internet-based environments. Therefore, their usage is gradually decreased.
- Role-Based Access Control (RBAC): The most widely used access control model in modern environments (Sandhu, Coyne & Feinstein, 1996). Subjects are assigned roles that are categorizations of subjects according to their duties in an organization. Thus, every subject is assigned roles that in their part have some authorizations. Therefore, the number of the needed policies is highly decreased. RBAC model is a super set, since it can also express DAC and MAC policies. It is appropriate for distributed heterogeneous networks offering Internet access. Moreover, it is recommended to protect hypertext documents (HTML or XML files). Lately, a generalized RBAC model has been proposed where objects and environmental conditions are assigned roles (Moyer & Ahamad, 2001).