Introduction

XML files are in a standardized text format used for represent, store and transport structured information. Therefore, **XML can be used as a neutral data storage method**, as the information stored in XML files can be moved from one machine to another without any problems. This makes it an ideal method for storing historical data that ensures that it can be re-read in the future without any problems.

This has made it increasingly difficult for organizations to have more and more XML documents, and it is becoming increasingly difficult to locate documents at the right time. So there is a need to organize your XML data in a way that allows you to locate it quickly.

But it's not just a matter of locating the documents, as often what you are looking for will be divided into different documents. You also need some way to be able to query and manipulate the data contained in the XML files. This query system should be able to search documents and modify data if necessary, but always with a minimum of efficiency.

In the face of these needs, the solution is usually given in two ways:

1. Since most organizations already have organized data storage systems based on relational data, one possible solution might be to **convert XML files to relational data**:
    - Data storage will be fully organized and centralized at one point where the other data in the organization already exists.
    - Its query language, SQL, is well known and widely used, so users will not need to learn a new language.
2. Another solution is to create a storage system designed for XML files only. These are the systems known as native XML databases. These:
    - Provide a place to neatly store XML files.
    - They have their own query language: XQuery.

Although this distinction seems easy to make, in practice organizations often use mixed storage systems, in which they store in relational data those data that need to be stored or queried, and in XML those data that must be represented in web environments or to be exchanged or transported to other systems.

Relational databases

Almost all organizations have their data organized with some relational system, as database management systems have become an indispensable element in the functioning of companies. They are used to control many of the operating data of companies: turnover, accounting, stocks…

Therefore, in the face of the emergence of a new type of data, one of the easy reasons would be: "If we already have a well-functioning data organization system, why can't we put that data in the same system?" . In this way, the data obtained from XML files would achieve a very efficient storage system and a method of manipulating information that has been widely tested for many years and is well known to many people.

The inclusion of XML files in relational database management systems can be done in two ways:

1. Convert data from XML files to relational data:
    - This system has the advantage that the data, once within the relational system, will be identical to the existing ones.
    - The biggest downside is that if you need the original XML document again, it can be very difficult to regenerate it, as there will often be information about the XML structure that will not be stored.
2. Store entire XML documents in databases:
    - Entire XML files will be placed in a field in a database table, so it will be like the other data.
    - In order to work well, the database will need to provide some moderately efficient way to search the contents of XML documents.

Convert XML data to relational

One solution that may seem simple but not so simple is to take the data from the XML files and turn it into relational data. Once the data has been added to the database, the XML files can be deleted.

Because the structure of the data is already defined in the XML document, and XML documents are often associated with a vocabulary, we need to:

1. Generate the structure of relational tables can be done by analyzing the structure of the XML document.
2. Vocabulary definition fields can be obtained by simply looking at the contents of the XML document.

And besides, there are systems that allow you to transform XML data into other file types (for example, XSLT).

Convert an XML document into relational data

If you start from the following XML document:

```
<?xml version = "1.0"?>
<students>
   <student>
      <name> Frederic </name>
      <cognom> Pi </cognom>
   </student>
   <student>
      <name> Philomeno </name>
      <cognom> Garcia </cognom>
   </student>
</students>
```

It is relatively easy to see that the structure of the document consists of a list of "student" items. And that `<student>` will have two data fields which are `<name>` and `<surname>`.

Therefore, the relational structure of this file will simply be to create a "student" table with a first and last name. This is trivial with the SQL CREATE TABLE command:

```
CREATE TABLE student (first name VARCHAR (30), last name VARCHAR (30))
```

Getting the data to fill the table shouldn't be a big deal either. Simply insert each data field:

```
INSERT INTO student VALUES ("Frederic", "Pi");
INSERT INTO student VALUES ("Filomeno", "Garcia");
```

The easiest way to do this is to create an XSLT template that will transform the XML file into SQL rules.

Despite the apparent simplicity of this system, it is not always easy to do this conversion, as relational and XML systems are based on quite different concepts:

- The relational system is based on two-dimensional data without hierarchy or order, while the XML system is based on hierarchical trees in which order is significant.
- In an XML document there may be repeated data, while relational systems escape repetitions.
- Relationships and structures within XML documents are not always obvious.
- What if we need to have the XML document back? Doing the reverse process is not always trivial. One of the difficult concepts is to determine which data were attributes and which elements.

Therefore, this is usually not the most advisable system but it is a valid system that can be useful in specific cases.

Relational systems with XML extensions

Until the advent of XML, virtually everyone stored data that needed to be quickly queried or modified by some relational system. Database management systems have been the king of data management in organizations for years.

But the increase in information in XML that needed to be queried or required to perform tasks that were previously reserved in relational databases has led to some kind of XML support having to be included in the database management systems.

All major database systems such as Oracle, IBM DB2, or Microsoft SQL Server have some kind of XML support, which is usually specified in the following:

- Allow exporting relational data in some XML format to transport the data.
- Have some way to be able to store XML documents as fields in relational tables.
- Allow searches and changes to stored XML documents.
- Generate XML from relational data in the database.

!!! info Although SQL/XML is part of the SQL standard, not all database managers support it.

Because SQL (the quintessential relational data query language) did not support XML in 2003, the SQL language standard was modified to add the **SQL/XML extension**.

SQL/XML

SQL/XML is an extension of the SQL standard that allows you to work with the XML language using SQL statements.

This extension was developed by a group in which there were large database companies (Microsoft, Oracle, IBM, SyBase, DataDirect Technologies…) and is already implemented in some database system.

!!! info Despite participating in the development of the standard, Microsoft announced that it did not intend to incorporate SQL/XML into Microsoft SQL Server. Instead, Microsoft SQL Server uses its own system called Microsoft SQLXML or OPENXML to work with SQL and XML.

SQL/XML defines a series of functions for publishing XML files from relational data, defines a type of XML data and a way to query and manipulate stored XML data.

XQuery

One of the most basic requirements for searching for data is to have a query language that is powerful enough to meet the needs of those who need to use it. That's why the XQuery language was developed.

!!! important XQuery is a query language designed to become the standard way to retrieve data from XML document collections.

It's a functional language, so instead of telling you the steps to do a task, all you have to do is evaluate the expressions against the XML file and generate a result. Unlike common programming languages, XQuery specifies what e isl what you want and not the way you have to do it to get it.

!!! info "XQuery 3.0"

```
A new version of XQuery is already under development that will increase the
power of queries by adding new clauses such as group by or dynamic error
checking. You can check it out at www.w3.org/TR/xquery-30
```

Among the most interesting features of XQuery, it allows:

- Select the information according to criteria. Sort, group, add data.
- Filter the information you want from the data stream.
- Search for information in a document or group of documents.
- Join data from multiple documents.
- Transform and restructure XML.
- Don't be limited to search, as it can perform numerical and character manipulation operations.
- You can work with namespaces and documents defined by DTD or XSD.

An important part of XQuery 1.0 is the XPath 2.0 language, which is the part that allows you to make selections of information and browse the document.

Native XML databases

If we do not go into technical details, the definition of a database is a place where data can be stored. In the case of an XML database, it is simply a matter of storing the XML files at a specific point in the operating system.

The increase in XML documents to be stored has made it difficult to have some way to automate the process, even though the storage can be done manually. To facilitate this automation, native XML (NXD) databases have appeared.

When we talk about native XML databases, we are referring to databases designed to contain and store data in XML format.

Unlike relational databases, which have been in operation for many years and have an important theoretical basis behind them, NXDs do not have definite standards for doing things, and the theory behind them is not very well defined. As a result, each database often does things in a way that may be completely different from another.

Native XML databases are primarily used to store data that contains:

- Narrative content.
- Which are less predictable than those normally stored in relational databases.
- That should generate outputs for web environments.
- They have to be transported from one system to another.