XML DATABASES

XML Hierarchical (Tree) Data Model

The basic object in XML is the XML document. Two main structuring concepts are used to construct an XML document: elements and attributes.

An example of an XML element called . As in HTML, elements are identified in a document by their start tag and end tag. The tag names are enclosed between angled brackets < ... >, and end tags are further identified by a slash.</.....>.

Complex elements are constructed from other elements hierarchically, whereas simple elements contain data values. A major difference between XML and HTML is that XML tag names are defined to describe the meaning of the data elements in the document, rather than to describe how the text is to be displayed. This makes it possible to process the data elements in the XML document automatically by computer programs.

Also, the XML tag (element) names can be defined in another document, known as the schema document, to give a semantic meaning to the tag names that can be exchanged among multiple users. In HTML, all tag names are predefined and fixed; that is why they are not extendible. It is possible to characterize three main types of XML documents:

Data-centric XML documents.

These documents have many small data items that follow a specific structure and hence may be extracted from a structured database. They are formatted as XML documents in order to exchange them over or display them on the Web. These usually follow a predefined schema that defines the tag names.

Document-centric XML documents.

These are documents with large amounts of text, such as news articles or books. There are few or no structured data elements in these documents.

Hybrid XML documents.

These documents may have parts that contain structured data and other parts that are predominantly textual or unstructured. They may or may not have a predefined schema

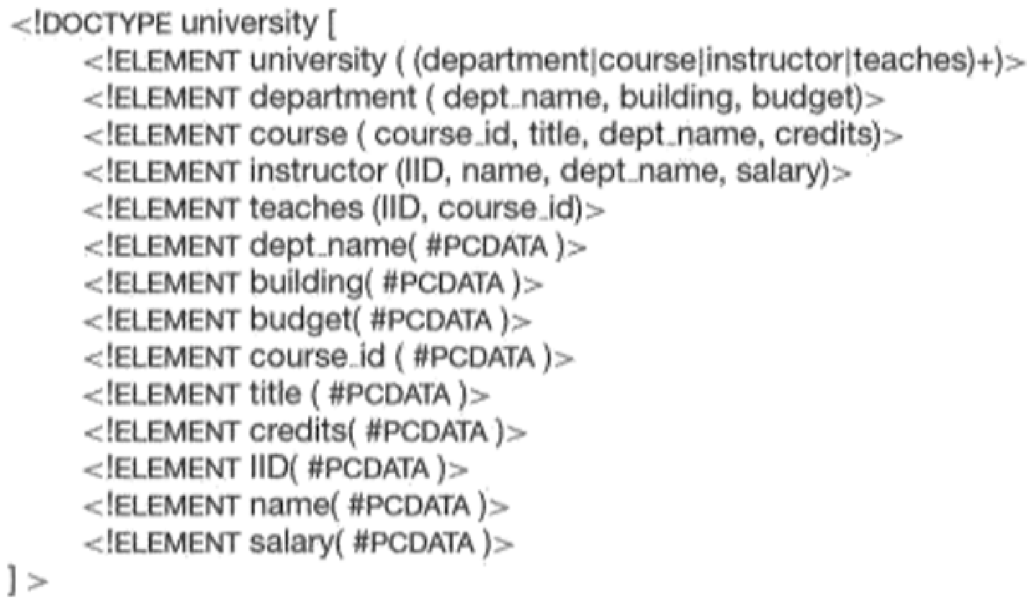
```
<?xml version= “1.0” standalone=“yes”?>
<Projects>
        <Project>
                <Name>ProductX</Name>
        <Number>1</Number>
        <Location>Bellaire</Location>
        <Dept_no>5</Dept_no>
        <Worker>
                <Ssn>123456789</Ssn>
                <Last_name>Smith</Last_name>
                <Hours>32.5</Hours>
        </Worker>
        <Worker>
                <Ssn>453453453</Ssn>
                 <First_name>Joyce</First_name>
                <Hours>20.0</Hours>
        </Worker>
</Project>
 <Project>
        <Name>ProductY</Name>
        <Number>2</Number>
         <Location>Su garland</Location>
        <Dept_no>5</Dept_no>
        <Worker>
                <Ssn>123456789</Ssn>
                <Hours>7.5</Hours>
        </Worker>
        <Worker>
                <Ssn>453453453</Ssn>
                <Hours>20.0</Hours>
        </Worker>
        <Worker>
                <Ssn>333445555</Ssn>
                <Hours>10.0</Hours>
        </Worker>
</Project>
```

```
        ...
</Projects>
```

XML documents that do not follow a predefined schema of element names and corresponding tree structure are known as schemaless XML documents. It is important to note that data-centric XML documents can be considered either as semistructured data or as structured data

4.2 DOCUMENT TYPE DEFINITION (DTD)

The document type definition (DTD) is an optional part of an XML document. The main purpose of a DTD is much like that of a schema: to constrain and type the information present in the document.



```
<!DOCTYPE university [
    <!ELEMENT university ( (department|course|instructor|teaches)+)>
    <!ELEMENT department ( dept_name, building, budget)>
    <!ELEMENT course ( course_id, title, dept_name, credits)>
    <!ELEMENT instructor (IID, name, dept_name, salary)>
    <!ELEMENT teaches (IID, course_id)>
    <!ELEMENT dept_name( #PCDATA )>
    <!ELEMENT building( #PCDATA )>
    <!ELEMENT budget( #PCDATA )>
    <!ELEMENT course_id ( #PCDATA )>
    <!ELEMENT title ( #PCDATA )>
    <!ELEMENT credits( #PCDATA )>
    <!ELEMENT IID( #PCDATA )>
    <!ELEMENT name( #PCDATA )>
    <!ELEMENT salary( #PCDATA )>
] >
```

However, the DTD does not in fact constrain types in the sense of basic types like integer or string. Instead, it constrains only the appearance of sub elements and attributes within an element.

The DTD is primarily a list of rules for what pattern of sub elements may appear within an element.

Example of a DTD

Thus, in the DTD, a university element consists of one or more course, department, or instructor elements;

the operator specifies “or”

while the + operator specifies “one or more”. Although not shown here,

the ∗ operator is used to specify “zero or more”

while the? operator is used to specify an optional element (that is, “zero or one”).

The course element contains sub elements course id, title, dept name, and credits (in that order).

Similarly, department and instructor have the attributes of their relational schema defined as sub elements in the DTD. Finally, the elements course id, title, dept name, credits, building, budget, IID, name, and salary are all declared to be of type #PCDATA.

The keyword #PCDATA indicates text data; it derives its name, historically, from “parsed character data”.

Two other special type declarations are empty, which says that the element has no contents, and any, which says that there is no constraint on the sub elements of the element; that is, any elements, even those not mentioned in the DTD, can occur as sub elements of the element. The absence of a declaration for an element is equivalent to explicitly declaring the type as any.