XML structures may be nested to arbitrary depth. New tags may be defined at will. Differences from HTML in at least 3 different ways. Used to describe content rather than presentation.

CH. 3: XML (Extensible Markup Language)
Abbreviated to `<married>`

- Empty elements `<married>` can be
  - `PDATA` (Parsed Character Data).

Because XML treats all data as text, this is referred to as

- Quotation marks around strings have disappeared in XML.

**Note:** Element includes the start and end tag

```xml
<person/>
<email>ge@abc.com</email>
<age>42</age>
<name>Alen</name>

<person/>
```

Element: Piece of text bounded by user-determined matching tags:

3.1 XML Elements

3.1.1 XML Syntax
<table>
  <thead>
    <tr>
      <th>Person</th>
      <th>Person</th>
      <th>Person</th>
      <th>Person</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Name: Ryan</td>
      <td>Name: Alex</td>
      <td>Age: 25</td>
      <td>Age: 30</td>
    </tr>
    <tr>
      <td>Name: Party</td>
      <td>Name: Alan</td>
      <td>Age: 36</td>
      <td>Age: 40</td>
    </tr>
    <tr>
      <td>Name: Alan</td>
      <td>Name: Alex</td>
      <td>Age: 42</td>
      <td>Age: 45</td>
    </tr>
  </tbody>
</table>

The collection of all persons on the 4th floor:

Collections are expressed using repeated structures.

EX: The collection of all persons on the 4th floor:
- Attribute values must be enclosed within quotation marks.
- As with tags, user may define any number of attributes.

```xml
<product/>
<address/>
<country>France</country>
<city>Strasbourg</city>
<zip>67100</zip>
<street>23 Rue Croix-Bosset</street>
<address format="XML56" language="French">
  <price currency="Euro" value="420.12"/>
  <name language="French" translate="true"/>
</address>

3.1.2 XML Attributes
- Expressed as a name-value pair
- Attributes define some properties of elements;
or sub-elements.

Repeat any number of times and their values may be string data.

On the other hand tags defining elements/sub-elements can

Its value is always a string.

A given attribute can occur only once within a tag:

Differences between attributes and tags:
or other combinations of attributes such as:

```xml
<person>
  <email>abh@abc.com</email>
  <age>42</age>
  <person name="Jian" />
</person>
```

to use attributes or elements:

In data exchange applications, there is ambiguity on whether
Well-formed XML documents must parse into a labeled tree always.

- Attributes must be unique within an element.
- Tags must nest properly.

3.1.3 Well-formed XML documents
correspondingssd-expression:

<person>
<name>"Alan"</name>
<age>42</age>
<email>"ab@abc.com"</email>
</person>
data are not easy to recognize. Beyond this simple analogy, however, XML and semi-structured

\[
\begin{align*}
\text{T} & = \text{atomically} \\
\text{T} & = \text{atomically} \\
\end{align*}
\]

XML is straightforward:

The following transformation from sdd-expressions to

one-to-one correspondence:

Tree structured structures:
create a new root node. Figure 3.3

Simply "lift" the node edges up one level,

Easy to transform XML tree into ssd tree.

ssd-expressions: Labeled edges

XML element: Labeled nodes

expressed as trees with the following distortion:

For tree data, XML element and ssd-expressions can be

3.2.1 XML Graph Model
Example:

previously defined reference.

Use id attribute (in an empty element) to refer to a

Use id attribute to define a reference (similar to old).

3.2.2 XML References
Detected example:
3.2.3 Order

The semi-structured data model is based on unordered collections, whereas XML is ordered.

The following two pieces of semi-structured data are equivalent:

    person: {firstname: "John", lastname: "Smith:"}
    person: {lastname: "Smith", firstname: "John"}

but the following two XML data are not:

    <person><firstname>John</firstname>
    <lastname>Smith</lastname>
    </person>

    <person><lastname>Smith</lastname>
    <firstname>John</firstname>
    </person>

To make matters worse, attributes are NOT ordered in XML; following two are equivalent:

    <person firstname="John" lastname="Smith"/>

    <person lastname="Smith" firstname="John"/>
we need to introduce new standard tags for the PCDATA.

To translate such XML data into ssd-expressions,

but from a document perspective, this is quite natural!

This seems un-natural from a database perspective,

endance

<person/>

I am not sure of the following email

This is my best friend

XML allows us to mix PCDATA and sub-elements within an element.

3.2.4 Mixing Elements and Text
that process XML titles.

such instructions are passed on to applications

<?xml version="1.0"?>

<?xml - stylesheet href="book.css" type="text/css"?>

Processing Instruction (Pi): Comment:

|-- this is a comment --

Comments:

for data exchange purposes.

These additional constructs in XML have little or no use

3.2.5 Other XML Constructs
Users can define new entities in XML:

Entities: &lt; stands for >

<![CDATA[&lt;start&gt;this is not an element&lt;/start&gt;]]>

CDATA (character data):

otherwise would be considered markup: used to write escape blocks containing text that
The structure of an entire XML document:

```
<!DOCTYPE name [markup-declarations]

<name> ... name
```

and their associated structure.

that define the tags that are permitted in the document
name of root tag followed by several markup declarations

```
<!DOCTYPE name [markup-declarations]

Document Type Declaration (DTD):
underlying data.

The DTD may serve as the schema for the

XML document and is part of the XML language.

The DTD serves as a grammar for the underlying

3.3 Document Type Definitions (DTDs)
name, age, email are string type
- person has three sub-elements: name, age, email
- db consists of 0 or more persons

<[
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT age (#PCDATA)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT person (name, age, email)>]
  <!ELEMENT db (person*)>
] DOCTYPE db>

A DTD for the above XML document:

  ...  
  <person>
  <email>ab@abcd.com</email>
  </person>

3.3.1 A simple DTD:
person --> name age email
db -- person | person db

In the above example, a DTD is precisely a context-free grammar for the document.

3.3.2 DTD as grammars are some of the other regular expressions allowed.

* | + | | | e | e | e (0 or one)
Consider the relational database example:

- The data types are limited, though.
  - name, age, and email.
- The db requires the person element to have 3 fields:

To a certain extent, DTDs can be used as schemas. For example,

3.3 DTDs as schemas
This DTD correctly constrains r1 elements to have 3 sub-elements: a, b, and c in that order.
followed by 0 or one b, followed by one or more c's.

requires dl element to have exactly one a,

\texttt{\textless \texttt{ELEMENT} dl \texttt{(c',d') \texttt{| c',d')}}

more flexibility in dtd's! for example,

\texttt{\textless \texttt{ELEMENT} dp \texttt{(rr | rz)*\texttt{| (c',d')}}

be interspersed:

but the following change allows rr and rz elements to

the dtd constraints rz elements to follow rr elements,

\texttt{\textless \texttt{ELEMENT} rz \texttt{(c',d') \texttt{| c',d')}}

the dtd to allow different orders:

the order of sub-elements are fixed. we could redefine

\texttt{\textless \texttt{ELEMENT} rr \texttt{(d',c') \texttt{| d',c')}}
where the schema title is available publicly on the Web.

or even

DDFs can be stored in files and the title can be included in the XML document as follows:

```xml
<DOCTYPE db SYSTEM="http://triman.cs.ensu.edu/~ray/schemex.dtd">
```
Both attributes are string type.
Language which is required and department which is optional.
Element name has two attributes:

<ATTLIST price currency CDATA #IMPLIED>
<ATTLIST department CDATA #IMPLIED>
<ATTLIST name language CDATA #REQUIRED

The DTD for this document includes attribute declarations as follows:

<product
 price currency="euro"/><price
 complete="true"/><name
 language="french" department="music"/>

Consider the XML document:

3.3.4 Declaring Attributes in DTD
Example: Family Tree specification

IDREF is used to refer to a list of objects ids separated by spaces

IDREF is used to refer to an object id

ID is used to define object identifier

ID, IDREF attributes:
An XML element that conforms to the above DTD is:

```xml
<family/>
<person/>

<name>Jack Smith/name>
<person id="Jack" mother="Mary" father="John"> -- child</p>

<name>Mary Smith/name>
<person id="Mary" children="Jane Jack"> -- mother
  <person id="Jane" children="John"> -- father
  <person id="John"> -- child</p>

<name>John Doe/name>
<person id="John" children="Jane Jack"> -- father
  <person id="Jane"> -- child</p>

<name>Jane Doe/name>
```
-- Empty indicates that this element will always be empty

</]>

pll:state-of:dept:id #REQUIRED
  pll:state-of:empty
  pll:code (#PCDATA)
  pll:city id
  pll:city (code, code, state-of).
</pll:state-of:dept:id #REQUIRED>

pll:state-of:dept:in:dept:required #REQUIRED
  pll:state-of:empty
  pll:capital-id #REQUIRED
  pll:code (#PCDATA)
  pll:state (code, state, capital, city)
This way citites-in element can be written as:

-- instead of IDREF

<APPLET citites-in=\"c1\" citites-in=\"c2\"></APPLET>

<ELEMENT citites-in=\"required\" #REQUIRED>

<ELEMENT citites-in=\"empty\">

* -- instead of citites-in

<ELEMENT state (code, name, capital, citites-in)>

The above DTD can be modified as follows to take advantage of IDREFs:

The attribute type IDREFS allows an attribute to refer to multiple entities.
<state/>
<state id="st" capital="ct" cities-in=ct=ct c3">

So, a state element can be described as:

cities-in IDREFS #REQUIRED
capital IDREF #REQUIRED

 frightful state id ID #REQUIRED

sub-elements as in:

to make capital and cities-in attributes instead of

Another way to make the representation more compact is
An example illustrating entities that refer to external data using a URL.
<report>
  <meta keywords="xml,www,web,semistructured"
       author="abiteboul,Buneman,Suciu"
       date="25.12.98"/>
  <title>Data on the Web</title>
  %abstract
  %content
</report>
The IDREFS point to:

There is no restriction on the types of objects

IDREFS must refer to existing IDs.

DTD specifications and the tags used must be those
defined in the DTD. IDs must be unique and

1. e. elements must be nested only in the way the

Valid XML document: one which has a DTD AND the document

no duplicate attributes

Well-formed XML document: matching nested tags,'
This is not possible in DDS.

We would like to constrain the type of IDREFS.

- DDS do not constrain the type of IDREFS.
- two different identifiers : personname and courseame
- different structures: in XMI we are forced to use
- ex. name of person and name of course may have
two different element types
- global names (cannot use same identifier for
- not possible
- constraints such as age between 0 and 120
- No atomic types except PCDATA
- DDS impose order

3.3.6 Limitations of DDS as Schemas