Methods and Tools for Management Information Systems

Lecture 2

26. Oktober 2009

XML Syntax

- A simple yet complete example:

```
<person>
  Alan Turing
</person>
```

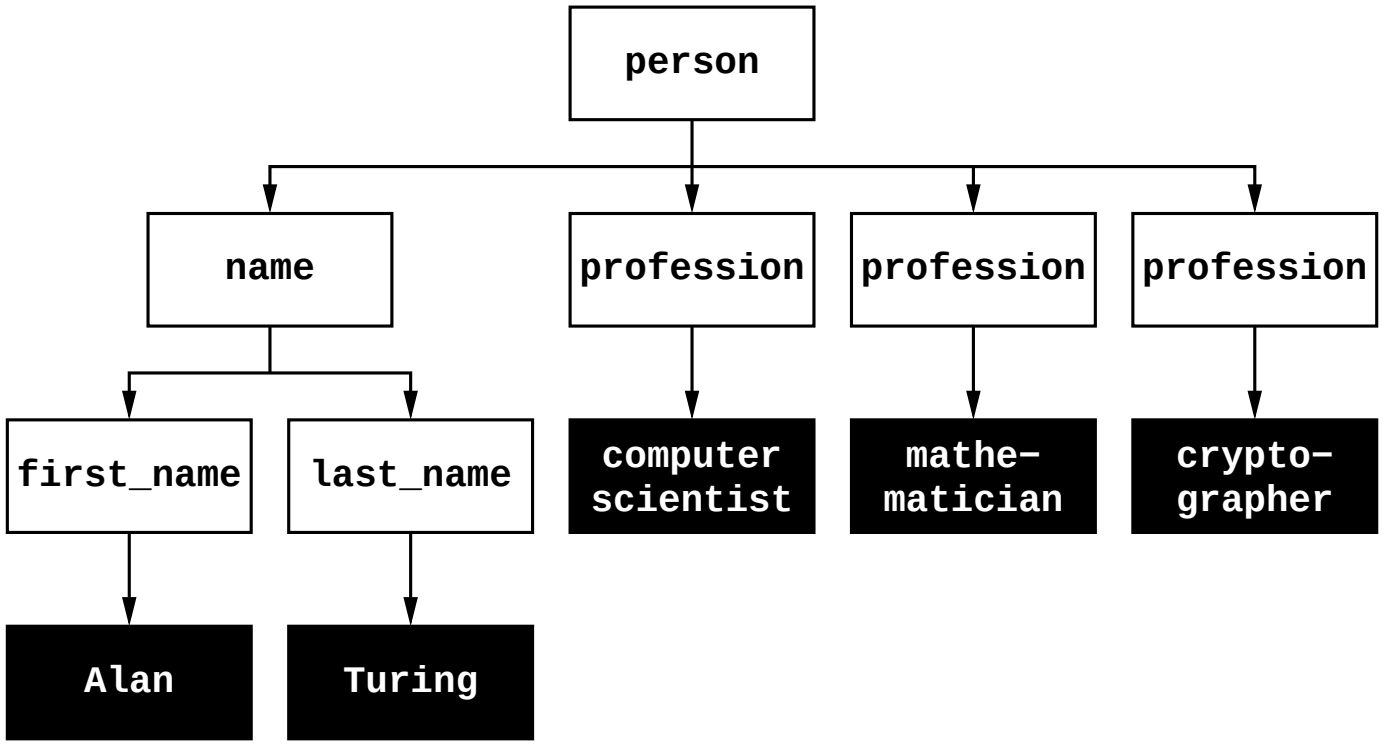
  - No binary data but completely built from text ...
  - Looks like HTML ...
  - Tags describe the meaning of the data rather than its layout ...

- A more complex example:

```
<person born="1912-06-23" died="1954-06-07">
  <name>
    <first_name>Alan</first_name>
    <last_name>Turing</last_name>
  </name>
  <profession>computer scientist</profession>
  <profession>mathematician</profession>
  <profession>cryptographer</profession>
</person>
```

  - Types of elements: root, parent, child ...
  - Attribute(s) ...
  - Element's contents either child elements or data ...

- XML tree for Example 2:



- Another example using attributes instead of child elements:

```
<person born="1912-06-23" died="1954-06-07">
  <name first="Alan" last="Turing">
    </name>
  <profession value="computer scientist">
    </profession>
  <profession value="mathematician"/>
  <profession value="cryptographer"/>
</person>
```

  - Attribute names have to be unique (on the element level)
  - Attributes appropriate for meta data
  - Element-based structure more flexible and extensible

- A narrative-organized XML document:

```
<biography>
  <name><first_name>Alan</first_name> <last_name>Turing</last_name>
  </name> was one of the first people to truly deserve the name
  <emphasize>computer scientist</emphasize>. Although his
  contributions to the field are too numerous to list, his
  best-known are the eponymous <emphasize>Turing Test</emphasize>
  and <emphasize>Turing Machine</emphasize>.

  <definition>The <term>Turing Test</term> is to this day the standard
  test for determining whether a computer is truly intelligent. This
  test has yet to be passed.</definition>

  <!-- ... -->
</biography>
```

- *Things* an XML document may contain:
  - Character data, using specific encodings (UTF-8, UTF-16, etc.)
  - Whitespace, e. g. blanks, tabulators, empty lines, etc.
  - XML names ...
    - for elements und attributes
    - start with a letter or underscore
    - followed by a letter, digit, dot, hyphen or underscore, no whitespace
  - Character references ...
    - decimal, e. g. `&#38;` for & (ampersand)
    - hexadecimal, e. g. '`&#xa9;`' for © (copyright)
  - Predefined entities ('`<`', '`>`', '`&`', '`&apos;`', '`&quot;`')

- `CDATA` (character data) sections ...
  - may not be nested
  - may not contain the character sequence `]]>`
  - Example:

    ```
    <![CDATA[unescaped character & markup data]]>
    ```

- Entity references ...
  - Macro replacement facility, i. e., references are replaced with the entity's text while parsing
  - Two different types:
    - Parameter entity references (`'%name;'`)
    - General entity references (`'&name;'`)

- Entity references (cont'd) ...
  - Example:

```
<!ENTITY % YEAR "2001">
<!ENTITY COPYRIGHT "&#xa9; %YEAR;">

<copyright_notice>&COPYRIGHT;</copyright_notice>

<copyright_notice>© 2001</copyright_notice>
```

- Comments

```
<!-- This is a comment. -->
```


- Processing instructions

  ```
  <?target [processing-instruction data]?>
  ```

  - Escape mechanism to include instructions that are not part of the XML markup or character data
  - target can be any legal XML name, except xml in any combination of upper- and lowercase
  - Example:

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

- XML declaration

```
<?xml version="1.0" [encoding="name"]
                    [standalone="yes"|"no"]?>
```

  - Must be the very first item in a document
  - Version information attribute denotes the version of the XML specification used to create the document
  - Encoding attribute indicates the character-encoding scheme
  - Attribute `standalone` denotes if a document is completely self contained, i. e., the DTD, if there is one, is contained completely within the original document
  - Example:

```
<?xml version="1.0" ecoding="UTF-8" standalone="no"?>
```

- Well-formedness of XML documents:
  - Every document must either use a DTD or set the value of the attribute `standalone` to `no`
  - Documents without DTD may only contain `CDATA` elements and/or attributes
  - Every start-tag must have a matching end-tag
  - Elements may nest, but may not overlap
  - There must be exactly one root element
  - Attribute values must be quoted
  - An element may not have two attributes with the same name
  - Comments and processing instructions may not appear inside tags
  - No unescaped < or & signs may occur in the character data of an element or attribute

- *Things* a Document Type Definition (DTD) may contain:
  - Document type declarations
    - internal

```
<?xml version="1.0"?>

<!DOCTYPE people [
  <!ELEMENT person (name, firstname, birthdate, title?)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT firstname (#PCDATA)>
  <!ELEMENT birthdate (#PCDATA)>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT people (person*)>
]>
```


- external

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE people SYSTEM "people.dtd">
```

- mixed

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE people SYSTEM "people.dtd" [
  <!ELEMENT address (country, city, zipcode, street, number)>
  <!ELEMENT country (#PCDATA)>
  <!ELEMENT city (#PCDATA)>
  <!ELEMENT zipcode (#PCDATA)>
  <!ELEMENT street (#PCDATA)>
  <!ELEMENT number (#PCDATA)>
]>
```


- Element declarations:

  `<!ELEMENT name contents>`

  - `name` must be a valid XML name
  - `contents` may consist of `PCDATA` (Parsed Character Data) and/or a sequence/choice of child elements
  - Cardinality (number of occurrences) of an element:

    `?` – zero or one element
    `*` – any number of elements
    `+` – at least one element

  - If elements contain mixed content ...
    - `PCDATA` must be the first child in the sequence
    - no cardinalities may be given

- Empty elements

  ```
  <!ELEMENT name EMPTY>
  ```

- Unspecified elements

  ```
  <!ELEMENT name ANY>
  ```

Example:

```
<?xml version="1.0"?>
<!DOCTYPE document [
  <!ELEMENT document (heading+, paragraph+)>
  <!ELEMENT heading (letter+, number?)>
  <!ELEMENT paragraph (letter*, number*)>
  <!ELEMENT letter (lower* | upper*)>
  <!ELEMENT lower (a | b | c | d | e)>
  <!ELEMENT upper (A | B | C | D | E)>
  <!ELEMENT number (one* | two* | three* | four*)>
]>
<document>
  <heading>
    <letter/><number/></heading>
  <paragraph/>
</document>
```


- Attribute declarations

```
<!ATTLIST element_name attribute_name type default>
```

  - `element_name` and `attribute_name` must be valid XML names
  - XML attribute types:

`CDATA`	– any string of text
`NMTOKEN`	– all characters valid in XML names
`NMTOKENS`	– list of `NMTOKEN`
enumeration	– list of all possible values
`ID`	– unique XML name
`IDREF`	– reference to an `ID` attribute
`IDREFS`	– list of `IDREF`
`ENTITY`	– name of an unparsed entity

- XML attribute types (cont'd):

  `ENTITIES` – list of `ENTITY`
  `NOTATION` – name of a notation

- Attribute defaults:

  `#IMPLIED` – attribute is optional, no default
  `#REQUIRED` – attribute is mandatory, no default
  `#FIXED` – attribute value is constant and immutable
  literal – actual default value


Examples:

```
<!ATTLIST picture width  CDATA #IMPLIED
                 height CDATA #IMPLIED
                 source CDATA #REQUIRED
                 alt    CDATA #IMPLIED>
<picture source="tower.jpeg" alt="The Tower."/>

<!ATTLIST article year NMTOKEN #REQUIRED>
<article year="1992"/>

<!ATTLIST project milestones_due NMTOKENS #REQUIRED>
<project milestones_due="12-01-2004 02-01-2005">
  Groupware Server Setup
</project>
```


```
<!ATTLIST date day (1 | 2 | 3 | 4 | ... | 31) #REQUIRED>
<!ATTLIST date month (January | ... | December) #REQUIRED>
<!ATTLIST date year (1999 | 2000 | ... | 2009) #REQUIRED>
<date day="27" month="May" year="2005"/>

<!ATTLIST employee SSN ID #REQUIRED>
<!ATTLIST team_member member_id IDREF #REQUIRED>
<!ATTLIST project PID ID #REQUIRED
                  team IDREFS #IMPLIED>
<employee SSN="_277711968"/>
<employee SSN="_274574668"/>
<project PID="project-011">
  <team_member member_id="_277711968">
    Will Smith</team_member>
  <team_member member_id="_274574668">
    John Smith</team_member>
</project>
```

```
<!ATTLIST movie source ENTITY #REQUIRED>
<movie source="movie.mpeg"/>

<!ATTLIST slideshow slides ENTITIES #REQUIRED>
<slideshow slides="slide1 slide2 slide3 slide4 slide5">
  Holidays 2004
</slideshow>
<!NOTATION gif SYSTEM "image/gif">
<!NOTATION tiff SYSTEM "image/tiff">
<!NOTATION jpeg SYSTEM "image/jpeg">
<!NOTATION png SYSTEM "image/png">
<!ATTLIST image type NOTATION (gif | tiff | jpeg | png) #REQUIRED>
<!ATTLIST file name CDATA #REQUIRED>
<image type="gif">
  <file name="peter.gif"/>
</image>
```


- General entity declarations:

```
<!ENTITY name "replacement text">
```

  - `name` must be a valid XML name
  - `replacement text` ...
    - must be well-formed
    - may also contain entity references
    - may not be self-referencing or circular
    - may not be used to define the DTD itself

- External parsed general entities:

  ```
  <!ENTITY name SYSTEM "URI">
  ```

  - `URI` references the replacement text
  - not allowed as attribute values
  - resulting document must be well-formed, i. e., the referenced document must be well-formed but may not contain XML and/or document type declarations
  - referenced document may contain text declarations:

    ```
    <?xml [version="1.0"] encoding="ISO-8859-15"?>
    ```

    similar to XML declarations, but while the `version` attribute is optional, the `encoding` attribute is mandatory

- External unparsed entities and notations:
  ... to reference non-XML data in XML documents, e. g. ASCII texts, multimedia, etc.

```
<!ENTITY name SYSTEM "system_literal" NDATA notation_name>
<!ENTITY name PUBLIC "public_literal" "system_literal"
    NDATA notation_name>
```

Example:

```
<!ENTITY me SYSTEM "http://images.org/me.jpg" NDATA jpeg>

<!ELEMENT image EMPTY>
<!ATTLIST image source ENTITY #REQUIRED>

<image source="me"/>
```


- Notations:

  . . . to describe the character content of an element

```
<!ENTITY notation_name SYSTEM "system_literal">
<!ENTITY notation_name PUBLIC "public_literal">
<!ENTITY notation_name PUBLIC "public_literal" "system_literal">
```

Example:

```
<!NOTATION java_code PUBLIC "Java source code">
<!NOTATION c_code    PUBLIC "C source code">
<!NOTATION perl_code PUBLIC "Perl source code">
<!ELEMENT code_fragment (#PCDATA)>
<!ATTLIST code_fragment
    code_lang NOTATION (java_code | c_code | perl_code) #REQUIRED>
]>
```

- Parameter entities:

```
<!ENTITY % name "replacement text">
<!ENTITY % name SYSTEM "system_literal">
<!ENTITY % name PUBLIC "public_literal" "system_literal">
```

  - Only valid within the context of the DTD
  - As a macro replacement facility ...

```
<!ENTITY % residential_content "address, footage, rooms, baths">
<!ENTITY % rental_content "rent">
<!ENTITY % purchase_content "price">
<!ELEMENT apartment (%residential_content;, %rental_content;)>
<!ELEMENT house (%residential_content;, %purchase_content;)>
```

  - As an import facility for external DTDs ...

```
<!ENTITY % external SYSTEM "external.dtd">
%external;
```

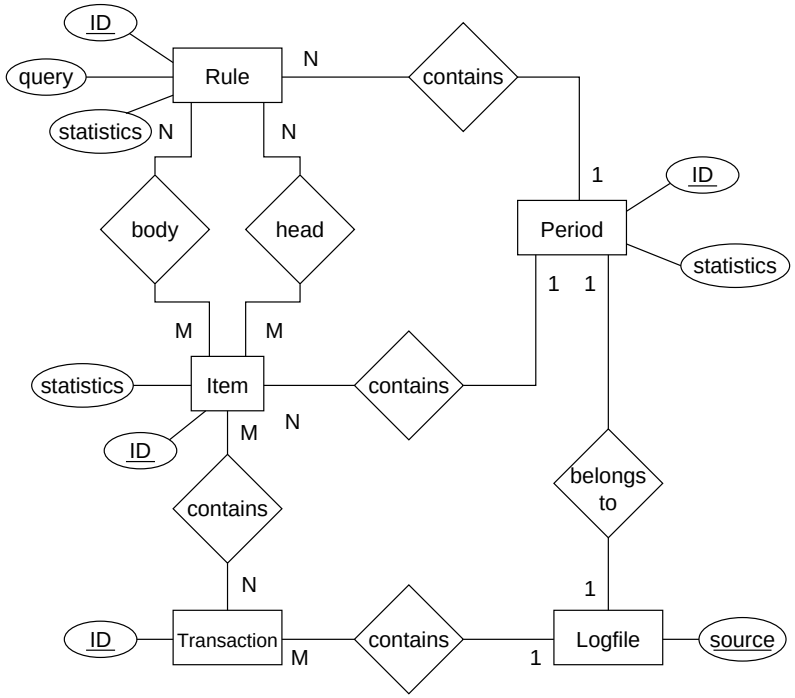
- Externally declared parameter entities can be redefined within the internal part of the DTD
- Conditional sections in the external part of a DTD ...

```
<![IGNORE[
  <!ELEMENT production_note (#PCDATA)> ]]>

<![INCLUDE[
  <!ELEMENT production_note (#PCDATA)>
]]>

<!ENTITY % notes_allowed "INCLUDE">
<![%notes_allowed;[
  <!ELEMENT production_note (#PCDATA)> ]]>
```

Example: A Data Mining Application



```
<!-- rulebase.dtd -->

<?xml version="1.0" encoding="ISO-8859-15"?>

<!ELEMENT rulebase ( ( logfile, period, item*, rule* )* )>

<!ELEMENT logfile ( transaction* )>
  <!ATTLIST logfile filename CDATA #REQUIRED>
<!ELEMENT transaction ( tid )>
  <!ATTLIST transaction tid ID #REQUIRED
                        item IDREFS #REQUIRED>
<!ELEMENT tid ( #PCDATA )>
<!ELEMENT period ( number, cardinality )>
  <!ATTLIST period pid ID #REQUIRED>
<!ELEMENT number ( #PCDATA )>
<!ELEMENT cardinality ( #PCDATA )>
```


```
<!ELEMENT item ( content, cardinality, support )>
  <!ATTLIST item iid ID #REQUIRED
                 period IDREF #REQUIRED>
<!ELEMENT content ( #PCDATA )>
<!ELEMENT support ( #PCDATA )>

<!ELEMENT rule ( query, cardinality, support, confidence )>
  <!ATTLIST rule rid    ID #REQUIRED
                 period IDREF #REQUIRED
                 body   IDREFS #REQUIRED
                 head   IDREFS #REQUIRED>
<!ELEMENT query ( #PCDATA )>
<!ELEMENT confidence ( #PCDATA )>
```

```
<!-- rulebase.xml -->

<?xml version="1.0" encoding="ISO-8859-15"?>

<!DOCTYPE rulebase SYSTEM "rulebase.dtd">

<rulebase>
  <logfile filename="january.log">
    <transaction tid="tid-1" item="iid-ABC iid-DEF">
      <tid>1</tid>
    </transaction>
  </logfile>
  <period pid="pid-1">
    <number>1</number>
    <cardinality>1</cardinality>
  </period>
```

```
  <item iid="iid-ABC" period="pid-1">
    <content>ABC</content>
    <cardinality>1</cardinality>
    <support>1.0</support>
  </item>
  <item iid="iid-DEF" period="pid-1">
    <content>DEF</content>
    <cardinality>1</cardinality>
    <support>1.0</support>
  </item>
  <rule rid="rid-1" period="pid-1" body="iid-ABC" head="iid-DEF">
    <query>support &gt; 0.2, confidence &gt; 0.8</query>
    <cardinality>1</cardinality>
    <support>1.0</support>
    <confidence>1.0</confidence>
  </rule>
</rulebase>
```

- Motivation:
  - To distinguish between elements and attributes from different vocabularies with different meanings (but the same name)
  - To group all the related elements and attributes from a single XML application together
  - To combine markup from multiple XML applications
- Mechanism: qualified names consisting of *prefixes* and *local parts*


- Namespace syntax:
  - Binding prefixes to URIs:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/TR/REC-rdf-syntax#">
  <rdf:Description
      about="http://www.cafeconleche.org/examples/impressionist.xml">
    <title>Impressionist Paintings</title>
    <creator>Elliotte Rusty Harold</creator>
    <description>A list of famous impressionist paintings organized
        by painter and date</description>
    <date>2000-08-22</date>
  </rdf:Description>
</rdf:RDF>
```

    ⇒ Name of prefix is *irrelevant*

- Setting a default namespace:

```
<svg xmlns="http://www.w3.org/2000/svg" width="12cm" height="10cm">
  <ellipse rx="110" ry="130" />
  <rect x="4cm" y="1cm" width="3cm" height="6cm" />
</svg>
```

- Fixed default namespace:

```
<!ATTLIST svg xmlns CDATA #FIXED "http://www.w3.org/2000/svg">

<svg width="12cm" height="10cm">
  <ellipse rx="110" ry="130" />
  <rect x="4cm" y="1cm" width="3cm" height="6cm" />
</svg>
```

- Namespaces are completely independent of DTDs, i. e., namespaces do not change DTD syntax

- Data Mining application revisited ...

```
<?xml version="1.0" encoding="ISO-8859-15"?>

<!ELEMENT rulebase:rulebase ( ( rulebase:logfile, rulebase:period,
    rulebase:item*, rulebase:rule* )* )>
<!ELEMENT rulebase:logfile ( rulebase:transaction* )>
  <!ATTLIST rulebase:logfile rulebase:filename CDATA #REQUIRED>
<!ELEMENT rulebase:transaction ( rulebase:tid )>
  <!ATTLIST rulebase:transaction rulebase:tid ID #REQUIRED
                                 rulebase:item IDREFS #REQUIRED>
<!ELEMENT rulebase:tid ( #PCDATA )>
<!ELEMENT rulebase:period ( rulebase:number, rulebase:cardinality )>
  <!ATTLIST rulebase:period rulebase:pid ID #REQUIRED>
<!ELEMENT rulebase:number ( #PCDATA )>
```

```
<!ELEMENT rulebase:cardinality ( #PCDATA )>

<!ELEMENT rulebase:item ( rulebase:content, rulebase:cardinality,
    rulebase:support )>
  <!ATTLIST rulebase:item rulebase:iid ID #REQUIRED
                          rulebase:period IDREF #REQUIRED>
<!ELEMENT rulebase:content ( #PCDATA )>
<!ELEMENT rulebase:support ( #PCDATA )>

<!ELEMENT rulebase:rule ( rulebase:query, rulebase:cardinality,
    rulebase:support, rulebase:confidence )>
  <!ATTLIST rulebase:rule rulebase:rid    ID #REQUIRED
                          rulebase:period IDREF #REQUIRED
                          rulebase:body   IDREFS #REQUIRED
                          rulebase:head   IDREFS #REQUIRED>
<!ELEMENT rulebase:query ( #PCDATA )>
<!ELEMENT rulebase:confidence ( #PCDATA )>
```

```
<?xml version="1.0" encoding="ISO-8859-15"?>

<!DOCTYPE rulebase:rulebase SYSTEM "rulebase.dtd">

<rulebase:rulebase xmlns:rulebase="http://somewhere.org/rulebase">
  <rulebase:logfile rulebase:filename="january.log">
    <rulebase:transaction rulebase:tid="tid-1"
        rulebase:item="iid-ABC iid-DEF">
      <rulebase:tid>1</rulebase:tid>
    </rulebase:transaction>
  </rulebase:logfile>
  <rulebase:period rulebase:pid="pid-1">
    <rulebase:number>1</rulebase:number>
    <rulebase:cardinality>1</rulebase:cardinality>
  </rulebase:period>
  <!-- .... -->
```


- Validation fails ...

```
$ validate -v -n rulebase_ns.xml
[Error] rulebase_ns.xml:5:67: Attribute "xmlns:rulebase"
must be declared for element type
"rulebase:rulebase".
```

- Adding a namespace attribute ...

```
<!ATTLIST rulebase:rulebase xmlns:rulebase CDATA #IMPLIED>
```

What if the namespace prefix changes?


- Parameter entity references for namespace prefixes:

```
<!ENTITY % ns-prefix "prefix">
<!ENTITY % ns-colon ":">

<!ENTITY % ns-elem1 "%ns-prefix;%ns-colon;elem1">
<!ENTITY % ns-elem2 "%ns-prefix;%ns-colon;elem2">
<!ENTITY % ns-elem3 "%ns-prefix;%ns-colon;elem3">
<!ENTITY % ns-elem4 "%ns-prefix;%ns-colon;elem4">

<!ELEMENT %ns-elem1; (#PCDATA)>
<!ELEMENT %ns-elem2; (#PCDATA)>
<!ELEMENT %ns-elem3; (#PCDATA)>
<!ELEMENT %ns-elem4; ( %ns-elem1; | %ns-elem2; | %ns-elem3; )>

<!ENTITY % ns-prefix "">
<!ENTITY % ns-colon  "">
```

- Data Mining application revisited (2) ...

```
<?xml version="1.0" encoding="ISO-8859-15"?>

<!ENTITY % ns-prefix "rulebase">
<!ENTITY % ns-colon ":">

<!ENTITY % rulebase "%ns-prefix;%ns-colon;rulebase">
<!ENTITY % xmlns-prefix "xmlns">
<!ENTITY % xmlns "%xmlns-prefix;%ns-colon;%ns-prefix;">

<!ENTITY % logfile "%ns-prefix;%ns-colon;logfile">
<!ENTITY % filename "%ns-prefix;%ns-colon;filename">
<!ENTITY % transaction "%ns-prefix;%ns-colon;transaction">
<!ENTITY % tid "%ns-prefix;%ns-colon;tid">
<!ENTITY % item "%ns-prefix;%ns-colon;item">
```

```
<!ENTITY % period "%ns-prefix;%ns-colon;period">
<!ENTITY % pid "%ns-prefix;%ns-colon;pid">
<!ENTITY % number "%ns-prefix;%ns-colon;number">
<!ENTITY % cardinality "%ns-prefix;%ns-colon;cardinality">

<!ENTITY % iid "%ns-prefix;%ns-colon;iid">
<!ENTITY % content "%ns-prefix;%ns-colon;content">
<!ENTITY % support "%ns-prefix;%ns-colon;support">

<!ENTITY % rule "%ns-prefix;%ns-colon;rule">
<!ENTITY % rid "%ns-prefix;%ns-colon;rid">
<!ENTITY % body "%ns-prefix;%ns-colon;body">
<!ENTITY % head "%ns-prefix;%ns-colon;head">
<!ENTITY % query "%ns-prefix;%ns-colon;query">
<!ENTITY % confidence "%ns-prefix;%ns-colon;confidence">
```

```
<!ELEMENT %rulebase; ( ( %logfile;, %period;, %item;*, %rule;* )* )>
  <!ATTLIST %rulebase; %xmlns; CDATA #IMPLIED>

<!ELEMENT %logfile; ( %transaction;* )>
  <!ATTLIST %logfile; %filename; CDATA #REQUIRED>
<!ELEMENT %transaction; ( %tid; )>
  <!ATTLIST %transaction; %tid; ID #REQUIRED
                          %item; IDREFS #REQUIRED>
<!ELEMENT %tid; ( #PCDATA )>

<!ELEMENT %period; ( %number;, %cardinality; )>
  <!ATTLIST %period; %pid; ID #REQUIRED>
<!ELEMENT %number; ( #PCDATA )>
<!ELEMENT %cardinality; ( #PCDATA )>
```

```
<!ELEMENT %item; ( %content;, %cardinality;, %support; )>
  <!ATTLIST %item; %iid; ID #REQUIRED
                   %period; IDREF #REQUIRED>
<!ELEMENT %content; ( #PCDATA )>
<!ELEMENT %support; ( #PCDATA )>

<!ELEMENT %rule; ( %query;, %cardinality;, %support;, %confidence; )>
  <!ATTLIST %rule; %rid; ID #REQUIRED
                   %period; IDREF #REQUIRED
                   %body; IDREFS #REQUIRED
                   %head; IDREFS #REQUIRED>
<!ELEMENT %query; ( #PCDATA )>
<!ELEMENT %confidence; ( #PCDATA )>
```