Key constraints

users.dimi.uniud.it/~massimo.franceschet/caffe-xml/xschema/xschema-keys.html

Key constraints extend the ID/IDREF mechanism of DTD. This mechanism is limited in DTD since it is not possible to specify multi-value keys and it is not possible to restrict the uniqueness domain to a fragment of the entire document. These limitations are overcome in XML Schemas.

Consider the following instance document. It contains a bibliography of either article or book items. Each item is identified by a `key` attribute and has zero, one or more `cite` elements each one pointing, via an `item` attribute, to some bibliographic item:

```
<?xml version="1.0"?>
<bibliography>
  <article key="G03">
    <author>Georg Gottlob</author>
    <title>XPath processing in a nutshell</title>
    <year>2003</year>
    <cite item="HM04"/>
    <cite item="G03"/>
    <journal>SIGMOD Records</journal>
  </article>
  <book key="HM04" isbn="0-596-00764-7">
    <author>Elliotte Harold</author>
    <author>Scott Means</author>
    <title>XML in a nutshell</title>
    <year>2004</year>
    <publisher>O'Reilly</publisher>
  </book>
</bibliography>
```

We could use the usuale ID/IDREF mechanism, encapsulated in XML Schema, to describe the above document and its constraints:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="author" type="xs:string"/>
  <xs:element name="title" type="xs:string"/>
  <xs:element name="year" type="xs:gYear"/>
  <xs:element name="cite" type="citeType"/>
  <xs:element name="journal" type="xs:string"/>
  <xs:element name="publisher" type="xs:string"/>
  <xs:element name="coreItem" type="coreItemType"/>
  <xs:element name="article" type="articleType"/>
  <xs:element name="book" type="bookType"/>
  <xs:element name="bibliography" type="bibliographyType"/>

  <xs:attribute name="isbn" type="xs:string"/>
  <xs:attribute name="key" type="xs:ID"/>
  <xs:attribute name="item" type="xs:IDREF"/>

  <xs:complexType name="coreItemType">
    <xs:sequence>
      <xs:element ref="author" maxOccurs="unbounded"/>
      <xs:element ref="title"/>
      <xs:element ref="year"/>
      <xs:element ref="cite" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute ref="key"/>
  </xs:complexType>

  <xs:complexType name="citeType">
    <xs:attribute ref="item"/>
  </xs:complexType>

  <xs:complexType name="articleType">
    <xs:complexContent>
      <xs:extension base="coreItemType">
        <xs:sequence>
          <xs:element ref="journal"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="bookType">
    <xs:complexContent>
      <xs:extension base="coreItemType">
        <xs:sequence>
          <xs:element ref="publisher"/>
        </xs:sequence>
        <xs:attribute ref="isbn"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="bibliographyType">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="article"/>
      <xs:element ref="book"/>
    </xs:choice>
  </xs:complexType>

</xs:schema>
```

Notice how the key attribute has type ID and the item attribute has type IDREF. These types are built-in primitive types in XML Schema and have the same meaning as in DTDs: an ID attribute must have unique values in the *entire* document, that is, no two attributes of type ID can have the same value. Moreover, an IDREF attribute must contain a valid ID value, that is, a value of any attribute of type ID in the document.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
<xs:element name="author" type="xs:string"/>
<xs:element name="title" type="xs:string"/>
<xs:element name="year" type="xs:gYear"/>
<xs:element name="cite" type="citeType"/>
<xs:element name="journal" type="xs:string"/>
<xs:element name="publisher" type="xs:string"/>
<xs:element name="coreItem" type="coreItemType"/>
<xs:element name="article" type="articleType"/>
<xs:element name="book" type="bookType"/>

<xs:element name="bibliography" type="bibliographyType">
  <xs:key name="biblioKey">
    <xs:selector xpath="*"/>
    <xs:field xpath="@key"/>
  </xs:key>
  <xs:keyref name="biblioKeyRef" refer="biblioKey">
    <xs:selector xpath="*/cite"/>
    <xs:field xpath="@item"/>
  </xs:keyref>
</xs:element>

<xs:attribute name="isbn" type="xs:string"/>
<xs:attribute name="key" type="xs:string"/>
<xs:attribute name="item" type="xs:string"/>

<xs:complexType name="coreItemType">
  <xs:sequence>
    <xs:element ref="author" maxOccurs="unbounded"/>
    <xs:element ref="title"/>
    <xs:element ref="year"/>
    <xs:element ref="cite" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute ref="key"/>
</xs:complexType>

<xs:complexType name="citeType">
  <xs:attribute ref="item"/>
</xs:complexType>

<xs:complexType name="articleType">
  <xs:complexContent>
    <xs:extension base="coreItemType">
      <xs:sequence>
        <xs:element ref="journal"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="bookType">
  <xs:complexContent>
    <xs:extension base="coreItemType">
      <xs:sequence>
```

```
        <xs:element ref="publisher"/>
      </xs:sequence>
      <xs:attribute ref="isbn"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="bibliographyType">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element ref="article"/>
    <xs:element ref="book"/>
  </xs:choice>
</xs:complexType>

</xs:schema>
```

In the above schema document, both key and item attributes have type string. The key constraint is enforced using the XML Schema key element in the declaration of bibliography element, while the key reference constraint is asserted using the XML Schema keyref element in the declaration of bibliography element. XML Schema relies on a subset of XPath for expressing keys and references.

The key element works as follows:

1. the key element must appear as a child of an element element, called the *current element*, following the element type; the name of the key is specified by the name attribute;
2. the *uniqueness domain* of the key, that is the elements that have a value for the defined key, is specified by the selector child of the key element. There must be exactly one selector. The selector has an attribute xpath that contains an XPath expression. The XPath expression is evaluated with the current element as context node and must result in a set of element nodes that forms the uniqueness domain for the key;
3. the value of the key for each node in the uniqueness domain is given by the field child of the key element. If there are more than one field element, then the key is formed by concatenating the value of each field. The field element has an xpath attribute that contains an XPath expression. For each selected node, the XPath expression is evaluated and must result in *at most one* attribute or simple-typed element node.

The result of this process is an **identity-constraint table** for each current node present in the document. This table contains the selected elements and the associated lists of field values. The **key constraint** is satisfied if, and only if, there are no two selected elements having equal lists of field values. Notice that if there are more than one current node in the document, the process is repeated independently for each current node, each generating a difference table.

The XPath expression in the selector must be an abbreviated location path using the child (/) and descendant-or-self (//) axes and the self::* (.) abbreviation only. However, descendant-or-self can only be used at the beginning of the location path in the .// form. Allowed node tests are element name, * and prefix:*. Predicates are disallowed. The vertical bar (|) can be used to union different expressions. For instance, .//author/* | */editor is a valid selector expression, but author//info/@key is not.

The XPath expression in the field is restricted as for the selector with one difference: the abbreviated attribute axis (@) can be used in the last step of the location path. Hence, the expression .//author/@key is legal in the field.

For example, consider the following key constraint extracted from the above schema and applied to the bibliography element:

```
<xs:key name="biblioKey">
  <xs:selector xpath="*"/>
  <xs:field xpath="@key"/>
</xs:key>
```

For each bibliography element (only one is expected in the instance document), the selector retrieves the child elements, which are the bibliography items, whatever are their names, and the only field retrieves the key attribute of the items. Hence, the value of the key attribute must be unique in the domain of all bibliography items.

As another example, consider the following key constraint:

```
<xs:key name="authorKey">
  <xs:selector xpath="author"/>
  <xs:field xpath="."/>
</xs:key>
```

Suppose we apply it to the article element. It states that, for each article, there must be no equal author elements. However, there may be equal authors in different articles. Notice that in this case current elements are author elements, hence there is a uniqueness domain and an identity-constraint table for each article instance in the document.

The selector and field elements of keyref element work as the key element. However, the **keyref constraint** is satisfied if, and only if, each field value list being produced matches a corresponding field value list in the identity-constraint table referred to by the keyref element. For some reason, the XML Schema key element that is referred to must be a child of one of the ancestors of the XML Schema keyref element (this refers to the schema document).

As an example, consider the following keyref constraint extracted from the schema above:

```
<xs:keyref name="biblioKeyRef" refer="biblioKey">
  <xs:selector xpath="*/cite"/>
  <xs:field xpath="@item"/>
</xs:keyref>
```

It asserts that item attributes of cite elements of bibliographic items must point to a valid key associated to some bibliographic item.

One advantage of the key/keyref mechanism in XML Schema is the possibility to specify keys on simple-typed elements and to specify multi-value keys. In the following instance document, for instance, each bibliographic item is identified by a multi-value key composed by the title and the year elements:

```
<?xml version="1.0"?>
<bibliography>
  <article>
    <author>Georg Gottlob</author>
    <title>XPath processing in a nutshell</title>
    <year>2003</year>
    <cite>
      <citeTitle>XML in a nutshell</citeTitle>
      <citeYear>2004</citeYear>
    </cite>
    <cite>
      <citeTitle>XPath processing in a nutshell</citeTitle>
      <citeYear>2003</citeYear>
    </cite>
    <journal>SIGMOD Records</journal>
  </article>
  <book isbn="0-596-00764-7">
    <author>Elliotte Harold</author>
    <author>Scott Means</author>
    <title>XML in a nutshell</title>
    <year>2004</year>
    <publisher>O'Reilly</publisher>
  </book>
</bibliography>
```

The above instance is described by the following schema:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="author" type="xs:string"/>
  <xs:element name="title" type="xs:string"/>
  <xs:element name="year" type="xs:gYear"/>
  <xs:element name="cite" type="citeType"/>
  <xs:element name="citeTitle" type="xs:string"/>
  <xs:element name="citeYear" type="xs:gYear"/>
  <xs:element name="journal" type="xs:string"/>
  <xs:element name="publisher" type="xs:string"/>
  <xs:element name="coreItem" type="coreItemType"/>
```

```
<xs:element name="article" type="articleType">
  <xs:key name="articleAuthorKey">
    <xs:selector xpath="author"/>
    <xs:field xpath="."/>
  </xs:key>
</xs:element>

<xs:element name="book" type="bookType">
  <xs:key name="bookAuthorKey">
    <xs:selector xpath="author"/>
    <xs:field xpath="."/>
  </xs:key>
</xs:element>

<xs:element name="bibliography" type="bibliographyType">
  <xs:key name="biblioKey">
    <xs:selector xpath="*"/>
    <xs:field xpath="title"/>
    <xs:field xpath="year"/>
  </xs:key>
  <xs:keyref name="biblioKeyRef" refer="biblioKey">
    <xs:selector xpath="*/cite"/>
    <xs:field xpath="citeTitle"/>
    <xs:field xpath="citeYear"/>
  </xs:keyref>
</xs:element>

<xs:attribute name="isbn" type="xs:string"/>

<xs:complexType name="coreItemType">
  <xs:sequence>
    <xs:element ref="author" maxOccurs="unbounded"/>
    <xs:element ref="title"/>
    <xs:element ref="year"/>
    <xs:element ref="cite" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="citeType">
  <xs:sequence>
    <xs:element ref="citeTitle"/>
    <xs:element ref="citeYear"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="articleType">
  <xs:complexContent>
    <xs:extension base="coreItemType">
      <xs:sequence>
        <xs:element ref="journal"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```
  <xs:complexType name="bookType">
    <xs:complexContent>
      <xs:extension base="coreItemType">
        <xs:sequence>
          <xs:element ref="publisher"/>
        </xs:sequence>
        <xs:attribute ref="isbn"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="bibliographyType">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="article"/>
      <xs:element ref="book"/>
    </xs:choice>
  </xs:complexType>

</xs:schema>
```

The XML Schema element has also a unique element that works like the key element except for the following difference: in the case of key, the specified field attributes or elements must have a value, in the case of unique, the specified field attributes or elements may be absent. Notice that the nil value of an absent element or attribute is never equal to other values. In particular, it is different from other nil values. A keyref element may refer to a unique constraint as well.

Finally, notice how key and keyref constraints resembles **primary key** and **foreign key** constraints in relational databases. In particular, the uniqueness domain retrieved by the selector of a key constraint corresponds to the table over which the primary key holds in the relational case. Moreover, the unique constraint is similar to the candidate key constraint in relational databases.