Selector and field

Maps to UML TaggedValues named “selector” and “field” of a UML Attribute representing key, keyRef, or unique. The “selector” tag has a value representing the “xpath” and “field,' a list of values representing the field “xpath”. ID values shall be skipped and annotation documentation will be applied to the tagged value according to the annotation rule (see Annotation). For the values annotation field, documentation shall be merged into one.

selector and field mapping to UML attributes

```
<selector
        id = ID

        xpath = a subset of XPath expression, see below

        {any attributes with non-schema namespace…}>

        Content: (annotation?)

</selector>

<field

        id = ID

        xpath = a subset of XPath expression, see below

        {any attributes with non-schema namespace…}>

        Content: (annotation?)

</field>
```

selector and field XML code sample

```
<xs:key name = "fullName">

        <xs:selector xpath=".//person"/>
        <xs:field xpath="forename"/>
        <xs:field xpath="surname"/>
</xs:key>


<xs:keyref name="personRef" refer="fullName">
        <xs:selector xpath=".//personPointer"/>
        <xs:field xpath="@first"/>

        <xs:field xpath="@1ast"/>

</xs:keyref>


<xs:unique name="nearlyID">
        <xs:selector xpath=".//*"/>
        <xs:field xpath="@id"/>
</xs:unique>
```

XML representations for the three kinds of identity-constraint definitions

Sample of XML representations for the three kinds of identity-constraint definitions

```
<xs:element name="state">

        <xs:complexType>

                <xs:sequence>

                        <xs:element name="code" type="twoLetterCode"/>
                        <xs:element ref="vehicle" maxOccurs="unbounded"/>
                        <xs:element ref="person" maxOccurs="unbounded"/>
                </xs:sequence>

        </xs:complexType>
```

```
        <xs:key name="reg"> <!-- vehicles are keyed by their plate within states -->
                <xs:selector xpath=".//vehicle"/>
                <xs:field xpath="@plateNumber"/>
        </xs:key>
</xs:element>

<xs:element name="root">
        <xs:complexType>
                <xs:sequence>
                        …
                        <xs:element ref="state" maxOccurs="unbounded"/>
                        …
                </xs:sequence>
        </xs:complexType>

        <xs:key name="state"> <!-- states are keyed by their code -->
                <xs:selector xpath=" .//state"/>
                <xs:field xpath="code"/>
        </xs:key>

        <xs:keyref name="vehicleState" refer="state"> <!-- every vehicle refers to its state -->
                <xs:selector xpath=".//vehicle"/>
                <xs:field xpath="@state"/>
        </xs:keyref>

        <xs:key name="regKey"> <!-- vehicles are keyed by a pair of state and plate-->
                <xs:selector xpath=".//vehicle"/>
                <xs:field xpath="@state"/>
                <xs:field xpath="@plateNumber"/>
        </xs:key>

        <xs:keyref name="carRef" refer="regKey"› <!-- people's cars are a reference -->
                <xs:selector xpath=".//car"/>
                <xs:field xpath="@regState"/>
                <xs:field xpath="@regPlate"/>
        </xs:keyref>
</xs:element>

<xs:element name="person"›
        <xs:complexType>
                <xs:seguence>
...
                        <xs:element name="car">
                                <xs:complexType>
                                        <xs:attribute name="regState" type="twoLetterCode"/>
                                        <xs:attribute name="regPlate" type="xs:integer"/>
                                </xs:complexType>
                        </xs:element>
                </xs:sequence>
        </xs:complexType>
</xs:element>
```

A state element is defined, which contains a code child and some vehicle and person children. A vehicle in turn has a plateNumber attribute, which is an integer, and a state attribute. State's code s are a key for them within the document. Vehicle's plateNumber s are a key for them within states, and state and plateNumber is asserted to be a key for vehicle within the document as a whole. Furthermore, a person element has an empty car child, with regState and regPlate attributes, which are then asserted together to refer to vehicles via the carRef constraint. The requirement that a vehicle's state match its containing state's code is not expressed here.

For selector and field UML model example, see Keyref.