LanguageManual XPathUDF

Documentation for Built-In User-Defined Functions Related To XPath

UDFs

xpath, xpath_short, xpath_int, xpath_long, xpath_float, xpath_double, xpath_number, xpath_string

- Functions for parsing XML data using XPath expressions.
- Since version: 0.6.0

Overview

The *xpath* family of UDFs are wrappers around the Java XPath library `javax.xml.xpath` provided by the JDK. The library is based on the XPath 1.0 specification. Please refer to http://java.sun.com/javase/6/docs/api/javax/xml/xpath/package-summary.html for detailed information on the Java XPath library.

All functions follow the form: `xpath_*(xml_string, xpath_expression_string)`. The XPath expression string is compiled and cached. It is reused if the expression in the next input row matches the previous. Otherwise, it is recompiled. So, the xml string is always parsed for every input row, but the xpath expression is precompiled and reused for the vast majority of use cases.

Backward axes are supported. For example:

```
> select xpath ('<a><b id="1"><c/></b><b
id="2"><c/></b></a>','/descendant::c/ancestor::b/@id') from t1 limit 1 ;
[1","2]
```

Each function returns a specific Hive type given the XPath expression:

- `xpath` returns a Hive array of strings.
- `xpath_string` returns a string.
- `xpath_boolean` returns a boolean.
- `xpath_short` returns a short integer.
- `xpath_int` returns an integer.
- `xpath_long` returns a long integer.
- `xpath_float` returns a floating point number.
- `xpath_double`,`xpath_number` returns a double-precision floating point number (`xpath_number` is an alias for `xpath_double`).

The UDFs are schema agnostic - no XML validation is performed. However, malformed xml (e.g., `<a><b>1</b></aa>`) will result in a runtime exception being thrown.

Following are specifics on each xpath UDF variant.

xpath

The `xpath()` function always returns a hive array of strings. If the expression results in a non-text value (e.g., another xml node) the function will return an empty array. There are 2 primary uses for this function: to get a list of node text values or to get a list of attribute values.

Examples:

Non-matching XPath expression:

```
> select xpath('<a><b>b1</b><b>b2</b></a>','a/*') from src limit 1 ;
[]
```

Get a list of node text values:

```
> select xpath('<a><b>b1</b><b>b2</b></a>','a/*/text()') from src limit 1 ;
[b1","b2]
```

Get a list of values for attribute 'id':

```
> select xpath('<a><b id="foo">b1</b><b id="bar">b2</b></a>','//@id') from src limit 1
;
[foo","bar]
```

Get a list of node texts for nodes where the 'class' attribute equals 'bb':

```
> SELECT xpath ('<a><b class="bb">b1</b><b>b2</b><b>b3</b><c
class="bb">c1</c><c>c2</c></a>', 'a/*[@class="bb"]/text()') FROM src LIMIT 1 ;
[b1","c1]
```

xpath_string

The `xpath_string()` function returns the text of the first matching node.

Get the text for node 'a/b':

```
> SELECT xpath_string ('<a><b>bb</b><c>cc</c></a>', 'a/b') FROM src LIMIT 1 ;
bb
```

Get the text for node 'a'. Because 'a' has children nodes with text, the result is a composite of text from the children.

```
> SELECT xpath_string ('<a><b>bb</b><c>cc</c></a>', 'a') FROM src LIMIT 1 ;
bbcc
```

Non-matching expression returns an empty string:

```
> SELECT xpath_string ('<a><b>bb</b><c>cc</c></a>', 'a/d') FROM src LIMIT 1 ;
```

Gets the text of the first node that matches '//b':

```
> SELECT xpath_string ('<a><b>b1</b><b>b2</b></a>', '//b') FROM src LIMIT 1 ;
b1
```

Gets the second matching node:

```
> SELECT xpath_string ('<a><b>b1</b><b>b2</b></a>', 'a/b[2]') FROM src LIMIT 1 ;
b2
```

Gets the text from the first node that has an attribute 'id' with value 'b_2':

```
> SELECT xpath_string ('<a><b>b1</b><b id="b_2">b2</b></a>', 'a/b[@id="b_2"]') FROM
src LIMIT 1 ;
b2
```

xpath_boolean

Returns true if the XPath expression evaluates to true, or if a matching node is found.

Match found:

```
> SELECT xpath_boolean ('<a><b>b</b></a>', 'a/b') FROM src LIMIT 1 ;
true
```

No match found:

```
> SELECT xpath_boolean ('<a><b>b</b></a>', 'a/c') FROM src LIMIT 1 ;
false
```

Match found:

```
> SELECT xpath_boolean ('<a><b>b</b></a>', 'a/b = "b"') FROM src LIMIT 1 ;
true
```

No match found:

```
> SELECT xpath_boolean ('<a><b>10</b></a>', 'a/b < 10') FROM src LIMIT 1 ;
false
```

xpath_short, xpath_int, xpath_long

These functions return an integer numeric value, or the value zero if no match is found, or a match is found but the value is non-numeric. Mathematical operations are supported. In cases where the value overflows the return type, then the maximum value for the type is returned.

No match:

```
> SELECT xpath_int ('<a>b</a>', 'a = 10') FROM src LIMIT 1 ;
0
```

Non-numeric match:

```
> SELECT xpath_int ('<a>this is not a number</a>', 'a') FROM src LIMIT 1 ;
0
> SELECT xpath_int ('<a>this 2 is not a number</a>', 'a') FROM src LIMIT 1 ;
0
```

Adding values:

```
> SELECT xpath_int ('<a><b class="odd">1</b><b class="even">2</b><b
class="odd">4</b><c>8</c></a>', 'sum(a/*)') FROM src LIMIT 1 ;
15
> SELECT xpath_int ('<a><b class="odd">1</b><b class="even">2</b><b
class="odd">4</b><c>8</c></a>', 'sum(a/b)') FROM src LIMIT 1 ;
7
> SELECT xpath_int ('<a><b class="odd">1</b><b class="even">2</b><b
class="odd">4</b><c>8</c></a>', 'sum(a/b[@class="odd"])') FROM src LIMIT 1 ;
5
```

Overflow:

```
> SELECT xpath_int ('<a><b>2000000000</b><c>40000000000</c></a>', 'a/b * a/c') FROM
src LIMIT 1 ;
2147483647
```

xpath_float, xpath_double, xpath_number

Similar to xpath_short, xpath_int and xpath_long but with floating point semantics. Non-matches result in zero. However, non-numeric matches result in NaN. Note that `xpath_number()` is an alias for `xpath_double()`.

No match:

```
> SELECT xpath_double ('<a>b</a>', 'a = 10') FROM src LIMIT 1 ;
0.0
```

Non-numeric match:

```
> SELECT xpath_double ('<a>this is not a number</a>', 'a') FROM src LIMIT 1 ;
NaN
```

A very large number:

```
SELECT xpath_double ('<a><b>2000000000</b><c>40000000000</c></a>', 'a/b * a/c') FROM
src LIMIT 1 ;
8.0E19
```

UDAFs

UDTFs