XML 使用指导手册

文档控制

测试数据准备

```
Create TABLE EMPLOYEES
(
id NUMBER,
data XMLTYPE
);

Insert INTO EMPLOYEES
        VALUES (1, xmltype ('<Employees>
      <Employee emplid="1111" type="admin">
            <firstname>John</firstname>
            <lastname>Watson</lastname>
            <age>30</age>
            <email>johnwatson@sh.com</email>
      </Employee>
      <Employee emplid="2222" type="admin">
            <firstname>Sherlock</firstname>
            <lastname>Homes</lastname>
            <age>32</age>
            <email>sherlock@sh.com</email>
      </Employee>
      <Employee emplid="3333" type="user">
            <firstname>Jim</firstname>
            <lastname>Moriarty</lastname>
            <age>52</age>
            <email>jim@sh.com</email>
      </Employee>
      <Employee emplid="4444" type="user">
            <firstname>Mycroft</firstname>
            <lastname>Holmes</lastname>
            <age>41</age>
            <email>mycroft@sh.com</email>
      </Employee>
</Employees>'));
```

1.1. Xmltype 兼容类型

PG 的 xml 数据类型为 xml，oracle 数据类型为 xmltype。为了兼容，创建 xmltype 数据类型：

```
create domain xmltype as xml
check(
    xml_is_well_formed_document(value::text) = 't'::boolean
);

create or replace function xmltype(character varying) returns xml as
```

```
begin
   return $1::xmltype;
end;
```

1.2. XPath 表达式

XPath 使用路径表达式来选择 XML 文档中的节点或节点列表。看下面的列表:

Expression	Description
nodename	选择所有名称为"nodename"的节点
/	选择根节点
//	从当前节点选择文档中相匹配的节点，无论他们在哪里
.	选择当前节点
..	选择当前节点的父节点
@	选择属性
employee	选择所有名称为"employee"的节点
employees/employee	选择所有子节点为 employee 的 employees 节点
//employee	选择所有 employee 的元素，无论他们在哪里

Path Expression	Result
/employees/employee[1]	选择第一个 employee 节点,它是 employees 的子节点。
/employees/employee[last()]	选择最后一个 employee 元素，它是 employees 的子节点
/employees/employee[last()-1]	选择是 employees 子元素的倒数第二个 employee 元素
//employee[@type='admin']	选择所有具有与'admin'的值的属性命名类型的 employee 元素

1.3. XMLTable 函数

1.3.1. 读取 Employees 中所有 firstname 和 lastname

```
SELECT T.ID,
    X.*
FROM EMPLOYEES T,
    XMLTABLE('/Employees/Employee' PASSING T.DATA COLUMNS FIRST_NAME VARCHAR2(30) PATH 'firstname',
                    LAST_NAME VARCHAR2(30) PATH 'lastname') X
WHERE T.ID = 1;
```

Data Output | Explain | Messages | Notifications

	id numeric	first_name character varying (30)	last_name character varying (30)
1	1	John	Watson
2	1	Sherlock	Homes
3	1	Jim	Moriarty
4	1	Mycroft	Holmes

注 XMLTable 函数的语法:

XMLTable('<XQuery>'
PASSING <xml column>
COLUMNS <new column name> <column type> PATH <XQuery path>)

XMLTABLE 函数包含一个 XQuery 行表达式和由一个或多个列表达式组成的 COLUMNS 子句。在上面的语句中，行表达式是 XPath /Employees/Employee。PASSING 子句中的 t.data 指的是 employees 表中的 XML 列中的数据。COLUMNS 子句用于将 XML 数据转换成关系数据，这里每个参数都定义了一个列名和 SQL 数据类型。在上面的查询中，我们定义了 first_name 和 last_name 列并指向 PATH 的 firstname 和 lastname 或者选定的节点。

1.3.2. 使用 text()读取节点值

```
SELECT T.ID,
    X.*
FROM EMPLOYEES T,
    XMLTABLE('/Employees/Employee/firstname' PASSING T.DATA COLUMNS FIRS_TNAME VARCHAR2(30) PATH 'text()') X
```

Data Output | Explain | Messages | Notifications

	id numeric	firs_tname character varying (30)
1	1	John
2	1	Sherlock
3	1	Jim
4	1	Mycroft

1.3.3. 读取所选节点的属性

```
SELECT EMP.ID,
    X.*
FROM EMPLOYEES EMP,
    XMLTABLE('/Employees/Employee' PASSING EMP.DATA COLUMNS FIRSTNAME VARCHAR2(30) PATH 'firstname',
                TYPE VARCHAR2(30) PATH '@type') X;
```

Data Output | Explain | Messages | Notifications

	id numeric	firstname character varying (30)	type character varying (30)
1	1	John	admin
2	1	Sherlock	admin
3	1	Jim	user
4	1	Mycroft	user

1.3.4. 使用 ID 读取特定的记录

```
SELECT T.ID,
    X.*
FROM EMPLOYEES T,
    XMLTABLE('/Employees/Employee[@emplid=2222]' PASSING T.DATA COLUMNS FIRSTNAME VARCHAR2(30) PATH 'firstname',
                    LASTNAME VARCHAR2(30) PATH 'lastname') X
```

Data Output | Explain | Messages | Notifications

	id numeric	firstname character varying (30)	lastname character varying (30)
1	1	Sherlock	Homes

1.3.5. 读取所有类型是 admin 的员工的 firstname 和 lastname

```
SELECT T.ID,
    X.*
FROM EMPLOYEES T,
    XMLTABLE('/Employees/Employee[@type="admin"]' PASSING T.DATA COLUMNS FIRSTNAME VARCHAR2(30) PATH 'firstname',
                    LASTNAME VARCHAR2(30) PATH 'lastname') X
WHERE T.ID = 1;
```

Data Output | Explain | Messages | Notifications

	id numeric	firstname character varying (30)	lastname character varying (30)
1	1	John	Watson
2	1	Sherlock	Homes

1.3.6. 读取年龄超过 40 的所有员工的 firstname 和 lastname

```
SELECT T.ID,
    X.*
FROM EMPLOYEES T,
    XMLTABLE('/Employees/Employee[age>40]' PASSING T.DATA COLUMNS FIRSTNAME VARCHAR2(30) PATH 'firstname',
                    LASTNAME VARCHAR2(30) PATH 'lastname',
                    AGE VARCHAR2(30) PATH 'age') X
WHERE T.ID = 1;
```

Data Output | Explain | Messages | Notifications

	id numeric	firstname character varying (30)	lastname character varying (30)	age character varying (30)
1	1	Jim	Moriarty	52
2	1	Mycroft	Holmes	41

1.4. 其他函数

1.4.1. Xmlforest

将指定列以 xml 格式查询出来,可指定生成的 xml 节点名称

```
1
2  select xmlforest('abc' as "ID");
```

Data Output | Explain | Messages | Notifications

	xmlforest xml
1	<ID>abc</ID>

1.4.2. Xmlelement

为查询出来的 xml 添加挂载的父节点,并将 xml 字符串格式化成 xml ,与 xmlforest 函数配套使用

```
3  select  xmlelement(name "ROW", xmlforest('abc' as "ID"));
```

Data Output | Explain | Messages | Notifications

	xmlelement xml
1	<ROW><ID>abc</ID></ROW>

1.4.3. Extractvalue

查询节点值，不带节点名

```
3  SELECT extractvalue(xml('<a>中</a>'), '/a') FROM dual;
```

Data Output | Explain | Messages | Notifications

	extractvalue text
1	中

1.4.4. Existsnode

判断节点是否存在，表示存在，0 表示不存在

```
3  SELECT existsnode('<a><c>33</c></a>','/a/b') FROM dual;
```

Data Output | Explain | Messages | Notifications

	existsnode bigint
1	0

1.4.5.XML 增删改

```
test=# SELECT updatexml('<a><c id="111">33</c></a>', '/a/c/@id', '55') FROM dual;
     updatexml
---------------------
 <a>                 +
   <c id="55">33</c>+
 </a>

test=# SELECT deletexml('<a><c id="111">33</c></a>', '/a/c/@id') FROM dual;
  deletexml
-------------
 <a>        +
   <c>33</c>+
 </a>

test=# select insertchildxml(xml('<a><b id="属性1">222</b></a>'),'/a/b','@nm','333');
       insertchildxml
-----------------------------------
 <a>                              +
   <b id="属性1" nm="333">222</b>+
 </a>

test=# SELECT appendchildxml(xml('<a><b id="属性1">222</b></a>'),'/a',xml('<d>值1</d>'));
     appendchildxml
-------------------------
 <a>                    +
   <b id="属性1">222</b>+
   <d>值1</d>           +
 </a>
```