xml类型可用于存储XML数据。 使用字符串类型（如text类型） 也可以存储XML数据， 但text类型不能保证其中存储的是合法的XML数据， 通常需要由应用程序来负责保证所输入数据的正确性， 这会增加应用程序的开发难度， 而使用xml类型就不存在此类问题，数据库会对输入的数据进行检查， 使不符合XML标准的数据不能存放到数据库中， 同时还提供了函数对其类型进行安全性检查。

注意， 要使用xml数据类型， 在编译PostgreSQL源码时必须使用以下参数：

configure --with-libxml

**xml类型的输入**

与其他类型类似， 可以使用下面两种语法来输入xml类型的数据：

xml '<pg>hello world</pg>'

'<pg>hello world</pg>'::xml

示例如下：

select xml '<pg>hello world</pg>';

select '<pg>hello world</pg> ::xml;

xml中存储的XML数据有以下两种：

·由XML标准定义的documents。

·由XML标准定义的content片段。

content片段可以有多个顶级元素或character节点。 但documents只能有一个顶级元素。可以使用“xmlvalue IS DOCUMENT”来判断一个特定的XML值是一个documents还是content片段。

PostgreSQL的xmloptions参数用来指定输入的数据是documents还是content片段， 默认情况下此值为content片段， 所以输入的xml可以有多个顶级元素， 但如果我们把此参数设置成“document”， 将不能输入有多个顶级元素的内容， 示例如下：

show xmloption;

select xml '<a>a</a><b>b</b>';

SET xmloption TO document;

select xml '<a>a</a><b>b</b>';--报错

也可以通过函数xmlparse将字符串转换为XML， 使用xmlparse函数是SQL标准中将字串转换成xml的唯一方法。

函数xmlparse的语法如下：

XMLPARSE ( { DOCUMENT | CONTENT } value)

此函数中的参数“DOCUMENT”和“CONTENT”表示指定XML数据的类型。

示例如下：

**字符集的处理**

PostgreSQL数据库在客户端与服务器之间传递数据时， 会自动进行字符集的转换。 如果客户端的字符集与服务端不一致， PostgreSQL会自动进行字符集转换。 但也正是因为这一特性， 用户在传递XML数据时需要格外注意。 对于XML文件来说， 可以通过类似“encoding=”XXX“”的方式指定自己文件的字符集， 但当这些数据在PostgreSQL之间传递时， PostgreSQL会把其原始内容的字符集变成数据库服务端的字符集， 这会导致一系列问题， 因为这意味着XML数据中的字符集编码声明在客户端和服务器之间传递时， 可能变得无效。 为了应对该问题， 提交输入到xml类型的字符串中的编码声明将会被忽略，同时内容的字符集会被认为是当前数据库服务器的字符集。

正确处理XML字符集的方式是， 将XML数据的字符串在当前客户端中编码成当前客户端的字符集， 在发送到服务端后， 再转换成服务端的字符集进行存储。 当查询xml类型的值时， 此数据又会被转换成客户端的字符集， 所以客户端收到的XML数据的字符集就是客户端的字符集。

所以通常来说， 如果XML数据的字符集编码、 客户端字符集编码以及服务器字符集编码完全一致， 那么用PostgreSQL来处理XML数据将会大大减少字符集问题， 并且处理效率也会很高。 通常XML数据都是以UTF-8编码格式进行处理的， 因此把PostgreSQL数据库服务器端编码也设置成UTF-8将是一种不错的选择。

**xml类型函数**

|  |  |  |
| --- | --- | --- |
| 操作符 | 描述 | 例子 |
| xmlcomment | 创建注释 | xmlcomment('hello'); → <!--hello--> |
| xmlconcat | 拼接 | xmlconcat('<id>1</id>', '<name>zhangsan</name>'); → <id>1</id><name>zhangsan</name> |
| xmlelement | 生成元素 | xmlelement(name id); → <id/>  xmlelement(name id, xmlattributes('32' as length)); →  <id length="32"/> |
| xmlagg | 聚合多行 | CREATE TABLE test (id int, info xml);  INSERT INTO test VALUES (1, '<id>1</id>');  INSERT INTO test VALUES (2, '<name>zhangsan</name>');  SELECT xmlagg(info) FROM test;  <id>1</id><name>zhangsan</name> |

PostgreSQL还提供了把数据库中的内容导出成XML数据的函数：

创建测试数据：

CREATE TABLE t1(id int, info text);

INSERT INTO t1 select n, repeat(n::text, 2) from generate\_series(1,5) as n;

select table\_to\_xmlschema('t1'::regclass, true, true, ''); --把表的定义转换成了xml格式

select query\_to\_xmlschema('SELECT \* FROM t1', true, true, ''); --把查询结果中行的定义转成了xml格式

select table\_to\_xml\_and\_xmlschema('t1'::regclass, true, true, '');

select query\_to\_xml\_and\_xmlschema('SELECT \* FROM t1', true, true, '');

schema\_to\_xml、 schema\_to\_xmlschema和schema\_to\_xml\_and\_xmlschema

select schema\_to\_xml('public', true, true, ''); --把schema中的数据全部导成xml格式

select schema\_to\_xmlschema('public', true, true, ''); --只导出schema的定义

select schema\_to\_xml\_and\_xmlschema('public', true, true, '');--导出全部定义及数据

database\_to\_xml、 database\_to\_xmlschema、 database\_to\_xml\_and\_xmlschema：针对某个数据库对象