DB2 XML 文档

拆分指导手册

DB2 XML 文档拆分指导手册

DB2 9 系列产品引入了 PureXML 技术，它能够在数据库表中以原有格式存储 XML 数据。在本次技术手册中，我们将为您讲解如何在 DB2 中对 XML 文档进行拆分。其中包括了经过拆分的 XML 文档示例、XML 的相关语法参数以及使用 XDBDECOMPXML 存储过程进行文档拆分的方法。通过阅读，您可以对 XML 的管理与操作有一个全新的认识。

理解 XML shredding 的概念

在许多 XML 应用场合，XML 数据的结构与使用率都决定了拆分工作的困难。这也就是为什么 DB2 支持 XML 列，使得用户无需转换就可以对 XML 数据进行索引与查询。有些时候，你会发现部分拆分或 XML 混合存储可以最大地满足你的应用需求。

- XML 拆分的优缺点
- 在 DB2 中进行 XML 拆分的两种方法对比

如何用 DB2 的 XMLTABLE 函数拆分 XML

XMLTABLE 函数是一个 SQL 表函数，可以通过 XQuery 表达式从 XML 输入文档中创建关系数据行。在本部分中，我们阐述了如何在 SQL 插入语句中使用 XMLTABLE 函数来实现对 XML 的分割。

- 如何用 DB2 的 XMLTABLE 函数分割 XML
- DB2 中的混合 XML 存储
- 针对 XML 数据的关系型视图

用带有 Annotations 标记的 XML schema 把 XML 文档拆分成关系表

本部分讲述了把 XML 文档分割为关系型表的另一种方法。该方法叫做 annotated schema 分割法，或者叫 annotated schema 解构法，因为它是基于 XML schema 中的 annotations 标记实现的。

- 如何给 XML Schema 增加 Annotations 标记
- 用 IBM Data Studio 定义 Schema Annotations 标记
- 注册带有 Annotations 标记的 Schema

单一与批量处理 XML 文档的示例

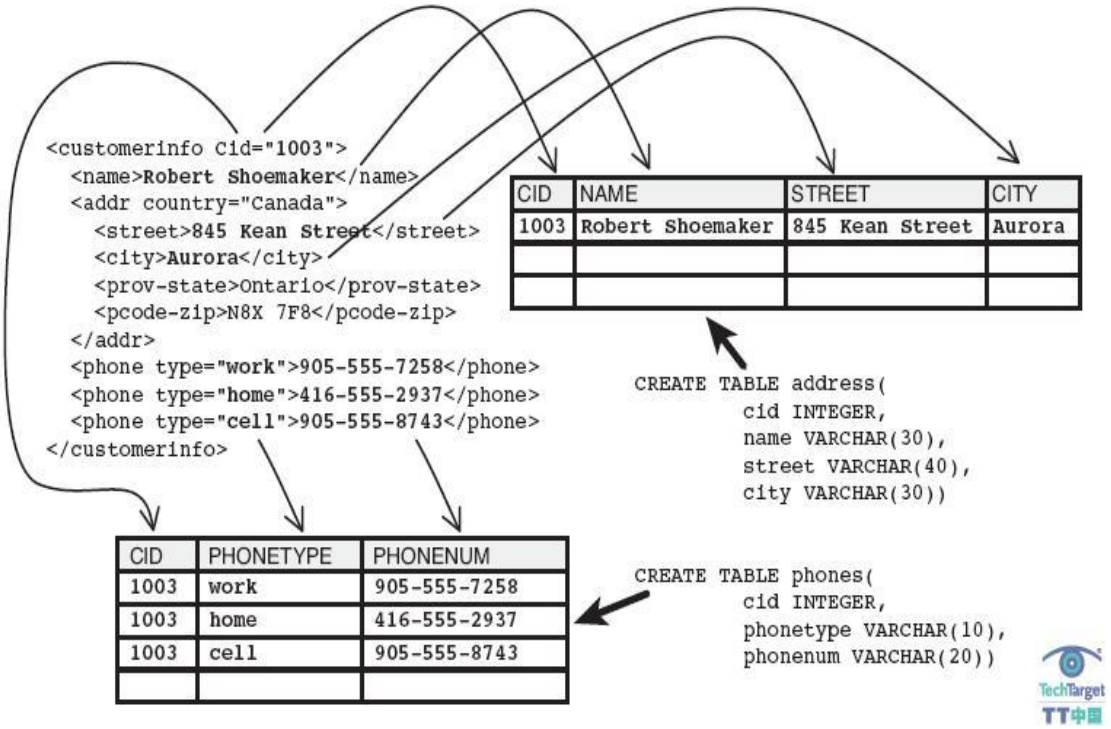
当你考虑把 XML 文档分割到关系型表时，记住 XML 和关系型数据是基于不同的数据模型的。关系型表是扁平的，无顺序的行集，有严格的列类型要求，表中的每一行都必须有相同的结构。使用多个表和表之间的连接来表达一对多的关系。

- 单一 XML 文档拆分示例
- 批量拆分 XML 文档示例
- 总结：在 DB2 中进行 XML 文档拆分

XML 拆分的优缺点

要理解 XML shredding 的概念，可以参考下面的图示。在这个例子中，XML 文档中包含了客户名称、地址以及电话等信息，它们被映射到两个关系表当中。文档可以包含多个电话元素，因为客户与电话是一对多的关系(手机、家庭座机、办公室座机)。所以，电话号码就被拆分到一个单独的表当中。在关系型目标架构中，每一个重复的元素(如电话信息)都可以分配到额外的一个表当中。假设客户信息还包含多个 Email 地址、多个账户、最近订单列表以及每个订单对应的多种商品等重复元素，那么关系型目标架构中的表数量将急剧增加。将 XML 文档拆分成数量巨大的表，你的逻辑业务对象中会产生复杂的碎片，直接导致应用程序开发的难度与出错率提升。而查询拆分的数据或重新组装原始文档需要复杂的多路连接。

XML 文档拆分示例图



相反地，根据 XML 文档的复杂程度、可变性以及用途，适当地进行拆分或许也是一个不错的选择。下表总结了将 XML 文档拆分成关系表的优缺点。

何时需要进行拆分	何时不需要拆分
• 输入的 XML 数据只用于现有的关系型数据库 • XML 文档不代表被保护的逻辑业务对象 • 你的主要目标是让现有的应用可以访问 XML 数据 • 你对现有的关系架构很满意，并希望尽可能多地使用它 • 你的 XML 数据可以比较容易地映射到关系表当中 • 你的 XML 格式相对稳定，并不经常改变 • 你很少重建拆分文件 • 使用 SQL 查询或更新数据比插入性能更加重要	• 你的 XML 数据相互嵌套，非常复杂并且难以映射到关系型架构中 • 将XML格式映射到关系型架构之后会产生数量巨大的表 • 你的XML结构可变性很大并随时间推移而不断变化 • 你的主要目标是作为完整业务对象来管理 XML 文档 • 你需要经常重建拆分文档或其中一部分 • 告诉摄入XML文件到数据库中对你的应用程序非常重要

在许多 XML 应用场合，XML 数据的结构与使用率都决定了拆分工作的困难。这也就是为什么 DB2 支持 XML 列，使得用户无需转换就可以对 XML 数据进行索引与查询。有些时候，你会发现部分拆分或 XML 混合存储可以最大地满足你的应用需求。

- 部分拆分的概念，就是将 XML 文档中元素或属性的一个子集拆分成关系表。当应用不需要 XML 的所有数据值时，这样的方法将很有效。
- XML 混合存储就是当向一个 XML 列插入一个 XML 文档时，抽取选定的元素或属性并将它们冗余存储在关系列中。

无论你选择全部拆分还是部分拆分 XML 文档，DB2 都提供强大的功能，可以实现以下效果：

- 再插入关系列之前，执行数据值的自定义转化。
- 将相同的元素或属性拆分到表的多个列中。
- 将不同的元素或属性拆分到表的同一列中。

- 指定条件来管理是否对相应元素进行拆分。
- 在拆分过程中使用 XML Schema 进行 XML 文档验证。
- 同拆分数据一同存储完整 XML 文档。

（作者：Matthias Nicola,Pav Kumar-Chatterjee 译者：冯昀晖 来源：TT 中国）

原文标题：XML 拆分的优缺点

链接：http://www.searchdatabase.com.cn/showcontent_29822.htm

在 DB2 中进行 XML 拆分的两种方法对比

DB2 9 for z/OS 同 DB2 9.x for Linux, UNIX, and Windows 支持两种拆分方法：

- 使用 XML TABLE 函数的 SQL INSERT 语句。这个函数可以导航到一个输入文档，然后生成一个或多个关系行，用来插入一个关系表。
- 用带注解的 XML Schema 进行分解。由于一个 XML Schema 定义了 XML 文档的结构，因此可以在 Schema 中添加注解来定义元素与属性如何映射。

下面两个表讨论了这两种方法的优缺点：

XMLTABLE 方法的优点	XMLTABLE 方法的缺点
• 无论有没有 XML Schema，你都可以进行数据拆分 • 你不需要会 XML Schema 语言或注解。 • 通常比 Schema 注释方法要简单，因为它是基于 SQL 与 XPath 的。 • 你可以使用熟悉的 XPath, XQuery, 或 SQL 函数来抽取或修改数据值。 • 在进行 XML Schema 演变时不需要做很多事。 • 拆分过程可以使用多个 XML 源的数据，比如 DB2 序列值。 • 性能更佳优秀。	• 对于每一个进行拆分后所生成的表，你都需要使用 INSERT 语句。 • 你需要将多个 INSERT 语句整合到存储过程中。 • 没有 GUI 支持。对 XPath 与 SQL/XML 要足够熟悉。

Schema 注释方法的优点	Schema 注释方法的缺点
• 可在 IBM Data Studio Developer 中使用 GUI 来定义映射。 • 如果是拆分复杂的 XML 数据，使用此方法进行编码比 XMLTABLE 函数方法更简单。 • 它提供批处理模式，当拆分失败时还有详细诊断功能。	• 不支持非 XML Schema 的拆分。 • 当使用新版本的 XML Schema 时，你需要手动拷贝注释。 • 虽然有 GUI 支持，但是你需要书序 XML Schema 语言。 • XML Schema 注释可能会非常复杂，特别是 XML Schema 本身就很复杂的时候。

(作者：Matthias Nicola,Pav Kumar-Chatterjee 译者：孙瑞 来源：TT 中国)

原文标题：在 DB2 中进行 XML 拆分的两种方法对比

链接：http://www.searchdatabase.com.cn/showcontent_29823.htm

如何用 DB2 的 XMLTABLE 函数分割 XML

XMLTABLE 函数是一个 SQL 表函数，可以通过 XQuery 表达式从 XML 输入文档中创建关系数据行。在本文中，我们阐述了如何在 SQL 插入语句中使用 XMLTABLE 函数来实现对 XML 的分割。我们以图 11.1 中的分割方案为例进行讲解。

第一步是创建关系型目标表(如果他们还不存在的话)。对于图 11.1 中的例子，目标表定义如下：

```
CREATE TABLE address(cid INTEGER, name VARCHAR(30),
        street VARCHAR(40), city VARCHAR(30))
CREATE TABLE phones(cid INTEGER, phonetype VARCHAR(10),
        phonenum VARCHAR(20))
```

基于目标表的定义，你可以构造分割输入 XML 文档的插入语句。插入语句必须是“INSERT……INTO……SELECT&……FROM……XMLTABLE”的格式，如图 11.2 所示。每个 XMLTABLE 函数包含一个标记为(“?”)的参数，应用程序可以通过该参数传递要分割的 XML 文档。SQL 类型规则要求参数标记必须映射到合适的数据类型上。SELECT 从句选择由 XMLTABLE 函数产生的列，然后分别插入到地址表和电话表中。

```
INSERT INTO address(cid, name, street, city)
  SELECT x.custid, x.custname, x.str, x.place
  FROM XMLTABLE('$i/customerinfo' PASSING CAST(? AS XML) AS "i"
       COLUMNS
         custid    INTEGER     PATH '@Cid',
         custname  VARCHAR(30) PATH 'name',
         str       VARCHAR(40) PATH 'addr/street',
         place     VARCHAR(30) PATH 'addr/city' ) AS x ;


INSERT INTO phones(cid, phonetype, phonenum)
  SELECT x.custid, x.ptype, x.number
  FROM XMLTABLE('$i/customerinfo/phone'
                PASSING CAST(? AS XML) AS "i"
       COLUMNS
         custid    INTEGER     PATH '../@Cid',
         number    VARCHAR(15) PATH '.',
         ptype     VARCHAR(10) PATH './@type') AS x ;
```

图 11.2 把 XML 元素和属性值插入到关系列中。

要按图 11.1 所示把数据载入这两个目标表，每个插入语句都必须以相同的 XML 文档输入执行。一种做法是应用程序把两个插入语句放到一个事务里，并给每个语句把同一个 XML 文档绑定到参数标记(“?”)上。这种方法运行没有问题，但是我们可以进一步优化一下。因为同一个 XML 文档被从客户端送到服务端两次，在 DB2 服务器端解析了两次，每句插入语句都会解析一次。这部分浪费的性能可以通过绑定两句插入语句到一个存储过程中来避免。这样，该应用只需要做一次存储过程调用，传递一次输入文档就行了，而不用管存储过程中有几句插入语句。第十八章《在存储过程，用户定义函数(UDF)，触发器中使用 XML》讲述了这样一个存储过程，以及其他在存储过程和自定义函数中操作 XML 数据的例子。

另一种选择是，在图 11.2 中的插入语句可以从 XML 列中读入一组入参文档。假定文档都已经被载入到 customer 表的 XML 列信息中。此时，你需要修改图 11.2 中每一句插入语句中的一行来从 customer 表中读取入参文档：

```
FROM customer, XMLTABLE('$i/customerinfo' PASSING info AS "i"
```

如果你需要分割许多文档，那么把输入文档载入分段表是有好处的。载入工具可以并行解析 XML 文档，这样可以节约把这些文档载入数据库的时间。当这些文档以解析好的格式存储到 XML 列中以后，XMLTABLE 函数无需 XML 解析就可以分割这些文档。

插入语句可以通过 XQuery 或者 SQL 函数或者连接语句(join)得以加强，从而针对特定需求调整分割过程。图 11.3 提供了一个例子。SELECT 从句包括了函数 RTRIM 来删除列“x.ptype”尾部的空格。XMLTABLE 函数表达式的生成行包括了把家庭电话号码排除掉，不让这部分数据被分割进目标表的语句。电话号码生成列的表达式使用了 XQuery 函数“normalize-space”(用来清理空格)，“normalize-space”函数会去掉元素前后的空格，替换重复空格为单一空格。该语句还对查找表的地区代码执行了一次连接，这样一来只有地区代码包含在地区代码表中的电话号码会插入到电话表中。

```
INSERT INTO phones(cid, phonetype, phonenum)
  SELECT x.custid, RTRIM(x.ptype), x.number
  FROM areacodes a,
       XMLTABLE('$i/customerinfo/phone[@type != "home"]'
                PASSING CAST(? AS XML) AS "i"
        COLUMNS
          custid    INTEGER     PATH '../@Cid',
          number    VARCHAR(15) PATH 'normalize-space(.)',
          ptype     VARCHAR(10) PATH './@type') AS x
  WHERE SUBSTR(x.number,1,3) = a.code;
```

图 11.3 使用函数和连接定制分割功能。

(作者：Matthias Nicola,Pav Kumar-Chatterjee 译者：冯昀晖 来源：TT 中国)

原文标题：如何用 DB2 的 XMLTABLE 函数分割 XML

链接：http://www.searchdatabase.com.cn/showcontent_30638.htm

DB2 中的混合 XML 存储

在许多情况下，XML 文档结构非常复杂，使得分割非常困难，低效率，而且不方便。除了分割的性能问题，分散 XML 文档的值跨越存储在大量表中，也使得应用程序开发人员很难理解并查询这些数据。要改善 XML 插入性能并降低你数据库中表的数量，你可能想以混合方式存储 XML 文档。这种方法提取选定 XML 元素和属性的值并把它们挨着完整的 XML 文档存储在关系列中。

前面部分的示例使用了两个表作为分割客户文档的目标表，这两个表是地址表和电话表。你可能更希望使用包含关系列“客户 id”，“名称”，和“所在城市”的单个表，把包含重复电话元素的完整的 XML 文档和其他信息放到一个 XML 列中。你可以按下面的语句定义表：

```
CREATE TABLE hybrid(cid INTEGER NOT NULL PRIMARY KEY,

name VARCHAR(30), city VARCHAR(25), info XML)
```

图 11.4 展示了执行迁移到该表的插入语句。XMLTABLE 函数通过参数标记符号（“?”）把 XML 文档作为输入参数。XMLTABLE 函数生成的四个列中的该列定义与目标表混合的定义相匹配。XMLTABLE 函数中的生成行表达式只是“$i”，它会生成完整输入文档。该表达式是 XMLTABLE 函数“COLUMNS”从句生成列表达式的输入。特别要注意的是，列表达式“.”原样返回完整输入文档并为给该目标表插入“info”列生成 XML 列文档。

```
INSERT INTO hybrid(cid, name, city, info)
  SELECT x.custid, x.custname, x.city, x.doc
  FROM XMLTABLE('$i' PASSING CAST(? AS XML) AS "i"
        COLUMNS
          custid   INTEGER     PATH 'customerinfo/@Cid',
          custname VARCHAR(30) PATH 'customerinfo/name',
          city     VARCHAR(25) PATH 'customerinfo/addr/city'
          doc      XML         PATH '.' ) AS x;
```

图 11.4 以混合方式存储 XML 文档。

目前，在 DB2 中还不可能实现定义检查约束限制来确保同一行的一个 XML 文档关系列和值的完整性。然而，你可以给表定义插入和修改的触发器，这样无论何时插入或修改了一个文档，都会自动迁移关系列。

在 DB2 命令行处理器(CLP)中测试这类插入语句很有意义。要做到这一点，你可以把参数标记符(“?”)替换为 XML 文档原文，如图 11.5 所示。XML 文档原文是一个字符串，必须被单引号括起来并用 XMLPARSE 函数转换为 XML 数据类型。同样，你还可以用一个自定义函数从文件系统读取输入文档。用户自定义函数的用法请参见图 11.6 中的内容。

```
INSERT INTO hybrid(cid, name, city, info)
  SELECT x.custid, x.custname, x.city, x.doc
  FROM XMLTABLE('$i' PASSING
    XMLPARSE(document
     '<customerinfo Cid="1001">
        <name>Kathy Smith</name>
        <addr country="Canada">
          <street>25 EastCreek</street>
          <city>Markham</city>
          <prov-state>Ontario</prov-state>
          <pcode-zip>N9C 3T6</pcode-zip>
        </addr>
        <phone type="work">905-555-7258</phone>
      </customerinfo>') AS "i"
        COLUMNS
          custid   INTEGER     PATH 'customerinfo/@Cid',
          custname VARCHAR(30) PATH 'customerinfo/name',
          city     VARCHAR(25) PATH 'customerinfo/addr/city',
          doc      XML         PATH '.' ) AS x;
```

图 11.5 使用 XML 文档原文的混合插入语句。

```
INSERT INTO hybrid(cid, name, city, info)
  SELECT x.custid, x.custname, x.city, x.doc
  FROM XMLTABLE('$i' PASSING
    XMLPARSE(document
      blobFromFile('/xml/mydata/cust0037.xml')) AS "i"
        COLUMNS
          custid   INTEGER     PATH 'customerinfo/@Cid',
          custname VARCHAR(30) PATH 'customerinfo/name',
          city     VARCHAR(25) PATH 'customerinfo/addr/city',
          doc      XML         PATH '.' ) AS x;
```

图 11.6 使用“FromFile”用户自定义函数的混合插入语句。

图 11.4，图 11.5 和图 11.6 中的插入逻辑是相等价的。唯一差异是输入文档的提供方式：通过参数标记(“?”)提供;以单引号括起来的原文字符串提供;或者通过用户自定义函数从文件系统中读取文档的方式提供。

(作者：Matthias Nicola,Pav Kumar-Chatterjee 译者：冯昀晖 来源：TT 中国)

原文标题：DB2 中的混合 XML 存储

链接：http://www.searchdatabase.com.cn/showcontent_30643.htm

针对 XML 数据的关系型视图

你可以针对 XML 数据使用 XMLTABLE 函数创建关系型视图。这种功能可以使你提供给应用程序一个关系型或者一个 XML 数据的混合视图，而无需把数据存储为关系型格式或者混合格式。如果你想避免把大量 XML 数据转换为关系型格式的压力，这一点很有用。前面章节中介绍的在插入语句中使用基本的“SELECT……FROM……XMLTABLE”结构的方式同样也可以被用在创建视图的语句中。

作为举例，假定你想根据客户表中的 XML 文档元素创建一个关系型视图，得到客户标识符，名称，所在街道和城市的值。图 11.7 列出了相应的视图定义，同时列出了对该视图的一个查询。

```
CREATE VIEW custview(id, name, street, city)
AS
  SELECT x.custid, x.custname, x.str, x.place
  FROM customer,
       XMLTABLE('$i/customerinfo' PASSING info AS "i"
        COLUMNS
          custid    INTEGER     PATH '@Cid',
          custname  VARCHAR(30) PATH 'name',
          str       VARCHAR(40) PATH 'addr/street',
          place     VARCHAR(30) PATH 'addr/city' ) AS x;


SELECT id, name FROM custview WHERE city = 'Aurora';

ID          NAME
----------- ------------------------------
       1003 Robert Shoemaker

  1 record(s) selected.
```

图 11.7 根据 XML 数据创建视图。

图 11.7 中对该视图的查询包括了一个 SQL 从句，其中使用视图中的“city”列作为查询条件。“city”列中的值来自于隐藏在 XML 列中的 XML 元素。你可以通过给“customer”表“info”列的“/customerinfo/addr/city”上创建 XML 索引来加快该查询的速度。针对 z/OS(IBM z 系列操作系统)的 DB2 9 版本和针对 Linux，UNIX 和 Windows 的 DB2 9.7 版本可以把关系型从句“city=‘Aurora’”转换为关于潜在 XML 列的 XML 从句，以便 XML 索引可以起作用。这在针对 Linux，UNIX 和 Windows 的 DB2 9.1 版本和 DB2 9.5 版本中是不可能实现的。在前面提到 DB2 的这些版本中，在视图定义中包括了 XML 列而且可以以 XML 从句形式书写查询条件，如下面的查询语句所示。否则，XML 索引不能起作用。

```
SELECT id, name
  FROM custview
WHERE XMLEXISTS('$INFO/customerinfo/addr[city = "Aurora"]')
```

(作者：Matthias Nicola,Pav Kumar-Chatterjee 译者：冯昀晖 来源：TT 中国)

原文标题：针对 XML 数据的关系型视图

链接：http://www.searchdatabase.com.cn/showcontent_30645.htm

如何给 XML Schema 增加 Annotations 标记

本文讲述了把 XML 文档分割为关系型表的另一种方法。该方法叫做 annotated schema 分割法，或者叫 annotated schema 解构法，因为它是基于 XML schema 中的 annotations 标记实现的。这些 annotations 标记定义了你 XML 数据中的 XML 元素和属性如何映射到关系表中列。

要想执行 annotated schema 分割，需要执行以下步骤：

- 确认或者创建要保存分割数据的目标关系表。
- 给 XML Schema 中增加 Annotations 标记，定义从 XML 数据到关系型表的映射。
- 在 DB2 XML Schema 资源库中注册该 XML Schema。
- 使用命令行处理器(CLP)命令或者内建存储过程分割 XML 文档。

假定你已经定义了你想存储分割数据的关系表，我们看看如何给 XML Schema 增加 Annotations 标记。

Schema Annotations 标记是 XML Schema 中附加的元素和属性，用来提供映射信息。DB2 可以使用该信息把 XML 文档分割为关系表。Annotations 标记中的内容不会改变原始 XML 模式的语义。如果 Annotations Schema 的文档是有效的，那么它对原始 Schema 也是有效的，反之亦然。你可以使用增加了 Annotations 标记的 Schema 验证 XML 文档，就像使用原来的 XML Schema 文档一样。

下面一行来自于 XML Schema：

该行内容定义了叫做“street”的 XML 元素，并且声明了它的数据类型是“xs:string”，而且该元素必须至少出现一次。你可以给该元素定义添加一个简单的 Annotations 标记，以表明该元素应该被分割到表“ADDRESS”的列“STREET”中。该 Annotations 标记由该元素定义中的两个附加属性组成，如下所示：

```
db2-xdb:rowSet="ADDRESS" db2-xdb:column="STREET"/>
```

相同的 Annotations 标记还可以被提供为 schema 元素，而不是属性，如下面所示。你会在后面的图 11.8 中明白为什么这一点很有用。

```
< xs:element name="street" type="xs:string" minOccurs="1" >
                    < xs:annotation >
                      < xs:appinfo >
                  < db2-xdb:rowSetMapping >
          < db2-xdb:rowSet >ADDRESS< /db2-xdb:rowSet >
                  STREET< /db2-xdb:column >
                 < /db2-xdb:rowSetMapping >
                      < /xs:appinfo >
                    < /xs:annotation >
                      < xs:element/ >
```

前缀“xs”对属于 XML Schema 的所有结构都有用，而前缀“db2-xdb”只对所有 DB2 专用 schema 注释有用。这就提供了一个清晰的区分，确保增加了 Annotations 标记的 schema 可以验证它与原来的 XML 文档是同样的文档。

有十四种不同类型的 Annotations 标记。它们支持你指定分割什么，在哪里分割，如何过滤或者转换分割的数据，以及以什么样的顺序向目标表执行插入。表 11.4 提供了一个可用 Annotations 标记概览，根据用户任务进行了逻辑分组。部分 Annotations 标记在表 11.5 中有进一步阐述。

表 11.4 Schema Annotations 标记概览和功能分组。

想实现的功能	使用的 Annotations 标记
指定分割保存数据的目标表。	db2-xdb:rowSet db2-xdb:column db2-xdb:SQLSchema db2-xdb:defaultSQLSchema
指定要分割的内容。	db2-xdb:contentHandling
分割时转换数据值。	db2-xdb:expression db2-xdb:normalization db2-xdb:truncate
过滤数据。	db2-xdb:condition db2-xdb:locationPath
映射一个元素或者属性到多个列。	db2-xdb:rowSetMapping
映射多个元素或者属性到同一个列中。	db2-xdb:table
定义行插入目标表的顺序，避免违反参考完整性限制。	db2-xdb:rowSetOperationOrder db2-xdb:order</TD< tr>

表 11.5 XML Schema Annotations 标记功能表

Annotations 标记	功能描述
db2-xdb:defaultSQLSchema	目标表的默认关系 schema。
db2-xdb:SQLSchema	对不同的表覆盖默认的 schema。
db2-xdb:rowSet	该元素或者属性要映射到的表名。
db2-xdb:column	该元素或者属性要映射为的列名。
db2-xdb:contentHandling	对于一个 XML 元素，该 Annotations 标记定义了如何获取要插入目标列的值。你可以只选择该元素（text）的文本值，该元素文本相关内容和它的所有子节点文本（stringValue），或者该元素序列化的 XML（包括所有标签）和所有子节点（serializeSubtree）。如果你省略了这个标记，DB2 会根据各元素的情况选择一个合适的默认值。
db2-xdb:truncate	指定如果一个值的长度大于目标表中列的长度的话，是否应该截取该值。
db2-xdb:normalization	指定如何对待空格：是截取掉，还是合并连续空格，还是保持原样。
db2-xdb:expression	指定要在插入到目标表之前应用到数据上的表达式。
db2-xdb:locationPath	基于 XML 内容过滤。例如，如果他是一个客户地址，那么分割到“cust”表中，如果是员工地址，那就分割到“employee”表中。
db2-xdb:condition	指定值采用的条件。这样只有在所有条件满足时，才给目标表插入数据。
db2-xdb:rowSetMapping	对于一个元素或者属性，支持用户指定多个映射给相同的或者不同的表。
db2-xdb:table	映射多个元素或者属性到单一列中。
db2-xdb:order	指定多个表中行的插入顺序。
db2-xdb:rowSetOperationOrder	把多个“db2-xdb:order” Annotations 标记组合为一组。

为了演示增加了 Annotations 标记的 schema 分解过程，我们采用图 11.1 为例介绍它。假定目标表已经按图 11.1 定义好了。带 Annotations 标记的 schema 如图 11.8，其中定义了期望的映射。第一行黑体字定义了命名空间前缀“db2-xdb”，它用在整个 schema 中，用来将 DB2 专用的 Annotations 标记和常规 XML Schema Annotations 标记进行区分。

Annotations 标记中第一次使用该前缀的是“db2-xdb：defaultSQLSchema”，它定义了目标表的关系模式。下一个出现的 Annotations 标记是原始名称的定义。这两个注释属性【db2-xdb：rowSet=“ADDRESS”】和【db2-xdb：column=“NAME”】给该名称元素定义了该目标表和列。同样，街道和城市元素也映射到了“ADDRESS”表的相应字段。接下来两个 Annotations 标记把电话号码和类别属性映射到“PHONE”表的列中。最后的 Annotations 标记段属于 XML Schema 的 Cid 属性定义。因为 Cid 属性值变成了表“ADDRESS”和表“PHONE”的关联键，这要求使用“annotation”元素，而不是“annotations”属性。第一个“db2-xdb：rowSetMapping”把 Cid 属性映射到了表“ADDRESS”的列“CID”上。第二个“db2-xdb：rowSet”映射把 Cid 属性赋值到表“PHONE”的列“CID”上。

图 11.8 在图 11.1 中实现分割用的带有 Annotations 标记的 Schema。

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:db2-xdb="http://www.ibm.com/xmlns/prod/db2/xdb1" >
  <xs:annotation>
    <xs:appinfo>
    <db2-xdb:defaultSQLSchema>db2admin</db2-xdb:defaultSQLSchema>
    </xs:appinfo>
  </xs:annotation>

  <xs:element name="customerinfo">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string" minOccurs="1"
         db2-xdb:rowSet="ADDRESS" db2-xdb:column="NAME"/>
        <xs:element name="addr" minOccurs="1"
         maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="street" type="xs:string"
               minOccurs="1" db2-xdb:rowSet="ADDRESS"
               db2-xdb:column="STREET"/>
              <xs:element name="city" type="xs:string"
               minOccurs="1" db2-xdb:rowSet="ADDRESS"
               db2-xdb:column="CITY"/>
              <xs:element name="prov-state" type="xs:string"
               minOccurs="1" />
              <xs:element name="pcode-zip" type="xs:string"
               minOccurs="1" />
            </xs:sequence>
            <xs:attribute name="country" type="xs:string" />
          </xs:complexType>
        </xs:element>
        <xs:element name="phone" minOccurs="0"
         maxOccurs="unbounded" db2-xdb:rowSet="PHONES"
         db2-xdb:column="PHONENUM">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="xs:string">
                <xs:attribute name="type" form="unqualified"
                 type="xs:string" db2-xdb:rowSet="PHONES"
                 db2-xdb:column="PHONETYPE"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="Cid" type="xs:integer">
        <xs:annotation>
```

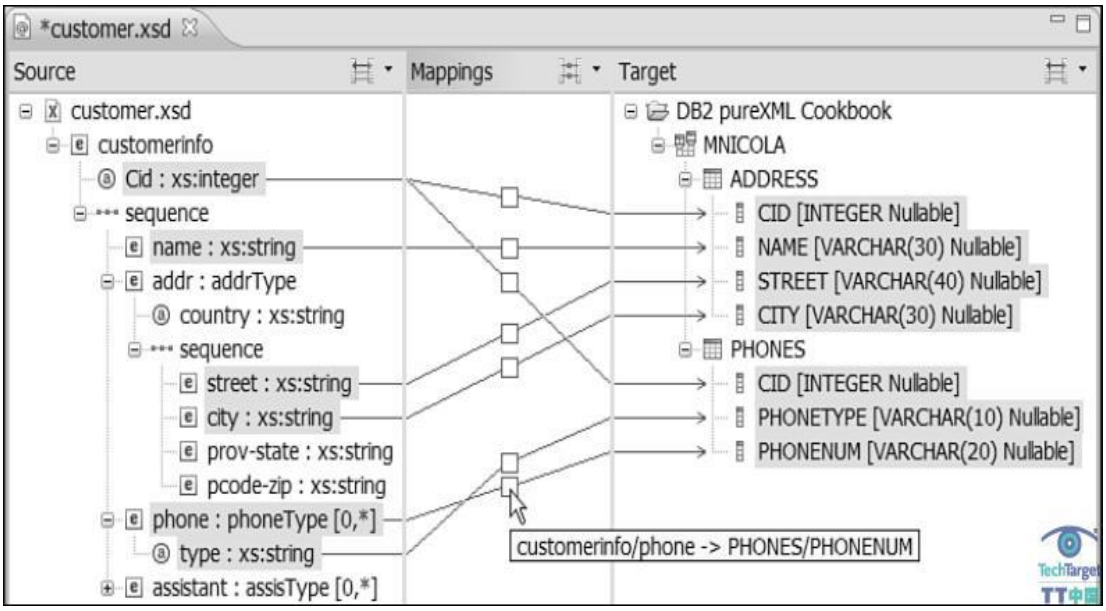
（作者：Matthias Nicola,Pav Kumar-Chatterjee 译者：冯昀晖 来源：TT 中国）

原文标题：如何给 XML Schema 增加 Annotations 标记

链接：http://www.searchdatabase.com.cn/showcontent_30691.htm

用 IBM Data Studio 定义 Schema Annotations 标记

你可以手工给 XML Schema 添加 Annotations 标记，使用任何文本编辑器或者 XML Schema 编辑器都可以。同样，你也可以使用 IBM Data Studio Developer 的“Annotated XSD 映射编辑器”来进行编辑。要打开该编辑器，请在一个 XML Schema 名称上用鼠标右击，选择“打开方式”，在其中选择“Annotated XSD 映射编辑器”。图 11.9 是映射编辑器的一个截图。编辑器左侧显示了 XML Schema(源)定义的文档层次结构。右侧显示关系目标模式(目标)的表和列。你可以通过连接源项到目标列来添加映射关系。还有一个自动搜索功能来找到可能存在的映射关系。已经映射的关系在映射编辑器中会画上从源元素到目标列的线条。



(作者：Matthias Nicola, Pav Kumar-Chatterjee 译者：冯昀晖 来源：TT 中国)

原文标题：用 IBM Data Studio 定义 Schema Annotations 标记

链接：http://www.searchdatabase.com.cn/showcontent_30693.htm

注册带有 Annotations 标记的 Schema

在你创建了带有 Annotations 标记的 XML Schema 以后，你需要把它注册到数据库的 XML Schema 资产库中。对于图 11.8 中已添加 Annotations 标记的 schema，足够执行带有“COMPLETE”和“ENABLE DECOMPOSITION”选项的“REGISTER XMLSCHEMA”命令了，如图 11.10 所示。在本例中，假定 XML Schema 存储在文件“/xml/myschemas/cust2.xsd”中。注册后它被分配给 SQL 标识符“db2admin.dust2xsd”。这个标识符可以在后面用来引用 schema。该命令的“COMPLETE”选项表示没有额外的 XML Schema 文档被添加。“ENABLE DECOMPOSITION”选项表示 XML Schema 不只可以被用来校验，而且可以用来执行分割。

图 11.0 注册已增加 Annotations 标记的 XML schema。

```
REGISTER XMLSCHEMA 'http://pureXMLcookbook.org'
FROM '/xml/myschemas/cust2.xsd'
AS db2admin.cust2xsd COMPLETE ENABLE DECOMPOSITION;
```

从图 11.11 中你可以了解到，可以查询 DB2 目录视图“syscat.xsrobjects”来查看注册的 schema 启用了分解(记为 Y)，还是没有(记为 N)。

图 11.11 检查已标注 Annotations 标记的 XML Schema 状态。

```
SELECT SUBSTR(objectname,1,10) AS objectname,
       status, decomposition
FROM syscat.xsrobjects ;


OBJECTNAME STATUS DECOMPOSITION
---------- ------ -------------
CUST2XSD   C      Y
```

如果某些目标表被删除或者某些目标列被修改了，已注释 schema 的分解状态会自动变成 X(无效状态)，分解功能也会被禁用。这种情况下，接下来试图进行分割 schema 将会

失败，而且不会有警告提示。你还可以使用下面的命令禁用或者启用分割用的 Annotated schema 功能：

```
ALTER XSROBJECT cust2xsd DISABLE DECOMPOSITION;

ALTER XSROBJECT cust2xsd ENABLE DECOMPOSITION;
```

（作者：Matthias Nicola,Pav Kumar-Chatterjee 译者：冯昀晖 来源：TT 中国）

原文标题：注册带有 Annotations 标记的 Schema

链接：http://www.searchdatabase.com.cn/showcontent_30696.htm

单一 XML 文档拆分示例

在你注册并启用了带有 Annotations 标记的 XML Schema 之后，你可以用“DECOMPOSE XML DOCUMENT”命令或者内建的存储过程分解 XML 文档。“DECOMPOSE XML DOCUMENT”命令在 DB2 命令行处理器(CLP)中用起来很方便，而存储过程可以在应用程序或者 CLP 中调用。CLP 命令有两个输入参数：要分割的 XML 文档文件名和带 Annotations 标记 schema 的 SQL 识别符。请看下面的例子：

```
DECOMPOSE XML DOCUMENT /xml/mydocuments/cust01.xml

XMLSCHEMA db2admin.cust2xsd VALIDATE;
```

关键字“VALIDATE”是可选的，它用来标识 XML 文档在分割过程中，是否需要被该 schema 验证。在分割过程中，DB2 会遍历 XML 文档和带 Annotations 标记的 schema 文件，并检测基本的 schema 约束是否被违反，尽管没有指定“VALIDATE”关键字。例如，如果一个必填元素没有填写，尽管该元素不会被分割，而且“VALIDATE”关键字也没有指定，分割过程也会失败并报错。同样，无关元素或者数据类型违反也会引起分解失败。原因是分割过程会马上浏览一遍该带 Annotations 标记的 XML Schema 和示例文档，而且会“无条件”检测许多 schema 的非法属性，尽管 XML 解析器并没有执行验证。

要从应用程序中分解 XML 文档，需要使用存储过程“XDBDECOPXML”。该存储过程的参数如下图 11.12 所示，参数详细说明在表 11.6 中。

```
>>-XDBDECOMPXML--(--rschema--,--xmlschemaname--,--xmldoc--,---->

>--documentid--,--validation--,--reserved--,--reserved--,------>

>--reserved--)-------------------------------------------------><
```

图 11.12 存储过程“XDBDECOPXML”的语法和参数。

参数	说明
rschema	两部分 Annotations 标记的 XML Schema SQL 识别符的关系 schema 部分。例如，如果 XML Schema 的 SQL 识别符是“db2admin.cust2xsd”，那么你应该把字符串“db2admin”传递到这个参数中。在 z/OS 的 DB2 中，该值必须是“SYSXSR”或者“NULL”。
xmlschemaname	两部分 Annotations 标记的 XML Schema SQL 识别符的关系 schema 的第二部分。如果 XML Schema 的 SQL 识别符是“db2admin.cust2xsd”，那么你应该把“cust2xsd”传递到这个参数中。该值不能为“NULL”。
Xmldoc	在 Linux，UNIX 和 Windows 版的 DB2 中，该参数是 BLOB（1M）类型，接受待分解的 XML 文档。在 z/OS 的 DB2 中该参数是“CLOB AS LOCATOR”类型。该参数不能为“NULL”。
documented	调用者可以用来指定输入 XML 文档的字符串。提供的值将被用来替换在 Annotations 标记“db2-xdb:expression”或者“db2-xdb:condition”中任何地方的“$DECOMP_DOCUMENTID”变量。
Validation	可能的值有“0”（无验证）和“1”（执行验证）。该参数在 z/OS 的 DB2 中不存在。
reserved	留给将来使用的保留参数。这些变量传递的值必须是“NULL”。该参数在 z/OS 的 DB2 中不存在。

表 11.6 存储过程“XDBDECOPXML”的参数说明。

下图 11.13 中展示了一段 Java 代码片段，其中使用了参数标记（“?”）调用了存储过程。

图 11.13 调用存储过程“XDBDECOMPXML”的 Java 代码段。

```
CallableStatement callStmt = con.prepareCall(
"call SYSPROC.XDBDECOMPXML(?,?,?,?,?, null, null, null)");

File xmldoc = new File("c:\mydoc.xml");
FileInputStream xmldocis = new FileInputStream(xmldoc);

callStmt.setString(1, "db2admin" );
callStmt.setString(2, "cust2xsd" );

// document to be shredded:
callStmt.setBinaryStream(3,xmldocis,(int)xmldoc.length() );

callStmt.setString(4, "mydocument26580" );

// no schema validation in this call:
callStmt.setInt(5, 0);

callStmt.execute();
```

如果在 z/OS 的 DB2 中，XML 文档的输入参数是“CLOB AS LOCATOR”类型，在 Linux，Unix 和 Windows 版本的 DB2 中就是 BLOB(1M)类型。如果你预计你的 XML 文档会大于 1MB，那就使用表 11.7 中列出的一个存储过程。这些存储过程都是相等价的，除了他们的名称和输入的 XML 文档参数大小不同。在你调用存储过程的时候，DB2 会根据声明的入参大小分配内存。例如，如果你的所有输入文档大小最多 10MB，那存储过程“XDBDECOMPXML10MB”就是一个不错的选择，他可以最合理地利用内存。

存储过程	文档大小	支持版本
XDBDECOMPXML	≤1MB	DB2 9.1
XDBDECOMPXML10MB	≤10MB	DB2 9.1
XDBDECOMPXML25MB	≤25MB	DB2 9.1
XDBDECOMPXML50MB	≤50MB	DB2 9.1
XDBDECOMPXML75MB	≤75MB	DB2 9.1
XDBDECOMPXML100MB	≤100MB	DB2 9.1
XDBDECOMPXML500MB	≤500MB	DB2 9.5 FP3
XDBDECOMPXML1GB	≤1GB	DB2 9.5 FP3
XDBDECOMPXML1_5GB	≤1.5GB	DB2 9.7
XDBDECOMPXML2GB	≤2GB	DB2 9.7

表 11.7 不同大小的文档使用不同的存储过程(Linux，UNIX 和 Windows 版本的 DB2)。

至于平台兼容性，z/OS 的 DB2 支持存储过程“XDBDECOMPXML100MB”，入参与 Linux，UNIX 和 Windows 版本的 DB2 的相同，包括对参数的校验。

(作者：Matthias Nicola,Pav Kumar-Chatterjee 译者：冯昀晖 来源：TT 中国)

原文标题：单一 XML 文档拆分示例

链接：http://www.searchdatabase.com.cn/showcontent_30799.htm

批量拆分 XML 文档示例

Linux，UNIX 和 Windows 版本的 DB2 9.7 版引入了一个新的存储过程叫做“XDB_DECOMP_XML_FROM_QUERY”。它利用带有 Annotations 标记的 schema 来分解一个或者多个从类型为“XML”，“BLOB”或者“VARCHAR FOR BIT DATA”的列选择出来的 XML 文档。它与存储过程“XDBDECOMPXML”的主要差异在于，“XDB_DECOMP_XML_FROM_QUERY”把 SQL 查询看作参数，并可以执行它来从 DB2 表中获取输入文档。对于大数量的文档，“LOAD”操作是批量进行分解的，它比对每个文档依次调用存储过程分别分割效率更高。图 11.14 展示了这个存储过程的参数。“commit_Count”参数和“allow_access”参数是输出参数，提供批量分割过程的结果信息。所有参数都在表 11.8 中有介绍说明。

```
>>--XDB_DECOMP_XML_FROM_QUERY--(--rschema--,--xmlschema--,-->

>--query--,--validation--,--commit_count--,--allow_access--,---->

>--reserved--,--reserved2--,--continue_on_error--,--------------->

>--total_docs--,--num_docs_decomposed--,--result_report--)--><
```

图 11.14 存储过程“XDB_DECOMP_XML_FROM_QUERY”。

参数	说明
rschema	等同于“XDBDECOMPXML”。
xmlschema	等同于“XDBDECOMPXML”的“xmlschemaname”。
query	类型 CLOB（1GB）的查询字符串，不能为“NULL”。查询必须是一个 SQL 或者 SQL/XML 查询语句，而且必须返回两列。第二列是 XML 文档的结果集，第一列必须包含唯一的对应于第一列 XML 文档的文档标识符。第二列包含待分割的 XML 文档，它的类型必须是“XML”,“BLOB”,“VARCHAR FOR BIT DATA”或者“LONG VARCHAR FOR BIT DATA”。
validation	可能的值为：“0”（无验证）和“1”（执行验证）。
commit_count	大于等于“0”的整数值。值“0”表示存储过程不执行任何提交。值“n”表示在每成功分解“n”个文档后执行一次提交。
allow_access	值为“1”或者“0”。如果值为“0”，那么存储过程对带有 Annotations 标记的 XML Schema 引用到的所有表会获取一个排它锁。如果值是“1”，那么存储过程会采用共享锁。
reserved, reserved2	这些是保留将来使用的参数，必须为“NULL”。
continue_on_error	可以是“1”或者“0”。值“0”表示存储过程在遇到第一个不能分解的文档时就会停止执行。例如：如果文档不能匹配 XML Schema。
total_docs	表示存储过程试图分解文档数量的输出参数。
num_docs_decomposed	表示成功分解的文档数量的输出参数。
result_report	类型为 BLOB（2GB）的输出参数。它包含一个 XML 文档，其中提供了给每一个分解不成功的文档的诊断信息。如果所有的文档分割都成功了，则不会生成该报告。这里选择 BLOB 字段（而不是 CLOB）的原因是为了避免代码页转换和潜在的截取或者数据丢失，如果应用代码页与数据库代码页差异很大的话。

表 11.8 存储过程“XDB_DECOMP_XML_FROM_QUERY”的参数。

图 11.15 展示了在 CLP 中一次存储过程“XDB_DECOMP_XML_FROM_QUERY”的调用过程。这个存储过程调用读取“customer”表“info”列中的所有 XML 文档，并用 Annotations

标记 XML Schema“db2admin.cust2xsd”把他们分割。该存储过程每 25 个文档提交一次，即使有文档没有被分割成功的话，也不会停止执行。

```
call SYSPROC.XDB_DECOMP_XML_FROM_QUERY
('DB2ADMIN', 'CUST2XSD', 'SELECT cid, info FROM customer',
  0, 25, 1, NULL, NULL, '1',?,?,?) ;

  Value of output parameters
  --------------------------
  Parameter Name  : TOTALDOCS
  Parameter Value : 100

  Parameter Name  : NUMDOCSDECOMPOSED
  Parameter Value : 100

  Parameter Name  : RESULTREPORT
  Parameter Value : x''

  Return Status = 0
```

图 11.15 调用存储过程“SYSPROC.XDB_DECOMP_XML_FROM_QUERY”。

如果你需要频繁地在 CLP 中执行批量分割，那就使用命令“DECOMPOSE XML DOCUMENTS”，而不要使用存储过程。使用命令行更方便，而且与存储过程“XDB_DECOMP_XML_FROM_QUERY”执行的是相同的任务。图 11.16 列出了该命令的语法。该命令的各种从句和关键字与相应的存储过程参数意义相同。例如：“query”是提供这些输入文档的查询语句，而“xml-schema-name”是两部分 Annotations 标记 XML Schema 的 SQL 识别符。

```
>>-DECOMPOSE XML DOCUMENTS IN----'query'----XMLSCHEMA------->

                                  .-ALLOW NO ACCESS-.
>--xml-schema-name--+----------+--+-----------------+---------->
                    '-VALIDATE-'  '-ALLOW ACCESS----'

>--+----------------------+--+------------------+-------------->
   '-COMMITCOUNT--integer-'  '-CONTINUE_ON_ERROR-'

>--+--------------------------+-------------------------------><
   '-MESSAGES--message-file-'
```

图 11.16 “DECOMPOSE XML DOCUMENTS”命令的语法。

图 11.17 介绍了 DB2 命令行处理器(CLP)中的“DECOMPOSE XML DOCUMENTS”命令。

```
DECOMPOSE XML DOCUMENTS IN 'SELECT cid, info FROM customer'
XMLSCHEMA db2admin.cust2xsd MESSAGES decomp_errors.xml ;

DB216001I  The DECOMPOSE XML DOCUMENTS command successfully
  decomposed all "100" documents.
```

图 11.17 “DECOMPOSE XML DOCUMENTS”命令的示例。

如果你不指定“message-file”，那么错误报告会被写到标准输出。图 11.18 列出了一个错误报告示例。对于每一个分割失败的文档，错误报告会显示出该文档的识别符(xdb:documented)。这个识别符是从“DECOMPOSE XML DOCUMENTS”SQL 语句产生的在第一列中获得的。错误报告还包括每一个分割失败的文档的 DB2 错误信息。图 11.18 显示：文档 1002 包含了一个意料之外的 XML 属性调用状态，而文档 1005 包含一个无效的元素或者属性值“abc”，因为该 XML Schema 预期找到一个类型为“xs:integer”的整数值。

```
<?xml version='1.0' ?>
<xdb:errorReport
     xmlns:xdb="http://www.ibm.com/xmlns/prod/db2/xdb1">
   <xdb:document>
      <xdb:documentId>1002</xdb:documentId>
      <xdb:errorMsg>SQL16271N Unknown attribute "status" at or
       near line "1" in document "1002".</xdb:errorMsg>
   </xdb:document>
   <xdb:document>
      <xdb:documentId>1005</xdb:documentId>
      <xdb:errorMsg> SQL16267N An XML value "abc" at or near
        line "1" in document "1005" is not valid according to
        its declared XML schema type "xs:integer" or is outside
        the supported range of values for the XML schema type
      </xdb:errorMsg>
   </xdb:document>
</xdb:errorReport>
```

图 11.18 批量分割 XML 文档的错误报告示例。

（作者：Matthias Nicola,Pav Kumar-Chatterjee 译者：冯昀晖 来源：TT 中国）

原文标题：批量拆分 XML 文档示例

链接：http://www.searchdatabase.com.cn/showcontent_30801.htm

总结：在 DB2 中进行 XML 文档拆分

当你考虑把 XML 文档分割到关系型表时，记住 XML 和关系型数据是基于不同的数据模型的。关系型表是扁平的，无顺序的行集，有严格的列类型要求，表中的每一行都必须有相同的结构。使用多个表和表之间的连接来表达一对多的关系。与之相反的是，XML 文档倾向于有一个分级的，嵌套的结构，可以在一个单一文档中表现多个一对多关系。XML 允许元素重复任意次，而且 XML Schema 可以定义数以百计或者数以千计的可选元素和属性，它们可能在任何给定的文档中存在，也可以不存在。由于这些差异，分割 XML 数据到关系型表是困难的，低效的，有时甚至是过度复杂的。

如果你的 XML 数据结构复杂度有限，比较容易映射为关系型表，而且 XML 格式不可能随着时间变更，那么 XML 分割有时对满足现存关系型应用和报表软件是有好处的。

DB2 为分割 XML 数据提供了两种方法。第一种方法采用 SQL 插入语句，使用“XMLTABLE”函数。每一个目标表需要一个这样的插入语句，而多个语句可以被合并到一个存储过程中来避免重复解析相同的 XML 文档。这种分割语句中可以包括 XQuery 和 SQL 函数，对其他表的连接，或者对 DB2 序列的引用。这些特性支持定制，在分割过程中具有高度的灵活性，但是需要手工编码。第二种分割 XML 数据的方法采用 Annotations 标记的 XML Schema 来定义从 XML 向关系型表和列的映射。IBM Data Studio Developer 提供了一个可视化的界面，用来方便的地创建这种映射，需要很少或者不需要手工编码。

(作者：Matthias Nicola,Pav Kumar-Chatterjee 译者：冯昀晖 来源：TT 中国)

原文标题：总结：在 DB2 中进行 XML 文档拆分

链接：http://www.searchdatabase.com.cn/showcontent_30804.htm