

Как зарегистрировать схему XML в XML DB и как ЭТИМ ВОСПОЛЬЗОВАТЬСЯ

Обновление ноябрем 2007 года

Владимир Пржиялковский

Преподаватель технологий Oracle
prz@yandex.ru,
www.ccas.ru/prz

*Чиновник по особым порученьям,
Который их до места проводил,
С заботливым Попова попеченьем
Сдал на руки дежурному.*

А. К. Толстой. Сон Попова

Введение

Эта статья является продолжением ранее опубликованных статей «XML DB – новое измерение в организации данных в Oracle» и «Что дает репозиторий XML DB и как с ним работать». Здесь говорится о регистрации в БД пользователя схем XML, что возможно после установки XML DB (об этом рассказывалось ранее). Показано, как выполняется регистрация, и как зарегистрированную схему XML можно использовать при работе с данными типа XMLTYPE.

Как зарегистрировать схему XML

Рассмотрим несложную схему XML, определение которой создадим в файле *bookCover.xsd*:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="cover" type="coverType"/>
  <xsd:complexType name="coverType">
    <xsd:sequence>
      <xsd:element name="title" type="xsd:string"/>
      <xsd:element name="author" type="xsd:string"
        maxOccurs="unbounded"/>
      <xsd:element name="publisher" type="xsd:string"/>
      <xsd:element name="pubdate" type="xsd:string"/>
      <xsd:element name="isbn" type="xsd:string"/>
      <xsd:element name="pages" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Поместим файл *bookCover.xsd* в репозиторий XML DB в папку */public* (любым методом: командной строкой, программно, графическим способом, по WebDAV или же по FTP). Получим ресурс */public/bookCover.xsd* репозитория. Зарегистрируем схему, являющуюся содержанием этого ресурса, с помощью пользователя SCOTT.

Для возможности регистрации схемы пользователь БД должен обладать привилегиями

```
ALTER SESSION
CREATE TYPE
```

```
CREATE TABLE
CREATE PROCEDURE
CREATE TRIGGER
```

В версии 10 содержание роли CONNECT было сведено только к привилегии CREATE SESSION, поэтому после версии 9 пользователю SCOTT потребуется так или иначе выдать привилегию ALTER SESSION, например:

```
CONNECT / AS SYSDBA

GRANT ALTER SESSION TO scott;

CONNECT scott/tiger
```

Следующая далее команда не является технологической необходимостью и выдается для возможности дальнейшего анализа произошедшего:

```
ALTER SESSION SET EVENTS='31098 trace name context forever';
```

Регистрация:

```
BEGIN
DBMS_XMLSCHEMA.REGISTERSCHEMA (
  schemaur1 => 'http://localhost:8080/public/bookCover.xsd'
, schemadoc => XDBURITYPE ( '/public/bookCover.xsd' )
);
END;
/
```

Описание схемы не обязательно заводить в репозитории в качестве ресурса. Для регистрации можно воспользоваться и внешним файлом, явно указанным документом XML или же документом, указанным через внешний URL (подтипы URITYPE).

Проверка:

```
SQL> COLUMN schema_url FORMAT A50 WORD
SQL> COLUMN local          FORMAT A5
SQL> SELECT  schema_url, local FROM user_xml_schemas;
```

SCHEMA_URL	LOCAL
-----	-----
http://localhost:8080/public/bookCover.xsd	YES

Интерес представляют также другие поля таблицы USER_XML_SCHEMAS, не указанные в этом запросе за экономией места. Например, поле QUAL_SCHEMA_URL указывает на полное (реальное), а не сокращенное имя ресурса, созданного в результате регистрации схемы: в нашем случае */sys/schemas/SCOTT/localhost:8080/public/bookCover.xsd*. Если бы регистрируя схему мы объявили ее глобальной (что требует явного указания и наличия прав), полный адрес ресурса был бы иной: */sys/schemas/PUBLIC/localhost:8080/public/bookCover.xsd*.

Выдача команды ALTER SESSION выше привела к фиксации в трассировочном файле сеанса некоторых скрытых действий СУБД, например:

```
CREATE OR REPLACE TYPE "SCOTT"."author342_COLL" AS VARRAY(2147483647) OF VARCHAR2(4000
CHAR)
/
CREATE OR REPLACE TYPE "SCOTT"."coverType341_T" AS OBJECT ("SYS_XDBPD$" "XDB"."XDB
$RAW_LIST_T", "title" VARCHAR2(4000 CHAR), "author" "author342_COLL", "publisher"
VARCHAR2(4000 CHAR), "pubdate" VARCHAR2(4000 CHAR), "isbn" VARCHAR2(4000 CHAR), "pages"
VARCHAR2(4000 CHAR)) NOT FINAL INSTANTIABLE
/
CREATE TABLE "SCOTT"."cover343_TAB" OF SYS.XMLTYPE XMLSCHEMA "http://localhost:
8080/public/bookCover.xsd" ID '6927F846BBDA487F84B9BCCEC191A059' ELEMENT "cover" ID
2664 TYPE "SCOTT"."coverType341_T"
/
```

Результат этих действий, а также созданную заодно триггерную процедуру "cover343_TAB\$xd" (в нашем случае), можно

наблюдать в таблице USER_OBJECTS. (*Упражнение.* Узнать из таблицы USER_OBJECTS перечень появившихся после регистрации схемы XML объектов). Обратите внимание, что сведения о таблице "cover343_TAB" не отражены в USER_TABLES, хотя представлены в прочих системных таблицах, например, в USER_TAB_COLUMNS и USER_TAB_COLS.

Эти сопутствующие объекты будут технически использоваться для организации хранения данных XMLTYPE, уточненных зарегистрированной схемой. Для возможно повторяющегося элемента author хранение организовано в виде массива VARRAY.

Если мы захотим снять схему с регистрации, да еще удалить эти сопутствующие объекты, нужно при вызове соответствующей процедуры указать особое значение специальному параметру:

```
BEGIN
DBMS_XMLSCHEMA.DELETESHEMA (
  schemaurl      => 'http://localhost:8080/public/bookCover.xsd'
, delete_option => DBMS_XMLSCHEMA.DELETE_CASCADE
);
END;
/
```

Для краткости (но в ущерб ясности кода) можно сразу указать цифровое значение, например в SQL*Plus:

```
EXECUTE DBMS_XMLSCHEMA -
        .DELETESHEMA ( 'http://localhost:8080/public/bookCover.xsd', 3 )
```

Использование зарегистрированной схемы

Проверка действия схемы

Метод CREATESCHEMABASEDXML типа XMLTYPE позволяет устроить проверку соответствия документа XML схеме.

Ради краткости записи следующих примеров имя зарегистрированной схемы занесем в переменную SQL*Plus:

```
VARIABLE bookschema VARCHAR2 ( 100 )
EXECUTE :bookschema := 'http://localhost:8080/public/bookCover.xsd'
```

Следующие запросы проработают *без* ошибок:

```
SELECT
XMLTYPE ( '<cover></cover>' ) .CREATESCHEMABASEDXML ( :bookschema )
FROM dual
;
```

```
SELECT
XMLTYPE ( '<cover><author>Einstein</author></cover>' )
.CREATESCHEMABASEDXML ( :bookschema )
FROM dual
;
```

```
SELECT
XMLTYPE (
'
<cover>
  <title>Java Programming with Oracle JDBC</title>
  <author>Donald Bales</author>
  <publisher>OReilly and Associates</publisher>
  <pubdate>December 2001</pubdate>
  <isbn>0-596-00088-x</isbn>
  <pages>496</pages>
</cover>
'
) .CREATESCHEMABASEDXML ( :bookschema )
FROM dual
```

```
;
```

Следующие запросы проработают с *ошибками* ввиду противоречий документа схеме:

```
SELECT
XMLTYPE ( '<c/>' ).CREATESCHEMABASEDXML ( :bookschema )
FROM dual
;
```

```
SELECT
XMLTYPE ( '<cover><a>Einstein</a></cover>' )
.CREATESCHEMABASEDXML ( :bookschema )
FROM dual
;
```

```
SELECT
XMLTYPE ( '<cover><title>A</title><title>B</title></cover>' )
.CREATESCHEMABASEDXML ( :bookschema )
FROM dual
;
```

Использование для дополнительной типизации XMLTYPE в базовых таблицах

Просто указанный XMLTYPE требует лишь, чтобы помещаемый в БД документ был правильно оформлен. Зарегистрированная в XML DB схема XML позволяет указать для размещаемых в конкретные поля документов XML-образные ограничения целостности, позволяя хранить только «годные» (valid) из них.

Создадим таблицу объектов XMLTYPE, уточненную ссылкой на зарегистрированную схему:

```
CREATE TABLE xtbooks OF XMLTYPE
XMLSCHEMA "http://localhost:8080/public/bookCover.xsd"
ELEMENT "cover"
;
```

Проверка занесения данных:

```
INSERT INTO xtbooks VALUES (
XMLTYPE ( '<cover><author>Einstein</author></cover>' )
);
```

```
INSERT INTO xtbooks VALUES (
XMLTYPE ( '
<cover>
<title>Java Programming with Oracle JDBC</title>
<author>Donald Bales</author>
<publisher>OReilly and Associates</publisher>
<pubdate>December 2001</pubdate>
<isbn>0-596-00088-x</isbn>
<pages>496</pages>
</cover>
' )
);
```

```
COMMIT;
```

Упражнение. Проверьте реакцию СУБД на попытку вставить в таблицу XTBOOKS правильно оформленный документ XML, не соответствующий описанию схемы.

Аналогично накладываются дополнительные ограничения на данные столбца XMLTYPE в обычной таблице:

```
CREATE TABLE tbooks (
id NUMBER ( 9 )
, description XMLTYPE
```

```

)
XMLTYPE description STORE AS OBJECT RELATIONAL
XMLSCHEMA "http://localhost:8080/public/bookCover.xsd"
ELEMENT "cover"
;

```

В приведенных примерах данные в БД фактически будут храниться «объектно-реляционно», то есть будучи распределены по разным структурам БД, в том числе по спрятанным столбцам таблиц. Однако типизируя таблицу или столбец типа XMLTYPE схемой, можно потребовать фактического хранения документа в виде объекта CLOB, подобно тому, как это происходит при отсутствии типизации схемой.

Пример:

```

CREATE TABLE xtbooksclob OF XMLTYPE
XMLTYPE STORE AS CLOB
XMLSCHEMA "http://localhost:8080/public/bookCover.xsd"
ELEMENT "cover"
;

```

Упражнение. Проверить возникновение новых объектов БД и их структуры по результатам заведения таблиц XTBOOKS, TBOOKS и XTBOOKSCLOB. Воспользоваться для этого таблицами USER_OBJECTS, USER_TAB_COLS, USER_TYPES, USER_TRIGGERS, USER_LOBS.

Использование для дополнительной типизации XMLTYPE в виртуальных таблицах (views)

При отсутствии XML DB не запрещено создавать виртуальную таблицу (view), выдающую данные в виде таблицы документов XML (типа XMLTYPE) на основе данных из обычных таблиц. Рассмотрим, как при наличии XML DB можно дополнительно уточнить такую виртуальную таблицу схемой XML.

В этом примере регистрируется схема, не взятая из ресурса репозитория, как ранее, а явно выписанная в виде текста XML:

```

BEGIN
DBMS_XMLSCHEMA.REGISTERSCHEMA (
  schemaurl => 'http://localhost:8080/public/employee.xsd'
, schemadoc =>
  '<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="employee">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="ename" type="xs:string"/>
          <xs:element name="job" type="xs:string"/>
          <xs:element name="sal" type="xs:integer"/>
        </xs:sequence>
        <xs:attribute name="empno" type="xs:integer"/>
      </xs:complexType>
    </xs:element>
  </xs:schema>
  '
);
END;
/

```

Пример создания виртуальной таблицы:

```

CREATE OR REPLACE VIEW empxml_schema_view OF XMLTYPE
XMLSCHEMA "http://localhost:8080/public/employee.xsd"
ELEMENT "employee"
WITH OBJECT ID
( EXTRACT ( SYS_NC_ROWINFO$, '/employee/@empno' ).GETNUMBERVAL ( ) )
AS
SELECT
  XMLELEMENT (

```

```

    "employee"
  , XMLATTRIBUTES ( e.empno AS "empno" )
  , XMLFOREST ( e.ename AS "ename", e.job AS "job", e.sal AS "sal" )
)
FROM emp e
;

```

Наличие данных, противоречащих схеме, не препятствует извлечению данных результата при обращении к созданной виртуальной таблице. Однако ж имеется метод, способный выполнить проверку соответствия схеме:

```

SELECT VALUE ( v ) .ISSHEMAVALID ( ) FROM empxml_schema_view v;

```

Упражнение. Создать виртуальную таблицу EMLXML_VIEW по аналогии с EMLXML_SCHEMA_VIEW, не уточненной схемой (для этого потребуется удалить из определения выше две строки, выделенные жирным шрифтом). Выполнить следующие проверки:

```

SELECT VALUE ( v ) .ISSHEMAVALID ( ) FROM empxml_view v;

```

```

SELECT
  VALUE ( v )
  .ISSHEMAVALID ( 'http://localhost:8080/public/employee.xsd' )
FROM empxml_view v
;

```

Упражнение. Внести в схему <http://localhost:8080/public/employee.xsd> пользователя SCOTT изменение (снять с регистрации и зарегистрировать исправленный текст). Заменить в схеме существующее описание зарплаты:

```

<xs:element name="sal" type="xs:integer"/>

```

на следующее:

```

<xs:element name="sal">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:maxInclusive value="1500"/>
      <xs:minInclusive value="1000"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

```

Проверить поведение последних трех приведенных выше запросов.

Последний пример показывает использование чуть более сложных ограничений целостности на структуру и данные документов XML (сформулированы минимальное и максимальное значения зарплат).