Automatic Conversion of XSD to RDBMS

Author

Colin James III, Principal Scientist, CEC Services, LLC, 1613 Morning Dr, Loveland CO 80538-4410, xsd-rdbms@cec-services.com, (719) 210 - 9534

Abstract

XSD (XML Schema Definition) describes document structures and data types. Draft EIA-836 standard for configuration management is published in XSD format. By contrast, the ancestor standard Mil-STD-2549, now cancelled, presented hundreds of database tables as requirements. Tools exist to convert the description of relational databases (RDBMS) into XSD code. However the reverse, to convert XSD into RDBMS descriptions from scratch, is difficult. The problem is to convert Draft EIA-836 in XSD into RDBMS scripts in structured query language (SQL). This paper describes a successful conversion in the development of the vendor tool XSD-RDBMS. Assumptions are made and presented. XSD-RDBMS produces table and index scripts that process in real time on IBM DB2 RDBMS.

Keywords

configuration management, Draft EIA-836, GEIA, Government Electronics and Information Technology Association, IBM DB2, layered logic tables, LLT, logic table technology, LTT, Mil-STD-2549, RDBMS, relational database management system, scripts, structured query language, SQL, TrueBASIC®, XML Schema Definition, XSD

Introduction

Draft EIA-836 is a standard for configuration management as published by the Government Electronics and Information Technology Association (GEIA) which is a consortium of public and private sector members. Draft EIA-836 is presented in XML Schema Definition (XSD) that describes document structures and data types. The previous standard Mil-STD-2549, now cancelled, was presented as hundreds of required database tables and relationships. Many vendors advertise to convert database schema, such as Mil-STD-2549, into XSD. However, to effect the reverse of converting XSD into database schema is problematic. Draft EIA-836 is presented in XSD format with content significantly different than that of its predecessor. The problem is to convert Draft EIA-836 in XSD into RDBMS scripts in structured query language (SQL). The difficulty is that elements of XSD do not necessarily map directly into the components of RDBMS.

Analysis

As used in the Draft EIA-836, elements of XSD are name, reference, type, attribute, and enumeration. A sample of XSD in graphical and code format is respectively in Figure 1 and 2 below.

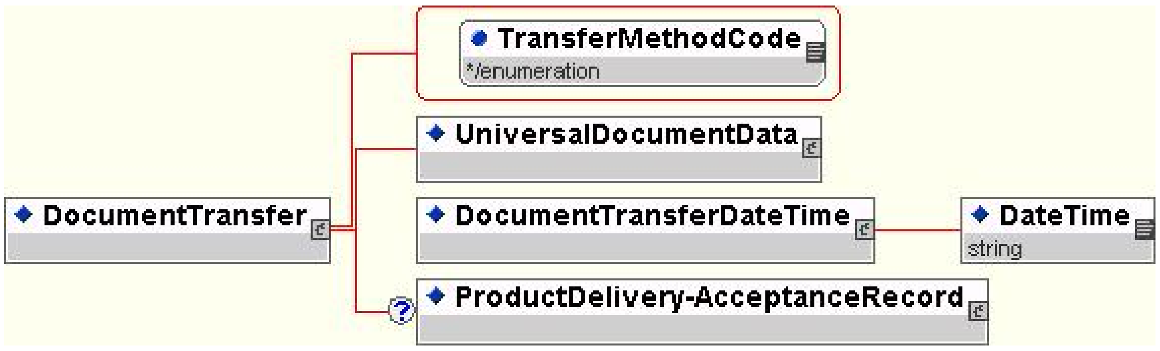

Figure 1

```
   <xsd:element name="DocumentTransferDateTime">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="DateTime"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="DocumentTransfer">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="BusinessObjectHeader"/>
            <xsd:element ref="UniversalDocumentData"/>
            <xsd:element ref="DocumentTransferDateTime"/>
            <xsd:element ref="ProductDelivery-AcceptanceRecord"
                minOccurs="0" maxOccurs="1"/>
              [ minOccurs="1" maxOccurs="1"/> ]
         </xsd:sequence>
         <xsd:attribute name="TransferMethodCode" use="required"

type="DocumentTransferTransferMethodCodetype"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:simpleType name="DocumentTransferTransferMethodCodetype">
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="Access"/>
         <xsd:enumeration value="Attachment"/>
      </xsd:restriction>
   </xsd:simpleType>
```

Figure 2

In Figure 2, the constraint in brackets [minOccurs="1" …] means at least one instance of the attribute is required. That constraint is not present in the graphical diagram, but inserted for discussion below.

From Figures 1 and 2, the following descriptive details are apparent:

1. Document Transfer is of attribute type Transfer Method Code with enumerated values of "Access" and "Attachment".
2. Document Transfer contains mandatory references to Universal Document Data and Document Transfer Date Time and optional reference to Product Delivery – Acceptance Record.
3. Document Transfer Date Time contains mandatory reference to Date Time.
4. Date Time is a string of Date Time type.

The components of RDBMS are names of tables and columns, constraints such as NOT NULL indexes, relationships between columns of the same type in different tables, views of combined tables and columns, and enumerated values.

The descriptions from Figures 1 and 2 are effectively translated into the rules below:

5. If an element has an attribute then that element is a column.
6. If an element has only one child then that element cannot be a table and thus is a column.
7. If an element has an attribute, then that attribute becomes a column.
8. An element with no child and only literal attributes is enumerated.
9. If an element name has a null name, a child reference name with attribute enumerated, and a child reference name with no attribute enumerated, then the child reference name with no attribute

enumerated is promoted to an element name, and the child reference name with attribute enumerated replaces the promoted child reference name.

10. In table T1, if a column named C1 is also the same name as table T2, then the table named T2_T1 is view for table name T1 and table name T2.

Design

To design the rules in 5-10 above, scripts in structured query language (SQL) perform the following tasks.

11. Tables are made by defining table names, column names, and data types.
12. Indexes are made by altering table columns as primary keys.
13. Views are made by relating columns in tables to each other.
14. Enumerated data is made by inserting data into table columns on a row-by-row basis.

The assumptions are that the data input in XSD format is consistent with the emerging standards and logically sound according to established practices and that:

15. Element names map directly to table names.
16. Element references map directly to column names.
17. Attributes map directly to column data types.
18. Simple sub-types map directly to column enumerations.

The design for mapping XSD into scripts for RDBMS is decomposed into the two main parts of parsing input and writing outputs.

19. Parsing input XSD

19.1. Parse input
19.1.1 Get element name
19.1.2 Get element reference name
19.1.3 Get element reference if attribute
19.1.4 Get enumeration values for the attribute
19.1.5 Get remaining attributes

19.2. Convert singleton element names with no attributes to element references with no attributes

19.3. Check label syntax for compliance with scripting conventions such as those of IBM DB2 where:
19.3.1 In names the characters "-", ".", and ":" are converted to character "_".
19.3.2 The length of column names, in particular, must be limited to 30 characters.

19.4. Check orphans for promotion. [See description in item 9 above.]

20. Writing output scripts for RDBMS

20.1. Make table script: Those element references not connected to a table are collected into an orphan table.

20.2. Make primary index script: The basis for a primary key is that a column is NOT NULL and therefore mandatory.

20.3. Make view script: This finds repeated instances of the same name for a table and column which connects the two by making a combined-name view of the respective tables.

20.4. Make orphan scripts: The orphans are counted and merged into orphan tables named after the originating table before making the script.

20.5. Make insert script for enumerations: This collates the values into separate insert commands.

20.6. The output script is sorted for all XSD files within Draft EIA-836 with duplicate statements removed and in the script order for CREATE TABLE, ALTER TABLE, CREATE VIEW, and INSERT INTO.

Implementation

The design is primarily concerned with the manipulation of input text so as to produce a transformation in the output text. The programming language chosen was ANSI BASIC for ease in managing text. The most robust version is TrueBASIC® authored by John Kemeny (deceased) and Thomas Kurtz, the two inventors of the language at Dartmouth College in about 1963, and implemented by Christopher Sweeney. The current version avoids the dynamic link library (DLL) with direct access to the programming interface of the host operating system. This is achieved in small engines by free download for Windows, Unix, and mainframe on which the tokenized pseudo code runs identically. It is noteworthy that TrueBASIC® has achieved the platform portability which the Java community promised but could not deliver.

TrueBASIC® also allows for extensive exception processing and recovery, dynamic memory allocation of arrays, and a wealth of built in functions and utilities. With dynamic memory management, arrays can be copied and resized in simple commands. String handling includes functions to convert the case of text. Extensive visual and graphics capabilities were available but not exploited because of the utility nature of the problem. The implemented product was named XSD-RDBMS and executes very fast in real time. The available demonstration version limits input file size to 1500 characters.

Solution

Output from the implemented product named XSD-RDBMS is in Figure 3 below.

```
CREATE TABLE DocumentTransferDateTime ( DateTime TIMESTAMP) ;

CREATE TABLE DocumentTransfer ( BusinessObjectHeader VARCHAR( 32),
UniversalDocumentData VARCHAR( 32), DocumentTransferDateTime VARCHAR( 32),
ProductDelivery_AcceptanceReco VARCHAR( 32) [NOT NULL], TransferMethodCode
VARCHAR( 32)) ;

        [ ALTER TABLE DocumentTransfer ADD PRIMARY KEY (
        ProductDelivery_AcceptanceReco) ; ]

CREATE VIEW DocumentTransfer__DocumentTransferDateTime AS SELECT * FROM
DocumentTransferDateTime, DocumentTransfer ;

INSERT INTO DocumentTransfer ( TransferMethodCode ) VALUES ( 'Access' ) ;
INSERT INTO DocumentTransfer ( TransferMethodCode ) VALUES ( 'Attachment' ) ;
```

Figure 3

From Figure 2, the constraint in brackets [minOccurs="1" …] is also reflected in Figure 3 in brackets as follows. The CREATE TABLE command represents the column of interest also as NOT NULL. The ALTER TABLE command makes the column of interest into a PRIMARY KEY.

Draft EIA-836 consisted of 106 DTD files which were manually converted into XSD files by loading and writing each into an XML-to-XSD editor. The files were serially translated into scripts by XSD-RDBMS. The scripts without documentation as comments were copied into one file of size 0.2 MB. The Uml.xsd script was excluded for reasons in the section below. The number of sorted scripts was: 605 "CREATE TABLE", 53 "ALTER TABLE", 216 "CREATE VIEW", and 287 "INSERT INTO". The scripts ignored were one for "CREATE TABLE" as nearly a duplicate, and 20 for "INSERT INTO" as attempting to insert on a column without insert on the respective key column. The 1090 scripts were manually inserted as a block into and processed by IBM DB2 Command Center without error.

The performance statistics were as follows. The conversion of XSD code to RDBMS scripts processed in 3.5 minutes. IBM DB2 parsed the input scripts in 0.6 minutes and executed the scripts in 2.0 minutes for a total of 2.6 minutes. The particular test environment was a very inexpensive 700 MHz Hz VIA CPU with 0.75 GB RAM running Microsoft Windows 2000 Pro SP4.

Where XSD does not convert to RDBMS

XSD does not convert to RDBMS where faulty logic is allowed by XSD. Such an example is in the generalized case of the XSD code and resultant RDMBS script in Figure 4.

```
<xsd:element name="table_01">
      <xsd:complexType>
            <xsd:sequence>
                  <xsd:element ref="column_01"
                      minOccurs="1" maxOccurs="unbounded"/>
                  <xsd:element ref="column_02"/>
            </xsd:sequence>
            <xsd:attribute name="column_02" use="optional"
                type="column_02_type"/>
      </xsd:complexType>
</xsd:element>
      <xsd:simpleType name="column_02_type">
            <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="test_value"/>
            </xsd:restriction>
      </xsd:simpleType>
```

```
CREATE TABLE table_01 (column_01 VARCHAR( 32) NOT NULL, column_02
VARCHAR( 32) ;

[ ALTER TABLE table_01 ADD PRIMARY KEY ( column_01) ; ]

INSERT INTO table_01 ( column_02) VALUES ( ‘test_value’) ;
```

Figure 4

From the XSD code, column_01 has a minimum of one instance. The RDBMS script translates this correctly as NOT NULL and as a resultant ALTER TABLE command which is in brackets as optional to the example. From the XSD code, column_02 has an enumerated value of “test_value”. The RDBMS script translates this correctly into an INSERT INTO statement on column_02. However, the script will not execute in IBM DB2 because there is no value already present in the NOT NULL or primary key column_01 which disallows insert into any subsequent nullable and non-key column such as column_02.

In Draft EIA-836 there were six such exceptions for the following element names: AsMaintained-ModifiedStatusReport; ConfigAuditAction; DocumentRevisionNotice; Hardware; ModificationInstruction; and ReleaseRecord.

XSD is sometimes difficult to convert into RDBMS script because of arbitrary conventions adopted by the XML community. For example use of Hungarian notation presents a name such as “transfer method code” as run together without spaces and using capitalization as TransferMethodCode. A lesson learned is that Hungarian notation is almost as unreadable to humans as is XSD code. A better convention by test is to keep all of the labels in lower case for the reasons that lower case is easier to read and upper case affects the sorting order and increases possibility of typo errors in mixed case systems. In addition, the underscore character works well as a space or capital letter substitute such as in transfer_method_code. The disadvantage of all lower case with underscores is that the length of the name is greater by the number of spaces replaced than the equivalent Hungarian notation that is lesser by the number of spaces excluded.

The length of the names becomes irrelevant in wordy systems such as XSD. However, one of the annoyances with Draft EIA-836 is that name references are greater than 30 characters which IBM DB2 does not allow. (ORACLE in addition does not allow table names greater than 30 characters.) The instant solution was simply to truncate the names. A consequence of this was the fact that the uml.xsd file, comprising more about 60% of the total XSD code of Draft EIA-836 at 0.6 MB, when translated into RDBMS script contained multiple instances of tables containing different column names but which truncated to the same column name, were thus not differentiated, and would not process in IBM DB2.

Conclusion

As sometimes is the case in project development, the importance of the intermediate steps or tools may eclipse the required purpose of the project. XSD-RDBMS as a general-purpose tool has implications beyond the purpose of a database implementing Draft EIA-836. Presenting documents in XSD format is one of the few things to come out of computer science from 2000 to 2004. Therefore the adoption of XSD in about 2000 as a trendy delivery system for Draft EIA-836 was not a well-advised strategic business decision, and consideration should be made to reverse it and present the standard in some other format.

The delivery of an RDBMS implementation of Draft EIA-836 marks recent advances in software development. XSD-RDBMS came about as the first useful translation tool to move XSD code into RDBMS. As an extension of Draft EIA-836, the database should reflect the current requirements of government and industry in an integrated configuration management database platform.

To implement the configuration management database for portable, maintainable, and scalable use in real time requires the assistance of reentrant layered logic tables (LLT) in logic table technology (LTT) as established in the literature since 1997.

Acknowledgments

Thanks are due for helpful comments to: Linda King Baroni, ManTech Advanced Systems International, Inc., Fairmont WV; and Larry Cagg, CEC Services, LLC, Westminster CO.

References

James, C. 2003.8, “Encryption Engine Implemented in Reentrant Layered Logic Tables”, unpublished, CEC Services, LLC, Loveland CO.

James, C. 2003.7, “MySQL and PostgreSQL: Non Competes in RDBMS”, unpublished, CEC Services, LLC, Loveland CO.

James, C. 2003.6, “Non Invertive Encryption in Reentrant Layered Logic Tables (RLLT), unpublished, CEC Services, LLC, Loveland CO.

James, C. 2003.5, “Implementing Performance in Reentrant Layered Logic Tables (RLLT)”, unpublished, CEC Services, LLC, Loveland CO.

James, C. 2003.4, “Reentrant Layered Logic Tables (RLLT)”, [in submission to the Industrial Water Conference 2003 (IWC2003)], CEC Services, LLC, Loveland CO.

James, C. 2003.3, “Layered Logic Tables (LLT)”, unpublished, CEC Services, LLC, Loveland CO.

James, C. 2003.2, "Software Factory", unpublished brochure, CEC Services, LLC, Loveland CO.

James, C. 2003.1, "Software Factory", unpublished poster, CEC Services, LLC, Loveland CO.

James, C. 2002.5, "The Software Development Methodology [SDM]", unpublished, CEC Services, LLC, Loveland CO.

James, C. 2002.4, "Reentrant Logic Table Technology", unpublished, CEC Services, LLC, Loveland CO.

James, C. 2002.3, "Static and Dynamic Driver Triggers", unpublished, CEC Services, LLC, Loveland CO.

James, C. 2002.2, "Additional Information", unpublished, CEC Services, LLC, Loveland CO.

James, C. 2002.1, “Implementation Details for Multiple Billing”, CEC Services, LLC, Loveland CO.

James, C. 2001.2, "Report Accounts [RA] v 1.2 Inventory / Point of Sale", unpublished, CEC Services, LLC, Loveland CO.

James, C., 2001.1, "Report Accounts [RA] v 1.2", unpublished, CEC Services, LLC, Loveland CO.

James, C. 1999.1, “Recent Advances in Logic Tables for Reusable Database Engines”, Proceedings of the American Society of Mechanical Engineers International, Petroleum Division, 75th Anniversary Conference, Energy Sources Technology Conference & Exhibition, Houston, Texas.

James, C., 1998.5, "Multiple and Self-Modifying Logic Tables with Queries", unpublished, CEC Services, LLC, Loveland CO.

James, C. 1998.4, “A Reusable Database Engine for Accounting Arithmetic”, Proceedings of The Third Biennial World Conference on Integrated Design & Process Technology, Vol. 2, pp. 25-30, Berlin, Germany.

James, C., 1998.3, "Competency test for CEC Services, LLC", unpublished, CEC Services, LLC, Loveland CO.

James, C. 1998.2, “Theory and Application of Logic Tables in Relational Database Engines”, Doctoral Dissertation, Pacific Western University, Los Angeles.

James, C., 1998.1, "Ticket Reservations [TR] ver 1.1", unpublished, CEC Services, LLC, Loveland CO.

James, C., 1997.2, "User Documentation", unpublished, CEC Services, LLC, Loveland CO.

James, C., 1997.1, "Logic Table Design for Reports in RA", unpublished, CEC Services, LLC, Loveland CO