XSD2DDL Requirements Specification

1. Scope

1.1 Overview

The XSD2DDL component will create Data Definition Language (DDL) statements from an XSD document. The component will initially create DDL that is valid for an Oracle 10g database. It will provide interfaces to allow support for other databases to be added easily. The goal of the component is not to produce DDL from any possible XSD, but rather to define a subset of XSD that can be used to produce DDL for a variety of RDBMSes and to be able to easily add features in the future.

1.2 Logic Requirements

1.2.1 Simple Types

The XSD 'simpleType' element will be used to define column types. For the Oracle implementation the base type will be used to determine the column type and any restriction will be used to determine column size or produce a check constraint on that column. Other RDBMS products will handle this differently.

1.2.2 Primary Keys

Primary keys will be identified in the XSD using the 'key' element. The component will produce appropriate DDL to enforce the primary key.

1.2.3 Foreign Keys

Foreign keys will be identified in the XSD using the keyref element. The component will produce appropriate DDL to enforce the foreign key relationship.

1.2.4 Constraints

All XSD restrictions will be enforced in the database either through structural elements (column sizes, cardinality, etc.) or through constraint definition.

1.2.5 Nullable Columns

The 'nillable' field will be used to indicate whether a given column in the database will be nullable.

1.2.6 Example XSD

The XSD below is an example only. The designer may deviate as needed, but the final result must be an XSD-based syntax.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://topcoder.university.com"
  xmlns:univ="http://topcoder.university.com">

      <xsd:simpleType name="ssn">
            <xsd:restriction base="xsd:string">
                  <xsd:pattern value="\d\d\d-\d\d-\d\d\d\d"/>
                  <xsd:maxLength value="11"/>
            </xsd:restriction>
      </xsd:simpleType>

      <xsd:simpleType name="classNameType">
            <xsd:restriction base="xsd:string">
                  <xsd:maxLength value="128"/>
            </xsd:restriction>
      </xsd:simpleType>
```

```
      <xsd:simpleType name="studentNameType">
            <xsd:restriction base="xsd:string">
                  <xsd:maxLength value="256"/>
            </xsd:restriction>
      </xsd:simpleType>

      <xsd:complexType name="StudentType">
            <xsd:sequence>
                  <xsd:element name="StudentName"
type="univ:studentNameType" nillable="false" minOccurs="1"
maxOccurs="1" />
                  <xsd:element name="SocialSecurityNumber"
type="univ:ssn" minOccurs="1" maxOccurs="1" />
            </xsd:sequence>
      </xsd:complexType>

            <xsd:complexType name="RootType">
        <xsd:sequence>
            <xsd:element name="Student" type="univ:StudentType">
                  <xsd:key name="StudentPK">
                        <xsd:selector xpath="Student"/>
                        <xsd:field xpath="SocialSecurityNumber"/>
                  </xsd:key>
            </xsd:element>

            <xsd:element name="Course">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="CourseName"
type="univ:classNameType" minOccurs="1" maxOccurs="1"/>
                        </xsd:sequence>
                        <xsd:attribute name="id" type="xsd:integer"/>
                  </xsd:complexType>
                  <xsd:key name="CoursePK">
                        <xsd:selector xpath="Course"/>
                        <xsd:field xpath="@id"/>
                  </xsd:key>
            </xsd:element>

            <xsd:element name="Enrollment">
                  <xsd:complexType>
                        <xsd:sequence>
                           <xsd:element name="EnrollmentID"
type="xsd:integer"/>
                              <xsd:element name="CourseID"
type="xsd:positiveInteger"/>
                              <xsd:element name="SocialSecurityNumber"
type="univ:ssn"/>
                        </xsd:sequence>
                  </xsd:complexType>
                  <xsd:keyref name="ClassFK" refer="univ:CoursePK">
                        <xsd:selector xpath="Enrollment"/>
                        <xsd:field xpath="ClassID"/>
                  </xsd:keyref>
                  <xsd:keyref name="StudentFK" refer="univ:StudentPK">
                        <xsd:selector xpath="Enrollment"/>
```

```
                        <xsd:field xpath="SocialSecurityNumber"/>
                    </xsd:keyref>
                    <xsd:key name="EnrollmentPK">
                        <xsd:selector xpath="Enrollment"/>
                        <xsd:field xpath="EnrollmentID"/>
                    </xsd:key>
                </xsd:element>
                </xsd:sequence>
                </xsd:complexType>

                <xsd:element name="University" type="univ:RootType"
nillable="false"/>
</xsd:schema>
```

1.2.7 Example DDL

The following DDL (or it's equivalent) will be produced from the XML in section 1.2.6

```
CREATE TABLE student
( studentname VARCHAR2(256) NOT NULL,
  socialsecuritynumber VARCHAR2(11) PRIMARY KEY );

ALTER TABLE student
  ADD CONSTRAINT student_ssn_check
CHECK( REGEXP_LIKE(ssn,'\d\d\d-\d\d-\d\d\d\d') );

CREATE TABLE course
( coursename VARCHAR2(128),
  id integer PRIMARY KEY );

CREATE TABLE enrollment
( enrollmentid integer PRIMARY KEY,
  courseid integer NOT NULL
  socialsecuritynumber VARCHAR2(11) NOT NULL,

  FOREIGN KEY(socialsecuritynumber)
    REFERENCES student(socialsecuritynumber),

  FOREIGN KEY(courseid)
    REFERENCES course(id)
);
```

1.3 Required Algorithms

None defined.

1.4 Example of the Software Usage

This component could be used to produce XML schemas for both data storage and data transmission.

1.5 Future Component Direction

Future versions will include support for other RDBMS as well as support for features such as table comments. The goal of version 1.0 is to build a solid framework that can be easily extended.

2. Interface Requirements

2.1.1 *Graphical User Interface Requirements*

None

2.1.2 *External Interfaces*

None

2.1.3 *Environment Requirements*

- Development language: C# 1.1
- Compile target: C# 1.1

2.1.4 *Package Structure*

`TopCoder.DB.Generation.Xsd2Ddl`

3. Software Requirements

3.1 Administration Requirements

3.1.1 *What elements of the application need to be configurable?*

None Defined.

3.2 Technical Constraints

3.2.1 *Are there particular frameworks or standards that are required?*

XSD – XML Schema Definition

3.2.2 *TopCoder Software Component Dependencies:*

**Please review the TopCoder Software component catalog for existing components that can be used in the design.

3.2.3 *Third Party Component, Library, or Product Dependencies:*

None

3.2.4 *QA Environment:*

- Windows 2000
- Windows 2003

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines. Modifications to these guidelines for this component should be detailed below.

3.4 Required Documentation

3.4.1 *Design Documentation*

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification
- XML Schema for supported XSD elements

3.4.2 *Help / User Documentation*

- Design documents must clearly define intended component usage in the ‘Documentation’ tab of Poseidon.