Proceeding, Seminar Ilmiah Nasional Komputer dan Sistem Intelijen (KOMMIT 2008)
Auditorium Universitas Gunadarma, Depok, 20-21 Agustus 2008 ISSN : 1411-6286

EVALUASI XML EDITOR SEBAGAI XML VALIDATOR

Oviliani Yenty Yuliana

Jurusan Teknik Informatika Universitas Kristen Petra
Jl. Siwalan Kerto 121-131 Surabaya 60236 (ovi@peter.petra.ac.id)

ABSTRAK

Penelitian bertujuan untuk mengevaluasi XML Editor sebagai validator XML Document berdasarkan XML Schema. XML Editor populer yang dievaluasi adalah XMLwriter, oXygen, ALTOVA dan Visual Studio. XML Editor dievaluasi atas ketepatan dan kelengkapan hasil validasi terhadap property: null status, data type, dan data constraint. Untuk keperluan evaluasi XML Editor maka dibuat kasus berupa XML Schema dan XML Document untuk masing-masing property tersebut. Seluruh XML Editor dapat memvalidasi kesalahan XML Document berdasarkan XML Schema. XML Editor oXygen dan Visual Studio unggul dalam membedakan huruf besar dan huruf kecil pada pattern string. Hanya ALTOVA dan Visual Studio yang dapat mendeteksi ketidakvalidan dari element yang berperan sebagai referential integrity antara complexType dengan tipe sama tetapi berbeda size.

***Kata Kunci**: Property, XML Schema, XML Document, XML Editor, XML Validator*

1. PENDAHULUAN

Saat ini XML merupakan standar untuk pertukaran, penyimpanan dan pengaksesan data untuk aplikasi yang berbeda sistem operasi atau berbeda sistem basis data. Hal tersebut dimungkinkan karena XML berupa file teks. XML dibedakan menjadi XML *Document* dan XML *Schema*. XML *Document* digunakan untuk penyimpanan data, sedangkan XML *Schema* berperan sebagai struktur dari XML *Document*.

Pemanfaatan XML semakin meningkat pesat. Akibat semakin banyak XML *Editor* yang ditawarkan secara komersial. XML *Editor* yang populer adalah XMLwriter, oXygen, ALTOVA, dan Visual Studio. Pada umumnya XML *Editor* tersebut memanfaatkan XML *Schema* untuk mengisi data ke dalam XML *Document*. Selain itu, XML *Schema* digunakan untuk memvalidasi XML *Document*. Penelitian sebelumnya pernah dilakukan oleh Yuliana (2006) mengenai evaluasi XML *Designer* dengan pendekatan model data *Entity Relationship Diagram* untuk pembuatan XML *Schema*. Sejauh ini penelitian yang terkait dengan evaluasi terhadap XML *Editor* sebagai XML *validator* belum dilakukan.

Tujuan dari penelitian adalah mengevaluasi fasilitas XML *Editor* sebagai *validator* XML *Document* berdasarkan XML *Schema*. Evaluasi tersebut perlu dilakukan mengingat file XML berupa file teks, sehingga isi file tersebut dengan mudah diubah, dihapus, atau ditambah menggunakan *text editor*, seperti *notepad*. Hal tersebut dapat mengakibatkan data dalam XML *Document* tidak konsisten terhadap XML *Schema* (*well validated*). Manfaat dari penelitian ini adalah untuk merekomendasi atau sebagai bahan pertimbangan dalam pemilihan XML *Editor*.

2. TINJAUAN PUSTAKA

Property Basis Data

Kroenke (2006:171-212) menyatakan ada empat *property* yang perlu ditetapkan pada setiap kolom dalam suatu tabel. Keempat *property* tersebut adalah *null status*, *data type*, *default value*, dan *data constraint*. *Property default value* berupa suatu nilai yang diberikan oleh DBMS saat suatu baris baru disisipkan dalam suatu tabel.

Suatu kolom dapat berstatus *NULL* atau *NOT NULL*. *Primary key* selalu *NOT NULL* dan unik. *Data type* tergantung pada DBMS yang digunakan. *Data type* yang umum adalah *CHAR*(n), *VARCHAR*(n), *DATE*, *TIME*, *MONEY*, *INTEGER*, dan *DECIMAL*. Dimana n pada *CHAR* dan *VARCHAR* digunakan untuk menyatakan jumlah karakter (*length*) pada tipe data tersebut.

Data constraint meliputi *domain constraint*, *range constraint*, *intrarelation constraint*, dan *interrelation constraint*. *Domain constraint* menetapkan sekumpulan nilai-nilai yang dimungkinkan untuk suatu kolom. *Range constraint* menetapkan suatu interval nilai yang diijinkan untuk suatu kolom. *Intrarelation constraint* membandingkan antar kolom dalam tabel yang sama. Sedangkan *interrelation constraint* membandingkan antar kolom dari tabel yang berbeda. *Referential integrity constraint* adalah salah satu contoh dari *interrelation constraint*. Grauer (2006:224) menyatakan *field* yang berperan sebagai *referential integrity* harus memiliki *data type* dan *size*/*length* yang sama. Sedangkan nama *field* boleh berbeda.

Property XML

W3C menetapkan *property* untuk XML. *Null* dalam XML *Schema* dapat dinyatakan dengan *attribute nillable="true"* atau *minOccurs*="0". Sedangkan *element* yang tidak menyertakan *attribute* tersebut dinyatakan sebagai *not null*. Dalam *data type* XML memiliki *simple type restriction* untuk menetapkan: panjang (*length*) suatu *string*, format *string* (*pattern*). *Simple type restriction* dapat juga digunakan untuk menetapkan *domain constraint* (*enumeration*) dan *range constraint* (*minInclusive dan maxInclusive*). Untuk keperluan *interrelation constraint* W3C menetapkan *element key* dan *keyref*.

Software Quality Assurance

Galin (2004:180) mendefinisikan evaluasi *software* sebagai suatu proses formal yang dilakukan oleh tim penguji khusus, dimana suatu unit *software*, beberapa unit *software* yang terintegrasi, atau keseluruhan paket *software* diuji dengan menjalankan program tersebut pada komputer. Semua proses evaluasi dilakukan menurut prosedur pemeriksaan dan studi kasus yang dirancang.

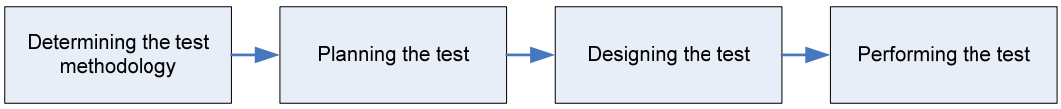
Tujuan dari evaluasi *software* menurut Galin (2004:181) adalah untuk mengidentifikasi dan mengungkap sebanyak mungkin kesalahan suatu *software* yang dievaluasi. Selain itu agar *software* yang telah dikoreksi kesalahannya dapat dievaluasi kembali, sampai kepada suatu tingkat kualitas yang dapat diterima.

Klasifikasi evaluasi *software* yang dilakukan dalam penelitian ini adalah *Black Box*. Galin (2004:187) mendefinisikan *Black Box* sebagai pengujian yang mengabaikan mekanisme internal suatu komponen atau sistem dan hanya memusatkan kepada keluaran yang dihasilkan sebagai jawaban atas kondisi pelaksanaan dan masukan. Selain itu pengujian dilakukan untuk mengevaluasi pemenuhan suatu komponen atau sistem dengan syarat fungsional yang dibutuhkan.

3. METODE PENELITIAN

Metode penelitian mengikuti

langkah-langkah yang diusulkan oleh Galin (2004), seperti yang tampak pada Gambar 1. Pada langkah pertama ditetapkan metode pengujian. Berdasarkan tujuan penelitian yaitu mengevaluasi suatu fasilitas *software* XML *Editor* yang berfungsi sebagai *validator* XML *Document* berdasarkan XML *Schema*. Untuk itu klasifikasi pengujian yang diperlukan adalah ketepatan dan kelengkapan hasil validasi XML *Editor*. Adapun *software* yang akan dievaluasi adalah XMLwriter 2.7 (XMLwriter), <oXygen/> XML Editor 9.3 (oXygen), ALTOVA xmlspy 2008 Enterprise Edition version 2008 rel. 2 sp1 (ALTOVA), Microsoft Visual Studio 2005 Professional Edition Version 8.0.50727.42 (Visual Studio).



Gambar 1. Proses Evaluasi
Sumber: Galin (2004:218)

Pada langkah kedua dirancang pengujian. Dalam penelitian ini, 3 *property* basis data, yakni: *null status*, *data type*, dan *data constraint* dalam XML *Document* divalidasi berdasarkan XML *Schema*. Lebih lanjut *data constraint* yang akan dievaluasi adalah *domain constraint*, *range constraint*, dan *interrelation constraint*. Untuk melakukan pengujian diperlukan XML *Editor* yang sudah diinstal dalam komputer dan XML *Document* serta XML *Schema* yang akan diuji. Pengujian hanya dilakukan satu kali karena manfaat dari penelitian ini hanya untuk mendapatkan informasi yang tepat dan lengkap dari proses validasi XML *Editor*.

Langkah selanjutnya adalah mendesain pengujian. Klasifikasi evaluasi *software* yang dilakukan dalam penelitian ini adalah *Black Box*. Untuk itu, pengujian hasil validasi XML *Editor* berdasarkan informasi yang dihasilkan oleh XML *Editor* tersebut. Sebagai masukan XML *Editor* dibuatkan contoh kasus XML *Schema* untuk setiap *property*. Setelah itu mengisi data pada XML *Document* dengan data yang tidak valid. Proses dilanjutkan dengan memvalidasi XML *document* berdasarkan XML *Schema*. Kemudian mengevaluasi informasi validasi yang dihasilkan oleh XML *Editor*.

Langkah terakhir adalah menjalankan prosedur yang dibuat dengan memuat XML *Schema* dan XML *Document* ke dalam masing-masing XML *Editor*. Kemudian menjalankan fasilitas XML *validator* untuk memvalidasi XML *Document* berdasarkan XML *Schema*. Berdasarkan pesan kesalahan validasi dapat diketahui seberapa tepat dan lengkap hasil validasi masing-masing XML *Editor*. Kemudian mendokumentasi hasil validasi ke dalam suatu tabel sebagai pembanding antara beberapa XML *Editor*.

4. HASIL DAN PEMBAHASAN

Validasi *Null Status*

XML *Schema* yang tampak pada Gambar 2 menunjukkan *element* SSN dan *element* Name berstatus *Not Null*. Sedangkan *element* JobCode berstatus *Null*. Dalam XML *Schema*, *Null Status* dinyatakan dengan *nillable="true"*. Dalam XML *Document*, *Null Status* untuk *element* bertipe data selain *string* dituliskan dengan *<JobCode xs:nil="true"></JobCode>*, sebagai contoh pada EMPLOYEE dengan SSN E002 ditunjukkan pada Gambar 2. Dengan memperhatikan aturan tersebut, maka validasi XML *Document* berdasarkan XML *Schema* dilakukan oleh XMLwriter,

oXygen, ALTOVA, dan Visual Studio. Hasil validasi menunjukkan bahwa XML *Document* tersebut valid. Penulisan *Null Status* pada *element* JobCode yang bertipe bukan *string* dengan <*JobCode*></*JobCode*> pada EMPLOYEE dengan SSN E001 menyebabkan XML *Document* tidak valid. Keseluruhan XML *Editor* dapat memvalidasi kesalahan tersebut, seperti yang ditunjukkan pada Tabel 1 No 1.

XML *Schema*

```
<xs:element name="EMPLOYEE">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="SSN" type="xs:string" />
      <xs:element name="Name" type="xs:string" />
      <xs:element name="JobCode" type="xs:byte"
          nillable="true" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

XML *Document*

```
<EMPLOYEE>
  <SSN>E001</SSN>
  <Name>Susi Setiawati</Name>
  <JobCode></JobCode>
</EMPLOYEE>
<EMPLOYEE>
  <SSN>E002</SSN>
  <Name>Amir</Name>
  <JobCode xs:nil="true"></JobCode>
</EMPLOYEE>
```

Gambar 2. Validasi null status dengan nillable JobCode

Berdasarkan XML *Schema* yang sama, divalidasi XML *Document* dengan membuang salah satu *Not Null element*. Dalam kasus ini, *element* yang dibuang adalah *element* Name seperti yang tampak pada Gambar 3. Dari XML *Schema* tampak *element* tersebut harus ada dalam XML *Document*. Keseluruhan XML *Editor* dapat memvalidasi XML *Document* berdasarkan XML *Schema* seperti yang ditunjukkan pada Tabel 1 No 2.

XML *Schema*

```
<xs:element name="EMPLOYEE">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="SSN" type="xs:string" />
      <xs:element name="Name" type="xs:string"/>
      <xs:element name="JobCode" type="xs:byte"
        nillable="true" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

XML *Document*

```
<EMPLOYEE>
    <SSN>E002</SSN>
    <JobCode
    xs:nil="true"></JobCode>
</EMPLOYEE>
```

Gambar 3. Validasi dengan membuang not null element Name pada XML Document

Tag element yang berstatus *Null* tetap harus dituliskan dalam XML *Document* dengan memperhatikan ketentuan sebelumnya. Pada Gambar 4 tampak *tag element* JobCode tidak disertakan dalam XML *Document*. Keseluruh XML *Editor* dapat memvalidasi kasus tersebut tampak pada Tabel 1 No 3.

XML *Schema*

```
<xs:element name="EMPLOYEE">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="SSN" type="xs:string" />
      <xs:element name="Name" type="xs:string"/>
      <xs:element name="JobCode" type="xs:byte"
          nillable="true" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

XML *Document*

```
<EMPLOYEE>
    <SSN>E002</SSN>
    <Name>Amir</Name>
</EMPLOYEE>
```

Gambar 4. Validasi dengan membuang null element JobCode pada XML Document

Dalam XML ada cara lain untuk menyatakan *Not Null* selain menggunakan *nillable="true"*, yakni dengan *attribute minOccurs*="0". Pada Gambar 5 tampak *element* JobCode *minOccurs*="0" (*null*), sedangkan *element* SSN dan Name secara *default minOccurs*="1" (*not null*). Semua XML *Editor* menyatakan XML *Document* tersebut valid terhadap XML *Schema*. Hal yang membedakan *minOccurs* dari *nillable* adalah *tag element* dengan *minOccurs*="0" dapat tidak dituliskan dalam XML *Document*. *Tag element* dengan *minOccurs*="0" dapat dituliskan dalam XML *Document* seperti yang tampak pada Gambar 6 untuk tipe data *string*. Dalam contoh kasus, tipe data adalah *byte* maka seluruh XML *Editor* menyatakan XML *Document* tersebut tidak valid terhadap XML *Schema*, seperti yang tampak pada Tabel 1 No 4.

XML *Schema*

```
<xs:element name="EMPLOYEE">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="SSN" type="xs:string"/>
      <xs:element name="Name" type="xs:string"/>
      <xs:element name="JobCode" type="xs:byte"
          minOccurs="0" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

XML *Document*

```
<EMPLOYEE>
    <SSN>E001</SSN>
    <Name>Susi Setiawati</Name>
    <JobCode>1</JobCode>
</EMPLOYEE>
<EMPLOYEE>
    <SSN>E002</SSN>
    <Name>Amir</Name>
</EMPLOYEE>
```

Gambar 5. Validasi null status dengan minOccurs pada element JobCode

XML *Schema*

```
<xs:element name="EMPLOYEE">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="SSN" type="xs:string"/>
      <xs:element name="Name" type="xs:string"/>
      <xs:element name="JobCode" type="xs:byte"
          minOccurs="0" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

XML *Document*

```
<EMPLOYEE>
    <SSN>E001</SSN>
    <Name>Susi Setiawati</Name>
    <JobCode>1</JobCode>
</EMPLOYEE>
<EMPLOYEE>
    <SSN>E002</SSN>
    <Name>Amir</Name>
    <JobCode></JobCode>
</EMPLOYEE>
```

Gambar 6. Validasi null status dengan minOccurs dan tag JobCode pada XML Document

Validasi Data *Type*

Pada bagian ini dilakukan validasi tipe data XML *Document* berdasarkan XML *Schema*. Tampak pada Gambar 7, SSN bertipe *integer* sedangkan dalam XML *Document* SSN diisi dengan E001 (*string*). Hal tersebut menunjukkan XML *Document* tidak valid berdasarkan XML *Schema*. Semua XML *Editor* dapat memvalidasi kesalahan seperti yang tampak pada Tabel 1 No 5.

Dalam *data type* juga dievaluasi panjang (*length*) suatu *element*. Tampak pada Gambar 8, *element* Name bertipe data *string* dengan panjang maksimum 10. Sedangkan dalam XML *Document*, jumlah karakter "Susi Setiawati" adalah 14. Untuk itu XML *Document* tidak valid terhadap XML *Schema*. Hasil evaluasi terhadap seluruh XML *Editor* tampak pada Tabel 1 No 6.

XML *Schema*

```
<xs:element name="EMPLOYEE">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="SSN" type="xs:integer" />
      <xs:element name="Name" type="xs:string" />
      <xs:element name="JobCode" type="xs:byte" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

XML *Document*

```
<EMPLOYEE>
  <SSN>E001</SSN>
  <Name>Susi Setiawati</Name>
  <JobCode>1</JobCode>
</EMPLOYEE>
```

Gambar 7. Validasi data type

XML *Schema*

```
<xs:element name="Name">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="10" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

XML *Document*

```
<EMPLOYEE>
    <SSN>11-AB</SSN>
    <Name>Susi Setiawati</Name>
    <JobCode>1</JobCode>
</EMPLOYEE>
```

Gambar 8. Validasi *length*

Validasi *pattern* untuk tipe data *string* XML *Document* berdasarkan XML *Schema* ditunjukkan pada Gambar 9. Hasil evaluasi terhadap keseluruhan XML *Editor* tampak pada Tabel 1 No 7. XML *Editor* oXygen dan Visual Studio lebih unggul dari kedua XML *Editor* lainnya karena XML *Editor* tersebut dapat membedakan huruf besar dan huruf kecil pada *pattern string*.

XML *Schema*

```
<xs:element name="SSN">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="\d{2}-[A-Z]{2}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

XML *Document*

```
<EMPLOYEE>
    <SSN>E001</SSN>
    . . .
</EMPLOYEE>
<EMPLOYEE>
    <SSN>11-Ab</SSN>
    . . .
</EMPLOYEE>
<EMPLOYEE>
    <SSN>12-AB</SSN>
. . .
</EMPLOYEE>
```

Gambar 9. Validasi pattern

Validasi *Domain Constraint*

Validasi *domain constraint* ditunjukkan pada Gambar 10. Dari gambar tersebut tampak kemungkinan Language adalah VB.Net, C++, atau Pascal. Sedangkan dalam XML *Document* Language diisi dengan Cobol, yang bukan merupakan *domain* Language. Semua XML *Editor* dapat menvalidasi kesalahan tersebut, seperti yang tampak pada Tabel 1 No 8.

XML *Schema*

```
<xs:element name="Language">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value ="VB.Net" />
      <xs:enumeration value ="C++" />
      <xs:enumeration value ="Pascal" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

XML *Document*

```
<PROGRAMMER>
    <SSN>E001</SSN>
    <Language>Cobol</Language>
    <OperatingSystem>Window</OperatingSystem>
</PROGRAMMER>
```

Gambar 10. Validasi domain constraint

Validasi *Range Constraint*

Pada Gambar 11 tampak JobCode minimum adalah 1 dan JobCode maksimum adalah 3. Sedangkan dalam XML *Document* JobCode berisi 4. Keseluruhan XML *Editor* dapat memvalidasi kesalahan tersebut seperti yang tampak pada Tabel 1 No 9.

XML *Schema*

```
<xs:element name="JobCode">
  <xs:simpleType>
    <xs:restriction base="xs:byte">
      <xs:minInclusive value="1"/>
      <xs:maxInclusive value="3"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

XML *Document*

```
<EMPLOYEE>
    <SSN>E002</SSN>
    <Name>Amir</Name>
    <JobCode>4</JobCode>
</EMPLOYEE>
```

Gambar 11. Validasi range constraint

Validasi *Interrelation Constraint*

Pada Gambar 12 tampak isi *element* SSN dalam *complexType* COMPUTER adalah E001. Data tersebut tidak ditemukan dalam *complexType* EMPLOYEE. Keseluruhan XML *Editor* dapat melakukan validasi terhadap data yang tidak konsisten, seperti yang tampak pada Tabel 1 No 10.

```
<EMPLOYEE>
    <SSN>11-AB</SSN>
    <Name>Susi Setiawati</Name>
    <JobCode>1</JobCode>
</EMPLOYEE>
<EMPLOYEE>
    <SSN>12-AB</SSN>
    <Name>Amir</Name>
    <JobCode>3</JobCode>
</EMPLOYEE>
```

```
<COMPUTER>
    <SerialNumber>CNF32113XW</SerialNumber>
    <Make>USA</Make>
    <DiskCapacity>40 GB</DiskCapacity>
    <SSN>E001</SSN>
</COMPUTER>
```

Gambar 12. Validasi referential integrity

Pada Gambar 13 tampak *element* SerialNumber berperan sebagai *primary key* pada *complexType* COMPUTER. Pada XML *Document* tampak SerialNumber berisi data yang sama untuk komputer yang berbeda. Keseluruhan XML *Editor* dapat melakukan validasi terdapat *primary key* yang duplikat, seperti tampak pada Tabel 1 No 11.

Pada Gambar 14 tampak *element* SSN pada *complexType* EMPLOYEE yang berperan sebagai *primary key* dan bertipe *integer*. Sedangkan *element* SSN pada *complexType* COMPUTER bertipe *decimal* dan berperan sebagai *foreign key*. Meskipun kedua *element* tersebut bertipe data *number* tetapi berbeda *size*. XML *Editor* XMLwriter dan oXygen tidak dapat mendeteksi kesalahan tersebut. Sedangkan ALTOVA dan Visual Studio dapat mendeteksi kesalahan seperti yang tampak pada Tabel 1 No 12.

XML *Schema*

```
<xs:key name="COMPUTERKey">
    <xs:selector xpath=".//COMPUTER" />
    <xs:field xpath="SerialNumber" />
</xs:key>
```

XML *Document*

```
<COMPUTER>
    <SerialNumber>CNF32113XW</SerialNumber>
    <Make>USA</Make>
    <DiskCapacity>40 GB</DiskCapacity>
    <SSN>E001</SSN>
</COMPUTER>
<COMPUTER>
    <SerialNumber>CNF32113XW</SerialNumber>
    <Make>USA</Make>
    <DiskCapacity>30 GB</DiskCapacity>
</COMPUTER>
```

Gambar 13. Validasi terhadap dublikat primary key

XML *Schema*

```
<xs:element name="EMPLOYEE">
    . . .
    <xs:element name="SSN" type="xs:integer" />
    . . .
</xs:element>
<xs:element name="COMPUTER">
    . . .
    <xs:element name="SSN" type="xs:decimal"
        minOccurs="0" />
    . . .
</xs:element>
. . .
<xs:keyref name="EMPLOYEE-COMPUTER"
refer="EMPLOYEEKey" >
    <xs:selector xpath=".//COMPUTER" />
    <xs:field xpath="SSN" />
</xs:keyref>
```

XML *Document*

```
<EMPLOYEE>
    <SSN>1</SSN>
    <Name>Susi Setiawati</Name>
    <JobCode>1</JobCode>
</EMPLOYEE>
<COMPUTER>
    <SerialNumber>CNF32113XW</SerialNumber>
    <Make>USA</Make>
    <DiskCapacity>40 GB</DiskCapacity>
    <SSN>1</SSN>
</COMPUTER>
    <COMPUTER>
    <SerialNumber>CNF32111AB</SerialNumber>
    <Make>USA</Make>
    <DiskCapacity>30 GB</DiskCapacity>
</COMPUTER>
```

Gambar 14. Validasi terhadap tipe data primary key dan foreign key

Tabel 1.
Ketepatan Dan Kelengkapan Informasi Validasi XML Editor

No	Validasi	XMLwriter	oXygen	ALTOVA	Visual Studio
1.	*Null status* tanpa *nillable* pada *tag* JobCode	*√	*√	*√	*√
2.	Membuang *Not Null Element* Name Pada XML *Document*	*√	*√	*√	*√
3.	Membuang *Null Element* JobCode Pada XML *Document*	*√	*√	*√	*√
4.	*Null status* dengan *minOccurs* dan *tag* JobCode pada XML *Document*	*√	*√	*√	*√
5.	Data *Type*	*√	*√	*√	*√
6.	*Length*	*√	*√	*√	*√
7.	*Pattern*	√	*√	√	*√
8.	*Domain constraint*	*√	*√	*√	*√
9.	*Range constraint*	*√	*√	*√	*√
10.	*Referential integrity*	*√	*√	*√	*√
11.	Dublikat *primary key*	*√	*√	*√	*√
12.	Tipe data *primary key* dan *foreign key*			*√	*√

Keterangan: *=tepat dan √=lengkap

5. KESIMPULAN DAN SARAN

Informasi yang dihasilkan dari proses validasi XML *Document* berdasarkan XML *Schema* oleh XML *Editor* XMLwriter, oXygen, ALTOVA, dan Visual Studio sudah tepat dan lengkap. XML *Editor* oXygen dan Visual Studio lebih unggul dibandingkan

dengan kedua XML *Editor* lainnya karena dapat membedakan huruf besar dan huruf kecil pada *pattern string*. Hanya XML *Editor* ALTOVA dan Visual Studio yang dapat mendeteksi ketidakvalidan *element* yang berperan sebagai *referential integrity* antara *complexType* yang bertipe sama tetapi berbeda *size.*

Penelitian ini dapat dikembangkan dengan melakukan validasi terhadap *intrarelation constraint* dan *interrelation constraint* yang terkait dengan minimum *cardinality mandatory-mandatory* antara *complex type parent* dan *complex type child*.

6. DAFTAR PUSTAKA

ALTOVA. 10-07-2008. ALTOVA xmlspy 2008 Enterprise Edition version 2008 rel. 2 sp1. http://www.altova.com/download/xmlspy/xml_editor_enterprise.html

Galin Daniel. 2004. *Software Quality Assurance: From Theory To Implementation 1st*. Pearson Prentice Hall. New Jersey.

Grauer, Robert T. And Maryann Barber. 2006. *Microsoft Office Access 2003: Revised Comprehensive*. Pearson Prentice Hall. New Jersey.

Kroenke, David M. 2006. *Database Processing: Fundamentals, Design, and Implementation 10th*. Pearson Prentice Hall. New Jersey.

oXygen. 10-07-2008. <oXygen/> XML Editor 9.3. http://www.oxygenxml.com

Visual Studio, Visual Studio 2005 Professional Edition Version 8.0.50727.42

W3C. 01-07-2008. XML Schema Part 0: Primer Second Edition. http://www.w3.org/TR/xmlschema-0.

W3C. 01-07-2008. XML Schema Part 1: Structures Second Edition. http://www.w3.org/TR/xmlschema-1.

W3C. 01-07-2008. XML Schema Part 2: Datatypes Second Edition, http://www.w3.org/TR/xmlschema-2.

XMLwriter. 10-07-2008. XMLwriter Version 2.7. http://xmlwriter.net

Yuliana, Oviliani Yenty. 2006. "An Evaluation of XML Designer in Microsoft Visual Studio.Net." Proceeding of the second International Seminar Information and Communication Technology Seminar. Volume 1, nomor 1. Surabaya.