Penilaian Kesamaan *Entity Relationship Diagram* dengan Algoritme *Tree Edit Distance*

Humasak Simanjuntak[1], Rosni Lumbantoruan[2], Wiwin Banjarnahor[3], Erisha Sitorus[4], Magdalena Panjaitan[5], Sintong Panjaitan[6]

***Abstract*—Main competency in database learning is ability to design Entity Relationship Diagram (ERD). Generally, lecturer gives task to students to design an ERD with some requirements. These ERDs are then assessed by comparing them with the answers. In practice, the process takes long time and it is possible that the lecturer grades the students inconsistently. Furthermore, plagiarism could be occured without being noticed by the lecturer. This research aims to design and build an application that assess similarity of ERD. The application apply tree edit distance algorithm in checking ERD similarity. ERD is exported into XMI document and then processed using the tree edit distance algorithm. The results show that ERD similarity value depends on number of insert, delete, and rename operation in tree edit distance Algorithm rather than number of difference component.**

***Intisari*—Pada pembelajaran basis data, salah satu kompetensi yang harus dicapai oleh mahasiswa adalah mampu merancang *Entity Relationship Diagram* (ERD) dengan benar. Untuk mengukur kemampuan tersebut, dosen memberikan tugas untuk merancang ERD yang sesuai dengan kebutuhan sistem. Kemudian, dosen memeriksa dan memberikan nilai berdasarkan kebenaran model ERD yang dibuat oleh mahasiswa dengan membandingkannya dengan jawaban ERD yang diberikan. Pada pelaksanaannya, pemberian nilai ERD tersebut membutuhkan waktu yang cukup lama (berbanding lurus dengan jumlah mahasiswa) dan sangat mungkin terjadi ketidakkonsistenan dalam pemberian nilai. Selain itu, sebuah ERD sangat mungkin merupakan hasil plagiarisme terhadap model yang lain tanpa diketahui oleh dosen. Makalah ini bertujuan untuk merancang dan membuat sebuah aplikasi yang memberikan nilai kesamaan dari sebuah ERD dengan ERD yang lain. Aplikasi menggunakan algoritme *tree edit distance* dalam memeriksa kesamaan ERD. ERD yang telah diubah menjadi XMI dokumen akan menjadi masukan bagi algoritme *tree edit distance*. Hasil pengujian menunjukkan bahwa nilai kesamaan ERD bergantung kepada jumlah operasi yang dilakukan oleh algoritme *tree edit distance*. Jumlah beda antar ERD tidak terlalu berpengaruh apabila tidak memberikan efek yang signifikan terhadap struktur ERD.**

Kata Kunci— Entity Relationship Diagram, tree edit distance, ERD similarity value

I. PENDAHULUAN

Basis data merupakan salah satu komponen dalam sebuah sistem informasi yang berfungsi sebagai tempat penyimpanan data. Sebelum basis data diimplementasikan, perancang membuat sebuah diagram yang disebut *Entity Relationship Diagram* (ERD). ERD merupakan sebuah diagram yang menggambarkan setiap entitas yang terkait pada suatu sistem. Kemudian, ERD akan diterjemahkan menjadi *Conceptual Data Model* dan *Physical Data Model*. Pada *Physical Data Model*, setiap komponen yang ada pada ERD akan menjadi objek-objek yang disebut tabel, atribut, *data type*, *relationship*, *primary key*, *foreign key*, dan objek lain. Objek-objek tersebut diterjemahkan ke dalam *Database Management System* (DBMS) menggunakan SQL maupun fasilitas lain yang telah disediakan oleh DBMS.

Dalam pembelajaran basis data, perancangan ERD merupakan proses yang sangat penting untuk menghasilkan sebuah sistem informasi yang baik. Setiap mahasiswa harus mampu menghasilkan ERD yang baik dan benar sesuai dengan bisnis proses sistem. ERD yang baik dan benar akan menghasilkan sebuah basis data (tabel, kolom, tipe data, relasi) yang sesuai dengan kebutuhan sistem dan memberikan perekaman data yang akurat.

Pada saat ini, pemberian penilaian ERD oleh tim pengajar masih dilakukan secara manual, yaitu dengan membandingkan setiap komponen ERD yang dihasilkan oleh mahasiswa dengan model ERD yang disediakan oleh tim pengajar. Hal ini membutuhkan waktu yang lama dan berbanding lurus dengan jumlah mahasiswa (semakin banyak mahasiswa semakin banyak ERD yang akan dinilai). Selain itu, dengan banyaknya ERD yang diperiksa, kemungkinan *human error* juga akan sangat sering terjadi, sehingga tim pengajar memberikan nilai yang tidak konsisten terhadap dua atau lebih ERD mahasiswa yang sama ataupun mirip.

Dengan memperhatikan hal tersebut, maka diperlukan sebuah perangkat lunak untuk menilai kesamaan antara beberapa ERD dan pada akhirnya memberikan nilai atau skor secara otomatis terhadap ERD yang dihasilkan oleh mahasiswa. Beberapa penelitian tentang pengecekan kesamaan ERD telah dilakukan. Salah satunya menggunakan pendekatan *graph based*, yaitu dengan membandingkan beberapa gambar ERD. Dalam penelitian tersebut, nilai yang diberikan belum akurat karena belum dapat membandingkan struktur ERD yang lengkap, seperti *cardinality*, *weak entity*, dan *primary key* [1]. Oleh karena itu, pada makalah ini dirancang sebuah perangkat lunak yang mampu mendeteksi kesamaan beberapa ERD dan memberikan nilai berdasarkan kesamaan tersebut. ERD yang akan dibandingkan terlebih dahulu diubah ke dalam format dokumen *Extensible Markup Language* (XML) [2], [3]. XML merupakan sebuah format teks sederhana yang berasal dari *Standard Generalized Markup Language* (SGML) yang digunakan untuk pertukaran

[1,2,3] *Dosen, Program Studi Sarjana Sistem Informasi, Institut Teknologi Del, Jl. Sisingamangaraja, Sitoluama, Laguboti, 22381, INDONESIA; (e-mail: humasak@gmail.com, rosni.ltoruan@gmail.com, wiwin.banjarnahor@gmail.com)*

[4,5,6] *Mahasiswa, Program Studi D3 Teknik Informatika, Institut Teknologi Del, Jl. Sisingamangaraja, Sitoluama, Laguboti, 22381.*

data [4]. XML *Metadata Interchange* (XMI) merupakan sebuah standar yang memungkinkan pengguna untuk menampilkan objek-objek dengan menggunakan dokumen XML [4]. Dokumen XML dimodelkan sebagai *tree* dengan *root*, *nodes*, *edge*, dan *leaf* [4]. Dengan merepresentasikan setiap komponen ERD ke dalam *tree* (dokumen XML), maka entitas, atribut, relasi, dan *cardinality* dapat teridentifikasi, sehingga nilai kesamaan ERD yang dihasilkan akurat dan konsisten.

II. DETEKSI KESAMAAN ERD

Beberapa penelitian terkait penilaian tingkat kesamaan ERD secara otomatis telah dilakukan. Sebuah penelitian telah merekomendasikan dua metode yang dapat digunakan untuk melakukan penilaian terhadap tingkat kesamaan ERD, yaitu penilaian tingkat kesamaan ERD menggunakan algoritme *tree edit distance* dan penilaian kesamaan ERD menggunakan teknik klasifikasi pada *machine learning* [2]. Penelitian tersebut menjadi sumber referensi utama makalah ini. Berdasarkan penelitian tersebut, penilaian kesamaan ERD dilakukan dengan terlebih dahulu menghasilkan struktur XML dari setiap ERD. Kemudian, komponen ERD akan diidentifikasi melalui *tag-tag* pada XML untuk kemudian dibandingkan. Teknik yang sama juga telah diimplementasikan pada UML Class Diagram [3]. Hasil deteksi kesamaan yang dilakukan sangat tergantung pada jumlah elemen, *access modifier*, arah relasi kelas, dan urutan elemen.

Selain itu, penelitian mengenai pemeriksaan kesamaan dari diagram ERD juga telah dilakukan. Pengecekan persamaan antar ERD tersebut dilakukan dengan membandingkan *minimal meaningful units* (MMUs) ERD [1]. ERD memiliki tiga tipe MMU, yaitu *entities*, *relationships*, dan *subtypes*. Lima langkah yang dilakukan dalam percobaan tersebut yaitu menerjemahkan *raster-based input* menjadi sekumpulan *diagrammatic primitives*, seperti bentuk kotak, garis, dan teks, yang kemudian akan diidentifikasikan sebagai *low-level*, ciri (*feature*) domain yang spesifik, dan akan disebut sebagai MMU. Langkah selanjutnya yang dilakukan yaitu menggabungkan MMU menjadi *high-level* sebagai *abstract features* dan akan dihasilkan makna dari diagram MMU tersebut. Setelah perbandingan MMU, dilakukan penghitungan ukuran kesamaan untuk setiap pasangan MMUs dan pemberian nilai sesuai dengan kesamaan MMU yang dimiliki.

Beberapa penggunaan algoritme lain terkait pendeteksian kesamaan sudah pernah dilakukan, seperti penelitian mengenai *cloning* pada *class diagram* dengan teknik *suffix array* untuk mendeteksi elemen-elemen yang berulang pada *class diagram* dan menghapus elemen tersebut jika diperlukan [5]. Elemen yang dimaksud adalah kelas, atribut, dan operasi. Pendekatan yang dilakukan pada penelitian tersebut dimulai dengan pemodelan *class diagram* dengan menggunakan perangkat lunak UML. *Class diagram* diubah ke dalam bentuk XML dan kemudian diuraikan ke dalam token. Kemudian, pencocokan token dilakukan dengan teknik *suffix array* untuk memperoleh jumlah *clone* yang dideteksi. Penelitian tersebut bertujuan untuk meningkatkan kualitas model, mengurangi kebutuhan jika *clone*-nya telah dideteksi, dan memudahkan pemeliharaan suatu model ketika terdeteksi sudah ada. Hasil yang diperoleh pada penelitian tersebut adalah mengetahui jumlah *clone* yang berhasil dideteksi pada *class diagram* yang telah dibandingkan.

Penelitian mengenai kesamaan dokumen XML dengan menggunakan algoritme *tree edit distance* juga sudah dilakukan [6]. Penelitian ini menerapkan teknik *Dynamic Programming* untuk memperoleh *edit distance* antar *string*. Penggunaan teknik tersebut bertujuan untuk menemukan operasi *edit* dalam metransformasikan satu *tree* ke *tree* yang lain. Selanjutnya, PrestoSoft juga telah menyediakan perangkat lunak untuk membandingkan *file* (termasuk dokumen XML) yang disebut ExamDiff Pro. Versi gratis aplikasi ini disebut DiffNow dan dapat diakses di situs http://www.diffnow.com/ [7]. Perangkat lunak ini dikembangkan berdasarkan algoritme Eugene Myers untuk membandingkan beberapa *file* dan akan digunakan sebagai pembanding terhadap aplikasi yang dikembangkan (AssERSi).

III. METODOLOGI

Seperti yang telah dijelaskan sebelumnya, makalah ini disusun untuk menguji metode kedua yang telah diusulkan pada penelitian sebelumnya [2]. Keluaran dari penelitian dalam makalah ini adalah perangkat lunak atau aplikasi untuk melakukan penilaian kesamaan ERD. Namun, penghitungan nilai kesamaan dilakukan menggunakan formula yang berbeda, sehingga nilai kesamaan yang dihasilkan sesuai dengan perbedaan jumlah komponen antara dua ERD.

Pada makalah ini, metode penilai kesamaan ERD dibagi menjadi beberapa tahapan sebagai berikut.

1. Mengubah ERD ke dokumen XML.
2. Optimisasi Struktur XML.
3. Menyimpan komponen ERD pada basis data.
4. Menghitung Nilai Kesamaan ERD.

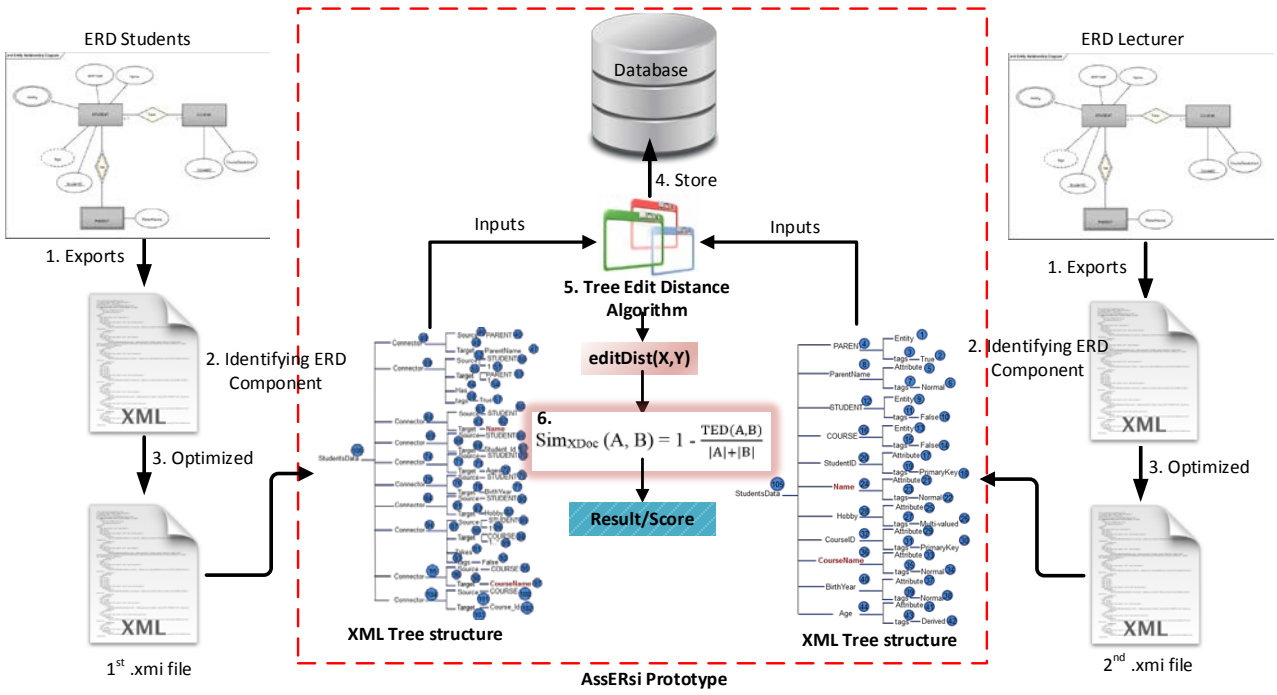
Proses lengkap metode yang dilakukan untuk menilai kesamaan ERD ditunjukkan pada Gbr. 1.

A. Mengubah ERD ke Dokumen XML

Pada makalah ini, struktur ERD yang digunakan berupa format XML. Struktur ERD digambarkan dengan menggunakan perangkat lunak Sparx Systems Enterprise Architect. Setelah ERD yang lengkap dihasilkan, Sparx memungkinkan kita untuk mengubah struktur ERD tersebut ke dalam formal XML, dengan ektensi *.xmi*. Versi *xmi* yang digunakan adalah xmi 2.1.

B. Identifikasi Komponen ERD

Setelah dokumen XML dihasilkan, dilakukan identifikasi komponen dari struktur ERD. Pada makalah ini, komponen ERD utama yang digunakan dalam pemeriksaan kesamaan adalah entitas, atribut, relasi, *cardinality*, tipe entitas, dan tipe atribut. Oleh karena itu, pada tahap ini akan dilakukan pemrosesan terhadap dokumen XML untuk menemukan semua entitas, atribut dari entitas, relasi antar entitas, dan *cardinality* dari setiap relasi.



Gbr. 1 Metode untuk menilai kesamaan *Entity Relationship Diagram.*

C. Optimasi Struktur XML

Dokumen XML merupakan representasi dari sebuah struktur *tree*. Struktur *tree* memiliki banyak *node* (elemen xml atau *tag* xml) yang merepresentasikan semua komponen pada ERD. Komponen ERD utama yang digunakan adalah entitas, atribut, relasi, *cardinality*, tipe entitas, dan tipe atribut. Dengan demikian, akan terdapat banyak *tag* yang tidak berkaitan dengan komponen-komponen tersebut. Oleh karena itu, beberapa struktur XML dari ERD akan dieliminasi berdasarkan komponen-komponen yang telah teridentifikasi. Kemudian, sebuah struktur *tree* yang lebih sederhana akan dihasilkan berdasarkan struktur *tree* dari XML sebelumnya. Dengan kata lain, struktur tidak berubah, namun beberapa *tag* yang tidak perlu akan dieliminasi.

D. Menyimpan Komponen ERD pada Basis Data

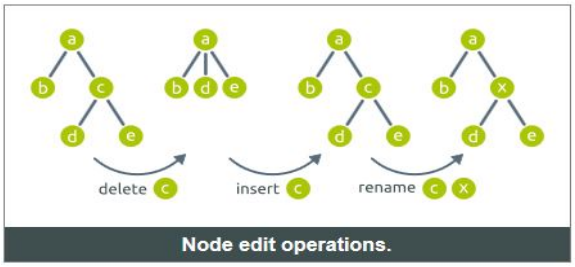
Untuk menghasilkan aplikasi AssERSi yang dapat memproses banyak ERD, aplikasi harus mampu menyimpan semua ERD ke dalam basis data. Struktur basis data yang digunakan dalam makalah ini berdasar pada data model dalam penelitian sebelumnya [2].

E. Algoritme Tree Edit Distance

Penelitian dalam makalah ini menggunakan algoritme *tree edit distance* untuk menghitung jarak atau jumlah operasi yang dilakukan untuk mentransformasikan satu *tree* ke *tree* yang lain. Jumlah operasi ini akan menjadi pembeda antar *tree* tersebut.

Dalam penerapan algoritme *tree edit distance*, terdapat tiga operasi *edit* untuk sebuah *tree* dengan node label berurutan, yaitu sebagai berikut [8] - [10].

1. *Delete*, yaitu operasi untuk menghapus *node* dan menghubungkan *children* dari *node* yang dihapus menjadi *children* dari *parent node* yang dihapus tersebut.
2. *Insert*, yaitu operasi untuk menyisipkan ataupun menambahkan *node* ke *tree* yang ada dan mengurutkan *node* tersebut berdasarkan label.
3. *Rename*, yaitu operasi mengubah nama dari label pada setiap *node*. Gbr. 2 merupakan operasi mengubah *node* pada *tree edit distance.*



Gbr. 2 Operasi algoritme *tree edit distance.*

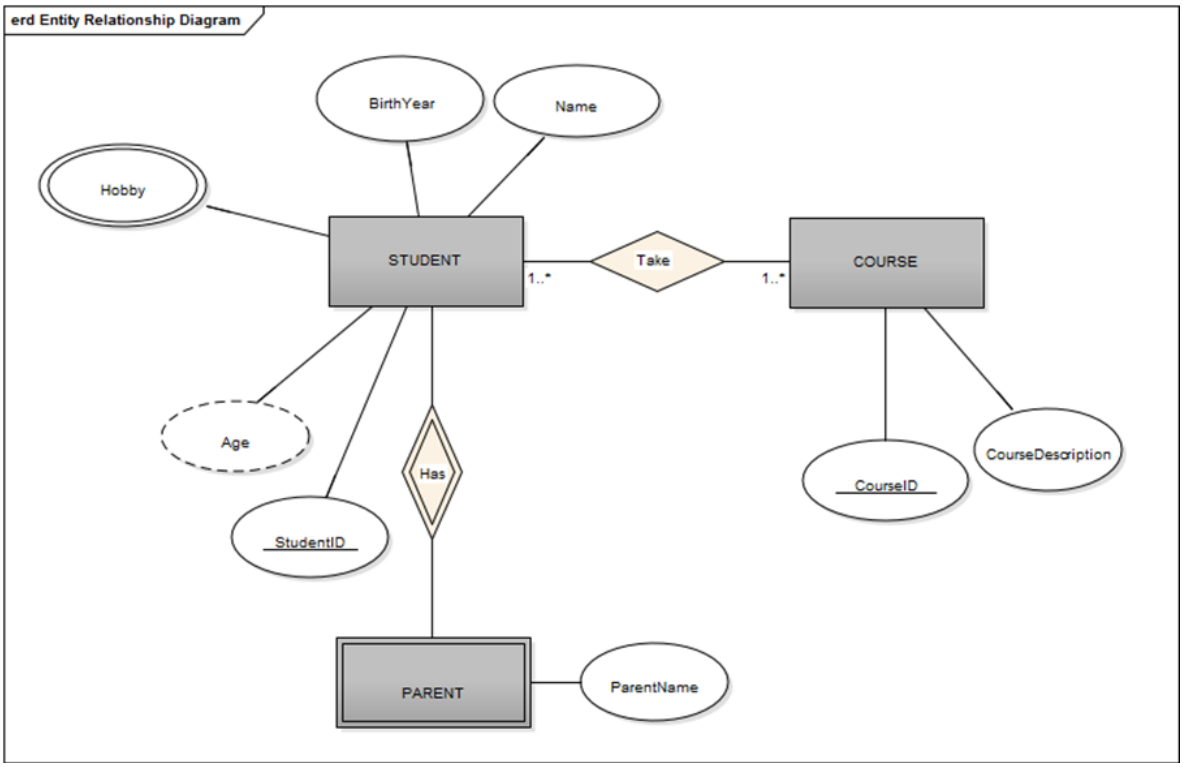
Pada awalnya, *edit distance* adalah sebuah algoritme *dynamic programming* yang digunakan untuk menghitung nilai perbedaan (*distance*) antara dua *string* [11]. Kemudian, algoritme *edit distance* dikembangkan agar dapat melakukan perbandingan terhadap *tree*, sehingga disebut sebagai algoritme *tree edit distance*. *Pseudocode* algoritme *tree edit distance* dapat dilihat pada Gbr. 3 [2].

F. Menghitung nilai kesamaan

Keluaran algoritme *tree edit distance* adalah ***editDist(A,B)***, yaitu jumlah minimum operasi (*insert*, *delete*, *rename*) yang diperlukan untuk melakukan tranformasi dari satu *tree* ke *tree* yang lain.

```
function TreeEditDistance (Input m: integer , n:integer)(Output retval: integer)
Variable:
        A,B : Tree; E : Array of Integer
        Insert,delete,rename : integer
Algoritma:
        {Tree A, B already defined}
        For i ← 0 to m do
              E[i,0] ← i
        endFor;
        For j ← 1 to n do
              E[0,j] ← j
        endFor;
        for i ← 1 to m do
             for j ← 1 to n do
                    insert    = E[i-1,j] + 1
                    delete    = E[i,j-1] + 1
                    rename    = E[i-1,j-1]
                     if A[i] ≠ B[j]
                             rename += 1
                     E[i,j] ← min (insert, delete, rename)
              Endfor
         Endfor;
         retval ← E[m,n]
         return retval
```

Gbr. 3 Algoritme *tree edit distance.*



Gbr. 4 Contoh ERD StudentData.

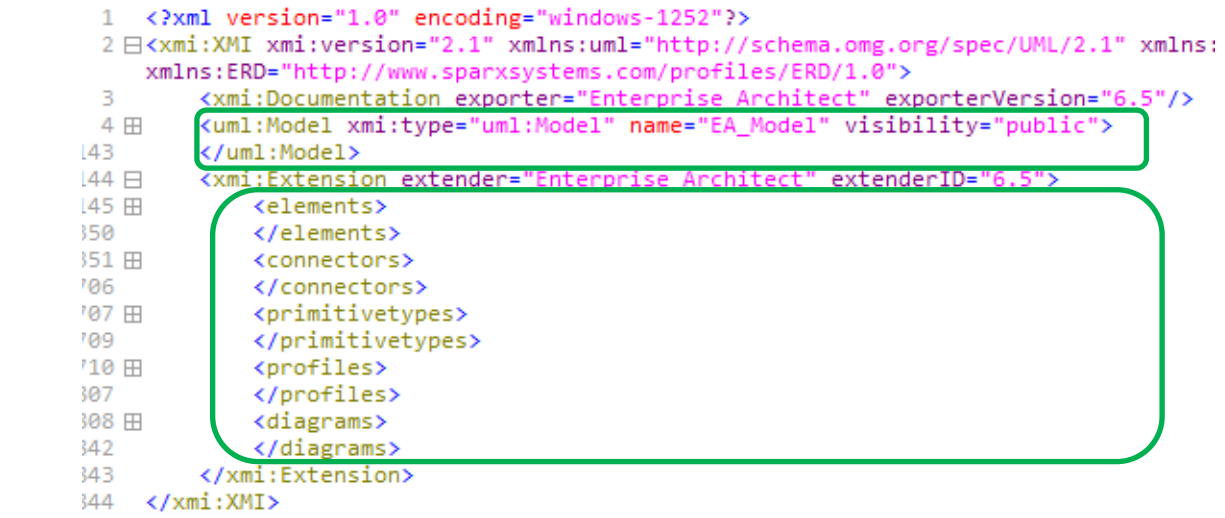
EditDist(A,B) akan menjadi masukan dari fungsi *similarity* yang akan digunakan untuk menghasilkan nilai kesamaan antara ERD mahasiswa dengan ERD jawaban yang diberikan oleh tim pengajar. Berikut ini adalah fungsi *similarity* yang digunakan untuk menghasilkan nilai kesamaan antar beberapa ERD [12].

$$Sim_{XDoc}(A,B) = 1 - \frac{editDist(A,B)}{|A||B|} \quad (1)$$

Hasil fungsi *similarity* ini akan berkisar pada nilai 0 – 1. Apabila nilai semakin besar (mendekati 1), berarti ERD yang dibandingkan memiliki tingkat kesamaan yang tinggi. Setelah nilai kesamaan dihasilkan, maka perangkat lunak AssERSi juga akan menghasilkan semua operasi transformasi yang dilakukan.

IV. ANALISIS KOMPONEN ERD PADA DOKUMEN XML

Pada bagian ini dijelaskan mengenai komponen ERD yang diubah ke dalam dokumen xmi versi 2.1. Berikut merupakan hasil analisis komponen ERD yang direpresentasikan dalam bentuk xmi.



Gbr. 5 Struktur *tag* XMI dokumen dari sebuah gambar ERD.

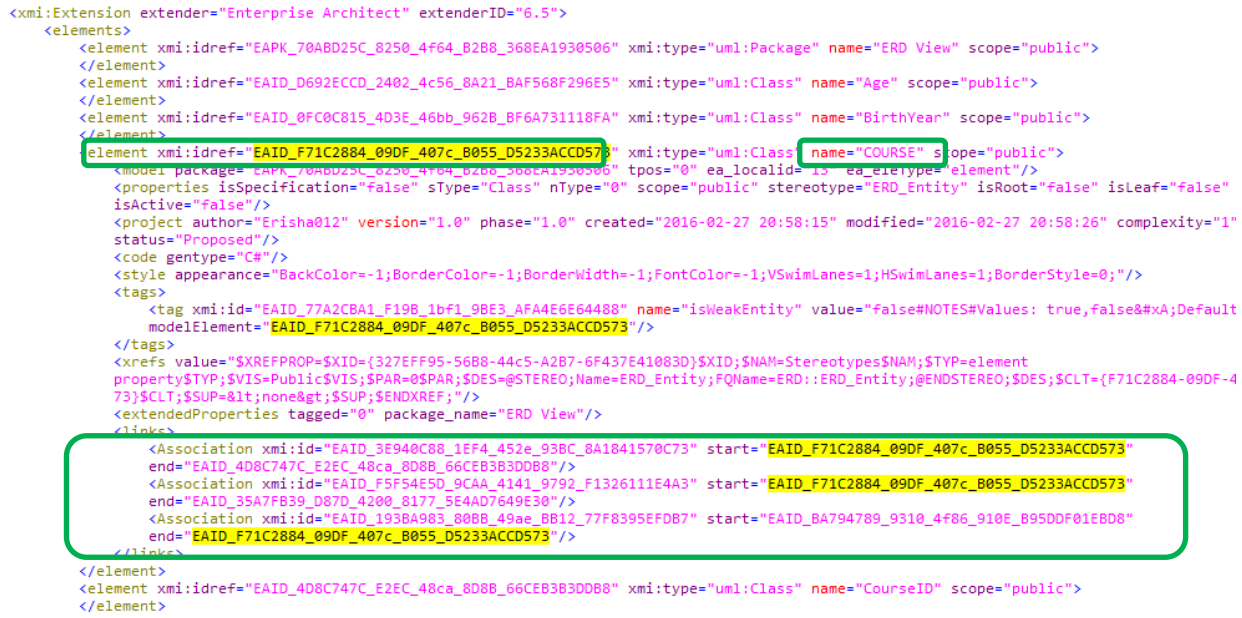
Gbr. 6 *Tag* komponen entitas, atribut, dan relasi pada XMI dokumen.

A. Representasi Komponen ERD pada Dokumen XMI

Sebuah contoh ERD ditunjukkan pada Gbr. 4. Terlihat ERD memiliki tiga entitas, *parent*, *student*, dan *course*, dengan *parent* adalah *weak entity*. Entitas *parent* memiliki satu atribut *ParentName*, entitas *student* memiliki lima atribut, yaitu *studentid*, *name*, *age*, *birthyear*, dan *hobby*. Sedangkan entitas *course* memiliki dua atribut, yakni *courseid* dan *coursedescription*. Setiap *student* memiliki *parent* dan seorang *student* akan mengambil *course* tertentu.

Dengan menggunakan perangkat lunak Sparx Systems Enterprise Architect untuk mengubah ERD ke dalam bentuk dokumen xmi, maka *tag-tag* umum yang merepresentasikan semua komponen ERD pada Gbr. 4 dapat dilihat pada Gbr. 5.

Dengan memperhatikan struktur *tag* pada Gbr 5, tampak sebuah dokumen XMI ERD hanya memiliki dua elemen utama, yaitu elemen **<uml:Model>** dan **<xmi:Extension>**. Jika elemen **<uml:Model>** di-*expand*, dihasilkan struktur dokumen XMI yang lebih lengkap seperti pada Gbr. 6.



Gbr. 7 *Tag* detail informasi setiap komponen ERD pada XMI dokumen.

Pada Gbr. 6 ditunjukkan bahwa semua komponen yang ada pada ERD dapat diidentifikasi melalui elemen *tag* yang berada di dalam *tag* **<uml:Model>,** yang terdiri atas

- <ERD:ERD_Entity>, yang menunjukkan entitas yang terdapat pada ERD,
- <ERD:ERD_Attribute>, yang menunjukkan atribut yang terdapat pada ERD,
- <ERD:ERD_Relationship>, yang menunjukkan relasi yang terjadi antara satu entitas dengan entitas yang lain, dan
- <ERD:ERD_Connector>, menunjukkan keterhubungan antara entitas dengan atributnya.

Selain itu, beberapa *tag* juga memiliki atribut yang menunjukkan tipe dari entitas maupun atribut, yaitu

- *attributeType*, yakni atribut pada *tag* <ERD:ERD_Attribute> yang menunjukkan tipe dari atribut. Contoh: *derived*, normal,
- *isWeakEntity*, yaitu atribut pada *tag* <ERD:ERD_Entity> yang menunjukkan tipe dari entitas (*strong* dan *weak entity*), dan
- *isWeak*, yaitu atribut pada *tag* <ERD:ERD_Relationship> yang menunjukkan tipe dari relasi, yaitu *strong* atau *weak*.

Dengan mengetahui *tag-tag* tersebut, maka telah diketahui komponen-komponen apa saja (entitas, atribut, relasi, jenis entitas, jenis atribut, dan jenis relasi) yang terdapat pada sebuah ERD.

B. Representasi Detail Komponen ERD pada XMI

Informasi yang diperoleh melalui *tag-tag* tersebut masih belum lengkap. Dengan mengidentifikasi *tag* sebelumnya, nama dari setiap atribut dan keterhubungan antara entitas – atribut serta entitas – relasi masih belum secara lengkap diketahui. Contohnya, pada *tag* tersebut, hanya diberikan informasi untuk atribut dengan *base_Class*= "EAID_F71C2884_09DF_407c_B055_D5233ACCD573" tanpa memberikan informasi detail mengenai nama atribut, detail relasi dengan entitas maupun atribut, informasi *cardinality*, dan informasi lainnya yang terdapat pada ERD. Oleh karena itu, perlu diidentifikasi *tag-tag* lain yang memberikan informasi detail ini.

Dengan melakukan analisis pada XMI dokumen, maka beberapa *tag* yang menampilkan informasi detail terkait komponen pada ERD ditunjukkan pada Gbr. 7.

Berdasarkan XMI dokumen pada Gbr. 7, *tag* yang merepresentasikan detail informasi dari setiap komponen dapat dilihat pada *sub tag* pada *tag* **<elements>.** Identifikasi untuk setiap komponen menggunakan atribut *base_Class* sebagai *identifier*. Pada *tag* **<element>**, atribut *base_Class* diacu dengan menggunakan atribut **xmi:idref**. Dengan demikian, *tag* yang menampilkan detail informasi dari komponen ERD adalah sebagai berikut.

- Atribut *name* pada tag <element> menampilkan nama entitas atau nama atribut atau nama relasi.
- *Sub tag* <Association> menampilkan detail informasi relasi yang dimiliki oleh komponen tertentu, baik antara entitas – entitas atau entitas – atribut. Informasi terkait entitas maupun atribut yang terhubung ditampilkan oleh atribut *start* dan *end*.

  Jika atribut xmi:id pada *tag* <Association> memiliki *id* yang sama dengan atribut *base_Association* pada *tag* ERD:ERD_Relationship, berarti *tag* <Association> menampilkan informasi relasi antara entitas dengan entitas.

  Jika atribut xmi:id pada *tag* <Association> memiliki *id* yang sama dengan atribut *base_Association* pada *tag* ERD:ERD_Connector, berarti *tag* <Association> menampilkan informasi relasi antara entitas dengan atribut yang dimiliki entitas tersebut.

Gbr. 8 Representasi relasi pada dokumen XMI.

```
function getERDComponent (Input XMI_File: string)(Output retval: string)

function optimizedXMI (Input arrERDComponent: string)(Output retval: string)

function TreeEditDistance (Input m: integer, n:integer)(Output retval: integer)

function countSimilarityScore(Input EditDistance: integer)(Output retval: integer)

void storedComponentERD(Input arrERDComponent: string)

require: XMI_File_location

        read XMI_File
        arr_m[][]=getERDComponent()

        storedComponentERD(arr_m)

        for i=0 to arr_m.length()

                XMI_Optimzed=optimizedXMI(m)

        end for;

        editDistance = TreeEditDistance(XMI_Optimzed_1,XMI_Optimzed_2)
        score = countSimilarityScore(editDistance)

        Print score
        Print ERD_Component

end;
```

Gbr. 9 Gambaran umum modul aplikasi AssERSi.

C. Representasi Relasi dan Cardinality pada XMI

Setiap entitas memiliki keterkaitan dengan entitas yang lain yang digambarkan melalui sebuah relasi. Pada dokumen XMI sebelumnya, informasi detail antara relasi entitas dengan entitas lain belum diberikan. Relasi antara entitas akan meliputi entitas asal relasi (*source*) dan entitas tujuan relasi (*target*). Setiap relasi pasti memiliki *cardinality* yang menunjukkan jumlah maksimum objek dari entitas yang dapat berelasi dengan objek pada himpunan entitas yang lain. Berikut ini adalah *tag* pada XMI dokumen yang menggambarkan relasi dan *cardinality* pada ERD.

XMI dokumen pada Gbr. 8 menunjukkan hubungan antara entitas dengan entitas yang direpresentasikan dengan elemen *tag* **<connector>**. *Tag* **<connector>** memiliki sub elemen *tag* **<source>** dan *tag* **<target>**. *Tag* **<source>** mendefinisikan entitas sumber dan *tag* **<target>** merepresentasikan nama entitas maupun atribut yang berhubungan dengan entitas yang ada pada *tag* **<source>**.

Cardinality relasi antar entitas ditunjukkan pada setiap *tag* **<type>** dengan atribut *multiplicity* yang menjadi sub elemen dari *tag* **<source>** dan *tag* **<target>**. Sebagai contoh, entitas *STUDENT* dengan atribut *PARENT* memiliki nilai *multiplicity*=”1..*. Notasi “1..*” memiliki arti *mandatory* dan *many*, sedangkan notasi “0..1” memiliki arti *one* dan *optional*. Kemudian, *tag* **<label>** dengan atribut mt=”has” merupakan *tag* yang merepresentasikan nama relasi yang dimiliki antara *students* dengan *parents* yaitu “has”.

Struktur *tag* tersebut juga berlaku untuk menggambarkan relasi antara entitas dengan atribut. Namun, pada relasi entitas dengan atribut tidak dimiliki informasi mengenai *cardinality*.

Dengan demikian, *tag* yang menampilkan detail informasi dari relasi dan *cardinality* entitas pada ERD adalah sebagai berikut.

- *Sub tag* <source> dan <target> yang terdapat pada *tag* <connector>. *Sub tag* <model> memiliki atribut nama yang menunjukkan nama entitas maupun atribut yang terhubung.
- *Sub tag* <type> dengan atribut *multiplicity* menunjukkan *cardinality* dari relasi antara entitas, baik pada *source* maupun *target*.

- *Sub tag* <label> pada *tag* <connector> menunjukkan nama relasi antara entitas. Nama ini ditunjukan dengan atribut mt.

V. IMPLEMENTASI PERANGKAT LUNAK ASSERSI

A. Gambaran Umum Aplikasi AssERSi

Setelah dilakukan analisis XMI dokumen yang akan diperiksa kesamaannya, dilakukan perancangan dan implementasi perangkat lunak AssERSi. Salah satu perancangan yang dilakukan adalah merancang fungsi utama atau modul-modul utama yang dimiliki oleh aplikasi. Gambaran umum fungsi atau modul aplikasi ditunjukkan pada Gbr. 9.

Gbr. 9 menunjukkan bahwa terdapat lima fungsi/prosedur utama aplikasi yang melakukan pengecekan kesamaan XMI dokumen dari ERD. Kelima fungsi/prosedur tersebut adalah sebagai berikut.

1) Fungsi getERDComponent: Fungsi ini membaca dokumen XMI dan memproses (*parsing*) *tag-tag* XML untuk memperoleh semua komponen ERD. Keluaran fungsi ini adalah *array string* dari *tag* XML dan komponen ERD

2) Procedure storedComponentERD: Digunakan untuk menyimpan semua komponen ERD yang berhasil diidentifikasi untuk digunakan dalam membandingkan banyak ERD.

3) Fungsi optimizedXMI: Fungsi ini membaca keluaran dari *getERDComponent* (*array string tag* XML dan komponen ERD), kemudian mengeliminasi *tag-tag* yang terdapat pada *file* XMI tanpa mengubah struktur *tree* awal, sehingga struktur *tree* dokumen XMI lebih sederhana.

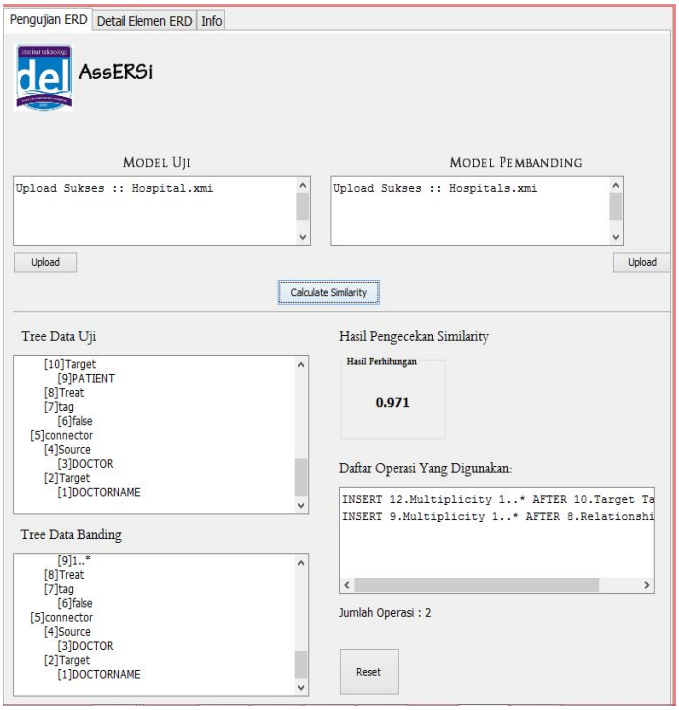
4) Fungsi TreeEditDistance: Fungsi yang membandingkan struktur *tree* dari dua dokumen XMI (XMI yang telah dioptimasi) untuk mencari nilai *editDistance* (jumlah minimal operasi yang diperlukan untuk mentransformasi struktur *tree file* xmi ke struktur *tree file* xmi yang lain)

5) Fungsi countSimilarityScore: Fungsi yang menerima masukan berupa nilai *editDistance*, kemudian menghitung *score* kesamaan berdasarkan formula yang telah ditentukan.

B. Antarmuka Aplikasi AssERSi

Antarmuka aplikasi AssERSi untuk menilai kesamaan ERD ditunjukkan pada Gbr. 10.

Gbr. 10 menunjukkan *form* utama aplikasi AssERSi untuk menerima dua masukan ERD berupa dokumen XML dan kemudian membandingkan kedua dokumen XML tersebut menggunakan algoritme *tree edit distance*. Kemudian, aplikasi menampilkan nilai kesamaan dua buah ERD tersebut. Aplikasi juga menghasilkan struktur *tree* yang diberikan oleh kedua dokumen XML setelah optimasi dan operasi (*insert*, *delete*, atau *rename*) yang dilakukan untuk mentransformasikan satu *tree* ke *tree* yang lain. Perangkat lunak AssERSi ini diimplementasikan menggunakan bahasa pemograman Java.



Gbr. 10 Form utama aplikasi AssERSi.

VI. HASIL DAN PENGUJIAN

Pengujian dan validasi aplikasi AssERSi dilakukan dengan menguji aplikasi tersebut menggunakan lima belas contoh kasus ERD yang dihasilkan dari ERD pada Gbr. 4. Hasil pengujian dibandingkan dengan hasil yang diberikan oleh aplikasi Diffnow dalam mendeteksi perbedaan dokumen XML, walaupun aplikasi Diffnow tidak memberikan nilai kesamaan terhadap dokumen XML yang dibandingkan.

TABEL I
DESKRIPSI DATA UJI

No	Deskripsi
1	Entitas yang lengkap (data uji = data banding)
2	Entitas memiliki atribut yang berbeda
3	Perbedaan nama entitas dan atribut *primary key*
4	Entitas memiliki atribut *primary key* yang berbeda
5	Nama Entitas berbeda dan *primary key multi-value*
6	Entitas memiliki nama atribut berbeda
7	Terdapat relasi *many-to-many*
8	Tipe entitas *weak entity*
9	Perubahan nama dan tipe entitas
10	Perubahan nama dan tipe entitas, serta entitas tidak memiliki *primary key*
11	Perbedaan nama entitas dan atribut yang berbeda
12	Entitas tidak memiliki *primary key*
13	Entitas tidak meiliki *primary key* dan *foreign key*
14	Nama relasi berbeda dan *cardinality* relasi yang berbeda
15	*Primary key* pada *weak entity* yang tidak lengkap

Kasus yang digunakan merupakan beberapa kasus yang paling sering ditemui dalam perancangan basis data. Tabel I menampilkan deskripsi lima belas kasus yang digunakan untuk menguji aplikasi AssERSi.

Pada pengujian ini, jumlah komponen yang berbeda antara ERD data uji dan ERD data banding dihitung secara manual untuk membandingkan setiap komponen yang berhasil dideteksi oleh aplikasi, sehingga hasil yang ditampilkan memberikan hasil yang benar. Data banding yang digunakan pada pengujian ini memiliki dua entitas, dua tipe entitas, lima atribut, lima tipe atribut, satu relasi, satu tipe relasi, dan satu *cardinality*.

Tabel II menampilkan jumlah entitas, jumlah tipe entitas, jumlah atribut, jumlah tipe atribut, jumlah relasi, jumlah tipe relasi, dan jumlah *cardinality* data uji ERD yang digunakan.

TABEL II
JUMLAH KOMPONEN ERD DATA UJI

No	Data Uji						
	E	TE	A	TA	R	TR	K
1	2	2	5	5	1	1	1
2	2	2	5	5	1	1	0
3	2	2	5	5	1	1	1
4	2	2	5	5	1	1	1
5	2	2	5	5	1	1	1
6	2	2	2	2	1	1	1
7	1	1	5	5	0	0	0
8	2	2	5	5	1	1	1
9	2	2	5	5	1	1	1
10	2	2	5	5	1	1	1
11	2	2	2	2	1	1	1
12	2	2	6	6	1	1	1
13	2	2	4	4	1	1	1
14	2	2	0	0	1	1	1
15	1	1	0	0	0	0	0

Keterangan:
E = Entitas, A = Atribut, R = Relasi, K = *Cardinality*, TE = Tipe Entitas, TA = Tipe Atribut, TR = Tipe Relasi.

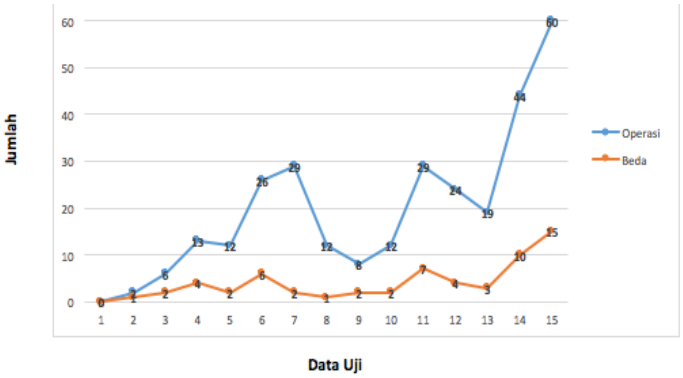
TABEL III
HASIL PENGUJIAN PADA APLIKASI ASSERSI DAN DIFFNOW

No	AssERSi			DiffNow		
	Beda	Op	Sim	Beda	Op	Sim
1	0	0	1	0	0	-
2	1	2	0,984	78	178	-
3	2	6	0,952	75	179	-
4	4	13	0,897	77	182	-
5	2	12	0,913	75	179	-
6	6	26	0,727	49	229	-
7	2	29	0,701	49	213	-
8	1	12	0,913	75	179	-
9	2	8	0,944	75	180	-
10	2	12	0,913	75	179	-
11	7	29	0,707	47	244	-
12	4	24	0,836	77	286	-
13	3	19	0,838	67	232	-
14	10	44	0,444	27	266	-
15	15	60	0,119	15	413	-

Keterangan:
- Beda = Jumlah komponen yang dideteksi berbeda
- Op = Jumlah operasi insert, delete atau renam yang dilakukan
- Sim = Nilai kesamaan ERD data uji dan data banding.

Tabel III menyajikan hasil pengujian aplikasi AssERSi dan aplikasi DiffNow. Jumlah beda komponen ERD yang diberikan oleh AssERSi sudah benar (akurat) jika dibandingkan dengan jumlah beda yang dihitung manual, sehingga nilai kesamaan yang dihasilkan sangat akurat.

Aplikasi AssERSi menghasilkan nilai kesamaan antara data uji dan data banding, sedangkan aplikasi DiffNow tidak menghasilkan nilai kesamaan. Aplikasi DiffNow hanya mampu menampilkan perbedaan antara data uji dan data banding, serta jumlah operasi yang dilakukan. Jumlah perbedaan antara data uji dan data banding pada aplikasi AssERSi lebih kecil jika dibandingkan dengan jumlah perbedaan yang dihasilkan oleh aplikasi DiffNow, karena aplikasi AssERSi melakukan optimasi pada dokumen XML dengan mengeliminasi *tag-tag* yang tidak dibutuhkan pada *tree*. Hal ini juga mempengaruhi jumlah operasi yang akan dilakukan. Tabel III menunjukkan bahwa semakin besar jumlah operasi, maka nilai kesamaan akan semakin rendah.



Gbr. 11 Perbandingan jumlah beda dan operasi pada aplikasi AssERSi.

Gbr. 11 menampilkan perbandingan jumlah beda pada data uji ERD dengan data banding ERD dan jumlah operasi yang dilakukan oleh aplikasi AssERSi untuk mentransformasi *tree* data uji ke *tree* data banding. Jumlah operasi yang dilakukan oleh aplikasi AssERSi dalam melakukan pengecekan kesamaan ERD tidak selalu berbanding lurus dengan jumlah beda. Hal ini dapat dilihat pada kasus 6 dan 7 (beda 6, operasi 26 dibandingkan dengan beda 2, operasi 29). Hal ini sangat bergantung kepada operasi yang dilakukan pada algoritme *tree edit distance*. Hasil pengujian menunjukkan jumlah beda antar ERD tidak terlalu berpengaruh apabila tidak memberikan efek yang signifikan terhadap struktur ERD. Sebagai contoh, perbedaan nama entitas tidak akan terlalu berpengaruh pada nilai kesamaan ERD.

VII. KESIMPULAN

Penilaian kesamaan ERD dengan menggunakan algoritme *tree edit distance* sangat bergantung kepada struktur *tree* ERD yang terdapat pada *file* XMI. Hasil dari algoritme *tree edit distance* adalah minimum operasi transformasi sebuah *tree* ke *tree* yang lain. Jumlah operasi yang dilakukan memengaruhi nilai kesamaan yang dihasilkan. Setiap operasi *insert*, *delete*, dan *rename* memiliki bobot yang sama untuk pemberian nilai kesamaan ERD.

Penggunaan struktur XML dan algoritme *tree edit distance* merupakan metode yang baik digunakan dalam pemberian nilai atau skor terhadap ERD serta pengecekan plagiarisme. Namun, metode ini perlu dikembangkan dengan menambahkan konsep pemeriksaan semantik ERD, yaitu struktur ERD yang berbeda dapat memiliki arti yang sama.

UCAPAN TERIMA KASIH

Terimakasih disampaikan kepada Kementerian Riset, Teknologi, dan Pendidikan Tinggi yang telah membantu pelaksanaan penelitian melalui pemberian hibah Penelitian Dosen Pemula.

REFERENSI

[1] P. Thomas, K. Waugh, and N. Smith, "Generalized diagram revision tools with automatic marking", *ITiCSE '09 Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education*, 2009, p. 318–322.

[2] H. Simanjuntak, "Proposed Framework for Automatic Entity Relationship Diagram Grading System", *ICITEE 2015 IEEE Proceedings of the 7th International Conference on information Technology and Electrical Engineering*, 2015, p. 141-146.

[3] H. Simanjuntak, D. hutajulu, E. manurung, and B. Hutagaol, "Similarity Assesment of UML Class Diagram Using Tree Edit Distance Algorithm", *Advanced Science Letters*, vol. 21 no. 11, pp. 3577-3582, Nov. 2015.

[4] (2015) W3C website. [Online]. Available: https://www.w3.org

[5] H. Kaur, M. Kaur, "Detecting Clones in Class Diagrams Using Suffix Array", *International Journal of Engineering and Advanced Technology*, vol. 3, pp. 243–246, April 2014.

[6] J. Tekli, R. Chbeir, and K. Yetongnon, "An overview on XML similarity: background, current trends and future directions", *Elsevier Computer Science Review*, vol. 3, pp. 151-173, August 2009.

[7] (2016) ExamDiff Pro Website. [Online]. Available: https://www.diffnow.com/

[8] T. Akutsu, "Tree Edit Distance Problems: Algorithm and Applications to Bioinformatics", *IEICE Transactions on Information and Systems*, vol. E93-D, pp. 208-218, February 2010.

[9] K. Zhang, D. Shasha, "Simple fast algorithms for the editing distance between trees and related problems", *SIAM Journal on Computing*, vol. 18, pp. 1245-1262, December 1989.

[10] (2016) Tree Edit Distance Website. [Online]. Available: http://tree-edit-distance.dbresearch.uni-salzburg.at/

[11] H. Edelsbrunner, Z. Gu, "Design and Analysis of Algorithms", Durham: Duke University, 2008.

[12] J. Tekli, R. Chbeir, "A novel XML document structure comparison framework based-on sub-tree commonalities and label semantics", *Journal Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 11, pp. 14-40, March 2012.