E-ISSN: 2715-2731

Analisis dan Perancangan Software Pengukur Kemiripan Desain Database Relasional

Nada Filsa Chaitra[1,*], Muhammad Ainul Yaqin[2], Rodhiyatus Sa'adah[3], Riska Dwi Anggraeni[4]

Jurusan Teknik Informatika, Universitas Islam Negeri Maulana Malik Ibrahim, Indonesia

[1]nada.chaitra@gmail.com; [2]yaqinov@ti.uin-malang.ac.id, [3]rodhiyatus1@gmail.com; [4]riskadwianggi@gmail.com
* corresponding author

INFO ARTIKEL

Sejarah Artikel
Diterima: 29 Desember 2019
Direvisi: 19 Mei 2020
Diterbitkan: 30 Desember 2020

Kata Kunci
Kemiripan Desain
Database Relational
Entity Relationship Diagram

ABSTRAK

Pengukuran kemiripan *entity relational diagram* (ERD) dilakukan untuk mendapatkan nilai kemiripan secara semantik dan struktural pada dua ERD yang dibandingkan. Kemiripan struktural didapatkan dengan mendapatkan elemen-elemen yang ada pada ERD. kemiripan semantik dilakukan dengan membandingkan setiap kata pada data *definition language* (DDL) dari ERD. Untuk mengetahui nilai kesamaan tersebut pengecekan dilakukan dengan secara manual. Hal seperti ini membutuhkan waktu yang cukup lama. Selain itu, terjadinya *human error* juga sangat mungkin terjadi. Dalam penelitian ini penulis akan menganalisa dan merancang sebuah sistem yang dapat mengukur kemiripan ERD. Ada banyak metode yang digunakan untuk menganalisa kemiripan desain basis data relasional. Untuk kemiripan struktural menggunakan metode *jaccard similarity, cosine coefficient, dice's coefficient* dan *overlap coefficient*. dan untuk kemiripan semantic digunakan *levensthein*. Hasil kemiripan ERD berupa nilai dari hasil perhitungan menggunakan metode *jaccard similarity, cosine coefficient, dice's coefficient,* serta *overlap coefficient* untuk kemiripan struktural dan metode *levensthein* untuk kemiripan semantik

PENDAHULUAN

Perkembangan teknologi semakin berkembang sehingga menjadikan pekerjaan manusia lebih mudah terutama dalam mengelola sebuah sistem informasi. Dalam sistem informasi terdapat salah satu komponen yang sangat penting yaitu basis data. Basis data merupakan salah satu komponen dalam sebuah sistem informasi yang berfungsi sebagai tempat penyimpanan data [1] .Basis data berfungsi sebagai penyimpanan data. Perangkat lunak yang digunakan untuk mengelola dan memanggil perintah (*Query*) basis data disebut sistem management basis data (*Database Management System*, DBMS). Untuk menyimpan data di dalam sebuah basis data, pembuatan diagram akan dilakukan terlebih dahulu. Diagram yang digunakan ialah *Entity Relational Diagram* (ERD). ERD digunakan untuk menggambarkan struktur *logical* database dalam bentuk diagram. ERD akan dibuat dalam format *Conceptual Data Model* dan *Physical Data Model* [2].

Setiap ERD satu dengan ERD lainnya memiliki nilai dan isi yang berbeda-beda. Tetapi terkadang terdapat beberapa ERD yang memiliki kesamaan. Untuk memeriksa kesamaan ERD masih dilakukan secara manual. Jika pemberian nilai pada ERD yang dibandingkan lebih dari dua ERD, maka akan tinggi kemungkinan terjadinya kesalahan atau *human error*. Selain itu penilaian secara manual ini membutuhkan waktu yang cukup lama. Dari permasalahan tersebut, maka perlu adanya sebuah sistem yang dapat mencari kesamaan dari beberapa ERD dengan metode yang digunakannya. Setelah dicari nilai dari kemiripannya satu persatu, diharapkan muncul nilai akhir kesamaan dari beberapa ERD yang telah diukur

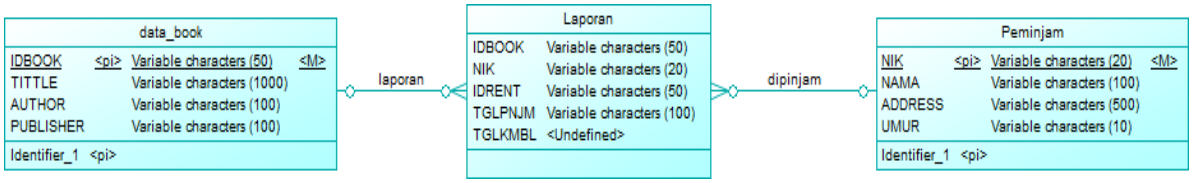
tersebut. Pencarian nilai kesamaan dengan cara menghitung kemiripan desain database relasional menggunakan beberapa proses. Tahap yang dilakukan untuk memenuhi tujuan paper ini adalah representasi ERD. Pada representasi ERD, kami menggunakan metode *Levenshtein Distance* untuk mengukur kemiripan secara semantik, dan *jaccard similarity, cosine coefficient, dice's coefficient*, *overlap coefficient* untuk kemiripan secara struktural. Paper ini mengidentifikasi bahwa antara database satu dengan database lainnya terdapat kemiripan yang dapat diukur melalui atribut-atribut yang digunakan, *key*, dan relasinya. Jika sistem ini dibuat, sistem akan memudahkan seseorang untuk mencari kesamaan pada sebuah ERD. Pencarian kesamaan semakin mudah dengan sistem ini karena dapat menghemat waktu dan mengurangi beberapa kesalahan yang terjadi dalam pencarian kesamaan.

METODE

Metode yang Prosedur penelitian ini menggunakan 2 tahap, yaitu permodelan database ke ERD, kemudian menghitung kemiripan desain. Berikut penjelasan detailnya:

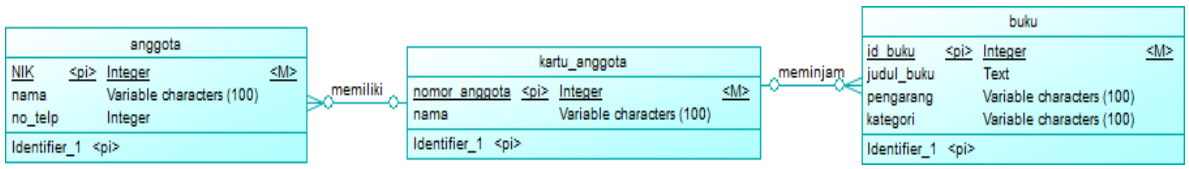
Permodelan Database Relational ke ERD

Pada tahap ini dilakukan permodelan *entity relational diagram* (ERD) dengan CDM. Database dimodelkan sehingga didapatkan model seperti pada Gambar 1 hingga Gambar 3.



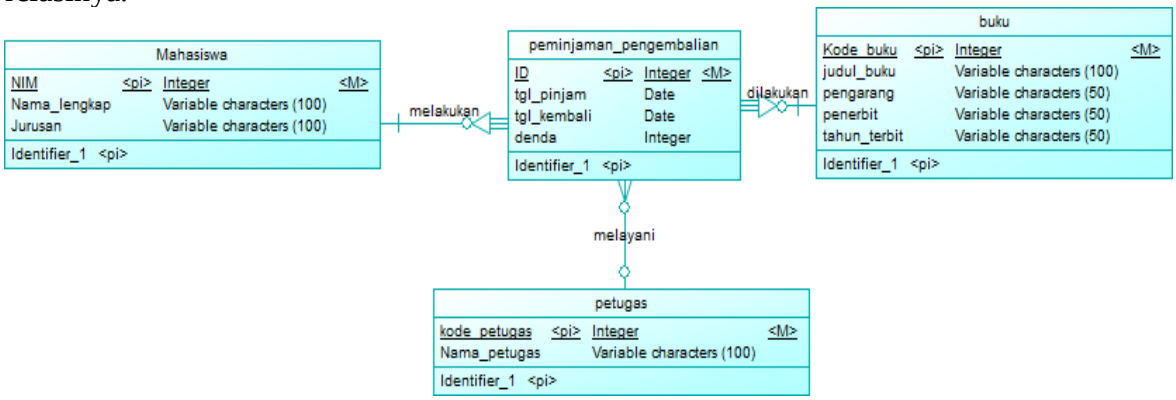
Gambar 1 ERD pemijaman buku model A

Gambar 1 menunjukkan ERD peminjaman buku model A dengan entitas, atribut, dan relasinya.



Gambar 2 ERD peminjaman buku model B

Gambar 2 menunjukkan ERD peminjaman buku model B dengan entitas, atribut, dan relasinya.



Gambar 3 ERD peminjaman buku model C

Gambar 2 menunjukkan ERD peminjaman buku model B dengan entitas, atribut, dan relasinya.

Pengubahan ERD ke SQL

Pada penelitian ini, struktur pada ERD yang digunakan berupa format SQL. Struktur SQL diilustrasikan melalui *software power designer*. Setelah suatu ERD telah dihasilkan, *software power designer* memungkinkan kita untuk mengubah struktur ERD ke dalam format SQL. Sebelum diubah menjadi SQL, ERD diubah ke format *.pdm terlebih dahulu. Setelah terbentuk format *pdm, maka dapat dikonversikan ke bentuk*sql. Format yang dihasilkan adalah *.sql.

Perhitungan Kemiripan ERD

Setelah didapatkan model ERD dan hasil *parsing* ke SQL, kegiatan selanjutnya adalah menghitung kemiripan yang terbentuk berdasarkan ERD tersebut. Perhitungan kemiripan ERD dilakukan dalam dua aspek yaitu kemiripan struktural dan semantik. Kemiripan struktural dapat dilakukan dengan cara sebagai berikut.
ERD Gambar 1 = E1,E2,E3,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,E1-E2,E2-E3
Dari sini ERD peminjaman buku model A 18 elemen.
Dimana:
E1-E3= entitas 1hingga entitas3
a1-13 = atribut 1 sampai atribut 13
E1-E2 = relasi entitas 1 dan 2
E2-E3 = relasi entitas 2 dan 3
ERD Gambar 2 = E1,E2,E3,a1,a2,a3,a4,a5,a6,,a7,a8,a9,E1-E2,E2-E3
Dari sini ERD peminjaman buku model B terdapat 9 elemen.
Dimana :
E1,E2,E3= entitas 1, entitas 2 entitas3
a1-9= atribut 1 sampai atribut 9
E1-E2 = relasi entitas 1 dan 2
E2-E3 = relasi entitas 2 dan 3
ERD Gambar 2 = E1,E2,E3,E4,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,E1-E2,E2-E3,E2-E4
Dari sini ERD ERD peminjaman buku model C terdapat 21 elemen.
E1-E4= entitas 1 hingga entitas 2
a1-14= atribut 1 sampai atribut 14
E1-E2 = relasi entitas 1 dan 2
E2-E3= relasi entitas 2 dan 3
E2-E4 = relasi entitas 2 dan 4
Kemudian angka-anngka tersebut dimasukkan ke dalam formula perhitungan kemiripan menggunakan metode *jaccard similarity, cosine coefficient, Dice's coefficient,* dan *overlap coefficient*. Berikut perhitungan kemiripannnya.

Jaccard similarity

Jaccard Similarity atau *Jaccard Coefficient* merupakan algoritma yang fungsinya untuk membandingkan dua *sample* yaitu dokumen yang satu dengan yang lainnya berdasarkan kata yang dimilikinya [2]. Persamaan 1 menunjukkan rumus *jaccard similarity*.

$$jaccard\ (A,B) = \frac{|\mathrm{A} \cap \mathrm{B}|}{|A| + |B| - |A \cap B|} \quad (1)$$

Dimana |A ∩ B| adalah jumlah elemen yang sama pada data A dan B, |A| adalah jumlah elemen pada data A dan |B| adalah jumlah elemen pada data B.

Tabel 1 perhitungan kemiripan dengan jaccard similarity

ERD	**ERD A**	**ERD B**	**ERD C**
ERD A	**0**	0.5	0.85
ERD B	0.5	**0**	0.42
ERD C	0.85	0.42	**0**

Tabel 1 menunjukkan pengukuran kemiripan menggunakan *jaccard similarity* dari ERD A dengan ERD B sebesar 0.5, ERD A dengan ERD C sebesar 0.85 dan ERD B dengan C sebesar 0.42.

Cosine coefficient

Cosine Similarity merupakan metode yang digunakan untuk menghitung tingkat kesamaan(*similarity*) antar dua objek [3]. Ditunjukkan pada Persamaan 2

$$cos\theta = \frac{|A \cap B|}{|A|^{1/2}|B|^{1/2}} \quad (2)$$

Dimana |A ∩ B| adalah jumlah elemen yang sama pada data A dan B, $|A|^{1/2}$adalah akar dua dari jumlah elemen pada data A dan $|B|^{1/2}$ adalah akar dua dari jumlah elemen pada data B.

Tabel 2 perhitungan persamaan menggunakan cosine coefficient

ERD	**ERD A**	**ERD B**	**ERD C**
ERD A	**0**	0.71	0.92
ERD B	0.71	**0**	0.65
ERD C	0.92	0.65	**0**

Tabel 2 menunjukkan pengukuran kemiripan menggunakan *cosine coefficient* dari ERD A dengan ERD B sebesar 0.71, ERD A dengan ERD C sebesar 0.92 dan ERD B dengan C sebesar 0.65.

Dice's coefficient

$$dice(A,B) = 2 * \frac{|A \cap B|}{|A| + |B|} \quad (3)$$

Untuk kemiripan *dice coefficient*, $2|A \cap B|$ adalah jumlah elemen yang sama pada data A dan B dikalikan dengan 2. |A| adalah jumlah elemen pada data A dan |B| adalah jumlah elemen pada data B. Ditunjukkan oleh Persamaan 3.

Tabel 3 perhitungan menggonakan dice coefficient

ERD	**ERD A**	**ERD B**	**ERD C**
ERD A	**0**	0.67	0.92
ERD B	0.67	**0**	0.6
ERD C	0.92	0.6	**0**

Tabel 3 menunjukkan pengukuran kemiripan menggunakan *dice coefficient* dari ERD A dengan ERD B sebesar 0.67, ERD A dengan ERD C sebesar 0.92 dan ERD B dengan C sebesar 0.6.

Overlap coefficient

Overlap coefficient juga disebut sebagai *Szymkiewicz-Simpson coefficient* yang mana pengukuran kemiripannya masih terkait dengan indeks *jaccard* [4].

$$O(A,B) = \frac{|A \cap B|}{\min(|A|,|B|)} \quad (4)$$

Untuk kemiripan *overlap coefficient*, $|A \cap B|$ adalah jumlah elemen yang sama pada data A dan B. $\min(|A|,|B|)$ adalah jumlah minimum dari data A dan B. Ditunjukkan oleh Persamaan 4.

Tabel 4 perhitungan menggunakan metode overlap coefficient

ERD	**ERD A**	**ERD B**	**ERD C**
ERD A	**0**	1	1
ERD B	1	**0**	1
ERD C	1	1	**0**

Tabel 4 menunjukkan pengukuran kemiripan menggunakan *jaccard similarity* dari ERD A dengan ERD B sebesar 1, ERD A dengan ERD C sebesar 1 dan ERD B dengan C sebesar 1. Perhitungan kemiripan semantik dilakukan dengan membandingkan DDL pada ERD satu dengan ERD lainnya. Setiap kata pada DDL tersebut akan dihitung kecocokan semanticnya menggunakan metode *Levensthein Distance*. *Levenshtein distance* adalah sebuah matriks *string* yang digunakan untuk mengukur perbedaan atau jarak (*distance*) antara dua *string*. Nilai *distance* antara dua *string* ini ditentukan oleh jumlah minimum dari operasi-operasi perubahan yang diperlukan untuk melakukan transformasi dari suatu *string* menjadi *string* lainnya[5].Setelah dilakukan perhitungan *Levensthein Distance*, akan di dapat nilai *distace* dari setiap kata yang dibandingkan. Berikut merupakan tabel perhitungan kemiripan semantikk dengan metode *levensthein.*

Tabel 5 Perhitungan Levensthein Distance

ERD	ERD 1	ERD 2	ERD 3
ERD 1	**0**	103	74
ERD 2	103	**0**	48
ERD 3	74	48	**0**

Tabel 5 menunjukkan besar *distance* dari ERD 1 dengan ERD 2 sebesar 82, ERD 1 dengan 3 sebesar 74 dan ERD 2 dengan 3 sebesar 48. *Levenshtein distance* melakukan perhitungan bobot *similarity* setelah mendapatkan nilai *distance* dari dua dokumen yang dibandingkan. Kemudian menggunakan suatu persamaan dalam menentukan bobot *similarity*[5].

$$Bobot\ similarity = 1 - \frac{d[m,n]}{\max(S,T)} * 100\% \quad 5$$

Diketahui pada Persamaan 5 bahwa:
d[m,n] = nilai distance pada baris ke-m dan kolom ke-n.
S = Panjang string awal.
T= panjang string target.
Max(S,T) = adalah panjang String terbesar dari string awal dan string target
Berikut adalah hasil dari pembototan dari distance yang telah dicari.

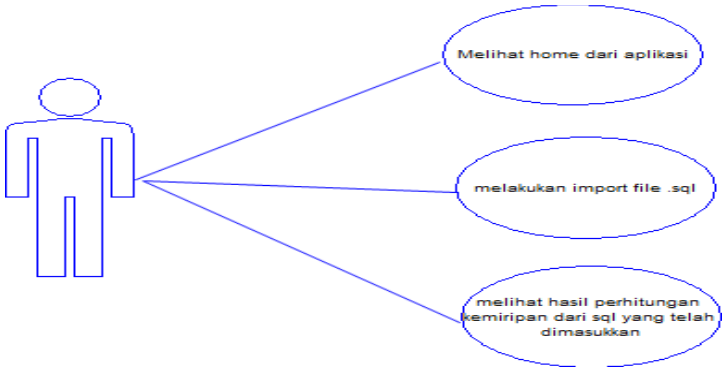
Tabel 6 hasil perhitungan bobot kemiripan

ERD	ERD A	ERD B	ERD C
ERD A	**0**	0,35	0,21
ERD B	0,35	**0**	0,33
ERD C	0,21	0,33	**0**

Tabel 1 menunjukkan pengukuran kemiripan semantik menggunakan *Levensthein similarity* dari ERD A dengan ERD B sebesar 0.35, ERD A dengan ERD C sebesar 0.21 dan ERD B dengan C sebesar 0.33.

HASIL DAN PEMBAHASAN

Pemetaan Use Case dan Diagram Alur ERDchecker



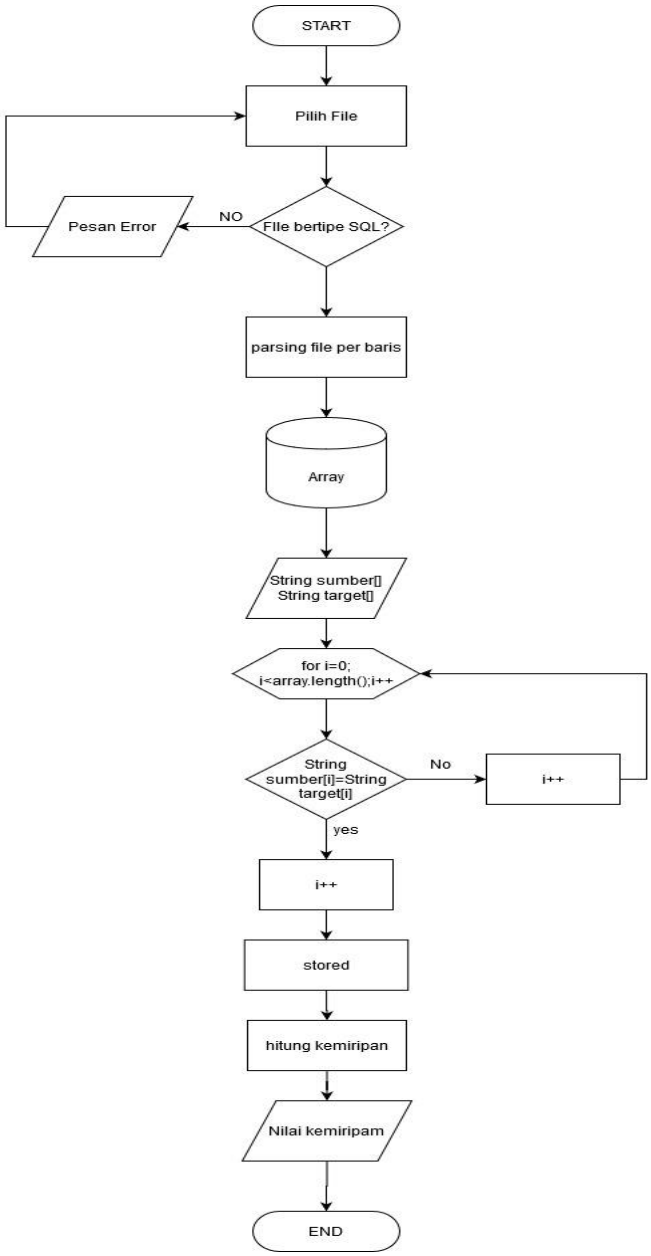
Gambar 4. Use Case Sistem

Gambar 4 menjelaskan tentang *use case* dari sistem. *Use case* tersebut kemudian dijelaskan pada Tabel 7.

Tabel 7. Penjelasan Use Case

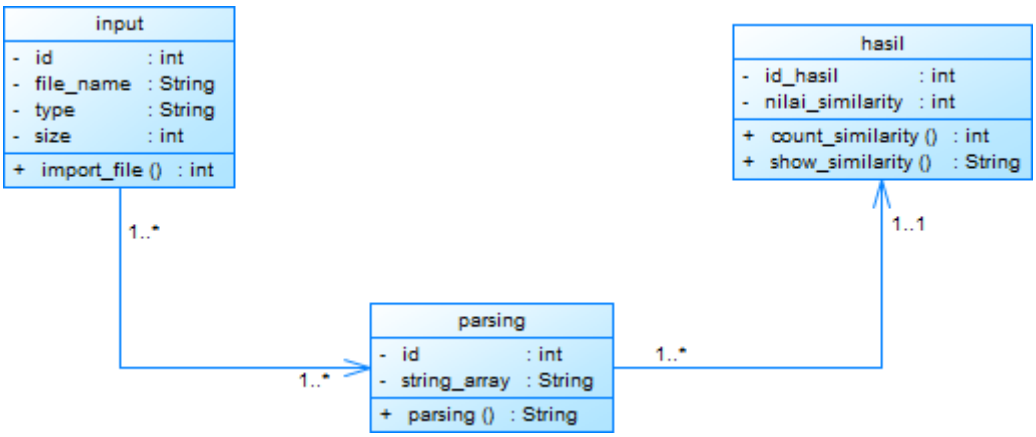
No	Use case	Deskripsi
1	Melihat halaman *home* dari aplikasi	*User* dapat melihat halaman *home* aplikasi ERDcheker
2	Melakukan *import* dokumen .sql	User dapat menginputkan dokumen .sql yang akan dihtung
3	Melihat hasil perhitungan dari dokumen yang dimasukkan	User dapat melihat nilai kemiripan dari dokumen yan telah di-*import*-kan

Flowchart **(diagram alur)**



Gambar 5. Diagram alur sistem

Gambar 5 menunjukkan *flowchart* dari sistem.



Gambar 5. class diagram sistem

Gambar 6 menunjukkan class diagram dari sistem. Adapun alur prosesnya adalah pertama *user* diminta untuk meng-*import* dokumen SQL yang akan dihitung kemiripannya. Kemudian, sistem akan mengecek apakah dokumen yang di-*import* merupakan dokumen berformat .sql atau bukan. Jika dokumen yang di-import bukan dokumen dengan format .sql, maka sistem akan menampilkan sebuah peringatan bahwa dokumen yang di-*import* bukan merupakan dokumen yang memiliki format .sql. Sedangkan jika dokumen yang di-*import* merupakan dokumen berformat .sql, maka sistem akan melanjutkan proses. Selanjutnya, dokumen .sql yang sudah di-*import*, akan di *parsing* pada setiap baris dan disimpan pada sebuah *ArrayList*. Dengan menggunakan *looping* sebanyak *ArrayList*, akan dilakukan perbandingan antara *string* target dengan *string* sumber untuk mencari kemiripan hingga *looping* memenuhi syarat. Kemudian hasil pada perbandingan tersebut akan disimpan, dan dilakukan perhitungan kemiripan hingga di dapatkan nilai kemirinnya.

Gambaran Umum Aplikasi ERDchecker

```
Function get ERDcomponent (input SQL_file : string)(output : String)
Function optimasisasi : (input :string) (output: array string)
Function comparing(input : array string)(output: integer)
Function countsimilarity(input comparing : integer)(output: integer)
Void storedcomponentERD(input array string)
Require : SQL_file
        Read SQL_file
        Array_a[]=getERDcomponent()
        Array_b[]=getERDcomponent()
        storedcomponentERD(Array_a,Arrray_b)
        for i=0 to arr_a.length()
                compare= comparing(Array_a[i],Arrayb[i])
        end for;
score = countsimilarity(compare)
print score
end;
```

Gambar 6. Gambaran umum fungsi sistem

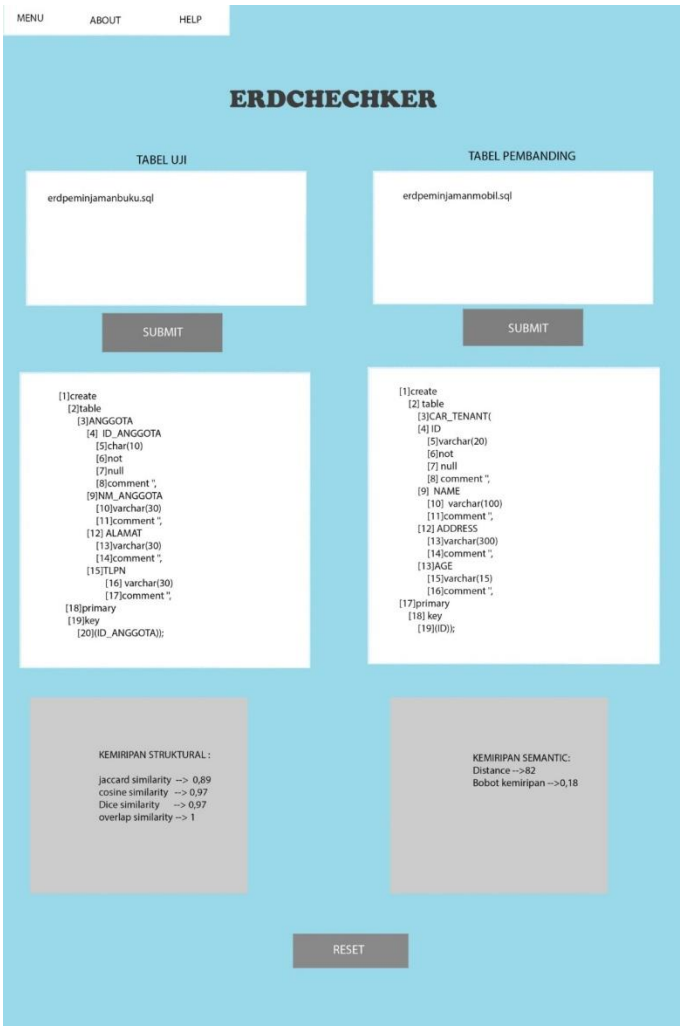
Setelah dilakukan analisis SQL dokumen yang telah diperiksa, dilakukan perancangan dan implementasi perangkat lunak ERDchecker. Salah satu perancangan yang dilakukan adalah menentukan menu-menu utama yang ada di aplikasi tersebut. Gambaran umum fungsi dan menu aplikasi ditunjukkan pada Gambar 7. Gambar 7 menunjukkan fungsi utama aplikasi yang akan melakukan pengecekan kemiripan secara structural dan semantic. Beberapa fungsi pada aplikasi adalah sebagai berikut

1. *Function getERDcomponent* : fungsi ini akan melakukan pembacaan dokumen sql dengan dilakukan import pada document sql. Dokumen sql akan diparsing untuk memperoleh DDL yang menggambarkan komponen dari ERD. Keluaran dari fungsi ini berupa array string DDL dan komponen ERD.
2. *Function storedcomponentERD*: komponen ERD yang telah diparsing, dalam bentuk array string akan disimpan dan dibandingkan dengan ERD pembanding.
3. *Function comparing*: setiap kata dan struktur pada DDL sumber akan dibandingkan dengan ERD target. Untuk setiap kata akan dicari levenshtein distance-nya. Dan untuk struktur ERD akan dihitung kemiripan elemen-elemen yang yang ada.
4. *Function countsimilarity*: setelah dibandingkan pada tahap sebelumnya, hasil keluaran yang telah didapat dihitung kemiripannya. Keluaran dari fungsi ini berupa nilai kemiripan dari ERD yang telah dihitung.
5. *Printscore* : digunakan untuk mencetak hasil perhitungan kemiripan yang telah dilakukan

Untuk melakukan pengukuran diperlukan *input* dokumen sql. setelah dinputkan dilakukan fungsi *getERDcomponent()*. Hasil *output* dari *getERDcomponent()* akan dilakukan fungsi *storedcomponentERD()* dimana data akan disimpan untuk dibandingkan. Data akan dibandingkan dengan fungsi *comparing*. Hasil dari *comparing* akan dihitung kemiripan dengan fungsi *countsimilarity* dan nilai kemiripan dapat dicetak menggunakan fungsi *printscore.*

Interface Aplikasi ERDChecker

Interface aplikasi pengukur kemiripan ERD ditunjukkan pada Gambar 8. Pada Gambar 8 dapat dilihat terdapat dua *textfield* untuk memasukkan dokumen sql. Pada dua *textfield* di baris kedua menunjukkan isi kedua DDL dari dokumen sql yang telah berhasil dimasukkan dan di-*parsing* berupa array. Array *string* tersebut akan dibandingkan lalu dihitung kemiripannya. Hasil kemiripan ditunjukkan pada dua *textfield* pada baris terakhir.



Gambar 7. Gambaran Sistem

KESIMPULAN

Berdasarkan hasil penelitian Analasis dan Perancangan Software Pengukur Kemiripan Desain Database Relasional dapat diambil kesimpulan bahwa kemiripan suatu ERD dapat dipengaruhi oleh DDL penyusunnya dan susunan struktural komponen ERD. Kemiripan yang dihasilkan dalam perancangan software ini berupa nilai kemiripan semantik dan struktural. Untuk perhitungan secara struktural, metode *overlap coefficient* dinilai kurang untuk mendapatkan nilai kemiripan. Sedangkan untuk pengukuran kemiripan semantic menggunakan metode *Levensthein* sudah didapatkan hasil yang cukup baik. Saran yang tepat untuk menyempurnakan program Analisis dan Perancangan Software Pengukur Kemiripan Desain Database Relasional yaitu bagi pembaca dapat melakukan penelitian lebih lanjut tentang pengukuran kemiripan ERD dengan metode *jaccard,cosine,dice,overlap*, dan *levensthein distance*.dan epneliti lanjutan dapat melakukan evaluasi program yang telah dibuat agar lebih disempurnakan dan dapat digunakan secara maksimal.

REFERENSI

[1] H. Simanjuntak, R. Lumbantoruan, W. Banjarnahor, E. Sitorus, M. Panjaitan, and S. Panjaitan, "Penilaian Kesamaan Entity Relationship Diagram dengan Algoritme Tree Edit Distance," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 6, no. 1, 2017, doi: 10.22146/jnteti.v6i1.289.

[2] S. Sunardi, A. Yudhana, and I. A. Mukaromah, "Implementasi Deteksi Plagiarisme Menggunakan

Metode N-Gram Dan Jaccard Similarity Terhadap Algoritma Winnowing," *Transmisi*, vol. 20, no. 3, p. 105, 2018, doi: 10.14710/transmisi.20.3.105-110.

[3] S. Sugiyamto, B. Surarso, and A. Sugiharto, "Analisa Performa Metode Cosine Dan Jacard Pada Pengujian Kesamaan Dokumen," *J. Masy. Inform.*, vol. 5, no. 10, pp. 1–8, 2014, doi: 10.14710/jmasif.5.10.1-8.

[4] V. M.K and K. K, "A Survey on Similarity Measures in Text Mining," *Mach. Learn. Appl. An Int. J.*, vol. 3, no. 1, pp. 19–28, 2016, doi: 10.5121/mlaij.2016.3103.

[5] B. P. Pratama and S. A. Pamungkas, "Analisis Kinerja Algoritma Levenshtein Distance dalam Mendeteksi Kemiripan Dokumen Teks," *J. "LOG!K@,"* vol. 6, no. 2, pp. 131–143, 2016.