3 ANALISIS DAN DESAIN SISTEM

3.1 Analisis Masalah

Saat ini, proses *mapping* Rodex Tours & Travel dilakukan dengan membandingkan *similarity* masing-masing *field* antar data vendor dengan data *master.* Data *master* yang dibandingkan difilter terlebih dahulu berdasarkan *field* City dan Destination yang sama dengan data vendor. Hal ini dilakukan data vendor tidak dibandingkan dengan setiap *record* dari data *master* karena akan membuat proses *mapping* menjadi lama. Ketika sudah didapatkan data *master* dengan Destination yang sama, akan dilakukan proses perbandingkan antar *field* dengan data vendor.

Masing-masing *field* ini mempunyai bobot atau skor maksimal berbeda. Untuk *field* Star, City, Destination, dan Country skor maksimalnya adalah 1 dengan nilai 0 atau 1. Jika hanya ada 1 karakter yang berbeda maka akan otomatis dianggap sebagai data yang berbeda atau tidak *match*. Untuk *field* Name dan Address #1 skor maksimalnya adalah 5 dengan *range* nilai 0-5. Semakin besar nilai yang dihasilkan mengindikasikan nilai kemiripan yang besar pula. Untuk *field* Longitude dan Latitude, skor maksimalnya adalah 1 dengan *range* nilai 0-1.

Nilai kemiripan ini ditentukan dari jumlah karakter yang sama dari nilai yang dibandingkan. Terdapat beberapa asumsi yang digunakan untuk menghitung skor kemiripan masing-masing *field.* Untuk Longitude dan Latitude, dilakukan pengecekan terhadap karakter pertama, apabila tidak sama maka otomatis dianggap 0 atau tidak *match*. Apabila karakter pertama sama, dilakukan pengecekan untuk 8 karakter di belakang koma. Apabila 8 karakter tersebut semuanya *match*, maka skor *similarity*-nya akan maksimal (5). Untuk *field* Name dan Address #1, terdapat beberapa kata-kata yang tidak dimasukkan dalam perhitungan skor *similarity* seperti “Hotel”, “Jl.”, “Jalan”, dan nama kota dari hotel terkait. Untuk penentuan kata-kata yang diabaikan ini dilakukan secara manual.

Nantinya skor per *field* ini akan dijumlahkan dan dibagi dengan jumlah dari skor maksimal untuk menghasilkan skor total *similarity*-nya. Apabila skor akhirnya di atas 70, maka akan dianggap *match,* dan jika skor nya di bawah 70 akan dicek secara manual untuk menentukan apakah kedua data yang dibandingkan *match.* Namun jika skor akhirnya di bawah 20, maka otomatis akan dianggap tidak sesuai dan di-*reject*. Apabila terdapat lebih dari 1 data yang memiliki skor di atas 70, maka akan diambil data dengan skor yang tertinggi sebagai target *mapping*.

Data vendor yang sudah *match* dengan data *master* kemudian akan di-*merge* terhadap data *master* tersebut. Dalam proses *merge* ini juga memungkinkan adanya nilai dari beberapa *field* data *master* yang perlu di-*override* dengan data vendor. Proses *override* ini bertujuan untuk meng-*update* data *master* dengan data yang lebih detail dan *update*. Penentuan *field* yang akan di-*override* dilakukan berdasarkan perbandingan antar *field* yang dilakukan sebelumnya. Apabila *field* dari data vendor memiliki panjang String yang lebih besar dibandingkan data *master*, maka nilai dari *field* tersebut di data *master* akan di-*override* oleh data vendor.

Proses *mapping* data vendor hanya akan menghasilkan 1 data *master* yang sesuai. Apabila data *master* yang menjadi target *mapping* data vendor lebih dari 1, maka akan menyebabkan kesalahan *mapping*, yang mempengaruhi data yang ditampilkan di *website* yang disajikan ke *user*. Proses *mapping* ini memerlukan banyak operasi manual yang perlu dilakukan seperti menentukan kata yang akan diabaikan ketika membandingkan nama hotel, filter data *master* sesuai Destination dan City dari data vendor, serta mengecek hasil *mapping*. Selain itu, sistem *mapping* saat ini hanya melakukan perbandingan antar String, sehingga apabila terdapat perbedaan karakter yang signifikan, maka tentunya akan menghasilkan skor *similarity* yang rendah walaupun sebenarnya kedua data merujuk ke satu identitas yang sama.

3.2 Analisis Kebutuhan

Berdasarkan analisa masalah yang dijabarkan sebelumnya, Rodex Tours & Travel memerlukan pendekatan baru untuk menghasilkan skor *similarity.* Skor *similarity* ini merupakan faktor yang sangat krusial dalam proses *mapping* karena akan menentukan data hotel mana saja yang merujuk ke satu identitas yang sama. Dengan menggunakan *graph database*, skor *similarity* yang dihasilkan tidak hanya berdasarkan String, namun juga dapat memperhitungkan *relation* antar *node* yang ada.

3.3 Dataset

Data yang digunakan pada skripsi ini adalah data hotel yang dimiliki oleh Rodex Tours & Travel. Data ini terdiri dari 1 *file csv* dengan jumlah 4111 data. Data diperoleh dari *file* excel/csv yang diberikan vendor atau *request* API oleh Rodex Tours & Travel kepada vendor. Terdapat 9 vendor dengan jumlah data hotel per vendor yang berbeda-beda. Vendor A8 memiliki 1266 data hotel, A14 sebanyak 296 data, A6 sebanyak 672 data, A1 sebanyak 172 data, A5 sebanyak 138 data, A9 sebanyak 659 data, A4 203 data, A2 sebanyak 389 data, dan A3 sebanyak 315 data. Data yang ada dari berbagai vendor ini dapat memiliki data identitas hotel yang berbeda-beda

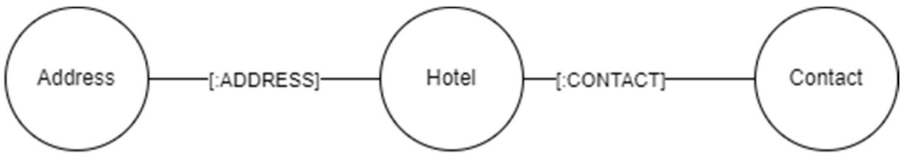
namun sebenarnya merujuk ke 1 entitas hotel yang sama. Untuk pengerjaan skripsi ini akan menggunakan vendor dengan jumlah data terbanyak yaitu A8 sebagai data *master,* dan vendor lainnya sebagai data vendor yang akan di-*mapping.* Daftar *field* untuk data ini dapat dilihat pada Tabel 3.1

Tabel 3.1

Field dari Data Hotel

Nama Field	Keterangan
ID	Kode unik hotel
Name	Nama hotel
Destination	Destinasi wisata, kecamatan atau nama kota dari hotel
City/Name	Kota tempat hotel berada
Country/Country Name	Negara tempat hotel berada
Star	*Rating* atau bintang dari suatu hotel
Provider	*Vendor* yang memiliki data hotel
Address #1	Alamat pertama dari suatu hotel (berupa nama jalan)
Address #2	Alamat kedua dari suatu hotel
Address #3	Alamat ketiga dari suatu hotel (berupa nama kecamatan)
Email	Kontak *email* hotel
Fax	Kontak faksimile hotel
Latitude	Koordinat *latitude* dari hotel
Longitude	Koordinat *longitude* dari hotel
Phone	Kontak nomor telepon hotel

3.4 Desain Schema Graph



Gambar 3.1 *Schema Graph*

Gambar 3.1 merupakan *schema* awal yang akan digunakan dalam *graph database*. Terdapat 3 tipe *node* yang dilambangkan dengan simbol lingkaran, serta 2 *relationship* yang dilambangkan dengan garis *solid*.

3.4.1 Node Hotel

Node Hotel merupakan *node* yang menyimpan nama dan bintang dari suatu hotel. *Node* hotel memiliki dua *property* utama, yaitu name dan star.

Tabel 3.2

Node Hotel

Nama Property	Keterangan
name	Nama hotel (dari *field* Name)
star	Bintang hotel (dari *field* Star_Fixed
alt_names	*List* yang menyimpan alternatif name dari suatu hotel.

3.4.2 Node Address

Node Address menyimpan data terkait alamat dari suatu hotel. Dalam beberapa kasus, terdapat lebih dari satu hotel yang memiliki alamat yang sama, sehingga memungkinkan untuk beberapa *node* hotel merujuk ke satu *node* Address yang sama.

Tabel 3.3

Node Address

Nama Property	Keterangan

address_detail	JSON yang menyimpan country, city, address, dan longlat sebagai nilainya. Address#1, Address#2, Address#3, dan Destination digabungkan menjadi satu String dan disimpan dalam *key* address. Longitude dan Latitude dijadikan dalam satu *list* dengan *key* longlat. Contoh:
	{
	country: Singapore,
	city: Singapore,
	address: 366 Orchard Road Singapore,
	longlat: [103831422,1305951]
	}

alt_address	*List* yang menyimpan alternatif address_detail dari suatu hotel.

3.4.3 Node Contact

Node Contact merupakan *node* yang menyimpan kontak dari suatu hotel. *Node* hotel memiliki dua *property* utama, yaitu phone dan email.

Tabel 3.4

Node Contact

Nama Property	Keterangan
email	Email dari suatu hotel
phone	Nomor telepon dari suatu hotel
alt_phone	*List* yang menyimpan alternatif nomor telepon dari suatu hotel.
alt_email	*List* yang menyimpan alternatif email dari suatu hotel.

3.4.4 Relationship

Terdapat dua *relationship* dari *schema* ini, yaitu ADDRESS dan CONTACT.

Tabel 3.5

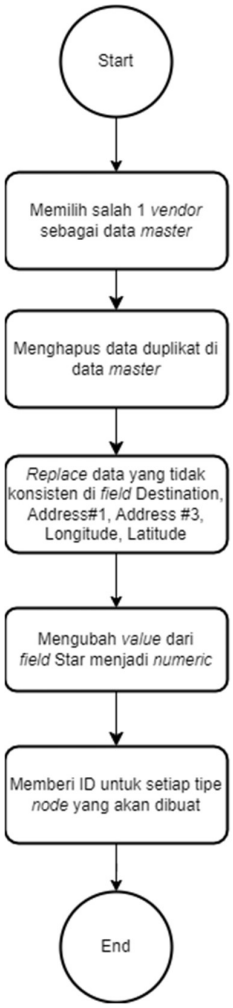
Relationship

Nama Relationship	Keterangan
ADDRESS	Menghubungkan *node* Address dan Hotel
CONTACT	Menghubungkan *node* Hotel dan Contact

3.5 Desain Sistem

3.5.1 Preprocessing Data

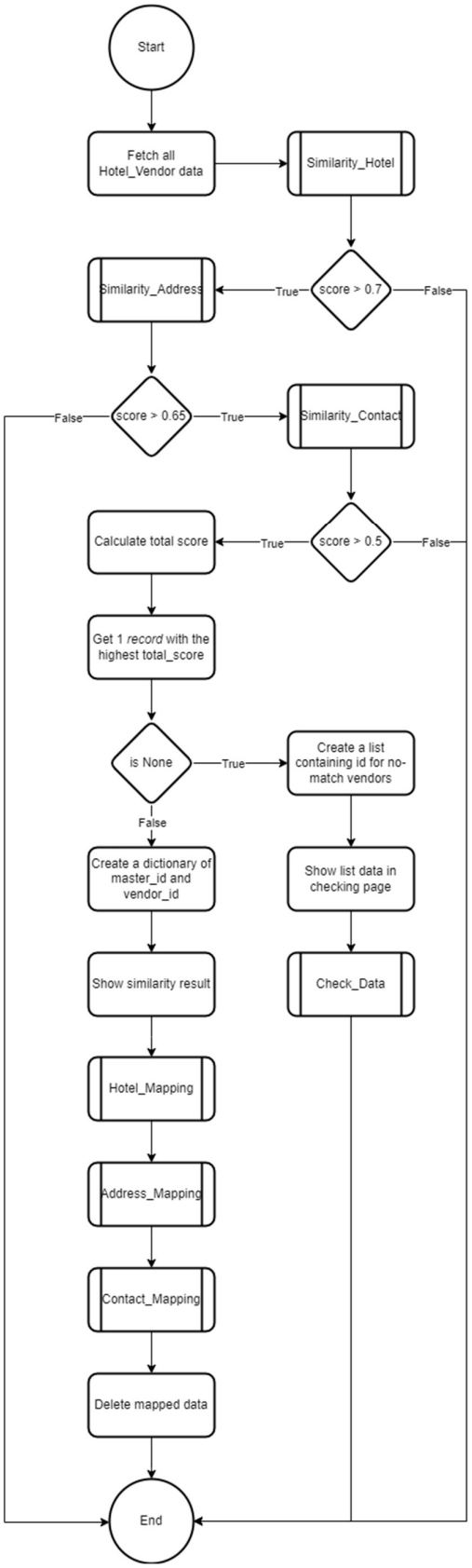
Sebelum di-*import* ke Neo4j, akan dilakukan beberapa proses sebagai terhadap data *master* hotel. Proses ini dilakukan agar nilai dari *initial data master* yang digunakan konsisten.



Gambar 3.2 *Preprocessing* Data

3.5.2 Proses *Similarity Search* dan *Mapping*

Setelah data master dan data vendor di-*import* ke dalam *database*, akan dilakukan dua proses, yaitu pencarian data master yang memiliki skor *similarity* tertinggi untuk setiap data vendor dan *mapping* untuk data vendor tersebut terhadap data master terkait. Alur untuk melakukan kedua proses tersebut dapat dilihat pada Gambar 3.3.


Start
Fetch all Hotel_Vendor data
Similarity_Hotel
score > 0.7
True
False
Similarity_Address
score > 0.65
False
True
Similarity_Contact
Calculate total score
True
score > 0.5
False
Get 1 record with the highest total_score
is None
True
Create a list containing id for no-match vendors
False
Create a dictionary of master_id and vendor_id
Show list data in checking page
Show similarity result
Check_Data
Hotel_Mapping
Address_Mapping
Contact_Mapping
Delete mapped data
End

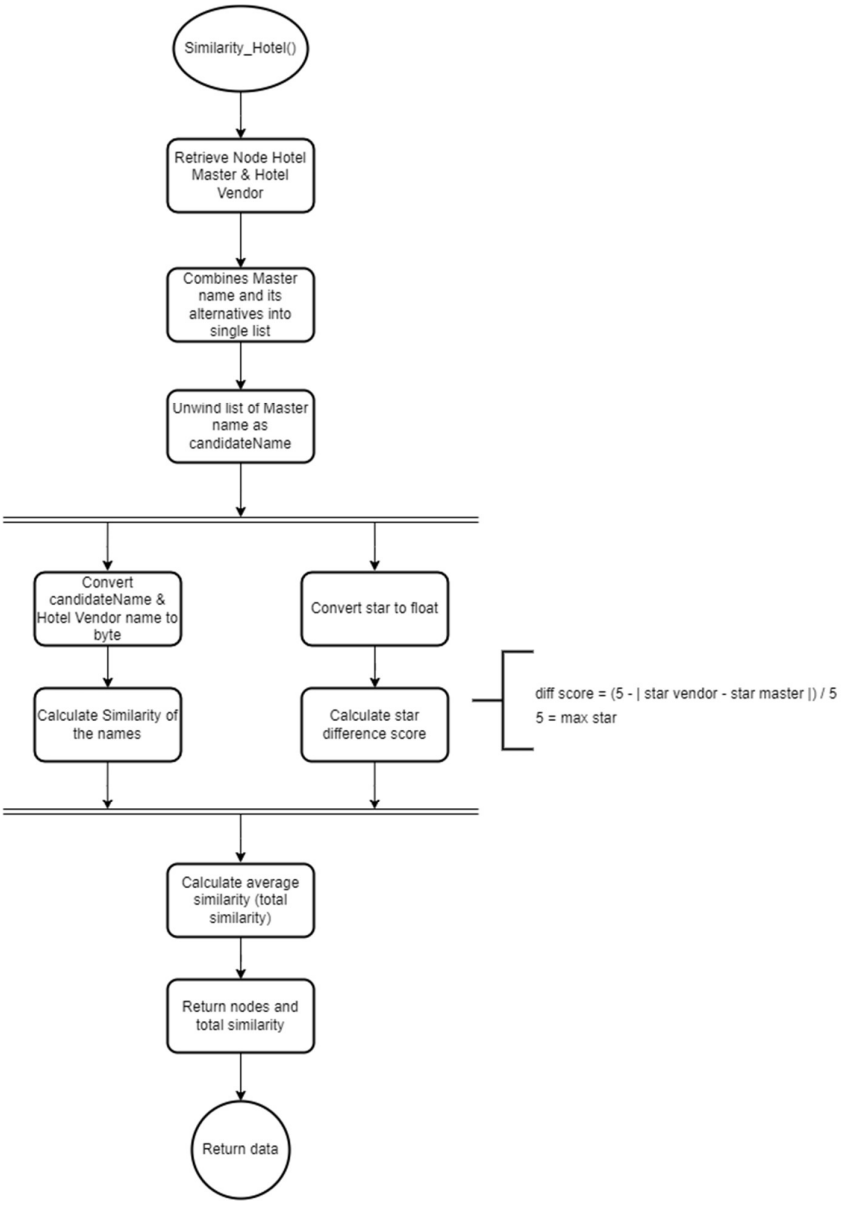
Gambar 3.3 *Flowchart* Alur *Similarity Search* dan *Mapping*

3.5.3 Perhitungan *Similarity*

Perhitungan *similarity* digunakan untuk mencari nilai dari *property node master* yang paling mirip dengan nilai dari *property node* vendor. Mekanisme untuk menghasilkan nilai *similarity* untuk setiap tiga tipe *node* yang ada disesuaikan dengan *property* yang dimiliki oleh *node* tersebut.

3.5.3.1 Perhitungan *Similarity Node* Hotel

Untuk perhitungan *similarity* dari *node* hotel, nilai dari *property* name dan semua nilai yang ada dalam *property* alt_names di *node* master akan dijadikan satu dalam satu *list*. Setelah itu, semua nilai dari *list* ini akan dihitung *similarity*-nya terhadap nilai dari *property* name dari *node* vendor, dan untuk nilai akhirnya diperoleh dari rata-rata nilai *similarity* nama dan nilai *similarity* star.



Gambar 3.4 *Flowchart* Perhitungan *Similarity Node* Hotel

Untuk nilai *similarity* dari star, semakin besar selisih antara nilai star dari *node* vendor dan master akan menghasilkan nilai *similarity* yang lebih kecil, dan demikian pula sebaliknya. Perhitungan nilai *similarity* untuk star dilakukan dengan mencari selisih dari nilai star vendor dan star master, lalu selisih tersebut dimutlakkan agar nilainya tidak negatif. Kemudian akan dicari selisih dari 5 dengan nilai mutlak tersebut. 5 adalah skor maksimal dari *property* star. Hasilnya akan dibagi dengan 5, dengan nilai akhir 1 menandakan nilai yang identik, dan 0 untuk nilai yang tidak identik. Contoh untuk perhitungan nilai star terdapat pada Tabel 3.6.

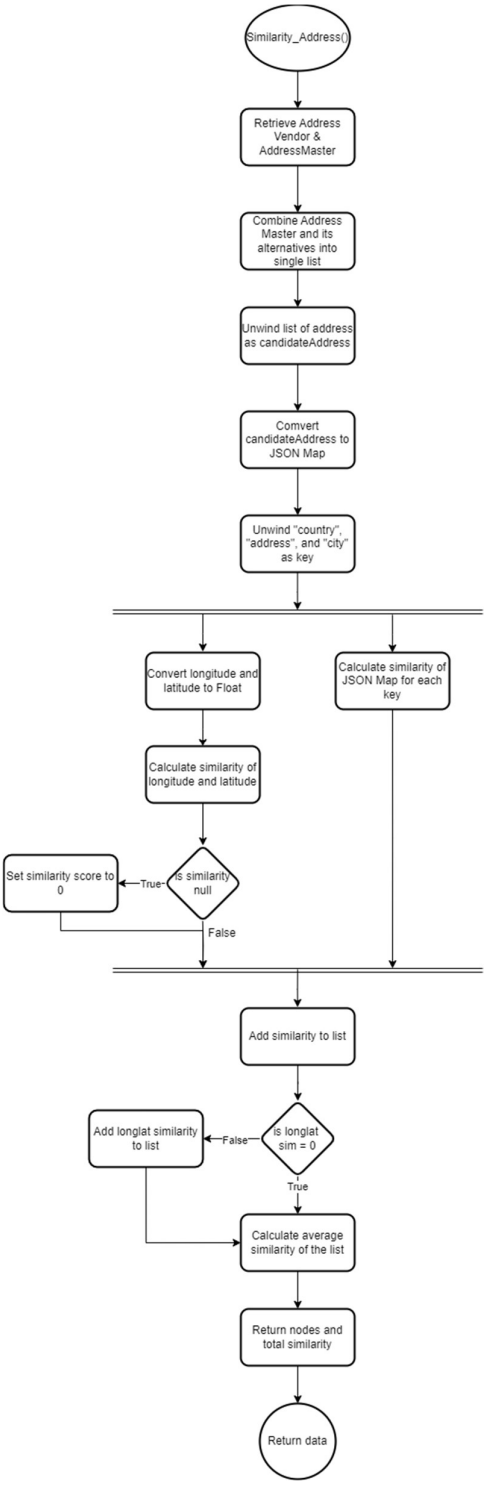
Tabel 3.6

Perhitungan Nilai Star

Star Vendor	Star Master	Nilai Star
4	4	(5 - | 4 - 4 |) / 5 = (5 - 0) / 5 = 1
2	5	(5 - | 2 - 5 |) / 5 = (5 - 3) / 5 = 0.4
5	0	(5 - | 5 - 0 |) / 5 = (5 - 5) / 5 = 0

3.5.3.2 Perhitungan *Similarity Node* Address

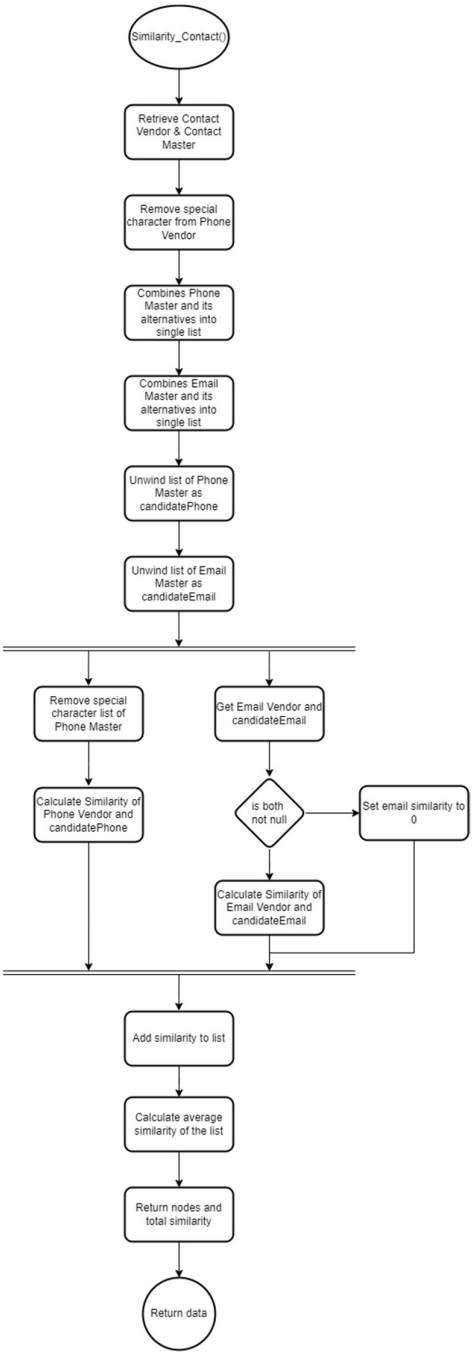
Untuk perhitungan *similarity* dari *node* Address, nilai dari *property* address_detail dan semua nilai yang ada dalam *property* alt_address di *node* master akan dijadikan satu dalam satu *list*. Setelah itu, semua nilai dari *list* ini akan dihitung *similarity*-nya terhadap nilai dari *property* address_detail dari *node* vendor.



Gambar 3.5 *Flowchart* Perhitungan *Similarity Node* Address

3.5.3.3 Perhitungan *Similarity Node* Contact

Untuk perhitungan *similarity* dari *node* Contact nilai dari *property* phone dan semua nilai yang ada dalam *property* alt_phone di *node* master akan dijadikan satu dalam satu *list*. Demikian juga untuk email, sehingga akan terdapat dua *temporary list* yaitu *list* untuk semua nilai phone dan *list* untuk semua nilai email. Setelah itu, semua nilai dari *list* ini akan dihitung *similarity*-nya terhadap nilai dari *property* phone dan email dari *node* vendor, dan untuk nilai akhirnya diperoleh dari rata-rata nilai *similarity* phone dan nilai *similarity* email.



Gambar 3.6

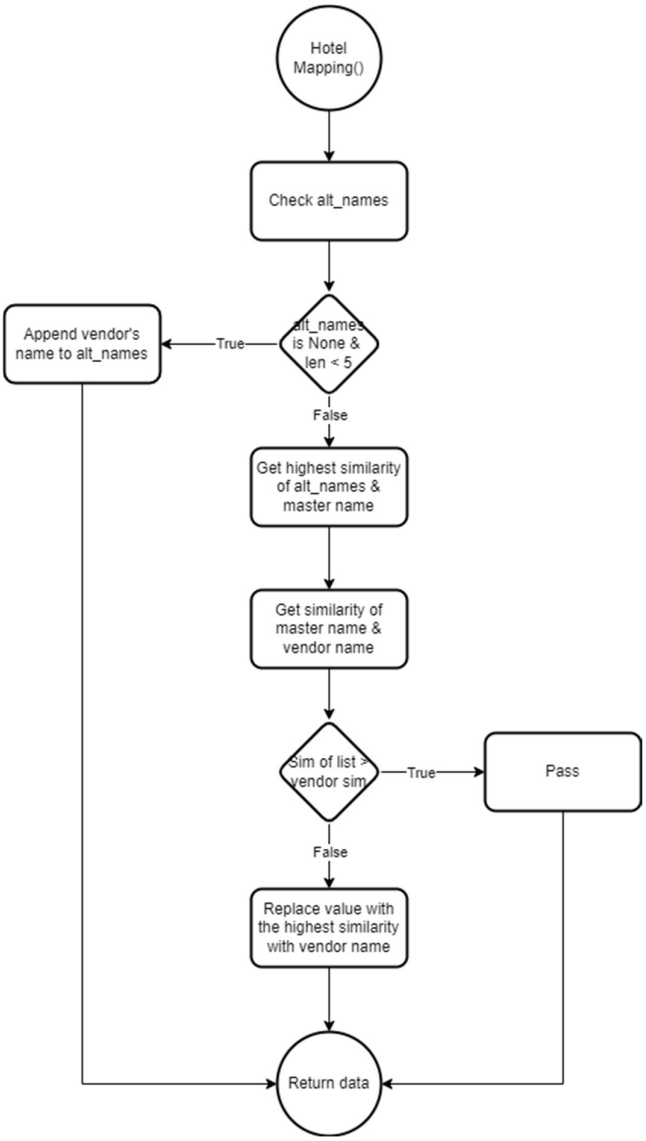
Flowchart Perhitungan *Similarity Node* Contact

3.5.4 Proses *Mapping*

Setelah didapatkan *record* master dengan nilai *similarity* yang paling tinggi terhadap *record* vendor, maka akan dilakukan proses *mapping* dengan cara menambahkan nilai-nilai dari *property* vendor ke *property* master terkait.

3.5.4.1 *Mapping Node* Hotel

Proses *mapping* dilakukan dengan mengecek apakah terdapat *property* alt_names pada *node* Hotel_Master. Apabila tidak terdapat *property* alt_names, maka program akan membuat *property* alt_names dan menambahkan nilai *property* name dari *node* Hotel_Vendor ke dalam alt_names. Apabila jumlah data pada alt_names kurang dari 5, maka nilai *property* name dari vendor akan langsung ditambahkan ke dalam alt_names. Apabila ternyata *node* Hotel_Master sudah memiliki *property* alt_names dan jumlah data di dalamnya adalah 5, maka program akan mengecek nilai *similarity* tertinggi dari nama-nama hotel yang ada dalam *list* tersebut terhadap *property* name pada Hotel_Master serta menghitung nilai *similarity* dari *property name* pada Hotel_Vendor. Kedua nilai ini akan dibandingkan, dan apabila nilai *similarity* tertinggi dari alt_names lebih tinggi dari pada nilai *similarity* dengan nama hotel pada Hotel_Vendor, maka nama hotel dengan *similarity* tertinggi tersebut akan diganti dengan nama pada Hotel_Vendor. Apabila nilai *similarity* tertinggi dari alt_names lebih rendah dari pada nilai *similarity* dengan nama hotel pada Hotel_Vendor, maka tidak akan ada perubahan dalam *list* alt_names. *Flowchart* untuk proses *mapping node* Hotel dapat dilihat pada Gambar 3.7.

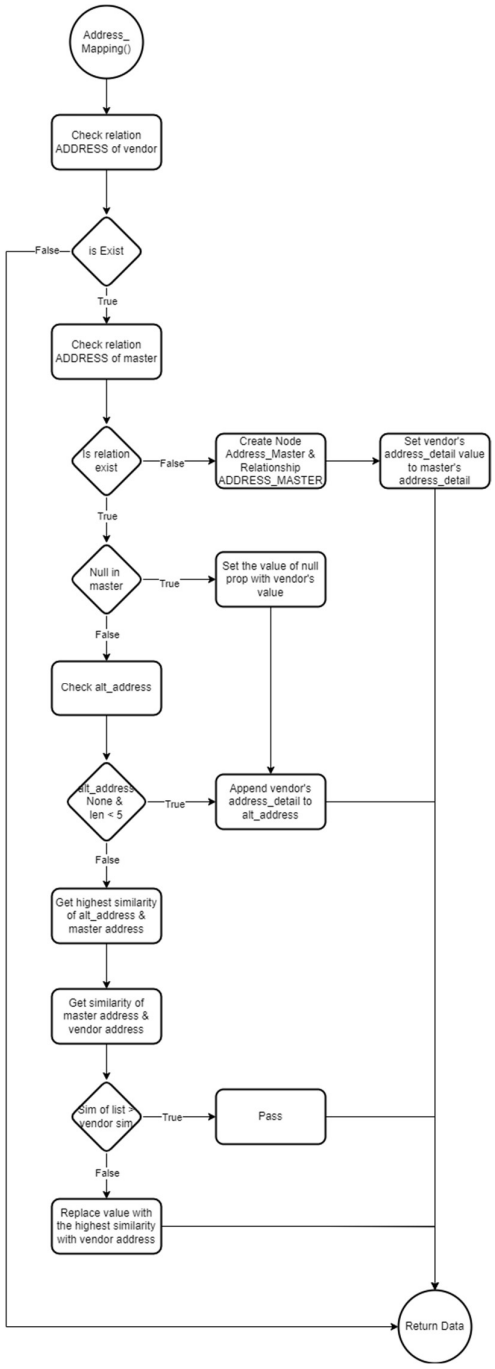


Gambar 3.7 *Flowchart* Proses *Mapping Node* Hotel

3.5.4.2 *Mapping Node* Address

Proses *mapping* dilakukan dengan mengecek apakah terdapat *property* alt_address pada *node* Address_Master. Apabila tidak terdapat *property* alt_address, maka program akan membuat *property* alt_address dan menambahkan nilai *property* address_detail dari *node* Address_Vendor ke dalam alt_address. Apabila jumlah data pada alt_address kurang dari 5, maka nilai *property* address_detail dari vendor akan langsung ditambahkan ke dalam alt_address. Apabila ternyata *node* Address_Master sudah memiliki *property* alt_address dan jumlah data di dalamnya adalah 5, maka program akan mengecek nilai *similarity* tertinggi dari

alamat-alamat hotel yang ada dalam *list* tersebut terhadap *property* address_detail pada Address_Master serta menghitung nilai *similarity* dari *property* address_detail pada Address_Vendor. Kedua nilai ini akan dibandingkan, dan apabila nilai *similarity* tertinggi dari alt_address lebih tinggi dari pada nilai *similarity* dengan nama hotel pada Address_Vendor, maka alamat hotel dengan *similarity* tertinggi tersebut akan diganti dengan alamat pada Address_Vendor. Apabila nilai *similarity* tertinggi dari alt_address lebih rendah dari pada nilai *similarity* dengan alamat hotel pada Address_Vendor, maka tidak akan ada perubahan dalam *list* alt_address. *Flowchart* untuk proses *mapping node* Address dapat dilihat pada Gambar 3.8

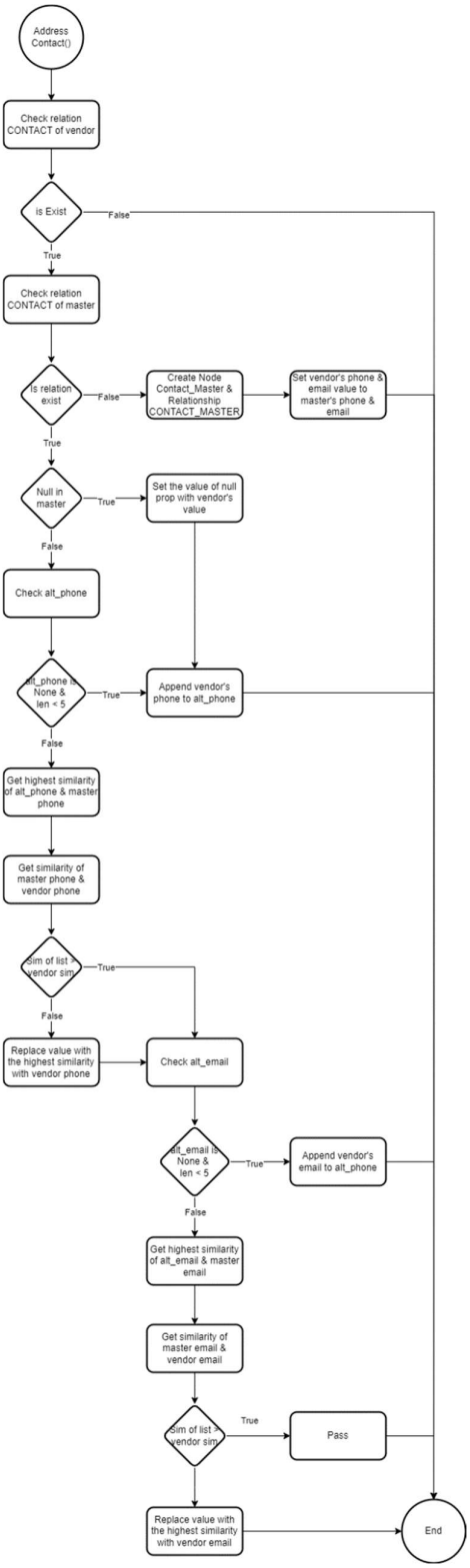


Gambar 3.8 *Flowchart* Proses *Mapping Node Address*

3.5.4.3 *Mapping Node* Contact

Proses *mapping* dilakukan dengan mengecek apakah terdapat *property* alt_phone dan alt_email pada *node* Contact_Master. Apabila tidak terdapat *property* alt_phone, maka program akan membuat *property* alt_phone dan menambahkan nilai *property* phone dari *node*

Contact_Vendor ke dalam alt_phone. Apabila jumlah data pada alt_phone kurang dari 5, maka nilai *property phone* dari vendor akan langsung ditambahkan ke dalam alt_phone. Apabila ternyata *node* Contact_Master sudah memiliki *property* alt_phone dan jumlah data di dalamnya adalah 5, maka program akan mengecek nilai *similarity* tertinggi dari nomor telepon hotel yang ada dalam *list* tersebut terhadap *property* phone pada Contact_Master serta menghitung nilai *similarity* dari *property* phone pada Contact_Vendor. Kedua nilai ini akan dibandingkan, dan apabila nilai *similarity* tertinggi dari alt_phone lebih tinggi dari pada nilai *similarity* dengan nama hotel pada Contact_Vendor, maka nomor telepon dengan *similarity* tertinggi tersebut akan diganti dengan nomor telepon pada Contact_Vendor. Apabila nilai *similarity* tertinggi dari alt_phone lebih rendah dari pada nilai *similarity* nomor telepon pada Hotel_Vendor, maka tidak akan ada perubahan dalam *list* alt_phone. Setelah pengecekan terhadap *property* phone, dijalankan proses yang sama untuk mengecek *property* email. *Flowchart* untuk proses *mapping node* hotel dapat dilihat pada Gambar 3.9.



Gambar 3.9 *Flowchart* Proses *Mapping Node Contact*