Penerapan Algoritma Dijkstra untuk Menentukan Rute Terpendek Pembacaan *Water Meter* Induk PDAM Tirta Kerta Raharja Kabupaten Tangerang

Ferdiansyah[#1], Ahmad Rizal[#2]

[#] *Program Studi Teknik Informatika Fakultas Teknologi Informasi Universitas Budi Luhur*
Jl. Raya Ciledug, Petukangan Utara, Kebayoran Lama, Jakarta Selatan 12260
Telp. (021) 5853753, Fax. (021) 5866369
[1]ferdiansyah@budiluhur.ac.id
[2]rizaldicaprio@yahoo.com

Abstraksi— Pencarian rute terpendek merupakan salah satu persoalan yang dihadapi oleh pegawai PDAM Tirta Kerta Raharja Kabupaten Tangerang yang bertugas untuk melakukan dan mencatat pembacaan terhadap *water meter* pelanggan yang terpasang di rumah atau lokasi dimana pelanggan berada. Lokasi pelanggan yang tersebar di beberapa wilayah pelayanan di Kota Tangerang, Kabupaten Tangerang bahkan sampai wilayah DKI Jakarta tentu saja memerlukan perhatian khusus seperti penentuan rute dan waktu yang harus ditempuh mulai dari Kantor Pusat ke lokasi para pelanggan. Hal ini dikarenakan banyaknya alternatif jalan yang dapat dilalui oleh petugas pembaca meter. Persoalan penentuan jalur terpendek ini bisa diselesaikan dengan algoritma Dijkstra. Algoritma Dijkstra menggunakan prinsip Greedy, yaitu mencari jalur terpendek dari satu *node* (titik/vertek) ke node lainnya yang terhubung. Prinsip ini digunakan untuk memecahkan solusi optimum dalam konteks yang baik, dengan mengambil apa saja yang diperoleh sekarang. Data-data pendukung perhitungan algoritma Dijkstra ini menggunakan data yang berasal dari informasi bagian Pelanggan mengenai nama dan lokasi pelanggan dan data koordinat lokasi dari GoogleMaps. Pada penelitian ini dihasilkan aplikasi website untuk mencari rute terpendek ke beberapa lokasi pelanggan, sehingga diharapkan para petugas pembaca meter dapat dengan mudah dan cepat untuk mencapai lokasi tujuan dengan mengaksesnya melalui komputer desktop, handphone, smartphone atau iPad.

***Kata kunci*— algoritma dijkstra, node, rute perjalanan terpendek, greedy, *water meter*, website.**

Shortest route is one of the problems faced by staff of PDAM Tirta Kerta Raharja Kabupaten Tangerang when performing and recording water meters installed in the home or the location where the customer is located. Customer locations spread across multiple service areas in Kota Tangerang, even DKI Jakarta of course requires special attention such as determining the route and time that must be taken from the central office to the customer area.That is to many alternate path or route can be traversed by staff. The issue of determining the shortest path can be solved by Dijkstra's algorithm, its uses the principle of the Greedy that find the shortest route from one node (point/vertex) to other nodes that are connected. This principle is used to solve the optimum solution, by taking whatever is obtained. The data supporting to calculation of the Dijkstra's algorithm uses data derived from the customer information about the name and customer location, and also the location coordinate from GoogleMaps. This research generated web application to find the shortest route customer area, so it is expected staff water meter can easily and quickly to reach the destination by access them via desktop computers, mobile phone, smartphone or iPad.

Keyword: Dijkstra's algorithm, node, Shortest route, greedy, water meter, website

I. PENDAHULUAN

Penerapan Teknologi Informasi dalam mendukung kegiatan operasional suatu perusahaan dewasa ini sudah menjadi satu keharusan bahkan kewajiban sebagai media komunikasi dan informasi. Berbagai media teknologi informasi telah dirancang untuk dapat memberikan fasilitas komunikasi tidak hanya berlaku bagi perusahaan besar tapi juga perusahaan menengah bahkan perusahaan kecil. Salah satu media yang sudah berkembang pesat adalah media *internet* dimana penyampaian dan penerimaan informasi dapat dilakukan secara mudah disetiap saat, kondisi dan waktu.

Sebagai perusahaan milik pemerintah daerah Kabupaten Tangerang, PDAM Tirta Kerta Raharja Kabupaten Tangerang telah mengupayakan penerapan teknologi informasi dalam mendukung kegiatan operasional pegawai dalam meningkatkan kinerja perusahaan. Namun kondisi saat ini masih ada beberapa kegiatan atau pekerjaan yang masih belum tersentuh secara langsung oleh teknologi informasi salah satunya adalah aplikasi untuk penentuan rute terpendek bagi para pembaca meter dalam melakukan perjalanan dan pembacaan water meter pelanggan skala besar yang biasa dikenal sebagai pelanggan curah yang tersebar diberbagai wilayah Tangerang dan DKI Jakarta. Pembacaan water meter induk merupakan salah satu kegiatan yang sangat berpengaruh besar dikarenakan pelanggan curah merupakan golongan pelanggan yang memberikan pendapatan cukup besar bagi

perusahaan. Hal inilah yang menjadi pilihan dan konsentrasi penelitian ini.

Gbr 1 : *Water meter* induk

Masalah pencarian rute terpendek diatas, bisa diselesaikan salah satunya dengan penerapan algoritma *Dijkstra*. Algoritma *Dijkstra* menggunakan prinsip *Greedy*, yaitu mencari jalur terpendek dari satu node(titik/vertek) ke *node* lainnya yang searah (*directed graph*) mulai dari *node* asal sampai *node* tujuan. *Node-node* yang dihitung didapat dari beberapa lokasi strategis yang dapat dikenal secara umum seperti kantor, persimpangan jalan atau lokasi umum yang mudah diingat oleh petugas. Data-data pendukung perhitungan algoritma *Dijkstra* ini menggunakan data yang berasal dari informasi bagian Pelanggan mengenai nama dan lokasi pelanggan sedangkan untuk data koordinat lokasi didapat dari *GoogleMaps*.

Pada penelitian ini dihasilkan aplikasi *website* untuk mencari rute terpendek ke beberapa lokasi pelanggan, sehingga diharapkan para petugas pembaca meter dapat dengan mudah dan cepat untuk mencapai lokasi tujuan dengan mengaksesnya melalui komputer *desktop, handphone, smartphone* atau *iPad*.

II. LANDASAN TEORI

Algoritma *Dijkstra* ditemukan oleh Edger Wybe Dijkstra yang merupakan salah satu jenis bentuk algoritma populer dalam pemecahan persoalan yang terkait dengan masalah optimasi dan bersifat sederhana [1]. Algoritma ini menyelesaikan masalah mencari sebuah lintasan terpendek dari *vertex a* ke *vertex z* dalam *graph* berbobot, bobot tersebut adalah bilangan positif jadi tidak dapat dilalui oleh *node* negatif, namun jika terjadi demikian, maka penyelesaian yang diberikan adalah infiniti atau jumlah tak terbatas.

Algoritma *Dijkstra* melibatkan pemasangan label pada vertex. Misalkan L(v) menyatakan label dari vertex v. Pada setiap pembahasan, beberapa vertex mempunyai label sementara dan yang lain mempunyai label tetap. Misalkan T menyatakan himpunan vertex yang mempunyai label sementara [2]. Dalam menggambarkan algoritma tersebut kumpulan vertex yang mempunyai label tetap akan dilingkari. Selanjutnya jika L(v) adalah label tetap dari vertex v, maka L(v) merupakan label sementara ke tetap. Pada bagian ini L(z) merupakan panjang kan merupakan lintasan terpendek dari a ke z. P pada algoritma *Dijkstra node* digunakan, karena algoritma Dijkstra menggunakan diagram pohon untuk penentuan jalur lintasan n terpendek dan menggunakan *graph* yang berarah.

Rumusan Algoritma *Dijkstra* [3] :

$V(G)$ = $\{v_1,v_2,v_3,..,v_n\}$
L = Himpunan *node* V(G) yang sudah terpilih dalam jalur terpendek
$D(j)$ = Jumlah bobot jarak terkecil dari v_1 ke v_j
$W(i,j)$ = Bobot garis dari *node* v_i ke *node* v_j
$W^*(1,j)$ = Jumlah bobot jarak terkecil dari v_1 ke v_j

Secara formal, algoritma *Dijkstra* untuk mencari jarak terpendek adalah :

1. L = { }
2. V = $\{v_2, v_3, ..., v_n\}$
3. Untuk i = 2, 3, ... , n, lakukan D(i) = w(1,i)
4. Selama vn $\in$ L lakukan :
   a. Pilih node v_k V-L dengan D(k) terkecil
      L = L $\cup$ $\{v_k\}$
   b. Untuk setiap vj V-L lakukan :
      Jika D(j) > D(k) + w(k,j) maka ganti D(j)
      dengan D(k) + w(k,j)

Untuk setiap $v_j \in$ V, $w^*(1,j) = D(j)$.

A. *Penerapan Algoritma Dijkstra*

Penerapan algoritma *Dijkstra* dilakukan dengan pertama-tama tentukan titik mana yang akan menjadi *node* awal, lalu beri bobot jarak pada *node* pertama ke *node* terdekat satu per satu, Dijkstra akan melakukan pengembangan pencarian dari satu titik ke titik lain dan ke titik selanjutnya tahap demi tahap. Inilah urutan logika dari algoritma *Dijkstra* [4] :

1. Beri nilai bobot (jarak) untuk setiap titik ke titik lainnya, lalu set nilai 0 pada *node* awal dan nilai tak hingga terhadap *node* lain (belum terisi)
2. Set semua *node* "Belum dilewati" dan set *node* awal sebagai "*Node* keberangkatan"
3. Dari *node* keberangkatan, pertimbangkan *node* tetangga yang belum dilewati dan hitung jaraknya dari titik keberangkatan. Sebagai contoh, jika titik keberangkatan 1 ke 2 memiliki bobot jarak 7 dan dari 2 ke 3 berjarak 10, maka jarak ke 3 melewati 2 menjadi 7+10=17. Jika jarak ini lebih kecil dari jarak sebelumnya (yang telah terekam sebelumnya) hapus data lama, simpan ulang data jarak dengan jarak yang baru.
4. Saat kita selesai mempertimbangkan setiap jarak terhadap *node* tetangga, tandai *node* yang telah dilewati sebagai "*Node* dilewati". *Node* dilewati tidak akan pernah dicek kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya.
5. Set "*Node* belum dilewati"dengan jarak terkecil (dari *node* keberangkatan) sebagai "*Node* Keberangkatan" selanjutnya dan lanjutkan dengan kembali ke *step* 3.

Algoritma *Dijkstra* juga dapat menggunakan larik S=[s_i] dengan *pseudocode* [5] sebagai berikut :

S_i=1, jika simpul i termasuk ke dalam lintasan terpendek
S_i=0, jika simpul i tidak termasuk ke dalam lintasan Terpendek dan larik/tabel D=[d_i] yang dalam hal ini,
d_i=panjang lintasan dari simpul awal ke simpul i

Algoritma Lintasan Terpendek *Dikstra* (mencari lintasan terpendek dari simpul a ke simpul lain)

Langkah 0 (inisialisasi) :
Inisialisasi s_i=0 dan d_i=m_{ij} untuk i=1,2,….,*n*
Langkah 1 :
Isi s_a dengan 1 (karena simpul a adalah simpul asal lintasan terpendek, jadi sudah pasti terpilih)
Isi d_a dengan ∞ (tidak ada lintasan terpendek dari simpul a ke a)
Langkah 2,3,…,*n*-1 :
Cari *j* sedemikian hingga s_j=0 dan d_j=min {d_1,d_2,…,d_n}
Isi s_j =dengan 1
Perbarui d_i , dengan i=1,2,3,…,n
dengan: d_i (baru)=min{d_i (lama),d_j+m_{ij}}

B. *HTML*

HTML merupakan kependekan dari *Hyper Text Markup Language*. Dokumen HTML adalah *file text* murni yang dapat dibuat dengan editor teks sembarang. Dokumen ini dikenal sebagai *web page*. Dokumen HTML merupakan dokumen yang disajikan dalam *browser web surfer*. Dokumen ini umumnya berisi informasi ataupun *interface* aplikasi didalam *internet*.

Dokumen HTML disusun oleh elemen-elemen. Elemen merupakan istilah bagi komponen-komponen dasar pembentukan dokumen HTML diantaranya adalah *Head, body, table,* dll. Elemen yang dibutuhkan untuk membuat suatu dokumen HTML dinyatakan dengan tag. Tag HTML terdiri atas sebuah kurung sudut kiri "<" sebuah nama tag dan sebuah kurung sudut kanan ">". Tag pada umumnya berpasangan dan yang menjadi pasangannya selalu diawali dengan karakter garis miring "/".

C. PHP

PHP adalah skrip yang dijalankan di *server*. Keuntungan PHP, kode yang menyusun program tidak perlu diedarkan ke pemakai sehingga kerahasiaannya kode dapat dilindungi. Hal menarik yang didukung oleh PHP adalah dapat digunakan untuk mengakses berbagai macam *database* seperti Acces, Oracle, MySQL,dll.

Pada PHP juga mengenal *variable* yang berfungsi untuk menyimpan suatu nilai dan nilai yang ada didalamnya dapat diubah sewaktu-waktu. Dalam membuat nama *variable*, nama yang dipilih harus memenuhi aturan pengenal. Pengenal (*identifier*) banyak digunakan dalam program untuk memberikan nama *variable*, fungsi atau kelas.

D. MySQL

MySQL merupakan database server dimana pemrosesan data terjadi di *server*, dan *client* hanya mengirimkan data serta meminta data. Oleh karena pemrosesan terjadi di server sehingga pengaksesan data tidak terbatas. MySQL mempunyai lisensi yang cukup kompleks, yaitu jika di instal pada sistem operasi Microsoft windows adalah *shareware* tetapi tidak mempunyai *expired date*. Sedangkan jika di instal pada sistem operasi selain Microsoft Windows adalah *free* sesuai dengan *General Public Licence* (GPL).

MySQL termasuk dalam kategori *database management system*, yaitu *database* yang terstruktur dalam pengolahan dan penampilan data. Ada beberapa alasan mengapa MySQL menjadi program *database* yang sangat populer dan digunakan oleh banyak orang. Alasan-alasan di antaranya adalah:

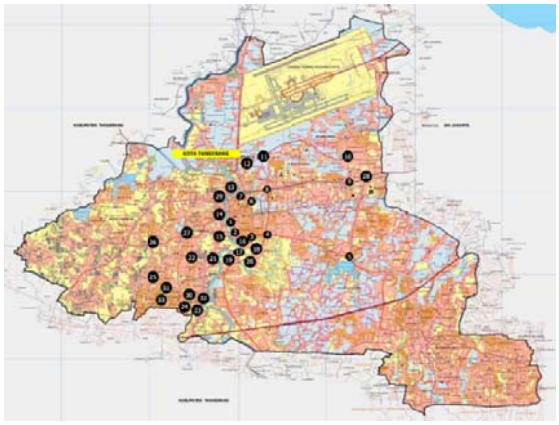
1. MySQL merupakan *database* yang memiliki kecepatan yang tinggi dalam melakukan pemrosesan data, dapat diandalkan dan mudah digunakan serta mudah dipelajari.
2. MySQL mendukung banyak bahasa pemrograman seperti C, C++, Perl, Phython, Java, Visual Basic dan PHP.
3. MySQL dapat menangani *database* dengan skala yang sangat besar dengan jumlah *record* mencapai lebih dari 50 juta.
4. MySQL merupakan *software database* yang bersifat *free* atau gratis, jadi kita tidak perlu susah-susah mengeluarkan uang hanya untuk sekedar membiayai lisensi.

III. ANALISA MASALAH DAN RANCANGAN PROGRAM

Penelitian ini berupaya membangun sebuah *prototype* sistem yang menunjukan lintasan terpendek antara beberapa *node* atau lokasi yang diimplementasikan pada sistem Rute Baca Meter dimana selanjutnya water meter pada *node-node* ini akan dibaca secara berkala oleh Petugas pembaca water meter. Karena banyak lokasi water meter yang harus dimasukan (tersebar di Kabupaten Tangerang, Kota Tangerang, Kota Tangerang Selatan dan DKI Jakarta) maka sistem ini menggunakan penelitian yang bersifat simulatif dan deskriptif dengan mengambil *sample* lokasi water meter di Kota Tangerang yang diasumsikan sebagai *node* (*vertek*), dimana *node-node* tersebut dihubungkan oleh garis-garis yang disimulasikan sebagai jalan yang diberikan nilai (satuan jarak dalam Kilometer). Jarak didapat dari perhitungan koordinat *latitude* dan *longitude* yang didapat dari aplikasi *GoogleMaps* dan atau berdasarkan pengukuran langsung menggunakan *tool* pengukuran GPS (*Global Positioning System*) dari *node* asal ke *node* tujuan. Pengukuran dengan GPS dilakukan secara langsung pada titik *node* yang akan dipergunakan pada aplikasi ini.

A. *Sebaran Node*

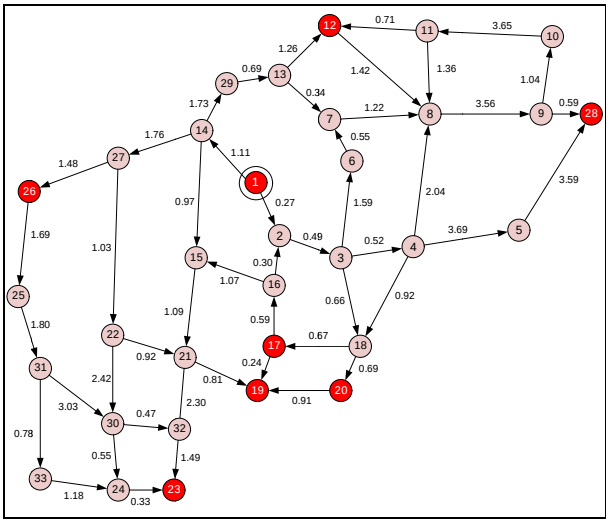
User akan memilih *node-node* yang akan dikunjungi atau terlewati dalam jalur rute baca meter, kemudian sistem akan melakukan perhitungan dan menentukan jalur terpendek yang harus dilalui dengan algoritma *Dijkstra* dimulai dari Kantor Pusat PDAM TKR sebagai *node* awal hingga *node* terakhir yang akan dikunjungi. Gambar 2 dibawah ini menunjukan sebaran *node* pada peta wilayah Kota Tangerang yang akan dipergunakan pada aplikasi Rute Baca Meter :



Gbr 2 . Sebaran *Node* pada peta Kota Tangerang

Node-node merupakan lokasi dimana water meter terpasang, berupa kantor, bangunan atau lokasi yang secara umum diketahui oleh Pembaca Meter seperti pertigaan, perempatan atau fasilitas umum seperti pusat perbelanjaan (*mall*). *Node* yang dimasukan selanjutnya diberikan penomoran untuk memudahkan identifikasi lokasi pada *graph*.

Bentuk *graph* dari *node-node* pada peta di gambar 2 digambarkan pada gambar 3 dibawah ini :



Gbr 3. *Graph* Sebaran *Node* Rute Baca Meter

Node-node tersebut diberikan nama sesuai lokasi dimana *node* tersebut berada, seperti pada tabel 1 dibawah ini :

TABEL I
NAMA LOKASI NODE

Node	Lokasi	Node	Lokasi
1	Kantor Pusat PDAM	18	UNIS – LP Wanita
2	Pertigaan Bintang	19	WM Induk IPA Cikokol
3	Perempatan Adipura	20	SPBU TangCity
4	Fly Over Cipondoh	21	Perempatan Teuku Umar
5	Pertigaan Cipondoh Makmur	22	Perempatan Beringin
6	Puspem Kota Tangerang	23	WM Induk IPA Perumnas
7	Jembatan Pasar Baru	24	Perempatan Betet Raya
8	Perempatan HAKI Tanah Tinggi	25	Pertigaan Perumnas IV
9	Pertigaan Kebon Besar	26	WM Induk Cibodas Niaga
10	Jurumudi	27	Terminal Cimone
11	Perumahan Angkasa Pura II	28	WM Budi Indah Perkasa
12	WM Induk Bandara	29	Pertigaan Strada
13	Gedung Cisadane	30	Perempatan BTN
14	Pertigaan Otista	31	Pertigaan Prambanan
15	Perempatan BCA	32	Pertigaan Kavling Pemda
16	Jembatan UNIS	33	Pertigaan Kecipir
17	WM Induk TangCity		

Permodelan rute yang menjadi bahan kajian adalah membuat perhitungan algoritma dari *node* 1 yang merupakan Kantor Pusat PDAM TKR sebagai lokasi awal pembacaan menuju ke beberapa *node* sebagai lokasi water meter yaitu *node* 28, *node* 12, *node* 17, *node* 19, *node* 20, *node* 23 dan *node* 26.

Dikarenakan sifat *Dijkstra* yaitu *greedy* (rakus) dimana proses perhitungan dikenakan kepada seluruh *node-node* yang ada, maka untuk kemudahan perhitungan terhadap rute baca meter ini maka digunakan *Bounding Box* yaitu dengan memilih *node-node* tertentu saja yang searah dimana *node* tersebut kemungkinan besar dilalui atau terlewati dalam satu rute pembacaan. Hal ini membuat perhitungan *Dijkstra* lebih cepat dan efisien. Sebagai contoh adalah untuk rute pembacaan dari *Node* 1 ke *Node* 17 maka *node* yang dimasukan kedalam *bounding box* adalah *node* 2, *node* 3, *node* 4, *node* 18, *node* 17.

B. *Pseudocode pencarian node terdekat*

Dijalankan melalui beberapa proses yaitu proses menghitung seluruh jarak *node* dari tujuan dalam satuan

kilometer (km) dilanjutkan dengan mencari jarak terkecil. *Pseudocode* untuk menghitung jarak adalah sebagai berikut :

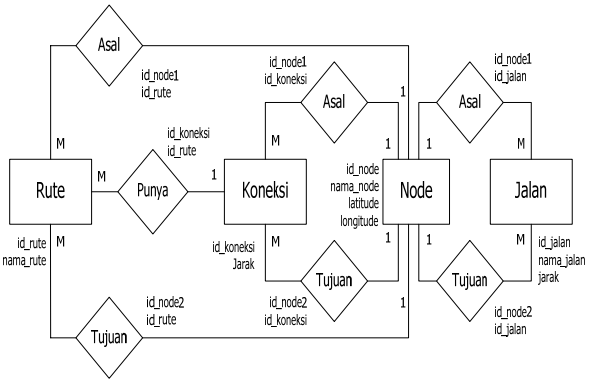
```
Procedure jarak semua node (INPUT lat1:latitude tujuan,
lng1: longitude tujuan, lat2:latitude node, lng2: longitude
node) {
Radian = 0.0174532925199433
a = 6378.137
f = 0.0033528107
lat1 = (latitude1 x radian)
long1 = (longitude1 x radian)
lat2 = (latitude2 x radian)
long2 = (longitude2 x radian)
U = (lat1 + lat2)/2;
G = (lat1 – lat2)/2;
J = (long1 – long2)/2;
M = (sin(G) x sin(G) x cos(J) x cos(J)) + (cos(U) x
      cos(U) x sin(J) x sin(J));
N = (cos(G) x cos(G) x cos(J) x  cos(J)) + (sin(U) x
      sin(U) x sin(J) x sin(J));
w = sqrt(M/N);
P = sqrt(M x N)/w;
D = 2 x w x a;
E1 = (3 x P-1)/(2 x N);
E2 = (3 x P+1)/(2 x M);
s = D x (1 + f x E1 x sin(U) x sin(U) x cos(G) x
     cos(G)-f x E2 x cos(U) x cos(U) x sin(G) x
     sin(G));
jarak (km) = s;
}
```

Selanjutnya *pseudocode* untuk mencari jarak terkecil adalah :

```
Procedure jarak node terdekat (INPUT hasil km[n]:matriks)
{
Urutan = Jumlah matriks
while Urutan bukan 0
    if hasil km[n] lebih kecil dari nilai sebelumnya
       tetapkan ini sebagai nilai terkecil
    endif
endwhile
hasilkan angka terkecil
}
```

C. *Entity Relationship Diagram* (ERD)

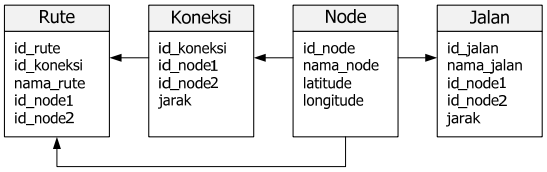
Entity Relationship Diagram (ERD) merupakan suatu model untuk menjelaskan hubungan antar data dalam basis data berdasarkan objek-objek dasar data yang mempunyai relasi. Untuk memodelkan struktur data dan hubungan antar data digunakan beberapa notasi dan simbol. Gambar 4 merupakan ERD dari *database* program aplikasi ini :



Gbr 4. *Entity Relationship Diagram*

D. *Logical Record Structure (LRS)*

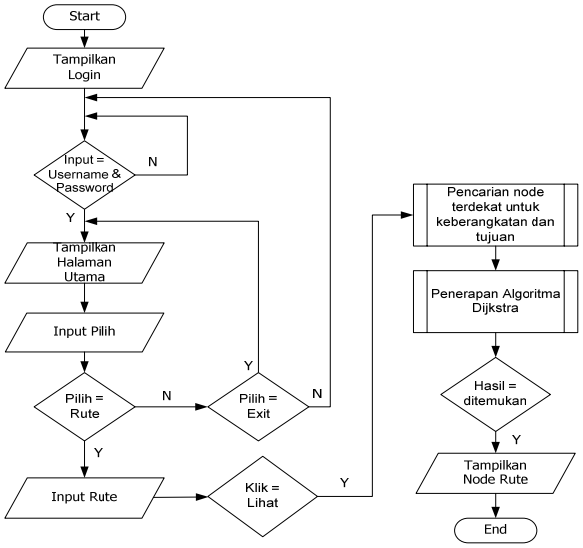
LRS dapat ditentukan jika proses transformasi ERD ke LRS telah dilakukan [6]. Bentuk LRS dari hasil transformasi ERD tampak seperti gambar 5 dibawah ini :



Gbr 5 . *Logical Record Structure*

E. *Flow chart*

Gambar 6 dibawah ini menggambarkan salah satu *flow chart* dari aplikasi rute baca meter yaitu *flow chart* rute dimana proses penghitungan algoritma *Dijkstra* dilakukan.



Gbr 6 : *Flow chart Rute*

Penjelasan algoritma untuk *flow chart* rute pada gambar 6 adalah :

1. Tampilan Login
2. Input Pilih
3. **If** Pilih = Username dan Password **Then**
4. Tampilan Halaman Utama
5. **Input** Pilih
6. **If** Pilih = Home **Then**
7. Tampilan pilihan Rute, tombol Lihat
8. **Input** pilih Rute
9. **If** Pilih = Lihat
10. Proses Pencarian dengan penerapan Dijkstra
11. **Endif**
12. **Else If**
13. **Else if** Pilih = Exit **Then**
14. **Endif**

IV. Implementasi Dan Analisa Hasil Uji Coba Program

Sistem ini mempunyai dua sisi, yaitu sisi *client* dan sisi *server*. Sisi *client* merupakan *user* yang menggunakan komputer *desktop, handphone, smartphone* dan komputer *tablet* yang berfungsi untuk pencarian data dan membaca data. Sisi server terdiri dari *web server* yang menggunakan bahasa pemrograman PHP dan *server database* yang menggunakan MySQL. Karena menggunakan *web based* maka perangkat yang dipergunakan haruslah memiliki *web browser* dan koneksi internet.

A. Implementasi Algoritma Dijkstra

Pengujian terhadap perhitungan pada aplikasi dapat diuji coba atau dibandingkan dengan perhitungan secara manual berdasarkan rumusan algoritma *Dijkstra*. Contoh perbandingan perhitungan adalah sebagai berikut : rute pembacaan water meter pada rute 100 yaitu dari *node* 1 ke *node* 17 dimana *node-node* yang kemungkinan dilewati adalah *node* 2 dengan jarak 0.27 km, *node* 3 dengan jarak 0.49 km , *node* 4 dengan jarak 0.52 km dan *node* 18 dengan jarak 0.66 km. Pada aplikasi dihasilkan bahwa rute terpendek dari *node* 1 ke *node* 17 adalah : 1 – 2 – 3 – 18 – 17 dengan total jarak 2.09 km.

Hasil perhitungan algoritma *Dijkstra* [3] adalah sebagai berikut :

1. L={1}
   d[2]=0.27
   d[3]= ∞
   d[4]= ∞
   d[18]= ∞
   d[17]= ∞
   Rute terpendek yang terhubung dengan *node* 1 adalah *node* 2 karena diantara yang lain memiliki bobot jarak terpendek.

2. Iterasi 1
   W=min{2,3,4,18,17} = 2
   L={1,2}
   d[3] = min(d[3],d[2]+c[2,3])= min (∞,0.27+0.49)= 0.76
   d[4] = min(d[4],d[2]+c[2,4])= min (∞,0.27+∞)= ∞
   d[18] = min(d[18],d[2]+c[2,18])= min (∞,0.27+∞)= ∞
   d[17] = min(d[17],d[2]+c[2,17])= min (∞,0.27+∞)= ∞
   Rute selanjutnya dari *node* 2 adalah *node* 3, karena merupakan *node* yang memiliki bobot jarak terpendek dari *node* yang lain.

3. Iterasi 2
   W=min{3,4,18,17} = 3
   L={1,2,3}
   d[4] = min(d[4],d[3]+c[3,4])= min (∞,0.76+0.52)= 1.28
   d[18]=min(d[18],d[3]+c[3,18])=min(∞,0.76+0.66)= 1.42
   d[17] = min(d[17],d[3]+c[3,17])= min (∞,0.76+∞)= ∞
   Rute selanjutnya yang terhubung dengan *node* 3 adalah *node* 4 karena memiliki bobot jarak terpendek.

4. Iterasi 3
   W=min{4,18,17} = 4
   L={1,2,3,4}
   d[18]= min(d[18],d[4]+c[4,18])= min(∞,1.28+0.92)= 2.2
   d[17] = min(d[17],d[4]+c[4,17])= min (∞,1.28+∞)= ∞
   Node yang terhubung dari *node* 4 adalah *node* 18, namun bobot jarak yang dimiliki *node* 18 pada iterasi 3 lebih besar dari bobot jarak dari *node* 3 ke n*ode* 18 pada iterasi 2, untuk itu maka yang dipilih adalah bobot jarak pada iterasi ke 2 sehingga hubungan yang terjadi selanjutnya adalah dari *node* 3 ke *node* 18.

5. Iterasi 4
   W=min{18,17} = 18
   L={1,2,3,18}
   d[17]=min(d[17],d[18]+c[18,17])=min(∞,2.2+0.67)
   = 2.87
   Node selanjutnya yang terhubung dari *node* 18 adalah *node* 17. Karena *node* 17 merupakan *node* tujuan maka iterasi diakhiri.

6. Iterasi 5
   W=min{17} = 17
   L={1,2,3,18,17}
   Hasil akhir didapat bahwa rute terpendek dari *node* 1 ke *node* 17 adalah : 1 – 2 – 3 – 18 – 17.

B. Tampilan Layar Rute

Salah satu tampilan layar pada aplikasi ini adalah Tampilan Layar Rute yang berisi pilihan menu untuk melihat daftar rute dari *node* asal yaitu Kantor Pusat PDAM TKR ke beberapa *node* tujuan dimana water meter pelanggan dipasang. Untuk melihat rute terpendek pada tiap-tiap rute didapat dengan mengklik menu Lihat Rute.

Gbr 7 : Tampilan Layar Rute

C. *Tampilan Layar Hasil Pencarian Algoritma Dijkstra*

Gbr 8. Tampilan Layar Hasil Pencarian Algoritma *Dijkstra*

Pada akhir aplikasi akan tampil *node-node* jalur terpendek pada rute baca meter sebagai hasil dari perhitungan algoritma *Dijkstra*. Selain itu akan ditampilkan pula data jarak terpendek dalam satuan Kilometer (Km) yang dihitung dari *node* asal sampai *node* tujuan.

V. PENUTUP

A. *Kesimpulan*

Dari hasil coba aplikasi rute baca meter dapat ditarik kesimpulan yaitu :

1. Aplikasi pembacaan water meter sangat diperlukan dalam rangka mendukung pekerjaan yang dilakukan oleh PDAM Tirta Kerta Raharja Kabupaten Tangerang dimana aplikasi ini dapat memudahkan bagi Petugas Pembaca Watermeter dalam menentukan rute perjalanan pembacaan secara cepat, efektif dan efisien dengan dukungan teknologi informasi.
2. Algoritma Dijkstra dapat diimplementasikan untuk pencarian rute, dalam proses pencarian rute tersebut terdapat faktor yang mempengaruhi diantaranya jarak dan koordinat.
3. Hasil pencarian rute dengan memanfaatkan alat komunikasi seperti handphone akan mempermudah user dengan menunjukan rute yang dikehendaki sesuai hasil pencarian yang terdapat pada data.
4. Dengan sistem informasi yang terkomputerisasi, maka tingkat keakuratan data dapat dikontrol dengan baik, waktu pengelolaan dapat efisien dan dapat memenuhi kebutuhan data yang dibutuhkan oleh manajemen secara cepat dan akurat.
5. Meskipun aplikasi yang dibuat terlihat rumit dan membutuhkan keahlian khusus, namun tetap diupayakan aplikasi ini dapat dioperasikan secara mudah oleh *administrator* maupun *user* ketika berinteraksi langsung dengan sistem.

B. *Saran*

Guna kesempurnaan aplikasi rute baca meter ini , maka dapat disampaikan saran sebagai berikut :

1. Pada suatu perusahaan, informasi merupakan hal yang terpenting dan saat ini memegang peranan utama dalam mendukung kegiatan operasional dan kelangsungan hidup perusahaan, untuk itu manajemen perlu memperhatikan dan berkomitmen untuk menerapkan sistem informasi dalam perusahaannya.
2. Aplikasi ini perlu dikembangkan lagi dikarenakan data yang dipergunakan masih dalam batas data berupa jarak, sementara masih banyak data yang sebenarnya dapat dimasukan kedalam algoritma *Dijkstra* seperti waktu dan lain-lain sehingga aplikasi ini nantinya akan lebih sempurna dan banyak memberikan informasi yang dibutuhkan.
3. Aplikasi yang dapat dikembangkan dengan menambahkan peta *digital* sehingga aplikasi lebih tambah informatif.

DAFTAR PUSTAKA

[1] http://www.arsitektur-routing.blogspot.com/ 2011/ 05/algoritma-dijkstra.html, dilihat 10 Juni 2013.
[2] Assyadiq, Hasbi 2009. Dijkstra–pencarian–jalur-terpendek, dilihat 26 Maret 2012, <http://www.asyadeeq.files.wordpress.com>.
[3] Munir, Rinaldi 2012. Matematika Diskrit, Edisi Kelima. Bandung : Informatika.
[4] http://www.w3schools.com/php/, dilihat 6 Apil 2013.
[5] Suarga, Drs., M.Sc., M.Math., Ph.D 2004. Algoritma Pemrograman, Yogyakarta : Andi.
[6] Indrajani, S.Kom, MM 2011. Perancangan Basis Data dalam All in 1, Jakarta : PT Elex Media Komputindo.