BAB II
TINJAUAN PUSTAKA

2.1 Teori Judul

2.1.1 Pengertian *Traveling Salesman Problem*

Menurut Kesy, Annisa dan Dewi, Rahmasari (2017), *Traveling Salesman Problem* (TSP) mempertimbangkan batasan kapasitas muatan dan atau waktu tempuh. TSP memiliki fungsi tujuan untuk meminimumkan jarak atau waktu atau biaya tempuh dengan batasan *salesman* hanya mengunjungi lokasi konsumen hanya satu kali.

Menurut Irham, Muhammad (2017), *Travelling Salesman Problem* adalah permasalahan dimana seorang salesman harus mengunjungi semua kota yang mana tiap kota hanya dikunjungi sekali, dan harus kembali ke kota asal.

Berdasarkan pengertian diatas, penulis menyimpulkan bahwa *Travelling Salesman Problem* adalah suatu permasalahan yang dihadapi oleh seorang kurir dimana kurir harus megunjungi semua lokasi konsumen hanya satu kali tanpa melalui tempat yang sama.

2.1.2 Pengertian Bisnis *Laundry*

Menurut Nur, Cahaya (2015), Bisnis *Laundry* adalah masuk dalam kategori bisnis yang berkelanjutan dan selalu dibutuhkan orang. Praktisnya demikian: ”Selama manusia masih mengenakan pakaian, maka selama itu manusia akan selalu membutuhkan *laundry*.”

Menurut Dwi, Yenita, dkk (2016) bisnis laundry termasuk dalam kategori bisnis dengan perputaran yang cepat. Maksudnya rentang waktu permintaan pelanggan antara permintaan pertama dan permintaan selanjutnya pada jasa ini yang memakan waktu relatif singkat.

Berdasarkan pengertian diatas, penulis menyimpulkan bahwa usaha laundry merupakan usaha berkelanjutan yang selalu dibutuhkan orang dengan perputaran yang cepat, pelanggan akan terus menggunakan jasa *laundry* selama manusia masih menggunakan pakaian.

2.1.3 Pengertian Algoritma *Dijkstra*

Menurut Ramadhani (2019:125), Algoritma *Dijkstra* merupakan salah satu tipe dari algoritma rakus (*greedy algorithm*). Algoritma ini digunakan untuk memecahkan masalah pencarian lintasan terpendek pada *Single Source Shortest Path* (3SP). Pada algoritma *Dijkstra*, proses pencarian dilakukan satu per satu untuk semua *vertex* sehingga ditemukan lintasan terpendek untuk *vertex* yang diinginkan.

2.1.4 Pengertian *Android*

Menurut Yudhanto (2018:1), *Android* adalah sistem operasi berbasis *Linux* yang dirancang untuk perangkat bergerak layar sentuh seperti telepon pintar dan *computer tablet. Android* awalnya dikembangkan oleh *android* Inc., dengan dukungan finansial dari Google, yang kemudian membelinya pada tahun 2005.

Menurut Safaat, Nazruddin (2015), Android adalah sebuah sistem operasi untuk perangkat mobile berbasis linux yang mencakup sistem operasi, *middleware* dan aplikasi.

Berdasarkan pengertian diatas, penulis menyimpulkan bahwa *Android* adalah suatu sistem operasi berbasis linux yang dirancang untuk perangkat layar sentuh.

2.1.5 Pengertian Penyelesaian *Travelling Salesman Problem* pada 24 *Hours Laundry*

Penyelesaian *Traveling Salesman Problem* pada 24 *Hours Laundry* merupakan suatu usaha untuk menyelesaikan permasalahan yang dihadapi oleh kurir, dimana kurir harus mengantar barang ke semua tempat dan setiap tempat hanya dikunjungi satu kali, sedangkan kurir kurang mengetahui rute terpendek dalam perjalanan. Sehingga diperlukan suatu cara mencari rute terpendek dari perjalan kurir. Solusi optimal dari permasalahan *Traveling Salesman Problem* ini akan sangat membantu perusahaan 24 *Hours Laundry* dalam melakukan pengembalian barang baik dari segi waktu maupun dana.

2.2 Teori Khusus

2.2.1 *Object Oriented Program* (OOP)

Menurut Abdullah (2017), OOP (*Object Oriented Program)* merupakan teknik pemograman dengan menggunakan konsep objek. Tujuan dari OOP adalah untuk memudahkan programmer dalam pembuatan program dengan menggunakan konsep objek yang ada dalamm kehidupan sehari-hari. Jadi setiap bagian permasalahan adalah objek, dan objek itu sendiri merupakan gabungan dari beberapa objek yang lebih kecil.

Sebuah objek pada OOP memiliki data atau disebut property yang menjelaskan tentang sifat-sifat objek tersebut. Seperti sebuah handpone dapat memiliki data warna, merk, ukuran layar, dan sebagainnya. Begitu juga dengan objek-objek yang ada didalamnya seperti layar memiliki data berupa lebar, tinggi dan sebagainnya.

Selain memiliki property, sebuah objek dalam OOP memiliki method berupa fungsi yang dapat dipanggil untuk melakukan tindakan atau merubah nilai dari property yang ada di dalamnya. Seperti handphone dapat melakukan tindakan merekam, restrart, memanggil, mengirim pesan dan sebagainnya. Handphone juga dapat diganti *casing* untuk mengubah warnanya (mengubah nilai property).

2.2.2 *Unified Modelling Language* (UML)

Menurut (Salahuddin, 2014), pada perkembangan teknologi perangkat lunak, diperlukan adanya bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standarisasi agar orang di berbagai negara dapat mengerti pemodelan perangkat lunak.

Pada perkembangan perkembangan teknik pemograman berorientasi objek, muncullah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemodelan berorientasi objek, yaitu *Unified Modeling Language* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak.

UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung.

2.2.3 *Class Diagram*

Menurut (Salahuddin, 2014), Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebuat atribut dan metode atau operasi.

Diagram kelas dibuat agar pembuat program atau *programmer* membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron.

Tabel 2.1 Simbol *Class Diagram*

No	Simbol	Deskripsi
1	Kelas nama_kelas + atribut + operasi()	Kelas pada struktur sistem.
2	Antarmuka / *interface* name_interface	sama dengan konsep *interface* dalam pemograman berorintasi objek.
3	Asosiasi / *association*	Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
4	Asosiasi berarah / *directed association*	Relasi antarkelas dengan makna kelas yang saat digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*.
5	Generalisasi	Relasi antarkelas dengan makna generalisasi-spesialisasi (umum khusus).

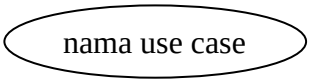
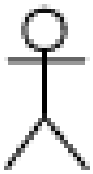
Lanjutan **Tabel 2.1**

No	Simbol	Deskripsi
6	Kebergantungan / *dependency* 	Relasi antarkelas dengan makna kebergantungan antar kelas.
7	Agregasi / *aggregation* 	Relasi antarkelas dengan makna semua-bagian (*whole-part)*

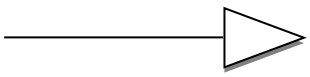
2.2.4 *Use Case Diagram*

Menurut (Salahuddin, 2014), *Use case* atau diagram *use case* merupakan pemodelan untuk kelakukan (*behavior)* sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor denagan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Tabel 2.2 Simbol *Use Case Diagram*

No	Simbol	Deskripsi
1	*Use case* 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan anta unit atau aktor; dinyatakan dengan kata kerja di awal fase nama *use case.*
2	Aktor / *actor* 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem sistem informasi yang akan dibuat itu sendiri, biasanya dinyatakan menggunkanan kata benda di awal fase nama aktor.
3	Asosiasi / *association*	Komunikasi antara aktor dan *use case* yang berpatisipasi pada *use case* atau *use case* memiliki interkasi dengan aktor.

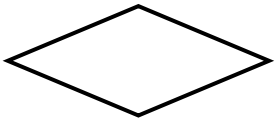
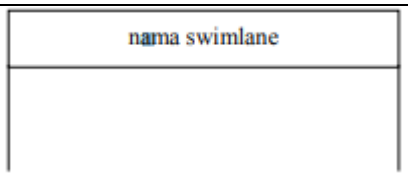
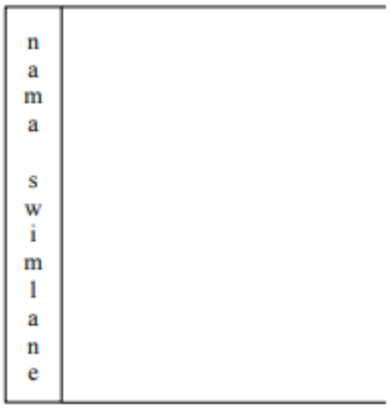
Lanjutan **Tabel 2.2**

No	Simbol	Deskripsi
4	Ekstensi / *extend* < < extend> > 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu; mirip dengan prinsip *ingeritance* pada pemograman berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case*.
5	Generalisasi / *generalization* 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
6	Menggunakan / *include / uses* < < include > > 	Relasi *use case* tambahan ke sebuah *use case* di mana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini.

2.2.5 *Activity Diagram*

A.S Rosa dan M. Shalahuddin (2015:161), "Diagram aktivitas atau *activity* diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak". Berikut adalah simbol-simbol yang ada pada diagram aktivitas menurut Rosa dan Shalahuddin (2015:162):

Tabel 2.3 Simbol Activity Diagram

No	Simbol	Deskripsi
1	Status Awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
2	Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
3	Percabangan / *decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
4	State	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
5	Status Akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
6	 Atau 	Swimlane memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

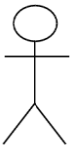
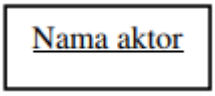
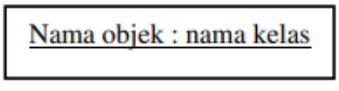
2.2.6 *Sequence Diagram*

A.S Rosa dan M. Shalahuddin (2015:165), "*Sequence diagram* atau diagram sekuen menggambarkan kelakuan objek pada use case dengan mendeskripsikan waktu hidup objek dan message yang dikirim dan diterima antar objek".

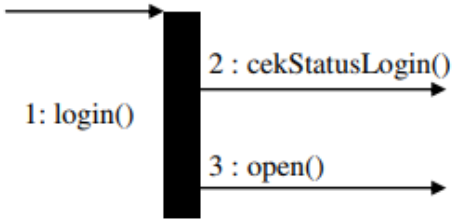
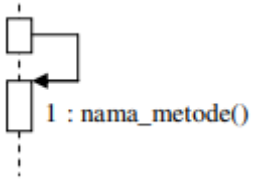
Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian use case yang memiliki proses sendiri atau yang penting semua use case yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak use case yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak.

Berikut adalah simbol-simbol yang ada pada diagram sekuen menurut Rosa dan Shalahuddin (2015:146):

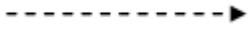
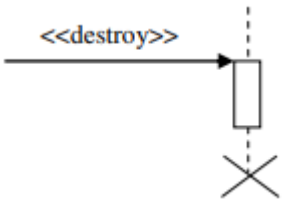
Tabel 2.4 Simbol *Sequence Diagram*

No.	Simbol	Keterangan
1.	Aktor  nama aktor atau  tanpa waktu aktif	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya dinyatakan dalam menggunakan kata benda diawal frase nama aktor.
2.	Garis Hidup / lifeline 	Menyatakan kehidupan suatu objek.
3.	Objek  	Menyatakan objek yang berinteraksi pesan.

Lanjutan **Tabel 2.4**

No	Simbol	Deskripsi
4.	Waktu Aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semuanya yang terhubungdengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya, misalnya  Maka cekStatusLogin() dan open() dilakukan didalam metode login(). Aktor tidak memiliki waktu aktif.
5.	Pesan tipe *create* <<create>> 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat
6.	Pesan tipe *call* 1 : nama_metode() 	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri,  Arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil

Lanjutan **Tabel 2.4**

No	Simbol	Deskripsi
		operasi/metode maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi.
7.	Pesan tipe *send* 1 : masukkan 	Menyatakan bahwa suatu objek mengirimkan data/masukkan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim.
8.	Pesan tipe *return* 1 : keluaran 	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.
9.	Pesan tipe *destroy* 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaliknya jika ada create maka ada destroy.

2.2.7 Pengertian Kamus Data

A.S Rosa dan M. Shalahuddin (2015:73), "Kamus data adalah kumpulan daftar elemen data yang yang mengalir pada sistem perangkat lunak sehingga masukan *(input)* dan keluaran *(output)* dapat dipahami secara umum (memiliki standar cara penulisan)."

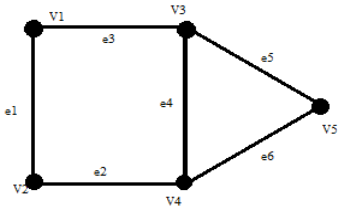
A.S Rosa dan M. Shalahuddin (2015:73) menjelaskan, bahwa kamus data memiliki beberapa simbol untuk menjelaskan informasi tambahan, yaitu sebagai berikut:

Tabel 2.5 Simbol dalam Kamus Data

No	Notasi	Arti
1.	=	disusun atau terdiri dari
2.	+	*Dan*
3.	[|]	baik… atau…
4.	{ }^{n}	n kali diulang/bernilai banyak
5.	( )	data operasional
6.	*…*	Batas komentar

2.2.8 Teori *Graph*

Menurut Ramadhani (2019:2), Teori *Graph* merupakan salah satu cabang ilmu matematika yang mempresentasikan objek-objek diskret dan hubungan antara objek-objek tersebut. Setiap objek yang dibuat akan direpresentasikan sebagai sebuah simpul (*vertex*), sedangkan hubungan antara objek direpresentasikan sebagai sebuah garus (*edge*).



Gambar 2.1 Contoh *Graph*

2.3 Teori Program

2.3.1 Pengertian *Android Studio*

Aziz, Abdul (2018), *Android Studio* merupakan perangkat lunak buatan *Google* untuk para *developer android* dalam membuat dan mengembangkan aplikasi *android*. *Android Studio* menawarkan banyak fitur yang memungkinkan alur kerja pengembangan menjadi lebih mudah dan menyenangkan dalam satu set.

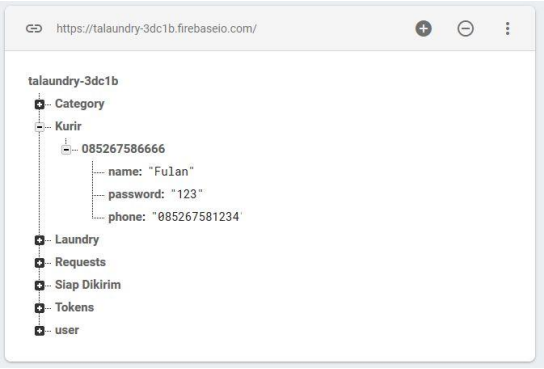
2.3.2 Pengertian *Java*

Nofriadi (2018), Bahasa pemrograman *Java* merupakan salah satu dari sekian banyak bahasa pemrograman yang dapat dijalankan di berbagai sistem operasi termasuk telepon genggam. Bahasa pemrograman ini merupakan pengembangan dari bahasa pemrograman C++ karena banyak mengadopsi sintak C dan C++. Secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi atau pun aplikasi berbasis *web*.

2.3.3 Pengertian *Firebase Realtime Database*

Kumar, Ashok (2018), Pada bulan April 2012 James Tamplin dan Andrew Lee meluncurkan *Firebase*. pada tahap awal, *firebase* menyediakan platform BaaS (Backend as a Service) yang dominan api yang terus meningkatkan pengalaman cloud dengan memperkenalkan fitur dan fungsi baru. *Firebase* adalah satu-satunya penyedia dengan fungsi basis data *autosyncing*, itu memungkinkan untuk menumbuhkan aplikasi yang hebat, mengembangkan pekerjaan Anda jauh lebih unggul bersama.

Firebase realtime database memberikan bahasa aturan basis ekspresi yang dapat diadaptasi, yang disebut aturan keamanan *database realtime firebase*, basis data *realtime firebase* mendukung *android, iOS, web* dan lainnya. Semua data disimpan dalam format JSON dan setiap perubahan dalam data tercermin segera dengan melakukan operasi sinkronisasi di setiap platform.



Gambar 2.2 Struktur Data *Firebase Realtime Database*

2.3.4 Pengertian *SQLite*

Siahaan, Vivian dan R.H. Sianipar (2019). SQLite merupakan mesin database yang mudah digunakan. Secara mendasar, SQLite merupakan database ringan yang dikhususkan untuk aplikasi-aplikasi berukuran kecil yang dapat disimpan pada suatu file disk. Mesin database ini sangat popular yang digunakan pada telepon seluler, tablet, peralatan dan instrument elektronis. SQLite tidak memerlukan proses server terpisah, dan tidak memerlukan konfigurasi apapun.

2.3.5 Pengertian *Google Maps* API

Hanief, Shofwan dan Dian, Pramana (2018), *Google* meluncurkan *Google Maps* API pada bulan Juni 2005 untuk memungkinkan pengembang untuk mengintegrasikan *Google Maps* ke situs *web* mereka. *Google maps* memiliki berbagai macam API yang membiarkan pengguna memasukkan fungsi dan tampilan data.

2.3.6 Pengertian Database

Menurut Abdulloh (2018:103), Database atau basis data adalah kumpulan informasi yang disimpan dalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi.

Menurut Enterprise (2017:1), Database adalah suatu aplikasi yang menyimpan sekumpulan data. Setiap database mempunyai perintah tertentu untuk membuat, mengakses, mengatur, mencari, dan menyalin data yang ada di dalamnya.

Maka dari beberapa pendapat dapat disimpulkan bahwa database atau basis data adalah suatu aplikasi yang berisi sekumpulan informasi yang tersimpan secara sistematik di komputer.

Terdapat istilah-istilah dalam basis data, yang didefinisikan sebagai berikut: (Bin Ladjamudin, 2015:132)

1. BIT

Bit merupsksn bagian data yang terkeci; yang bisa diwakili dengan numeric, symbol khusus, gambar-gambar dan alphabetis.

2. Byte

Byte adalah kumpulan dari pada bit-bit yang sejenis. Satu byte identik dengan satu karakter

3. Field

Field merupakan sekumpulan byte-byte yang sejenis akan membentuk suatu field.

4. Atribut

Atribut merupakan relasi fungsional dari satu object set ke object set yang lain. Tiap tipe entitas memiliki sekumpulan atribut yang berkaitan dengannya. Dibawah ini diberikan contoh beberapa tipe entitas beserta atributnya:

a. MAHASISWA : NPM, NAMA, ALAMAT, RT, RW, KOTA, KODEPOS
b. MOBIL : NO_MOBIL, WARNA, JENIS, CC
c. PEGAWAI : NIP, NAMA, ALAMAY, KEAHLIAN

5. Tuple/ Record

Dalam basis data istilah yang lebih tepat untuk menyatakan suatu baris data dalam suatu relasi adalah tuple, sebenarnya pengertian tuple bisa diidentikkan dengan record. Tuple terdiri dari kumpulan atribut-atribut dan atribut atribut tersebut saling berkaitan dalam menginformasikan tentang suatu entitas/relasi secara lengkap. Contoh : terdapat suatu relasi/ table mahasiswa dengan struktur dibawah ini.

Tabel 2.6 Contoh Tuple/Record

NIM	NAMA	ALAMAT
9455500001	ABDULLAH	BENDA RAYA NO.4
9455500002	AMINAH	CILEDUG TENGAH NO.4
9455500003	BUDIMAN	HALIMUN NO.7

Dari relasi/table diatas maka :

9455500001 ABDULLAH BENDA RAYA NO.4

Adalah satu tuple/record.

6. Entitas/ File

File merupakan kumpulan dari record-record yang sejenis dan mempunyai elemen yang sama, atribut yang sama, namun berbeda-beda data dan valuenya. Database terbentuk dari kumpulan file. File dalam pemrosesan aplikasi di kategorikan sebagai berikut :

a. File Induk (Master File)

File induk merupakan file yang penting dalam sistem dan akan tetap ada selama siklus berputar. File master ini dibedakan menjadi 2 macam yaitu :

1) File induk acuan (reference master file), yaitu file induk yang recordnya relatif statis, jarang berubah nilainya.
2) File induk dinamik (dinamic master file), yaitu file induk yang nilai dari recordrecordnya sering berubah atau sering dimutakhirkan (update) sebagai akibat dari suatu transaksi.

b. File Transaksi (Transaction File)

File transaksi disebut juga dengan nama file input (input file). File ini digunakan untuk merekam data hasil dari suatu transaksi yang terjadi.

c. File Laporan (Report File)

File laporan adalah file yang berisi dengan informasi yang akan ditampilkan. Biasanya struktur dari file laporan ada beberapa macam. Hal ini akan disesuaikan dengan kepada siapa saja laporan tersebut didistribusikan.

d. File Sejarah (History File)

File yang berisi dengan data masa lalu yang sudah tidak aktif lagi, tetapi perlu disimpan untuk mencari data yang hilang

e. File Pelindung (Back up File)

File pelindung merupakan salinan dari file-file yang masih aktif di database pada suatu saat tertentu dan digunakan sebagai cadangan atau pelindung bila file database yang aktif rusak atau hilang.

f. File Kerja (Working File)

File kerja dibuat oleh suatu proses program secara sementara karena memori komputer tidak mencukupi, atau untuk menghemat pemakaian memori selama proses, dan akan dihapus bila proses telah selesai.

7. Domain

Domain adalah kumpulan dari nilai-nilai yang diperbolehkan untuk berada dalam satu atau lebih atribut. Setiap atribut dalam suatu basis data relasional didefinisikan sebagai suatu domain. Contoh :

Tabel 2.7 Contoh Domain

Atribut	Nama Domain
N_Cab	Nomor_Cabang
Jalan	Nama_Jalan
Wilayah	Nama_Wilayah
K_Pos	Kode_Pos
Tg.L	Tanggal_Lahir
Kota	Nama_Kota
No_Tel	Nomor_Tel

8. Kunci Elemen Data (Key)

Key adalah elemen record yang dipakai untuk menemukan record tersebut pada waktu akses, atau bisa juga digunakan untuk mengidentifikasi setiap entity/record/baris. Jenis-jenis key, yaitu :

a. Superkey

Superkey merupakan satu atau lebih atribut (kumpulan atribut) dari suatu table yang dapat digunakan untuk mengidentifikasi entity/record dari tavel tersebut secara unik. (tidak semua atribut dapat menjadi superkey).

b. Candidate Key

Superkey dengan jumlah atribut minimal, disebut candidate key.

c. Primary Key

Salah satu atribut dari candidate key dapat dipilih/ditentukan menjadi primary key dengan tiga kriteria sebagai berikut :

1) Key tersebut lebih natural untuk digunakan sebagai acuan.
2) Key tersebut lebih sederhana.
3) Key tersebut terjamin keunikannya.

d. Foreign Key

Foreign key merupakan sembarangan atribut yang menunjuk kepada primary key pada table lain.

Komponen penting dalam sistem basis data adalah : (Yanto, 2016:13)

1. Data

   Merupakan informasi yang disimpan dalam suatu struktur tertentu yang terintegrasi.

2. Hardware

   Merupakan perangkat keras berupa komputer dengan media penyimpanan yang digunakan untuk menyimpan data karena pada umumnya basis data memiliki ukuran yang besar.

3. Sistem Operasi

   Program yang mengaktifkan dan memfungsikan sistem komputer, mengendalikan seluruh sumber daya dalam komputer, dan melakukan operasi dasar dalam komputer meliputi input, proses dan output.

4. Basis Data

   Basis data sebagai inti dari sistem basis data. Basis data menyimpan data serta struktur sistem basis data baik untuk entitas maupun objek-objek secara detail.

5. *Database Management System*

   Merupakan perangkat lunak yang digunakan untuk melakukan pengelolaan basis data.

6. User

   Merupakan Penggunaan yang menggunakan data yang tersimpan dan terkelola. User dapat berupa seseorang yang mengelola basis data yang disebut *database administrator* (DBA), bisa juga disebut end user.

7. Aplikasi Lainnya

   Program yang dibuat untuk memberikan interface kepada user sehingga lebih mudah dan terkontrol dalam mengakses basis data.

2.4 Penelitian Terdahulu

Berdasarkan penelitian oleh Harahap dan Nurul (2017), penelitian ini adalah membuat sebuah simulasi pencarian jalur terpendek dengan menggunakan Algoritma *Dijkstra* yang dapat membantu dalam pencarian jalur terpendek dengan skema yang telah ditetapkan dan digambarkan. *Shortest path* merupakan persoalan untuk mencari lintasan terpendek antara dua buah *vertex* pada *graph* berbobot yang memiliki gabungan nilai jumlah bobot pada *edge graph* yang dilalui dengan jumlah yang paling minimum. Pada proses Tentukan *Shortest Path* membutuhkan data masukan berupa data asal dan data akhir, dan kemudian proses akan mengambil data dari *database node* dan jarak. Lalu sistem akan melakukan pemeriksaan pada *database* jalur *shortest path*, bila data awal dan data akhir yang dipilih sudah pernah diproses sebelumnya, maka akan diambil hasil proses dari *database* jalur *shortest path*, apabila belum pernah, maka sistem akan memproses data tersebut, selanjutnya sistem akan menyimpan hasil perhitungan dan *query* jalur *shortest path* pada *database* jalur *shortest path*.

Menurut penelitian Irfan (2017), penelitian ini bertujuan untuk mengetahui penyelesaian masalah TSP dengan menggunakan algoritma *Hill Climbing* baik *Simple Hill Climbing* (SHC) maupun *Steepest-Ascent Hill Climbing* (SAHC). Berdasarkan hasil kajian teori yang dilakukan, dapat disimpulkan bahwa algoritma *Hill Climbing* dapat digunakan untuk menyelesaikan TSP. Algoritma *Hill Climbing* dalam menyelesaikan masalah TSP yaitu: menentukan *initial state*, melakukan pengujian panjang lintasan, melakukan kombinasi penukaran dua kota, dan kemudian melakukan pengujian terhadap nilai heuristiknya. Penyelesaian masalah TSP dengan menggunakan algoritma *Hill Climbing* masing-masing mempunyai karakteristik yang berbeda-beda. Dalam menyelesaikan masalah TSP, SHC memilih keadaan yang lebih baik tanpa melakukan pengujian dikombinasi pertukaran kota pada iterasi yang sama. Sedangkan SAHC membandingkan keadaan yang lebih baik sebelum memilih keadaan tersebut sebagai *new state*. Selanjutnya untuk membantu proses komputasi pada saat menyelesaikan TSP, dibuat sebuah program dengan model algoritma *Hill Climbing* menggunakan

MATLAB. Program yang terbentuk memberikan solusi berupa rute perjalanan yang disajikan dalam bentuk diagram.

Penelitian Prianto dan Kusnadi (2018), kurangnya informasi mengenai posisi suatu tempat dan menemukan rute terbaik untuk menuju sebuah tempat tujuan termasuk hal yang cukup rumit terutama mencari rute dalam menemukan tempat parkir. Tidak tersedianya aplikasi yang menyediakan informasi serta menavigasi pengguna parkir untuk menemukan rute terbaik secara realtime menjadi suatu masalah yang harus dicari jawabannya. Hasil yang telah dicapai dari penelitian ini adalah algoritma *Dijkstra* dapat menjawab masalah yang dihadapi yakni memberikan navigasi rute terbaik.

Menurut penelitian Nugraha, K.P., dkk, (2017), pada tugas akhir ini telah dikembangkan salah satu fitur LBS (*Location Base Service*) , yaitu *shortest path finder* atau pencari rute terpendek, dengan pembuatan suatu aplikasi berbasis *android* yang bertujuan memberikan layanan berupa informasi lokasi kepada pengguna perangkat *mobile* di kawasan kampus Telkom University. Pada sistem kerja aplikasi ini pengguna dari perangkat *mobile* akan dideteksi posisinya oleh GPS yang terintegrasi pada aplikasi ini. Selanjutnya Pengguna perangkat *mobile* menentukan tempat yang diinginkan. Setelah itu aplikasi akan memberikan rute terpendek ketempat tujuan dari posisi pengguna perangkat *mobile* yang sebelumnya telah terdeteksi oleh GPS. Hasil yang dicapai dalam tugas akhir ini terbentuknya aplikasi berbasis *android* yang dapat menentukan rute terpendek dengan algoritma *Dijkstra* dari suatu titik ke suatu titik tujuan yang telah ditentukan pada kawasan kampus Telkom University.

Yusuf, M.S., dkk, (2017), berdasarkan penelitian yang sudah dilakukan, terdapat beberapa kebutuhan fungsional yang meliputi: menampilkan peta kebun raya, mencari posisi tanaman, menampilkan rute dan menampilkan daftar tanaman. Selain itu terdapat kebutuhan non-fungsional usability dan compatibility. Fungsional yang menerapkan implementasi algortima *dijkstra* adalah menampilkan rute. Persimpangan yang ada di kebun raya Purwodadi direpresentasikan sebagai *vertex* atau node. Sedangkan jarak antar persimpangan direpresentasikan sebagai edge. Sehingga dari data yang didapatkan di lapangan,

bisa dihitung jarak terpendek antara titik pengguna dan titik tujuan atau tanaman. Hasil implementasi algoritma *dijkstra* dalam menentukan jarak terpendek dari lokasi pengguna ke tanaman yang di tuju menghasilkan nilai usability sebesar 70.916%, yang diambil dari 30 responden. Nilai tersebut berada di rentang interval 60% - 80% yang dapat dikategorikan sebagai aplikasi yang memuaskan.