BAB II

LANDASAN TEORI

2.1. Konsep Dasar Sistem

2.1.1. Sistem Informasi

Menurut (Anggraeni, 2015) Sistem adalah “kumpulan orang yang saling bekerja sama dengan ketentuan- ketentuan aturan yang sistematis dan terstruktur untuk membentuk satu kesatuan yang melaksanakan suatu fungsi untuk mencapai tujuan .”

Menurut (Anggraeni, 2015) Informasi adalah “hasil dari pengolahan data dalam suatu bentuk yang lebih berguna dan lebih berarti bagi penerimanya yang menggambarkan suatu kejadian-kejadian yang nyata yang digunakan untuk pengambilan keputusan.”

Menurut (Anggraeni, 2015) Sistem informasi adalah “suatu kombinasi teratur dari orang-orang hardware, software, jaringan komunikasi dan sumber daya data yang mengumpulkan, mengubah, dan menyebarkan informasi dalam sebuah organisasi.”

2.1.2. Sistem Berorientasi Objek/ OOP (Object Oriented Programming)

Pengertian pemrograman berorientasi objek atau object-oriented programming Menurut (Kadek, 2015) adalah :

Pemrograman berorientasi objek atau object-oriented programming merupakan suatu pendekatan pemrograman yang menggunakan object dan class. Saat ini konsep OOP sudah semakin berkembang. Hampir semua programmer maupun pengengmbang aplikasi menerapkan konsep OOP. OOP bukanlah sekedar cara penulisan sintaks program yang berbeda, namun lebih dari itu, OOP merupakan cara pandang dalam menganalisa sistem dan permasalahan pemrograman. Dalam OOP, setiap bagian dari program adalah object. Sebuah object mewakili suatu bagian program yang akan diselesaikan.

Beberapa konsep OOP dasar menurut (Kadek, 2015), antara lain:

a. Encapsulation (Class dan Object)
b. Inheritance (Penurunan sifat)
c. Polymorphisme PHP

2.1.3. Basis Data

Menurut Rosa dan Shalahuddin (2015:43) “basis data merupakan salah satu bagian dalam rekayasa perangkat lunak yang terkomputerisasi dan bertujuan utama memelihara data yang sudah diolah atau media penyimpanan informasi agar dapat diakses dengan mudah dan cepat”. Sedangkan menurut Yakub dan Hisbanarto (2015:25) menjelaskan, “basis data (database) merupakan kumpulan data yang saling berhubungan atau punya relasi”.

Dapat disimpulkan bahwa basis data bagian dari rekayasa perangkat lunak yang terkomputerisasi sebagai media penyimpanan informasi yang saling berhubungan atau punya relasi untuk penyimpanan data informasi agar dapat diakses dengan mudah dan cepat.

Menurut Rosa dan Shalahuddin (2015:46) SQL (Structured Quety Laguage) adalah “bahasa yang digunakan untuk mengolola data pada RDBMS (Relational DBMS) yang dikembangkan berdasarkan teori aljabar relasional dan kalkulus”. Sedangkan menurut Manurung (2015:33) “SQL (Structured Quety Laguage) merupakan bahasa pemograman yang dirancang untuk mengelola data dalam database management system DBMS”. Berdasarkan pendapat ahli diatas dapat disimpulkan bahwa SQL (Structured Quety Laguage) merupakan bahasa komputer standar untuk berkomunikasi dengan basis data dan dikembangkan berdasarkan teori aljabar yang terstruktur digunakan untuk menelola RDBMS maupun sebuah alat pengaksesan data yang tersimpan dalam database.

2.2. Teori Pendukung (Tool System)

2.2.1. ERD (Entity Relationship Diagram)

(Rosa dan Shalahuddin 2015:50) mengemukakan bahwa: “ERD (Entity Relationship Diagram) merupakan konseptual yang mendeskripsikan hubungan antara penyimpanan penyimpanan basis data menggunakan OODBMS (Object

Oriented Database Management System)". Pemodelan awal basis data yang paling banyak digunakan adalah menggunakan Entity Relationship Diagram (ERD). ERD dikembangkan berdasarkan teori himpunan dalam dalam bidang matematika. ERD digunakan untuk pemodelan basis data relational, sehingga penyimpanan menggunakan OODBMS (Object Oriented DBMS).

Notasi-notasi simbolik didalam *Diagram Entity Relationship* yang dapat kita gunakan adalah:

1. Persegi panjang, menyatakan Himpunan Entitas.
2. Lingkaran/ Elips, menyatakan Atribut (atribut yang berfungsi sebagai *key* digaris bawahi).
3. Belah Ketupat, menyatakan Himpunan Relasi.
4. Garis, sebagai penghubung antara himpunan entitas dengan atributnya.
5. Kardinalitas Relasi dapat dinyatakan dengan banyaknya garis cabang atau dengan pemakaian angka (1 dan 1 untuk relasi satu-ke-satu, dan N untuk relasi satu-ke-banyak atau N dan N untuk relasi banyak-ke-banyak).

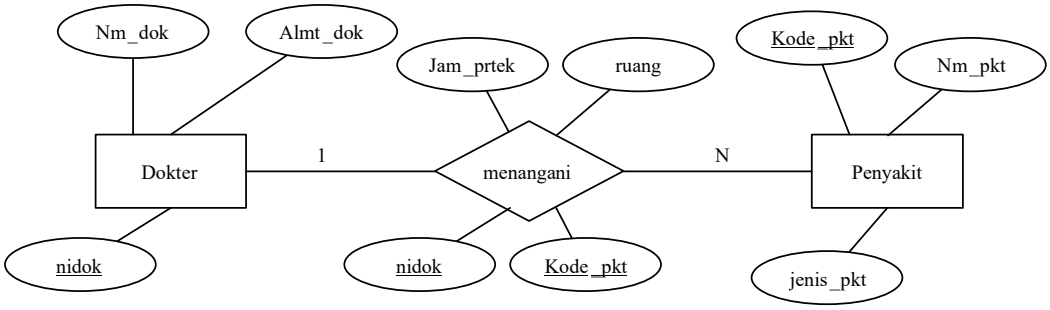
Berikut adalah contoh penggambaran relasi antar himpunan entitas lengkap dengan kardinalitas relasi dan atribut-atributnya:

1. Relasi satu-ke-satu (one-to-one)

Sumber: Priyadi (2015)

Gambar II.1. Diagram E-R Relasi satu-ke-satu

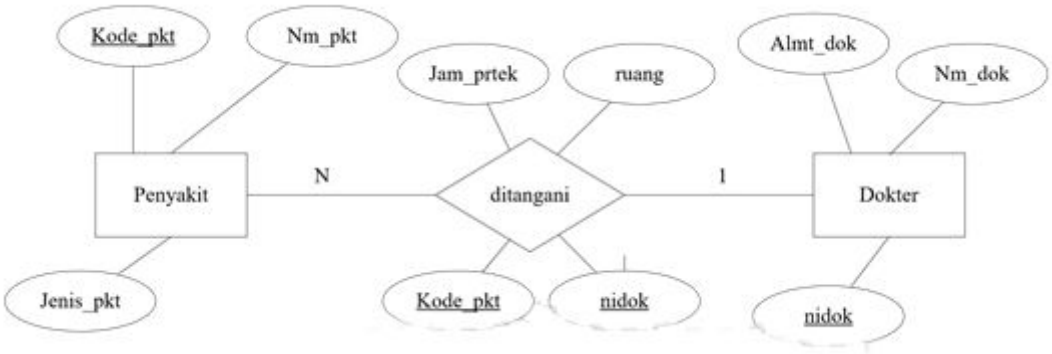
2. Relasi satu-ke-banyak (one-to-many)



Sumber: Priyadi (2015)

Gambar II.2. Diagram E-R Relasi satu-ke-banyak

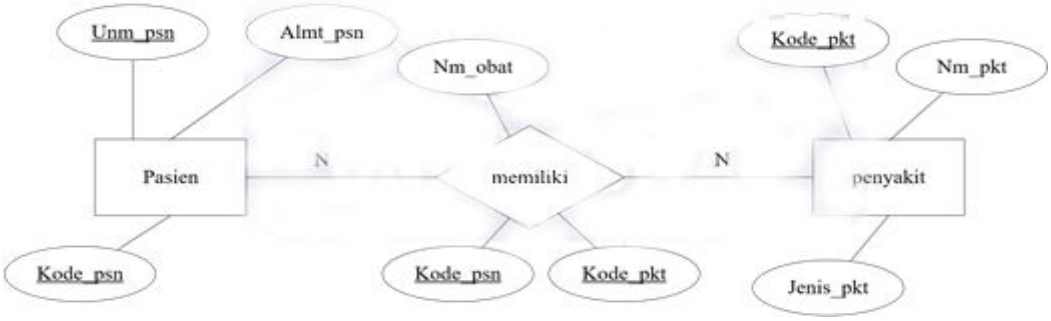
3. Relasi banyak-ke-satu (many-to-one)



Sumber: Priyadi (2015)

Gambar II.3. Diagram E-R Relasi banyak-ke-satu

4. Relasi banyak-ke-banyak (many-to-many)



Sumber: Priyadi (2015)

Gambar II.4. Diagram E-R Relasi banyak-ke-banyak

2.2.2. Logical Record Structure (LRS)

Menurut Hasugian dan Shidiq dalam (Puspitasari, 2016) mengartikan bahwa:

LRS adalah Sebuah model sistem yang digambarkan dengan sebuah diagram-ER akan mengikuti pola/ aturan pemodelan tertentu dalam kaitannya dengan konversi ke LRS. Setiap entitas akan diubah kebentuk kotak, sebuah atribut relasi disatukan dalam sebuah kotak bersama entitas jika hubungan yang terjadi pada diagram-ER 1:M (relasi bersatu dengan cardinality M) atau tingkat hubungan 1:1 (relasi bersatu dengan cardinality yang paling membutuhkan referensi) Sebuah relasi dipisah dalam sebuah kotak tersendiri (menjadi entitas baru) jika tingkat hubungannya M:M (many to many) dan memiliki foreign key sebagai primary key yang diambil dari kedua entitas yang sebelumnya saling berhubungan.

2.2.3. UML (Unified Modelling Language)

UML (*Unified Modelling Language*) adalah standarisasi internasional untuk notasi dalam bentuk grafik. Menurut (Wakhid, 2017) mengatakan bahwa: "Unified Modelling Language (UML) adalah salah satu standard bahasa yang banyak digunakan di dunia industri untuk mendefinisikan requirement, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek".

Menurut (Hendini, 2016) *Unified Modelling Language* (UML) bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak. UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem".

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML

tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C. Beberapa diagram pada UML adalah:

1. Use Case Diagram

Use-case diagram merupakan model diagram UML yang digunakan untuk menggambarkan requirement fungsional yang diharapkan dari sebuah sistem. Usecase diagram adalah diagram usecase yang digunakan untuk menggambarkan secara ringkas siapa yang menggunakan sistem dan apa saja yang bisa dilakukannya. Use case class digunakan untuk memodelkan dan menyatakan unit fungsi/layanan yang disediakan oleh sistem (or bagian sistem: subsistem atau class) ke pemakai. Diagram use case tidak menjelaskan secara detail tentang penggunaan usecase, namun hanya memberi gambaran singkat hubungan antara usecase, aktor, dan sistem.

Melalui diagram usecase dapat diketahui fungsi-fungsi apa saja yang ada pada sistem (Rosa-Salahudin, 2015). Use case bekerja dengan cara mendeskripsikan tipe interaksi antara user sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Use Case Diagram kerap digunakan untuk mendokumentasikan dan menjelaskan proses-proses yang berlangsung di dalam sebuah sistem. Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Use case Diagram, adalah gambaran efek fungsionalitas yang diharapkan oleh sistem.

Tujuan Use Case

1. Memetakkan kebutuhan sistem
2. Merepresentasikan interaksi pengguna terhadap sistem
3. Untuk mengetahui kebutuhan diluar sistem

Deskripsi Use Case

1. Diagram use case merupakan pemodelan untuk menggambarkan kelakuan (behavior) sistem yang akan dibuat.

2. Diagram use case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat.
3. Diagram use case digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Yang ditekankan pada diagram ini adalah "apa" yang diperbuat sistem, dan bukan "bagaimana".
4. Sebuah use case merepresentasikan sebuah interaksi antara aktor (user atau sistem lainya) dengan sistem.
5. Use case menjelaskan secara sederhana fungsi sistem dari sudut pandang user.

2. Use Case Diagram

Menurut Rosa dan Shalahuddin (2015: 155) Use case atau diagram use case merupakan pemodelan untuk kelakuan (behavior) sistem informasi yang akan dibuat. Syarat penamaan pada use case adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada use case yaitu pendefinisian apa yang disebut Aktor dan use case.

Use Case Diagram memiliki tiga area pokok :

1. Nama (dan stereotype)
2. Atribut
3. Metoda

Hubungan antar class diantaranya adalah:

1. Asosiasi, yaitu hubungan statis antar class. Umumnya menggambarkan class yang memiliki atribut berupa class lain, atau class yang harus mengetahui eksistensi class lain. Panah navigability menunjukkan arah query antar class.
2. Agregasi, yaitu hubungan yang menyatakan bagian ("terdiri atas.."). generalisasi.
3. Pewarisan yaitu hubungan hirarkis antar *class. Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah

4. Hubungan dinamis, yaitu rangkaian pesan (message) yang di-passing dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.

3. Activity Diagram

Diagram aktivitas atau activity diagram menggambarkan workflow (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak (Rosa dan Shalahuddin, 2015: 161).

Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. *Activity diagram* dapat dibagi menjadi beberapa object swimlane untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

4. Sequence Diagram

Sequence diagram menurut Rosa dan Shalahuddin (2015: 165) diagram sekuen menggambarkan kelakuan objek pada use case dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek.

Sequence diagram biasa digunakan untuk menggambarkan scenario atau rangkaian langkah – langkah yang dilakukan sebagai respon dari sebuah event untuk menghasilkan output tertentu.