BAB 2

LANDASAN TEORI

1.1 Teori Umum.

Teori umum yang akan dibahas oleh penulis pada sub bab ini meliputi Data, *Database*, *Database Management System* (DBMS), Keuntungan dan Kerugian *Database Management System* (DBMS), Model Relasional, *Database Language, Database Design, Entity Relationship Modelling,* Normalisasi, MySQL, *Unified Modifying Language* (UML), *Hypertext Preprocessor* (PHP), PHP *MyAdmin*, *Adobe Dreamweaver,* Internet, *World Wide Web* (WWW), *Browser.*

2.1.1 Data.

Menurut Connolly dan Begg (2005,p20), data adalaha sekumpulan komponen terpenting dari *Database Management System* (DBMS) yang berasal dari sudut pandang *end-users.*

2.1.2 Database.

Menurut Connolly dan Begg (2005, p15), *database* adalah sekumpulan data logikal yang saling terkait dan dirancang untuk memperoleh informasi yang dibutuhkan oleh organisasi atau perusahaan.

2.1.3 *Database Management System* (DBMS).

Menurut Connolly and Begg (2005, p16), *Database Management System* (DBMS) adalah suatu sistem *software* yang memugkinkan *user* untuk mendefinisikan, membuat, memelihara, dan mengatur akses ke *database.*

2.1.4 Keuntungan dan Kerugian *Database Management System* (DBMS).

Menurut Connolly dan Begg (2005, pp26-30), keuntungan dan kerugian Database Management System adalah :

Keuntungan DBMS adalah sebagai berikut:

1. Kontrol atas perulangan data (*data redundancy*).
2. Konsistensi data.
3. Banyaknya informasi dari data yang sama.
4. Jumlah data dapat banyak.
5. Membagikan data (*data sharing*)
6. Meningkatkan integritas data.
7. Meningkatkan keamanan data.
8. Penetapan standarisasi pelaksanaan.
9. Mengurangi biaya.
10. Menyeimbangkan konflik dari kebutuhan yang ada.
11. Meningkatkan aksesbilitas dan responsif data.
12. Meningkatkan produktivitas.
13. Meningkatkan pemeliharaan melalui independensi data.
14. Meningkatkan keakuratan (*concurrency*).
15. Meningkatkan layanan *backup* dan *recovery*.

Sedangkan kerugian dari DBMS adalah sebagai berikut :

a. Kompleksitas.
b. Ukuran.
c. Biaya dari penggunaan DBMS.
d. Biaya konversi.
e. Kinerja.
f. Dampak yang tinggi dari kegagalan.

2.1.5 Model Relasional.

Menurut Connolly dan Begg (2005, p74), model relasional adalah kumpulan relasi yang telah dinormalisasi dengan memiliki nama-nama relasi yang berbeda.

2.1.6 *Database Languages*.

2.1.6.1 *Data Definition Language* (DDL).

Menurut Connolly dan Begg (2005, p40), *Data Definition Language* (DDL) adalah suatu bahasa yang memungkinkan *Database Administrator* (DBA) atau pengguna untuk mendefinisikan dan member nama entitas, atribut, dan relasi yang dibutuhkan oleh aplikasi, dengan beberapa kesatuan asosiasi integritas dan batasan keamanan.

2.1.6.2 *Data Manipulation Language* (DML).

Menurut Connolly dan Begg (2005, pp41-42), *Data Manipulation Language* (DML) adalah bahasa yang menyediakan sekumpulan operasi untuk mendukung operasi-operasi dasar dalam memanipulasi data yang ada dalam *database*.

Biasanya operasi manipulasi data yang digunakan adalah :

1. Menyisipkan data baru kedalam *database*.
2. Memodifikasi atau mengubah data yang telah disimpan dalam *database*.
3. Mengambil data yang terdapat dalam *database*.
4. Menghapus data yang terdapat dalam *database*.

Menurut Connolly dan Begg (2005, pp41-42), *Data Manipulation Language* (DML) dapat dibagi menjadi 2 tipe, yaitu :

1. Procedural DML.

   Procedural DML adalah sebuah bahasa yang memungkinkan pengguna mengatur sistem untuk menentukan data yang diinginkan dan bagaimana cara mendapatkan data secara tepat.

2. Non-Procedural DML.

   Non-Procedural DML adalah sebuah bahasa yang memungkinkan pengguna untuk memperkirakan data yang diperlukan daripada bagaimana data tersebut didapat.

2.1.7 Database *Design.*

Menurut Connolly dan Begg (2005, p291-292), perancangan basis data merupakan proses pembuatan suatu desain untuk sebuah basis data yang akan mendukung operasional dan sasaran suatu perusahaan. Ada 2 pendekatan untuk merancang sebuah basis data, yaitu :

a. *Bottom-Up.*

Dimulai pada tingkat awal dari atribut (yaitu properti dari entiti dan *relationship)* yang mana melalui analisis dari asosiasi antar atribut, dikelompokkan menjadi hubungan yang merepresentasikan jenis-jenis entitas dan hubungan antar entitas. Pendekatan ini cocok untuk merancang basis data yang sederhana dengan jumlah atribut yang tidak banyak. (Connolly dan Begg, 2005, p291)

b. *Top-Down.*

Digunakan pada basis data yang lebih kompleks, yang dimulai dengan pengembangan dari model data yang mengandung beberapa entitas dan hubungan tingkat tinggi dan kemudian memakai perbaikan *top down* berturut-turut untuk mengidentifikasi entitas, hubungan dan atribut berkaitan tingkat rendah. Pendekatan ini biasanya digambarkan melalui ER (*entity relationship)*. (Connolly dan Begg, 2005, p292)

c. *Inside Out.*

Pendekatan ini berhubungan dengan pendekatan *buttom-up*, perbedaannya adalah pendekatan ini mengidentifikasi sekumpulan entity utama

dan kemudian menyebar ke entity-entity yang lain, *relationship-relationship,* atribut-atribut yang berkaitan dengan hal-hal yang sudah diidentifikasi sebelumnya. (Connolly dan Begg, 2005, p292)

d. *Mixed Strategy.*

Pendekatan ini menggunakan pendekatan *buttom-up* dan *top-down* untuk bagian model yang berbeda sebelum akhirnya menggabungkan semua bagian. (Connolly dan Begg, 2005, p292)

Didalam perancangan basis data terbagi ke dalam tiga fase utama yang meliputi *Conceptual Databasbe Design, Logical Database Design, Physical Database Design.*

1) *Conceptual Database Design*.

Menurut Connolly dan Begg (2005, p.439), hal yang pertama dilakukan dalam membuat *conceptual design* adalah dengan membuat model data secara konseptual dari perusahaan yang bersangkutan. *Conceptual database design* seluruhnya *independent* dari implementasi seperti target DBMS *software*, program aplikasi, bahasa pemograman, atau *physical consideration* lainnya. Data tersebut merupakan informasi-informasi mengenai perusahaan. Dalam *conceptual database design*, data yang ada dikembangkan dengan representasi secara konseptual yang mencakup mengidentifikasi *entity*, *relationship*, dan *atribut* yang sangat penting dalam perancangan bisnis tersebut.

Langkah-langkah untuk membuat data model lokal yang konseptual untuk setiap *user view* dapat digambarkan sebagai berikut :

a) Penentuan Jenis *Entitas*.

Tujuannya adalah untuk mengidentifikasi dan membuat kelas-kelas dari obyek yang ada berikut penjelasannya serta menentukan *entitas* utama yang dibutuhkan. Salah satu metode untuk mengidentifikasi *entity* adalah dengan menguji spesifikasi kebutuhan dari pengguna. Dari spesifikasi ini dapat mengidentifikasikan *noun* dan *noun phrases* yang disebutkan. Selain itu juga dapat melihat objek utama seperti orang, tempat atau konsep dari ketertarikan diluar *noun* lainnya yang merupakan kualitas dari objek lain. (Connolly dan Begg,2005, p.439)

b) Penentuan Jenis Hubungan Antar *Entitas*.

Tujuannya adalah untuk mengidentifikasi hubungan-hubungan yang ada antara entitas yang telah diidentifikasikan. Dalam mengidentifikasi tipe *relasi* yang ada dengan menggunakan *Entity Relationship Diagram* (ERD), mencari batasan dari tipe *relationship*, memeriksa *fan* dan *chasm traps*, memeriksa masing masing *entity* ikut serta setidaknya dalam satu *relationship*, dan dokumentasikan tipe *relationship*. (Connolly dan Begg,2005, p.439)

c) Penentuan dan Penghubungan *Atribut* Dengan *Entitas*.

Tujuannya adalah untuk mengidentifikasikan dan menghubungkan *atribut-tribut* yang berkaitan dengan *entitas* atau tipe *relationship* yang telah sesuai. (Connolly dan Begg,2005, p.439)

d) Penentuan *Domain* Terhadap *Atribut*.

Tujuannya adalah untuk menetapkan *domain* untuk tiap-tiap *atribut* dalam model data konseptual lokal dan mendokumentasikan setiap detail dari *domain*. Suatu domain adalah suatu kelompok nilai yang

dari mana satu atau lebih atribut mengambil nilainya. (Connolly dan Begg,2005, p.439)

e) Penentuan *Atribut Candidate Key, Primary Key,*dan A*lternate Key.*

Tujuannya adalah untuk menentukan *candidate key* dan *primary key* dari kumpulan *atribut-atribut* yang telah ditentukan pada tiap *entitas*. *Candidate key* adalah satu atau lebih *atribut* dari suatu *entitas* yang dapat dijadikan *primary key*. *Primary key* adalah satu *atribut* dari suatu *entitas* yang dipakai sebagai ciri yang paling unik dari *entitas* tersebut. (Connolly dan Begg,2005, p.439)

f) Mempertimbangkan Kegunaan Dari Konsep *Enhanced Modeling* (*optional*).

Tujuannya adalah untuk mengembangkan *ER model* dengan menggunakan konsep enhanced *modeling*, seperti spesialisasi, generalisasi, penggabungan (*aggregation*) dan komposisi (*composition*). (Connolly dan Begg,2005, p.439)

g) Pemeriksaan Model Terhadap *Redudansi*.

Tujuannya adalah untuk memeriksa konsep model data apakah masih mengandung data maupun *entitas* serta *atribut* yang berulang atau tidak. Hal pertama yang dilakukan adalah memeriksa kembali hubungan-hubungan yang ada apabila terdapat suatu hubungan yang mirip. (Connolly dan Begg,2005, p.439)

h) Validasi Model Konseptual Lokal Terhadap Transaksi Pengguna.

Tujuannya untuk memastikan model konseptual local mendukung transaksi yang dibutuhkan oleh *view*. Diuji dua pendekatan untuk memastikan model data konseptual local mendukung transaksi yang dibutuhkan, dengan cara:

(1) Mendeskripsikan Transaksi-transaksi.

Memeriksa seluruh informasi (*entitas*, *relationship*, dan *atribut*) yang dibutuhkan oleh setiap transaksi yang telah disediakan oleh model, dengan mendokumentasikan setiap kebutuhan transaksi. (Connolly dan Begg,2005, p.439)

(2) Mengunakan Jalur-jalur Transaksi.

Untuk validasi model data terhadap transaksi yang dibutuhkan termasuk representasi diagram jalur yang digunakan oleh setiap transaksi langsung pada *ER diagram*. (Connolly dan Begg,2005, p.439)

i) *Review Model* Data Konseptual Dengan Pengguna.

Tujuannya untuk mengkaji ulang model data konseptual lokal dengan pengguna untuk memastikan model tersebut adalah representasi sebenarnya dari *view*. Model data konseptual ini termasuk *ER diagram* dan dokumentasi pendukung yang mendeskripsikan model data. Bila ada kejanggalan (*anomali*) dalam model data, maka harus dibuat perubahan yang sesuai yang mungkin membutuhkan pengulangan langkah-langkah sebelumnya. (Connolly dan Begg,2005, p.439)

2) *Logical Database Design*.

Menurut Connolly dan Begg (2005, p.439), *logical design* merupakan proses pembuatan model data dengan menggunakan informasi yang diperoleh dari perusahaan serta berdasarkan pada model data spesifik. Model data yang telah diperoleh dalam *conceptual database design* diubah dalam bentuk *logical* model dimana data yang ada dipengaruhi oleh model data yang menjadi tujuan *database*.

Hal ini dilakukan untuk menerjemahkan representasi konseptual ke dalam bentuk struktur *logic* dalam database yang akan dijadikan sumber informasi dalam merancang *physical database design* serta memberikan sarana yang membantu para perancang *database* dalam merancang *physical database*.

Hasil akhir dari logikal *database design* ini berupa sebuah kamus data yang berisi atribut-atribut beserta *key*-nya (*primary key, alternate key*, dan *foreign key*) dan ERD keseluruhan dengan *atribut key*-nya. Langkah-langkah untuk membuat logikal *database design* dapat digambarkan sebagai berikut :

a) Menentukan *relations* untuk *logical data model*

b) Validasi *relation* dengan normalisasi

c) Validasi *relations* terhadap *user transaction*

d) Memeriksa *integrity constraints*

e) Meninjau ulang *logical data model* dengan user

f) Gabungkan model data logikal lokal menjadi model *global*

g) Memeriksa perkembangan selanjutnya

3) *Physical Database Design*.

Menurut Connolly dan Begg (2005, p.439), *physical database design* merupakan proses pembuatan deskripsi dari suatu implementasi basis data pada

secondary storage. Hal ini mendeskripsikan *base relation*, organisasi data, dan indeks yang digunakan untuk mencapai efisiensi akses ke dalam data, dan *associated integrity constraints* yang lainnya dan *security measures*. Physical database design merupakan fase ketiga dan terakhir dalam desain basis data.

Tujuan utama dari *physical database design* adalah untuk mendeskripsikan bagaimana perancang bermaksud untuk mengimplementasikan secara fisik dari *logikal database design*. Langkah-langkah dalam pembuatan *physical database design* adalah sebagai berikut:

a) Merancang *relasi* dasar

b) Mendesain representasi dari *derived data*

c) Mendesain *Enterprise Constraint*

d) Menganalisa transaksi

e) Pemilihan DMBS (*Database Management Sistem*)

f) Penerapan *View* dalam database

g) Pembuatan *Index* setiap *entitas*

h) Estimasi kebutuhan *disk*

i) Merancang mekanisme keamanan

2.1.8 *Entity Relationship Modeling.*

Menurut Connolly dan Begg (2005, p342), *Entity Relationship Modelling* adalah salah satu model yang dapat digunakan untuk mendapatkan pengertian yang tepat dari sifat data dan bagaimana data tersebut dapat digunakan oleh perusahaan.

Entity Relationship Modelling menggunakan pendekatan top-down dalam melakukan perancangan basis data. Dengan melakukan pendekatan top-down ini, perancangan dimulai dengan mengidentifikasikan data penting yang biasanya disebut sebagai *entity* dan perancangan hubungan antar data yang direpresentasikan dalam model yang biasanya disebut *relationship*. Selain itu dibutuhkan juga keterangan tambahan seperti *atribut* dan *constraint* untuk *entities, relationship,* dan *attributes*.

a. Entity Types.

Menurut Connolly dan Begg (2005, p343), *entity types* adalah sekumpulan obyek dengan properti yang sama yang diidentifikasi oleh perusahaan dan memiliki keberadaan yang *independen*.

Entity occurrence adalah suatu obyek dari *entity type* yang diidentifikasikan secara unik (Connolly dan Begg, 2005, p345). *Entity Type* dapat diklasifikasikan menjadi 2 yaitu *strong entity* dan *weak entity*.

1) Strong Entity.

Strong Entity adalah *entity type* yang keberadaannya tidak bergantung pada *entity type* lainnya (Connolly dan Begg, 2005, p354).

2) Weak Entity.

Weak Entity merupakan kebalikan dari *strong entity* yaitu *entity type* yang keberadaannya bergantung pada *entity type* lainnya (Connolly dan Begg, 2005, p355).

b. Relationship Types.

Relationship Types adalah sekumpulan asosiasi di antara *entity types* (Connolly dan Begg, 2005, p346).

Relationship Occurrence adalah sekumpulan asosiasi yang diidentifikasi secara unik, termasuk satu kejadian/*occurrence* dari masing-masing *entity type* yang berpartisipasi (Connolly dan Begg, 2005, p346).

c. Attribute.

Attribute merupakan properti/sifat dari suatu *entity type* atau *relationship type* (Connolly dan Begg, 2005, p350).

Attribute domain adalah sekumpulan nilai yang diijinkan untuk satu atau beberapa *atribut* (Connolly dan Begg, 2005, p350).

Attribute dapat dibedakan menjadi empat jenis, yaitu:

1) *Simple and Composite Attributes*.

Simple Attribute adalah atribut yang terdiri dari satu komponen tunggal yang keberadaannya *independen* (Connolly dan Begg, 2000, p351). Atribut ini sering disebut juga sebagai *atomic attribute*. *Simple attribute* tidak dapat dibagi lagi menjadi komponen yang lebih kecil.

Composite Attribute adalah atribut yang terdiri dari banyak komponen yang keberadaannya *independen* (Connolly dan Begg, 2005, p351). Atribut ini dapat dibagi menjadi komponen lain yang lebih kecil, yang masing-masing memiliki keberadaan yang independen.

2) *Single-Valued and Multi-Valued Attributes*.

Single-Valued Attribute adalah atribut yang memiliki nilai tunggal untuk setiap kejadian/*occurence* pada setiap *entity* (Connolly dan Begg, 2005, p351).

Multi-Valued Attribute adalah atribut yang memiliki banyak nilai untuk setiap kejadian/*occurrence* pada setiap *entity* (Connolly dan Begg, 2005, p351).

3) Derived Attributes.

Derived atribute adalah atribut yang nilainya dihasilkan dari satu atau beberapa atribut lainnya dan tidak harus berasal dari *entity* yang sama (Connolly dan Begg, 2005, p351).

4) Keys.

a) *Candidate Key.*

Candidate key merupakan sejumlah kecil atribut dari suatu *entity* yang dapat mengidentifikasikan secara unik suatu kejadian dari *entity* tersebut (Connolly dan Begg, 2005, p352). *Candidate key* tidak boleh berupa *null*.

b) *Primary Key*.

Primary key adalah *candidate key* yang dipilih untuk mengidentifikasikan secara unik setiap kejadian pada entity tersebut (Connolly dan Begg, 2005, p352). Pemilihan *primary key* pada suatu *entity* didasarkan pada pertimbangan panjang atribut, jumlah minimal atribut yang dibutuhkan serta keunikannya.

Candidate key yang tidak dipilih menjadi *primary key* pada suatu entity disebut *alternate key* (Connolly dan Begg, 2005, p352).

c) *Composite Key.*

Composite key adalah *candidate key* yang terdiri dari dua atau lebih atribut (Connolly dan Begg, 2005, p352).

d) *Foreign Key.*

Foreign key adalah sebuah atribut atau sekumpulan atribut pada suatu relasi yang sesuai dengan *candidate key* yang ada di relasi yang sama ataupun relasi yang lainnya (Connolly dan Begg, 2010, p352).

2.1.9 Normalisasi.

Menurut Connolly dan Begg (2005, p.416), normalisasi adalah suatu teknik yang menghasilkan seperangkat relasi untuk properti yang diinginkan, dengan data yang diberikan oleh suatu perusahaan. Tujuan utama dari suatu normalisasi adalah untuk mengurangi terjadinya redudansi data dan mengurangi masalah yang terjadi pada suatu relasi yang dikenal dengan *anomaly*.

Anomaly adalah suatu masalah yang timbul seperti data ganda, data hilang, tempat penyimpanan memori yang boros, dan data yang tidak konsisten akibat dari proses penghapusan data, pembaruan data, pemasukan data dan pergantian data. (Connolly dan Begg, 2005, p.388)

Tingkatan normalisasi terdiri dari beberapa tahap, meliputi :

a. *Unnormalized Form* (UNF).

Unnormalized Form adalah suatu table yang terdiri dari satu atau lebih kelompok yang berulang (*repeating group*) (Connolly dan Begg, 2005, p.430).

Repeating group adalah sebuah atribut atau himpunan atribut di dalam table yang memiliki lebih dari satu nilai (multiple values) untuk sebuah *single*

occurrence (kejadian) dari atribut *key* yang dinominasikan untuk *table* tersebut (Connolly dan Begg, 2005, p.403).

b. Bentuk Normal Pertama (*First Normal Form* atau 1NF)

Suatu relasi dikatakan 1NF jika titik temu tiap baris dan kolom pada relasi tersebut mengandung satu dan hanya satu nilai. (Connolly dan Begg, 2005, p.403)

Sebuah relasi akan berada dalam bentuk 1NF jika *repeating groupnya* telah hilang. Ada dua pendekatan untuk menghilangkan repeating group pada tabel yang tidak normal (*unnormalized table*), yaitu:

1) Dengan memasukkan data yang sesuai ke dalam kolom kosong dimana baris-barisnya mengandung data yang berulang.
2) Dengan menempatkan data yang berulang bersamaan dengan duplikat dari atribut kunci.

c. Bentuk Normal Kedua (*Second Normal Form* atau 2NF)

Relasi dikatakan 2NF jika relasi tersebut telah berada pada 1NF dan setiap atribut yang bukan *primary key* bergantung sepenuhnya (*fully functionally dependent*) terhadap *primary key* (Connolly dan Begg, 2005, p.407).

Fully functional dependency terjadi jika A dan B merupakan atribut dari suatu relasi, dan B dikatakan bergantung penuh terhadap A (A-> B), namun bukan merupakan subset A (Connolly dan Begg, 2005, p.395).

d. Bentuk Normal Ketiga (*Third Normal Form* atau 3NF).

Suatu relasi dikatakan 3NF jika relasi tersebut sudah berada dalam bentuk 1NF dan 2NF serta tidak ada atribut yang bukan *primary key* bergantung secara

transitif (*transitively dependent*) terhadap primary key (Connolly dan Begg, 2005, p.409).

Transitive dependencies terjadi dalam kondisi dimana A, B dan C merupakan atribut dari suatu relasi serta A -> B dan B -> C. Maka dapat dikatakan C bergantung secara transitif terhadap A melalui B (asalkan A tidak bergantung secara fungsional terhadap B atau C) (Connolly dan Begg, 2005, p.397).

2.1.10 MySQL.

Menurut Welling dan Thomson (2008, p3-4) MySQL berarti sistem manajemen hubungan antar basis data yang sangat cepat dan sempurna. MySQL merupakan alat bantu untuk manipulasi basis data, sehingga basis data dapat dengan mudah diisi, diambil, disusun, dan diubah datanya. *Server* MySQL pun dapat mengatur kontrol akses dari data, sehingga beberapa pengguna dapat sekaligus bekerja pada waktu yang bersamaan.

Beberapa kelebihan MySQL, dibandingkan dengan sistem basis data sejenis seperti Microsoft SQL server, Oracle adalah :

1. Kemampuan yang tinggi
2. Tidak dibutuhkan biaya untuk mendapatkan SQL
3. Mudah untuk konfigurasi dan dipelajari

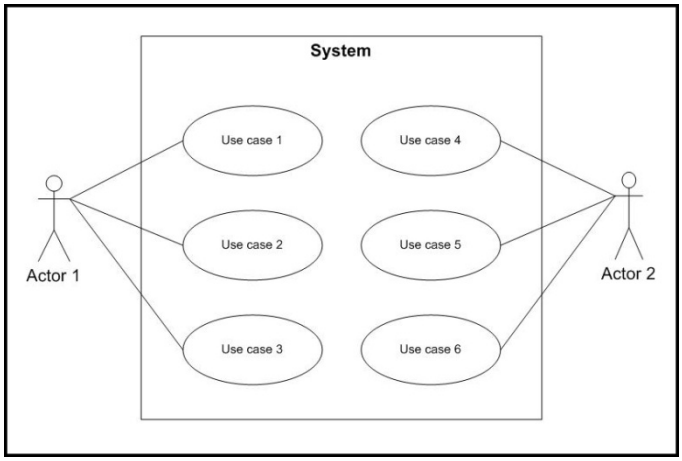
   Dapat dijalankan pada beberapa sistem operasi seperti sistem Unix dan Microsoft Windows

2.1.11 *Unified Modeling Language* (UML).

Menurut Whitten et al. (2007, p371), UML adalah pemodelan yang digunakan untuk menggambarkan sebuah sistem piranti lunak yang terkait dengan objek. UML terdiri dari beberapa tipe diagram antara lain :

1. *Use Case Diagram.*

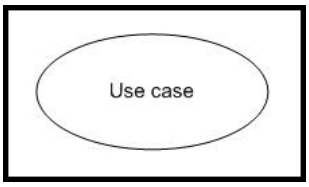
Menurut Whitten et al. (2007, p246), *use case diagram* adalah diagram yang menggambarkan interaksi antara sistem, eksternal sistem, dan pengguna. Diagram ini menyediakan informasi mengenai siapa saja yang akan menggunakan sistem tersebut dan bagaimana cara untuk menggunakannya.



Gambar 2.1 Contoh *Use Case Diagram*

a. *Use Cases.*

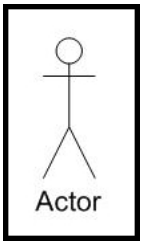
Menurut Whitten et al. (2007, p246), *use-case modeling* mengidentifikasi serta mendeskripsikan fungsi dari sebuah sistem dengan menggunakan peralatan yang dinamakan *use cases*. *Use cases* juga mendeskripsikan fungsi sistem tersebut dilihat dari sisi perspektif pengguna luar yang dimengerti oleh mereka.



Gambar 2.2 *Use Cases*

b. *Actors.*

Menurut Whitten et al. (2007, p247), *actors* merupakan sesuatu yang berinteraksi dengan sistem untuk saling bertukar informasi. *Actors* tidak harus berupa manusia, tetapi dapat berupa suatu organisasi atau sistem informasi.



Gambar 2.3 *Actors*

c. *Relationships.*

Menurut Whitten et al. (2007, p248), *relationship* digambarkan sebagai garis antara dua simbol pada *use case diagram*. Arti dari setiap *relationship* berbeda tergantung dari bagaimana garis tersebut ditarik dan jenis simbol yang terhubung. Berikut ini adalah jenis *relationship* yang ditemukan pada *use case diagram*, yaitu :

i. *Associations.*

Hubungan antara *actor* dan *use case* terjadi apabila *use case* tersebut mendeskripsikan interaksi antara kedua belah pihak. Hubungan ini ditujukkan sebagai sebuah *association*. *Association*

ditandai dengan tebal antara *actor* dan *use case*, *association* yang mempunyai mata panah di ujungnya menandakan bahwa *use case* telah diimitasi oleh *actor* di tabel lainnya. Apabila *association* tidak memiliki mata panah di ujungnya menandakan interaksi antara *use case* dan *external server*. *Association* juga dapat diartikan sebagai lebih dari satu arah (*bidirectional*) atau satu arah (*unidirectional*).

ii. *Extends.*

Hubungan yang terjadi antara *extention use case* dan *use case* yang berkembang dinamakan *extends relationship*. Sebuah *use case* diperbolehkan untuk mempunyai banyak *extends relationship*, tetapi *extension use case* hanya dapat dilakukan apabila bersama dengan *use case* yang sedang berkembang. *Extends relationship* ditunjukkan dengan garis yang mempunyai mata panah di ujungnya (garis tebal atau garis putus-putus). Bermula dari *extension use case* dan menunjuk kepada *use case* yang berkembang.

iii. *Uses* (*or Includes*).

Sebuah *use cas*e abstrak dapat digunakan oleh *use case* lain yang membutuhkan fungsionalitas dari *use case* abstrak tersebut. Hubungan antara *use case* abstrak dan *use case* tersebut dapat diartikan sebagai *uses relationship*. *Uses relationship* dapat ditunjukan dengan garis yang mempunyai mata panah di ujungnya (garis tebal atau garis putus-putus). Bermula dari *original use case* dan menunjuk ke *use case* yang digunakan. Setiap garis mempunyai label di bawahnya.

iv. *Depends on.*

Dengan menggunakan bank sebagai contoh, *use case* pengambilan atau penarikan uang tidak dapat dilakukan sampai *use case* deposit telah selesai dilakukan terlebih dahulu, serta *use case* deposit juga tidak dapat dilakukan sebelum *use case* konfirmasi *account* bank dipastikan. *Depends-on relationship* ditunjukkan dengan garis yang mempunyai mata panah di ujungnya (garis tebal atau garis putus-putus). Bermula dari satu *use case* dan menunjuk ke *use case* yang lain. *Depends-on relationship* mempunyai label di bawahnya.

v. *Inheritance.*

Ketika dua atau lebih *actor* mempunyai peraturan yang sama dalam kata lain dapat menginisiasi *use case* yang sama ada baiknya untuk memperkirakan kemungkinan dari kesamaan peraturan ini dan menunjuknya sebagai *abstract actor* yang baru dengan tujuan untuk mengurangi redundansi komunikasi antar sistem.

2. *Class Diagram.*

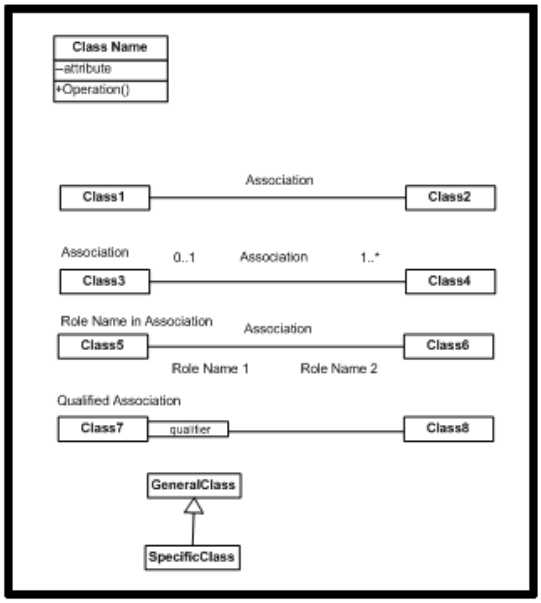
Menurut Whitten et al. (2007, p382), *class diagram* menggambarkan struktur objek yang terdapat pada sebuah sistem. Diagram ini menunjukkan objek-objek yang terdapat pada suatu sistem serta relasi antar objek-objek tersebut. *Class diagram* mempunyai 2 jenis dalam penggunaannya, yaitu :

a. Generalisasi.

Menurut Whitten et al. (2007, p373), generalisasi adalah sebuah teknik dimana atribut merupakan beberapa jenis *object class* dan dikelompokan menjadi *class* mereka sendiri dan biasa disebut dengan *supertype*.

b. *Inheritance.*

Menurut Whitten et al. (2007, p373), *inheritance* adalah konsep dimana metode dan atribut yang dapat dianggap sebagai *object class* dapat diwariskan (*inherited*) atau digunakan oleh *object class* lainnya.

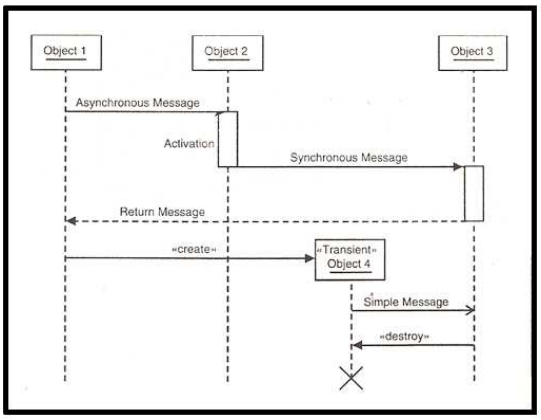


Gambar 2.4 Contoh *Class Diagram*

3. *Sequence Diagram.*

Menurut Whitten et al. (2007, p394), *sequence diagram* merupakan sebuah gambaran yang menjelaskan mengenai interaksi antara *actor* dengan sistem yang digunakan dalam skenario *use case* yang sedang berlangsung.

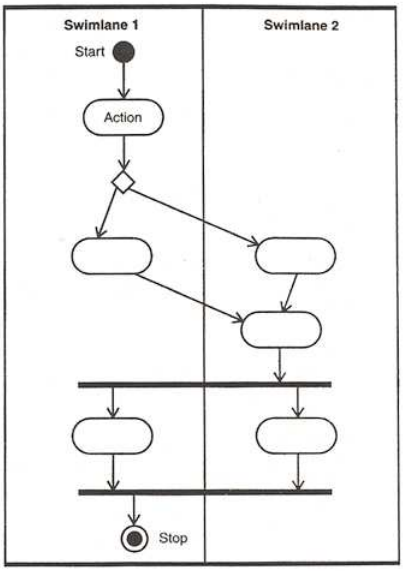
Diagram ini menggambarkan bagaimana pesan dikirim dan diterima antar objek dan urutannya.



Gambar 2.5 Contoh *Sequence Diagram*

4. *Activity Diagram.*

Menurut Whitten et al (2007, p382), *activity diagram* menggambarkan aliran berurutan dari sebuah proses *use case* atau *business process*. Selain itu, dapat juga digunakan untuk logika model dengan sistem yaitu., menggambarkan tindakan (a*ction*) yang akan dijalankan ketika suatu proses sedang berjalan dan beserta hasil dari proses yang dijalankan tersebut.



Gambar 2.6 Contoh *Activity Diagram*

Berikut ini merupakan komponen-komponen dalam *activity diagram* menurut Whitten et al. (2007, p391), yaitu :

a. *Initial Node.*

Bentuk lingkaran berisi penuh melambangkan awal dari suatu proses.

b. *Actions.*

Bentuk persegi panjang yang mempunyai ujung lingkaran yang melambangkan tahap-tahap per individu. *Sequence* dari *actions* menunjukan total dari aktivitas yang dilihat dari diagram.

c. *Flow.*

Panah pada diagram mengindikasikan kemajuan (*progress*) dari sebuah *actions*. Kebanyakan *flow* tidak membutuhkan kata untuk mengidentifikasikan mereka kecuali kata tersebut keluar dari *decision*.

d. *Decision.*

Bentuk wajib dengan satu *flow* yang masuk beserta dua atau lebih *flow* yang keluar. *Flow* yang keluar ditandai untuk mengindikasikan beberapa kondisi.

e. *Merge.*

Bentuk wajib dengan dua atau lebih *flow* yang masuk beserta satu *flow* yang keluar. Kombinasi *flow* ini sebelumnya dipisahkan oleh *decision*. Proses akan berlanjut dengan adanya satu *flow* yang menuju ke *merge*.

f. *Fork.*

Satu *bar* hitam dengan satu *flow* yang masuk beserta dua atau lebih *flow* yang keluar. *Actions* dengan *flow* pararel di bawah *fork* dapat terjadi dengan adanya urutan secara bersamaan.

g. *Join.*

Satu *bar* hitam dengan dua atau lebih *flow* yang masuk beserta satu *flow* yang keluar, tercatat pada akhir dari proses secara bersamaan. Semua *actions* yang menuju *join* harus lengkap sebelum proses dapat berlanjut.

h. *Activity Final.*

Bentuk lingkaran berisi penuh yang berada di dalam lingkaran kosong, melambangkan akhir dari sebuah proses.

i. *Subactivity Indicator.*

Simbol seperti sisir terbalik yang berada pada *actions* mengindikasikan bahwa *actions* telah keluar menuju *activity diagram* yang lain. Hal ini dapat membantu *activity diagram* agar tidak menjadi terlalu kompleks.

j. *Connector*.

Huruf yang berada di dalam lingkaran dan memberikan alat untuk mengatur kompleksitas. *Flow* yang menuju *connector* melompati *flow* yang keluar dari *connector* dengan huruf yang sama.

2.1.12 Interaksi Manusia dan Komputer (IMK).

Menurut Shneiderman dan Plaisant (2010, p88-89), disebutkan pula bahwa ada delapan aturan emas yang digunakan dalam merancang antarmuka, yaitu :

1. Mencoba Untuk Konsisten.

Konsistensi merupakan aturan yang sering dilanggar karena memiliki banyak bentuk konsistensi, antara lain urutan aksi, istilah yang digunakan, warna, *layout*, kapitalisasi, huruf dan lain-lain. Dengan tampilan yang konsisten akan membantu *user* untuk merasa tetap berada dalam aplikasi yang sama walaupun telah berpindah halaman.

2. Memenuhi Kebutuhan *Universal*.

Memenuhi kebutuhan universal maksudnya adalah memahami kebutuhan *user* yang bermacam-macam dan membuat desain fleksibel yang mendukung perubahan dalam konten. Perbedaan *Novice-Expert*, jarak umur, kecacatan fisik, serta beragam teknologi masing-masing merupakan syarat yang harus menjadi pertimbangan dalam desain.

3. Memberikan Umpan Balik Yang Informatif.

Umpan balik dari sistem harus ada pada setiap aksi *user*. Untuk aksi kecil yang sering dilakukan, tanggapan dapat dibuat dengan sederhana. Sedangkan untuk aksi besar dan jarang dilakukan, respon hendaknya dibuat lebih tegas dan jelas agar *user* dapat mengerti dengan jelas.

4. Dialog Untuk Keadaan Akhir.

Urutan aksi hendaknya disusun menjadi kategori awal, tengah, dan akhir. Untuk memberikan kepuasan pencapaian, kelegaan, dan sebagai tanda untuk mempersiapkan diri memasuki kategori aksi selanjutnya, dibuatlah umpan balik yang informatif pada penyelesaian salah satu kategori aksi.

5. Pencegahan Kesalahan.

Sedapat mungkin sistem didesain agar *user* tidak dapat membuat kesalahan yang serius. Contohnya tidak memperbolehkan karakter alfabet pada kotak *entry* nomor. *Interface* harus mendeteksi kesalahan dan memberikan instruksi yang mudah dimengerti, membangun, dan jelas untuk memperbaikinya, jika *user* membuat kesalahan.

6. Pembalikan Aksi Yang Sederhana.

Dalam suatu aplikasi, pada setiap aksi harus terdapat pembalikan aksi. Fitur I ni dapat memperkecil kesalahan, karena *user* tahu bahwa aksi bisa dibatalkan. Pembalikan bisa saja atas satu aksi seperti saat memasukkan data, atau serangkaian aksi seperti memasukkan nama dan alamat di kotak pengisian.

7. Mendukung Pusat Kendali *Internal*.

User yang sudah terbiasa dengan suatu aplikasi, biasanya ingin memiliki kendali atas antarmuka dan tanggapan pada aksinya. Aksi antarmuka yang tidak seperti biasanya, rangkaian pemasukan data yang membosankan, tidak bisa atau sulit mendapatkan informasi yang diperlukan, dan tidak bisa menghasilkan aksi yang diinginkan dapat menimbulkan keresahan dan ketidakpuasan pada *user*.

8. Mengurangi Beban Ingatan Jangka Pendek.

Dikarenakan oleh keterbatasan manusia dalam memproses informasi dalam jangka pendek, dibutuhkan tampilan yang ringan, penggabungan halaman-halaman, pengurangan frekuensi *window-motion*, pemberian waktu latihan yang cukup untuk kode-kode, hafalan, dan rangkaian aksi. Oleh karena itu, dalam setiap perancangan aplikasi dibutuhkan alur aplikasi yang mudah diingat oleh *user*.

2.1.13 *Hypertext Preprocessor* (PHP).

Menurut Ullman (2005, pxi), PHP merupakan *server-side* dan *cross-platform technology*. *Server-side* mengacu pada fakta bahwa semua *script* PHP bekerja pada *server*. *Cross-platform* berarti PHP dapat bekerja pada kebanyakan *operating-system*, termasuk Windows, Unix, dan Machintos.

Berdasarkan keunggulan dan kemampuannya (dan dimulainya dalam pemakaian untuk keperluan profesional), PHP berubah menjadi PHP:*Hypertext Preprocessor*.

Beberapa kelebihan PHP dibandingkan dengan bahasa pemrograman sejenis seperti Perl, Microsoft Active Server Pages (ASP), Java Server Pages (JSP), dan Allaire Cold Fusion adalah :

1. Kemampuan yang tinggi
2. Kemampuan untuk dapat terhubung dengan banyak sistem basis data, seperti : MySQL, PostgreSQL, mSQL, Oracle, dbm, *file*pro, Hyperwave Informix, dan Interbase
3. Tidak dibutuhkan biaya untuk mendapatkan PHP

4. Mudah dipelajari dan digunakan, karena PHP dibuat berdasarkan bahasa pemrograman dasar, yaitu bahasa C dan Perl

5. Dapat berjalan pada berbagai sistem operasi, seperti Linux, Solaris, dan beberapa versi Microsoft Windows.

2.1.14 Internet.

Internet adalah sistem global dari seluruh jaringan komputer yang saling terhubung menggunakan standar *Internet Protocol Suite* (TCP/IP). Menurut William/Sawyer (2007, p60), internet adalah ratusan ribu jaringan kecil yang menghubungkan organisasi pendidikan, komersial, nirlaba, militer, dan bahkan perorangan. Jaringan-jaringan interkoneksi itu saling bertukar informasi dengan menggunakan standar pertukaran informasi yang berlaku.

2.1.15 *World Wide Web* (WWW).

World wide web yang biasa disingkat menjadi www atau biasa dikenal dengan web adalah sistem dokumen *hypertext* yang saling diakses melalui internet. Dengan web *browser*, kita dapat melihat halaman web yang mungkin berisi tentang teks, gambar, video, dan multimedia lainnya, serta melakukan navigasi pengiriman dari satu web ke web yang lainnya (*hyperlink*).

Web menyediakan cara termudah dalam mengakses informasi dan menjalankan beberapa program yang disimpan dalam komputer yang terkoneksi oleh internet. Ditambah lagi, web memiliki memori didalamnya dimana informasi tersebut dapat diperlihatkan, disimpan, dan diakses. Informasi yang disimpan ini dapat diakses oleh seluruh komputer yang terkoneksi oleh internet. Konsep dasar dari web sendiri dibagi menjadi lima, yaitu :

1. *Hypermedia*, bentuk dasar dari web yang berfungsi untuk menyimpan informasi di dalam web.
2. HTML, bahasa pemrograman yang digunakan untuk kode *hypermedia*.
3. URL, alamat yang digunakan untuk mengakses web *resources* seperti dokumen dan program dari *hypermedia*.
4. HTTP, protokol yang digunakan untuk mengakses web *resources*.

Gateways, halaman web yang dirancang untuk menarik pengunjung dan *search engine* ke situs web tertentu.

2.2 Teori Khusus.

2.2.1 Investasi.

Investasi adalah penanaman modal untuk satu atau lebih aktiva yang dimiliki dan biasanya berjangka waktu lama dengan harapan mendapatkan keuntungan di masa-masa yang akan datang.

Jenis-jenis Investasi.

a.) Tabungan di Bank.

Dengan menyimpan uang di tabungan, maka akan mendapatkan suku bunga tertentu yang besarnya mengikuti kebijakan bank bersangkutan. Produk tabungan biasanya memperbolehkan kita mengambil uang kapanpun yang kita inginkan.

b.) Deposito di Bank.

Produk deposito hampir sama dengan produk tabungan. Bedanya, dalam deposito tidak dapat mengambil uang kapanpun yang diinginkan, kecuali apabila uang tersebut sudah menginap di bank selama jangka waktu tertentu (tersedia pilihan antara satu, tiga, enam, dua belas, sampai dua puluh empat bulan, tetapi ada juga yang harian). Suku bunga deposito biasanya lebih tinggi daripada suku bunga tabungan. Selama deposito kita belum jatuh tempo, uang tersebut tidak akan terpengaruh pada naik turunnya suku bunga di bank.

a.) Saham.

Saham adalah kepemilikan atas sebuah perusahaan tersebut. Dengan membeli saham, berarti membeli sebagian perusahaan tersebut. Apabila perusahaan tersebut mengalami keuntungan, maka pemegang saham biasanya

akan mendapatkan sebagian keuntungan yang disebut deviden. Saham juga bisa dijual kepada pihak lain, baik dengan harga yang lebih tinggi yang selisih harganya disebut capital gain maupun lebih rendah daripada kita membelinya yang selisih harganya disebut capital loss. Jadi, keuntungan yang bisa didapat dari saham ada dua yaitu deviden dan capital gain.

b.) Properti.

Investasi dalam properti berarti investasi dalam bentuk tanah atau rumah.

1. Keuntungan yang bisa didapat dari properti ada dua yaitu : Menyewakan properti tersebut ke pihak lain sehingga mendapatkan uang sewa.
2. Menjual properti tersebut dengan harga yang lebih tinggi.

e.) Barang-barang Koleksi.

Contoh barang-barang koleksi adalah perangko, lukisan, barang antik, dan lain-lain. Keuntungan yang didapat dari berinvestasi pada barang-barang koleksi adalah dengan menjual koleksi tersebut kepada pihak lain.

f.) Emas.

Emas adalah barang berharga yang paling diterima di seluruh dunia setelah mata uang asing dari negara-negara G-7 (sebutan bagi tujuh negara yang memiliki perekonomian yang kuat, yaitu Amerika, Jepang, Jerman, Inggris, Italia, Kanada, dan Perancis). Harga emas akan mengikuti kenaikan nilai mata uang dari negara-negara G-7. Semakin tinggi kenaikan nilai mata uang asing tersebut, semakin tinggi pula harga emas. Selain itu harga emas biasanya juga berbanding searah dengan inflasi. Semakin tinggi inflasi, biasanya akan semakin

tinggi pula kenaikan harga emas. Seringkali kenaikan harga emas melampaui kenaikan inflasi itu sendiri.

g.) Mata Uang Asing.

Segala macam mata uang asing biasanya dapat dijadikan alat investasi.

Investasi dalam mata uang asing lebih beresiko dibandingkan dengan investasi dalam saham, karena nilai mata uang asing di Indonesia menganut sistem mengambang bebas (free float) yaitu benar-benar tergantung pada permintaan dan penawaran di pasaran. Di Indonesia mengambang bebas membuat nilai mata uang rupiah sangat fluktuatif.

h.) Obligasi.

Obligasi atau sertifikat obligasi adalah surat utang yang diterbitkan oleh pemerintah maupun perusahaan, baik untuk menambah modal perusahaan atau membiayai suatu proyek pemerintah. Karena sifatnya yang hampir sama dengan deposito, maka agar lebih menarik investor suku bunga obligasi biasanya sedikit lebih tinggi dibanding suku bunga deposito. Selain itu seperti saham kepemilikan obligasi dapat juga dijual kepada pihak lain baik dengan harga yang lebih tinggi maupun lebih rendah daripada ketika membelinya.

2.2.2 Investor.

Investor adalah orang perorangan atau lembaga baik domestik atau non domestik yang melakukan suatu investasi (bentuk penanaman modal sesuai dengan jenis investasi yang dipilihnya) baik dalam jangka pendek atau jangka panjang.

Tipe-tipe investor menurut profil resiko dalam berinvestasi dapat dideskripsikan berikut :

1. *Defensive.*

Investor dengan tipe *defensive, investor* ini berusaha untuk mendapatkan keuntungan dan menghindari resiko sekecil apapun dari investasi yang dilakukan. *Investor* tipe ini tidak mempunyai keyakinan yang cukup dalam hal spekulasi, dan lebih memilih untuk menunggu saat-saat yang tepat dalam berinvestasi agar investasi yang dilakukan terbebas dari resiko.

2. *Conservative.*

Investor dengan tipe *conservative*, biasanya berinvestasi untuk meningkatkan kualitas hidup keluarga dan dengan rentang waktu investasi yang cukup panjang, misalnya, untuk pendidikan perguruan tinggi anak atau biaya hidup di hari tua. Investor tipe ini memiliki kecenderungan menanam investasi dengan keuntungan yang layak saja dan tidak memiliki resiko besar, karena filosofi investasi mereka untuk menghindari resiko. Walaupun *investor conservative* sering berinvestasi, *investor* ini umumnya mengalokasikan sedikit waktu untuk menganalisa dan mempelajari portofolio investasinya.

3. *Balanced.*

Investor dengan tipe *balanced*, merupakan tipe *investor* yang menginginkan resiko menengah. *Investor* tipe ini selalu mencari proporsi yang seimbang antara resiko yang dimungkinkan terjadi dengan pendapatan yang dapat diraih. Tipikal *investor* ini bahwa mereka akan selalu berhati-hati dalam memilih jenis investasi, dan hanya investasi yang proporsional antara resiko dan penghasilan yang bisa diperoleh yang akan dipilih.

4. *Moderately Aggressive*.

Moderately aggressive, merupakan tipe *investor* yang tenang atau tidak ekstrim dalam menghadapi resiko. *Investor* ini cenderung memikirkan kemungkinan terjadinya resiko dan kemungkinan bisa mendapatkan keuntungan.

Dalam hal ini, investor dengan tipe *moderately aggressive* selalu tenang dalam mengambil keputusan investasi karena keputusan yang ditetapkan sudah dipikirkan sebelumnya

5. *Aggressive.*

Investor aggressive, atau biasa disebut 'pemain', adalah kebalikan dari *investor conservative*. Mereka sangat teliti dalam menganalisa portofolio yang dimiliki.

Semakin banyak angka-angka dan fakta yang bisa dianalisa adalah semakin baik. Investor tipe ini umumnya berinvestasi dengan rentang waktu relatif pendek karena mengharapkan adanya keuntungan yang besar dalam waktu singkat. Walaupun tidak berharap untuk merugi, namun setiap investor aggressive menyadari bahwa kerugian adalah bagian dari permainan.

(www.danareksa.com/home/index_produk.cfm?act=investasiRepot).

2.2.3 Deposit.

Deposit atau deposito adalah instrumen investasi dengan resiko kecil. Melalui investasi atau tabungan deposito berjangka, anda dapat menjaga nilai pokok dari uang yang anda investasikan.

(http://www.belajarinvestasi.net/lain/tabungan-deposito).