2. LANDASAN TEORI

2.1. *Entity Relationship Diagram* (ERD)

ERD adalah representasi data dari suatu organisasi secara detail, masuk akal dan digambarkan dalam bentuk grafik. ERD merupakan model dari *entity* dalam suatu elemen bisnis, relasi antara *entity* dan atribut atau *property* dari *entity* dan relasinya. Berikut adalah beberapa simbol yang digunakan dalam membuat suatu ERD *(Valacich, George, and Hoffer, 2004)*:

2.1.1. *Entity*

Suatu *entity* adalah seseorang, tempat, objek, kejadian atau konsep dalam lingkungan *user*, dimana suatu organisasi membutuhkan untuk memelihara data. *Entity* mempunyai identitas sendiri, dimana antar satu *entity* dengan yang lain berbeda. Beberapa contoh dari *entity* adalah:

1. Orang : PEGAWAI, PELAJAR, PASIEN
2. Tempat : NEGARA, DAERAH, KOTA
3. Objek : MESIN, BANGUNAN, PRODUK
4. Kejadian : PENJUALAN, PEMBELIAN, REGISTRASI
5. Konsep : AKUNTANSI

Suatu *entity* digambarkan dengan kotak, dimana terdapat nama yang mengidentifikasikan *entity* tersebut, contoh EMPLOYEE *entity* dapat dilihat pada Gambar 2.1:

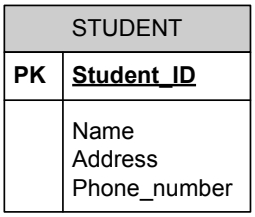
Gambar 2.1 *Entity*

Sumber: Valacich, George, and Hoffer, 2004.

2.1.2. *Attributes*

Setiap *entity* mempunyai beberapa properti atau karakteristik. Atribut menggambarkan tentang rincian dari *entity*, misalnya:

STUDENT: Student_ID, Name, Address, Phone_number



Gambar 2.2 *Entity* dengan Atribut

Sumber: Valacich, George, and Hoffer, 2004.

2.1.3. *Identifier* atau *primary key*

Primary key adalah attribut *unique* yang mengidentifikasi setiap *entity*. *Unique* di sini yang dimaksud adalah dalam *entity* tersebut tidak boleh ada *primary key* yang kembar. Pada Gambar 2.2. di atas, *primary key* untuk *entity* STUDENT adalah STUDENT_ID.

2.1.4. *Relationship*

Relationship adalah relasi atau hubungan antar *entity.* Berikut adalah beberapa istilah dalam penggolongan jenis relasi antar *entity (Kendall, Kenneth E., 2002)*:

1. *Cardinality*

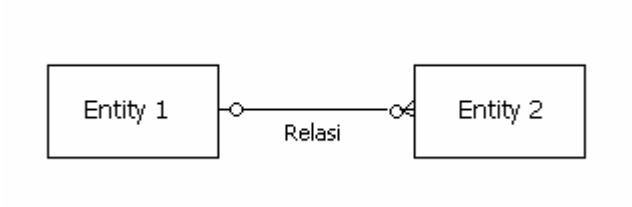
Menandai jumlah *entity* yang muncul pada *entity* lainnya. Misalkan, terdapat dua *entity* A dan B, *cardinality* adalah angka dari B yang dapat diassosiasikan dengan *entity* A dan sebaliknya. Nilai untuk *cardinality* ini ada dua yaitu *one* atau *many*. Berikut adalah relasi dengan *cardinality*:

a. One to one

Gambar 2.3 *One to One Relationship*

Sumber: Kendall, Kenneth E., 2002.

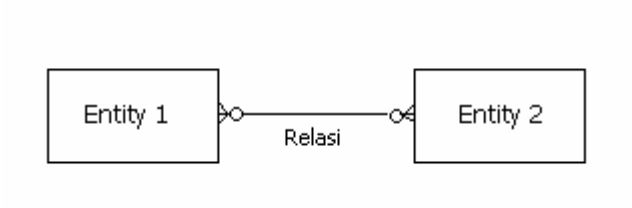
b. *One to many*



Gambar 2.4 *One to Many Relationship*

Sumber: Kendall, Kenneth E., 2002.

c. *Many to many*

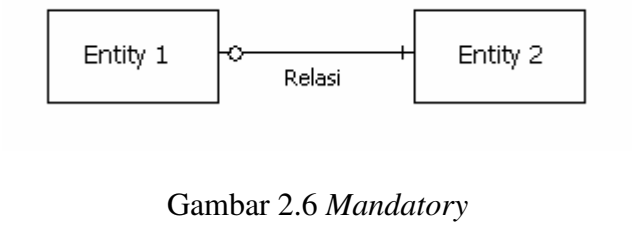


Gambar 2.5 *Many to Many Relationship*

Sumber: Kendall, Kenneth E., 2002.

2. *Mandatory*

Sedangkan *mandatory* adalah untuk menunjukan apakah *entity* tersebut harus berelasi dengan *entity* lain atau tidak. Jika iya bernilai 1 sedangkan jika tidak maka bernilai 0.



Gambar 2.6 *Mandatory*

Sumber: Kendall, Kenneth E., 2002.

2.2. *Data Warehouse*

Sebelum pembuatan OLAP *tools,* diperlukan suatu tempat untuk menyimpan data yang telah diproses dan didesain sedemikian rupa yang

digunakan untuk melakukan analisa. Disini *data warehouse* menyediakan tempat yang terbaik untuk analisa dan mudah untuk diakses *(Ponniah, 2003)*.

2.2.1. Definisi *Data Warehouse*

Data warehouse adalah suatu *database* yang mendukung dalam melakukan analisa untuk pengambilan keputusan yang menggambarkan *historical* data dari suatu organisasi. *Data warehouse* dapat berfungsi sebagai *Decision Support System* (DSS) dan *Executive Information System* (EIS). *Data warehouse* mempunyai struktur khusus dalam pembuatan *query* dan analisa. *Data warehouse* tidak dapat membuat keputusan tetapi hanya menyediakan informasi untuk membuat keputusan. *Data warehouse* menyediakan akses data lebih akurat dan menggabungkan informasi dari sumber *internal* maupun *eksternal*. Menurut Bill Inmon yaitu *father of Data Warehousing (Ponniah, 2003)*, *data warehouse* adalah kumpulan data yang *subject oriented*, *integrated*, *nonvolatile* dan *time variant* dalam mendukung pengambilan keputusan. Pengertiannya adalah sbb:

1. *Subject Oriented*

   Data yang dibuat, dirancang, dan dibangun untuk melakukan analisa terhadap subjek tertentu. Misalnya analisis mengenai data penjualan dalam suatu organisasi bisnis. Semua informasi disimpan dalam suatu sistem *data warehouse.*

2. *Integrated*

   Untuk dapat memenuhi kebutuhan analisa secara menyeluruh dan konsisten, suatu *data warehouse* harus dapat mengintegrasikan data dari berbagai sumber yang berbeda.

3. *Nonvolatile*

   Data dalam *database* tidak dapat dihapus atau ditumpuk (*over-written*), karena setiap data merupakan historis yang digunakan untuk melakukan analisa.

4. *Time Variant*

   Setiap perubahan data disimpan sehingga data dapat dibuat berdasarkan perubahan terhadap waktu. Dengan kata lain data dapat digunakan untuk melakukan analisa atas kejadian lalu, berhubungan dengan informasi sekarang dan dapat melakukan perkiraan untuk masa depan.

Beda antara *database* biasa dengan *data warehouse* adalah *user* tidak dapat melakukan *update* terhadap data dalam *data warehouse.* Data dalam *data warehouse* hanya dapat dibaca untuk dianalisa.

2.2.2. Proses Dalam *Data Warehouse*

Dalam pembuatan suatu *data warehouse,* yang perlu dilakukan pada awalnya adalah menentukan subyek area yang akan dibuat dan juga definisi yang akan digunakan. Proses ini memerlukan analisa dari kebutuhan informasi. Dengan adanya permasalahan yang ada, akan dibuat sebuah *data warehouse.*

Langkah-langkah dalam pembuatan suatu *data warehouse* adalah sebagai berikut:

1. *Extraction*

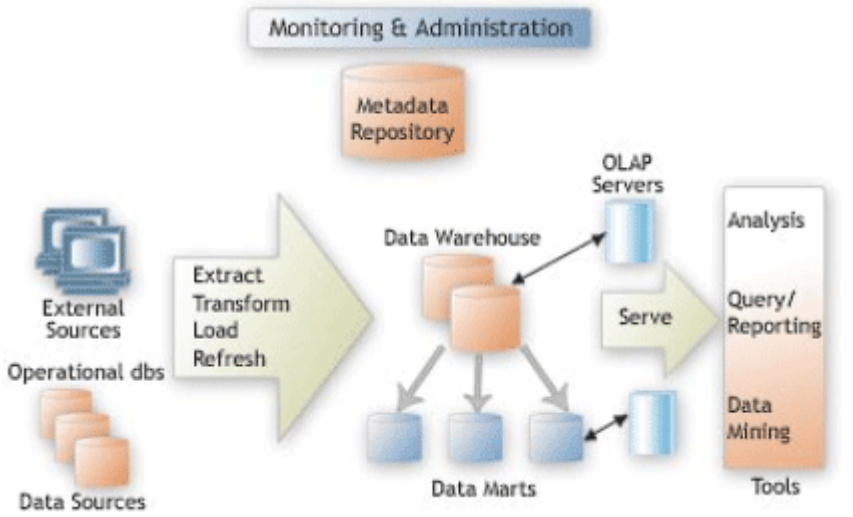
Ekstrak adalah proses pengambilan data dari sumber data. Disebut ekstrak, karena proses pengambilan data ini tidak mengambil keseluruhan data yang ada di *database* operasional, melainkan hanya mengambil data matang saja. Proses ini meliputi penyaringan data yang akan digunakan dalam pembuatan *data warehouse.* Dapat langsung dimasukkan langsung dalam *data warehouse* atau dimasukkan dalam tempat penampungan sementara terlebih dahulu.

2. *Transformation*

Proses yang dilakukan adalah mentransformasi data yang telah diekstrak ke dalam format yang diperlukan. Hal ini perlu dilakukan mengingat data yang diambil berasal dari sumber yang berbeda yang kemungkinan memiliki standarisasi yang berbeda pula. Standarisasi diperlukan untuk nantinya memudahkan pembuatan laporan.

3. *Load*

Proses *load* adalah suatu proses mengirimkan data yang telah menjalani proses transformasi ke *data warehouse*.



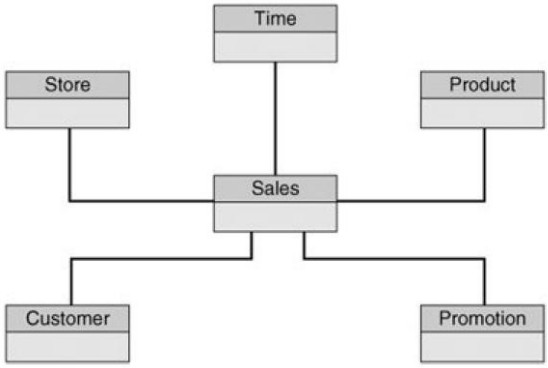
Gambar 2.7 Proses *Data Warehouse*

Sumber: K.A.H & Associates Inc., *Data Warehousing*, 2006. http://kahassoc.com/datawarehousing.htm

Gambar 2.7 menunjukkan proses *data warehouse* dimana data dari *data sources* diolah (*extract, transform, load*) menjadi *data warehouse* untuk selanjutnya di-*query* untuk menghasilkan *report* yang diinginkan. *Data mart* berisikan kumpulan data yang digunakan untuk pengambilan keputusan. *Data mart* hampir sama seperti *data warehouse,* tetapi perbedaannya adalah data *mart* lebih spesifik dan ditujukan untuk bagian-bagian tertentu.

Dalam pembuatan suatu *data warehouse,* ada dua jenis *schema* yang sering digunakan yaitu *Star Schema* dan *Snowflake Schema*. *Star Schema* merupakan suatu *relational database* yang mengandung sebuah *fact table* sebagai pusatnya. *Fact table* memuat ukuran yang dikelilingi dengan beberapa *dimension table*. Setiap *dimension table* mengandung informasi detail yang membantu untuk menyimpulkan informasi dalam *fact table* dengan cara yang berbeda.

Gambaran dari *star schema* secara umum adalah sebagai berikut:



Gambar 2.8 *Star schema*

Sumber: Sonny Setiawan, 2007.

Tabel *sales* pada Gambar 2.8 yang berada di tengah akan menjadi *fact table*-nya. Kemudian tabel *store, time, product, promotio*n dan *customer* disebut sebagai *dimension table*.

Keuntungan dari penggunaan *star schema (Ponniah, 2003)* adalah:

1. Memudahkan *user* untuk lebih mengerti, karena *star schema* menggambarkan bagaimana *user* berpikir dan memerlukan data untuk query dan analisa.
2. Lebih sesuai untuk proses *query.*

2.3. *Online Analytical Processing* (OLAP)

OLAP adalah salah satu *tools* yang digunakan untuk mengakses *data warehouse.* Secara umum, suatu *data warehouse* menyediakan kesempatan terbaik untuk melakukan analisa, dan OLAP adalah kendaraan atau alat untuk menyajikan analisa tersebut. OLAP menawarkan metode analisis data secara kompleks yang disesuaikan dengan kebutuhan informasi dari *user.* Data yang dikelola oleh OLAP berasal dari *user.* Keuntungan dari penggunaan OLAP *(Ponniah, 2003)* adalah:

1. Fleksibilitas dari OLAP membuat *user* dapat melakukan analisanya sendiri tanpa bantuan dari bagian IT.
2. Meningkatkan produktivitas dari manager, eksekutif dan analis.
3. Lebih efisien terutama dalam mengurangi waktu pada saat menjalankan *query*.

4. Keuntungan untuk pengembang IT karena *software* memang ditujukan untuk hasil dari *system development* dengan kecepatan pengiriman aplikasi.

OLAP memproses data dalam *data warehouse* dengan struktur multidimensi, sehingga dapat menyediakan jawaban untuk *query* analisis yang kompleks. Sistem dari OLAP menyediakan fleksibilitas dalam mendukung proses analisi secara *real time*.

2.3.1. OLAP *Cube* (Kubus OLAP)

Kubus OLAP adalah kumpulan data struktur multidimensi yang menggambarkan data yang ada dalam *data warehouse*. Data yang digambarkan berupa *fact table* dan *dimension table* dalam *data warehouse.* Beberapa operasi umum dalam kubus OLAP adalah:

1. *Roll-Up*

Melakukan pengumpulan pada tingkatan yang berbeda dari hirarki dimensi. Misalnya untuk setiap kota diberikan total penjualan, maka untuk total penjualan tiap propinsi bisa didapatkan dengan menambah total penjualan pada semua kota dalam satu propinsi.

2. *Drill Down*

Sebaliknya dengan *Roll Up*, misalnya untuk tiap propinsi diberikan total penjualan, maka total penjualan tiap kota dapat di *Drill Down*. Jadi disini, *Drill Down* merupakan teknik untuk memecah informasi menjadi beberapa informasi yang lebih detail lagi.

3. *Slice and Dice / Rotation*

Dengan *Slice and Dice* memungkinkan untuk melihat kubus dari sudut pandang yang berbeda sesuai yang diinginkan. Dengan ini, *user* dapat melakukan analisa dari lebih banyak hal. *Slice and Dice* merupakan keunggulan dari OLAP.

4. *Pivot*

Melakukan agregasi pada dimensi yang dipilih dengan menukarkan dimensi data. Dengan *pivoting*, *user* dapat menganalisa data dari berbagai sudut pandang.

2.3.2. OLAP Models

Tempat penyimpanan data untuk OLAP ada beberapa macam *(Ponniah, 2003)*, yaitu ROLAP *(Relational Online Analytical Processing)*, MOLAP *(Multidimensional Online Analytical Processing)*, dan HOLAP *(Hybrid Online Analytical Processing).* MOLAP adalah salah satu cara yang umum digunakan dan digunakan dalam pembuatan Tugas Akhir. MOLAP menyimpan data dalam bentuk *array* multidimensi. Dengan demikian, *user* dapat melakukan *query* pada basis data MOLAP dengan cepat karena dimensi dapat diakses dengan mudah. Beberapa keuntungan dari MOLAP adalah:

1. *Query* dapat dilakukan dengan cepat
2. Perhitungan secara otomatis dari level agregasi data yang lebih tinggi
3. *Array* model menyediakan index secara alami

2.4. *Software* Pendukung Aplikasi

Dalam penyelesaian tugas akhir ini, bahasa pemrograman yang digunakan adalah Java Netbeans IDE 6.1 dan Oracle 9i .

2.4.1. Java

Java adalah bahasa pemrograman *open-sources* yang menggunakan *syntax* seperti bahasa pemrograman C dan C++. Pemrograman pada Java juga menggunakan konsep *Object Oriented Programming.* Selain itu ada beberapa kelebihan dalam penggunaan Java, diantaranya adalah sebagai berikut:

1. *Multiplatform*

Java dapat dijalankan pada beberapa *platform*/sistem operasi komputer. Kelebihan ini memungkinkan untuk suatu program berbasis Java dikerjakan di salah satu OS dan dapat dijalankan di OS lainnya. *Platform* yang didukung sampai saat ini adalah Windows, LINUX, Mac OS dan Sun Solaris.

2. Perpustakaan yang lengkap

Java terkenal dengan kelengkapan *library,* sehingga memudahkan *user* untuk membangun aplikasinya. Selain itu meluasnya penggunaan Java telah menyebabkan banyaknya komunitas-komunitas yang dapat membantu dalam pembangunan aplikasi.

Java juga mempunyai kekurangan yaitu penggunaan memori yang besar. Memori yang dibutuhkan jauh lebih besar dari pada penggunaan bahasa pemrograman C/C++, Delphi dan Pascal. Hal ini tidak jadi masalah untuk pengguna teknologi terbaru.

Untuk Java IDE ada beberapa macam, misalnya JCreator, Eclipse, Sun Java Studio Creator, Netbeans, JBuilder, dll. IDE yang digunakan dalam aplikasi ini adalah Netbeans 6.1.

2.4.2. Oracle 9i

Oracle 9i merupakan suatu *database* relasional yang terdiri dari kumpulan data dalam suatu sistem manajemen basis data (RDBMS). Oracle mendukung semua aplikasi *database*. Oracle RDBMS menyimpan data logis ke dalam *tablespace* dan data fisik ke dalam *data files*. Oracle dapat menyimpan dan mengeksekusi *stored procedure* dan *functions* dengan sendirinya. Berbeda dengan MS Excell, Oracle dapat dikatakan lebih kompleks. Oracle terdiri dari beberpa bagian utama, diantaranya:

1. *Schema*

Oracle *database* dapat mengelompokkan beberapa tabel ke dalam suatu *schema*. *Schema* disini juga dapat dibatasi oleh hak akses (*username*). Untuk kebanyakan *database* Oracle mempunyai *default schema* sebagai berikut:

a. SYS
b. SYSTEM
c. OUTLN
d. BI, IX, HR, PM, OH

2. *Tablespace*

Tablespace merupakan tempat untuk menyimpan data yang dapat dibagi berdasarkan segmen-segmen. Jadi untuk pembuatan suatu *schema* dapat ditentukan akan disimpan di *tablespace* yang diinginkan.

Default tablespace dari Oracle adalah sebagai berikut:

a. SYSTEM
b. SYSAUX
c. TEMP

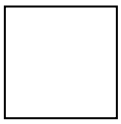
d. UNDOTBS1
e. USERS

2.5. *DATA FLOW DIAGRAM* (DFD)

DFD adalah representasi secara grafik untuk menjelaskan aliran data dari suatu organisasi. DFD digunakan untuk mendokumentasikan sistem yang sudah ada. Suatu DFD terdiri dari empat elemen utama yaitu *data sources* dan tujuan, *data flow*, proses transformasi dan penyimpanan data. Setiap elemen digambarkan dengan satu simbol. Keempat simbol ini digabungkan untuk menunjukan bagaimana suatu data diproses. Elemen tersebut digambarkan dengan simbol sebagai berikut:

2.5.1. *Data Sources* dan tujuan

Data Sources dan tujuan menggambarkan suatu organisasi atau individu yang mengirim atau menerima data yang digunakan atau dihasilkan oleh sistem. Sebuah *entity* dapat menjadi sumber atau tujuan dari data. Data sources dan tujuan digambarkan oleh suatu persegi (Gambar 2.9).

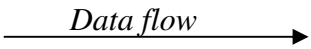


Gambar 2.9 *Data Sources*

Sumber: Kendall, Kenneth E., 2002.

2.5.2. *Data Flow*

Data flow menggambarkan aliran data antar proses, *data stores*, *data sources* dan tujuan. *Data flow* menggambarkan aliran data secara *logical*. Suatu data yang akan melewati *data stores* dan *data sources* atau tujuan harus melewati suatu proses terlebih dahulu yang dihubungkan dengan suatu *data flow*. *Data flow* digambarkan dengan panah (Gambar 2.10.) dimana untuk setiap aliran data diberi label untuk mengetahui aliran apa yang sedang terjadi dalam suatu kejadian.

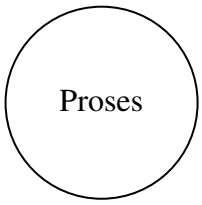


Gambar 2.10 *Data flows*

Sumber: Kendall, Kenneth E., 2002.

2.5.3. Proses

Suatu proses adalah suatu kejadian yang menyebabkan suatu data ditransformasi, disimpan atau di distribusikan. Contoh proses pembayaran, proses penggajian dll. Proses digambarkan dalam bentuk lingkaran dimana label akan dicantumkan didalam lingkaran tersebut untuk menunjukkan proses yang dilakukan.

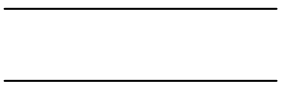


Gambar 2.11 Proses

Sumber: Kendall, Kenneth E., 2002.

2.5.4. *Data Stores*

Data stores adalah tempat penyimpanan data secara permanen maupun *temporary.* DFD tidak menunjukkan media penyimpanan secara fisik (kertas dan disk) untuk menyimpan data. Nama dari *data stores* adalah deskripsi dari data apa yang disimpan. Contoh data tentang *customer*, permintaan *customer*, dll. Simbol dari *data stores* adalah dua garis horisontal (Gambar 2.12).

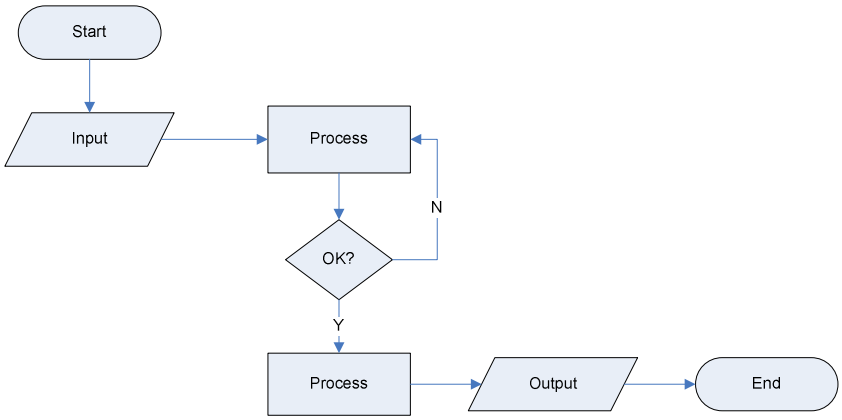


Gambar 2.12 *Data Stores*

Sumber: Kendall, Kenneth E., 2002.

2.6. *Flowchart*

Flowchart merupakan representasi grafik dari algoritma dengan menggunakan simbol –simbol tertentu yang masing-masing mempunyai fungsi yang khusus. *Flowchart* menggambarkan logika atau langkah dari sistem (proses, operasi, fungsi, atau aktifitas). *Flowchart* mempunyai beberapa simbol yang digunakan, contoh untuk *flowchart* ditunjukkan pada Gambar 2.13.



Gambar 2.13 *Flowchart*

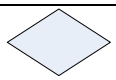
Sumber: Arsana, 2008.

Gambar 2.13 menunjukkan simbol-simbol dasar yang dipakai dalam pembuatan *flowchart*. Beberapa simbol lain yang dipakai dan penjelasannya dapat dilihat pada Tabel 2.1.

Tabel 2.1 Tabel Simbol *Flowchart*

Simbol	Nama	Keterangan
	Terminator	Untuk memulai atau mengakhiri program
	Garis Alir (Flow Line)	Arah aliran suatu proses dalam program
	Preparation	Proses inisialisasi/pemberian harga awal

Tabel 2.1 Tabel Simbol *Flowchart* (Lanjutan)

	Proses	Proses perhitungan/pengolahan data
	Input/Output Data	Proses masukkan/keluaran data, parameter, informasi
	Predefined Process (Sub Program)	Permulaan sub program/proses menjalankan sub program
	Decision (keputusan dalam pengujian)	Perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya
	On Page Connector	Penghubung bagian flowchart yang terpisah tetapi masih dalam satu halaman
	Off Page Connector	Penghubung bagian flowchart yang terpisah pada halaman berbeda