BAB III

LANDASAN TEORI

3.1 Definisi Capacity Management (Manajemen Kapasitas)

Manajemen Kapasitas adalah disiplin ilmu yang memastikan infrastruktur TI disediakan pada :

- Pada waktu yang tepat
- Dalam volume yang tepat
- Dengan harga yang tepat
- Dan memastikan bahwa TI digunakan dalam cara yang paling efisien.

Ini melibatkan *input* dari berbagai bidang usaha untuk mengidentifikasi :

- Layanan apa yang (atau akan) diperlukan
- Infrastruktur TI apa yang dibutuhkan untuk mendukung layanan ini
- Seberapa besar tingkat Kontingensi akan dibutuhkan, dan
- Berapa biaya infrastruktur ini.

Alasan Menggunakan Capacity Management :

- Bisnis ingin fokus pada Perencanaan Bisnis mereka
- TI dapat membuat sebuah Strategi Bisnis
- Laju perubahan dalam dunia bisnis sangat tinggi
- Hal ini menghasilkan peningkatan laju perubahan yang membutuhkan TI
- komponen, dan meningkatkan fleksibilitas.
- Bisnis semakin mengharapkan "On Demand" TI.
- Layanan terukur, Harga terjangkau, terukur skala waktunya.

- Ini adalah elemen kunci dalam Layanan baru Berorientasi Arsitektur (SOA) inisiatif. (Dennis Adams. 2004 : 2, 4)

3.2 Tujuan Capacity Management (Manajemen Kapasitas)

Untuk memastikan bahwa jumlah biaya yang disetujui selalu ada untuk jenis sumber daya berikut:

- Hardware (Sistem Operasi, Database, Power, Pemanas, Ruang Data Center)
- Peralatan Jaringan (LAN, WAN, Bridge, Router)
- Peripherals (Storage, Printer)
- Software (OS, jaringan SW)
- HR (di mana hal itu berdampak kapasitas TI)

Hal ini bertujuan untuk memastikan bahwa kapasitas ini cocok untuk saat ini dan kebutuhan bisnis masa depan. (Dennis Adams. 2004 : 3)

3.3 Definisi Capacity Management Information System (CMIS)

Capacity Management Information System (CMIS) atau Sistem Informasi Manajemen Kapasitas adalah kumpulan penggunaan infrastruktur TI, informasi kapasitas dan kinerja yang telah dikumpulkan secara konsisten dan disimpan dalam satu atau lebih database. Ini adalah satu buku catatan untuk semua penggunaan, kapasitas, dan data kinerja, lengkap terkait bisnis, aplikasi dan layanan Statistik. Setiap staf TI yang perlu akses ke kapasitas manajemen data berpotensi dapat menggunakan CMIS.

Proses manajemen layanan TI yang sering mengakses data CMIS adalah:

- Perencanaan kapasitas
- Manajemen kinerja
- Manajemen tingkat layanan
- Bantuan layanan meja
- Incident management
- Masalah manajemen
- Manajemen konfigurasi

Konsep CMIS adalah konsep baru dengan ITIL Versi 3. Dalam versi ITIL sebelumnya, Manajemen Kapasitas Database (CDB) adalah penyimpanan data pusat tetapi pengembang ITIL menyadari bahwa itu terlalu singkat dari apa yang diperlukan untuk membawa Manajemen Kapasitas ke tingkat berikutnya. CDB adalah koleksi data, tapi tidak ada standar mengenai koleksi dan arsip atau integrasi antara teknologi yang berbeda.

Proses Manajemen Kapasitas Versi 3 memiliki langkah-langkah baru untuk memastikan keakuratan dan integritas dari data. Sementara memperbarui proses, ITIL Versi 3 juga memperluas ruang lingkup informasinya yang disimpan, yang sekarang termasuk item tambahan seperti ramalan bisnis dan metrik. (Ron Potter. 2010 : 2)

3.4 Analisa dan Perancangan Sistem

Analisa sistem dilakukan dengan tujuan untuk mengidentifikasi dan mengevaluasi permasalahan yang terjadi dan kebutuhan yang diharapkan, sehingga dapat diusulkan perbaikannya. Perancangan sistem merupakan

penguraian suatu sistem informasi yang utuh ke dalam bagian komputerisasi yang dimaksud, mengidentifikasi dan mengevaluasi permasalahan, menentukan kriteria, menghitung konsistensi terhadap kriteria yang ada, serta mendapatkan hasil atau tujuan dari masalah tersebut serta mengimplementasikan seluruh kebutuhan operasional dalam membangun sebuah aplikasi.

Menurut Kendall (2003: 7), analisa dan perancangan sistem dipergunakan untuk menganalisis, merancang dan mengimplementasikan peningkatan-peningkatan fungsi bisnis yang dapat dicapai melalui penggunaan sistem informasi terkomputerisasi.

Tahap analisis merupakan tahap yang kritis dan sangat penting, karena kesalahan di dalam tahap ini juga akan menyebabkan kesalahan di tahap selanjutnya. Dalam tahap analisis sistem terdapat langkah-langkah dasar yang harus dilakukan oleh analis sistem sebagai berikut:

1. *Identify*, yaitu mengidentifikasi masalah.
2. *Understand*, yaitu memahami kerja dari sistem yang ada.
3. *Analyze*, yaitu menganalisis sistem.
4. *Report*, yaitu membuat laporan hasil analisis.

Setelah tahap analisis sistem selesai dilakukan, maka analis sistem telah mendapatkan gambaran dengan jelas apa yang harus dikerjakan. Tiba waktunya sekarang bagi analis sistem untuk memikirkan bagaimana membentuk sistem tersebut. tahap ini disebut desain sistem.

3.5 Konsep Dasar Sistem Informasi

Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran tertentu. Informasi adalah data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya. Data merupakan bentuk yang masih mentah yang belum dapat bercerita banyak, sehingga perlu diolah lanjut. (Jogiyanto, 1998, hal. 8)

Untuk memahami apa yang dimaksud dengan sistem informasi, kita perlu mendefnisikan istilah informasi dan sistem. Produk dari sistem informasi adalah informasi yang dihasilkan. Informasi tidak sama dengan data. Data adalah fakta, angka bahkan simbol mentah. Secara bersama-sama mereka merupakan masukan bagi suatu sistem informasi. Sebaliknya, informasi terdiri dari data yang telah ditransformasi dan dibuat lebih bernilai melalui suatu pemrosesan. Idealnya, informasi adalah pengetahuan yang berarti dan berguna untuk mencapai sasaran.

Sistem adalah suatu kerangka kerja terpadu yang mempunyai satu sasaran atau lebih. Sistem ini mengkoordinasi sumber daya yang dibutuhkan untuk mengubah masukan-masukan menjadi keluaran. Sumber daya dapat berupa bahan (material) atau mesin ataupun tenaga kerja, bergantung pada macam sistem yang dibahas. Sistem informasi karenanya adalah suatu kerangka kerja dengan mana sumber daya (manusia dan komputer) dikoordinasikan untuk mengubah masukan (data) menjadi keluaran (informasi), guna mencapai sasaran-sasaran perusahaan.

Definisi lain dari sistem informasi adalah sekumpulan hardware, software, brainware, prosedur dan atau aturan yang diorganisasikan secara integral untuk

mengola data menjadi informasi yang bermanfaat guna memecahkan masalah dan pengambilan keputusan. Sistem informasi adalah satu kesatuan data olahan yang terintegrasi dan saling melengkapi yang menghasilkan output baik dalam bentuk gambar, suara maupun tulisan.

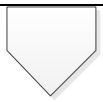
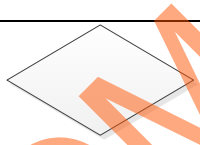
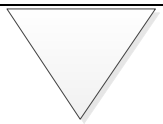
Sistem informasi adalah sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan.

3.6 Bagan Alir Dokumen

Bagan alir dokumen (*document flowchart*) atau di sebut juga bagan alir formulir (*form flowchart*) atau *paperwork flowchart* merupakan bagan alir yang menunjukkan dokumen gambaran arus data dengan menggunakan simbol seperti pada tabel berikut:

Tabel 3.1 Simbol-Simbol *Flowchart*

No.	Simbol	Nama Simbol *Flowchart*	Fungsi
1.		Dokumen	Untuk menujukkan dokumen input dan output baik untuk proses manual, mekanik atau komputer.

No.	Simbol	Nama Simbol *Flowchart*	Fungsi
2.		Proses Komputerisasi	Menunjukkan kegiatan dari operasi program komputer.
3.		Database	Untuk menyimpan data.
4.		Penghubung	Menunjukkan hubungan di halaman yang sama.
5.		Penghubung Halaman Lain	Menunjukkan hubungan di halaman lain.
6.		Terminator	Menandakan awal/akhir dari suatu sistem.
7.		Decision	Menggambarkan logika keputusan dengan nilai *true* atau *false*.
8.		Kegiatan Manual	Untuk menunjukkan pekerjaan yang dilakukan secara manual.
9.		Simpanan Offline	Untuk menujukkan file non-komputer yang diarsip urut angka.

3.7 *Data Flow Diagram* (DFD)

Menurut Kendall (2003: 241), *Data Flow Diagram* menggambarkan pandangan sejauh mungkin mengenai masukan, proses dan keluaran sistem, yang berhubungan dengan masukan, proses, dan keluaran dari model sistem yang dibahas. Serangkaian diagram aliran data berlapis juga bisa digunakan untuk merepresentasikan dan menganalisis prosedur-prosedur mendetail dalam sistem. Prosedur-prosedur tersebut yaitu konseptualisasi bagaimana data-data berpindah di dalam organisasi, proses-proses atau transformasi dimana data-data melalui, dan apa keluarannya. Jadi, melalui suatu teknik analisa data terstruktur yang disebut *Data Flow Diagram*, penganalisis sistem dapat merepresentasi proses-proses data di dalam organisasi. Menurut Kendall (2003: 265), dalam memetakan *Data Flow Diagram*, terdapat beberapa simbol yang digunakan antara lain:

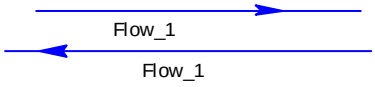
1. *External entity*

Suatu *external entity* atau entitas merupakan orang, kelompok, departemen, atau sistem lain di luar sistem yang dibuat dapat menerima atau memberikan informasi atau data ke dalam sistem yang dibuat.

Gambar 3.1 Simbol *External Entity*

2. *Data Flow*

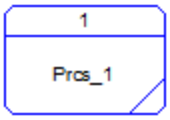
Data Flow atau aliran data disimbolkan dengan data tanda panah. Aliran data menunjukkan arus data atau aliran data yang menghubungkan dua proses atau *entitas* dengan proses.



Gambar 3.2 Simbol *Data Flow*

3. *Process*

Suatu proses dimana beberapa tindakan atau sekelompok tindakan dijalankan.



Gambar 3.3 Simbol *Process*

4. *Data Store*

Data store adalah simbol yang digunakan untuk melambangkan proses penyimpanan data.

Gambar 3.4 Simbol *Data Store*

3.8 Entity Relationship Diagram (ERD)

Entity relationship diagram (ERD) adalah gambaran pada sistem dimana di dalamnya terdapat hubungan antara *entity* beserta relasinya. *Entity* merupakan sesuatu yang ada dan terdefinisikan di dalam suatu organisasi, dapat abstrak dan nyata. Untuk setiap *entity* biasanya mempunyai *attribute* yang merupakan ciri *entity* tersebut. *Attribute* yaitu uraian dari entitas dimana mereka dihubungkan atau dapat dikatakan sebagai *identifier* atau *descriptors* dari entitas.

Entitas digolongkan menjadi *independent* atau *dependent entity*. *Independent entity* adalah apa yang tidak bersandar pada yang lain sebagai identifikasi. Suatu *dependent entity* adalah apa yang bersandar pada yang lain sebagai identifikasi. Selain digolongkan menjadi *independent* atau *dependent entity*, terdapat jenis- jenis entitas khusus yaitu:

1. *Associative Entity*

Associative Entity (juga dikenal sebagai *intersection entity*) adalah entitas yang *digunakan* oleh rekanan dua entitas atau lebih untuk menyatukan suatu hubungan benyak - ke - banyak (*Many to Many*)

2. *Subtypes Entity*

Subtypes Entity digunakan di dalam hierarki generalisasi (*generalization hierarchies*) untuk menyajikan suatu subset kejadian dari entitas orangtua, yang disebut *supertype*, tetapi yang memiliki atribut atau hubungan yang berlaku hanya untuk *subset*.

Menurut Marlinda (2004: 28), *atribute* sebagai kolom di sebuah relasi mempunyai macam-macam jenis *atribute* yaitu :

a. *Key Atribute*

Atribute ini merupakan *atribute* yang unik dan tidak dimiliki oleh *atribute* lainnya, misalnya entity mahasiswa yang *atribute*-nya NIM.

Gambar 3.5 *Key Attribute*

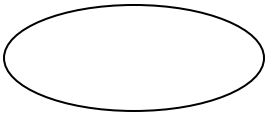
b. *Particial key Atribute*

Adalah *Attribute* yang tidak menjadi atau merupakan anggota dari *Key Primer*. Misalnya antara Cabang (toko) dan kode cabang.

Gambar 3.6 *Particial Key Attribute*

c. *Single Vallue Atribute*

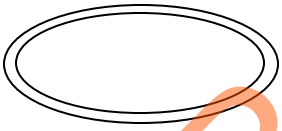
Atribute yang hanya memiliki satu nilai harga, misalnya *entity* mahasiswa dengan *atribute*-nya Umur (Tanggal lahir).



Gambar 3.7 *Single Value Attribute*

d. *Multi Vallue Atribute*

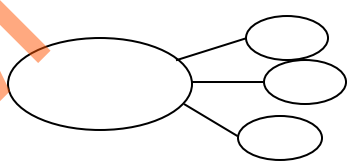
Atribute yang banyak memiliki nilai harga, misalnya *entity* mahasiswa dengan *atribute*-nya pendidikan (SD, SMP, SMA).



Gambar 3.8 *Multi Value Attribute*

e. *Composite Atribute*

Atribute yang memiliki dua harga, misalnya nama besar (nama kerja) dan nama kecil (nama asli)



Gambar 3.9 *Composite Attribute*

f. *Derived Attribute*

Attribute yang yang nilai-nilainya diperoleh dari pengolahan atau dapat diturunkan dari table *Attribute* atau table lain yang berhubungan.

Gambar 3.10 *Derived Attribute*

Model *Entity - Relationship* (ER) mula-mula diusulkan oleh Peter pada tahun 1976 sebagai cara untuk mempersatukan pandangan basis data jaringan dan relasional. Langkah sederhana dari model ER adalah model data konseptual yang memandang dunia nyata sebagai kesatuan (*entities*) dan hubungan (*relationship*).

Komponen dasar model merupakan diagram *entity-relationship* yang digunakan untuk menyajikan objek data secara *visual*. *Entity Relationship Diagram* mengilustrasikan struktur logis dari basis data yang mempunyai metodologi sebagai berikut:

Tabel 3.2 Ilustrasi pembuatan ERD

Proses	**Keterangan**
1. Menentukan Entitas	Menentukan peran, kejadian, lokasi, hal nyata, dan konsep dimana pengguna akan menyimpan data.
2. Menentukan Relasi	Tentukan hubungan antara pasangan entitas menggunakan matriks relasi.
3. Gambar ERD Sementara	Entitas digambarkan dengan kotak dan relasi dengan garis yang menghubungkan entitas.
4. Isi Kardinalitas	Tentukan jumlah kejadian dari satu entitas untuk sebuah kejadian pada entitas yang berhubungan.
5. Tentukan Kunci Utama	Tentukan atribut yang mengidentifikasi satu dan hanya satu kejadian pada masing-masing entitas.

6. Gambar ERD berdasar Kunci	Hilangkan relasi *Many-to-Many* dan masukkan *primary* dan kunci tamu pada masing-masing entitas.
7. Menentukan Atribut	Tuliskan *field-field* yang diperlukan oleh sistem.
8. Pemetaan Atribut	Pasangkan atribut dengan satu entitas yang sesuai pada masing-masing atribut.
9. Gambar ERD dengan Atribut	Aturlah ERD dari langkah 6 dengan menambahkan entitas atau relasi yang ditemukan pada langkah 8.
10. Periksa Hasil	Apakah ERD sudah menggambar sistem yang akan dibangun.

Entity Relationship Diagram ini diperlukan agar dapat menggambarkan hubungan antar *entity* dengan jelas, dapat menggambarkan batasan jumlah *entity* dan partisipasi antar *entity*, mudah dimengerti pemakai dan mudah disajikan oleh perandang *database*. Untuk itu, *entity relationship diagram* dibagi menjadi dua jenis model, yaitu:

1. *Conceptual Data model*

Conceptual Data model (CDM) adalah jenis model data yang menggambarkan hubungan antar tabel secara konseptual.

2. *Physical Data Model*

Physical Data Model (PDM) adalah jenis model data yang menggambarkan hubungan antar tabel secara fisikal.

3.9 Use Case Diagram

Use case adalah rangkaian/uraian sekelompok yang saling terkait dan membentuk sistem secara teratur yang dilakukan atau diawasi oleh sebuah aktor. Use case digunakan untuk membentuk tingkah-laku benda/ things dalam sebuah model serta di Realisasikan oleh sebuah collaboration.

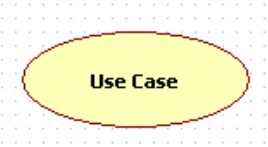
Umumnya use case digambarkan dengan sebuah elips dengan garis yang solid, biasanya mengandung nama.Use case menggambarkan proses system (kebutuhan system dari sudut pandang user).Secara umum use case adalah:

- Pola perilaku system
- Urutan transaksi yang berhubungan yang dilakukan oleh satu actor

Use case diagram terdiri dari :

- Use case

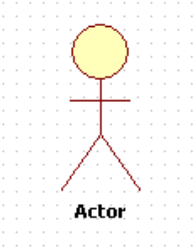
  Use case adalah gambaran fungsionalitas dari suatu sistem, sehingga *customer* atau pengguna sistem paham dan mengerti mengenai kegunaan sistem yang akan dibangun



Gambar 3.11 Use Case

- Actors

  Actor tersebut mempresentasikan seseorang atau sesuatu (seperti perangkat, sistem lain) yang berinteraksi dengan sistem.



Gambar 3.12 Aktor

- Relationship

  Menunjukan hubungan antara sebuah use case dengan aktor. Ada 3 jenis relationship yaitu :

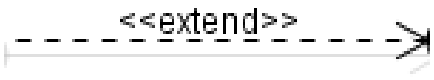
  - Assotiation Relationship

Gambar 3.13 Hubungan Asosiasi

  - Include Relationship

Gambar 3.14 Hubungan Include

  - Extends Relationship



Gambar 3.15 Hubungan Extends

- System boundary boxes (optional)
- Packages (optional)

3.10 Program Penunjang

Untuk membuat rancang bangun aplikasi penggajian pada Dealer Honda Saver, dibutuhkan beberapa perangkat lunak untuk memudahkan perancangan *design* maupun sistem. Perangkat lunak tersebut antara lain:

3.10.1 *Power Designer*

Power designer merupakan suatu *tool* berupa *software* untuk mendesain sistem dan rancangan *Entity Relationship Diagram* (ERD) yang dikembangkan oleh *Sybase Inc*. Ada dua model data, yaitu: *Entity Relationship Diagram* (ERD) dan model relasional. Keduanya menyediakan cara untuk mendeskripsikan perancangan basis data pada peringkat logika.

a. Model ERD atau *Conceptual Data Model* : model yang di buat berdasarkan anggapan bahwa dunia nyata terdiri dari koleksi obyek-obyek dasar yang dinamakan entitas serta hubungan (*relationship*) antara entitas-entitas itu.
b. Model Relasional atau *Physical Data Model* : model yang menggunakan sejumlah tabel untuk menggambarkan data serta hubungan antara data-data tersebut. Setiap tabel mempunyai sejumlah kolom dimana setiap kolom memiliki nama yang unik.

3.10.2 *Visual Basic .NET*

Microsoft Visual Basic .NET adalah sebuah alat untuk mengembangkan dan membangun aplikasi yang bergerak diatas sistem *.NET Fremawork*, dengan menggunakan bahasa *BASIC*. Dengan menggunakan alat ini, para pembuat program dapat membangun aplikasi *Windows Forms*. Alat ini dapat diperoleh secara terpisah dari beberapa produk lainnya (seperti *Microsoft Visual C++*,

Visual C#, atau *visual j#*) atau juga dapat diperoleh secara terpadu dalam *Microsoft visual Studion .NET* (Yuwanto, 2005).

Bahasa *Visual Basic .NET* sendiri menganut paradigma bahasa pemrograman berorientasi objek yang dapat dilihat sebagai evolusi dari *Microsot Visual Basic* versi sebelumnya yang dimplementasikan diatas *.NET Framework*. Peluncurannya mengundang kontrovensi, mengingat banyak sekali perubahan yang dilakukan oleh *Microsoft*, dan versi baru ini tidak kompatibel dengan versi terdahulu.

3.10.3 *NET Framework*

Microsoft .NET Framework (di baca *Microsoft dot Net Framework*) adalah sebuah komponen yang dapat ditambahkan ke sistem operasi *Microsoft Windows* atau telah terintegrasi ke dalam *Windows* (mulai dari *Windows server* 2003 dan versi-versi *Windows* Terbaru). Kerangka kerja ini menyediakan sejumlah besar solusi-solusi program untuk memenuhi kebutuhan-kebutuhan umum suatu program baru, dan mengatur eksekusi program-program yang ditulis secara khusus untuk framework ini. *.NET Framework* adalah kunci penawaran utama dari *Microsoft*, dan dimaksudkan untuk digunakan oleh sebagian besar aplikasi-aplikasi baru yang dibuat untuk *platform Windows* (Yuwanto, 2005).

Pada dasarnya, *.NET framework* memiliki 2 komponen utama: *CLR* dan *.NET Framework Class Library*. Program-program yang ditulis untuk *.NET Framework* dijalankan pada suatu lingkungan *software* yang mengatur persyaratan-persyaratan *runtime* program. *Runtime environment* ini, yang juga merupakan suatu bagian dari *.NET Framework*, dikenal sebagai *Common Language Runtime* (CLR). CLR menyediakan penampilan dari *application virtual*

machine, sehingga para *programmer* tidak perlu mengetahui kemampuan CPU tertentu yang akan menjalankan program. CLR juga menyediakan layanan-layanan penting lainnya seperti jaminan keamanan, pengaturan *memori*, *garbage collection* dan *exception handling* atau penanganan kesalahan pada saat *runtime*.

Class Library dan CLR ini merupakan komponen inti dari *.NET Framework*. Kerangka kerja itu pun dibuat sedemikian rupa agar para *programmer* dapat mengembangkan program komputer dengan jauh lebih mudah, dan juga untuk mengurangi kerawanan aplikasi dan juga komputer dari beberapa ancaman keamanan. CLR adalah turunan dari CLI (*Common Language Infrastructure*) yang saat ini merupakan standar ECMA.

Solusi-solusi program pembentuk *Class Library* dari *.NET Framework* mengcover area yang luas dari kebutuhan program pada bidang *user interface*, pengaksesan data, koneksi basis data, *kriptografi*, pembuatan aplikasi berbasis *web*, *algoritma numerik*, dan komunikasi jaringan. Fungsi-fungsi yang ada dalam *class library* dapat digabungkan oleh *programmer* dengan kodenya sendiri untuk membuat suatu program aplikasi baru

3.10.4 *SQL Server 2008*

Microsoft SQL Server adalah sebuah sistem manajemen basis data relasional (RDBMS) produk *Microsoft*. Bahasa kueri utamanya adalah *transact-SQL* yang merupakan implementasi dari *SQL standar ANSI/ISO* yang digunakan oleh *Microsoft* dan *Sybase*. SQL (*Structured Query Language*) adalah sebuah bahasa yang dipergunakan untuk mengakses data dalam basis data relasional (Yuwanto, 2007).

Umumnya *SQL Server* digunakan di dunia bisnis yang memiliki basis data berskala kecil sampai dengan menengah, tetapi kemudian berkembang dengan digunakannya *SQL Server* pada basis data besar. Penulis menggunakan *SQL Server 2010* untuk merancang database yang digunakan pada sistem.

3.10.5 Crystal Report

Merupakan *software* yang digunakan untuk pembuatan laporan. Dengan cara mengoneksi nama tabel yang akan dibuatkan laporannya. Setelah tampilan data ada maka klik dan *drag* semua *field* yang ada sesuai dengan tampilan yang diinginkan. Biasanya *crystal report* adalah komponen dari *VB.NET*.

3.10.6 Highcharts

Highcharts adalah Perpustakaan grafik yang ditulis dalam JavaScript yang murni, yang menawarkan cara mudah untuk menambahkan grafik interaktif ke situs web atau aplikasi web. Highcharts saat ini mendukung garis, spline, area, areaspline, kolom, bar, pie, bubar, alat pengukur sudut, arearange, areasplinerange, columnrange, bubble, box plot, error bar, funnel, waterfall dan (polar chart types). (Torstein, 2011)