BAB II

TINJAUAN PUSTAKA

II.1. Konsep Sistem

Sistem merupakan kumpulan elemen yang saling berhubungan satu sama lain yang membentuk satu kesatuan dalan usaha mencapai suatu tujuan. Informasi adalah hasil pemprosesan data yang diperoleh dari setiap elemen sistem tersebut menjadi bentuk yang mudah dipahami dan merupakan pengetahuan yang relevan yang dibutuhkan oleh orang untuk menambah pemahamannya tehadap fakta-fakta yang ada (Budi sutejo : 2006 : 168).

II.1.1. Sistem Informasi

Sistem informasi secara sederhana dapat diartikan sebagai kumpulan dari beberapa komponen yang saling berinteraksi untuk mencapai hasil dari satu tujuan. Pengertian sederhana ini sesuai dengan pendapat O'Brien (2006: 5)" Sistem Informasi dapat merupakan kombinasi teratur apapun dari orang-orang, *hardware*, *software*, jaringan komunikasi, dan sumber daya data yang mengumpulkan, mengubah dan menyebarkan informasi dalam sebuah organisasi". Menurut Whitten (2004:12) *"information system is an arrangement of people, data, process and information technology that interact to collect, process, store, and provide as output the information needed to support an organization*". Defenisi tersebut dapat dijelaskan sistem informasi adalah susunan dari orang,

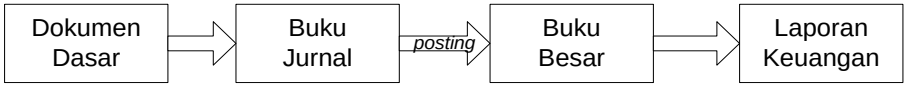
data, pemprosesan dan teknologi informasi yang dibutuhkan untuk mendukung sebuah organisasi (Henny Hendarti: 2011: 1)

II.2 Sistem Informasi Akuntansi

Menurut Rudianto (2009: 4) Akuntansi adalah sebuah sistem informasi yang menghasilkan informasi keuangan kepada pihak-pihak yang berepentingan mengenai aktivitas ekonomi dan kondisi suatu perusahaan.

Akuntansi adalah aktivitas mengumpulkan, menganalisis, menyajikan dalam bentuk angka, mengklasifikasikan, mencatat, meringkas dan melaporkan aktivitas/transaksi perusahaan dalam bentuk informasi keuangan.

Dengan demikian, untuk sampai pada penyajian informasi keuangan yang dibutuhkan berbagai pihak, maka akuntansi harus melewati suatu proses yang disebut dengan siklus akuntansi. Siklus akuntansi adalah urutan kerja yang harus dibuat oleh akuntan, sejak awal hingga menghasilkan laporan keuangan suatu perusahaan.



Gambar II.1 Siklus Akuntansi

Sumber : Rudianto (2009 : 14)

1. **Dokumen Dasar** adalah bukti transaksi yang dijadikan dasar oleh akuntan untuk mencatat, seperti : faktur, kuitansi, nota penjualan, *invoice*, dll.
2. **Jurnal** (*Journal*) adalah aktivitas meringkas dan mencatat transaksi perusahaan berdasarkan dokumen dasar. Tempat untuk mencatat dan meringkas transaksi disebut dengan Buku Jurnal.

3. **Posting** adalah aktivitas memindahkan catatan di buku jurnal ke dalam buku besar sesuai dengan jenis transaksi dan nama perkiraan masing-masing.
4. **Buku Besar** (*General Ledger*) adalah kumpulan dari semua akun/perkiraan yang dimiliki suatu perusahaan yang saling berhubungan satu dengan lainnya dan merupakan suatu kesatuan.
5. **Akun/Perkiraan** (*Account*) adalah suatu kelas informasi di dalam suatu sistem akuntansi. Atau, suatu media yang digunakan untuk mencatat informasi sumber daya perusahaan dan informasi lainnya berdasarkan jenisnya. Misalnya perkiraan kas, perkiraan piutang, akun modal, dsb.

Sistem informasi akuntansi (SIA) adalah sistem berbasis komputer yang dirancang untuk mentransformasi data akuntansi menjadi informasi (george H. Bodnar dan Wlliam S. Hopwood : 2006 : 8).

II.3. Unified Modeling Language (UML)

UML (*Unified Modeling Language*) adalah suatu alat Bantu yang sangat handal di dunia pengembangan sistem yang berorientasi objek (Munawar ; 2005 : 17). Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembangan sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan mereka dengan yang lain.

Meskipun UML sudah banyak menyediakan diagram yang bisa membantu mendefenisikan suatu aplikasi, tidak berarti bahwa semua diagram tersebut akan

bisa menjawab persoalan yang ada. Adapun tipe diagram UML yang ada seperti pada Tabel II.1.

Tabel II.1 Tipe Diagram UML

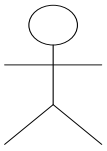
Diagram	Tujuan	Keterangan
Activity	Prilaku prosedural dan paralel	Sudah ada di UML 1
Class	Class, fitur dan relasinya	Sudah ada di UML 1
Communication	Interaksi diantara objek. Lebih menekankan kepada link	Di UML 1 disebut collaboration
Component	Struktur dan koneksi dari komponen	Sudah ada di UML 1
Composite Structure	Dekomposisi sebuah class saat runtime	Baru untuk UML 2
Deployment	Penyebaran/instalasi ke klien	Sudah ada di UML 1
Interaction Overview	Gabungan dari activity dan sequence diagram	Baru untuk UML 1
Object	Contoh konfigurasi instance	Tidak resmi ada di UML 1
Package	Struktur hierarki saat kompilasi	Tidak resmi ada di UML 1
Sequence	Interaksi antara objek. Lebih menekankan pada urutan.	Sudah ada di UML 1
State Machine	Bagaimana event mengubah sebuah objek	Sudah ada di UML 1
Timing	Interaksi antar objek. Lebih menekankan pada waktu	Sudah ada di UML 1
Use Case	Bagaimana user berinteraksi dengan sebuah sistem	Sudah ada di UML 1

Sumber :Munawar (2009 : 23)

II.3.1. Notasi Dasar UML

1. Actor

Actor adalah *abstraction* dari orang dan *system* yang lain yang mengaktifkan fungsi dari target *system*. Orang atau system bisa muncul dalam bebrapa peran. Perlu dicatat bahwa *actor* berinteraksi dengan *use case*, tetapi tidak memiliki kontrol atas *use case*. Berikut notasi actor dalam UML :

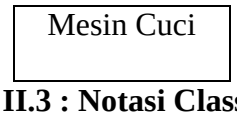


Gambar II.2 : Notasi Actor pada UML

Sumber : Munawar (2009 : 64)

2. *Class*

Calss, dalam notasi UML digambarkan dengan kotak. Nama class menggunakan huruf besar diawal kalimatnya dan diletakkan diatas kotak. Bila calass mempunyai nama yang terdiri dari 2 suku kata atau lebih, maka semua suku kata digabungkan tanpa spasi dengan huruf awal tiap suku kata menggunakan huruf besar. Berikut notasi class dalam UML:



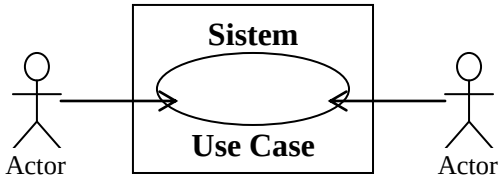
Gambar II.3 : Notasi Class di UML

Sumber : Munawar (2009 : 35)

3. Use Case

Use Case adalah alat bantu terbaik guna menstimulasi pengguna potensial untuk mengatakan tentang suatu *system* dari sudut pndangnya. Tidak selalu mudah bagi pengguna untuk menyatakan bagaimana mereka bermaksud

menggunakan sebuah *system*. Karena system pengembangan tradisional sering ceroboh dalam melakukan analisis, akibatnya pengguna seringkali susah menjawabnya tatkala dimintai masukan tentang sesuatu. Notasi *use cas*e dapat dilihat pada gambar II.3 :



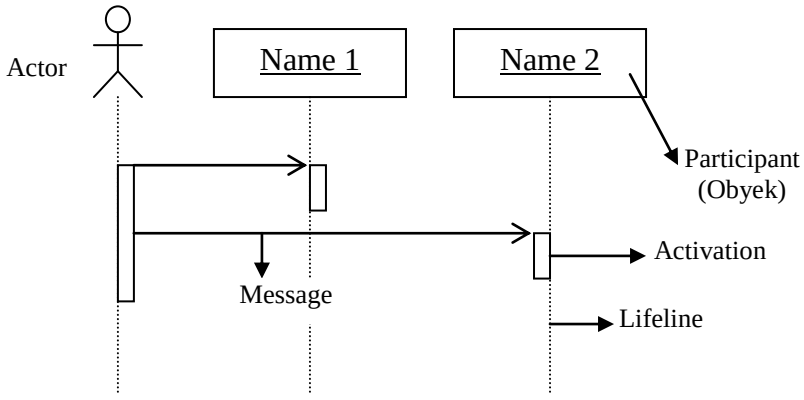
Gambar II.4 : Notasi Use Case pada UML

Sumber : (Munawar ; 2009 : 64)

4. Sequence Diagram

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah scenario. Diagram ini menunjukkan sejumlah contoh obyek dan *message* (pesan) yang diletakkan diantara obyek-obyek ini dalam *use case*.

Komponen utama squence diagram terdiri atas obyek yang dituliskan dengan kotak segiempat bernama. Message dengan tanda panah dan waktu yang ditunjukkan dengan progress vertical. Berikut Contoh *sequence diagram* :



Gambar II.5 : Simbol-simbol yang ada pada Sequence Diagram

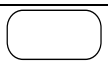
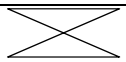
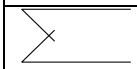
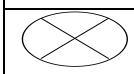
Sumber : Munawar (2009 : 89)

5. Activity Diagram

Activity diagram adalah teknik untuk mendiskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity Diagram* mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaannya dengan *flowchart* adalah *activity diagram* bisa mendukung prilaku paralel sedangkan *flowchart* tidak bisa.

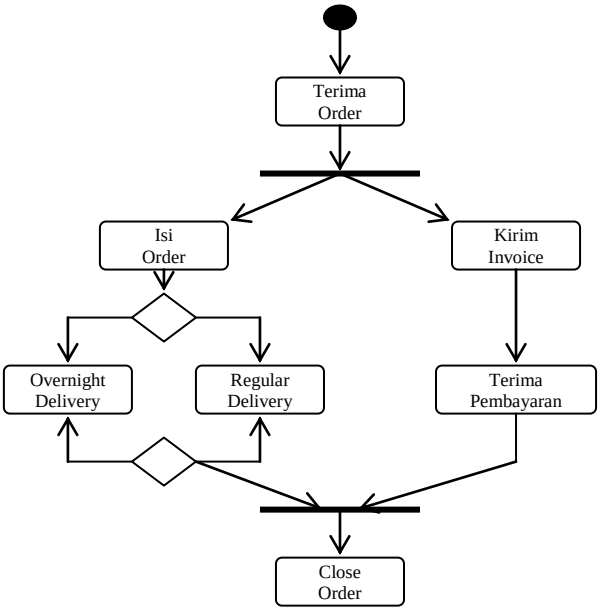
Berikut adalah simbol-simbol yang sering digunakan pada saat pembuatan activity diagram.

Tabel II.2 Simbol-simbol yang sering dipakai pada activity diagram

Simbol	Keterangan
	Titik awal
	Titik akhir
	Activity
	Pilihan untuk pengambilan keputusan
	Fork; digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu.
	Rake; menunjukkan adanya dekomposisi
	Tanda waktu
	Tanda pengiriman
	Tanda penerimaan
	Aliran akhir (Flow Final)

Sumber : Munawar (2009 : 109)

Adapun contoh dari Activity Diagram dapat di lihat pada Gambar II.8.



Gambar II.6 : Contoh Activity Diagram Sederhana

Sumber : Munawar (2009 : 111)

II.4. Pengertian Database

Database merupakan komponen terpenting dalam pembangunan SI, karena menjadi tempat untuk menampung dan mengorganisasikan seluruh data yang ada dalam sistem, sehingga dapat dieksplorasi untuk menyusun-menyusun informasi-informasi dalam berbagai bentuk. *Database* merupakan himpunan kelompok data yang saling berkaitan. Data tersebut diorganisasikan sedemikian rupa agar tidak terjadi duplikasi yang tidak perlu, sehingga dapat diolah atau di eksplorasi secara cepat dan mudah untuk menghasilkan informasi (Budi Sutedjo Dharma Oetomo : 2006: 99).

II.4.1. Hierarki Data Dalam Database

Data dalam sebuah *database* disusun berdasarkan sistem hierarki yang unik, yaitu:

1. **Database**, merupakan kumpulan file yang saling terkait satu sama lain, misalnya file data induk karyawan, file jabatan file penggajian dan lain sebagainya. Kumpulan file yang tidak saling terkait satu sama lain tidak dapat disebut *database*, misalnya file data induk karyawan, file tamu undangan perkawinan, file barang retail pasar swalayan.
2. **File**, yaitu kumpulan dari *record* yang saling terkait dan memiliki format *field* yang sama dan sejenis.
3. **Record**, yaitu kumpulan *field* yang menggambarkan suatu unit data individu tertentu.
4. **Field**, yaitu atribut dari *record* yang menunjukkan suatu item dari data, seperti nama, alamat, dan lain sebagainya.
5. **Byte**, yaitu atribut dari *field* yang berupa huruf yang membentuk nilai dari sebuah *field*. Huruf tersebut dapat berupa numerik maupun abjad atau karakter khusus
6. **Bit**, yaitu bagian terkecil dari data secara keseluruhan, yaitu berupa karakter ASCII nol atau satu yang merupakan komponen pembentuk *byte* (Budi Sutedjo Dharma Oetomo: 2006: 102).

II.4.2. SQL Server 2005

RDBMS memiliki kepanjangan *Relational Database Management System*. Merupakan salah satu produk andalan yang dibuat oleh *Microsoft* yang berfungsi sebagai *relational database.* Entah mengapa nama *Microsoft SQL Server* tersebut lebih mengena dengan sebutan *SQL Server*. Mungkin memang karena penyebutannya yang sudah tak asing lagi ditelinga kita. Penulis pun kadang-kadang suka menyebutnya dengan *SQL Server*. *Microsoft SQL Server* mendukung *SQL* sebagai bahasa pemroses *query*. Seperti yang kita ketahui, *SQL* merupakan bahasa standar *international* untuk proses *query database* dan *SQL* ini sudah banyak sekali digunakan pada hampir semua aplikasi, baik itu *e-commerce*, pendidikan, organisasi, pemerintah, atau bahkan personal sekalipun. Sehingga dengan demikian, *SQL Server* patut kita pertimbangkan sebagai *database* program kita (Agus Saputra: 2012: 11).

II.5. Kamus Data

Kamus data (KD) atau *data dictionary* (DD) atau disebut juga dengan istilah *systems data dictionary* adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi. Dengan menggunakan KD, analisis sistem dapat mendefinisikan data yang mengalir di sistem dengan lengkap. KD dibuat pada tahap analisis sitem dan digunakan baik pada tahap analisis maupun pada tahap perencanaan sistem (Jogiyanto: 2005: 725).

Tabel II.3 Notasi Kamus Data

Notasi	Arti
=	Terbentuk dari (is composed) atau terdiri dari (consist of) atau sama dengan (is equivalent of)
+	AND
[]	Salah satu dari (memilih salah satu dari elemen-elemen data di dalam kurung bracket ini)
|	Sama dengan simbol []
M{ }M	Intensi (elemen data didalam kurung brace berinterasi mulai minimum N kali dan maksimum M kali)
()	Optional (elemen data di dalam kurung parenthesis sifatnya optional, dapat ada dan dapat tidak ada)
*	Keterangan setelah tanda ini adalah komentar

Sumber : Jogiyanto (2005: 730)

II.6. Entity Relationship Diagram – ERD

II.6.1. Model-model Data

Struktur yang mendasari suatu basisdata adalah model data yang merupakan kumpulan alat-alat konseptual untuk mendeskripsikan data, relasi data, data semantik, dan batasan konsistensi. Untuk mengilustrasikan konsep model data, berikut disajikan dua model data, yaitu *entity relationship model* dan *relational model.* Kedua model menyediakan cara mendeskripsikan rancangan basisdata pada tingkatan logis.

II.6.2. Entity Relationship Model

Entity Relationship (ER) data model didasarkan pada persepsi terhadap dunia nyata yang tersusun atas kumpulan objek-objek dasar yang disebut entitas dan hubungan antar objek. Entitas adalah sesuatu atau objek dalam dunia nyata yang dapat dibedakan dari objek lain. Sebagai contoh, masing-masing mahasiswa adalah entitas dan mata kuliah dapat pula dianggap sebagai entitas.

Entitas digambarkan dalam basisdata dengan kumpula atribut. Misalnya atribut nim, nama, alamat dan kota bisa menggambarkan data mahasiswa tertentu dalam suatu universitas. Atribut-atribut membentuk entitas mahasiswa. Demikian pula, atribut kodeMK, namaMK, dan SKS mendeskripsikan entitas mata kuliah.

Atribut NIM digunakan untuk mengidentifikasi mahasiswa secara unik karena dimungkinkan terhadap dua mahasiswa dengan nama, alamat, dan kota yang sama. Pengenal unik harus diberikan pada masing-masing mahasiswa.

Relasi adalah hubungan antara beberapa entitas. Sebagai contoh, relasi menghubungkan mahasiswa dengan mata kuliah yang di ambilnya. Kumpulan semua entitas bertipe sama disebut kumpulan entitas (*entity set)*, sedangkan kumpulan semua relasi bertipe sama disebut kumpulan relasi (*relationship set)*.

Struktur logis (skema *database*) dapat ditunjukkan secara grafis dengan diagram ER yang dibentuk dari komponen-komponen berikut :

Tabel II.4 Notasi ERD (Entity Relationship Diagram)

	Persegi panjang mewakili kumpulan entitas
	Elips mewakili atribut
	Belah ketupat mewakili relasi
	Garis menghubungkan atribut dengan kumpulan entitas dan kumpulan entitas dengan relasi

Sumber : Janner Simarmata & Imam Prayudi (2006 :59)

II.7. Normalisasi

Normalisasi adalah teknik perancangan yang banyak digunakan sebagai pemandu dalam merancang basis data relasional. Pada dasarnya, normalisasi adalah proses dua langkah yang meletakkan data dalam bentuk tabulasi dengan menghilangkan kelompok berulang lalu menghilangkan data yang terduplikasi dari tabel relational.

Teori normalisasi didasarkan pada konsep bentuk normal. Sebuah tabel relasional dikatakan berada pada bentuk normal tertentu jika tabel memenuhi himpunan batasan tertentu. Ada lima bentuk normal yang telah ditemukan.

1. **Bentuk Normal Pertama (1NF/*First Normal Form*)**, bentuk normal pertama adalah suatu bentuk relasi dimana atribut bernilai banyak (*multivalues atribute*) telah dihilangkan sehingga kita akan menjumpai nilai tunggal (mungkin saja nilai *null*) pada perpotongan setiap baris dan kolom satu nilai untuk irisan baris dan kolom pada tabel.

2. **Bentuk Normal Kedua (2NF/*Second Normal Form*)**, semua kebergantungan fungsional (*functional dependeny*) yang bersifat sebagian (*partial functional dependeny*) telah dihilangkan.
3. **Bentuk Normal Ketiga (3NF/*Third Normal Form*)**, semua kebergantungan transitif (*transitive dependeny*) telah dihilangkan.
4. **Boyce-Codd Normal Form (BCNF/*Boyce-Codd Normal Form*)**, semua anomali yang tersisa dari hasil penyempurnaan kebergantungan fungsional (*functional dependeny*) diatas telah dihilangkan.
5. **Bentuk Normal Keempat (4NF/*Fifth Normal Form*)**, semua anomali yang berasal dari kebergantungan banyak-nilai (*multivalues dependeny*) telah dihilangkan (Adi Nugroho; 2010: 34).

Tujuan normalisasi adalah membuat kumpulan tabel relasional yang bebas dari data berulang yang dapat dimodifikasi secara benar dan konsisten. Ini berarti bahwa semua tabel pada basisdata relasional harus berada pada bentuk normal ketiga (3NF). Sebuah tabel relasional berada pada 3NF jika dan hanya jika semua kolom bukan kunci adalah (a) saling independen dan (b) sepenuhnya tergantung pada kunci utama. Saling independen berarti bahwa tidak ada kolom bukan kunci yang tergantung pada senbarang kombinasi kolom lainnya. Dua bentuk normal pertama adalah langkah antara untuk mencapai tujuan, yaitu mempunyai semua tabel dalam 3NF (Janner Simarmata & Imam Prayudi: 2006: 77).

II.8. Bahasa Pemrograman Visual Basic 2008

Visual Basic 2008 atau *visual Basic 9* merupakan paket teknologi bahasa pemprograman dari *visual studio* yang dikembangkan oleh Microsoft. Bahasa

visual basic digunakan untuk membuat aplikasi *Windows* yang berbasis GUI (*Graphical User Interface*). GUI adalah sebuah aplikasi yang menampilkan antarmuka secara grafis. Hal ini memudahkkan *user*/pemakai dalam mengoperasikan aplikasi.

Visual basic merupakan salah satu bahasa *Object Oriented Programming* (OOP) dimana pemprograman di fokuskan pada suatu objek tertentu. Selain itu, *Visual Basic* juga dikenal dengan sebutan *Event Driven Programming*, yaitu program yang akan bekerja setelah ada respon dari *user* berupa *event*/kejadian tertentu (seperti tombol diklik, menu dipilih, memasukkan data pada *text field*, dan lain-lain). Ketika sebuah *event* terdeteksi, kode yang berada pada objek yang mendapat *event* akan dijalankan.

II.8.1. Fitur Baru Visual Basic 2008

Visual Basic 2008 mempunyai beberapa fitur baru dalam kompiler dan bahasa, IDE (*Integrated Development Environments*), akses data dan lain-lain. Fitur baru dalam dukungannya terhadap akses data/aplikasi berbasis *database* adalah sebagai berikut :

1. Language- Integrated Query (LINQ)

   Language-Integrated Query (LINQ) adalah sebuah teknologi baru pada *Visual Basic* 2008 yang mendukung sintaks *query* dan bahasa baru lainnya. LINQ membuat seolah-olah sebuah *query database* merupakan bagian dari sintaks/perintah *Visual Basic*. Hal ini menambah kemampuan *query Visual Basic* dengan perintah yang sederhana dan kemampuan yang tangguh saat bekerja dengan macam-macam data. LINQ memungkinkan anda melakukan

query data dari *databse SQL Server*, dokumen XML, data *array* dan *collections*, ADO.NET *datasets*, dan banyak lagi *remote* atau sumber data lokal yang mendukung LINQ.

2. Object Relational Designer (O/R Designer)

   Object Relational Designer atau disingkat dengan O/R *Designer* menyediakan desain visual untuk membuat kelas *entity* LINQ to SQL dan menghubungkannya berdasarkan objek pada *database*. Fitur ini akan memanduan para pengembang untuk membuat dan mengedit objek LINQ to SQL yang menghubungkan antara kode aplikasi dengan database.

3. N-Tier

   N-Tier mendukung penulisan datasets dengan menyediakan tingkatan (strata) lapisan *Dataset Designer* yang membuat lebih mudah untuk memisahkan kode *dataset* ke dalam *Table Adapter* dan menuliskan kode *dataset* kedalam proyek terpisah. N-Tier juga disebut sebagai “aplikasi” terdistribusi “dan“ aplikasi “*multitier*”. Aplikasi N-Tier memisahka pengolahan/pengaksesan data ke dalam tingkatan-tingkatan yang menyusun aplikasi secara terpisah yang didistribusikan antar klien dan *server*.

4. Peningkatan kemampuan hierarki

   Hierarchical update capabilities (peningkatan kemampuan hierarki) telah disatukan ke dalam *Dataset Designer*. Peningkatan ini menyediakan penciptaan kode yang menyimpan logika yang dibutuhkan untuk pemeliharaan integritas referensi antar tabel yang berkaitan di dalam suatu *database*.

5. Local Database Caching

   Local database caching menyertakan *database SQL Server Cmpact* 3.5 ke dalam sebuah aplikasi dan mengatur sinkronisasi periodik data dengan *remote database* pada *server*. *Local database caching* memungkinkan aplikasi mengurangi banyaknya *roundtrips* (keluar-masuk data) antara aplikasi dengan *server database*, ketika data jarang berubah atau saat aplikasi berhubungan dengan database *server*.

6. Microsoft SQL Server Compact 3.5

   Microsoft SQL Server Compact 3.5 adalah sebuah *database compact* yang dapat digunakan pada *computer desktop*, *smart devices*, dan *Tablet PCs. SQL Server Compact* 3.5 menyediakan model pemprograman yang umum untuk mengembangkan dan mengatur aplikasi (Andi: 2009: 2).