BAB 2
TINJAUAN PUSTAKA

2.1 Profil Perusahaan

PT. Matahari Sentosa didirikan pada tahun 1985 oleh Mr. Sung Yaw Kwok. PT. Matahari Sentosa telah berkembang dengan pesat untuk dikenal sebagai salah satu produsen resleting terbesar dan terbaik dan eksportir. Mr. Sung Yaw Kwok adalah investor asing. Beliau membawa pengalaman dan teknologi formosanya dari Republik Cina (Taiwan) ke Indonesia. PT. Matahari Sentosa sekarang dikenal atas pelayanan mereka yang baik dan produk yang berkualitas tinggi. Semua barang diproduksi secara hati-hati dari bahan baku sampai jadinya resleting terhadap Asia, Eropa, Timur Tengah, dan Amerika Selatan.

PT. Matahari Sentosa menempati lebih dari setengah juta meter persegi di Bandung Indonesia dan dengan pabrik yang ada di Shenzhen Cina. Pabrik ini dilengkapi dengan teknologi terbaru dari Taiwan dan Jerman seperti mesin pemutar pama dan mesin formosa mono film. Produk PT. Matahari Sentosa yang berkualitas, kuantitas, dan yang paling penting harga ekonomis dalam setiap tahun dapat memenuhi permintaan pasar.

Produk PT. Matahari Sentosa meliputi berbagai warna dan berbagai macam koleksi resleting, seperti berbagai macam penggeser dan pembuatan penggeser untuk pribadi. Pilihan ini mungkin termasuk desain penggeser yang diminta oleh pelanggan pribadi dan juga dapat dipersiapkan untuk disempurnakan menjadi berbagai nama merek.

PT. Matahari Sentosa akan memastikan pelanggannya tersenyum, karena PT. Matahari Sentosa menjamin kliennya dengan produk yang berkualitas paling baik dengan harga yang terbaik dan pengiriman tanpa penundaan. Ini akan menjadi suatu kehormatan dan kesenangan perusahaan ini untuk melayani pelanggan dalam masa mendatang.

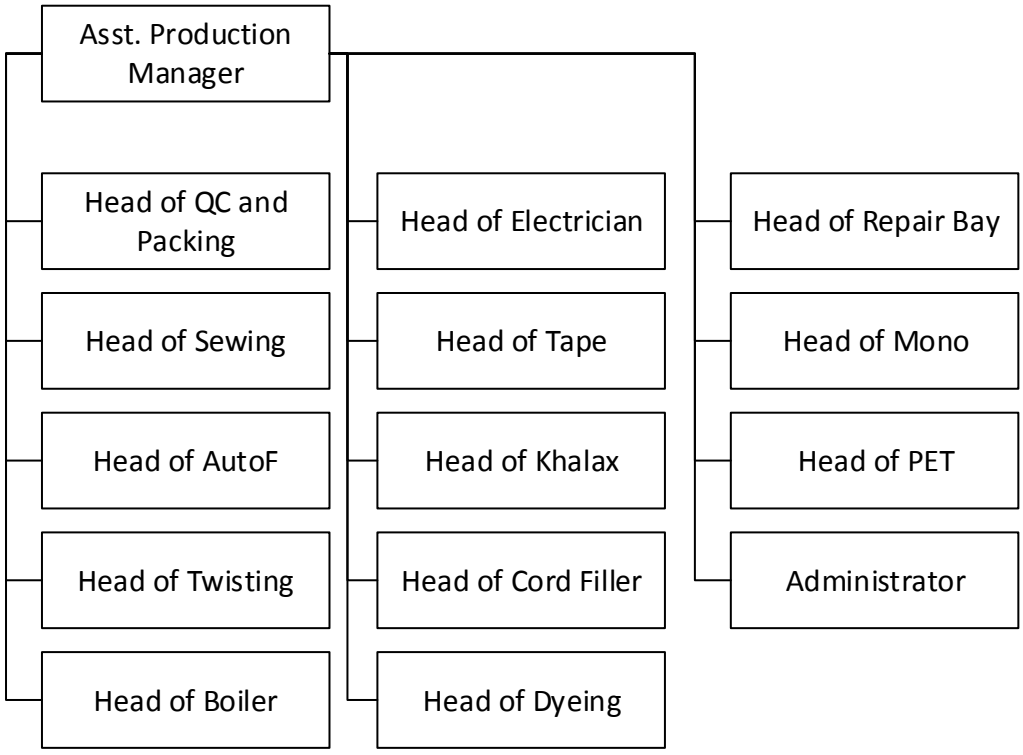
2.1.1 Visi dan Misi Perusahaan

Visi dari PT. Matahari Sentosa adalah menjadi perusahaan resleting yang terbaik.

Adapun misi dari PT. Matahari Sentosa adalah menghasilkan produk yang berkualitas tinggi, mempertahankan harga yang kompetitif, melayani pelanggan dengan ramah, meningkatkan sumber daya manusia, teknologi dan proses produksi secara terus menerus, menjaga akuntabilitas sosial terhadap karyawan dengan menyediakan wilayah kerja yang ramah lingkungan dan sadar terhadap kesehatan.

2.1.2 Struktur Organisasi

Struktur organisasi adalah pola hubungan antara bagian-bagian dari instansi atau menggambarkan dengan jelas pemisah kegiatan pekerjaan antar bagian dalam suatu instansi. Berikut struktur organisasi divisi produksi di PT. Matahari Sentosa dapat dilihat pada Gambar 2.1.



Gambar 2.1 Struktur Organisasi PT. Matahari Sentosa

2.2 Landasan Teori

Pada sub bab ini berisi teori-teori yang digunakan dalam proses analisis dan perancangan serta implementasi dalam tugas akhir ini.

2.2.1 Data Warehouse

Data warehouse merupakan kumpulan data yang berorientasi subjek, terintegrasi, tidak dapat di-*update*, memiliki dimensi waktu, yang digunakan untuk mendukung proses manajemen pengambilan keputusan [2]. Karakteristik *data warehouse* menurut Inmon adalah sebagai berikut :

1. Berorientasi Subjek (*Subject Oriented*)

   Data warehouse berorientasi subjek artinya *data warehouse* didesain untuk menganalisa data berdasarkan subjek-subjek tertentu dalam organisasi, bukan pada proses atau fungsi aplikasi tertentu. *Data warehouse* diorganisasikan disekitar subjek-subjek utama dari perusahaan (*customers, product* dan *sales*), hal ini dikarenakan kebutuhan dari *data warehouse* untuk menyimpan data-data yang bersifat penunjang suatu keputusan.

2. Terintegrasi (*Integrated*)

   Data warehouse dapat menyimpan data-data yang berasal dari sumber-sumber yang terpisah kedalam suatu format yang konsisten dan saling terintegrasi satu dengan yang lainnya.

3. Rentang Waktu (*Time variant*)

   Seluruh data pada *data warehouse* dapat dikatakan akurat atau *valid* pada rentang waktu tertentu. Untuk melihat interval waktu yang digunakan dalam mengukur keakuratan suatu *data warehouse* dapat menggunakan beberapa cara, antara lain :

   a. Cara yang peling sederhana adalah menyajikan *data warehouse* pada rentang waktu tertentu, misal antara 5 sampai 10 tahun ke depan.

   b. Cara yang kedua, dengan menggunakan variasi atau perbedaan waktu yang disajikan dalam *data warehouse* dengan unsur waktu dalam hari, minggu, dan bulan.

4. *Nonvolatile*

   Data warehouse tidak dapat di-*update* secara *real time* tetapi di-*referesh* dari sistem operasional secara reguler. Data yang baru selalu ditambahkan kedalam *database* itu sendiri. *Database* tersebut secara terus-menerus menerima dan menyimpan data baru, kemudian disatukan dengan data sebelumnya.

2.2.2 Data Mart

Data mart merupakan suatu bagian dari *data warehouse* yang mendukung pembuatan laporan dan analisis data pada suatu unit. *Data mart* berisi informasi yang relefan bagi pengguna yang ingin mengambil keputusan.

Ada empat tugas yang dapat dilakukan dengan adanya *data mart* [3], empat tugas tersebut adalah sebagai berikut :

1. Pembuatan laporan

   Pembuatan laporan merupakan salah satu kegunaan *data mart* yang paling umum dilakukan. Dengan menggunakan *query* sederhana didapatkan laporan per hari, per bulan, per tahun, atau jangka waktu kapanpun yang diinginkan.

2. *On-Line Analytical Processing* (OLAP)

   Dengan adanya *data mart*, semua informasi baik detail maupun hasil *summary* yang dibutuhkan dalam proses analisa mudah didapat. OLAP merupakan konsep data multidimensi dan memungkinkan pada pemakai menganalisa data sampai mendetail, tanpa mengetikkan satupun perintah SQL. Fasilitas lain adalah *roll-up* dan *drill-down*. *Drill-down* adalah kemampuan untuk melihat detail dari suatu informasi dan *roll-up* adalah kebalikannya.

3. *Data Mining*

   Data mining merupakan proses untuk menggali (*mining*) pengetahuan dan informasi baru dari data yang berjumlah banyak pada *data mart*.

4. Proses informasi *executive*

   Data mart dapat membuat ringkasan informais yang penting dengan tujuan membuat keputusan bisnis, tanpa harus menjelajahi keseluruhan data. Dengan

menggunakan *data mart* segala laporan telah diringkas dan dapat pula mengetehaui segala rinciannya secara lengkap, sehingga mempermudah proses pengambilan keputusan.

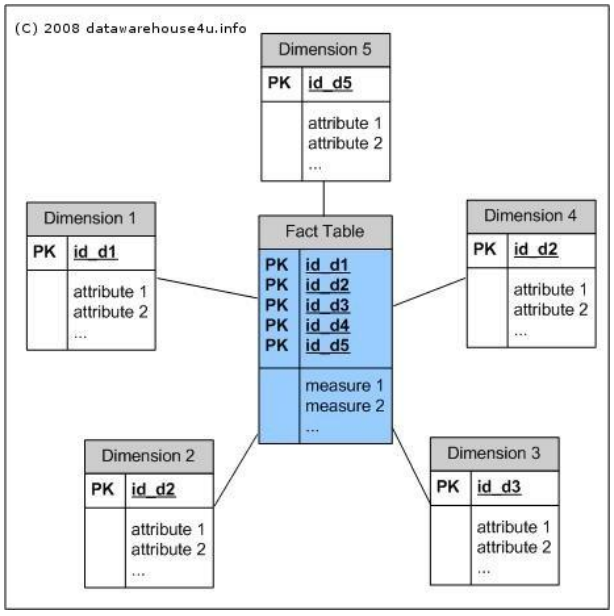
2.2.2.1 Model Dimensional Data Mart

Model dimensional pada *data mart* terdiri dari tabel fakta dan tabel dimensi. Tabel fakta merupakan tabel yang berisi kumpulan *foreign key* dari *primary* key yang terdapat pada masing-masing tabel dimensi, sedangkan tabel dimensi merupakan tabel yang berisi data detail yang menjelaskan *foreign key* yang terdapat pada tabel fakta.

Ada beberapa model skema yang terdapat pada pemodelan *data mart*, yaitu skema *star*, skema *snowflake*, dan skema *constellation*. Penjelasan dari masing-masing model skema adalah sebagai berikut :

1. Skema Bintang (*Star Schema*)

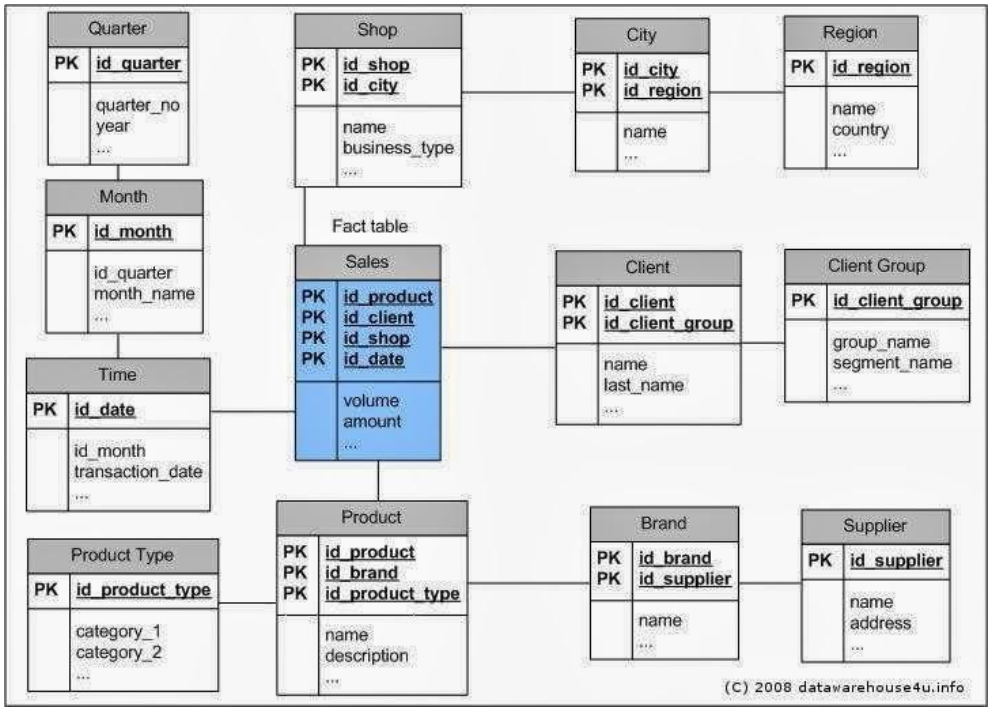
   Skema ini mengikuti bentuk bintang, dimana terdapat satu tabel fakta di pusat bintang dengan beberapa tabel dimensi yang mengelilinginya. Semua tabel dimensi berhubungan dengan ke tabel fakta. Tabel fakta memiliki beberapa *primary key* dalam tabel dimensi. Berikut adalah contoh skema bintang dapat dilihat pada Gambar 2.2.



Gambar 2.2 Skema Bintang

2. Skema Bola Salju (*Snowflake Schema*)

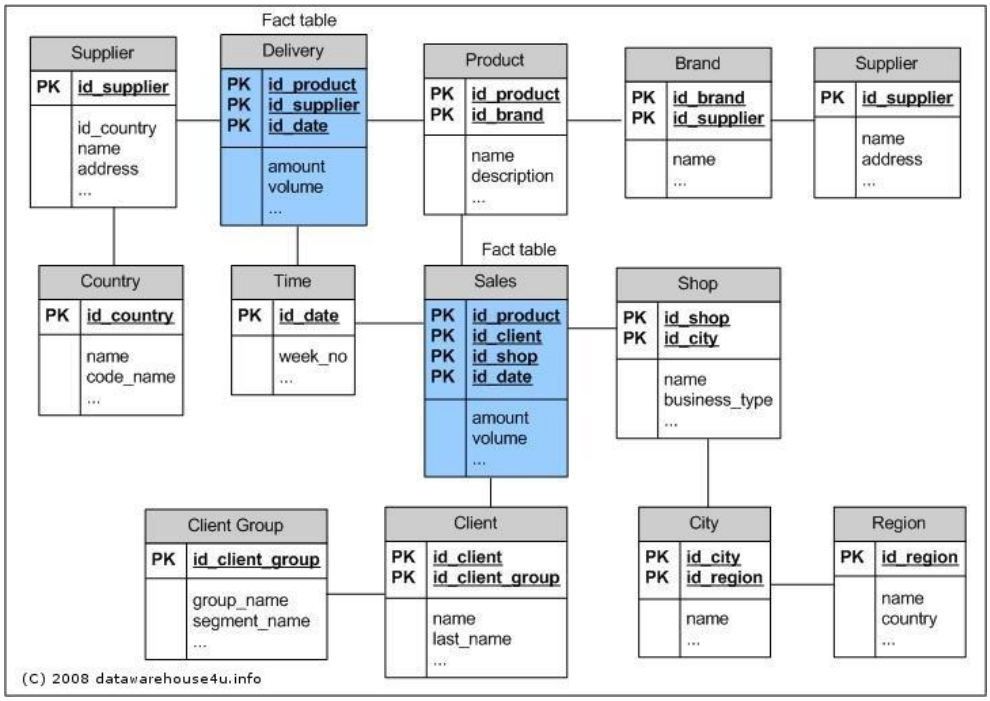
   Skema bola salju merupakan perluasan dari skema bintang dengan tambahan beberapa tabel dimensi yang tidak berhubungan secara langsung dengan tabel fakta. Tabel dimensi tersebut berhubungan dengan tabel dimensi yang lain. Berikut adalah contoh skema bola salju dapat dilihat pada Gambar 2.3.



Gambar 2.3 Skema Bola Salju

3. Skema *Constellation*

   Pada skema ini terdapat beberapa tabel fakta yang menggunakan satu atau beberapa tabel dimensi. Berikut adalah contoh skema constellation dapat dilihat pada Gambar 2.4.



Gambar 2.4 Skema *Constellation*

2.2.2.2 ETL (*Extract, Transform, Loading*)

Proses ETL (*Extract*, *Transform*, *Loading*) merupakan proses yang harus dilalui dalam pembentukan *data mart* [4]. ETL merupakan fase pemrosesan data dari sumber data kemudian masuk ke dalam *data mart*. Tujuan ETL adalah mengumpulkan, menyaring, mengolah dan menggabungkan data dari berbagai sumber untuk disimpan ke dalam *data mart*. Berikut adalah penjelasan dari tiap proses ETL :

1. Ekstraksi Data (*Extract*)

   Ekstraksi data adalah proses pengambilan data dari berbagai sistem operasional, baik menggunakan *query* atau aplikasi ETL. Terdapat beberapa fungsi ekstraksi data, yaitu :

   a. Ekstraksi data secara otomatis dari aplikasi sumber.
   b. Penyaringan atau seleksi data hasil ekstraksi.
   c. Pengiriman data dari berbagai *platform* aplikasi ke sumber data.
   d. Perubahan format *layout* data dari format aslinya.

e. Penyimpanan dalam *file* sementara untuk penggabungan dengan hasil ekstraksi dari sumber lain.

2. Transformasi Data (*Transformation*)

   Transformasi adalah proses dimana data hasil ekstraksi disaring dan diubah sesuai dengan aturan bisnis yang berlaku. Langkah-langkah dalam transformasi data adalah sebagai berikut :

   a. Memetakan data input dari skema data aslinya ke skema *data mart*.
   b. Melakukan konversi tipe data.
   c. Membersihkan dan membuang data yang sama (duplikasi).
   d. Memeriksa referensi data.
   e. Mengisi nilai-nilai yang kosong dengan nilai *default*.
   f. Menggabungkan data.

3. Pengisian Data (*Loading*)

   Pengisian data adalah proses memasukkan data yang didapatkan dari hasil transformasi ke dalam *data mart*. Cara untuk memasukkan data adalah dengan menjalankan *SQL script* secara periodik.

2.2.2.3 OLAP (*On-Line Analytical Processing*)

OLAP (*On-Line Analytical Processing*) adalah teknologi yang memproses data ke dalam struktur multidimensi, menyediakan jawaban yang cepat untuk *query* analisis yang kompleks dengan tujuan untuk mengorganisir sejumlah data yang besar, agar bisa dianalisa dan dievaluasi dengan cepat serta menyediakan kecepatan dan fleksibelitas untuk mendukung analisa dalam waktu nyata [2].

Ada beberapa karakteristik OLAP, yaitu :

1. Mengijinkan pelaku bisnis melihat data dari sudut pandang *logical* dan multidimensional pada *data warehouse*.
2. Memfasilitasi *query* yang kompleks dan analisa bagi pengguna.
3. Mengijinkan pengguna melakukan *drill-down* untuk menampilkan data yang lebih detail atau *roll-up* untuk agregasi dari suatu dimensi atau beberapa dimensi.
4. Menyediakan proses kalkulasi dan perbandingan data.

5. Menampilkan hasil dalam bentuk tabel atau grafik.

Keuntungan dari OLAP, yaitu :

1. Meningkatkan produktivitas pemakai akhir bisnis, pengembang IT, dan keseluruhan organisasi.
2. Pengawasan yang lebih dan akses tepat waktu terhadap informasi strategis dapat membuat pengambilan keputusan lebih cepat.
3. Mengurangi pengembangan aplikasi bagi staf IT dengan membuat pemakaian akhir dapat mengubah skema dan membuat model sendiri.
4. Penyimpanan pengawasan organisasi melalui integritas data koorporasi sebagai aplikasi OLAP tergantung pada *data warehouse* dan sistem OLTP untuk memperbaharui sumber tingkatan data.

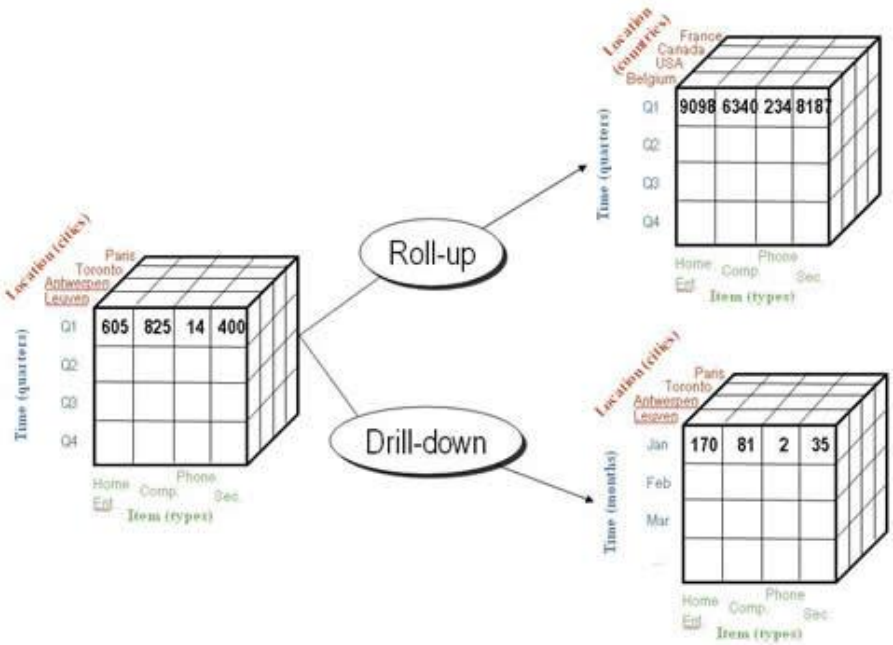
OLAP dapat digunakan untuk melakukan seperti [2] :

1. Konsolidasi (*roll-up*)

   Konsolidasi melibatkan pengelompokan data.

2. *Drill-down*

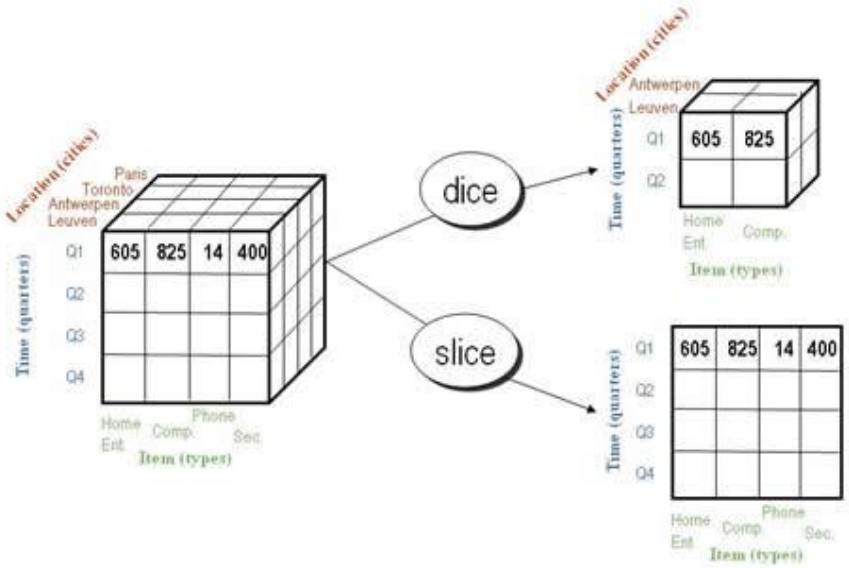
   Suatu bentuk yang merupakan kebalikan dari konsolidasi untuk menjabarkan data yang ringkas menjadi data yang lebih detail. Gambaran untuk *roll-up* dan *drill-down* dapat dilihat pada Gambar 2.5.



Gambar 2.5 *Roll-up* dan *Drill-down*

3. *Slicing* dan *dicing*

   Menjabarkan pada kemampuan untuk melihat data dari sudut pandang. Gambaran untuk *slicing* dan *dicing* dapat dilihat pada Gambar 2.6.



Gambar 2.6 *Slicing* dan *Dicing*

2.2.3 Microsoft SQL Server

Microsoft SQL Server adalah sebuah sistem manajemen basis data relasional (RDBMS) produk *Microsoft*. Bahasa *query* utamanya adalah *Transact-SQL* yang merupakan implementasi dari *SQL* standar ANSI/ISO yang digunakan oleh *Microsoft* dan *Sybase*. Umumnya *SQL Server* digunakan di sunia bisnis yang memiliki basis data yang berskala kecil sampai dengan menengah, tetapi kemudian berkembang dengan digunakannya *SQL Server* pada basis data besar [5]. Pengguna yang menggunakan *SQL Server* dapat menyimpan banyak data dan dapat untuk diimplementasi untuk kepentingan bisnis dan perusahaan.

2.2.4 SSIS (*SQL Server Integration Service*)

SSIS (*SQL Server Integration Services*) adalah suatu platform untuk membangun sistem yang handal untuk integrasi data, *extraction*, *transformation*, dan *loading* yang digunakan pada *data warehousing* [6]. SSIS menawarkan solusi

dalam menghadapi permasalahan data integrasi. Selain itu, *tools* ini membatu untuk menigkatkan efisiensi waktu pembuatan.

Arsitektur *SQL Server Integration Services* secara umum berisi berbagai macam komponen, diantaranya yaitu :

1. *SSIS Deginer*. Merupakan *tools* yang digunakan untuk membuat dan mengatur paket *integration service*. Pada *SQL Server 2012*, *tools* ini sudah terintegrasi dengan *Visual Studio 2010* yang merupakan bagian *project Bussiness Intelegence*.
2. *Runtime Engine*. Komponen ini berguna untuk menjalankan semua paket SSIS yang sudah dibuat.
3. *Task* dan *executable binary*.
4. *Data Flow Engine* dan *Data Flow*. Komponen *data flow* merupakan enkapsulasi *data flow engine* yang menyediakan *buffer* didalam memori dan bertugas memindahkan data dari sumber data ke tujuan data. Sedangkan *data flow* merupakan sumber data, tujuan data, dan transformasi.
5. *Integration Services service*. Memungkinkan *SQL Server Management Studio* dapat digunakan untuk melakukan monitoring paket SSIS dan mengatur SSIS *storage* yang digunakan.
6. *SQL Server Import* dan *Export Wizard*. *Tools* ini digunakan untuk melakukan *copy* data dari sumber ke tujuan data.

2.2.5 UML (*Unified Modeling Language*)

UML (*Unified Modelling Language*) adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi objek. Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembang untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah untuk dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan mereka yang lain [7].

UML merupakan kesatuan dari bahasa pemodelan yang dikembangkan oleh Booch, *Object Modelling Technique* (OMT) dan *Object Oriented Software Engineering* (OOSE) [7]. Metode Booch dari Grady Booch sangat terkenal

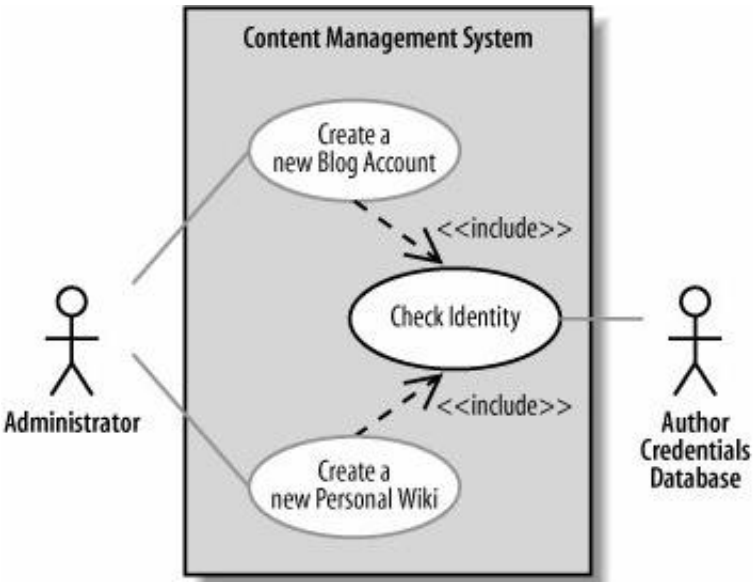
dengan nama metode *Design Object Oriented*. Metode ini menjadikan proses analisis dan design ke dalam empat tahapan iteratif, yaitu identifikasi kelas-kelas dan objek-objek, identifikasi semantik dari hubungan objek dan kelas tersebut, perincian interface dan implementasi. Keunggulan metode Booch adalah pada detail kayanya dengan notasi dan elemen. Pemodelan OMT yang dikembangkan oleh Rumbaugh didasarkan pada analisis terstruktur dan pemodelan *entity-relationship*. Tahapan utama dalam metodologi ini adalah analisis, design sistem, design objek dan implementasi. Keunggulan metode ini adalah dalam penotasian yang mendukung semua konsep OO. Metode OOSE dari Jacobson lebih memberi penekanan pada *use case*. OOSE memiliki tiga tahapan yaitu membuat model *requirement* dan analisis, design dan implementasi, dan model pengujian (*test model*). Keunggulan metode ini adalah mudah dipelajari karena memiliki notasi yang sederhana namun mencakup seluruh tahapan dalam rekayasa perangkat lunak.

Dengan UML, metode Booch, OMT dan OOSE digabungkan dengan membuang elemen-elemen dari metode lain yang lebih efektif dan elemen-elemen baru yang belum ada pada metode terdahulu sehingga UML lebih ekspresif dan seragam daripada metode lainnya.

2.2.5.1 Use Case Diagram

Use Case adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem disebut skenario. Setiap skenario mendeskripsikan urutan kejadian. Setiap urutan diinisialisasi oleh orang, sistem lain, perangkat keras atau urutan waktu. Dengan demikian secara singkat biasa dikatakan *use case* adalah serangkaian skenario yang digabungkan bersama-sama oleh tujuan umum pengguna [7].

Berikut ini adalah contoh *Use Case Diagram*:



Gambar 2.7 *Use Case Diagram* [8]

2.2.5.2 Use Case Scenario

Sebuah diagram yang menunjukkan kasus penggunaan dan aktor dapat menjadi titik awal yang bagus, tetapi tidak memberikan cukup detail untuk desainer sistem untuk benar-benar memahami persis bagaimana kekhawatiran sistem akan dipenuhi. Use Case Skenario adalah cara terbaik untuk mengungkapkan informasi penting ini dalam bentuk penggunaan description setiap kasus berbasis teks harus disertai dengan satu [8].

Berikut ini adalah contoh *Use Case Scenario*:

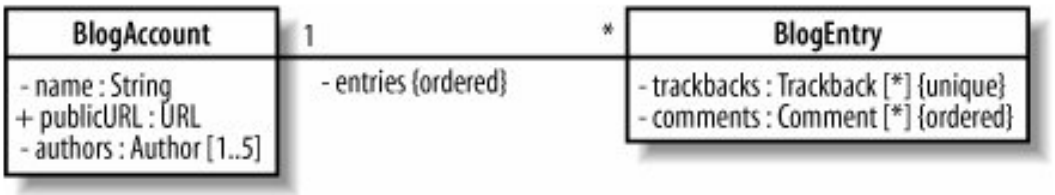
Use case name	**Create a new Blog Account**	
Related Requirements	Requirement A.1.	
Goal In Context	A new or existing author requests a new blog account from the `Administrator`.	
Preconditions	The system is limited to recognized authors and so the author needs to have appropriate proof of identity.	
Successful End Condition	A new blog account is created for the author.	
Failed End Condition	The application for a new blog account is rejected.	
Primary Actors	Administrator.	
	Secondary Actors	Author Credentials Database.
	Trigger	The `Administrator` asks the CMS to create `a new blog account`.
Main Flow	Step	Action
	1	The `Administrator` asks the system to create a new blog account.
	2	The `Administrator` selects an account type.
	3	The `Administrator` enters the author's details.
	4	The author's details are verified using the Author Credentials Database.
	5	The new blog account is created.
	6	A summary of the new blog account's details are emailed to the author.
Extensions	Step	Branching Action
	4.1	The Author Credentials `Database` does not verify the author's details.
	4.2	The author's new blog account application is rejected.

Gambar 2.8 *Use Case Scenario* [8]

2.2.5.3 Class Diagram

Class Diagram adalah diagram yang digunakan untuk menampilkan beberapa kelas serta paket-paket yang ada dalam sistem/perangkat lunak yang sedang kita kembangkan. Diagram kelas memberikan gambaran/diagram statis tentang sistem/perangkat lunak dan relasi-relasi yang ada di dalamnya [9].

Berikut ini adalah contoh *Class Diagram*:



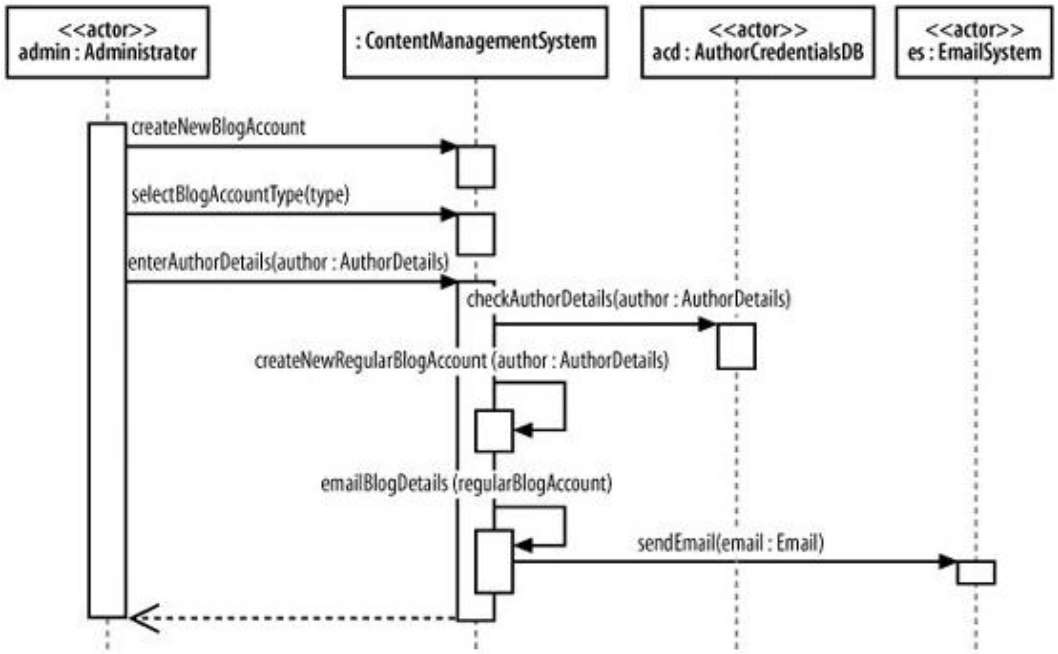
Gambar 2.9 *Class Diagram* [8]

2.2.5.4 Sequence Diagram

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

Komponen utama *sequence diagram* terdiri atas objek yang dituliskan dengan kotak segiempat bernama. *Message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan *progress* vertikal [7].

Berikut ini adalah contoh *Sequence Diagram*:

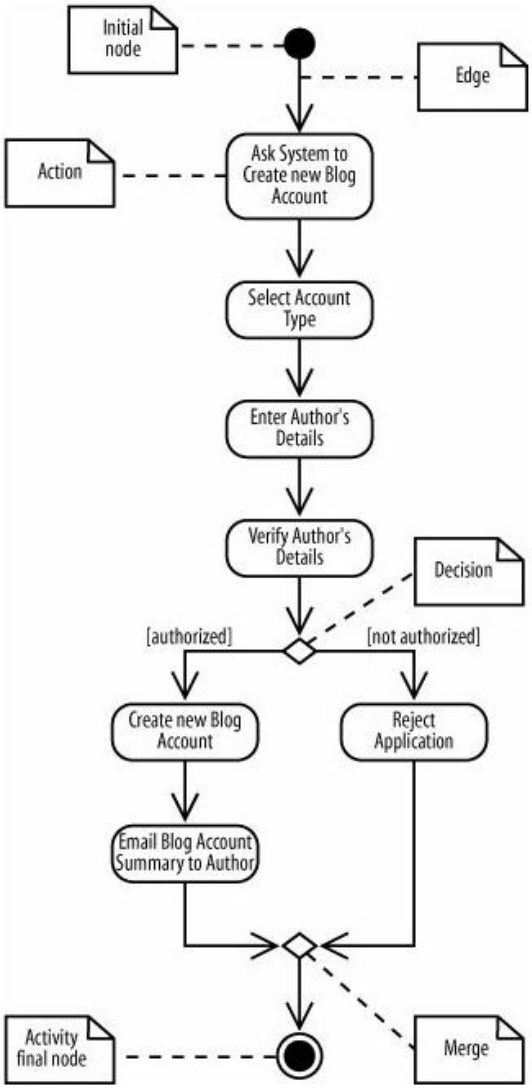


Gambar 2.10 *Sequence Diagram* [8]

2.2.5.5 Activity Diagram

Activity Diagram adalah teknik untuk mendeskripsikan 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 mendukung perilaku paralel sedangkan *flowchart* tidak bisa [7].

Berikut ini adalah contoh *Activity Diagram*:



Gambar 2.11 ***Activity Diagram*** [8]