BAB 2
DASAR TEORI DAN TINJAUAN PUSTAKA

Dalam bab ini akan diuraikan mengenai teori-teori dan tinjauan dari hasil penelitian-penelitan sebelumnya yang berhubungan dan dapat digunakan dalam pengembangan sistem informasi manajemen tambang di PT Sebuku Iron Lateritic Ores.

2.1 Dasar Teori

2.1.1 Pengertian Sistem

Menurut Lucas (2000) mendefinisikan sistem sebagai suatu komponen atau variable yang terorganisir, saling berinteraksi, saling bergantung, satu sama lain dan terpadu. Begitu pula Robert (1993), mendefinisikan sistem sebagai seperangkat elemen-elemen yang terintegrasi dengan maksud yang sama untuk mencapai suatu tujuan bersama.

2.1.2 Konsep Dasar Informasi

Menurut Gordon (1992) mendefinisikan informasi sebagai data yang telah diolah menjadi bentuk yang lebih berarti dan berguna bagi penerimanya untuk mengambil keputusan masa kini maupun yang akan datang. Informasi mempunyai ciri benar atau salah, baru, tambahan dan korektif. Menurut Raymond (2001) definisi informasi adalah data yang telah diolah menjadi bentuk yang lebih berarti dapat meliputi elemen komputer, elemen non komputer atau kombinasinya.

Menurut Burch dan Grudnitski (1986), agar informasi dihasilkan lebih berharga, maka informasi harus memenuhi kriteria sebagai berikut :

a. Informasi harus akurat, sehingga mendukung pihak manajemen dalam mengambil keputusan.
b. Informasi harus relevan, benar-benar terasa manfaatnya bagi yang membutuhkan.
c. Informasi harus tepat waktu, sehingga tidak ada keterlambatan pada saat dibutuhkan.

Kegunaan informasi adalah untuk mengurangi ketidakpastian didalam proses pengambilan keputusan tentang suatu keadaan. Informasi yang digunakan di dalam suatu informasi umumnya digunakan untuk beberapa kegunaan. Informasi digunakan tidak hanya oleh satu orang pihak di dalam organisasi.

2.1.3 Pengertian Sistem Informasi

Menurut Ladjamudin (2005) sistem informasi dapat didefinisikan sebagai berikut.

a. Suatu sistem yang dibuat oleh manusia yang terdiri dari komponen-komponen dalam organisasi untuk mencapai tujuan yait menyajikan informasi.
b. Sekumpulan prosedur organisasi yang pada saat dilaksanakan akan memberikan informasi bagi pengambilan keputusan dan atau untuk mengendalikan organisasi.
c. Suatu sistem didalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak

luar tertentu dengan laporan-laporan yang diperlukan.

2.1.4 Metode *Waterfall Model*

Menurut Sommerville (2003) *Waterfall Model* merupakan salah satu model proses perangkat lunak yang mengambil kegiatan proses dasar seperti spesifikasi, pengembangan, validasi dan evolusi dengan mempresentasikannya sebagai fase-fase proses yang berbeda seperti analisis dan definisi persyaratan, perancangan perangkat lunak, implementasi dan pengujian unit, integrasi dan pengujian sistem, operasi dan pemeliharaan.

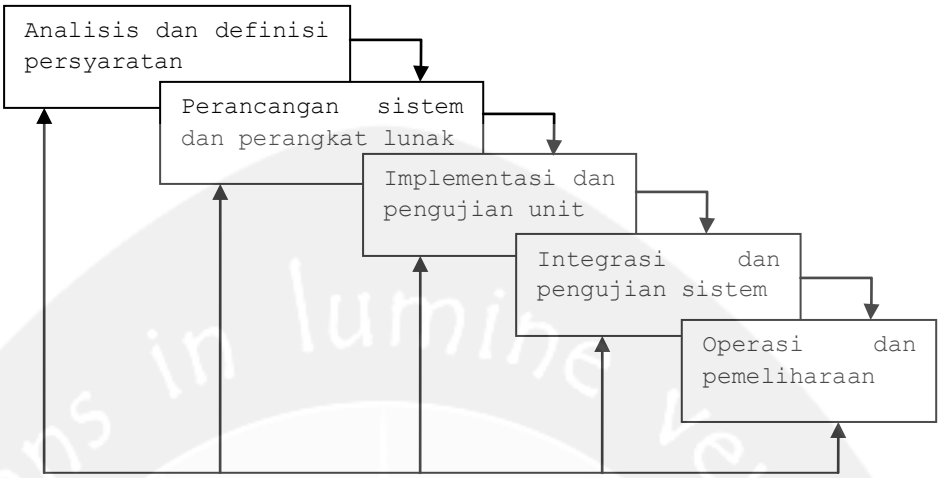
Pada Gambar 2.1, Sommerville (2003) menjelaskan bahwa tahap-tahap utama dari *waterfall model* adalah memetakan kegiatan-kegiatan pengembangan dasar yaitu:

a. Analisis dan Definisi Persyaratan

   Proses mengumpulkan informasi kebutuhan sistem/perangkat lunak melalui konsultasi dengan *user system.* Proses ini mendefinisikan secara rinci mengenai fungsi-fungsi, batasan dan tujuan dari perangkat lunak sebagai spesifikasi sistem yang akan dibuat.

b. Perancangan Sistem dan Perangkat Lunak

   Proses perancangan sistem difokuskan pada empat atribut, yaitu struktur data, arsitektur perangkat lunak, representasi antar muka, dan detail (algoritma) prosedural. Yang dimaksud struktur data adalah representasi dari hubungan logis antara elemen-elemen data individual.



Gambar 2.1. *Warefall Model* (Sommerville, 2003)

c. Implementasi dan Pengujian Unit

Pada tahap ini, perancangan perangkat lunak direalisasikan sebagai program atau unit program. Kemudian pengujian unit melibatkan verifikasi bahwa setiap unit program telah memenuhi spesifikasinya.

d. Integrasi dan Pengujian Sistem

Unit program/program individual diintegrasikan menjadi sebuah kesatuan sistem dan kemudian dilakukan pengujian. Dengan kata lain, pengujian ini ditujukan untuk menguji keterhubungan dari tiap-tiap fungsi perangkat lunak untuk menjamin bahwa persyaratan sistem telah terpenuhi. Setelah pengujian sistem selesai dilakukan, perangkat lunak dikirim ke pelanggan/*user*.

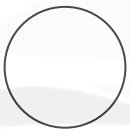
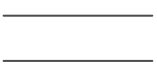
e. Operasi dan Pemeliharaan

Tahap ini biasanya memerlukan waktu yang paling lama. Sistem diterapkan (di-*install*) dan dipakai. Pemeliharaan mencakup koreksi dari beberapa kesalahan yang tidak diketemukan pada tahapan

sebelumnya, perbaikan atas implementasi unit sistem dan pengembangan pelayanan sistem, sementara persyaratan-persyaratan baru ditambahkan.

2.1.5 *Data Flow Diagram*

Menurut Hartono (1999) *Data Flow Diagram* (DFD) adalah diagram yang menggunakan notasi atau simbol untuk menggambarkan arus data sistem. DFD digunakan untuk membuat dokumentasi dari sistem informasi yang ada dan untuk menyusun sistem informasi yang baru. Pada gambar 2.2 akan dijelaskan Simbol DFD versi De Marco.

Nama Simbol	Simbol DFD versi De Marco
Arus Data	
Proses	
Penyimpanan Data	

Gambar 2.2. Simbol DFD versi De Marco (Hartono, 1999)

2.1.6 Normalisasi Data

Menurut Octaviani (2010) normalisasi data adalah proses dimana tabel-tabel pada database dites dalam hal kesalingtergantungan di antara *field-field* pada sebuah tabel. Pada proses normalisasi data, aturan yang dijadikan acuan adalah metode ketergantungan fungsional. Teorinya adalah bahwa tiap kolom dalam

sebuah tabel selalu memiliki hubungan yang unik dengan sebuah kolom kunci.

Ada beberapa langkah dalam normalisasi tabel, yaitu:

a. *Decomposition*, dekomposisi adalah proses mengubah bentuk tabel supaya memenuhi syarat tertentu sebagai tabel yang baik.
b. Bentuk tidak normal, pada bentuk ini semua data yang ada pada tiap *entity* (diambil atributnya) masih ditampung dalam satu tabel besar.
c. *Normal Form* pertama (*1st Normal Form*), pada tahapan ini tabel di-dekomposisi dari tabel bentuk tidak normal yang kemudian dipisahkan menjadi tabel-tabel kecil yang memiliki kriteria tidak memiliki atribut yang bernilai ganda dan komposit.
d. Normal Tahap Kedua (*2nd Normal Form*), pada tahapan ini tabel dianggap memenuhi normal kedua jika pada tabel tersebut semua atribut yang bukan kunci primer bergantung penuh terhadap kunci primer tabel tersebut.
e. Bentuk Normal Ketiga (*3rd Normal Form*),setiap atribut pada tabel selain kunci primer atau kunci utama harus bergantung penuh pada kunci utama.

2.1.7 *Entity Relationship Diagram*

Menurut Octaviani (2010) ERD (*Entity Relationship Diagram*) adalah sebuah diagram yang secara konseptual memetakan hubungan antar penyimpanan pada diagram DFD di atas. ERD ini digunakan untuk melakukan permodelan terhadap struktur data dan hubungannya. Penggunaan ERD ini dilakukan untuk mengurangi tingkat kerumitan

penyusunan sebuah database yang baik. *Entity* dapat berarti sebuah obyek yang dapat dibedakan dengan obyek lainnya. Ada dua macam atribut yang dikenakan dalam *entity* yaitu atribut yang berperan sebagai kunci primer dan atribut deskriptif. Hal ini seperti setiap entity memiliki himpunan yang diperlukan sebuah *primary key* untuk membedakan anggota-anggota dalam himpunan tersebut.Atribut dapat memiliki sifat-sifat sebagai berikut :

a. *Atomic*, *atomic* adalah sifat dari atribut yang menggambarkan bahwa atribut tersebut berisi nilai yang spesifik dan tidak dapat dipecah lagi.
b. *Multivalued*, sifat ini menandakan atribut ini bisa memiliki lebih dari satu nilai untuk tiap entity tertentu.
c. *Composite*, atribut yang bersifat komposit adalah atribut yang nilainya adalah gabungan dari beberapa atribut yang bersifat atomik.

Menurut Octaviani (2010) ada beberapa derajat relasi yang dapat terjadi yaitu:

a. Hubungan Satu – ke – Satu *(one to one)*
   Menggambarkan bahwa antara 1 anggota *entity* A hanya dapat berhubungan dengan 1 anggota *entity* B. Biasanya derajat relasi ini digambarkan dengan simbol 1-1.
b. Hubungan satu – ke – banyak *(one to many)*
   Menggambarkan bahwa 1 anggota *entity* A dapat memiliki hubungan dengan lebih dati 1 anggota *entity* B. Biasanya derajat relasi ini digambarkan dengan simbol 1-N.

c. Hubungan Banyak – ke – banyak *(many to many)*

Menggambarkan bahwa lebih dari satu anggota A dapat memiliki hubungan dengan lebih dari satu anggota *entity* B. Simbol yang digunakan adalah N-N.

2.1.8 Basis Data

Octaviani (2010) menyebutkan basis data adalah sekumpulan data yang memiliki hubungan secara logika dan diatur dengan susunan tertentu serta disimpan dalam media penyimpanan komputer. Data itu sendiri adalah representasi dari semua fakta yang ada pada dunia nyata.

Menurut Octaviani (2010) data mempunyai jenjang mulai karakter-karakter *(character)*, item data (data item atau *field)*, *record*, *file* dan *database*. Berikut penjelasan tentang jenjang data yaitu sebagai berikut :

a. Karakter

Karakter adalah bagian data yang terkecil, dapat berupa karakter numerik, huruf ataupun karakter – karakter khusus *(Special Character)* yang membentuk suatu item data.

b. Field

Suatu *field* menggambarkan suatu atribut dari *record* yang menunjukkan suatu item dari data seperti nama, alamat, dan lain sebagainya. Kumpulan dari *field* membentuk suatu *record*. Ada tiga hal yang penting dalam suatu *field*, yaitu:

- Nama dari *field (field name)*
  Field harus diberi nama untuk membedakan *field* yang satu dengan *field* yang lain.
- Representasi dari *field (field representation)*

Representasi dari *field* menunujukkan tipe dari *field* (*field type*) serta lebar dari *field (field width).* Lebar dari *field* menunjukkan ruang maksimum dari *field* yang dapat diisi dengan karakter – karakter data.

- Nilai dari *field (field value)*

  Nilai dari *field* menunjukkan isi dari *field* untuk masing – masing *record*.

a. *Record*

   Merupakan kumpulan dari *field* membentuk suatu *record*. *Record* menggambarkan suatu unit data individu tertentu. Kumpulan dari *record* membentuk suatu *field*. Misalnya *field* karyawan, tiap – tiap *record* dapat mewakili data tiap – tiap karyawan yang telah teregistrasi.

b. *File*

   File terdiri dari *record* – *record* yang menggambarkan satu kesatuan data yang sejenis. Misalnya *file* departemen yang berisi data tentang semua departemen yang ada dalam perusahaan.

c. *Database*

   Database adalah kumpulan dari *file* yang mempunyai hubungan keterkaitan antara suatu *file* dengan *file* yang lainnya, sehingga membentuk satu kesatuan data untuk menghasilkan sebuah informasi yang dibutuhkan.

2.1.9 Sistem Database

Menurut Octaviani (2010) sebuah sistem *database* adalah sebuah kumpulan dari komponen-komponen *database-database* yang meliputi :

a. *Database*

b. *Database Server*

c. Komponent *Client Software*

d. Aplikasi *Database*

Octaviani (2010) menyebutkan aplikasi *database* adalah sebuah *software* khusus yang didesain dan digunakan oleh *user* atau pihak lainnya seperti penyedia jasa pemrograman dan konsultan. Sedangkan *client software* adalah salah satu komponen yang termasuk dalam sistem *database* yang memampukan *software* aplikasi *database* mengakses data secara *remote* pada sebuah *server database*. Fungsi utama dari sebuah *server database* adalah menangani manajemen data. Tiap *client software* berkomunikasi dengan server untuk menyalurkan permintaan data lewat sql dan server memprosesnya dalam urutan tertentu dan mengirimkan data tersebut kembali ke *client software*. Kewajiban utama dari sistem *database* adalah menyediakan antarmuka (*interface*) kepada *user* untuk membuat *database*, dan mengolahnya (mencari, menghapus dan mengedit).

2.1.10 Arsitektur Sistem

Menurut Hidayatullah (2012) arsitektur sistem adalah konfigurasi sistem secara keseluruhan yang mengelola sistem basis data, basis data dan aplikasi yang memanfaatkannya.

Beberapa jenis arsitektur sistem yang digunakan adalah :

a. Sistem Mandiri/Tunggal (*stand-alone*).

Pada arsitektur sistem ini, sistem basis data, basis data serta aplikasi basis data ditempatkan

pada komputer yang sama. Arsitektur ini paling sederhana dan murah. Arsitektur sering digunakan pada sistem basis data yang dikelola tidak terlalu besar. Selain itu sistem ini juga digunakan untuk pengembangan modul-modul pada pembuatan *software*.

b. Sistem tersentralisasi (*Centralized System*)

Jika yang disentralisasi adalah sistem basis datanya maka *server*-nya disebut *application server*. Jika yang disentralisasi adalah basis datanya, maka *server*-nya disebut *file server*. Sistem tersentralisasi menggunakan *application server* kurang baik untuk spesifikasi *server* yang rendah. Sistem tersentralisasi menggunakan *file server* kurang baik untuk jaringan yang terlalu luas karena transaksi datanya bias sangat berat dan keamanan kurang terjaga.

c. Sistem *Client-Server*

Arsitektur ini dibuat untuk menutupi kelemahan pada sistem tersentralisasi. Beban *server* jadi tidak terlalu berat karena *Client* juga memiliki sistem basis data sehingga proses yang bisa dilakukan di *Client* akan dilakukan di *Client*.

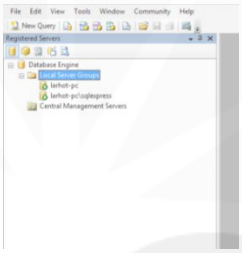
2.1.11 Sistem Database Relasional

Menurut Octaviani (2010) sistem database relational berdasarkan pada metode matematika. Sistem ini menggantikan metode lama yang berdasarkan *network* dan *hierarki*. Metode relational ini bekerja berdasarkan keterkaitan antar tabel. SQL Server Management Studio memiliki beberapa komponen penting yang mewakili kegunaannya dalam perancangan database dan melakukan

pengaturan sistem secara keseluruhan. Komponen-komponen tersebut adalah :

a. *Registered Server*

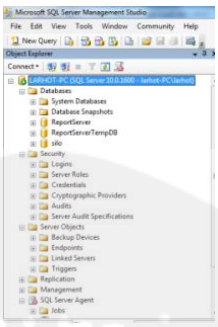
Pada Gambar 2.3 menunjukkan panel *Registered Server* dimungkinkan untuk menjaga koneksi-koneksi dengan *server-server* yang pernah digunakan. Koneksi-koneksi ini dapat digunakan untuk memeriksa status dari *server* tersebut (*online* atau *offline*) atau melakukan pengaturan pada obyek-obyeknya. Setiap *user* memiliki daftar tersendiri dari *Register Servers* ini yang disimpan pada mesin lokal.



Gambar 2.3. *Registered Server* (Octaviani, 2010)

b. *Object Explorer*

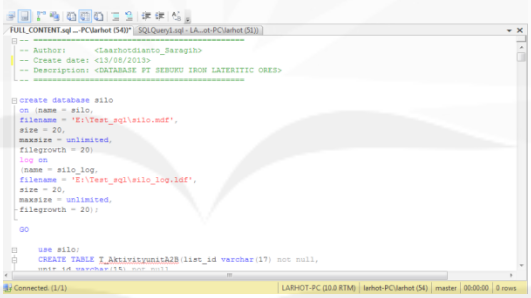
Pada Gambar 2.4 menunjukkan *Object explorer* berfungsi untuk melakukan koneksi maupun diskoneksi. *Object Explorer* juga memungkinkan untuk melakukan koneksi ke lebih dari satu *server* pada panel yang sama. *Server* tersebut dapat berasal dari berbagai tipe *server* seperti *Database Engine*, *Analysis Server*, *Reporting Server* atau *Server Integrasi*.



Gambar 2.4. *Object Explorer* (Octaviani, 2010)

c. Query Editor

Panel query editor pada Gambar 2.5 digunakan untuk membuat, melakukan perubahan perintah-perintah T-SQL dan mengeksekusi perintah tersebut.



Gambar 2.5. *Query Editor* (Octaviani, 2010)

2.1.12 Query Language

Menurut Octaviani (2010) bahasa SQl berawal pada sebuah proyek dengan kode system R yang diprakarsai oleh IBM pada tahun 1980. Proyek ini membuktikan bahwa mungkin saja membuat sebuah sistem relasional database berdasarkan pendekatan yang diperkenalkan oleh E.F.Codd. Dengan bahasa SQL ini *user* dapat melakukan permintaan terhadap lebih dari satu/sekumpulan *record*

pada database. Bukan itu saja, *user* dimungkinkan untuk melakukan permintaan data pada sekumpulan *record-record* baik yang ada pada satu tabel atau bahkan tabel yang lainnya dan itu dapat dilakukan hanya dengan menggunakan sebuah *statement* SQL.SQL dibedakan menjadi dua jenis sub bahasa yaitu :

a. *Data Definition Language* (DLL), bahasa ini digunakan untuk "membangun" struktur database. Contoh dari bahasa ini adalah *CREATE*, *DROP*, *ALTER*. Bahasa ini dikenakan pada database, tabel, kolom (*field*) dan *index*.
b. *Data Manipulation Language* (DML), jenis SQL ini berfungsi untuk melakukan manipulasi terhadap data yang ada seperti record field. Contoh perintahnya adalah *DELETE*, *INSERT*, *UPDATE* dan *SELECT*.

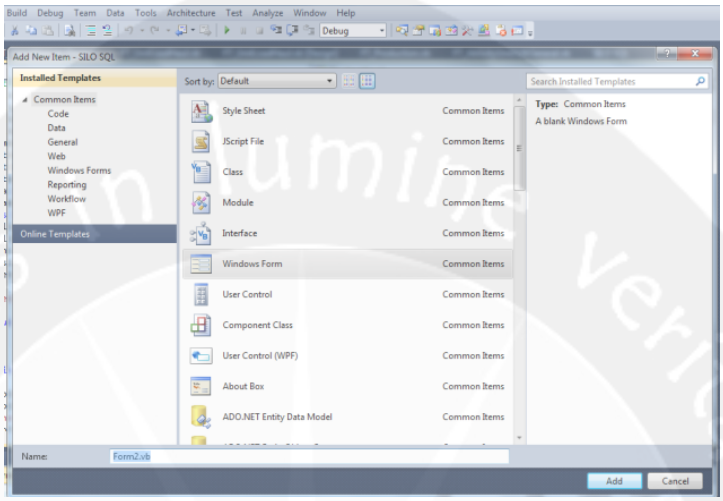
2.1.13 Aplikasi Pemrograman Visual

Menurut Junindar (2010) Microsoft Visual Studio 2010 memiliki beberapa komponen penting yang mewakili kegunaannya dalam perancangan aplikasi dan melakukan pengaturan sistem secara keseluruhan. Komponen-komponen tersebut adalah :

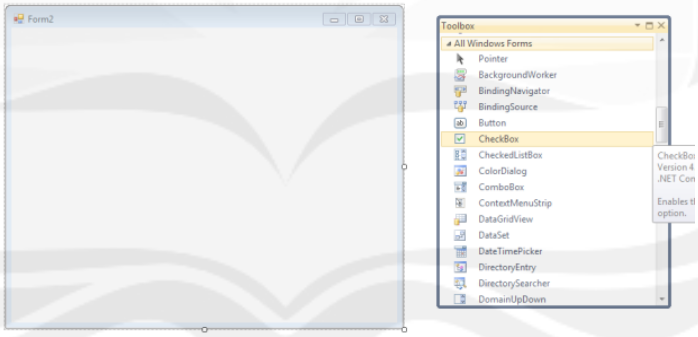
a. *Templates*

Pada Gambar 2.6 menunjukkan *Templates* yang umum digunakan pada pembuatan aplikasi adalah *windows form*. Pada Gambar 2.7 menunjukkan Form merupakan tempat dimana kontrol-kontrol dari *toolbox* yang ditunjukkan pada diletakkan. Form juga berfungsi sebagai tempat pembuatan tampilan atau antarmuka (user interface) dari sebuah aplikasi Windows. Selain *windows form* masih ada *template* lain yang

dapat digunakan yaitu *class*, *module*, *splash screen*, *reporting* dan lain sebagainya.



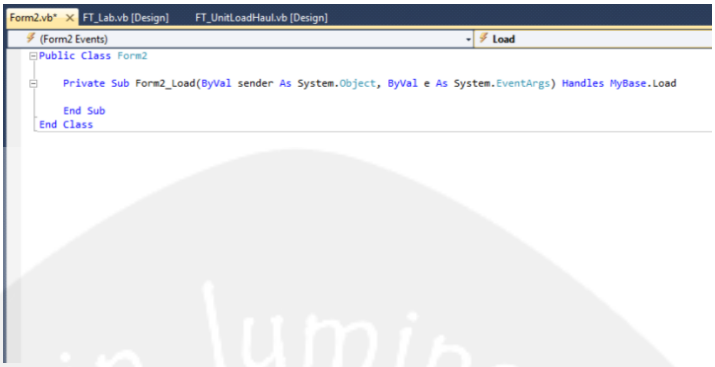
Gambar 2.6. *Templates* (Junindar, 2010)



Gambar 2.7. *Windows Form* & *ToolBox* (Junindar, 2010)

b. *Code Editor*

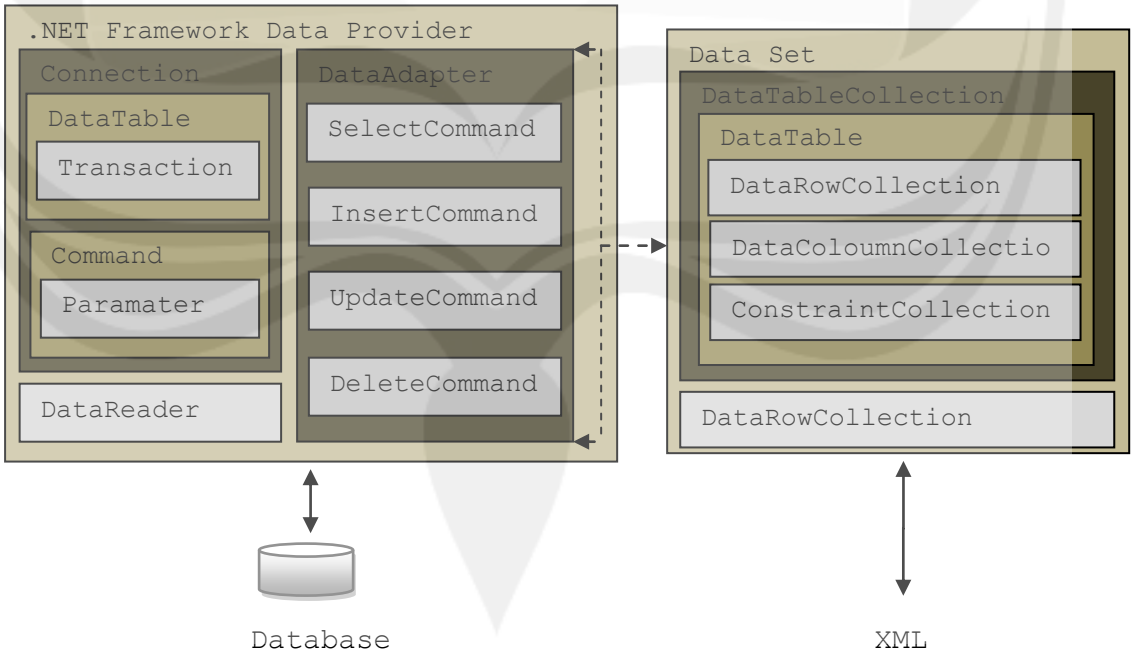
Pada Gambar 2.8 menunjukkan *Code Editor* adalah tempat dimana kita meletakkan atau menuliskan algoritma program dari program kita. Pada *code editor* terdapat bagian *objek* dan *event* dari kontrol.



Gambar 2.8. *Code Editor* (Junindar, 2010)

2.1.14 Algoritma Pemrograman

Menurut Junindar (2010) algoritma yang digunakan pada penelitian ini adalah ADO.NET. Algoritma ADO.NET (*ActiveX Data Object* untuk *.Net Framework*) adalah kumpulan *class* yang berisi komponen untuk melakukan koneksi, akses dan manipulasi data.



Gambar 2.9. Arsitektur ADO.NET (Junindar, 2010)

Pada gambar 2.9 menunjukkan arsitektur dari ADO.NET. ADO.NET merupakan pengembangan dari pendahulunya yaitu ADO 2.6 yang terdapat pada VB.6. Berikut penjelasan tentang VB.Net Arsitektur :

a. Object Connection

Object Connection digunakan untuk membuat koneksi ke database. Misalnya dengan menggunakan objek *SqlConnection* ketika kita menggunakan SQL Server dan *OleDBConnection* untuk sumber data lainnya. Dalam sebuah *Object Connection* dibutuhkan suatu *string* (*ConnectionString*) yang berisikan beberapa atribut yang dibutuhkan sesuai dengan database yang digunakan yaitu :

- *Provider* : *Driver* yang digunakan pada Database
- *DataSource* : Lokasi dimana database disimpan.
- *Database Password* : Password untuk database (*optional security*)

b. Object Command

Object command dapat memanggil data dari sumber data. Perintah dibentuk oleh properti *command text*. Properti *CommandText* berisi nama tabel, pernyataan SQL atau nama dari *store procedure* dari SQL Server.

Untuk mengatur bagaimana ADO.NET menerjemahkan perintah yang dibuat menggunakan properti *CommandText* maka ditugaskan suatu konstanta ke properti *CommandType*. Untuk menghindari *SQL Injection* maka diperlukan paramter pada *Object Command*.

Properti *CommandType* memiliki tiga konstanta sebagai berikut :

- *Text*, perintah berupa pernyataan SQL ini adalah nilai *default* dari *CommandType*.
- *StoreProcedure*, perintah berupa *store procedure*.
- *TableDirect*, perintah berupa tabel.

c. Object DataReader

Object DataReader merupakan mekanisme secara tepat untuk memanggil aliran data yang bersifat *forward-only* dan *read-only*. Kita dapat membuat *object DataReader* menggunakan metode *ExecuteReader*() dari *ObjectCommand*.

d. *Object DataAdapter*

Object DataAdapter akan melakukan kordinasi antara representasi data dalam memori dari sumber data permanen. *Object DataAdapter* bekerja sebagai duta kita dengan mekanisme akses data sehingga kita dapat memanggil dan menyimpan data dari sumber data dan *Object DataSet*.

DataAdapter berisi sebuah metode yang penting dalam ADO.NET yaitu metode *Fill*(). Metode *Fill*() akan melakukan *populate* atau mengisi suatu *DataSet* dan proses pengisian tersebut terjadi ketika *DataSet* menyentuh secara langsung koneksi database. Secara fungsional mekanisme metode *fill*() untuk *populating*/pengisian *DataSet* bekerja seperti membuat suatu kursor *client side static* dalam ADO klasik.

e. Object DataSet

DataSet merupakan representasi data dalam memori yang memberikan model pemrograman relasional yang konsisten, tanpa memperhatikan asal dari sumber data. Suatu DataSet mewakili serangkaian dara secara

lengkap termasuk tabel yang berhubungan constrains (aturan) dan relasi antar tabel.

Ada beberapa cara untuk bekerja dengan *DataSet*, yaitu :

- Menggunakan kode program dengan membuat *DataTable*, *DataRelation* dan *Constrains* dalam *DataSet* dan *populate* dengan data.
- *Populate DataSet* dari sitem manajemen database relasional menggunakan *DataAdapter*.
- Memanggil *DataSet* menggunakan XML.

f. DataTable

DataTable adalah komponen dimana baris data yang diperoleh dari sumber data ditampung sebelum diproses lebih lanjut. Baris data ini kemudian direferensi melalui *object DataTable*. Perubahan yang terjadi pada *DataTable* ditangani *library.Net Framework*, jadi sumber data sudah tidak terlibat lagi.

2.2 Tinjauan Pustaka

Menurut Asmoro (2010) Hasil analisa pada lingkungan internal dan eksternal SI/TI perusahaan, dapat disimpulkan bahwa PT.SILO memiliki sistem-sistem yang beroperasi secara terpisah satu sama lainnya. Hal ini menyebabkan pelaporan data dan informasi menjadi sulit ditelusuri asal-muasalnya.

Penelitian selanjutnya yang menjadi acuan adalah "Perencanaan Implementasi Sistem ERP Berbasis SAP Modul Material Management Pada Proses Purchasing Lokal (Material) PT. Krakatau Steel" oleh Amanda (2010). Penelitian ini menganalisa proses bisnis dari proyek

implementasi sistem ERP berbasis SAP pada modul Material Management. Hasil analisa yang dijadikan acuan adalah implementasi modul Material Management pada material *non-sparepart* khususnya *raw material*. Karakteristik material hampir sama dengan yang ada di PT Sebuku Iron Lateritic Ores dikarenakan PT Krakatau Steel juga membeli *raw material* dari PT Sebuku Iron Lateritic Ores.

Penelitian selanjutnya yang menjadi acuan adalah "Analisis dan Perancangan Sistem Informasi *Rig Non Productive Time* (NPT) pada Proses Pengeboran di PT. XYZ" oleh Armando (2013). Penelitian ini merancang Sistem Informasi *Rig Non Productive Time* guna dapat menghasilkan laporan NPT yang dipergunakan untuk analisis produktivitas rig. Karakteristik proses bisnis dan faktor yang mempengaruhi produktivitas unit produksi bisnis perminyakan dan pertambangan hampir sama. Proses bisnis terdiri dari eksplorasi, eksploitasi dan pengapalan. Faktor cuaca dan ketepatan hasil survey eksplorasi sangat berpengaruh pada produktivitas unit produksi pada bisnis perminyakan dan pertambangan.

Penelitian yang akan dilakukan selanjutnya adalah melakukan analisis dan perancangan sistem informasi management tambang di PT Sebuku Iron Lateritic Ores.