BAB 2
LANDASAN TEORI

2.1. Basis Data

Basis Data terdiri dari 2 kata, yaitu Basis dan Data. Basis kurang lebih dapat diartikan sebagai markas atau gudang, tempat bersarang atau berkumpul, sedangkan Data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, siswa, pembeli, pelanggan), barang, hewan, peristiwa, konsep, keadaan, dan sebagainya, yang diwujudkan dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya [1]. Basis Data adalah suatu ruang atau tempat penyimpanan yang dapat diibaratkan seperti gudang yang menyimpan berbagai data dari berbagai objek nyata seprti data pegawai, siswa, guru, dan lain-lain, data tersebut tersimpan dalam berbagai bentuk seperti angka, huruf, simbol, atau kombinasi lainnya.

Sebagai satu kesatuan istilah, Basis Data (*Database*) sendiri dapat didefinisikan dalam sejumlah sudut pandang seperti berikut [1].

a. Himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar dapat dimanfaatkan kembali dengan cepat dan mudah.
b. Kumpulan data yang saling behubungan yang disimpan secara bersama sedimikian rupa dan tanpa pengulangan (*redudancy*) yang tidak perlu, untuk memenuhi berbagai kebutuhan.
c. Kumpulan *file*/tabel/arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronis.

Operasi-operasi dasar yang dapat dilakukan pada suatu basis data sebagai berikut [1].

a. Pembuatan basis data baru (create database), merupakan pembuatan ruangan arsip yang baru.
b. Penghapusan basis data (drop database), merupakan penghancuran ruangan arsip beserta isinya.

c. Pembuatan tabel baru ke suatu basus data, merupakan penambahan map arsip baru ke sebuah ruangan arsip yang telah ada.
d. Penghapusan tabel dari suatu basis data (drop table), merupakan penghancuran map arsip lama yang ada pada suatu ruangan basis data.
e. Penambahan atau pengisian data baru ke sebuah tabel di sebuah basis data (insert), merupakan penambahan lembaran arsip ke sebuah map arsip.
f. Pengambilan data dari sebuah tabel (select), merupakan pencarian lembaran arsip dari sebuah map arsip.
g. Pengubahan data dari sebuah tabel (update), merupakan perbaikan isi lembaran arsip yang ada dari sebuah map arsip.
h. Penghapusan data dari sebuah tabel (delete), merupakan penghapusan sebuah lembaran arsip yang ada di sebuah map arsip.

2.2. *Structured Query Language*

SQL atau juga sering disebut sebagai *query* merupakan suatu bahasa (*language*) yang digunakan untuk mengakses *database*. SQL dikenalkan pertama kali dalam IBM pada tahun 1970 dan sebuah standar ISO dan ANSII ditetapkan untuk SQL. Standar ini tidak tergantung pada mesin yang digunakan (IBM, Microsoft, dan Oracle). Hampir semua *software database* mengenal atau mengerti SQL. Jadi, perintah SQL pada semua *software* DBMS hampir sama. SQL adalah suatu aturan bahasa yang digunakan untuk mengakses suatu basis data yang tersimpan secara elektronis.

2.2.1. *Data Definition Language*

Data Definition Language (DDL) adalah suatu metode perintah SQL yang digunakan untuk pendefinisian, pengubahan dan penghapusan suatu struktur basis data. Contoh penerapan DDL adalah sebagai berikut [9][10].

1. Membuat dan menghapus basis bata

   a. Membuat basis data

   Perintah umum SQL untuk membuat suatu *database* dapat dilihat pada gambar 2.1.

```
CREATE DATABASE [IF NOT EXISTS] nama_database;
```

Gambar 2. 1 Perintah Umum SQL Pembuatan Basis Data

Perintah pada gambar 2.1 akan membuat basis data baru dengan nama "nama_database". Aturan penamaan sebuah basis data sama seperti aturan penamaan sebuah variabel, dimana secara umum nama basis data boleh terdiri dari huruf, angka dan *under-score* (_). Jika basis data yang akan dibuat sudah ada, maka akan muncul pesan *error*. Namun jika ingin otomatis menghapus basis data yang lama jika sudah ada, maka diikuti perintah "if not exists". Contoh perintah untuk membuat basis data baru dengan nama "unikom" dapat dilihat pada gambar 2.2.

```
CREATE DATABASE unikom;
```

Gambar 2. 2 Contoh Perintah SQL Pembuatan Basis Data

   b. Menghapus basis data

   Menghapus suatu *database*, sintaks umumnya dapat dilihat pada gambar 2.3.

```
DROP DATABASE [IF EXISTS] nama_database;
```

Gambar 2. 3 Perintah Umum Penghapusan Basis Data

Perintah pada gambar 2.3 akan menghapus basis data dengan nama "nama_database". Jika basis data ada maka basis data dan juga seluruh tabel di dalamnya akan dihapus. Jika nama basis data yang akan dihapus tidak ditemukan, maka akan ditampilkan pesan *error*. Cara untuk menghidari *error* maka dapat diikuti perintah "if exists" untuk memastikan bahwa suatu basis data benar-benar ada. Contoh perintah untuk

menghapus basis data dengan nama “unikom” dapat dilihat pada gambar 2.4.

```
DROP DATABASE unikom;
```

Gambar 2. 4 Contoh Perintah Penghapusan Basis Data

2. Membuat, mengubah dan menghapus tabel
   a. Membuat tabel baru

   Bentuk umum sintak untuk membuat suatu tabel baru dapat dilihat pada gambar 2.5.

```
CREATE TABLE nama_tabel (
field1 tipe(panjang),
field2 tipe(panjang),
...
fieldn tipe(panjang),
PRIMARY KEY (field_key)
);
```

Gambar 2. 5 Perintah Umum Pembuatan Tabel

Perintah pada gambar 2.5 merupakan bentuk umum pembuatan tabel yang sudah disederhanakan. Penamaan tabel dan kolom memiliki aturan yang sama dengan penamaan basis data. Contoh stuktur tabel yang akan dibuat dapat dilihat pada tabel 2.1.

Tabel 2. 1 Contoh Struktur Tabel Mahasiswa

No	Nama Field	Tipe	Panjang
1	nim	varchar	8
2	nama_mhs	varchar	30
3	alamat	text	-
4	telepon	varchar	20
5	email	varchar	50

Untuk membuat tabel pada basis data sesuai struktur yang sudah dibuat seperti pada tabel 2.1. *query* atau perintah SQL-nya dapat dilihat pada gambar 2.6.

```
CREATE TABLE mahasiswa (
      nim varchar(8) NOT NULL,
      nama_mhs varchar(30) NOT NULL,
      alamat text,
      telepon varchar (20),
      email varchar (50),
      PRIMARY KEY(nim)
);
```

Gambar 2. 6 Contoh Perintah Membuat Tabel

Perintah pada gambar 2.6 terdapat beberapa hal yang perlu diperhatikan, antara lain:

1. Kode ‘create table’ merupakan perintah dasar dari pembuatan tabel.
2. Kata ‘mahasiswa’ merupakan nama tabel yang akan dibuat.
3. Kata nim, nama, alamat, telepon dan email merupakan nama *field*.
4. varchar dan text merupakan tipe data dari *field*.
5. Kode ‘not null’ merupakan *option* untuk menyatakan bahwa suatu *field* tidak boleh kosong.
6. Kode ‘primary key’ merupakan perintah untuk menentukan *field* mana yang akan dijadikan kunci utama pada tabel.
7. Angka 5, 10, 30 dan 50 di belakang tipe data merupakan panjang maksimal dari suatu *field*.
8. Untuk tipe data ‘date’ dan ‘text’, panjang karakter maksimalnya tidak perlu ditentukan.
9. Perintah diakhiri dengan titik-koma (;).

b. Membuat tabel baru dengan menyalin struktur dari tabel lain

Perintah umum SQL untuk membuat suatu *database* dengan menyalin struktur tabel lain dapat dilihat pada gambar 2.7.

```
CREATE TABLE tabel_baru
AS (SELECT [field | *] FROM tabel_asli);
```

Gambar 2. 7 Sintak Umum Membuat Tabel Sesuai Tabel Lain

Contoh *query* membuat tabel baru sesuai pada tabel mahasiswa dengan struktur pada tabel 2.1 dapat dilihat pada gambar 2.8 dan 2.9.

```
CREATE TABLE backup_mahasiswa
AS (SELECT * FROM mahasiswa);
```

Gambar 2. 8 Contoh Membuat Tabel Sesuai Tabel Lain

```
CREATE TABLE backup_mahasiswa AS (SELECT nim, nama,
alamat FROM mahasiswa);
```

Gambar 2. 9 Contoh Membuat Tabel Sesuai Kolom Tabel Lain

c. Mengubah struktur tabel dengan alter

Bentuk umum perintah SQL untuk mengubah struktur suatu tabel dapat dilihat pada gambar 2.10.

```
ALTER TABLE nama_tabel alter_options;
```

Gambar 2. 10 Sintak Umum Mengubah Struktur Tabel

Perintah pada Gambar 2.10 memiliki beberapa hal yang perlu diperhatikan, yaitu:

1. ‘alter table’ merupakan perintah dasar untuk mengubah tabel.
2. ‘nama_tabel’ merupakan nama tabel yang akan diubah strukturnya.
3. ‘*alter_options*’ merupakan pilihan perubahan tabel. *Option* yang bisa digunakan yaitu add, add index, add primary key, change, modify, drop, dan rename to.

Beberapa contoh variasi perintah alter untuk mengubah struktur suatu tabel antara lain sebagai berikut.

1. Menambah kolom

   Fungsi alter untuk menambahan kolom pada tabel menggunakan *option* ‘add’. Contoh *query* untuk menambah kolom ‘tgllahir’ ke tabel mahasiswa dapat dilihat pada gambar 2.11.

```
ALTER TABLE mahasiswa ADD tgllahir date NOT NULL;
```

Gambar 2. 11 Contoh Perintah SQL Menambah *Field*

2. Menambahkan *index* kolom

   Fungsi alter untuk menambahkan *index* kolom pada tabel menggunakan *option* ‘add index’. Contoh *query* untuk menambah *index* “idx_nim” pada kolom nim di tabel mahasiswa dapat dilihat pada gambar 2.12.

```
ALTER TABLE mahasiswa ADD INDEX idx_nim(nim);
```

Gambar 2. 12 Contoh Perintah SQL Menambah Index Kolom

3. Menambahkan *primary key*

   Fungsi alter untuk menambahkan kata kunci utama pada salah satu kolom tabel menggunakan *option* ‘add primary key’. Contoh *query* untuk menambah *primary key* pada tabel dapat dilihat pada gambar 2.13.

```
ALTER TABLE mahasiswa ADD PRIMARY KEY(nim);
```

Gambar 2. 13 Contoh Perintah SQL Menambah *Primary Key*

4. Mengubah tipe kolom

   Fungsi alter untuk mengubah tipe data salah satu kolom pada tabel menggunakan *option* ‘modify’. Contoh *query* untuk mengubah tipe kolom ‘tgllahir’ menjadi varchar dalam tabel mahasiswa dapat dilihat pada gambar 2.14.

```
ALTER TABLE mahasiswa MODIFY tgllahir varchar(8) NOT
NULL;
```

Gambar 2. 14 Contoh Perintah SQL Mengubah Tipe *Field*

5. Mengubah nama kolom

   Fungsi alter untuk mengubah nama salah satu kolom pada tabel menggunakan *option* ‘change’. Contoh *query* untuk mengganti nama kolom ‘tgllahir’ dapat dilihat pada gambar 2.15.

```
ALTER mahasiswa CHANGE tgllahir tgl_lahir;
```

Gambar 2. 15 Contoh Perintah SQL Mengubah Nama Kolom

6. Menghapus *field*

   Fungsi alter untuk menghapus salah satu kolom pada tabel menggunakan *option* ‘drop’. Contoh *query* untuk menghapus *field* ‘tgllahir’ dari tabel mahasiswa dapat dilihat pada gambar 2.16.

```
ALTER TABLE mahasiswa DROP tgllahir;
```

Gambar 2. 16 Contoh Perintah SQL Menghapus *Field*

7. Mengubah nama tabel

   Fungsi alter untuk mengubah nama tabel menggunakan *option* ‘rename to’. Contoh *query* untuk mengubah nama tabel mahasiswa dapat dilihat pada gambar 2.17.

```
ALTER TABLE mahasiswa RENAME TO tb_mahasiswa;
```

Gambar 2. 17 Contoh Perintah SQL Mengubah Nama Tabel

d. Mengubah nama tabel

   Untuk mengubah nama suatu tabel, dapat menggunakan perintah SQL yang dapat dilihat pada gambar 2.18.

```
RENAME TABLE mahasiswa TO tb_mahasiswa;
```

Gambar 2. 18 Contoh Perintah SQL Mengubah Nama Tabel

e. Menghapus Tabel

   Untuk menghapus sebuah tabel, bentuk umum dari perintah SQL dapat dilihat pada gambar 2.19.

```
DROP TABLE nama_tabel;
```

Gambar 2. 19 Perintah Umum SQL Menghapus Tabel

Sebagai contoh untuk menghapus tabel dengan nama ‘mahasiswa’ maka perintah SQL-nya dapat dilihat pada gambar 2.20.

```
DROP TABLE mahasiswa;
```

Gambar 2. 20 Contoh Perintah SQL Menghapus Tabel

2.2.2. *Data Manipulation Language*

Data Manipulation Language (DML) adalah suatu metode perintah SQL yang digunakan untuk memanipulasi data atau *record* suatu tabel. DML terdiri atas perintah – perintah yang beroprasi pada data di dalam database. Ini mencakup pernyataan – pernyataan untuk menambahkan data ke dalam tabel atau pernyataan yang dipakai untuk melakukan *query* terhadap database [9]. Contoh penerapan DML adalah sebagai berikut [9][10].

1. Mengambil data dari tabel

   Perintah select digunakan untuk menampilkan sesuatu. Sesuatu di sini bisa berupa sejumlah data dari tabel dan bisa juga berupa suatu ekspresi. Dengan perintah select dapat mengatur tampilan atau keluaran data sesuai yang diinginkan. Perintah umum untuk mengambil data dari tabel pada gambar 2.21.

   ```
   SELECT [field | *] FROM nama_tabel [WHERE kondisi];
   ```

   Gambar 2. 21 Perintah Umum Mengambil Data

   Perhatikan beberapa contoh penggunaan perintah select dari tabel mahasiswa sebagai berikut.

   a. Menampilkan seluruh kolom (*) dari tabel mahasiswa pada gambar 2.19.

   ```
   SELECT * FROM mahasiswa;
   ```

   Gambar 2. 22 Perintah SQL Tampil Seluruh Data

b. Menampilkan *field* nim dan nama_mhs dari seluruh mahasiswa dalam tabel mahasiswa pada gambar 2.23.

```
SELECT nim, nama_mhs FROM mahasiswa;
```

Gambar 2. 23 Perintah SQL Tampil Beberapa *Field*

2. Menambah data pada tabel

Pernyataan insert untuk menambahkan data ke dalam tabel. Perintah umumnya dapat dilihat pada gambar 2.24.

```
INSERT INTO nama_tabel ([field | *])
VALUES ([nilai | *]);
```

Gambar 2. 24 Perintah Umum Menambah Banyak Data

Adapun perintah insert tanpa menentukan kolom tabel dapat dilihat pada gambar 2.25.

```
INSERT INTO nama_tabel VALUES ([nilai | *]);
```

Gambar 2. 25 Perintah Menambah Data Tanpa Kolom

Perhatikan beberapa contoh penggunaan perintah insert pada tabel mahasiswa sebagai berikut.

a. Menambahkan data pada tabel mahasiswa.

```
INSERT INTO mahasiswa (nim, nama)
VALUES (10114497,Iqram Anwar);
```

Gambar 2. 26 Perintah SQL Menambahkan Data

b. Menambah data tanpa menentukan kolom pada tabel mahasiswa.

```
INSERT INTO mahasiswa
VALUES (10114497,Iqram Anwar);
```

Gambar 2. 27 Perintah SQL Mengubah Data Tanpa kolom

2.3. Pemrosesan Bahasa Alami (*Natural Language Processing*)

Pemrosesan bahasa alami (*Natural Language Processing*) adalah salah satu bidang ilmu kecerdasan buatan (*Artificial Intelligence*) yang mengkaji tentang komunikasi antar manusia dengan komputer melalui bahasa alami baik secara lisan maupun tulisan.

Proses representasi bahasa dalam bentuk yang lebih memungkinkan untuk dikomputasi telah dilakukan oleh seorang ahli bernama Chomsky pada tahun 1957 [11]. Pada bidang ilmu pemrosesan bahasa alami tidak mudah untuk dilakukan karena terdapat beberapa alasan yang menyulitkan pada pemrosesan bahasa alami. Salah satu kesulitan dalam pemrosesan bahasa alami adalah pemilihan arti yang tepat dari suatu kata bermakna ganda seperti kata ‘bisa’ yang dapat berarti ‘racun’ atau ‘dapat’ sesuai dengan kalimatnya [11]. Fenomena ini terjadi dalam penentuan jenis kata (*part of speech*) seperti kata ‘*advanced*’ yang dapat berfungsi sebagai kata kerja aktif lampau, kata kerja pasif, atau kata sifat. Selain itu, jumlah kosa kata (*vocabulary*) dalam bahasa alami besar dan terus berkembang dari waktu ke waktu [11].

Pemrosesan Bahasa Alami sering diterapkan pada aplikasi tertulis yang sering menemui kesulitan karena tulisan setiap orang tidak selalu baku sesuai dengan tata bahasa sehingga tidak sesuai dengan aturan – aturan yang berlaku. Salah satu aplikasi dari pemrosesan bahasa alami adalah sistem penerjemah bahasa alami (*natural language translator*) seperti sistem penerjemah dari bahasa Inggris ke bahasa Indonesia. *Translator* tidak hanya dapat menerjemahkan kata per kata (kamus) tetapi juga dapat mentranslasikan dari bahasa asal ke bahasa target dengan maksud yang dapat dimengerti [11].

2.4. Kalimat

Kalimat adalah satuan bahasa yang disusun konstituen dasar yang umumnya berupa klausa, kata penghubung (jika ada) dan intonasi final. Dalam ragam tulisan intonasi final ini dinyatakan dengan titik (.), tanda tanya (?), atau tanda seru (!) [12]. Menurut strukturnya, sebuah kalimat sederhana dalam bahasa Indonesia memiliki pola sebagai berikut [13].

Tabel 2. 2 Pola Kalimat Sederhana Berdasarkan Strukturnya

No	Struktur	Contoh
1	Subjek + Predikat (S-P)	1. Para pedagang mengeluh. 2. Orang itu sedang tidur.
2	Subjek + Predikat + Objek (S-P-O)	1. Ilmu mengatur pengetahuan. 2. Ayahnya membeli mobil baru.
3	Subjek + Predikat + Pelengkap (S-P-Pel)	Beliau menjadi ketua koperasi.
4	Subjek + Predikat + Objek + Pelengkap (S-P-O-Pel)	1. Ia menyebutnya si hitam. 2. Dian mengambilkan adiknya air minum.
5	Subjek + Predikat + Objek + Keterangan (S-P-O-K)	Pak Raden memasukan uang ke bank.
6	Subjek + Predikat + Keterangan (S-P-K)	Kami tinggal di jakarta

Dalam tata bahasa Indonesia, fungsi gramatikal *subjek, predikat,* dan *objek* dapat ditempati oleh berbagai macam jenis kata maupun frasa. Tabel 2.3 berikut adalah daftar fungsi gramatikal dan jenis kata atau frasa yang dapat menempatinya.

Tabel 2. 3 Daftar Fungsi Gramatikal dan Pengisiannya

Fungsi Gramatikal	Keterangan	Pengisi
Subjek	Sebuah kata atau frasa yang letaknya di depan predikat.	1. Kata benda / nomina 2. Frasa nomina 3. Kata kerja / verba 4. Kata bilangan
Predikat	Bagian dari klausa yang memberikan informasi tentang subjek.	1. Kata Kerja / verba 2. Frasa verba 3. Adjektiva 4. Kata benda / nomina 5. Frasa nomina 6. Kata / frasa
Objek	Bagian dari klausa yang terkena tindakan yang disebut di dalam predikat.	1. Kata benda / nomina 2. Frasa nomina

Tabel 2. 3 Daftar Fungsi Gramatikal dan Pengisiannya (Lanjutan)

Fungsi Gramatikal	Keterangan	Pengisi
Pelengkap	Bagian dari klausa yang mengikuti atau melengkapi predikat.	1. Adverbia 2. Frasa adverbia
Keterangan	Bagian dari klausa yang menjelaskan jenis tindakan yang dinyatak di dalam predikat. Biasanya keterangan berupa kata keterangan. Posisinya dalam kalimat bisa diubah (depan,tengah, belakang).	Frasa nomina

Berdasarkan jumlah subjek dan predikatnya (jumlah klausa), kalimat dikelompokan menjadi kalimat tunggal dan kalimat majemuk [13].

2.5. Kalimat Tunggal

Kalimat tunggal adalah kalimat yang terdiri atas satu klausa. Kalimat ini hanya mempunyai satu subjek dan satu predikat. Kalimat tunggal bisa berwujud kalimat dasar atau kalimat dasar yang diperluas, meskipun sudah mengalami perluasan, fungsi subjek dan predikatnya tetap berjumlah satu [13].

Contoh :

a. Separuh pesisir Pulau Bangka Rusak.
b. Separuh pesisir Pulau Bangka, Provinsi Kepulauan Bangka Belitung, rusak.

Kalimat pada contoh (a) merupakan kalimat dasar, sedangkan kalimat pada contoh (b) merupakan kalimat tunggal hasil perluasan karena subjek kalimat telah diperluas oleh unsur keterangan. Meskipun demikian, subjek pada kalimat tersebut tetap berjumlah satu.

2.6. Kalimat Majemuk

Kalimat majemuk adalah kalimat yang terdiri atas dua klausa atau lebih yang dimana antara klausa yang satu dengan klausa yang lain saling berhubungan [13]. Umumnya, ahli bahasa membagi hubungan antar klausa ke dalam dua jenis yaitu koordinasi dan subordinasi.

1. Hubungan koordinasi (Majemuk Setara)

   Hubungan koordinasi menggabungkan dua klausa atau lebih yang masing – masing mempunyai kedudukan setara dalam struktur kalimat [13]. Kalimat yang terbentuk dari dua klausa atau lebih dengan hubungan antar klausa terbentuk secara koordinatif dapat disebut dengan kalimat majemuk setara, sehingga kalimat tersebut terdiri dari klausa – klausa yang disusun secara koordinasi dengan kata penghubung yang dimana setiap klausa dapat berdiri sendiri menjadi kalimat.

   Contoh :

   "Candi Gedong Songo memiliki sembilan kelompok candi, tetapi sebagian kelompok candi itu sudah hilang."

   Kalimat diatas terdiri atas dua klausa : (a) Candi Gedong Songo memiliki sembilan kelompok candi; (b) sebagian kelompok candi itu sudah hilang. Kalimat tersebut dihubungkan oleh secara koordinasi oleh kata 'tetapi' sehingga terbentuk kalimat majemuk setara.

2. Hubungan Subordinasi (Majemuk Bertingkat)

   Hubungan subordinasi menunjukkan hubungan yang hierarkis, yakni menggabungkan dua klausa atau lebih secara bertingkat, ada yang berfungsi sebagai klausa utama dan ada yang berfungsi sebagai klausa bawahan [13]. Kalimat yang terbentuk dari beberapa klausa yang dihubungkan dengan kata penghubung yang bersifat subordinatif menyebabkan klausa yang satu menjadi bagian dari klausa yang lain, artinya salah satu klausa bergantung pada klausa lain dan tidak bisa berdiri sendiri. Kalimat tersebut disebut dengan kalimat majemuk bertingkat.

   Contoh :

   "Candi Gedong Songo merupakan mutiara kehidupan karena menjadi sumber nafkah bagi masyarakat sekitarnya."

   Kalimat di atas terdiri dari dua klausa, yaitu (a) "Candi Gedong Songo merupakan mutiara kehidupan", dan (b) " (Candi Gedong Songo) menjadi sumber nafkah bagi masyarakat sekitarnya". Kedua klausa

dihubungkan oleh konjungsi ‘karena’. Klausa (a) merupakan klausa utama, sedangkan klausa (b) merupakan klausa bawahan [13].

Daftar kata penghubung dan ciri – cirinya dapat dilihat pada Tabel 2.4.

Tabel 2. 4 Daftar Kata Penghubung Kalimat Majemuk

No	Kalimat Majemuk	Penghubung Klausa	Ciri-Ciri Penghubung
1	Majemuk setara / Koordinatif	1. dan 2. atau 3. tapi 4. namun 5. serta 6. lalu 7. kemudian 8. bahkan 9. sedangkan 10. sementara 11. padahal 12. bukan ...melainkan	lokasi penghubung tetap, yaitu di awal klausa koordinatif.
2	Majemuk bertingkat / Subordinatif	1. karena / sebab 2. ketika 3. sejak 4. jikalau 5. asalkan 6. meskipun / walaupun 7. bila / apabila 8. maka 9. sehingga	Subordinator selalu mengikuti unsur intern klausa utama sebagai informasi lanjutan. Karena fungsinya itu, subordinator selalui terletak di awal klausa subordinatif / bawahan.

2.7. Kalimat Perintah

Kalimat perintah adalah kalimat yang isinya mengharapkan adanya reaksi berupa tindakan atau perbuatan dari orang yang diajak bicara (pendengar atau pembaca) [14]. Kalimat perintah dilihat dari taraf tindakan yang diharapkan dapat dibedakan menjadi 3 yaitu kalimat perintah yang tegas, kalimat perintah yang biasa, dan kalimat perintah yang halus.

a. Kalimat perintah tegas

Kalimat perintah yang tegas dibentuk dari sebuah klausa tidak lengkap, biasanya hanya berupa kata kerja dasar, disertai dengan intonasi kalimat perintah dan dalam bahasa tulis, intonasi kalimat

perintah ini diganti atau dilambangkan dengan tanda seru [14]. Contoh :

1. Tangkap!
2. Tembak!

Jika contoh dilengkapi dengan objek atau keterangan, agar lebih jelas atau agar tidak menimbulkan salah paham. Misalnya contoh yang sebelumnya dibuat menjadi "Tangkap orang itu!".

b. Kalimat perintah biasa

Kalimat perintah yang biasa dibentuk dari sebuah klausa berpredikat kata kerja dasar yang diberi partikel "-lah", serta dengan menghilangkan atau melesapkan subjeknya [14]. Contoh :

1. Jagalah kebersihan!
2. Bayarlah dengan uang pas!
3. Belilah karcis di loket!

Jika kalimat perintah digunakan pada orang yang sudah ditentukan, maka subjek pada klausa tersebut harus ditampilkan, misalnya contoh yang sebelumnya dibuat menjadi "Iqram, jagalah kebersihan!".

c. Kalimat perintah halus

Kalimat perintah yang halus harus digunakan oleh yang lebih muda kepada yang lebih tua, yang lebih rendah status atau kedudukan sosialnya terhadap yang lebih tinggi atau yang lebih berkuasa, atau juga untuk menampilkan rasa hormat atau sopan santun terhadap orang yang diperintah [14]. Contoh :

1. Tolong bereskan buku – buku ini!
2. Silahkan duduk di sini!

2.8. *Case Folding*

Case folding adalah proses penyeragaman semua teks ke dalam *case* yang sama menjadi huruf kecil (*lowercase*) atau huruf capital (*uppercase*) [15]. Pada penelitian ini setiap huruf dalam teks masukan diseragamkan menjadi huruf kecil.

Contoh dari tahap *case folding* dapat dilihat pada tabel 2.5. Tabel tersebut menjelaskan bahwa setiap huruf kapital yang ada pada teks input akan diubah menjadi huruf kecil.

Tabel 2. 5 Contoh Tahap *Case Folding*

Kalimat Masukan	Sesudah *Case Folding*
Buatkan Database Dengan Nama Mahasiswa	**b**uatkan **d**atabase **d**engan **n**ama **m**ahasiswa

Penelitian ini menggunakan tahap – tahap proses *case folding* untuk menyeragamkan semua huruf pada kalimat masukan menjadi huruf kecil (*lowercase*).

2.9. *Filtering*

Filtering merupakan proses pengecekan pada setiap karakter pada teks selain karakter ‘a’ sampai ‘z’, ‘0’ sampai ‘9’, ‘_‘, ‘,’ dan spasi akan dihapus. Berikut adalah contoh pada tahap *filtering.*

Tabel 2. 6 Contoh Tahap *Filtering*

Kalimat Masukan	Sesudah *filtering*
buatkan **‘**database**’** dengan nama mahasiswa**!**	buatkan database dengan nama dbmahasiswa

Penelitian ini menggunakan tahap *filtering* untuk memindai setiap karakter lalu menghapus karakter lain selain karakter ‘a’ sampai ‘z’, ‘0’ sampai ‘9’, ‘_‘, ‘,’ dan spasi.

2.10. *Tokenizing* Kata

Tokenizing kata adalah proses memindai teks masukan yang akan dipotong menjadi kumpulan kata. Pemecahan kalimat menjadi kata – kata tunggal dilakukan dengan memindai kalimat dan memisahkan kata demi kata dengan spasi. Hasil keluaran dari proses *tokenizing* kata akan digunakan sebagai masukan dalam tahap transformasi teks. Contoh *tokenizing* kata dapat dilihat pada tabel 2.7.

Tabel 2. 7 Contoh Tahap *Tokenizing* Kata

Kalimat Masukan	Sesudah *Tokenizing* Kata
buatkan database dengan nama mahasiswa	buatkan | database | dengan | nama | mahasiswa

Penelitian ini menggunakan tahap *tokenizing* kata untuk memotong teks masukan menjadi kumpulan kata.

2.11. *Stemming*

Stemming merupakan suatu proses yang mentransformasi kata-kata yang berimbuhan ke kata-kata dasarnya (*root word*) [15]. Sebagai contoh, kata 'menyenangkan', 'menyenangi', dapat dikatakan serupa atau satu kelompok dan dapat diwakili oleh satu kata umum 'senang'. Proses *stemming* pada teks berbahasa Indonesia berbeda dengan stemming pada teks berbahasa Inggris. Pada teks berbahasa Inggris, proses yang diperlukan hanya proses menghilangkan sufiks, sedangkan pada teks berbahasa Indonesia, selain sufiks, prefiks, dan konfiks juga dihilangkan. Pada umumnya kata dasar pada bahasa Indonesia terdiri dari kombinasi:

Prefiks 1 + Prefiks 2 + Kata dasar + Sufiks 3 + Sufiks 2 + Sufiks 1 (2.1)

Pada penelitian ini, algoritma yang digunakan dalam proses *stemming* yaitu Nazief dan Andriani yang telah dikembangkan oleh Sastrawi. Algoritma *Stemming* Bahasa Indonesia M. Adriani dan B. Nazief ini mempunyai aturan imbuhan sendiri dengan model sebagai berikut.

[[[AW+]AW+]AW+] Kata-Dasar [[+AK][+KK][+P]] (2.2)

Dimana:

AW : Awalan

AK : Akhiran

KK : Kata Ganti kepunyaan

P : Partikel

Algoritma yang dibuat oleh Bobby Nazief dan Mirna Adriani ini memiliki tahap sebagai berikut.

1. Mencari kata yang akan diproses *stemming* dalam kamus. Jika ditemukan maka diasumsikan bahwa kata tesebut adalah *root word*, maka algoritma berhenti.

2. *Inflection Suffixes* (“-lah”, “-kah”, “-ku”, “-mu”, atau “-nya”) dibuang. Jika berupa *particles* (“-lah”, “-kah”, “-tah” atau “-pun”) maka langkah ini diulangi lagi untuk menghapus *Possesive Pronouns* (“-ku”, “-mu”, atau “-nya”), jika ada.
3. Hapus *Derivation Suffixes* (“-i”, “-an” atau “-kan”). Jika kata ditemukan dikamus, maka algoritma berhenti. Jika tidak maka ke langkah 3a.
    a. Jika “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k”, maka “-k” juga ikut dihapus. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan maka lakukan langkah 3b.
    b. Akhiran yang dihapus (“-i”, “-an” atau “-kan”) dikembalikan, lanjut ke langkah 4.
4. Hapus *Derivation Prefix*. Jika pada langkah 3 ada sufiks yang dihapus maka pergi ke langkah 4a, jika tidak pergi ke langkah 4b.
    a. Periksa tabel kombinasi awalan-akhiran yang tidak diijinkan. Jika ditemukan maka algoritma berhenti, jika tidak pergi ke langkah 4b.
    b. For i = 1 to 3, tentukan tipe awalan kemudian hapus awalan. Jika *root word* belum juga ditemukan lakukan langkah 5, jika sudah maka algoritma berhenti. Catatan: jika awalan kedua sama dengan awalan pertama algoritma berhenti.
5. Melakukan *Recoding*.
6. Jika semua langkah telah selesai tetapi tidak juga berhasil maka kata awal diasumsikan sebagai *root word* dan proses dianggap selesai.

Tipe awalan ditentukan melalui langkah-langkah sebagai berikut.

1. Jika awalannya adalah: “di-”, “ke-”, atau “se-” maka tipe awalannya secara berturut-turut adalah “di-”, “ke-”, atau “se-”.
2. Jika awalannya adalah “te-”, “me-”, “be-”, atau “pe-” maka dibutuhkan sebuah proses tambahan untuk menentukan tipe awalannya.
3. Jika dua karakter pertama bukan “di-”, “ke-”, “se-”, “te-”, “be-”, “me-”, atau “pe-“ maka berhenti.
4. Jika tipe awalan adalah “none” maka berhenti. Jika tipe awalan adalah bukan “none” maka awalan ditemukan, hapus awalan jika ditemukan.

Sebagai contoh dapat dilihat pada tabel 2.8 kata ‘buatkan’ akan diubah menjadi kata ‘buat’ karena pada kata ‘buatkan’ terdapat imbuhan “–kan” [16].

Tabel 2. 8 Contoh Tahap *Stemming*

Kalimat Masukan	Sesudah *Stemming*
buatkan database dengan nama mahasiswa	buat database dengan nama mahasiswa

Pada penelitian ini digunakan *library stemming* dengan PHP untuk bahasa Indonesia dari Sastrawi. Karena *library* ini menggunakan tiga algoritma yaitu algoritma *Nazief dan Adriani*, algoritma *CS* Stemmer dan algoritma *ECS Stemmer*. Dimana algoritma *CS Stemmer* dan algoritma *ECS Stemmer* memodifikasi beberapa aturan yang terdapat di algoritma *Nazief dan Adriani*. Pada penelitian ini juga beberapa kata tidak dilakukan proses *stemming* karena proses *stemming* hanya untuk mendapatkan kata kunci.

2.12. *Removing Stopword*

Stopword merupakan kata yang diabaikan dalam pemrosesan, kata-kata ini biasanya disimpan ke dalam *stop lists*. Karakteristik utama dalam pemilihan *stopword* biasanya adalah kata yang mempunyai frekuensi kemunculan yang tinggi misalnya kata penghubung seperti ‘dan’, ‘atau’, ‘tapi’, ‘akan’ dan lainnya. Tidak ada aturan pasti dalam menentukan *stopword* yang akan digunakan, penentuan s*topword* bisa disesuaikan dengan kasus yang sedang diselesaikan [17]. Beberapa kata yang termasuk *stopword* dapat dilihat pada tabel 2.9.

Tabel 2. 9 Beberapa Daftar Kata Yang Termasuk *Stopword*

No	Kata *Stopword*	No	Kata *Stopword*
1	coba	11	untuk
2	tolong	12	dengan
3	ada	13	apa
4	agar	14	maka
5	akan	15	lagi
6	berapa	16	lain
7	dan	17	jadi
8	yang	18	jika
9	pada	19	maka
10	yaitu	20	itu

Pada tabel 2.9 merupakan beberapa daftar kata yang termasuk *stopword*, penelitian ini hanya menggunakan 320 kata dasar dari 758 kata *stopword* yang digunakan oleh Fadillah Z Tala [17] yang dapat dilihat pada lampiran B-1 dan disesuaikan dengan menghapus 3 kata ("buat", "tambah", dan "semua") karena kata tersebut digunakan pada kamus kata kunci untuk menentukan perintah DDL dan menambah 3 kata ("tolong", "coba", "menjadi") karena kata tersebut tidak digunakan pada kamus kata kunci.

2.13. Akurasi

Untuk menghitung akurasi dari hasil query dilakukan perhitungan dengan rumus berikut [18].

$$Akurasi = \frac{Jumlah\ hasil\ query\ sistem\ benar}{Jumlah\ sampel\ data\ uji}\ X\ 100\% \quad (2.3)$$

Perhitungan akurasi digunakan untuk mengetahui tingkat keberhasilan dari proses translasi yang dilakukan oleh sistem dengan membandingakan *query* harapan dan *query* hasil translasi.

2.14. Pemodelan

Skema pemodelan data adalah metode yang memungkinkan untuk memodelkan gambaran proses sistem pada aplikasi yang akan dibangun.

2.14.1. *Flowchart*

Flowchart adalah bagan-bagan yang mempunyai arus yang menggambarkan langkah-langkah penyelesaian suatu masalah. *Flowchart* merupakan cara penyajian suatu algoritma. *Flowchart* adalah penggambaran secara grafik dari langkah-langkah dan urutan-urutan prosedur dari suatu program. *Flowchart* biasanya mempermudah penyelesaian suatu masalah khususnya masalah yang perlu dipelajari dan dievaluasi lebih lanjut [19].

Tujuan membuat *Flowchart* adalah Menggambarkan alur proses suatu penyelesaian masalah secara jelas menggunakan simbol – simbol standar.

2.14.2. Diagram Konteks

Diagram Konteks adalah sebuah diagram sederhana yang menggambarkan hubungan antara *entity* luar, masukan dan keluaran dari sistem. Diagram konteks direpresentasikan dengan lingkaran tunggal yang mewakili keseluruhan sistem [19]. Diagram konteks dimulai dengan penggambaran terminator, aliran data, aliran kontrol penyimpanan, dan proses tunggal yang menunjukan keseluruhan sistem. Ada beberapa aturan dalam menggambarkan diagram konteks, berikut di bawah ini menunjukan aturan tersebut.

1. Menggunakan hanya satu simbol proses.
2. Memberi label simbol proses tersebut untuk menggambar seluruh sistem, biasanya berupa kata kerja ditambah objek.
3. Tidak memberi nomor pada simbol proses.
4. Menyertakan semua terminator dari sistem.
5. Menunjukan semua arus data antara terminator dan sistem.

Pada penelitian ini diagram konteks digunakan untuk menggambarkan rancangan alur proses dan data pada sistem berbasis komputer secara umum dan menyeluruh.

2.14.3. DFD (*Data Flow Diagram*)

DFD adalah suatu model logika data atau proses yang dibuat untuk menggambarkan darimana asal data dan kemana tujuan data yang keluar dari sistem, dimana data disimpan, proses apa yang menghasilkan data tersebut dan interaksi antara data yang tersimpan dan proses yang dikenakan pada data tersebut [19].

1. Arus Data (*Data Flow*)

   Menunjukan arus dari data yang dapat berupa masukan untuk sistem atau dari proses sistem.

2. Proses

   Proses adalah kegiatan yang dilakukan oleh orang, komputer dari hasil arus data yang masuk ke dalam proses untuk dihasilkan arus data yang akan keluar dari proses.

3. Kesatuan Luar (*External Entity*)

   Kesatuan luar merupakan kesatuan di lingkungan luar sistem yang dapat berupa orang, organisasi atau sistem lain yang akan memberikan masukan atau menerima keluaran dari sistem.

4. *File*

   Kumpulan data yang disimpan dengan cara tertentu. Data yang mengalir disimpan dalam *file*. Aliran data di-update atau ditambahkan ke dalam file. Pada penelitian ini *Data Flow Diagram* (DFD) digunakan untuk menggambarkan rancangan dan alur proses sistem secara detail yang mudah dipahami oleh pemakai maupun pembuat sistem.

2.14.4. Kamus Data

Kamus data adalah kumpulan daftar elemen data yang mengalir pada sistem perangkat lunak sehingga masukan (*input*) dan keluaran (*output*) dapat dipahami secara umum [19]. Kamus data mendefinisikan elemen data dengan menjelaskan arti aliran data dan penyimpanan data dalam DFD, mendeskripsikan komposisi paket data yang bergerak melalui aliran, mendeskripsikan komposisi penyimpanan data, menspesifikasikan nilai dan satuan yang relevan bagi penyimpanan dan aliran, mendeskripsikan hubungan detil antar penyimpanan.

Kamus data berfungsi untuk membantu pelaku sistem untuk mengartikan alokasi secara detail dan mengorganisasikan semua elemen data yang digunakan dalam sistem secara persis sehingga baik pemakai atau penganalisis sistem mempunyai dasar pengertian yang sama tentang masukan, keluaran, penyimpanan dan proses. Kamus data biasanya berisi:

1. Nama - nama dari data
2. Digunakan pada – merupakan proses-proses yang terkait data
3. Deskripsi – merupakan deskripsi data
4. Informasi tambahan – seperti tipe data, nilai data, batas nilai data, dan komponen yang membentuk data.

Kamus data memiliki simbol untuk menjelaskan informasi tambahan seperti pada tabel 2.11.

Tabel 2. 10 Kamus Data

Simbol	Keterangan
=	disusun atau terdiri dari
+	dan
[|]	baik ... atau....
$\{\}^n$	n kali diulang/ bernilai banyak
()	data opsional
...	batas komentar

Pada penelitian ini kamus data digunakan untuk memberikan informasi mengenai detail penggunaan deskripsi dari proses yang ada pada DFD.

2.15. Bahasa Pemograman

2.15.1. PHP

PHP (*Personal Home Page*) merupakan *script* yang menyatu dengan kode HTML dan berjalan pada sisi server (*server side*), artinya semua sintaks yang kita berikan akan sepenuhnya dijalankan pada sisi server dan yang dikirmkan ke browser hanya hasilnya [20]. Keuntungan menggunakan PHP adalah seluruh *syntax* PHP yang telah kita *coding* tidak akan tampak pada halaman browser ketika pengguna menggunakan fungsi *inspect element*, sehingga kerahasiaan kode terjamin. PHP yang berjalan pada sisi server harus menggunakan *web server* untuk menggunakannya. Sistem yang dibangun pada penelitian ini menggunakan PHP versi 5.

2.15.2. *Java Script*

JavaScript adalah bahasa *script* berdasar pada objek yang memperbolehkan pemakai untuk mengendalikan banyak aspek interaksi pemakai pada suatu dokumen HTML, dimana objek tersebut dapat berupa suatu *window*, *frame*, URL, dokumen, *form*, *button*, atau item yang lain. Semuanya itu mempunyai properti yang saling berhubungan, dan masing-masing memiliki nama, lokasi, warna nilai, dan atribut lainnya [21]. *Javascript* digunakan untuk tampilan website agar lebih dinamis.

2.16. *Software* Pendukung

2.16.1. XAMPP

XAMPP adalah perangkat lunak bebas (*free software*), yang mendukung untuk banyak sistem operasi, yang merupakan kompilasi dari beberapa program. Fungsi XAMPP sendiri adalah sebagai *server* yang berdiri sendiri (localhost), yang terdiri beberapa program, antara lain yaitu *Apache* HTTP *Server*, *MySQL database*, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl. Nama XAMPP sendiri merupakan singkatan dari X (empat sistem operasi apapun), *Apache*, MySQL, PHP dan Perl. *MySQL* memiliki *tool* atau aplikasi yang disebut *PhpMyAdmin*. Melalui *PhpMyAdmin*, user melakukan perintah *query* tanpa harus mengetikkan seperti pada MS DOS. Untuk mendapatkanya XAMPP anda dapat mengunduh langsung dari web resminya. Pada penelitian ini xampp akan digunakan sebagai web server [22].