QUERY BAHASA INDONESIA PADA BASISDATA SOAL UJIAN DI FAKULTAS TEKNOLOGI INFORMASI UNIVERSITAS STIKUBANK DENGAN FORMAT DATA XML

Sariyun Naja Anwar, B.Sc, M.MSi
Dr. Drs. Yohanes Suhari, M.MSi
Setyawan Wibisono, S.Kom, M.Cs

Abstract *— Aplikasi dengan berbasis Natural Language Processing ini berfungsi merancang sebuah sistem pengakses basisdata. Metoda akses menggunakan query bahasa Indonesi dari 7 format pertanyaan yang telah ditetapkan untuk membantu pekerjaan pencarian data soal ujian dalam format XML. Cara pencarian data dalam aplikasi ini dilakukan dengan memberikan input berupa sebuah perintah dalam bahasa Indonesia. Pengguna memasukkan kalimat tanya atau kalimat perintah dengan format yang sudah ditetapkan. Aplikasi ini akan memberikan hasil dalam bentuk tabel kepada user. Jika format kalimat tidak sesuai dengan aturan produksi maka yang dimunculkan oleh aplikasi ini adalah sebuah peringatan kesalahan. Sedangkan jika input kalimat yang diberikan sudah sesuai dengan format aturan, tetapi data tidak tersedia maka aplikasi ini akan memunculkan sebuah tabel kosong.*

Keywords—natural language processing, query bahasa Indonesia, soal ujian, SQL, XML

1. LATAR BELAKANG

Untuk pencarian informasi pada sebuah basisdata, selain diperlukan pemahaman dalam bahasa Inggris, juga diperlukan pemahaman yang memadai tentang struktur penulisan query dan pemahaman antar muka suatu sistem basisdata. Untuk itu diperlukan suatu sistem yang dapat digunakan untuk melakukan pencarian data dengan cara yang lebih sederhana, daripada menuliskan pernyataan query yang cenderung tidak dipahami oleh pengguna awam. Sistem yang dibutuhkan adalah sistem yang dapat memberikan jawaban atas pertanyaan yang diajukan oleh pengguna awam dengan memasukkan pertanyaan dalam bahasa alami yang dikuasai oleh pengguna.

Dalam sistem informasi akademik, khususnya pada pengelolaan data ujian di Fakultas Teknologi Informasi, operator sistem adalah pengguna yang tidak mempunyai latar belakang pengetahuan query yang cukup. Sehingga kebutuhan untuk mendapatkan informasi menggunakan sistem dengan interaksi bahasa alami yaitu bahasa Indonesia sangatlah diperlukan. Demikian juga dalam hal tampilan antarmuka, operator akan lebih mudah berinteraksi dengan sistem, jika tampilan antarmuka dalam bentuk grafis maupun dalam bentuk formulir.

Maka sistem yang tepat sebagai solusi adalah sistem Pemrosesan Bahasa Alami (*Natural Language Processing*) yaitu sistem yang mengijinkan pengguna memberikan pernyataan tentang informasi yang dibutuhkan dalam bahasa alami dan sistem tidak memberikan jawaban dalam bentuk satu dokumen penuh dimana pengguna harus menentukan sendiri mana yang tepat sebagai jawaban, tetapi jawaban yang diberikan oleh sistem adalah suatu kutipan teks singkat atau bahkan sebuah frase (Wibisono, 2010).

XML (*eXtensible Markup Language)* mendeskripsikan susunan informasi dan berfokus pada informasi itu sendiri. XML terutama dibutuhkan untuk menyusun dan menyajikan informasi dengan format yang tidak mengandung format standar, layaknya seperti *heading*, *paragraph*, *table* dan lain sebagainya. *File* XML berbentuk teks sehingga bila diperlukan dapat dibaca tanpa memerlukan bantuan perangkat lunak khusus. Hal ini memudahkan pengembangan aplikasi yang menggunakan XML untuk men – *debug* programnya. Kelebihan lain dari XML adalah kemampuan untuk mempertukarkan informasi dari satu sistem ke sistem lain yang berbeda platform. Berdasarkan latar belakang di atas, maka dalam penelitian ini akan dibangun aplikasi pemrosesan bahasa alami untuk query basisdata akademik dengan format data XML. Diharapkan sistem yang dibuat akan lebih memudahkan pencarian informasi akademik oleh pemakai (Hartati dan Zuliarso, 2008).

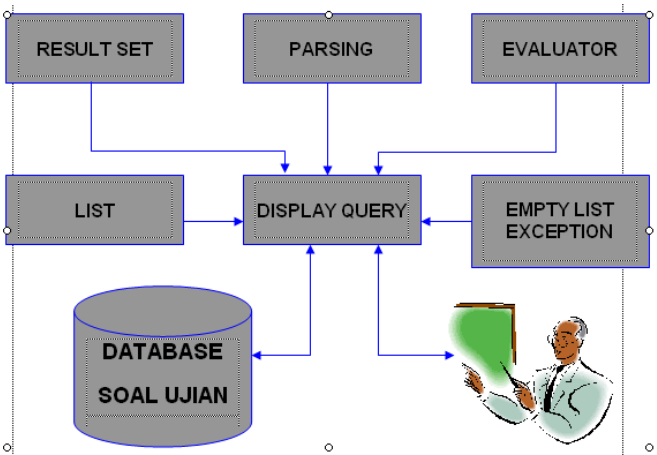
2. CARA PENELITIAN

Dalam perancangan aplikasi ini, langkah yang dilakukan adalah merancang basisdata XML dengan berdasar aturan yang diperoleh dari studi pustaka tentang basisdata XML. Kemudian melakukan pengamatan obyek penelitian, dalam hal ini merancang dan membuat soal ujian di Fakultas Teknologi Informasi dalam format XML. Dilanjutkan dengan merancang dan mengimplementasikan aplikasi query bahasa indonesia pada basisdata soal ujian di Fakultas Teknologi Informasi Universitas Stikubank dengan format data XML

3. PERANCANGAN

3.1. Deskripsi Sistem

Program aplikasi query Bahasa Indonesia pada basisdata soal ujian di fakultas teknologi informasi universitas stikubank dengan format data xml adalah program aplikasi digunakan oleh dosen, instruktur dan staf administrasi akademik untuk membantu pekerjaan pencarian data dalam format XML yang berkaitan dengan data soal ujian. Aplikasi ini dapat dipergunakan sebagai alat bantu dalam pencarian data dengan domain soal ujian. Cara pencarian data dalam aplikasi ini dilakukan dengan memberikan input berupa sebuah perintah dalam bahasa Indonesia.



Gambar 1. Blok diagram aplikasi pengelolan Query Bahasa Indonesia pada basisdata soal ujian dengan format XML

User memasukkan kalimat tanya atau kalimat perintah dengan format yang sudah ditetapkan. Format kalimat yang sudah ditetapkan disebut sebagai sebuah aturan produksi. Dalam aplikasi ini diberikan beberapa contoh kalimat yang sesuai dengan format aturan produksi. Aplikasi ini akan memberikan hasil dalam bentuk tabel kepada user, jika input kalimat yang diberikan sudah benar dan data tersedia sesuai permintaan. Jika format kalimat tidak sesuai dengan aturan produksi maka yang dimunculkan oleh aplikasi ini adalah sebuah peringatan kesalahan. Sedangkan jika input kalimat yang diberikan sudah sesuai dengan format aturan, tetapi data tidak tersedia maka aplikasi ini akan memunculkan sebuah tabel kosong. Prinsip kerja dari aplikasi ini adalah :

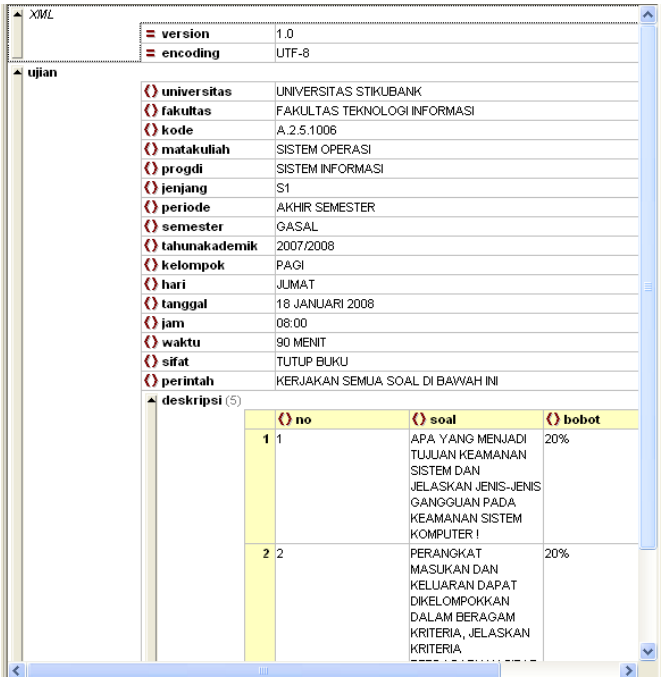
1. Menerima input kalimat.
2. Memisahkan input kalimat menjadi kata.
3. Dari tiap kata yang muncul, akan dibandingkan dengan kata – kata yang sudah terdaftar dalam program.
4. Dari hasil pembandingan akan didapat kelompok kata yang termasuk kata yang bermakna (token) dan kelompok kata yang tidak bermakna.
5. Kata yang tidak bermakna akan diabaikan, sedangkan kata yang bermakna akan dibandingkan dengan daftar kata yang termasuk atribut. Hal ini bertujuan untuk mendapatkan field yang sesuai dengan maksudk kalimat.
6. Selain dibandingkan dengan daftar atribut, kata – kata yang bermakna juga dibandingkan dengan daftar kata pelengkap yang berisi daftar kata - kata yang digunakan sebagai operator seperti kata – kata sama dengan, kurang dari, lebih dari dan lain – lain.
7. Kata – kata yang termasuk kondisi dari operator akan didapatkan dari hasil perbandingan kata – kata alami dengan daftar kata kondisi yang berisi kata - kata keterangan kondisi seperti kata, huruf, karakter dan lain-lain.
8. Dari kumpulan kata – kata yang termasuk kata – kata penting, akan dikonstruksikan sesuai dengan aturan produksi pembentuk kalimat. Jika sudah sesuai dengan aturan produksi, maka akan diterjemahkan dalam XQuery untuk mengakses basisdata XML, sehingga menghasilkan keluaran field yang sesuai dengan pertanyaan atau perintah masukan.

3.2. Struktur Data

Dalam penyimpanan data dengan format XML, maka data soal ujian dibuat dalam XML schema. Pembuatan XML Schema dimaksudkan untuk memastikan bahwa elemen – elemen dan atribut – atribut yang dimasukkan ke dalam dokumen sudah memenuhi aturan yang diterapkan dalam skema. XML Schema Tabel_Ujian untuk data yang disimpan adalah sebagai berikut :

1. universitas
2. fakultas
3. kode
4. mata kuliah
5. progdi
6. jenjang
7. periode
8. semester
9. tahun akademi
10. kelompok
11. hari
12. tanggal
13. jam
14. waktu
15. sifat
16. perintah
17. deskripsi :
    a. no
    b. soal
    c. bobot

Penggambaran data dengan format XML pada XML Grid akan terlihat seperti gambar 2. Field yang dimiliki mahasiswa akan diperlihatkan sebagai elemen utama, sedangkan field yang dimiliki oleh kuliah diperlihatkan sebagai elemen dengan susunan yang bersarang.



Gambar 2. XML Grid

4. IMPLEMENTASI

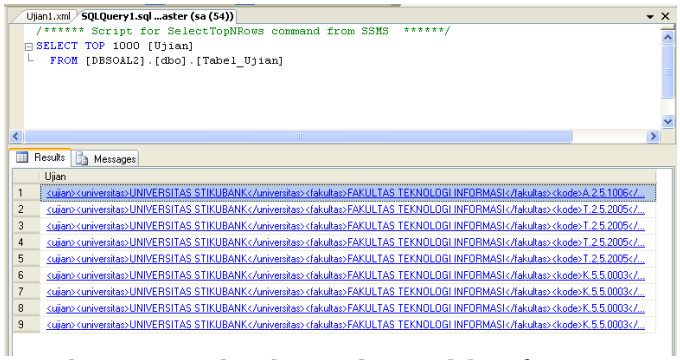
4.1. Basisdata XML

Basisdata yang digunakan dalam aplikasi ini adalah basisdata soal ujian dalam format XML. Data yang digunakan adalah data soal ujian dari beberapa periode ujian. Data diambil secara sampling secara acak. Tabel tersebut ditampilkan dalam struktur data seperti ditampilkan dalam gambar 3.

```
<ujian>
  <universitas>UNIVERSITAS
STIKUBANK</universitas>
  <fakultas>FAKULTAS TEKNOLOGI
INFORMASI</fakultas>
  <kode>A.2.5.1006</kode>
  <matakuliah>SISTEM OPERASI</matakuliah>
  <progdi>SISTEM INFORMASI</progdi>
  <jenjang>S1</jenjang>
  <periode>AKHIR SEMESTER</periode>
  <semester>GASAL</semester>
  <tahunakademik>2007/2008</tahunakademik>
  <kelompok>PAGI</kelompok>
  <hari>JUMAT</hari>
  <tanggal>18 JANUARI 2008</tanggal>
  <jam>08:00</jam>
  <waktu>90 MENIT</waktu>
  <sifat>TUTUP BUKU</sifat>
  <perintah>KERJAKAN SEMUA SOAL DI BAWAH
INI</perintah>
  <deskripsi>
    <no>1</no>
    <soal>APA YANG MENJADI TUJUAN KEAMANAN
```

Gambar 3. Struktur data soal ujian dalam format XML

Pembuatan data soal ujian diimplementasikan menggunakan SQL Server 2008 dalam format XML. Gambar 4 adalah tampilan data soal ujian di lingkungan Fakultas Teknologi Informasi, Universitas Stikubank dalam format XML.



Gambar 4. Tampilan data soal ujian dalam format XML

4.2. Antarmuka Display Query

Pada subprogram ini ditentukan koneksi aplikasi dengan database dan tabel yang digunakan dalam aplikasi ini. Database yang digunakan adalah ”SOAL2” dan tabel yang digunakan adalah ”Tabel_Ujian”, dengan username ”sa” dan password ”aliya”, seperti terlihat pada gambar 5.

```
public class DisplayQueryNlp extends JFrame
{
    static final String JDBC_DRIVER  =
    "sun.jdbc.odbc.JdbcOdbcDriver";
    static final String DATABASE_URL   =  "jdbc:odbc:SOAL2";
    static final String USERNAME       = "sa";
    static final String PASSWORD       = "aliya";
```

Gambar 5. Koneksi database

Ketika aplikasi dijalankan akan ditampilkan sebuah default query bahasa Indonesia yang sesuai dengan format kalimat. Dalam default query ini ditampilkan sebuah perintah dalam Bahasa Indonesia untuk menampilkan daftar mata kuliah seperti terlihat pada gambar 6.

```
static final String DEFAULT_QUERY  = "select
  akademik.value('/mahasiswa[1]/nama[1]','varchar(200)') as
  Nama_Mahasiswa from tabel_akademik2";
private ResultSetTableModel tableModel;
private JTextArea queryArea;
public DisplayQueryNlp()
{
  super ( "QUERY BAHASA INDONESIA" );
  try
  {
    tableModel = new ResultSetTableModel( JDBC_DRIVER,
    DATABASE_URL, USERNAME, PASSWORD,DEFAULT_QUERY );
    queryArea = new JTextArea( "Tampilkan mata kuliah",3, 100 );
  }
.......
}
```

Gambar 6. Default query bahasa Indonesia

Jika format query sudah benar, tetapi data yang dikehendaki tidak tersedia, maka akan ditampilkan tabel kosong. Begitu juga jika terdapat kesalahan dalam penulisan format query, maka akan ditampilkan pesan kesalahan, seperti terlihat pada gambar 7.

```
catch ( SQLException sqlException )
{
  JOptionPane.showMessageDialog(null,sqlException.getMessage(),
  "Database error Query Tidak berhasil",
  JOptionPane.ERROR_MESSAGE );
  try
  {
    ableModel.setQuery( DEFAULT_QUERY );
    queryArea.setText( DEFAULT_QUERY );
  }
  catch ( SQLException sqlException2 )
  {
    JOptionPane.showMessageDialog(null, sqlException2.
    getMessage(),Database error",JOptionPane.ERROR_MESSAGE );
    tableModel.disconnectFromDatabase();
    System.exit( 1 );
  }
}
```

Gambar 7. Pesan kesalahan

4.3. Antarmuka Result Set

Program Antarmuka ResultSetTableModel ini digunakan untuk melakukan koneksi antara program Java dengan SQL Server 2008 melalui JDBC. Selanjutnya melakukan eksekusi perintah XQuery dan menerima hasil record. Hasil record ditampilkan dalam bentuk tabel. Jika koneksi tidak berhasil maka muncul suatu peringatan "Not Connected to Database ", seperti terlihat pada gambar 8.

```
public ResultSetTableModel( String driver, String url,String
username, String password, String query )throws SQLException,
ClassNotFoundException
{
     Class.forName( driver );
     connection = DriverManager.getConnection( url, username,
     password );
     statement = connection.createStatement(
     ResultSet.TYPE_SCROLL_INSENSITIVE,
     ResultSet.CONCUR_READ_ONLY );
     connectedToDatabase = true;
     setQuery( query );
     public Class getColumnClass( int column ) throws
     IllegalStateException
     {
       if ( !connectedToDatabase )
       throw new IllegalStateException( "Not Connected to
       Database" );
       .............
     }
}
```

Gambar 8. ResultSetTableModel

4.4. Scanner

Dalam implementasi sistem ini, scanner yang digunakan adalah utilitas scanner yang ada dalam library Java. Untuk menggunakannya cukup dengan memanggilnya, dengan proses pemanggilan diimplementasikan seperti terlihat pada gambar 9.

```
import java.util.Scanner;
```

Gambar 9. Penggunaan utility scanner

4.5. Parsing

Analisis sintaks dilakukan berdasarkan aturan produksi yang telah ditentukan dan akan menghasilkan pohon sintaks (pohon parser). Seperti terlihat pada gambar 10. proses pengubahan pohon sintaks hasil parsing ke dalam tipe – tipe query diimplementasikan dalam bagian program ini.

```
// Variabel bersama querysatu dan querydua
 String fdan         = "dan";
 String fatau        = "atau";
 String fkomma       = ",";
 String sambung[]    = {"dan","," };
 StringBuffer strbff = new StringBuffer("");
 String ekorkon      = " ";
 String ekorkonp     = " "; //ekor query mahasiswa
 String utama[]      =
  {"universitas","fakultas","kode","matakuliah","progdi","jenj
  ang","periode","semester","tahunakademik","kelompok","hari",
  "tanggal","jam","waktu","sifat","perintah"};
 String deskripsi[]  = {"no","soal","bobot"};

 SortedSet<String> treeutama = new TreeSet< String >
   (Arrays.asList( utama ) );
 SortedSet< String > treedeskripsi = new TreeSet< String >
   (Arrays.asList( deskripsi ) );
 SortedSet< String > treesambung = new TreeSet< String >
   (Arrays.asList( sambung ) );

 // Variabel Khusus querydua
 String fkurang      = "kurang";
 String flebih       = "lebih";
 String suka         = "like"; //jika ada kata bukan/tidak
 String sukat        = "=";
 Boolean statkon     = false;
 String tolak[]       = {"bukan","tidak"};
 String danatau[]    = {"atau","dan"};
 SortedSet< String > treetolak = new TreeSet< String >
   ( Arrays.asList( tolak ) );
 SortedSet< String > treedanatau = new TreeSet< String >
```

Gambar 10. Deklarasi parser

Dalam perulangan, pengujian adanya token didefinisikan dengan regex dengan memanggil metode hasNext() dengan pola had sebagai argumen, seperti terlihat pada gambar 11. Bagian ini untuk menampung kata – kata yang nantinya dikelompokkan sebagai kata – kata yang tidak bermakna (bukan token).

```
String regex = "|dimana|apa|saja|tampilkan|tampil|cari|
                daftar|mengandung|di|pada|dari|yang|oleh|
                dengan|seluruh|disimpan";
Pattern had     = Pattern.compile(regex);
Scanner strScan = new Scanner(str);
int hadCount    = 0;
 while(strScan.hasNext())
 {
   if(strScan.hasNext(had))
   {
     ++hadCount;
     strScan.next();
   }
   else
   {
      ...
   }
 }
```

Gambar 11. Pengelompokan kata bukan token

Sedangkan bagian yang digunakan untuk menampung kata – kata yang nantinya dikelompokkan sebagai kata – kata yang bermakna (token), diperlihatkan, seperti terlihat pada gambar 12.

```
String regexlist = "dan|atau|,|universitas|fakultas|kode|
                   matakuliah|progdi|jenjang|periode|semester|
                   tahunakademik|kelompok|hari|tanggal|
                   jam|waktu|sifat|perintah|deskripsi|no|
                   soal|bobot";

Pattern hadlist = Pattern.compile(regexlist);
Scanner strScantok = new Scanner(sbfbantu.toString());
hadCount=0;
while(strScantok.hasNext() )
{
 if(strScantok.hasNext(hadlist))
 {
  ++hadCount;
  bantu=strScantok.next();
  list.insertAtBack(bantu);
 }
 else
 {
  ...
```

Gambar 12. Pengelompokan token

Dalam kalimat input dimungkinkan untuk mengakses data yang memiliki nama atribut yang terdiri dari dua kata yang dirangkai. Maka dibutuhkan suatu aturan, seperti terlihat pada gambar 13. yang dapat memastikan bahwa jika pengguna memberikan input dengan satu atribut, tetapi menuliskannya dengan menggunakan dua kata yang dipisahkan dengan spasi, maka input tersebut dapat diterima dengan benar.

```
String names[] = { "mata","matakuliah",
                   "tahun","tahunakademik" };
SortedSet< String > treex =  new TreeSet< String >
  ( Arrays.asList( names ) );
```

Gambar 13. Pengelompokan atribut yang terdiri dari 2 kata

Dalam mengimplementasikan parsing dilakukan dengan menggunakan teknik perulangan dan struktur if. Simpul yang ada di linked list akan diambil satu persatu dari depan. Seperti terlihat pada gambar 14. simpul ini dikemudian dievaluasi untuk menentukan apakah berupa atribut, operator ataukah data. Berikut ini contoh penguraian linked list dan penterjemahan menjadi bagian dari query dalam XQuery.

```
 String utama[]       = {"universitas","fakultas","kode",
                         "matakuliah","progdi","jenjang",
                         "periode","semester","tahunakademik",
                         "kelompok","hari","tanggal","jam",
                         "waktu","sifat","perintah"};
 String deskripsi[]   = {"no","soal","bobot"};
   SortedSet< String > treeutama = new TreeSet< String >
  ( Arrays.asList( utama ) );
 SortedSet< String > treedeskripsi = new TreeSet< String >
  ( Arrays.asList( deskripsi ) );
 SortedSet< String > treesambung = new TreeSet< String >
  ( Arrays.asList( sambung ) );
```

Gambar 14. Linked list

4.5. Translator dan Evaluator

Pohon sintaks yang dihasilkan dari proses parsing kemudian diterima translator. Translator berfungsi menghasilkan kode antara, sebelum memasuki proses evaluasi untuk menentukan jawaban akhir query. Komponen evaluator berfungsi mengevaluasi daftar *goal* dalam proses penentuan jawaban query, sehingga evaluasi banyak terlibat dengan entri – entri basisdata, seperti terlihat pada gambar 15.

```
bantu=fratribut.removeFromFront();
if (treeutama.contains(bantu))
{
 strbf.append(" Ujian.value('/ujian[1]/")
 .append(bantu)
 .append("[1]','varchar(200)') as ")
 .append(bantu);
 continue;
}
```

Gambar 15. Translasi atribut utama

Baris pertama akan mengambil satu simpul terdepan dari linked list token untuk dimasukkan ke dalam variabel bantu. Baris kedua akan menguji apakah himpunan treeutama mengandung string yang dimuat dalam variabel bantu. Jika ya, maka kemudian dilakukan penterjemahan ke dalam fragmen sintaks XQuery.

Untuk XML dengan data bersarang, proses pengubahan pohon sintaks hasil parsing ke dalam tipe – tipe query diimplementasikan dalam bagian program seperti terlihat pada gambar 16. Setelah dilakukan penguraian berdasarkan frase atribut dan frase kondisi. Selanjutnya dilakukan penguraian berdasarkan masing – masing frase.

```
if (treedeskripsi.contains(bantu))
{
 strbf.append("a.value('");
 strbf.append(bantu)
 .append("[1]','varchar(200)') as ")
 .append(bantu);
 ekorkonp="CROSS APPLY Ujian.nodes('/ujian/deskripsi') as
 result(a)";
 continue;
}
```

Right-click to display spelling suggestions

Gambar 16. Translasi atribut bersarang

Dalam tipe input kalimat dimana terdapat kalimat tanya atau kalimat perintah yang berisi lebih dari satu atribut dan untuk memisahkan atribut – atribut itu digunakan tanda koma (,) atau kata “dan”, maka akan diterjemahkan sebagai tanda koma (,) seperti terlihat pada gambar 17.

```
if ((fdan.equals(bantu)==true)||(fkomma.equals(",")==true))
{
 strbf.append(" , ");
}
```

Gambar 17. Translasi kata sambung

Proses berikutnya adalah melakukan pengecekan terhadap ekor atribut, apakah ekor merupakan sebuah kondisi atau ekor merupakan string kosong. Jika ekor merupakan string kosong atau tidak ada kata di belakang ekor, maka akan langsung dilakukan pemanggilan tabel. Tetapi jika tidak, maka dilakukan pemanggilan tabel, dilanjutkan dengan pencocokan pola ekor dengan kondisi, seperti pada gambar 18.

```
if (ekorkonp.equals(" ") == true) //diubah
{
 strbf.append(" from Tabel_Ujian ");
}
else
{
 strbf.append(" from Tabel_Ujian ")
 .append(ekorkonp);
}
```

Gambar 18. Translasi penulisan tabel

Selanjutnya dicek apakah ada atribut tambahan. Atribut tambahan ini dapat berupa kondisi dengan sebuah persyaratan. Sedangkan persyaratan dapat berupa persyaratan dalam bentuk positif atau negatif. Persyaratan dalam bentuk negatif akan diawali dengan

kata “bukan” atau “selain”. Kata ini terletak pada ekor kondisi. Kata yang menyatakan bukan atau selain akan dicek apakah termasuk dalam daftar kata tolak. Jika termasuk dalam daftar kata tolak, maka akan diterjemahkan sebagai ***not like***, dan ditambahkan di belakang pernyataan terjemahan SQL sebelumnya, serta ditambahkan tanda “ ***%*** ”, seperti terlihat pada gambar 19.

```
if (treetolak.contains(bantu))
{
  suka=" not like ";sukat=" != ";
  strbf.append(suka)
  .append("'%");
  bantu=frakondisi.removeFromFront();
}
else
{
  if (fdan.equals(bantu)!=true)
  {
    suka=" like ";sukat=" = ";
    strbf.append(suka)
    .append("'%");
  }
}
```

Gambar 19. Translasi operator bukan

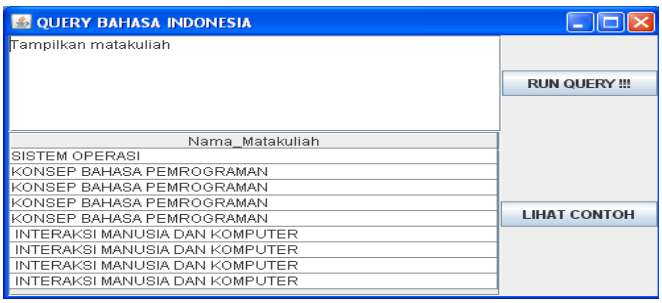
Atribut kondisi juga dapat berisi atribut kondisi yang bersifat perbandingan. Atribut kondisi ini bertujuan untuk mencari data yang berupa angka. Kondisi ini berupa kata “lebih”, “kurang”, “sesudah”, “sebelum” dan dideklarasikan dalam ***fbel*** dan ***fdah.*** Jika termasuk dalam ***fbel*** maka akan diterjemahkan dalam “ < “, sedangkan jika termasuk dalam ***fdah***, maka akan diterjemahkan dalam “ > “. Tetapi jika atribut diikuti kondisi secara langsung, maka akan diterjemahkan dalam “ = “, seperti terlihat pada gambar 20.

```
if (fbel.equals(bantu)== true) //sebelum tahun
{
  bantu=frakondisi.removeFromFront();
  strbf.append(" < ")
  .append(bantu);
}
else
{
  if (fdah.equals(bantu)==true)
  {
    bantu=frakondisi.removeFromFront();
    strbf.append(">")
    .append(bantu);
  }
  else
  {
    strbf.append(sukat)
    .append(bantu);
  }
}
```

Gambar 20. Translasi operator perbandingan

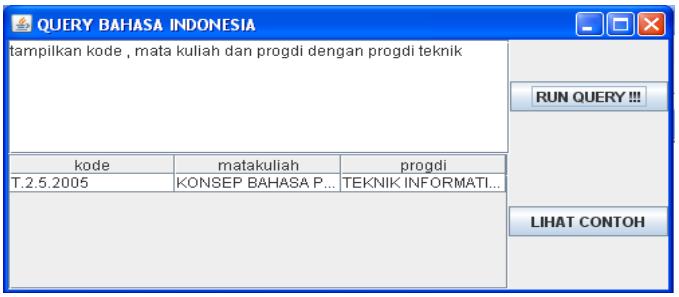
5. Tampilan Aplikasi

5.1. Tampilan Tipe q_a (query – atribut)

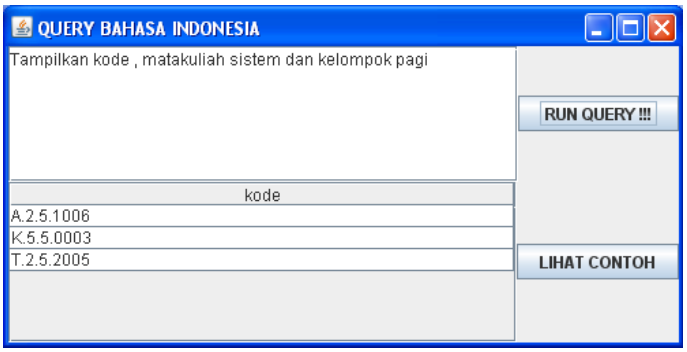


5.2. Tampilan Tipe q_a_a (query – atribut – atribut)

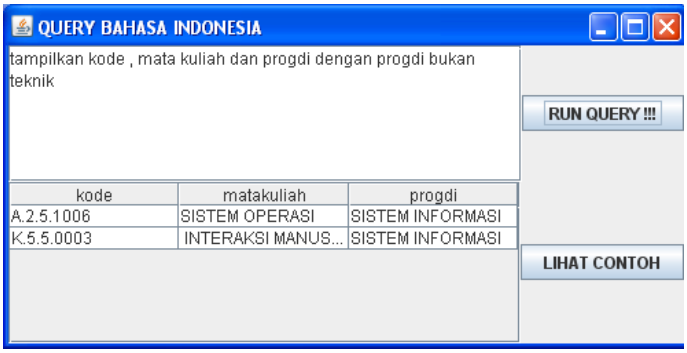
5.3. Tampilan Tipe q_a_opr (query – atribut – operator)



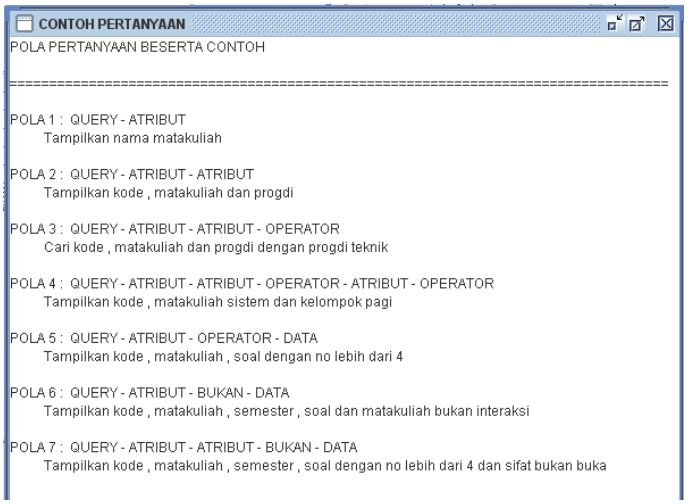
5.4. Tampilan Tipe q_a_a_opr (query – atribut – atribut – operator – atribut – operator)



5.5. Tampilan Tipe q_a_bukan (query – atribut – bukan – data)



5.6. Tampilan contoh-contoh format kalimat input



5. KESIMPULAN

Berdasarkan hasil penelitian ini,maka dapat disimpulkan beberapa hal sebagai berikut:

1. Aplikasi ini mampu memberikan informasi berkaitan soal ujian dari sebuah permintaan. Permintaan dituliskan dalam bahasa Indonesia.
2. Terdapat 7 format perintah/pertanyaan yang yang mampu dijawab oleh aplikasi ini.
3. Kesalahan input dapat disebabkan oleh kekeliruan format kalimat perintah/pertanyaa. Jika terdapat kesalahan dalam memberikan masukan query, maka aplikasi ini mampu memberikan. Peringatan yang ditampilkan adalah peringatan yang berkaitan dengan kesalahan query. Tetapi aplikasi ini belum mampu memberikan koreksi otomatis sebagai alternatif perbaikan kesalahan.
4. Aplikasi ini mampu memberikan keluaran berupa tabel yang berisi data. aplikasi ini juga mampu menampilkan XQuery hasil translasi dari kalimat masukan, dalam hal ini berupa pernyataan SQL dalam format XML.

DAFTAR PUSTAKA

Djuandi, F., 2008, *Jurus Baru Pemrograman SQL Server 2005*, Elex Media Komputindo, Jakarta.

Hartati, S., dan Zuliarso E., 2008, *Aplikasi Pengolah Bahasa Alami untuk Query Basisdata XML*, Dinamik, Vol XIII, No 2, Juli 2008, 168 – 175.

Junaedi, M., 2003, *Pengantar XML*, *www.Ilmukomputer.com/umum/junaedi-xml.php*, diakses tanggal : 9 Februari 2009

Liddy, E.D., 2001, *Natural Language Processing*, *In Encyclopedia of Library and Information Science*, 2nd Edition, Marcel Decker Inc, NY, USA.

Riskadewi dan Karya, G., 2004, *Representasi dan Sinkronisasi Basis Data Relasional dengan Dokumen XML*, INTEGRAL, Vol. 9 No. 1, Maret 2004

Utama, G., 2002, *Berfikir Objek: Cara Efektif Menguasai Java*, www.ilmukomputer.org, diakses tanggal : 4 Februari 2011

www.cnlp.org/publications/03nlp.lis.encyclopedia.pdf

www.w3schools.org.