BAB IV
IMPLEMENTASI

Bab ini akan membahas mengenai tahapan dari proses implementasi sistem berdasarkan hasil analisa kebutuhan dan proses perancangan perangkat lunak yang telah dibuat sebelumnya. Pembahasan bab ini dapat terdiri dari penjelasan tentang spesifikasi sistem, batasan-batasan dalam implementasi, implemantasi algoritma, implementasi antarmuka, dan implementasi basis data.

4.1. Spesifikasi Sistem

Sistem penggalian pola sekuensial dikembangkan pada lingkungan implementasi yang terdiri dari perangkat keras dan perangkat lunak

4.1.1. Implementasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan dalam implementasi aplikasi penggalian pola sekuensial dapat dilihat pada tabel 4.1

Tabel 4. 1 Spesifikasi Perangkat Keras

Nama Komponen	**Spesifikasi**
Pocessor	Intel(R) Core (TM) i5-3230M CPU @ 2.60GHz
Memori (RAM)	4.00 GB
Harddisk	1 TB

4.1.2. Implementasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan dalam implementasi aplikasi penggalian pola sekuensial dapat dilihat pada tabel 4.2

Tabel 4. 2 Spesifikasi Perangkat Lunak

Nama	**Spesifikasi**
Sistem Operasi	Microsoft Windows 8 Pro 64-bit
Bahasa Pemrograman	C#
Tools pemrograman	Microsoft Visual Studio 2012
Database Management System	MySQL versi 5.1.41

4.2. Batasan - Batasan Implementasi

Batasan – batasan dalam mengimplementasikan sistem penggalian pola sekuensial pada data akses pengguna *website* adalah sebagai berikut:

1. Sistem penggalian pola sekuensial pada data akses pengguna *website* ini dirancang dan dijalankan mengunakan *desktop application*.
2. Format log yang digunakan adalah *combined log format* dari *web server* Apache.
3. Data yang digunakan untuk penelitian ini diperoleh dari data akses pengguna yang disimpan dalam file *web access log* PTIIK Universitas Brawijaya dan dibatasi pada direktori content/read, kknp,monita, dan news.
4. Teknik data mining yang digunakan dalam menyelesaikan masalah adalah *Web Usage Mining* (WUM) dengan algoritma Pola sekuensial yang digunakan adalah Algoritma *PrefixSpan*.
5. Output yang dihasilkan oleh sistem adalah pola sekuensial dari pengguna dalam mengakses *website* PTIIK - UB

4.3. Implementasi Program

Sistem penggalian pola sekuensial ini terdiri dari beberapa proses yaitu proses *preprocessing*, penemuan pola dengan algoritma *PrefixSpan*, dan analisis pola.

4.3.1 Implementasi Prepocessing

Proses *preprocessing* yang dilakukan terdiri dari tiga tahap yaitu pembersihan data, identifikasi pengguna, dan identifikasi sesi. Masukan dari proses ini adalah file *web access log*. Hasil dari proses *preprocessing* ini adalah idpengguna dan halaman yang diakses pengguna yang telah dibagi berdasarkan sesinya.

4.3.1.1 Pembersihan data

Pada proses ini file log akan dipecah berdasarkan baris. Kemudian setiap baris akan dilakukan pengecekan terhadap halaman akes / URL. Jika URL tersebut memiliki akhiran dengan format gambar, video, dan file CSS, file JS, Swf, zip, php, xml maka baris log tersebut akan dihapus.

Pada gambar 4.1 merupakan *stored procedure* yang berfungsi untuk proses penghapusan baris-baris log yang tidak digunakan. Baris 6-25 adalah proses

penghapusan baris log yang mempunyai URL dengan akhiran nama file seperti “.gif”, “.jpeg”, “.jpg”, “.bmp”, “.png”, “.tiff”, “.mp4”, ”.avi”, ”.mkv”, “.pdf”,”.doc”, ”.docx”,”.txt”,”.xls”,’’.xml’’ ,’’.php’’. Penghapusan selanjutnya yaitu baris 26-27 akan dilakukan terhadap baris yang mencatat permintaan yang dilakukan robots (*crawler*) dari suatu mesin pencarian. Baris 28-29 menunjukkan penghapusan baris yang memiliki kode status HTTP yang tidak sama dengan 200 dan baris yang memiliki aksi selain GET.

```
1   /*------ potongan stored procedure untuk menghapus baris yang tidak
2   diperlukan -------*/
3
4   ...
5    SET @statement = CONCAT ('DELETE FROM seleksidata WHERE HalamanAkses
6    LIKE ''%.png%''
7          OR HalamanAkses LIKE ''%.jpg%''
8           OR HalamanAkses LIKE ''%.gif%''
9          OR HalamanAkses LIKE ''%.ico%''
10         OR HalamanAkses LIKE ''%.js%''
11         OR HalamanAkses LIKE ''%.pdf%''
12         OR HalamanAkses LIKE ''%.docx%''
13         OR HalamanAkses LIKE ''%.doc%''
14         OR HalamanAkses LIKE ''%.mp4%''
15         OR HalamanAkses LIKE ''%.mkv%''
16         OR HalamanAkses LIKE ''%.avi%''
17         OR HalamanAkses LIKE ''%.css%''
18         OR HalamanAkses LIKE ''%.txt%''
19         OR HalamanAkses LIKE ''%.xls%''
20         OR HalamanAkses LIKE ''%.zip%''
21         OR HalamanAkses LIKE ''%.woff%''
22         OR HalamanAkses LIKE ''%.swf%''
23         OR HalamanAkses LIKE ''%.php%''
24         OR HalamanAkses LIKE ''%.xml%''
25         OR HalamanAkses LIKE ''%feed%''
26         OR UserAgent LIKE ''%bot%''
27          OR Kode_HTTP!=200
28         OR Get!=''GET'''
29         OR HalamanAkses LIKE ''/lab%''
30         OR HalamanAkses LIKE ''%niceub%''
31         OR HalamanAkses LIKE ''%auth/%''
32         OR HalamanAkses LIKE ''%pkm%''
33         OR HalamanAkses LIKE ''wp-%''
34  );
    ...
```

Gambar 4. 1 Potongan Sourcecode Proses Pembersihan Data

Pada proses implementasi terdapat beberapa penambahan baris yang dihapus untuk mendapatkan data yang bersih, proses ini dapat dilihat pada baris 30-34. Dengan adanya penambahan proses ini diharapkan halaman-halaman yang digunakan untuk proses selanjutnya adalah halaman-halaman utama dari direktori news, content/post, content/read, monita dan kknp. Pada halaman laboratorium dilakukan penghapusan karena halaman ini masih dalam bentuk id, sehingga sulit untuk membedakan halaman utama dari laboratorium dan halaman pengumuman yang ada di laboratorium. *Website* PTIIK berbasis wordpress sehingga akan

ditemukan halaman dengan *prefix* wp- merupakan halaman admin dari wordpress. Baris yang mengandung halaman ini tidak digunakan, karena halaman ini tidak diakses oleh pengguna, halaman hanya diakses oleh adimnistrator.

4.3.1.2 Identifikasi Pengguna

Identifikasi pengguna digunakan untuk mengidentifikasi siapa yang mengakses *website* dan halaman apa yang diakses. Tidak semua data yang sudah dipecah dan dibersihkan digunakan, pada tahap ini akan dipilih beberapa atribut yang digunakan sebagai inputan identifikasi pengguna. Atribut-atribut yang digunakan adalah alamat IP, waktu dan tanggal akses, halaman yang diminta pengguna dan *useragent*. Alamat IP dan *user agent* digunakan untuk identifikasi pengguna, kemudian waktu dan tanggal akses digunakan untuk identifikasi sesi, dan halaman yang diminta untuk mendapatkan halaman-halaman yang diakses pengguna.

```
1   /*------ potongan stored procedure untuk membuat halaman akses menjadi
2   numerik -------*/
3   ...
4
5       DECLARE   cur1   CURSOR   FOR   SELECT   DISTINCT   HalamanAkses   FROM
6        seleksidata;
7
8        ...
9
10      SELECT COALESCE(MAX(idmapping),0)+1 INTO sesuatu FROM m_mapping;
11
12      read_loop : LOOP
13      FETCH cur1 INTO a;
14
15       IF done THEN
16       LEAVE read_loop;
17       END IF;
18
19       IF NOT EXISTS(SELECT * FROM m_mapping WHERE `contentmapping`= a)
20       THEN
21       INSERT    INTO    `m_mapping`    (idmapping,contentmapping)    VALUES
22       (sesuatu,a);
23       SET inc=inc+1;
24       SET sesuatu= sesuatu+1;
25       END IF;
26
27     END LOOP;
28     CLOSE cur1;
    ...
```

Gambar 4. 2 Potongan Sourcecode Proses Substitusi Halaman Akses Menjadi Numerik

Proses substitusi halaman akses menjadi angka numerik dapat dilihat pada gambar 4.2. baris 6 adalah deklarasi cursor cur1 yang digunakan untuk mengambil nilai halaman akses dari tabel seleksidata, nilai halaman akses ini

kemudian disimpan dalam variabel a pada baris 13. Baris 10 adalah statement untuk mengambil nilai maksimal dan menambahnya dengan nilai satu dari kolom idmapping dan disimpan dalam variabel sesuatu. Baris 10-25 adalah proses memasukkan nilai idmapping dan halaman akses ke dalam tabel m_mapping. Nilai dari Id mapping akan terus bertambah ketika ada halaman akses baru yang masuk. Setiap halaman akses akan memiliki satu id mapping.

```
1   /*------ potongan stored procedure untuk proses identifikasi pengguna -----
2   --*/
3   ...
4
5       DECLARE  cur1  CURSOR  FOR  SELECT  DISTINCT  IP,  UserAgent  FROM
6       seleksidata;
7   ...
8
9      SELECT COALESCE(MAX(iduser),0)+1 INTO sesuatu FROM m_user;
10
11      read_loop : LOOP
12      FETCH cur1 INTO a,b;
13
14       IF done THEN
15       LEAVE read_loop;
16       END IF;
17
18       IF NOT EXISTS(SELECT * FROM m_user WHERE `ip`=a AND `useragent` = b)
19       THEN
20       INSERT INTO `m_user` (iduser,ip, useragent) VALUES (sesuatu,a,b);
21       SET inc=inc+1;
22       SET sesuatu= sesuatu+1;
23       END IF;
24
25     END LOOP;
26     CLOSE cur1;
27
...
```

Gambar 4. 3 Potongan Sourcecode Proses Identifikasi Pengguna

Proses identifikasi pengguna dilakukan dengan cara IP yang berbeda menandakan pengguna yang berbeda. jika memiliki alamat IP sama, tetapi *user agent* (tipe dari browser dan sistem operasi) berbeda maka dianggap pengguna yang berbeda. Proses ini dapat dilihat pada potongan *stored procedure* yang ada pada gambar 4.3. Baris 6 adalah deklarasi cursor cur1 dan pengambilan nilai IP dan useragent dari tabel seleksidata, digunakan fungsi distinct untuk mendapatkan nilai yang berbeda untuk setiap barisnya. Alamat IP dan user agent ini kemudian disimpan dalam variabel a, b pada baris 12. Baris 9 adalah statement untuk mengambil nilai maksimal dan menambahnya dengan nilai satu dari kolom iduser dan disimpan dalam variabel sesuatu. Baris 10-25 adalah proses memasukkan nilai iduser, IP, dan useragent ke dalam tabel m_user. Nilai dari Iduser akan terus bertambah ketika ada IP dan useragent yang tidak terdapat dalam tabel m_user.

4.3.1.3 Identifikasi Sesi

Identifikasi dilakukan dengan cara memeriksa iduser yang ada dalam tabel ‘useraccess’, jika terdapat pengguna baru (iduser) maka terdapat sesi baru. Jika jarak antara dua akses pengguna yang sama melebihi 30 menit, maka dianggap terdapat sesi baru.

```
1   public void setsesi(string iduser)
2   {
3    ........
4       for (int i = 1; i < list1[0].Count; i++)
5       {
6          if (Convert.ToInt32(list1[1][i]) < (Convert.ToInt32(list1[1][i - 1]) +
7          1800))
8          {
9              List2[0].Add(list1[0][i]);
10             List2[1].Add(iduser);
11             List2[2].Add(temp);
12             List2[3].Add(list1[1][i]);
13             List2[4].Add(list1[2][i]);
14         }
15         else
16         {
17             x++;
18             temp = list1[0][i] + iduser + x;
19             List2[0].Add(list1[0][i]);
20             List2[1].Add(iduser);
21             List2[2].Add(temp);
22             List2[3].Add(list1[1][i]);
23             List2[4].Add(list1[2][i]);
24         }
25
26
27  ..................
28              }
29  }
```

Gambar 4. 4 Potongan Sourcecode Proses Identifikasi Sesi

Proses identifikasi dapat dilihat pada gambar 4.4 , proses identifikasi diawali dengan merubah tanggal dan waktu akses yang ada di basis data menjadi format unix timestamp. Pada baris 4-25 akan dibandingkan jarak waktu antara dua akses yang berurutan. Jika jarak waktu antara dua akses pengguna yang sama lebih dari 1800 detik /30 menit, maka pengguna tersebut dianggap memiliki sesi baru.

4.3.2. Implementasi Algoritma *PrefixSpan*

Proses algoritma *PrefixSpan* diawali dengan mencari length-1 yang akan dijadikan sebagai kandidat *prefix*. Untuk length-1 yang memiliki nilai *support* lebih besar dari minimum *support* akan dijadikan sebagai *prefix*. Selanjutnya *sequence database* akan dibagi berdasarkan *prefix*-nya (pembagian ruang pencarian) dan akan dicari *prefix* k-length. Keluaran dari proses algoritma *PrefixSpan* ini adalah himpunana pola sekuensial yang didiapatkan dari kumpulan *prefix* yang memenuhi minimum *support* (*frekuent itemset*).

```
1   /*------ potongan stored procedure untuk proses pencarian length-1 ------
2   -*/
3   CREATE DEFINER=`root`@`localhost` PROCEDURE `inssuffix1`(
4       IN inminsup DOUBLE,
5       IN initerasi INT
6       )
7   BEGIN
8   ......
9       DECLARE cur1 CURSOR FOR SELECT aa.idmapping,aa.support
10          FROM
11          (SELECT       a.idmapping,       COALESCE       (supportlength1
12          (inminsup,'','',a.idmapping,itera),0)  AS support
13          FROM
14          (SELECT DISTINCT idmapping FROM transaksi)a
15          )aa
16          WHERE aa.support !=0 GROUP BY idmapping;
17
18  .....
19
20          OPEN cur1;
21          read_loop : LOOP
22          FETCH cur1 INTO aprefix,bsupport;
23
24          IF done THEN
25          LEAVE read_loop;
26          END IF;
27
28          CALL `suffix1`(aprefix,itera);
29          END LOOP;
30          CLOSE cur1;
31  .......
32      END$$
```

Gambar 4. 5 Potongan Sourcecode Pencarian *Support Length*-1

Gambar 4.5 merupakan potongan dari *stored procedure* untuk pencarian *support* length 1, masukan dari *stored procedure* ini ada nilai dari minimum *support* dan nilai dari iterasi. Untuk nilai minimum *support* didapat dari inputan pengguna, sedangkan iterasi diberikan nilai iterasi = 1 pada pencarian support length-1. Baris 9-16 merupakan proses pencarian *prefix* length-1, proses ini dimulai dengan menghitung *support* dari semua item yang ada didalam *sequence database* dengan cara memanggil fungsi supportlengt1. Fungsi ini sekaligus

menyeleksi item-item yang memenuhi minimum *support*. Item length-1 yang memenuhi minimum *support* selanjutnya disebut sebagai *prefix* length-1. Tahap selanjutnya adalah pembagian ruang pencarian berdasarakan *prefix* untuk *suffix sequence*.

```
/*------ potongan stored procedure untuk proses pembentukan suffix sequence
pada iterasi 1-------*/

CREATE DEFINER=`root`@`localhost` PROCEDURE `suffix1`(
IN inprefix VARCHAR(10),
IN itera INT)

BEGIN
.....
    DECLARE  cur1  CURSOR  FOR  SELECT  a.idsesi,  cekmap(a.idsesi,inprefix)
cekpos FROM (SELECT DISTINCT idsesi FROM `transaksi`)a ;
...........

     IF done THEN
     LEAVE read_loop;
     END IF;
     IF (b!='kosong')
     THEN
     SELECT COALESCE(MAX(urutan),0) INTO itu FROM suffix;
     SELECT countfuncseq(c) INTO jum
jum_loop:LOOP
IF cnt>=jum
THEN LEAVE jum_loop;
END IF;
SET cnt=cnt+1;
SET isi=CONCAT('',isi,' union all select ',cnt,'');
END LOOP;

    SET        @statement        =CONCAT('INSERT        INTO        suffix
(urutan,idsesi,prefixawal,prefixbaru,iterasi,suffix)              SELECT
generate_urutan(),Z.idsesi,'''',Z.prefix,',itera,',SUBSTRING_INDEX(SUBSTRI
NG_INDEX(Z.suffix, '','', n.digit+1), '','', -1)
  FROM
  (SELECT A.idsesi,',inprefix,' AS prefix, ') ;
if b like ',%' and b not like '%,'
then  SET @statement =CONCAT(@statement, ' SUBSTR(A.map,length (A.map) +
LENGTH(substring(''',b,''',2))) AS suffix ' );
   else
   SET    @statement    =CONCAT(@statement,    '    SUBSTR(A.map,LOCATE(
''',b,''',A.map)+LENGTH(''',b,''')) AS suffix ' );
end if;
SET      @statement      =CONCAT(@statement,      '      FROM      (SELECT
a.idsesi,`funcseq`(a.idsesi)  map  FROM  (SELECT  DISTINCT  idsesi  FROM
`session` WHERE idsesi=',c,')a 0 A
WHERE A.map LIKE ',inprefix,' OR A.map LIKE ''',inprefix,',%'' OR A.map
LIKE ''%,',inprefix,',%'' OR A.map LIKE ''%,',inprefix,''') Z
  INNER JOIN(SELECT ',isi,') n
  ON LENGTH(REPLACE(Z.suffix, '','' , '''')) <= LENGTH(Z.suffix)-n.digit '
);
.................
     END IF;
     END LOOP;
    CLOSE cur1;
     END$$
..............
```

Gambar 4. 6 Potongan Sourcecode Pembagian Ruang Pencarian Iterasi 1

Gambar 4.6 adalah potongan dari *stored procedure* untuk proses pembentukan *Suffix sequence* iterasi 1. Masukan dari *stored procedure* ini adalah

prefix yang didapatkan dari proses gambar 4.5 dan juga iterasi. Nilai iterasi yang digunakan untuk *stored procedure* ini adalah 1. Cursor cur1 pada baris 10 digunakan untuk mencari posisi dari *prefix*, jika posisi dari *prefix* tidak sama dengan kosong maka dapat dilanjutkan dengan proses selanjutnya. Baris 29- 49 adalah proses untuk menghasilkan *Suffix sequence*. *Suffix sequence* didapatkan dari *sequence* yang berada setelah *prefix*. Hasil *prefix* dan *suffix sequence* kemudian akan disimpan di dalam tabel suffix.

```
1  /*------ potongan stored procedure untuk proses pencarian length-k pada
2  iterasi lebih dari 1 -------*/
3  CREATE DEFINER=`root`@`localhost` PROCEDURE `inssuffix2`(
4      IN inminsup DOUBLE,
5      IN initerasi INT
6      )
7  BEGIN
8  ........
9
10 DECLARE cur1 CURSOR FOR SELECT ab.prefixawal, ab.prefixbaru, ab.suffix,
11 ab.support
12 FROM
13   (SELECT a.prefixawal, a.prefixbaru, a.suffix, COALESCE( `supportlength1`
14   (inminsup,  a.prefixawal,a.prefixbaru,  a.suffix,  initerasi)  ,0)  AS
15   support FROM
16     (SELECT DISTINCT prefixawal,prefixbaru,suffix,iterasi FROM suffix) a
17     WHERE a.iterasi=(initerasi-1)) ab
18 WHERE  ab.support != 0 GROUP BY ab.prefixawal,ab.prefixbaru,ab.suffix;
19 .....
20
21 SELECT COUNT(*) INTO dcnt
22 FROM (
23 SELECT ab.prefixawal, ab.prefixbaru, ab.suffix, ab.support
24 FROM(
25 SELECT a.prefixawal, a.prefixbaru, a.suffix, COALESCE( `supportlength1`(
26 inminsup, prefixawal, prefixbaru, suffix,initerasi),0) AS support
27 FROM (SELECT DISTINCT prefixawal,prefixbaru,suffix,iterasi FROM suffix) a
28 WHERE a.iterasi=(initerasi-1)) ab
29 WHERE  ab.support != 0 GROUP BY ab.prefixawal,ab.prefixbaru,ab.suffix)b;
30
31
32   OPEN cur1;
33   read_loop : LOOP
34   FETCH cur1 INTO aprefixawal, aprefix,bsuffix,csupport;
35   IF done THEN
36   LEAVE read_loop;
37   END IF;
38         IF (dcnt > 0)
39         THEN
40         CALL `suffix1dinamis`(aprefixawal,aprefix,bsuffix,initerasi);
41         END IF;
42   END LOOP;
43   CLOSE cur1;
44   END$$
45
46 .....
```

Gambar 4. 7 Potongan *Source Code* Pencarian *Support Length*-k

Potongan *Stored procedure* pada gambar 4.7 digunakan untuk pencarian support length-k. Masukan dari stored procedure adalah minimum *support* yang dimasukkan oleh pengguna dan iterasi. Iterasi dimulai dari nilai 2 dan seterusnya

sampai iterasi selesai yaitu tidak ditemukan lagi item yang memenuhi minimum *support*.

Baris 10-18 merupakan proses pencarian *prefix* length-k, proses ini dimulai dengan menghitung support dari semua item yang ada didalam *suffix sequence* hasil proses iterasi sebelumnya. Proses menghitung *support* dilakukan dengan cara memanggil fungsi supportlengt1. Fungsi ini sekaligus menyeleksi item-item yang memenuhi minimum *support*. Item length-k yang memenuhi minimum *support* selanjutnya disebut sebagai *prefix* length-k. Baris 21-29 digunakan untuk menghitung berapa jumlah item yang masih memenuhi minimum *support*. ketika nilai dcnt sama dengan 0 yang artinya sudah tidak ada item yang memenuhi minimum *support* sehingga iterasi akan berhenti.

pembentukan *Suffix sequence* iterasi 2 dan selanjutnya dapat dilihat pada potongan dari *stored procedure* gambar 4.8. Masukan dari *stored procedure* ini adalah *prefix* awal dan *prefix* baru dari tabel suffix yang didapatkan dari proses gambar 4.6 dan juga iterasi. Cursor cur1 pada baris 12-14 digunakan untuk mencari posisi dari *prefix*, jika posisi dari *prefix* tidak sama dengan kosong maka dapat dilanjutkan dengan proses selanjutnya. Baris 40- 65 adalah proses untuk menghasilkan *Suffix sequence* dan juga *prefix*baru. *Prefix* baru = prefix lama, item(yang memenuhi minimum *support*). Selanjutnya akan dicari *suffix sequence* dari *prefix* baru tersebut. Proses pencarian *prefix* baru dan *suffix sequence* ini akan terus menerus diulang dan berhenti ketika sudah tidak ada item yang memenuhi minimum *support*. Semua *prefix* yang telah disimpan dalam tabel suffix adalah himpunan pola sekuensial yang merupakan keluaran dari algoritma *PrefixSpan*.

```
1  /*------ potongan stored procedure untuk proses pembentukan suffix sequence
2  pada iterasi 2 dan seterusnya-------*/
3  .........
4  CREATE DEFINER=`root`@`localhost` PROCEDURE `suffix1dinamis`(
5          IN inprefixawalawal VARCHAR(20),
6          IN inprefixawal VARCHAR(20),
7          IN inprefixbaru VARCHAR(10),
8          IN itera INT
9      )
10 BEGIN
11     ......
12     DECLARE cur1 CURSOR FOR SELECT a.idsesi,
13 cekmapsuffix(a.idsesi,inprefixawalawal,inprefixawal,inprefixbaru,itera)
14 cekpos FROM (SELECT DISTINCT idsesi FROM `suffix`)a ;
15     .......
16     OPEN cur1;
17     SET inc=1;
```

```
18      SET cnt=0;
19      SET isi='0 digit';
20      read_loop : LOOP
21      FETCH cur1 INTO c,b;
22
23       IF done THEN
24       LEAVE read_loop;
25       END IF;
26
27       IF (b!='kosong')
28       THEN
29       SELECT COALESCE(MAX(urutan),0) INTO itu FROM suffix;
30
31       SELECT countfunctseq(c) INTO jum;
32          jum_loop:LOOP
33          IF cnt>=jum
34          THEN LEAVE jum_loop;
35          END IF;
36          SET cnt=cnt+1;
37          SET isi=CONCAT('',isi,' union all select ',cnt,'');
38          END LOOP;
39
40    SET        @statement         =CONCAT('INSERT         INTO         suffix
41    (urutan,idsesi,prefixawal,prefixbaru,iterasi,suffix)            SELECT
42    generate_urutan(),Z.idsesi,Z.prefixawal,Z.prefixbaru,',itera,',
43     SUBSTRING_INDEX(SUBSTRING_INDEX(Z.suffix, '','', n.digit+1), '','', -1)
44  sufix
45    FROM
46    (SELECT A.idsesi,concat('''',inprefixawalawal,'''','','',',inprefixawal,')
47  AS prefixawal,',inprefixbaru ,' as prefixbaru,
48    ');
49    IF b LIKE ',%' AND b NOT LIKE '%,'
50  THEN   SET @statement =CONCAT(@statement, ' SUBSTR(A.map,length (A.map) +
51  LENGTH(substring('''',b,'''',2))) AS suffix ' );
52     ELSE
53     SET     @statement     =CONCAT(@statement,     '     SUBSTR(A.map,LOCATE(
54  '''',b,'''',A.map)+LENGTH('''',b,'')) AS suffix ' );
55  END IF;
56     SET       @statement       =CONCAT(@statement,       'FROM       (SELECT
57  a.idsesi,funcseqsuffix(a.idsesi,'''',inprefixawalawal,'''',',inprefixawal,',
58  ',itera,') map FROM
59     (SELECT DISTINCT idsesi FROM `session` WHERE idsesi=',c,')a) A
60          WHERE    A.map    LIKE    ',inprefixbaru,'    OR    A.map    LIKE
61  '''',inprefixbaru,',%''  OR  A.map  LIKE  ''%,',inprefixbaru,',%''  OR  A.map
62  LIKE ''%,',inprefixbaru,'''') Z
63    INNER JOIN
64  (SELECT ',isi,') n
65   ON  LENGTH(REPLACE(Z.suffix, '','' , ''''))  <=  LENGTH(Z.suffix)-n.digit
66  ');
67  ....
68        END IF;
69        END LOOP;
70       CLOSE cur1;
         END$$
  ......
```

Gambar 4. 8 Potongan Source Code Pembagian Ruang Pencarian Iterasi 2 dan selanjutnya

4.3.3. Implementasi Analisis Pola

Proses analisis pola dilakukan dengan cara melakukan perhitungan *confidence* terhadap *rule* yang didapat. Perhitungan dimulai dari *rule* yang memiliki length-2 sampai pola yang memiliki length-k. Dalam penggalian pola sekuensial untuk mendapatkan *rule* dilakukan dengan cara mengambil semua

himpunan pola sekuensial yang mempunyai length lebih besar sama dengan 2. Kemudian untuk length-2 maka *rule* yang terbentuk adalah item pertama menjadi antecedent dan item kedua menjadi consequent. Untuk pola yang memiliki length lebih dari 2 maka yang dijadikan consequent adalah item terakhir, kemudian semua item sebelum item terakhir menjadi antecedent.

```
1   .........
2   IF (alength = 1)
3     THEN
4     SET      @statement      =CONCAT('insert      into      hasilpola(SELECT
5     concat(a.prefixawal,''->'', a.prefixbaru),
6     a.kemunculan1/trans.jumtak)*100 AS support,
7     (a.kemunculan1/b.kemunculan2)*100 as confidence,
8     (c.munculbeck/trans.jumtak)*100 as beckconf
9     FROM
10    (SELECT  COUNT(DISTINCT  idsesi,prefixawal,prefixbaru)  AS  kemunculan1,
11    prefixawal,prefixbaru FROM suffix WHERE prefixawal ='''',inprefixawal,'''
12    AND prefixbaru = '''',inprefixbaru,''') a,
13    (SELECT COUNT(DISTINCT idsesi,prefixbaru)AS kemunculan2 ,prefixbaru FROM
14    suffix       WHERE       prefixbaru       =       (SELECT       DISTINCT
15    SUBSTR('''',inprefixawal,'''',2) FROM suffix WHERE iterasi=2)
16    GROUP BY prefixbaru)b,
17    (SELECT   COUNT(aa.a)   AS   munculbeck   FROM   (SELECT   (COUNT(DISTINCT
18    idmapping))   AS   a,   idmapping,idsesi   FROM   `transaksi`   GROUP   BY
19    `idmapping`,idsesi) aa
20    where idmapping ='''',inprefixbaru,'''GROUP BY `idmapping`)c,
21    (SELECT COUNT(DISTINCT idsesi) AS jumtak FROM transaksi) trans)');
22  ......
23    ELSE
24    SET      @statement      =CONCAT('insert      into      hasilpola(SELECT
25    CONCAT(SUBSTRING_INDEX(prefixawal,'','',LENGTH(prefixawal)-
26    LENGTH(REPLACE(prefixawal,'','',''''))+',cnt2,'),''-
27    >'',SUBSTRING_INDEX(prefixawal,'','',(LENGTH(prefixawal)-
28    LENGTH(REPLACE(prefixawal,'','',''''))-(LENGTH(prefixawal)-
29    LENGTH(REPLACE(prefixawal,'','',''''))))-',cnt,'),'','', a.prefixbaru),
30      (a.kemunculan1/trans.jumtak)*100 AS support,
31      (a.kemunculan1/b.kemunculan2)*100 as confidence,
32      (c.munculbeck/trans.jumtak)*100 as beckconf
33    FROM(SELECT      COUNT(DISTINCT      idsesi,prefixawal,prefixbaru)      AS
34    kemunculan1,prefixawal,prefixbaru   FROM   suffix   WHERE   prefixawal
35    ='''',inprefixawal,''' AND prefixbaru ='''',inprefixbaru,''') a,
36    (SELECT  COUNT(DISTINCT  idsesi,prefixawal)  AS  kemunculan2  FROM  suffix
37    WHERE
38    prefixawal=(SUBSTRING_INDEX('''',inprefixawal,'''','','',(',alength,'-
39    ',cnt,')))AND                                                  prefixbaru
40    =(SUBSTRING_INDEX(SUBSTRING_INDEX('''',inprefixawal,'''','','',(',alength,
41    '-',cnt,')+1),'','',-1))) b,  /*kemunculan antecedent*/
42    (SELECT   COUNT(aa.a)   AS   munculbeck   FROM   (SELECT   (COUNT(DISTINCT
43    idmapping))   AS   a,   idmapping,idsesi   FROM   `transaksi`   GROUP   BY
44    `idmapping`,idsesi) aa
45          where idmapping ='''',inprefixbaru,'''
46          GROUP BY `idmapping`
            )c,
            (SELECT COUNT(DISTINCT idsesi) AS jumtak FROM transaksi) trans)');
    .......
```

Gambar 4. 9 Potongan Source code Perhitungan *Confidence* dari *rule* yang terbentuk

Pada gambar 4.9 adalah proses perhitungan *confidence* dari *rule* yang sudah terbentuk. Baris 3-21 adalah perhitungan support dan *confidence* dari pola yang

memiliki dua item, untuk pola yang memiliki item lebih dari dua proses perhitungan support dan *confidence* terdapat pada baris 23-46.

4.3.4. Implementasi Antarmuka

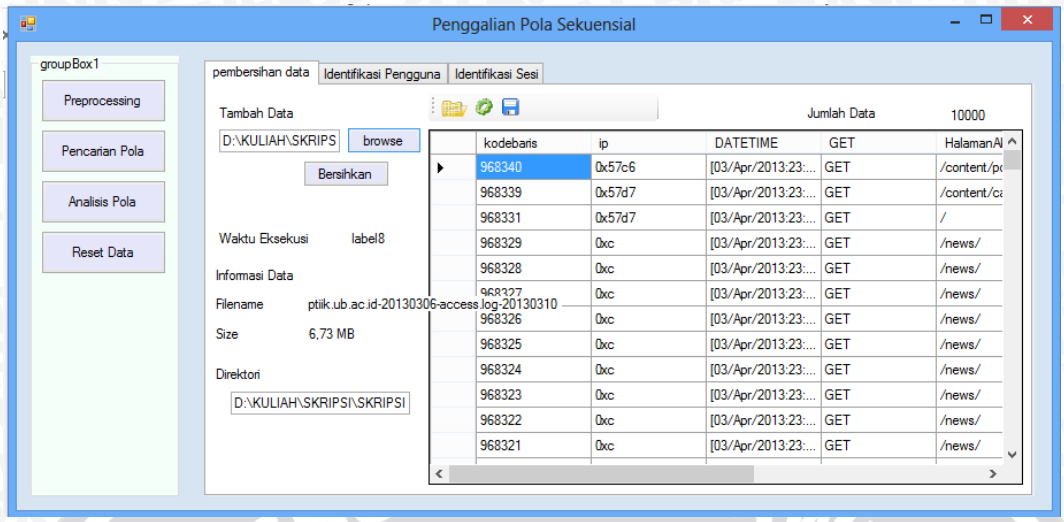
Antarmuka berfungsi sebagai media untuk berinteraksi antara pengguna dengan sistem. Antarmuka sistem penggalian pola sekuensial ini dibagi menjadi 3 bagian utama yaitu, antarmuka halaman *preprocessing*, halaman pencarian pola, dan halaman analisis pola.

4.4.1. Antarmuka *preprocessing*

Antamuka *preprocessing* terdiri dari tiga bagian, yaitu proses pembersihan data, identifikasi pengguna, dan identifikasi sesi.

- Antarmuka pembersihan data

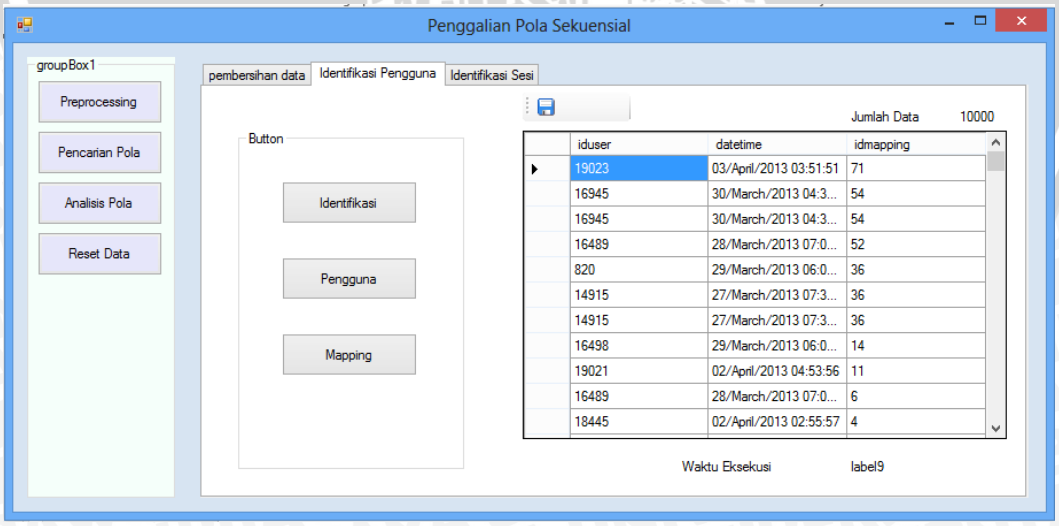
Pada gambar 4.10 merupakan antarmuka dari halaman pembersihan data. Halaman ini data dapat digunakan oleh pengguna untuk memasukkan file web access log dengan cara menekan tombol ‘browse’ . File masukan dari pembersihan data hanya bisa satu file untuk setiap prosesnya. Ketika tombol browse ditekan maka pengguna dapat memilih file web acces log mana yang akan di masukkan, setelah file dipilih akan terdapat informasi dari file tersebut yaitu informasi nama file, ukuran file, dan direktori file. Untuk memulai proses pemberishan data, pengguna cukup menenkan tombol “bersihkan”. Setelah pembersihan data selesai akan ditampilkan waktu yang diperlukan sistem untuk membersihkan satu file web access log tersebut. Hasil proses pembersihan data akan ditampilkan dalam bentuk tabel.



Gambar 4. 10 Antarmuka Pembersihan Data

- Antarmuka identifikasi pengguna

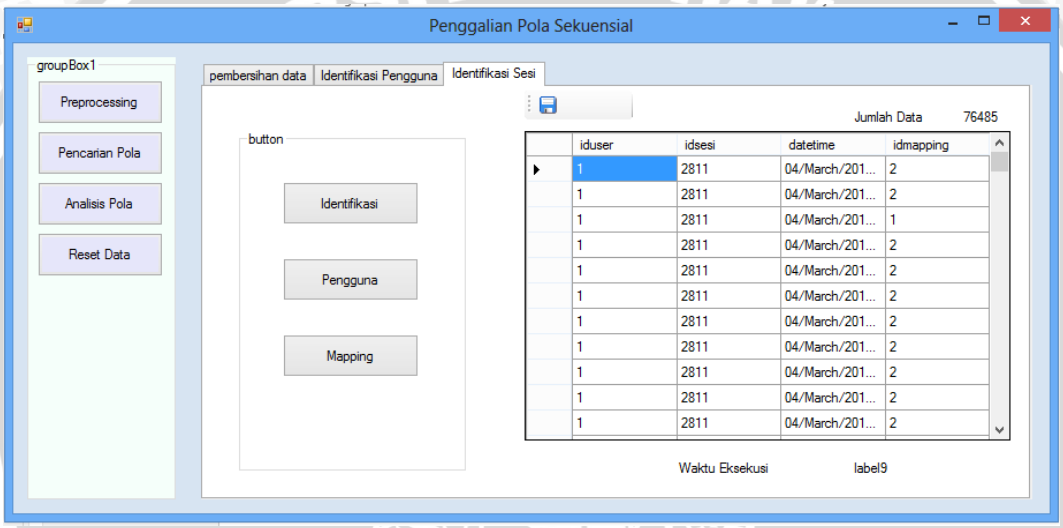
Pada tampilan identifikasi pengguna yang ditunjukkan oleh gambar 4.11 dapat dilihat bahwa halaman tersebut hanya terdapat tiga tombol yaitu tombol identifikasi yang digunakan untuk memulai proses identifikasi pengguna, tombol pengguna yang digunakan untuk melihat id user, alamat ip , dan user agent. Kemudian tombol mapping yang digunakan untuk melihat mapping angka dari halaman akses. Pada halaman ini juga terdapat label yang menunjukan berapa lama waktu yang dibutuhkan untuk menyelesaikan proses identifikasi pengguna.



Gambar 4. 11 Antarmuka Identifikasi Pengguna

- Antarmuka identifikasi sesi

Sama seperti tampilan identifikasi pengguna, identifikasi sesi juga terdapat tiga button yaitu button identifikasi yang digunakan untuk memulai proses identifikasi sesi, button pengguna yang digunakan untuk melihat id user, alamat ip , dan user agent. Kemudian tombol mapping yang digunakan untuk melihat mapping angka dari halaman akses. Pada halaman ini juga terdapat label yang menunjukan berapa lama waktu yang dibutuhkan untuk menyelesaikan proses identifikasi sesi. Antarmuka halaman identifikasi sesi dapat dilihat pada gambar 4.12

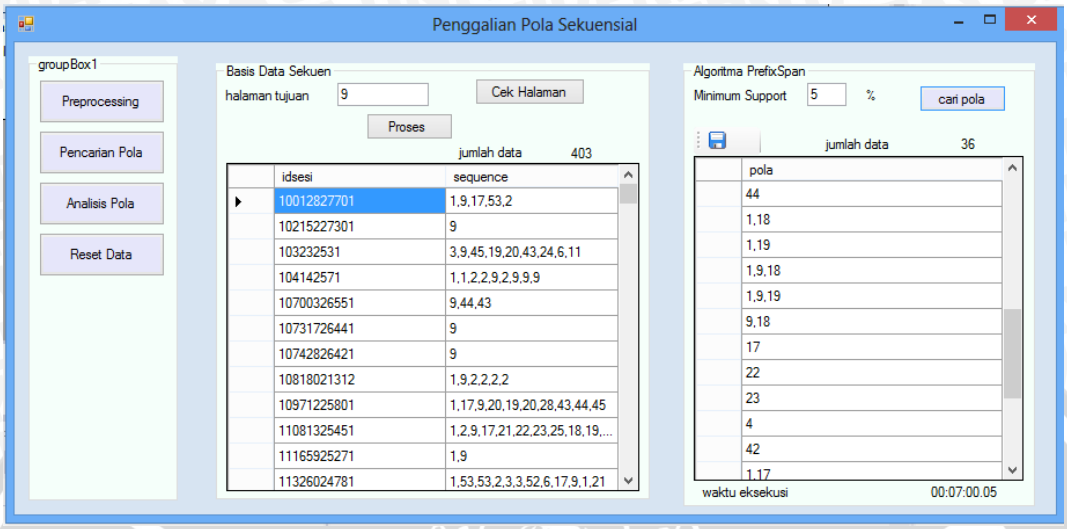


Gambar 4. 12 Antarmuka Identifikasi Sesi

4.4.2. Antarmuka Pencarian Pola

Pada antarmuka ini terdapat dua bagian, yang pertama adalah untuk membuat *sequence database*. *Sequence database* akan terbentuk sesuai dengan masukan dari pengguna. Misalnya pengguna memberikan masukan idmapping '5'. Maka semua transaksi yang pernah mengaskes halaman 5 akan terpilih sebagai *sequence database*. Bagian yang kedua adalah bagian pencarian pola menggunakan algoritma *PrefixSpan*. Pengguna hanya perlu memasukkan minimum *support* yang di inginkan. Minimum *support* yang dimasukkan mulai dari 1-100, kemudian menekan tombol proses, maka proses pencarian pola akan mulai berlangsung. Saat proses pencarian pola selesai hasil pola akan ditampilkan

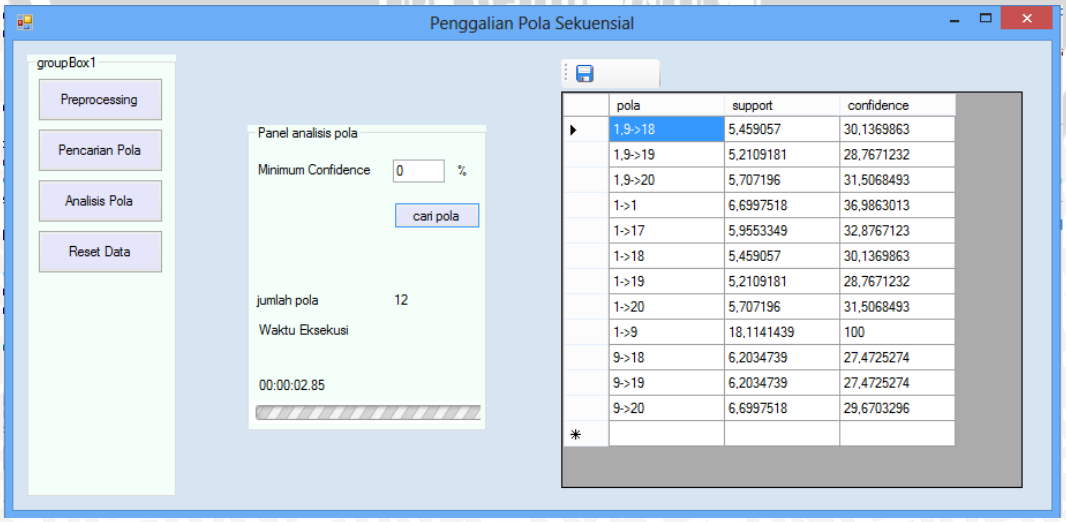
dalam bentuk tabel. Antamuka untuk proses pencarian pola dapat dilihat pada gambar 4.13.



Gambar 4. 13 Antarmuka Pencarian Pola

4.4.3. Antarmuka Analisis Pola

Pada gambar 4.14 dapat dilihat bahwa pengguna cukup memasukkan minimum *confidence* yang di inginkan kemudian menekan tombol 'cari pola'. Setelah proses selesai akan ditampilkan nilai *support* dan *confidence* dari masing-masing pola yang lebih besar sama dengan nilai minimum *confidence* dari yang dimasukkan oleh pengguna.



Gambar 4. 14 Antarmuka Identifikasi Pola