# 4. IMPLEMENTASI SISTEM

Pada bab ini dibahas implementasi sistem, yaitu penerapan teori penunjang dan analisa sistem menjadi sebuah aplikasi yang sudah dapat digunakan. Teori-teori dan algoritma penyelesaian oleh perangkat lunak diimplementasikan dalam sistem melalui *source code*.

## 4.1 Implementasi Pembuatan Tabel Matriks untuk Mencari Semua Jalur Dari Satu Poin ke Poin Lainnya

### 4.1.1. Pencarian Matriks Derajat Satu

Matriks derajat satu diperoleh dari tabel jalan. Matriks ini menandakan ada tidaknya jalan dari poin asal ke poin tujuan dengan hanya melewati satu jalan. Bila pada tabel jalan terdapat jalan dari poin asal ke poin tujuan maka informasi tentang jalan tersebut ditambahkan pada matriks derajat satu. Pembuatan matriks derajat satu ini berguna untuk pembuatan matriks derajat dua dan selanjutnya. *Listing* program pembuatan matriks derajat satu yaitu pada Segmen Program 4.1.

Segmen Program 4.1. Pencarian matriks derajat satu

```
//cari jumlah jalan dari no poin asal ke no poin tujuan
ado.ADOQuery1.SQL.Text:='select count(*)as jumlah from jalan where
     no_grup_asal = '+inttostr(no_grup)+' and no_grup_tujuan =
     '+inttostr(no_grup)+' and
     no_poin_asal='+inttostr(no_poin_asal)+' and
     no_poin_tujuan='+inttostr(no_poin_tujuan)+'';
ado.ADOQuery1.Open;
if(ado.ADOQuery1['jumlah']>0)then
    begin
    //input ke tabel matriks bila terdapat jalur dari poin asal ke
      poin tujuan
    sql:='insert into matriks(derajat, no_grup_asal, no_poin_asal,
          no_grup_tujuan, no_poin_tujuan, jalur,no_urut,
          sub_poin_asal, sub_poin_tujuan)';
    sql:=sql+' values (1, '+inttostr(no_grup)+', '+
          inttostr(no_poin_asal)+','+ inttostr(no_grup)+','+
          inttostr(no_poin_tujuan)+',1,1,'+ inttostr(
          no_poin_asal)+','+ inttostr(no_poin_tujuan)+')';
    ado.ADOQuery1.SQL.Text:=sql;
    ado.ADOQuery1.ExecSQL;
    end;
```

Segmen Program 4.1. tersebut menunjukkan perintah SQL untuk mencari apakah terdapat jalan dari poin asal ke poin tujuan yang tersimpan dalam tabel jalan. Apabila memang terdapat jalan antara poin asal dan poin tujuan maka pada matriks derajat satu tersebut ditambahkan informasi bahwa terdapat sebuah jalan yang menghubungkan poin asal dan poin tujuan.

### 4.1.2. Pencarian Matriks Derajat Lebih Dari Satu

Matriks derajat n berarti menandakan bahwa dari poin asal, mobil pengirim barang dapat menuju ke poin tujuan dengan melewati n buah jalan. Matriks derajat dua didapat dari hasil operasi antara matriks derajat satu dengan matriks derajat satu. Sedangkan matriks derajat tiga didapat dari hasil operasi matriks derajat dua dengan matriks derajat satu. *Listing* program untuk pembuatan matriks derajat n untuk derajat lebih dari satu adalah sebagai berikut.

Segmen Program 4.2. Pengecekan valid atau tidak

```
//cek_valid
//cek apakah ada sub poin asal atau sub poin tujuan yang sama
  dengan poin tujuan
//bila tidak ada maka valid
sql:='select count(*)as jumlah from (SELECT jalur, no_urut,
      sub_poin_asal, sub_poin_tujuan FROM matriks WHERE
      derajat = '+inttostr(derajat_ke-1)+' AND no_grup_asal =
      '+inttostr(no_grup)+' AND no_poin_asal =
      '+inttostr(no_poin_asal)+' AND no_grup_tujuan =
      '+inttostr(no_grup)+' AND no_poin_tujuan = '+inttostr
      (sub_poin_tujuan)+' and jalur = '+ inttostr(jalur_ke)+')as
      table1';

sql:=sql+' where '+inttostr (no_poin_asal)+ '=' +
      IntToStr(poin_tujuan) + ' or sub_poin_asal =
      '+IntToStr(poin_tujuan)+' or sub_poin_tujuan =' +
      IntToStr(poin_tujuan) + '';

ado.ADOQuery3.SQL.Text:=sql;
ado.ADOQuery3.Open;
if(ADO.ADOQuery3['jumlah']>0)then
  valid:=false
  else valid:=true;
```

Setelah didapatkan sebuah jalur dari poin asal ke poin tujuan, diperlukan pengecekan valid atau tidaknya jalur dari poin asal ke poin tujuan. Seperti yang terdapat pada Segmen Program 4.2. jalur tersebut dikatakan valid apabila sebuah poin tidak dilewati sebanyak dua kali atau lebih dalam sebuah jalur tersebut.

Apabila ternyata terdapat poin yang sama dalam sebuah jalur maka jalur tersebut tidak ditambahkan pada tabel matriks. Apabila jalur tersebut valid maka sub poin asal dan sub poin tujuan yang dilewati jalur tersebut diambil dari matriks derajat sebelumnya dan ditambahkan dengan sebuah jalan menuju poin tujuan dari matriks derajat satu untuk dimasukkan pada matriks derajat tertentu yang sedang di-*generate*.

Segmen Program 4.3. Pencarian jalan dari poin asal ke sub poin tujuan

```
//cari apakah ada jalan dari poin_asal ke sembarang poin
no_poin_asal:=map.poin[poin_asal_ke-1].no_poin;

ado.ADOQuery1.SQL.Text:='SELECT distinct derajat, no_grup_asal,
            no_poin_asal, no_grup_tujuan, no_poin_tujuan FROM
            matriks WHERE no_grup_asal = '+inttostr(no_grup)+' and
            no_poin_asal = '+inttostr(no_poin_asal)+' and derajat
            = '+inttostr(derajat_ke-1)+' and no_poin_tujuan <>
            no_poin_asal';

ado.ADOQuery1.Open;
jumlah_sub_poin_tujuan:=ado.ADOQuery1['jumlah'];
```

Langkah pertama yang dilakukan dalam mencari jalur antara poin asal dan poin tujuan adalah mencari poin mana saja yang dapat dijangkau dari poin asal. Poin-poin yang dapat dijangkau dari poin asal tersebut dianggap sebagai sub poin tujuan. Semua sub poin tujuan yang diperoleh disimpan terlebih dahulu untuk diproses lebih lanjut pada Segmen Program 4.4.. Perintah SQL yang digunakan untuk melakukan langkah ini adalah seperti yang terdapat pada Segmen Program 4.3.

Setelah pencarian sub poin tujuan tersebut yang dilakukan selanjutnya adalah mencari dari tabel matriks derajat satu, poin mana saja yang dapat dijangkau dari sub poin tujuan ke poin yang lain yang menjadi poin tujuan. Proses ini seperti yang terdapat pada Segmen Program 4.4. Setelah itu, jalur dari poin asal ke sub poin tujuan dan dari sub poin tujuan ke poin tujuan diseleksi apakah jalur tersebut termasuk valid atau tidak. Pengecekan tersebut seperti yang telah dijelaskan pada Segmen Program 4.2. Apabila valid maka jalur tersebut ditambahkan pada matriks yang sedang di-*generate*.

Segmen Program 4.4. Pencarian jalan dari sub poin tujuan ke sembarang poin

```
//cari sub poin tujuan
ado.ADOQuery2.SQL.Text:='select distinct derajat, no_grup_asal,
                no_poin_asal, no_grup_tujuan, no_poin_tujuan from
                matriks where no_grup_asal = ' +
                inttostr(no_grup)+' and no_poin_asal = ' +
                inttostr(sub_poin_tujuan)+' and derajat = 1 and
                no_poin_tujuan <> no_poin_asal';
ado.ADOQuery2.Open;
for poin_tujuan_ke:=1 to jumlah_poin_tujuan do begin
  poin_tujuan:=ado.ADOQuery2['no_poin_tujuan'];
  //ambil semua jalur pada matriks>1
  ado.ADOQuery3.SQL.Text:='select count(*) as jumlah from (SELECT
                 DISTINCT jalur FROM matriks WHERE derajat =
                 '+inttostr (derajat_ke-1)+' AND no_grup_asal =
                 '+inttostr (no_grup)+' AND no_poin_asal =
                 '+inttostr (no_poin_asal)+' AND no_grup_tujuan =
                 '+inttostr(no_grup)+' AND no_poin_tujuan =
                 '+inttostr (sub_poin_tujuan)+') as table1';
  ado.ADOQuery3.Open;
  jumlah_jalur:=ado.ADOQuery3['jumlah'];
  if(jumlah_jalur>0)then begin
    //cek valid untuk masing-masing jalan yang dilalui
    //alur program untuk pengecekan valid tidaknya jalur yang
      ditemukan dapat dilihat pada Segmen Program 4.2
    end;
  end;
```

## 4.2 Implementasi Pencarian Kombinasi Pengiriman Barang

Terdapat dua macam kombinasi dalam pengiriman barang ini. Kombinasi pertama yang perlu dicari adalah kombinasi pembagian daerah pengiriman ke dalam mobil yang tersedia dan kombinasi urutan pengiriman dalam sebuah mobil pengirim barang. Kombinasi pembagian daerah pengiriman ke mobil yang ada tersebut perlu dicari karena apabila terdapat lebih dari satu buah mobil pengiriman maka tentu terdapat banyak kombinasi dalam membagi daerah pengiriman tersebut ke dalam mobil-mobil yang tersedia. Dalam sebuah mobil pengiriman pun terdapat berbagai kemungkinan urutan dalam mengirimkan barang ke tempat masing-masing pelanggan

Kombinasi-kombinasi tersebut didapatkan dengan cara rekursi. Pada dasarnya algoritma rekursi untuk mendapatkan kombinasi-kombinasi tersebut hampir sama sehingga dapat digabung menjadi sebuah prosedur. Yang membedakan antara kombinasi tersebut adalah variabel tempat penyimpanan akhir kombinasi yang diperoleh. *Listing* program tersebut terdapat pada Segmen Program 4.5.

Segmen Program 4.5. Pencarian kombinasi urutan pengiriman pada sebuah mobil

```
procedure TFormTransitiveClosure.cari_semua_urutan_pengiriman4
(jumlah_max, pengiriman_ke: integer; temp_array: array of integer;
mode:integer);
var nota_ke, isi_array_ke, temp_daerah_pengiriman_ke:integer;
    valid:boolean;
    temp_str:string;

begin
//bila jml pengiriman kurang dari jml pelanggan yg hrs dikirimi
if(nota_ke<=jumlah_max)then
   begin
   for nota_ke:=1 to jumlah_max do
      begin

      //cek valid
      valid:=true;
      if(pengiriman_ke>1)then
      //pengecekan apakah pelanggan tersebut sudah pernah
        dimasukkan dalam urutan pengiriman atau belum
      for temp_daerah_pengiriman_ke:=1 to pengiriman_ke-1 do
         begin

         //jika no pelanggan tsb sudah ada maka urutan pengiriman
           tersebut dianggap tdk valid dan tdk diproses lebih
           lanjut
         if(array_kombinasi_pengiriman[temp_daerah_pengiriman_ke-
           1] = nota_ke)then
            valid:=false;
         end;

      //jika no pelanggan tersebut belum ada dalam urutan
        pengiriman maka tambahkan no pelanggan dalam urutan
        pengiriman
      if(valid=true) then
          begin
          array_kombinasi_pengiriman[pengiriman_ke-1]:=nota_ke;
          //cari pelanggan yg mendapat pengiriman selanjutnya bila
            ada pelanggan yg blm dimasukkan dlm urutan pengiriman


          cari_semua_urutan_pengiriman4(jumlah_max,pengiriman_ke+
          1, mode);
          end;
      end;
   end else
      begin
      //jika semua pelanggan sudah ada dalam urutan kombinasi
      if(mode = 1)then
        //tambahkan kombinasi urutan pengiriman pada array
kombinasi
      if(mode = 2)then
        //tambahkan kombinasi pembagian daerah pada mobil yang ada
      end;
end;
```

Pada Segmen Program 4.5. tersebut, daftar pembagian dibuat dalam bentuk *array*. Apabila belum semua daerah pengiriman tersimpan dalam *array* maka prosedur ini dipanggil lagi untuk menambahkan daerah pengiriman pada *array* tersebut. Apabila daerah pengiriman sudah pernah ada maka perangkat lunak mencoba menambahkan daerah pengiriman yang lain dalam *array* tersebut. Setelah semua daerah pengiriman dimasukkan dalam *array*, urutan pengiriman atau pembagian daerah pengiriman tersebut ditambahkan dalam kombinasi yang ada.

Segmen Program 4.6. Pengecekan Kapasitas Mobil

```
function TFormTransitiveClosure.cek_kapasitas_mobil(kode_mobil,
  urutan_nota: string): boolean;
var temp_pos,total_berat,temp_berat,kapasitas_mobil:integer;
    temp_nota:string;
begin
   //kode_mobil = 'M0001'
   //urutan_nota = 'N0001|N0002|N0003|'
   total_berat:=0;
   temp_pos:=1;
   if(length(urutan_nota)>0)then
   while (temp_pos<=length(urutan_nota)) do
      begin
      temp_nota:=potong(urutan_nota,temp_pos);
      temp_pos:=temp_pos+length(temp_nota)+1;
      ado.ADOQuery4.SQL.Text:='select sum(jumlah*berat) as
            total_berat from detail_penjualan,barang where
            detail_penjualan.kode_barang=barang.kode_barang and
            detail_penjualan.no_nota='''+temp_nota+'''';
      ado.ADOQuery4.Open;
      temp_berat:=ado.ADOQuery4['total_berat'];
      total_berat:=total_berat+temp_berat;
      end;
   ado.ADOQuery4.SQL.Text:='select * from mobil where
            kode='''+kode_mobil+'''';
   ado.ADOQuery4.Open;
   kapasitas_mobil:=ado.ADOQuery4['berat_max'];
   if(total_berat<=kapasitas_mobil)then
      cek_kapasitas_mobil:=true
      else cek_kapasitas_mobil:=false;
end;
```

Perangkat lunak juga dapat diatur untuk memperhatikan kapasitas mobil yang digunakan untuk melakukan pengiriman barang. Apabila penempatan barang-barang dalam mobil juga memperhatikan kapasitas mobil yang digunakan maka dilakukan pengecekan seperti yang terdapat pada Segmen Program 4.6. Pada Segmen Program 4.6. tersebut dilakukan pencarian terhadap jumlah total

berat barang-barang yang dibeli oleh pelanggan pada sebuah nota. Segmen Program 4.6. tersebut meminta masukan berupa kode mobil yang digunakan untuk mengirimkan barang dan nota-nota penjualan yang dikirimkan menggunakan mobil tersebut.

Segmen Program 4.6. tersebut akan mengembalikan nilai *boolean* yaitu *true* dan *false*. Hal itu berarti bahwa apabila nilai yang diperoleh dari hasil Segmen Program 4.6. adalah *true* maka mobil tersebut dapat melakukan pengiriman barang yang terdapat pada nota-nota tersebut. Akan tetapi apabila nilai keluarannya adalah *false* maka mobil tersebut tidak dapat melakukan pengiriman barang-barang pada nota tersebut yang diakibatkan oleh keterbatasan kapasitas mobil yang bersangkutan.

Cara kerja Segmen Program 4.6. tersebut adalah sebagai berikut. Jumlah total berat pada satu nota tersebut ditambahkan dengan total berat nota lainnya yang dikirim dengan menggunakan satu buah mobil. Total berat dari semua nota tersebut dibandingkan dengan kapasitas mobil yang dapat digunakan. Apabila ternyata kapasitas mobil tersebut tidak mencukupi maka Segmen Program 4.6. tersebut akan mengembalikan nilai *false*. Namun sebaliknya, jika kapasitas mobil tersebut dapat mengangkut semua barang yang terdapat pada nota-nota tersebut maka Segmen Program 4.6. tersebut akan mengembalikan nilai *true*.

Segmen Program 4.6. tersebut digunakan oleh prosedur lain yang mencari mobil mana yang dapat melakukan pengiriman barang dengan total berat barang tersebut. Namun apabila semua mobil yang tersedia memiliki kapasitas yang lebih kecil daripada total berat barang yang dikirim maka kombinasi pengiriman tersebut tidak dapat dilakukan. Perangkat lunak tidak mencari jalur pengiriman tercepat apabila pengiriman tersebut tidak memiliki mobil yang dapat digunakan untuk melakukan pengiriman barang.

## 4.3 Implementasi Pencarian Waktu Keberangkatan dan Jalur Pengiriman Tercepat

### 4.3.1. Pencarian Waktu Bongkar Barang

Penghitungan total waktu bongkar barang yang terjual pada sebuah nota dilakukan seperti yang terdapat pada Segmen Program 4.7. Segmen Program 4.7.

tersebut meminta masukan berupa nomor nota yang akan dihitung waktu untuk pembongkaran barang yang terdapat pada sebuah nota dan menghasilkan keluaran berupa total waktu yang dibutuhkan untuk membongkar semua barang pada nota tersebut. Sebuah nota tersebut tentunya dikirim pada seorang pelanggan. Setiap pelanggan memiliki waktu luang tertentu pada hari tertentu pula. Oleh karena itu penghitungan total waktu bongkar barang ini diperlukan untuk menyesuaikan dengan waktu luang pelanggan yang bersangkutan. Langkah-langkah penghitungan total waktu bongkar barang yang terdapat pada sebuah nota adalah seperti yang terdapat pada Segmen Program 4.7.

Segmen Program 4.7. Penghitungan Waktu Bongkar Barang

```
CREATE PROCEDURE sp_Cari_Waktu_Bongkar
(     @no_nota varchar(6), @total_waktu_bongkar varchar(5) output
)AS

SET @tempTotalWaktuBongkarInt = 0

DECLARE curWaktuBongkar INSENSITIVE CURSOR FOR
SELECT jumlah,waktu_bongkar
      FROM detail_penjualan,barang
      WHERE detail_penjualan.kode_barang = barang.kode_barang and
            no_nota = @no_nota
OPEN curWaktuBongkar
FETCH NEXT FROM curWaktuBongkar INTO
      @curJmlPembelianBrg, @curWaktuBongkar

WHILE @@FETCH_STATUS =  0
BEGIN
      exec convert_time_to_int @curWaktuBongkar,
            @tempWaktuBongkarInt output
      SET @tempWaktuBongkarInt = @tempWaktuBongkarInt *
            @curJmlPembelianBrg
      SET @tempTotalWaktuBongkarInt = @tempTotalWaktuBongkarInt +
            @tempWaktuBongkarInt

      FETCH NEXT FROM curWaktuBongkar INTO
            @curJmlPembelianBrg, @curWaktuBongkar

END
CLOSE curWaktuBongkar
DEALLOCATE curWaktuBongkar

exec convert_int_to_time_str @tempTotalWaktuBongkarInt,
      @tempTotalWaktuBongkarStr output

SET @total_waktu_bongkar = @tempTotalWaktuBongkarStr
```

4.3.2. Pencarian Interval Waktu Keberangkatan dan Alternatif Jalur Pengiriman

Untuk mencari rute tercepat dan memenuhi interval waktu pelanggan yang mendapat pengiriman barang, yang dilakukan pertama kali oleh perangkat lunak adalah mencari interval waktu keberangkatan dari gudang dan mendapatkan semua alternatif jalur yang ada untuk pengiriman barang ke semua pelanggan. Pencarian interval keberangkatan menjadi cukup sulit apabila dilakukan dari gudang hingga pengiriman pada pelanggan yang terakhir. Namun perangkat lunak ini mencari interval keberangkatan mulai dari pelanggan yang paling akhir hingga ditemukan interval waktu dimana mobil harus berangkat dari gudang.

Segmen Program 4.8. Pencarian jam keberangkatan dari poin satu ke poin lain

```
--cari jam keberangkatan dari poin satu ke poin lainnya
DECLARE @jam_awal varchar(5), @no_urut int, @waktu_tempuh
        varchar(5), @jam_akhir varchar(5)
DECLARE abc SCROLL CURSOR FOR
SELECT daftar_jam.no_urut, daftar_jam.jam, jalan.waktu_tempuh
FROM daftar_jam INNER JOIN
    jalan ON daftar_jam.no_urut = jalan.no_urut
WHERE (no_grup_asal=@no_grup) and (no_grup_tujuan=@no_grup) and
      (no_poin_asal = @cur_sub_poin_asal) AND (no_poin_tujuan =
      @cur_sub_poin_tujuan) AND (CONVERT(datetime, daftar_jam.jam)
      + CONVERT(datetime, jalan.waktu_tempuh)
BETWEEN @tempRangeCustAwal AND @tempRangeCustAkhir)
ORDER BY daftar_jam.no_urut
open abc

--cari jam awal & jam akhir keberangkatan
--ambil jam pada record pertama sebagai jam awal
fetch First from abc into @no_urut, @jam_awal, @waktu_tempuh

--ambil jam pada record terakhir sebagai jam akhir
fetch LAST from abc  into @no_urut, @jam_akhir, @waktu_tempuh

--bila tidak ada record yang di-select maka tidak ada jam
keberangkatan yang dapat sampai di tempat pelanggan lain tepat
waktu
fetch First from abc
if @@Fetch_status <> 0
    begin
    set @jam_awal = '00:00'
    set @jam_akhir = '00:00'
    SET @ada_perpotongan = 0
    end

SET @tempRangeCustAwal=@jam_awal
SET @tempRangeCustAkhir=@jam_akhir

close abc
deallocate abc
```

Segmen Program 4.9. Pengisian pada *database* bila ada interval atau jalur baru

```
if(@ada_perpotongan = 1)
    begin
    --cari no jalur_ke dan interval_ke untuk diurutkan dlm
database
    SELECT @temp_jml_jalur = count(jalur_ke)
    FROM Tb_temp_jam_berangkat
    WHERE pengiriman_ke=@pengiriman_ke and
                      urutan_ke=@temp_urutan_ke
    if(@temp_jml_jalur > 0)
        begin

        SELECT @db_jalur_ke = jalur_ke
        FROM Tb_temp_jam_berangkat
        WHERE pengiriman_ke = @pengiriman_ke and jalur =
              @temp_jalur + @temp_jalur_awal and urutan_ke =
              @temp_urutan_ke
        if(@@ROWCOUNT = 0)
            begin
            --interval pasti dimulai dari 1 bila jalur baru
            SET @db_interval_ke = 1

            SELECT @db_jalur_ke = MAX(jalur_ke) + 1
            FROM Tb_temp_jam_berangkat
            WHERE pengiriman_ke = @pengiriman_ke and urutan_ke =
                  @temp_urutan_ke
            if(@@ROWCOUNT = 0)
                begin
                SET @db_jalur_ke = 1
                end
            end else
                begin
                --cari no interval terakhir bila jalur tsb pernah
                  ada
                SELECT @db_interval_ke = MAX(interval_ke) + 1
                FROM Tb_temp_jam_berangkat
                WHERE pengiriman_ke = @pengiriman_ke and jalur_ke =
                      @db_jalur_ke and urutan_ke = @temp_urutan_ke
                if not(@db_interval_ke > 0)
                    begin
                    SET @db_interval_ke = 1
                    end
                end
        END else
            begin
            SET @db_jalur_ke = 1
            SET @db_interval_ke = 1
            end
    --masukkan dalam database bila telah ditemukan waktu harus
      berangkat dari masing-masing jalur alternatif
    INSERT INTO Tb_temp_jam_berangkat (pengiriman_ke, jalur_ke,
           interval_ke, urutan_ke, jam_awal, jam_akhir, jalur)
    VALUES (@pengiriman_ke, @db_jalur_ke, @db_interval_ke,
           @temp_urutan_ke, @tempRangeCustAwal,
           @tempRangeCustAkhir, @temp_jalur + @temp_jalur_awal)
    end
```

Pencarian interval waktu keberangkatan dimulai dari interval waktu pelanggan yang paling akhir. Kemudian didapatkan berapa interval keberangkatan dari pelanggan sebelumnya agar sampai di tempat pelanggan akhir pada jam tersebut. Proses tersebut dilakukan terus menerus hingga nantinya ditemukan pada interval waktu berapa, mobil pengirim barang harus berangkat dari gudang.

Pada Segmen Program 4.8. terdapat langkah-langkah untuk mencari jam keberangkatan dari sebuah poin apabila telah diketahui pada interval waktu berapa mobil pengirim barang tersebut harus sampai di poin lainnya. Pada Segmen Program 4.8. tersebut, dicari jam keberangkatan mana saja yang apabila ditambah dengan waktu tempuh maka waktu tibanya berada di dalam interval waktu kedatangan pada poin selanjutnya.

Jam-jam keberangkatan yang memenuhi interval waktu keberangkatan pada poin selanjutnya tersebut disusun mulai dari yang terkecil hingga yang terbesar. Jam keberangkatan terkecil digunakan sebagai batas awal interval jam keberangkatan yang dapat dilakukan. Sedangkan jam keberangkatan terbesar digunakan sebagai batas akhir interval jam keberangkatan yang dapat dilakukan. Apabila tidak terdapat jam keberangkatan yang memenuhi interval waktu pada poin selanjutnya tersebut maka jalur tersebut dengan interval waktu pelanggan akhir tersebut tidak dapat digunakan.

Apabila interval awal dan akhir jam keberangkatan telah diperoleh pada Segmen Program 4.9. maka interval tersebut ditambahkan dalam *database*. Penambahan pada *database* tersebut membutuhkan nomor jalur dan nomor interval. Oleh karena itu dilakukan pengecekan apakah jalur tersebut sudah pernah dimasukkan pada *database*. Apabila jalur sudah pernah terdaftar maka nomor jalur tersebut diambil untuk memasukkan interval yang baru pada *database*.

Segmen Program 4.10 digunakan untuk mencari jalur-jalur alternatif dari tempat pelanggan pertama ke pelanggan kedua. Jalur-jalur alternatif tersebut didapatkan dari tabel matriks semua derajat. Setiap jalur alternatif dari tabel tersebut terdiri dari beberapa jalan. Hal itu ditentukan dari jumlah derajat pada jalur. Dari interval waktu harus tiba di pelanggan kedua, dicari waktu berangkat pada setiap jalan yang dilalui pada jalur tersebut hingga nantinya diperoleh pada interval berapa, mobil pengirim barang harus berangkat dari pelanggan pertama.

Segmen Program 4.10. Pencarian waktu keberangkatan dari tabel matriks

```
--looping semua jalur dr semua derajat yg didapat dari tb matriks
DECLARE curJalur INSENSITIVE CURSOR FOR
SELECT derajat, no_grup_asal, no_poin_asal, no_grup_tujuan,
       no_poin_tujuan, jalur, no_urut, sub_grup_asal,
       sub_poin_asal, sub_grup_tujuan, sub_poin_tujuan
FROM matriks
WHERE no_grup_asal=@no_grup and no_grup_tujuan=@no_grup and
      no_poin_asal=@no_poin_asal and
      no_poin_tujuan=@no_poin_tujuan
ORDER BY derajat, jalur, no_urut DESC
OPEN curJalur
FETCH NEXT FROM curJalur INTO
    @cur_derajat, @cur_no_grup_asal, cur_no_poin_asal,
    @cur_no_grup_tujuan, @cur_no_poin_tujuan, @cur_jalur,
    @cur_no_urut, @cur_sub_grup_asal, @cur_sub_poin_asal,
    @cur_sub_grup_tujuan, @cur_sub_poin_tujuan
WHILE @@FETCH_STATUS = 0
    BEGIN
    SET @ada_perpotongan = 1
    SET @jalur_ke = @jalur_ke + 1
    --diulang selama hasil query belum habis
    --ambil derajat ke berapa
    SET @derajat=@cur_derajat
    set @i = 0
    SET @temp_jalur = ''
    --isi range awal dan akhir dengan range cust tujuan
    SET @tempRangeCustAwal = @curRangeCustAwal
    SET @tempRangeCustAkhir = @curRangeCustAkhir

    --potong range cust tujuan dgn wkt bongkar brg di cust tujuan
    SET @tempRangeCustAkhir = convert(varchar(5),
        cast(@curRangeCust Akhir as datetime)-
        cast(@temp_waktu_bongkar as datetime), 8)
    --looping sebanyak derajatnya
    while @i < @derajat
        begin
        set @i = @i + 1
        --simpan jalur pada temp_jalur
        if (@i = 1)
            SET @temp_jalur = cast(@cur_sub_poin_tujuan as varchar)
                + '|' --+ @temp_jalur
        SET @temp_jalur = cast(@cur_sub_poin_asal as varchar) +
                '|' + @temp_jalur
        --cari jam berangkat dr poin pada Segmen Program 4.8.
        --lanjutkan cari record jalur yang lain
        FETCH NEXT FROM curJalur INTO
            @cur_derajat, @cur_no_grup_asal, @cur_no_poin_asal,
            @cur_no_grup_tujuan, @cur_no_poin_tujuan, @cur_jalur,
            @cur_no_urut, @cur_sub_grup_asal, @cur_sub_poin_asal,
            @cur_sub_grup_tujuan, @cur_sub_poin_tujuan
        end  --end while @i
    --bila terdapat interval awal dan akhir jam berangkat (Segmen
      Program 4.9.
    --fetch dr curJalur ada di dalam derajat sebelum end while @i
    END
    CLOSE curJalur
    DEALLOCATE curJalur
```

Segmen Program 4.11. Pencarian waktu berangkat untuk pengiriman ke semua pelanggan

```
--looping masing2 range jadwal cust tujuan
if(@pengiriman_ke = 1) begin
    --jika pengiriman pada pelanggan terakhir (urutan pengiriman
      telah dibalik) maka ambil waktu tiba dari range pelanggan
      yang terakhir mendapat pengiriman barang
    DECLARE curJadwalCust INSENSITIVE CURSOR FOR
    SELECT no_urut,jam_awal,jam_akhir
    FROM jadwal_pelanggan
    WHERE no_pelanggan=@no_pelanggan_tujuan and tanggal_pengiriman
          = @tanggal_pengiriman
    OPEN curJadwalCust
    FETCH NEXT FROM curJadwalCust INTO
        @curNoUrutRange, @curRangeCustAwal, @curRangeCustAkhir
    end else begin
        --jika bukan pengiriman pada pelanggan terakhir, ambil
          interval waktu harus tiba ke pelanggan selanjutnya dari
          tabel Tb_temp_jam_berangkat
        DECLARE curJadwalCust INSENSITIVE CURSOR FOR
        SELECT jalur_ke,jam_awal,jam_akhir,jalur
        FROM Tb_temp_jam_berangkat
        WHERE pengiriman_ke = @pengiriman_ke - 1 and urutan_ke =
              @temp_urutan_ke
        OPEN curJadwalCust
        FETCH NEXT FROM curJadwalCust INTO @curJalurKeAwal,
            @curRangeCustAwal, @curRangeCustAkhir, @curJalurAwal
        end

WHILE @@FETCH_STATUS = 0 BEGIN
    SET @interval_ke = @interval_ke + 1
    --ambil temp_jalur_awal
    SET @temp_jalur_awal = ''
    if (@pengiriman_ke <> 1)
        SET @temp_jalur_awal = @curJalurAwal
    SET @jalur_ke = 0

    --looping semua jalur alternatif dari semua derajat yang
       didapatkan dari tabel matriks pada Segmen Program 4.10.
    --potong jam keberangkatan dengan range cust asal
    --kecuali untuk pengiriman dari gudang
    if(@pengiriman_ke<>@jml_pelanggan)
        exec sp_CariIrisanDgnCustAsal3 @no_pelanggan_asal,
            @tanggal_pengiriman, @pengiriman_ke, @temp_urutan_ke
        --Segmen Program 4.16.
    --fetch next dari cursor curJadwalCust
    if(@pengiriman_ke = 1) begin
        FETCH NEXT FROM curJadwalCust INTO
            @curNoUrutRange,@curRangeCustAwal,@curRangeCustAkhir
        end else begin
            FETCH NEXT FROM curJadwalCust INTO @curJalurKeAwal,
                @curRangeCustAwal, @curRangeCustAkhir,
                @curJalurAwal
            end
END
CLOSE curJadwalCust
DEALLOCATE curJadwalCust
```

Pencarian waktu keberangkatan dari gudang ke semua pelanggan dimulai dengan range pelanggan yang paling akhir. Seperti yang terdapat pada Segmen Program 4.11, masing-masing interval waktu pelanggan terakhir dicari waktu keberangkatannya dari gudang. Untuk pengiriman pada pelanggan yang terakhir, dilakukan pengambilan interval waktu pelanggan sebagai waktu harus tiba di pelanggan tujuan. Akan tetapi untuk pengiriman selain ke pelanggan terakhir diambil dari perhitungan sebelumnya yang disimpan dalam tabel sementara.

Segmen Program 4.12.1. Pencarian interval waktu berangkat dan alternatif jalur

```
CREATE PROCEDURE sp_CariIntervalJamBktDanAlternatifJalur2
(   @no_grup int, @urutan_no_nota
    varchar(1000),@tanggal_pengiriman datetime
)AS

DECLARE
    --untuk pendeklarasian variabel-variabel yang digunakan
    --cari no poin gudang dari grup tertentu
    SELECT @no_poin_gudang = no_poin_gudang
    FROM gudang
    WHERE no_grup = @no_grup

    SET @temp_urutan_nota = @urutan_no_nota
    --balik urutan pelanggan yang mendapat pengiriman untuk proses
      penghitungan interval jam keberangkatan
    --mencari apakah sudah ada urutan pelanggan seperti ini pada
      tabel daftar_urutan pelanggan
    SET @temp_urutan_ke = 0
    SELECT @temp_urutan_ke = urutan_ke
    FROM daftar_urutan_pengiriman
    WHERE urutan_nota = @urutan_nota_tanpa_jml and urutan_ke IS
          NOT NULL

    --bila urutan ini belum ada dalam tabel
      daftar_urutan_pelanggan
    if(@temp_urutan_ke=0)
        begin
        --cari index yang masih kosong untuk urutan pengiriman
          3|2|1| atau yang lain
        SELECT @temp_urutan_ke = ISNULL(max(urutan_ke),0)+1
        FROM daftar_urutan_pengiriman
        --insert urutan pengiriman 3|2|1| ke tabel
          daftar_urutan_pelanggan
        INSERT INTO daftar_urutan_pengiriman (tanggal_pengiriman,
          urutan_ke, urutan_nota, jml_nota)
        VALUES (@tanggal_pengiriman, @temp_urutan_ke,
          @urutan_nota_tanpa_jml, @jml_nota)
        end
    --tambahkan no poin gudang pada bagian belakang urutan
      pengiriman pada nota yang telah dibalik
    SET @temp_reverse_nota = @reverse_nota + cast(@no_poin_gudang
    as varchar) + '|'
```

Segmen Program 4.12.2. Pencarian interval waktu berangkat dan alternatif jalur(lanjutan)

```
    --cari semua cust yang dikirimi
    SET @pengiriman_ke = 0

    WHILE (@pengiriman_ke<@jml_nota)
        begin

        --inisialisasi no pelanggan asal dan no pelanggan tujuan
        SET @pengiriman_ke = @pengiriman_ke + 1

        SELECT @temp_pos = CHARINDEX( '|', @temp_reverse_nota)
        SET
        @temp_no_nota_tujuan=LEFT(@temp_reverse_nota,@temp_pos-1)
        SET @temp_reverse_nota =
        RIGHT(@temp_reverse_nota,LEN(@temp_reverse_nota) -
        @temp_pos)

        SELECT @no_pelanggan_tujuan = no_pelanggan
        FROM penjualan
        WHERE no_nota = @temp_no_nota_tujuan
        SELECT @temp_pos = CHARINDEX( '|', @temp_reverse_nota)
        SET @temp_no_nota_asal =
        LEFT(@temp_reverse_nota,@temp_pos-1)

        SELECT @no_pelanggan_asal = no_pelanggan
        FROM penjualan
        WHERE no_nota = @temp_no_nota_asal

        --cari no poin tempat pelanggan asal dan tujuan
        if(@pengiriman_ke<>@jml_nota)
            begin
            SET @no_poin_asal = (SELECT no_poin FROM pelanggan
                WHERE no_pelanggan=@no_pelanggan_asal)
            end else
                begin - untuk pengiriman dari poin gudang
                SET @no_poin_asal = @temp_no_nota_asal
                end

        SET @no_poin_tujuan = (SELECT no_poin
            FROM pelanggan
            WHERE no_pelanggan=@no_pelanggan_tujuan)

        --ambil waktu bongkar barang di pelanggan tujuan
        exec sp_Cari_Waktu_Bongkar @temp_no_nota_tujuan,
        @temp_waktu_bongkar output

        SET @interval_ke = 0
        --looping masing2 range jadwal cust tujuan seperti yang
          terdapat pada Segmen Program 4.11.
        end-- while pengiriman ke

--menambahkan no grup pada jalur
update Tb_temp_jam_berangkat set jalur=cast(@no_grup as
    varchar)+'|'+jalur
GO
```

Bagian utama dari pencarian interval jam keberangkatan dan alternatif jalur terdapat pada Segmen Program 4.12.1. dan Segmen Program 4.12.2. Pada segmen program ini, yang dilakukan pertama kali adalah mengecek apakah kombinasi tersebut sudah ada pada tabel daftar urutan pelanggan. Apabila kombinasi ini belum pernah ada maka kombinasi tersebut ditambahkan pada tabel daftar urutan pelanggan. Langkah selanjutnya adalah membalik urutan pengiriman sehingga pelanggan yang paling akhir terletak pada bagian awal deretan pengiriman. Kemudian no poin gudang ditambahkan pada bagian akhir deretan pengiriman tersebut. Ambil no poin tempat pelanggan asal maupun tujuan. Kecuali nomor gudang yang menyimpan no gudangnya langsung. Mulai dari pengiriman ke satu hingga pengiriman sepanjang jumlah pelanggan, dilakukan Segmen Program 4.11.

### 4.3.3. Pencarian Irisan Waktu Harus Berangkat dan Interval Waktu Pelanggan yang Mendapat Pengiriman Barang

Waktu tiba mobil pengirim barang di tempat seorang pelanggan ditentukan oleh waktu harus berangkat mobil pengirim barang tersebut ke tempat pelanggan selanjutnya dan interval waktu pelanggan yang mendapat pengiriman barang tersebut. Jadi untuk mengetahui pada interval waktu berapakah mobil pengirim barang sebaiknya tiba di tempat pelanggan, perangkat lunak mencari irisan antara waktu harus berangkat dari tempat pelanggan dan interval waktu waktu pelanggan yang bersangkutan. Dengan demikian maka mobil pengirim barang tidak sampai di tempat pelanggan di luar interval waktu waktu pelanggan dan tidak terlambat untuk sampai di tempat pelanggan selanjutnya. *Listing* program untuk mencari irisan antara waktu harus berangkat dari seorang pelanggan dan interval waktu pelanggan tersebut adalah sebagai berikut.

Pengecekan ada perpotongan atau tidaknya antara interval waktu harus berangkat dan interval waktu pelanggan adalah seperti yang terdapat pada Segmen Program 4.13. Segmen program tersebut mencari interval awal dan interval akhir dari irisan yang didapat dari dua interval waktu tersebut. Interval pertama yang dibatasi oleh awal1 dan akhir1 digunakan sebagai interval waktu harus berangkat

untuk menuju ke pelanggan selanjutnya. Sedangkan interval kedua digunakan sebagai interval waktu pelanggan yang bersangkutan.

Segmen Program 4.13. Irisan antara waktu tiba dan interval waktu pelanggan

```
--pencarian perpotongan waktu tiba dan interval waktu pelanggan
  asal
SET @ada_perpotongan=1
--posisi awal2 berada di awal sebelum awal1
if @awal2<=@awal1
    begin
    SET @awal3=@awal1
    if(@akhir2<@awal1)
        begin
        SET @ada_perpotongan=0
        end
    if((@akhir2>=@awal1)and(@akhir2<=@akhir1))
        begin
        SET @akhir3=@akhir2
        end
    if(@akhir2>@akhir1)
        begin
        SET @akhir3=@akhir1
        end
    end
--posisi awal2 berada di tengah awal1 dan akhir1
if ((@awal2>@awal1)and(@awal2<=@akhir1))
    begin
    SET @awal3=@awal2
    if((@akhir2>=@awal1)and(@akhir2<=@akhir1))
        begin
        SET @akhir3=@akhir2
        end
    if(@akhir2>=@akhir1)
        begin
        SET @akhir3=@akhir1
        end
    end
--posisi awal2 berada setelah akhir1
if (@awal2>@akhir1)
    begin
    SET @ada_perpotongan=0
    end
if(@ada_perpotongan=0)
    begin
    SET @awal3=cast('00:00' as datetime)
    SET @akhir3=cast('00:00' as datetime)
    end
```

Perpotongan yang dicari pada Segmen Program 4.13. antara waktu harus berangkat dan interval waktu pelanggan yang bersangkutan tentunya harus memperhatikan semua interval waktu yang dimiliki oleh pelanggan yang bersangkutan. Hal ini dikarenakan setiap pelanggan dapat memiliki lebih dari

sebuah interval waktu. Waktu harus berangkat ke pelanggan selanjutnya pun dapat memiliki irisan dengan lebih dari sebuah interval waktu pelanggan. Hasil perpotongan disimpan pada tabel sementara terlebih dahulu. *Listing* program ini dapat dilihat pada Segmen Program 4.14.

Segmen Program 4.14. Pencarian semua interval waktu pelanggan

```
--cari semua range pelanggan asal
DECLARE curJadwalCustAsal CURSOR FOR
SELECT no_urut,jam_awal,jam_akhir
FROM jadwal_pelanggan
WHERE no_pelanggan=@no_pelanggan_asal and tanggal_pengiriman =
       @tanggal_pengiriman
OPEN curJadwalCustAsal
FETCH NEXT FROM curJadwalCustAsal INTO @curNoUrutRangeAsal,
    @curRangeCustAwalAsal,@curRangeCustAkhirAsal
WHILE @@FETCH_STATUS = 0
BEGIN
    SET @tempRangeCustAwalAsal = @curRangeCustAwalAsal
    SET @tempRangeCustAkhirAsal = @curRangeCustAkhirAsal

    --potong range pelanggan asal tsb dgn waktu bongkar brg di
      pelanggan asal
    SET @tempRangeCustAkhirAsal = convert(varchar(5),
        cast(@curRangeCustAkhirAsal as datetime)-
        cast(@waktu_bongkar as datetime), 8)

    SET @awal1 = cast(convert(varchar(5), cast(@curJam_awal as
        datetime), 8) as datetime)
    SET @awal2 = cast(convert(varchar(5),
        cast(@tempRangeCustAwalAsal as datetime), 8) as datetime)
    SET @akhir1= cast(convert(varchar(5), cast(@curJam_akhir as
        datetime), 8) as datetime)
    SET @akhir2= cast(convert(varchar(5),
        cast(@tempRangeCustAkhirAsal as datetime), 8) as datetime)

    --pencarian apakah ada perpotongan waktu tiba dan range
    pelanggan asal seperti yang terdapat pada Segmen Program 4.13

    --jika memang ada perpotongan, masukkan ke database
    if(@ada_perpotongan = 1) begin
        --cari jam kedatangan dari range jam keberangkatan
        INSERT INTO #DummyIrisan1 (pengiriman_ke, jalur_ke,
               interval_ke, urutan_ke, jam_awal, jam_akhir, jalur)
        VALUES (@curPengiriman_ke, @curJalur_ke, @curInterval_ke,
               @no_urutan_pelanggan, convert(varchar(5), @awal3,
               8), convert(varchar(5), @akhir3, 8),@curJalur)
        end

    FETCH NEXT FROM curJadwalCustAsal INTO
    @curNoUrutRangeAsal,@curRangeCustAwalAsal,@curRangeCustAkhirAs
al
END
CLOSE curJadwalCustAsal
DEALLOCATE curJadwalCustAsal
```

Setelah semua irisan telah dimasukkan dalam tabel sementara, perlu dilakukan pengurutan nomor jalur dan nomor interval yang tersimpan. Dalam tabel sementara sebelumnya, nomor jalur dan nomor interval diisi berdasarkan nomor jalur dan interval yang didapatkan pada waktu harus berangkat ke pelanggan selanjutnya. Padahal belum tentu setiap interval tersebut memiliki irisan dengan interval pelanggan satt ini. Oleh karena itu untuk mengurutkan nomor jalur dan nomor interval pada pilihan jalur yang telah didapatkan, harus dilakukan pengurutan ulang dan dipindah pada tabel sementara lain. Proses ini seperti yang terdapat pada Segemen Program 4.15.1 dan Segmen Program 4.15.2.

Segmen Program 4.15.1. Pengurutan nomor jalur dan nomor interval

```
--cari termasuk alternatif jalur dan pilihan interval ke berapa
SELECT @max_jalur = COUNT(pengiriman_ke)
FROM Tb_temp_jam_berangkat
WHERE urutan_ke = @no_urutan_pelanggan

if(@max_jalur>0)
    BEGIN
    SET @max_jalur = (SELECT max(jalur_ke)
    FROM Tb_temp_jam_berangkat
    WHERE urutan_ke = @no_urutan_pelanggan)
    END

SET @jalur_ke = 0
SET @db_jalur_ke = 0

WHILE (@jalur_ke < @max_jalur)
begin
    SET @jalur_ke = @jalur_ke + 1

    SELECT @max_interval = COUNT(pengiriman_ke)
    FROM #DummyIrisan1
    WHERE jalur_ke = @jalur_ke

    if(@max_interval > 0)
    BEGIN
        SELECT @max_interval = MAX(pengiriman_ke)
        FROM #DummyIrisan1
        WHERE jalur_ke = @jalur_ke
    END

    if(@max_interval > 0)
    BEGIN
        SET @db_jalur_ke = @db_jalur_ke + 1
        SET @db_interval_ke = 0
        --urutkan penomoran jalur alternatif dan penomoran pilihan
          interval keberangkatan
```

Segmen Program 4.15.2. Pengurutan nomor jalur dan nomor interval (lanjutan)

```
        DECLARE curIntervalKe CURSOR FOR
        SELECT pengiriman_ke, jalur_ke, interval_ke, jam_awal,
            jam_akhir, jalur
        FROM #DummyIrisan1
        WHERE jalur_ke = @jalur_ke
        ORDER BY pengiriman_ke,jalur_ke,interval_ke
        OPEN curIntervalKe
        FETCH NEXT FROM curIntervalKe INTO
            @curPengiriman_ke, @curJalur_ke, @curInterval_ke,
            @curJam_awal, @curJam_akhir, @curJalur
        WHILE @@FETCH_STATUS = 0
        BEGIN
            SET @db_interval_ke = @db_interval_ke + 1

            --pindahkan record yang sudah diatur penomorannya ke
              dalam tabel Dummy yang ke 2
            INSERT INTO #DummyIrisan2 (pengiriman_ke, jalur_ke,
                interval_ke, urutan_ke, jam_awal, jam_akhir, jalur)
            VALUES (@curPengiriman_ke, @db_jalur_ke,
                @db_interval_ke, @no_urutan_pelanggan,
                @curJam_awal, @curJam_akhir, @curJalur)

            FETCH NEXT FROM curIntervalKe INTO
                @curPengiriman_ke, @curJalur_ke, @curInterval_ke,
                @curJam_awal, @curJam_akhir, @curJalur
        END
        CLOSE curIntervalKe
        DEALLOCATE curIntervalKe

    END
end
```

Struktur pencarian irisan antara waktu harus berangkat ke pelanggan selanjutnya dan interval waktu pelanggan yang bersangkutan secara garis besar adalah seperti yang terdapat pada Segmen Program 4.16. Pertama kali yang dilakukan adalah mengambil waktu yang diperlukan untuk melakukan pembongkaran barang pada pelanggan yang bersangkutan. Setelah itu semua interval harus berangkat menuju pelanggan selanjutnya diambil. Masing-masing waktu harus berangkat tersebut dicari perpotongannya dengan masing-masing interval waktu pelanggan yang bersangkutan seperti yang telah dijelaskan sebelumnya pada Segmen Program 4.14. Kemudian dilakukan pengurutan nomor jalur dan nomor interval yang ada seperti yang terdapat pada Segmen Program 4.15.1. dan Segmen Program 4.15.2. Hasil pengurutan yang selama ini ditampung pada tabel sementara, dipindahkan ke tabel lain yang ditampilkan agar dapat diakses oleh prosedur yang lain.

Segmen Program 4.16. Pencarian irisan dengan interval waktu pelanggan secara keseluruhan

```
CREATE PROCEDURE sp_CariIrisanDgnCustAsal4
(   @no_pelanggan_asal int,
    @tanggal_pengiriman datetime,
    @pengiriman_ke int,
    @no_urutan_pelanggan int,
    @no_nota varchar(5)
)AS
DECLARE
--pendeklarasian variabel-variabel yang digunakan

--pembuatan tabel Dummy (hapus tabel Dummy yang lama)

--select waktu bongkar barang di cust asal
exec sp_Cari_Waktu_Bongkar @no_nota, @waktu_bongkar output

SET @db_jalur_ke = 0
SET @db_interval_ke = 0
--select semua record pada Tb_temp_jam_berangkat
--ambil jam harus berangkat dari sub poin asal
DECLARE curJamKeberangkatanTabel CURSOR FOR
SELECT pengiriman_ke, jalur_ke, interval_ke, jam_awal, jam_akhir,
    jalur
FROM Tb_temp_jam_berangkat
WHERE pengiriman_ke = @pengiriman_ke and urutan_ke =
    @no_urutan_pelanggan
ORDER BY pengiriman_ke,jalur_ke,interval_ke
OPEN curJamKeberangkatanTabel
FETCH NEXT FROM curJamKeberangkatanTabel INTO
    @curPengiriman_ke, @curJalur_ke, @curInterval_ke,
    @curJam_awal, @curJam_akhir, @curJalur

WHILE @@FETCH_STATUS = 0
BEGIN
    --cari semua range pelanggan asal (Segmen Program 4.14.)

    FETCH NEXT FROM curJamKeberangkatanTabel INTO
    @curPengiriman_ke, @curJalur_ke, @curInterval_ke,
    @curJam_awal, @curJam_akhir, @curJalur
END
CLOSE curJamKeberangkatanTabel
DEALLOCATE curJamKeberangkatanTabel

--cari termasuk alternatif jalur dan pilihan interval ke berapa
   (Segmen Program 4.15.1. dan Segmen Program 4.15.2.)

--hapus isi Tb_temp_jam_berangkat
delete from Tb_temp_jam_berangkat
where urutan_ke = @no_urutan_pelanggan

--pindahkan isi #DummyIrisan2 ke Tb_temp_jam_berangkat
INSERT INTO Tb_temp_jam_berangkat
select * from #DummyIrisan2
GO
```

4.3.4. Pencarian Jam Keberangkatan Berdasarkan Interval Waktu Keberangkatan

Sewaktu alternatif jalur dan interval keberangkatan yang ada disimpan dalam *database*, waktu tempuh dari masing-masing jalur belum dapat diketahui. Hal ini disebabkan karena dibutuhkan waktu tempuh yang berbeda apabila mobil melewati suatu jalur pada waktu keberangkatan yang berbeda. Waktu tempuh dapat dihitung apabila waktu keberangkatan bukanlah suatu interval melainkan suatu waktu tertentu.

Perangkat lunak mengambil jam-jam yang berada di antara batas awal interval dan batas akhir interval serta batas awal dan akhir interval tersebut. Perangkat lunak nantinya juga mencari waktu tempuh apabila sebuah jalur dilewati mobil dengan waktu keberangkatan adalah jam-jam tersebut. *Listing* program untuk mendapatkan jam-jam keberangkatan tersebut adalah sebagai berikut.

Segmen Program 4.17. Pencarian jam antara batas awal dan akhir interval

```
--cari semua jam yang berada di antara awal dan akhir interval
DECLARE curDaftarJamYgDipilih SCROLL CURSOR FOR
SELECT jam
FROM daftar_jam
WHERE jam between cast(@curJamAwal as datetime) and
      cast(@curJamAkhir as datetime)
ORDER BY no_urut
OPEN curDaftarJamYgDipilih
--cari jam awal & jam akhir keberangkatan
fetch First from curDaftarJamYgDipilih into @JamAwal
fetch LAST  from curDaftarJamYgDipilih into @JamAkhir

--apabila batas awal interval tidak termasuk dalam daftar jam maka
  cari waktu tempuh bila berangkat pada batas awal interval
if(@curJamAwal <> @JamAwal) begin
    exec sp_CariWaktuTempuh2 @curJamAwal, @curJalur,
         @curUrutanNota, @tanggal_pengiriman, @curJalurKe,
         @waktu_tempuh output
    end
--cursor dikembalikan pada awal record
FETCH FIRST FROM curDaftarJamYgDipilih INTO @curJam
WHILE @@FETCH_STATUS = 0
BEGIN
--cari waktu tempuh dari semua jam diantara batas awal interval
  dan batas akhir interval keberangkatan
sp_CariWaktuTempuh2 @curJamAwal, @curJalur, @curUrutanNota,
     @tanggal_pengiriman, @curJalurKe, @waktu_tempuh output
FETCH NEXT FROM curDaftarJamYgDipilih INTO @curJam
END
CLOSE curDaftarJamYgDipilih
DEALLOCATE curDaftarJamYgDipilih
```

Segmen Program 4.17. itulah digunakan yang mencari jam-jam yang terdapat pada daftar jam dimana berada dalam batas awal dan batas akhir interval. Masing-masing jam tersebut dicari waktu tempuhnya pada prosedur yang berbeda. Hal ini dilakukan seperti yang telah dijelaskan sebelumnya karena pada jam keberangkatan yang berbeda, waktu tempuh total untuk pengiriman barang dapat berbeda pula.

Segmen Program 4.18. Pengambilan interval dari semua jalur yang diperoleh

```
CREATE PROCEDURE sp_cari_waktu_tempuh_dr_jam_berangkat3
(   @pengiriman_ke int,
    @tanggal_pengiriman datetime,
    @urutan_no_nota varchar(1000)
)AS

--pendeklarasian variabel yang digunakan
--cursor untuk mengambil semua alternatif jalur yang ada
DECLARE curJalurKe INSENSITIVE CURSOR FOR
SELECT jalur_ke,interval_ke,jam_awal,jam_akhir,jalur,urutan_nota
FROM Tb_temp_jam_berangkat,daftar_urutan_pengiriman
WHERE pengiriman_ke=@pengiriman_ke and
Tb_temp_jam_berangkat.urutan_ke =
daftar_urutan_pengiriman.urutan_ke and urutan_nota =
@urutan_no_nota

OPEN curJalurKe
FETCH NEXT FROM curJalurKe INTO
    @curJalurKe, @curIntervalKe, @curJamAwal, @curJamAkhir,
    @curJalur, @curUrutanNota

WHILE @@FETCH_STATUS = 0
BEGIN
    --cari semua jam yang berada di antara awal interval dan akhir
      interval (Segmen Program 4.17.)

    --bila batas akhir interval tidak termasuk dalam daftar jam
      dan batas akhir interval tidak sama dgn batas awal interval
      maka cari waktu tempuh bila berangkat pada batas akhir
      interval
    if ((@curJamAkhir<>@JamAkhir) and (@curJamAkhir<>@curJamAwal))
        begin
        exec sp_CariWaktuTempuh2 @curJamAkhir, @curJalur,
        @curUrutanNota, @tanggal_pengiriman, @curJalurKe,
        @waktu_tempuh output
        end

FETCH NEXT FROM curJalurKe INTO
    @curJalurKe, @curIntervalKe, @curJamAwal, @curJamAkhir,
    @curJalur, @curUrutanNota
END
CLOSE curJalurKe
DEALLOCATE curJalurKe
GO
```

Pencarian daftar jam yang termasuk dalam sebuah interval tersebut tidak hanya dilakukan dalam sebuah interval saja melainkan dilakukan untuk semua jalur pengiriman dan semua interval keberangkatan dari gudang yang telah diperoleh. *Listing* program untuk pencarian jam-jam keberangkatan untuk semua jalur dan semua interval ini seperti yang terdapat pada Segmen Program 4.19.

### 4.3.5. Pencarian Waktu Tempuh Total Dari Jalur Pengiriman dan Waktu Keberangkatan yang Telah Ditentukan

Setelah menentukan waktu keberangkatan dan sebuah jalur yang dilalui, perangkat lunak menghitung waktu tempuh dari waktu perjalanan dan waktu yang dibutuhkan untuk menurunkan barang di tempat pelanggan yang mendapat pengiriman barang. *Listing* program selengkapnya adalah sebagai berikut.

Pada penghitungan waktu tempuh total dari sebuah jalur, tentunya akan dibutuhkan waktu untuk melakukan perjalanan atau pembongkaran barang di tempat pelanggan. Waktu-waktu yang dibutuhkan tersebut akan ditambahkan untuk mencari total waktu pengiriman barang melalui jalur tersebut. Apabila dilakukan perjalanan dari sebuah poin ke poin yang lain maka seperti yang terdapat pada Segmen Program 4.19, perangkat lunak mencari waktu mulai memasuki jalan tersbeut terlebih dahulu yang didapatkan dari penambahan antara waktu berangkat dan waktu tempuh total sementara. Dari informasi tersebut, prosedur lain akan mencari waktu yang dibutuhkan untuk melalui jalan tersebut. Setelah itu, waktu total sementara akan ditambahkan dengan waktu perjalanan yang baru saja didapatkan untuk menghitung waktu tempuh seluruhnya.

Segmen Program 4.19. Penghitungan waktu perjalanan dari satu poin ke poin lain

```
--waktu perjalanan pengiriman barang
SET @temp_waktu_sekarang = convert(varchar(5),
    cast(@waktu_berangkat as datetime)+cast(@waktu_tempuh as
    datetime),8)

exec cr_wkt_tempuh_dr_param_jam_keberangkatan
     @temp_waktu_sekarang, @no_grup, @no_poin_asal, @no_grup,
     @no_poin_tujuan, @temp_waktu_perjalanan output

SET @waktu_tempuh = convert(varchar(5), cast(@waktu_tempuh as
    datetime) + cast(@temp_waktu_perjalanan as datetime) ,8)
```

Selain waktu perjalanan, faktor lain yang diperlukan untuk menghitung waktu tempuh total adalah waktu pembongkaran barang di tempat pelanggan. Seperti yang terdapat pada Segmen Program 4.20, langkah yang dilakukan pertama kali adalah mencari nomor pelanggan yang mendapat pengiriman barang. Dari nomor pelanggan tersebut diperoleh waktu bongkar barang di tempat pelanggan yang bersangkutan dari tabel bongkar barang. Waktu bongkar barang tersebut kemudian ditambahkan pada waktu tempuh total.

Segmen Program 4.20. Penghitungan waktu bongkar barang di tempat pelanggan

```
--waktu bongkar barang di tempat pelanggan
--no poin asal sama dengan poin tujuan kecuali pengiriman pada
  pelanggan terakhir

SELECT @temp_pos_nota = CHARINDEX( '|', @temp_urutan_nota)

SET @temp_no_nota = LEFT(@temp_urutan_nota, @temp_pos_nota-1)

SET @temp_urutan_nota = RIGHT(@temp_urutan_nota,
    LEN(@temp_urutan_nota)-@temp_pos_nota)

EXEC sp_Cari_Waktu_Bongkar @temp_no_nota, @temp_waktu_bongkar
                  output

--tambahkan waktu tempuh dgn wkt bongkar barang di tempat
                    pelanggan
SET @waktu_tempuh = convert(varchar(5), cast(@waktu_tempuh as
    datetime) + cast(@temp_waktu_bongkar as datetime) ,8)
```

Secara keseluruhan, alur dari pencarian waktu tempuh pada suatu jalur pengiriman dan pada suatu jam keberangkatan adalah seperti yang terdapat pada Segmen Program 4.21. Pada jalur yang diterima oleh prosedur tersebut, akan dilakukan pencarian nomor poin asal dan nomor poin tujuan hingga semua poin pada jalur tersebut telah diproses. Setelah menemukan nomor poin asal dan nomor poin tujuan tersebut, dilakukan pencarian waktu perjalanan seperti yang terdapat pada Segmen Program 4.19. untuk ditambahkan pada waktu tempuh total. Apabila nomor poin asal dan nomor poin tujuan sama, maka waktu yang dicari bukanlah waktu perjalanan melainkan waktu penurunan barang di tempat pelanggan. Pada akhir pencarian waktu tempuh tersebut juga dilakukan pencarian waktu bongkar barang di tempat pelanggan yang terakhir seperti pada Segmen Program 4.20.

Segmen Program 4.21. Penghitungan waktu tempuh

```
CREATE PROCEDURE sp_CariWaktuTempuh2
(   @waktu_berangkat varchar(5), @jalur varchar(1000),
    @urutan_no_nota varchar(1000), @tanggal_pengiriman datetime,
    @jalur_ke int, @waktu_tempuh varchar(5) output
)AS
--contoh @jalur = '2|5|4|2|3|3|1|4|2|2|1|'--angka 2 ditulis 2x
  artinya ada perhentian
--pendeklarasian variabel yang digunakan
SET @no_grup = -1 SET @no_poin_asal = -1  SET @no_poin_tujuan = -1
SET @waktu_tempuh = '00:00' SET @temp_urutan_nota =
@urutan_no_nota
--cari jumlah karakter jalur
SET @JumlahCharJalur = len(@jalur)
SET @temp_jalur = @jalur
SELECT @temp_pos = CHARINDEX( '|', @temp_jalur)
SET @no_grup = LEFT(@temp_jalur,@temp_pos-1)
SET @temp_jalur = RIGHT(@temp_jalur,LEN(@temp_jalur) - @temp_pos)
--ambil no poin asal dan tujuan selama @temp_jalur belum habis
WHILE (LEN(@temp_jalur)>0)
    begin
    --ambil no poin asal
    SELECT @temp_pos = CHARINDEX( '|', @temp_jalur)
    SET @no_poin_asal = LEFT(@temp_jalur,@temp_pos-1)
    SET @temp_jalur =RIGHT(@temp_jalur,LEN(@temp_jalur)-@temp_pos)
    --ambil no poin tujuan
    if(LEN(@temp_jalur)>0) begin
        SELECT @temp_pos = CHARINDEX( '|', @temp_jalur)
        SET @no_poin_tujuan = LEFT(@temp_jalur,@temp_pos-1)
        end else
            SET @no_poin_tujuan = -1
    --cari apakah belum habis @temp_jalurnya
    if(@no_poin_asal>0)and(@no_poin_tujuan>0) begin
        SET @temp_waktu_sekarang = convert(varchar(5),
            cast(@waktu_ berangkat as datetime)+cast(@waktu_tempuh
            as datetime),8)
        if(@no_poin_asal <> @no_poin_tujuan) begin
            --waktu perjalanan pengiriman brg(Segmen Program 4.19)
            end else begin--waktu bongkar brg di tempat pelanggan
                --no poin asal sama dengan poin tujuan seperti yang
                  terdapat pada Segmen Program 4.20.)
                end
        end
    end
SET @temp_waktu_sekarang = convert(varchar(5),
  cast(@waktu_berangkat as datetime)+cast(@waktu_tempuh as
  datetime),8)
--tambahkan waktu tempuh total dengan waktu bongkar barang di
  pelanggan terakhir (Segmen Program 4.20.)
SET @urutan_ke = (SELECT urutan_ke FROM daftar_urutan_pelanggan
    WHERE urutan_pelanggan = @urutan_pelanggan)
--pindahkan kemungkinan jalur dan jam berangkat yang dapat
    digunakan ke dalam tabel pilihan jalur
INSERT INTO pilihan_jalur (no_grup,tanggal_pengiriman,jalur_ke,
    wkt_berangkat,urutan_ke,wkt_tempuh,jalur)
VALUES (@no_grup,tanggal_pengiriman,@jalur_ke, @waktu_berangkat,
    @urutan_ke,@waktu_tempuh,@jalur)
GO
```

4.3.6. Pencarian Waktu Tempuh Perjalanan Pada Waktu Keberangkatan Tertentu

Perangkat lunak mencari waktu tempuh untuk melewati sebuah jalan dengan waktu keberangkatan tertentu. Waktu keberangkatan melewati sebuah jalan tidak selalu berada dalam daftar jam. Apabila waktu keberangkatan berada dalam daftar jam maka perangkat lunak hanya perlu mencari waktu tempuh melewati jalan tersebut dari *database*.

Namun apabila waktu keberangkatan tersebut tidak termasuk dalam daftar jam maka diambil waktu tempuh melewati jalan tersebut pada jam sebelum dan sesudah jam keberangkatan. Kemudian untuk mendapatkan waktu tempuh pada waktu keberangkatan tersebut, perangkat lunak mencari rata-rata waktu tempuh dari waktu tempuh sebelum dan sesudah waktu keberangkatan tersebut. *Listing* program untuk mencari waktu tempuh tersebut dengan waktu keberangkatan yang telah ditentukan dapat dilihat pada Segmen Program 4.22.1. dan Segmen Program 4.22.2.

Segmen Program 4.22.1. Penghitungan waktu tempuh melewati suatu jalan

```
CREATE PROCEDURE cr_wkt_tempuh_dr_param_jam_keberangkatan
(   @WaktuKeberangkatan     varchar(5),     @no_grup_asal     int,
    @no_poin_asal int, @no_grup_tujuan int, @no_poin_tujuan int,
    @WaktuTempuh varchar(5) output
)AS
--inisialisasi dan deklarasi variabel
SET @JumlahJam = 0
--pencarian apakah wkt keberangkatan tersebut ada pada daftar jam
DECLARE curDaftarJam SCROLL CURSOR FOR
SELECT jam
FROM daftar_jam WHERE jam>=@WaktuKeberangkatan
ORDER BY no_urut ASC
OPEN curDaftarJam

--ambil record awal dari daftar jam yang lebih besar atau sama
  dengan jam keberangkatan
FETCH FIRST from curDaftarJam into @curJam
SET @JamAkhir = @curJam
SET @JumlahJam = @JumlahJam + 1
close curDaftarJam
deallocate curDaftarJam

if(@JamAkhir <> @WaktuKeberangkatan) begin
    --apabila jam akhir tidak sama dengan waktu keberangkatan maka
      waktu keberangkatan tersebut tidak terdapat pada daftar jam
    --cari jam awal atau jam sebelum waktu keberangkatan tersebut
    DECLARE curDaftarJam SCROLL CURSOR FOR
    SELECT jam FROM daftar_jam WHERE jam<=@WaktuKeberangkatan
    ORDER BY no_urut DESC
```

Segmen Program 4.22.2. Penghitungan waktu tempuh melewati suatu jalan (lanjutan)

```
    OPEN curDaftarJam
    FETCH FIRST from curDaftarJam into @curJam
    SET @JamAwal = @curJam
    SET @JumlahJam = @JumlahJam + 1
    close curDaftarJam
    deallocate curDaftarJam
    end
if(@JumlahJam = 1) begin
    --apabila waktu keberangkatan termasuk dalam daftar jam
    --ambil waktu tempuh melewati poin asal ke poin tujuan pada
      waktu keberangkatan tersebut
    SET @WaktuTempuhAkhir = (SELECT jalan.waktu_tempuh
        FROM jalan INNER JOIN
            daftar_jam ON jalan.no_urut = daftar_jam.no_urut
        WHERE (no_grup_asal = @no_grup_asal) AND (no_grup_tujuan =
        @no_grup_tujuan) AND (no_poin_asal = @no_poin_asal) AND
        (no_poin_tujuan = @no_poin_tujuan) AND (jalan.jam =
        @JamAkhir))
    SET @WaktuTempuh = @WaktuTempuhAkhir
    end else if(@JumlahJam = 2) begin
        --bila jam berangkat tersebut tdk termasuk dlm daftar jam
        --cari waktu tempuh bila melewati jalan tersebut pada jam
          sesudah jam keberangkatan
        SET @WaktuTempuhAkhir = (SELECT jalan.waktu_tempuh
        FROM jalan INNER JOIN
            daftar_jam ON jalan.no_urut = daftar_jam.no_urut
        WHERE (no_grup_asal = @no_grup_asal) AND (no_poin_asal =
            @no_poin_asal) AND (no_poin_tujuan = @no_poin_tujuan)
            AND (jalan.jam = @JamAkhir))
        --cari waktu tempuh bila melewati jalan tersebut pada jam
          sebelum jam keberangkatan
        SET @WaktuTempuhAwal = (SELECT jalan.waktu_tempuh
        FROM jalan INNER JOIN
            daftar_jam ON jalan.no_urut = daftar_jam.no_urut
        WHERE (no_grup_asal = @no_grup_asal) AND (no_grup_tujuan =
            @no_grup_tujuan) AND (no_poin_asal = @no_poin_asal) AND
            (no_poin_tujuan=@no_poin_tujuan)AND(jalan.jam=
            @JamAwal))
        --cari rata2 antara waktu tempuh awal dan akhir
        exec convert_time_to_int @WaktuTempuhAkhir,
            @WaktuTempuhAkhir_int output
        exec convert_time_to_int @WaktuTempuhAwal,
            @WaktuTempuhAwal_int output
        exec convert_time_to_int @WaktuKeberangkatan,
            @WaktuKeberangkatan_int output
        exec convert_time_to_int @JamAwal,@JamAwal_int output
        exec convert_time_to_int @JamAkhir,@JamAkhir_int output
        SET @WaktuTempuh_int = @WaktuTempuhAwal_int +
            ((@WaktuTempuhAkhir_int - @WaktuTempuhAwal_int) *
            (@WaktuKeberangkatan_int - @JamAwal_int) /
            (@JamAkhir_int - @JamAwal_int))
        exec convert_int_to_time_str @WaktuTempuh_int,
            @WaktuTempuh output
        end
GO
```

## 4.4 Implementasi Pengubahan Jam Dari Bentuk Waktu (hh:mm) ke Bentuk Menit

Perangkat lunak menggunakan tipe data string untuk menyimpan waktu-waktu yang diperlukan. Namun ada kalanya perangkat lunak harus menggunakan operasi matematika untuk menghitung operasi matematika antara kedua buah jam. Operasi matematika berupa pengurangan atau penambahan dapat dilakukan dengan menggunakan perintah *cast* atau *convert* yang tersedia dalam SQL Server. Namun untuk proses pembagian atau perkalian, dua buah waktu tersebut harus diubah dalam format menit terlebih dahulu agar dapat diproses dengan operasi pembagian atau perkalian.

### 4.4.1. Pengubahan dari Bentuk Waktu (hh:mm) ke Bentuk Menit

Pengubahan menjadi bentuk menit adalah agar waktu tersebut dapat disimpan pada variabel dengan tipe data integer. Proses pengubahan format waktu hh:mm ke dalam bentuk menit adalah seperti yang terdapat pada Segmen Program 4.23.

Segmen Program 4.23. Pengubahan dari bentuk waktu (hh:mm) ke bentuk menit

```
CREATE PROCEDURE convert_time_to_int
(   @waktu varchar(5),
    --output
    @waktu_int int output
)AS

DECLARE
    @jam varchar(2), @menit varchar(2), @temp_pos int
    --pencarian posisi ':' pada format waktu yang diproses
    SELECT @temp_pos = CHARINDEX( ':', @waktu)

    --ambil jumlah jam pada waktu tersebut dari posisi 1 hingga
      posisi sebelum ':'
    SET @jam = LEFT(@waktu,@temp_pos-1)

    --ambil jumlah jam pada waktu tersebut dari posisi setelah ':'
      hingga posisi terakhir
    SET @menit = RIGHT(@waktu,2)

    --jumlah jam dikalikan dengan 60 dan ditambahkan dengan menit
      sekarang sehingga didapatkan waktu dalam bentuk menit
    SET @waktu_int = cast(@jam as int)*60 + cast(@menit as int)
    return 0
GO
```

4.4.2. Pengubahan dari Bentuk Menit ke Bentuk Waktu (hh:mm)

Setelah melakukan operasi matematika pada waktu dalam bentuk menit, waktu tersebut harus dikembalikan menjadi format waktu (hh:mm). Hal ini dikarenakan format waktu itulah yang digunakan oleh perangkat lunak untuk proses lebih lanjut. Proses pengubahan format waktu dalam bentuk menit ke format waktu hh:mm adalah dengan *listing* program berikut.

Segmen Program 4.24. Pengubahan dari bentuk menit ke bentuk waktu (hh:mm)

```
CREATE PROCEDURE convert_int_to_time_str
(   @waktu_int int,
    --output
    @waktu varchar(5) output
)AS

DECLARE
    @jam_int int, @menit_int int, @jam varchar(2),
    @menit varchar(2)

    --pencarian jam dari waktu dalam bentuk integer
    SET @jam_int = FLOOR(@waktu_int/60)
    --ambil sisa pembagian antara waktu dengan 60 untuk
      mendapatkan menit
    SET @menit_int = @waktu_int % 60
    --ubah menit dan jam dari bentuk integer ke bentuk string
    SET @jam = cast(@jam_int as varchar)
    SET @menit = cast(@menit_int as varchar)
    --Apabila panjang jam atau panjang menit kurang dari 2 maka
      tambahkan angka 0 di bagian depan
    if(LEN(@jam)<2)
        SET @jam = '0' + @jam
    if(LEN(@menit)<2)
        SET @menit = '0' + @menit
    --Gabungkan jam dan menit menjadi sebuah format waktu (hh:mm)
    SET @waktu = @jam + ':' + @menit
GO
```

## 4.5 Penulisan Pada *File*

Untuk membuat *file*, seperti *logfile* dalam melakukan penghitungan pengiriman barang maka dibutuhkan fungsi pada SQL Server untuk menulis penghitungan bersamaan dengan dieksekusinya sebuah prosedur pada SQL Server untuk melakukan penghitungan tersebut. Penulisan penghitungan tersebut berfungsi untuk memberikan penjelasan apakah memang jalan yang diambil tersebut merupakan jalan yang paling singkat. Proses penulisan penghitungan

jarak tersingkat dalam melakukan pengiriman barang adalah dengan *listing* program berikut.

Segmen Program 4.25. Penulisan pada *file*

```
CREATE PROCEDURE WriteToFile
(   @filename varchar(1000),
    @kalimat varchar(2000)
)AS

DECLARE @temp_str2 varchar(2000)

SET @kalimat = REPLACE(@kalimat,'|','_')
SET @kalimat = REPLACE(@kalimat,'->',' ke ')
SET @temp_str2='echo '+@kalimat+' >> '+@filename

exec master..xp_cmdshell @temp_str2

GO
```