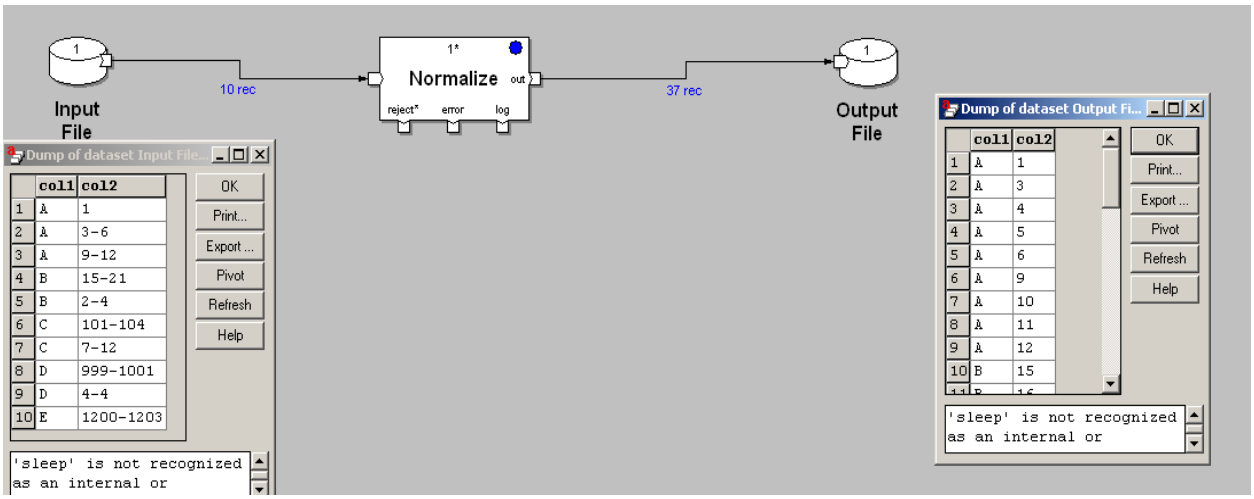


INPUT DML:

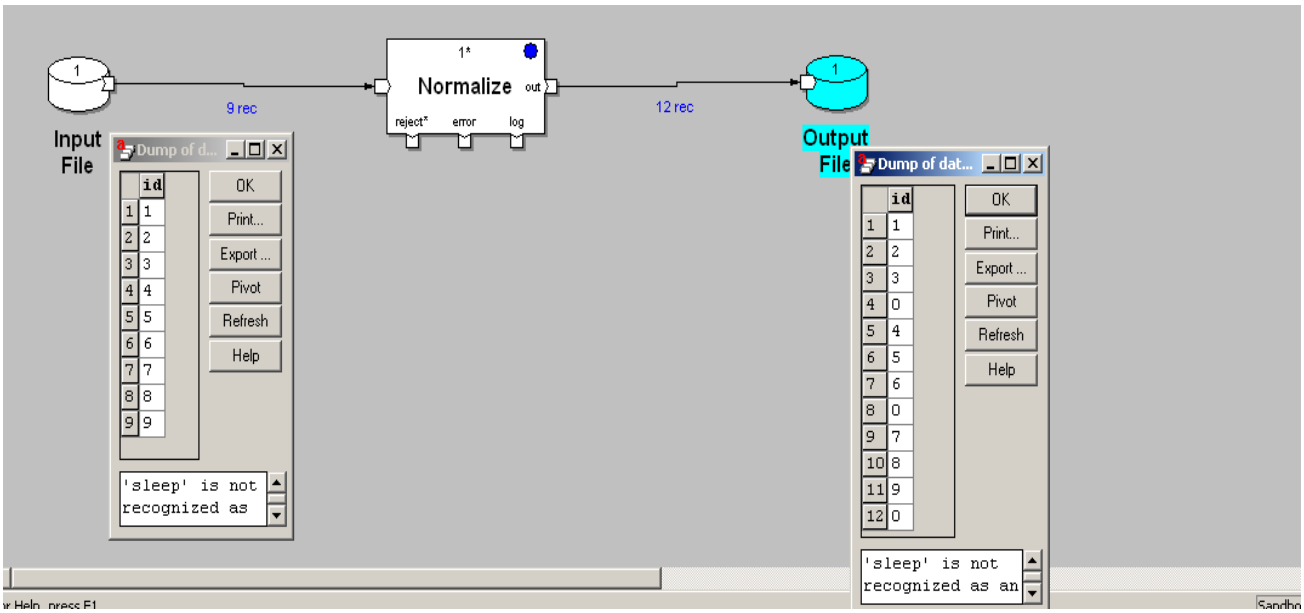
```
Record

  string("|") col1;
  string ("\r\n") col2;
end;
```

OUTPUT DML:

```
record
  string("") col1;
  string("") col2;
end;
```

TRANSFORM LOGIC :

```
out::length(in) =

begin
  out :: if(string_index(in.col2, "-")>0)
          ((decimal("")) (string(""))string_substring(in.col2,
string_index(in.col2, "-")+1, string_length(in.col2)-string_index(in.col2, "-
")))-
          ((decimal("")) (string(""))string_substring(in.col2, 1,
string_index(in.col2, "-")-1))+1

else string_length(in.col2);
end;
out::normalize(in, index) =
begin
  out.col1 :: in.col1;
  out.col2 :: if(index==0 && string_index(in.col2, "-")>0)
string_lrtrim((decimal(""))
(string(""))(string_substring(in.col2,1,string_index(in.col2, "-")-1 ))) else
if (index!=0 && string_index(in.col2, "-")>0)
string_lrtrim((decimal(""))
(string(""))(string_substring(in.col2,1,string_index(in.col2, "-")-1
))+index) else
string_lrtrim(in.col2);
end;
```



INPUT DML::

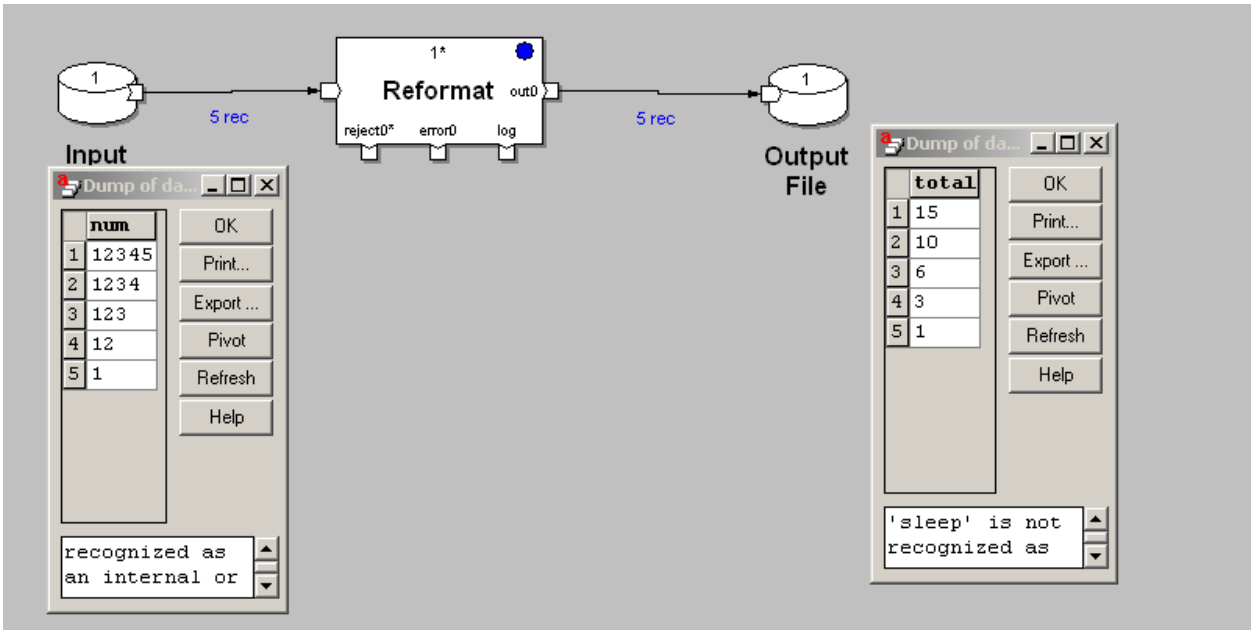
```
record
decimal ("\r\n") id;
end
```

OUTPUT DML::

```
record
  decimal(1) id;
end;
```

TRANSFORM LOGIC ::

```
out::length(in) =
begin
  out :: if (next_in_sequence()%3==0) 2 else 1;
end;


out::normalize(in, index) =
begin
  out.id :: if (index==0) in.id else 0;
end;
```



INPUT DML ::

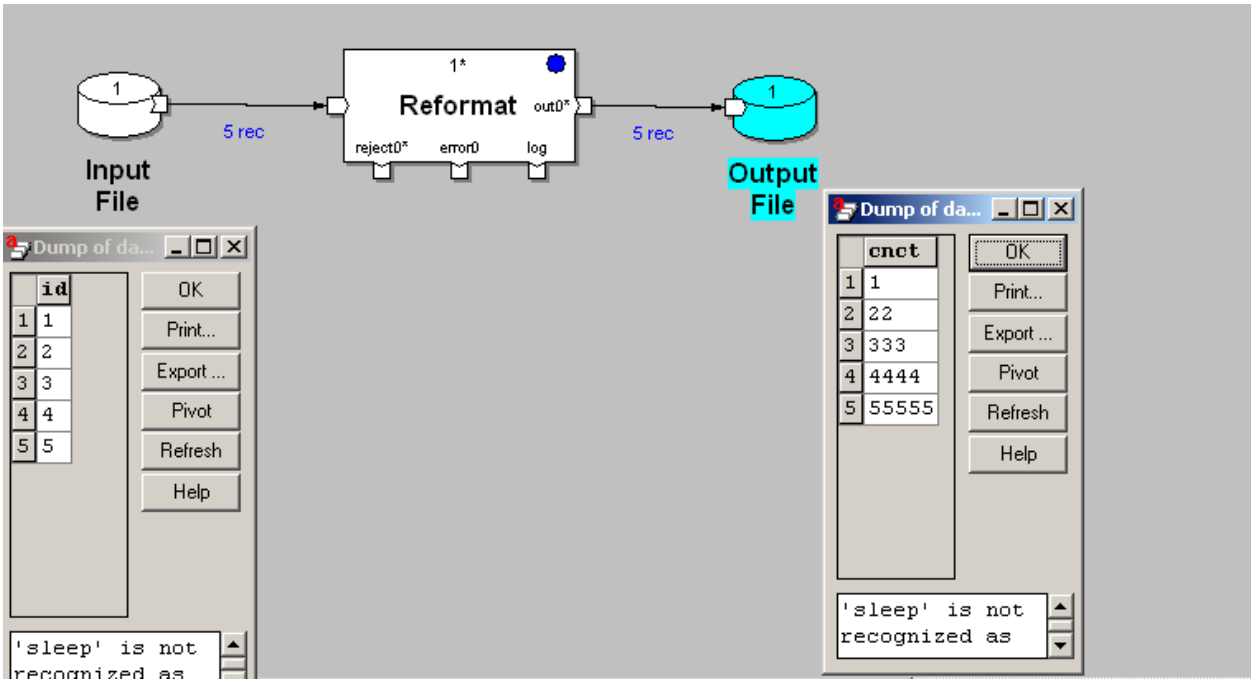
```
record
string ("\r\n") num;
end
```

OUTPUT DML::

```
record
  decimal("") total;
end;
```

TRANSFORM LOGIC ::

```
out::reformat(in) =
begin
  let integer(8) i;
  let decimal("") v_total =0;

for( i, i< length_of(in.num))

v_total= v_total+ (decimal(""))(string(""))string_substring(in.num,i+1, 1);

 out.total :: v_total;

end;
```



INPUT DML ::

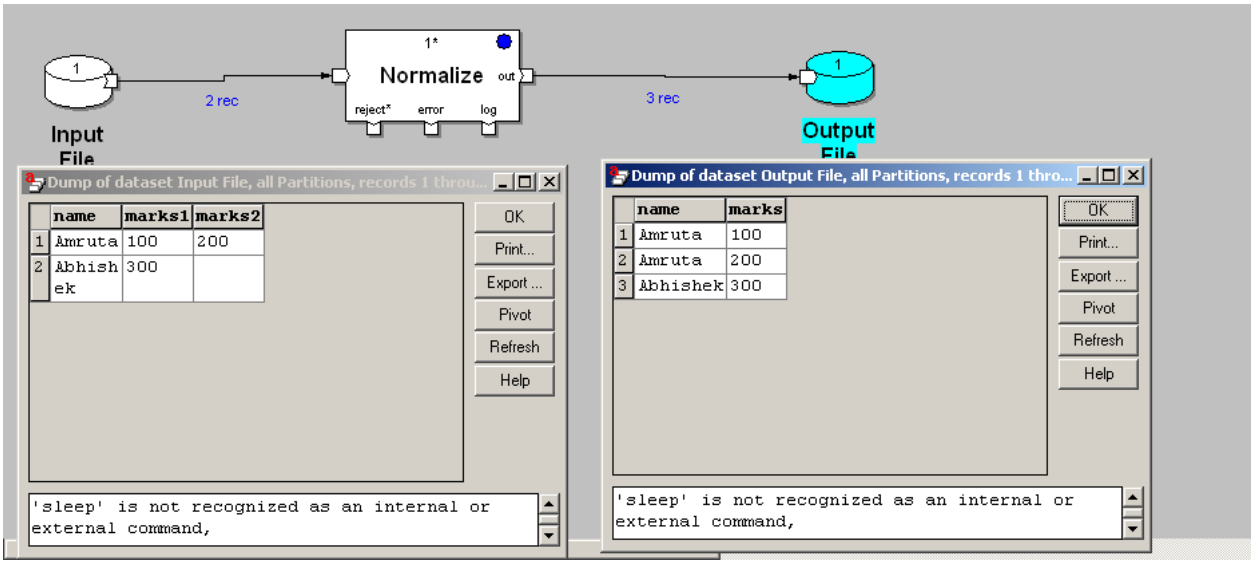
```
record
decimal   ("\r\n") id;
end
```

OUTPUT DML ::

```
record
  decimal("") cnct;
end;
```

TRANSFORM LOGIC ::

```
/*Reformat operation*/
out::reformat(in) =
begin
   let int i;
  let decimal("") v_cnct = "";
    for (i, i < in.id)
    v_cnct=string_concat(v_cnct, in.id);

  out.cnct::v_cnct;

end
```



INPUT DML ::

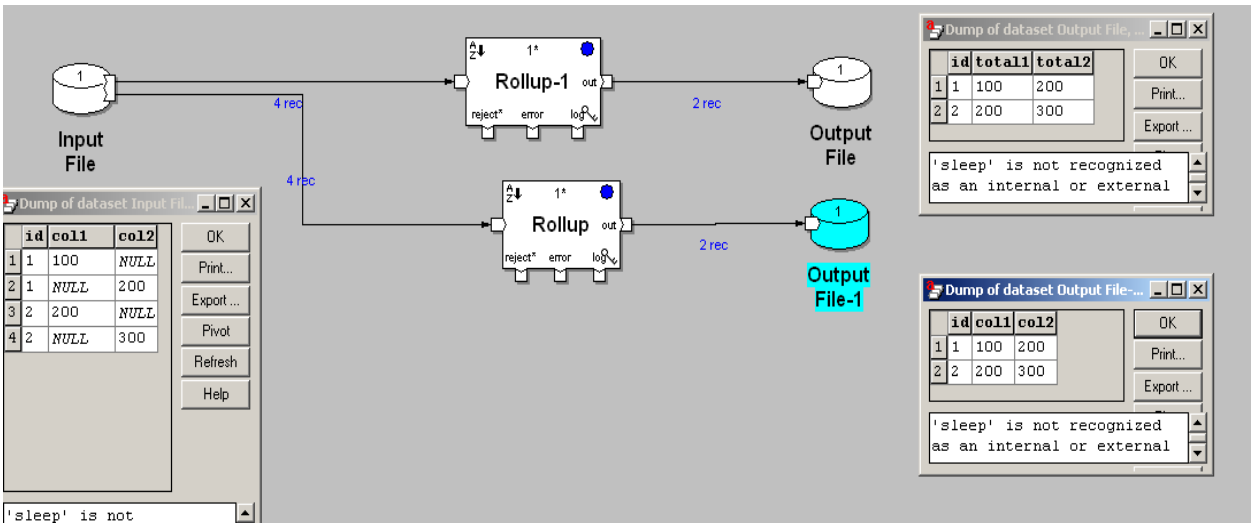
```
record
string (" ") name;
string (",") marks1;
string("\r\n") marks2;
end
```

OUTPUT DML ::

```
record
  string(" ") name;
  ascii string("\0") marks;
end;
```

TRANSFORM LOGIC ::

```
/*Number of records to output*/
out::length(in) =
begin
  out :: 2;
end;

/*Do computation*/
temp::normalize(in, index) =
begin
  temp.name :: in.name;
  temp.marks :: if (index==0) in.marks1 else in.marks2;
     end;

out::output_select(out) =
begin

  out :: !is_blank(out.marks) ;
end;
```



INPUT DML ::

```
record
decimal (" ") id;
decimal (" ") col1=NULL("");
decimal ("\r\n") col2=NULL("");
end
```

OUTPUT DML ::

```
record
  decimal(" ") id;
  decimal(" ") total1 = NULL("");
  decimal("\r\n") total2 = NULL("");
end;
```

TRANSFORM LOGIC ::

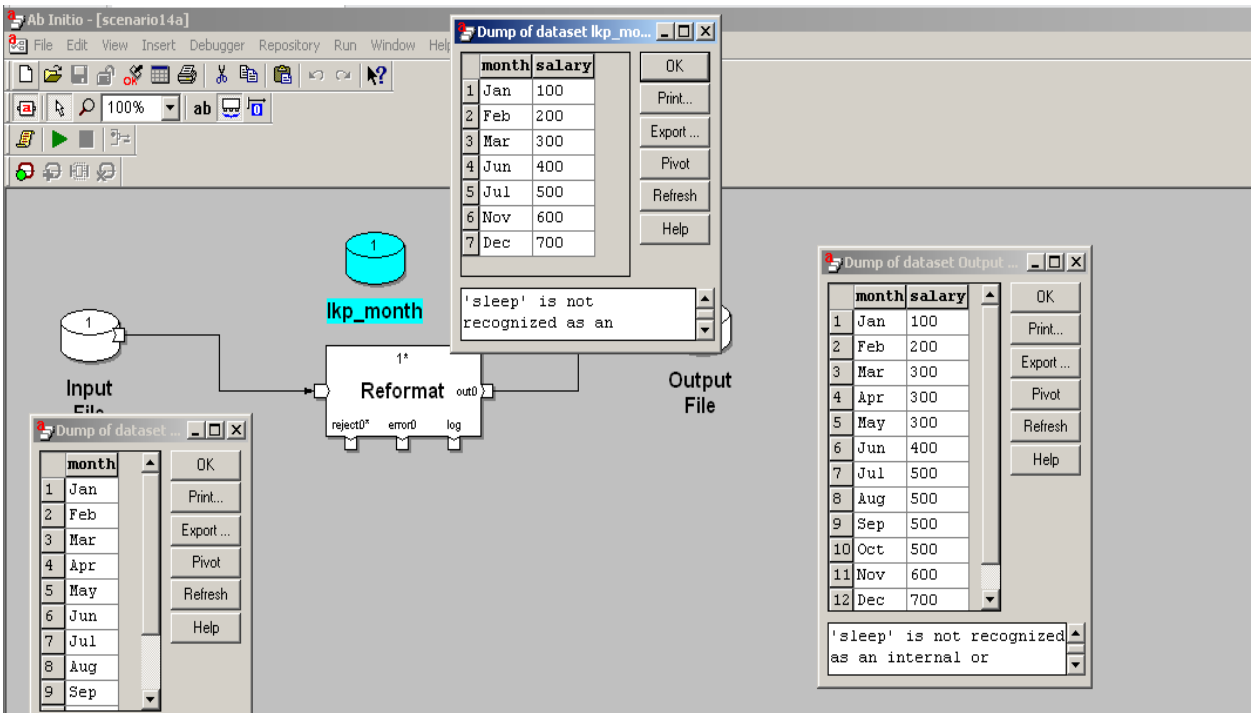
ROLLUP-1 ::

```
type temporary_type = record
  double sum_tmp_0;
  double sum_tmp_1;
end;

out::initialize(in) =
begin
  out.sum_tmp_0 :: 0;  /**GENERATED*'sum(first_defined(in.col1, 0))'*/
  out.sum_tmp_1 :: 0;  /**GENERATED*'sum(first_defined(in.col2, 0))'*/
end;

out::rollup(tmp, in) =
begin
  out.sum_tmp_0 :: tmp.sum_tmp_0 + (first_defined(in.col1, 0));
/**GENERATED*'sum(first_defined(in.col1, 0))'*/
  out.sum_tmp_1 :: tmp.sum_tmp_1 + (first_defined(in.col2, 0));
/**GENERATED*'sum(first_defined(in.col2, 0))'*/
end;

/**/
```

```
out::finalize(tmp, in) =
begin
  out.id :: in.id;  /**GENERATED*'in.id'* */
  out.total1 :: (tmp.sum_tmp_0);  /**GENERATED*'sum(first_defined(in.col1,
0))'* */
  out.total2 :: (tmp.sum_tmp_1);  /**GENERATED*'sum(first_defined(in.col2,
0))'* */
end;
```

ROLLUP ::

```
type temporary_type =
record
  decimal("") v_tmp_col1;
  decimal("") v_tmp_col2;
end;

out::initialize(in) =
begin
  out.v_tmp_col1 :: "";
  out.v_tmp_col2 :: "";
end;


out::rollup(tmp, in) =
begin
  out.v_tmp_col1 :: string_concat((first_defined(in.col1, "")),
tmp.v_tmp_col1);
  out.v_tmp_col2 :: string_concat((first_defined(in.col2, "")),
tmp.v_tmp_col2);
end;


out::finalize(tmp, in) =
begin
  out.id :: in.id;
  out.col1 :: tmp.v_tmp_col1;
  out.col2 :: tmp.v_tmp_col2;
end;
```



INPUT DML ::

```
record
string ("\r\n") month;
end
```

OUTPUT DML ::

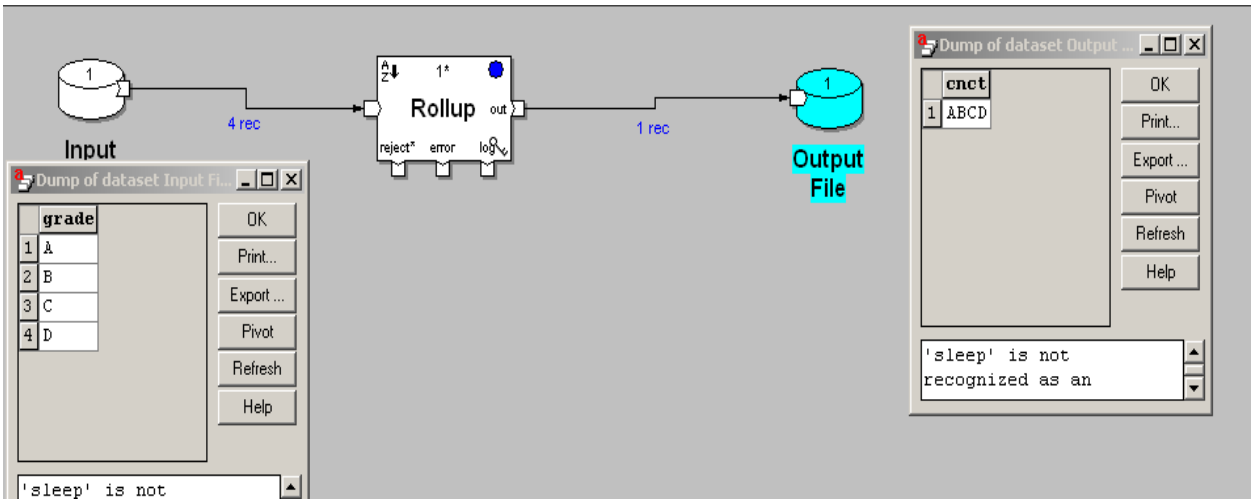
```
record
  string(" ") month;
  decimal("\r\n") salary;
end;
```

LOOKUP DML::

```
record
string(" ") month;
decimal("\r\n") salary;
end
```

TRNASFORM LOGIC ::

```
let decimal("") v_salary =0;

out::reformat(in) =
begin
v_salary= if((first_defined(lookup("lkp_month", in.month) .salary, 0))!=0)
(lookup("lkp_month", in.month) .salary)
 else v_salary;

  out.month :: in.month;
  out.salary :: v_salary;
end;
```



INPUT DML ::

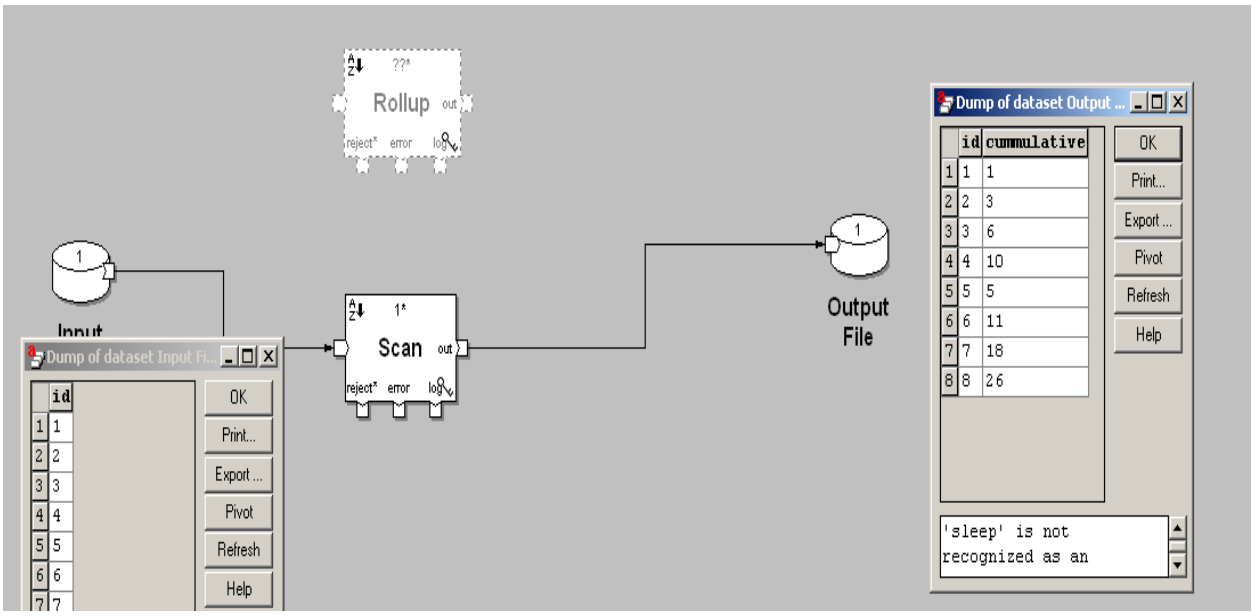
```
record
string ("\r\n") grade;
end
```

OUTPUT DML ::

```
record
  string("") cnct;
end;
```

TRANSFORM LOGIC :: KEY AS {}

```
type temporary_type =
record
  string("") v_cnct;
end;

out::initialize(in) =
begin
  out.v_cnct :: "";
end;


out::rollup(tmp, in) =
begin
  out.v_cnct :: string_concat(tmp.v_cnct, in.grade);
end;


out::finalize(tmp, in) =
begin
  out.cnct :: tmp.v_cnct;
end;
```



INPUT DML :: `record`

```
string("\r\n") id;
end
```

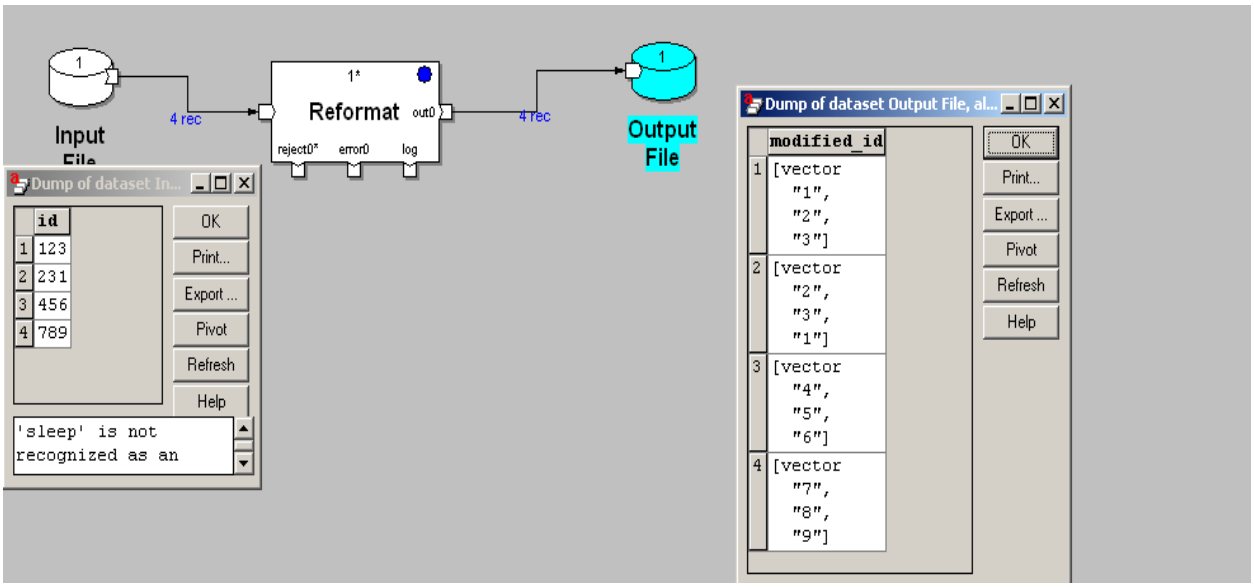
OUTPUT DML :: `record`

```
  string("") id;
  decimal("") cummulative;
end;
```

TRANSFORM LOGIC ::

KEY METHOD :USE KEY CHANGE FUNCTION

```
type temporary_type =
record
  decimal("") v_sum;
            end /* Temporary variable*/;
temp::initialize(in) =
begin
  temp.v_sum :: 0;
end;
temp::scan(temp, in) =
begin
  temp.v_sum :: temp.v_sum+(decimal("")) (string(""))in.id;
end;
out::finalize(temp, in) =
begin
  out.id :: in.id;
  out.cummulative :: temp.v_sum;
end;
out::key_change(in1, in2) =
begin
  out :: next_in_sequence()%4==0;
end;
```



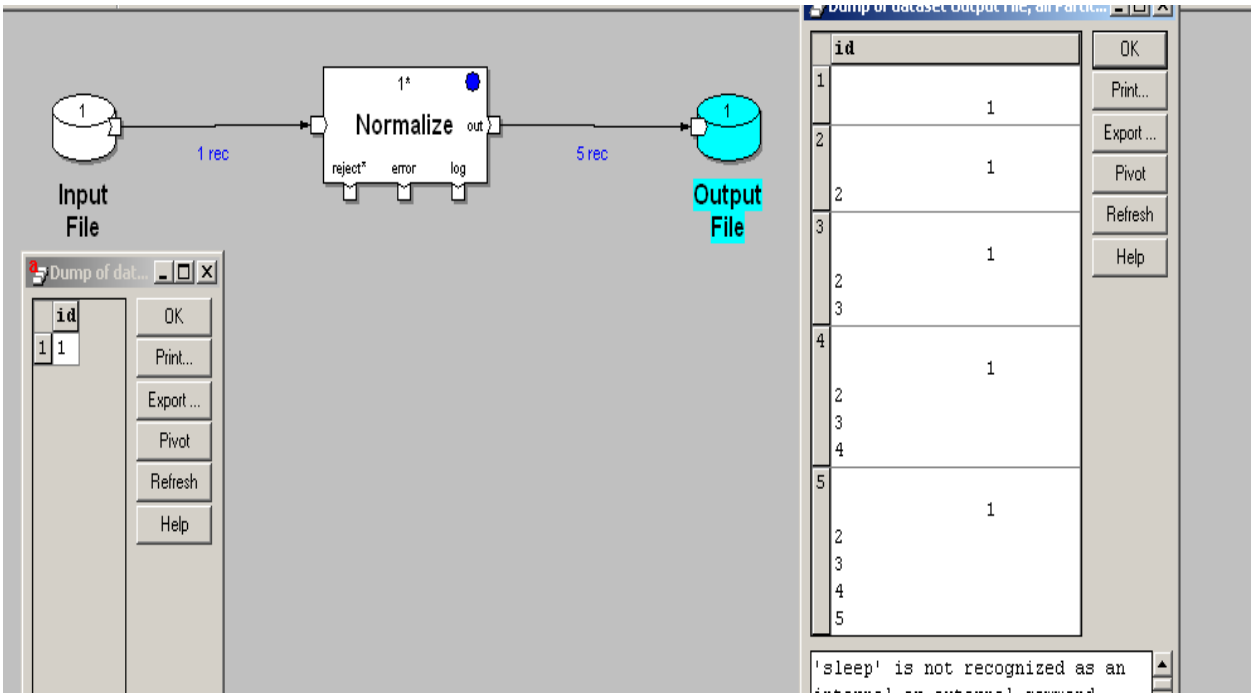
INPUT DML ::

```
record
string("\r\n") id;
end
```

OUTPUT DML ::

```
record
  string("")[3] modified_id;
end;
```

```
aout::reformat(in) =
begin
  out.modified_id :: reinterpret_as(string(1)[3],in.id);
end;
```



INPUT DML :: record

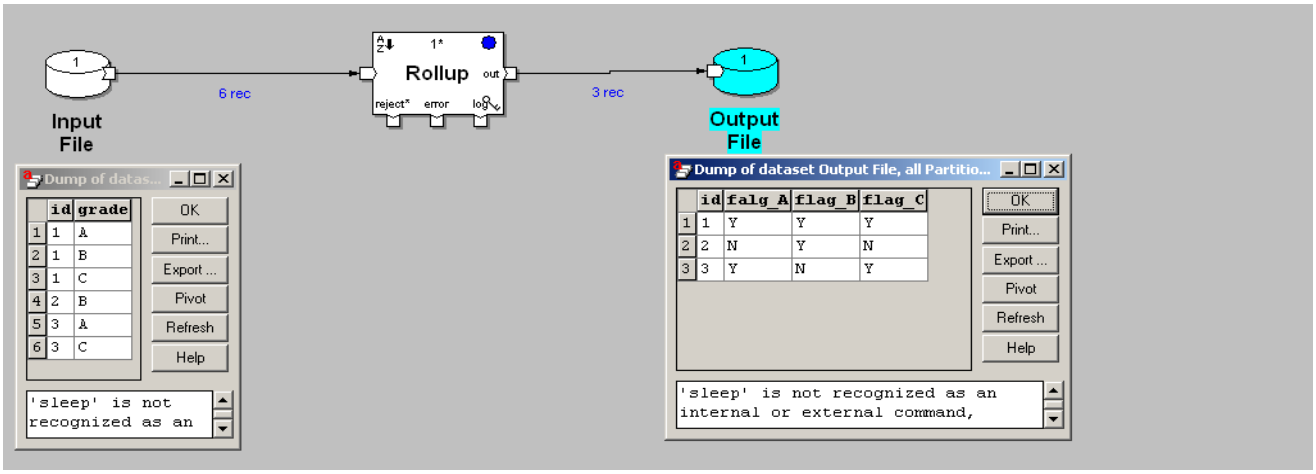
```
string("\r\n") id;
end
```

OUTPUT DML :: record

```
  string("") id;
end;
```

TRANSFORM LOGIC ::

```
out::length(in) =
begin
  out :: 5;
end;
type temporary_type =
record
  string("") v_id;
end /* Temporary variable*/;

temp::initialize(in) =
begin
  temp.v_id :: "";
end;
temp::normalize(temp, in, index) =
begin
        temp.v_id :: string_concat(temp.v_id, index+(decimal(""))
      (string(""))in.id);
end;out::finalize(temp, in) =
begin
  out.id :: temp.v_id;
end;
```



INPUT DML ::

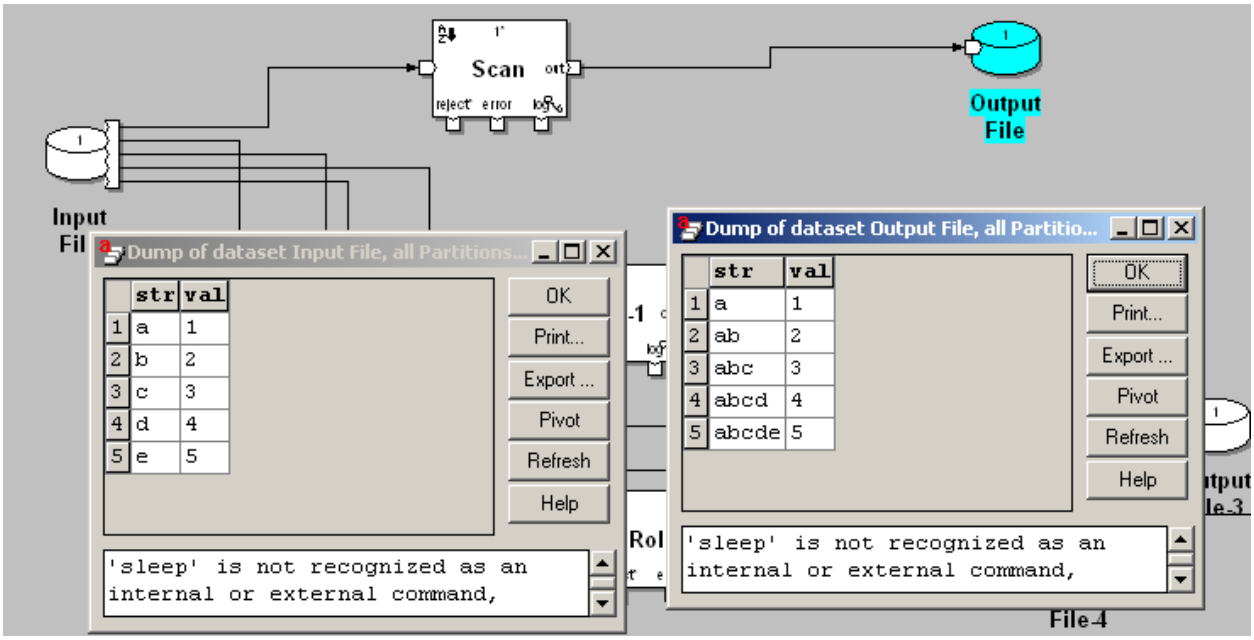
```
record
string (" ") id;
string("\r\n") grade;
end
```

OUTPUT DML ::

```
record
  string(" ") id;
  string("") falg_A;
  string("") flag_B;
  string("") flag_C;
end;
```

TRANSFORM LOGIC :: KEY AS {ID}

```
type temporary_type =
record
  string("") v_type_A;
  string("") v_type_B;
  string("") v_type_C;
end;
   out::initialize(in) =
begin
  out.v_type_A :: "N";
  out.v_type_B :: "N";
  out.v_type_C :: "N";
end;
   out::rollup(tmp, in) =
begin
  out.v_type_A :: if(in.grade=="A") "Y" else tmp.v_type_A;
  out.v_type_B :: if(in.grade=="B") "Y" else tmp.v_type_B;
  out.v_type_C :: if(in.grade=="C") "Y" else tmp.v_type_C;
end;
 out::finalize(tmp, in) =
begin
  out.id :: in.id;
  out.falg_A :: tmp.v_type_A;
  out.flag_B :: tmp.v_type_B;
  out.flag_C :: tmp.v_type_C;
end;
```



INPUT DML :: `record`

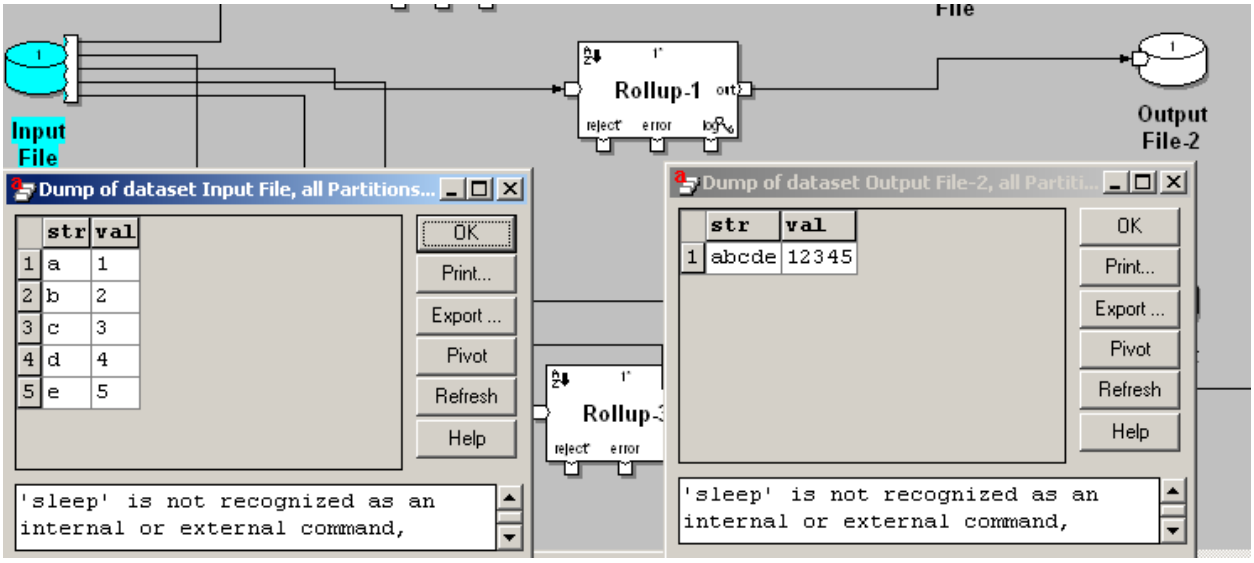
```
string (" ") str;
decimal ("\r\n") val;
end
```

OUTPUT DML :: `record`

```
  string("") str;
  decimal("") val;
end;
```

TRANSFORM LOGIC ::KEY AS {}

```
type temporary_type =
record
  string("") v_str;
end /* Temporary variable*/;

temp::initialize(in) =
begin
  temp.v_str :: "";
end;


temp::scan(temp, in) =
begin
  temp.v_str :: string_concat(temp.v_str,in.str);
end;


out::finalize(temp, in) =
begin
  out.str :: temp.v_str;
  out.val :: in.val;
end;
```

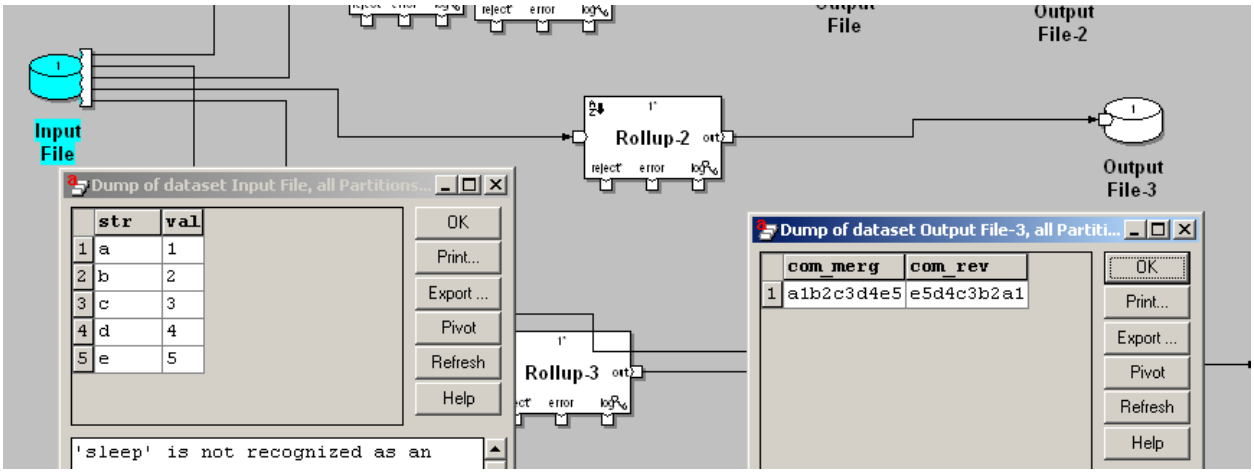


INPUT DML ::
```
record
string (" ") str;
decimal ("\r\n") val;
end
```

OUTPUT DML ::
```
record
  string("") str;
  decimal("") val;
end;
```

TRANSFORM LOGIC :: KEY AS {}

```
type temporary_type =
record
  string("") v_str;
  decimal("") v_val;
end;

out::initialize(in) =
begin
  out.v_str :: "";
  out.v_val :: "";
end;


out::rollup(tmp, in) =
begin
  out.v_str :: string_concat(tmp.v_str,in.str);
  out.v_val :: string_concat(tmp.v_val,in.val);
end;


out::finalize(tmp, in) =
begin
  out.str :: tmp.v_str;
  out.val :: tmp.v_val;
end;
```



INPUT DML :: `record`

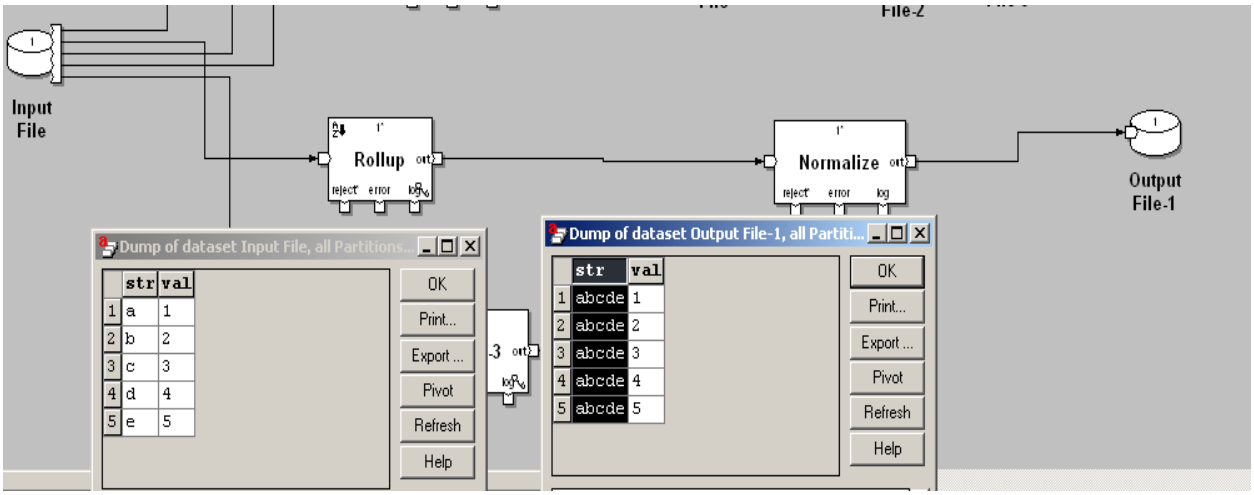
```
string (" ") str;
decimal ("\r\n") val;
end
```

OUTPUT DML :: `record`

```
  string("") com_merg;
  string("") com_rev;
end;
```

TRANSFORM LOGIC :: KEY AS {}

```
type temporary_type =
record
  string("") v_com;
  string("") v_com_rev;
end;

out::initialize(in) =
begin
  out.v_com :: "";
  out.v_com_rev :: "";
end;


out::rollup(tmp, in) =
begin
  out.v_com :: string_concat(tmp.v_com ,in.str, in.val);
  out.v_com_rev :: string_concat(in.str, in.val, tmp.v_com_rev);
end;


out::finalize(tmp, in) =
begin
  out.com_merg :: tmp.v_com;  /**GENERATED*'string_concat(in.str,in.val)'* */
  out.com_rev :: tmp.v_com_rev;
end;
```



INPUT DML ::
```
record
string (" ") str;
decimal ("\r\n") val;
end
```

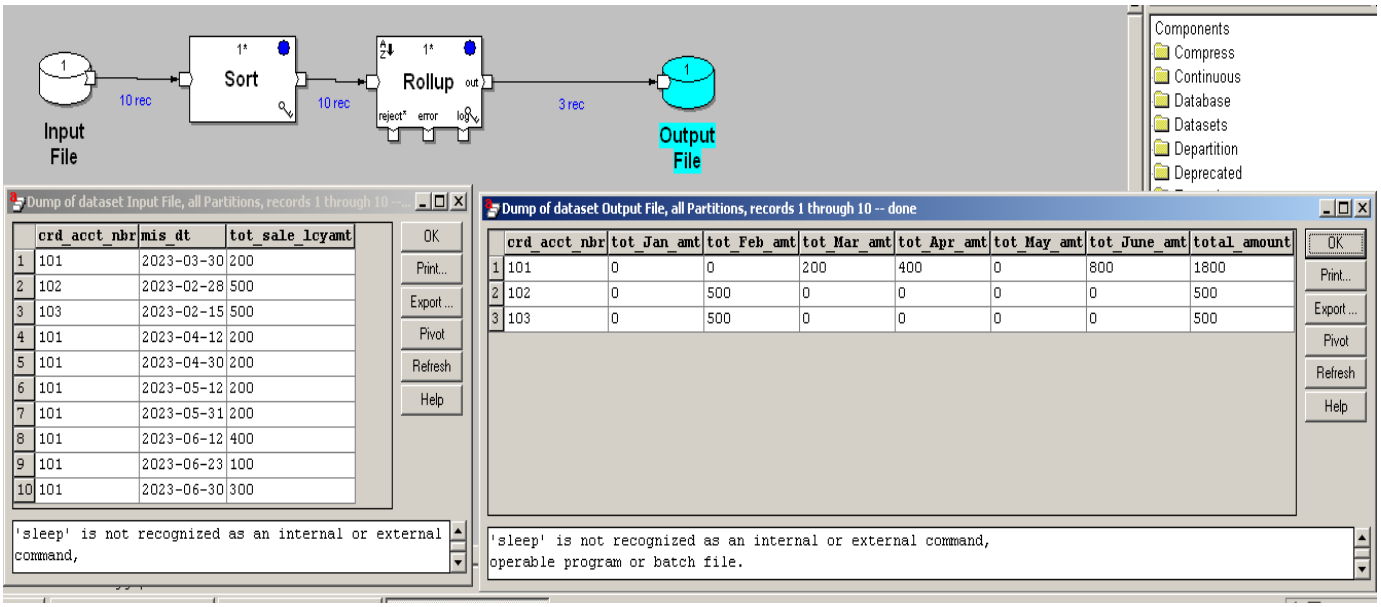
OUTPUT DML ::
```
record
  string("") str;
  decimal("") val;
end;
```

TRANSFORM LOGIC :: ROLLUP :: KEY AS {}

```
type temporary_type =
record
  string("") v_str;
end;
out::initialize(in) =
begin
  out.v_str :: "";
end;
out::rollup(tmp, in) =
begin
  out.v_str :: string_concat(tmp.v_str,in.str);
end;
out::finalize(tmp, in) =
begin
  out.str :: tmp.v_str;
  out.val :: in.val;
end;
```

TRANSFORM LOGIC :: NORMALIZE ::

```
out::length(in) =
begin
  out :: in.val;
end;

out::normalize(in, index) =
begin
  out.str :: in.str;
  out.val :: index+1;
end;
```



INPUT DML ::

```
record
string (" ") crd_acct_nbr;
date ("YYYY-MM-DD")(" ") mis_dt;
decimal ("\r\n") tot_sale_lcyamt;
end
```

OUTPUT DML ::

```
record
  string("") crd_acct_nbr;
  decimal("") tot_Jan_amt;
  decimal("") tot_Feb_amt;
  decimal("") tot_Mar_amt;
  decimal("") tot_Apr_amt;
  decimal("") tot_May_amt;
  decimal("") tot_June_amt;
  decimal("") total_amount;
end;
```

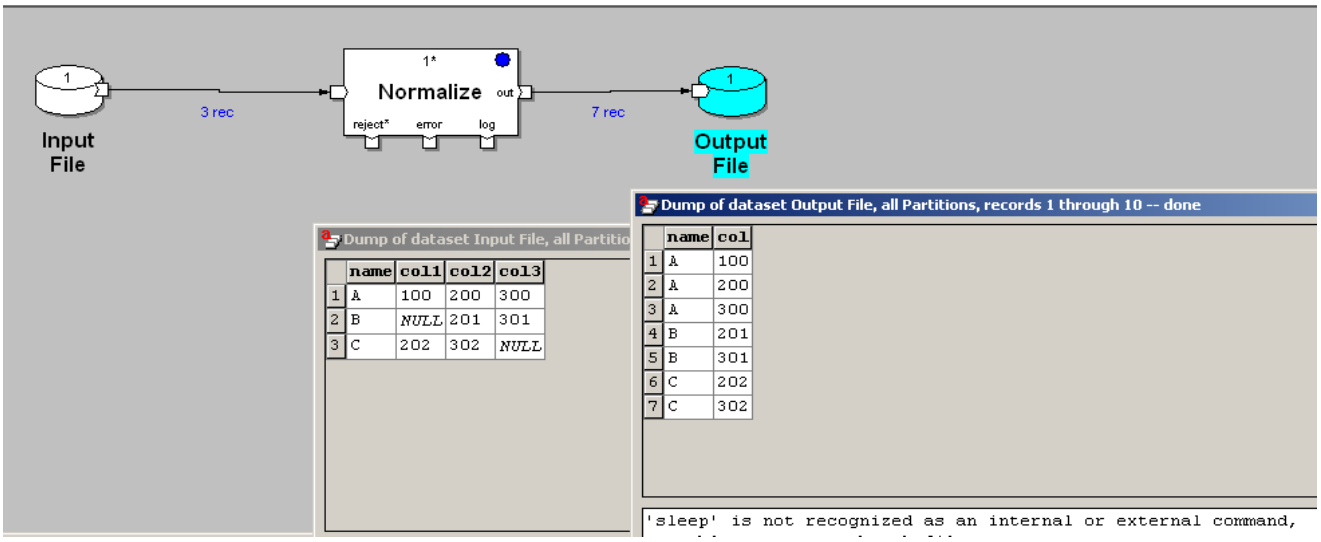
TRANSFORM LOGIC ::

SORT -→KEY→ {crd_acct_nbr}

ROLLUP :

```
type temporary_type =
record
  decimal("") v_Jan;
  decimal("") v_Feb;
  decimal("") v_Mar;
  decimal("") v_Apr;
  decimal("") v_May;
  decimal("") v_Jun;
  decimal("") v_total;
end;
```

```
out::initialize(in) =
begin
  out.v_Jan :: 0;
  out.v_Feb :: 0;
  out.v_Mar :: 0;
  out.v_Apr :: 0;
  out.v_May :: 0;
  out.v_Jun :: 0;
  out.v_total :: 0;
end;


out::rollup(tmp, in) =
begin
  out.v_Jan :: if( date_month(in.mis_dt)==1) tmp.v_Jan+in.tot_sale_lcyamt
else tmp.v_Jan;
  out.v_Feb :: if( date_month(in.mis_dt)==2) tmp.v_Feb+in.tot_sale_lcyamt
else tmp.v_Feb;
  out.v_Mar :: if( date_month(in.mis_dt)==3) tmp.v_Mar+in.tot_sale_lcyamt
else tmp.v_Mar;
  out.v_Apr :: if( date_month(in.mis_dt)==4) tmp.v_Apr+in.tot_sale_lcyamt
else tmp.v_Apr;
  out.v_May :: if( date_month(in.mis_dt)==1) tmp.v_May+in.tot_sale_lcyamt
else tmp.v_May;
  out.v_Jun :: if( date_month(in.mis_dt)==6) tmp.v_Jun+in.tot_sale_lcyamt
else tmp.v_Jun;
  out.v_total :: tmp.v_total+in.tot_sale_lcyamt;
end;


out::finalize(tmp, in) =
begin
  out.crd_acct_nbr :: in.crd_acct_nbr;
  out.tot_Jan_amt :: tmp.v_Jan;
  out.tot_Feb_amt :: tmp.v_Feb;
  out.tot_Mar_amt :: tmp.v_Mar;
  out.tot_Apr_amt :: tmp.v_Apr;
  out.tot_May_amt :: tmp.v_May;
  out.tot_June_amt :: tmp.v_Jun;
  out.total_amount :: tmp.v_total;
end;
```



INPUT DML::

```
record
string(" ") name;
decimal(3) col1=NULL("");
decimal(3) col2=NULL("");
decimal("\r\n") col3=NULL("");
end
```

OUTPUT DML ::

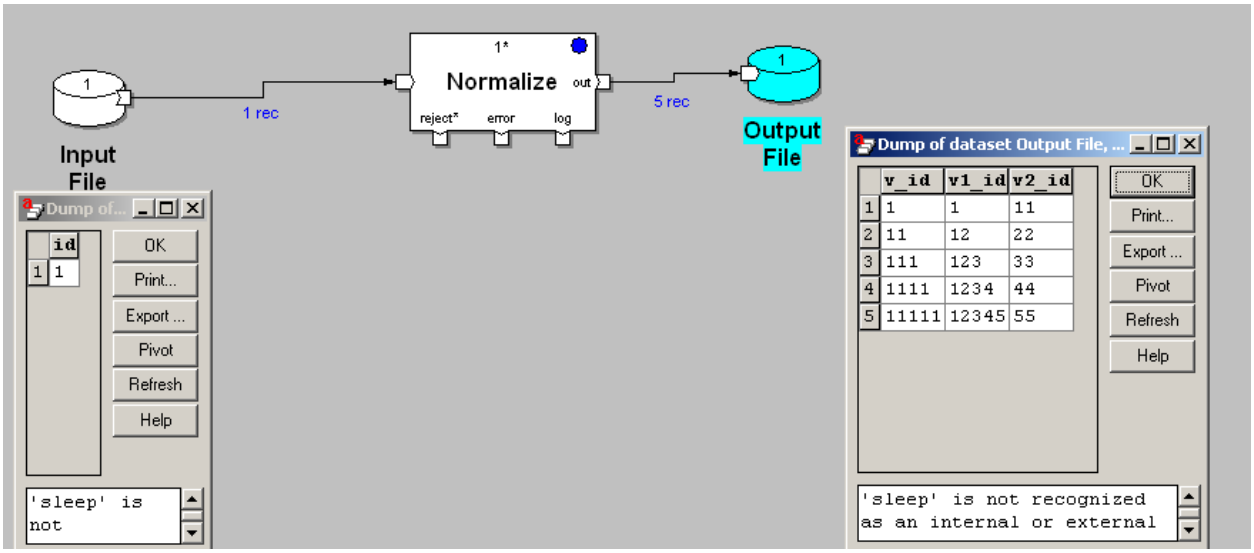
```
record
  string(" ") name;
  ascii decimal("\0") col;
end;
```

TRANSFORM LOGIC ::

```
out::length(in) =
begin
  out :: 3;
end;
```

```
out::normalize(in, index) =
begin
  out.name :: in.name;
  out.col :: first_defined(if(index==0)in.col1 else if (index==1) in.col2
else in.col3, " ");
end;
```

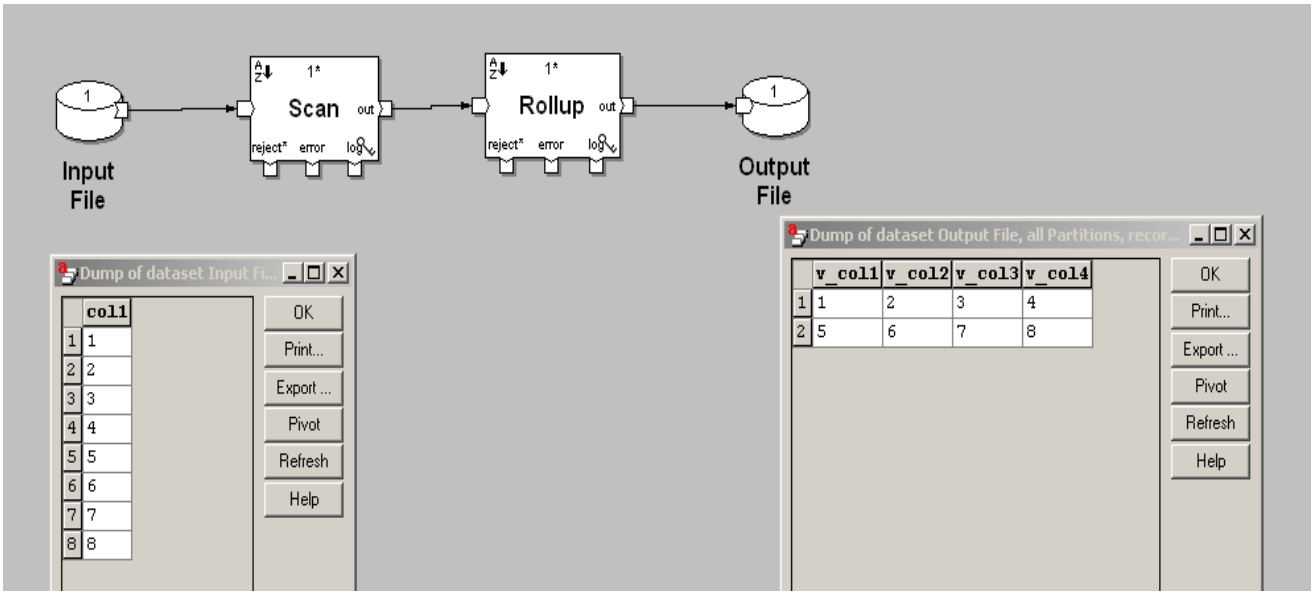
```
out::output_select(out) =
begin
  out :: !is_blank(out.col);
end;
```



INPUT DML ::
```
record
decimal("\r\n") id;
end
```
OUTPUT DML ::
```
record
  string("") v_id;
  decimal("") v1_id;
  decimal("") v2_id;
end;
```
TRANSFORM LOGIC ::

```
out::length(in) =
begin
  out :: 5;
end;
type temporary_type =
record
  decimal("") v_id;
  decimal("") v1_id;
  decimal("") v2_id;
          end /* Temporary variable*/;
temp::initialize(in) =
begin
  temp.v_id :: "";
  temp.v1_id :: "";
  temp.v2_id :: 0;
         end;
temp::normalize(temp, in, index) =
begin
  temp.v_id :: string_concat(temp.v_id,in.id);
  temp.v1_id :: string_concat(temp.v1_id,(decimal(""))(in.id+index));
  temp.v2_id::
string_concat((decimal(""))(in.id+index),(decimal(""))(in.id+index));
   end;
```

```
out::finalize(temp, in) =
begin
  out.v_id :: temp.v_id;
  out.v1_id :: temp.v1_id;
  out.v2_id :: temp.v2_id;
end;
```



INPUT DML :: `record`

```
decimal("\r\n") col1;
end
```

OUTPUT DML :: `record`

```
  decimal("") v_col1;
  decimal("") v_col2;
  decimal("") v_col3;
  decimal("") v_col4;
end;
```

TRANSFORM LOGIC ::

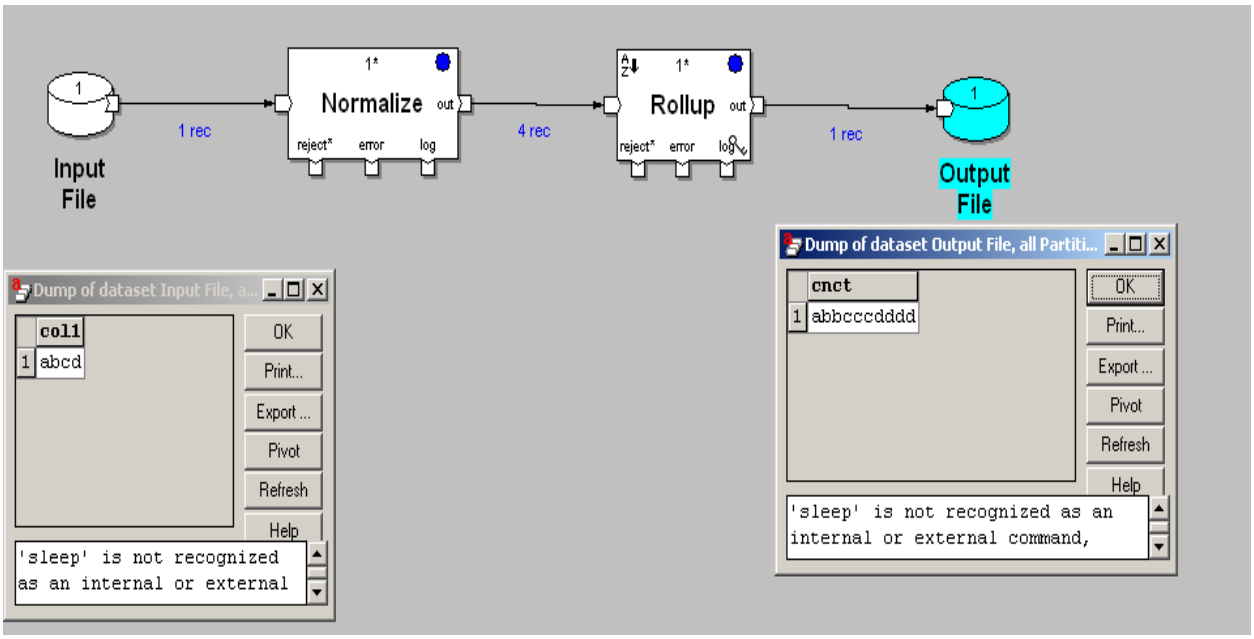
SCAN : KEY METHOD -→USE KEY CHANGE FUNCTION

```
type temporary_type =
record
  decimal("") v_count;
end /* Temporary variable*/;

temp::initialize(in) =
begin
  temp.v_count :: 0;
end;
```

```
temp::scan(temp, in) =
begin
  temp.v_count :: temp.v_count+1;
end;


out::finalize(temp, in) =
begin
  out.col1 :: in.col1;
  out.count :: temp.v_count;
end;


out::key_change(in1, in2) =
begin
  out :: in1.col1%4==0;
end;
```

ROLLUP :KEY METHOD -→KEY CHANGE FUNCTION

```
type temporary_type =
record
  decimal("") v_col1;
  decimal("") v_col2;
  decimal("") v_col3;
  decimal("") v_col4;
end;

out::initialize(in) =
begin
  out.v_col1 :: 0;
  out.v_col2 :: 0;
  out.v_col3 :: 0;
  out.v_col4 :: 0;
end;


out::rollup(tmp, in) =
begin
  out.v_col1 :: if (in.count==1) in.col1 else tmp.v_col1;
  out.v_col2 :: if (in.count==2) in.col1 else tmp.v_col2;
  out.v_col3 :: if (in.count==3) in.col1 else tmp.v_col3;
  out.v_col4 :: if (in.count==4) in.col1 else tmp.v_col4;
end;

out::finalize(tmp, in) =
begin
  out.v_col1 :: tmp.v_col1;
  out.v_col2 :: tmp.v_col2;
  out.v_col3 :: tmp.v_col3;
  out.v_col4 :: tmp.v_col4;
end;
out::key_change(in1, in2) =
begin
  out :: in1.col1%4==0;
end;
```



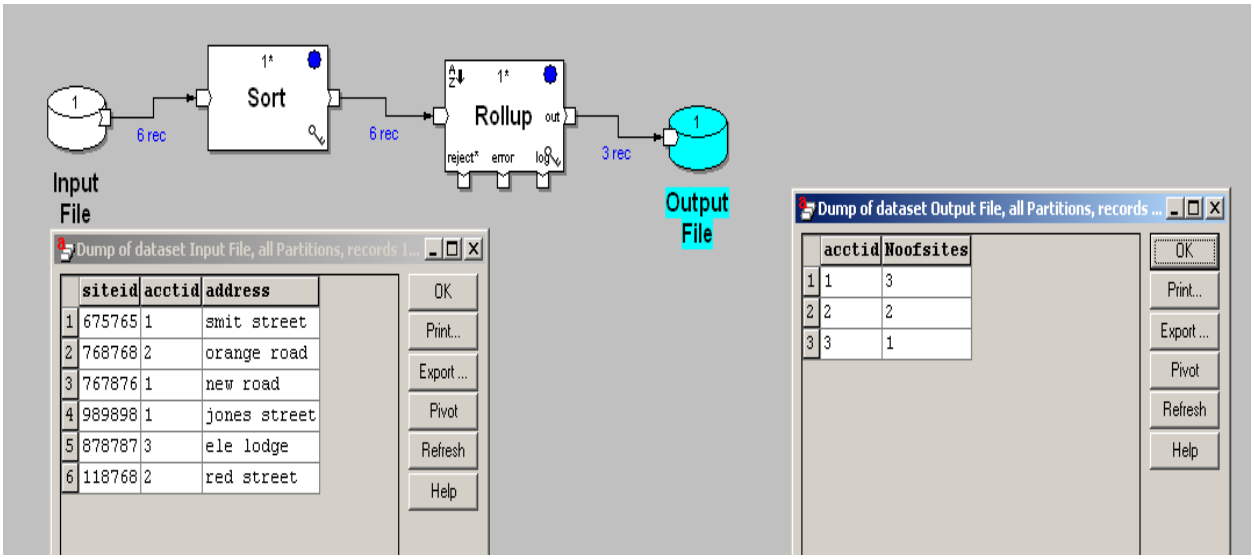
INPUT DML :: `record`
```
string("\r\n") col1;
end
```
OUTPUT DML :: `record`
```
  string("") cnct;
end;
```
TRANSFORM LOGIC:

Normalize :

```
out::length(in) =
begin
  out :: length_of(in.col1);
end;

out::normalize(in, index) =
begin
let integer(8) i;
let string("") v_cnct="";
for (i, i<(index+1))
v_cnct=string_concat(v_cnct,string_substring(in.col1,index+1,1));
  out.col1 :: v_cnct;

end;
```
ROLLUP ::

```
type temporary_type =
record
  string("") v_cnct;
end;

out::initialize(in) =
begin
  out.v_cnct :: "";
end;
```

```
out::rollup(tmp, in) =
begin
  out.v_cnct :: string_concat(tmp.v_cnct,in.col1);
end;


out::finalize(tmp, in) =
begin
  out.cnct :: tmp.v_cnct;
end;
```



INPUT DML ::

```
record
decimal(",") siteid;
decimal(",") acctid;
string("\r\n") address;
end
```

OUTPUT DML ::

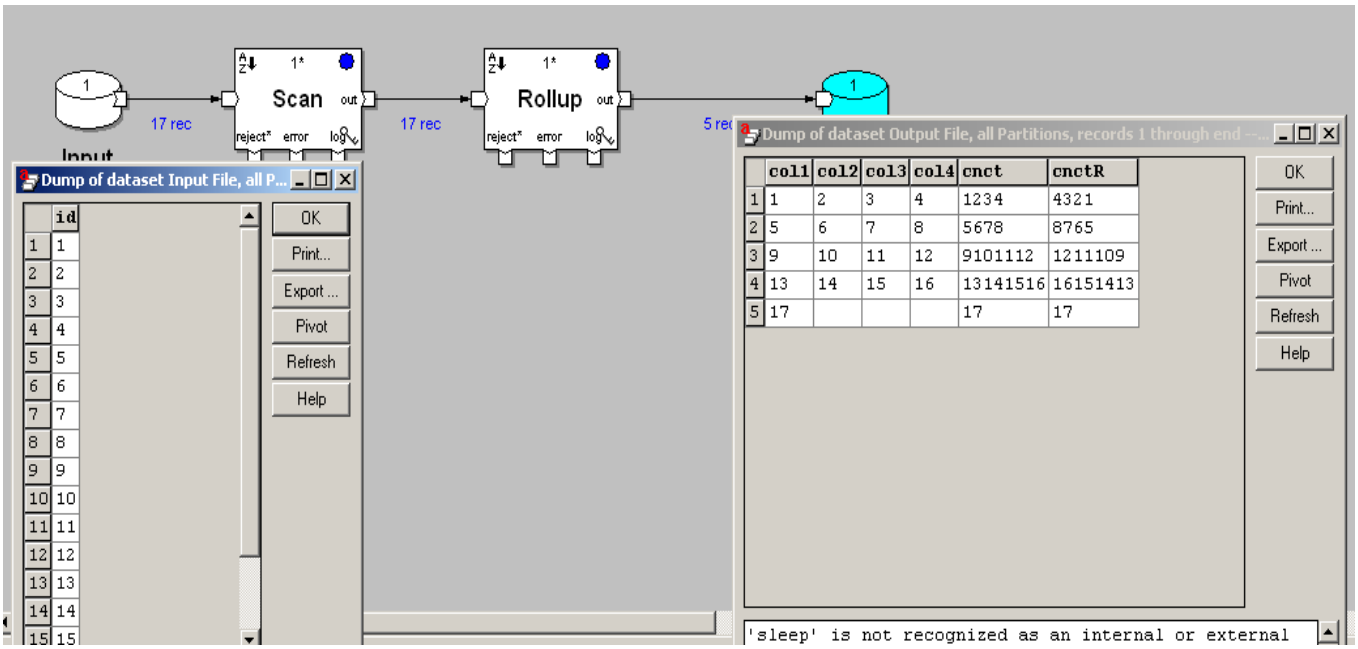
```
record
  decimal("") acctid;
  decimal("") Noofsites;
end;
```

TRANSFORM LOGIC ::

SORT ----------→KEY----------→ {acctid}

ROLLUP:

```
type temporary_type = record
  unsigned long count_tmp_0;
end;



out::initialize(in) =
begin
  out.count_tmp_0 :: 0;  /**GENERATED*'count(in.siteid)'*/
end;

out::rollup(tmp, in) =
begin
  out.count_tmp_0 :: tmp.count_tmp_0 + 1;
end;

/**/

out::finalize(tmp, in) =
begin
  out.acctid :: in.acctid;
  out.Noofsites :: (tmp.count_tmp_0);
end;
```



id
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

col1	col2	col3	col4	cnct	cnctR
1	2	3	4	1234	4321
5	6	7	8	5678	8765
9	10	11	12	9101112	1211109
13	14	15	16	13141516	16151413
17				17	17

INPUT DML :: `record`

```
decimal ("\r\n") id;
end
```

OUTPUT DML :: `record`

```
  decimal("") col1;
  decimal("") col2;
  decimal("") col3;
  decimal("") col4;
  decimal("") cnct;
  decimal("") cnctR;
end
```

TRANSFORM LOGIC ::

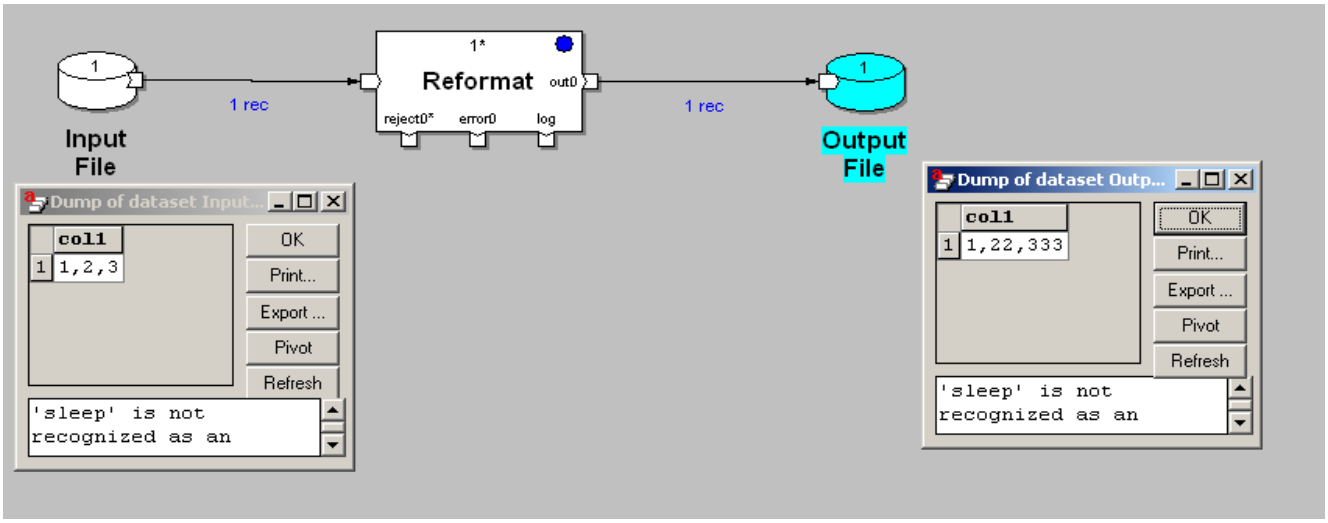
SCAN : KEY AS KEY CHANGE FUNCTION

```
type temporary_type =
record
  decimal("") v_count;
end /* Temporary variable*/;

temp::initialize(in) =
begin
  temp.v_count :: 0;
end;
temp::scan(temp, in) =
begin
  temp.v_count :: temp.v_count+1;
end;

out::finalize(temp, in) =
begin
  out.id :: in.id;
  out.seq :: temp.v_count;
end;
```

```
out::key_change(in1, in2) =
begin
  out :: in1.id%4==0;
end;
```

ROLLUP :: KEY AS CHANGE FUNCTION

```
type temporary_type =
record
  decimal("") v_col1;
  decimal("") v_col2;
  decimal("") v_col3;
  decimal("") v_col4;
  decimal("") v_cnct;
  decimal("") v_cnctR;
end;

temp::initialize(in) =
begin
  temp.v_col1 :: "";
  temp.v_col2 :: "";
  temp.v_col3 :: "";
  temp.v_col4 :: "";
  temp.v_cnct :: "";
  temp.v_cnctR :: "";
end;

out::rollup(tmp, in) =
begin
  out.v_col1 :: if (in.seq==1) in.id else tmp.v_col1;
  out.v_col2 :: if (in.seq==2) in.id else tmp.v_col2;
  out.v_col3 :: if (in.seq==3) in.id else tmp.v_col3;
  out.v_col4 :: if (in.seq==4) in.id else tmp.v_col4;
  out.v_cnct :: string_concat(tmp.v_cnct,in.id);
  out.v_cnctR :: string_concat(in.id, tmp.v_cnctR);
end;


out::finalize(tmp, in) =
begin
  out.col1 :: tmp.v_col1;
  out.col2 :: tmp.v_col2;
  out.col3 :: tmp.v_col3;
  out.col4 :: tmp.v_col4;
  out.cnct :: tmp.v_cnct;
  out.cnctR :: tmp.v_cnctR;
end;


out::key_change(in1, in2) =
begin
  out :: in1.id%4==0;
end;
```



INPUT DML :: record

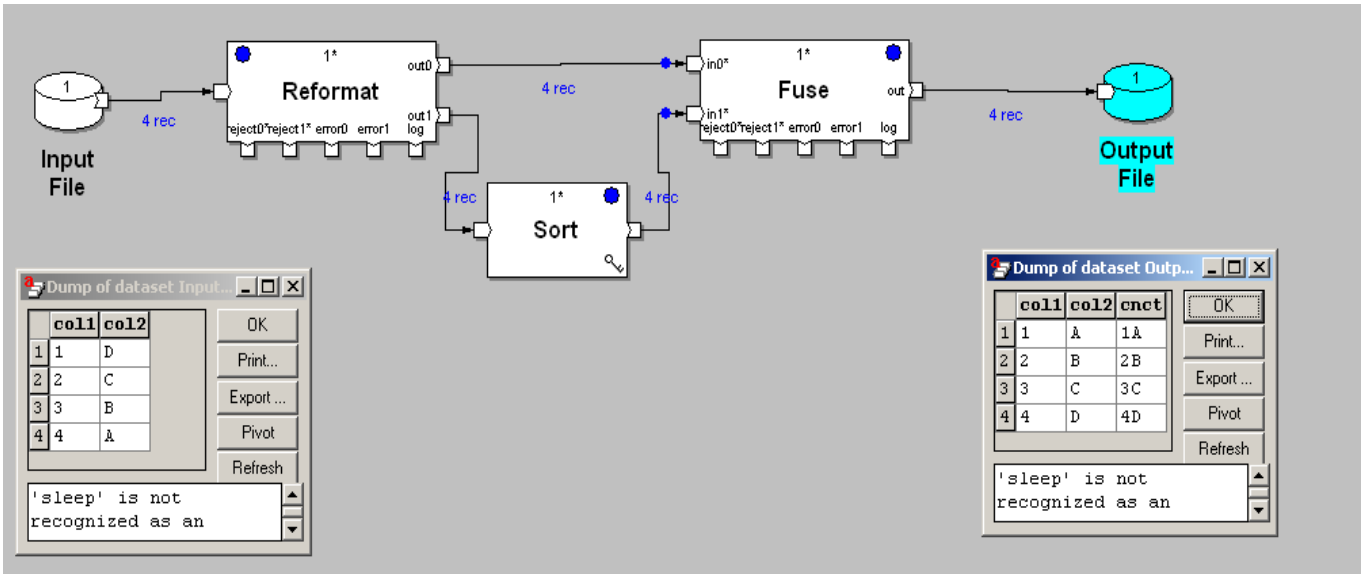
```
string ("\r\n") col1;
end
```

OUTPUT DML :: record

```
  string("\r\n") col1;
end;
```

TRANSFORM LOGIC :

```
 out::reformat(in) =
begin
  let integer(4) i;
  let integer(4) j;
  let decimal("") v_cnct = "";

for (i, i < length_of(in.col1))
    begin
    if (string_substring(in.col1, i+1, 1)==",")
    v_cnct=string_concat(v_cnct,string_substring(in.col1, i+1, 1));
    else
    for (j,j<(decimal(""))string_substring(in.col1, i+1, 1))
    v_cnct=string_concat(v_cnct,string_substring(in.col1, i+1, 1));
    end

  out.col1 :: v_cnct;
end;
```



INPUT DML: `record`

```
string (" ") col1;
decimal ("\r\n") col2;
end
```

OUTPUT DML: `record`

```
  string(" ") col1;
  decimal("") col2;
  string("") cnct;
end;
```

TRANSFORM LOGIC ::

REFORMAT-0 ::

```
out::reformat(in) =
begin
  out.col1 :: in.col1;
end;
```

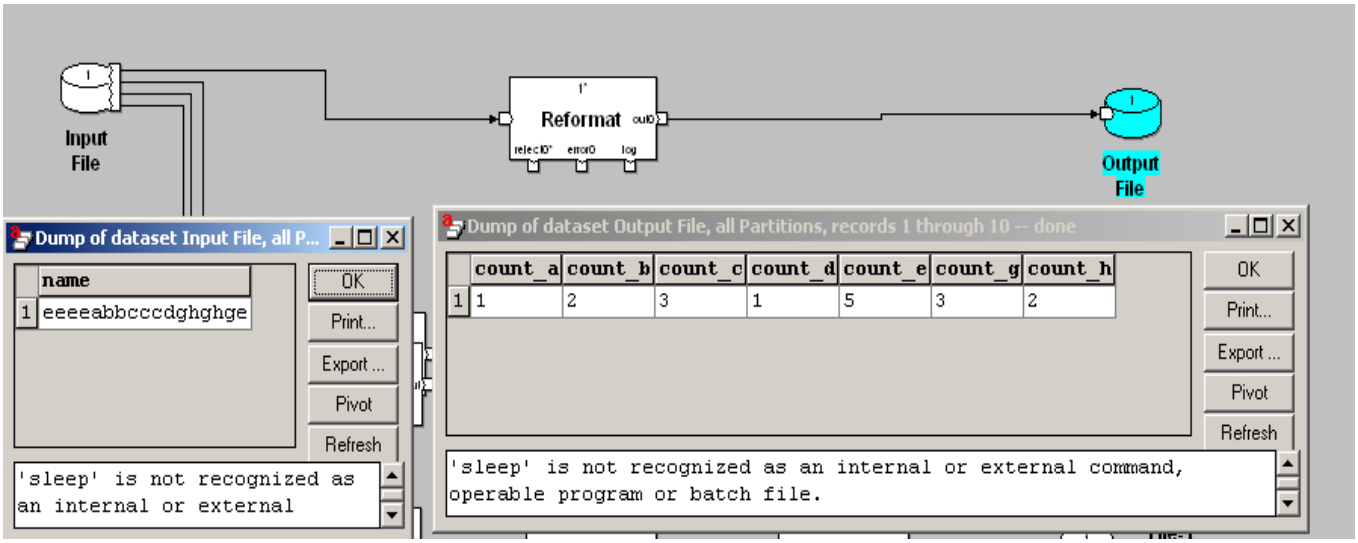
REFORMAT-1 ::

```
out::reformat(in) =
begin
  out.col2 :: in.col2;
  out.seq :: next_in_sequence();
end;
```

SORT --→KEY-----→{seq descending}

FUSE ::

```
out::fuse(in0, in1) =
begin
  out.col1 :: in0.col1;
  out.col2 :: in1.col2;
  out.cnct :: string_concat(in0.col1,in1.col2);
end;
```



INPUT DML ::

```
record
string("\r\n") name;
end
```

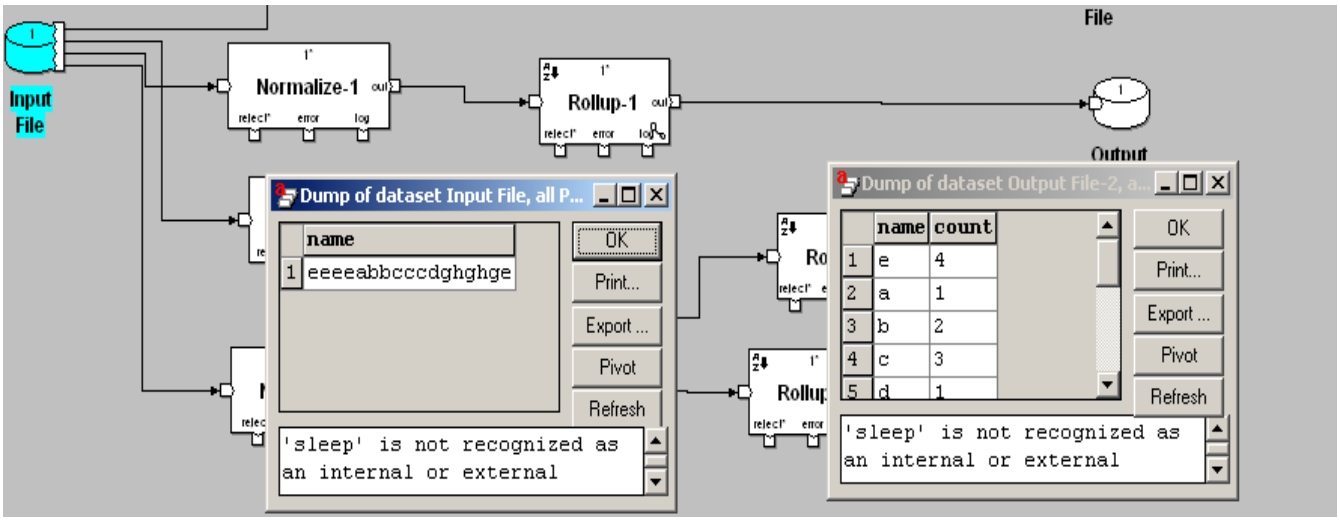
OUTPUT DML ::

```
record
  decimal("") count_a;
  decimal("") count_b;
  decimal("") count_c;
  decimal("") count_d;
  decimal("") count_e;
  decimal("") count_g;
  decimal("") count_h;
end;
```

TRANSFORM LOGIC ::

```
out::reformat(in) =
begin
  let integer(4) i;
  let decimal("") v_count_a = 0;
  let decimal("") v_count_b = 0;
  let decimal("") v_count_c = 0;
  let decimal("") v_count_d = 0;
  let decimal("") v_count_e = 0;
  let decimal("") v_count_g = 0;
  let decimal("") v_count_h = 0;

for (i, i < length_of(in.name))
     if (string_substring(in.name, i+1, 1)=="a")
    v_count_a = v_count_a +1; else
if (string_substring(in.name, i+1, 1)=="b")
    v_count_b = v_count_b +1; else
if (string_substring(in.name, i+1, 1)=="c")
    v_count_c = v_count_c +1;  else
if (string_substring(in.name, i+1, 1)=="d")
    v_count_d = v_count_d +1;  else
if (string_substring(in.name, i+1, 1)=="e")
    v_count_e = v_count_e +1;  else
```

```
if (string_substring(in.name, i+1, 1)=="g")
    v_count_g = v_count_g +1;  else
if (string_substring(in.name, i+1, 1)=="h")
    v_count_h = v_count_h +1 ;

  out.count_a :: v_count_a;
  out.count_b :: v_count_b;
  out.count_c :: v_count_c;
  out.count_d :: v_count_d;
  out.count_e :: v_count_e;
  out.count_g :: v_count_g;
  out.count_h :: v_count_h;
end;
```



INPUT DML ::

```
record
string("\r\n") name;
end
```

OUTPUT DML ::

```
record
  string("\r\n") name;
  integer(8) count;
end;
```

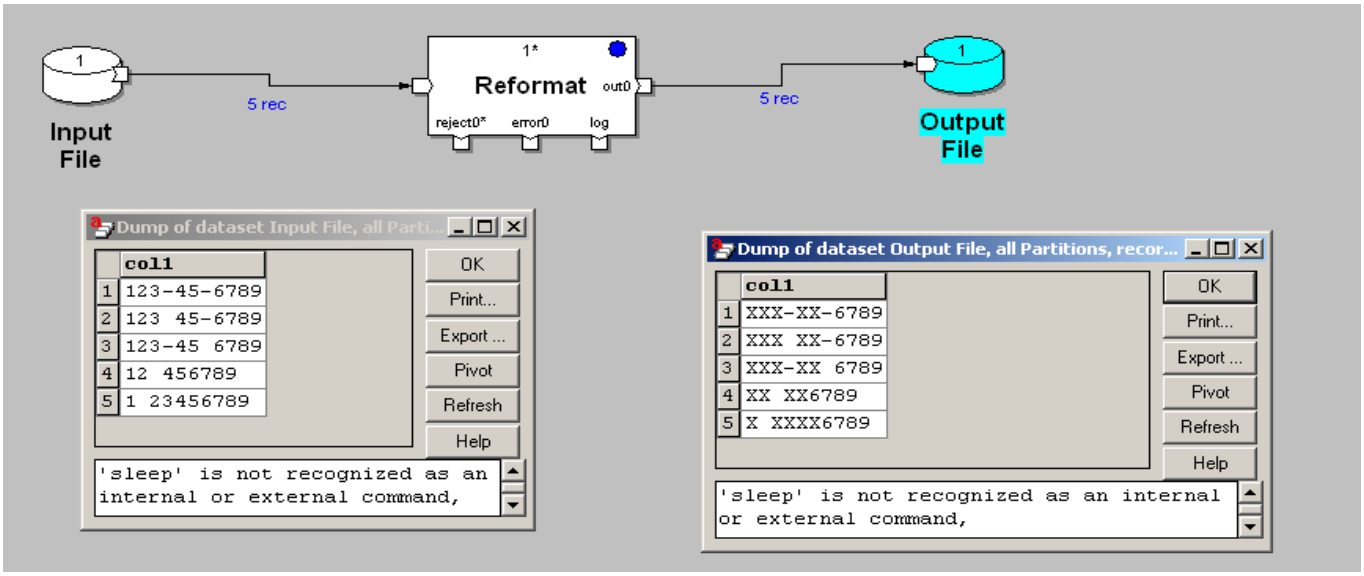
TRANSFORM LOGIC ::

NORMALIZE :

```
out::length(in) =
begin
  out :: length_of(in.name);
end;


out::normalize(in, index) =
begin
  out.name :: string_substring(in.name, index+1, 1);
end;
```

ROLLUP :KEY-METHOD AS KEY CHANGE FUNCTION

```
type temporary_type =
record
  decimal("") v_count;
end;

out::initialize(in) =
begin
  out.v_count :: 0;
end;


out::rollup(tmp, in) =
begin
  out.v_count :: tmp.v_count+1;
end;


out::finalize(tmp, in) =
begin
  out.name :: in.name;
  out.count :: tmp.v_count;
end;


out::key_change(in1, in2) =
begin
  out :: in1.name!=in2.name;
end;
```



INPUT DML :: `record`

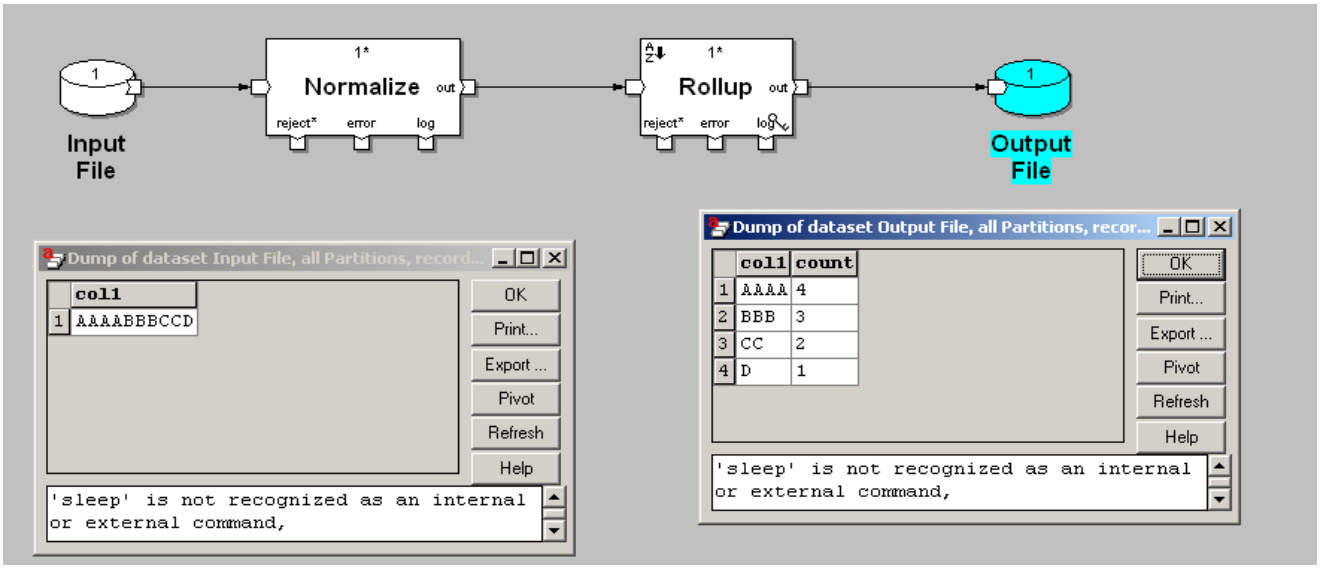
```
string("\r\n") col1;
end
```

OUTPUT DML ::

```
record
  string("\r\n") col1;
end;
```

TRANSFORM LOGIC ::

```
out::reformat(in) =
begin
let int i;
let string("") v_col1=in.col1;
for (i, i< (length_of(in.col1)-4))
begin
if (string_substring(in.col1, i+1, 1)==" " || string_substring(in.col1, i+1,
1)=="-")
v_col1 = v_col1;
else
v_col1= string_replace(v_col1,string_substring(v_col1, i+1, 1), "X");
end
out.col1 :: v_col1;
end
```



INPUT DML :: `record`

```
string("\r\n") col1;
end
```

OUTPUT DML :: `record`

```
  string("") col1;
  decimal("") count;
end;
```

TRANSFORM LOGIC ::

NORMALIZE ::

```
out::length(in) =
  begin
 out :: length_of(in.col1);
  end;
```

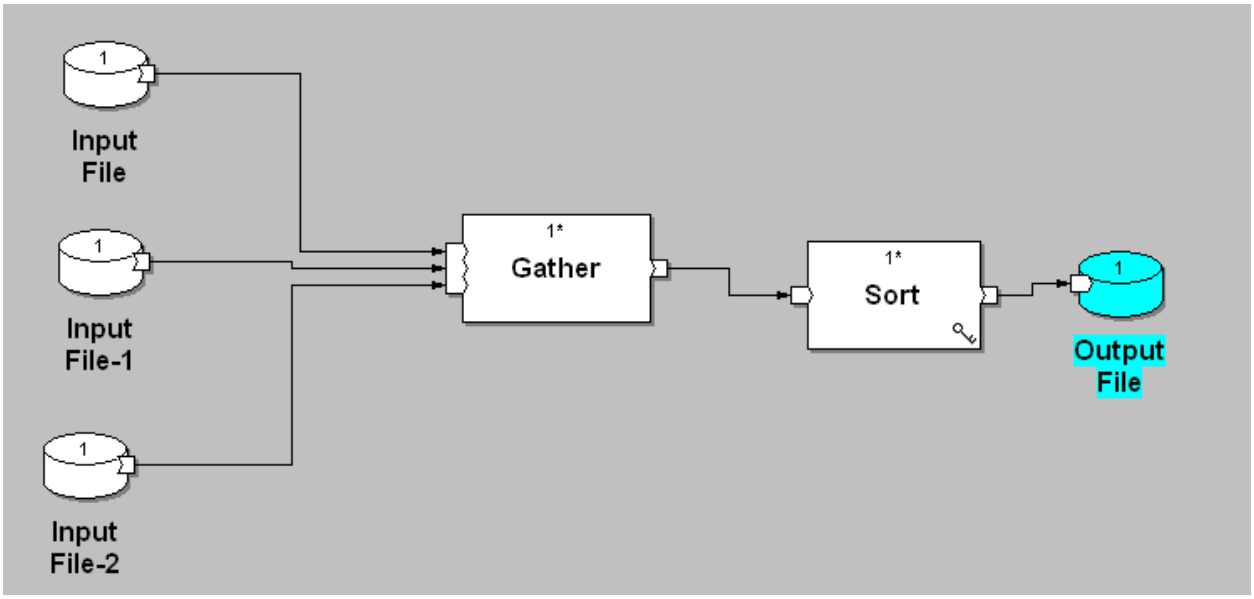
```
out::normalize(in, index) =
begin
  out.col1 :: string_substring(in.col1, index+1, 1);
end;
```

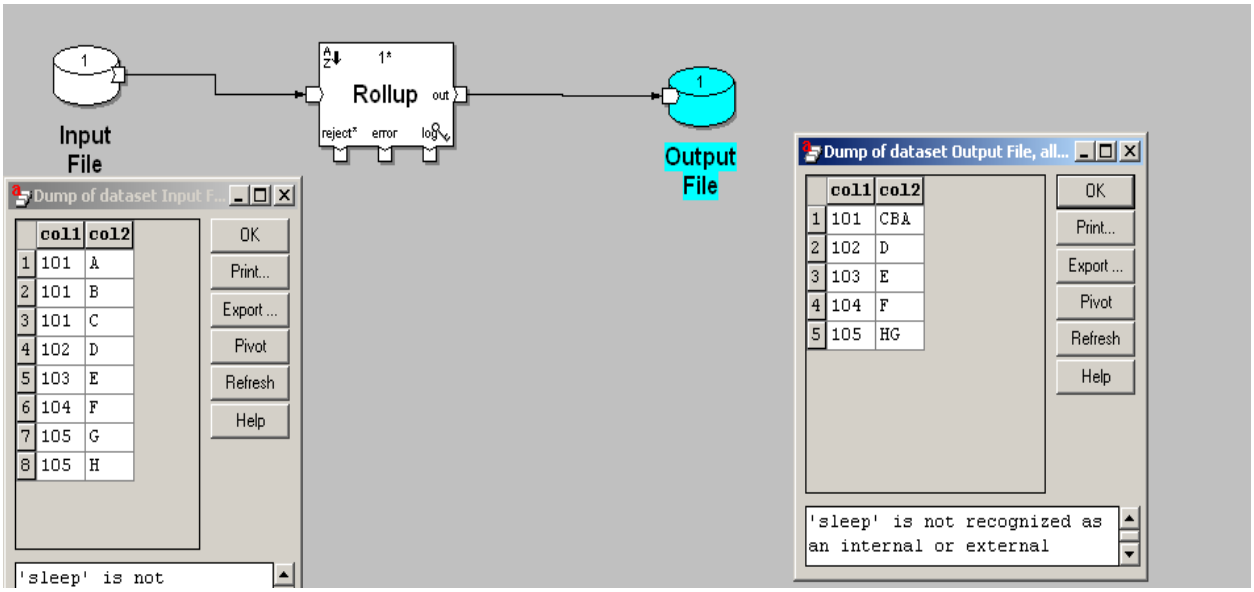
ROLLUP : KEY AS (COL1)

```
type temporary_type =
record
  string("") v_col1;
  decimal("") v_count;
end;

out::initialize(in) =
begin
  out.v_col1 :: "";
  out.v_count :: 0;
end;
```

```
out::rollup(tmp, in) =
begin
  out.v_col1 :: string_concat(tmp.v_col1,in.col1);
  out.v_count :: tmp.v_count+1;
end;


out::finalize(tmp, in) =
begin
  out.col1 :: tmp.v_col1;
  out.count :: tmp.v_count;
end;
```

SCENARIO-56





INPUT DML :: `record`

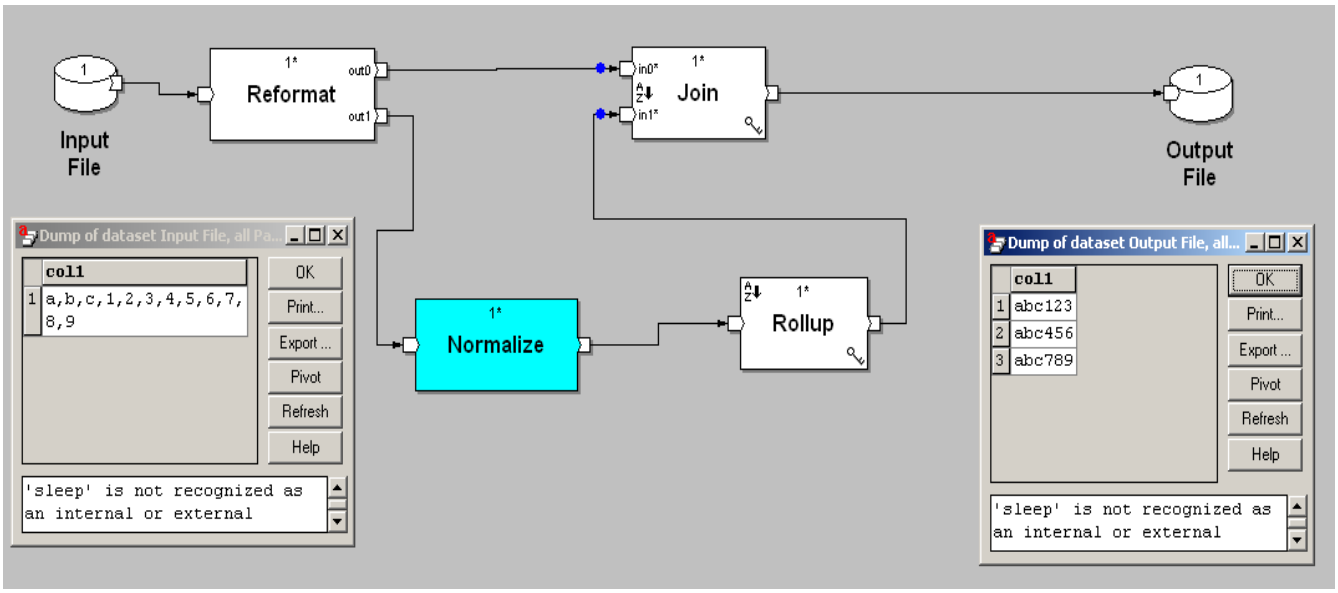
```
decimal(" ") col1;
string("\r\n") col2;
end
```

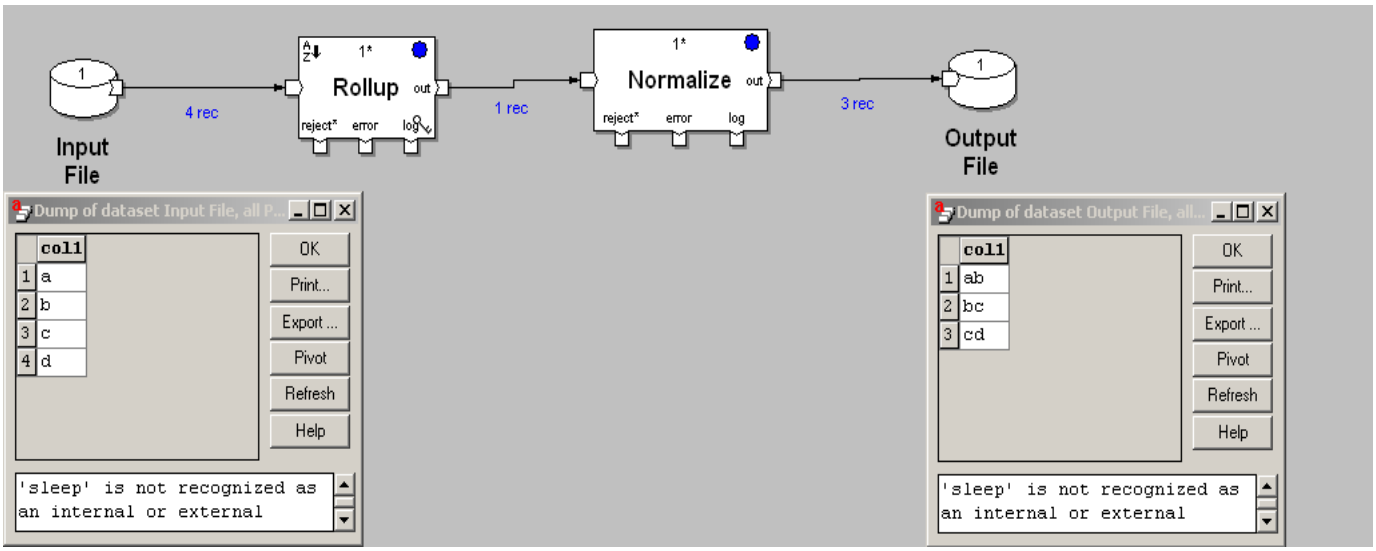
OUTPUT DML : : `record`

```
  decimal(" ") col1;
  string("\r\n") col2;
end;
```

TRANSFORM LOGIC :

```
type temporary_type =
record
  string("") v_col2;
end;

out::initialize(in) =
begin
  out.v_col2 :: "";
end;


out::rollup(tmp, in) =
begin
  out.v_col2 :: string_concat(in.col2,tmp.v_col2);
end;


out::finalize(tmp, in) =
begin
  out.col1 :: in.col1;  /**GENERATED*'in.col1'* */
  out.col2 :: tmp.v_col2;
end;
```


Input File
Reformat
out0
out1
Join
in0*
in1*
Output File
Normalize
Rollup
Dump of dataset Input File, all Pa...
col1
1 a,b,c,1,2,3,4,5,6,7,8,9
OK
Print...
Export ...
Pivot
Refresh
Help
'sleep' is not recognized as an internal or external
Dump of dataset Output File, all...
col1
1 abc123
2 abc456
3 abc789
OK
Print...
Export ...
Pivot
Refresh
Help
'sleep' is not recognized as an internal or external



INPUT DML :: `record`

```
string ("\r\n") col1;
end
```

OUTPUT DML :: `record`

```
  string("") col1;
end;
```

TRANSFORM LOGIC ::

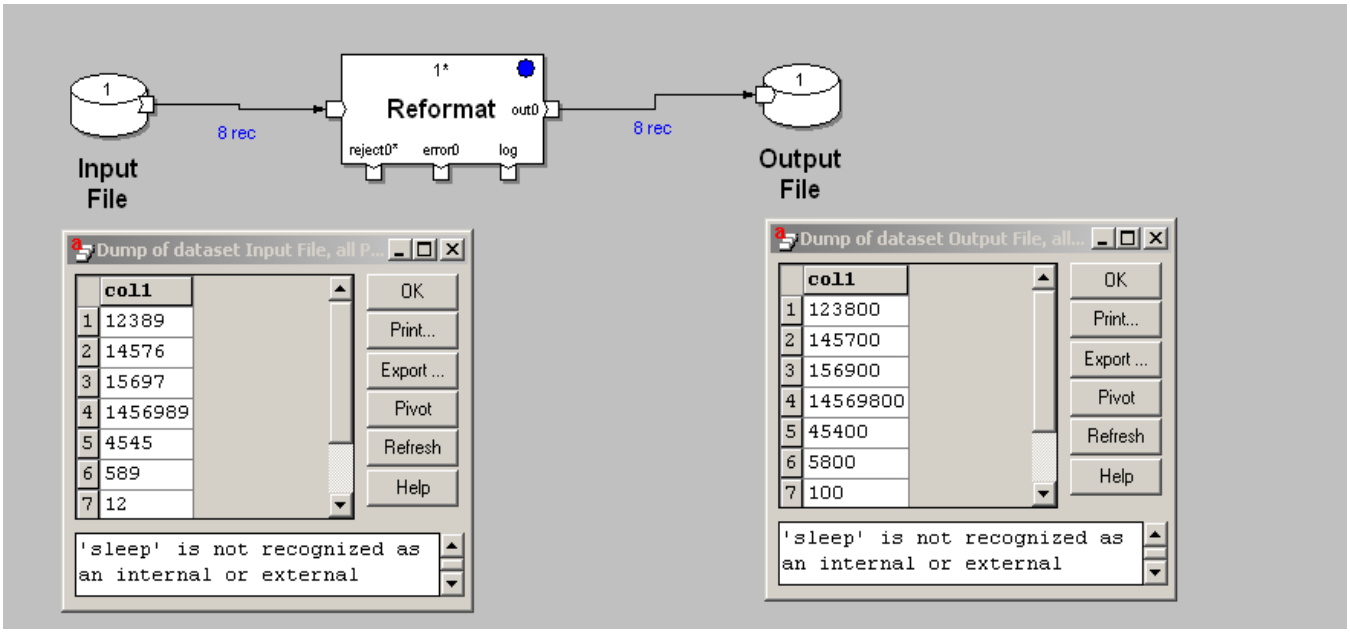
ROLLUP :: KEY SPECIFIER -→KEY AS {}

```
type temporary_type =
record
  string("") v_col1;
end;

out::initialize(in) =
begin
  out.v_col1 :: "";
end;

out::rollup(tmp, in) =
begin
  out.v_col1 :: string_concat(tmp.v_col1,in.col1);
end;

out::finalize(tmp, in) =
begin
  out.col1 :: tmp.v_col1;
end;
```

NORMALIZE ::

```
out::length(in) =
begin
  out :: length_of(in.col1)-1;
end;
```

```
out::normalize(in, index) =
begin
  out.col1 :: string_substring(in.col1,index+1, 2);
end;
```



INPUT DML :: `record`

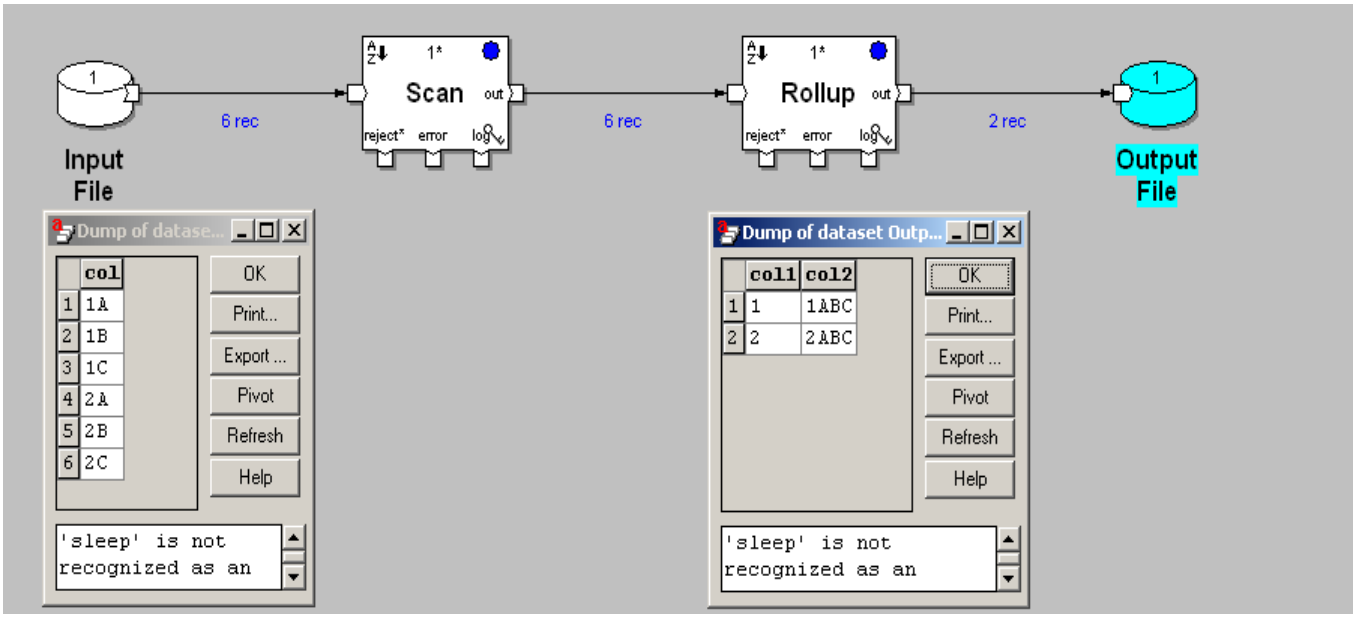
```
string("\r\n") col1;
end
```

OUTPUT DML :: `record`

```
  string("\r\n") col1;
end;
```

TRANSFORM LOGIC ::

```
out::reformat(in) =
begin
  out.col1 :: string_concat(string_substring(in.col1,1, length_of(in.col1)-
1),"00");
end;
```



INPUT DML :: `record`

```
string ("\r\n") col;
end
```

OUTPUT DML :: `record`

```
  decimal("") col1;
  string("") col2;
end;
```

TRANSFORM LOGIC ::

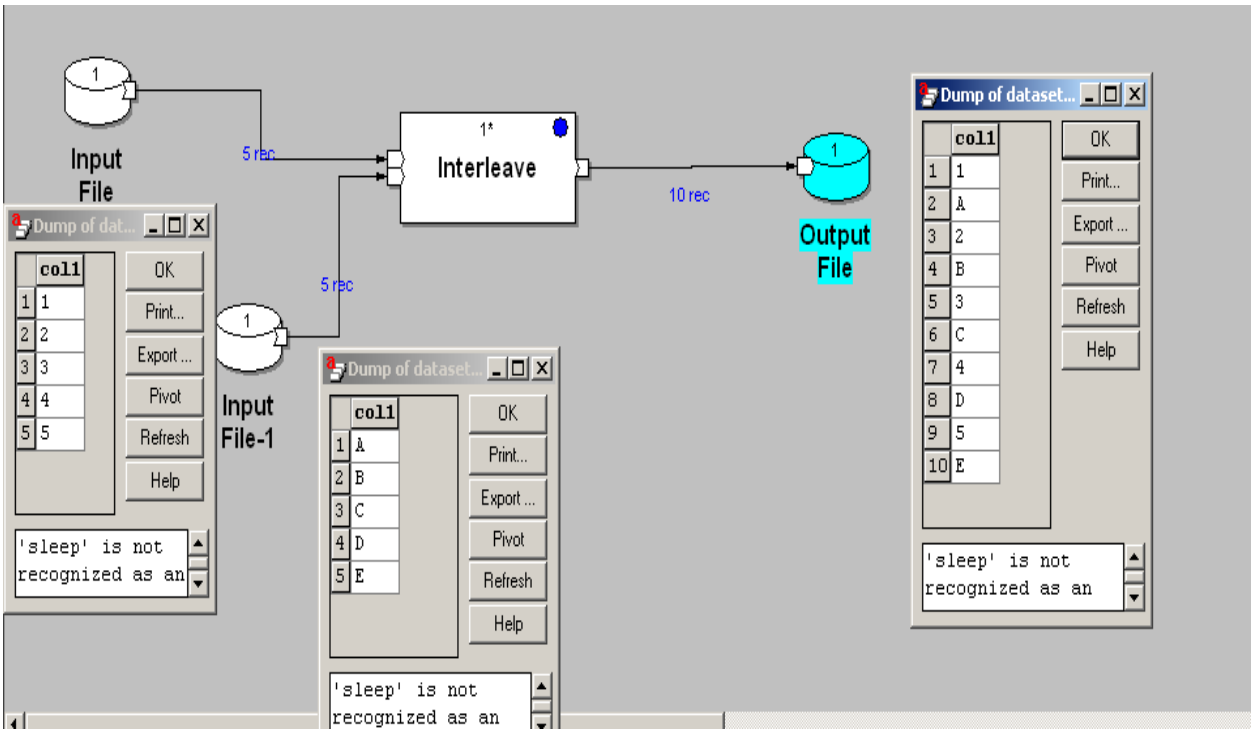
SCAN :KEY METHOD →USE KEY CHANGE FUNCTION

```
type temporary_type =
record
  decimal("") v_count;
end /* Temporary variable*/;

temp::initialize(in) =
begin
  temp.v_count :: 0;
end;

temp::scan(temp, in) =
begin
  temp.v_count :: temp.v_count+1;
end;

out::finalize(temp, in) =
begin
  out.v_count :: temp.v_count;
  out.col :: in.col;
end;
```

```
out::key_change(in1, in2) =
begin
  out :: next_in_sequence()%3==0;
end;
```

ROLLUP ::KEY-METHOD AS KEY CHANGE FUNCTION

```
type temporary_type =
record
  string("") v_col;
end;
temp::initialize(in) =
begin
  temp.v_col :: "";
end;
out::rollup(tmp, in) =
begin
  out.v_col :: if (in.v_count==1) in.col
               else string_concat(tmp.v_col,string_substring(in.col, 2,1));
end;

out::finalize(tmp, in) =
begin
  out.col1 :: next_in_sequence();
  out.col2 :: tmp.v_col;
end;


out::key_change(in1, in2) =
begin
  out :: in1.v_count==3;
end;
```



Input dml ::

```
record
string ("\r\n") col1;
end
```

Input dml -1::

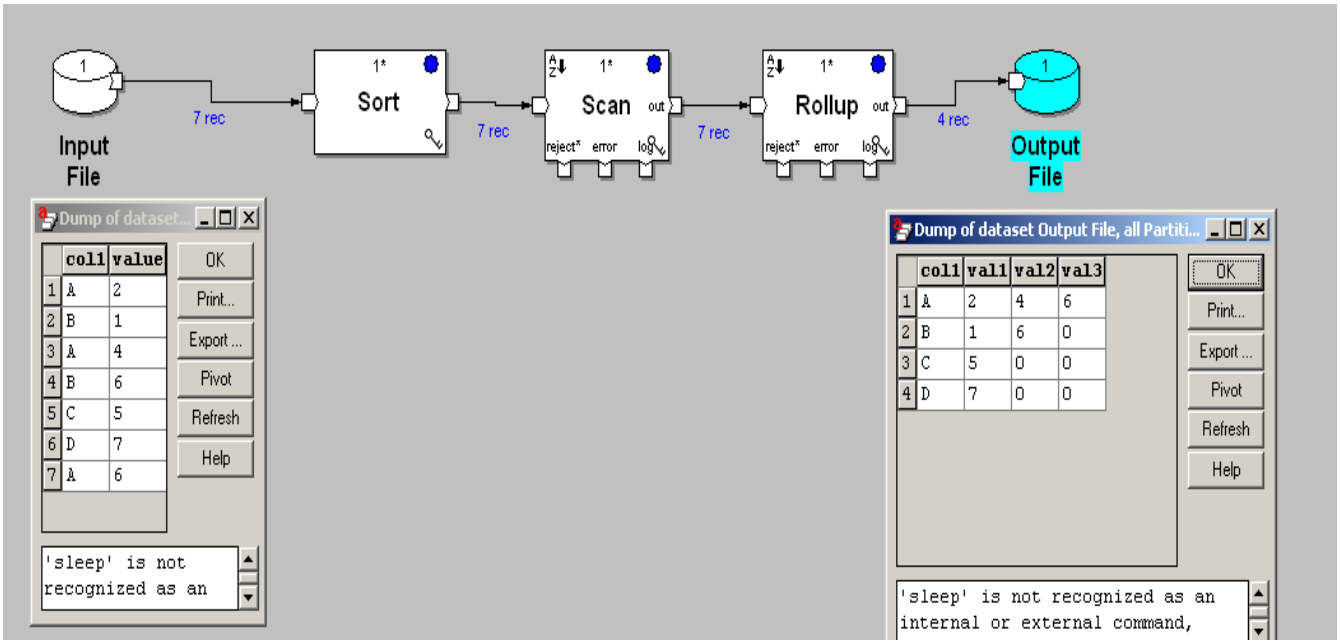
```
record
string ("\r\n") col1;
end
```

output dml ::

```
record
string ("\r\n") col1;
end
```

TRANSFORM LOGIC ::

INTERLEAVE -→BLOCKSIZE→1



INPUT DML :: `record`

```
string(" ") col1;
decimal("\r\n") value;
end
```

OUTPUT DML :: `record`

```
  string("") col1;
  decimal("") val1;
  decimal("") val2;
  decimal("") val3;
end;
```

TRANSFORM LOGIC ::

SORT→ {col1; value}

SCAN ::

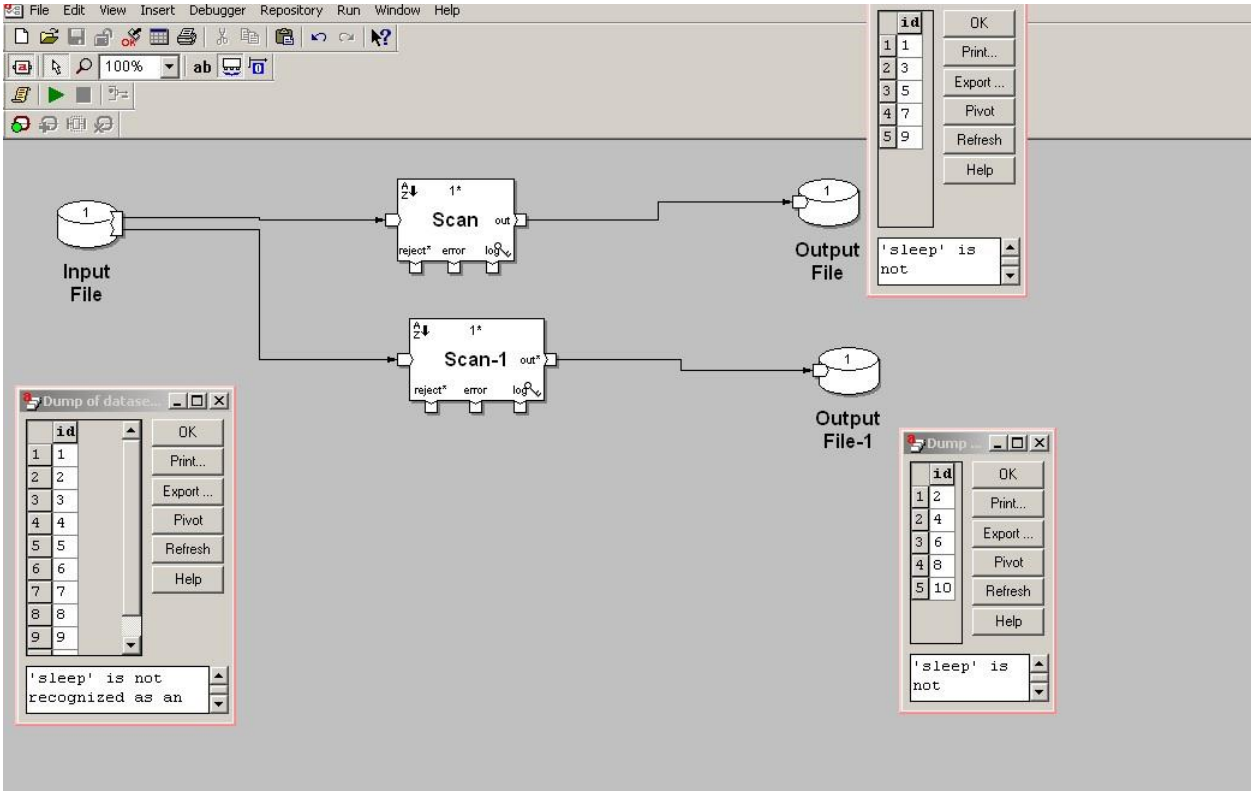
KEY→{COL1}

```
type temporary_type =
record
  decimal("") v_count;
end /* Temporary variable*/;

temp::initialize(in) =
begin
  temp.v_count :: 0;
end;


temp::scan(temp, in) =
begin
  temp.v_count :: temp.v_count+1;
end;
```

```
out::finalize(temp, in) =
begin
  out.col1 :: in.col1;
  out.value :: in.value;
  out.v_count :: temp.v_count;
end;
```

ROLLUP ::KEY→{COL1}

```
type temporary_type =
record
  decimal("") v_val1;
  decimal("") v_val2;
  decimal("") v_val3;
end;

out::initialize(in) =
begin
  out.v_val1 :: 0;
  out.v_val2 :: 0;
  out.v_val3 :: 0;
end;


out::rollup(tmp, in) =
begin
  out.v_val1 :: if (in.v_count==1) in.value else tmp.v_val1;
  out.v_val2 :: if (in.v_count==2) in.value else tmp.v_val2;
  out.v_val3 :: if (in.v_count==3) in.value else tmp.v_val3;
end;


out::finalize(tmp, in) =
begin
  out.col1 :: in.col1;
  out.val1 :: tmp.v_val1;
  out.val2 :: tmp.v_val2;
  out.val3 :: tmp.v_val3;
end;
```



INPUT DML :: `record`

```
  decimal("\r\n") id;
end;
```

OUTPUT DML :: `record`

```
  decimal("\r\n") id;
end;
```

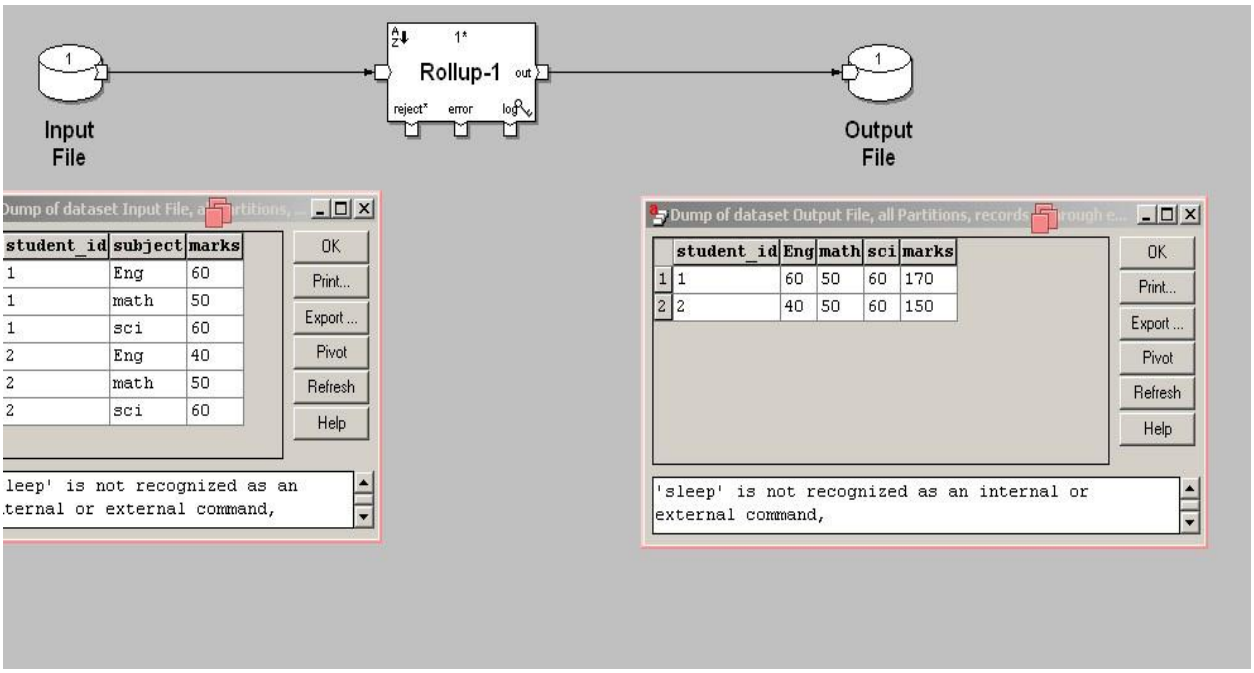
TRANSFORM LOGIC ::

SCAN :: KEY AS {}

```
type temporary_type =
record
end /* Temporary variable*/;

temp::scan(temp, in) =
begin
end;

out::finalize(temp, in) =
begin
  out.id :: in.id;
end;

out::output_select(out) =
begin
  out :: out.id%2!=0;
end
```

SCAN-1 :: KEY AS {}

```
type temporary_type =
record
end /* Temporary variable*/;

temp::scan(temp, in) =
begin
end;

out::finalize(temp, in) =
begin
  out.id :: in.id;
end;

out::output_select(out) =
begin
  out :: out.id%2==0;
end;
```



INPUT DML :: `record`

```
decimal("|") student_id;
string ("|") subject;
decimal("\r\n") marks;
end
```

OUTPUT DML :: `record`

```
  decimal("") student_id;
  decimal("") Eng;
  decimal("") math;
  decimal("") sci;
  decimal("") marks;
end;
```

TRANSFORM LOGIC ::

ROLLUP :: KEY AS {student_id}

```
type temporary_type =
record
  decimal("") v_eng_marks;
  decimal("") v_math_marks;
  decimal("") v_sci_marks;
  decimal("") v_sum_tmp_0; /* *GENERATED*'sum(in.marks)'*/
end;
```

```
out::initialize(in) =
begin
  out.v_sum_tmp_0 :: 0;  /**GENERATED*'sum(in.marks)'*/
  out.v_eng_marks :: "";
  out.v_math_marks :: "";
  out.v_sci_marks :: "";
end;


out::rollup(tmp, in) =
begin
  out.v_sum_tmp_0 :: tmp.v_sum_tmp_0 + in.marks;
  out.v_eng_marks :: if(in.subject=="Eng") in.marks else tmp.v_eng_marks;
  out.v_math_marks :: if(in.subject=="math") in.marks else tmp.v_math_marks;
  out.v_sci_marks :: if(in.subject=="sci") in.marks else tmp.v_sci_marks;
end;


out::finalize(tmp, in) =
begin
  out.student_id :: in.student_id;   /**GENERATED*'in.student_id'* */
  out.Eng :: tmp.v_eng_marks;   /**GENERATED*'if (in.subject=="Eng")
(in.marks) else if (in.subject=="math") (in.marks) else if
(in.subject=="sci") (in.marks) else 0'* */
  out.math :: tmp.v_math_marks;  /**GENERATED*'if (in.subject=="Eng")
(in.marks) else if (in.subject=="math") (in.marks) else if
(in.subject=="sci") (in.marks) else 0'* */
  out.sci :: tmp.v_sci_marks;  /**GENERATED*'if (in.subject=="Eng")
(in.marks) else if (in.subject=="math") (in.marks) else if
(in.subject=="sci") (in.marks) else 0'* */
  out.marks :: tmp.v_sum_tmp_0;

end;
```