



**Some pages of this thesis may have been removed for copyright restrictions.**

If you have discovered material in AURA which is unlawful e.g. breaches copyright, (either yours or that of a third party) or any other law, including but not limited to those relating to patent, trademark, confidentiality, data protection, obscenity, defamation, libel, then please read our [Takedown Policy](#) and [contact the service](#) immediately

RULES FOR MAPPING A CONCEPTUAL MODEL  
ONTO VARIOUS  
DATA BASE MANAGEMENT SYSTEMS.

Volume 2

A thesis submitted for the degree of Doctor of Philosophy

by

Jayasri Chaudhuri

University of Aston in Birmingham

February 1988

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no quotation from the thesis and no information derived from it may be published without the author's prior, written consent.

## TABLE OF CONTENTS

Appendix C	PROGRAM LISTINGS .....	1
	ENTEST.P .....	1
	RELTEST.P .....	6
	FUNCTEST.P .....	16
	ANALYSIS.P .....	29
	RELMAP.P .....	40
	CODMAP.P .....	52
	LABEL.P .....	64
	ING.P .....	79
	MIM.P .....	92
Appendix D	TYPICAL RUNS OF PROGRAMS .....	105
	A Typical Run of the Program ENTEST.P .....	105
	A Typical Run of the Program RELTEST.P .....	107
	A Typical Run of the Program FUNCTEST.P .....	109
Appendix E	OUTPUTS FROM PROGRAMS .....	111
	Output of ENTEST.P .....	111
	Output of RELTEST.P .....	116
	Output of FUNCTEST.P .....	118
	Output of ANALYSIS.P .....	121
	Output of RELMAP.P .....	129
	Output of CODMAP.P .....	132
	Output of LABEL.P .....	140
	Output of ING.P .....	147
	Output of MIM.P .....	151
Appendix F	LOGICAL MODEL DEFINITION FOR INGRES .....	155

## APPENDIX C

### PROGRAM LISTINGS

#### ENTEST.P

{THIS PROGRAM DOCUMENTS THE ENTITIES IDENTIFIED BY THE DESIGNER}

```
program entest (input, output, entfile);

const maxstrlength = 20;
      maxnoent = 40;
      maxnoatt = 20;
      primarykey = 'PRIMARY KEY          ';

type str = array [1..maxstrlength] of char;
      attrib = str;
      attributes = array [0..maxnoatt] of attrib;
      entity = record
          ename: str; {NAME OF THE ENTITY}
          keycount : integer; {THE NUMBER OF ATTRIBUTES THAT MAKE THE KEY}
          noatts: integer;    {TOTAL NUMBER OF ATTRIBUTES}
          entatt: attributes {NAMES OF THE ATTRIBUTES}
          end;

var entchart: array [1..maxnoent] of entity; {THE LIST OF ENTITIES}
    noofentities: integer; {TOTAL NO OF EXISTING ENTITIES}
    entfile: text;
    i, j, k, l: integer;
    nonewent: integer; {NUMBER OF NEW ENTITIES TO BE INSERTED}

procedure readstr (var f: text; var s: str); {READS A STRING OF CHARACTERS}
    var ptr: integer;
    begin
        ptr := 0;
        while not eoln (f) and (ptr < maxstrlength) do
            begin
                ptr := ptr + 1;
                read (f, s[ptr])
            end;
        while ptr < maxstrlength do
            begin
                ptr := ptr + 1;
                s[ptr] := ' '
            end
        end;
    end;

procedure writestr (var f: text; var s: str); {WRITES A STRING OF CHARACTERS}
    var i: integer;
    begin
        for i := 1 to maxstrlength do
            write (f, s[i]);
        end;
    end;
```

```

begin
{READ IN THE LIST OF ENTITIES FROM THE FILE INTO THE MEMORY}
{FOR EACH ENTITY READ ITS NAME, NUMBER OF ATTRIBUTES IN THE KEY}
{TOTAL NUMBER OF ATTRIBUTES, NAME OF THE ATTRIBUTES}
reset (entfile, 'hospent');
readln (oldent, noofentities);
for i := 1 to noofentities do
  with entchart[i] do
    begin
      readln (oldent);
      readstr (oldent, ename);
      readln (oldent);
      readln (oldent, keycount);
      readln (oldent, noatts);
      readln (oldent);
      for l := 0 to keycount do readstr (oldent, entatt[l]);
      readln (oldent);
      readln (oldent);
      k := 1;
      for j := (keycount + 1) to noatts do
        begin
          readstr (oldent, entatt[j]);
          k := k+1;
          if k > 4 then
            begin
              readln (oldent);
              k := 1
            end
          end;

          readln (oldent);
          readln (oldent)
        end;

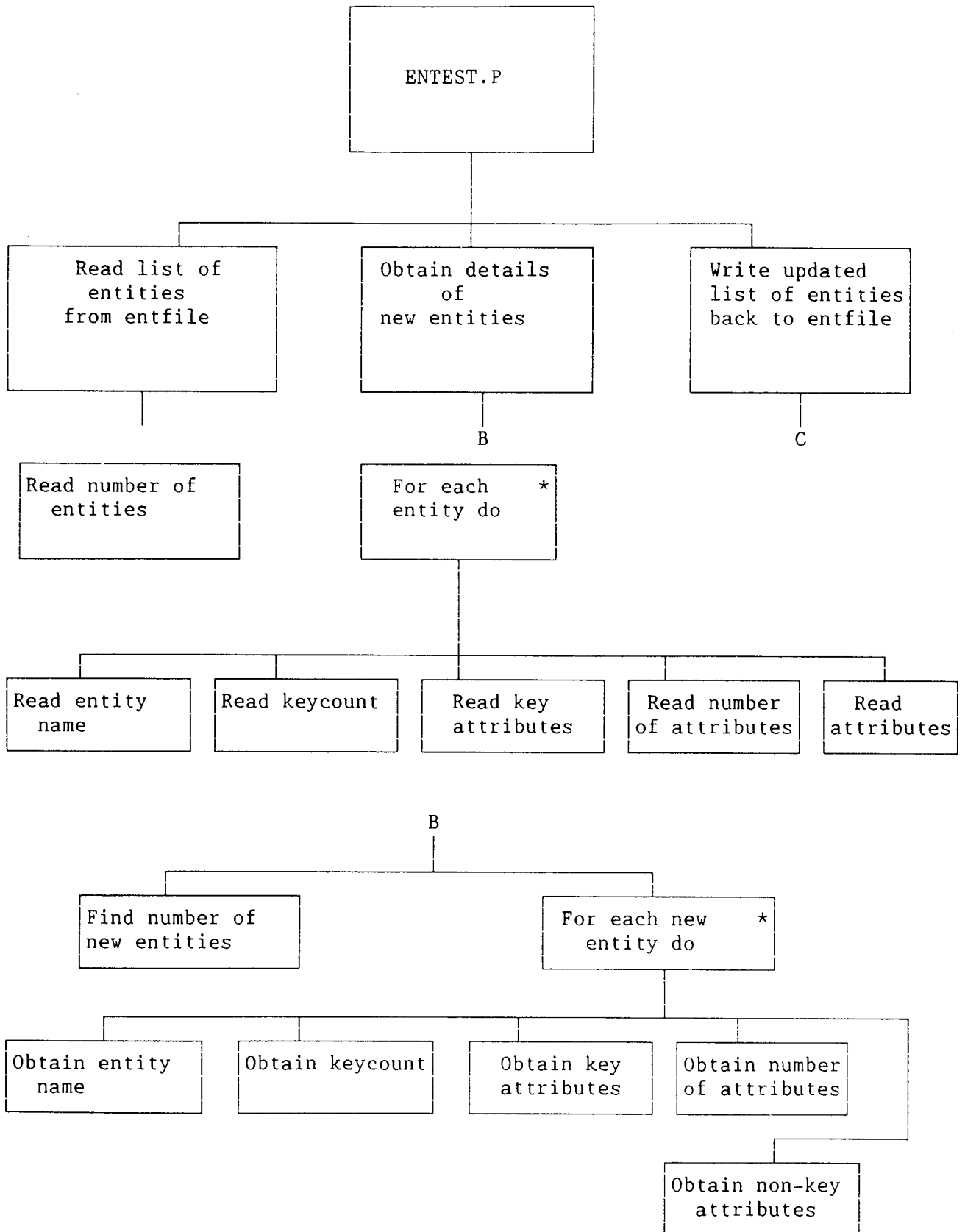
      readln (oldent);
      readln (oldent)
    end;

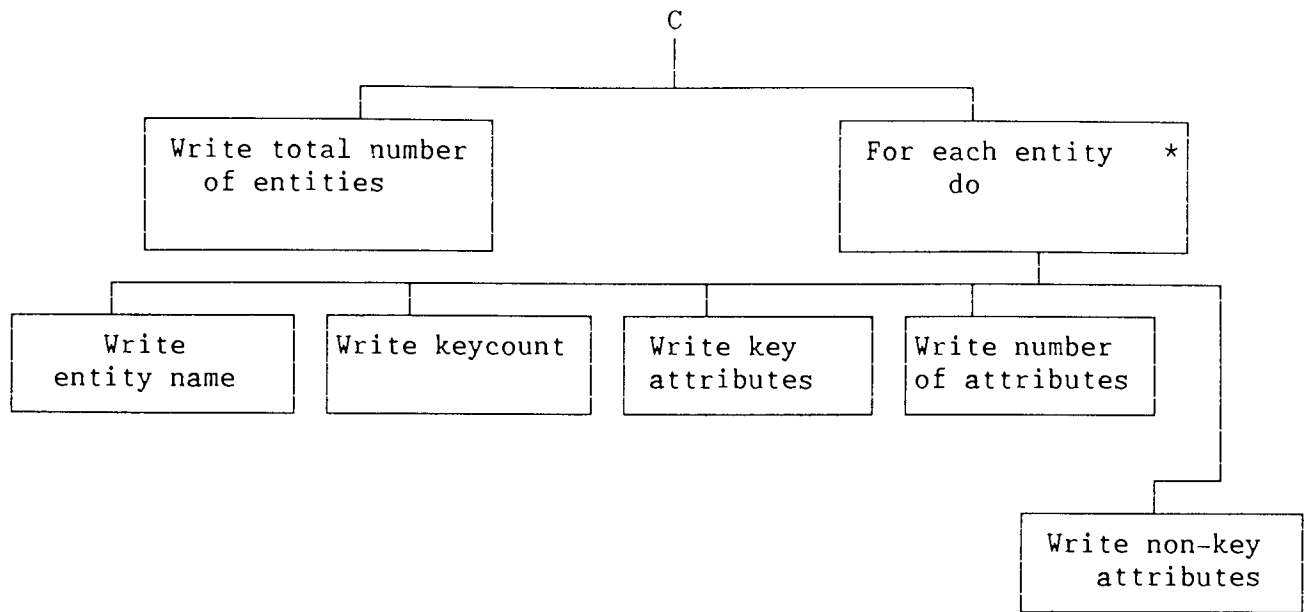
{READ IN THE NEW ENTITIES FROM THE TERMINAL}
writeln ('Number of new entities');
readln (nonewent); {NOTE THE NUMBER OF NEW ENTITIES TO BE INSERTED}
for i := 1 to nonewent do
  begin
    noofentities := noofentities + 1{INCREASE THE TOTAL NUMBER OF ENTITIES}
    with entchart[noofentities] do
      begin
        writeln;
        writeln ('Entity name');
        readstr (input, ename);
        readln;
        writeln ('NO of attributes in the primary key');
        readln (input, keycount);
        writeln ('No of attributes');
        readln (input, noatts);
        entatt[0] := primarykey;
        writeln ('Input the attributes one per line');
        for j := 1 to noatts do
          begin
            readstr (input, entatt[j]);
            readln
          end
        end
      end
    end;
end;

```

```
{WRITE THE ENTITIES BACK TO THE ENTITY FILE}
rewrite (entfile, 'hospent');
writeln (newent, noofentities);
for i := 1 to noofentities do
  with entchart[i] do
    begin
      writeln (newent);
      writestr (newent, ename);
      writeln (newent);
      writeln (newent, keycount);
      writeln (newent, noatts);
      writeln (newent);
      for l := 0 to keycount do writestr (newent, entatt[l]);
      writeln (newent);
      writeln (newent);
      k := 1;
      for j := (keycount + 1) to noatts do
        begin
          writestr (newent, entatt[j]);
          k := k+1;
          if k > 4 then
            begin
              writeln (newent);
              k:= 1
            end
          end;
        writeln (newent);
        writeln (newent)
      end;
    end;
end.
end.
```

ENTEST.P







## RELTEST.P

{THIS PROGRAM DOCUMENTS THE RELATIONSHIPS IDENTIFIED BY THE DESIGNER}

```
program relationtest (input, output, relfile);

const maxstrlength = 20;
      maxnorel = 40;
      maxnoent = 40;
      maxnoatt = 20;
      maxrelatt = 10;
      awith = 'WITH';
      awithout = 'WITHOUT';

type str = array [1..maxstrlength] of char;
relstat = (weth, wethout); {DENOTES IF RELATIONSHIP HAS ANY ATTRIBUTE}
relation = record
  rname: str; {NAME OF THE RELATIONSHIP}
  entitya: integer; {POINTER TO ENTITYA }
  degenta: char; {DEGREE OF ENTITYA}
  membshpa: char; {MEMBERSHIP CLASS OF ENTITYA}
  entityb: integer; {POINTER TO ENTITYB}
  degentb: char; {DEGREE OF ENTITYB}
  membshpb: char; {MEMBERSHIP CLASS OF ENTITYB}
  case rs : relstat of
    weth: (norelatt : integer; {NUMBER OF ATTRIBUTES IF ANY}
           relatt : array [1..maxrelatt] of str); {ATTRIBUTES}
    wethout: ()
  end;

attributes = array [0..maxnoatt] of str;

entity = record
  ename: str;
  keycount: integer;
  noatts: integer;
  entatt : attributes
end;

var relchart: array [1..maxnorel] of relation; {THE LIST OF RELATIONSHIPS}
    entchart : array [1..maxnoent ] of entity;
    noofrelations: integer;
    relfile: text;
    relform : text;
    i, j, k, l : integer;
    noelfile: integer;
    noofentities: integer;
    entfile : text;
    tempbuff : str;
    found : boolean;
    correct : boolean;
    answer : str;
```

```

procedure readstr (var f: text; var s: str);
  var ptr: integer;
  begin
    ptr := 0;
    while not eoln (f) and (ptr < maxstrlength) do
      begin
        ptr := ptr + 1;
        read (f, s[ptr]);
      end;
    while ptr < maxstrlength do
      begin
        ptr := ptr + 1;
        s[ptr] := ' '
      end
    end;
end;

procedure writestr (var f: text; var s: str);
  var i: integer;
  begin
    for i := 1 to maxstrlength do
      write (f, s[i]);
    end;
end;

function equalstr(a, b: str): boolean; {COMPARES TWO STRINGS}
  var ptr : integer;
      equal : boolean;
  begin
    equal := true;
    ptr := 0;
    while equal and (ptr < maxstrlength ) do
      begin
        ptr := ptr + 1;
        if a[ptr] <> b[ptr] then
          equal := false
        end;
      equalstr := equal
    end;
end;

procedure findent (var int : integer); {SEARCHES IF AN ENTITY EXISTS}
  var l : integer;
  begin
    l := 0;
    found := false;
    while (not found) and (l < noofentities) do
      begin
        l := l+ 1;
        if equalstr (tempbuff, entchart[l].ename) then
          begin
            int := l;
            found := true
          end
        end
      end;
end;

```

```

begin
{READ IN THE ENTITY FILE}
reset (entfile, 'hospent');
readln (entfile, noofentities);
for i := 1 to noofentities do
  with entchart[i] do
    begin
      readln (entfile);
      readstr (entfile, ename);
      readln (entfile);
      readln (entfile, keycount);
      readln (entfile, noatts);
      readln (entfile);
      for l := 0 to keycount do readstr (entfile, entatt[l]);
      readln (entfile);
      readln (entfile);
      k := 1;
      for j := (keycount + 1) to noatts do
        begin
          readstr (entfile, entatt[j]);
          k := k+1;
          if k > 4 then
            begin
              readln (entfile);
              k := 1
            end
          end;
        readln (entfile);
        readln (entfile)
      end;
end;

{READ IN THE RELATIONSHIPS FROM THE FILE INTO THE MEMORY}
{FOR EACH RELATIONSHIP READ ITS NAME, POINTERS TO THE PARTICIPATING ENTITIES}
{DEGREE AND MEMBERSHIP CLASSES OF THE ENTITIES}
{ATTRIBUTES OF THE ENTITIES}

reset (relfile, 'hosprel');
readln (relfile, noofrelations);
for i := 1 to noofrelations do
  with relchart[i] do
    begin
      readln (relfile);
      readstr (relfile, rname);
      readln (relfile);
      readstr (relfile, tempbuff);
      findent (entitya);
      readln (relfile, degenta);
      readln (relfile, membshpa);
      readstr (relfile, tempbuff);
      findent (entityb);
      readln (relfile, degentb);
      readln (relfile, membshpb);
      readstr (relfile, tempbuff);
      if tempbuff = awith then rs := weth
      else rs := wewithout;
      readln (relfile);
    end;
  end;
end;

```

```

    if rs = weth then
      begin
        readln (relfile, norelatt);
        k := 1;
        for j := 1 to norelatt do
          begin
            readstr(relfile, relatt[j]);
            k := k+1;
            if k > 4 then
              begin
                readln (relfile);
                k := 1
              end
            end
          end
        end
      end
    end;

```

```

{READ IN FROM THE TERMINAL}
writeln (' Number of new relationships');
readln (norelfile); {NOTE THE NUMBER OF NEW RELATIONSHIPS TO BE INSERTED}
for i := 1 to norelfile do
  begin
    noofrelations := noofrelations + 1; {INCREASE RELATIONSHIP TOTAL}
    with relchart[noofrelations] do
      begin
        writeln;
        writeln ('Relationship name');
        readstr (input, rname);
        readln;
        writeln ('First entity');
        repeat
          readstr (input, tempbuff);
          readln;
          findent (entitya);
          if (not found) then writeln ('THIS ENTITY DOES NOT EXIST')
          until found = true;
          writeln ('Degree of first entity 1 or m');
          readln (input, degenta);
          writeln ('Membership of first entity o for obligatory n for
non-obligatory');
          readln (input, membshpa);
          writeln ('Second entity');
          repeat
            readstr (input, tempbuff);
            readln;
            findent (entityb);
            if (not found) then writeln ('THIS ENTITY DOES NOT EXIST')
            until found = true;
            writeln ('Degree of second entity 1 or n');
            readln (input, degentb);
            writeln ('Membership of second entity o or n');
            readln (input, membshpb);

```

```

if membshpb = 'o' then
  begin

    {CHECK IF THE KEY OF ENTITYA FORMS PART OF THE KEY OF ENTITYB}
    {IF IT IS THEN ASSIGN MEMBERSHIP CLASS S TO ENTITYB}

    found := false;
    j := 0;
    while (not found) and (j < entchart[entityb].keycount) do
      begin
        j := j+1;
        if equalstr (entchart[entitya].entatt[1], entchart[entityb].
                    entatt[j]) then
          found := true
        end;
        if found then membshpb := 's'
        end;
correct := false;
repeat
  writeln ('State WITH or WITHOUT Attributes');
  readstr (input, answer);
  readln;
  if answer = awith then
    begin
      rs := weth;
      correct := true
    end
  else if answer = awithout then
    begin
      rs:= wewithout;
      correct := true
    end
  else writeln ('INVALID RESPONSE')
until correct = true;
if rs = weth then
  begin
    writeln ('Number of Attributes');
    readln (input, norelatt);
    writeln ('State one attriute per line');
    for j := 1 to norelatt do
      begin
        readstr (input, relatt[j]);
        readln
      end
    end
  end
end
end;

```

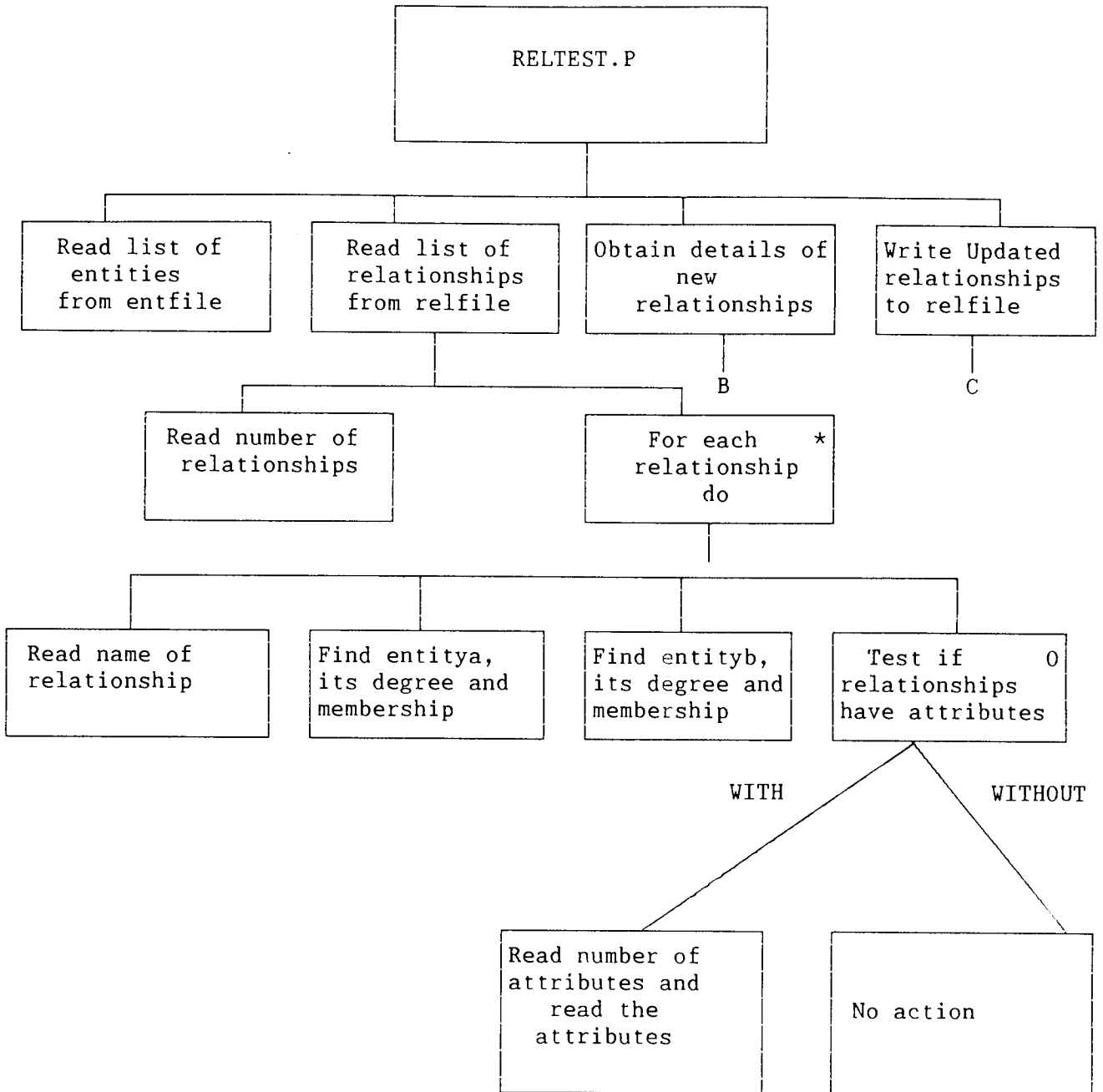
```
{WRITE THE RELATIONSHIP BACK TO THE RELFILE}

rewrite (relfile, 'hosprel');
writeln (relfile, noofrelations);
for i := 1 to noofrelations do
  with relchart[i] do
    begin
      writeln (relfile);
      writestr (relfile, rname);
      writeln (relfile);
      writestr (relfile, entchart[entitya].ename);
      writeln (relfile, degenta);
      writeln (relfile, membshpa);
      writestr ( relfile, entchart[entityb].ename);
      writeln (relfile, degentb);
      writeln (relfile, membshpb);
      if rs = weth then write(relfile, awith)
      else write (relfile, awithout);
      writeln (relfile );
      if rs = weth then
        begin
          writeln (relfile, norelatt);
          k := 1;
          for j := 1 to norelatt do
            begin
              writestr (relfile, relatt[j]);
              k := k+1;
              if k> 4 then
                begin
                  writeln(relfile);
                  k := 1
                end
            end
          end
        end
      end
    end;
end;
```

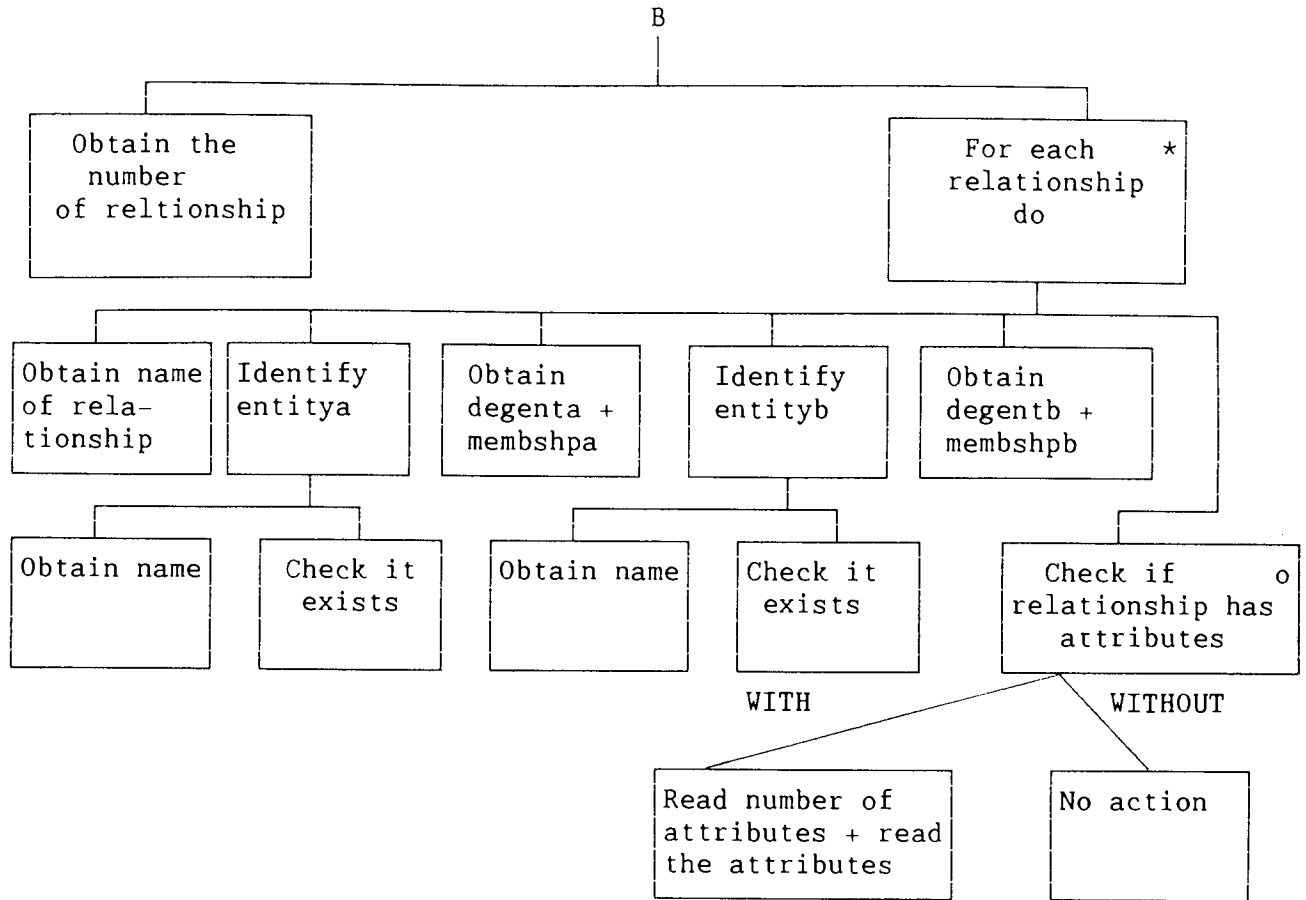
{WRITE THE RELATION IN THE FORMATTED FORM}

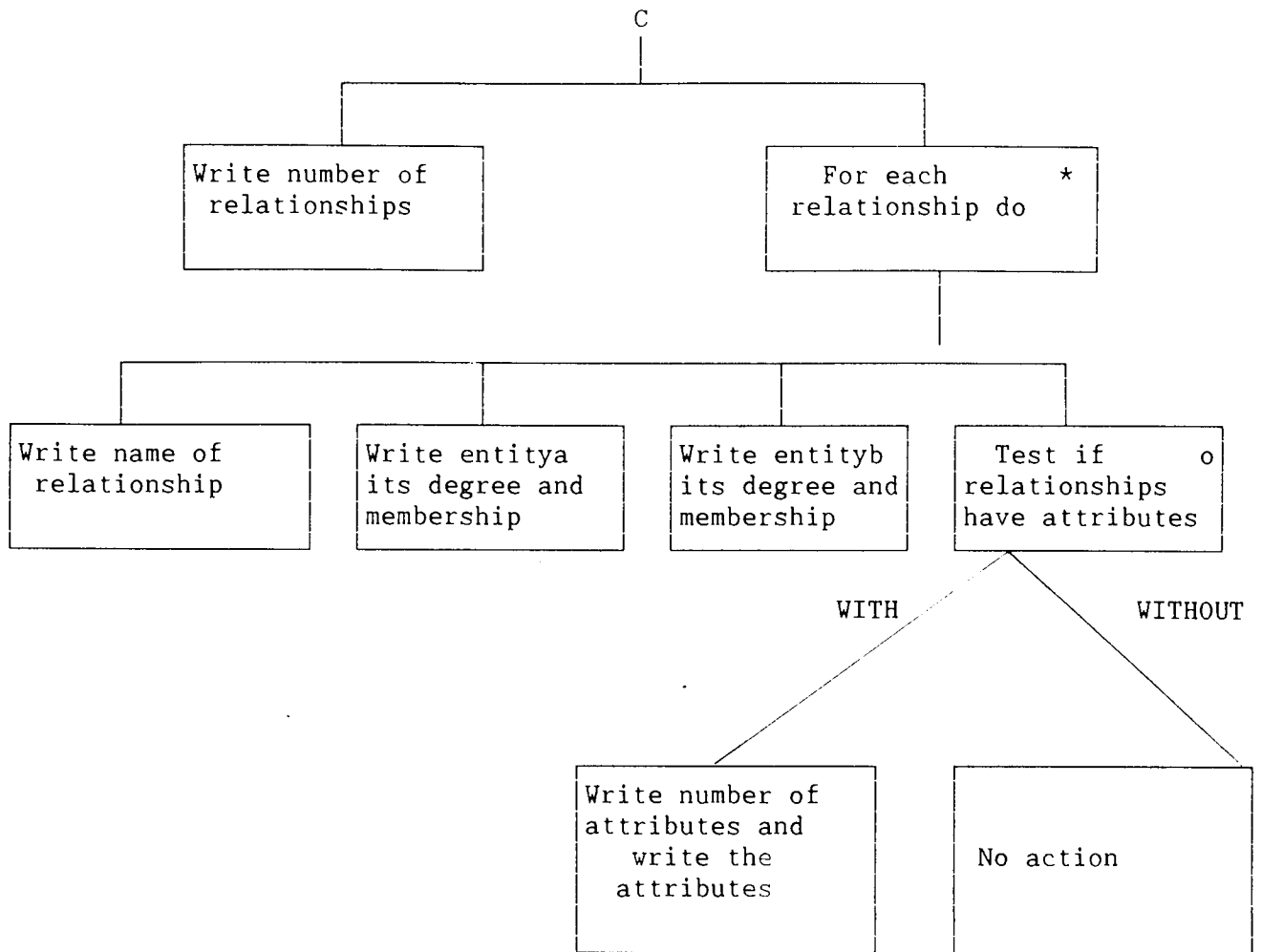
```
rewrite (relform, 'format-rel');
write(relform, 'RELATION-NAME      ');
write(relform, 'ENTITY-A          ');
write(relform, 'ENTITY-B          ');
write(relform, 'DEGREE-A          ');
write(relform, 'DEGREE-B          ');
write(relform, 'MEMB-A            ');
writeln(relform, 'MEMB-B');
writeln (relform);
for i := 1 to noofrelations do
  with relchart[i] do
    begin
      writestr (relform, rname);
      writestr (relform, entchart[entitya].ename);
      writestr (relform, entchart[entityb].ename);
      write (relform, '          ');
      write(relform, degenta);
      write (relform, '          ');
      write (relform, '          ');
      write (relform, degentb);
      write (relform, '          ');
      write (relform, '          ');
      write (relform, membshpa);
      write(relform, '          ');
      writeln(relform, membshpb);
      writeln (relform)
    end
  end.
end.
```

RELTEST.P









FUNCTEST.P

{THIS PROGRAM DOCUMENTS INFORMATION ABOUT THE IDENTIFIED FUNCTIONS}

```
program functest (input, output, funcfile);
```

```
const maxstrlength = 20;
      maxnofunc = 20;
      maxnoacc = 15;
      maxselcrit = 4;
      maxnoatt = 15;
      maxnoent = 40;
      maxnorel = 40;
      primary = 'PRIMARY          ';
      secondary = 'SECONDARY      ';
      ent = 'ENTITY                ';
      reln = 'RELATIONSHIP        ';
      selbyrel = 'SELECT.BY.RELATION  ';
      selbyatt = 'SELECT.BY.ATTRIBUTES';
      selbykey = 'SELECT.BY.KEY      ';
      range = 'RANGE              ';
      equijoin = 'EQUIJOIN          ';
      awith = 'WITH';
      awithout = 'WITHOUT';
      maxrelatt = 10;
```

```
type str = array [1..maxstrlength] of char;
      relstat = (weth, without);
      selopts = (sbyr, sbya, sbyp);
      selectdet = record
          case opts : selopts of
              sbyr : (relpnt : integer);
              sbyp : (ppnt : char);
              sbya : (apnt : integer;
                     aclaus : char);
          end;
      accent = record
          entname : integer; {POINTER TO THE ENTITY}
          eselectcrit : selectdet {SELECTION CRITERIA}
          end;
      entacc = array [1..maxnoacc] of accent;

      func = record
          funcname : str; {NAME OF THE FUNCTION}
          funcfreq : integer; {FREQUENCY OF THE FUNCTION}
          funcstatus : integer; {STATUS OF THE FUNCTION}
          noacc : integer; {NUMBER OF ENTITIES ACCESSED}
          entarr : entacc {LIST SHOWING THE ENTITIES ACCESED}
          end;

      attributes = array [0..maxnoatt] of str;

      entity = record
          ename: str;
          keycount: integer;
          noatts : integer;
          entatt : attributes
          end;
```

```

relation = record
    rname: str;
    entitya: integer;
    degenta: char;
    membshpa : char;
    entityb: integer;
    degentb: char;
    membshpb: char;
    case rs: relstat of
        weth: (norelatt : integer;
              relatt : array [1..maxrelatt] of str);
        without : ()
    end;

var funcchart: array [1..maxnofunc] of func;
    entchart : array [1..maxnoent] of entity;
    relchart : array [1..maxnorel] of relation;
    nooffunc: integer;
    funcfile: text;
    entfile : text;
    relfile : text;
    i, j, k, l: integer;
    noofnewfunc: integer;
    noofentities : integer;
    noofrelations : integer;
    entindex : integer;
    tempbuff : str;
    found : boolean;

procedure readstr (var f: text; var s: str);
    var ptr: integer;
    begin
        ptr := 0;
        while not eoln (f) and (ptr < maxstrlength) do
            begin
                ptr := ptr + 1;
                read (f, s[ptr])
            end;
        while ptr < maxstrlength do
            begin
                ptr := ptr + 1;
                s[ptr] := ' '
            end
        end;
    end;

procedure writestr (var f: text; var s: str);
    var i: integer;
    begin
        for i := 1 to maxstrlength do
            write (f, s[i]);
        end;
    end;

```

```

function equalstr (a, b: str ) : boolean;
  var ptr : integer;
      equal : boolean;
  begin
    equal := true;
    ptr := 0;
    while equal and (ptr < maxstrlength ) do
      begin
        ptr := ptr + 1;
        if a[ptr] <> b[ptr] then
          equal := false;
        end;
      end;
    equalstr := equal
  end;

```

```

procedure findent (var int : integer);
  var l : integer;
  begin
    l := 0;
    found := false;
    while (not found) and (l < noofentities ) do
      begin
        l := l+1;
        if equalstr (tempbuff, entchart[l].ename) then
          begin
            int := l;
            found := true
          end
        end
      end;
    end;

```

```

procedure findrel (var int : integer); {SEARCHES IF A RELATIONSHIP EXISTS}
  var l : integer;
  begin
    l := 0;
    found := false;
    while (not found) and (l < noofentities) do
      begin
        l := l+1;
        if equalstr (tempbuff, relchart[l].rname) then
          begin
            int := l;
            found := true
          end
        end
      end
    end;

```

```

procedure findatt (var int : integer; var entindex : integer);
{SEARCHES IF AN ATTRIBUTE EXISTS}
  var l : integer;
  begin
  l := 0;
  found := false;
  while (not found) and (l < entchart[entindex].noatts) do
  begin
  l := l+1;
  if equalstr (tempbuff, entchart[entindex].entatt[l]) then
  begin
  int := l;
  found := true
  end
  end
  end;

```

```

procedure findrelatt (var int: integer; var relindex : integer);
{SEARCHES IF AN ATTRIBUTE OF A RELATIONSHIP EXISTS}
  var l : integer;
  begin
  l := 0;
  found := false;
  while (not found) and (l < relchart[relindex].norelatt ) do
  begin
  l := l + 1;
  if equalstr (tempbuff, relchart[relindex].relatt[l]) then
  begin
  int := l;
  found := true
  end
  end
  end;

```

```
begin
```

```
{READ IN THE ENTITY FILE}
```

```

reset (entfile, 'hospent');
readln (entfile, noofentities);
for i := 1 to noofentities do
  with entchart [i] do
  begin
  readln (entfile);
  readstr (entfile, ename);
  readln (entfile);
  readln (entfile, keycount);
  readln (entfile, noatts);
  readln (entfile);
  for l := 0 to keycount do readstr (entfile, entatt[l]);
  readln (entfile);
  readln (entfile);
  k := 1;

```

```

    for j := (keycount + 1) to noatts do
      begin
        readstr (entfile, entatt[j]);
        k := k+1;
        if k > 4 then
          begin
            readln (entfile);
            k := 1
          end
        end;
      readln (entfile);
      readln (entfile)
    end;

{READ IN THE RELATION FILE}

reset (relfile, 'hosprel');
readln (relfile, noofrelations );
for i := 1 to noofrelations do
  with relchart[i] do
    begin
      readln (relfile);
      readstr (relfile, rname);
      readln (relfile);
      readstr (relfile, tempbuff);
      findent (entitya);
      readln (relfile, degenta);
      readln (relfile, membshpa);
      readstr (relfile, tempbuff);
      findent (entityb);
      readln (relfile, degentb);
      readln (relfile, membshpb);
      readstr (relfile, tempbuff);
      if tempbuff = awith then rs := weth
      else rs := without;
      readln (relfile);
      if rs = weth then
        begin
          readln (relfile, norelatt);
          k := 1;
          for j := 1 to norelatt do
            begin
              readstr (relfile, relatt[j]);
              k := k + 1;
              if k > 4 then
                begin
                  readln (relfile);
                  k := 1
                end
              end
            end
          end
        end;
      end;
    end;
  end;
end;

```

```
{READ IN THE FUNCTIONS FROM THE FILE CONTAINING THE LIST OF FUNCTIONS}
```

```
reset (funcfile, 'hosfunc');
readln (funcfile, nooffunc);
for i := 1 to nooffunc do
  with funcchart[i] do
```

```
{FOR EACH FUNCTION NOTE ITS NAME, STATUS, FREQUENCY AND NUMBER OF ENTITIES
ACCESSED}
```

```
  begin
    readln (funcfile);
    readstr (funcfile, funcname);
    readstr (funcfile, tempbuff);
    if equalstr (tempbuff, primary) then funcstatus := 1
    else funcstatus := 2;
    readln (funcfile, funcfreq);
    readln (funcfile, noacc);
```

```
{FOR EACH ENTITY ACCESSED NOTE ITS NAME, HOW IT IS SELECTED}
{IDENTIFY THE RELATION OR ATTRIBUTE/S USED TO SELECT THE ENTITY}
{IF SELECTED BY ATTRIBUTE NOTE WHETHER RANGE OR EQUALITY CLAUSE WAS USED}
```

```
  for j := 1 to noacc do
    with entarr[j] do
      begin
        readstr (funcfile, tempbuff);
        findent (entname);
        readstr (funcfile, tempbuff);
        readln(funcfile);
        if equalstr (tempbuff, selbyrel) then
          eselectcrit.opts := sbyr
        else
          if equalstr (tempbuff, selbyatt) then
            eselectcrit.opts := sbya
          else
            eselectcrit.opts := sbyp;
          if eselectcrit.opts = sbyr then
            begin
              readstr (funcfile, tempbuff);
              readln (funcfile);
              findrel (eselectcrit.relpnt)
            end
          else
            if eselectcrit.opts = sbya then
              begin
                readstr (funcfile, tempbuff);
                findatt (eselectcrit.apnt, entname);
                readstr (funcfile, tempbuff);
                if equalstr (tempbuff, range) then
                  eselectcrit.aclaus := 'r'
                else
                  eselectcrit.aclaus := 'e';
                readln(funcfile)
              end
            end
```



```

        else
            begin
                for k := 1 to entchart[entname].keycount do
                    readstr (funcfile, tempbuff);
                    readln (funcfile)
                end;
            readln (funcfile)
        end
    end;
end;

```

```

{FIND OUT THE NUMBER OF NEW FUNCTIONS TO BE INSERTED}
{FOR EACH FUNCTION NOTE THE DETAILS DESCRIBING THE FUNCTION}

```

```

writeln ('Number of new function');
readln (noofnewfunc);
for i := 1 to noofnewfunc do
    begin
        nooffunc := nooffunc + 1;
        with funcchart[nooffunc] do
            begin
                writeln;
                writeln ('Function name');
                readstr ( input, funcname );
                readln;
                writeln ('Status of function');
                readstr (input, tempbuff);
                if equalstr (tempbuff, primary ) then funcstatus := 1
                else funcstatus := 2;
                readln;
                writeln ('Frequency of the function' );
                readln (input, funcfreq);
                writeln ('Number of ENTITY accessed');
                readln ( input, noacc);
                for j := 1 to noacc do
                    with entarr[j] do
                        begin

```

```

{CHECK WHETHER THE ENTITY EXISTS}

writeln ('Name of entity');
repeat
readstr (input, tempbuff) ;
readln;
findent( entname);
if (not found) then
  writeln ('THIS ENTITY DOES NOT EXIST. TRY AGAIN')
until found = true;
writeln ('State whether SELECT.BY.KEY/RELATION/ATTRIBUTES');
readstr(input,tempbuff);
if equalstr (tempbuff, selbyrel) then
  eselectcrit.opts := sbyr
else if equalstr (tempbuff, selbyatt) then
  eselectcrit.opts := sbya
else
  eselectcrit.opts := sbyp;
readln;
if eselectcrit.opts = sbyr then
  begin
    writeln('State the name of the relation used');

    {CHECK WHETHER THE STATED RELATIONSHIP EXISTS}

    repeat
      readstr (input, tempbuff);
      readln;
      findrel (eselectcrit.relpnt);
      if (not found) then
        writeln( 'THIS RELATION DOES NOT EXIST. TRY AGAIN.')
      until found = true
    end
  else
    if eselectcrit.opts = sbya then
      begin
        writeln( 'State the attribute');
        repeat
          readstr (input, tempbuff);
          readln;
          findatt(eselectcrit.apnt,entname);
          if (not found) then
            writeln ('THIS ATTRIBUTE DOES NOT EXIST. TRY AGAIN')
          until found = true ;
          writeln ('State selection clause RANGE or EQUIJOIN. ');
          readstr (input, tempbuff);
          if equalstr (tempbuff, range) then
            eselectcrit.aclaus := 'r'
          else
            eselectcrit.aclaus := 'e'
          end
        end
      end
    end
  end
end;

```

```

{WRITE BACK ALL THE FUNCTION DESCRIPTIONS IN THE FUNCFILE};

rewrite(funcfile, 'hosppfunc');
writeln (funcfile,nooffunc);
for i := 1 to nooffunc do
with funcchart[i] do
  begin
  writeln (funcfile);
  writestr (funcfile,funcname); {WRITE FUNCTION NAME}
  if funcstatus =1 then tempbuff := primary
  else tempbuff := secondary;

  {WRITE WHETHER FUNCTION IS PRIMARY OR SECONDARY}
  writestr (funcfile, tempbuff);
  writeln (funcfile, funcfreq); {WRITE FREQUENCY OF THE FUNCTION}
  writeln (funcfile, noacc); {WRITE NUMBER OF ENTITIES ACCESSED}
  for j := 1 to noacc do
  with entarr[j] do
    begin
    writestr (funcfile, entchart[entname].ename); {WRITE THE NAME OF THE}
                                                    {ENTITY}

    {CHECK WHETHER SELECTED BY RELATIONSHIP/KEY/ATTRIBUTE}

    if eselectcrit.opts = sbyr then
      tempbuff := selbyrel
    else
    if eselectcrit.opts = sbya then
      tempbuff := selbyatt
    else
      tempbuff := selbykey;

    {WRITE WHETHER SELECTED BY KEY/ATTRIBUTE/ RELATIONSHIP}
    writestr (funcfile, tempbuff);
    writeln (funcfile);

```

```

if eselectcrit.opts = sbyr then
  {IF SELECTED BY RELATIONSHIP THEN WRITE NAME OF RELATIONSHIP}
  begin
    writestr (funcfile, relchart[eselectcrit.relpnt].rname);
    writeln (funcfile)
  end
else
if eselectcrit.opts = sbyp then

{IF SELECTED BY KEY THEN WRITE THE KEY ATTRIBUTES}

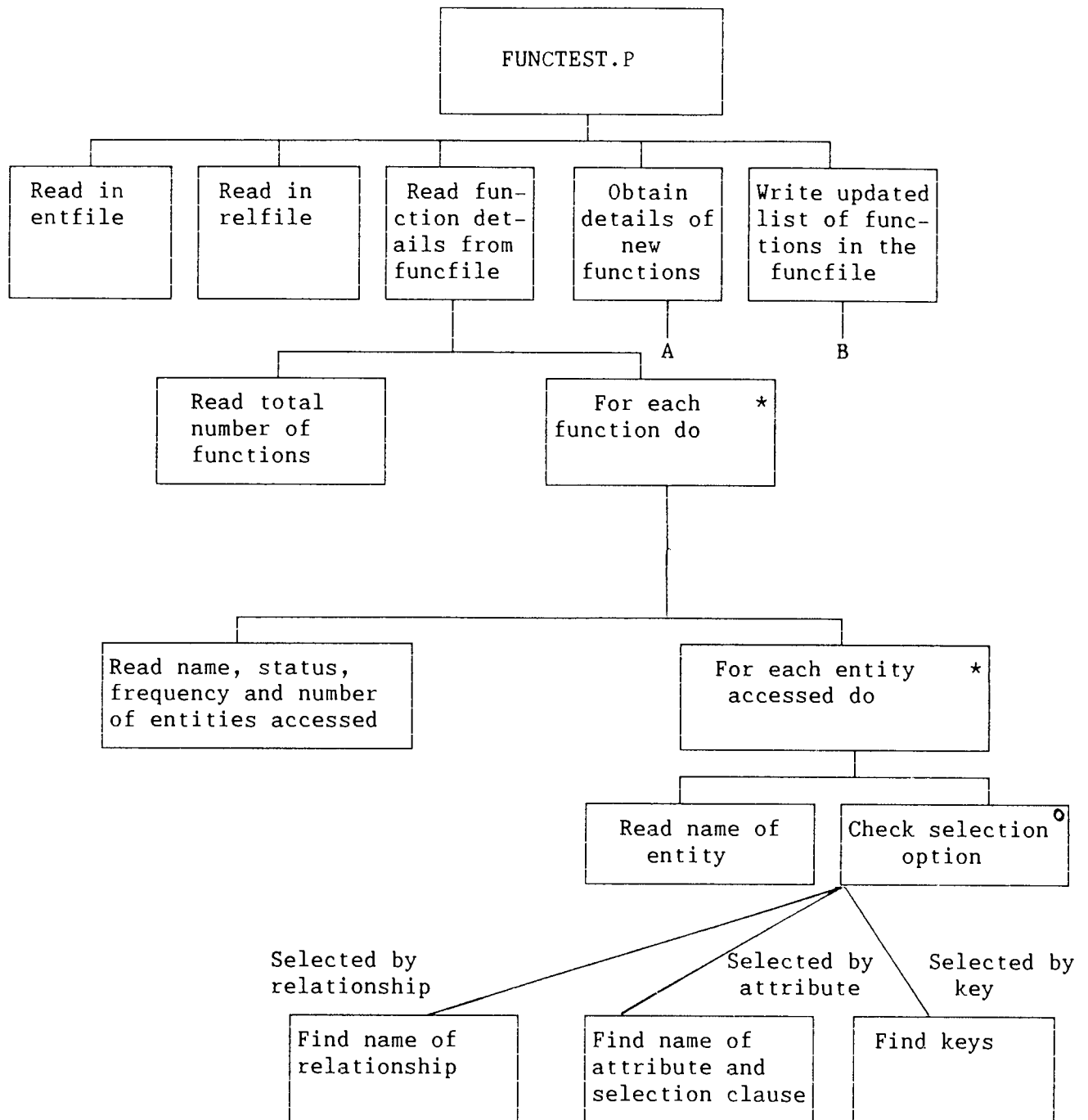
  begin
    for k := 1 to entchart[entname].keycount do
      writestr (funcfile, entchart[entname].entatt[k]);
      writeln (funcfile)
    end
  else
    begin

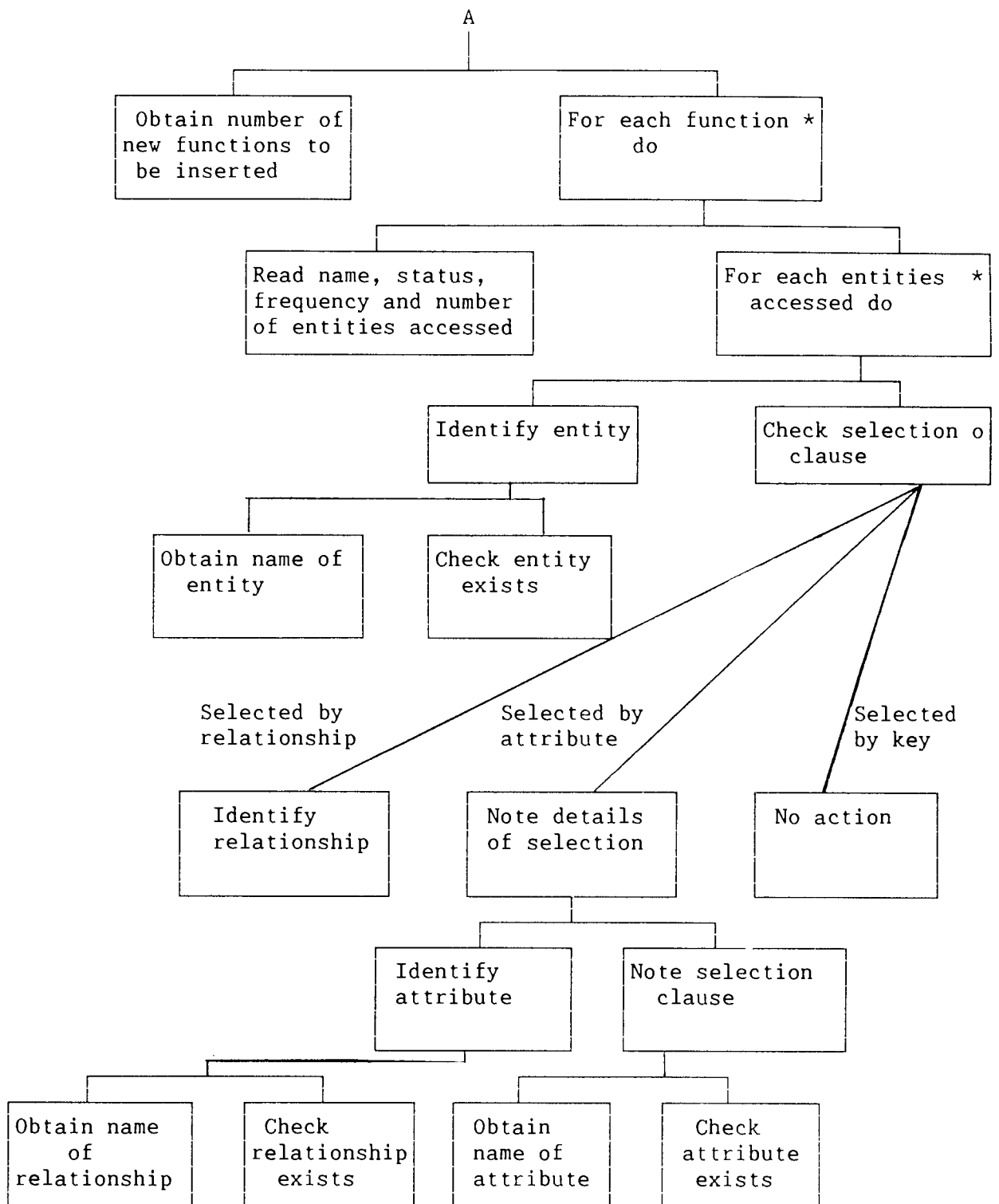
{IF SELECTED BY ATTRIBUTE THEN WRITE THE ATTRIBUTE}

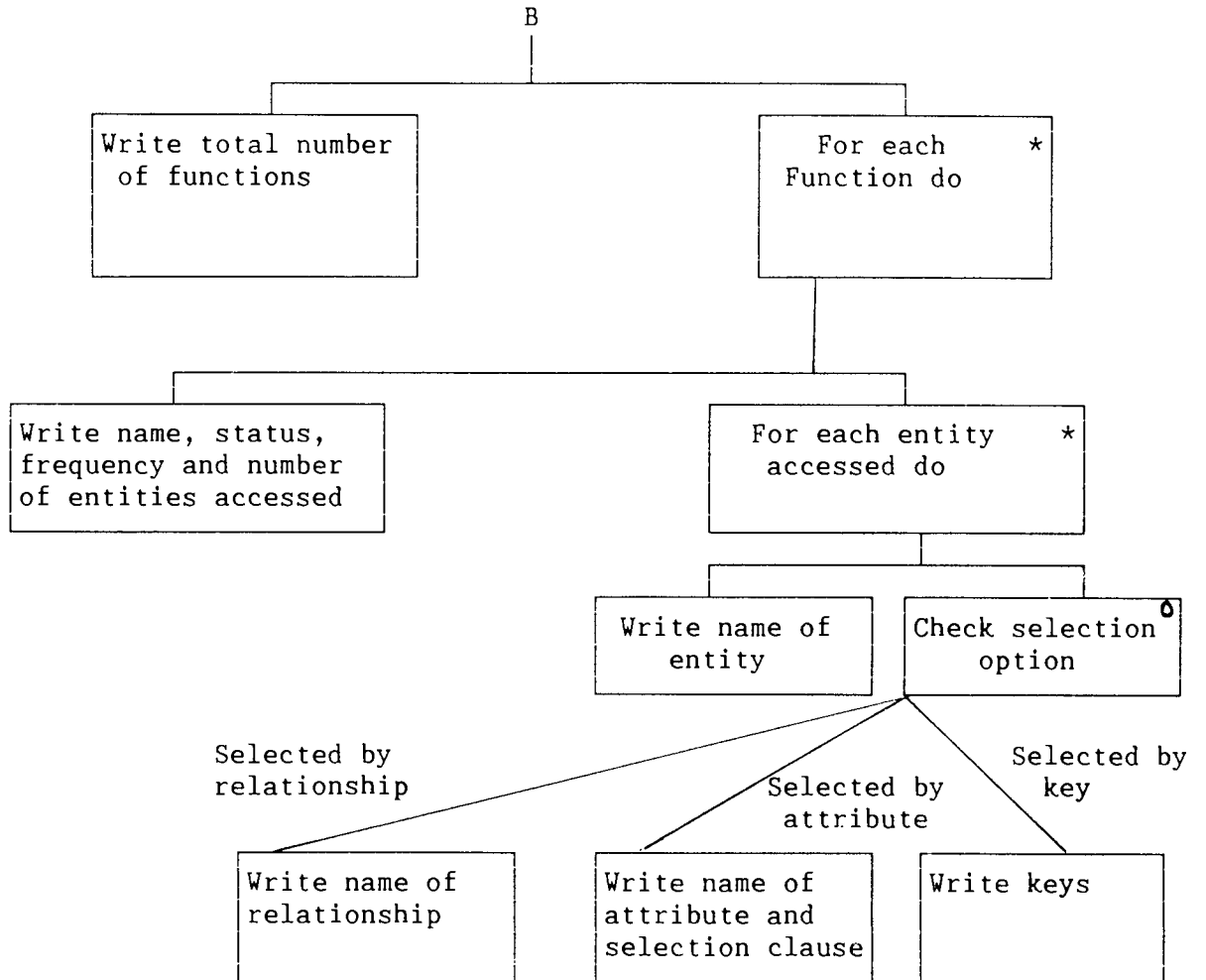
      writestr (funcfile, entchart[entname].entatt[eselectcrit.apnt]);
      if eselectcrit.aclaus = 'r' then
        tempbuff := range
      else
        tempbuff := equijoin;
      writestr (funcfile, tempbuff);  {WRITE WHETHER RANGE OR EQUIJOIN}
                                       {ACCESS}
      writeln (funcfile)
    end;
  writeln (funcfile)
end
end
end.

```

FUNCTEST.P







## ANALYSIS.P

{THIS PROGRAM ANALYSES THE RESULT OF THE FUNCTIONAL ANALYSIS}

```

program analysis (input, output, funcfile);

const maxstrlength = 20;
      maxnofunc = 20;
      maxnoacc = 15;
      maxselcrit = 4;
      maxnoatt = 15;
      maxnoent = 40;
      maxnorel = 40;
      maxoutedge = 20;
      maxinedge = 20;
      maxnoedge = 400;
      primary = 'PRIMARY          ';
      secondary = 'SECONDARY      ';
      ent = 'ENTITY                ';
      reln = 'RELATIONSHIP         ';
      selbyrel = 'SELECT.BY.RELATION  ';
      selbyatt = 'SELECT.BY.ATTRIBUTES';
      selbykey = 'SELECT.BY.KEY       ';
      awith = 'WITH                  ';
      awithout = 'WITHOUT            ';
      maxrelatt = 10;
      range = 'RANGE                ';
      equijoin = 'EQUIJOIN           ';

type str = array [1..maxstrlength] of char;
      relstat = (weth, without);
      selopts = (sbyr, sbya, sbyp);
      selectdet = record
          case opts : selopts of
              sbyr : (relpnt : integer);
              sbyp : (ppnt : char);
              sbya : (apnt : integer;
                     aclus : char);
          end;

      accent = record
          entname : integer;
          eselectcrit : selectdet
      end;

      entacc = array [1..maxnoacc] of accent;

      func = record
          funcname : str;
          funcfreq : integer;
          funcstatus : integer;
          noacc : integer;
          entarr : entacc
      end;

      attributes = array [0..maxnoatt] of str;
  
```



```

entity = record
    ename: str;
    keycount: integer;
    noatts : integer;
    entatt : attributes
end;

relation = record
    rname: str;
    entitya: integer;
    degenta: char;
    membshpa : char;
    entityb: integer;
    degentb: char;
    membshpb: char;
    case rs: relstat of
        weth: (norelatt : integer;
              relatt : array [1..maxrelatt] of str);
        without : ()
    end;

accdet = record
    rangefreq: integer; {FREQUENCY OF USAGE IN RANGE CLAUSE}
    equifreq : integer  {FREQUENCY OF USAGE IN EQUALITY CLAUSE}
end;

analdet = record
    primdet: accdet; {USAGE OF AN ATTRIBUTE IN PRIMARY FUNCTION}
    seconddet: accdet {USAGE OF AN ATTRIBUTE IN SECONDARY FUNCTION}
end;

eusagedet= record {USAGE DETAIL OF AN ENTITY}
    usagedet : array[1..maxnoatt] of analdet
end;

rusagedet = record {USAGE DETAIL OF A RELATIONSHIP}
    primfreq : integer; {FREQUENCY OF USAGE IN PRIMARY FUNCTION}
    secfreq : integer  {FREQUENCY OF USAGE IN SECONDARY FUNCTION}
end;

var funcchart: array [1..maxnofunc] of func;
    entchart : array [1..maxnoent] of entity;
    relchart : array [1..maxnorel] of relation;
    eusagemat : array [1..maxnoent] of eusagedet; {ATTRIBUTE USAGE DETAILS}
    rusagemat : array [1..maxnorel] of rusagedet; {RELATIONSHIP USAGE DETAILS}
    nooffunc: integer;
    funcfile: text;
    entfile : text;
    relfile : text;
    analysisfile : text;
    i, j, k, l: integer;
    noofnewfunc: integer;
    noofentities : integer;
    noofrelations : integer;
    entindex : integer;
    tempbuff : str;
    found : boolean;

```

```

procedure readstr (var f: text; var s: str);
  var ptr: integer;
  begin
    ptr := 0;
    while not eoln (f) and (ptr < maxstrlength) do
      begin
        ptr := ptr + 1;
        read (f, s[ptr])
      end;
    while ptr < maxstrlength do
      begin
        ptr := ptr + 1;
        s[ptr] := ' '
      end
    end;
  end;

procedure writestr (var f: text; var s: str);
  var i: integer;
  begin
    for i := 1 to maxstrlength do
      write (f, s[i]);
    end;
  end;

function equalstr (a, b: str ) : boolean;
  var ptr : integer;
      equal : boolean;
  begin
    equal := true;
    ptr := 0;
    while equal and (ptr < maxstrlength ) do
      begin
        ptr := ptr + 1;
        if a[ptr] <> b[ptr] then
          equal := false;
        end;
      end;
    equalstr := equal
  end;

procedure findent (var int : integer);
  var l : integer;
  begin
    l := 0;
    found := false;
    while (not found) and (l < noofentities ) do
      begin
        l := l+1;
        if equalstr (tempbuff, entchart[l].ename) then
          begin
            int := l;
            found := true
          end
        end
      end
    end;
  end;

```

```
procedure findrel (var int : integer);
  var l : integer;
  begin
    l := 0;
    found := false;
    while (not found) and (l < noofentities) do
      begin
        l := l+1;
        if equalstr (tempbuff, relchart[l].rname) then
          begin
            int := l;
            found := true
          end
        end
      end
    end;

procedure findatt (var int : integer; var entindex : integer);
  var l : integer;
  begin
    l := 0;
    found := false;
    while (not found) and (l < entchart[entindex].noatts) do
      begin
        l := l+1;
        if equalstr (tempbuff, entchart[entindex].entatt[l]) then
          begin
            int := l;
            found := true
          end
        end
      end
    end;

procedure findrelatt (var int: integer; var relindex : integer);
  var l : integer;
  begin
    l := 0;
    found := false;
    while (not found) and (l < relchart[relindex].norelatt ) do
      begin
        l := l + 1;
        if equalstr (tempbuff, relchart[relindex].relatt[l]) then
          begin
            int := l;
            found := true
          end
        end
      end
    end;
end;
```

```

begin

{READ IN THE ENTITY FILE}
reset (entfile, 'hospent');
readln (entfile, noofentities);
for i := 1 to noofentities do
  with entchart [i] do
    begin
      readln (entfile);
      readstr (entfile, ename);
      readln (entfile);
      readln (entfile, keycount);
      readln (entfile, noatts);
      readln (entfile);
      for l := 0 to keycount do readstr (entfile, entatt[l]);
      readln (entfile);
      readln (entfile);
      k := 1;
      for j := (keycount + 1) to noatts do
        begin
          readstr (entfile, entatt[j]);
          k := k+1;
          if k > 4 then
            begin
              readln (entfile);
              k := 1
            end
          end;
        readln (entfile);
        readln (entfile)
      end;
    end;

{READ IN THE RELATION FILE}
reset (relfile, 'hosprel');
readln (relfile, noofrelations );
for i := 1 to noofrelations do
  with relchart[i] do
    begin
      readln (relfile);
      readstr (relfile, rname);
      readln (relfile);
      readstr (relfile, tempbuff);
      findent (entitya);
      readln (relfile, degenta);
      readln (relfile, membshpa);
      readstr (relfile, tempbuff);
      findent (entityb);
      readln (relfile, degentb);
      readln (relfile, membshpb);
      readstr (relfile, tempbuff);
      if tempbuff = awith then rs := weth
      else rs := without;
      readln (relfile);
    end;
  end;
end;

```

```

if rs = weth then
  begin
  readln (relfile, norelatt);
  k := 1;
  for j := 1 to norelatt do
    begin
    readstr (relfile, relatt[j]);
    k := k + 1;
    if k > 4 then
      begin
      readln (relfile);
      k := 1
      end
    end
  end
end;

```

```

{READ IN THE FUNCTION FILE}
reset (funcfile, 'hospfunc');
readln (funcfile, nooffunc);
for i := 1 to nooffunc do
  with funcchart[i] do
    begin
    readln (funcfile);
    readstr (funcfile, funcname);
    readstr (funcfile, tempbuff);
    if equalstr(tempbuff, primary) then
      funcstatus := 1
    else funcstatus := 2;
    readln (funcfile, funcfreq);
    readln (funcfile, noacc);
    for j := 1 to noacc do
      with entarr[j] do
        begin
        readstr (funcfile, tempbuff);
        findent (entname);
        readstr (funcfile, tempbuff);
        readln (funcfile);
        if equalstr (tempbuff, selbyrel) then
          eselectcrit.opts := sbyr
        else
          if equalstr (tempbuff, selbyatt) then
            eselectcrit.opts := sbya
          else
            eselectcrit.opts := sbyp;
          if eselectcrit.opts = sbyr then
            begin
            readstr (funcfile, tempbuff);
            readln (funcfile);
            findrel (eselectcrit.relpnt)
            end
          else
            if eselectcrit.opts = sbya then
              begin
              readstr (funcfile, tempbuff);
              findatt (eselectcrit.apnt, entname);
              readstr (funcfile, tempbuff);
              if equalstr (tempbuff, range) then
                eselectcrit.aclus := 'r'

```



```

else
  {IF SELECTION CRITERIA IS THE KEY}

  if eselectcrit.opts = sbyp then
    begin
      if funcstatus = 1 then
        begin
          for k := 1 to entchart[entname].keycount do
            begin

              {UPDATE THE EQUIJOIN FREQUENCY OF THE KEY ATTRIBUTE}

              eusagemat[entname].usagedet[k].primdet.equifreq :=
                eusagemat[entname].usagedet[k].primdet.equifreq + funcfreq
            end
          end
        end
      else
        begin
          for k := 1 to entchart[entname].keycount do
            begin
              eusagemat[entname].usagedet[k].seconddet.equifreq :=
                eusagemat[entname].usagedet[k].seconddet.equifreq + funcfreq
            end
          end
        end
      end
    end
  else

    {UPDATE THE RELATIONSHIP USAGE FREQUENCY}

    begin
      if funcstatus = 1 then
        rusagemat[eselectcrit.relpt].primfreq :=
          rusagemat[eselectcrit.relpt].primfreq + funcfreq
      else
        rusagemat[eselectcrit.relpt].secfreq :=
          rusagemat[eselectcrit.relpt].secfreq + funcfreq
      end
    end
  end;

  {WRITE THE RESULT OF THE ENTITY ANALYSIS}

  rewrite (analysisfile, 'analentity');
  for i := 1 to noofentities do
    with entchart[i] do
      with eusagemat[i] do
        begin
          writeln (analysisfile);
          write (analysisfile, 'Entity-Name :-  ');
          writestr (analysisfile, ename);
          writeln (analysisfile);
          write(analysisfile, ' ');
          write(analysisfile, ' Primary functions ');
          write(analysisfile, ' Secondary functions ');
          writeln (analysisfile);
          write(analysisfile, ' ');
          write(analysisfile, ' Rfrequency ');
          write(analysisfile, ' Efrequency ');
        end
      end
    end
  end;

```

```

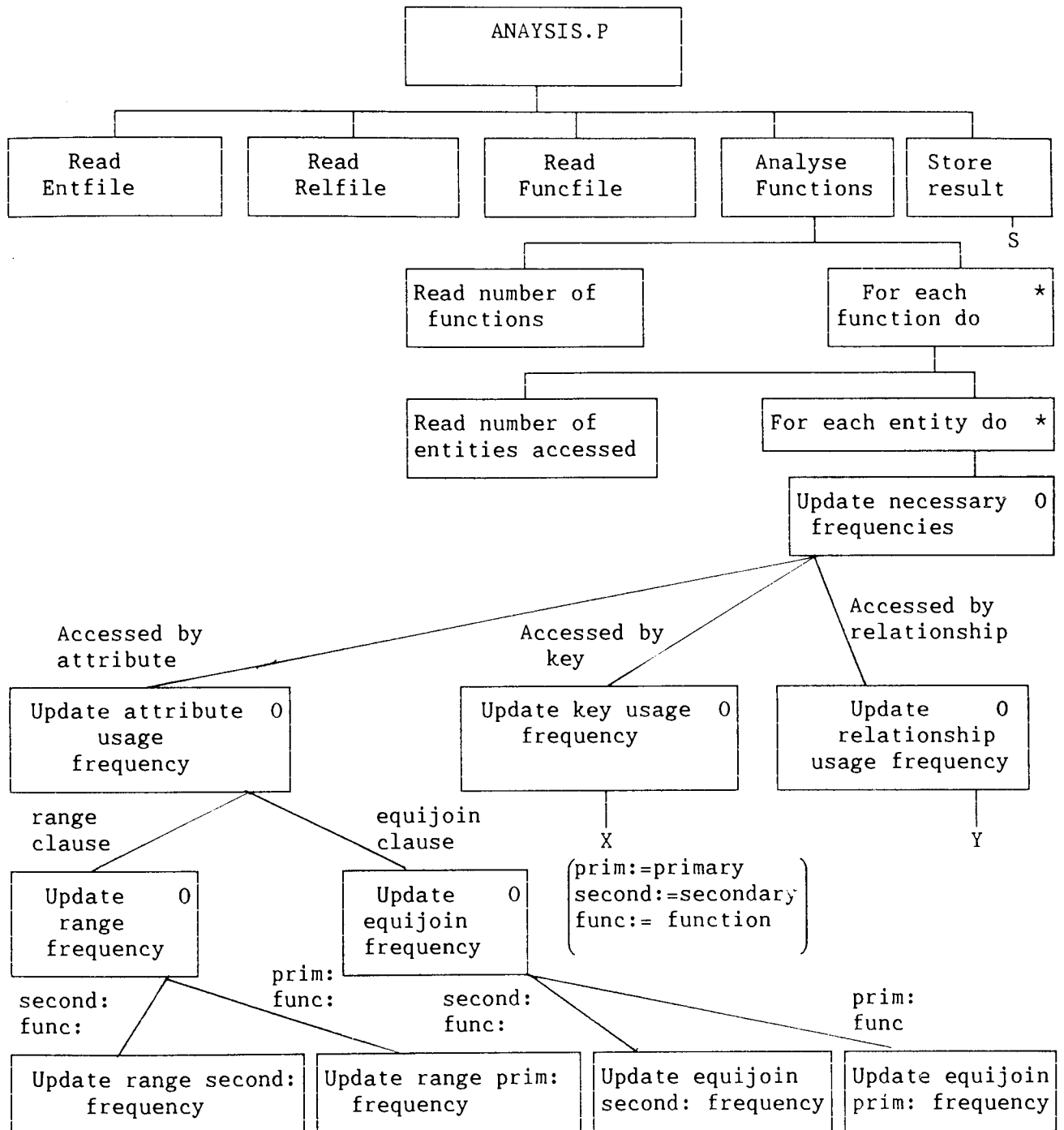
write(analysisfile, ' Rfrequency ');
write(analysisfile, ' Efrequency ');
writeln (analysisfile);
write (analysisfile, 'Primary-key ');
write (analysisfile, usagedet[1].primdet.rangefreq);
write (analysisfile, usagedet[1].primdet.equifreq);
write (analysisfile, ' ');
write (analysisfile, usagedet[1].seconddet.rangefreq);
write (analysisfile, usagedet[1].seconddet.equifreq);
writeln (analysisfile);
for j := (keycount + 1) to noatts do
  begin
    writestr (analysisfile, entatt[j]);
    write (analysisfile, usagedet[j].primdet.rangefreq);
    write (analysisfile, usagedet[j].primdet.equifreq);
    write(analysisfile, ' ');
    write (analysisfile, usagedet[j].seconddet.rangefreq);
    write (analysisfile, usagedet[j].seconddet.equifreq);
    writeln (analysisfile);
  end
end;

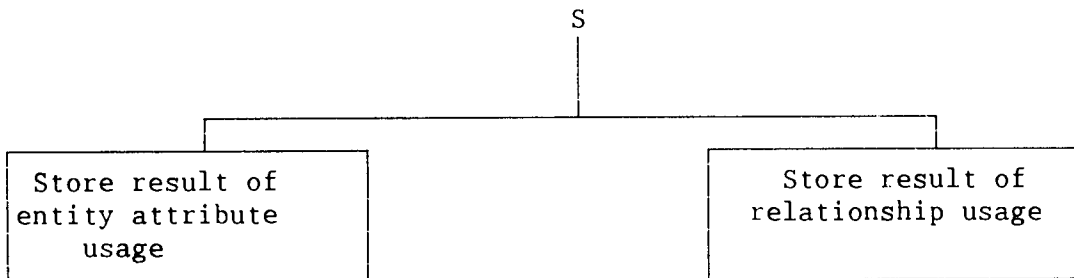
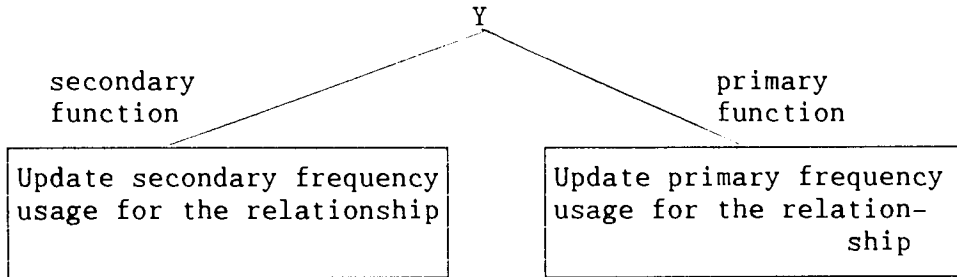
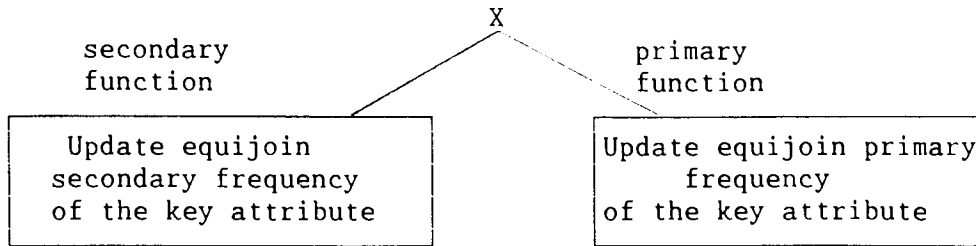
{WRITE THE RESULT OF THE RELATIONSHIP ANALYSIS}
rewrite (analysisfile, 'analreln');
writeln (analysisfile);
write (analysisfile, ' ');
write (analysisfile, ' Relationship Name ');
write (analysisfile, ' Primary Function Frequency ');
write (analysisfile, ' Secondary Function Frequency');
writeln (analysisfile);
for i := 1 to noofrelations do
  with relchart[i] do
    with rusagemat[i] do
      begin
        write (analysisfile, ' ');
        writestr (analysisfile, rname);
        write (analysisfile, ' ');
        write (analysisfile, primfreq);
        write (analysisfile, ' ');
        write (analysisfile, ' ');
        write (analysisfile, secfreq);
        writeln (analysisfile);
      end
    end
  end.

```



ANALYSIS.P





## RELMAP.P

{THIS PROGRAM MAPS THE ENTITIES AND RELATIONSHIPS INTO NORMALISED}  
{RELATIONAL STRUCTURE}

```
program relmap (input, output, relfile);
```

```
const maxstrlength = 20;
      maxnoel = 40;
      maxnoent = 40;
      maxnoatt = 20;
      maxrelatt = 20;
      maxnokey = 5;
      maxnoindent = 5;
      maxnoprops = 20;
      awith = 'WITH';
      awithout = 'WITHOUT';
```

```
type str = array [1..maxstrlength] of char;
      relstat = (weth, without);
```

```
      props = record
        entptr: integer;
        attptr: integer;
      end;
```

```
      relation = record
        rname : str;
        entitya : integer;
        degenta : char;
        membshpa : char;
        entityb : integer;
        degentb : char;
        membshpb : char;
        case rs : relstat of
          weth : (norelatt : integer;
                  relatt : array [1..maxrelatt] of str);
          without : ();
        end;
```

```
      attributes = array[0..maxnoatt] of str;
```

```
      entity = record
        ename : str;
        keycount : integer;
        noatts : integer;
        entatt : attributes;
      end;
```

{NORMENT DESCRIBES A RELATION THAT REPRESENTS AN ENTITY IN A RELATIONAL SCHEMA}

```
      norment = record
        nename : str;
        noofprops : integer;
        identcnt: integer;
        norentatts: array[1..maxnoprops] of props;
      end;
```

```
{NORMREL DESCRIBES A RELATION THAT REPRESENTS A RELATIONSHIP IN A RELATIONAL}
{SCHEMA}
```

```
normrel = record
    nrname : integer;
    identcnt : integer;
    norrelidnt : array [1..maxnoidnt] of props;
    nrkeycnt : integer;
    norrelkey : array [1..maxnokey] of props;
    case nrs : relstat of
        weth : (nrattcnt : integer;
                nrelatt : array [1..maxrelatt] of integer);
        without : ();
    end;

var relchart : array [1..maxnorel] of relation;
    entchart : array [1..maxnoent] of entity;
    norrelchtr : array [1..maxnorel] of normrel;
    norentchtr : array [1..maxnoent] of norment;
    noofrelations : integer;
    relfile : text;
    i, j, k, l : integer;
    norelfile : integer;
    noofentities : integer;
    entfile : text;
    mapfile : text;
    tempbuff : str;
    found : boolean;
    correct : boolean;
    answer : str;
    noofnormrel : integer;

procedure readstr (var f : text; var s : str);
    var ptr : integer;
    begin
        ptr := 0;
        while not eoln (f) and (ptr < maxstrlength ) do
            begin
                ptr := ptr + 1;
                read (f, s[ptr]);
            end;
        while ptr < maxstrlength do
            begin
                ptr := ptr + 1;
                s[ptr] := ' '
            end
        end;
    end;

procedure writestr (var f : text; var s : str);
    var i : integer;
    begin
        for i := 1 to maxstrlength do
            write (f, s[i]);
        end;
    end;
```

```

function equalstr (a, b : str): boolean;
  var ptr : integer;
      equal : boolean;
  begin
    equal := true;
    ptr := 0;
    while equal and (ptr < maxstrlength ) do
      begin
        ptr := ptr + 1;
        if a[ptr] <> b[ptr] then equal := false
        end;
      equalstr := equal
    end;

procedure findent (var int : integer);
  var l : integer;
  begin
    l := 0;
    found := false;
    while (not found) and (l < noofentities) do
      begin
        l := l + 1;
        if equalstr (tempbuff, entchart[l].ename) then
          begin
            int := l;
            found := true
          end
        end
      end;

begin
  {READ IN THE ENTITY FILE}
  reset (entfile, 'hospent');
  readln (entfile, noofentities);
  for i := 1 to noofentities do
    with entchart[i] do
      begin
        readln (entfile);
        readstr (entfile, ename);
        readln (entfile);
        readln (entfile, keycount);
        readln (entfile, noatts);
        readln (entfile);
        for l := 0 to keycount do readstr (entfile,entatt[l]);
        readln (entfile);
        readln (entfile);
        k := 1;
        for j := (keycount + 1) to noatts do
          begin
            readstr (entfile, entatt[j]);
            k := k + 1;
            if k > 4 then
              begin
                readln (entfile);
                k := 1
              end
            end;
          readln (entfile);
          readln (entfile)
        end;
      end;
end;

```

```
{READ IN THE RELATION FILE}
reset (relfile, 'hosprel');
readln (relfile, noofrelations);
for i := 1 to noofrelations do
with relchart[i] do
    begin
        readln (relfile);
        readstr (relfile, rname);
        readln (relfile);
        readstr (relfile, tempbuff);
        findent (entitya);
    readln (relfile, degenta);
        readln (relfile, membshpa);
        readstr (relfile, tempbuff);
        findent (entityb);
        readln (relfile, degentb);
        readln (relfile, membshpb);
        readstr (relfile, tempbuff);
        if tempbuff = awith then rs := weth
        else rs := without;
        readln (relfile);
        if rs = weth then
            begin
                readln (relfile, norelatt);
                k := 1;
                for j := 1 to norelatt do
                    begin
                        readstr (relfile, relatt[j]);
                        k := k + 1;
                        if k > 4 then
                            begin
                                readln (relfile);
                                k := 1;
                            end
                        end
                    end
                end
            end
        end;
end;
```

```
{FORM THE RELATIONAL SCHEMA}
```

```
j := 0;
for i := 1 to noofrelations do
with relchart[i] do
begin
```

```
{DO NOT SELECT THOSE RELATIONSHIPS WHICH ARE 1:1 or 1:n AND HAVE AN OBLIGATORY}
{MEMBERSHIP FOR ENTITY-B}
```

```
if (degenta <> '1' ) and (membshpb <> 'o' ) then
begin
j := j + 1;
with norrelchtr [j] do
begin
if (degenta = '1') and (degentb = '1') and (membshpb = 'n') then
begin
nrname := i;
identcnt := entchart[entitya].keycount;
for k := 1 to identcnt do
begin
norrelidnt[k].entptr := entitya;
norrelidnt[k].attptr := k
end;
nrkeycnt := entchart[entityb].keycount;
for k := 1 to nrkeycnt do
begin
norrelkey[k].entptr := entityb;
norrelkey[k].attptr := k
end
end
else
```

```
{IF THE RELATIONSHIP IS OF DEGREE 1:n AND THE MEMBERSHIP OF ENTITY-B}
{IS NOT OBLIGATORY THEN CREATE A RELATION}
```

```
if (degenta = '1') and (degentb = 'n') and (membshpb = 'n') then
begin
nrname := 1;
identcnt := entchart[entityb].keycount;
for k := 1 to identcnt do
begin
norrelidnt[k].entptr := entityb;
norrelidnt[k].attptr := k
end;
nrkeycnt := entchart[entitya].keycount;
for k := 1 to nrkeycnt do
begin
norrelkey[k].entptr := entitya;
norrelkey[k].attptr := k
end
end
else
```

```
{CREATE RELATIONS TO REPRESENT RELATIONSHIP OF DEGREE m:n}
```

```

begin
  nrname := i;
  identcnt := (entchart[entitya].keycount) +
              (entchart[entityb].keycount);
  for k := 1 to entchart[entitya].keycount do
    begin
      norrelidnt[k].entptr := entitya;
      norrelidnt[k].attptr := k
    end;
  l := 0;
  for k := ((entchart[entitya].keycount)+1) to identcnt do
    begin
      l := l + 1;
      norrelidnt[k].entptr := entityb;
      norrelidnt[k].attptr := l
    end;
  nrkeycnt := 0
end;
nrs := rs;
if nrs = weth then
  begin
    nrattcnt := norelatt;
    for k := 1 to nrattcnt do
      begin
        nrelatt[k] := k
      end
    end
  end
end
end;
noofnormrel := j;

{WRITE THE NORMALISED RELATIONS REPRESENTING THE RELATIONSHIPS IN THE MAPFILE}

rewrite (mapfile, 'norhosprel');
writeln (mapfile, noofnormrel);
for i := 1 to noofnormrel do
  with norrelchtr[i] do
    begin
      writestr (mapfile, relchart[nrname].rname);
      writeln (mapfile);
      writeln (mapfile, identcnt);
      k := 1;
      for j := 1 to identcnt do
        with norrelidnt[j] do
          begin
            writestr (mapfile, entchart[entptr].entatt[attptr]);
            k := k + 1;
            if k > 4 then
              begin
                writeln (mapfile);
                k := 1
              end
            end;
          end;
        end;
      end;
    end;
  end;
end;

```



```
for j := 1 to nrkeycnt do
with norrelkey[j] do
begin
writestr (mapfile, entchart[entptr].entatt[attptr]);
k := k + 1;
if k > 4 then
begin
writeln(mapfile);
k := 1
end;
end;
if nrs = weth then
begin
for j := 1 to nrattcnt do
with relchart[nrname] do
begin
writestr (mapfile, relatt[nrelatt[j]]);
k := k + 1;
if k > 4 then
begin
writeln (mapfile);
k := 1
end
end
end;
writeln (mapfile)
end;
```

```

{MAP ALL THE ENTITIES}

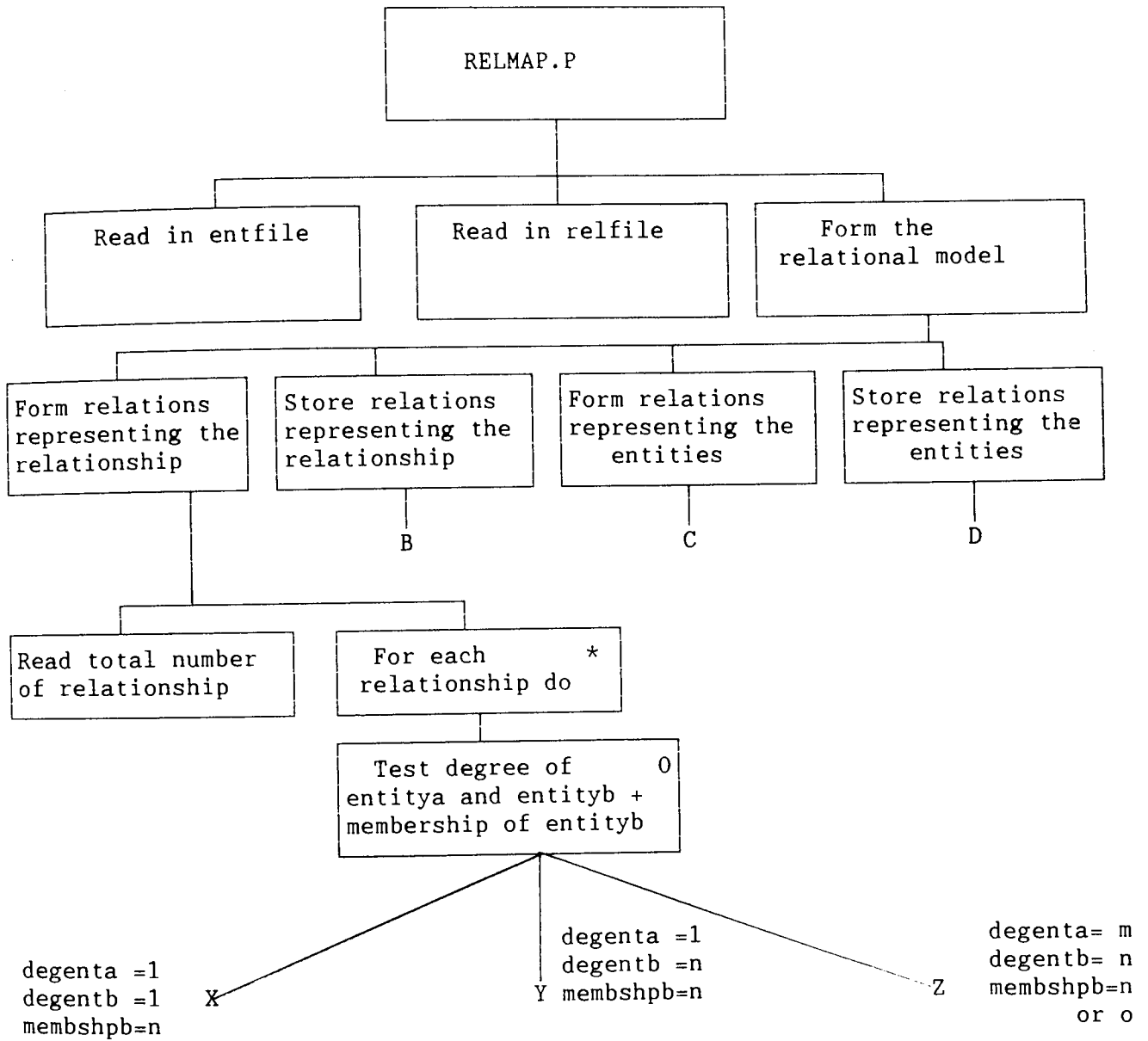
for i := 1 to noofentities do
  with norentchtr[i] do
    begin
      nename := entchart[i].ename;
      identcnt := entchart[i].keycount;
      noofprops := entchart[i].noatts;
      for j := 1 to entchart[i].noatts do
        begin
          norentatts[j].entptr := i;
          norentatts[j].attptr := j
        end;
      l := 0;
      while l < noofrelations do
        begin
          {IF THE ENTITY IS THE ENTITYB OF A RELATIONSHIP OF DEGREE 1:1 or 1:n}
          {AND THE MEMBERSHIP OF ENTITYB IS OBLIGATORY, THEN POST THE IDENTIFIER}
          {OF ENTITYA AS A FOREIGN KEY TO ENTITYB AND INCREASE THE NUMBER OF}
          {ATTRIBUTES ENTITYB}

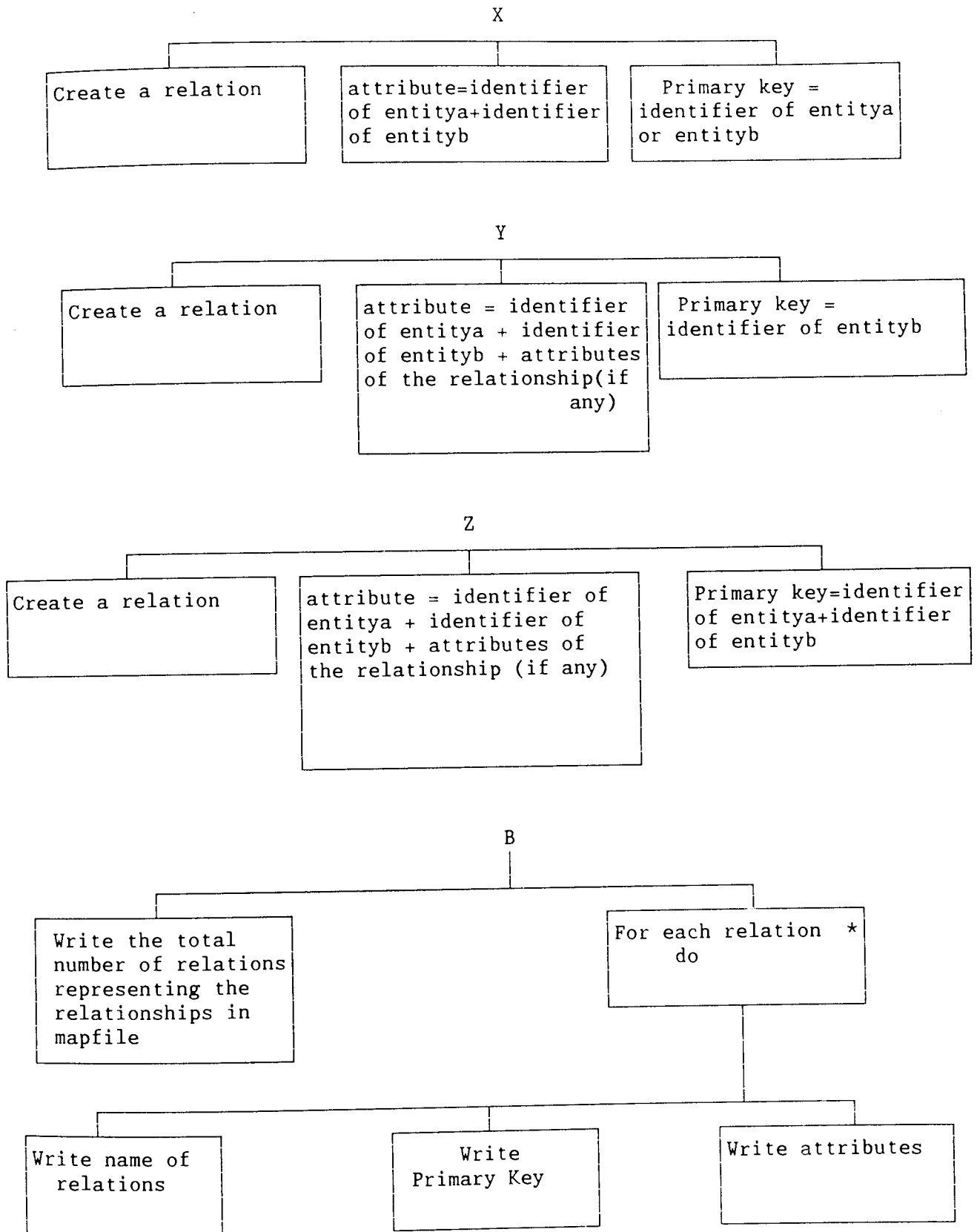
          l := l+1;
          if relchart[l].entityb = i then
            begin
              if (relchart[l].degenta = '1') and (relchart[l].membshpb = 'o') then
                begin
                  for k := 1 to entchart[relchart[l].entitya].keycount do
                    begin
                      noofprops := noofprops + 1;
                      norentatts[noofprops].entptr := relchart[l].entitya;
                      norentatts[noofprops].attptr := k
                    end
                  end
                end
            end
          end
        end;
      end;

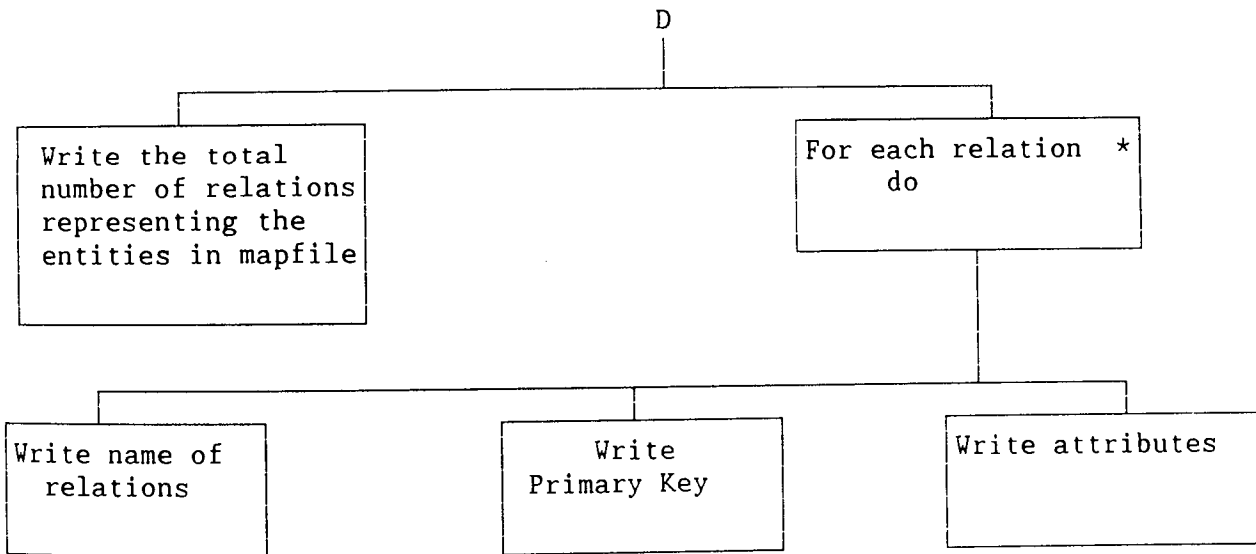
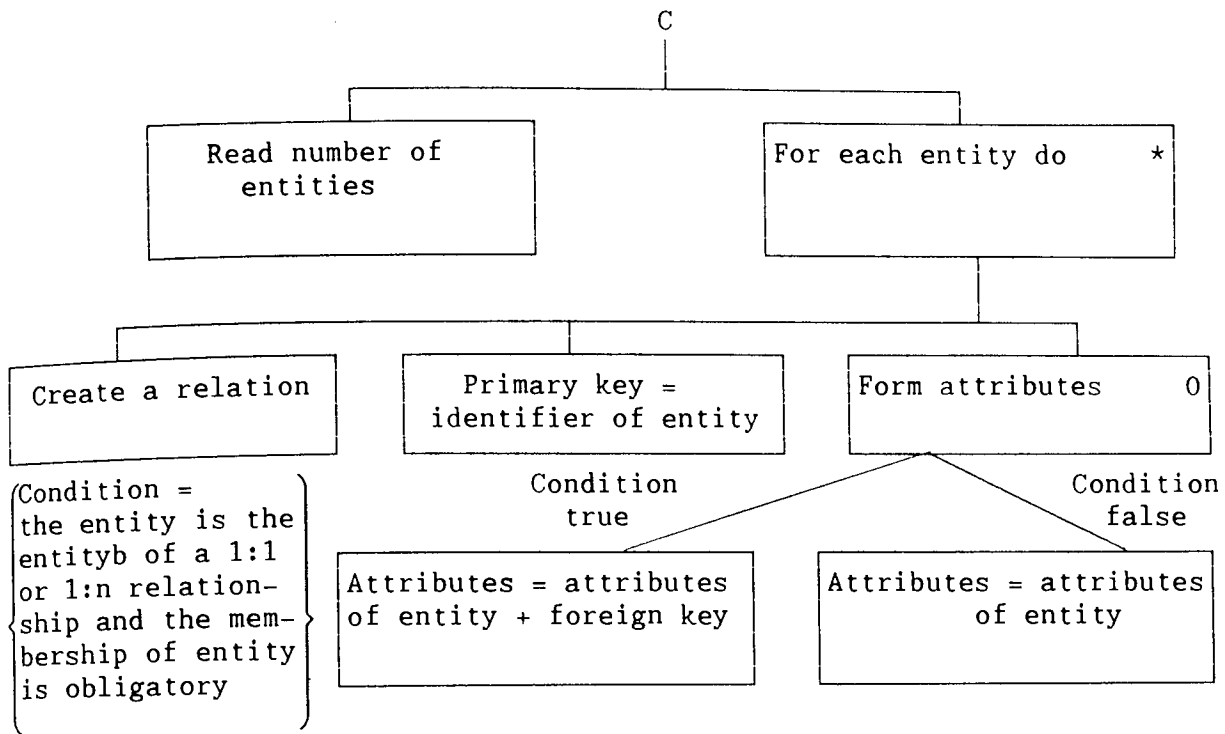
```

```
{WRITE TO THE MAPFILE}
rewrite (mapfile, 'hospmmap');
writeln (mapfile, noofentities);
for i := 1 to noofentities do
  with norentchtr[i] do
    begin
      writeln (mapfile);
      writestr (mapfile, nename);
      write (mapfile, noofprops);
      write (mapfile, ' ');
      writeln (mapfile, identcnt);
      for j := 1 to identcnt do
        with norentatts[j] do
          writestr (mapfile, entchart[entptr].entatt[attptr]);
          writeln(mapfile);
          k := 1;
          for j := (identcnt + 1) to noofprops do
            with norentatts[j] do
              begin
                writestr (mapfile, entchart[entptr].entatt[attptr]);
                k := k+1;
                if k > 4 then
                  begin
                    writeln (mapfile);
                    k := 1
                  end
                end;
              writeln (mapfile);
              writeln (mapfile)
            end
          end
        end
      end
    end
  end
end.
```

RELMAP.P







CODMAP.P

{THIS PROGRAM MAPS THE ENTITIES AND RELATIONSHIPS INTO A CODASYL STRUCTURE}

```

program codmap (input, output, relfile);

const maxstrlength = 20;
      maxnorel = 40;
      maxmodrel = 50;
      maxnoent = 40;
      maxnoatt = 20;
      maxrelatt = 10;
      maxnokey = 5;
      maxnoidnt = 5;
      awith = 'WITH';
      awithout = 'WITHOUT';
      primarykey = 'PRIMARY KEY          ';

type str = array [1..maxstrlength] of char;
      relstat = (weth, without);
      props = record
          entptr: integer;
          attptr: integer;
          end;

      relation = record
          rname: str;
          entitya: integer;
          degenta: char;
          membshpa: char;
          entityb: integer;
          degentb: char;
          membshpb: char;
          case rs : relstat of
              weth: (norelatt : integer;
                    relatt : array [1..maxrelatt] of str);
              without: ();
          end;

      attributes = array [0..maxnoatt] of str;
      entity = record
          ename: str;
          keycount: integer;
          noatts: integer;
          entatt : attributes;
          end;

      modreln = record
          mrtype : char; {REPRESENTS WHETHER MODIFIED OR ORIGINAL}
          mrname : str; {NAME OF THE RELATIONSHIP}
          mentitya : integer; {POINTER TO ENTITYA}
          mdegenta : char; {DEGREE OF ENTITYA}
          mmembshpa : char; {MEMBERSHIP OF ENTITYA}
          mentityb : integer; {POINTER TO ENTITYB}
          mdegentb : char; {DEGREE OF ENTITYB}
          mmembshpb : char; {MEMBERSHIP OF ENTITYB}
          end;

```

```
var relchart: array [1..maxnrel] of relation;
entchart : array [1..maxnoent ] of entity;
modrelchart : array [1..maxmodrel] of modreln;
noofrelations: integer;
relfile: text;
codent: text;
codrel: text;
i, j, k, l : integer;
norelfile: integer;
noofentities: integer;
noofmodrel : integer;
entfile : text;
mapfile : text;
modrelfile : text;
modent : text;
tempbuff : str;
found : boolean;
correct : boolean;
answer : str;
noofnormrel : integer;
tempstr : str;
```

```
procedure readstr (var f: text; var s: str);
  var ptr: integer;
  begin
    ptr := 0;
    while not eoln (f) and (ptr < maxstrlength) do
      begin
        ptr := ptr + 1;
        read (f, s[ptr]);
      end;
    while ptr < maxstrlength do
      begin
        ptr := ptr + 1;
        s[ptr] := ' ';
      end
    end;
  end;
```

```
procedure writestr (var f: text; var s: str);
  var i: integer;
  begin
    for i := 1 to maxstrlength do
      write (f, s[i]);
    end;
```



```

function equalstr(a, b: str): boolean;
  var ptr : integer;
      equal : boolean;
  begin
    equal := true;
    ptr := 0;
    while equal and (ptr < maxstrlength ) do
      begin
        ptr := ptr + 1;
        if a[ptr] <> b[ptr] then
          equal := false
        end;
      end;
    equalstr := equal
  end;

```

```

procedure findent (var int : integer);
  var l : integer;
  begin
    l := 0;
    found := false;
    while (not found) and (l < noofentities) do
      begin
        l := l+ 1;
        if equalstr (tempbuff, entchart[l].ename) then
          begin
            int := l;
            found := true
          end
        end
      end
    end;

```

```

procedure formstring (var r: str; var s: str; var t: str);

```

```

{FORMS THE NAME OF THE MODIFIED ENTITY}
  begin
    write (' The entity name is ');
    writestr (output, s);
    writeln;
    write (' The name of the entity formed from the relationship is ');
    writestr (output, t );
    writeln;
    write (' State the name of the relationship between these entities ');
    writeln;
    readstr (input, r);
    readln
  end;

```

```
{READ IN THE ENTITY FILE}
```

```
begin
reset (entfile, 'hospent');
readln (entfile, noofentities);
for i := 1 to noofentities do
  with entchart[i] do
    begin
      readln (entfile);
      readstr (entfile, ename);
      readln (entfile);
      readln (entfile, keycount);
      readln (entfile, noatts);
      readln (entfile);
      for l := 0 to keycount do readstr (entfile, entatt[l]);
      readln (entfile);
      readln (entfile);
      k := 1;
      for j := (keycount + 1) to noatts do
        begin
          readstr (entfile, entatt[j]);
          k := k+1;
          if k > 4 then
            begin
              readln (entfile);
              k := 1
            end
          end;
        readln (entfile);
        readln (entfile)
      end;
    end;
end;
```

```
{READ IN THE RELATION FILE}
```

```
reset (relfile, 'hosprel');
readln (relfile, noofrelations);
for i := 1 to noofrelations do
  with relchart[i] do
    begin
      readln (relfile);
      readstr (relfile, rname);
      readln (relfile);
      readstr (relfile, tempbuff);
      findent (entitya);
      readln (relfile, degenta);
      readln (relfile, membshpa);
      readstr (relfile, tempbuff);
      findent (entityb);
      readln (relfile, degentb);
      readln (relfile, membshpb);
      readstr (relfile, tempbuff);
      if tempbuff = awith then rs := weth
      else rs := without;
      readln (relfile);
      if rs = weth then
        begin
          readln (relfile, norelatt);
          k := 1;
```

```

        for j := 1 to norelatt do
            begin
                readstr(relfile, relatt[j]);
                k := k+1;
                if k > 4 then
                    begin
                        readln (relfile);
                        k := 1
                    end
                end
            end
        end;

{MODIFY THE RELATIONS INTO CODASYL COMPATIBLE FORM}

k := 0;
for j := 1 to noofrelations do
with relchart[j] do
    begin

{1:n RELATIONS WITHOUT ATTRIBUTES DO NOT NEED TO BE MODIFIED}

        if (degenta = '1') and (rs = without) then
            begin
                k := k + 1;
                with modrelchart[k] do
                    begin
                        mrtype := '0';
                        mrname := rname;
                        mentitya := entitya;
                        mdegenta := degenta;
                        mmembshpa := membshpa;
                        mentityb := entityb;
                        mdegentb := degentb;
                        mmembshpb := membshpb
                    end
                end
            end
        else

{1:n RELATIONS WITH ATTRIBUTES NEED TO BE MODIFIED}

            if (degenta = '1') and (rs = weth) then
                begin

{A NEW ENTITY IS CREATED TO REPLACE THE RELATIONSHIP}

                    noofentities := noofentities + 1;
                    with entchart[noofentities] do
                        begin
                            ename := rname;
                            keycount := entchart[entityb].keycount;
                            noatts := keycount + norelatt;
                            entatt[0] := primarykey;
                            for l := 1 to keycount do
                                begin
                                    entatt[l] := entchart[entityb].entatt[l]
                                end;
                            end;
                        end
                    end
                end
            end
        end
    end
end;

```

```

    i := 0;
    for l:= (keycount+1) to noatts do
        begin
            i := i + 1;
            entatt[l] := relatt[i]
        end
    end;

```

{TWO ADDITIONAL RELATIONSHIPS NEED TO BE CREATED}

```

    k :=k + 1;
    formstring (tempstr, entchart[entitya].ename, rname);
    with modrelchart[k] do
        begin
            mrtype := 'C';
            mrname := tempstr;
            mentitya:= entitya;
            mdegenta :='1';
            mmembshpa := membshpa;
            mentityb:= noofentities;
            mdegentb := degentb;
            mmembshpb := 'o'
        end;
    k := k + 1;
    formstring (tempstr, entchart[entityb].ename, rname);
    with modrelchart[k]do
        begin
            mrtype := 'C';
            mrname :=tempstr;
            mentitya := entityb;
            mdegenta := '1';
            mmembshpa := membshpb;
            mentityb := noofentities;
            mdegentb := '1';
            mmembshpb := 'o'
        end
    end

```

{IF THE DEGREE OF THE RELATIONSHIP IS m:n THEN A NEW ENTITY}  
 {AND TWO ADDITIONAL RELATIONSHIPS ARE CREATED}

```

else
    begin
        noofentities := noofentities + 1;
        with entchart[noofentities] do
            begin
                ename := rname;
                keycount := (entchart[entitya].keycount) +
                    (entchart[entityb].keycount);
                noatts := keycount;
                entatt[0] := primaryKey;
                for l := 1 to entchart[entitya].keycount do
                    entatt[l] := entchart[entitya].entatt[l];
                i := 0;
                for l := (entchart[entitya].keycount + 1) to keycount do
                    begin
                        i := i + 1;
                        entatt[l] := entchart[entityb].entatt[i];
                    end;
            end;
        end;
    end;

```

```

    if rs = weth then
        begin
            noatts := noatts + norelatt;
            i := 0;
            for l := (keycount + 1) to noatts do
                begin
                    i := i+1;
                    entatt[l] := relatt[i]
                end
            end;
        k := k + 1;
        formstring (tempstr, entchart[entitya].ename, rname);
        with modrelchart[k] do
            begin
                mrtype := 'C';
                mrname := tempstr;
                mentitya := entitya;
                mdegenta := '1';
                mmembshpa := membshpa;
                mentityb := noofentities;
                mdegentb := 'n';
                mmembshpb := 'o'
            end;
        k := k + 1;
        formstring(tempstr, entchart[entityb].ename, rname);
        with modrelchart [k] do
            begin
                mrtype := 'C';
                mrname := tempstr;
                mentitya := entityb;
                mdegenta := '1';
                mmembshpa := membshpb;
                mentityb := noofentities;
                mdegentb := 'n';
                mmembshpb := 'o'
            end
        end
    end
end;
noofmodrel := k;

{WRITE THE ORIGINAL AND NEWLY CREATED ENTITIES BACK INTO THE MODENTFILE.}

rewrite (modent, 'modhospent');
writeln (modent, noofentities);
for i := 1 to noofentities do
with entchart[i] do
    begin
        writeln (modent);
        writestr (modent, ename);
        writeln (modent);
        writeln (modent, keycount);
        writeln (modent, noatts);
        writeln (modent);
    end
end;

```

```

k := 1;
for l := 0 to keycount do
  begin
    writestr (modent, entatt[l]);
    k := k + 1;
    if k > 4 then
      begin
        writeln (modent);
        k := 1
      end
    end;
  writeln (modent);
k := 1;
for j := (keycount + 1) to noatts do
  begin
    writestr (modent, entatt[j]);
    k := k + 1;
    if k > 4 then
      begin
        writeln (modent);
        k := 1
      end
    end;
  writeln (modent);
  writeln (modent)
end;

```

{WRITE THE ORIGINAL AND MODIFIED RELATIONSHIPS INTO THE MODRELFIL}

```

rewrite (modrelfile, 'modhosprel');
writeln (modrelfile, noofmodrel);
for i := 1 to noofmodrel do
with modrelchart[i] do
  begin
    writestr (modrelfile, mrname);
    writeln (modrelfile);
    writestr (modrelfile, entchart[mentitya].ename);
    write (modrelfile, mdegenta);
    write (modrelfile, ' ');
    write (modrelfile, mmembshpa);
    writeln (modrelfile);
    writestr (modrelfile, entchart[mentityb].ename );
    write (modrelfile, mdegentb);
    write (modrelfile, ' ');
    write (modrelfile, mmembshpb);
    writeln (modrelfile);
    writeln (modrelfile)
  end;

```

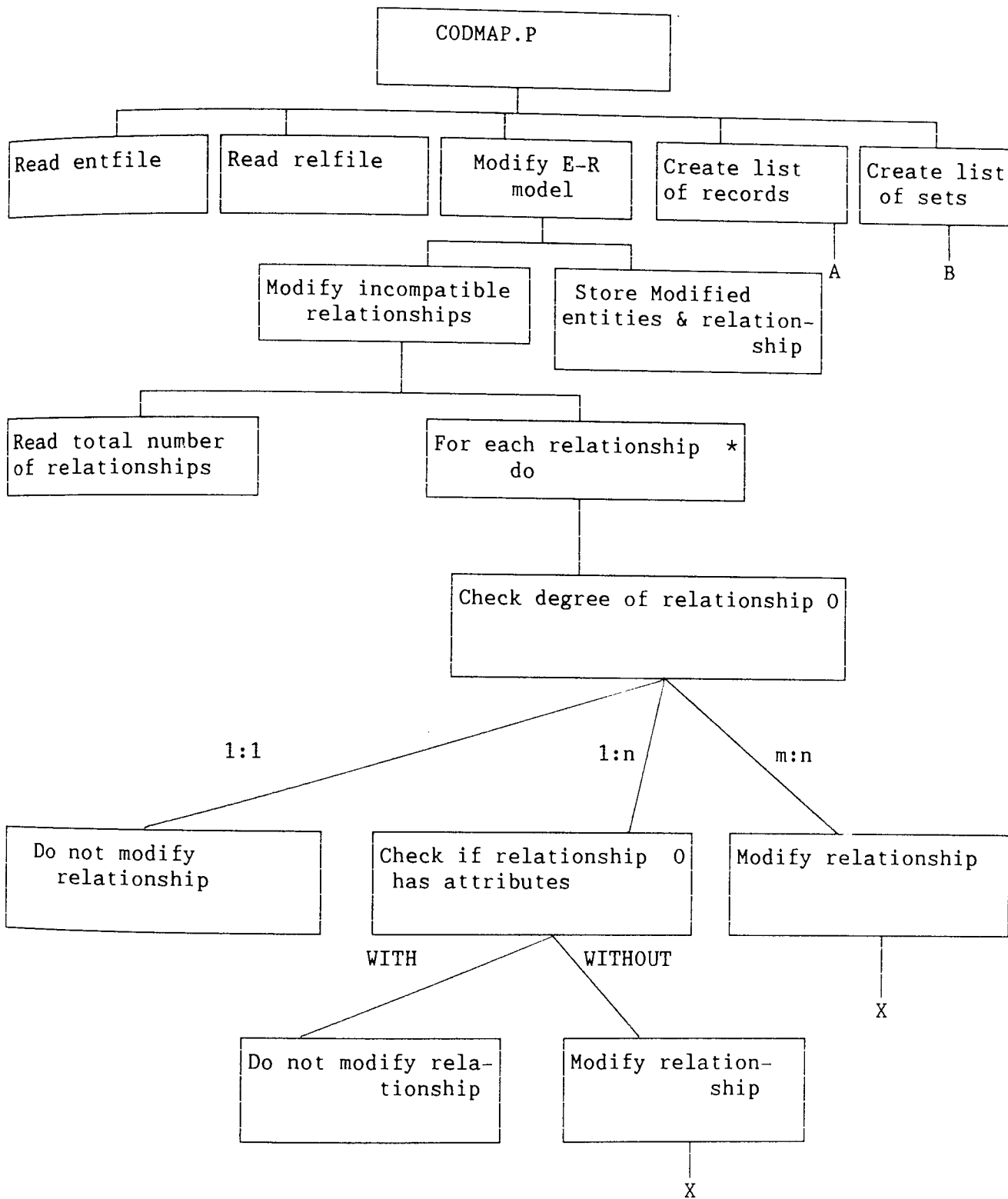
```
{FORM A RECORD CORRESPONDING TO EACH ENTITY}
{WRITE THE RECORDS IN THE RECORD FILE}
```

```
rewrite (codent, 'hosprec');
for i := 1 to noofentities do
with entchart[i] do
begin
write (codent, 'Record name is ');
writestr(codent, ename);
writeln (codent);
writestr(codent, ename);
write(codent, 'KEY is ');
for j := 1 to keycount do
begin
writestr(codent, entatt[j]);
if (j < keycount) then write (codent, ',')
end;
writeln (codent);
for j := (keycount + 1) to noatts do
begin
write(codent, ' ');
writestr(codent, entatt[j]);
writein (codent, ';')
end;
writeln (codent);
writeln (codent)
end;
end;
```

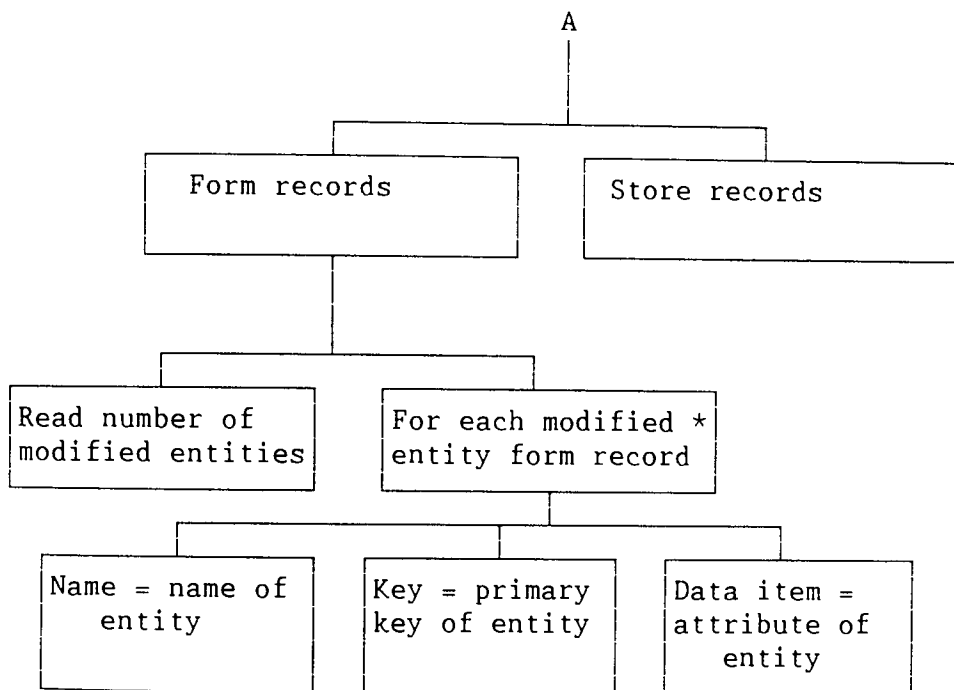
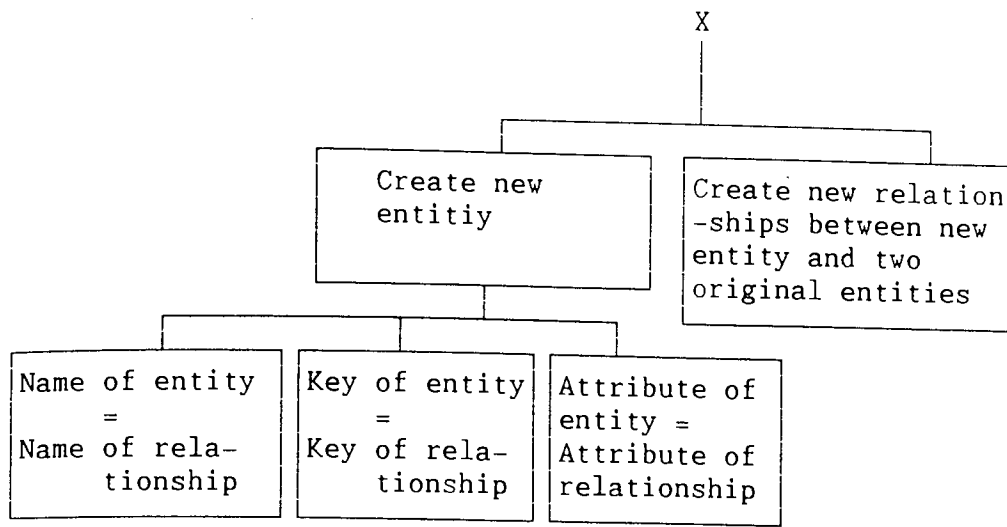
```
{CORRESPONDING TO EACH RELATIONSHIP IN MODIFIED RELATIONSHIP FILE}
{FORM A SET AND WRITE THE SETS IN THE SET FILE}
```

```
rewrite(codrel, 'hospset');
for i := 1 to noofmodrel do
with modrelchart[i] do
begin
write (codrel, 'Set name is ');
writestr(codrel, mrname);
writeln(codrel);
write(codrel, 'Owner record is ');
writestr(codrel, entchart[mentitya].ename);
writeln(codrel);
writeln(codrel, 'Member record is ');
writestr(codrel, entchart[mentityb].ename);
if (mmembshpb = 'o') and (mrtype = '0') then
write(codrel, ' AUTOMATIC MANDATORY')
else
if (mmembshpb = 's') and (mrtype = '0') then
write (codrel, 'AUTOMATIC FIXED')
else
if (mmembshpb = 'o') and (mrtype = 'C') then
write(codrel, 'AUTOMATIC FIXED')
else
write(codrel, 'MANUAL OPTIONAL');
writeln(codrel);
writeln(codrel)
end;
end.
```

CODMAP.P









LABEL.P

{THIS PROGRAM DETERMINES WAYS OF ORGANISING DATA}

program edgelabel (input, output, funcfile);

```

const maxstrlength = 20;
    maxnofunc = 20;
    maxnoacc = 15;
    maxselcrit = 4;
    maxnoatt = 15;
    maxnoent = 40;
    maxnorel = 40;
    maxoutedge = 40;
    maxinedge = 40;
    maxnoedge = 400;
    primary = 'PRIMARY           ';
    secondary = 'SECONDARY       ';
    ent = 'ENTITY                 ';
    reln = 'RELATIONSHIP         ';
    selbyrel = 'SELECT.BY.RELATION ';
    selbyatt = 'SELECT.BY.ATTRIBUTES';
    selbykey = 'SELECT.BY.KEY     ';
    awith = 'WITH                 ';
    awithout = 'WITHOUT          ';
    maxrelatt = 10;
    range = 'RANGE               ';
    equijoin = 'EQUIJOIN         ';

type str = array [1..maxstrlength] of char;
relstat = (weth, without);
selopts = (sbyr, sbya, sbyp);
selectdet = record
    case opts : selopts of
        sbyr : (relpnt : integer);
        sbyp : (ppnt : char);
        sbya : (apnt : integer;
               aclaus : char);
    end;

accent = record
    entname : integer;
    eselectcrit : selectdet
end;

entacc = array [1..maxnoacc] of accent;

func = record
    funcname : str;
    funcfreq : integer;
    funcstatus : integer;
    noacc : integer;
    entarr : entacc
end;

attributes = array [0..maxnoatt] of str;

```

```

entity = record
    ename: str;
    keycount: integer;
    noatts : integer;
    entatt : attributes
end;

relation = record
    rname: str;
    entitya: integer;
    degenta: char;
    membshpa : char;
    entityb: integer;
    degentb: char;
    membshpb: char;
    case rs: relstat of
        weth: (norelatt : integer;
              relatt : array [1..maxrelatt] of str);
        without : ()
    end;
end;

accdet = record
    rangefreq: integer;
    equipfreq : integer
end;

analdet = record
    primdet: accdet;
    seconddet: accdet
end;

eusagedet= record
    usagedet : array[1..maxnoatt] of analdet
end;

rusagedet = record
    primfreq : integer;
    secfreq : integer
end;

outdet = record
    oedgetype : char; {EDGE TYPE}
    oedgepnt : integer; {POINTER TO KEY/ATTRIBUTE/RELATIONSHIP}
    oedgefreq : integer; {USAGE FREQUENCY}
    oedgelabel: char {DATA ORGANISATION LABEL}
end;

indet = record
    iedgepnt : integer; {POINTER TO THE RELATIONSHIP}
    iedgefreq : integer; {USAGE FREQUENCY}
    iedgelabel : char {DATA ORGANISATION LABEL}
end;

```

```

    edgedet = record
        edgelab : char; {LABEL ASSIGNED TO THE EDGE}
        edgefreq : integer; {FREQUENCY OF USAGE}
        edgetype : char; {TYPE OF THE EDGE}
        edgepnt1 : integer;
        edgepnt2 : integer;
        edgepnt3 : integer;
        edgepnt4 : integer
    end;

    graphdet = record
        noofoutedge : integer; {NUMBER OF OUTEDGES}
        noofinedge : integer; {NUMBER OF INEDGES}
        outedge : array[1..maxoutedge] of outdet; {OUTEDGE DETAILS}
        inedge : array[1..maxinedge] of indet {INEDGE DETAILS}
    end;

    sortedge = record
        sortfreq : integer;
        sortpnt : integer
    end;

var funcchart: array [1..maxnofunc] of func;
    entchart : array [1..maxnoent] of entity;
    relchart : array [1..maxnorel] of relation;
    eusagemat : array [1..maxnoent] of eusagedet;
    rusagemat : array [1..maxnorel] of rusagedet;
    sortlist : array [1..maxnoedge] of sortedge;
    labelmat : array [1..maxnoent] of graphdet; {ENTITY EDGE DESCRIPTION}
    edgemat : array [1..maxnoedge] of edgedet; {EDGE DESCRIPTION}
    noofedges : integer;
    labelc : integer;
    ilabelw : integer;
    olabelw : integer;
    noofswap : integer;
    tempfreq : integer;
    tempnt : integer;
    nooffunc : integer;
    funcfile : text;
    entfile : text;
    relfile : text;
    analysisfile : text;
    labelfile : text;
    edgefile : text;
    i, j, k, l : integer;
    noofnewfunc : integer;
    noofentities : integer;
    noofrelations : integer;
    entindex : integer;
    tempbuff : str;
    found : boolean;

```

```
procedure readstr (var f: text; var s: str);
  var ptr: integer;
  begin
    ptr := 0;
    while not eoln (f) and (ptr < maxstrlength) do
      begin
        ptr := ptr + 1;
        read (f, s[ptr])
      end;
    while ptr < maxstrlength do
      begin
        ptr := ptr + 1;
        s[ptr] := ' '
      end
    end;
end;

procedure writestr (var f: text; var s: str);
  var i: integer;
  begin
    for i := 1 to maxstrlength do
      write (f, s[i]);
    end;
end;

function equalstr (a, b: str ) : boolean;
  var ptr : integer;
      equal : boolean;
  begin
    equal := true;
    ptr := 0;
    while equal and (ptr < maxstrlength ) do
      begin
        ptr := ptr + 1;
        if a[ptr] <> b[ptr] then
          equal := false;
        end;
      end;
    equalstr := equal
  end;

procedure findent (var int : integer);
  var l : integer;
  begin
    l := 0;
    found := false;
    while (not found) and (l < noofentities ) do
      begin
        l := l+1;
        if equalstr (tempbuff, entchart[l].ename) then
          begin
            int := l;
            found := true
          end
        end
      end
    end;
end;
```

```
procedure findrel (var int : integer);
  var l : integer;
  begin
  l := 0;
  found := false;
  while (not found) and (l < noofentities) do
    begin
    l := l+1;
    if equalstr (tempbuff, relchart[l].rname) then
      begin
      int := l;
      found := true
      end
    end
  end;

procedure findatt (var int : integer; var entindex : integer);
  var l : integer;
  begin
  l := 0;
  found := false;
  while (not found) and (l < entchart[entindex].noatts) do
    begin
    l := l+1;
    if equalstr (tempbuff, entchart[entindex].entatt[l]) then
      begin
      int := l;
      found := true
      end
    end
  end;

procedure findrelatt (var int: integer; var relindex : integer);
  var l : integer;
  begin
  l := 0;
  found := false;
  while (not found) and (l < relchart[relindex].norelatt ) do
    begin
    l := l + 1;
    if equalstr (tempbuff, relchart[relindex].relatt[l]) then
      begin
      int := l;
      found := true
      end
    end
  end;
end;
```

```
begin
```

```
{READ IN THE ENTITY FILE}
```

```
reset (entfile, 'hospent');
readln (entfile, noofentities);
for i := 1 to noofentities do
  with entchart [i] do
    begin
      readln (entfile);
      readstr (entfile, ename);
      readln (entfile);
      readln (entfile, keycount);
      readln (entfile, noatts);
      readln (entfile);
      for l := 0 to keycount do readstr (entfile, entatt[l]);
      readln (entfile);
      readln (entfile);
      k := 1;
      for j := (keycount + 1) to noatts do
        begin
          readstr (entfile, entatt[j]);
          k := k+1;
          if k > 4 then
            begin
              readln (entfile);
              k := 1
            end
          end;
        readln (entfile);
        readln (entfile)
      end;
    end;
```

```
{READ IN THE RELATION FILE}
```

```
reset (relfile, 'hosprel');
readln (relfile, noofrelations );
for i := 1 to noofrelations do
  with relchart[i] do
    begin
      readln (relfile);
      readstr (relfile, rname);
      readln (relfile);
      readstr (relfile, tempbuff);
      findent (entitya);
      readln (relfile, degenta);
      readln (relfile, membshpa);
      readstr (relfile, tempbuff);
      findent (entityb);
      readln (relfile, degentb);
      readln (relfile, membshpb);
      readstr (relfile, tempbuff);
      if tempbuff = awith then rs := weth
      else rs := without;
      readln (relfile);
      if rs = weth then
        begin
          readln (relfile, norelatt);
          k := 1;
          for j := 1 to norelatt do
```



```

begin
  readstr (relfile, relatt[j]):
  k := k + 1;
  if k > 4 then
    begin
      readln (relfile);
      k := 1
    end
  end
end
end;

{READ IN THE FUNCTION FILE}
reset (funcfile, 'hosppfunc');
readln (funcfile, nooffunc);
for i := 1 to nooffunc do
  with funcchart[i] do
    begin
      readln (funcfile);
      readstr (funcfile, funcname);
      readstr (funcfile, tempbuff);
      if equalstr(tempbuff, primary) then
        funcstatus := 1
      else funcstatus := 2;
      readln (funcfile, funcfreq);
      readln (funcfile, noacc);
      for j := 1 to noacc do
        with entarr[j] do
          begin
            readstr (funcfile, tempbuff);
            findent (entname);
            readstr (funcfile, tempbuff);
            readln (funcfile);
            if equalstr (tempbuff, selbyrel) then
              eselectcrit.opts := sbyr
            else
              if equalstr (tempbuff, selbyatt) then
                eselectcrit.opts := sbya
              else
                eselectcrit.opts := sbyp;
            if eselectcrit.opts = sbyr then
              begin
                readstr (funcfile, tempbuff);
                readln (funcfile);
                findrel (eselectcrit.relpnt)
              end
            else
              if eselectcrit.opts = sbya then
                begin
                  readstr (funcfile, tempbuff);
                  findatt (eselectcrit.apnt, entname);
                  readstr (funcfile, tempbuff);
                  if equalstr (tempbuff, range) then
                    eselectcrit.aclus := 'r'
                  else
                    eselectcrit.aclus := 'e';
                  readln (funcfile)
                end
              end
            end
          end
        end
      end
    end
  end
end

```

```

        else
        begin
        for k := 1 to entchart[entname].keycount do
        readstr (funcfile, tempbuff);
        readln (funcfile)
        end;
        readln (funcfile)
        end
        end;

{ANALYSE THE FUNCTIONS}
for i := 1 to nooffunc do
with funcchart [i] do
begin
  for j := 1 to noacc do
  with entarr[j] do
  begin
  if eselectcrit.opts = sbya then
  begin
    if eselectcrit.aclaus = 'r' then
    begin
      if funcstatus = 1 then
      eusagemat[entname].usagedet[eselectcrit.apnt].primdet.rangefreq:=
      eusagemat[entname].usagedet[eselectcrit.apnt].primdet.rangefreq +
      funcfreq

      else
      eusagemat[entname].usagedet[eselectcrit.apnt].seconddet.rangefreq:=
      eusagemat[entname].usagedet[eselectcrit.apnt].seconddet.rangefreq +
      funcfreq

      end
      else
      begin
      if funcstatus = 1 then
      eusagemat[entname].usagedet[eselectcrit.apnt].primdet.equifreq:=
      eusagemat[entname].usagedet[eselectcrit.apnt].primdet.equifreq +
      funcfreq

      else
      eusagemat[entname].usagedet[eselectcrit.apnt].seconddet.equifreq :=
      eusagemat[entname].usagedet[eselectcrit.apnt].seconddet.equifreq +
      funcfreq

      end
    end
  end
end
end

else
if eselectcrit.opts = sbyp then
begin
  if funcstatus = 1 then
  begin
    for k := 1 to entchart[entname].keycount do
    begin
      eusagemat[entname].usagedet[k].primdet.equifreq :=
      eusagemat[entname].usagedet[k].primdet.equifreq + funcfreq
    end
  end
end
end

```

```

    else
      begin
        for k := 1 to entchart[entname].keycount do
          begin
            eusagemat[entname].usagedet[k].seconddet.equifreq :=
            eusagemat[entname].usagedet[k].seconddet.equifreq + funcfreq
          end
        end
      end
    else
      begin
        if funcstatus = 1 then
          rusagemat[eselectcrit.relptnt].primfreq :=
          rusagemat[eselectcrit.relptnt].primfreq + funcfreq
        else
          rusagemat[eselectcrit.relptnt].secfreq :=
          rusagemat[eselectcrit.relptnt].secfreq + funcfreq
        end
      end
    end
  end;

noofedges := 0;
for i := 1 to noofentities do
  with eusagemat[i] do
  with entchart[i] do
  with labelmat[i] do
    begin
      noofoutedge := 1;
      noofinedge := 0;

      {THE FIRST OUTEDGE OF THE ENTITY IS THE PRIMARY KEY EDGE}
      {FILL IN THE NECESSARY DETAILS FOR THE FIRST EDGE}

      outedge[noofoutedge].oedgetype := 'p';
      outedge[noofoutedge].oedgepnt := 1;
      outedge[noofoutedge].oedgefreq := usagedet[1].primdet.rangefreq +
      usagedet[1].primdet.equifreq;
      outedge[noofoutedge].oedgelabel := 'I';

      {FILL IN THE DETAILS FOR THE REST OF THE EDGES}

      noofedges := noofedges + 1;
      with edgemat[noofedges] do
        begin
          edgelab := 'I';
          edgefreq := outedge[noofoutedge].oedgefreq;
          edgetype := 'p';
          edgepnt1 := i;
          edgepnt2 := 1;
          edgepnt3 := noofoutedge;
        end;
      for j := (keycount + 1) to noatts do
        begin
          noofoutedge := noofoutedge + 1;
          outedge[noofoutedge].oedgetype := 'a';
          outedge[noofoutedge].oedgefreq := usagedet[j].primdet.rangefreq +
          usagedet[j].primdet.equifreq;

          outedge[noofoutedge].oedgelabel := 'I';
          noofedges := noofedges + 1;
        end;
      end;
    end;
  end;
end;

```

```

    with edgemat[noofedges] do
      begin
        edgelab := 'I';
        edgefreq := outedge[noofoutedge].oedgefreq;
        edgetype := 'a';
        edgepnt1 := i;
        edgepnt2 := j;
        edgepnt3 := noofoutedge
      end
    end
  end;

for k := 1 to noofrelations do
with relchart[k] do

{RELATIONSHIP EDGES ARE OUTEDGES FOR ENTITYA}
{RELATIONSHIP EDGES ARE INEDGES FOR ENTITYB}

begin
noofedges := noofedges + 1;
with labelmat[entitya] do
begin
noofoutedge := noofoutedge + 1;
outedge[noofoutedge].oedgetype := 'r';
outedge[noofoutedge].oedgepnt := k;
outedge[noofoutedge].oedgefreq := rusagemat[k].primfreq;
outedge[noofoutedge].oedgelabel := 'I';
edgemat[noofedges].edgepnt3 := noofoutedge
end;
with labelmat[entityb] do
begin
noofinedge := noofinedge + 1;
inedge[noofinedge].iedgepnt := k;
inedge[noofinedge].iedgefreq := rusagemat[k].primfreq;
inedge[noofinedge].iedgelabel := 'I';
edgemat[noofedges].edgepnt4 := noofinedge
end;
edgemat[noofedges].edgepnt1 := entitya;
edgemat[noofedges].edgepnt2 := entityb;
edgemat[noofedges].edgetype := 'r';
edgemat[noofedges].edgelab := 'I';
edgemat[noofedges].edgefreq := rusagemat[i].primfreq
end;

{FORM THE LIST FOR SORTING}
for i := 1 to noofedges do
begin
sortlist[i].sortfreq := edgemat[i].edgefreq;
sortlist[i].sortpnt := i;
end;

```

```

{SORT THE LIST INTO FREQUENCY ORDER}
noofswap :=1;
repeat
noofswap := 0;
for i := 1 to noofedges do
  begin
    if sortlist[i+1].sortfreq > sortlist[i].sortfreq then
      begin
        tempfreq := sortlist[i].sortfreq;
        tempnt := sortlist[i].sortpnt;
        sortlist[i].sortfreq := sortlist[i+1].sortfreq;
        sortlist[i].sortpnt := sortlist[i+1].sortpnt;
        sortlist[i+1].sortfreq := tempfreq;
        sortlist[i+1].sortpnt := tempnt;
        noofswap := noofswap + 1
      end
    end
  until noofswap = 0;

{TRAVERSE IN FREQUENCY ORDER AND ASSIGN LABEL C WHERE FEASIBLE}

for i := 1 to noofedges do
with sortlist [i] do
  begin
    labelc := 0;
    with edgemat[sortpnt] do
      begin
        for j := 1 to labelmat[edgepnt1].noofoutedge do
with labelmat[edgepnt1] do
  begin
    if outedge[j].oedgelabel = 'C' then
      labelc := labelc + 1
    end;
    if labelc = 0 then
      begin
        edgelab := 'C';
        labelmat[edgepnt1].outedge[edgepnt3].oedgelabel := 'C';
        if edgetype = 'r' then
          labelmat[edgepnt2].inedge[edgepnt4].iedgelabel := 'C'
        end
      end
    end;
  end;
end;

```

```

{TRAVERSE IN FREQUENCY ORDER AND ASSIGN LABEL W WHERE FEASIBLE}
for i := 1 to noofedges do
with sortlist[i] do
  begin
  labelc := 0;
  olabelw := 0;
  ilabelw := 0;
  with edgemat[sortpnt] do
    begin
    if edgelab = 'C' then
      begin
      for j := 1 to labelmat[edgepnt1].noofoutedge do
with labelmat[edgepnt1] do
        begin
        if outedge[j].oedgelabel = 'C' then
          begin
          if edgepnt3 <> j then
            labelc := labelc + 1
          end;
          if outedge[j].oedgelabel = 'W' then
            begin
            if edgepnt3 <> j then
              olabelw := olabelw + 1
            end
          end;
        end;
      for j := 1 to labelmat[edgepnt1].noofinedge do
with labelmat[edgepnt1] do
        begin
        if inedge[j].iedgelabel = 'W' then
          begin
          ilabelw := ilabelw + 1
          end
        end;
      end;
    if (labelc = 0) and (olabelw = 0) and (ilabelw = 0) then
      begin
      edgelab := 'W';
      labelmat[edgepnt1].outedge[edgepnt3].oedgelabel := 'C';
      if edgetype = 'r' then
        labelmat[edgepnt2].inedge[edgepnt4].iedgelabel := 'C'
      end
    end
  end
end
end;

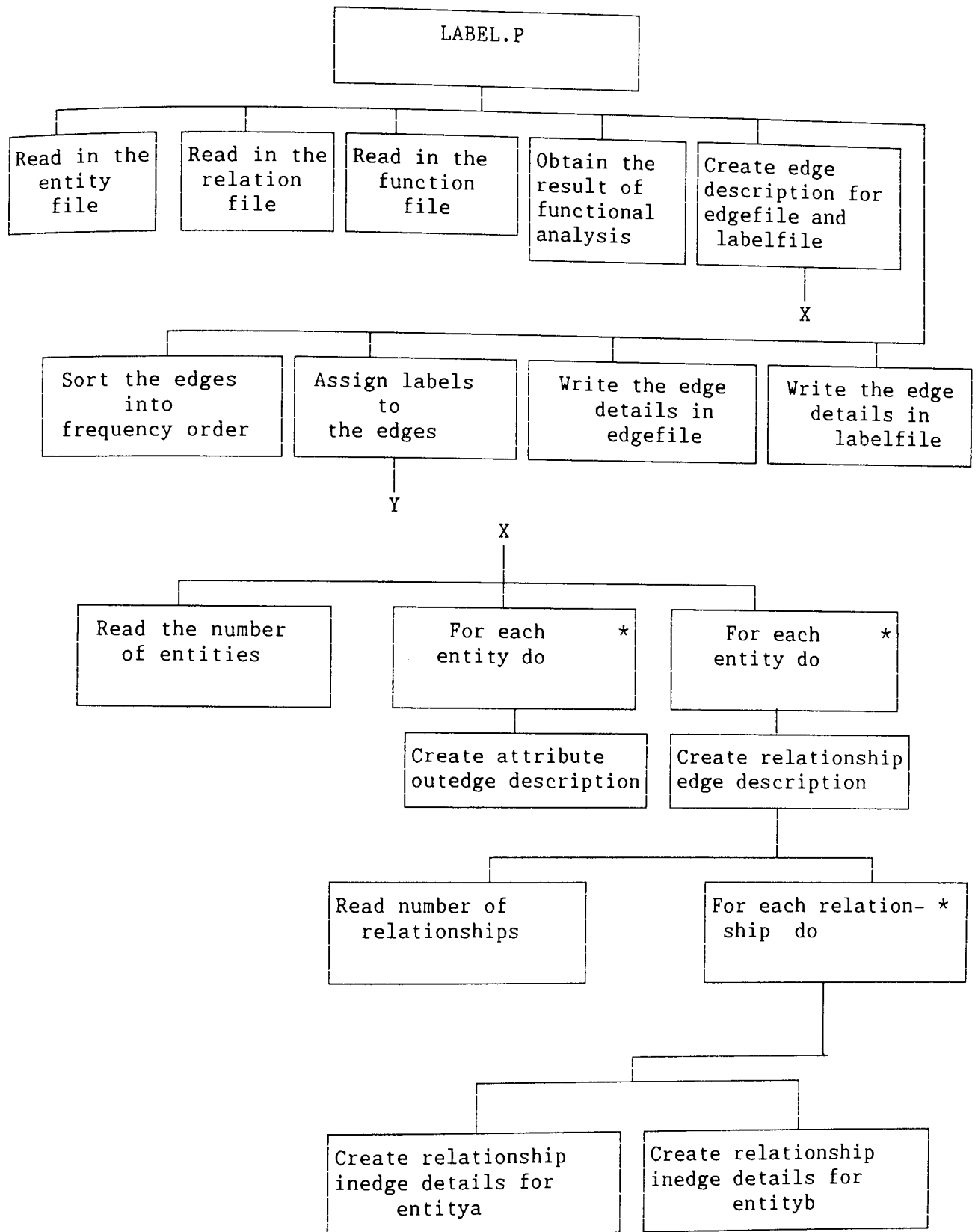
```

```
{WRITE IN THE EDGE DETAILS}
rewrite (edgefile, 'hospedge');
writeln (edgefile, noofedges);
for i := 1 to noofedges do
with edgemat[i] do
  begin
  write (edgefile, edgelab);
  write (edgefile, edgefreq);
  write (edgefile, edgetype);
  write (edgefile, edgepnt1);
  write (edgefile, edgepnt2);
  write (edgefile, edgepnt3);
  write (edgefile, edgepnt4);
  writeln (edgefile)
  end;

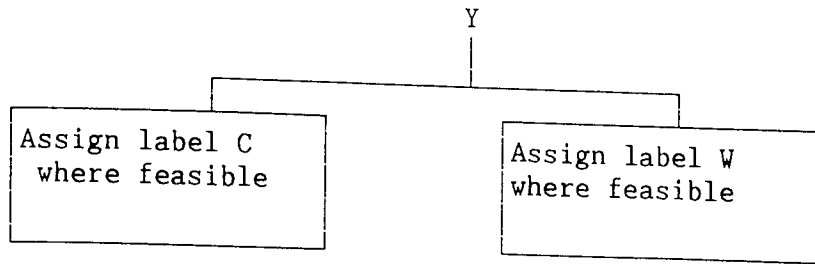
{WRITE IN THE LABELLED FILE}

rewrite (labelfile, 'hosplabel');
writeln (labelfile, noofentities);
for i := 1 to noofentities do
with labelmat[i] do
  begin
  writeln (labelfile, noofoutedge);
  writeln (labelfile, noofinedge);
  for j := 1 to noofoutedge do
with outedge[j] do
  begin
  write (labelfile, oedgetype);
  write (labelfile, oedgepnt);
  write (labelfile, oedgefreq);
  write (labelfile, oedgelabel);
  writeln (labelfile)
  end;
  for j := 1 to noofinedge do
with inedge[j] do
  begin
  write (labelfile, iedgepnt);
  write (labelfile, iedgefreq);
  write (labelfile, iedgelabel);
  writeln (labelfile)
  end
  end
end.
end.
```

LABEL.P







ING.P

{THIS PROGRAM CREATES A LOGICAL MODEL SUITABLE FOR INGRES DBMS}

```

program ingimp (input, output, relfile);

const maxstrlength = 20;
      maxnorel = 40;
      maxnoent = 40;
      maxnoatt = 20;
      maxrelatt = 20;
      maxnokey = 5;
      maxnoidnt = 5;
      awith = 'WITH';
      awithout = 'WITHOUT';
      maxnoprops = 20;
      maxinedge = 40;
      maxoutedge = 40;
      maxnoedge = 400;

type str = array [1..maxstrlength] of char;
      relstat = (weth, wewithout);
      props = record
        entptr: integer;
        attptr: integer
      end;
      relation = record
        rname : str;
        entitya : integer;
        degenta : char;
        membshpa : char;
        entityb : integer;
        degentb : char;
        membshpb : char;
        case rs : relstat of
          weth : (norelatt : integer;
                 relatt : array [1..maxrelatt] of str);
          wewithout : ()
        end;
        attributes = array[0..maxnoatt] of str;

      entity = record
        ename : str;
        keycount : integer;
        noatts : integer;
        entatt : attributes
      end;

      normrel = record
        nrname : integer;
        identcnt : integer;
        norrelidnt : array [1..maxnoidnt] of props;
        nrkeycnt : integer;
        norrelkey : array [1..maxnokey] of props;
        case nrs : relstat of
          weth : (nrattcnt : integer;
                 nrelatt : array [1..maxrelatt] of integer);
          wewithout : ()
        end;
end;

```

```

norment = record
    nename : str;
    noofprops : integer;
    identcnt : integer;
    norentatts : array [1..maxnoprops] of props
end;

outdet = record
    oedgetype : char;
    oedgepnt : integer;
    oedfreq : integer;
    oedgelabel : char
end;

indet = record
    iedgepnt : integer;
    iedfreq : integer;
    iedgelabel : char
end;

edgedet = record
    edgelab : char;
    edfreq : integer;
    edgetype : char;
    edgepnt1 : integer;
    edgepnt2 : integer;
    edgepnt3 : integer;
    edgepnt4 : integer
end;

graphdet = record
    noofoutedge : integer;
    noofinedge : integer;
    outedge : array[1..maxoutedge] of outdet;
    inedge : array[1..maxinedge] of indet
end;

ingent = record
    struc : char;
    edgeno : integer;
    diredge : char
end;

var relchart : array [1..maxnorel] of relation;
    entchart : array [1..maxnoent] of entity;
    norrelchtr : array [1..maxnorel] of normrel;
    norentchrt : array [1..maxnoent] of norment;
    ingentchart : array [1..maxnoent] of ingent;
    edgemat : array[1..maxnoedge] of edgedet;
    labelmat : array[1..maxnoent] of graphdet;
    noofrelations : integer;
    relfile : text;
    i, j, k, l : integer;
    noelfile : integer;
    noofentities : integer;
    entfile : text;
    mapfile : text;

```

```
    tempbuff : str;
    found : boolean;
    correct : boolean;
    answer : str;
    noofnormrel : integer;
    totnfreq : integer;
    condition : boolean;
    icondition : boolean;
    edgefile : text;
    labelfile : text;
    noofedges : integer;

procedure readstr (var f : text; var s : str);
    var ptr: integer;
    begin
    ptr := 0;
    while not eoln (f) and (ptr < maxstrlength ) do
        begin
        ptr := ptr + 1;
        read (f, s[ptr]);
        end;
    while ptr < maxstrlength do
        begin
        ptr := ptr + 1;
        s[ptr] := ' '
        end
    end;

procedure writestr (var f : text; var s : str);
    var i : integer;
    begin
    for i := 1 to maxstrlength do
        write (f, s[i]);
    end;

function equalstr (a, b : str): boolean;
    var ptr : integer;
        equal : boolean;
    begin
    equal := true;
    ptr := 0;
    while equal and (ptr < maxstrlength ) do
        begin
        ptr := ptr + 1;
        if a[ptr] <> b[ptr] then equal := false
        end;
    equalstr := equal
    end;
```

```

procedure findent (var int : integer);
  var l : integer;
  begin
    l := 0;
    found := false;
    while (not found) and (l < noofentities) do
      begin
        l := l + 1;
        if equalstr (tempbuff, entchart[l].ename) then
          begin
            int := l;
            found := true
          end
        end
      end;
end;

begin
  {READ IN THE ENTITY FILE}
  reset (entfile, 'hospent');
  readln (entfile, noofentities);
  for i := 1 to noofentities do
    with entchart[i] do
      begin
        readln (entfile);
        readstr (entfile, ename);
        readln (entfile);
        readln (entfile, keycount);
        readln (entfile, noatts);
        readln (entfile);
        for l := 0 to keycount do readstr (entfile,entatt[l]);
        readln (entfile);
        readln (entfile);
        k := 1;
        for j := (keycount + 1) to noatts do
          begin
            readstr (entfile, entatt[j]);
            k := k + 1;
            if k > 4 then
              begin
                readln (entfile);
                k := 1
              end
            end;
          readln (entfile);
          readln (entfile)
        end;
      end;
    end;
  end;

  {READ IN THE RELATION FILE}
  reset (relfile, 'hosprel');
  readln (relfile, noofrelations);
  for i := 1 to noofrelations do
    with relchart[i] do

```

```

begin
  readln (relfile);
  readstr (relfile, rname);
  readln (relfile);
  readstr (relfile, tempbuff);
  findent (entitya);
  readln(relfile, degenta);
  readln (relfile, membshpa);
  readstr (relfile, tempbuff);
  findent (entityb);
  readln (relfile, degentb);
  readln (relfile, membshpb);
  readstr (relfile, tempbuff);
  if tempbuff = awith then rs := weth
  else rs := wewithout;
  readln (relfile);
  if rs = weth then
    begin
      readln (relfile, norelatt);
      k := 1;
      for j := 1 to norelatt do
        begin
          readstr (relfile, relatt[j]);
          k := k + 1;
          if k > 4 then
            begin
              readln (relfile);
              k := 1
            end
          end
        end
      end
    end
  end;

```

```
{MAP THE RELATIONS}
```

```

j := 0;
for i := 1 to noofrelations do
  with relchart[i] do
    begin
      if (degenta <> '1' ) and (membshpb <> 'o' ) then
        begin
          j := j + 1;
          with norrelchtr [j] do
            begin
              if (degenta = '1') and (degentb = '1') and (membshpb = 'n') then
                begin
                  nrname := i;
                  identcnt := entchart[entitya].keycount;
                  for k := 1 to identcnt do
                    begin
                      norrelidnt[k].entptr := entitya;
                      norrelidnt[k].attptr := k
                    end;
                  nrkeycnt := entchart[entityb].keycount;
                  for k := 1 to nrkeycnt do
                    begin
                      norrelkey[k].entptr := entityb;
                      norrelkey[k].attptr := k
                    end
                  end
                end
            end
          end
        end
      end
    end
  end
end

```

```

else
if (degenta = '1') and (degentb = 'n') and (membshpb = 'n') then
begin
nrname := 1;
identcnt := entchart[entityb].keycount;
for k := 1 to identcnt do
begin
norrelidnt[k].entptr := entityb;
norrelidnt[k].attptr := k
end;
nrkeycnt := entchart[entitya].keycount;
for k := 1 to nrkeycnt do
begin
norrelkey[k].entptr := entitya;
norrelkey[k].attptr := k
end
end
else
begin
nrname := i;
identcnt := (entchart[entitya].keycount) +
(entchart[entityb].keycount);
for k := 1 to entchart[entitya].keycount do
begin
norrelidnt[k].entptr := entitya;
norrelidnt[k].attptr := k
end;
l := 0;
for k := ((entchart[entitya].keycount)+1) to identcnt do
begin
l := l + 1;
norrelidnt[k].entptr := entityb;
norrelidnt[k].attptr := l
end;
nrkeycnt := 0
end;
nrs := rs;
if nrs = weth then
begin
nrattcnt := norelatt;
for k := 1 to nrattcnt do
begin
nrelatt[k] := k
end
end
end
end
end;
noofnormrel := j;

```

```

{MAP ALL THE ENTITIES}
for i := 1 to noofentities do
  with norentchrt[i] do
    begin
      nename := entchart[i].ename;
      identcnt := entchart[i].keycount;
      noofprops := entchart[i].noatts;
      for j := 1 to entchart[i].noatts do
        begin
          norentatts[j].entptr := i;
          norentatts[j].attptr := j;
        end;
      l := 0;
      while l < noofrelations do
        begin
          l := l + 1;
          if relchart[l].entityb = i then
            begin
              if (relchart[l].degenta = '1') and (relchart[l].membshpb = 'o')
              then
                begin
                  for k := 1 to entchart[relchart[l].entitya].keycount do
                    begin
                      noofprops := noofprops + 1;
                      norentatts[noofprops].entptr := relchart[l].entitya;
                      norentatts[noofprops].attptr := k;
                    end
                  end
                end
            end
          end
        end
      end;
end;

```

```

{READ THE EDGE DETAILS}

```

```

reset (edgefile, 'hospedge');
readln (edgefile, noofedges);
for i := 1 to noofedges do
  with edgemat[i] do
    begin
      read (edgefile, edgelab);
      read (edgefile, edgefreq);
      read (edgefile, edgetype);
      read (edgefile, edgepnt1);
      read (edgefile, edgepnt2);
      read (edgefile, edgepnt3);
      read (edgefile, edgepnt4);
      readln (edgefile)
    end;
end;

```

```

{READ THE LABELLED FILE}

```

```

reset (labelfile, 'hosplabel');
readln (labelfile, noofentities);
for i := 1 to noofentities do
  with labelmat[i] do
    begin
      readln (labelfile, noofoutedge);
      readln (labelfile, noofinedge);
      for j := 1 to noofoutedge do

```



```

with outedge[j] do
  begin
    read (labelfile, oedgetype);
    read (labelfile, oedgepnt);
    read (labelfile, oedgefreq);
    read (labelfile, oedgelabel);
    readln (labelfile)
  end;
for j := 1 to noofinedge do
with inedge[j] do
  begin
    read (labelfile, iedgepnt);
    read (labelfile, iedgefreq);
    read (labelfile, iedgelabel);
    readln (labelfile)
  end
end;

for i := 1 to noofentities do
with norentchrt[i] do
with labelmat[i] do
  begin
    j := 0;
    condition := false;
    while (j < noofoutedge) and (not condition) do
      begin
        j := j + 1;
        if (outedge[j].oedgelabel = 'W' ) or
            (outedge[j].oedgelabel = 'C') then
          condition := true
        end;
      totinfreq := 0;
      for k := 1 to noofinedge do
        begin
          totinfreq := totinfreq + inedge[k].iedgefreq
        end;

      l:= 0;
      icondition := false;
      while (l< noofinedge) and (not icondition) do
        begin
          l := l +1;
          if (inedge[l].iedgelabel = 'W') or
              (inedge[l].iedgelabel = 'C') then
            icondition := true
          end;

      {IF fi + SUM(INEDGE FREQUENCY) > FREQUENCY OF EDGE MARKED 'W' or 'C'}
      {THEN HASH ON IDENTIFIER EDGE}

      if condition then
        begin
          if outedge[1].oedgefreq + totinfreq >= outedge[j].oedgefreq then
            begin
              ingentchart[i].struc := 'H';
              ingentchart[i].edgeno := 1;
              ingentchart[i].diredge := 'o'
            end;

```

```
{IF OUTEDGE MARKED 'W' OR 'C' IS AN ATTRIBUTE THEN ISAM ON THAT PROPERTY}
```

```
  if (outedge[1].oedgefreq + totinfreq < outedge[j].oedgefreq) and
    (outedge[j].oedgetype = 'a') then
    begin
      ingentchart[i].struc := 'M';
      ingentchart[i].edgeno := j;
      ingentchart[i].diredge := 'o'
    end
```

```
end
```

```
{IF INEDGE MARKED 'W' or 'C' HASH ON PRIMARY KEY}
else if icondition then
```

```
  begin
    ingentchart[i].struc := 'H';
    ingentchart[i].edgeno := relchart[inedge[1].iedgepnt].entitya;
    ingentchart[i].diredge := 'i'
  end
```

```
else
```

```
  begin
    ingentchart[i].struc := 'H';
    ingentchart[i].edgeno := 1;
    ingentchart[i].diredge := 'o'
  end
```

```
end;
```

```
{WRITE TO THE MAPFILE SUITABLE FOR INGRES SCHEMA}
```

```
rewrite (mapfile, 'ingfile');
writeln (mapfile,noofentities);
for i := 1 to noofentities do
  with norentchrt[i] do
  with ingentchart[i] do
    begin
      writeln (mapfile);
      writestr (mapfile,nename);
      write (mapfile, noofprops);
      write (mapfile,' ');
      writeln (mapfile,identcnt);
      for j := 1 to identcnt do
        with norentatts[j] do
          writestr (mapfile, entchart[entptr].entatt[attptr]);
          writeln (mapfile);
          k:= 1;
          for j:= (identcnt + 1) to noofprops do
            with norentatts[j] do
              begin
                writestr (mapfile, entchart[entptr].entatt[attptr]);
                k := k+1;
                if k > 4 then
                  begin
                    writeln (mapfile);
                    k := 1
                  end
                end;
              end;
            end;
          end;
        end;
      end;
      writeln (mapfile);
      if struc = 'H' then
        write (mapfile, 'Hash it on');
```

```

if struc = 'M' then
write (mapfile, 'ISAM it on ');
if diredge = 'o' then
begin
if edgeno = 1 then
begin
for j := 1 to identcnt do
writestr (mapfile, entchart[i].entatt[j])
end
else
writestr (mapfile, entchart[i].entatt[edgeno])
end
else
begin
for j := 1 to entchart[edgeno].keycount do
writestr (mapfile, entchart[edgeno].entatt[j])
end;
writeln (mapfile)
end;
end;

```

(WRITE THE NORMALISED RELATIONS SUITABLE FOR INGRES SCHEMA IN THE MAPFILE)

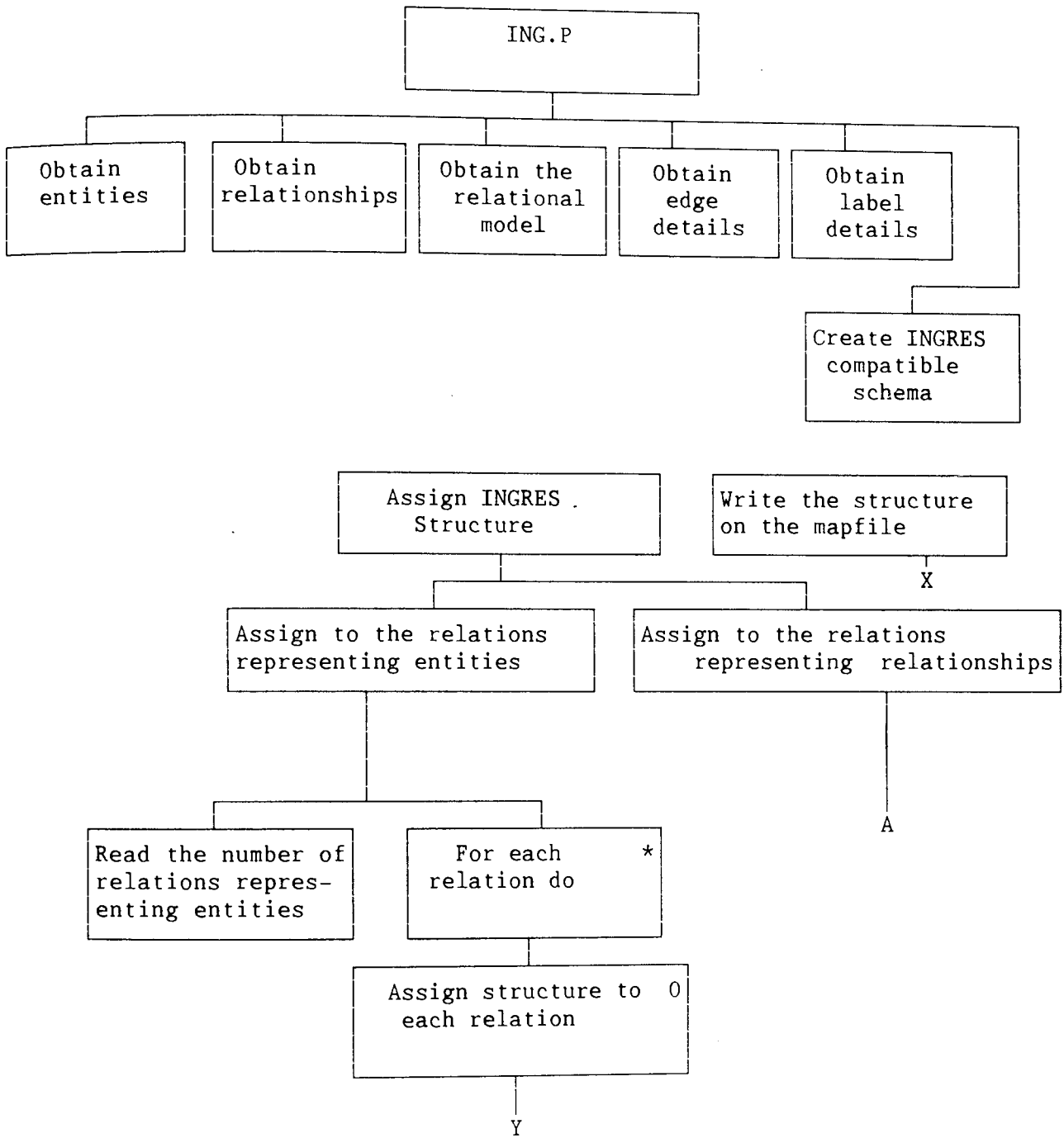
```

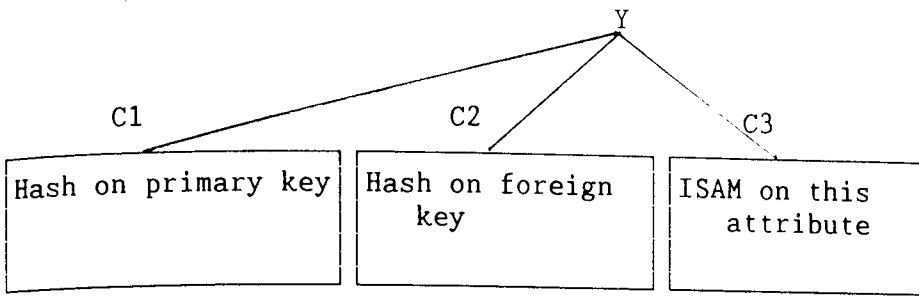
rewrite (mapfile, 'ingrel');
writeln (mapfile, noofnormrel);
for i := 1 to noofnormrel do
with norrelchtr[i] do
begin
writestr (mapfile, relchart[nrname].rname);
writeln(mapfile);
writeln (mapfile, identcnt);
k := 1;
for j := 1 to identcnt do
with norrelidnt[j] do
begin
writestr (mapfile, entchart[entptr].entatt[attptr]);
k := k+1;
if k > 4 then
begin
writeln(mapfile);
k := 1
end
end;
for j := 1 to nrkeycnt do
with norrelkey[j] do
begin
writestr(mapfile, entchart[entptr].entatt[attptr]);
k := k + 1;
if k > 4 then
begin
writeln (mapfile);
k := 1
end
end;
end;
end;

```

```
if nrs = weth then
  begin
    for j := 1 to nrattent do
      with relchart[nrname] do
        begin
          writestr(mapfile, relatt[nrelatt[j]]);
          k := k + 1;
          if k > 4 then
            begin
              writeln(mapfile);
              k := 1
            end
          end
        end
      end;
    writeln(mapfile);
    write(mapfile, 'Hash it on ');
    for k := 1 to entchart[relchart[nrname].entitya].keycount do
      begin
        writestr (mapfile, entchart[relchart[nrname].entitya].entatt[k])
      end;
    writeln (mapfile)
  end
end.
```

ING.P

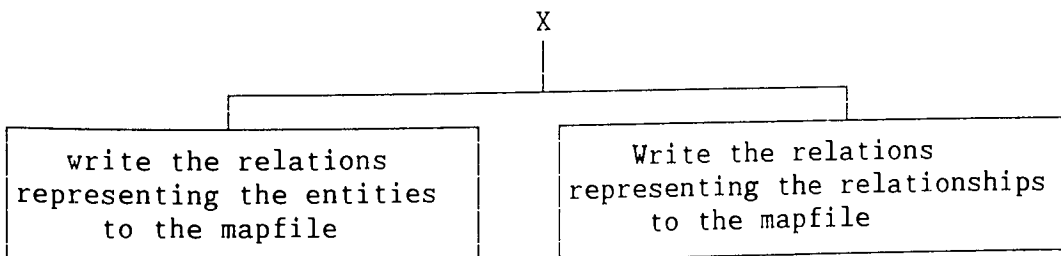
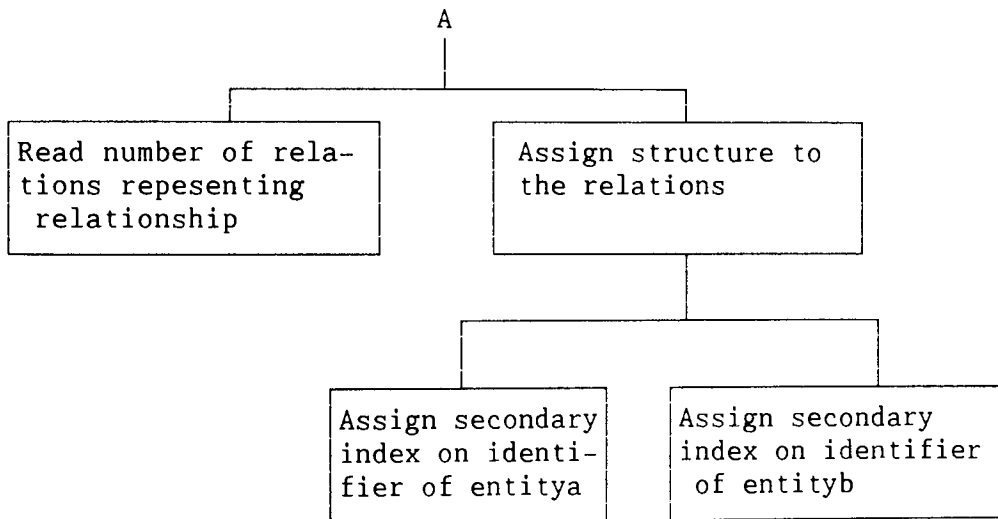




C1 = Outedge is labelled W or C  
 sum of frequency of inedges  
 is greater than the frequency  
 of outedge

C2 = Inedge is labelled W or C

C3 = Outedge is labelled W or C  
 and sum of inedge frequency  
 is less than the frequency  
 of outedge



MIM.P

{THIS PROGRAM CREATES A SCHEMA SUITABLE FOR MIMER DBMS}

```

program mimimp (input, output, relfile);

const maxstrlength = 20;
      maxnorel = 40;
      maxnoent = 40;
      maxnoatt = 20;
      maxrelatt = 20;
      maxnokey = 5;
      maxnoidt = 5;
      awith = 'WITH';
      awithout = 'WITHOUT';
      maxnoprops = 20;
      maxinedge = 40;
      maxoutedge = 40;
      maxnoedge = 400;

type str = array [1..maxstrlength] of char;
      relstat = (weth, wewithout);
      props = record
        entptr: integer;
        attptr: integer
      end;
      relation = record
        rname : str;
        entitya : integer;
        degenta : char;
        membshpa : char;
        entityb : integer;
        degentb : char;
        membshpb : char;
        case rs : relstat of
          weth : (norelatt : integer;
                  relatt : array [1..maxrelatt] of str);
          wewithout : ()
        end;
      attributes = array[0..maxnoatt] of str;

      entity = record
        ename : str;
        keycount : integer;
        noatts : integer;
        entatt : attributes
      end;

      normrel = record
        nrname : integer;
        identcnt : integer;
        norrelidnt : array [1..maxnoidt] of props;
        nrkeycnt : integer;
        norrelkey : array [1..maxnokey] of props;
        case nrs : relstat of
          weth : (nrattcnt : integer;
                  nrelatt : array [1..maxrelatt] of integer);
          wewithout : ()
        end;

```

```

norment = record
    nename : str;
    noofprops : integer;
    identcnt : integer;
    norentatts : array [1..maxnoprops] of props
end;

outdet = record
    oedgetype : char;
    oedgepnt : integer;
    oedfreq : integer;
    oedgelabel : char
end;

indet = record
    iedgepnt : integer;
    iedfreq : integer;
    iedgelabel : char
end;

edgedet = record
    edgelab : char;
    edfreq : integer;
    edgetype : char;
    edgepnt1 : integer;
    edgepnt2 : integer;
    edgepnt3 : integer;
    edgepnt4 : integer
end;

graphdet = record
    noofoutedge : integer;
    noofinedge : integer;
    outedge : array[1..maxoutedge] of outdet;
    inedge : array[1..maxinedge] of indet
end;

miment = record
    struc : char;
    edgeno : integer;
    diredge : char
end;

var relchart : array [1..maxnorel] of relation;
    entchart : array [1..maxnoent] of entity;
    norrelchtr : array [1..maxnorel] of normrel;
    norentchrt : array [1..maxnoent] of norment;
    mimentchart : array [1..maxnoent] of miment;
    edgemat : array[1..maxnoedge] of edgedet;
    labelmat : array[1..maxnoent] of graphdet;
    noofrelations : integer;
    relfile : text;
    i, j, k, l : integer;
    noelfile : integer;
    noofentities : integer;
    entfile : text;
    mapfile : text;
    tempbuff : str;

```



```
found : boolean;
correct : boolean;
answer : str;
noofnormrel : integer;
totinfreq : integer;
condition : boolean;
icondition : boolean;
edgefile : text;
labelfile : text;
noofedges : integer;

procedure readstr (var f : text; var s : str);
var ptr: integer;
begin
ptr := 0;
while not eoln (f) and (ptr < maxstrlength ) do
begin
ptr := ptr + 1;
read (f, s[ptr]);
end;
while ptr < maxstrlength do
begin
ptr := ptr + 1;
s[ptr] := ' '
end
end;

procedure writestr (var f : text; var s : str);
var i : integer;
begin
for i := 1 to maxstrlength do
write (f, s[i]);
end;

function equalstr (a, b : str): boolean;
var ptr : integer;
equal : boolean;
begin
equal := true;
ptr := 0;
while equal and (ptr < maxstrlength ) do
begin
ptr := ptr + 1;
if a[ptr] <> b[ptr] then equal := false
end;
equalstr := equal
end;
```

```

procedure findent (var int : integer);
  var l : integer;
  begin
    l := 0;
    found := false;
    while (not found) and (l < noofentities) do
      begin
        l := l + 1;
        if equalstr (tempbuff, entchart[l].ename) then
          begin
            int := l;
            found := true
          end
        end
      end
    end;

begin
  {READ IN THE ENTITY FILE}
  reset (entfile, 'hospent');
  readln (entfile, noofentities);
  for i := 1 to noofentities do
    with entchart[i] do
      begin
        readln (entfile);
        readstr (entfile, ename);
        readln (entfile);
        readln (entfile, keycount);
        readln (entfile, noatts);
        readln (entfile);
        for l := 0 to keycount do readstr (entfile,entatt[l]);
        readln (entfile);
        readln (entfile);
        k := 1;
        for j := (keycount + 1) to noatts do
          begin
            readstr (entfile, entatt[j]);
            k := k + 1;
            if k > 4 then
              begin
                readln (entfile);
                k := 1
              end
            end;
          readln (entfile);
          readln (entfile)
        end;
      end;
    end;
  end;
end;

```

```

{READ IN THE RELATION FILE}
reset (relfile, 'hosprel');
readln (relfile, noofrelations);
for i := 1 to noofrelations do
with relchart[i] do
begin
readln (relfile);
readstr (relfile, rname);
readln (relfile);
readstr (relfile, tempbuff);
findent (entitya);
readln (relfile, degenta);
readln (relfile, membshpa);
readstr (relfile, tempbuff);
findent (entityb);
readln (relfile, degentb);
readln (relfile, membshpb);
readstr (relfile, tempbuff);
if tempbuff = awith then rs := weth
else rs := wewithout;
readln (relfile);
if rs = weth then
begin
readln (relfile, norelatt);
k := 1;
for j := 1 to norelatt do
begin
readstr (relfile, relatt[j]);
k := k + 1;
if k > 4 then
begin
readln (relfile);
k := 1
end
end
end
end;

{MAP THE RELATIONS}

j := 0;
for i := 1 to noofrelations do
with relchart[i] do
begin
if (degenta <> '1' ) and (membshpb <> 'o' ) then
begin
j := j + 1;
with norrelchtr [j] do
begin
if (degenta = '1') and (degentb = '1') and (membshpb = 'n') then
begin
nrname := i;
identcnt := entchart[entitya].keycount;
for k := 1 to identcnt do
begin
norrelidnt[k].entptr := entitya;
norrelidnt[k].attptr := k
end;
nrkeycnt := entchart[entityb].keycount;

```

```

    for k := 1 to nrkeycnt do
        begin
            norrelkey[k].entptr := entityb;
            norrelkey[k].attptr := k
        end
    end
else
if (degenta = '1') and (degentb = 'n') and (membshpb = 'n') then
    begin
        nrname := 1;
        identcnt := entchart[entityb].keycount;
        for k := 1 to identcnt do
            begin
                norrelidnt[k].entptr := entityb;
                norrelidnt[k].attptr := k
            end;
        nrkeycnt := entchart[entitya].keycount;
        for k := 1 to nrkeycnt do
            begin
                norrelkey[k].entptr := entitya;
                norrelkey[k].attptr := k
            end
        end
    end
else .

    begin
        nrname := i;
        identcnt := (entchart[entitya].keycount) +
            (entchart[entityb].keycount);
        for k := 1 to entchart[entitya].keycount do
            begin
                norrelidnt[k].entptr := entitya;
                norrelidnt[k].attptr := k
            end;
        l := 0;
        for k := ((entchart[entitya].keycount)+1) to identcnt do
            begin
                l := l + 1;
                norrelidnt[k].entptr := entityb;
                norrelidnt[k].attptr := l
            end;
        nrkeycnt := 0
    end;
nrs := rs;
if nrs = weth then
    begin
        nrattcnt := norelatt;
        for k := 1 to nrattcnt do
            begin
                nrelatt[k] := k
            end
        end
    end
end
end
end;
noofnormrel := j;

```

```

{MAP ALL THE ENTITIES}
for i := 1 to noofentities do
  with norentchrt[i] do
    begin
      nename := entchart[i].ename;
      identcnt := entchart[i].keycount;
      noofprops := entchart[i].noatts;

      for j := 1 to entchart[i].noatts do
        begin
          norentatts[j].entptr := i;
          norentatts[j].attptr := j
        end;
      l := 0;
      while l < noofrelations do
        begin
          l := l + 1;
          if relchart[l].entityb = i then
            begin
              if (relchart[l].degenta = '1') and (relchart[l].membshpb = 'o')
then
                begin
                  for k := 1 to entchart[relchart[l].entitya].keycount do
                    begin
                      noofprops := noofprops + 1;
                      norentatts[noofprops].entptr := relchart[l].entitya;
                      norentatts[noofprops].attptr := k
                    end
                  end
                end
              end
            end
          end;
end;

{READ THE EDGE DETAILS}

reset (edgefile, 'hospedge');
readln (edgefile, noofedges);
for i := 1 to noofedges do
  with edgemat[i] do
    begin
      read (edgefile, edgelab);
      read (edgefile, edgefreq);
      read (edgefile, edgetype);
      read (edgefile, edgepnt1);
      read (edgefile, edgepnt2);
      read (edgefile, edgepnt3);
      read (edgefile, edgepnt4);
      readln (edgefile)
    end;
end;

```

```

{READ THE LABELLED FILE}

reset (labelfile, 'hosplabel');
readln (labelfile, noofentities);
for i := 1 to noofentities do
with labelmat[i] do
begin
readln (labelfile, noofoutedge);
readln (labelfile, noofinedge);
for j := 1 to noofoutedge do
with outedge[j] do
begin
read (labelfile, oedgetype);
read (labelfile, oedgepnt);
read (labelfile, oedfreq);
read (labelfile, oedgelabel);
readln (labelfile)
end;
for j := 1 to noofinedge do
with inedge[j] do
begin
read (labelfile, iedgepnt);
read (labelfile, iedfreq);
read (labelfile, iedgelabel);
readln (labelfile)
end
end;

for i := 1 to noofentities do
with norentchrt[i] do
with labelmat[i] do
begin
j := 0;
condition := false;

{CHECK WHETHER ANY OUTEDGE OF THE ENTITY IS LABELLED 'W' or 'C'}
while (j < noofoutedge) and (not condition) do
begin
j := j + 1;
if (outedge[j].oedgelabel = 'W' ) or
(outedge[j].oedgelabel = 'C') then
condition := true
end;

l:= 0;
icondition := false;

{CHECK WHETHER ANY INEDGE OF THE ENTITY IS LABELLED 'W' or 'C'}
while (l< noofinedge) and (not icondition) do
begin
l := l +1;
if (inedge[l].iedgelabel = 'W') or
(inedge[l].iedgelabel = 'C') then
icondition := true
end;

```

```

{IF THE LABELLED OUTEDGE IS NOT THE PRIMARY KEY}
{CREATE SECONDARY INDEX ON THAT ATTRIBUTE}

if (condition) and (j<> 1) then
  begin

    mimentchart[i].struc := 'I';
    mimentchart[i].edgeno := j;
    mimentchart[i].diredge := 'o'
  end
else

{IF AN INEDGE IS LABELLED 'W' OR 'C' THEN CREATE SECONDARY INDEX ON}
{THE KEY OF THE ENTITYA OF THE RELATIONSHIP}
  if icondition then
    begin
      mimentchart[i].struc := 'I';
      mimentchart[i].edgeno := relchart[inedge[1].iedgepnt].entitya;
      mimentchart[i].diredge := 'i'
    end
  else
    begin
      mimentchart[i].struc := 'N';
      mimentchart[i].edgeno := 0;
      mimentchart[i].diredge := 'n'
    end

end;

{WRITE THE MIMER SCHEMA ON THE MAPFILE}
{FIRST WRITE THE RELATIONS CORRESPONDING TO THE ENTITIES}

rewrite (mapfile, 'mimfile');
writeln(mapfile, noofentities);
for i := 1 to noofentities do
  with norentchrt[i] do
  with mimentchart[i] do
    begin
      writeln (mapfile);
      writestr (mapfile, nename);
      write(mapfile,noofprops);
      write(mapfile,' ');
      writeln (mapfile,identcnt);
      for j := 1 to identcnt do
        with norentatts[j] do
          writestr (mapfile, entchart[entptr].entatt[attptr]);
        writeln(mapfile);
        k := 1;
        for j := (identcnt + 1) to noofprops do
          with norentatts[j] do
            begin
              writestr (mapfile, entchart[entptr].entatt[attptr]);
              k := k +1;
              if k > 4 then
                begin
                  writeln (mapfile);
                  k := 1
                end
            end
          end;
        end;
      end;

```

```

writeln (mapfile);
if struc <> 'N' then
begin
write (mapfile, 'Invert it on ');
if diredge = 'o' then
  writestr(mapfile, entchart[i].entatt[edgeno])
else
  begin
  for j := 1 to entchart[edgeno].keycount do
    writestr (mapfile, entchart[edgeno].entatt[j])
  end
end;
writeln (mapfile);
end;

```

{THEN WRITE THE RELATIONS CORRESPONDING TO THE RELATIONSHIPS IN THE MAPFILE}

```

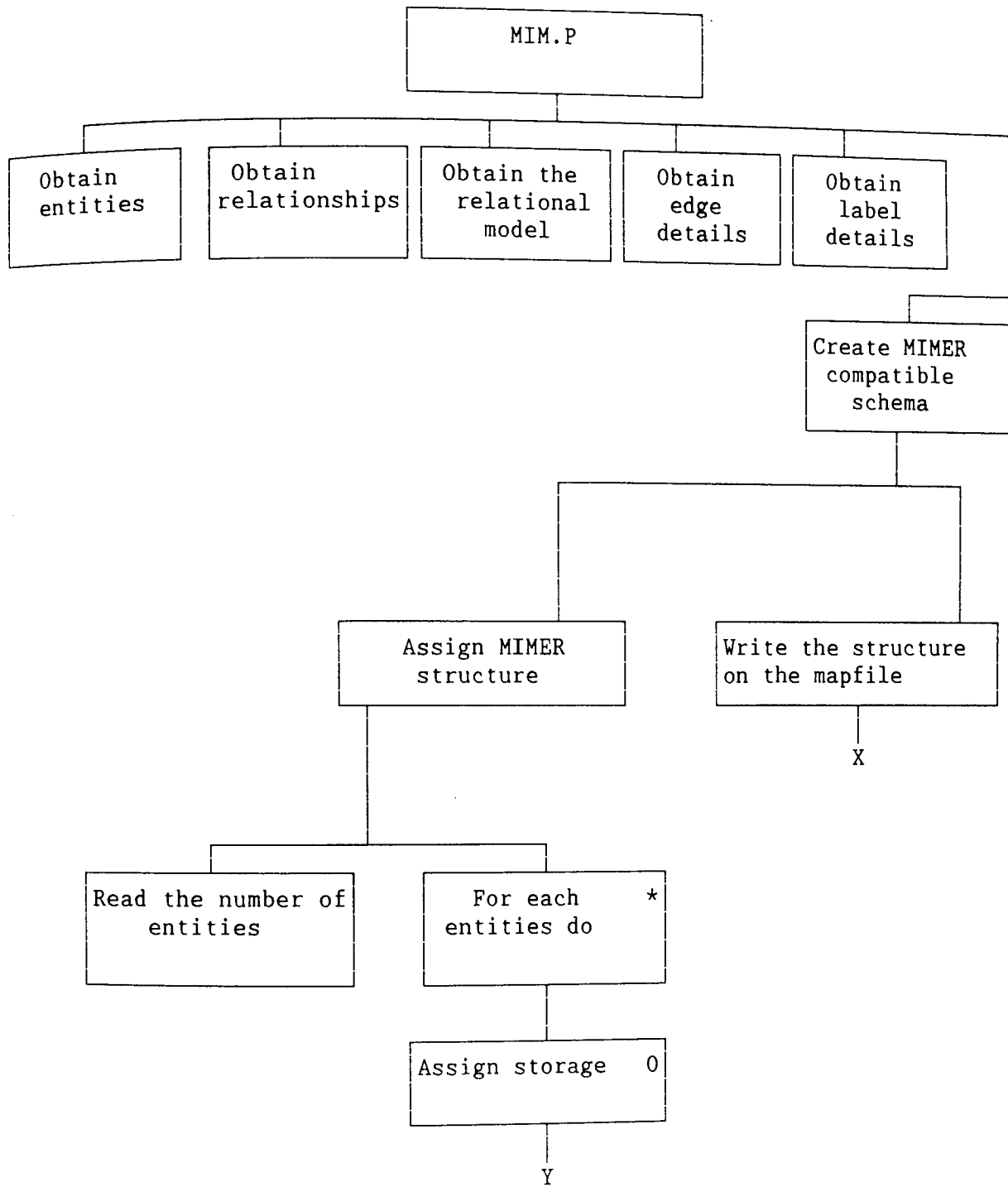
rewrite (mapfile, 'mimrel');
writeln (mapfile, noofnormrel);
for i := 1 to noofnormrel do
with norrelchtr[i] do
  begin
  writestr (mapfile, relchart[nrname].rname);
  writeln (mapfile);
  writeln (mapfile, identcnt);
  k := 1;
  for j := 1 to identcnt do
  with norrelidnt[j] do
    begin
    writestr (mapfile, entchart[entptr].entatt[attptr]);
    k := k + 1;
    if k > 4 then
      begin
      writeln (mapfile);
      k := 1
      end
    end;
  for j := 1 to nrkeycnt do
  with norrelkey[j] do
    begin
    writestr (mapfile, entchart[entptr].entatt[attptr]);
    k := k + 1;
    if k > 4 then
      begin
      writeln(mapfile);
      k := 1
      end;
    end;
  end;
end;

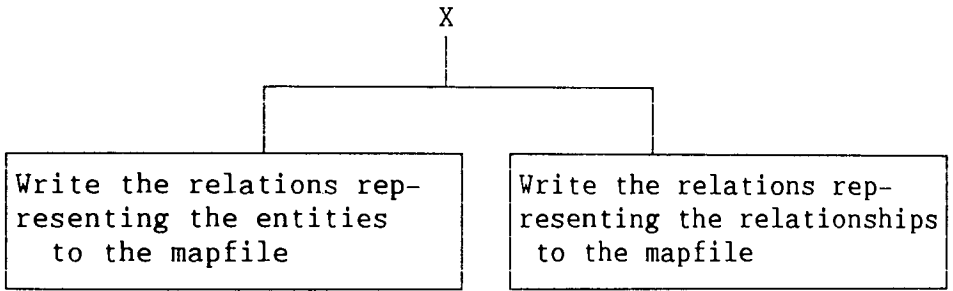
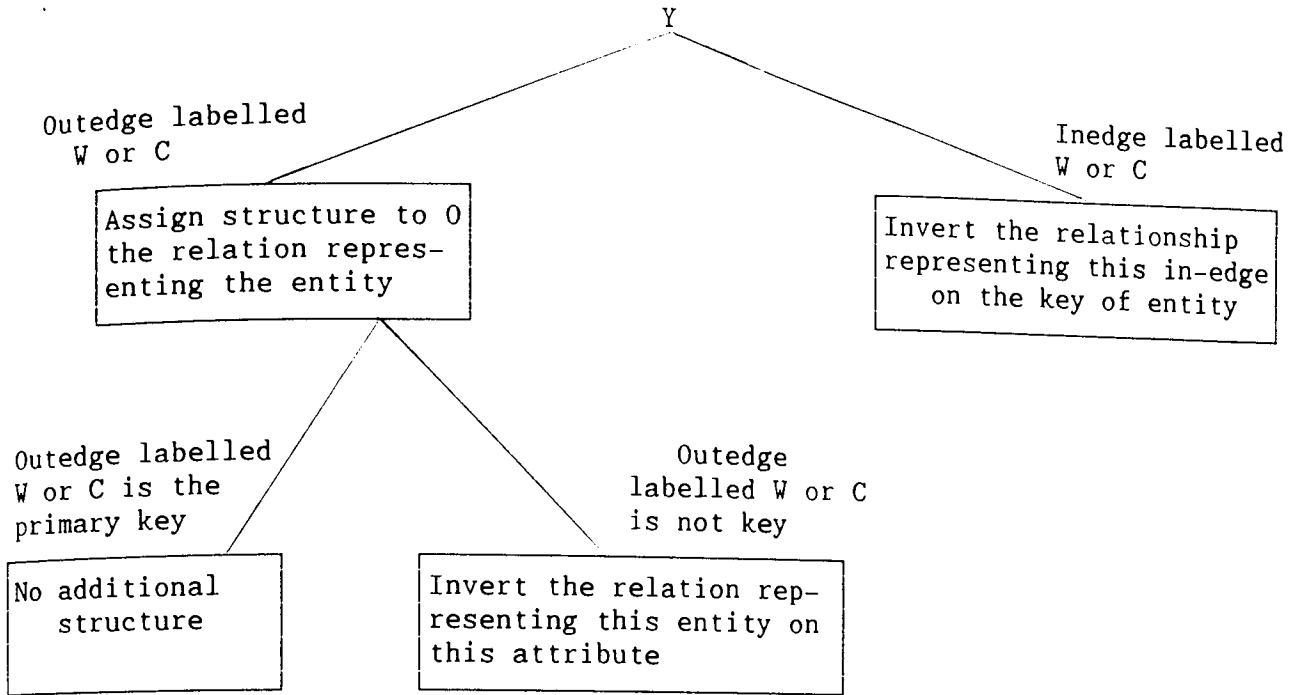
```



```
    if nrs = weth then
        begin
            for j := 1 to nrattcnt do
                with relchart[nrname] do
                    begin
                        writestr (mapfile, relatt[nrelatt[j]]);
                        k := k + 1;
                        if k > 4 then
                            begin
                                writeln (mapfile);
                                k := 1
                            end
                        end
                    end;
                writeln (mapfile);
            write (mapfile, 'Invert it on ');
            for k := 1 to entchart[relchart[nrname].entitya].keycount do
                begin
                    writestr (mapfile, entchart[relchart[nrname].entitya].entatt[k])
                end;
                writeln (mapfile)
            end
        end.
end.
```

MIM.P





APPENDIX D

TYPICAL RUNS OF PROGRAMS

A Typical Run of the Program ENTEST.P

Script started on Mon Feb 23 16:23:01 1987  
\$ px entest.ex

Number of new entities

1

Entity name

**Supplier**

NO of attributes in the primary key

1

No of attributes

4

Input the attributes one per line

**supplier-no**

**name**

**address**

**status**

23293 statements executed in 0007.840 seconds cpu time.  
script done on Mon Feb 23 16:29:56 1987

The run of the program 'ENTEST.P' shows the insertion of a new entity named 'Supplier'. The responses of the user of the tools are in bold type.

The run shows that the user intends to insert one entity. The entity name is 'Supplier'. The entity has four attributes which are

supplier-no  
name  
address  
status

The identifier of the entity is a single attribute, which is the 'supplier-no'.

## A Typical Run of the Program RELTEST.P

Script started on Mon Feb 23 16:33:53 1987  
\$ pix reltest.p  
Execution begins...

Number of new relationships

1

Relationship name

**Supplied drug**

First entity

**Supplier**

THIS ENTITY DOES NOT EXIST

**Supplier**

Degree of first entity 1 or m

m

Membership of first entity o for obligatory n for non-obligatory

n

Second entity

**Drug**

Degree of second entity 1 or n

n

Membership of second entity o or n

n

State WITH or WITHOUT Attributes

**WITHOUT**

Execution terminated.

34850 statements executed in 0011.760 seconds cpu time.

\$ ^D

script done on Mon Feb 23 16:38:22 1987

The run of the Program 'RELTEST.P' shows the insertion of a new relationship named 'Supplied drug'. The responses of the user are in bold type.

The name of the first entity is 'Supplier'. On the first instance, the user makes a mistake with the spelling of the entity 'Supplier'. Due to this mistake, the program does not recognise the entity and reports that the entity does not exist in the entity list. The user tries again, and types the name of the entity. The program recognises the entity and accepts it as the first entity. The program then enquires about the degree of the first entity. The user types in 'm'. The program then enquires about the membership of the first entity. The user types in 'n' for non-obligatory. The program then asks questions about the second entity. The user types in the name, degree and the membership of the second entity, which are 'Drug', 'n' and 'n' respectively. The relationship has no attributes.

**A Typical Run of the Program FUNCTEST.P**

Script started on Mon Feb 23 12:03:41 1987  
\$ px functest.ex

Number of new function

1

Function name

**Select-nurse**

Status of function

**PRIMARY**

Frequency of the function

400

Number of ENTITY accessed

1

Name of entity

**Nurse**

THIS ENTITY DOES NOT EXIST. TRY AGAIN

**Nurses**

State whether SELECT.BY.KEY/RELATION/ATTRIBUTES

**SELECT.BY.ATTRIBUTES**

State the attribute

**grade**

State selection clause RANGE or EQUIJOIN.

**RANGE**

42252 statements executed in 0014.380 seconds cpu time.

\$ ^D

script done on Mon Feb 23 12:07:11 1987



The run of the Program 'FUNCTEST.P' shows the user inserting information regarding a new function named 'Select-nurse'. The function has a 'Primary' status. The responses of the user is in bold type.

The number of times the function is performed in a day is 400. A single entity 'Nurses' is accessed by the function. The entity is selected by an attribute. The attribute 'grade' is used to select the entity and the selection clause used is the 'range' clause.

As the entity name is 'Nurses' the program does not recognise the when the user types in 'Nurse' and gives an error message.

APPENDIX E

OUTPUTS FROM PROGRAMS

Output of ENTEST.P  
ENTFILE

26

Patient

1  
12

PRIMARY KEY

pat-no

pat-name

pat-address

pat-category

reference

sex

date-of-birth

marital status

next-of-kin

blood group

allergy

x-ray information

Disease

1  
4

PRIMARY KEY

disease-no

disease-name

contageous

treatment

Consultant

1  
4

PRIMARY KEY

emp-no

con-name

con-address

speciality

Clinical-session

2  
9

PRIMARY KEY

clinic-no

date

time-of-start

time-of-finish

speciality

est: new pat

est: repeat pat

new pat booked

rep: pat booked

Clinic-time-schedule

3  
5

PRIMARY KEY

clinic-no

date

time

free/booked

new/repeat

O/p-W/l

1  
4

PRIMARY KEY

W/l-no

speciality

list size

selection criteria

Surgical W/l

1  
4

PRIMARY KEY

S-W/l-no

speciality

list length

selection criteria

Surgical-session

2  
8

PRIMARY KEY

session-no

date

start time

end time

max: major cases

booked major cases

max: minor cases

booked minor cases

Admission W/l

1  
4

PRIMARY KEY

A-W/l-no

speciality

list length

selection criteria

Blood request

1  
9

PRIMARY KEY

Request-no

reason

date.

save serum

reserve blood units

reserve conc: units doctors no:

blood taker's no:

date of request

Pat-rec-blood

1  
8

PRIMARY KEY

ref-no

ward-code

blood-group

haemoglobin level

pregnancies

past-trans

antibody

prev:/ref-no

Antibody cases

1  
8

PRIMARY KEY

anti-ref

blood group  
B.T.S confirmphenotype  
dateantibody  
comment

E.D.C

Drug

1  
5

PRIMARY KEY

drug-code

name

strength

pack-size

expiry-code

supplier

1  
4

PRIMARY KEY

supplier-no

name

address

status

Prescription

1  
3

PRIMARY KEY

pres-no

pres-date

prescriber

X-ray request

1  
5

PRIMARY KEY

request-no

investigation

requestor

urgency

reason

X-ray sessions

4  
6

PRIMARY KEY

date

unit

morning/afternoon

x-ray type

maximum load

numbers booked

Staff details

1  
5

PRIMARY KEY

emp-no  
name grade F.T/P.T address

Wards

1  
6

PRIMARY KEY

Ward-code  
no-of-beds w-type w-description nursing load  
nursing dependency

Ward bed

2  
4

PRIMARY KEY

Ward-code bed-no  
male/female free/occupied

Nurses

1  
10

PRIMARY KEY

emp-no  
N.I.no name grade unit code  
entry grade location code qualification code date of birth  
full equivalent

Duty

4  
5

PRIMARY KEY

grade date shift W-code

number required

Ind: workload

2  
4

PRIMARY KEY

week commencing emp-no  
max-hours hours-booked

Appendix E

Outputs

In-patient  
4  
8

PRIMARY KEY  
date entering

pat-no

w-code

bed-no

discharge date

illness

patient type

comment

N-absence rec  
4  
5

PRIMARY KEY  
year

emp-no

day

month

reason

Supplier  
1  
4

PRIMARY KEY

supplier-no

name

address

status

Output from RELTEST.P  
RELFILE

22

Pat-Cons	
Patient	m
o	
Consultant	n
n	
WITH	
1	
Date registered	
Pat-Disease	
Patient	m
n	
Disease	n
n	
WITH	
1	
Date diagnosed	
Cons-Clinic	
Consultant	1
n	
Clinical-session	n
o	
WITHOUT	
Clinic/schedule	
Clinical-session	1
o	
Clinic-time-schedulen	
s	
WITHOUT	
Appointment	
Patient	1
n	
Clinic-time-schedulen	
n	
WITHOUT	
Consultant/S-W/l	
Consultant	1
n	
Surgical W/l	n
o	
WITHOUT	
Consultant/S-session	
Consultant	1
n	
Surgical-session	n
o	
WITHOUT	

Pat/Blood req				
Patient	1			
n				
Blood request	n			
o				
WITHOUT				
Pat/Pat-rec-blood				
Patient	1			
n				
Pat-rec-blood	n			
o				
WITHOUT				
Patient/Antibody				
Patient	1			
n				
Antibody cases	1			
o				
WITHOUT				
Pat/prescription				
Patient	1			
n				
Prescription	n			
o				
WITHOUT				
Pat/X-ray request				
Patient	1			
n				
X-ray request	n			
o				
WITHOUT				
X-ray sess/Pat				
X-ray sessions	m			
n				
Patient	n			
n				
WITH				
1				
suggestion				
Staff/session				
Staff details	m			
n				
X-ray sessions	n			
o				
WITHOUT				
Pat/O/p-W/l				
Patient	m			
n				
O/p-W/l	n			
n				
WITH				
4				
position	reference	complaints	urgency	



Pat/S-W/l				
Patient	m			
n				
Surgical W/l	n			
n				
WITH				
5				
urgency	op-code	suggestion	non-availability	
position				
Consultant/O/p-W/l				
Consultant	1			
n				
O/p-W/l	n			
o				
WITHOUT				
Duty-rota				
Nurses	m			
n				
Duty	n			
n				
WITHOUT				
Nurses/Wards				
Nurses	m			
n				
Wards	n			
n				
WITHOUT				
Nurses/Workload				
Nurses	1			
n				
Ind: workload	n			
s				
WITHOUT				
Pat/In-pat				
Patient	1			
n				
In-patient	n			
s				
WITHOUT				
Supplied drug				
Supplier	m			
n				
Drug	n			
n				
WITHOUT				

RELATION-NAME	FORMAT-REL		DEGREE-A	DEGREE-B
	ENTITY-A	ENTITY-B		
Pat-Cons o n	Patient	Consultant	m	n
Pat-Disease n n	Patient	Disease	m	n
Cons-Clinic n o	Consultant	Clinical-session	1	n
Clinic/schedule o s	Clinical-session	Clinic-time-schedule	1	n
Appointment n n	Patient	Clinic-time-schedule	1	n
Consultant/S-W/l n o	Consultant	Surgical W/l	1	n
Consultant/S-session n o	Consultant	Surgical-session	1	n
Pat/Blood req n o	Patient	Blood request	1	n
Pat/Pat-rec-blood n o	Patient	Pat-rec-blood	1	n
Patient/Antibody n o	Patient	Antibody cases	1	1
Pat/prescription n o	Patient	Prescription	1	n
Pat/X-ray request n o	Patient	X-ray request	1	n
X-ray sess/Pat n n	X-ray sessions	Patient	m	n
Staff/session n o	Staff details	X-ray sessions	m	n
Pat/O/p-W/l n n	Patient	O/p-W/l	m	n
Pat/S-W/l n n	Patient	Surgical W/l	m	n
Consultant/O/p-W/l n o	Consultant	O/p-W/l	1	n
Duty-rota n n	Nurses	Duty	m	n

Appendix E

			Outputs	
Nurses/Wards n                    n	Nurses	Wards	m	n
Nurses/Workload n                    s	Nurses	Ind: workload	1	n
Pat/In-pat n                    s	Patient	In-patient	1	n
Supplied drug n                    n	Supplier	Drug	m	n

Output from FUNCTEST.P  
FUNCFILE

7			
Enquirey	PRIMARY		300
4			
Patient	SELECT.BY.KEY		
pat-no			
Clinic-time-schedule	SELECT.BY.RELATION		
Appointment			
Clinical-session	SELECT.BY.RELATION		
Clinic/schedule			
Consultant	SELECT.BY.RELATION		
Cons-Clinic			
Query blood	PRIMARY		350
2			
Patient	SELECT.BY.ATTRIBUTES		
pat-name	EQUIJOIN		
Pat-rec-blood	SELECT.BY.RELATION		
Pat/Pat-rec-blood			
Fix-Appt:	PRIMARY		350
5			
Patient	SELECT.BY.KEY		
pat-no			
O/p-W/l	SELECT.BY.RELATION		
Pat/O/p-W/l			
Consultant	SELECT.BY.RELATION		
Consultant/O/p-W/l			
Clinical-session	SELECT.BY.RELATION		
Cons-Clinic			
Clinic-time-schedule	SELECT.BY.RELATION		
Clinic/schedule			
Sel-nurse-rota	SECONDARY		150
2			
Duty	SELECT.BY.ATTRIBUTES		
grade	EQUIJOIN		
Ind: workload	SELECT.BY.RELATION		
Nurses/Workload			

Appendix E

Outputs

Check rota 2	PRIMARY	200	
Duty date	SELECT.BY.KEY shift		W-code grade
Nurses Duty-rota	SELECT.BY.RELATION		
Pat/dis enq 2	PRIMARY	200	
Patient pat-no	SELECT.BY.KEY		
Disease Pat-Disease	SELECT.BY.RELATION		
Select-nurse 1	PRIMARY	400	
Nurses grade	SELECT.BY.ATTRIBUTES RANGE		

Output from ANALYSIS.P  
ANALYSISFILE(Entity Analysis)

Entity-Name :- Patient

	Primary functions		Secondary functions	
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	850	0	0
pat-name	0	350	0	0
pat-address	0	0	0	0
pat-category	0	0	0	0
reference	0	0	0	0
sex	0	0	0	0
date-of-birth	0	0	0	0
marital status	0	0	0	0
next-of-kin	0	0	0	0
blood group	0	0	0	0
allergy	0	0	0	0
x-ray information	0	0	0	0

Entity-Name :- Disease

	Primary functions		Secondary functions	
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	0	0	0
disease-name	0	0	0	0
contageous	0	0	0	0
treatment	0	0	0	0

Entity-Name :- Consultant

	Primary functions		Secondary functions	
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	0	0	0
con-name	0	0	0	0
con-address	0	0	0	0
speciality	0	0	0	0

Entity-Name :- Clinical-session

	Primary functions		Secondary functions	
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	0	0	0
time-of-start	0	0	0	0
time-of-finish	0	0	0	0
speciality	0	0	0	0
est: new pat	0	0	0	0
est: repeat pat	0	0	0	0
new pat booked	0	0	0	0
rep: pat booked	0	0	0	0

Entity-Name :- Clinic-time-schedule

	Primary functions		Secondary functions	
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	0	0	0
free/booked	0	0	0	0
new/repeat	0	0	0	0

Entity-Name :- O/p-W/l

	Primary functions		Secondary functions	
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	0	0	0
speciality	0	0	0	0
list size	0	0	0	0
selection criteria	0	0	0	0

Entity-Name :- Surgical W/l

	Primary functions		Secondary functions	
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	0	0	0
speciality	0	0	0	0
list length	0	0	0	0
selection criteria	0	0	0	0

Entity-Name :- Surgical-session

	Primary functions		Secondary functions	
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	0	0	0
start time	0	0	0	0
end time	0	0	0	0
max: major cases	0	0	0	0
booked major cases	0	0	0	0
max: minor cases	0	0	0	0
booked minor cases	0	0	0	0

Entity-Name :- Admission W/l

	Primary functions		Secondary functions	
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	0	0	0
speciality	0	0	0	0
list length	0	0	0	0
selection criteria	0	0	0	0

Entity-Name :- Blood request

	Primary functions		Secondary functions	
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	0	0	0
reason	0	0	0	0
date	0	0	0	0
save serum	0	0	0	0
reserve blood units	0	0	0	0
reserve conc: units	0	0	0	0
doctors no:	0	0	0	0
blood taker's no:	0	0	0	0
date of request	0	0	0	0

Entity-Name :- Pat-rec-blood

	Primary functions		Secondary functions	
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	0	0	0
ward-code	0	0	0	0
blood-group	0	0	0	0
haemoglobin level	0	0	0	0
pregnancies	0	0	0	0
past-trans	0	0	0	0
antibody	0	0	0	0
prev:/ref-no	0	0	0	0

Entity-Name :- Antibody cases

	Primary functions		Secondary functions	
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	0	0	0
blood group	0	0	0	0
phenotype	0	0	0	0
antibody	0	0	0	0
E.D.C	0	0	0	0
B.T.S confirm	0	0	0	0
date	0	0	0	0
comment	0	0	0	0

Entity-Name :- Drug

	Primary functions		Secondary functions	
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	0	0	0
name	0	0	0	0
setlength	0	0	0	0
pack-size	0	0	0	0
expirey-code	0	0	0	0

Entity-Name :- supplier

	Primary functions		Secondary functions	
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	0	0	0
name	0	0	0	0
address	0	0	0	0
status	0	0	0	0

Entity-Name :- Prescription

	Primary functions		Secondary functions	
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	0	0	0
pres-date	0	0	0	0
prescriber	0	0	0	0



Entity-Name :-	Primary functions		Secondary functions	
	Rfrequency	Efrequency	Rfrequency	Efrequency
X-ray request				
Primary-key	0	0	0	0
investigation	0	0	0	0
requestor	0	0	0	0
urgency	0	0	0	0
reason	0	0	0	0
Entity-Name :-	X-ray sessions			
	Primary functions	Secondary functions	Rfrequency	Efrequency
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	0	0	0
maximum load	0	0	0	0
numbers booked	0	0	0	0
Entity-Name :-	Staff details			
	Primary functions	Secondary functions	Rfrequency	Efrequency
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	0	0	0
name	0	0	0	0
grade	0	0	0	0
F.T/P.T	0	0	0	0
address	0	0	0	0
Entity-Name :-	Wards			
	Primary functions	Secondary functions	Rfrequency	Efrequency
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	0	0	0
no-of-beds	0	0	0	0
w-type	0	0	0	0
w-description	0	0	0	0
nursing load	0	0	0	0
nursing dependency	0	0	0	0
Entity-Name :-	Ward bed			
	Primary functions	Secondary functions	Rfrequency	Efrequency
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	0	0	0
male/female	0	0	0	0
free/occupied	0	0	0	0

Entity-Name :- Nurses

	Primary functions		Secondary functions	
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	0	0	0
N.I.no	0	0	0	0
name	0	0	0	0
grade	400	0	0	0
unit code	0	0	0	0
entry grade	0	0	0	0
location code	0	0	0	0
qualification code	0	0	0	0
date of birth	0	0	0	0
full equivalent	0	0	0	0

Entity-Name :- Duty

	Primary functions		Secondary functions	
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	200	0	0
number required	0	0	0	0

Entity-Name :- Ind: workload

	Primary functions		Secondary functions	
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	0	0	0
max-hours	0	0	0	0
hours-booked	0	0	0	0

Entity-Name :- In-patient

	Primary functions		Secondary functions	
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	0	0	0
discharge date	0	0	0	0
illness	0	0	0	0
patient type	0	0	0	0
comment	0	0	0	0

Entity-Name :- N-absence rec

	Primary functions		Secondary functions	
	Rfrequency	Efrequency	Rfrequency	Efrequency
Primary-key	0	0	0	0
reason	0	0	0	0

## ANALYSISFILE(Relationship Analysis)

Relationship Name	Primary Function Frequency	Secondary Function Frequency
Pat-Cons	0	0
Pat-Disease	200	0
Cons-Clinic	650	0
Clinic/schedule	650	0
Appointment	300	0
Consultant/S-W/l	0	0
Consultant/S-session	0	0
Pat/Blood req	0	0
Pat/Pat-rec-blood	350	0
Patient/Antibody	0	0
Pat/prescription	0	0
Pat/X-ray request	0	0
X-ray sess/Pat	0	0
Staff/session	0	0
Pat/O/p-W/l	350	0
Pat/S-W/l	0	0
Consultant/O/p-W/l	350	0
Duty-rota	200	0
Nurses/Wards	0	0
Nurses/Workload	0	150
Pat/In-pat	0	0

Output from RELMAP.P  
MAPFILE

25

Patient	12	1		
pat-no				
pat-name	pat-address	pat-category	reference	
sex	date-of-birth	marital status	next-of-kin	
blood group	allergy	x-ray information		
Disease	4	1		
disease-no				
disease-name	contageous	treatment		
Consultant	4	1		
emp-no				
con-name	con-address	speciality		
Clinical-session	10	2		
clinic-no	date			
time-of-start	time-of-finish	speciality	est: new pat	
est: repeat pat	new pat booked	rep: pat booked	emp-no	
Clinic-time-schedule	5	3		
clinic-no	date	time		
free/booked	new/repeat			
O/p-W/l	5	1		
W/l-no				
speciality	list size	selection criteria	emp-no	
Surgical W/l	5	1		
S-W/l-no				
speciality	list length	selection criteria	emp-no	
Surgical-session	9	2		
session-no	date			
start time	end time	max: major cases	booked major cases	
max: minor cases	booked minor cases	emp-no		
Admission W/l	4	1		
A-W/l-no				
speciality	list length	selection criteria		

Blood request	10	1		
Request-no				
reason	date		save serum	reserve blood units
reserve conc: units	doctors no:		blood taker's no:	date of request
pat-no				
Pat-rec-blood	9	1		
ref-no				
ward-code	blood-group		haemoglobin level	pregnancies
past-trans	antibody		prev:/ref-no	pat-no
Antibody cases	9	1		
anti-ref				
blood group	phenotype		antibody	E.D.C
B.T.S confirm	date		comment	pat-no
Drug	5	1		
drug-code				
name	strength		pack-size	expiry-code
supplier	4	1		
supplier-no				
name	address		status	
Prescription	4	1		
pres-no				
pres-date	prescriber		pat-no	
X-ray request	6	1		
request-no				
investigation	requestor		urgency	reason
pat-no				
X-ray sessions	6	4		
date	unit		morning/afternoon	x-ray type
maximum load	numbers booked			
Staff details	5	1		
emp-no				
name	grade		F.T/P.T	address

## Appendix E

## Outputs

Wards	6	1		
Ward-code				
no-of-beds	w-type	w-description	nursing load	
nursing dependency				
Ward bed	4	2		
Ward-code	bed-no			
male/female	free/occupied			
Nurses	10	1		
emp-no				
N.I.no	name	grade	unit code	
entry grade	location code	qualification code	date of birth	
full equivalent				
Duty	5	4		
date	shift	W-code	grade	
number required				
Ind: workload	4	2		
week commencing	emp-no			
max-hours	hours-booked			
In-patient	8	4		
pat-no	w-code	bed-no	date entering	
discharge date	illness	patient type	comment	
N-absence rec	5	4		
emp-no	day	month	year	
reason				

Output from CODMAP.P  
CODENT

Record name is Patient

Patient KEY is pat-no  
 pat-name ;  
 pat-address ;  
 pat-category ;  
 reference ;  
 sex ;  
 date-of-birth ;  
 marital status ;  
 next-of-kin ;  
 blood group ;  
 allergy ;  
 x-ray information ;

Record name is Disease

Disease KEY is disease-no  
 disease-name ;  
 contagious ;  
 treatment ;

Record name is Consultant

Consultant KEY is emp-no  
 con-name ;  
 con-address ;  
 speciality ;

Record name is Clinical-session

Clinical-session KEY is clinic-no ,date  
 time-of-start ;  
 time-of-finish ;  
 speciality ;  
 est: new pat ;  
 est: repeat pat ;  
 new pat booked ;  
 rep: pat booked ;

Record name is Clinic-time-schedule

Clinic-time-schedule KEY is clinic-no ,date ,time  
 free/booked ;  
 new/repeat ;

Record name is O/p-W/l

O/p-W/l KEY is W/l-no  
 speciality ;  
 list size ;  
 selection criteria ;

Record name is Surgical W/l  
 Surgical W/l KEY is S-W/l-no  
 speciality ;  
 list length ;  
 selection criteria ;

Record name is Surgical-session  
 Surgical-session KEY is session-no ,date  
 start time ;  
 end time ;  
 max: major cases ;  
 booked major cases ;  
 max: minor cases ;  
 booked minor cases ;

Record name is Admission W/l  
 Admission W/l KEY is A-W/l-no  
 speciality ;  
 list length ;  
 selection criteria ;

Record name is Blood request  
 Blood request KEY is Request-no  
 reason ;  
 date ;  
 save serum ;  
 reserve blood units ;  
 reserve conc: units ;  
 doctors no: ;  
 blood taker's no: ;  
 date of request ;

Record name is Pat-rec-blood  
 Pat-rec-blood KEY is ref-no  
 ward-code ;  
 blood-group ;  
 haemoglobin level ;  
 pregnancies ;  
 past-trans ;  
 antibody ;  
 prev:/ref-no ;

Record name is Antibody cases  
 Antibody cases KEY is anti-ref  
 blood group ;  
 phenotype ;  
 antibody ;  
 E.D.C ;  
 B.T.S confirm ;  
 date ;  
 comment ;



Record name is Drug

```
Drug                KEY is drug-code
  name                ;
  strength            ;
  pack-size          ;
  expiry-code        ;
```

Record name is supplier

```
supplier            KEY is supplier-no
  name                ;
  address             ;
  status              ;
```

Record name is Prescription

```
Prescription        KEY is pres-no
  pres-date          ;
  prescriber         ;
```

Record name is X-ray request

```
X-ray request        KEY is request-no
  investigation      ;
  requestor          ;
  urgency             ;
  reason              ;
```

Record name is X-ray sessions

```
X-ray sessions      KEY is date
, morning/afternoon , x-ray type      , unit
  maximum load      ;
  numbers booked    ;
```

Record name is Staff details

```
Staff details        KEY is emp-no
  name                ;
  grade               ;
  F.T/P.T             ;
  address             ;
```

Record name is Wards

```
Wards                KEY is Ward-code
  no-of-beds         ;
  w-type              ;
  w-description       ;
  nursing load        ;
  nursing dependency ;
```

Record name is Ward bed

```
Ward bed             KEY is Ward-code      , bed-no
  male/female        ;
  free/occupied      ;
```

Record name is Nurses

Nurses KEY is emp-no  
 N.I.no ;  
 name ;  
 grade ;  
 unit code ;  
 entry grade ;  
 location code ;  
 qualification code ;  
 date of birth ;  
 full equivalent ;

Record name is Duty

Duty KEY is date ,shift ,W-code  
 ,grade  
 number required ;

Record name is Ind: workload

Ind: workload KEY is week commencing ,emp-no  
 max-hours ;  
 hours-booked ;

Record name is Pat-Cons

Pat-Cons KEY is pat-no ,emp-no  
 Date registered ;

Record name is Pat-Disease

Pat-Disease KEY is pat-no ,disease-no  
 Date diagnosed ;

Record name is X-ray sess/Pat

X-ray sess/Pat KEY is date ,unit  
 ,morning/afternoon ,x-ray type ,pat-no  
 suggestion ;

Record name is Staff/session

Staff/session KEY is emp-no ,date ,unit  
 ,morning/afternoon ,x-ray type

Record name is Pat/O/p-W/l

Pat/O/p-W/l KEY is pat-no ,W/l-no  
 position ;  
 reference ;  
 complaints ;  
 urgency ;



## CODREL

Set name is Pat/Pat-Cons  
Owner record is Patient  
Member record is  
Pat-Cons                   AUTOMATIC FIXED

Set name is Cons/Pat-Cons  
Owner record is Consultant  
Member record is  
Pat-Cons                   AUTOMATIC FIXED

Set name is Pat/Pat-Dis  
Owner record is Patient  
Member record is  
Pat-Disease                AUTOMATIC FIXED

Set name is Dis/Pat-Dis  
Owner record is Disease  
Member record is  
Pat-Disease                AUTOMATIC FIXED

Set name is Cons-Clinic  
Owner record is Consultant  
Member record is  
Clinical-session          AUTOMATIC MANDATORY

Set name is Clinic/schedule  
Owner record is Clinical-session  
Member record is  
Clinic-time-scheduleAUTOMATIC FIXED

Set name is Appointment  
Owner record is Patient  
Member record is  
Clinic-time-scheduleMANUAL OPTIONAL

Set name is Consultant/S-W/1  
Owner record is Consultant  
Member record is  
Surgical W/1               AUTOMATIC MANDATORY

Set name is Consultant/S-session  
Owner record is Consultant  
Member record is  
Surgical-session          AUTOMATIC MANDATORY

Set name is Pat/Blood req  
Owner record is Patient  
Member record is  
Blood request              AUTOMATIC MANDATORY

Set name is Pat/Pat-rec-blood  
Owner record is Patient  
Member record is  
Pat-rec-blood              AUTOMATIC MANDATORY

Set name is Patient/Antibody  
 Owner record is Patient  
 Member record is  
 Antibody cases           AUTOMATIC MANDATORY

Set name is Pat/prescription  
 Owner record is Patient  
 Member record is  
 Prescription            AUTOMATIC MANDATORY

Set name is Pat/X-ray request  
 Owner record is Patient  
 Member record is  
 X-ray request           AUTOMATIC MANDATORY

Set name is X-sess/Detail  
 Owner record is X-ray sessions  
 Member record is  
 X-ray sess/Pat         AUTOMATIC FIXED

Set name is Pat/Detail  
 Owner record is Patient  
 Member record is  
 X-ray sess/Pat         AUTOMATIC FIXED

Set name is Staff session detail  
 Owner record is Staff details  
 Member record is  
 Staff/session           AUTOMATIC FIXED

Set name is X-ray/staff detail  
 Owner record is X-ray sessions  
 Member record is  
 Staff/session           AUTOMATIC FIXED

Set name is Pat-0/p-W/l det:  
 Owner record is Patient  
 Member record is  
 Pat/0/p-W/l            AUTOMATIC FIXED

Set name is 0/p-W/l/Pat det:  
 Owner record is 0/p-W/l  
 Member record is  
 Pat/0/p-W/l            AUTOMATIC FIXED

Set name is Pat-S-W/l det:  
 Owner record is Patient  
 Member record is  
 Pat/S-W/l              AUTOMATIC FIXED

Set name is S-W/l Pat Det:  
 Owner record is Surgical W/l  
 Member record is  
 Pat/S-W/l              AUTOMATIC FIXED

Set name is Consultant/0/p-W/l  
 Owner record is Consultant  
 Member record is  
 0/p-W/l                AUTOMATIC MANDATORY

Set name is Nurses/Duty-rota  
Owner record is Nurses  
Member record is  
Duty-rota                   AUTOMATIC FIXED

Set name is Duty/Duty-rota  
Owner record is Duty  
Member record is  
Duty-rota                   AUTOMATIC FIXED

Set name is Nurses/Wards details  
Owner record is Nurses  
Member record is  
Nurses/Wards               AUTOMATIC FIXED

Set name is Wards/Nurses details  
Owner record is Wards  
Member record is  
Nurses/Wards               AUTOMATIC FIXED

Set name is Nurses/Workload  
Owner record is Nurses  
Member record is  
Ind: workload               AUTOMATIC FIXED

Output of LABEL.P  
EDGEFILE

	152				
W	850p	1	1	1	0
I	350a	1	2	2	0
I	0a	1	3	3	0
I	0a	1	4	4	0
I	0a	1	5	5	0
I	0a	1	6	6	0
I	0a	1	7	7	0
I	0a	1	8	8	0
I	0a	1	9	9	0
I	0a	1	10	10	0
I	0a	1	11	11	0
I	0a	1	12	12	0
W	0p	2	1	1	0
I	0a	2	2	2	0
I	0a	2	3	3	0
I	0a	2	4	4	0
W	0p	3	1	1	0
I	0a	3	2	2	0
I	0a	3	3	3	0
I	0a	3	4	4	0
W	0p	4	1	1	0
I	0a	4	3	2	0
I	0a	4	4	3	0
I	0a	4	5	4	0
I	0a	4	6	5	0
I	0a	4	7	6	0
I	0a	4	8	7	0
I	0a	4	9	8	0
W	0p	5	1	1	0
I	0a	5	4	2	0
I	0a	5	5	3	0
W	0p	6	1	1	0
I	0a	6	2	2	0
I	0a	6	3	3	0
I	0a	6	4	4	0
W	0p	7	1	1	0
I	0a	7	2	2	0
I	0a	7	3	3	0
I	0a	7	4	4	0
W	0p	8	1	1	0
I	0a	8	3	2	0
I	0a	8	4	3	0
I	0a	8	5	4	0
I	0a	8	6	5	0
I	0a	8	7	6	0
I	0a	8	8	7	0
W	0p	9	1	1	0
I	0a	9	2	2	0
I	0a	9	3	3	0
I	0a	9	4	4	0
W	0p	10	1	1	0
I	0a	10	2	2	0
I	0a	10	3	3	0
I	0a	10	4	4	0
I	0a	10	5	5	0

I	0a	10	6	6	0
I	0a	10	7	7	0
I	0a	10	8	8	0
I	0a	10	9	9	0
W	0p	11	1	1	0
I	0a	11	2	2	0
I	0a	11	3	3	0
I	0a	11	4	4	0
I	0a	11	5	5	0
I	0a	11	6	6	0
I	0a	11	7	7	0
I	0a	11	8	8	0
W	0p	12	1	1	0
I	0a	12	2	2	0
I	0a	12	3	3	0
I	0a	12	4	4	0
I	0a	12	5	5	0
I	0a	12	6	6	0
I	0a	12	7	7	0
I	0a	12	8	8	0
W	0p	13	1	1	0
I	0a	13	2	2	0
I	0a	13	3	3	0
I	0a	13	4	4	0
I	0a	13	5	5	0
W	0p	14	1	1	0
I	0a	14	2	2	0
I	0a	14	3	3	0
I	0a	14	4	4	0
W	0p	15	1	1	0
I	0a	15	2	2	0
I	0a	15	3	3	0
W	0p	16	1	1	0
I	0a	16	2	2	0
I	0a	16	3	3	0
I	0a	16	4	4	0
I	0a	16	5	5	0
W	0p	17	1	1	0
I	0a	17	5	2	0
I	0a	17	6	3	0
W	0p	18	1	1	0
I	0a	18	2	2	0
I	0a	18	3	3	0
I	0a	18	4	4	0
I	0a	18	5	5	0
W	0p	19	1	1	0
I	0a	19	2	2	0
I	0a	19	3	3	0
I	0a	19	4	4	0
I	0a	19	5	5	0
I	0a	19	6	6	0
W	0p	20	1	1	0
I	0a	20	3	2	0
I	0a	20	4	3	0
I	0p	21	1	1	0
I	0a	21	2	2	0
I	0a	21	3	3	0
W	400a	21	4	4	0
I	0a	21	5	5	0



I	Oa	21	6	6	0
I	Oa	21	7	7	0
I	Oa	21	8	8	0
I	Oa	21	9	9	0
I	Oa	21	10	10	0
W	200p	22	1	1	0
I	Oa	22	5	2	0
W	Op	23	1	1	0
I	Oa	23	3	2	0
I	Oa	23	4	3	0
W	Op	24	1	1	0
I	Oa	24	5	2	0
I	Oa	24	6	3	0
I	Oa	24	7	4	0
I	Oa	24	8	5	0
W	Op	25	1	1	0
I	Oa	25	5	2	0
I	Or	1	3	13	1
I	Or	1	2	14	1
I	Or	3	4	5	1
I	Or	4	5	9	1
I	Or	1	5	15	2
I	Or	3	7	6	1
I	Or	3	8	7	1
I	Or	1	10	16	1
I	Or	1	11	17	1
I	Or	1	12	18	1
I	Or	1	15	19	1
I	Or	1	16	20	1
I	Or	17	1	4	1
I	Or	18	17	6	1
I	Or	1	6	21	1
I	Or	1	7	22	2
I	Or	3	6	8	2
I	Or	21	22	11	1
I	Or	21	19	12	1
I	Or	21	23	13	1
I	Or	1	24	23	1

## LABELFILE

	25	
	23	
	1	
p	1	850C
a	0	350I
a	0	0I
a	0	0I
a	0	0I
a	0	0I
a	0	0I
a	0	0I
a	0	0I
a	0	0I
r	1	0I
r	2	200I
r	5	300I
r	8	0I
r	9	350I
r	10	0I
r	11	0I
r	12	0I
r	15	350I
r	16	0I
r	21	0I
	13	0I
	4	
	1	
p	1	0C
a	0	0I
a	0	0I
a	0	0I
	2	200I
	8	
	1	
p	1	0C
a	0	0I
a	0	0I
a	0	0I
r	3	650I
r	6	0I
r	7	0I
r	17	350I
	1	0I
	9	
	1	
p	1	0C
a	0	0I
a	0	0I
a	0	0I
a	0	0I
a	0	0I
a	0	0I
r	0	0I
	4	650I
	3	650I

	3	
	2	
p	1	0C
a	0	0I
a	0	0I
	4	650I
	5	300I
	4	
	2	
p	1	0C
a	0	0I
a	0	0I
a	0	0I
	15	350I
	17	350I
	4	
	2	
p	1	0C
a	0	0I
a	0	0I
a	0	0I
	6	0I
	16	0I
	7	
	1	
p	1	0C
a	0	0I
a	0	0I
a	0	0I
a	0	0I
a	0	0I
a	0	0I
	7	0I
	4	
	0	
p	1	0C
a	0	0I
a	0	0I
a	0	0I
	9	
	1	
p	1	0C
a	0	0I
a	0	0I
a	0	0I
a	0	0I
a	0	0I
a	0	0I
a	0	0I
a	0	0I
	8	0I
	8	
	1	
p	1	0C
a	0	0I
a	0	0I
a	0	0I
a	0	0I
a	0	0I

Appendix E

Outputs

a	0	0I
a	0	0I
	9	350I
	8	
	1	
p	1	0C
a	0	0I
a	0	0I
a	0	0I
a	0	0I
a	0	0I
a	0	0I
	10	0I
	5	
	0	
p	1	0C
a	0	0I
a	0	0I
a	0	0I
a	0	0I
	4	
	0	
p	1	0C
a	0	0I
a	0	0I
a	0	0I
	3	
	1	
p	1	0C
a	0	0I
a	0	0I
	11	0I
	5	
	1	
p	1	0C
a	0	0I
a	0	0I
a	0	0I
a	0	0I
	12	0I
	4	
	1	
p	1	0C
a	0	0I
a	0	0I
r	13	0I
	14	0I
	6	
	0	
p	1	0C
a	0	0I
a	0	0I
a	0	0I
a	0	0I
r	14	0I
	6	
	1	
p	1	0C

a	0	OI
a	0	OI
a	0	OI
a	0	OI
a	0	OI
	19	OI
	3	
	0	
p	1	OC
a	0	OI
a	0	OI
	13	
	0	
p	1	OI
a	0	OI
a	0	OI
a	0	400C
a	0	OI
a	0	OI
a	0	OI
a	0	OI
a	0	OI
a	0	OI
r	18	200I
r	19	OI
r	20	OI
	2	
	1	
p	1	200C
a	0	OI
	18	200I
	3	
	1	
p	1	OC
a	0	OI
a	0	OI
	20	OI
	5	
	1	
p	1	OC
a	0	OI
a	0	OI
a	0	OI
a	0	OI
	21	OI
	2	
	0	
p	1	OC
a	0	OI

Output of ING.P  
MAPFILE (Entities)

25

Patient	12	1		
pat-no				
pat-name	pat-address	pat-category	reference	
sex	date-of-birth	marital status	next-of-kin	
blood group	allergy	x-ray information		
Hash it onpat-no				
Disease	4	1		
disease-no				
disease-name	contageous	treatment		
Hash it ondisease-no				
Consultant	4	1		
emp-no				
con-name	con-address	speciality		
Hash it onemp-no				
Clinical-session	10	2		
clinic-no	date			
time-of-start	time-of-finish	speciality	est: new pat	
est: repeat pat	new pat booked	rep: pat booked	emp-no	
Hash it onclinic-no	date			
Clinic-time-schedule	5	3		
clinic-no	date	time		
free/booked	new/repeat			
Hash it onclinic-no	date	time		
O/p-W/l	5	1		
W/l-no				
speciality	list size	selection criteria	emp-no	
Hash it onW/l-no				
Surgical W/l	5	1		
S-W/l-no				
speciality	list length	selection criteria	emp-no	
Hash it onS-W/l-no				
Surgical-session	9	2		
session-no	date			
start time	end time	max: major cases	booked major cases	
max: minor cases	booked minor cases	emp-no		
Hash it onsession-no	date			
Admission W/l	4	1		
A-W/l-no				
speciality	list length	selection criteria		
Hash it onA-W/l-no				

Blood request	10	1		
Request-no				
reason	date		save serum	reserve blood units
reserve conc:	units doctors no:		blood taker's no:	date of request
pat-no				
Hash it on	Request-no			
Pat-rec-blood	9	1		
ref-no				
ward-code	blood-group		haemoglobin level	pregnancies
past-trans	antibody		prev:/ref-no	pat-no
Hash it on	ref-no			
Antibody cases	9	1		
anti-ref				
blood group	phenotype		antibody	E.D.C
B.T.S confirm	date		comment	pat-no
Hash it on	anti-ref			
Drug	5	1		
drug-code				
name	strength		pack-size	expiry-code
Hash it on	drug-code			
supplier	4	1		
supplier-no				
name	address		status	
Hash it on	supplier-no			
Prescription	4	1		
pres-no				
pres-date	prescriber		pat-no	
Hash it on	pres-no			
X-ray request	6	1		
request-no				
investigation	requestor		urgency	reason
pat-no				
Hash it on	request-no			
X-ray sessions	6	4		
date	unit		morning/afternoon	x-ray type
maximum load	numbers booked			
Hash it on	date	unit	morning/afternoon	x-ray type
Staff details	5	1		
emp-no				
name	grade		F.T/P.T	address
Hash it on	emp-no			

Wards		6	1		
Ward-code					
no-of-beds	w-type			w-description	nursing load
nursing dependency					
Hash it on	Ward-code				
Ward bed		4	2		
Ward-code	bed-no				
male/female	free/occupied				
Hash it on	Ward-code				
					bed-no
Nurses		10	1		
emp-no					
N.I.no	name			grade	unit code
entry grade	location code			qualification code	date of birth
full equivalent					
ISAM it on	grade				
Duty		5	4		
date	shift			W-code	grade
number required					
Hash it on	date			shift	W-code
					grade
Ind: workload		4	2		
week commencing	emp-no				
max-hours	hours-booked				
Hash it on	week commencing				emp-no
In-patient		8	4		
pat-no	w-code			bed-no	date entering
discharge date	illness			patient type	comment
Hash it on	pat-no			w-code	bed-no
entering					date
N-absence rec		5	4		
emp-no	day			month	year
reason					
Hash it on	emp-no			day	month
					year



## MAPFILE (Relationships)

Pat-Cons	7				
Hash it on pat-no	2	emp-no	Date registered		
Pat-Disease					
Hash it on pat-no	2	disease-no	Date diagnosed		
X-ray sess/Pat					
Hash it on date type	5	unit suggestion	morning/afternoon	x-ray type	
		unit	morning/afternoon	x-ray	
Pat/O/p-W/l					
Hash it on pat-no	2	W/l-no urgency	position	reference	
Pat/S-W/l					
Hash it on pat-no	2	S-W/l-no non-availability	urgency position	op-code	
Duty-rota					
Hash it on emp-no	5	date	shift	W-code	
Nurses/Wards					
Hash it on emp-no	2	Ward-code			

Output of MIM.P  
MAPFILE (Entities)

25

Patient	12	1		
pat-no				
pat-name				
sex	pat-address	pat-category	reference	
blood group	date-of-birth	marital status	next-of-kin	
	allergy	x-ray information		
Disease	4	1		
disease-no				
disease-name	contageous	treatment		
Consultant	4	1		
emp-no				
con-name	con-address	speciality		
Clinical-session	10	2		
clinic-no	date			
time-of-start	time-of-finish	speciality	est: new pat	
est: repeat pat	new pat booked	rep: pat booked	emp-no	
Clinic-time-schedule	5	3		
clinic-no	date	time		
free/booked	new/repeat			
O/p-W/l	5	1		
W/l-no				
speciality	list size	selection criteria	emp-no	
Surgical W/l	5	1		
S-W/l-no				
speciality	list length	selection criteria	emp-no	
Surgical-session	9	2		
session-no	date			
start time	end time	max: major cases	booked major cases	
max: minor cases	booked minor cases	emp-no		
Admission W/l	4	1		
A-W/l-no				
speciality	list length	selection criteria		

## Appendix E

## Outputs

Blood request	10	1		
Request-no				
reason	date		save serum	reserve blood units
reserve conc: units	doctors no:		blood taker's no:	date of request
pat-no				
Pat-rec-blood	9	1		
ref-no				
ward-code	blood-group		haemoglobin level	pregnancies
past-trans	antibody		prev:/ref-no	pat-no
Antibody cases	9	1		
anti-ref				
blood group	phenotype		antibody	E.D.C
B.T.S confirm	date		comment	pat-no
Drug	5	1		
drug-code				
name	strength		pack-size	expiry-code
supplier	4	1		
supplier-no				
name	address		status	
Prescription	4	1		
pres-no				
pres-date	prescriber		pat-no	
X-ray request	6	1		
request-no				
investigation	requestor		urgency	reason
pat-no				
X-ray sessions	6	4		
date	unit		morning/afternoon	x-ray type
maximum load	numbers booked			
Staff details	5	1		
emp-no				
name	grade		F.T/P.T	address
Wards	6	1		
Ward-code				
no-of-beds	w-type		w-description	nursing load
nursing dependency				

Appendix E

Outputs

Ward bed	4	2	
Ward-code	bed-no		
male/female	free/occupied		
Nurses	10	1	
emp-no			
N.I.no	name	grade	unit code
entry grade	location code	qualification code	date of birth
full equivalent			
Invert it on grade			
Duty	5	4	
date	shift	W-code	grade
number required			
Ind: workload	4	2	
week commencing	emp-no		
max-hours	hours-booked		
In-patient	8	4	
pat-no	w-code	bed-no	date entering
discharge date	illness	patient type	comment
N-absence rec	5	4	
emp-no	day	month	year
reason			

MAPFILE (Relationships)

```

    7
Pat-Cons
    2
pat-no          emp-no          Date registered
Invert it on pat-no
Pat-Disease
    2
pat-no          disease-no      Date diagnosed
Invert it on pat-no
X-ray sess/Pat
    5
date           unit             morning/afternoon  x-ray type
pat-no        suggestion
Invert it on date           unit             morning/afternoon  x-ray
type
Pat/O/p-W/l
    2
pat-no          W/l-no          position          reference
complaints     urgency
Invert it on pat-no
Pat/S-W/l
    2
pat-no          S-W/l-no        urgency           op-code
suggestion     non-availability  position
Invert it on pat-no
Duty-rota
    5
emp-no         date             shift            W-code
grade
Invert it on emp-no
Nurses/Wards
    2
emp-no         Ward-code
Invert it on emp-no

```

APPENDIX F

LOGICAL MODEL DEFINITION FOR INGRES

```

**
** booking (b) - "booking for patient coming in"
**
create booking(
    pat-no           = c7,           patient id
    ward-code        = c3,           ward id
    bedno            = i3,           bed number
    entrydate        = c10,          date entered
    stay             = i2,           expected length of stay in days
    comment          = c80)          comment

```

```

modify booking to isam on pid, wid, bedno, entrydate
save booking until june 21 1986

```

```

**
** pat-outpat (po) - "patient and out-patient waiting list"
**

```

```

create pat-op(
    pat-no           = c7,           patient id
    W/l-no          = c3,           out-patient waiting list id
    entrydate        = c10,          date entered the waiting list
    urgency          = c10,          medical urgency
    operation        = c4,           operation code
    remark           = c30,          consultant's suggestion
    notavail         = c25,          dates patient not available
    source           = c7)           source patient came from

```

```

modify pat-outpat to hash on pat-no
save pat-outpat until june 21 1986

```

```

**
** pat-admin (pad) - "patient and admission waiting list"
**

```

```

create pat-admission(
    pat-no           = c7,           patient id
    A-W/L-no        = c3,           admission waiting list id
    entrydate        = c10,          date entered
    urgency          = c40,          medical urgency
    notavail         = c35,          dates not available for admission
    position         = i2,           position in waiting list
    comment          = c80)          consultant's comment

```

```

modify pat-admin to isam on pat-no,
save pat-admin until june 21 1986

```

```

**
** disease (dis) - "contains details of different medical conditions"
**
create disease(
    disease-no      = c7,          identifying code/number
    name            = c30,         scientific/identifying name
    contag          = c1,          contagious (Y/N)
    treatment       = c30)         description of treatment using codes

```

```

modify disease to hash on id
save disease until june 21 1986

```

```

**
** pat-dis (pd) - "patient and disease"
**

```

```

create pat-dis(
    pid            = c7,          patient id
    did            = c7,          disease id
    date           = c10)         date diagnosed with disease

```

```

modify pat-dis to isam on pid, did, date
save pat-dis until june 21 1986

```

```

**
** patient (p)
**

```

```

create patient(
    pat-no         = c7,          identifying number (e.g. p999999)
    pat-name       = c30,         full name
    pat-address    = c80,         full address & telephone no.
    sex            = c1,          M = male and F = Female
    dob            = c10,         date of birth
    status         = c10,         marital status
    kin            = c30,         full name of next of kin
    blood          = c10,         blood group details
    x-ray          = c80)         x-ray information

```

```

modify patient to hash on pat-no
save patient until june 21 1986

```

```

** ** outpatient-wl (o) - "out-patient waiting list"
**

```

```

create outpatient-wl(
    W/l-no        = c3,          identifying code/number
    speciality    = c20,
    criteria      = c30,         selection criteria for waiting list
    list size     = i2           maximum size for waiting list
    emp-no        = c7)         identifier of consultant

```

```

modify outpatient-wl to hash on W/-no
save outpatient-wl until june 21 1986

```

```

**
** admission-wl (a) - "admission waiting list"
**
create admission(
    A-W/l-no          = c3,          identifying code/number
    speciality        = c20
    length            = i2,          current length of the waiting list
    criteria          = c30)         selection criteria for the waiting list

modify admission-wl to hash on A-W/l-no
save admission-wl until june 21 1986

```

```

**
** pat-surg (psurg) - "patient and surgical waiting list"
**
create pat-surg(
    pat-no           = c7,          patient id
    S-W/l-no        = c3           surgical waiting list id
    urgency          = c40
    position         = i2
    reference        = c7
    complaints       = c40)

modify pat-surg to isam on pat-no
save pat-surg until june 21 1986

```

```

**
** pat-consult (pc) - "patient and consultant"
**
create pat-consult(
    pat-no          = c7,          patient id
    emp-no          = c7           consultant id
    date reg:       = c6)

modify pat-consult to hash on pat-no
save pat-consult until june 21 1986

```

```

**
** pat-prog (pp) - "patient progress"
**
create pat-prog(
    pat-no          = c7,          patient id
    clinic-no       = c3,          clinical session id
    date            = c10,         date of clinical session
    start           = i2,          start time of clinical session
    drugs           = c30,         medicine/drugs being taken
    progress        = c30,
    nexttime        = c10,         date of next appointment
    arrangement     = c20)         travel arrangements for appointment

modify pat-prog to isam on pat-no
save pat-prog until june 21 1986

```



```

**
** surg-wl (surg) - "surgical waiting list"
**
create surg-wl(
    S-W/l-no      = c3,          identifying code/number
    speciality    = c20,
    criteria      = c30,          selection criteria for waiting list
    maxsize      = i2            maximum size for waiting list
    emp-no       = c7 )          identifier of consultant

modify surg-wl to hash on S-W/l-no
save surg-wl until june 21 1986

**
** con-surg (csurg) - "consultant and surgical waiting list"
**
create con-surg(
    emp-no        = c7,          consultant id
    S-W/l-no     = c10)         surgical waiting list id

modify con-surg to isam on emp-no
save con-surg until june 21 1986

**
** consultant (c)
**
create consultant(
    emp-no        = c7,          identifying code/number (e.g. c999999)
    name          = c30,          full name
    address       = c80,          full address & telephone no.
    speciality    = c20)         consultant's medical speciality

modify consultant to hash on emp-no
save consultant until june 21 1986

**
** pat-session (ps) - "patient and surgical session"
**
create pat-session(
    pat-no        = c7,          patient id
    session-no    = c3,          surgical session id
    date          = c10)         date of surgical session

modify pat-session to isam on pat-no
save pat-session until june 21 1986

```

```

**
** session-wl (s) - "surgical session waiting list"
**
create session-wl(
    S-W/l-no      = c3,          identifying number
    date          = c10,         date of surgical session
    start         = i2,          start time of surgical session
    finish        = i2,          finish time of surgical session
    maxmajor      = i2,          maximum major cases allowed
    maxminor      = i2,          maximum minor cases allowed
    majorbooked   = i2,          number of major cases booked so far
    minorbooked   = i2)         number of minor cases booked so far

modify session-wl to isam on S-/l-no
save session-wl until june 21 1986

**
** con-session (cs) - "consultant and surgical session"
**
create con-session(
    emp-no        = c3,          consultant id
    session-no    = c3,          surgical session id
    date          = c10)         date of surgical session

modify con-session to isam on emp-no
save con-session until june 21 1986

**
** clinical (clin) - "clinical session"
**
create clinical(
    clinic-no     = c3,          identifying number
    date          = c10,         date of clinic
    cid           = c7,          consultant id
    start         = i2,          start time of clinic
    finish        = i2,          finishing time of clinic
    doctor        = c7,          id of doctor in charge
    speciality    = c20,         clinic's speciality
    maxnew        = i2,          maximum new patients allowed
    maxrep        = i2,          maximum repeat patients allowed
    newbooked     = i2,          number of new patients booked so far
    repbooked     = i2)         number of repeat patients booked so far

modify clinical to hash on clinic-no,date
save clinical until june 21 1986

**
** schedule (sch) - "clinical session's time schedule"
**
create schedule(
    clinic-no     = c3,          clinical session's id
    date          = c10,         date of clinic
    time          = i2,          time of patient's appointment
    status        = c1,          free or booked
    type          = c3)         "new" or "old" (i.e. repeat)

modify schedule to hash on clinic-no, date, time
save schedule until june 21 1986

```

```

create anticoag
    ( id = c3,
      pat-no c7,
      did = c7,
      reason = c15,
      urgency = c10 )
      anticoagulant #
      patient id.
      doctor sending
      reason for sending
      urgency
modify anticoag to hash on id
save anticoag until june 21 1986

**      bloodreq (br) - requests for blood
**
create bloodreq
    ( reqno = c5,
      pat-no= c7,
      reason = c15,
      timereq = i2,
      datereq = c10,
      serum = c1,
      rwbu = c1,
      rccu = c1,
      date = c10,
      taker = c30 )
      request #
      patient id.
      reason for request
      time required
      date required
      saved serum or not
      reserve whole blood units
      reserve concentrated cell units
      date
      blood taker
modify bloodreq to hash on reqno
save bloodreq until june 21 1986

**      patreceive (pr) - patient receiving blood
**
create patreceive
    ( patrefno = c5,
      pat-no = c7,
      ward = c3,
      haem = c5,
      preg = c1,
      pasttran = c1,
      antibody = c10,
      previous = c5,
      quantity = c5,
      date = c10 )
      patient reference #
      patient id.
      ward patient in
      patient's current haemoglobin
      pregnancies or not
      past transfusion or not
      known antibody in blood
      last previous reference #
      quantity
      date
modify patreceive to hash on patrefno
save patreceive until june 21 1986

**      proteinpat (prp) - protein patint record
**
create proteinpat
    ( labcode = c6,
      date = c10,
      pid = c7,
      testcode = c5,
      result = c10 )
      lab code
      date
      patient id.
      test code
      result
modify proteinpat to isam on labcode,date,pid
save proteinpat until june 21 1986

```

```

**      proteinday (prd) - protein day record
**
create proteinday
      ( labcode = c6,                lab code
        date = c10,                 date
        notest = i2 )               number of patients tested
modify proteinday to isam on labcode,date
save proteinday until june 21 1986

**      antibodyprob (ap) - antibody problem cases
**
create antibodyprob
      ( id = c6,                    anti-ref
        pat-no = c7,                patient id.
        phenotype = c2,              phenotype
        antibody = c10,              antibody
        edc = c1,                   edc
        bts = c1,                   bts confirmation
        date = c10,                 date
        comment = c15 )              comment
modify antibodyprob to hash on id
save antibodyprob until june 21 1986

**      proteinspec (prs) - protein special cases
**
create proteinspec
      ( prospref = c6,               prospref
        pid = c7,                   patient id.
        diag = c10,                  diagnosis
        treat = c20,                 treatment
        qereport = c15 )             textual report from QE
modify proteinspec to hash on prospref
save proteinspec until june 21 1986

**      proteinint (pri) - protein interesting cases
**
create proteinint
      ( prointref = c6,              prointref
        pid = c7,                   patient id.
        diag = c10,                  diagnosis
        treat = c20 )                treatment
modify proteinint to hash on prointref
save proteinint until june 21 1986

```

```

**      coagpat (cp)      - coag clinic patient
**
create coagpat
      ( coagref = c6,                coag-ref
        pid = c7,                   patient id.
        diag = c15,                 diagnosis
        regeme = c15,              drug regeme
        commther = c10,            commence therapy
        nextclinic = c10,          next clinic date
        comment = c20 )            comment
modify coagpat to hash on coagref
save coagpat until june 21 1986

**      coagpatprog (cpp) - coag clinic patient progress
**
create coagpatprog
      ( coagref = c6,                coag-ref
        date = c10,                 date
        pid = c7,                   patient id.
        result = c15,               result
        dose = c10 )                dose
modify coagpatprog to isam on coagref,date
save coagpatprog until june 21 1986

**      coagclinic (cc) - coag clinic
**
create coagclinic
      ( date = c10,                 date
        time = i2,                  time
        maxpat = i2,                max. number of patients in list
        nopat = i2 )                number of patients in list
modify coagclinic to isam on date,time
save coagclinic until june 21 1986

**      spectestreq (spt) - specimen test requests
**
create spectestreq
      ( reqno = i2,                 request #
        reqcode = c5,               request code
        pid = c7,                   patient id.
        date = c10,                 date of request
        source = c10,               source of request
        status = c6 )               status
modify spectestreq to hash on reqno
save spectestreq until june 21 1986

```

\*\* specimen (sp) - specimen  
\*\*

create specimen

( specno = i2,	specimen #
spectype = c10,	specimen type
reqno = i2,	request #
invest = c1,	investigation required
urgency = c10,	urgency
testcode = c5,	test code
labcode = c6,	lab code
result = c10,	result
comment = c20 )	comment

modify specimen to hash on specno  
save specimen until june 21 1986

\*\*\*\* drug - Not abbreviated

\*\*\*\*

\*\*\*\* Information relating to any single drug

create drug (

drug-code = c7,	Drug identification number
name = c40,	Name of drug
descr = c80,	Description of Drug, including use
strength = c10,	Concentration of drug, eg 40mg/litre
packsize = i4,	Number in packs held in stock
expiry-code = c10 )	Roughly best before codes

modify drug to hash on drug-code  
save drug until june 21 1986

\*\*\*\* prescr-dets - Abbreviated to pred

\*\*\*\*

\*\*\*\* Details of individual drugs on any single prescription

create prescr-dets (

prescr-no = c7,	These details relate to prescription
drugid = c7,	Drug identification number for this
qty = i2,	Quantity prescribed
dose = c10,	Number to be taken per time period
duration = c10 )	Time over which drug must be taken,

modify prescr-dets to isam on prescr-no  
save prescr-dets until june 21 1986

```

****          prescrip - Abbreviated to presd
****
****          Fields in a prescription that occur ONCE per prescription

```

```

create prescrip (
    prescr-no      = c7,          Prescription number
    date           = c10,         Date prescribed
    prescrib-id    = c7,          Prescribers identification number
    patient-id     = c7 )         Patients identification number

```

```

modify prescrip to hash on prescr-no
save prescrip until june 21 1986

```

```

****          ward-order - Abbreviated to wo
****
****          Details of a bulk order for a whole ward

```

```

create ward-order (
    id              = c7,          Ward order identification number
    ward-id         = c7,          Ward identification number
    date-req        = c10 )       Date required at ward for distribution

```

```

modify ward-order to isam on id, ward-id
save ward-order until june 21 1986

```

```

****          wo-details - Abbreviated to wod
****
****          Item details for a ward order

```

```

create wo-details (
    woid           = c7,          Ward order identification number
    drug-id        = c7,          Drug identification number
    qty            = i4 )         Quantity required by whole ward

```

```

modify wo-details to isam on id, drug-id
save wo-details until june 21 1986

```

```

****          supplier - Abbreviated to sup
****
****          Supplier for various types of drugs

```

```

create supplier (
    supplier-no    = c7,          Supplier identification number
    name           = c30,         Name and address of supplier
    address        = c80,
    status         = c8 )

```

```

modify supplier to hash on supplier-no
save supplier until june 21 1986

```

```

****          purch-dets - Abbreviated to purd
****
****          Details of purchases from a supplier, itemised in purch-items

```

```

create purch-dets (
    id          = c7,          Purchase details identifier, used in
                                purch-items
    supplier    = c7,          Supplier code who supplied the items
    order-no    = c10 )       Hospitals order number placed on
                                supplier

```

```

modify purch-dets to isam on id
save purch-dets until june 21 1986

```

```

****          purch-items - Abbreviated to puri
****
****          Item list for a purchase detail (above)

```

```

create purch-items (
    pdid        = c7,          Purchase-Detail identification number
    drugid      = c7,          Drug identification number purchased
    qty-ordered = i4,          -
    date-ordered = c10,       -
    qty-recd    = i4,          -
    date-recd   = c10 )       -

```

```

modify purch-items to isam on pdid, drugid
save purch-items until june 21 1986

```

```

****          stock-rec - Abbreviated to srec
****
****          Stock record, current holdings of any drug ; also older versions
: i.e. stocks on previous days

```

```

create stock-rec (
    drugid      = c7,          Drug identification number
    date        = c10         Date relating to this stock record
    balance     = i4,          Closing balance on this drug for today
    stock-recd  = i4,          Stock received today
    stock-issued = i4 )       Stock issued (Prescriptions/ward orders)
today

```

```

modify stock-rec to isam on stock-code

```

```

**
** ward (w) - ward details
**

```

```

create ward(
    ward-code    = c3,          ward code
    type         = c10,         ward type
    description   = c10,
    nobeds       = i3,          number of beds
    nursesreq    = i1,          number of nurses required
    dependency    = i1)         nursing dependency

```

```

modify ward to hash on ward-code
save ward until june 21 1986

```



```

**
** ward-bed (wb) - details for each bed in ward
**
create ward-bed(
    ward-code      = c3,          ward code
    bedno          = i3,          bed number
    sex            = c1,          sex
    occupied       = c1)         free or occupied

```

```

modify ward-bed to hash on ward-code, bedno
save ward-bed until june 21 1986

```

```

**
** entry (e) - inpatient entry record ~ uniquely identifies a inpatient
**
create entry(
    pat-no         = c7,          patient identifier
    ward-code      = c3,          ward code
    bedno          = i3,          bed number
    entrydate      = c10,        date entering ward
    id             = c7)         patient entry identifier

```

```

modify entry to hash on pid, wid, bedno, entrydate
save entry until june 21 1986

```

```

**
** inpatient (i) - inpatient details
**
create inpatient(
    pat-no         = c7
    w-code         = c3
    bedno          = i3
    date ent:      = c10
    expdischarge   = c10,        date expected for discharge
    illness        = c20,        type of illness
    activities     = c20,        daily activities
    type           = c1,          type of patient(P=private/S=state)
    comment        = c30)        comment

```

```

modify inpatient to hash on pat-no,w-code,bedno,date ent:
save pat-bed until june 21 1986

```

```

**
** consult-bed (cb) - consultant attending a particular bed
**
create consult-bed(
    wid            = c3,          ward code
    bedno         = i3,          bed number
    cid           = c7)         consultant identifier

```

```

modify consult-bed to isam on wid, bedno, cid
save consult-bed until june 21 1986

```

```

**
** drug-pat (dp) - details of drug being given to patient
**

```

```

create drug-pat(
    eid           = c7,           patient entry identifier
    drugid        = c5,           drug code
    root          = c10,         root of drug
    dose          = c10,         dose
    ntimes       = i1,           number of times
    prescriber    = c7)          prescriber

```

```

modify drug-pat to isam on eid, drugid
save drug-pat until june 21 1986

```

```

**
** pat-med (pm) - details of when medication has been given to patient
**

```

```

create pat-med(
    eid           = c7,           patient entry identifier
    drugid        = c5,           drug code
    date         = c10,         date
    time         = i2,           time
    givenby      = c7,           given by(e.g. nurse code)
    comment      = c30)          comment

```

```

modify pat-med to isam on eid, drugid, date, time
save pat-med until june 21 1986

```

```

**
** pat-report (pr) - report entry of patients condition (temperature, blood
pressure)
**

```

```

create pat-report(
    eid           = c7,           patient entry identifier
    date         = c10,         date
    time         = i2,           time
    temp         = i1,           temperature
    pressure     = i1,           blood pressure
    seenby      = c7,           seen by (doctor's code)
    comment      = c30)          comment

```

```

modify pat-report to hash on eid, date, time
save pat-report until june 21 1986

```

```

**
** disch-pat (dp) - details relating to a discharged patient
**

```

```

create disch-pat(
    eid           = c7,           patient entry identifier
    datedisch    = c10,         date discharged
    condition     = c20,         condition when discharged
    treatment     = c20,         back-up treatment
    homecare     = c1,           home care required (Y/N)
    presdrug     = c1)          prescribed drug

```

```

modify disch-pat to isam on eid, datedisch
save disch-pat until june 21 1986

```

```

**
** nurse (n) - record of nurse details
**
create nurse(
    id             = c7,           nurse identifier
    natno          = c9,           national insurance number
    name           = c30,         name
    grade          = c3,           grade
    gradedate      = c10,         entry to grade
    location       = c3,           location code
    qual           = c15,         qualification code
    dob            = c10,         date of birth
    wholetime      = f1)          whole time equivalent

```

```

modify nurse to isam on grade
save nurse until june 21 1986

```

```

**
** absence (a) - details for a day when a nurse is absent
**
create absence(
    id             = c7,           nurse identifier
    date           = c10,         date absent
    reason         = c20)         reason for absence

```

```

modify absence to hash on id, date
save absence until june 21 1986

```

```

**
** duty (d) - duty details
**
create duty(
    id             = c3,           duty code
    date           = c10,         date
    shift          = i1,          shift
    wid            = c3,           ward code
    grade          = c3,           grade required
    noreq          = i1)          number of nurses required

```

```

modify duty to hash on date, shift, wid, grade
save duty until june 21 1986

```

```

**
** rota (r) - duty performed by nurse
**
create rota(
    nid           = c7,           nurse identifier
    dutyid        = c3)          duty code

```

```

modify rota to hash on nid
save rota until june 21 1986

```

\*\*

\*\* load (l) - record of work done by a nurse for a particular week

\*\*

```

create load(
    commence      = c10,      week commencing
    nid            = c7,      nurse identifier
    totalhrs      = i1,      total hours for week
    bookedhrs     = i1)      number of hours booked

```

```

modify load to isam on commence, nid
save load until june 21 1986

```

\*\*

\*\* ward-nurse (wn) - ward which nurse works on

\*\*

```

create ward-nurse(
    wid           = c3,      ward code
    nid           = c7)      nurse identifier

```

```

modify ward-nurse to hash on wid, nid
save ward-nurse until june 21 1986

```

\*\*

\*\* holiday (h) - holiday details for a nurse

\*\*

```

create holiday(
    nid           = c7,      nurse identifier
    year          = i2,      year commencing
    maxhols       = i1,      maximum holiday days
    daystaken     = i1)      number of days taken

```

```

modify holiday to isam on nid, year
save holiday until june 21 1986

```

\*\*

\*\* nurseleft (nl) - details about a nurse which has left the hospital

\*\*

```

create nurseleft(
    natno         = c7,      national insurance number
    name          = c30,     name
    address       = c80,     last known address
    dateleft      = c10,     date left
    qualleft      = c15,     qualification when left
    gradeleft     = c3,      grade when left
    reason        = c15,     reason for leaving
    comment       = c30)     comment

```

```

modify nurseleft to isam on natno
save nurseleft until june 21 1986

```