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**

<table>
<thead>
<tr>
<th>PROGRAM LISTINGS</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTEST.P</td>
<td>1</td>
</tr>
<tr>
<td>RELTEST.P</td>
<td>6</td>
</tr>
<tr>
<td>FUNCTEST.P</td>
<td>16</td>
</tr>
<tr>
<td>ANALYSIS.P</td>
<td>29</td>
</tr>
<tr>
<td>RELMAP.P</td>
<td>40</td>
</tr>
<tr>
<td>CODMAP.P</td>
<td>52</td>
</tr>
<tr>
<td>LABEL.P</td>
<td>64</td>
</tr>
<tr>
<td>ING.P</td>
<td>79</td>
</tr>
<tr>
<td>MIM.P</td>
<td>92</td>
</tr>
</tbody>
</table>

**Appendix D**

<table>
<thead>
<tr>
<th>TYPICAL RUNS OF PROGRAMS</th>
<th>105</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Typical Run of the Program ENTEST.P</td>
<td>105</td>
</tr>
<tr>
<td>A Typical Run of the Program RELTEST.P</td>
<td>107</td>
</tr>
<tr>
<td>A Typical Run of the Program FUNCTEST.P</td>
<td>109</td>
</tr>
</tbody>
</table>

**Appendix E**

<table>
<thead>
<tr>
<th>OUTPUTS FROM PROGRAMS</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output of ENTEST.P</td>
<td>111</td>
</tr>
<tr>
<td>Output of RELTEST.P</td>
<td>116</td>
</tr>
<tr>
<td>Output of FUNCTEST.P</td>
<td>118</td>
</tr>
<tr>
<td>Output of ANALYSIS.P</td>
<td>121</td>
</tr>
<tr>
<td>Output of RELMAP.P</td>
<td>129</td>
</tr>
<tr>
<td>Output of CODMAP.P</td>
<td>132</td>
</tr>
<tr>
<td>Output of LABEL.P</td>
<td>140</td>
</tr>
<tr>
<td>Output of ING.P</td>
<td>147</td>
</tr>
<tr>
<td>Output of MIM.P</td>
<td>151</td>
</tr>
</tbody>
</table>

**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;
attri = 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}
nooentities: 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;

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;
APPENDIX C

Program Listings

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;

  {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;
APPENDIX C

(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);
    end;
  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.
RELTEST.P

{THIS PROGRAM DOCUMENTS THE RELATIONSHIPS IDENTIFIED BY THE DESIGNER}

program relationtest (input, output, relfile);

const maxstrlength = 20;
  maxnrel = 40;
  maxnoent = 40;
  maxnoatt = 20;
  maxrelatt = 10;
  avith = 'WITH ';
  awithout = 'WITHOUT ';

type str = array [1..maxstrlength] of char;
relstat = (weth, without);  {DENOTES IF RELATIONSHIP HAS ANY ATTRIBUTE}
relation = record
  rname: str;  {NAME OF THE RELATIONSHIP}
  entityya: integer;  {POINTER TO ENTITYYA }
  degenta: char;  {DEGREE OF ENTITYYA}
  membshpa: char;  {MEMBERSHIP CLASS OF ENTITYYA}
  entityyb: integer;  {POINTER TO ENTITYYB}
  degentb: char;  {DEGREE OF ENTITYYB}
  membshpb: char;  {MEMBERSHIP CLASS OF ENTITYYB}
case rs : relstat of
  weth: (nrelatt : integer;  {NUMBER OF ATTRIBUTES IF ANY}
    relatt : array [1..maxrelatt] of str);  {ATTRIBUTES}
  without: ()
end;

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

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

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

6
procedure readstr (var f: text; var s: str);
    var ptr: integer;
    begin
        ptr := 0;
        while not eoln (f) and (ptr < maxstrlen) do
            begin
                ptr := ptr + 1;
                read (f, s[ptr]);
            end;
        while ptr < maxstrlen 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 maxstrlen do
            write (f, s[i]);
    end:

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

Program Listings

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 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 := wethout;
      readln (relfile);
APPENDIX C

Program Listings

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

[READ IN FROM THE TERMINAL]
write ('Number of new relationships');
readln (norelfile);  [NOTE THE NUMBER OF NEW RELATIONSHIPS TO BE INSERTED]
for i := 1 to norelfile do
  begin
    nooofrelations := nooofrelations + 1;  [INCREASE RELATIONSHIP TOTAL]
    with relchart[nooofrelations] 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 := wethout;
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 attribute per line');
for j := 1 to norelatt do
begin
readstr (input, relatt[j]);
readln
end
end
APPENDIX C

{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);
      writeln (relfile, i.name);
      writeln (relfile);
      writeln (relfile, entchart[entitya].ename);
      writeln (relfile, degenta);
      writeln (relfile, membshpa);
      writeln (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
              writeln (relfile, relatt[j]);
              k := k+1;
              if k> 4 then
                begin
                  writeln(relfile);
                  k := 1
                end
            end
        end
    end
end;
APPENDIX C

{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.

12
C

Write number of relationships

For each relationship do

Write name of relationship

Write entity a
its degree and membership

Write entity b
its degree and membership

Test if relationships
have attributes

WITH

Write number of attributes and write the attributes

WITHOUT

No action
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 = (veth, without);
  seloolds = (sbyr, sbya, sbyp);
  selectdet = record
    case opts : seloolds 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;
APPENDIX C

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;
refile: text;
i, j, k, l: integer;
noofnentfunc: integer;
nofentities: integer;
nofrelations: 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;

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;

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;
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 = avith 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;
APPENDIX C

[READ IN THE FUNCTIONS FROM THE FILE CONTAINING THE LIST OF FUNCTIONS]

reset (funcfile, 'hospfunc');
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
            esselectcrit.opts := sbyr
          else
            if equalstr (tempbuff, selbyatt) then
              esselectcrit.opts := sbya
            else
              esselectcrit.opts := sbyp;
          if esselectcrit.opts = sbyr then
            begin
              readstr (funcfile, tempbuff);
              readln (funcfile);
              findrel (esselectcrit.relpnt)
            end
          else
            if esselectcrit.opts = sbya then
              begin
                readstr (funcfile, tempbuff);
                findatt (esselectcrit.apnt, entname);
                readstr (funcfile, tempbuff);
                if equalstr (tempbuff, range) then
                  esselectcrit.aclas := 'r'
                else
                  esselectcrit.aclas := 'e';
                readln(funcfile)
              end

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

end

{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;

23
[WRITE BACK ALL THE FUNCTION DESCRIPTIONS IN THE FUNCFILE];

rewrite(funcfile, 'hospfunc');
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
writeln (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
writeln (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;
Obtain number of new functions to be inserted

For each function * do

Read name, status, frequency and number of entities accessed

For each entities * accessed do

Identify entity

Check selection of clause

Obtain name of entity

Check entity exists

Selected by relationship

Selected by attribute

Selected by key

Identify relationship

Note details of selection

No action

Identify attribute

Note selection clause

Obtain name of relationship

Check relationship exists

Obtain name of attribute

Check attribute exists
B

Write total number of functions

For each Function do

Write name, status, frequency and number of entities accessed

For each entity accessed do

Write name of entity

Check selection option

Selected by relationship

Write name of relationship

Selected by attribute

Write name of attribute and selection clause

Selected by key

Write keys
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';
avith = '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;

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..maxnoent] 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 < maxstrlen) do
      begin
        ptr := ptr + 1;
        read (f, s[ptr])
      end;
    while ptr < maxstrlen 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 maxstrlen 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 < maxstrlen ) 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;
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;
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;
      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 = veth 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;

{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
  funcsstatus := 1
else funcsstatus := 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
  eeselectcrit.opts := sbyr
else if equalstr(tempbuff, selbyatt) then
  eeselectcrit.opts := sbya
else
  eeselectcrit.opts := sbyp;
if eeselectcrit.opts = sbyr then
begin
readstr (funcfile, tempbuff);
readln (funcfile);
findrel (eeselectcrit.relpnt)
end
else
if eeselectcrit.opts = sbya then
begin
readstr (funcfile, tempbuff);
findatt (eeselectcrit.apnt, entname);
readstr (funcfile, tempbuff);
if equalstr (tempbuff, range) then
  eeselectcrit.aclaus := 'r'
else
else
eeselectcrit.aclaus := 'e';
readln (funcfile)
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  \{FOR EACH FUNCTION PERFORM THE FOLLOWING\}
with funcchart [i] do
begin
for j := 1 to noacc do  \{FOR EACH ENTITY ACCESSED\}
with entarr[j] do
begin
if eselectedopts = sbya then  \{IF SELECTED BY AN ATTRIBUTE\}
begin
if eselectedaclaus = 'r' then  \{IF THE SELECTION CLAUSE IS RANGE\}
begin

{UPDATE NECESSARY RANGE FREQUENCY}
if funcstatus = 1 then
eusagemat[entname].usagedet[eselectedcrit.apnt].primdet.rangefreq :=
eusagemat[entname].usagedet[eselectedcrit.apnt].primdet.rangefreq +
funcfreq
else
eusagemat[entname].usagedet[eselectedcrit.apnt].seconddet.rangefreq :=
eusagemat[entname].usagedet[eselectedcrit.apnt].seconddet.rangefreq +
funcfreq
end
else
{IF SELECTION CLAUSE IS EQUIJOIN THEN UPDATE NECESSARY EQUIJOIN FREQUENCY}
begin
if funcstatus = 1 then
eusagemat[entname].usagedet[eselectedcrit.apnt].primdet.equifreq :=
eusagemat[entname].usagedet[eselectedcrit.apnt].primdet.equifreq +
funcfreq
else
eusagemat[entname].usagedet[eselectedcrit.apnt].seconddet.equifreq :=
eusagemat[entname].usagedet[eselectedcrit.apnt].seconddet.equifreq +
funcfreq
end
end
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.equipfreq :=
            eusagemat[entname].usagedet[k].primdet.equipfreq + funcfreq
            end
          end
      else
        begin
          for k := 1 to entchart[entname].keycount do
            begin
              eusagemat[entname].usagedet[k].seconddet.equipfreq :=
              eusagemat[entname].usagedet[k].seconddet.equipfreq + funcfreq
              end
            end
        end
    else
      begin

        [UPDATE THE RELATIONSHIP USAGE FREQUENCY]

        begin
          if funcstatus = 1 then
            rusagemat[eselectcrit.relpnt].primfreq :=
            rusagemat[eselectcrit.relpnt].primfreq + funcfreq
          else
            rusagemat[eselectcrit.relpnt].secfreq :=
            rusagemat[eselectcrit.relpnt].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, 'Primary functions ');
            write(analysisfile, 'Secondary functions ',);
            writeln (analysisfile);
            write(analysisfile, 'Ffrequency ');
            write(analysisfile, 'Effrequency ');
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');
write (analysisfile);
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.

37
ANAYSIS.P

Read Entfile
Read Relfile
Read Funcfile
Analyse Functions
Store result

S

Read number of functions
For each function do

Read number of entities accessed
For each entity do

Update necessary 0 frequencies

Accessed by attribute

Update attribute 0 usage frequency

Accessed by key

Update key usage 0 frequency

Accessed by relationship

Update 0 relationship usage frequency

range clause
Update 0 range frequency
second: func:
Update range second: frequency

equijoin clause
Update 0 equijoin frequency
prim: func:
second: func:
Update equijoin second: frequency
prim: func:

prim:=primary second:=secondary func:= function

X

Y
RELMAP.P

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

program relmap (input, output, relfile);

const maxstrlength = 20;
  maxnorel = 40;
  maxnoent = 40;
  maxnoatt = 20;
  maxrelatt = 20;
  maxnokey = 5;
  maxnoidnt = 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 : (norealatt : 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;

{nornment 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;
APPENDIX C

(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;
nooofrelations : integer;
refile : text;
i, j, k, l : integer;
norefile : integer;
nooofentities : integer;
entfile : text;
mapfile : text;
tempbuff : str;
found : boolean;
correct : boolean;
answer : str;
nooofnormrel : 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 < maxstrlen) 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 < noo) do
        begin
            l := l + 1;
            if equalstr (tempbuff, entchart[l].ename) then
                begin
                    int := l;
                    found := true
                end
        end;
begin
    [READ IN THE ENTITY FILE]
reset (entfile, ‘hospent’);
readln (entfile, noo);
for i := 1 to noo 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;
        end;
readln (entfile);
readln (entfile)
(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;
APPENDIX C

(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 norrelcht[r][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
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
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
                norelatt[k] := k
            end
    end
end

noofnormrel := j;

rewrite (mapfile, 'norhosprel');
writeln (mapfile, noofnormrel);
for i := 1 to noofnormrel do
    with norrelchtr[i] do
        begin
            writeln (mapfile, relchart[nrname].rname);
            writeln (mapfile);
            writeln (mapfile, identcnt);
            k := 1;
            for j := 1 to identcnt do
                with norrelidnt[j] do
                    begin
                        writeln (mapfile, entchart[entptr].entatt[attptr]);
                        k := k + 1;
                        if k > 4 then
                            begin
                                writeln (mapfile);
                                k := 1
                            end;
                    end;
            end;
end;
for j := 1 to nrkeycnt do
  with norrelkey[j] do
    begin
      writeln(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
            writeln(mapfile, relatt[nrelatt[j]]);
            k := k + 1;
            if k > 4 then
              begin
                writeln(mapfile);
                k := 1
              end
          end;
      writeln(mapfile)
    end;
{MAP ALL THE ENTITIES}

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

end;
APPENDIX C

{WRITE TO THE MAPFILE}
rewrite (mapfile, 'hospmap');
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
          begin
            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.
APPENDIX C

C

Read number of entities

For each entity do *

Create a relation
(Condition = the entity is the entityb of a 1:1 or 1:n relationship and the membership of entity is obligatory)

Primary key = identifier of entity

Form attributes 0

Condition true

Attributes = attributes of entity + foreign key

Condition false

Attributes = attributes of entity

D

Write the total number of relations representing the entities in mapfile

For each relation *

do

Write name of relations

Write Primary Key

Write attributes
program codmap (input, output, refile);

const maxstrlength = 20;
  maxnrel = 40;
  maxmodrel = 50;
  maxnoent = 40;
  maxnoatt = 20;
  maxrelatt = 10;
  maxnokey = 5;
  maxnoidnt = 5;
  avith = '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 = array [0..maxnoatt] of 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..maxnoret] of relation;
entchart: array [1..maxnoent] of entity;
modrelchart: array [1..maxmodrel] of modreln;
noofofrelations: integer;
relfile: text;
codent: text;
codrel: text;
i, j, k, l: integer;
norelfile: integer;
noofofentities: integer;
noofofmodrel: integer;
entfile: text;
mapfile: text;
modrelfile: text;
modent: text;
tempbuff: str;
found: boolean;
correct: boolean;
answer: str;
nooofnormrel: 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;

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 < maxstrlen) 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;

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;
Appendix C

(READ IN THE ENTITY FILE)

begin
reset (entfile, 'hospent');
readln (entfile, noofoentities);
for i := 1 to noofoentities 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, nooforelations);
for i := 1 to nooforelations 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 = a with then rs := weth
      else rs := without;
      readln (relfile);
      if rs = weth then
        begin
          readln (relfile, norelatt);
          k := 1;
        end;
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;

{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
  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;
Appendix C

Program Listings

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

{TWO ADDITIONAL RELATIONSHIPS NEED TO BE CREATED}

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

57
if rs = weth then
begin
noatts := noatts + norelatt;
i := 0;
for l := (keycount + 1) to noatts do
begin
i := i+1;
entatt[l] := relat[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';
mmembsp := membspa;
mentityb := noofentities;
mdegentb := 'n';
membshpb := 'o'
end;
k := k + 1;
formstring(tempstr, entchart[entityb].ename, rname);
with modrelchart [k] do
begin
mrtype := 'C';
mrname := tempstr;
mentitya := entityb;
mdegenta := '1';
mmembsp := membshpa;
mentityb := noofentities;
mdegentb := 'n';
membshpb := 'o'
end
end
end;
end;
noofmodrel := k;

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

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

Program Listings

\[ k := 1; \]
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 MODRELFILe)

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

{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]);
    writeln (codent, '; ')
   end;
  writeln (codent);
  writeln (codent)
  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, mrtype);
  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 (mmbshpb = 'o') and (mrtype = 'o') then
   write(codrel, ' AUTOMATIC MANDATORY')
  else
   if (mmbshpb = 's') and (mrtype = 'o') then
    write (codrel, 'AUTOMATIC FIXED')
  else
   if (mmbshpb = 'o') and (mrtype = 'c') then
    write (codrel, 'AUTOMATIC FIXED')
   else
    write (codrel, 'MANUAL OPTIONAL');
  writeln(codrel);
  writeln(codrel)
  end.

60
Create new entity

Create new relationship between new entity and two original entities

Name of entity = Name of relationship
Key of entity = Key of relationship
Attribute of entity = Attribute of relationship

Form records
Store records

Read number of modified entities
For each modified entity form record

Name = name of entity
Key = primary key of entity
Data item = attribute of entity
B

Form sets

Store sets

Read number of modified relationships

For each modified relationship form a set

Set name = relationship name

Owner record is entitya

Member record is entityb

Define insertion/ 0 retention

C1

AUTOMATIC MANDATORY

C2

AUTOMATIC FIXED

C3

AUTOMATIC FIXED

C4

MANUAL OPTIONAL

C1 = membership of entityb is obligatory + type of relationship is original
C2 = primary key of entityb is extension of the primary key of entitya + type of relationship is original
C3 = membership of entityb is obligatory + type of relationship is modified
C4 = any other condition
[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'
sebbyrel = 'SELECT.BY.RELATION'
sebbyatt = 'SELECT.BY.ATTRIBUTES'
sebbykey = '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;
    caser s: relstat of
           weth: (norelatt: integer;
               relatt : array [1..maxrelatt] of str);
       without : ()
    end;

accdet = record
    rangefreq: integer;
    equifreq : 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}
ooofinedge : 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;
ilabelv : integer;
olabelv : integer;
noofswap : integer;
tempfreq : integer;
temppnt : 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;
Appendix C

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;
Appendix C

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].nrelatt ) 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, noofoentities);
for i := 1 to noofoentities do
  with entchart [i] do
    begin
      readln (entfile);
      readstr (entfile, ename);
      readln (entfile);
      readln (entfile, keycount);
      readln (entfile, noaatts);
      readln (entfile);
      for l := 0 to keycount do readstr (entfile, entatt[l]);
      readln (entfile);
      readln (entfile);
      k := 1;
      for j := (keycount + 1) to noaatts 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 (refile, 'hospre');
readln (refile, nooforelations);
for i := 1 to nooforelations do
  with relchart[i] do
    begin
      readln (refile);
      readstr (refile, rname);
      readln (refile);
      readstr (refile, tempbuff);
      findent (entitya);
      readln (refile, degenta);
      readln (refile, membshpa);
      readstr (refile, tempbuff);
      findent (entityb);
      readln (refile, degentb);
      readln (refile, membshpb);
      readstr (refile, tempbuff);
      if tempbuff = awith then rs := weth
      else rs := without;
      readln (refile);
      if rs = weth then
        begin
          readln (refile, norelatt);
          k := 1;
          for j := 1 to norelatt do
begin
readstr (relfile, relatt[j]);
k := k + 1;
if k > 4 then
begin
  begin
    readln (relfile);
k := 1
  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.aclaus := 'r'
                  else
                    eselectcrit.aclaus := 'e';
                  readln (funcfile)
                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 eeselectcrit.opts = sbya then
begin
if eeselectcrit.aclaus = 'r' then
begin
if funcstatus = 1 then
eusagemat[entname].usagedet[eeselectcrit.apnt].primdet.rangefreq:=
eusagemat[entname].usagedet[eeselectcrit.apnt].primdet.rangefreq +
funcfreq
else
eusagemat[entname].usagedet[eeselectcrit.apnt].seconddet.rangefreq:=
eusagemat[entname].usagedet[eeselectcrit.apnt].seconddet.rangefreq +
funcfreq
end
else
begin
if funcstatus = 1 then
eusagemat[entname].usagedet[eeselectcrit.apnt].primdet.equipfreq:=
eusagemat[entname].usagedet[eeselectcrit.apnt].primdet.equipfreq +
funcfreq
else
eusagemat[entname].usagedet[eeselectcrit.apnt].seconddet.equipfreq :=
eusagemat[entname].usagedet[eeselectcrit.apnt].seconddet.equipfreq +
funcfreq
end
end
end

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

71
else
    begin
    for k := 1 to entchart[entname].keycount do
        begin
            eusagemat[entname].usagedet[k].seconddet.equipreq :=
            eusagemat[entname].usagedet[k].seconddet.equipfreq + funcfreq
        end
    end
else
    begin
    if funcstatus = 1 then
        rusagemat[eeselectcrit.relport].primfreq :=
        rusagemat[eeselectcrit.relport].primfreq + funcfreq
    else
        rusagemat[eeselectcrit.relport].secfreq :=
        rusagemat[eeselectcrit.relport].secfreq + funcfreq
    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.equipfreq;
            outedge[noofoutedge].oedgelabel := 'I';

            [FILL IN THE DETAILS FOR THE REST OF THE EDGES]

            noofedges := noofedges + 1;
            with edgemat[noofedges] do
                begin
                    edgelab := 'I';
                    edgfreq := 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.equipfreq;
            outedge[noofoutedge].oedgelabel := 'I';
            noofedges := noofedges + 1;
with edgemat[noofedges] do
begin
  edgelab := 'I';
  edgfreq := outedge[noofoutedge].oedgedtype := 'r';
  edgelabel := 'I';
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].edgedtype := k;
    outedge[noofoutedge].oedgedtype := rusagemat[k].primfreq;
    outedge[noofoutedge].edgelabel := 'I';
    edgemat[noofedges].edgpnt3 := noofoutedge
end;

with labelmat[entityb] do
begin
  noofinedge := noofinedge + 1;
  inedge[noofinedge].edgedtype := k;
  inedge[noofinedge].iedgefreq := rusagemat[k].primfreq;
  inedge[noofinedge].iedgelabel := 'I';
  edgemat[noofedges].edgpnt4 := noofinedge
end;

edgemat[noofedges].edgpnt1 := entitya;
edgemat[noofedges].edgpnt2 := entityb;
edgemat[noofedges].edgytype := 'r';
edgemat[noofedges].edgelab := 'I';
edgemat[noofedges].edgfreq := rusagemat[i].primfreq
end;

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

(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;
temppnt := 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 := temppnt;
end
noofswap := noofswap + 1
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[edgept1].noofoutedge do
with labelmat[edgept1] do
begin
if outedge[j].oedgelabel = 'C' then
labelc := labelc + 1
end;
if labelc = 0 then
begin
edgelab := 'C';
labelmat[edgept1].outedge[edgept3].oedgelabel := 'C';
if edgetype = 'r' then
labelmat[edgept2].inedge[edgept4].iedgelabel := 'C'
end
end
end;
Appendix C

{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;
            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;

75
Appendix C

{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.
Y

Assign label C where feasible

Assign label W where feasible
program ingimp (input, output, relfile);

const maxstrlength = 20;
maxnoret = 40;
maxnoent = 40;
maxnoatt = 20;
maxrelatt = 20;
maxnokey = 5;
maxnoidnt = 5;
avith = 'WITH';
avwithout = 'WITHOUT';
maxnopro = 20;
maxinedge = 40;
maxoutedge = 40;
maxnoedge = 400;

type str = array [1..maxstrlength] of char;
relstat = (weth, without);
props = record
  entptr: integer;
  attrptr: integer
end;
relation = record
  rname : str;
  entitya: integer;
  degenta: char;
  membshpa: char;
  entityb: integer;
  degentb: char;
  membshpb: char;
case rs: relstat of
  weth: (noreal : 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;
normrel = record
  nrname: integer;
  idcntent: 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;
norentment = record
  nename : str;
nooforops : integer;
identcnt : integer;
norentatts : array [1..maxnprops] of props
end;

outdet = record
  oedgectype : char;
oedgепnt : integer;
oedgefreq : integer;
oedgelabel : char
end;

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

edgedet = record
  edgelab : char;
edgefreq : integer;
edgetype : char;
edgetpl1 : integer;
edgetpl2 : integer;
edgetpl3 : integer;
edgetpl4 : integer
end;

graphdet = record
  noofoutedge : integer;
nofinedge : 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;
norentchttr : array [1..maxnoent] of norent;
ingentchart : array [1..maxnoent] of ingent;
edgemat : array[1..maxnoedg] of edgedet;
labelmat : array[1..maxnoedg] of graphdet;
noofofrelations : integer;
relfile: text;
i, j, k, l : integer;
norelfile : integer;
nopofentities : integer;
tenfile : text;
mapfile : text;
tempbuff : str;
found : boolean;
correct : boolean;
answer : str;
noofnormrel : integer;
totinfreq : integer;
condition : boolean;
icondition : boolean;
edgfile : 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;
  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;
    begin
    {READ IN THE RELATION FILE}
    reset (relfile, 'hosprel');
    readln (relfile, noofrelations);
    for i := 1 to noofrelations do
      with relchart[i] do
begin
readln (refile);
readstr (refile, rname);
readln (refile);
readstr (refile, tempbuff);
findent (entitya);
readln(refile, degenta);
readln (refile, membshpa);
readstr (refile, tempbuff);
findent (entityb);
readln (refile, degentb);
readln (refile, membshpb);
readstr (refile, tempbuff);
if tempbuff = awith then rs := weth
else rs := wewithout;
readln (refile);
if rs = weth then
begin
readln (refile, norelatt);
k := 1;
for j := 1 to norelatt do
begin
readstr (refile, relatt[j]);
k := k + 1;
if k > 4 then
begin
readln (refile);
k := 1
end
end
end

{MAP THE RELATIONS}
j := 0;
for i := 1 to noofrelations do
with relchart[i] do
begin
if (degenta <> '1') and (membshpb <> '0') then
begin
j := j + 1;
with norrelchtr [j] do
begin
if (degenta = '1') and (degentb = '1') and (membshpb = 'n') then
begin
nrrname := 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
else
  if (degenta = 'l') 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
      norelatt[k] := k
      end
    end
  end;
noofnormrel := j;
Appendix C

{MAP ALL THE ENTITIES}
for i := 1 to noofentities do
  with norentcht[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 = '0')
                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

{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, edgepoint1);
      read (edgefile, edgepoint2);
      read (edgefile, edgepoint3);
      read (edgefile, edgepoint4);
      readln (edgefile)
    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, oedgepnt);
    read (labelfile, oedgedtype);
    read (labelfile, oedgedfreq);
    read (labelfile, oedgelabel);
    readln (labelfile)
  end;
  for j := 1 to noofinedge do
    with inedge[j] do
      begin
        read (labelfile, iedgepnt);
        read (labelfile, iedgedtype);
        read (labelfile, iedgelabel);
        readln (labelfile)
      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].iedgedfreq
            end;
          l := 0;
          icontion := false;
          while (l < noofinedge) and (not icontion) do
            begin
              l := l + 1;
              if (inedge[l].iedgelabel = 'W') or
                 (inedge[l].iedgelabel = 'C') then
                icontion := true
            end;
        end;
        IF fi + SUM(INEDGE FREQUENCY) > FREQUENCY OF EDGE MARKED 'W' or 'C'
        THEN HASH ON IDENTIFIER EDGE
      end;
    if condition then
      begin
        if outedge[1].oedgedfreq + totinfreq >= outedge[j].oedgedfreq then
          begin
            nrentchart[i].struc := 'H';
            nrentchart[i].edgeno := 1;
            nrentchart[i].diredge := 'o'
          end;
      end;
(IF OUTEDGE MARKED 'W' OR 'C' IS AN ATTRIBUTE THEN ISAM ON THAT PROPERTY)

    if (outedge[l].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

(IF INEDGE MARKED 'W' OR 'C' HASH ON PRIMARY KEY)
else if icondition then
    begin
        ingentchart[i].struc := 'H';
        ingentchart[i].edgeno := relchart[inedge[l].edgept].entitya;
        ingentchart[i].diredge := 'i'
    end
else
    begin
        ingentchart[i].struc := 'H';
        ingentchart[i].edgeno := 1;
        ingentchart[i].diredge := 'o'
    end;

(WRITE TO THE MAPFILE SUITABLE FOR INGRES SCHEMA)

    rewrite (mapfile, 'ingfile');
    writeln (mapfile,noofentities);
    for i := 1 to noofentities do
    with norentchart[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 noentatts[j] do
        begin
            writestr (mapfile, entchart[entptr].entatt[attptr]);
        end;
        writeln (mapfile);
        k := 1;
        for j := (identcnt + 1) to noofprops do
        with noentatts[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);
        if struc = 'H' then
        write (mapfile, 'Hash it on');
if struc = 'M' then
write (mapfile, 'ISAM it on ');
if diredge = '0' 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;

(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;
if nrs = veth 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;
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

Obtain entities | Obtain relationships | Obtain the relational model | Obtain edge details | Obtain label details

Create INGRES compatible schema

Assign INGRES Structure | Write the structure on the mapfile

Assign to the relations representing entities | Assign to the relations representing relationships

Read the number of relations representing entities | For each relation do *

Assign structure to 0 each relation

X

Y
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

A

Read number of relations representing relationship

Assign structure to the relations

Assign secondary index on identifier of entity a
Assign secondary index on identifier of entity b

X

write the relations representing the entities to the mapfile

Write the relations representing the relationships to the mapfile
program mimimp (input, output, relfile);

const maxstrlength = 20;
  maxnrel = 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, without);
props = record
  entp: integer;
  attp: integer
end;
relation = record
  rname : str;
  entitya : integer;
  degenta : char;
  membshpa : char;
  entityb : integer;
  degentb : char;
  membshpb : char;
case rs : relstat of
  weth : (nrelatt : 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;
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 : (nrtattcnt : integer;
    nrelatt : array [1..maxrelatt] of integer);
  without : ()
end;
norment = record
    nename : str;
    noofprops : integer;
    identcnt : integer;
    norentatts : array [1..maxnoprops] of props
end;

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

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

edgedet = record
    edgelab : char;
    edgefreq : 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..maxnoremel] of normrel;
  norentchtr : array [1..maxnoent] of norment;
  mimentchart : array [1..maxnoent] of miment;
  edgemat : array[1..maxnoedge] of edgedet;
  labelmat : array[1..maxnoent] of graphdet;
  nooofrelations : integer;
  relfile: text;
  i, j, k, l : integer;
  norelfile : integer;
  nooofentities : integer;
  entfile : text;
  mapfile : text;
  tempbuff : str;
found : boolean;
correct : boolean;
answer : str;
nofnormrel : integer;
totinfreq : integer;
condition : boolean;
icondition : boolean;
edgefile : text;
labelfile : text;
nofedges : 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 := 1;
        found := true
      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;
        begin
          if k > 4 then
            begin
              readln (entfile);
              k := 1
            end
        end;
      readln (entfile);
      readln (entfile)
    end;

end;
Appendix C

[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 = a with 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 <> '0') then
        begin
            j := j + 1;
            with norrelchtr [j] do
                begin
                    if (degenta = '1') and (degentb = '1') and (membshpb = 'n') then
                        begin
                            nnname := i;
                            identcnt := entchart[entitya].keycount;
                            for k := 1 to identcnt do
                                begin
                                    norrelidnt[k].entptr := entitya;
                                    norrelidnt[k].attptr := k
                                end;
                            nkeycnt := entchart[entityb].keycount;
                        end
                    else if (degenta = '1') and (membshpb = 'n') then
                        begin
                            nnname := i;
                            identcnt := entchart[entitya].keycount;
                            for k := 1 to identcnt do
                                begin
                                    norrelidnt[k].entptr := entitya;
                                    norrelidnt[k].attptr := k
                                end;
                            nkeycnt := entchart[entityb].keycount;
                        end
                end
        end
end;
for k := 1 to nrkeycnt do
    begin
        norrelkey[k].entptr := entityb;
        norrelkey[k].attptr := k
    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
        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
                nrtcnt := norelatt;
                for k := 1 to nrtcnt do
                    begin
                        nrelatt[k] := k
                    end
            end
        end
end;  
noofnormrel := j;
{MAP ALL THE ENTITIES}
for i := 1 to noofentities 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 = 'l') and (relchart[l].membshpb = '0')
              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;
{READ THE EDGE DETAILS}
reset (edgefile, 'hospedge');
readln (edgefile, noofedges);
for i := 1 to noofedges 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;
(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;
  end;
for i := 1 to noofentities do
  with norentschr[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 := '0'
end
else

{IF AN INEDGE IS LABELLED 'W' OR 'C' THEN CREATE SECONDARY INDEX ON} 
{THE KEY OF THE ENTITYA OF THE RELATIONSHIP}
if condition then
begin
  mimentchart[i].struc := 'I';
  mimentchart[i].edgeno := relchart[inedge[i].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}

rewite (mapfile, 'mimfile');
writeln(mapfile, noofentities);
for i := 1 to noofentities do
  with norentchart[i] do
  with mimentchart[i] do
    begin
      writeln (mapfile);
      writestr (mapfile, nename);
      write(mapfile, noofprops);
      write(mapfile,' ');
      writeln (mapfile, idenctnt);
      for j := 1 to idenctnt do
        with norentatts[j] do
          writestr (mapfile, entchart[entptr].entatt[attptr]);
      writeln(mapfile);
      k := 1;
      for j := (idenctnt + 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;
writeln (mapfile);
if struct <> '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 norrelchr[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;
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;
  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.
Appendix C

Program Listings

Y

Outedge labelled W or C
Assign structure to 0 the relation representing the entity

Inedge labelled W or C
Invert the relationship representing this in-edge on the key of entity

Outedge labelled W or C is the primary key
No additional structure

Outedge labelled W or C is not key
Invert the relation representing this entity on this attribute

X

Write the relations representing the entities to the mapfile

Write the relations representing the relationships to the mapfile
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 'ENTEIST.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'.

106
A Typical Run of the Program RELTEST.P

Script started on Mon Feb 23 16:33:53 1987
S 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 1or 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.
S "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

<table>
<thead>
<tr>
<th>PRIMARY KEY</th>
<th>pat-no</th>
<th>pat-address</th>
<th>pat-category</th>
<th>reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>pat-name</td>
<td>date-of-birth</td>
<td>marital status</td>
<td>next-of-kin</td>
<td></td>
</tr>
<tr>
<td>sex</td>
<td>allergy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>blood group</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PRIMARY KEY</th>
<th>disease-no</th>
</tr>
</thead>
<tbody>
<tr>
<td>disease-name</td>
<td>contagious</td>
</tr>
<tr>
<td>consult</td>
<td>treatment</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PRIMARY KEY</th>
<th>emp-no</th>
</tr>
</thead>
<tbody>
<tr>
<td>con-name</td>
<td>con-address</td>
</tr>
<tr>
<td>consultant</td>
<td>speciality</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PRIMARY KEY</th>
<th>clinic-no</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>time-of-start</td>
<td>time-of-finish</td>
<td></td>
</tr>
<tr>
<td>est: repeat pat</td>
<td>new pat booked</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PRIMARY KEY</th>
<th>clinic-no</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>free/booked</td>
<td>new/repeat</td>
<td></td>
</tr>
<tr>
<td>Table Name</td>
<td>Primary Key</td>
<td>Field 1</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------</td>
<td>------------------</td>
</tr>
<tr>
<td>PRIMARY KEY</td>
<td>W/1-no</td>
<td>list size</td>
</tr>
<tr>
<td>Surgical W/1</td>
<td>S-W/1-no</td>
<td>list length</td>
</tr>
<tr>
<td>Surgical-session</td>
<td>session-no</td>
<td>date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Admission W/1</td>
<td>A-W/1-no</td>
<td>list length</td>
</tr>
<tr>
<td>Blood request</td>
<td>Request-no</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>reason</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reserve conc:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>units doctors no:</td>
</tr>
<tr>
<td>Pat-rec-blood</td>
<td>ref-no</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ward-code</td>
<td></td>
</tr>
<tr>
<td></td>
<td>past-trans</td>
<td></td>
</tr>
<tr>
<td></td>
<td>blood-group</td>
<td></td>
</tr>
<tr>
<td></td>
<td>antibody</td>
<td></td>
</tr>
</tbody>
</table>
# Appendix E

## Antibody cases

<table>
<thead>
<tr>
<th>1</th>
<th>8</th>
</tr>
</thead>
</table>

**PRIMARY KEY**

<table>
<thead>
<tr>
<th>anti-ref</th>
<th>phenotype</th>
<th>antibody</th>
<th>E.D.C</th>
</tr>
</thead>
</table>

**blood group**

**B.T.S confirm**

**Drug**

<table>
<thead>
<tr>
<th>1</th>
<th>5</th>
</tr>
</thead>
</table>

**PRIMARY KEY**

<table>
<thead>
<tr>
<th>drug-code</th>
<th>strength</th>
<th>pack-size</th>
<th>expiry-code</th>
</tr>
</thead>
</table>

**name**

**supplier**

<table>
<thead>
<tr>
<th>1</th>
<th>4</th>
</tr>
</thead>
</table>

**PRIMARY KEY**

<table>
<thead>
<tr>
<th>supplier-no</th>
<th>address</th>
<th>status</th>
</tr>
</thead>
</table>

**Prescription**

<table>
<thead>
<tr>
<th>1</th>
<th>3</th>
</tr>
</thead>
</table>

**PRIMARY KEY**

<table>
<thead>
<tr>
<th>pres-no</th>
<th>pres-date</th>
<th>prescriber</th>
</tr>
</thead>
</table>

**X-ray request**

<table>
<thead>
<tr>
<th>1</th>
<th>5</th>
</tr>
</thead>
</table>

**PRIMARY KEY**

<table>
<thead>
<tr>
<th>request-no</th>
<th>investigation</th>
<th>requestor</th>
<th>urgency</th>
<th>reason</th>
</tr>
</thead>
</table>

**X-ray sessions**

<table>
<thead>
<tr>
<th>4</th>
<th>6</th>
</tr>
</thead>
</table>

**PRIMARY KEY**

<table>
<thead>
<tr>
<th>date</th>
<th>unit</th>
<th>morning/afternoon</th>
</tr>
</thead>
<tbody>
<tr>
<td>x-ray type</td>
<td>maximum load</td>
<td>numbers booked</td>
</tr>
</tbody>
</table>
Appendix E

Staff details

1
5

<table>
<thead>
<tr>
<th>PRIMARY KEY</th>
<th>emp-no</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>grade</td>
</tr>
<tr>
<td></td>
<td>F.T/P.T</td>
</tr>
<tr>
<td></td>
<td>address</td>
</tr>
</tbody>
</table>

Wards

1
6

<table>
<thead>
<tr>
<th>PRIMARY KEY</th>
<th>Ward-code</th>
</tr>
</thead>
<tbody>
<tr>
<td>no-of-beds</td>
<td>w-type</td>
</tr>
<tr>
<td>nursing dependency</td>
<td>w-description</td>
</tr>
<tr>
<td></td>
<td>nursing load</td>
</tr>
</tbody>
</table>

Ward bed

2
4

<table>
<thead>
<tr>
<th>PRIMARY KEY</th>
<th>Ward-code</th>
</tr>
</thead>
<tbody>
<tr>
<td>male/female</td>
<td>free/occupied</td>
</tr>
<tr>
<td>bed-no</td>
<td></td>
</tr>
</tbody>
</table>

Nurses

1
10

<table>
<thead>
<tr>
<th>PRIMARY KEY</th>
<th>emp-no</th>
</tr>
</thead>
<tbody>
<tr>
<td>N.I.no</td>
<td>name</td>
</tr>
<tr>
<td>entry grade</td>
<td>grade</td>
</tr>
<tr>
<td>full equvalent</td>
<td>location code</td>
</tr>
<tr>
<td></td>
<td>qualification code</td>
</tr>
<tr>
<td></td>
<td>unit code</td>
</tr>
<tr>
<td></td>
<td>date of birth</td>
</tr>
</tbody>
</table>

Duty

4
5

<table>
<thead>
<tr>
<th>PRIMARY KEY</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>grade</td>
<td>shift</td>
</tr>
<tr>
<td>number required</td>
<td>W-code</td>
</tr>
</tbody>
</table>

Ind: workload

2
4

<table>
<thead>
<tr>
<th>PRIMARY KEY</th>
<th>week commencing</th>
</tr>
</thead>
<tbody>
<tr>
<td>max-hours</td>
<td>emp-no</td>
</tr>
<tr>
<td></td>
<td>hours-booked</td>
</tr>
</tbody>
</table>

114
Appendix E

In-patient
4
8

PRIMARY KEY
pat-no

w-code

bed-no
date entering

discharge date
illness

patient type

comment

N-absence rec
4
5

PRIMARY KEY
emp-no
day

month
year

reason

Supplier
1
4

PRIMARY KEY
supplier-no

name

address

status
22

Pat-Cons
Patient m
o
Consultant 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-schedule

WITHOUT

Appointment
Patient 1
n
Clinic-time-schedule
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
Appendix E

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 0/p-W/l n
n WITH
4 position reference complaints urgency

117
<table>
<thead>
<tr>
<th>Service</th>
<th>Op-code</th>
<th>Suggestion</th>
<th>Non-availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient</td>
<td>m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Surgical W/l</td>
<td>n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WITH 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Urgency position</td>
<td>op-code</td>
<td>suggestion</td>
<td>non-availability</td>
</tr>
<tr>
<td>Consultant/O/p-W/l</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consultant</td>
<td>n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>O/p-W/l</td>
<td>n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WITHOUT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duty rota Nurses</td>
<td>m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duty</td>
<td>n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WITHOUT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nurses/Wards Nurses</td>
<td>m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wards</td>
<td>n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WITHOUT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nurses/Workload Nurses</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ind: workload</td>
<td>n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WITHOUT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pat/in-pat Patient</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>In-patient</td>
<td>n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WITHOUT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supplied drug Supplier</td>
<td>m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Drug</td>
<td>n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WITHOUT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RELATION-NAME</td>
<td>ENTITY-A</td>
<td>ENTITY-B</td>
<td>DEGREE-A</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------</td>
<td>-----------------------</td>
<td>----------</td>
</tr>
<tr>
<td>Pat-Cons o n</td>
<td>Patient</td>
<td>Consultant</td>
<td>m</td>
</tr>
<tr>
<td>Pat-Disease n n</td>
<td>Patient</td>
<td>Disease</td>
<td>m</td>
</tr>
<tr>
<td>Cons-Clinic n o</td>
<td>Consultant</td>
<td>Clinical-session</td>
<td>1</td>
</tr>
<tr>
<td>Clinic/schedule o s</td>
<td>Clinical-session</td>
<td>Clinic-time-schedule</td>
<td>1</td>
</tr>
<tr>
<td>Appointment n n</td>
<td>Patient</td>
<td>Clinic-time-schedule</td>
<td>1</td>
</tr>
<tr>
<td>Consultant/S-W/l n o</td>
<td>Consultant</td>
<td>Surgical W/l</td>
<td>1</td>
</tr>
<tr>
<td>Consultant/S-sessionConsultant n o</td>
<td>Surgical-session</td>
<td>1</td>
<td>n</td>
</tr>
<tr>
<td>Pat/Blood req n o</td>
<td>Patient</td>
<td>Blood request</td>
<td>1</td>
</tr>
<tr>
<td>Pat/Pat-rec-blood n o</td>
<td>Patient</td>
<td>Pat-rec-blood</td>
<td>1</td>
</tr>
<tr>
<td>Patient/Antibody n o</td>
<td>Patient</td>
<td>Antibody cases</td>
<td>1</td>
</tr>
<tr>
<td>Pat/prescription n o</td>
<td>Patient</td>
<td>Prescription</td>
<td>1</td>
</tr>
<tr>
<td>Pat/X-ray request n o</td>
<td>Patient</td>
<td>X-ray request</td>
<td>1</td>
</tr>
<tr>
<td>X-ray sess/Pat n n</td>
<td>X-ray sessions</td>
<td>Patient</td>
<td>m</td>
</tr>
<tr>
<td>Staff/session n o</td>
<td>Staff details</td>
<td>X-ray sessions</td>
<td>m</td>
</tr>
<tr>
<td>Pat/O/p-W/l n n</td>
<td>Patient</td>
<td>O/p-W/l</td>
<td>m</td>
</tr>
<tr>
<td>Pat/S-W/l n n</td>
<td>Patient</td>
<td>Surgical W/l</td>
<td>m</td>
</tr>
<tr>
<td>Consultant/O/p-W/l n o</td>
<td>Consultant</td>
<td>O/p-W/l</td>
<td>1</td>
</tr>
<tr>
<td>Duty-rota n n</td>
<td>Nurses</td>
<td>Duty</td>
<td>m</td>
</tr>
</tbody>
</table>
## Appendix E

<table>
<thead>
<tr>
<th></th>
<th>Nurses/Wards</th>
<th>Nurses/Workload</th>
<th>Pat/In-pat</th>
<th>Supplied drug</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>n</td>
<td>n</td>
<td>s</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>Nurses/Wards</td>
<td>Nurses</td>
<td>Nurses</td>
<td>Ind: workload</td>
<td>Patient</td>
<td>1</td>
</tr>
<tr>
<td>n</td>
<td></td>
<td></td>
<td></td>
<td>In-patient</td>
<td>n</td>
</tr>
<tr>
<td>Supplied drug</td>
<td>Supplier</td>
<td></td>
<td></td>
<td>Drug</td>
<td>m</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outputs</th>
<th>m</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>n</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>n</td>
</tr>
<tr>
<td>m</td>
<td></td>
<td>n</td>
</tr>
</tbody>
</table>
### Outputs

<table>
<thead>
<tr>
<th>Enquiry</th>
<th>PRIMARY</th>
<th>300</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient</td>
<td>SELECT.BY.KEY</td>
<td></td>
</tr>
<tr>
<td>pat-no</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clinic-time-schedule</td>
<td>SELECT.BY.RELATION</td>
<td></td>
</tr>
<tr>
<td>Appointment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clinical-session</td>
<td>SELECT.BY.RELATION</td>
<td></td>
</tr>
<tr>
<td>Clinic/schedule</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consultant</td>
<td>SELECT.BY.RELATION</td>
<td></td>
</tr>
<tr>
<td>Cons-Clinic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Query blood</td>
<td>PRIMARY</td>
<td>350</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Patient</td>
<td>SELECT.BY.ATTRIBUTES</td>
<td></td>
</tr>
<tr>
<td>pat-name</td>
<td>EQUIJOIN</td>
<td></td>
</tr>
<tr>
<td>Pat-rec-blood</td>
<td>SELECT.BY.RELATION</td>
<td></td>
</tr>
<tr>
<td>Pat/Pat-rec-blood</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fix-Appt:</td>
<td>PRIMARY</td>
<td>350</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Patient</td>
<td>SELECT.BY.KEY</td>
<td></td>
</tr>
<tr>
<td>pat-no</td>
<td></td>
<td></td>
</tr>
<tr>
<td>O/p-W/l</td>
<td>SELECT.BY.RELATION</td>
<td></td>
</tr>
<tr>
<td>Pat/O/p-W/l</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consultant</td>
<td>SELECT.BY.RELATION</td>
<td></td>
</tr>
<tr>
<td>Consultant/O/p-W/l</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clinical-session</td>
<td>SELECT.BY.RELATION</td>
<td></td>
</tr>
<tr>
<td>Cons-Clinic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clinic-time-schedule</td>
<td>SELECT.BY.RELATION</td>
<td></td>
</tr>
<tr>
<td>Clinic/schedule</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sel-nurse-rota</td>
<td>SECONDARY</td>
<td>150</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duty</td>
<td>SELECT.BY.ATTRIBUTES</td>
<td></td>
</tr>
<tr>
<td>grade</td>
<td>EQUIJOIN</td>
<td></td>
</tr>
<tr>
<td>Ind: workload</td>
<td>SELECT.BY.RELATION</td>
<td></td>
</tr>
<tr>
<td>Nurses/Workload</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process</td>
<td>Function</td>
<td>Primary</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Check rota</td>
<td>PRIMARY</td>
<td>200</td>
</tr>
<tr>
<td>Duty date</td>
<td>SELECT.BY.KEY</td>
<td></td>
</tr>
<tr>
<td>Nurses Duty-rota</td>
<td>SELECT.BY.RELATION</td>
<td></td>
</tr>
<tr>
<td>Pat/dis enq</td>
<td>PRIMARY</td>
<td>200</td>
</tr>
<tr>
<td>Patient pat-no</td>
<td>SELECT.BY.KEY</td>
<td></td>
</tr>
<tr>
<td>Disease Pat-Disease</td>
<td>SELECT.BY.RELATION</td>
<td></td>
</tr>
<tr>
<td>Select-nurse</td>
<td>PRIMARY</td>
<td>400</td>
</tr>
<tr>
<td>Nurses grade</td>
<td>SELECT.BY.ATTRIBUTES</td>
<td>RANGE</td>
</tr>
</tbody>
</table>
### Output from ANALYSIS.P

#### ANALYSISFILE(Entity Analysis)

<table>
<thead>
<tr>
<th>Entity-Name</th>
<th>Primary functions</th>
<th>Secondary functions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rfrequency</td>
<td>Efrequency</td>
</tr>
<tr>
<td>Patient</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Primary-key</td>
<td>0</td>
<td>850</td>
</tr>
<tr>
<td>pat-name</td>
<td>0</td>
<td>350</td>
</tr>
<tr>
<td>pat-address</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pat-category</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>reference</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sex</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>date-of-birth</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>marital status</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>next-of-kin</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>blood group</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>allergy</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x-ray information</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Disease</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Primary-key</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>disease-name</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>contagious</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>treatment</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Consultant</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Primary-key</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>con-name</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>con-address</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>speciality</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Clinical-session</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Primary-key</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>time-of-start</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>time-of-finish</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>speciality</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>est: new pat</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>est: repeat pat</td>
<td></td>
<td></td>
</tr>
<tr>
<td>new pat booked</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rep: pat booked</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clinic-time-schedule</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Primary-key</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>free/booked</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>new/repeat</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Entity-Name: O/p-W/l</td>
<td>Primary functions</td>
<td>Secondary functions</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td></td>
<td>Rfrequency</td>
<td>Efrequency</td>
</tr>
<tr>
<td>Primary-key</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>speciality</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>list size</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>selection criteria</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Entity-Name: Surgical W/l</th>
<th>Primary functions</th>
<th>Secondary functions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rfrequency</td>
<td>Efrequency</td>
</tr>
<tr>
<td>Primary-key</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>speciality</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>list length</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>selection criteria</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Entity-Name: Surgical-session</th>
<th>Primary functions</th>
<th>Secondary functions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rfrequency</td>
<td>Efrequency</td>
</tr>
<tr>
<td>Primary-key</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>start time</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>end time</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>max: major cases</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>booked major cases</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>max: minor cases</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>booked minor cases</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Entity-Name: Admission W/l</th>
<th>Primary functions</th>
<th>Secondary functions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rfrequency</td>
<td>Efrequency</td>
</tr>
<tr>
<td>Primary-key</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>speciality</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>list length</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>selection criteria</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Entity-Name: Blood request</th>
<th>Primary functions</th>
<th>Secondary functions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rfrequency</td>
<td>Efrequency</td>
</tr>
<tr>
<td>Primary-key</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>reason</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>date</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>save serum</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>reserve blood units</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>reserve conc: units</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>doctors no:</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>blood taker’s no:</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>date of request</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### Appendix E

**Entity-Name**: Pat-rec-blood

<table>
<thead>
<tr>
<th>Primary functions</th>
<th>Secondary functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rfrequency</td>
<td>Efrequency</td>
</tr>
<tr>
<td>Primary-key</td>
<td>0</td>
</tr>
<tr>
<td>ward-code</td>
<td>0</td>
</tr>
<tr>
<td>blood-group</td>
<td>0</td>
</tr>
<tr>
<td>haemoglobin level</td>
<td>0</td>
</tr>
<tr>
<td>pregnancies</td>
<td>0</td>
</tr>
<tr>
<td>past-trans</td>
<td>0</td>
</tr>
<tr>
<td>antibody</td>
<td>0</td>
</tr>
<tr>
<td>prev:ref-no</td>
<td>0</td>
</tr>
</tbody>
</table>

**Entity-Name**: Antibody cases

<table>
<thead>
<tr>
<th>Primary functions</th>
<th>Secondary functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rfrequency</td>
<td>Efrequency</td>
</tr>
<tr>
<td>Primary-key</td>
<td>0</td>
</tr>
<tr>
<td>blood group</td>
<td>0</td>
</tr>
<tr>
<td>phenotype</td>
<td>0</td>
</tr>
<tr>
<td>antibody</td>
<td>0</td>
</tr>
<tr>
<td>E.D.C</td>
<td>0</td>
</tr>
<tr>
<td>B.T.S confirm</td>
<td>0</td>
</tr>
<tr>
<td>date</td>
<td>0</td>
</tr>
<tr>
<td>comment</td>
<td>0</td>
</tr>
</tbody>
</table>

**Entity-Name**: Drug

<table>
<thead>
<tr>
<th>Primary functions</th>
<th>Secondary functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rfrequency</td>
<td>Efrequency</td>
</tr>
<tr>
<td>Primary-key</td>
<td>0</td>
</tr>
<tr>
<td>name</td>
<td>0</td>
</tr>
<tr>
<td>setlength</td>
<td>0</td>
</tr>
<tr>
<td>pack-size</td>
<td>0</td>
</tr>
<tr>
<td>expiry-code</td>
<td>0</td>
</tr>
</tbody>
</table>

**Entity-Name**: supplier

<table>
<thead>
<tr>
<th>Primary functions</th>
<th>Secondary functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rfrequency</td>
<td>Efrequency</td>
</tr>
<tr>
<td>Primary-key</td>
<td>0</td>
</tr>
<tr>
<td>name</td>
<td>0</td>
</tr>
<tr>
<td>address</td>
<td>0</td>
</tr>
<tr>
<td>status</td>
<td>0</td>
</tr>
</tbody>
</table>

**Entity-Name**: Prescription

<table>
<thead>
<tr>
<th>Primary functions</th>
<th>Secondary functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rfrequency</td>
<td>Efrequency</td>
</tr>
<tr>
<td>Primary-key</td>
<td>0</td>
</tr>
<tr>
<td>pres-date</td>
<td>0</td>
</tr>
<tr>
<td>prescriber</td>
<td>0</td>
</tr>
<tr>
<td>Entity-Name</td>
<td>X-ray request</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Primary functions</td>
</tr>
<tr>
<td></td>
<td>Rfrequency</td>
</tr>
<tr>
<td>Primary-key</td>
<td>0</td>
</tr>
<tr>
<td>investigation</td>
<td>0</td>
</tr>
<tr>
<td>requester</td>
<td>0</td>
</tr>
<tr>
<td>urgency</td>
<td>0</td>
</tr>
<tr>
<td>reason</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Entity-Name</th>
<th>X-ray sessions</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Primary functions</td>
<td>Secondary functions</td>
</tr>
<tr>
<td></td>
<td>Rfrequency</td>
<td>Efrequency</td>
</tr>
<tr>
<td>Primary-key</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>maximum load</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>numbers booked</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Entity-Name</th>
<th>Staff details</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Primary functions</td>
<td>Secondary functions</td>
</tr>
<tr>
<td></td>
<td>Rfrequency</td>
<td>Efrequency</td>
</tr>
<tr>
<td>Primary-key</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>name</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>grade</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>F.T/P.T</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>address</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Entity-Name</th>
<th>Wards</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Primary functions</td>
<td>Secondary functions</td>
</tr>
<tr>
<td></td>
<td>Rfrequency</td>
<td>Efrequency</td>
</tr>
<tr>
<td>Primary-key</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>no-of-beds</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>w-type</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>w-description</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nursing load</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nursing dependency</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Entity-Name</th>
<th>Ward bed</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Primary functions</td>
<td>Secondary functions</td>
</tr>
<tr>
<td></td>
<td>Rfrequency</td>
<td>Efrequency</td>
</tr>
<tr>
<td>Primary-key</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>male/female</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>free/occupied</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Entity-Name :-</td>
<td>Nurses</td>
<td>Duties</td>
</tr>
<tr>
<td>---------------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td><strong>Primary-key</strong></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>N.I.no</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>name</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>grade</td>
<td>400</td>
<td>0</td>
</tr>
<tr>
<td>unit code</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>entry grade</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>location code</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>qualification code</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>date of birth</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>full equivalent</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Primary-key</strong></td>
<td>0</td>
<td>200</td>
</tr>
<tr>
<td>number required</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Primary-key</strong></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>max-hours</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>hours-booked</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Primary-key</strong></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>discharge date</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>illness</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>patient type</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>comment</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Primary-key</strong></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>reason</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Relationship Name</td>
<td>Primary Function Frequency</td>
<td>Secondary Function Frequency</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>Pat-Cons</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Pat-Disease</td>
<td>200</td>
<td>0</td>
</tr>
<tr>
<td>Cons-Clinic</td>
<td>650</td>
<td>0</td>
</tr>
<tr>
<td>Clinic/schedule</td>
<td>650</td>
<td>0</td>
</tr>
<tr>
<td>Appointment</td>
<td>300</td>
<td>0</td>
</tr>
<tr>
<td>Consultant/S-W/l</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Consultant/S-session</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Pat/Blood req</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Pat/Pat-rec-blood</td>
<td>350</td>
<td>0</td>
</tr>
<tr>
<td>Patient/Antibody</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Pat/prescription</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Pat/X-ray request</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>X-ray sess/Pat</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Staff/session</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Pat/0/p-W/l</td>
<td>350</td>
<td>0</td>
</tr>
<tr>
<td>Pat/S-W/l</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Consultant/0/p-W/l</td>
<td>350</td>
<td>0</td>
</tr>
<tr>
<td>Duty-rota</td>
<td>200</td>
<td>0</td>
</tr>
<tr>
<td>Nurses/Wards</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Nurses/Workload</td>
<td>0</td>
<td>150</td>
</tr>
<tr>
<td>Pat/In-pat</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Output from RELMAP.P MAPFILE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Patient</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pat-no</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>pat-name</td>
<td>pat-address</td>
<td></td>
</tr>
<tr>
<td>sex</td>
<td>date-of-birth</td>
<td></td>
</tr>
<tr>
<td>blood group</td>
<td>allergy</td>
<td></td>
</tr>
<tr>
<td>Disease</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>disease-no</td>
<td>contagious</td>
<td></td>
</tr>
<tr>
<td>disease-name</td>
<td>treatment</td>
<td></td>
</tr>
<tr>
<td>Consultant</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>emp-no</td>
<td>con-address</td>
<td></td>
</tr>
<tr>
<td>con-name</td>
<td>speciality</td>
<td></td>
</tr>
<tr>
<td>Clinical-session</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>clinic-no</td>
<td>date</td>
<td></td>
</tr>
<tr>
<td>time-of-start</td>
<td>time-of-finish</td>
<td></td>
</tr>
<tr>
<td>est: repeat pat</td>
<td>new pat booked</td>
<td></td>
</tr>
<tr>
<td>Clinic-time-schedule</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>clinic-no</td>
<td>date</td>
<td></td>
</tr>
<tr>
<td>free/booked</td>
<td>new/repeat</td>
<td></td>
</tr>
<tr>
<td>0/p-W/l</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>W/l-no</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>speciality</td>
<td>list size</td>
<td></td>
</tr>
<tr>
<td>Surgical W/l</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>S-W/l-no</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>speciality</td>
<td>list length</td>
<td></td>
</tr>
<tr>
<td>Surgical-session</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>session-no</td>
<td>date</td>
<td></td>
</tr>
<tr>
<td>start time</td>
<td>end time</td>
<td></td>
</tr>
<tr>
<td>max: minor cases</td>
<td>booked minor cases</td>
<td></td>
</tr>
<tr>
<td>admission W/l</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>A-W/l-no</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>speciality</td>
<td>list length</td>
<td></td>
</tr>
</tbody>
</table>

129
<table>
<thead>
<tr>
<th>Appendix E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blood request</td>
</tr>
<tr>
<td>Request-no</td>
</tr>
<tr>
<td>reason</td>
</tr>
<tr>
<td>reserve conc: units doctors no:</td>
</tr>
<tr>
<td>pat-no</td>
</tr>
<tr>
<td>date</td>
</tr>
<tr>
<td>save serum</td>
</tr>
<tr>
<td>blood taker's no:</td>
</tr>
<tr>
<td>reserve blood units</td>
</tr>
<tr>
<td>date of request</td>
</tr>
<tr>
<td>Pat-rec-blood</td>
</tr>
<tr>
<td>ref-no</td>
</tr>
<tr>
<td>ward-code</td>
</tr>
<tr>
<td>blood-group</td>
</tr>
<tr>
<td>antibody</td>
</tr>
<tr>
<td>haemoglobin level</td>
</tr>
<tr>
<td>prev:/ref-no</td>
</tr>
<tr>
<td>pregnancies</td>
</tr>
<tr>
<td>pat-no</td>
</tr>
<tr>
<td>Antibody cases</td>
</tr>
<tr>
<td>anti-ref</td>
</tr>
<tr>
<td>blood group</td>
</tr>
<tr>
<td>phenotype</td>
</tr>
<tr>
<td>antibody</td>
</tr>
<tr>
<td>E.D.C</td>
</tr>
<tr>
<td>pat-no</td>
</tr>
<tr>
<td>B.T.S confirm</td>
</tr>
<tr>
<td>date</td>
</tr>
<tr>
<td>comment</td>
</tr>
<tr>
<td>expirey-code</td>
</tr>
<tr>
<td>Drug</td>
</tr>
<tr>
<td>drug-code</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>strength</td>
</tr>
<tr>
<td>pack-size</td>
</tr>
<tr>
<td>supplier</td>
</tr>
<tr>
<td>supplier-no</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>address</td>
</tr>
<tr>
<td>status</td>
</tr>
<tr>
<td>Prescription</td>
</tr>
<tr>
<td>pres-no</td>
</tr>
<tr>
<td>pres-date</td>
</tr>
<tr>
<td>prescriber</td>
</tr>
<tr>
<td>pat-no</td>
</tr>
<tr>
<td>X-ray request</td>
</tr>
<tr>
<td>request-no</td>
</tr>
<tr>
<td>investigation</td>
</tr>
<tr>
<td>requestor</td>
</tr>
<tr>
<td>urgency</td>
</tr>
<tr>
<td>reason</td>
</tr>
<tr>
<td>X-ray sessions</td>
</tr>
<tr>
<td>date</td>
</tr>
<tr>
<td>maximum load</td>
</tr>
<tr>
<td>unit</td>
</tr>
<tr>
<td>numbers booked</td>
</tr>
<tr>
<td>morning/afternoon</td>
</tr>
<tr>
<td>x-ray type</td>
</tr>
<tr>
<td>Staff details</td>
</tr>
<tr>
<td>emp-no</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>grade</td>
</tr>
<tr>
<td>F.T/F.T</td>
</tr>
<tr>
<td>address</td>
</tr>
<tr>
<td>Field</td>
</tr>
<tr>
<td>----------------------------</td>
</tr>
<tr>
<td>Wards</td>
</tr>
<tr>
<td>Ward-code</td>
</tr>
<tr>
<td>no-of-beds</td>
</tr>
<tr>
<td>w-type</td>
</tr>
<tr>
<td>nursing dependency</td>
</tr>
<tr>
<td>Ward bed</td>
</tr>
<tr>
<td>Ward-code</td>
</tr>
<tr>
<td>male/female</td>
</tr>
<tr>
<td>Nurses</td>
</tr>
<tr>
<td>emp-no</td>
</tr>
<tr>
<td>N.I.no</td>
</tr>
<tr>
<td>entry grade</td>
</tr>
<tr>
<td>Duty</td>
</tr>
<tr>
<td>date</td>
</tr>
<tr>
<td>number required</td>
</tr>
<tr>
<td>Ind: workload</td>
</tr>
<tr>
<td>week commencing</td>
</tr>
<tr>
<td>max-hours</td>
</tr>
<tr>
<td>In-patient</td>
</tr>
<tr>
<td>pat-no</td>
</tr>
<tr>
<td>discharge date</td>
</tr>
<tr>
<td>N-absence rec</td>
</tr>
<tr>
<td>emp-no</td>
</tr>
<tr>
<td>reason</td>
</tr>
</tbody>
</table>

131
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 ;
Appendix E

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 ;

133
Appendix E

Record name is Drug
Drug       KEY is drug-code
  name        ;
  strength    ;
  pack-size   ;
  expirey-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
  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 ;
Appendix E

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/0/p-W/L
Pat/0/p-W/L KEY is pat-no ,W/L-no
  position ;
  reference ;
  complaints ;
  urgency ;
Appendix E

Record name is Pat/S-W/1
Pat/S-W/1 KEY is pat-no, S-W/1-no
  urgency ;
  op-code ;
  suggestion ;
  non-availability ;
  position ;

Record name is Duty-rota
Duty-rota KEY is emp-no, date, shift
  W-code , grade

Record name is Nurses/Wards
Nurses/Wards KEY is emp-no, Ward-code
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-schedule AUTOMATIC FIXED

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

Set name is Consultant/S-W/l  
Owner record is Consultant  
Member record is  
Surgical W/l 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
Appendix E

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
<table>
<thead>
<tr>
<th>W</th>
<th>850p</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>350a</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>1</td>
<td>7</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>1</td>
<td>8</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>1</td>
<td>9</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>1</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>1</td>
<td>11</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>1</td>
<td>12</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>W</td>
<td>0p</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>W</td>
<td>0p</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>W</td>
<td>0p</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>4</td>
<td>7</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>4</td>
<td>8</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>W</td>
<td>0p</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>W</td>
<td>0p</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>6</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>W</td>
<td>0p</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>7</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>7</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>W</td>
<td>0p</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>8</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>8</td>
<td>7</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>W</td>
<td>0p</td>
<td>9</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>9</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>9</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>9</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>W</td>
<td>0p</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>10</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>10</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>140</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>10</td>
<td>7</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>0p</td>
<td>11</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>11</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>11</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>11</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>11</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>11</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>11</td>
<td>7</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>11</td>
<td>8</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>0p</td>
<td>12</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>12</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>12</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>12</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>12</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>12</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>12</td>
<td>7</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>0p</td>
<td>13</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>13</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>13</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>13</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>13</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>0p</td>
<td>14</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>14</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>14</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>14</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>0p</td>
<td>15</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>15</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>15</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>0p</td>
<td>16</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>16</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>16</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>16</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>16</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>0p</td>
<td>17</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>17</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>17</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>0p</td>
<td>18</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>18</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>18</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>18</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>18</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>0p</td>
<td>19</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>19</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>19</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>19</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>19</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>19</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>0p</td>
<td>20</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>20</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>20</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0p</td>
<td>21</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>21</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>21</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>400a</td>
<td>21</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0a</td>
<td>21</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

141
| I   | 0a | 21 | 6  | 6  | 0  |
| I   | 0a | 21 | 7  | 7  | 0  |
| I   | 0a | 21 | 8  | 8  | 0  |
| I   | 0a | 21 | 9  | 9  | 0  |
| I   | 0a | 21 | 10 | 10 | 0  |
| W   | 200p | 22 | 1  | 1  | 0  |
| I   | 0a | 22 | 5  | 2  | 0  |
| W   | 0p | 23 | 1  | 1  | 0  |
| I   | 0a | 23 | 3  | 2  | 0  |
| W   | 0p | 24 | 4  | 3  | 0  |
| I   | 0a | 24 | 5  | 2  | 0  |
| I   | 0a | 24 | 6  | 3  | 0  |
| I   | 0a | 24 | 7  | 4  | 0  |
| I   | 0a | 24 | 8  | 5  | 0  |
| W   | 0p | 25 | 1  | 1  | 0  |
| I   | 0a | 25 | 5  | 2  | 0  |
| I   | 0r | 1  | 3  | 13 | 1  |
| I   | 0r | 1  | 2  | 14 | 1  |
| I   | 0r | 3  | 4  | 5  | 1  |
| I   | 0r | 4  | 5  | 9  | 1  |
| I   | 0r | 1  | 5  | 15 | 2  |
| I   | 0r | 3  | 7  | 6  | 1  |
| I   | 0r | 3  | 8  | 7  | 1  |
| I   | 0r | 1  | 10 | 16 | 1  |
| I   | 0r | 1  | 11 | 17 | 1  |
| I   | 0r | 1  | 12 | 18 | 1  |
| I   | 0r | 1  | 15 | 19 | 1  |
| I   | 0r | 1  | 16 | 20 | 1  |
| I   | 0r | 17 | 1  | 4  | 1  |
| I   | 0r | 18 | 17 | 6  | 1  |
| I   | 0r | 1  | 6  | 21 | 1  |
| I   | 0r | 1  | 7  | 22 | 2  |
| I   | 0r | 3  | 6  | 8  | 2  |
| I   | 0r | 21 | 22 | 11 | 1  |
| I   | 0r | 21 | 19 | 12 | 1  |
| I   | 0r | 21 | 23 | 13 | 1  |
| I   | 0r | 1  | 24 | 23 | 1  |
| p | a | a | a | a | a | a | a | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 25 | 23 | 1 | 850C | 350I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 200I | 300I | 0I | 350I | 0I | 0I | 350I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I |
| 4 | 1 | 0C | 0I | 0I | 0I | 0I | 200I | 0I | 650I | 0I | 0I | 350I | 0I | 0I | 0C | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I | 0I |
3
2
p  1  0C
a  0  0I
a  0  0I
4
5  650I
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

\[
\begin{array}{ccc}
a & 0 & 0 I \\
a & 0 & 0 I \\
9 & 350 I \\
8 & 1 \\
p & 1 & 0 C \\
a & 0 & 0 I \\
a & 0 & 0 I \\
a & 4 & 0 \\
p & 1 & 0 C \\
a & 0 & 0 I \\
a & 0 & 0 I \\
a & 3 & 1 \\
p & 1 & 0 C \\
a & 0 & 0 I \\
a & 11 & 0 I \\
5 & 1 \\
p & 1 & 0 C \\
a & 0 & 0 I \\
a & 0 & 0 I \\
a & 12 & 0 I \\
4 & 1 \\
p & 1 & 0 C \\
a & 0 & 0 I \\
a & 0 & 0 I \\
a & 13 & 0 I \\
r & 14 & 0 I \\
6 & 0 \\
p & 1 & 0 C \\
a & 0 & 0 I \\
a & 0 & 0 I \\
a & 0 & 0 I \\
r & 14 & 0 I \\
6 & 1 \\
p & 1 & 0 C \\
\end{array}
\]
<table>
<thead>
<tr>
<th>a</th>
<th>0</th>
<th>0</th>
<th>OI</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>1</td>
<td>0</td>
<td>OC</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>OI</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>OI</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>OI</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>OI</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>OI</td>
</tr>
<tr>
<td>p</td>
<td>1</td>
<td>0</td>
<td>OI</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>OI</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>OI</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>OI</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>OI</td>
</tr>
<tr>
<td>r</td>
<td>18</td>
<td>200I</td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>19</td>
<td>0I</td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>20</td>
<td>0I</td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>1</td>
<td>200C</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>OI</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>200I</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>OI</td>
</tr>
<tr>
<td>p</td>
<td>1</td>
<td>0</td>
<td>OC</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>OI</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>OI</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>OI</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>OI</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>OI</td>
</tr>
<tr>
<td>p</td>
<td>1</td>
<td>0</td>
<td>OC</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>OI</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>OI</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>OI</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>OI</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>OI</td>
</tr>
</tbody>
</table>

146
Appendix E

Output of ING.P
MAPFILE (Entities)

25

Patient
pat-no 12 1
pat-name pat-address 1
sex date-of-birth
blood group allergy
Hash it on: pat-no pat-category marital status next-of-kin
reference x-ray information

Disease
4 1
disease-no

treatment
Hash it on: disease-no
disease-name contagious

Consultant
4 1
emp-no
con-name con-address
Hash it on: emp-no speciality

Clinical-session
10 2
clinical-no date
time-of-start time-of-finish speciality
est: repeat pat new pat booked rep: pat booked
emp-no
Hash it on: clinical-no date

Clinic-time-schedule
5 3
clinical-no date
free/booked new/repeat
time
Hash it on: clinical-no date
time

O/p-W/l
5 1
W/l-no
speciality list size
selection criteria emp-no
Hash it on: W/l-no

Surgical W/l
5 1
S-W/l-no
speciality list length
selection criteria emp-no
Hash it on: S-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
time
Hash it on: session-no date

Admission W/l
4 1
A-W/l-no
speciality list length
selection criteria
Hash it on: A-W/l-no

147
<table>
<thead>
<tr>
<th>Appendix E</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blood request 10</td>
<td>1</td>
</tr>
<tr>
<td>Request-no</td>
<td>save serum</td>
</tr>
<tr>
<td>reason date</td>
<td>reserve blood units</td>
</tr>
<tr>
<td>reserve conc: units doctors no:</td>
<td>blood taker's no:</td>
</tr>
<tr>
<td>pat-no</td>
<td>date of request</td>
</tr>
<tr>
<td>Hash it onRequest-no</td>
<td></td>
</tr>
<tr>
<td>Pat-rec-blood 9</td>
<td>1</td>
</tr>
<tr>
<td>ref-no</td>
<td>haemoglobin level</td>
</tr>
<tr>
<td>ward-code blood-group antibody</td>
<td>pregnancies</td>
</tr>
<tr>
<td>past-trans antibody</td>
<td>pat-no</td>
</tr>
<tr>
<td>Hash it onref-no</td>
<td></td>
</tr>
<tr>
<td>Antibody cases 9</td>
<td>1</td>
</tr>
<tr>
<td>anti-ref</td>
<td>antibody</td>
</tr>
<tr>
<td>blood group phenotype date</td>
<td>E.D.C</td>
</tr>
<tr>
<td>B.T.S confirm date</td>
<td>comment</td>
</tr>
<tr>
<td>Hash it onanti-ref</td>
<td>pat-no</td>
</tr>
<tr>
<td>Drug 5</td>
<td>1</td>
</tr>
<tr>
<td>drug-code strength pack-size expiry-code</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td></td>
</tr>
<tr>
<td>Hash it ondrug-code</td>
<td></td>
</tr>
<tr>
<td>supplier 4</td>
<td>1</td>
</tr>
<tr>
<td>supplier-no</td>
<td></td>
</tr>
<tr>
<td>name address status</td>
<td></td>
</tr>
<tr>
<td>Hash it onsupplier-no</td>
<td></td>
</tr>
<tr>
<td>Prescription 4</td>
<td>1</td>
</tr>
<tr>
<td>pres-no</td>
<td></td>
</tr>
<tr>
<td>pres-date prescriber pat-no</td>
<td></td>
</tr>
<tr>
<td>Hash it onpres-no</td>
<td></td>
</tr>
<tr>
<td>X-ray request 6</td>
<td>1</td>
</tr>
<tr>
<td>request-no</td>
<td></td>
</tr>
<tr>
<td>investigation requestor urgency reason</td>
<td></td>
</tr>
<tr>
<td>pat-no</td>
<td></td>
</tr>
<tr>
<td>Hash it onrequest-no</td>
<td></td>
</tr>
<tr>
<td>X-ray sessions 6</td>
<td>4</td>
</tr>
<tr>
<td>date unit numbers booked x-ray type</td>
<td></td>
</tr>
<tr>
<td>maximum load unit numbers booked x-ray type</td>
<td></td>
</tr>
<tr>
<td>Hash it ondate</td>
<td></td>
</tr>
<tr>
<td>X-ray sessions 6</td>
<td>4</td>
</tr>
<tr>
<td>date unit numbers booked x-ray type</td>
<td></td>
</tr>
<tr>
<td>maximum load unit numbers booked x-ray type</td>
<td></td>
</tr>
<tr>
<td>Staff details 5</td>
<td>1</td>
</tr>
<tr>
<td>emp-no</td>
<td></td>
</tr>
<tr>
<td>name grade F.T/P.T address</td>
<td></td>
</tr>
<tr>
<td>Hash it onemp-no</td>
<td></td>
</tr>
</tbody>
</table>
## Appendix E

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

149
WAPPILE (Relationships)

7
Pat-Cons
2
pat-no
Hash it on pat-no
emp-no
Date registered
disease-no
Date diagnosed

X-ray sess/Pat
5
date
pat-no
Hash it on date
unit
suggestion
type
unit
morning/afternoon
x-ray
x-ray type

Pat/O/p-W/l
2
pat-no
complaints
Hash it on pat-no
W/l-no
urgency
position
reference

Pat/S-W/l
2
pat-no
suggestion
Hash it on pat-no
S-W/l-no
urgency
position
non-availability
op-code

Duty-rota
5
emp-no
grade
Hash it on emp-no
date
shift
W-code

Nurses/Wards
2
emp-no
Ward-code
Hash it on emp-no
### Appendix E

**Output of MIM.P**

**MAPFILE (Entities)**

<table>
<thead>
<tr>
<th>Entity Type</th>
<th>Value 1</th>
<th>Value 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>pat-no</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>pat-name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sex</td>
<td></td>
<td></td>
</tr>
<tr>
<td>blood group</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disease</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>disease-no</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>disease-name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consultant</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>emp-no</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>con-name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clinical-session</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>clinic-no</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>time-of-start</td>
<td></td>
<td></td>
</tr>
<tr>
<td>new pat booked</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clinic-time-schedule</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>free/booked</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>O/p-W/1</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>W/1-no</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>speciality</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Surgical W/1</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>S-W/1-no</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>speciality</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Surgical-session</td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>session-no</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>start time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>max: minor cases</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Admission W/1</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>A-W/1-no</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>speciality</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Outputs**

151
### Appendix E

<table>
<thead>
<tr>
<th>Field</th>
<th>Quantity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blood request request-no</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>Blood request reason</td>
<td>1</td>
<td>save serum</td>
</tr>
<tr>
<td>Blood request reserve conc:</td>
<td>1</td>
<td>blood taker's no:</td>
</tr>
<tr>
<td>Blood request units doctors no:</td>
<td>1</td>
<td>reserve blood units</td>
</tr>
<tr>
<td>Blood request pat-no</td>
<td>1</td>
<td>date of request</td>
</tr>
<tr>
<td>Pat-rec-blood ref-no</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>Pat-rec-blood ward-code</td>
<td>9</td>
<td>haemoglobin level</td>
</tr>
<tr>
<td>Pat-rec-blood past-trans</td>
<td>9</td>
<td>prev:/ref-no</td>
</tr>
<tr>
<td>Antibody cases anti-ref</td>
<td>9</td>
<td>pregnancies</td>
</tr>
<tr>
<td>Antibody cases blood group</td>
<td>9</td>
<td>E.D.C</td>
</tr>
<tr>
<td>Antibody cases B.T.S confirm</td>
<td>9</td>
<td>pat-no</td>
</tr>
<tr>
<td>Drug drug-code name</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Drug drug-code strength</td>
<td>5</td>
<td>pack-size</td>
</tr>
<tr>
<td>Drug drug-code expirey-code</td>
<td>5</td>
<td>expirey-code</td>
</tr>
<tr>
<td>supplier supplier-no</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>supplier supplier-no name</td>
<td>4</td>
<td>status</td>
</tr>
<tr>
<td>Prescription pres-no</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Prescription pres-date</td>
<td>4</td>
<td>prescriber</td>
</tr>
<tr>
<td>Prescription requestor</td>
<td>4</td>
<td>pat-no</td>
</tr>
<tr>
<td>Prescription investigation</td>
<td>4</td>
<td>urgency</td>
</tr>
<tr>
<td>Prescription pat-no</td>
<td>4</td>
<td>reason</td>
</tr>
<tr>
<td>X-ray request request-no</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>X-ray request investigation</td>
<td>6</td>
<td>requestor</td>
</tr>
<tr>
<td>X-ray request pat-no</td>
<td>6</td>
<td>urgency</td>
</tr>
<tr>
<td>X-ray request unit</td>
<td>6</td>
<td>x-ray type</td>
</tr>
<tr>
<td>X-ray sessions date</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>X-ray sessions numbers booked</td>
<td>6</td>
<td>morning/afternoon</td>
</tr>
<tr>
<td>Staff details emp-no name</td>
<td>5</td>
<td>F.T/P.T</td>
</tr>
<tr>
<td>Staff details grade</td>
<td>5</td>
<td>address</td>
</tr>
<tr>
<td>Staff details address</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Wards Ward-code no-of-beds</td>
<td>6</td>
<td>w-description</td>
</tr>
<tr>
<td>Wards Ward-code w-type</td>
<td>6</td>
<td>nursing load</td>
</tr>
<tr>
<td>Wards Ward-code nursing</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>Wards Ward-code dependency</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>Category</td>
<td>Code</td>
<td>Format</td>
</tr>
<tr>
<td>---------------------</td>
<td>------</td>
<td>--------------</td>
</tr>
<tr>
<td>Ward bed</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Ward-code</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>male/female</td>
<td></td>
<td>free/occupied</td>
</tr>
<tr>
<td>Nurses</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>emp-no</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>N.I.no</td>
<td></td>
<td>name</td>
</tr>
<tr>
<td>entry grade</td>
<td></td>
<td>grade</td>
</tr>
<tr>
<td>full equivalent</td>
<td></td>
<td>unit code</td>
</tr>
<tr>
<td>Invert it on grade</td>
<td></td>
<td>date of birth</td>
</tr>
<tr>
<td>Duty date</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>number required</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Ind: workload</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>week commencing</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>max-hours</td>
<td></td>
<td>emp-no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hours-booked</td>
</tr>
<tr>
<td>In-patient pat-no date</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>emp-no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>w-code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bed-no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>illness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>patient type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>date entering</td>
</tr>
<tr>
<td></td>
<td></td>
<td>comment</td>
</tr>
<tr>
<td>N-absence rec</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>emp-no</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>reason</td>
<td></td>
<td>day</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>month</td>
</tr>
<tr>
<td></td>
<td></td>
<td>year</td>
</tr>
</tbody>
</table>
MAPFILE (Relationships)

7
Pat-Cons
2
pat-no emp-no Date registered
Invert it on pat-no
Pat-Disease
2
disease-no Date diagnosed
Invert it on pat-no
X-ray sess/Pat
5
date unit morning/afternoon x-ray type
pat-no suggestion unit x-ray
Invert it on date
type
Pat/0/p-W/1
2
pat-no W/1-no position reference
complaints urgency
Invert it on pat-no
Pat/S-W/1
2
pat-no S-W/1-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
**
** 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)

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)

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)

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

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

modify output-wl to hash on W/-no  
save output-wl until june 21 1986
**
** admission-wl (a) - "admission waiting list"
**
create admission(
  A-W/l-no = c3,
  speciality = c20
  length = i2,
  criteria = c30)

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,
  S-W/l-no = c3
  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,
  emp-no = c7
  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,
  clinic-no = c3,
  date = c10,
  start = i2,
  drugs = c30,
  progress = c30,
  nexttime = c10,
  arrangement = c20)

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,  
speciality = c20,  
criteria = c30,  
maxsize = i2  
emp-no = c7 )  

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,  
S-W/l-no = c10)  

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

** consultant (c)

create consultant(  
emp-no = c7,  
name = c30,  
address = c80,  
speciality = c20)  

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,  
session-no = c3,  
date = c10)  

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-W/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 )
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,
    rvbu = c1,
    rccu = c1,
    date = c10,
    taker = c30 )
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 )
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 )
modify proteinpat to isam on labcode,date,pid
save proteinpat until june 21 1986
** proteinday (prd) - protein day record

create proteinday
  ( labcode = c6, date = c10, notest = c12 )
  lab code date 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, pat-no = c7, phenotype = c2, antibody = c10, edc = c1, bts = c1, date = c10, comment = c15 )
  anti-ref patient id. phenotype antibody edc bts confirmation date comment
modify antibodyprob to hash on id
save antibodyprob until june 21 1986

** proteinspec (prs) - protein special cases

create proteinspec
  ( prospref = c6, pid = c7, diag = c10, treat = c20, gereport = c15 )
  prospref patient id. diagnosis treatment 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, pid = c7, diag = c10, treat = c20 )
  prointref patient id. diagnosis treatment
modify proteinint to hash on prointref
save proteinint until june 21 1986

161
Appendix F

** coagpat (cp) - coag clinic patient

create coagpat

( coagref = c6,
  pid = c7,
  diag = c15,
  regeme = c15,
  commther = c10,
  nextclinic = c10,
  comment = c20 )

modify coagpat to hash on coagref
save coagpat until june 21 1986

** coagpatprog (cpp) - coag clinic patient progress

create coagpatprog

( coagref = c6,
  date = c10,
  pid = c7,
  result = c15,
  dose = c10 )

modify coagpatprog to isam on coagref,date
save coagpatprog until june 21 1986

** coagclinic (cc) - coag clinic

create coagclinic

( date = c10,
  time = i2,
  maxpat = i2,
  nopat = i2 )

modify coagclinic to isam on date,time
save coagclinic until june 21 1986

** spectestreq (spt) - specimen test requests

create spectestreq

( reqno = i2,
  reqcode = c5,
  pid = c7,
  date = c10,
  source = c10,
  status = c6 )

modify spectestreq to hash on reqno
save spectestreq until june 21 1986
** specimen (sp) - specimen
**
create specimen

```plaintext
(specno = i2,
spectype = c10,
reqno = i2,
invest = c1,
urgency = c10,
testcode = c5,
labcode = c6,
result = c10,
comment = c20 )
```

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,
name = c40,
descr = c80,
strength = c10,
packsize = i4,
expiry-code = c10 )

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,
drugid = c7,
qty = i2,
dose = c10,
duration = c10 )

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

Drug identification number
Name of drug
Description of Drug, including use
Concentration of drug, eg 40mg/litre
Number in packs held in stock
Roughly best before codes

These details relate to prescription
Drug identification number for this
Quantity prescribed
Number to be taken per time period
Time over which drug must be taken,
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

vo-details - Abbreviated to wod

Item details for a ward order

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

modify vo-details to isam on id, drug-id
save vo-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
Appendix F

*** 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, number of beds
nobeds = i3, number of nurses required
nursesreq = i1, nursing dependency
dependency = i1)

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

165
** ward-bed (vb) - 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 (1) - 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

169