DESIGNING A DATABASE FOR INGRES

by

L. E. Dumont

Memorandum No. UCB/ERL M82/23

1 February 1982

(cover)

Research sponsored by the National Science Foundation Grant ECS-8007683.

DESIGNING A DATABASE FOR INGRES

by

Lorna E. DuMont

Memorandum No. UCB/ERL M82/23

1 February 1982

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Research sponsored by the National Science Foundation Grant ECS-8007683.

DESIGNING A DATABASE FOR INGRES

The first step in developing an INGRES database is to select formats for the relations that will hold your data--that is, to design the logical schema. This tutorial explains how to use the command 'designdb' to help you do this. Most of the tutorial is available in the on-line explanations, but you should read through it before you begin a design, to give yourself an overview. Refer back to the tutorial or to the on-line version whenever you have questions.

REFERENCES: Some familiarity with INGRES is assumed; if you have never used INGRES, you should begin with "A Tutorial on INGRES" (UCB/ERL M77/25) and "Creating and Maintaining a Database Using INGRES" (UCB/ERL M77/71), both by Robert Epstein. The "INGRES Reference Manual" (UCB/ERL M79/43) provides a general purpose guide to INGRES. This design-aid is based to a large extent on a design methodology developed by Randy Katz and Eugene Wong; refer to "Database Design and Translation For Multiple Data Models" (UCB/ERL M80/24), by Randy Katz, if you are interested in the underlying approach.

PURPOSE: This design-aid provides: 1) a natural framework and orderly procedure for specifying your design, and 2) automatic creation of a normalized INGRES schema. The framework used is that of objects and relationships; the six steps within the design-aid 'designdb' will help you to select an Intermediate Design--of objects and relationships among them--based on knowledge of the data you eventually want in your database. When you are satisfied with your choices, step 7 of the design-aid will take the Intermediate Design and produce an INGRES schema. This schema will be normalized, so as to preclude unexpected side-effects as you update your data.

CALLING DESIGNDB: The information that you specify while using this design-aid is held in an INGRES database, designated the 'design- information database'. If you are begining a new design, you must create an empty database for this purpose, using the Unix command 'creatdb'. The command 'designdb' will start the design- aid, and you supply the name of the design-information database. If you stop the design-aid (see below), you may continue your design later by calling 'designdb' and supplying the same database name. You may also modify a completed design, by saving the design-information database and supplying that name to 'designdb'.

MENU OF STEPS: When 'designdb' has finished the initializtion, you will see a menu of steps, as below:

```
Select step:
0 =print general explanation     1 =choose entity sets
2 =choose relationships          3 =choose subschemas
4 =choose constraints            5 =analyze and display
6 =choose attributes             7 =produce design and quit
8 =quit, saving work

Please type a number from the menu.
```

It is suggested that you begin each design by selecting these steps in order. You can at any time return to an earlier step to modify your design; you can also select steps out of order whenever you wish, but you may get a message directing you back to complete an earlier step. You can always postpone working on a design by selecting step 8 (=quit, saving work); the choices you have made so far will be saved in the design-information database.

STEPS 1 TO 6 (MENU OF OPERATIONS): In steps 1 to 6 you are working on the intermediate design. Each step is explained in the corresponding section of the tutorial; an example is shown, with remarks (indicated by <----) for clarity and titles (e.g. <LIST>) for easy reference. Each step has its own menu of operations, similar in format to the menu of steps. Operation 0 on each menu will print the same explanation that is in the tutorial. The other operations are specific to the step; within each one, the design-aid will prompt you for the necessary information (see the examples).

THE RETURN CONVENTION: The 'return' key is your way of 'escaping' or 'backing up'. When you see a menu of operations and the 'operation?' prompt, you can 'back up' to the menu of steps by pressing return. You may then select another step. In general, when you respond to any prompt by pressing return, without typing anything (not even a space), this will let you 'escape' from that prompt and will back you up to an earlier level of prompt. This will become clearer as you look at the examples, but it is always safe to press return until you are back at one of the menus. (You won't be able to back up beyond the menu of steps). The only exceptions to this use of pressing return are clearly marked.

STEP 7: When you have completed steps 1 to 6, and are satisfied that your intermediate design adequately reflects

the real-world meaning of your application, select step 7 (=produce design and quit). This step produces two files, one with a listing of your intermediate design and the resulting INGRES schema, and one with 'create' statements for the relations in the schema. If you ever decide to modify your design, simply supply the name of the design-information database to 'designdb', make your modifications, and select step 7 once more.

THREE NAMES TO REMEMBER: It is especially important to remember the name of the design-information database, since you must know this name to continue or modify a design. It is also important to remember the two file names that you choose in step 7 for the design/schema file and the 'create-statement' file. (The file names will appear in your directory, once they have been created). You should be prepared to choose new names (or reuse previous names) when you use step 7 of designdb.

STEP 1: CHOOSE ENTITY SETS

1.1. INTRODUCTION

The first step toward your intermediate design is to identify those real-world objects about which you want to record information in your database. An object is called an 'entity'; for example, 'Matilda Zilch' and 'machinist2' could be entities. Entities are grouped into SETS (or TYPES), such as 'employees' and 'jobs'. In step 1, you choose the entity sets (entity types) for your design. Note that in the design stage you are specifying only the types of entities and not the entities themselves; 'employees' and 'jobs' become part of your design, but 'Matilda Zilch' and 'machinist2' will eventually be data in your database.

1.2. EXAMPLE

This is the begining of a small example, which will be continued in the later steps. Suppose you have read the explanation, and want to choose your entity sets. Starting at the menu of operations for step one, the terminal would look like this:

```
Step 1: Choose entity sets

Select operation:
0 =print explanation          1 =define entity set(s)
2 =delete entity set(s)       3 =change name of entity set(s)
4 =list entity sets

operation? 1                                        <DEFINE>

Define entity set(s):

entity set? employees
entity set? managers
entity set? buildings
entity set? departments
entity set?                   <----At this point, you realize
                                   that you don't really want
                                   'managers' to be a seperate
                                   entity set; they will be
                                   part of the employees
                                   entity set instead. So you
                                   press return without typing
                                   anything (not even a space).
```

```
Select operation:
0 =print explanation          1 =define entity set(s)
2 =delete entity set(s)       3 =change name of entity set(s)
4 =list entity sets

operation? 2                                    <DELETE>


Delete entity set(s):

entity set? managers          <----You correct the mistake, and
entity set?                   <----you press return.

Select operation:
0 =print explanation          1 =define entity set(s)
2 =delete entity set(s)       3 =change name of entity set(s)
4 =list entity sets

operation? 1                                    <DEFINE>

Define entity set(s):

entity set? job
entity set?                   <----You realize that you left off
                                   the 's', so you press return.

Select operation:
0 =print explanation          1 =define entity set(s)
2 =delete entity set(s)       3 =change name of entity set(s)
4 =list entity sets

operation? 3                                    <CHANGE>

Change name of entity set(s):

old entity-set-name? job
new entity-set-name? jobs
old entity-set-name?          <----You have fixed the mistake.

Select operation:
0 =print explanation          1 =define entity set(s)
2 =delete entity set(s)       3 =change name of entity set(s)
4 =list entity sets

operation? 4                                    <LIST>
```

List of entity sets:

```
entlist relation                <-----INGRES prints these
                                      headings;
¦enm          ¦                <-----please ignore them.
¦-------------¦
¦buildings    ¦
¦departments  ¦
¦employees    ¦
¦jobs         ¦
¦-------------¦
```

```
Select operation:
0 =print explanation            1 =define entity set(s)
2 =delete entity set(s)         3 =change name of entity set(s)
4 =list entity sets

operation?                      <----You are satisfied with your
                                     choices, so you press return.
```

You would then return to the menu of steps, and continue with step 2.

STEP 2: CHOOSE RELATIONSHIPS

2.1. INTRODUCTION

The second step is to identify the ways in which your entity sets are related; this leads to choosing the relationships for your intermediate design. Suppose employees are assigned to jobs; then you want to choose a relationship 'assignment', which relates entity sets 'employees' and 'jobs'. If employees can be qualified for different jobs, independent of what the current assignments are, then you will choose 'qualified' to be another relationship between these two entity sets.

Please do not confuse relationships with relations. A relationship is part of your intermediate design; it indicates how certain entity sets are related. A relation is a table of data in an INGRES database.

TAGS: Sometimes an entity set plays more than one role in a relationship. For example, in the relationship 'manager', you must distinguish between the two roles that the 'employees' entity set takes on: the role of the manager/supervisor and that of the employee being supervised. As you will see in the example, the design-aid prompts you to choose a three letter 'tag' to identify each role. Thus 'manager' involves 'employees' (tag='sup') for the supervisory role and 'employess' (tag='emp') for the supervised role.

2.2. EXAMPLE

```
Step 2: Choose relationships

Select operation:
0 =print explanation
1 =define relationship(s) or          2 =delete relationship(s) or
   add entity set(s) to rel.             remove entity set(s) from rel.
3 =change name of relationship        4 =list relationships
5 =list relationships that            6 =list entity sets
   entity set appears in

operation? 1                                        <DEFINE>

Define relationship(s) or add entity set(s) to relationship:

relationship? assignment
```

```
Add entity set(s) to relationship 'assignment':

entity set? employees
entity set? jobs
entity set?


relationship? qualified

Add entity set(s) to relationship 'qualified':

entity set? employees
entity set? jobs
entity set?

relationship? manager

Add entity set(s) to relationship 'manager':

entity set? employees
entity set? employees

Entity set 'employees' already appears once in
relationship 'manager'.
Do you want it to appear twice?
y or n? y

Each usage will need a tag of 3 or fewer characters,
to distinguish it from the other.
tag1? sup
tag2? emp

entity set?

relationship? works_in

Add entity set(s) to relationship 'works_in':

entity set? employees
entity set? jobs
entity set?                  <----You realize that you really want
                                  'departments' instead of 'jobs'.
relationship?


Select operation:
0 =print explanation
1 =define relationship(s) or          2 =delete relationship(s) or
   add entity set(s) to rel.             remove entity set(s) from rel.
3 =change name of relationship        4 =list relationships
5 =list relationships that            6 =list entity sets
   entity set appears in
```

```
operation? 2                                        <DELETE>

Delete relationship(s) or remove entity set(s) from relationship:

relationship? works_in

Do you want to delete the whole of relationship 'works_in'?
y or n? n                        <----You only want to remove 'jobs'.

Remove entity set(s) from relationship 'works_in':
entity set? jobs
tag (press return if no tag)?
entity set?

relationship?

Select operation:
0 =print explanation
1 =define relationship(s) or          2 =delete relationship(s) or
   add entity set(s) to rel.             remove entity set(s) from rel.
3 =change name of relationship        4 =list relationships
5 =list relationships that            6 =list entity sets
   entity set appears in

operation? 1                                        <DEFINE>

Define relationship(s) or add entity set(s) to relationship:

relationship? works_in

Note that relationship 'works_in' is already defined.
Add entity set(s) to relationship 'works_in':

entity set? departments
entity set?                      <----Now you have fixed the mistake.

relationship? depthead

Add entity set(s) to relationship 'depthead':

entity set? departments
entity set? employees
entity set?

relationship? allocation

Add entity set(s) to relationship 'allocation':

entity set? departments
entity set? jobs
entity set?
```

```
relationship? location

Add entity set(s) to relationship 'location':

entity set? departments
entity set? buildings
entity set?


relationship?

Select operation:
0 =print explanation
1 =define relationship(s) or          2 =delete relationship(s) or
   add entity set(s) to rel.             remove entity set(s) from rel.
3 =change name of relationship        4 =list relationships
5 =list relationships that            6 =list entity sets
   entity set appears in

operation? 4                                        <LIST>

List of relationships:
Relationship  Entity set   Tag

rellist relation                        <----INGRES prints these
                                             headings;
!relnm        !entnm        !tag    ! <----please ignore them.
!-----------------------------------!
!allocation   !departments  !       !
!allocation   !jobs         !       !
!assignment   !employees    !       !
!assignment   !jobs         !       !
!depthead     !departments  !       !
!depthead     !employees    !       !
!location     !buildings    !       !
!location     !departments  !       !
!manager      !employees    !emp    !
!manager      !employees    !sup    !
!qualified    !employees    !       !
!qualified    !jobs         !       !
!works_in     !departments  !       !
!works_in     !employees    !       !
!-----------------------------------!


Select operation:
0 =print explanation
1 =define relationship(s) or          2 =delete relationship(s) or
   add entity set(s) to rel.             remove entity set(s) from rel.
3 =change name of relationship        4 =list relationships
5 =list relationships that            6 =list entity sets
   entity set appears in
```

operation?

You would then return to the menu of steps.

2.3. FURTHER COMMENTS

Two or more entity sets may participate in a relationship. An entity set must be defined before you add it to a relationship; if you try to add an undefined entity set, the design aid will print a message and allow you to define it (without going back to step 1). You must go back to step 1 to delete an entity set, and first you must delete any relationships that it participates in (or change the relationships so they no longer involve that entity set); if you forget to do this, the design-aid will remind you.

ASSUMPTION: In order for the resulting schema to be normalized, the relationships you choose must be 'independent' and 'indecomposable'. Usually this will be true without any fuss, so only a brief definition is given here. An independent relationship represents a fact that is not represented by any other relationship (or combination of relationships) in your design; 'assignment' represents the fact that employees are assigned to jobs, which is not represented in any other way. An indecompsable relationship is one which you cannot break down into other relationships without loosing information. You cannot break 'assignment' down any further.

STEP 3: CHOOSE SUBSCHEMAS

3.1. INTRODUCTION

In step 3 you may partition your design into subschemas. A subschema is a group of entity sets and relationships that you feel belong together in some way. If you have a large intermediate design, choosing subschemas will help you keep its meaning clear. Subschemas may also be used in physical design to improve performance.

If you decide to have subschemas, you must put each entity set and each relationship into exactly one subschema. Furthermore, the subschemas must fulfill a closure property: if all the entity sets participating in one relationship are in one subschema, you must put that relationship in the same subschema. The 'check closure' operation will check this property, and your design must pass this check before you can do the analysis in step 5. If a relationship involves entity sets from two or more subschemas, you may put it in any one of those subschemas.

3.2. EXAMPLE

If the entity sets and relationships defined so far were the only ones in the design, there would be no reason for choosing subshemas. However, if they were part of a larger design, you might have a subschema having to do with employees:

```
Step 3: Choose subschemas

Do you want to partition your entity sets and relationships
into subschemas? (For explanation, press return.)
y, n or return? y

Please remember that in this step you are only partitioning
(grouping) the entity sets and relationships--nothing you do
in this step will define new entity sets or relationships, or
delete old ones from your design.

Select operation:
0 =print explanation
1 =set up or add to
   subschema(s)                    2 =eliminate or remove from
                                      subschema(s)
3 =change name of subschema(s)     4 =list (all/some) subschemas
5 =check closure                   6 =list relationships
7 =list entity sets
```

```
operation? 1                                    <SET UP SUBSCHEMA>

Set up subschema(s) and/or add entity sets and/or add
relationships:

subschema? emps

Do you want to add entity sets or relationships first?
e or r (press return if neither)? e
                                                <ADD ENTITY SETS>
Add entity set(s) to subschema 'emps':

entity set? employees
entity set? departments
entity set? jobs
entity set?

Do you want to add entity sets or relationships next?
e or r (press return if neither)? r
                                                <ADD RELATIONSHIPS>
Add relationship(s) to subschema 'emps':

relationship? assignment
relationship? qualified
relationship? manager
relationship? works_in
relationship? depthead
relationship? allocation
relationship? location
relationship?

Do you want to add entity sets or relationships next?
e or r (press return if neither)?

subschema?

Select operation:
0 =print explanation
1 =set up or add to                 2 =cancel or remove from
   subschema(s)                        subschema(s)
3 =change name of subschema(s)      4 =list (all/some) subschemas
5 =check closure                    6 =list relationships
7 =list entity sets

operation? 4                                        <LIST>

Please choose one of four list-subschema options:
1 =list just the names of all subschemas
2 =list all subschemas
3 =list individual subschema(s)
4 =list those entity sets and relationships not in a subschema
Return takes you back to the menu of subschema operations
1, 2, 3 or 4? 3
```

```
List individual subschema(s):

subschema? emps

Entity sets in subschema 'emps':

entlist relation

¦enm          ¦
¦-------------¦
¦departments  ¦
¦employees    ¦
¦jobs         ¦
¦-------------¦

Relationships in subschema 'emps':
Relationship  Entity set   Tag

rellist relation

¦relnm        ¦entnm        ¦tag    ¦
¦-------------------------------------¦
¦allocation   ¦departments  ¦       ¦
¦allocation   ¦jobs         ¦       ¦
¦assignment   ¦employees    ¦       ¦
¦assignment   ¦jobs         ¦       ¦
¦depthead     ¦departments  ¦       ¦
¦depthead     ¦employees    ¦       ¦
¦location     ¦buildings    ¦       ¦
¦location     ¦departments  ¦       ¦
¦manager      ¦employees    ¦emp    ¦
¦manager      ¦employees    ¦sup    ¦
¦qualified    ¦employees    ¦       ¦
¦qualified    ¦jobs         ¦       ¦
¦works_in     ¦departments  ¦       ¦
¦works_in     ¦employees    ¦       ¦
¦-------------------------------------¦

subschema?                    <----Pressing return 'backs you up'.

Please choose one of four list-subschema options:
1 =list just the names of all subschemas
2 =list all subschemas
3 =list individual subschema(s)
4 =list those entity sets and relationships not in a subschema
Return takes you back to the menu of subschema operations
1, 2, 3 or 4?
```

```
Select operation:
0 =print explanation
1 =set up or add to
   subschema(s)
2 =cancel or remove from
    subschema(s)
3 =change name of subschema(s)
4 =list (all/some) subschemas
5 =check closure
6 =list relationships
7 =list entity sets

operation?
```

At this point you would choose the other subschemas in your design, and then do operation 5 (=check closure).

STEP 4: CHOOSE CONSTRAINTS

4.1. INTRODUCTION

Suppose you know that each employee is assigned to exactly one job at any given time. You can put this information into your intermediate design by specifying that each 'employees' entity must participate at least once and at most once in the 'assignment' relationship, thereby being assigned to exactly one job.

You can choose to place either one or both of these two constraints (AT LEAST ONCE or AT MOST ONCE) on the role of any entity set in a relationship. However, you may not place the AT LEAST ONCE constraint on more than one entity set in a relationship, for a reason which will be explained in section 4.3.

When you list your relationships with the constraints, you will see that 't' and 's' appear in the heading printed by INGRES, underneath the AT LEAST ONCE and AT MOST ONCE column titles. The AT LEAST ONCE constraint is also called 'total', because all entities in the constrained entity set must appear in the relationship. The AT MOST ONCE constraint is also called 'single-valued', because each entity in the constrained entity set has at most a single appearance in the relationship.

4.2. EXAMPLE

Suppose you know that:
1) each employee is assigned to exactly one job at all times
2) only one employee can be assigned to a job
3) each employee works in exactly one department at all times
4) each employee has exactly one manager
5) there is only one department head for a department
6) each department must be located in at least one building

You could chose operation 1, which would prompt you for the constraints on each entity set in each relationship. However, suppose you decide to use operation 2. (All constraints start as 'no', so you can just modify the ones you want to make 'yes'). Step 4 might look like this:

```
Step 4: Choose constraints

Select operation:
0 =print explanation
1 =sequence through                    2 =modify constraints on
   relationships, choosing                selected relationship(s)
   constraints
3 =list relationships (with            4 =list relationships that
   constraints)                           selected entity set
                                          appears in (with cons.)

operation? 2                                    <MODIFY>


Modify constraints for selected relationship(s):

relationship? assignment

Modify constraints on entity set(s) in relationship 'assignment':
entity set? employees
tag (press return if no tag)?

Must each 'employees' (tag='') entity
participate AT LEAST ONCE in the 'assignment' relationship?
y or n? y                          <----From (1).

May each 'employees' (tag='') entity
participate AT MOST ONCE in the 'assignment' relationship?
y or n? y                          <----From (1).

entity set? jobs
tag (press return if no tag)?

Must each 'jobs' (tag='') entity
participate AT LEAST ONCE in the 'assignment' relationship?
y or n? n

May each 'jobs' (tag='') entity
participate AT MOST ONCE in the 'assignment' relationship?
y or n? y                          <----From (2).

entity set?

relationship? works_in

Modify constraints on entity set(s) in relationship 'works_in':
entity set? employees
tag (press return if no tag)?

Must each 'employees' (tag='') entity
participate AT LEAST ONCE in the 'works_in' relationship?
y or n? y                          <----From (3).
```

```
May each 'employees' (tag='') entity
participate AT MOST ONCE in the 'works_in' relationship?
y or n? y                        <----From (3).

entity set?

relationship? manager

Modify constraints on entity set(s) in relationship 'manager':
entity set? employees
tag (press return if no tag)? emp

Must each 'employees' (tag='emp') entity
participate AT LEAST ONCE in the 'manager' relationship?
y or n? y                        <----From (4).

May each 'employees' (tag='emp') entity
participate AT MOST ONCE in the 'manager' relationship?
y or n? y                        <----From (4).

entity set?

relationship? depthead

Modify constraints on entity set(s) in relationship 'depthead':
entity set? departments
tag (press return if no tag)?

Must each 'departments' (tag='') entity
participate AT LEAST ONCE in the 'depthead' relationship?
y or n? n

May each 'departments' (tag='') entity
participate AT MOST ONCE in the 'depthead' relationship?
y or n? y                        <----From (5).

entity set?

relationship? location

Modify constraints on entity set(s) in relationship 'location':
entity set? departments
tag (press return if no tag)?

Must each 'departments' (tag='') entity
participate AT LEAST ONCE in the 'location' relationship?
y or n? y                        <---From (6).

May each 'departments' (tag='') entity
participate AT MOST ONCE in the 'location' relationship?
y or n? n

entity set?
```

```
relationship?


Select operation:
0 =print explanation
1 =sequence through                    2 =modify constraints on
   relationships, choosing                selected relationship(s)
   constraints
3 =list relationships (with            4 =list relationships that
   constraints)                           selected entity set
                                          appears in (with cons.)

operation? 3                                        <LIST>


List of relationships with constraints:
Relationship  Entity set   Tag     At      At
                                   Least   Most
                                   Once    Once

rellist relation

|relnm        |entnm        |tag    |t      |s      |
|-----------------------------------------------|
|allocation   |departments  |       |n      |n      |
|allocation   |jobs         |       |n      |n      |
|assignment   |employees    |       |y      |y      |
|assignment   |jobs         |       |n      |y      |
|depthead     |departments  |       |n      |y      |
|depthead     |employees    |       |n      |n      |
|location     |buildings    |       |n      |n      |
|location     |departments  |       |y      |n      |
|manager      |employees    |emp    |y      |y      |
|manager      |employees    |sup    |n      |n      |
|qualified    |employees    |       |n      |n      |
|qualified    |jobs         |       |n      |n      |
|works_in     |departments  |       |n      |n      |
|works_in     |employees    |       |y      |y      |
|-----------------------------------------------|

Select operation:
0 =print explanation
1 =sequence through                    2 =modify constraints on
   relationships, choosing                selected relationship(s)
   constraints
3 =list relationships (with            4 =list relationships that
   constraints)                           selected entity set
                                          appears in (with cons.)

operation?
```

This would then return you to the menu of steps.

4.3. FURTHER COMMENTS

The constraints you choose for your intermediate design will have consequences when you come to insert and delete data from the resulting database (presuming you enforce the constraints).

The AT MOST ONCE constraints affect inserting data about the relationships. In the above example, when you want to make Peter Rabbit the head of the accounting department, you must make sure that 'accounting' doesn't already appear in 'depthead'. Similarly, when you want to assign an employee to a job, you must check that neither the employee nor the job appears in 'assignment'.

The AT LEAST ONCE constraints affect inserting and deleting entities. In the above example, consider inserting a new employee, Peter Rabbit. Since each employee must appear AT LEAST ONCE in 'assignment', you must link Peter Rabbit to some job via this relationship. If the job exists, fine; if not, you must insert the new job before you insert the new employee--this is an insert side-effect.

To hire Peter Rabbit, you must also link him to a department via the 'works_in' relationship. This illustrates how the side- effects can ripple: if the department doesn't yet exist, you must insert it before inserting Peter Rabbit, but to insert a department you must link it to a building via 'location'. This seems reasonable, since the building and the department should exist when you hire a new employee.

The AT MOST ONCE and AT LEAST ONCE constraints affect deleting entities. The delete side-effects are essentially the opposite of the insert side-effects, except that they are divided into two cases. In CASE 1, BOTH constraints apply (to an entity set in a relationship), and there WILL be delete side-effects. In CASE 2, only the AT LEAST ONCE constraint applies, and there MAY be delete side-effects. In the example, an employee must be assigned to exactly one job (case 1). If you delete a job, you will have to delete the employee currently assigned to that job. (You may, of course, move the employee to a different job before you do the the delete). Case 2 is similar, but a 'check' is involved. A department must be located in at least one building (case 2). If you delete a building, you may have to delete (or move) some of the departments currently located in it; for each such department, you must check whether this is the only building it is located in, and if so you must delete (or move) that department. Delete side-

effects can also ripple. If deleting a building causes you to delete a department, you must also delete (or move) all employees working in that department.

In this example, the insert and delete side-effects adequately represent the real-world behavior of the application. Sometimes, however, you can choose constraints that seem to fit your data, and later discover from the consequences that you made a poor choice. In step 5, the insert and delete side-effects are analyzed for you, so that you can see the consequences of the constraints you chose; if they do not represent your application correctly, you can return to this step and modify them.

LOOPS: The analysis will also point out any loops in the insertion order. Suppose you had decided that each building must have at least one department located in it, as well as each department being located in some building. Then, to insert a building you would have to link it to a department; but to insert a department you would have to link it to a building. You couldn't begin to enter your data, and therefore loops are illegal. This is why only one entity set can be constrained to participate AT LEAST ONCE in a given relationship.

TAGS: One further caution: when you put constraints on a tagged entity set, you are actually putting the constraints on the whole entity set. Thus, in the example, placing both constraints on the 'employees' (tag='emp') entity set in 'manager' means that EVERY employee entity must participate exactly once in 'manager' (in the 'emp' role); this is just what you want, since every employee must have exactly one manager. Suppose, however, that you wanted instead to put constraints indicating that 'every manager must supervise some employee'. You could not do this without creating a seperate entity set for managers, since placing the AT LEAST ONCE constraint on 'employees' (tag='sup') in 'manager' would mean that EVERY EMPLOYEE must manage someone.

STEP 5: ANALYZE AND DISPLAY

5.1. INTRODUCTION

At this step, the design aid analyzes the insert and delete side-effects for your intermediate design. You can then display the results of this analysis and decide whether they faithfully reflect the real-world consequences of the inserts and deletes.

The analysis also finds those entity sets not used in any relationship and those which form a subset pattern (see example below). These situations are not illegal (unlike loops), but you should check to make sure that they are meaningful.

If you chose subschemas, the display will be arranged by subschema and will point out which side-effects go outside the subschema. You may want to balance your choice of constraints and subschemas so as to minimize the side-effects which go outside a subschema.

5.2. EXAMPLE

In the display, the 'Trace of' column tells you which entity set began the ripple of side-effects, and 'Ripple' tells you the ripple number, i.e. how far from the original insert or delete the ripple has gone. Refer back to the earlier explanation (4.3) for an English description of the side-effects displayed here. Begining at the step 5 menu:

```
Select operation:
0 =print explanation          1 =display whole analysis
2 =display for selected       3 =display for insert/delete
   subschema(s)                  of selected entity (entities)
4 =display entity sets
   not used

operation? 1                                  <DISPLAY WHOLE ANALYSIS>
                                                    <SEE 4.3>

Display whole analysis:

Entity sets not used in any relationship:

notused relation

¦enm          ¦
¦-------------¦
¦-------------¦
```

<SUBSET>

Subset pattern--when one entity set is constrained to participate exactly once (AT LEAST ONCE and AT MOST ONCE) in a relationship, and another entity set is constrained to participate AT MOST ONCE, the first set is essentially a subset of the second. The following entity sets show this pattern.

Entity set (Tag) Relationship Entity set (Tag)

subset relation

doment	domta	rel	rangent	rangt
employees		assignment	jobs	

Display of insert side-effects (ordering): <INSERTS>

Trace of Ripple To add (Tag) Must link to (Tag) Via

projinsert relation

traceof	ripnum	toadd	adtag	linkto	lntag	via
departments	1	departments		buildings		location
employees	1	employees		departments		works_in
employees	1	employees		employees	sup	manager
employees	1	employees		jobs		assignment
employees	2	departments		buildings		location

Display of delete side-effects (further deletes): <DELETES>

Trace of Ripple To delete (Tag) Will/may delete (Tag) Via

projdel relation

traceof	ripnum	todel	dltag	w_m	w_mdel	wltag	via
buildings	1	buildings		may	departments		location
buildings	2	departments		will	employees		works_in
departments	1	departments		will	employees		works_in
employees	1	employees	sup	will	employees		manager
jobs	1	jobs		will	employees		assignment

```
Select operation:
0 =print explanation          1 =display whole analysis
2 =display for selected       3 =display for insert or delete
   subschema(s)                  of selected entity (entities)
4 =display entity sets
   not used

operation?
```

You would then return to the menu of steps, and go back to step 4 (if you weren't satisfied) or on to step 6.

STEP 6: CHOOSE ATTRIBUTES

6.1. INTRODUCTION

When you decide that your choice of entity sets, relationships and constraints faithfully reflects the meaning of your data, you are almost ready to produce the design. All that is needed is to choose the attributes of your entity sets, that is, the domain-name and format for each piece of data you want to record for each entity set. (See 'The INGRES Reference Manual', under QUEL, for a description of the allowable formats). Each entity set must have one attribute--its key or identifying attribute--which will have a unique value for every entity.

You can also choose attributes for relationships, but not the key attributes. The key of a relationship will be made up of the key attributes from the participating entity sets.

6.2. EXAMPLE

```
Step 6: Choose attributes

Select operation:
0 =print explanation
1 =sequence through entity sets, defining attribute(s)
2 =define attribute(s) for entity set or relationship
3 =delete attribute(s) of entity set or relationship
4 =list entity sets that have attributes
5 =list relationships that have attributes
6 =check keys (will list entity sets without keys)

operation? 1                                  <SEQUENCE THROUGH >

Sequence through entity sets, defining attributes:

Warning: the key attributes are used (together) as a surrogate
for an entity, so it is strongly suggested that you use only
one attribute as a key (instead of two or more). Note that you
must designate at least one attribute as a key; the 'check keys'
operation on the menu checks that each entity set has a key
attribute, and this check must be passed before the design will
be produced.

The allowable formats for attributes are: c1 to c255, i1, i2,
i4, f4 or f8.
See the INGRES Reference Manual for more details.
```

```
Define key or non-key attributes for entity set 'employees':

attribute? empno
format? i2
Is 'empno' a key (identifying) attribute for entity set 'employees'?
y or n? y


attribute? name
format? c15
Is 'name' a key (identifying) attribute for entity set 'employees'?
y or n? n

attribute? salary
format? f4
Is 'salary' a key (identifying) attribute for entity set 'employees'?
y or n? n

attribute?

continue (y or n)? y          <----This question allows you to stop
                                    the sequence if you want to.

Define key or non-key attributes for entity set 'buildings':

attribute? siteno
format? i2
Is 'siteno' a key (identifying) attribute for entity set 'buildings'?
y or n? y

attribute? capacity
format? i2
Is 'capacity' a key (identifying) attribute for entity set 'buildings'?
y or n? n

attribute? address
format? c30
Is 'address' a key (identifying) attribute for entity set 'buildings'?
y or n? n

attribute?

continue (y or n)? y

Define key or non-key attributes for entity set 'departments':

attribute? deptno
format? i2
Is 'deptno' a key (identifying) attribute for entity set 'departments'?
y or n? y
```

```
attribute? name
format? c10
Is 'name' a key (identifying) attribute for entity set 'departments'?
y or n? n


attribute?

continue (y or n)? y

Define key or non-key attributes for entity set 'jobs':

attribute? jobno
format? i2
Is 'jobno' a key (identifying) attribute for entity set 'jobs'?
y or n? y

attribute? position
format? c15
Is 'position' a key (identifying) attribute for entity set 'jobs'?
y or n? n

attribute? hours
format? c10
Is 'hours' a key (identifying) attribute for entity set 'jobs'?
y or n? n

attribute?
                                <----You have completed the sequence.

Select operation:
0 =print explanation
1 =sequence through entity          2 =define attribute(s) for
   sets, defining attribute(s)         entity
 set or relationship
3 =delete attribute(s) of           4 =list entity sets that have
   entity set or relationship          attributes
5 =list relationships that have     6 =check keys (will list
   attributes                          entity sets without keys)

operation? 4                                  <LIST>
```

```
List of entity sets that have attributes:
Entity set     Attribute     Format Key

entatlist relation

+esetnm       +eatnm         +eatfor+key    +
+-------------------------------------------+
+buildings    +address       +c30   +n      +
+buildings    +capacity      +i2    +n      +
+buildings    +siteno        +i2    +y      +
+departments  +deptno        +i2    +y      +
+departments  +name          +c10   +n      +
+employees    +empno         +i2    +y      +
+employees    +name          +c15   +n      +
+employees    +salary        +f4    +n      +
+jobs         +hours         +c10   +n      +
+jobs         +jobno         +i2    +y      +
+jobs         +position      +c15   +n      +
+-------------------------------------------+

Select operation:
0 =print explanation
1 =sequence through entity          2 =define attribute(s) for
   sets, defining attribute(s)         entity set or relationship
3 =delete attribute(s) of           4 =list entity sets that have
   entity set or relationship          attributes
5 =list relationships that have     6 =check keys (will list
   attributes                          entity sets without keys)

Pressing return key within an operation will return you to
this menu.  Pressing return key now will return you to the
menu of steps.
operation? 2                                    <DEFINE>

Define attribute(s) for selected entity sets and/or relationships:

The allowable formats for attributes are: c1 to c255, i1, i2,
i4, f4 or f8.
See the INGRES Reference Manual for more details.


Do you want to define attributes for entity sets or relationships first?
e or r (press return if neither)? r          <FOR RELATIONSHIPS>

Define attributes for relationship(s):
```

```
relationship? depthead

Define attributes for relationship 'depthead':

rellist relation

|relnm       |entnm        |tag   |
|-------------------------------|
|depthead    |departments  |      |    <----This reminds you which
|depthead    |employees    |      |         entity sets are involved
|-------------------------------|         in 'depthead'.

attribute? years
format? i1
                    <----Note that you cannot choose keys.
attribute?


relationship?

Do you want to define attributes for entity sets or relationships next?
e or r (press return if neither)?


Select operation:
0 =print explanation
1 =sequence through entity          2 =define attribute(s) for
   sets, defining attribute(s)         entity set or relationship
3 =delete attribute(s) of           4 =list entity sets that have
   entity set or relationship          attributes
5 =list relationships that have     6 =check keys (will list
   attributes                          entity sets without keys)

operation? 6                                  <CHECK KEY>,

Check keys:

Key check succeeds; all entity sets have key attributes.

Checking for compound keys...
                                 <----You have no compound keys.


Select operation:
0 =print explanation
1 =sequence through entity          2 =define attribute(s) for
   sets, defining attribute(s)         entity set or relationship
3 =delete attribute(s) of           4 =list entity sets that have
   entity set or relationship          attributes
5 =list relationships that have     6 =check keys (will list
   attributes                          entity sets without keys)

operation?
```

Now you would be ready to produce the design!

STEP 7: PRODUCE DESIGN AND QUIT

7.1. INTRODUCTION

When you are satisfied with your intermediate design, select this step to terminate the design and produce two files: one with a listing of your intermediate design and the resulting schema, and one with 'create' statements for the relations in the schema. To use the second file, create your database using the Unix commands 'createdb' and 'sysmod'. Then call INGRES, giving it this database name. After the asterisk prompt appears, type 'i filename' with the name of the create-statements file. This will cause INGRES to create the relations. You can then enter and modify your data.

In the schema, you will see a heading 'Foreign Key (of Relationship)'. 'Foreign key' means that an attribute which is a key in one relation is being used in a different relation (forming a connection between the two). In the example below, the key attribute of 'jobs' is 'jobno'. 'Employees' has an attribute 'assignment' which matches 'jobno' in format. The 'Foreign Key' column tells you that, for each tuple in the 'employees' relation, the value in the domain 'assignment' should be a job number which relates this tuple to a tuple in the jobs relation. Thus you could run the following Quel query on your database to get a list of your employees with their current jobs.

```
range of e is employees
range of j is jobs
retrieve into emplist (e.empno, e.name, currentjob = j.position)
                where (e.assignment = j.jobno)
```

7.2. EXAMPLE

This is how the design/schema file appears for the example of the other steps:

```
Intermediate Design


Entity sets:

employees
buildings
departments
jobs
```

Relationships (with constraints):

Relationship	Entity set	(Tag)	At Least Once	At Most Once
assignment	employees		y	y
assignment	jobs		n	y
qualified	employees		n	n
qualified	jobs		n	n
depthead	departments		n	y
depthead	employees		n	n
works_in	employees		y	y
works_in	departments		n	n
manager	employees	emp	y	y
manager	employees	sup	n	n
allocation	departments		n	n
allocation	jobs		n	n
location	departments		y	n
location	buildings		n	n

Schema

Relation: employees

Domain	Format	Key	Foreign Key	(of Relation)	
empno	i2	yes			
name	c15	no			
salary	f4	no			
assignment	i2	no	jobno	jobs	<----See
manager_sup	i2	no	empno	employees	note
works_in	i2	no	deptno	departments	above.

Relation: buildings

Domain	Format	Key	Foreign Key	(of Relation)
siteno	i2	yes		
capacity	i2	no		
address	c30	no		

Relation: departments

Domain	Format	Key	Foreign Key	(of Relation)
deptno	i2	yes		
name	c10	no		

Relation: jobs

Domain	Format	Key	Foreign Key	(of Relation)
jobno	i2	yes		
position	c15	no		
hours	c10	no		

```
Relation:  qualified
Domain          Format  Key
empno           i2      yes
jobno           i2      yes

Relation:  depthead
Domain          Format  Key
deptno          i2      yes
empno           i2      yes
depthe_years    i2      no

Relation:  allocation
Domain          Format  Key
deptno          i2      yes
jobno           i2      yes

Relation:  location
Domain          Format  Key
deptno          i2      yes
siteno          i2      yes
```

7.3. FURTHER COMMENTS

WARNING: Occasionally duplicate domain names are generated in a single relation. For example, suppose that each employee was assigned to an office as well as a job; then you might see the following in the schema file:

```
Relation:  employees
Domain          Format  Key   Foreign Key   (of Relation)
empno           i2      yes
name            c15     no
salary          f4      no
assignment      i2      no    jobno         jobs
assignment      i2      no    rmno          offices
manager_sup     i2      no    empno         employees
works_in        i2      no    deptno        departments
```

Duplicate domain names within a single relation are illegal in INGRES, so please check for them in the schema file and change one of the names. (Remember to change the names in the file of 'create' statements also).

Also note that any entity set with a compound key (more than one key attribute) requires multiple domains to represent it in a relation--this may be awkward and is not recommended.

At this point, you can modify your design by calling 'designdb' again, or change any names that seem unclear by editing the schema and 'create' files. If you are satisfied with the design, then you are ready to create your database, create your relations using the 'create' file, and enter data!