

# 2

---

*Understanding the electronic filing cabinet*

## The Database

### What is an IMAGE Database?

The HP3000, with its IMAGE<sup>†</sup> databases, was bought to replace steel filing cabinets and paper records. The advantage that accrues in making your records electronic is the ease with which you can now search, collate and extract information. You now have the capability to generate analytical summaries which you never would have previously done using the paper records, simply due to the magnitude of the effort required.

IMAGE is a particularly simple, but powerfully and efficiently implemented database. IMAGE was designed from its beginnings to be the *image* of a steel file cabinet (hence its name). QueryCalc extends this basic idea as a combination of two procedures that are common to every office: filing cabinets and spreadsheets. The filing cabinets and spreadsheets have simply become electronic. Even so, procedures remain unchanged from those you would normally perform if you were to gather up the information from the paper records. With QueryCalc, you are going to flip through the records in filing cabinets, calculate sums and averages, and enter that information directly onto the spreadsheet, creating complex summary reports rather easily.

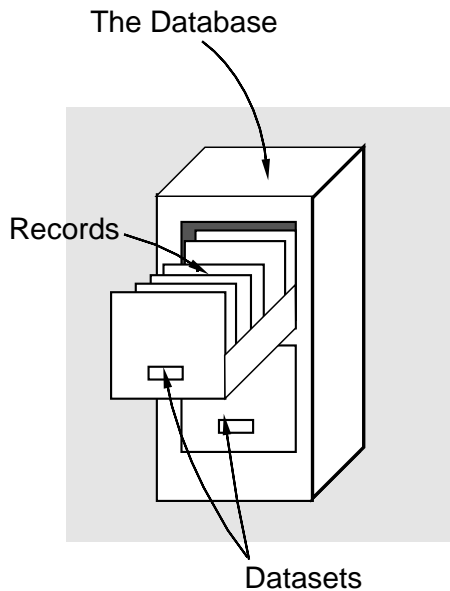
So, if QueryCalc is to be used effectively, it becomes important to understand how the filing cabinet is put together. To do this, we'll use the training database that was supplied with your copy of QueryCalc. This training database is a real database, containing 10,000 invoices and 6,000 labor tickets. The database was supplied by one of QueryCalc's user organizations, a construction company. The names of individual people, projects and companies have been altered, but otherwise, it's real.

---

<sup>†</sup>IMAGE is not the only database structure found on the HP3000, but it is the one most commonly used. Other common databases which QueryCalc supports are KSAM and MPE files (often called *flat files*). All database structures are fundamentally the same. And all database queries are identical in QueryCalc. Because IMAGE is the most commonly used database, this chapter emphasizes IMAGE. But, if you understand IMAGE, you automatically have a good feel for the others.

## Visualizing a Database

A *database* is the electronic equivalent of a filing cabinet. Each of its individual drawers are *datasets*. The names come from mathematical set theory, but the names are generally inconsequential. It would have been just as easy to call the database the filing cabinet and the datasets filing drawers. If IMAGE had been designed in today's "user-friendly" climate, the names would probably be different. But IMAGE was designed in the early 1970's



when the principal users were data processing professionals with substantial mathematical training. Databases and datasets were appropriate names to their education and backgrounds. We want you to think of IMAGE as a steel filing cabinet filled with drawers.

What should you expect to find when you open a file drawer (a dataset)? Hundreds of manila folders. In IMAGE, the folders are called *records*. An electronic database differs from a paper-filled cabinet drawer in one very particular way. In any single real cabinet drawer you may have a mix of folder types: some personnel records, some invoices, and anything else that might fit in the drawer. That won't happen in a database. An electronic file drawer (a dataset)

can contain only one type of record. The whole drawer must be dedicated to invoices (or payroll checks or employee records). The dataset (the drawer) may be very large or it may be very small, but it will always be consistent and uniform.

This same requirement of uniformity holds true for what's written into each manila folder (record). With paper records, you have the capacity to write information in the margins, scribbling notes anywhere where you can find room. Again, that can't happen in an electronic record. You can only write or read information from lines previously specified on the record. Every record in the dataset (the drawer) is identical in format. It's what's entered in those specific lines that makes up your database information. These line-by-line entries are called *dataitems* in IMAGE.

Who decided what dataitems make up a record? Quite often, it's you—or someone just like you sometime in the past. The electronic record in the dataset is generally no more than what you would have typed onto a paper

record if it had been up to you to generate a form. Presume that you are in charge of a construction company and it's your responsibility to generate a labor ticket form for the employees to fill out daily. What would you make the labor ticket look like if it had to be done on paper?

Quite likely, you would choose something like this:

```

                                EMPLOYEE LABOR TICKET
Name: _____
Date: _____
Job Number: _____
Regular hours: _____
Overtime hours: _____
    
```

## Designing the Database

The electronic version will be virtually identical, but with one important difference. Space in a computer is generally expensive. Text almost always requires more room than a number. Storing a person's name as text has two principal disadvantages. The first is the waste of space in storing a complete name where a number (employee number, social security number) would do. And the second is the ambiguity associated with names. Any one person can have a variety of things he or she may be called (Mary Smith, M.B. Smith, etc.). So rather than use names, it is almost always preferable to use a number to identify a person.

Thus, the computer-equivalent record would probably look like this:

```

                                LABOR DATASET
SOCSECNUM:
DATE:
JOBNUM:
REGULAR:
OVERTIME:

                                CAPACITY:10000    ENTRIES: 5792
    
```

While this looks a little computer-*ish*, the reasons for making it so are straightforward. Social Security Number, which now takes the place of the person's name, has been reduced to SOCSECNUM for three reasons: (1) spaces aren't allowed in a dataitem's name, (2) no dataitem name can be

longer than 16 characters, and (3) it's just simply easier to type. Other than that, the form of the record is identical to the paper form. This one form will be duplicated in the LABOR file drawer (dataset) perhaps 1,000 times, 10,000 times or possibly 100,000 times, depending on how many records you think you'll need to serve the time period the records should cover. The number of potential records in a dataset is called its *capacity*. The capacity of a dataset is not a fixed amount; it can be changed as more space is required.

## Retrieving Information from the Database

An IMAGE database (the filing cabinet) may have up to 199 separate datasets (drawers) in it, each with its own record format. For a construction company, other datasets we would need are for JOBS-IN-PROGRESS, EMPLOYEE RECORDS, and INVOICES.

Presume that current information fills the database. How would you find something in the database? Let's say the question you need to ask is:

*What was the sum of all invoices charged to job 8404  
during the first half of 1989?*

The important question to ask is: how would you go about obtaining the information if all of the records were in paper form? You would open the file drawer labeled INVOICES, find the invoices for job 8404, flip through them and add up the total charges for those with dates which properly qualify. That's exactly what you're going to do with QueryCalc and IMAGE too. The question in QueryCalc looks like this:

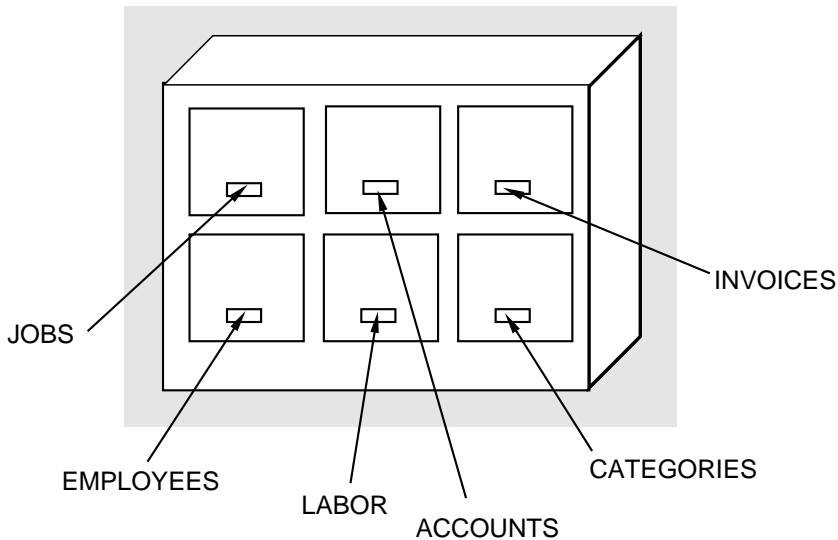
*@Using INVOICES, sum of AMOUNT when  
JOBNUM is 8404 and DATE is 890101,890630*

The question (called a *database query*) is written in a stylized English form both you and the computer can understand. It will always look something like this one, although you have a great many options in what you can ask.

The standard QueryCalc sentence will always occur in three basic parts: (1) the "*Using...*" phrase specifies which dataset (and optionally database) you wish to search through, (2) the "*sum of...*" phrase specifies what information you would like to retrieve, and (3) the "*when...*" phrase specifies the conditions that must be satisfied before the record is incorporated into whatever calculation is requested. As you'll see in a following chapter, the "*Using...*" phrase is not always necessary. QueryCalc can often deduce the

proper dataset (file drawer) from which to retrieve the information without it being fully specified.

The database QCDEMO and some of its datasets



An actual database is likely to have more than a few datasets. Indeed, a database will probably have more file drawers than can be easily drawn. Six of the fifteen datasets of the training database, QCDEMO, are shown here. These are among the more important datasets in the database and will figure prominently in the exercises which will appear throughout the remainder of this *Guide*.

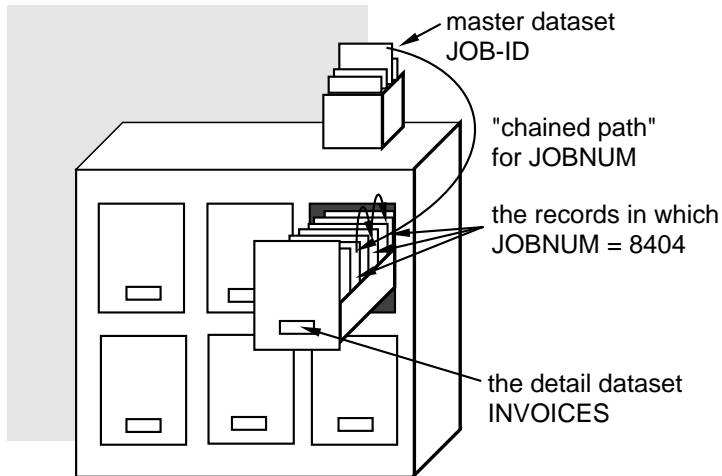
There are two ways to search a dataset for a particular piece of information. The first is straightforward. Open the dataset, begin at

the beginning and read every record, front to back, marking those records which meet the qualifying criteria. This is called a *serial search*. This is a perfectly legitimate way to search a dataset, but it is also the slowest possible way to obtain a summary of qualifying records. If there are a 100,000 records in the dataset, this method may be too time consuming to be acceptable.

## High Speed Searches

The second way to search a dataset for a particular record or set of records uses indexing datasets (called *master datasets*). A master dataset sits on top of the filing cabinet like a file of 3x5 index cards, not unlike the indexing cards in a library file. But in this case, the 3x5 master dataset cards contain record numbers. If we wanted to find all of the invoices in the INVOICES dataset which are associated with the the job 8404, and the dataitem JOBNUM was a *search item*, the master dataset will point to first and last invoices which have JOBNUM = 8404. Each succeeding invoice itself will point to next and previous records which also have JOBNUM = 8404, thus forming a *chain*. The first record obviously can't point to a previous record, so it becomes the *chain head*. Likewise, the last record is the *chain tail*. Now, instead of searching the entire dataset, record by record, we can simply go to the 3x5 index card file, find the record for JOBNUM = 8404,

go to that particular record in INVOICES, starting optionally at either the chain head or the tail, follow the chain and read only those records which have the proper JOBNUM, as shown. In place of reading 10,000 records, we now may have to read only 30 records. This form of search will be



much faster. The moral? *Always use a chained search if possible.* QueryCalc will do this automatically if one of the dataitems used in your query question is a search item. If there are multiple search items in your query question (implying multiple 3x5 card-like master datasets), QueryCalc will choose the chained path with the fewest number of entries. Alternatively, if your query question uses no search items at all, QueryCalc must use a serial search. Query-

Calc will announce that too, specifying how many records will have to *serially* searched, thereby providing you with an estimate of how long the query question is likely to take.

## Selecting the Search Items

There are two kinds of datasets in an IMAGE database, *detail* datasets (the file drawers) and *master* datasets (the 3x5 index cards). And there are two kinds of dataitems. Technically, these are called either *key items* (search items) or *attributes* (non-search items), although these names aren't commonly used in reference to IMAGE databases. A search dataitem is a dataitem which has a chaining path back to a 3x5 card master dataset. The question is: which items should be search items. And which items should not be? The answer partly depends on your use of the database.

Consider the employee LABOR tickets dataset that we just constructed. Which items are good candidates for search items? The better way to ask that question is: how many ways are you likely to want to retrieve information from the LABOR dataset? Certainly, you would like to know the sum of hours that a particular person has worked. That means that SOCSECNUM must be a search item. You would also like summaries of all of the

labor tickets charged to a particular job, thus JOBNUM must also be a search item. Therefore, we can be sure that these two items on the LABOR dataset need to be *search* items. Are there any others?

## LABOR DATASET

```

SOCSECNUM:      [search item]
    DATE:
    JOBNUM:      [search item]
    REGULAR:
    OVERTIME:

```

CAPACITY:10000    ENTRIES: 5792

Why shouldn't the data items for regular and overtime hours also be search items? Because you are rarely going to need to search for all of the records that have 13.1 hours of regular labor recorded in them. It's not the kind of question that you would normally ask. The only items that should be search items are those items that have some definite value, not an indeterminate value like 13.1. What about DATE then? It certainly has definite values. Should it be a search item? Possibly. Whether it should be or not wholly depends on what kind of questions you are likely to ask.

## Penalties Associated with Search Items

Are there penalties associated with making some items search items and others not? Yes, there are several, but they are generally small. IMAGE will only allow 16 items in any one dataset to be search items. This limit is not a severe restriction however. Good database design practice would generally not have more than about four, five or six search items in a single dataset. Each additional search item in a dataset requires that the dataset take up a bit more space on the disc; so there is a small penalty paid for disc space. A slightly more substantial penalty is that new entries placed into a dataset require greater processing time than they otherwise would. Existing search chains must be altered for each newly inserted dataset record. Or, alternatively, if a chain for the newly entered value does not already exist, a chain for the new value must be created. Again, these are generally not severe restrictions. The final penalty can occasionally be the most bothersome. A search item is a *key* item in the database and IMAGE will not allow key items to be modified or updated the way that non-key (*attribute*) items can be. The only easy way to modify a key item is to delete the entire record and re-enter it, thus automatically rebuilding all of the necessary chains.

*Search items will never be beneficial when entering data into a database. In general, they can only slow data entry down. But they will be invaluable when extracting and summarizing information from the database.* All report generators use available chained search paths, if possible. QueryCalc makes especially good use of the available chains. When, then, should search chains be defined? Fred White, one of the original members of the HP design team for IMAGE and now associated with Adager, offers these rules:

## Fred White's Rules for Chained Search Paths

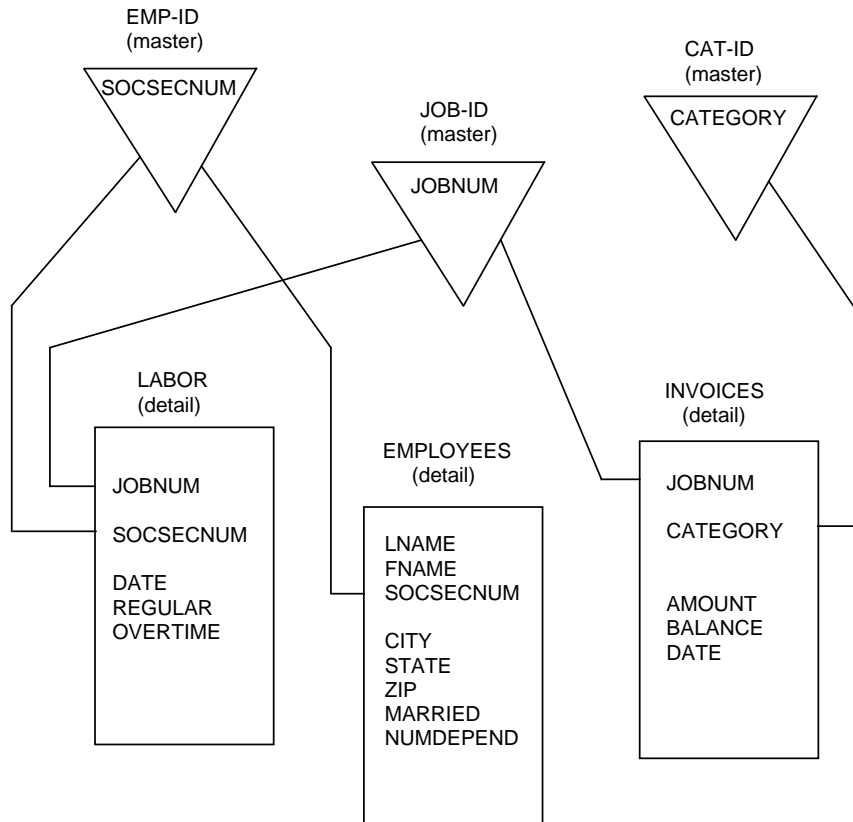
A chained path should be defined if:

- *it is necessary for the application,*
- *its speed of access is better than a serial search and it's frequently used, or*
- *its speed of access is so much better than a serial search that it's cost effective even if it's seldom used.*

If you are responsible for participating in the initial design of the database, the only question that usually arises concerns data items such as the DATE. Should DATE be a chained search item or not? Will it be used often enough to warrant a chain? Fred White's conclusion is: "*when in doubt, leave it out*". If the omission of a search item becomes a burden because of long serial search times, or searches are performed on the database in a manner not originally anticipated, that fact will become increasingly obvious. It's possible add a search path to an IMAGE database by using any one of the available products which restructure an IMAGE database (e.g., ADAGER from Adager, DBGENERAL from Bradmark Computer Systems, or DBCHANGE from Hewlett-Packard). But we politely disagree with Fred, who is otherwise a good friend. Our own experience suggests that if you have doubts, the item should be made a search item now rather than later.

## Schematizing The Database

An IMAGE database can quickly grow so complicated that attempting to draw it as a filing cabinet would become fairly messy. Thus, if you wish to draw the database, it becomes necessary to schematize it in the fashion shown on the facing page. Using standard symbols, master datasets (the 3x5 indexing cards) are drawn as triangles. Detail datasets (the file drawers) are drawn either as trapezoids (Hewlett-Packard's preferred notation) or as rectangles, as shown. Chained search paths are drawn as connecting lines from the master datasets to the detail datasets, landing on their respective



search items. But even the figure shown above is only a small fragment of the training database, QCDEMO; not all of the data items are shown for the detail datasets illustrated nor are all of the master datasets and their search paths shown. Most commonly, the structural form of the database is simply printed out rather than drawn.

How can you determine the form of your own databases? That's easily done using the @FORM command in QueryCalc. An examination of the structure of QCDEMO is a worthwhile brief exercise. Quite logically, you will not be able to fully utilize the information contained in your databases until you understand how the databases are constructed.

## A Walking Tour of QCDEMO

To see how QCDEMO, the construction company database, is put together, type the following at your terminal:

```
:HELLO USER.AICS
:RUN QC.QCPROGS.AICS
```

You are now in QueryCalc. Press RETURN to move into the first spreadsheet screen and type:

```
@OPENDB QCDEMO/FRONT
```

to open the database. The database password (FRONT) must be in all caps. To insure that the database was successfully opened, type:

```
@SHOWDB
```

If you have obtained proper entry into the database, QCDEMO will have an arrow pointing at its name, indicating that it is the *default* database (the database that will be chosen should there be several open at a time and you have not specified which you wish to use). *If the open has failed, the most likely cause is that QCDEMO has not yet been entered into QueryCalc's dictionary.* Press RETURN to return to the spreadsheet.

Now we are free to look at QCDEMO. Type:

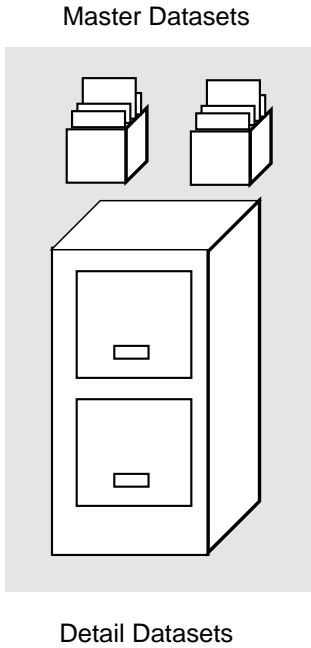
```
@FORM
```

The first screen displays the datasets of QCDEMO and is duplicated on the opposite page. Just as 3x5 index cards rest on top of a filing cabinet, QueryCalc will place the master datasets on top of the detail datasets in the map of datasets. The information displayed is the name of the datasets, their current number of entries and their maximum capacities. Datasets more than 80% full are marked with an asterisk (the 80% full mark is often considered to be the maximum useful density of a dataset). Additionally, if the dataset is a master dataset, @FORM will indicate whether the dataset is a *manual master* or an *automatic master*.

## Automatic vs. Manual Masters

The difference between an automatic and a manual master is illustrated by the following example. If you were entering new employee labor tickets into the LABOR dataset and typed JOBNUM = 8912, and the indexing master dataset for JOBNUM was an *automatic master* and there were no previous entries in the master dataset for 8912, that job number would *automatically* be entered into the master dataset and a chained search path would be immediately established for the new value, 8912. This feature can be either useful or troublesome, depending on your perspective. If 8912 was typed in error, you would never know it. But if the master dataset (JOB-ID) had

SETS OF DATABASE QCDEMO



MASTER DATASETS:		ENTRIES	CAPACITY
CAT-ID	automatic	120	501
SUBCAT-ID	automatic	70	101
JOB-ID	automatic	76	101
ACCOUNT-ID	automatic	483	1001
EMPLOYEE-ID	automatic	185	501
KEYNAME-ID	automatic	1	1 *
DUMMYSORT	automatic	476	1001
NAME-ID	automatic	133	501
DETAIL DATASETS:		ENTRIES	CAPACITY
JOB		53	102
CHANGES		59	78
JOBESTCOST		3598	4572
TAXRATE		55	507
CATEGORIES		120	510
SUBCATS		67	114
EMPLOYEES		181	505
PAYRATE		737	1005
LABOR		5796	7336
PAYRECORD		4051	5137
ACCOUNTS		482	1002
INVOICES		9962	12610
CHECKS		5445	6900
SCHEDULES		0	1004
FIXEDASSETS		82	504

The master and detail datasets of QCDEMO

been constructed as a *manual* master and the value 8912 had not already been entered *manually* into the master, a jobnum of 8912 would be rejected when you attempt entry into the detail dataset. All search item values entered into a detail dataset must be previously entered into their respective manual masters to prevent just this mistake.

Manual masters also have a few other attributes that differentiate them from automatic masters. An automatic master can never have more than one dataitem in it, the search item, and it must have at least one path to one detail dataset. Manual masters, in contrast, may contain many dataitems. And they may be "standalone" datasets (that is, they do not need to be associated with any detail dataset). However, neither form of the master dataset can ever have more than one search (key) dataitem in it..

## Mastering the Details

All of the master datasets in QCDEMO are automatic masters. That's often the case when the data entry (application) programs are written to check the validity of the data before it's entered into the database rather than use value checking features inherent to a manual master.

### FORM OF DATABASE QCDEMO

LABOR [detail dataset]			
ITEMS	DATA	TYPE	
SOCSECNUM	U10	[text]	[search item]
JOBNUM	I1	[num]	[search item]
REGULAR	R2	[num]	
OVERTIME	R2	[num]	
DATE	U6	[text]	[search item]
AMOUNT	R4	[num]	
SUBCAT	I1	[num]	
FICA	R2	[num]	
FUTA	R2	[num]	
NMESC	R2	[num]	
CAPACITY: 7336		ENTRIES: 5796	

### *The LABOR tickets dataset of QCDEMO*

To see what a dataset or dataitem looks like in detail, you can simply type the name of the dataset or dataitem once you are in @FORM. For example, type LABOR. What will appear on your terminal will be a map of the labor tickets dataset, as shown above.

This is the actual form of the LABOR tickets in QCDEMO, not all that different from the one designed a few pages earlier. The three chained search items, SOCSECNUM, JOBNUM, and DATE are marked. The maximum capacity of the dataset and its current entry count are shown, as are the dataitem types for each of the items. Only two types of data are stored in a database: text and numbers. QueryCalc is not particularly sensitive to the manner in which a number is stored. All numbers, regardless of their original data types, are automatically converted into high-resolution numbers (double reals) when they are imported from the database into the spreadsheet. Thus, displaying the numeric datatype is more for your information than for practical use. Although you are not likely to be greatly concerned about the actual datatypes of the individual dataitems, you'll find it useful to know what each data type means.

## Numeric Dataitem Types

Two basic types of number representations commonly exist in computers: *integers* (whole numbers) and *reals* (a "real" number is said to be a number with a decimal point) (Table 2-1). Both numeric types may be specified to be of various resolutions (that is, capable of resolving and accurately keeping track of perhaps a penny in a billion dollars). Real numbers are specified as either *short* (32-bit) or *long* (64-bit) numbers. Their respective data types are labeled R2 and R4 in IMAGE. Integer numbers (numbers without decimal points) may be either *short* (16-bit), *intermediate* (32-bit) or *long* (64-bit) integers, written as I1, I2 or I4 data types, respectively.

Slight modifications of the integer number types occur for the J1, J2 and J4 data types, which are used principally in COBOL, a common programming language. COBOL places additional restrictions on the range of values that an integer number may take. The J data types observe these restrictions. Another form of modified integers are the K data types, called *logicals*. A K1 or K2 number is identical to an I1 or I2 number, but is limited to positive values only.

Various other techniques have also been devised to store numbers. Two of these techniques involve the storage of numbers more like text than a normal numeric representation. Integer numbers stored in this fashion are called *packed* and *zoned integers* (P and Z data types, respectively). The resolution capacity of a packed or zoned integer is not fixed but is specified at the time of database creation.

## Text Dataitem Types

Only two types of text data types exist: either wholly upper case (the U data type) or both upper and lower case (the X data type). The length of the string of characters which can be entered into a text dataitem field is specified at the time the database is designed. Due to the manner in which text is stored, the length must always be an even number. Text dataitem lengths greater than 40 characters are rarely used.

## Printing the Form of the Database

If you are going to successfully extract information from your databases, you will find it useful to have printed copies of the structures of the relevant databases. In general, it is impossible to remember all of the dataitems and datasets if the databases are complex. Although the @FORM command is always available, printed copies of the database structures are a valuable reference.

DATA TYPE	IMAGE TYPE SPECIFICATION	COMMONLY CALLED	LENGTH IN BITS	RESOLUTION
NUMBER	R2	SHORT REAL	32 BITS	1 IN $\pm 4$ MILLION
NUMBER	R4	LONG REAL	64 BITS	1 IN $\pm 2 \times 10^{16}$
NUMBER	I1	SHORT INTEGER	16 BITS	1 IN $\pm 32,768$
NUMBER	I2	MED. INTEGER	32 BITS	1 IN $\pm 2$ BILLION
NUMBER	I4	LONG INTEGER	64 BITS	1 IN $\pm 2 \times 10^{19}$
NUMBER	J1	SHORT COBOL	16 BITS	1 IN $\pm 10,000$
NUMBER	J2	MED. COBOL	32 BITS	1 IN $\pm 1$ BILLION
NUMBER	J4	LONG COBOL	64 BITS	1 IN $\pm 1 \times 10^{19}$
NUMBER	K1	SHORT LOGICAL	16 BITS	1 IN 65,536
NUMBER	K2	LONG LOGICAL	32 BITS	1 IN 4 BILLION
NUMBER	P	PACKED	4 BITS/DIGIT	USER SPECIFIED
NUMBER	Z	ZONED	8 BITS/DIGIT	USER SPECIFIED
TEXT	X	TEXT STRING	8 BITS/CHAR	USER SPECIFIED
TEXT	U	UPPERCASE	8 BITS/CHAR	USER SPECIFIED

*Table 2-1. The data types used in IMAGE*

Three views of the database are recommended for printing. They are: function key [f1], *the sets of the database*; function key [f4], *the paths of the database*; and function key [f2], *the complete structure of the database*. To print these views to your system line printer, press first [f6], *print to the line printer*, and then in turn [f1], [f4], and [f2]. Press [f7], *print to terminal*, to redirect the output of @FORM back to your terminal's screen.

From these three views you have all of the information necessary to draw the database schema in the manner shown on page 2-9. All that needs to be done is to connect the master datasets with their respective search items in the detail datasets. The view [f4], *the paths of the dataset*, shows you how that is to be done. As you become more familiar with IMAGE, you will be able to draw the appropriate schemas more rapidly and will soon dispense with the need to draw them at all.

The complete printed @FORM of the training IMAGE database, QCDEMO, appears on the next several pages, printed in the order: Sets [f1], Paths [f4], and All [f2].

SETS OF IMAGE DATABASE QCDEMO
-------------------------------

---

MASTER DATASETS:		ENTRIES	CAPACITY
CAT-ID	automatic	120	501
SUBCAT-ID	automatic	70	101
JOB-ID	automatic	76	101
DATE-ID	automatic	951	3659
ACCOUNT-ID	automatic	483	1001
EMPLOYEE-ID	automatic	185	501
DUMMYSORT	automatic	1	1 *
KEYNAME-ID	automatic	476	1001
NAME-ID	automatic	133	501

DETAIL DATASETS:		ENTRIES	CAPACITY
JOB		53	102
CHANGES		59	78
JOBESTCOST		3598	4572
TAXRATE		55	507
CATEGORIES		120	510
SUBCATS		67	114
EMPLOYEES		181	505
PAYRATE		737	1005
LABOR		5796	7336
PAYRECORD		4051	5137
ACCOUNTS		482	1002
INVOICES		9962	12610
CHECKS		5445	6900
SCHEDULES		0	1004
FIXEDASSETS		82	504

\* The indicated dataset(s) are now more than 80% full.

*The [f1] printout. A listing of the master and detail sets in the selected database.*

PATHS OF DATABASE QCDEMO

---

this master      paths to these      using this      with this item  
 dataset has \_\_\_\_ detail datasets \_\_\_\_ search item \_\_\_\_ sorting the chain

---

CAT-ID	CATEGORIES	CATEGORY
	INVOICES	CATEGORY
	SCHEDULES	CATEGORY
	FIXEDASSETS	CATEGORY

---

SUBCAT-ID	JOBESTCOST	SUBCAT
	SUBCATS	SUBCAT
	INVOICES	SUBCAT

---

JOB-ID	JOBS	JOBNUM
	CHANGES	JOBNUM
	JOBESTCOST	JOBNUM
	TAXRATE	JOBNUM
	PAYRATE	JOBNUM
	LABOR	JOBNUM
	INVOICES	JOBNUM

---

The *Paths* printout compactly displays :

DATE-ID	LABOR	DATE
	PAYRECORD	DATE
	INVOICES	DATE
	CHECKS	DATE

---

ACCOUNT-ID	ACCOUNTS	ACCTCODE
	INVOICES	ACCTCODE
	CHECKS	ACCTCODE

---

EMPLOYEE-ID	EMPLOYEES	SOCSECNUM	
	PAYRATE	SOCSECNUM	
	LABOR	SOCSECNUM	DATE
	PAYRECORD	SOCSECNUM	

---

DUMMYSORT	EMPLOYEES	ENTER-AN-X	LNAME
	ACCOUNTS	ENTER-AN-X	KEYNAME

---

KEYNAME-ID	ACCOUNTS	KEYNAME
------------	----------	---------

---

NAME-ID	EMPLOYEES	LNAME
---------	-----------	-------

---

*The [f4] printout. A list of the chained paths which link the master datasets to detail datasets in the selected database.*

## SETS OF DATABASE QCDEMO

---

CAT-ID [automatic master]			
ITEMS	DATA TYPE		
CATEGORY	I1 [num]		[key item with 4 paths]
CAPACITY: 501	ENTRIES: 120		24% FULL

---

SUBCAT-ID [automatic master]			
ITEMS	DATA TYPE		
SUBCAT	I1 [num]		[key item with 3 paths]
CAPACITY: 101	ENTRIES: 70		69% FULL

---

JOB-ID [automatic master]			
ITEMS	DATA TYPE		
JOBNUM	I1 [num]		[key item with 7 paths]
CAPACITY: 101	ENTRIES: 76		75% FULL

---

DATE-ID [automatic master]			
ITEMS	DATA TYPE		
DATE	U6 [text]		[key item with 4 paths]
CAPACITY: 3659	ENTRIES: 951		26% FULL

---

ACCOUNT-ID [automatic master]			
ITEMS	DATA TYPE		
ACCTCODE	I1 [num]		[key item with 3 paths]
CAPACITY: 1001	ENTRIES: 483		48% FULL

---

EMPLOYEE-ID [automatic master]			
ITEMS	DATA TYPE		
SOCSECNUM	U10 [text]		[key item with 4 paths]
CAPACITY: 501	ENTRIES: 185		37% FULL

---

*The [f2] printout. A complete listing of all of the datasets, data-items, and dataset capacities for the selected database.*

2-18 / The Database

DUMMYSORT [automatic master]

ITEMS	DATA TYPE	
ENTER-AN-X	U2 [text]	[key item with 2 paths]
CAPACITY: 1	ENTRIES: 1	100% FULL

---

KEYNAME-ID [automatic master]

ITEMS	DATA TYPE	
KEYNAME	U14 [text]	[key item with 1 path]
CAPACITY: 1001	ENTRIES: 476	48% FULL

---

NAME-ID [automatic master]

ITEMS	DATA TYPE	
LNAME	U14 [text]	[key item with 1 path]
CAPACITY: 501	ENTRIES: 133	27% FULL

---

JOBS [detail set]

ITEMS	DATA TYPE	
JOBNUM	I1 [num]	[search item]
DESCRIPTION1	U40 [text]	
DESCRIPTION2	U40 [text]	
DESCRIPTION3	U40 [text]	
DESCRIPTION4	U40 [text]	
SITENUM	I1 [num]	
CONTRACTAMOUNT	R4 [num]	
BIDDATE	U6 [text]	
STARTDATE	U6 [text]	
MONTH	U2 [text]	
ESTCOMPDATE	U6 [text]	
ACTCOMPDATE	U6 [text]	
WCSEXEMPT	U2 [text]	
BUILDERSRISK	U2 [text]	
OWNER	U20 [text]	
ADDRESS	U40 [text]	
CITY	U16 [text]	
STATE	U2 [text]	
ZIP	U10 [text]	
PHONE	U12 [text]	
JOBCOST	R4 [num]	
WORKCOMPLETE	R4 [num]	
PRIORCOST	R4 [num]	
PRIOREARNED	R4 [num]	
BILLED	R4 [num]	
RETAINAGE	R4 [num]	
GRT	R4 [num]	
CAPACITY: 102	ENTRIES: 53	52% FULL

---

CHANGES [detail set]

ITEMS	DATA	TYPE	
JOBNUM	I1	[num]	[search item]
DESCRIPTION1	U40	[text]	
DESCRIPTION2	U40	[text]	
DESCRIPTION3	U40	[text]	
DESCRIPTION4	U40	[text]	
CAPACITY: 78		ENTRIES: 59	76% FULL

---

JOBESTCOST [detail set]

ITEMS	DATA	TYPE	
SUBCAT	I1	[num]	[search item]
JOBNUM	I1	[num]	[search item]
ESTCOST	R4	[num]	
CAPACITY: 4572		ENTRIES: 3598	79% FULL

---

TAXRATE [detail set]

ITEMS	DATA	TYPE	
JOBNUM	I1	[num]	[search item]
SITETAX	R2	[num]	
STARTDATE	U6	[text]	
TERMDATE	U6	[text]	
CAPACITY: 507		ENTRIES: 55	11% FULL

---

CATEGORIES [detail set]

ITEMS	DATA	TYPE	
CATEGORY	I1	[num]	[search item]
SCHEDULE	U2	[text]	
DESCRIPTION1	U40	[text]	
AMOUNT	R4	[num]	
DATE	U6	[text]	
CAPACITY: 510		ENTRIES: 120	24% FULL

---

SUBCATS [detail set]

ITEMS	DATA	TYPE	
SUBCAT	I1	[num]	[search item]
DESCRIPTION1	U40	[text]	
SUBCONTRACTED	U2	[text]	
CAPACITY: 114		ENTRIES: 67	59% FULL

---

2-20 / The Database

EMPLOYEES [detail set]

ITEMS	DATA	TYPE	
ENTER-AN-X	U2	[text]	[search item]
LNAME	U14	[text]	[search item] [sort item]
FNAME	U10	[text]	
SOCSECNUM	U10	[text]	[search item]
ADDRESS	U40	[text]	
CITY	U16	[text]	
STATE	U2	[text]	
ZIP	U10	[text]	
PHONE	U12	[text]	
MARRIED	U2	[text]	
NUMDEDUCTIONS	I1	[num]	
STARTDATE	U6	[text]	
TERMDATE	U6	[text]	
TERMREASON	U30	[text]	
INSURANCE	R2	[num]	
FICA	R2	[num]	
FUTA	R2	[num]	
NMESC	R2	[num]	

CAPACITY: 505

ENTRIES: 181

36% FULL

---

PAYRATE [detail set]

ITEMS	DATA	TYPE	
SOCSECNUM	U10	[text]	[search item]
JOBNUM	I1	[num]	[search item]
CLASS	U2	[text]	
RATE1	R2	[num]	
RATE2	R2	[num]	
RATE3	R2	[num]	
RATE4	R2	[num]	
RATE5	R2	[num]	
RATE6	R2	[num]	
RATE7	R2	[num]	
RATE8	R2	[num]	
RATE9	R2	[num]	

CAPACITY: 1005

ENTRIES: 737

73% FULL

---

## LABOR [detail set]

ITEMS	DATA TYPE	
SOCSECNUM	U10 [text]	[search item]
JOBNUM	I1 [num]	[search item]
REGULAR	R2 [num]	
OVERTIME	R2 [num]	
DATE	U6 [text]	[search item] [sort item]
AMOUNT	R4 [num]	
SUBCAT	I1 [num]	
FICA	R2 [num]	
FUTA	R2 [num]	
NMESC	R2 [num]	

CAPACITY: 7336

ENTRIES: 5796

79% FULL

## PAYRECORD [detail set]

ITEMS	DATA TYPE	
SOCSECNUM	U10 [text]	[search item]
GROSS	R4 [num]	
FICA	R2 [num]	
FUTA	R2 [num]	
NMESC	R2 [num]	
FEDWITH	R2 [num]	
STATEWITH	R2 [num]	
DEDUCTION	R2 [num]	
CHECKNUM	I1 [num]	
AMOUNT	R4 [num]	
DATE	U6 [text]	[search item]
REGULAR	R2 [num]	
OVERTIME	R2 [num]	
INSURANCE	R2 [num]	

CAPACITY: 5137

ENTRIES: 4051

79% FULL

## ACCOUNTS [detail set]

ITEMS	DATA TYPE	
ENTER-AN-X	U2 [text]	[search item]
DESCRIPTION1	U40 [text]	
KEYNAME	U14 [text]	[search item] [sort item]
ADDRESS	U40 [text]	
CITY	U16 [text]	
STATE	U2 [text]	
ZIP	U10 [text]	
PHONE	U12 [text]	
ACCTCODE	I1 [num]	[search item]

CAPACITY: 1002

ENTRIES: 482

48% FULL

2-22 / The Database

INVOICES [detail set]

ITEMS	DATA	TYPE	
CATEGORY	I1	[num]	[search item]
SUBCAT	I1	[num]	[search item]
JOBNUM	I1	[num]	[search item]
ACCTCODE	I1	[num]	[search item]
INVOICENUM	I1	[num]	
AMOUNT	R4	[num]	
DISCOUNT	R2	[num]	
BALANCE	R4	[num]	
DATE	U6	[text]	[search item]

CAPACITY: 12610      ENTRIES: 9962      79% FULL

---

CHECKS [detail set]

ITEMS	DATA	TYPE	
ACCTCODE	I1	[num]	[search item]
AMOUNT	R4	[num]	
DATE	U6	[text]	[search item]
CHECKNUM	I1	[num]	

CAPACITY: 6900      ENTRIES: 5445      79% FULL

---

SCHEDULES [detail set]

ITEMS	DATA	TYPE	
CATEGORY	I1	[num]	[search item]
DESCRIPTION1	U40	[text]	
DESCRIPTION2	U40	[text]	
DESCRIPTION3	U40	[text]	
DESCRIPTION4	U40	[text]	
AMOUNT	R4	[num]	
DATE	U6	[text]	

CAPACITY: 1004      ENTRIES: 0      0% FULL

---

FIXEDASSETS [detail set]

ITEMS	DATA	TYPE	
CATEGORY	I1	[num]	[search item]
DESCRIPTION1	U40	[text]	
DESCRIPTION2	U40	[text]	
AMOUNT	R4	[num]	
DATE	U6	[text]	
SCHEDULE	U2	[text]	
YEARDEPR	R4	[num]	
NETVALUE	R4	[num]	
TERMDATE	U6	[text]	

CAPACITY: 504      ENTRIES: 82      16% FULL

---

Concepts  
Introduced in  
Chapter 2

---

---

DETAIL DATASET	the basic "file drawer"
MASTER DATASET	the 3x5 card index dataset
SEARCH ITEM	a dataitem in a dataset for which a chained search path exists
CHAINED PATH	the linking together of all of the records which share a common search item value
NON-SEARCH ITEM	any of the other dataitems in a dataset for which no search chain exists
AUTOMATIC MASTER	search item values (for which no chained search path currently exists) are automatically created when an automatic master is used
SERIAL SEARCH	a search of all of the records in a dataset, first to last
DATABASE FORM	the schematic drawing of the database's structure
REAL NUMBERS	numbers with decimal points
INTEGER NUMBERS	whole (non-fractional) numbers
RESOLUTION	the largest usable number after which accuracy will be lost

---

---