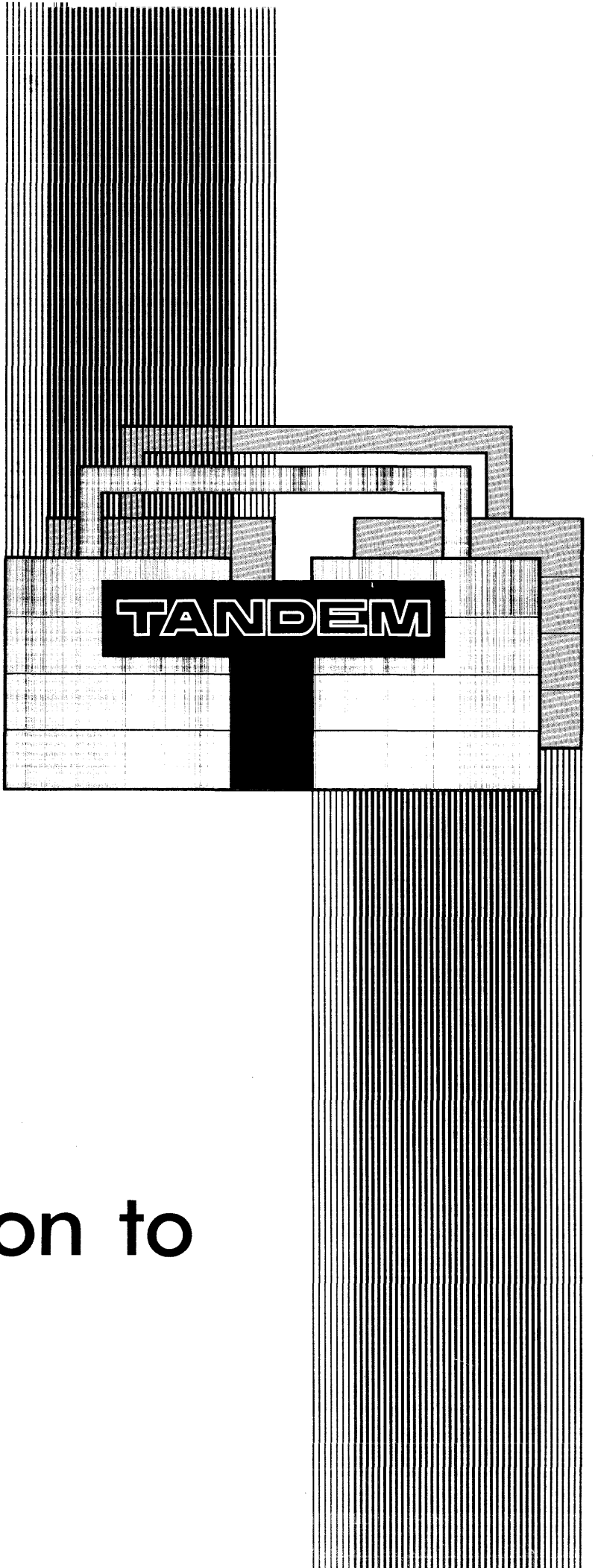


Introduction to ENFORM

INTRODUCTION TO ENFORM

Tandem Computers Incorporated
19333 Vallco Parkway
Cupertino, California 95014

P/N 82313 A00

October 1981
Printed in U.S.A.

PREFACE

This publication is an introduction to the ENFORM query and report writing language. ENFORM is a product in the ENCOMPASS Distributed Data Base Management System that can be used on the Tandem NonStop Computer System and on the Tandem NonStop II Computer System. The purpose of ENFORM is to enable all users of a Tandem system to query an online data base—to ask questions about their business and to retrieve the answers (as reports) on their terminals or as printed documents.

This publication is for anybody who is about to use ENFORM for the first time. It has several goals: to present basic ENFORM concepts; to start teaching you how to use ENFORM; and to enable you to start evaluating ENFORM's potential for use in an online transaction processing environment.

The organization of the publication is:

- Section 1—an overview of ENFORM's purpose and features. Anyone who is interested in ENFORM should read this section.
- Section 2—a quick summary (with examples) of using ENFORM language specifications to produce reports. Readers who are interested in learning how to write queries should read this section for terminology and basic features of the language.
- Section 3—a conceptual description of how ENFORM works. Readers who are primarily interested in writing queries can skip this section.
- Section 4—a simple tutorial on using ENFORM; it concentrates on teaching you how to do some of the tasks involved in generating a complete report. This section can be used without reading any of the preceding sections. Reading this section will help to prepare you for using the more detailed information in the *ENFORM Reference Manual*.

For additional information related to understanding and using ENFORM, see the following publications.

- *Introduction To Tandem Computer Systems*
- *ENFORM Reference Manual*
- *Data Defintion Language (DDL) Programming Reference Manual*
- *EDIT Manual*

CONTENTS

LIST OF FIGURES

SECTION 1

OVERVIEW OF ENFORM

Information about your business is critical to controlling the business. You need this information to make timely and correct decisions, to satisfy customer or management requests, and to plan for the future.

If your organization depends upon online transaction processing to run its business, you typically store large amounts of complex information as data files in a data base. The data can change constantly and rapidly. And for the data to be useful, you must be able to interrogate the data base: to select and retrieve its information quickly, accurately, and in a form that satisfies your needs (Figure 1-1).

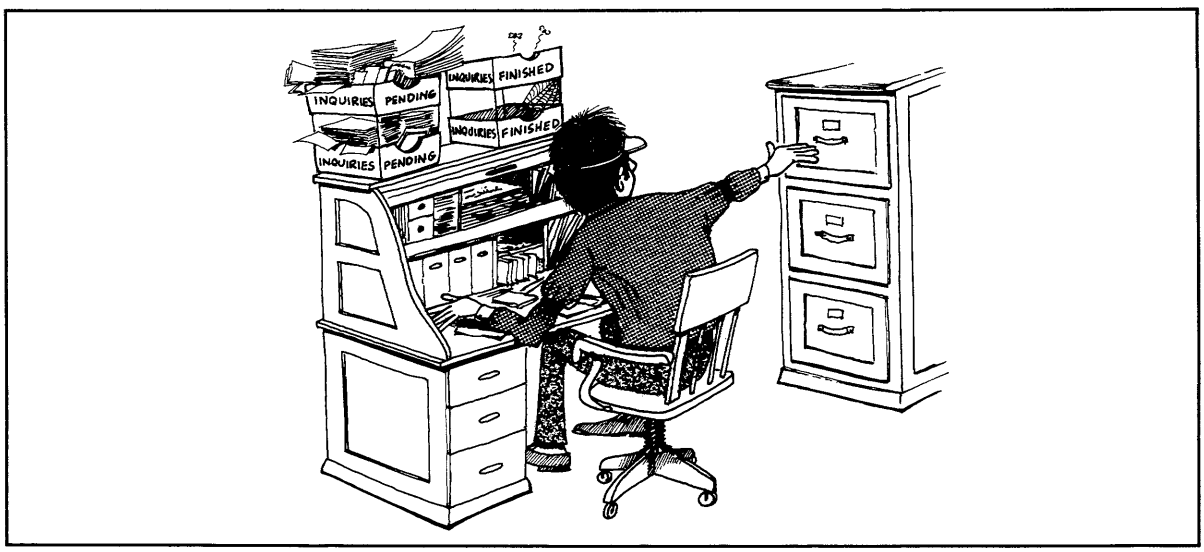


Figure 1-1. Data Retrieval

ENFORM is a product provided by Tandem, as part of its ENCOMPASS Distributed Data Base Management System, to help meet your need for retrieving information from the online data base. ENFORM enables all users (programmers and non-programmers alike) of a Tandem system to query the data base—to ask questions about their business and to retrieve the answers (as reports) on their terminals or as printed documents.

The following pages describe ENFORM and discuss its use in an ENCOMPASS online transaction processing environment.

INTRODUCING ENFORM

Formally, ENFORM is defined as "a powerful nonprocedural language for querying or developing reports on a relational data base." This definition is complete and it does describe ENFORM. But, what does it mean to you?

Language—Definition

First and most important, ENFORM is a *language*: a formal way of expressing your information needs to the system by using ENFORM statements. The statements, which resemble English, tell ENFORM what information you require and what you want the information to look like. Viewed simply, using a language like ENFORM is the first step in a sequence of events that transforms your ideas into actions that can be performed by the computer.

Nonprocedural—Definition

Nonprocedural means that you tell ENFORM what you need, but not how to produce it. ENFORM chooses a search strategy for accessing the data base and producing the required information. This frees you from the problem of deciding how to extract information from a large complex data base and allows you to concentrate on your most important problem: deciding exactly what information is necessary to answer your questions about the business.

Think of a nonprocedural language by comparing it to automated exposure on modern cameras. The camera handles the details of exposure times and lens opening, enabling you to think about, compose and take the picture without secondary distractions.

Queries and Reports—Definition

This is a query—a question regarding information in the data base:

```
OPEN parts;
LIST parts,                                    the LIST statement
    WHERE price LESS THAN 20000;               produces a report
```

And this is a report—the answer to the question posed by the query:

PARTNUM	PARTNAME	INVENTORY	LOCATION	PRICE
4102	DISC 10 MB	14	K87	8000.00
4103	DISC 50 MB	9	K45	14500.00

The query is your question—phrased in a language that ENFORM can understand. It is a specification of the exact information that should be selected from the data base. Simple queries can be entered by using a simple ENFORM statement. Complex queries can be entered by adding more specifications to the statement, precisely defining the information to be selected and its format.

Temporary queries can be entered to answer questions like "how many wrenches are in stock?" And complex queries can be developed, using the editor, stored and saved to answer long term needs like weekly inventory reports or monthly budget reports. The complex queries can be developed a piece at a time, tested, and saved for future use by you or other users.

Queries enable you to retrieve information spontaneously. If you have an immediate need for information, you can enter an ENFORM statement and retrieve the information quickly. Contrast this to having to develop a program in COBOL or FORTRAN where preliminary planning and programming development is required. Also, if the selected information is not quite what you need, the first time, the query can be successively and easily modified (in contrast to modifying a program) until the information is exactly right.

The report is the answer to your question: ENFORM'S output. It contains selected information; sorted and arranged to your query specifications; with page numbers, titles, headings, and subheading included; and with any specified arithmetic (such as averages and totals) performed for the report.

Relational Data Base—Definition

A relational data base enables you to think about and represent files in the data base as a collection of similarly structured tables: two-dimensional lists where the rows and columns contain specific pieces of information. Each table contains information about something related to your business; for example, parts or suppliers (Figure 1-2). Relationships between tables are established by including items from one table in another table.

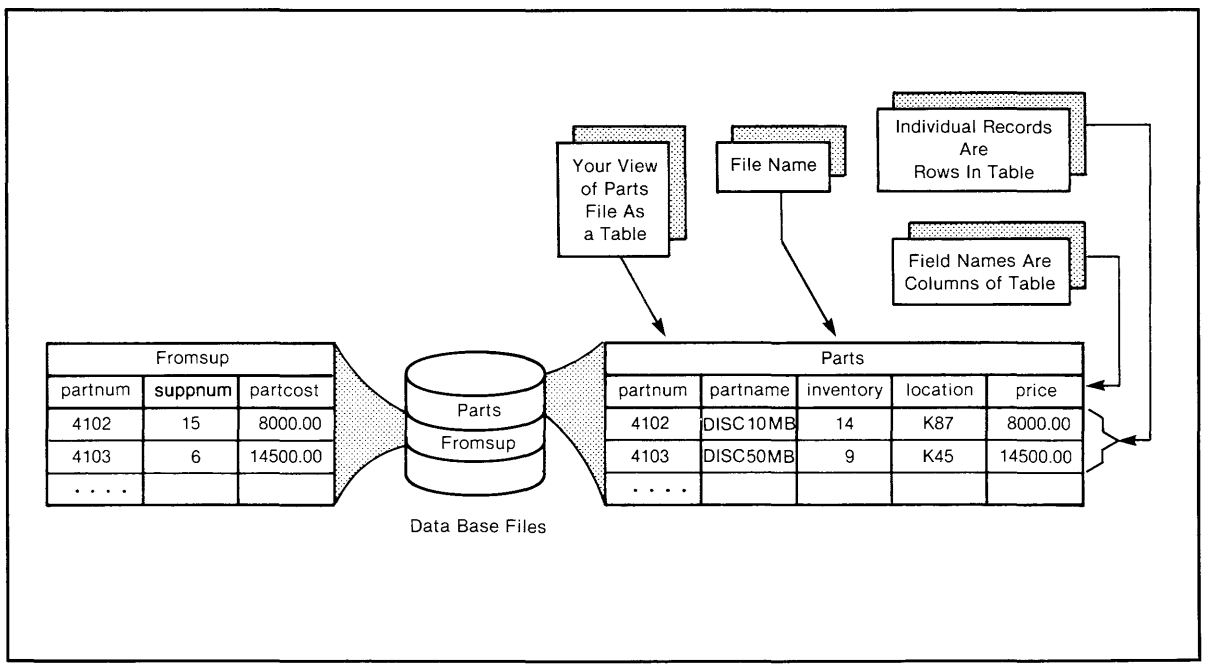


Figure 1-2. Relational View of Data

Since tables are a natural way for you to think about information, a relational data base is easy to use; it does not require understanding of how the data is physically stored or accessed in the computer. Instead, you only need to know what information you need and how it is related to other information.

Additionally, a relational data base allows you to establish simple views of the data—where all the information related to one part of a business is in one or two tables. And the simple views can be extended to more complex views—where multiple tables can be linked together to contain all of the information for the entire business (Figure 1-3). This allows you to synthesize information from related tables, when you need to, while still working with the basic concept of one table at a time.

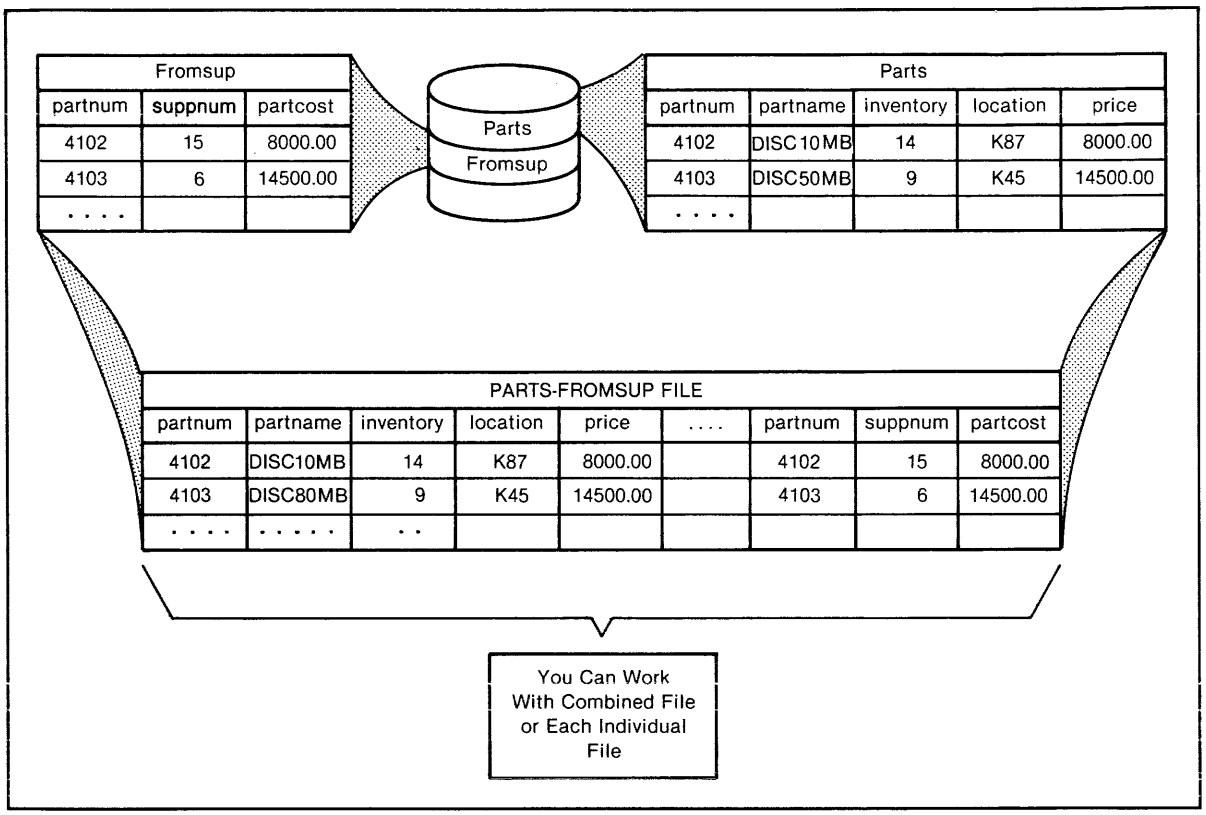


Figure 1-3. Relating Information in a Relational Database

Powerful—Definition

Powerful has two meanings with regard to ENFORM: (1) you can do a lot with a little and (2) the same language (ENFORM) is flexible enough to support a wide variety of users' needs and levels of sophistication.

ENFORM features a number of functions built into the language including: averages, totals, subtotals, titles, column headings, sorting, and qualifications for selecting information. These functions allow you to select data and to customize your reports with a minimal number of ENFORM specifications.

Anybody can use ENFORM. No programming background is required. Instead the only requirements are: (1) you must know how the information in the relational data base is named and related and (2) you must learn how to write and enter the queries. The complexity of the queries are under your control; simple queries require minimal understanding of ENFORM and of the data base; more complex queries with sophisticated report formatting require more knowledge.

ENFORM AND ENCOMPASS

ENCOMPASS is Tandem's Distributed Data Base Management System. It contains the following products:

- PATHWAY — provides a simplified terminal-oriented interface for developing and controlling modular transaction processing applications.
- ENABLE — automatically generates interactive update applications under PATHWAY.
- TMF — maintains consistency of data bases changed by transactions.
- DDL — used to create a dictionary for the relational data base; the dictionary describes the structure of all files in the data base.
- ENFORM — provides a data base query and reporting capability.

The purpose of ENCOMPASS is to simplify the development and control of online transaction processing applications (Figure 1-4).

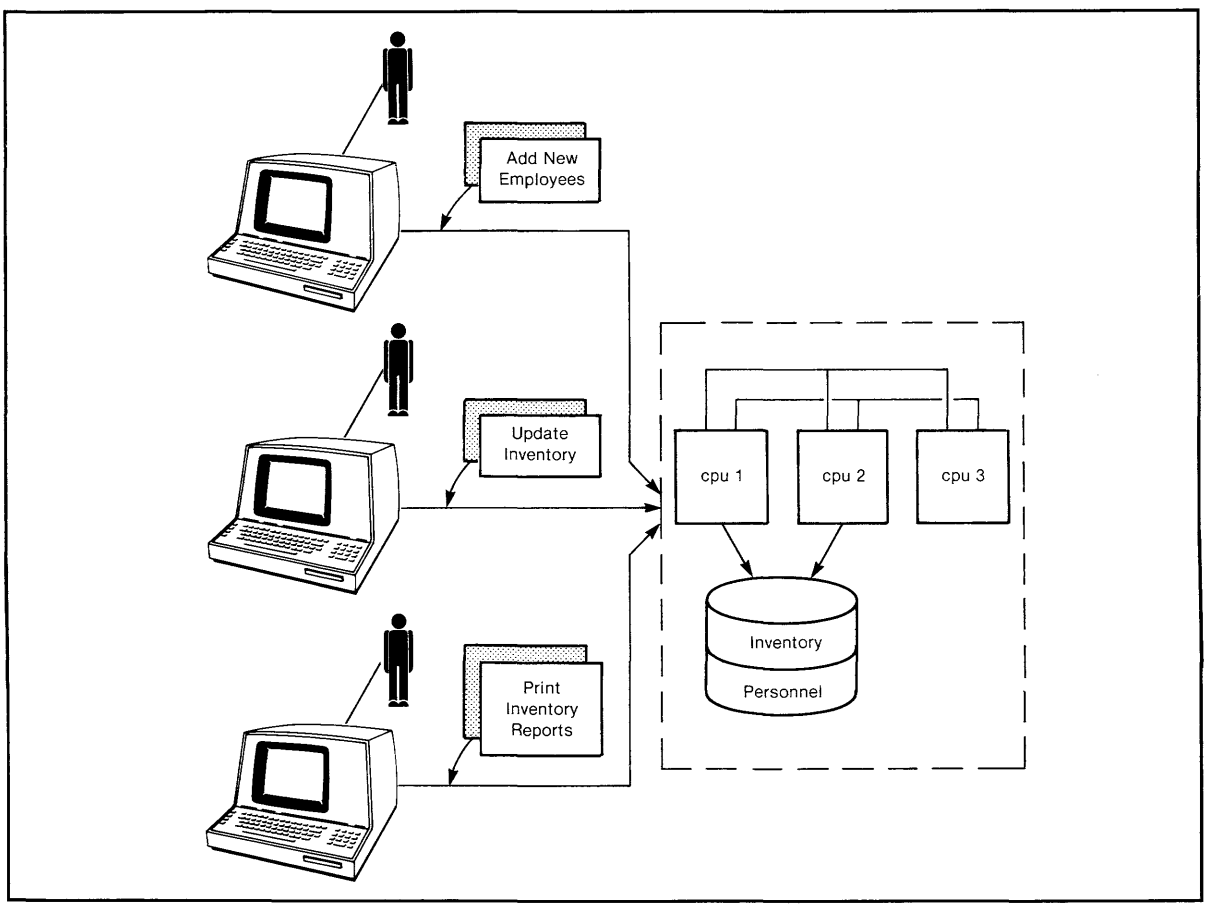


Figure 1-4. Online Transaction Processing Applications

Online Transaction Processing

As illustrated in Figure 1-4, online transaction processing applications typically control a number of terminals and disc drives that are connected to a host computer on which the application software runs. The software connects users (through their terminals) to the data base on the disc drives.

The application software consists of transactions that are presented to the system by operators at the terminals. Transactions are units of work defined by the business that uses the computer; for example, a change to inventory, the registration of students for a class, or a request to find out how many wrenches are in stock. A characteristic of transactions is that they require access to the data base—either to search for and modify information or simply to search for and retrieve information.

Transaction processing then means any system in which transactions are entered by terminal operators to query or modify a data base.

The word *online* means that the changes made by any transaction are, upon completion, available to all other terminal operators who access the data base. For example, if a transaction adds a new item to inventory, all succeeding inquiries (after the transaction completes) about the inventory status will reflect the new item.

Online Transaction Processing and ENFORM

Typical Tandem online transaction processing applications consist of a mix of retrieval-only transactions and transactions that both retrieve information and change it. That is, some transactions modify the data and some only extract data.

PATHWAY and TMF are the products used to build and control the application transactions that inspect and change the data base. ENFORM simplifies the data retrieval part of the application where the objective of the retrieval is to inquire and report about information in the data base, but not to change the data (Figure 1-5). This means that application programmers can use ENFORM to reduce the amount of development time required to produce standard formatted reports. Alternatively, non-programmers can satisfy their information needs without doing any programming development tasks and without the possibility of accidentially modifying information in the data base.

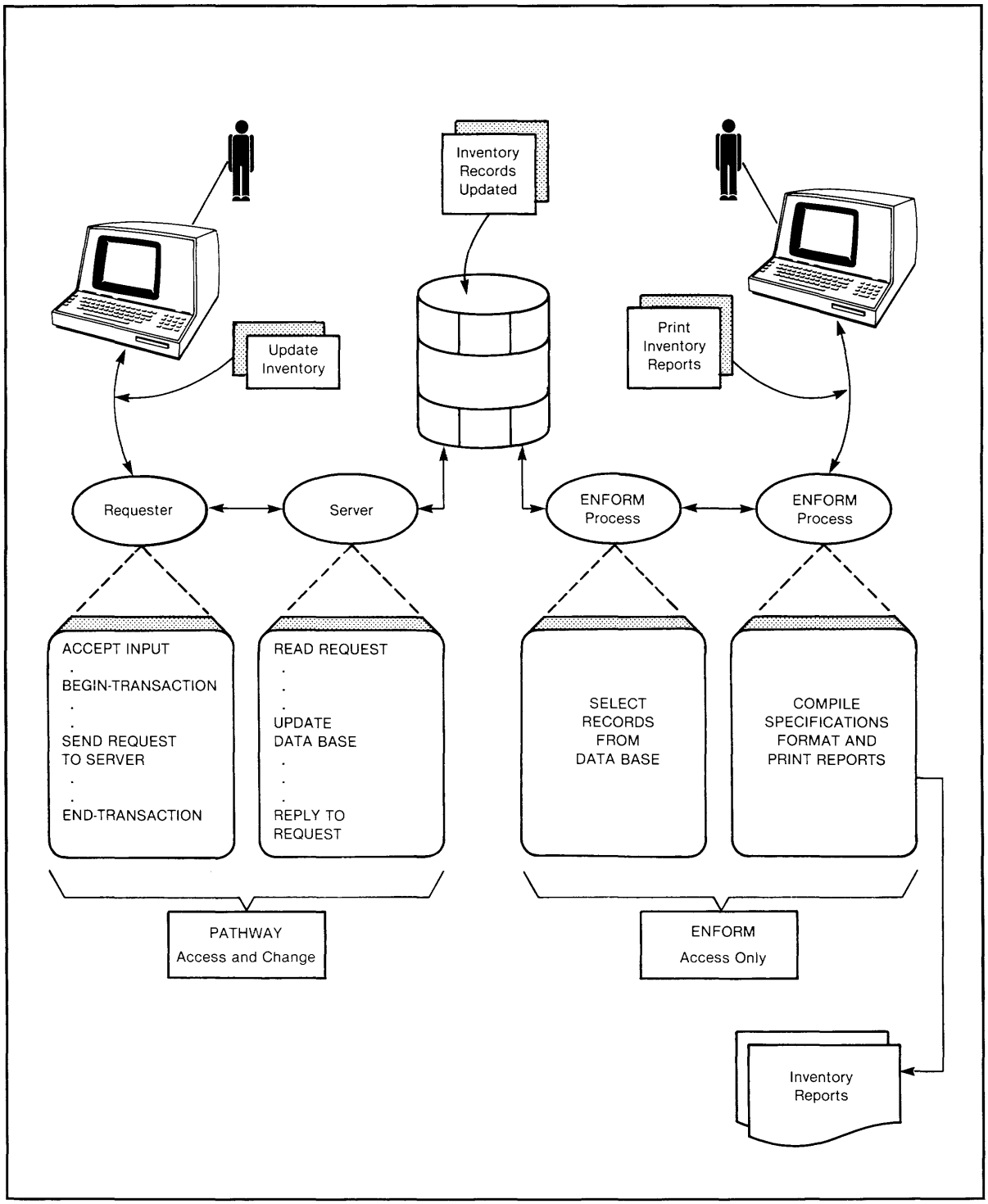


Figure 1-5. ENFORM In An Online Transaction Processing Environment

A key point about the use of ENFORM in this environment is that it has the potential to provide a snapshot of a consistent data base; a data base that is being changed by transactions that modify the data. The information you retrieve will be up-to-date and reflect the most recent changes to the data. So, reports produced by using ENFORM—either as the result of a simple inquiry, or as one of the standard application reports—can be based on all changes occurring to the data base and take advantage of all ENCOMPASS features.

SUMMARY OF ENFORM FEATURES

ENFORM features include:

- shared access to the data base through a centralized dictionary
- data base searching based on its own set of search algorithms
- support of use through a host language interface
- the flexibility to specify new relationships between data
- access to files distributed over an EXPAND network.

These features are described, briefly, in the following pages and, in more detail, in the remaining sections of this publication and in the *ENFORM Reference Manual.*

Shared Access

Before using ENFORM, the data base administrator must use DDL to build a dictionary that describes the data base. The data dictionary is a centralized set of files that form a permanent record of the data base description. Each file in the data base is represented by a record in the dictionary (Figure 1-6). Since the dictionary describes the structure of all files, their records, and the relationships between the files, ENFORM uses the dictionary as a source of information about file structures and locations.

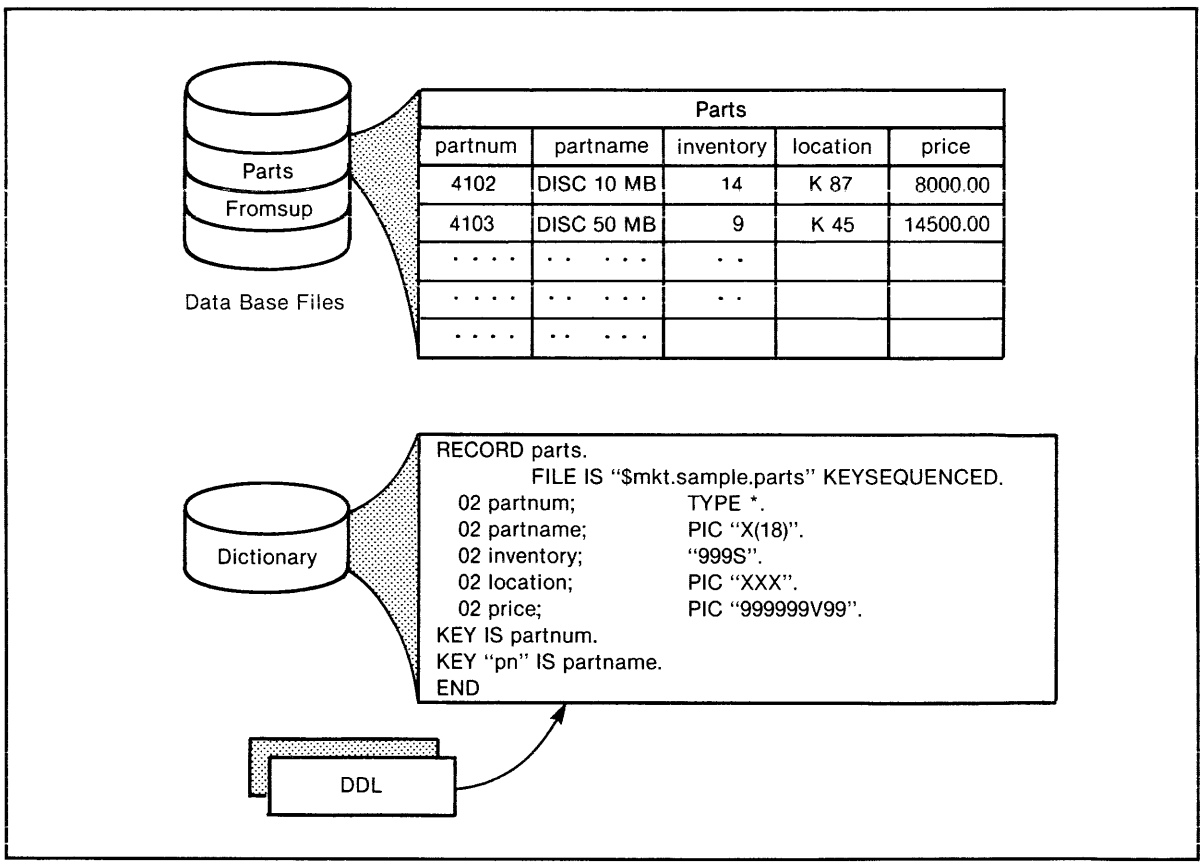


Figure 1-6. Data Dictionary

The dictionary and the data base files are shared resources; multiple terminal operators can access them concurrently, but each operator can think of their query as the only transaction in the system. Since ENFORM does not require an exclusive data base and dictionary, they can be shared with other application transactions written in COBOL, FORTRAN, or TAL.

An advantage of using DDL to produce the dictionary is that the data base can generally be extended or modified without forcing any changes to existing ENFORM specifications; you simply change the dictionary instead.

Data Base Searching

ENFORM has two processes: (1) the compiler/report writer and (2) the query processor. The compiler/report writer compiles the ENFORM specifications, accesses the dictionary to determine the structure of the information to be retrieved, and passes the specifications and data structure information to the query processor (Figure 1-7). The query processor then: chooses a strategy for accessing the data base; accesses the files specified in the query; processes the data according to the specifications; and returns the data to the compiler report/writer for report generation.

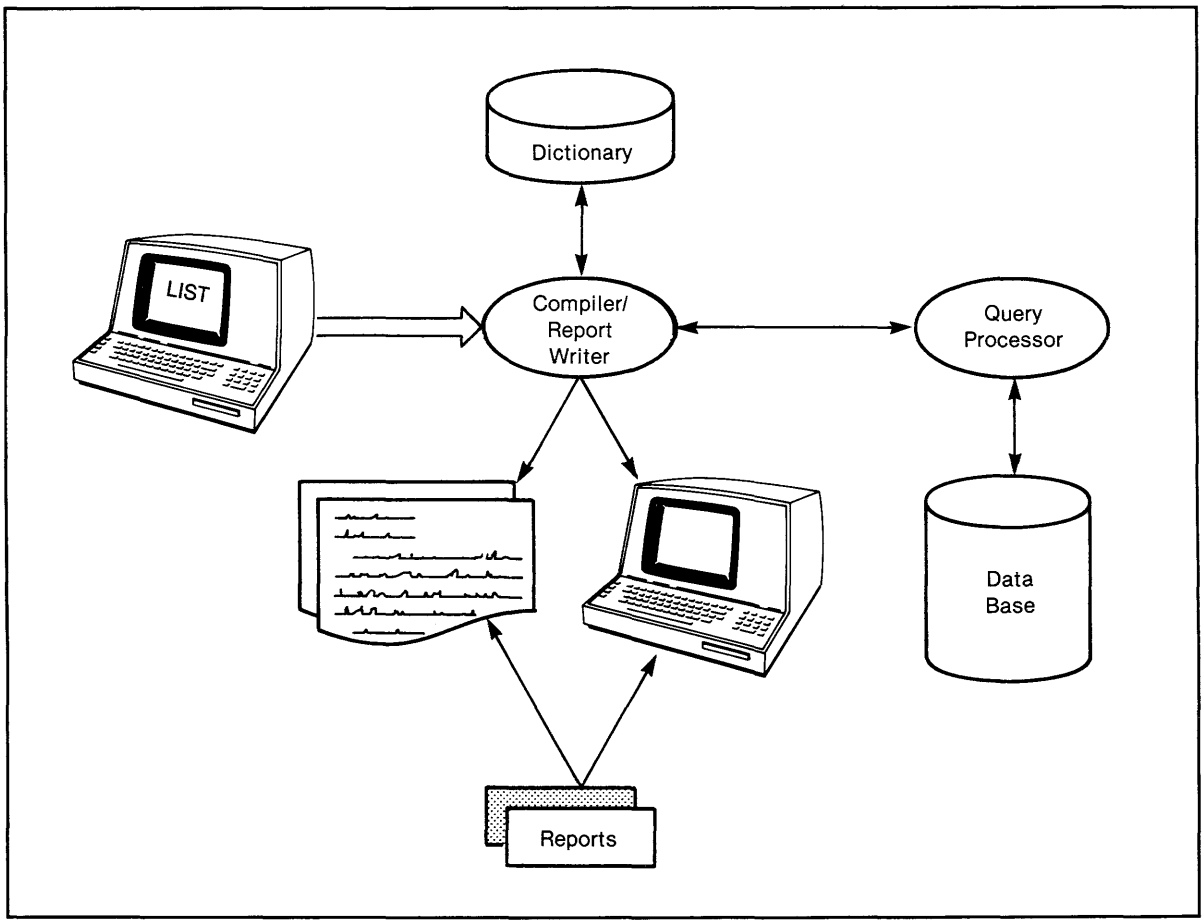


Figure 1-7. ENFORM Environment

In the environment illustrated in Figure 1-7, you need to specify what information should be selected, without worrying about how the data is to be accessed or where it is stored. The query processor automatically selects a search strategy for accessing the information by using information from the dictionary and the file system, and by using its own search algorithms to reduce the number of data base accesses (separate disc reads) required to produce your information.

Host Language Interface

Programs written in COBOL, TAL, or FORTRAN can use ENFORM to retrieve information from the data base by saving compiled queries in a file and using ENFORM procedures to pass the queries to the query processor host language interface. Records are then returned from the query processor to the program. Application development time and program maintenance can be reduced using this approach because:

- changes to a physical file or key in the data base do not require any changes to the host language program
- the query processor selects the access strategy for a complex ENFORM program
- record transformations are easier to express in ENFORM than in most other languages—you simply identify the before and after views of the records, not the steps required to do the transformations.

Compiled queries are generated as illustrated in Figure 1-8.

See the *ENFORM Reference Manual* for more details on this subject.

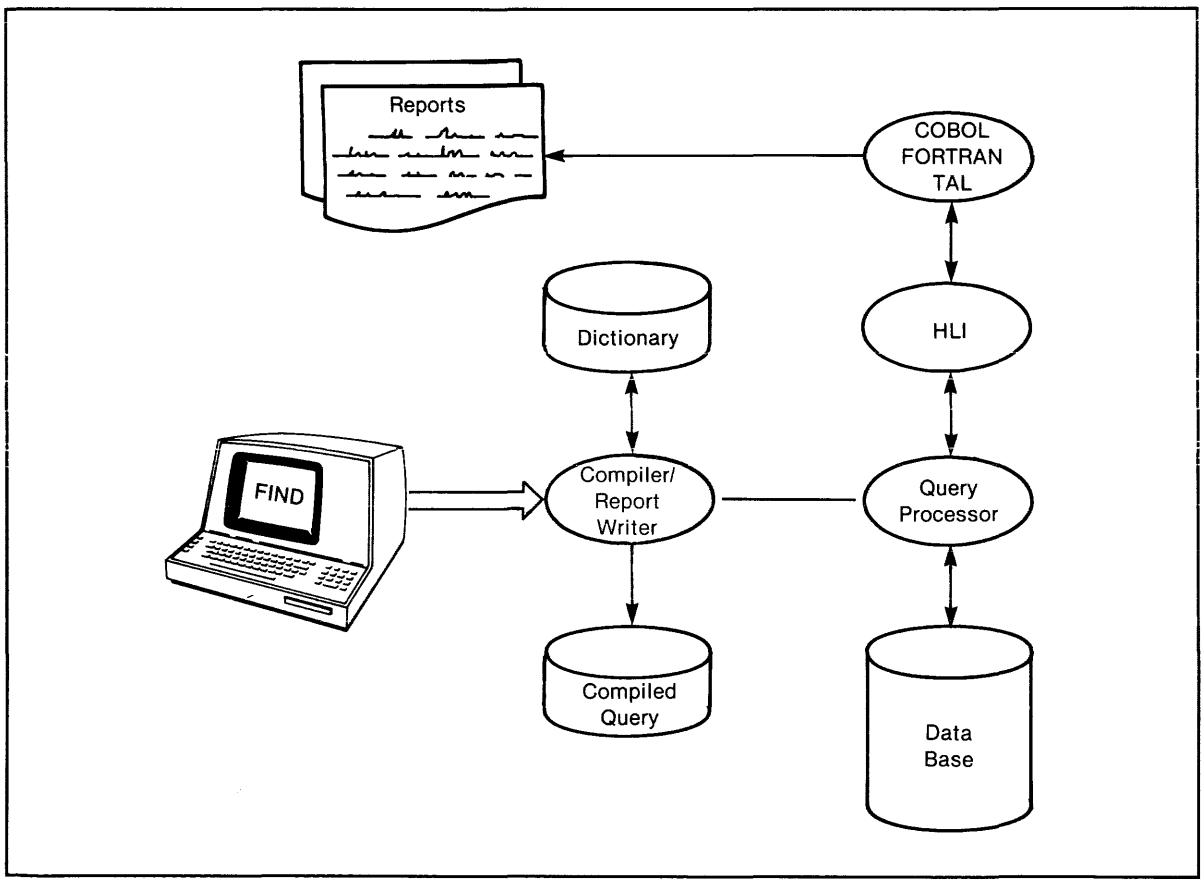


Figure 1-8. Generating Compiled Queries

New Data Relationships

The files of a data base are often interrelated. For example, Figure 1-9 shows two files (tables) that share a common field, *partnum*. The files were designed this way, intentionally, to associate inventory parts with information related to the suppliers that provide them. ENFORM has a LINK statement that allows you to specify a relationship between these two files (at run time) that appears to create a new file made up of information from each file. Information can be selected from the new file and synthesized in any report produced for the file.

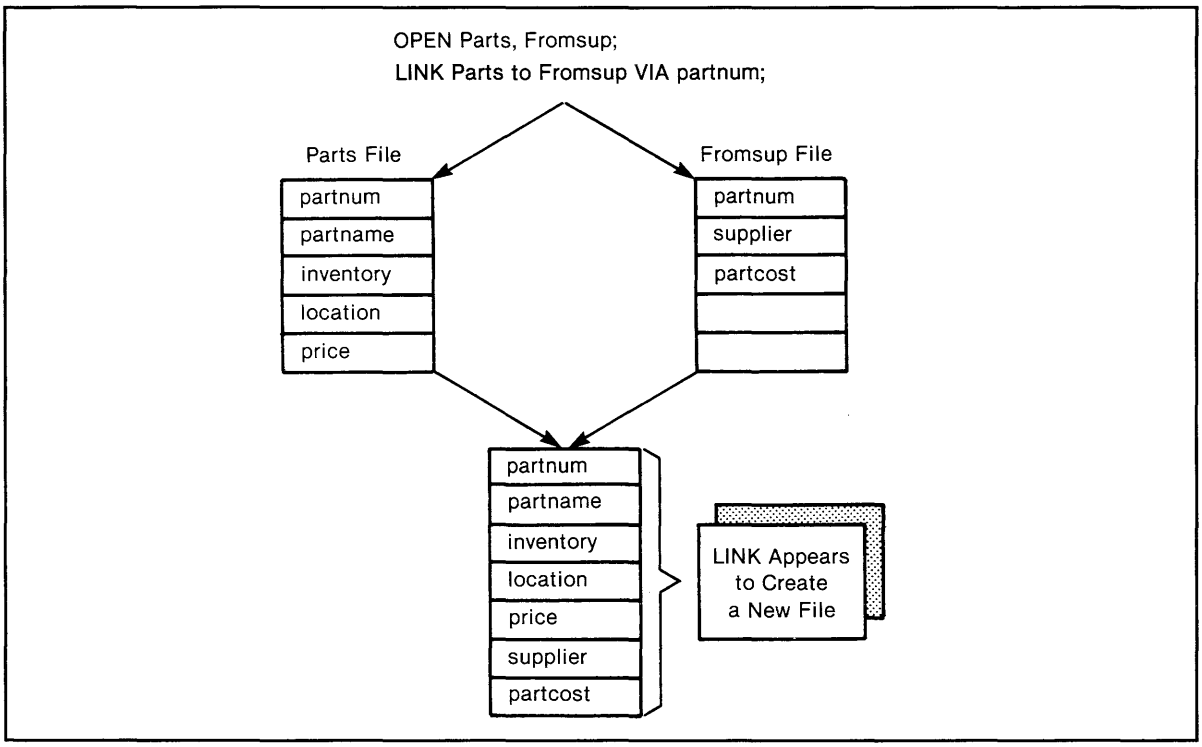


Figure 1-9. Linking Files

ENFORM uses the dictionary to identify the relationships and connect the files. You do not have to restructure the data base, change the dictionary, or worry about the location of the files. Instead you concentrate on connecting information in logical ways to satisfy your need for information. ENFORM takes care of the physical details associated with the linking.

Use Over an EXPAND Network

ENFORM allows you to select data from any files distributed over a network by using GUARDIAN/EXPAND. But, transmitting large amounts of data over a network takes time and increases network traffic. For example, if your query originates in New York and the related data files are in Chicago, all of the data might have to be transmitted to New York before processing on the query could start.

However, because ENFORM consists of two processes (the compiler/report writer and the query processor), it is possible to efficently process a distributed data base and decrease network traffic. The query processor can be created at the network node where most of the data resides (Figure 1-10). The query processor will then process the files at that node and select the information that satisfies the query specifications; this will be the only information transmitted over the network to the compiler/report writer.

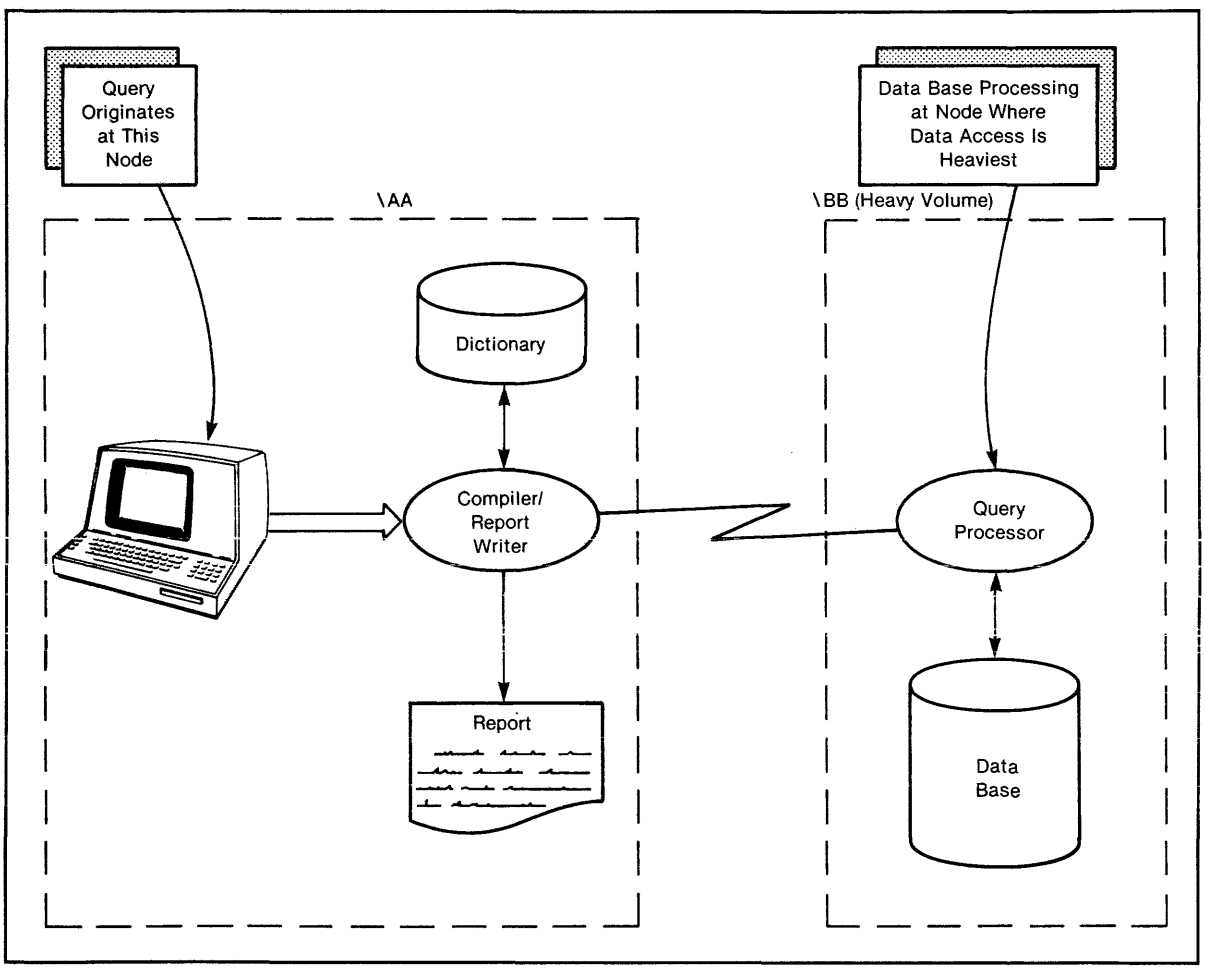


Figure 1-10. Using ENFORM Over a Network

SECTION 2

USING ENFORM—THE BASICS

A *session* is the term that describes the period of time that you interact with ENFORM. Basically, a conversational session (one where ENFORM specifications are entered a line at a time) consists of the steps illustrated in Figure 2-1.

Step	You Enter
1. Start ENFORM and identify the dictionary subvolume	`:ENFORM $data.rept`
2. Identify the data descriptions for the files	`>OPEN parts,fromsup;`
3. *Optionally, relate files to form a new view*	`>LINK parts TO fromsup` `>VIA partnum;`
4. Specify formatting information	`>TITLE "Inventory" CENTER;`
5. Specify information to be selected for report	`>LIST parts,` `>WHERE price LESS THAN 5000;`
6. *Repeat versions of steps 4 and 5 until all reports are produced*	
7. End ENFORM session	`>?EXIT`

Figure 2-1. Example of Conversational ENFORM Session

The steps on the preceding page are intended to give you an initial impression of how ENFORM is used; they show that an ENFORM session consists of starting ENFORM and then entering specifications: ENFORM statements, clauses, and commands. To illustrate these ideas further, the following pages contain:

- Description of different types of ENFORM queries.
- An overview of the steps involved in producing a report.
- A brief description of ENFORM specifications.
- Examples of queries and reports.

This section describes only the most basic types of query specifications—those used in the majority of queries. The "Using ENFORM To Produce Complete Reports" section of this publication and the *ENFORM Reference Manual* contain more details.

TYPES OF ENFORM QUERIES

There are three types of ENFORM queries:

1. temporary
2. stored
3. compiled.

Temporary Queries

A temporary query is illustrated in Figure 2-2. In this type of query, the ENFORM specifications are entered conversationally and not saved. The LIST statement is executed immediately after it is entered. Temporary queries are useful for spontaneous access to the data base when the query is relatively simple; i.e., a limited amount of information is required from one or two files only.

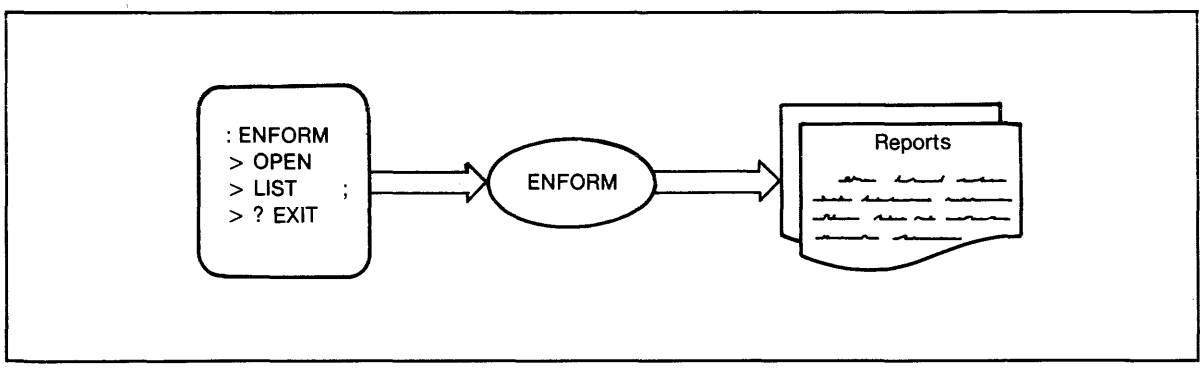


Figure 2-2. Temporary Queries

Stored Queries

A stored query is illustrated in Figure 2-3. In this type of query, ENFORM specifications are entered in an edit file and saved. The edit file can be used as a source file for ENFORM and run whenever you need it. Stored queries are useful for developing ENFORM queries that will be used repeatedly by different people in your installation.

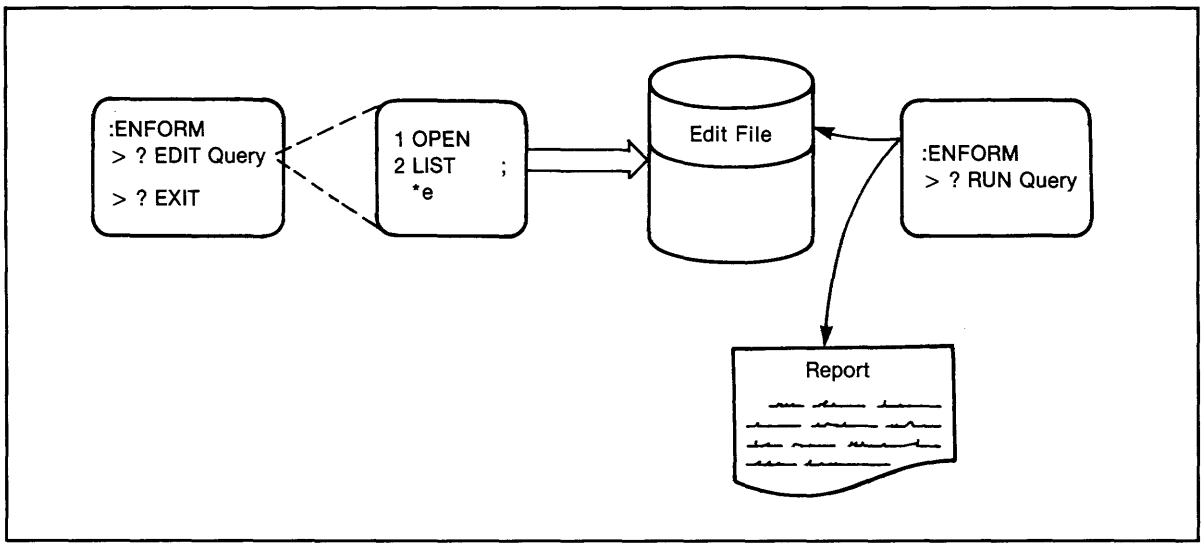


Figure 2-3. Stored Queries

Compiled Queries

A compiled query is illustrated in Figure 2-4. In this type of query, ENFORM specifications in an edit file are compiled into an ENFORM object file; the object file is saved and can be run from a TAL, COBOL, or FORTRAN program.

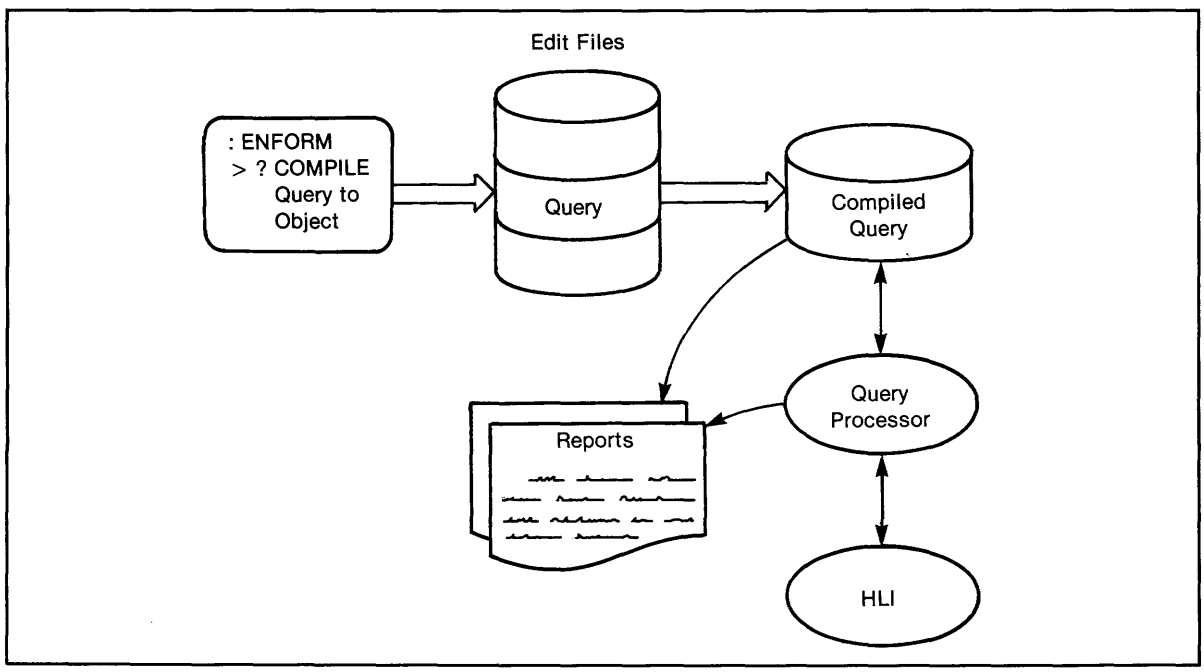


Figure 2-4. Compiled Queries

FIVE EASY STEPS TO USING ENFORM

When you use ENFORM, your major concern is the information that ultimately appears on a report. That is, how do you make data move from the data base files to the report and how do you control the appearance of the information on the report? Although this process is potentially complicated, you can initially reduce it to a simple set of steps by: (1) visualizing the data files that you work with and (2) visualizing how to select and format information from the files.

Figure 2-5 illustrates the steps involved in producing a report, and the following briefly explains each of the steps.

Step 1 — identify the files (tables) to be used in the query.

Step 2 — relate files to form new views.

Step 3 — select fields (columns) of information from the files.

Step 4 — select records (rows) of information from the tables; the rows are generally selected according to criteria that you specify.

Step 5 — format the appearance and sequence of the selected information.

Figure 2-5 also introduces four terms that will be used in the rest of this publication: *fields, records, files, and items.*

A field is a storage area for one item of information. Fields have names and occupy a specific location in relation to other fields. In Figure 2-5, the field names are the names of the columns; for example, partnum indicates a column of digits that form the part numbers for different parts in the data base. Records are groups of fields. Each record is named and occupies a specific location in relation to other records. In Figure 2-5, each record is a row in the table and contains information related to a specific part. A file is a collection of records; it is named and contains data about a group of related items; for example, parts. An item is something that ultimately appears on a report; it can be a field, a group of fields, or a derived item that is temporarily created by ENFORM while producing your report; for example, the sum of two fields.

The following pages discuss the ENFORM language specifications that enable you to produce a report and the "Using ENFORM To Produce Complete Reports" section of this publication concentrates, in more detail, on the major tasks involved in producing a report.

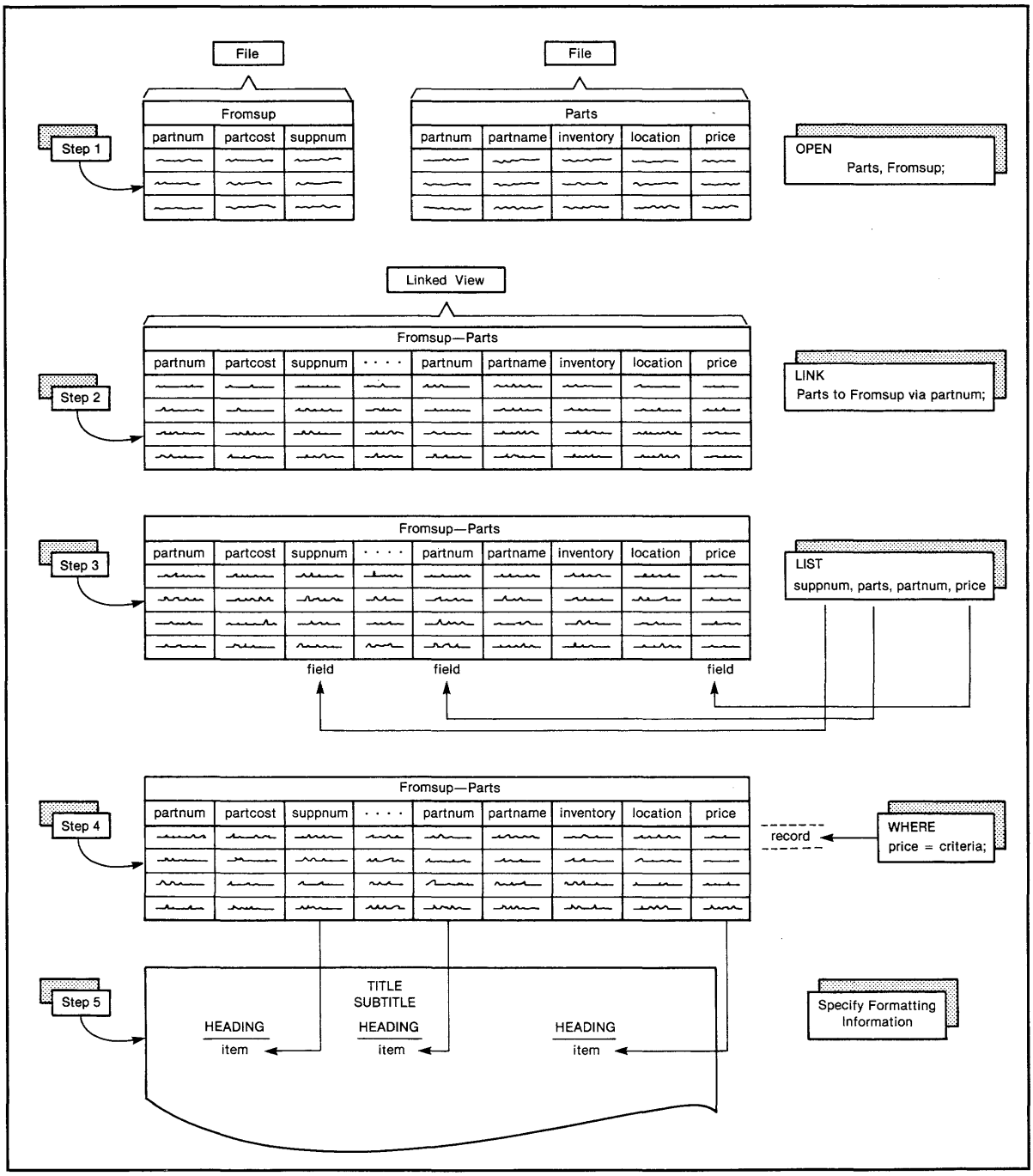


Figure 2-5. Five Steps to Using ENFORM

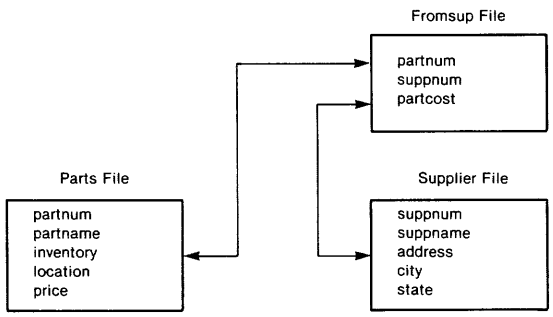
STATEMENTS, CLAUSES, AND COMMANDS

An ENFORM session (for any type of query) consists of executing:

- statements ..... the basic query specifications for selecting and formatting items from the data base
- clauses ..... optional parts of statements that provide added functions to augment the statements
- commands ..... specify operational details for the session.

The preceding components make up an ENFORM query; that is, specifications that will be executed by ENFORM to select data and produce reports.

The rest of this section shows how to use simple statements, clauses, and commands to produce ENFORM reports. All of the following examples (and the examples in Section 4) are based on the files illustrated in Figure 2-6 below.



Parts File Records

PARTNUM	PARTNAME	INVENTORY	LOCATION	PRICE
212	SYSTEM 192 KB CORE	7	J87	92000.00
244	SYSTEM 192 KB SEMI	3	B78	87000.00
4102	DISC 50 MB	9	K45	14500.00
4103	DISC 160 MB	7	K43	24500.00
6201	SYNC CONTROLLER	-16	A34	5800.00
7102	CABINET LARGE	20	F76	68000.00
7301	POWER MODULE	32	H76	2400.00

Fromsup File Records

PARTNUM	SUPPNUM	PARTCOST
212	1	92000.00
244	1	87000.00
4102	6	14500.00
4103	6	24500.00
6201	1	5800.00
7102	10	68000.00
7301	1	2400.00

Supplier File Records

SUPPNUM	SUPPNAME	ADDRESS	CITY	STATE
1	TANDEM COMPUTERS	19333 VALLCO PKWY	CPRTNO	CA.
6	INFORMATION STORAGE	1000 INDUSTRY DR.	LEXNTN	MASS
10	STEELWORK INC	6000 LINCOLN LANE	SNVALE	CA.

Figure 2-6. Parts, Fromsup, and Supplier Files

NOTE

The examples in this section show all statements terminated by a semicolon (;). With the exception of the LIST statement, this practice is not required by ENFORM syntax rules, but it does help you fix syntax errors—any error message produced for a statement will be displayed immediately after the statement if it is terminated by a semicolon.

Statements

ENFORM statements provide the basic query specifications of the program to be executed. The main ENFORM statement is LIST which specifies what information should appear on a report, and prints the formatted report to (usually) a terminal or printer; for example,

```
:ENFORM /out name of file/name of dictionary subvolume
>.....
>LIST parts;
```

will select and print all data items in the *parts* file. The report will be printed on the output device specified by the *out* parameter.

Alternatively, entering

```
>LIST partname,partnum,price;
```

in place of LIST parts, in the preceding example, will select and print the named items: part names, part numbers, and price from the *parts* file.

Additional ENFORM statements establish the query environment (specific conditions for processing the query) and provide some basic report formatting specifications.

The most often used environment statements are OPEN, which specifies file descriptions for the files to be processed by the query, and LINK, which specifies a connecting relationship between files. For example,

```
>OPEN parts,fromsup;
>LINK parts to fromsup VIA partnum;
```

will cause ENFORM to access the dictionary description of the parts and fromsup files and to produce a new relation that connects the files for each *partnum* that is identical in both file descriptions.

Typical report formatting statements are TITLE, which prints the title you specify at the top of each page of the report and SUBTITLE, which prints a subtitle, immediately following the title, at the top of each page. For example,

```
>TITLE "Inventory Report" CENTER;
>SUBTITLE "XYZ INC." CENTER;
```

will print and center the words "Inventory Report" on one line followed by "XYZ INC." on the next line at the top of each page for all reports produced during the session.

Summary of Statements

By entering four of the preceding statements as follows:

```
>OPEN parts;
>TITLE "Inventory Report" CENTER;
>SUBTITLE "XYZ INC."      CENTER;
>LIST partname,partnum,price;
```

you can use ENFORM to produce a report like Figure 2-7.

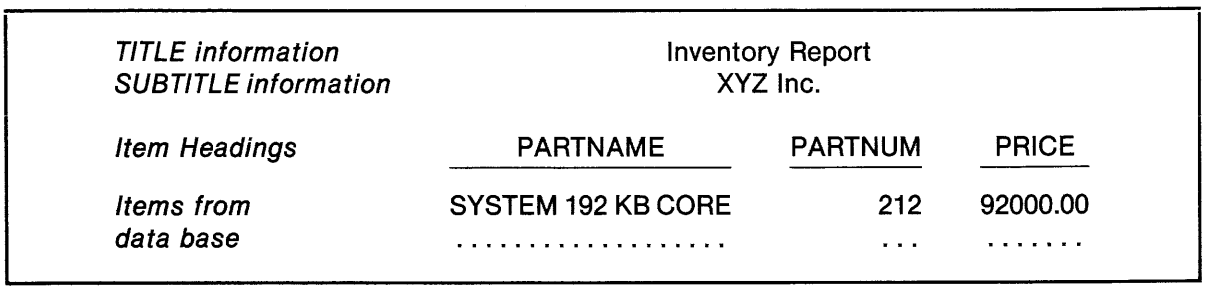


Figure 2-7. Example of Simple Formatted ENFORM Report

This is an example of using ENFORM to do a lot of work with a few specifications. Clearly, this is a simple example, but it does show the basic components of practically every ENFORM query.

Typically, you will use ENFORM to produce a report that:

- Prints data items sorted by groups; by supplier number, for example.
- Prints different headings for the columns of the report.
- Calculates and prints subtotals for the groups and totals for all groups; for example, the cost of all items in inventory from a particular supplier.
- Selects and prints items that meet some criteria like "inventory greater than $2500."

See Figure 2-10 for an example of this type of report.

Note that the word "item" simply means a specific piece of information from the data base; for example, part number *100212*.

ENFORM clauses, which are discussed in the following pages, enable to you do the preceding report by augmenting the basic statements.

Clauses

Clauses are optional components of ENFORM statements; they specify additional data selection and formatting criteria. This section discusses three types of clauses:

1. field selection and sorting
2. item formatting
3. new items.

FIELD SELECTION AND SORTING CLAUSES. These clauses can be used to specify selection criteria for the data that will contribute to the report and to specify how the data should be sorted. Typical clauses in this group are WHERE, BY, and ASCD.

The WHERE clause allows you to filter the data items that appear in the report. This clause is included as part of the LIST statement and combined with a logical expression such as GREATER THAN or LESS THAN to specify the selection criteria. For example, entering:

```
OPEN parts;
LIST parts WHERE price LESS THAN 20000;
```

will select and print only those inventory items that are priced less than $20000.00.

The BY clause allows you to sort data items, for the report, based on the value of a specified data item. A new group will be printed or displayed each time the data item changes. For example, if you specify *BY suppnum*, the report will be formatted in groups for each individual supplier number in the file and the printing of duplicate values of the supplier number will be suppressed. BY is included as part of the LIST statement.

The ASCD clause allows you to specify that an item should appear on the report in ascending (lowest to highest) order.

An example of using WHERE, BY, and ASCD is entering:

```
OPEN parts,fromsup;
LINK parts to fromsup VIA partnum;
LIST BY suppnum,
     parts.partnum,
     ASCD inventory,
     WHERE inventory GREATER THAN 0;
```

to produce a report like the one illustrated in Figure 2-8.

Group Item Headings and Item Headings	SUPPNUM	PARTNUM	INVENTORY
Items from database	1	244	3
		212	7
		7301	32
Items from database	6	4103	7
		4102	9
	.	....	.

Figure 2-8. Example of Grouping and Sorting Items in a Report

In this report:

- The rows are sorted (grouped) by *suppnum* from the *fromsup* file and the printing of duplicate values of the supplier number is supressed.
- Within a supplier number group, the items are sorted by inventory.
- *partnum* had to be entered as *parts.partnum* in the query. The reason for this qualification is that *partnum* appears in both files and you have to identify which file to select it from.
- Only those inventory items that are greater than 0 in the *inventory* field of the *fromsup* file are printed.

ITEM FORMATTING CLAUSES. These clauses can be used to specify display formats and column headings for items that appear in the report. One item formatting clause is HEADING which prints a column heading of your choice for each column of data items in the report. The default for the heading is the item name specified in the dictionary (through DDL). If you enter

```
LIST partnum HEADING "PART/NUMBER";
```

the heading at the top of the part number column will be PART
NUMBER
on two lines (the "/" means start a new line).

NEW ITEMS CLAUSES. One of the things that you can do with ENFORM is to use new item clauses to temporarily create new data from existing data and to produce the data on a report; for example, to print the sum of two fields, in the data base, under the column heading SUM. Typical new item clauses are SUBTOTAL, TOTAL, arithmetic expression clauses such as "*" (multiply), and aggregate clauses such as AVG.

The TOTAL clause allows you to specify that the total for any column of numeric data items should be produced on the report. And the SUBTOTAL clause produces the subtotal for any group of numeric data items. Both TOTAL and SUBTOTAL are included as part of the LIST statement.

Arithmetic expression clauses can be combinations of data items and numeric literals that can be added, subtracted, or multiplied to produce new values.

Aggregate clauses specify that one value should be computed (by ENFORM) for a set of items; for example, the average price of all parts supplied by a particular supplier.

An example of using SUBTOTAL, TOTAL, "*", and AVG is entering:

```
OPEN parts,fromsup;
LINK parts to fromsup VIA partnum;
LIST BY suppnum,
        parts.partnum,
        inventory,
        price,
        (price*inventory) HEADING "TOTAL COST",
        SUBTOTAL over suppnum,
        TOTAL,
        AVG ((price*inventory) OVER suppnum);
```

to produce a report like the one illustrated in Figure 2-9.

SUPPNUM	PARTNUM	INVENTORY	PRICE	TOTAL COST	AVG
1	212	7	92000.00	644000.00	222250.00
	244	3	87000.00	261000.00	
	6201	-16	5800.00	-92800.00	
	7301	32	2400.00	76800.00	
*subtotal for 1st supplier				889000.00	
6	4102	9	14500.00	130500.00	151000.00
	4103	7	24500.00	171500.00	
*subtotal for next supplier				302000.00	
10	7102	20	6800.00	136000.00	136000.00
*subtotal for next supplier				136000.00	
total for all suppliers				1327000.00	

Figure 2-9. Example of Item Formatting and New Items on a Report

In this report:

- A new data item that is the multiple of the price and inventory items is calculated and printed with a heading of TOTAL COST for the column of new items.
- A subtotal for the TOTAL COST items is calculated and printed whenever the item *suppnum* changes.
- A total for the TOTAL COST items is calculated and printed at the end of the report.
- The average value of all items in stock from a particular supplier is calculated and printed for each supplier.

The next page contains an annotated example of using the preceding clauses and the statement discussed earlier to produce a complete report.

Summary of Clauses

The previous clauses can be combined with their respective ENFORM statements and entered as follows:

```
OPEN parts,fromsup;                                    0
LINK parts TO fromsup VIA partnum;                     1
TITLE "Inventory Report" CENTER;                       2
SUBTITLE "XYZ Inc." CENTER;                            3
LIST by suppnum HEADING "SUPPLIER/NUMBER",             4
     parts.partnum heading "PART/NUMBER",              5
     inventory HEADING "QUANTITY/IN STOCK",            6
     price HEADING "COST",                             7
     (price*inventory) HEADING "TOTAL/COST"            8
     SUBTOTAL OVER suppnum,                            9
     TOTAL,                                            10
     AVG ((price*inventory) OVER suppnum);             11
```

to produce the report illustrated in Figure 2-10.

In this example, the statements specify the following:

0. Record descriptions of the files involved in the query.

1. Produce a new view by linking the descriptions.

2. The title "Inventory Report" should be printed and centered.

3. The subtitle "XYZ INC." should be printed and centered.

4. Sort the report by the field *suppnum*, print this field (for each record in the linked file) with a heading of "SUPPLIER NUMBER", and suppress printing of duplicate values of this field.

5. Print the *partnum* field (for each record in the linked file) with a heading of "PART NUMBER".

6. Print the *inventory* field (for each record in the linked file) with a heading of "QUANTITY IN STOCK".

7. Print the *price* field (for each record in the linked file) with a heading of "COST".

8. Calculate a new item that is the multiple of the *price* and *inventory* fields and print it (for each record in the the linked file) with a heading of "TOTAL COST".

9. Each time the *suppnum field* changes print a subtotal for all items in the "TOTAL COST" column of the report.

10. Print a total for all items in the "TOTAL COST" column.

11. For each unique supplier number, calculate and print the average of the "TOTAL COST" items.

Inventory Report
XYZ Inc.

	SUPPLIER NUMBER	PART NUMBER	QUANTITY IN STOCK	COST	TOTAL COST	AVG
	1	212	7	92000.00	644000.00	222250.00
		244	3	87000.00	261000.00	
		6201	-16	5800.00	-92800.00	
		7301	32	2400.00	76800.00	
*					889000.00	
	6	4102	9	14500.00	130500.00	151000.00
		4103	7	24500.00	171500.00	
					302000.00	
	10	7102	20	6800.00	136000.00	136000.00
*					136000.00	
					1327000.00	

Figure 2-10. Example of Typical ENFORM Report

COMMANDS

If you have read this far, you should have an overview of ENFORM language components. So, how do commands fit into the idea of a language composed of ENFORM statements?

Commands are unconditional one-word imperatives (to ENFORM) to indicate that you want to do something different within a session—something not necessarily connected with specifying information on the report. All commands are preceded by "?" and you can, for example, use them to:

- Display some of the current settings for ENFORM statements like LINK and OPEN (?SHOW command).
- Use the editor from an ENFORM session to create a source file that can be saved (?EDIT).
- Stop the current ENFORM session (?EXIT).
- Specify a listing device for the report (?OUT).
- Execute an ENFORM source program stored in an edit file (?RUN).

An example of using commands is entering:

```
:ENFORM
>?EDIT newquery!
CURRENT FILE IS VOLUME.SUBVOLUME.NEWQUERY
*XVS F
     OPEN parts
     LIST parts;
*e
>?RUN newquery
```

which will:

- Allow you to use the editor (within an ENFORM session) to create a source file named *newquery* with two ENFORM statements.
- Save the source file.
- Run the source file to produce a report.
- Continue with the ENFORM session.

See the *ENFORM Reference Manual* for a complete description of ENFORM commands.

SECTION 3

HOW ENFORM WORKS

In Section 2, the ENFORM language was introduced by showing how different types of query specifications produced different types of information on a report. This section will expand that idea by *conceptually* showing how a query is processed by ENFORM to produce a report. The purpose of this section is to enhance your understanding of ENFORM and to help prepare you for using the more detailed information in the *ENFORM Reference Manual.*

Skip this section if you are primarily interested in writing ENFORM queries.

Figure 3-1 is an overview of the ENFORM environment: the ENFORM processes, ENFORM files, and data files involved in query processing and report writing. The sequence of steps in the figure is explained in the remainder of this section.

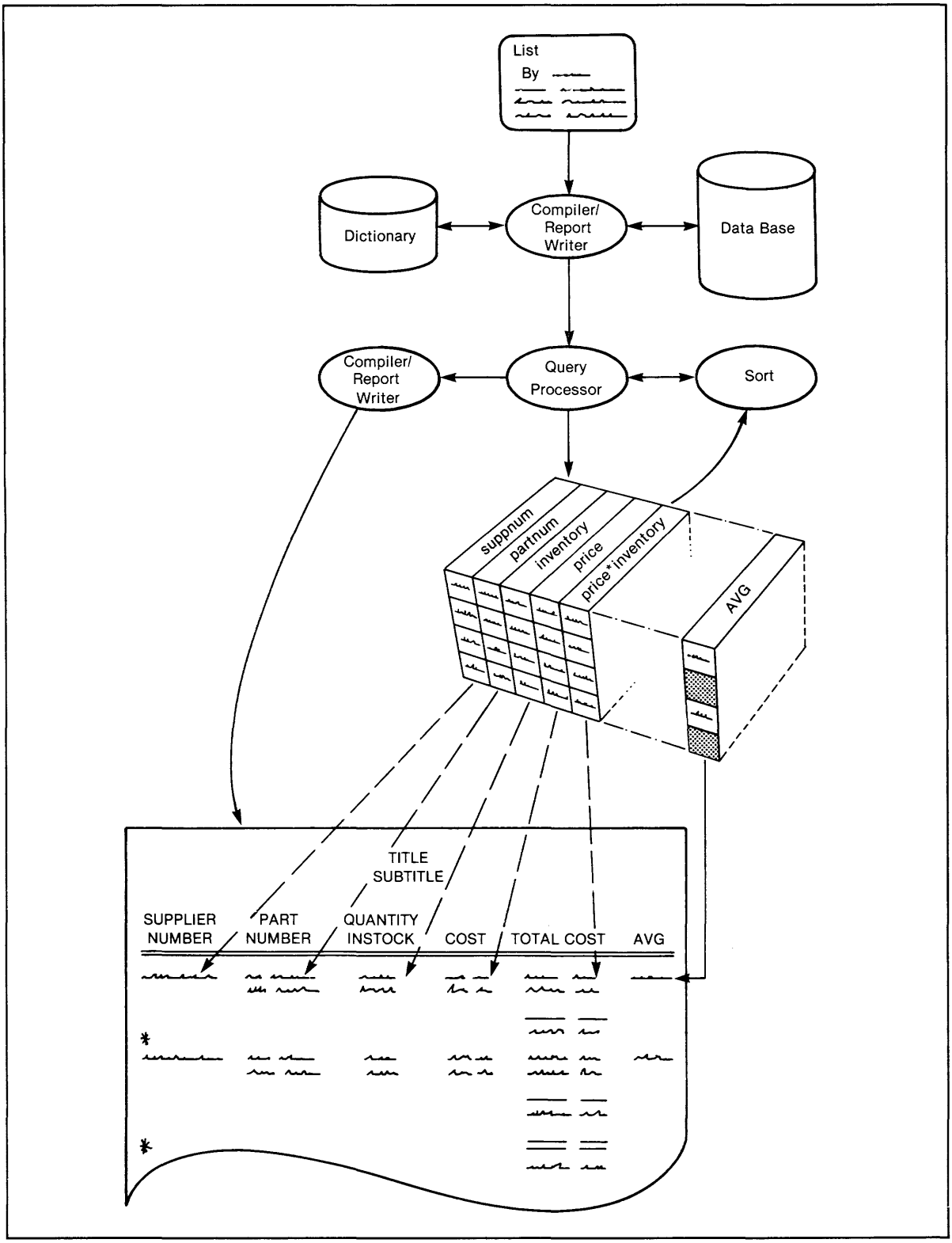


Figure 3-1. ENFORM Operational Environment

Figure 3-1 illustrates that the following sequence of steps is involved in processing an ENFORM LIST query:

1. Query compilation by the compiler/report writer process.

2. Transfer of control and dictionary-derived file information to the query processor: a server process that accesses the data base files specified in the query.

3. The production of a target (work) file by the query processor. The target file is the result of the query; it contains items of information (both from the data base and derived) that will appear on the report.

4. Transfer of control and the target file back to the compiler/report writer process to format and produce the report.

Basically, these steps illustrate that you should think of ENFORM as providing two logical functions: (1) report formatting and writing, and (2) query processing where the result of the query is a target file that will be processed by the compiler/report writer in step 4.

COMPILING THE QUERY

The compiler/report writer process has two logical functions or phases: (1) compiling the query and (2) formatting and writing the report. The purpose of the first phase is to ensure that the query processsor receives the information that it needs to process the query.

In the compilation phase, your query specifications are first checked for syntactical correctness: do they obey the rules of the language? Any errors detected at this point will stop query processing and result in a syntax error for the query.

If the source specifications are correct, they are transformed into an intermediate form that can be intepreted by the query processor. The intermediate form is a compiled representation of the query. It contains information:

- from the dictionary about the files and fields needed to build the target file

- that will be used to construct target records by the query processor; every target record looks like a template of the specifications in the query

- about related files

- about qualifications in the query specification that will limit the numbers of records read from the data base and written to the target file.

The compiled query is then sent to the query processor (as a message) using standard interprocess i/o.

The compiler/report writer also builds a table containing information that will allow it to format and produce the report during its report writing phase—when it receives control back from the query processor.

TRANSFERRING CONTROL TO THE QUERY PROCESSOR

The query processor is a server process that interprets the compiled query and accesses the data base to build a target file. It can occur in one or both of the following forms:

1. As a named server process that can be shared (sequentially) by several ENFORM processes. This type of server is persistent (it exists until it is stopped), has a backup process, and handles queries one at a time. To start this type of server, the system manager uses the QP command and the Command Interpreter commands ASSIGN and PARAM. Your ENFORM program can then access it by specifying its name in a ?ATTACH command.

2. As a dedicated server process that is created for and provides services to an individual ENFORM process; this process is created each time an individual ENFORM program is run.

There are two reasons for using named query processors instead of dedicated query processsors.

First, they maximize effective use of the system resources that are being shared by all applications in your installation. Starting and stopping dedicated processes is generally an expensive use of resources. It should be avoided whenever possible. Additionally, you can balance the system load by moving the named server to a lightly loaded cpu to increase its performance or off a heavily loaded cpu to increase the performance of other processes running on that cpu.

Second, they can be used for efficient processing of network distributed queries as discussed in Section 1. Simply start the query processor at the network node where most of the data resides.

Dedicated query processors, of course, will generally process an individual query quicker since they are not being shared.

BUILDING THE TARGET FILE

The target file, which is produced by the query processor, is the result of your query—before the data is formatted and generated as a report. In its final form, you can think about it as a collection of records that:

- are built by taking data out of the different files and putting it together in one file
- are sorted in the order that they will ultimately appear on the report
- contain all of the specified fields from the data base; again, the fields are in the order that they will appear on the report
- contain derived data items that are the results of arithmetic expressions clauses, and aggregate clauses; in this respect the target file can be viewed as a construct that allows calculations to be done over a group.

Figure 3-2 conceptually illustrates how the target file would be produced for the following type of query:

```
OPEN parts,fromsup;
LINK parts to fromsup VIA partnum;
LIST BY suppnum,
        parts.partnum,
        inventory,
        price,
        (price*inventory),
        SUM (price*inventory OVER suppnum),
        SUM (price*inventory OVER ALL),
        AVG ((price*inventory) OVER suppnum),
        WHERE inventory GREATER THAN 0;
```

The target file records in Figure 3-2:

- are sorted by supplier number—as a result of the BY clause
- contain fields for the suppnum, parts, price, and inventory items specified in the query
- contain a derived field (the product of price and inventory)
- contain fields that are the result of doing calculations such as AVG for the groups of records in the target file.
- do not contain target records for any records in the data base files where the *inventory* field is less than or equal to 0.

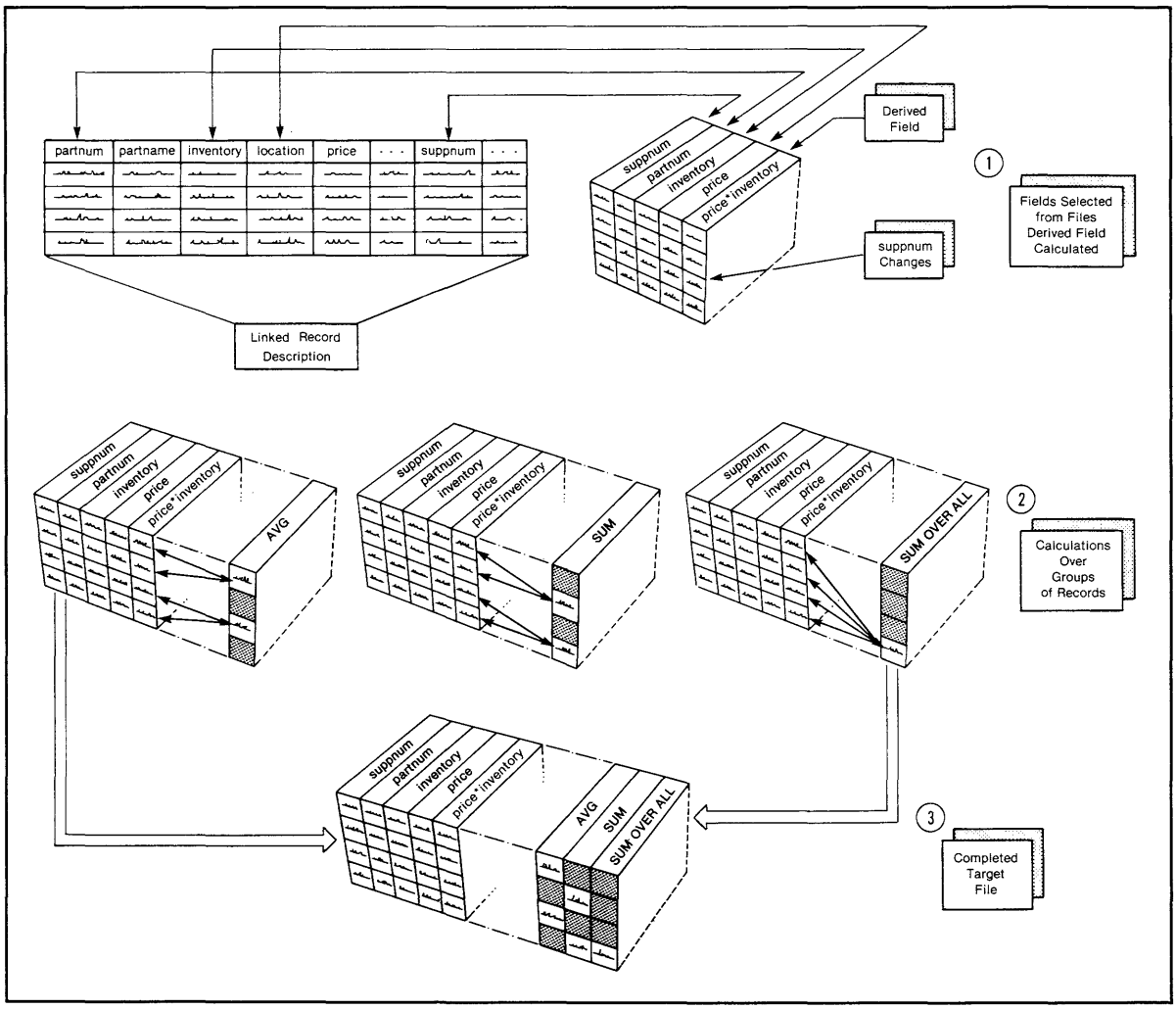


Figure 3-2. Building The Target File

Note that in this example, the target file is an intermediate file used by ENFORM to process the query. It is not a named physical file that you can access. However, if you use an ENFORM FIND statement in the query, the result of the query can be a physical output file that is structured like the target file.

To use the FIND statement, first create a DDL record description that looks like the target file and add it to the dictionary; then use FIND instead of LIST in the query specifications. The output file, produced by the FIND statement, can be processed by a host language program using ENFORM procedures.

See the *ENFORM Reference Manual* for more details on using the FIND statement.

REPORT WRITING

The second phase of the compiler/report writer process occurs when it receives control back from the query processor. As illustrated in Figure 3-3, the compiler/report writer:

- adds titles, subtitles, and headings to the report
- extracts data from the target file and adds it to the report
- does any other report formatting specified in the query
- writes the report to the output device specified in the query.

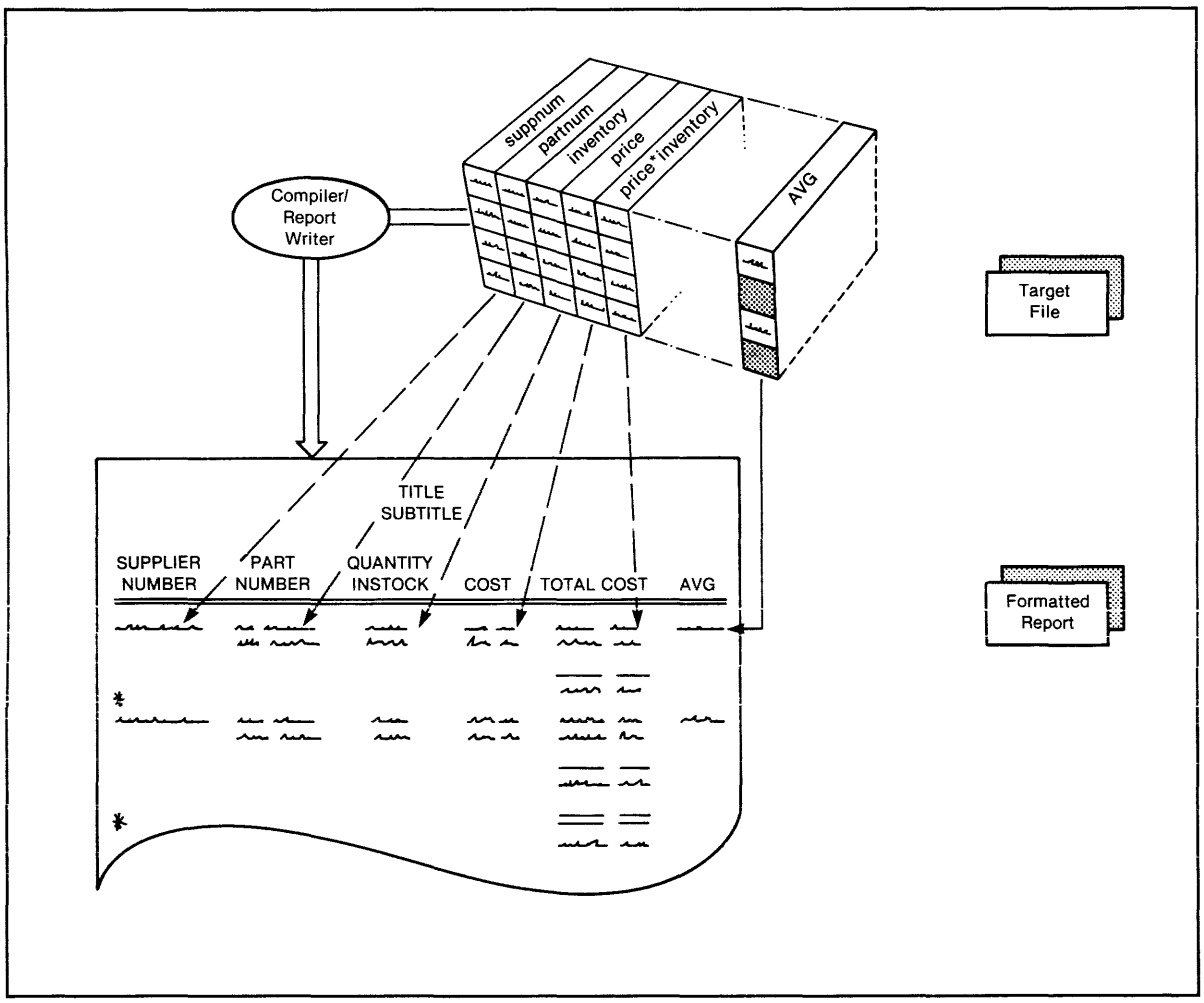


Figure 3-3. Report Writing Phase of Compiler/Report Writer

SECTION 4

USING ENFORM TO PRODUCE COMPLETE REPORTS

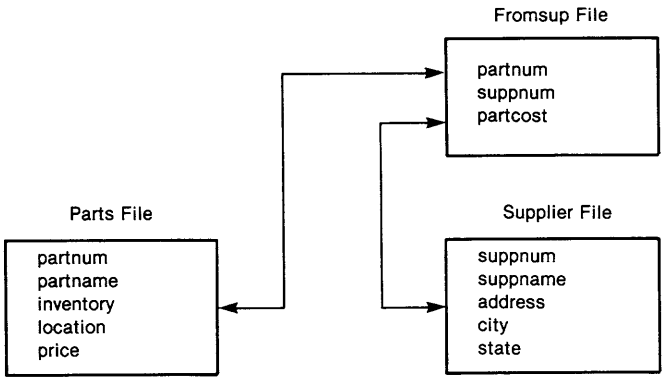
Section 2 described some of the basic things that you can do with ENFORM. It concentrated on a language view of ENFORM; that is, what are the specifications of the language and how are they used?

This section is a simple tutorial on using ENFORM. It extends Section 2 by concentrating on the tasks that you do to produce a complete report. The basic tasks (and their related specifications) that will be described are:

- starting ENFORM
- opening dictionary descriptions of files used in the query
- listing information from a file onto a report
- restricting the information that appears on a report
- linking files
- controlling the appearance of the report
- specifying report arithmetic.

Figure 4-1 illustrates the files and their relationships that will be used in all examples shown in this section.

Note that this section is not a substitute for the information in the *ENFORM Reference Manual* which should be used to determine the syntax and detailed considerations for all ENFORM specifications.



Parts File Records

PARTNUM	PARTNAME	INVENTORY	LOCATION	PRICE
212	SYSTEM 192 KB CORE	7	J87	92000.00
244	SYSTEM 192 KB SEMI	3	B78	87000.00
4102	DISC 50 MB	9	K45	14500.00
4103	DISC 160 MB	7	K43	24500.00
6201	SYNC CONTROLLER	-16	A34	5800.00
7102	CABINET LARGE	20	F76	6800.00
7301	POWER MODULE	32	H76	2400.00

Fromsup File Records

PARTNUM	SUPPNUM	PARTCOST
212	1	92000.00
244	1	87000.00
4102	6	14500.00
4103	6	24500.00
6201	1	5800.00
7102	10	6800.00
7301	1	2400.00

Supplier File Records

SUPPNUM	SUPPNAME	ADDRESS	CITY	STATE
1	TANDEM COMPUTERS	19333 VALLCO PKWY	CPRTNO	CA.
6	INFORMATION STORAGE	1000 INDUSTRY DR.	LEXNTN	MASS
10	STEELWORK INC	6000 LINCOLN LANE	SNVALE	CA.

Figure 4-1. Fromsup, Parts, and Supplier File Relationships

STARTING ENFORM

Since ENFORM is a program, you must start it before it can be used to produce a report. Once a program is started and has control of your terminal, it displays a prompt character at the left of the next line on the screen. Each program has its own prompt character; the Command Interpreter uses a colon (:); the Text Editor uses an asterisk (*); and ENFORM uses a greater than symbol (>). While using ENFORM you will normally have one of these prompts on the screen.

Generally, the Command Interpreter has initial control of your terminal and you can start ENFORM by responding to the colon prompt character with:

```
:ENFORM
```

The system accesses the ENFORM program, creates an ENFORM process, and displays the following message and a prompt character:

```
ENFORM - T9102xxx (yyyyyyy) DATE - TIME : 9/01/81 - 13:59:15

>
```

where xxx is the level of the executing version of ENFORM and yyyyyyy is the release date of that level.

There are several options that you can specify with the ENFORM command. You can specify the name of the subvolume that holds the data dictionary that is to be used:

```
:ENFORM $data.dictiony
```

If no subvolume is specified for the dictionary, ENFORM assumes that the current default subvolume is to be used.

There are times when you want to start ENFORM, run reports from an edit file, and exit. In cases like this, you can specify the edit file containing the ENFORM source code and the output device or file to which the report goes. These two functions are performed with the *in* and *out* parameters of the Command Interpreter command line.

```
:ENFORM /in $data.rpt.rpt1,out $s/ $data.dictiony
```

This command line causes ENFORM source code to be read from the edit file named with the *in* parameter and output to be sent to the spooler which is named $S in this system. When the processing is done, but before the report is printed, the ENFORM process finishes and control returns to the Command Interpreter.

OPENING A FILE

Before files can be used by ENFORM, their file descriptions (in the dictionary) must be identified and opened. File descriptions have the same name as the physical files they describe. Opening a description involves identifying the files to be used in the query and initializing pointers to make it possible for a program to read and write a file. Think of this procedure as being similar to opening a file drawer so that folders can be put into or taken out of the drawer. Opening a file description is accomplished by entering an ENFORM OPEN statement and the names of the descriptions to be opened.

You will get error messages if you try to use a file description that has not been opened; ENFORM will not be able to find any field names within the file.

The simple form of the OPEN statment is the word OPEN followed by the name of the description to be opened:

```
>OPEN parts;
```

Multiple descriptions can be opened with separate OPEN statements or with a single OPEN statement in which the names are separated with spaces or commas:

```
>OPEN parts;
>OPEN fromsup;
>OPEN supplier,order,odetail;
```

The files remain open until explicitly closed with a CLOSE statement, implicitly closed by a DICTIONARY command, or the ENFORM session is ended. It is advisable to close all files you no longer need after using them.

The CLOSE statement is as simple as the OPEN statement. It is the word CLOSE followed by the file description name separated by commas or spaces:

```
>CLOSE region,parts;
```

LISTING INFORMATION FROM A FILE ONTO A REPORT

As you already know, the purpose of ENFORM is to generate reports based on the content of data files. ENFORM lets you select the data, organize it, and structure a report by entering specifications. There are several ENFORM commands, statements, and clauses that make up complete ENFORM specifications, but the LIST statement is the core of the product. It is the LIST statement that specifies what is to be reported.

The simplest form of the LIST statement generates a list of an entire file. If headings for the individual data items are included in the data dictionary they are included in the report. Otherwise, the names of the individual fields are used as headings. If formatting controls are included in the dictionary, the report items are formatted.

The LIST statement consists of the word LIST followed by the name of an open file and ending with a semicolon (;). Because a LIST statement can run to several lines when its full capability is used, it must be terminated with a semicolon. Without a terminator, ENFORM does not know when LIST ends. Note that other statements such as OPEN do not require a terminator. However, the examples in this section, show all statements terminated with a semicolon. Following this practice will help you fix syntax errors.

Figure 4-2 shows the listing of a complete file from the data base (see Figure 4-1) used throughout this publication. The following is the ENFORM commands and statements that generated the report:

```
>OPEN fromsup;
>LIST fromsup;
```

SUPPNUM	PARTNUM	PARTCOST
1	212	92000.00
1	244	87000.00
6	4102	14500.00
1	6201	5800.00
6	4103	24500.00
10	7102	6800.00
1	7301	2400.00

Figure 4-2. Listing of a Single File

The report in Figure 4-2 is the complete listing of the file named *fromsup.* There are seven records in the file, one for each part. Each record has three fields: the supplier number, the part number, and the cost. The meaning of each field may not be obvious from the report because the data names for the fields are used as headings. Later, you will learn how to make reports more understandable.

Figure 4-2 shows a characteristic of ENFORM reports that helps in formatting complete reports. The fields that contain numeric values are aligned to the right. Additionally, if the report contained character strings, they would be aligned to the left. The alignment is not based on the characters in the field but on the definition of the field in the data dictionary. For example, a character string may contain only numbers (the digits 0 through 9). Even though the string looks like a number, it will be aligned to the left.

Listing Parts of a File

Listing the entire contents of a single file is not very useful except in a maintenance situation where you have to check all records in a file. Usually, you want to list certain fields in a certain order when they meet specific criteria.

The LIST statement can specify the fields within a file rather than a whole file. When you specify field names rather than the file name, only the specified fields are listed. For example, you can generate a report similar to the one in Figure 4-2, but containing only the supplier number and the part number of a part. This report is shown in Figure 4-3. The following ENFORM statements will generate the report:

```
>OPEN fromsup;
>LIST suppnum, partnum;
```

SUPPNUM	PARTNUM
1	212
1	244
6	4102
1	6201
6	4103
10	7102
1	7301

Figure 4-3. A Simple Report by Fields

Notice that the fields are printed in the order specified by the LIST statement.

Sorting a File Listing

The report in Figure 4-3 could be hard to use if you scan it by the supplier number. The numbers are in the order in which they occur in the file. If the list was very long it would be quite difficult to search the list for a specific supplier number/part number relationship.

To help the report reader in this kind of situation, ENFORM provides the capability of sorting the report into ascending or descending order by providing clauses that augment the LIST statement. Report sorting is controlled by the ASCD, DESC clauses, and BY clauses. ASCD specifies that a report be sorted by a field in ascending order. DESC specifies sorting in descending order. BY causes the report to be sorted by a specified field and additionally suppresses the printing of subsequent occurrences of that field with like values.

The ASCD or DESC clauses are placed before the field name that is to be sorted. For example to sort a report in ascending order by supplier number, you enter:

```
>LIST ASCD suppnum,partnum,partcost;
```

which will generate the report in Figure 4-4.

SUPPNUM	PARTNUM	PARTCOST
1	212	92000.00
1	244	87000.00
1	6201	5800.00
1	7301	2400.00
6	4102	14500.00
6	4103	24500.00
10	7102	6800.00

Figure 4-4. A Simple Report Sorted by a Field Value — Ascending

In Figure 4-4, one of the sorted values—supplier number—is duplicated in several records. When there is duplication of values, it is often useful to use the BY clause (in place of ASCD) to include the first occurrence of the value in the report and leave the remaining values blank. For example, entering:

```
>LIST BY suppnum,partnum,partcost;
```

will generate the report in Figure 4-5.

SUPPNUM	PARTNUM	PARTCOST
1	212	92000.00
	244	87000.00
	6201	5800.00
	7301	2400.00
6	4102	14500.00
	4103	24500.00
10	7102	6800.00

Figure 4-5. Sorting Multiple Value Occurrences with BY

You can use the BY modifier with descending values as well as ascending values. When dealing with values you want in ascending order, you must specify ASCD or BY. (An error occurs if you specify both ASCD and BY for the same field.) When requesting a sort in descending order with suppressed printing of duplicate values, you must specify BY and DESC. For example, to generate the report in Figure 4-5 in descending order you would use the following LIST statement:

```
>LIST BY DESC suppnum,partnum,partcost;
```

which will generate the report in Figure 4-6.

SUPPNUM	PARTNUM	PARTCOST
10	7102	6800.00
6	4102	14500.00
	4103	24500.00
1	212	92000.00
	244	87000.00
	6201	5800.00
	7301	2400.00

Figure 4-6. Sorting Multiple Value Occurrences with BY DESC

RESTRICTING THE INFORMATION THAT APPEARS ON A REPORT

Occasionally, you need all of the information in a file or all of the fields of a particular type. More often, you'll need to restrict the report to information that meets certain qualifications; for example, parts that cost more than $90000. ENFORM provides two clauses that can be used to test a particular field in a file for a condition and then to either select that field for the report, ignore it, or print something else in place of the field. The clauses are WHERE and IF/THEN/ELSE. For example, entering:

```
>LIST BY suppnum,partnum,partcost,
      WHERE partcost GREATER THAN 9000;
```

will generate the report in Figure 4-7.

SUPPNUM	PARTNUM	PARTCOST
1	212	92000.00
	244	87000.00
6	4102	14500.00
	4103	24500.00

Figure 4-7. Qualifying a Report with WHERE

This report is the same as Figure 4-6, with one exception—all records where the partcost field did not satisfy the condition *greater than 9000* are not in the report.

WHERE does not restrict you to simple greater than or less than conditions; it can be followed by any logical expression that can be tested for a true or not true condition. These expressions can be complex and sophisticated, which allows you to be very precise about qualifying information to be selected. See the description of logical expressions in the *ENFORM Reference Manual* for a detailed description of how to construct a logical expression.

An example of using IF/THEN/ELSE to qualify information and to print alternate values for a field is entering:

```
>LIST BY suppnum,partnum,
     (IF partcost GREATER THAN 9000 THEN partcost ELSE BLANKS);
```

to generate the report in Figure 4-8.

SUPPNUM	PARTNUM	PARTCOST
1	212	92000.00
	244	87000.00
	6201	
	7301	
6	4102	14500.00
	4103	24500.00
10	7102	

Figure 4-8. Qualification Using IF/THEN/ELSE

In one way, this report is like the one in Figure 4-7—a condition has been tested (partcost greater than 9000) and records have been printed for fields that satisfy the condition. However, all of the other records in the data base have also been printed, and the partcost fields have been printed as blanks for records that did not satisfy the condition. Again, see the *ENFORM Reference Manual* for all of the ways you can use IF/THEN/ELSE in conjunction with logical expressions.

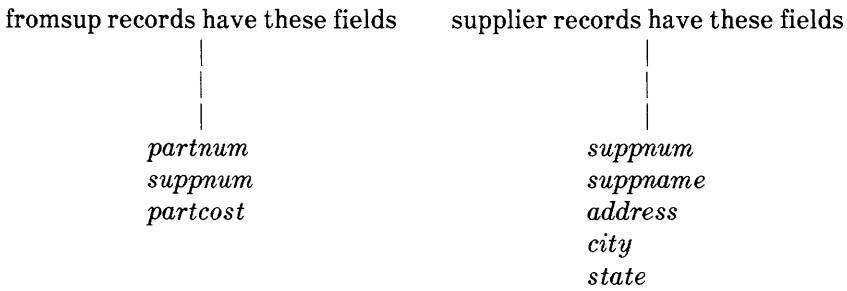
CONNECTING FILES

To this point, you have learned how to produce simple, qualified, *single-file reports.* Single-file reports are useful and a valid use of ENFORM, but they do not take full advantages of all the capabilities of ENFORM and relational data bases. Remember, one of the primary advantages of a relational data base is to allow you to work with either one file or multiple related files. Multiple files can be connected, logically, to satisfy your needs for information. This allows you to work with simple files—connecting them, when you have to, to form more complex views of the data base.

There are two ways that you can use ENFORM to connect files: (1) by qualification through the use of the WHERE clause and (2) by using the LINK statement.

Connecting Files By Using the WHERE Clause

Assume you have two files in the data base named fromsup and supplier.



By connecting the files, you can produce (for example) a report that contains the *partnum*, *partcost*, and *suppnum* fields from the fromsup file along with their matching *suppname* field from the supplier file. You do this by taking advantage of the fact that the field (data item) *suppnum* appears in both files.

An example of using the WHERE clause to connect the files is entering:

```
>OPEN fromsup,supplier;
>LIST BY fromsup.suppnum,BY suppname, partnum,partcost,
      WHERE fromsup.suppnum = supplier.suppnum;
```

to generate the report in Figure 4-9.

SUPPNUM	SUPPNAME	PARTNUM	PARTCOST
1	TANDEM COMPUTERS	212	92000.00
		244	87000.00
		6201	5800.00
		7301	2400.00
6	INFORMATION STORAGE	4102	14500.00
		4103	24500.00
10	STEELWORK INC	7102	6800.00

Figure 4-9. Connecting Files Using WHERE

The report in Figure 4-9 contains items from both files based on the connecting relationship expressed by the WHERE clause of the LIST statement. There are several things that you should notice about this LIST statement:

1. *suppnum* had to be qualified as fromsup.suppnum to identify which file to select it from. If you forget to do this and and try to list the field, ENFORM will generate the following message.

   ```
   Name not sufficiently qualified to avoid ambiguity:SUPPNUM
   ```

2. The BY clauses sort the report in ascending order by *suppnum* and by *suppname*, and suppresses the printing of multiple values of these fields.

3. The WHERE clause expresses the connecting relationship. Basically, it specifies that ENFORM should select items from both files only when the *suppnum* field of a record in the supplier file equals the *suppnum* field of a record in the fromsup file. Again, notice that the field names had to be qualified with their respective file names.

Using WHERE to connect files allows you to qualify the connection by using logical expressions to further define what information should appear on the report.

For example, entering:

```
>OPEN fromsup,supplier;
>LIST BY fromsup.suppnum,BY suppname, partnum,partcost,
      WHERE fromsup.suppnum = 1 AND
      fromsup.suppnum = supplier.suppnum;
```

will generate the report in Figure 4-10.

SUPPNUM	SUPPNAME	PARTNUM	PARTCOST
1	TANDEM COMPUTERS	212	92000.00
		244	87000.00
		6201	5800.00
		7301	2400.00

Figure 4-10. Connecting Files Using WHERE And Logical Expressions

In this report, the information has been more precisely defined by first testing the records to be selected to determine if they meet the condition—*suppnum* equal to 1. Otherwise, it is the same as the report in Figure 4-9.

Using the LINK Statement to Connect Files

The LINK statement is a convenient way of expressing a connecting relationship. Actually, LINK is just another form of a WHERE qualification with one difference—the linked file description is stored in an internal table and remains available for use in all subsequent LIST statements until it is cleared by an ENFORM command or the session is terminated.

To produce the report in Figure 4-9, using LINK instead of WHERE, you enter:

```
>OPEN fromsup,supplier;
>LINK fromsup TO supplier VIA suppnum;
>LIST BY fromsup.suppnum,BY suppname, partnum,partcost;
```

to generate the report in Figure 4-11.

SUPPNUM	SUPPNAME	PARTNUM	PARTCOST
1	TANDEM COMPUTERS	212	92000.00
		244	87000.00
		6201	5800.00
		7301	2400.00
6	INFORMATION STORAGE	4102	14500.00
		4103	24500.00
10	STEELWORK INC	7102	6800.00

Figure 4-11. Connecting Files Using LINK

In this example, the report is generated from a new logical description that has been produced by taking the value of the *suppnum* field of each record in the fromsup file and searching for an equivalent value in the *suppnum fields* of all records in the supplier file. Whenever there is a match, a record connecting all fields in both file records is produced for the logical description.

Since connected file descriptions occupy space in the internal table, they should be cleared whenever you are through using them. One way to clear them is to use the DELINK statement.

DELINK syntax looks like LINK syntax. For example, entering:

```
DELINK fromsup TO supplier VIA suppnum;
```

will clear the relationship established by:

```
LINK fromsup TO supplier VIA suppnum;
```

See the *ENFORM Reference Manual* for more information on the commands used to clear ENFORM's internal tables.

CONTROLLING THE APPEARANCE OF THE REPORT

ENFORM has a number of defaults that automatically control the appearance of information on the reports. These defaults control such things as headings for the columns, spacing between the columns, and alignment of data items within a column.

You can override these defaults by using the ENFORM clauses HEADING, SPACE, and CENTER. Each of these clauses is discussed with examples in the following pages. The examples show reports that use the parts files which has the following fields:

partnum partname inventory location price

Entering the following statements, which assume the ENFORM defaults, will produce the report in Figure 4-12.

```
OPEN parts;
LIST partnum,inventory,location;
```

PARTNUM	INVENTORY	LOCATION
212	7	J87
244	3	B78
4102	9	K45
4103	7	K43
6201	-16	A34
7102	20	F76
7301	32	H76

Figure 4-12. Standard Report Using ENFORM Formatting Defaults

In this report: numeric fields are right-aligned; headings are the defaults from the dictionary; character strings are left-aligned; and the spacing between columns is two spaces. These are the defaults. Now, let's change them to improve the readability of the report.

Specifying Column Headings

The column headings for the report in Figure 4-12 are not very meaningful since they use the field names from the dictionary description. There are two ways to generate meaningful headings on the report: (1) use the ENFORM HEADING clause, or (2) change the dictionary description of the record.

USING THE HEADING CLAUSE TO SPECIFY COLUMN HEADINGS. Entering

```
OPEN parts;
LIST partnum HEADING "PART NUMBER",
     inventory HEADING "ITEMS/IN STOCK",
     location HEADING "WAREHOUSE/LOCATION";
```

will generate the report in Figure 4-13.

PART NUMBER	ITEMS IN STOCK	WAREHOUSE LOCATION
212	7	J87
244	3	B78
4102	9	K45
4103	7	K43
6201	-16	A34
7102	20	F76
7301	32	H76

Figure 4-13. Standard Report With User-Specified Headings

In this report, ENFORM uses, as headings, the character string within the quotes and splits the string whenever it encounters a "/" character.

USING DDL TO ESTABLISH DEFAULT HEADINGS. The DDL record description for the *parts* file is:

```
RECORD parts.
     FILE IS "$mkt.sample.parts"  KEY-SEQUENCED.
  02 partnum        TYPE *.
  02 partname       PIC "X(18)".
  02 inventory      PIC "999S".
  02 location       PIC "XXX".
  02 price          PIC "999999V99";
KEY IS partnum.
KEY "pn" is partname.
END
```

By using the HEADING attribute of the DDL RECORD statement, the preceding description can be changed to establish meaningful default headings for the columns of fields or groups listed on an ENFORM report. Advantages of using this approach are: (1) all ENFORM reports that use the record description will have consistent headings and (2) it simplifies the query.

For example, to produce the column heading for the report in Figure 4-13 without using the ENFORM HEADING clause, modify the record description as follows.

```
RECORD parts.
     FILE IS "$mkt.sample.parts"  KEY-SEQUENCED
  02 partnum        TYPE * HEADING "part number".
  02 partname       PIC "X(18)".
  02 inventory      PIC "999S" HEADING "items/in stock".
  02 location       PIC "XXX" HEADING "warehouse/location".
  02 price          PIC "999999V99";
KEY IS partnum.
KEY "pn" is partname.
END
```

For more information about DDL, see the *Data Definition Language (DDL) Programming Manual.*

Using the CENTER Clause to Center Items Within a Column

ENFORM automatically right-aligns numeric items and left-aligns character strings unless you specify a different alignment with the CENTER clause.

For example entering:

```
OPEN parts;
LIST partnum HEADING "PART NUMBER" CENTER,
     inventory HEADING "ITEMS/IN STOCK",
     location HEADING "WAREHOUSE/LOCATION" CENTER;
```

will generate the report in Figure 4-14.

PART NUMBER	ITEMS IN STOCK	WAREHOUSE LOCATION
212	7	J87
244	3	B78
4102	9	K45
4103	7	K43
6201	-16	A34
7102	20	F76
7301	32	H76

Figure 4-14. Standard Report With Specific Data Items Centered

In this report, the items in the PART NUMBER and ITEMS IN STOCK column are centered and the items in the ITEMS IN STOCK column have defaulted to right-aligned.

If you prefer to center all data items on the report, the LIST statement can be conveniently entered as:

```
LIST partnum HEADING "PART NUMBER",
     inventory HEADING "ITEMS/IN STOCK",
     location HEADING "WAREHOUSE/LOCATION",
     CENTER ALL;
```

Using the Space Clause to Control Spacing Between Columns

Unless your printer or display screen is restricted in width, it is preferable to space the columns on the report instead of using the ENFORM default of two spaces between columns. Spacing columns with the ENFORM SPACE clause can improve the report's readability.

```
OPEN parts;
LIST partnum HEADING "PART NUMBER",
     SPACE 5 inventory HEADING "ITEMS/IN STOCK",
     SPACE 10 location HEADING "WAREHOUSE/LOCATION",
     CENTER ALL;
```

will generate the report in Figure 4-15.

PART NUMBER	ITEMS IN STOCK	WAREHOUSE LOCATION
212	7	J87
244	3	B78
4102	9	K45
4103	7	K43
6201	-16	A34
7102	20	F76
7301	32	H76

Figure 4-15. Standard Report Using ENFORM SPACE Clause

In this report, the spacing between the first column and the ITEMS IN STOCK columns and WAREHOUSE LOCATION columns has been increased to 5 and 10 spaces respectively.

ENFORM provides several additional report formatting clauses such as SKIP, TAB, and NOHEAD which enable you to control the report's appearance in much more detail than discussed here. For a detailed description of these clauses, see the *ENFORM Reference Manual.*

SPECIFYING REPORT ARITHMETIC

ENFORM clauses can be used to specify the following types of arithmetic operations:

- calculate and print totals and subtotals for the numeric data items in a column on the report
- calculate a derived data item that is the result of performing arithmetic operations on existing fields in the data base and print the derived items on the report
- calculate aggregate values for groups of data items and print the aggregate values on the report; in this context "aggregate" is used as a noun that means "a type of data item formed from a collection of other data items." ENFORM supports a number of predefined aggregates such as "average", "count", "minimum", and "maximum."

Each of the preceding operations will be illustrated in the following pages by using examples based on connecting the files named parts and fromsup. The logical description produced as a result of the LINK has the following fields:

|partnum|partname|inventory|location|price|.|partnum|suppnum|partcost|

Entering the following statements generates the report in Figure 4-16. This report contains no items that are the result of arithmetic operations. However it is a starting point for describing how to specify report arithmetic.

```
OPEN parts,fromsup;
LINK parts to fromsup VIA partnum;
LIST BY suppnum,
     parts.partnum,
     inventory,
     price;
```

SUPPNUM	PARTNUM	INVENTORY	PRICE
1	212	7	92000.00
	244	3	87000.00
	6201	-16	5800.00
	7301	32	2400.00
6	4102	9	14500.00
	4103	7	24500.00
10	7102	20	6800.00

Figure 4-16. Standard Report With No Arithmetic Operations

Using the TOTAL and SUBTOTAL Clauses

Use the TOTAL clause to specify that the sum of *all* items in a particular column or columns of the report should be printed after the last item in the column. You use TOTAL by specifying it after the name of the data item in the LIST statement. For example, entering:

```
OPEN parts,fromsup;
LINK parts to fromsup VIA partnum;
LIST BY suppnum,
     parts.partnum,
     inventory  TOTAL,
     price;
```

will generate the report in Figure 4-17.

SUPPNUM	PARTNUM	INVENTORY	PRICE
1	212	7	92000.00
	244	3	87000.00
	6201	-16	5800.00
	7301	32	2400.00
6	4102	9	14500.00
	4103	7	24500.00
10	7102	20	6800.00
		62	

Figure 4-17. Standard Report With One Item Totaled

In this report, the items from the *inventory* field are totaled and the total is printed after the last item in the ITEMS IN STOCK column.

SUBTOTAL is similar to TOTAL with the difference being that it prints a sum for a group of items. For example, in Figure 4-17, the report is grouped by the supplier number. And, you can specify that the total number of inventory items for each supplier should be printed by entering:

```
OPEN parts,fromsup;
LINK parts to fromsup VIA partnum;
LIST BY suppnum,
     parts.partnum,
     inventory SUBTOTAL OVER suppnum TOTAL,
     price;
```

to generate the report in Figure 4-18.

SUPPNUM	PARTNUM	INVENTORY	PRICE
1	212	7	92000.00
	244	3	87000.00
	6201	-16	5800.00
	7301	32	2400.00
*		26	
6	4102	9	14500.00
	4103	7	24500.00
*		16	
10	7102	20	6800.00
*		20	
		62	

Figure 4-18. Standard Report With One Item Totaled and Subtotaled

In this report, the sum of the *inventory* fields is printed for each unique supplier number and for all suppliers.

Using Arithmetic Expression Clauses

Arithmetic expression clauses are a powerful ENFORM feature that allow you to add, subtract, multiply, or divide combinations of numeric data fields, literals, and variables—producing derived values for these combinations that can be printed on a report.

The general form of an arithmetic expression clause is:

```
(item followed by a +, -, *, or / followed by item)
```

An example of using an arithmetic expression clause is to modify the report in Figure 4-18 to contain a new column of data items that is the result of multiplying the *price* and *inventory* fields. The heading for the new column will be TOTAL COST and it can be subtotaled and totaled by entering:

```
OPEN parts,fromsup;
LINK parts to fromsup VIA partnum;
LIST BY suppnum,
     parts.partnum,
     inventory SUBTOTAL OVER suppnum TOTAL,
     price,
     (price*inventory) HEADING "TOTAL COST"                    new item
                       SUBTOTAL OVER suppnum TOTAL;
```

to generate the report in Figure 4-19.

SUPPNUM	PARTNUM	INVENTORY	PRICE	TOTAL COST
1	212	7	92000.00	644000.00
	244	3	87000.00	261000.00
	6201	-16	5800.00	-92800.00
	7301	32	2400.00	76800.00
*		26		889000.00
6	4102	9	14500.00	130500.00
	4103	7	24500.00	171500.00
*		16		302000.00
10	7102	20	6800.00	
*		32		136800.00
		62		1327000.00

Figure 4-19. Standard Report With Derived Fields

Using Aggregate Clauses

Aggregate clauses can be used to specify that a single type of value should be computed and printed for a group of values. ENFORM provides five predefined aggregate clauses:

1. AVG ..... the average value of a group of items
2. COUNT ... the total number of occurrences of an item
3. MAX ..... the maximum value of an item
4. MIN ..... the minimum value of an item
5. SUM ..... the total sum of an item.

An example of using an aggregate clause is to generate the average cost of all items in stock for a particular supplier on the report in Figure 4-19.

Entering:

```
OPEN parts,fromsup;
LINK parts to fromsup VIA partnum;
LIST BY suppnum,
     parts.partnum,
     inventory SUBTOTAL OVER suppnum TOTAL,
     price,
     (price*inventory) HEADING "TOTAL COST"
                       SUBTOTAL OVER suppnum TOTAL,
     AVG (price*inventory over suppnum);
```

will generate the report in Figure 4-20.

SUPPNUM	PARTNUM	INVENTORY	PRICE	TOTAL COST	AVG
1	212	7	92000.00	644000.00	222250.00
	244	3	87000.00	261000.00	
	6201	-16	5800.00	-92800.00	
	7301	32	2400.00	76800.00	
*		26		889000.00	
6	4102	9	14500.00	130500.00	151000.00
	4103	7	24500.00	171500.00	
*		16		302000.00	
10	7102	20	6800.00		136800.00
*		32		136800.00	
		62		1327000.00	

Figure 4-20. Standard Report With Aggregated Value

In this report, the average value of the items in the TOTAL COST column is computed by taking the total value of items in the column and dividing it by the number of items in the column.

In addition to the predefined aggregates, ENFORM supports user-defined aggregates that extend your ability to specify report arithmetic. User-defined aggregates can contain arithmetic and logical expression clauses. See the explanation of the DECLARE statement in the *ENFORM Reference Manual* for an explanation of how to specify this type of aggregate.

INDEX

READER'S COMMENTS

Tandem welcomes your feedback on the quality and usefulness of its publications. Please indicate a specific *section* and *page* number when commenting on any manual. Does this manual have the desired completeness and flow of organization? Are the examples clear and useful? Is it easily understood? Does it have obvious errors? Are helpful additions needed?

Title of manual(s): _____

FOLD ➤

FOLD ➤

FROM:

Name _____

Company _____

Address _____

City/State _____ Zip _____

A written response is requested yes no

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 482 CUPERTINO, CA, U.S.A.

POSTAGE WILL BE PAID BY ADDRESSEE

TANDEM
COMPUTERS

19333 Vallco Parkway
Cupertino, CA U.S.A. 95014
Attn: Technical Communications—Software

FOLD

STAPLE HERE

82313 A00

TANDEM COMPUTERS INCORPORATED
19333 Vallco Parkway
Cupertino, CA 95014