Prescriptive methodology for training now-technical users in the use of query languages.

Friedel, Nancy E.
1982
PRESCRIPTIVE METHODOLOGY

FOR

TRAINING NON-TECHNICAL USERS

IN THE USE OF QUERY LANGUAGES

by

Nancy E. Friedel

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Industrial Engineering

Lehigh University

1987
This thesis is accepted and approved in partial fulfillment for the degree of Master of Science.

9/27/82
(date)

Professor in Charge

Chairman of Department
ACKNOWLEDGMENTS

This thesis would not have materialized were it not for the encouragement, direction, and guidance provided by Dr. Ben Wechsler. I sincerely want to thank him for the time and effort he devoted to acting as my major advisor even after his retirement from Lehigh.

I would also like to thank my minor advisor, Dr. Edwin Kay, for his constructive comments and editorial expertise.

In addition, I want to thank Carol Rauch, Kathy Horwath, Monica Morganello, Tim Foley, and Mati Vaze for participation in the training sessions conducted during this investigation.

Finally, I would like to express my appreciation to my husband, Don, and my children, Michael and Heather, for being patient and supportive throughout this endeavor.
## Table of Contents

**ABSTRACT**  
1  

1. Introduction  
1.1 Background  
1.2 Statement of the Problem  
1.3 Objective  
3  

2. Generic Characteristics of Query Languages  
2.1 Form  
2.2 User Interface  
2.3 Syntax  
2.4 Procedurality  
2.5 Extendibility  
2.6 Data Model  
2.7 Means-of-Access  
2.8 Capabilities  
2.9 Summary  
12  

3. Psychology of the User  
3.1 Fear of the Computer  
3.2 Lack of Motivation to Learn a Query Language  
3.3 Propensity for Making Errors  
3.4 Tendency to Forget Details  
3.5 Limited Time for Training  
3.6 Lack of Exposure to the Computer  
3.7 Poor Typing Ability  
3.8 Diverse Backgrounds  
3.9 Lack of Knowledge of Data Base Details  
3.10 Desirable Characteristics of Trainers  
3.11 Summary  
40  

4. Prescriptive Training Methodology  
4.1 Assess Potential Trainee Population  
4.2 Become Familiar with Available Data Bases  
4.3 Identify Products for Training  
4.4 Select and Educate Trainers  
4.5 Develop Basic Method for Presenting Each Product  
4.6 Compile End-Users Manual, Where Applicable  
4.7 Identify Target Group to be Trained  
4.8 Determine Class Participants  
4.9 Schedule Class at Trainees' Convenience  
4.10 Plan Training Exercises around Trainees'  
62
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>V. IQL Training Program</td>
<td>169</td>
</tr>
<tr>
<td>VI. Evaluation Forms</td>
<td>221</td>
</tr>
<tr>
<td>VII. Profile of Training Program Participants</td>
<td>225</td>
</tr>
<tr>
<td>VIII. IQL End-Users Guide</td>
<td>228</td>
</tr>
<tr>
<td>IX. Training Exercises</td>
<td>292</td>
</tr>
<tr>
<td>IX.1 TOPS-20 Training Exercises</td>
<td>293</td>
</tr>
<tr>
<td>IX.2 IQL Training Exercises - Day 1</td>
<td>307</td>
</tr>
<tr>
<td>IX.3 IQL Training Exercises - Day 2</td>
<td>333</td>
</tr>
<tr>
<td>VITA</td>
<td>346</td>
</tr>
</tbody>
</table>
**List of Figures**

| Figure 2-1: | Hierarchical Data Model | 32 |
| Figure 2-2: | Network Data Model | 32 |
| Figure 2-3: | Relational Data Model | 33 |
| Figure 2-4: | Generic Characteristics of Query Languages | 39 |
| Figure 3-1: | IQL Template | 49 |
| Figure 3-2: | Meaningful Learning Process | 54 |
| Figure 4-1: | Work Flow Diagram | 63 |
| Figure 4-2: | Prescriptive Methodology | 86 |
| Figure 5-1: | Tabulated Evaluation Form | 88 |
ABSTRACT

Keeping abreast of user demand is a significant problem in the data processing industry. An attempt at alleviating some of this problem has resulted in the introduction of high level query languages which provide non-technically trained users with a utility for developing their own ad-hoc reports.

The ultimate goal of a query language is to allow the user to communicate with the computer based data via the user's natural language. Since this goal has not been reached, a problem arises when attempting to train a non-technical person to use one of these languages. The objective of this thesis was therefore to develop a prescriptive methodology to train non-technical users in the use of query languages.

Prior to developing the methodology, an analysis was made of query languages themselves. Eight generic characteristics were proposed as a basis for this analysis: form, user interface, syntax, procedurality, extendability, data model, means-of-access, and capabilities.

In order to train non-technical users to use a query language, one must understand what distinguishes them from other users. The major traits considered were: fear of the computer, lack of motivation to learn a query
language, propensity for making errors, tendency to forget details, limited time for training, lack of exposure to the computer, poor typing ability, diverse backgrounds and lack of knowledge of data base principles.

The studies of the languages and the human factor elements were then incorporated into a fourteen step prescriptive methodology.

1. Assess Potential Trainee Population
2. Become Familiar with Available Data Bases
3. Identify Products for Training
4. Select and Educate Trainers
5. Develop Basic Method for Presenting Each Product
7. Identify Target Group to be Trained
8. Determine Class Participants
9. Schedule Class at Trainees’ Convenience
10. Plan Training Exercises around Trainees’ Backgrounds
11. Conduct Training Program
12. Evaluate Training Program
13. Provide Continued Support for Trainees
14. Determine Whether All Groups Have Been Trained

The prescribed methodology provided a sound preparation for an implementation involving training non-technical users to use IQL (Interactive Query Language). This implementation served as a basis for the evaluation and a step-by-step substantiation of the methodology.
CHAPTER 1
INTRODUCTION

1.1 Background

In the traditional organization, the data processing (DP) department has been responsible for the specification, design, and implementation of systems developed to use the computer. The involvement of the end-user in this process has been gradually increasing over the years. Organizations have learned that unless the end-user is involved in the process, it is unlikely that the completed system will satisfy the demands of the user. The process of specification, design, programming, testing, and implementation consumes time. The demand for new applications is rising faster than DP can satisfy them [Martin 82] which has caused an application backlog in most companies. Keeping abreast of user demand is a significant problem in the data processing industry. One study [Mollen 81] done by TBM indicates that the total number of applications waiting for development exceeds those already installed by 80%. There also exists an invisible backlog of applications which the end-users would like to have implemented but have not mentioned to the DP staff because they know it would be years before these applications could be implemented (if at all).
Another reason for this ever increasing backlog is the continuing drop in the cost of computers to the point where the computer is less expensive than the manpower needed to reap its benefits. Different means have been undertaken to increase productivity in DP environments, such as structured programming and top-down design but these methods are not sufficient.

An attempt at alleviating some of this problem has resulted in the introduction of high level query languages. Query languages are special purpose languages for construction of ad-hoc queries and/or simple user applications from a database stored in the computer [Reisner 81]. They provide non-technically trained users with a utility for developing their own ad-hoc reports and the better utilities frequently increase the productivity of the DP personnel. Martin [82] states that about "70% of the immediate needs of typical end users can be satisfied with query languages and report generators if an appropriate data base exists".

One would think that the DP industry would welcome these products as an aid in diminishing the backlog that exists. Many DP managers, however, are unwilling to relinquish power to the end-user. It seems that the very faction of the company that has automated other people's jobs refuses to have their own job automated [Martin 82].
The ideal situation would be one in which the DP
department directs, organizes, and trains end-users in
the use of these facilities. With or without the DP
department's blessing, the end-users will take the matter
into their own hands. They have already begun to use
time-sharing services or to purchase minicomputers to
satisfy their data processing needs rather than waiting
years for the DP department to respond. This could
result in a proliferation of incompatible hardware and
software within a company. In one actual manufacturing
case study cited in Martin's book "Application
Development Without Programmers," a manufacturing
department, after being told by the DP department that it
would take a few years to design and implement on-line
production control applications, took the matter into
their own hands, rented a UNIVAC computer and designed a
system themselves using a software system call MAPPER.

The DP department's reaction was "We're going to
give them enough rope to hang themselves. The machine
will be out of the door in 30 days!" After three
successful years, without DP help, this computer
installation had done more than most DP installations to
raise factory productivity, improve customer service, and
generally increase the corporate profitability. At this
point, DP wanted to take it over so that they could also
create applications faster. "The great strength that the end-users brought to the installation was their knowledge of their job and ability to adjust the procedures a small step at a time as the needs were perceived [Martin 82]."

The increased productivity in DP departments that have initiated "Information Centers" provides an example of the benefits that DP departments can derive via end-users developing their own reports. An information center is usually a separate entity within the DP department which provides end-users with the software to develop their own applications and the support to assist them in the development process. In an actual case, IBM Canada, LTD., created an information center for its employees in 1974. In five years, the data processing department maintenance load dropped from 70% to 40% because about 50% of the project requests were being implemented by the end-user [Mollen 81].

1.2 Statement of the Problem

The ultimate goal of a query language is to allow the user to communicate with the computer based data via the user's natural language. This goal has, however, not been reached. A problem therefore arises when attempting to train a non-technical person in the use of such languages. A recent issue of "ACM Computing Surveys" was
devoted entirely to the psychology of human computer interaction. Thomas Moran in his introduction to the issue noted "the debut of user psychology as a coherent subfield of computer science [Moran 81]."

The success of query languages lies (1) in getting users to use them to alleviate some of the pressure on the DP department, and (2) in providing users with faster responses. Unfortunately, users frequently resist using this type of tool.

Overcoming resistance to the use of these tools is a major problem to be addressed in this thesis. Users without previous exposure to the computer are often awed and intimidated by it. Fear is a major factor blocking conceptual learning. Dunn [68] found that high anxiety impaired concept attainment. Developing a training program which would introduce users to the computer in such a manner as to bolster their confidence and overcome their apprehensions would greatly increase the chances of their continued use of the computer.

In an interview for Computerworld, V. McConnel, who serves as President of a California based consulting firm, stated that "Effective training of non-technical end-users has become indispensable to high information
systems productivity." She cites training as "one of the computing field's most woefully neglected expense items". Of an approximately $500 billion spent annually for Information Processing in this country, only .5% is applied to training programmers and analysts while an amount so small that statistics are not even kept on it is used for training non-technical end-users (Beeler 81).

1.3 Objective

The objective of this thesis will therefore be to develop a prescriptive methodology for training users in the use of a query language.

In Chapter 2, the comparative studies of some of the better known query languages will be synthesized to arrive at a set of generic characteristics that differentiate query languages.

Chapter 3 will involve a study of the psychology of the user, researching what has been done in the field of cognitive psychology to deal with the human-computer interface. From the psychologist's point of view, the goal is to be able to "reliably" assure user-computer interaction just as we can reliably assure other performance aspects of computer systems (Moran 81). An analysis will be made of the characteristics of the naive user, and studies of cognitive processes involved in the
learning of a computer language will be synthesized. These studies will then be adapted to training a non-technical person in the use of a query language.

In Chapter 4, the findings of chapters two and three will be analyzed and integrated into a step-by-step methodology for developing and implementing a training program to train non-technical people to use a query language. This methodology will be adaptable to many situations depending upon the mesh of the generic characteristics of the query facilities and the characteristics of the users.

Since the objective of this thesis is to develop a set of guidelines for a training program for non-technical users of a query language, the main thrust will be to pinpoint problems so that a training methodology can compensate for them. It is beyond the scope of this thesis to dissect the developed methodology, vary all the elements, and compare the effect on different sample groups. Most of the documented studies of this nature have been done in controlled experiments using pencil-and-paper tests in which the respondents were tested on their ability to write, interpret, debug, and modify queries within a given time period. This framework, although conducive to experimental testing, lacks a major ingredient —
interaction with the computer. When the on-line aspect is introduced into the training procedure, some of the control needed to perform valid experimental tests is lost, however, insight is gained into the areas which have caused the most difficulty, and valuable feedback for improvement to the process can be provided.

Initial training of a non-technical person to use a query language requires a much more "individualized" approach than would be necessary in subsequent advanced training sessions. However, the success of this initial training will, to a great extent, determine whether the naive user will be willing to continue in the more advanced classes.

The prescriptive methodology developed in Chapter 4 will be implemented by determining those factors relevant to training naive users in the usage of IQL (Interactive Query Language), and by developing a relevant training program. This training program will be presented to the end-users of an inventory application, developed for the Lehigh University Computing Center (LUCC), which uses IQL as a basis for much of its reporting. Training the users of this system in IQL would greatly enhance the usefulness and flexibility of this system. This implementation could thus provide a test of the methodology which uses the LUCC application as the basis for explaining the features of IQL.
Training will be divided into an instructional period and a period of on-line use utilizing the "Photo" capability of the DEC Operating system to record the terminal sessions. An evaluation method for analyzing these "Photo" outputs is included as Appendix I.

By analyzing the recorded results of the query sessions and noting the types of questions asked during the hands-on session, problem areas in the training method or the language itself may be uncovered. Because of the rapport already developed during the original implementation of this system, it is felt that the users will be candid in their evaluation of strengths, weaknesses, and unevaluated areas of the methodology.

A final evaluation of the methodology can then be undertaken, incorporating any findings revealed during the training process. These findings along with recommendations for further research will be discussed in Chapter 5.
CHAPTER 2

GENERIC CHARACTERISTICS OF QUERY LANGUAGES

Prior to developing a prescriptive methodology for training non-technical persons in the use of query languages, one must first analyze query languages themselves. Eight generic characteristics are being proposed as a basis for this analysis; Form, User Interface, Syntax, Procedurality, Extendibility, Data Model, Means-of-Access, and Capabilities. Each of these will be considered individually in the following sections.

2.1 Form

2.1.1 Definition

Form refers to the type of system within which the query capacity is found. It could be within a Data Management System, a module within a Data Base Management System, or the integral language of a system combining the features of both a data management system and a data base management system.

2.1.2 DMS

A Data Management System (DMS) is defined as a stand-alone package intended primarily to permit access to and retrieval from already existing files [Datapro
81). DMS capabilities include query and response, update, and report generation, but not all packages are capable of all three. Some provide query capability but do not have a general purpose report writer. Others, like ON-LINE ENGLISH, work from read-only files, thereby having no update capability.

2.1.3 DBMS

A Data Base Management System (DBMS) is a software system intended to manage and maintain data in a prescribed structure for the purpose of being processed by many applications [Datapro 81]. Most DBMS's accommodate modules that can be interfaced to the system to handle the related inquiry/response or report generation needs. These various query packages can be procured from outside vendors as long as they are compatible with the user's DBMS. Numerous query packages can be interfaced with the more popular DBMS systems such as IMS and TOTAL.

There is a growing tendency among DBMS vendors to provide a "total system" concept. The vendors have found it more beneficial to themselves as well as to the customer to have a single supplier for all the elements of a DBMS. These vendors have either developed, or acquired from other vendors, compatible packages so as to provide a complete package from which the user can choose the needed modules.
2.1.4 DMS/DBMS

Certain products on the market combine the features of both a DMS and a DBMS. In some instances the product started out as a DMS and later developed into a full-fledged DBMS. Fourth generation languages such as FOCUS and NOMAD fall into this category. They provide a query capability as the integral non-procedural language of the DBMS. They can operate on an existing file structure or, after utilizing a powerful load command, convert the existing files into a prescribed structure. FOCUS, for example, can operate on a CODASYL (Conference on Data Systems Language) type data base by creating a redundant relational file structure. If one chooses to maintain both structures, redundancy does exist, but there are tradeoffs between time and space in such a decision. Martin [82] suggests that it is often good design policy to place the users' information systems on a peripheral computer, thus ensuring that the integrity or performance of the remote data base system will not be endangered. Limitations exist with respect to adaptation to a particular hardware-software system and whether conversion of existing files would be necessary to glean the full benefits of the proposed system.
2.1.5 Form Summary

The purpose for which the package is intended will determine which form the query language should take. If the user wants to have a package which would permit non-technical people to access information stored in a data base structure, a query language interface for the particular DBMS would probably be most applicable.

If, on the other hand, individual departments wish to keep files of information not contained in the corporate data bases, the stand alone DMS's would permit them to create, update, and report from files using one of the data management systems.

Should the user want to load, maintain, and access a data base structure, a system which incorporates the features of both a DMS and a DRMS should be considered.

2.1.6 Impact on Training

The form in which the query capability exists will indirectly affect the training program. The form may be transparent to the user. It is more a matter of how the query capability, in whichever form it exists, fulfills the other characteristics to be analyzed that will have a direct effect on the training program. One such characteristic is the User Interface.
2.2 User Interface

2.2.1 Definition

The "user interface" consists of all the messages that pass between a user and the system and the conditions under which they can occur [Schofield 80]. This interface refers to the messages produced by the query language as well as those produced by the command language.

2.2.2 Command Language Interface

One cannot consider the user's interface with the query language without first considering the nature of the command language interface. The command language interface is the vehicle through which users gain access to the system and to the environment in which they can use the query capability. Many package users are struggling to learn the command language at the same time they are learning to use the query language. Mann [75] suggests that "the major reason non-technical people have difficulties learning to use computer systems is the nature of the command language interface."

Since the user interacts with the query language through the command language, training in the command language must be considered prior to or in conjunction with training in the query language itself.
2.2.3 Query Language Interface

There may or may not be a consistency between the syntax and dialogue of the query language and the command language. The degree of inconsistency will affect not only the training program but also the chances the user will accept the package.

Having the same vendor produce both does not guarantee consistency. A subtle example of such an inconsistency lies with IQL (Interactive Query Language) and the DEC Tops-20 Editor. IQL can be used to access data via the DEC Tops-20 operating system. If a user wishes to create a query, he must become familiar with the DEC editor. To create a new file using the DEC editor two options exist; either the command CREATE filename is given, or the command EDIT filename can be issued which will cause the system to respond File Not Found, Creating New File. In either case, one is then able to enter the file information. To modify an existing file, the command EDIT filename is used.

In the IQL environment, one may modify an existing query by saying EDIT queriname. To create a new query, however, the command WRITE queriname must be given. CREATE has a completely different meaning in IQL and typing EDIT queriname will cause an error message. The system will not default to creating a new query if the
one you are editing does not exist. This inconsistency can be confusing to an experienced user let alone a user just being exposed to the system.

2.2.4 Impact on Training

In addition to the amount of consistency between the query language and the command language, certain other aspects should be examined in the user interface, and the degree to which they are present will have a direct bearing on the training program. They are:

- Meaningful, non-antagonistic messages with an option for abbreviated or detailed explanations
- Existence of on-line HELP facilities instead of user relying on printed documentation
- Clarifying dialogue whereby system communicates with user to overcome ambiguities
- Consistent conventions throughout the system
- Tolerance of minor errors

The training program must fill in the gap for any of the above features which are missing from the user interface. If the user interface is "user-friendly", the process of training will be much easier.

2.3 Syntax

In line with the user interface is another characteristic which can have a direct impact on the
training program. This characteristic concerns the syntax of the query language.

2.3.1 Definition

Syntax can be defined as the form in which users must phrase their queries to the computer. This form ranges from the use of natural English language to a structured English-like language or an almost wordless two-dimensional format.

2.3.2 Natural English Language

The goal of the natural English language approach is to allow the users to communicate with the computer in their own language. By using clarifying dialogue with the user, the system then attempts to resolve any ambiguities [Thomas 75].

A query language that approaches a natural English dialogue is ON-LINE-ENGLISH. Originally called ROBOT [Harris 77], it was subsequently named INTELLECT, IQS, and finally ON-LINE-ENGLISH by Cullinane.

A sample dialogue is depicted in Martin [82]:

USER: WHAT ARE THE NAMES OF ACCOUNTANTS IN NEW YORK

TERMINAL: PRINT NAMES OF ALL EMPLOYEES WITH CITY = NEW YORK AND JOB = ACCOUNTANT

The system thus restates the query as it has been
interpreted. It then prints the number of records that must be retrieved to satisfy the request and prints the answer.

TERMINAL: THE NUMBER OF RECORDS TO RETRIEVE IS 15
Jones, H.A.
Kendall, C.R.
.
.
.

The user may ask another question relating to the same query.

USER: WHICH ONES EARN MORE THAN 20,000

TERMINAL: PRINT NAMES OF ALL EMPLOYEES WITH
CITY = NEW YORK
AND JOB = ACCOUNTANT
AND SALARY > 20,000

There are problems with the natural English approach in addition to the inherent difficulties in producing such languages. The user may develop an undeserved respect for the "intelligence" of the system and begin to personify it. Naive users can become frustrated by the restrictions on syntax and vocabulary which are likely to be present. They are less likely to get frustrated if they are aware of these limitations rather than being told the system can understand English [Plum 77].
2.3.3 English-like Language

Instead of natural English, most query languages require users to state their questions in a formal language using an English-like syntax and grammar. To demonstrate variability of the English-like syntax, one simple query will be formulated using several different query languages [Reisner 81].

One table or relation will be involved: an Employee table (EMP) composed of NAME, DEPTNO, and SALARY. The sample query is: **FIND THE NAMES OF EMPLOYEES IN DEPT 50.**

**SQL:**
SELECT NAME 
FROM EMP 
WHERE DEPTNO = 50

**TABLET:**
FORM DEPTFIFTY FROM NAME, DEPTNO OF EMP 
KEEP ROWS WHERE DEPTNO = 50 
PRINT NAME

**IQL:**
OPEN EMP 
IF DEPTNO = "50" 
PRINT NAME

**NOMAD:**
DA EMP; 
LIST NAME WHERE DEPTNO = "50"

**QUEL:**
RANGE OF E IS EMP 
RETRIEVE (E.NAME) WHERE E.DEPTNO = "50"

**FOCUS:**
TABLE FILE EMP 
PRINT NAME; 
IF DEPTNO FO '50'

2.3.4 Two-Dimensional Form

An exception to the English-like query approach exists with the two-dimensional QUERY-BY-EXAMPLE. The
user requests a particular table and a skeleton of the table containing the name of the table and column headings is displayed:

```
EMP | NAME | SALARY | DEPT NO
-----|------|--------|----------
| | | |
| | | |
```

The user then enters an example of the information he would like to see:

```
EMP | NAME | SALARY | DEPT NO
-----|------|--------|----------
| | | |
| | | |
| p. Jones | | 50 |
```

The fact that "Jones" is underlined indicates that it is a sample and that all names satisfying any other prescribed conditions are to be printed (signified by "p."). This concept is analogous to algebra where one would instead place x under Name. Reference to this concept is purposely avoided in training because of the naive user's tendency to regard algebra with apprehension and foreboding. The "50", since it is not underlined, represents an actual value that is desired.

The resultant output would be a listing of the names of all employees in department 50.
2.3.5 Impact on Training

The syntax of the query package will directly affect the training procedure. Two-dimensional languages, such as QUERY-BY-EXAMPLE, are practically wordless and do not force users to generate their queries freestyle. This has caused them to be exceptionally easy languages for the naive user to learn [Chamberlain 81]. They will, however, require a unique training methodology.

Apart from the two-dimensional syntactic languages, the more English-like the language is, the less training that will be required.

2.4 Procedurality

2.4.1 Definition

Tied in to the English-like nature of the various query languages is the degree of procedurality the language displays. A language is termed non-procedural if the user is required to tell "what" is to be achieved but not "how" it is to be achieved. A language is termed procedural if it specifies a step by step method for achieving a result. Procedurality is a difficult characteristic to categorize because it is actually a continuum which cannot be described in absolute terms. Welty and Stemple [81] compared SQL and TABLET which differ mainly in degree of procedurality. SQL was felt
to be non-procedural while TABLET was claimed by its
designers to be procedural. As depicted earlier, the
TABLET query to "Find the names of employees in
department 50" would be of the following form:

```
FORM DEPTFIFTY FROM NAME, DEPTNO OF EMP
KEEP ROWS WHERE DEPTNO = 50
PRINT NAME
```

This format specifies what is to be done and in what
order and is therefore termed procedural. It is far less
procedural, however, than a COROL program written to do
the same thing. The user does not have to open files,
read individual records, or handle error conditions. For
the purposes of this thesis, "procedural" will refer to a
query language and not to programming languages in
general.

2.4.2 Non-procedural

Non-procedural query packages are becoming more and
more popular for use by the non-technical person. They
do not require a knowledge of the structure of the data
base nor do they require the user to handle files a
record at a time. Instead, the query language allows the
user to operate on sets of data.
2.4.3 Procedural:

Studies done by Welty and Stemple [80] comparing SQL and TABLET supported the hypothesis that "people more often write difficult queries correctly using a procedural query language than they do using a non-procedural query language."

Some query languages, such as NOMAD, include a mixture of non-procedural and procedural capabilities.

2.4.4 Procedurality Summary

A procedural query language, as described here, is actually non-procedural when compared to a procedural programming language like COROL. Very few experiments have documented the effect that procedurality of a query language has on ease of use by non-technical users. Even though Welty and Stemple [81] did show that for complex queries a procedural query language was easier to retain and understand, the majority of queries formulated by the non-technical user are found to be non-complex [Cuff 80].

2.4.5 Impact on Training

Up to a point, procedurality will have a definite impact on the training method. If, because of the degree of procedurality, the user must be taught the underlying structure of the data or how to handle error conditions, a more intensive training procedure will be required.
Once the procedurality has reached the point where the user can operate on sets of data rather than a record at a time, any further movements toward non-procedurality have a less significant effect on training.

2.5 Extendibility

A characteristic of query languages which must also be considered is their extendibility.

2.5.1 Definition

Extendibility refers to the range of activities that can be accomplished using a particular package or a compatible package without requiring the user to learn a new syntax and dialogue.

Some query languages perform the function of data base inquiry alone. To update the data base or perform general report writing tasks, another product must be used which may or may not resemble the query language that has been learned.

The complete range of activities being considered as falling under the extendibility characteristic is that a query language have the ability to perform the following functions:

- Pure Query Function
- Report Generator
- Graphics Generator
2.5.2 Pure Query Function

This is usually the initial involvement that the user has with the computer. The pure query function of a query language serves as a means of retrieving information from the database. Some query languages perform this function alone.

2.5.3 Report Generator

A generalized report generator is often included in the query facility. It provides a means for designing tailor made reports.

2.5.4 Graphics Generator

Many graphics packages exist on the market, some of which are included as part of a report generator. Their function is to provide pictorial representations of the data.

2.5.5 Data Base Update

Data base update packages will permit users to interactively update their data base.
2.5.6 Application Generator

Application generators are vehicles by which users can design their own applications.

2.5.7 High Level Programming Language

A high level programming language provides users with the facility for procedurally programming their own applications.

2.5.8 Extendibility Summary

Some software systems offer individual packages for each of the described functions: pure query function, report generator, graphics generator, data base update, application generator, and high level programming language. In many instances, there is little similarity among the packages even though they might be supplied by the same vendor.

Some packages, such as NOMAD and FOCUS, allow for the complete range of functions within the same product. The user can, therefore, progress within the same framework and learn the system in stages. This type of system is much more extendable.

Few users will actually progress to the level where they are programming in a high level programming language, but many will progress beyond the data base inquiry stage if the facilities exist and are compatible with the language they have already learned.
2.5.9 Impact on Training

The training program as developed will be geared towards the query level but much of the methodology will be applicable to subsequent training in more difficult areas especially if the packages are similar in syntax and dialogue.

2.6 Data Model

Having considered the extendibility of query languages, the next characteristic to be studied is the data model on which the language is based.

2.6.1 Definition

Data model refers to the user’s conceptual view of how the data is stored, which need not correspond to how it is physically stored. There are three predominant data models: hierarchical, network, and relational.

2.6.2 Hierarchical

In a hierarchical data model, the user perceives the data base as a tree-like structure with an owner-member (or parent-child) relationship. A member (or child) is permitted to have only one owner (or parent). Some hierarchies are limited to one parent-child level whereas others provide more than one but to a limited depth.
2.6.3 Network

In the network model, advocated by CODASYL, record occurrences are represented as nodes of a network, chained together by named, directed arcs. The arcs present logical links between the entities which can be traversed in the specified direction in order to navigate through the data base [Lough 77].

A network is more general than a hierarchy because a member record can have any number of owners, whereas in a hierarchy it can only have one. Using a link record, networks are capable of representing many-to-many relationships [Date 77].

2.6.4 Relational

The relational data model is one in which the user perceives the data in the form of two-dimensional tables which can be related to each other via common data items.

Flat files can also be considered as a relational data model because a relational model implies no predefined relationships among the tables, that is, no pointer linkage exists between tables.

2.6.5 Data Model Summary

Most query facilities include a Data Dictionary which at minimum contains the names and descriptions of all items stored in the data base. The user will need to
have access to the Data Dictionary in order to know the correct spellings for names of items in the data base.

Using the inventory application, which will be used in training non-technical users in IQL as an example, this same application will be shown as a hierarchical (Fig. 2-1), network (Fig. 2-2), and relational (Fig. 2-3) data model.

The inventory system contains information about computer terminals and modems owned and/or purchased through the university. Some of the data items recorded are:

- Vendor-No - code to designate manufacturer
- Vendor-Name - name of manufacturer
- Model-No - code to distinguish a particular model within a manufacturer
- Model-Name - description of model
- Serial-No - manufacturer's serial number for piece of equipment
- Cust-No - code to designate owner of equipment
- Cust-Name - name of owner of equipment
- Cust-Bldg-No - code of building in which customer resides
- Equip-Bldg-No - code of building in which equipment is housed
- Bldg-Name - name of university building.
Figure 2-1: Hierarchical Data Model

Figure 2-2: Network Data Model
Figure 2-3: Relational Data Model

The major way in which the three data models differ is in the manner in which they handle relationships between entities [Lochovsky 79]. The relational model is gaining popularity especially for use by non-technical people. It is flexible and easy for the naive user to relate to because of its simplicity.

The network and hierarchical models require the user to understand the structure of the underlying data model in order to navigate through the data base. For these reasons, most query languages use the relational data model.
2.6.6 Impact on Training

If the data model is relational, it will present fewer problems at training time because the non-technical person is probably already familiar with the table concept. If he must, instead, be taught to navigate through a network, a much more involved training procedure will be required.

2.7 Means-of-Access

Also to be considered in the analysis of query languages is the means of access to the computer.

2.7.1 Definition

Means-of-access is defined as the manner in which the user gets the desired information from the computer. The means-of-access can be either on-line or off-line.

2.7.2 On-line(Interactive)

Most query packages are designed for on-line use. Users interact directly with the computer using an English-like language and get an immediate response to their query needs. The user can formulate a query and then, based on the computer responses, can narrow the search step-by-step to arrive at the desired result.
2.7.3 Off-line (Forms-oriented)

Some query languages are designed for off-line use wherein the user fills in a form and submits it to the DP department. Informatics' MARK IV query language is forms-oriented.

2.7.4 Means-of-Access Summary

On-line, interactive use has many advantages over the off-line forms-oriented approach. There is the obvious savings in time in that the user receives an immediate response a query. The system also has the freedom to seek additional information from the user [Cuff 80] should the need arise, and the user has the option to modify the query to obtain additional information.

The query languages which have had the most success with the end user are those that have been interactive [Martin 82].

2.7.5 Impact on Training

The means-of-access has an obvious impact on the training program. The amount of training required to teach a non-technical person to fill in a form is minimal; however, this forms-oriented approach does not set the groundwork for using the computer for purposes beyond the present query needs.
2.8 Capabilities

The final characteristic to be analyzed involves the capabilities of a query language.

2.8.1 Definition

Capabilities refer to the various operations and functions that can be performed using a designated query package. Numerous capabilities are possible in a query language, but not all query packages provide them.

These capabilities can be summarized [Shneiderman 78] as follows:

- Simple mapping - Retrievability of a data value for one field when a known data value for another field is supplied
- Selection - Ability to select certain records based on a key
- Projection - Retrievability of an entire column or domain
- Boolean queries - Use of logical operators (and/or) to specify conditions of a request
- Set operation - Ability to find union and/or intersection of sets based on common fields
- Combination - Use of output of one query as input to another
- Built-in functions - Availability of functions such as MIN, MAX, SUM
- Grouping - Ability to group or sort items by a common column or domain
2.8.2 Impact on Training

The number of capabilities available and distinguishing which ones should be taught in an initial training period must be determined. Reisner advocates a layered approach to training naive users [Reisner 81]. Not all features are necessary for the user to get meaningful use from a system. To attempt to teach all aspects of the language at the beginning would only tend to confuse new users. They should, instead, be given the opportunity to interact with the system and build their confidence before tackling the more complicated aspects of the language. The single most important factor in building up a user's confidence is the capability to use the system. Users should, later, after they are comfortable with what they have learned, be able to extend their knowledge by returning to an advanced class, seek further training on-line, or have available to them, people who can answer questions and gradually train them to use the full capabilities of the system.

2.9 Summary

These characteristics, (synthesized in Figure 2.4) while not claiming to be complete, provide a framework in which to analyze query languages. There is a wide variation among the numerous products currently on the
market as to the mixture of these characteristics. Martin [82] has developed a table classifying many of these products as to Vendor, Machine-Type, Extendibility, Ease-of-Use, and Means-of-Access and this table is included as Appendix II.

Along with the aforementioned aspects of the language itself which will affect the development of a training procedure, the other critical element is the human factor. This will be studied in the next chapter by investigating the psychology of the user.
Form:
- DMS (2.1.2)
- DBMS (2.1.3)
- DMS/DBMS (2.1.4)

User Interface:
- Command Language (2.2.2)
- Query Language (2.2.3)

Syntax:
- Natural English Language (2.3.2)
- English-Like Language (2.3.3)
- Two-Dimensional Form (2.3.4)

Procedurality:
- Non-Procedural (2.4.2)
- Procedural (2.4.3)

Extendibility:
- Pure Query Function (2.5.2)
- Report Generator (2.5.3)
- Graphics Generator (2.5.4)
- Database Update (2.5.5)
- Application Generator (2.5.6)
- High-Level Programming (2.5.7)

Data Model:
- Hierarchical (2.6.2)
- Network (2.6.3)
- Relational (2.6.4)

Means-of-Access:
- On-line (Interactive) (2.7.2)
- Off-line (Forms-oriented) (2.7.3)

Capabilities:
- Mapping (2.8.1)
- Selection (2.8.1)
- Projection (2.8.1)
- Boolean Operators (2.8.1)
- Set Operation (2.8.1)
- Combination (2.8.1)
- Functions (2.8.1)
- Grouping (2.8.1)

Figure 2-4: Generic Characteristics of Query Languages
CHAPTER 3

PSYCHOLOGY OF THE USER

In order to train casual users to use a query language, one must understand what distinguishes casual users from other users. They can be identified as intermittent users who do not use the computer as their primary job function. Cuff [80] lists major traits or characteristics that they, as a class or as an individual may exhibit:

- Fear of the Computer
- Lack of Motivation to Learn a Query Language
- Propensity for Making Errors
- Tendency to Forget Details
- Limited Time for Training
- Lack of Exposure to the Computer
- Poor Typing Ability
- Diverse Backgrounds
- Lack of Knowledge of Data Base Principles

3.1 Fear of the Computer

Of prime concern, is the fear that many would-be users exhibit towards the computer. Many end-users regard the computer and "creation" of applications as a mystique entirely beyond reach [Martin 82]. One of the reasons that people form strong preconceptions about
computers is that their workings are invisible [Smith 80]. They fear what they do not understand. What is more significant is that they fear they will make mistakes and appear to be incompetent. As a defense against the latter, they avoid becoming involved with computers. This is especially true of managers who are used to being in control of their work environment. They are knowledgeable in their area of expertise and do not want to be put in a situation where others, employees perhaps, will see them in an embarrassing situation.

Weinberg [71] likens initial exposure to a computer system to the feeling of helplessness a person has when visiting a foreign country for the first time. Many programmers and analysts do not anticipate this attitude. They enjoy experimenting with terminals and often fail to comprehend that others are afraid of them. The barrier, that fear causes, cannot be removed by lecturing about the computer, but it can be removed by putting the learner in a situation where the machine gives him the experience he needs to overcome his fear [Weinberg 71].

3.1.1 Impact on Training

Fear of the computer poses a two-fold threat to a training program. In the first place, this fear can keep a user from even considering a training program and secondly, it can serve as a deterrent to meaningful
learning should the user decide, or be forced, to attend a training program.

Advance publicity can be helpful in the former instance, by stressing the fact that no prior computer experience is necessary to learn a query language. The trainees should be reassured that they will be receiving individualized assistance. Since the fear that most end-users experience is not so much of failure itself, but of having other people witness the failure [Weinberg 71], they would find reassurance in knowing that they will not be making mistakes in front of an audience. Techniques to be discussed in the next section to motivate a person to learn a query language would also be appropriate here, because strong motivation can overcome even fear.

In the training program itself, fear can best be overcome either through individualized one-on-one instruction or if this is not feasible by insuring that members of the class are grouped homogeneously. Placing a person who fears computers in a class with people who do not understand this problem would probably assure failure. This user in particular needs a safety net either from within the system via HELP facilities or through one-on-one interaction with the trainer. The safety net provides the user with the opportunity to
experiment with the system knowing that he will not be left in limbo. By hand-holding users through their initial experiences with the computer, much of the fear can be dissipated as the user gains confidence.

3.2 Lack of Motivation to Learn a Query Language

Lack of motivation ranks second only to fear in impact on a training program. Weinberg [71] states that "it could well be that if someone is not motivated there is no way to make them learn and if someone is motivated, there may be no way to stop them from learning." An individual needs to perceive that he will reap a personal benefit from learning a query language. This benefit could be that the facility will provide the user with a capability to do his work better or to have time to do more interesting work because of the time that will be saved by using the query facility.

Top management support and backing can play a significant role in the user’s conception of the value of this training program. If the end-user perceives that knowledge of this skill will better his chances for advancement, he will be more likely to be motivated to acquire this skill. Top management forcing employees to attend a training session will not provide the type of motivation desirable in this situation.
3.2.1 Impact on Training

Various means can and should be undertaken to encourage an end-user to learn to use a query language. There are two aspects to be considered here. Initially, a stimulus must be provided to encourage the end-users to agree to the training program and later, once they are in the training program, the motivation that brought them to the training program must be heightened or at least sustained.

Advance publicity should emphasize that by learning a query language, end-users will be able to generate their own ad-hoc reports instead of waiting for the DP department to design and implement them. This publicity should also stress the control the end-users will gain because they will not have to go through an intermediary who may misinterpret their requests. Increased productivity should be emphasized as a benefit of using this facility. This literature could also contain the candid comments of satisfied end-users who are currently using the query facility.

Development of an effective training program will be an evolutionary process. A good foundation would be laid by having a few initial successes i.e. a few satisfied users. They can serve as an excellent form of publicity. To help assure these initial successes, the trainer
should seek out enthusiasts or as Martin [82] terms them "early adapters" as the first trainees. These individuals should be people who like to experiment and try new things. Once convinced of the benefits of what they have learned, they can serve as valuable salesmen.

Assuming that the end-user has committed himself to the training program, his motivation must be reinforced. This can best be accomplished by structuring the training program so that the trainee is accomplishing something worthwhile in a short period of time. This is possible with almost all query languages by teaching a subset of commands and allowing the user to experiment with them and thereby gain a feeling of accomplishment.

3.3 Propensity for Making Errors

Casual users will make errors when they begin using a query language and inevitably, because of intermittent use, will continue to make them. The propensity for making errors will be heightened if the query language requires the user to remember numerous syntactic details. This trait is also interwoven with the amount of previous exposure the users have had to the computer and the attitude that they have brought to the training program - be it fear or enthusiasm.

Some users, realizing that they will be making
errors, also fear that their errors will damage files or even the computer itself. This makes them hesitate to become involved with the computer at all.

3.3.1 Impact on Training

Casual users should be reassured that they should expect to make errors initially and that the training program has been adjusted to compensate for this problem. This class of users in particular, needs a safety net whereby they are able to seek aid when an error occurs. Most query systems have also anticipated this problem and have intrinsic HELP facilities. In any event, users should be taught very early in the training program what options exist should an error occur. If on-line HELP facilities exist, users should be trained how to use them. If the on-line facilities are weak, trainees should be instructed where to find explanations of error messages via off-line documentation. Should both the system aids and documentation be inadequate, it will be necessary to compile an end-users manual explaining what to do in the event of errors.

The casual users should also be reassured at the onset of the training program that there is little, if any, likelihood that they will damage any files and no chance they will damage the computer itself by using a query facility. They should also be reassured that
should files become damaged, they can usually be reconstructed. Some organizations, anticipating this problem, create a duplicate database for access by end-users. Even though some query languages do permit the user to update the database, the commands that perform this function should not be taught until the user has become comfortable with the subset of commands used strictly for querying a database.

3.4 Tendency to Forget Details

A trait that casual users exhibit, especially if they use the computer intermittently, is that they have a tendency to forget the details of a query language. This is to be expected because they do not receive the reinforcement that constant usage provides. The degree to which this trait is manifested is directly associated with the complexity of the syntax (2.3) of the language being used. Some languages require the user to enter information in a prescribed format and are very inflexible. If a choice exists between a rigidly structured language and a more flexible one, the latter is more appropriate for this class of users.

3.4.1 Impact on Training

Since the casual user cannot be expected to remember all the details of a query language, it is better to
assure that the user acquires a basic understanding of the principles of the language. This can be facilitated by aiding the user in developing a conceptual model of the query language. A user’s conceptual model is how the user thinks the language works. A conceptual model of a query language based on a relational data model (2.6) might be [Reisner 81]:

1. The computer locates the table requested
2. Then it looks for the columns that have the information to be looked up
3. Then it looks in these columns for the information desired
4. Then it pulls out the rows that have this information
5. Then it looks in these rows for the answers desired.

If the user gleans a clear conceptual model of the query language from a training program, he will be more likely to remember how to use the language. He will understand that writing a query involves certain basic functions in order to satisfy the conceptual model. For the query mentioned earlier "Find the names of employees in Dept 50", he must specify:

1. What action is to be taken (Print, List, etc.)
2. Which data type or types are the action to be taken on (Name, Deptno in Employee record)
3. Qualifying characteristics that determine whether this action is to be taken (Dept 50)

The user can then translate the English request into the query language.

One method of aiding the user is this translation process is to generate a template [Resiner 81] of the major components of a query which the user can call up on the screen and modify to fit his present query. This method as applied to IQL would be as follows:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEADING &quot;________&quot;</td>
<td>(What title do you want for this report)</td>
</tr>
<tr>
<td>OPEN &quot;________&quot;</td>
<td>(What tables are required to satisfy this request)</td>
</tr>
<tr>
<td>IF &quot;________&quot;</td>
<td>(What selection criteria, if any)</td>
</tr>
<tr>
<td>SORT &quot;________&quot;</td>
<td>(What sequence)</td>
</tr>
<tr>
<td>PRINT &quot;________&quot;</td>
<td>(What data items do you want to see)</td>
</tr>
</tbody>
</table>

Figure 3-1: IQL Template
To satisfy the request "Find the names of employees in Dept 50" the user would utilize the template as follows:

HEADING "Employees in Dept 50"
OPEN EMP
IF DEPTNO = "50"
SORT BY NAME
PRINT NAME

The casual user may still forget syntactic details but he will understand the basics of what is required to generate a query and can seek help from the system or from off-line documentation when making a syntactic error.

3.5 Limited Time for Training

Generally speaking, casual users will not be willing to devote much time to learning how to use a query language, especially if they foresee using it only sporadically. They may not be able to comprehend that a couple of hours spent in training now could save them a couple of hours a day later on [Martin 82]. This trait is tied in with motivation. If users are strongly motivated and perceive that knowledge of a query language will be of personal value, they will be more likely to try and set aside time for training. Users must be furnished "realistic" estimates of the amount of time
they will need to expend now and what specific advantages
(i.e., increased productivity) that acquisition of this
skill will bring [Nickerson et al].

3.5.1 Impact on Training

Every effort should be made to adapt the training
program to the user-trainee's schedule. In some
instances, it may be necessary to schedule classes
outside the regular working day if that is the only time
a busy executive can spare or is willing to allow his
employees to devote to a training program.

The trainees should be assured that with a small
investment of time, they will be able to carry out useful
work with the query language. The amount of time
required for training will depend on the particular query
language. With many of the query packages available
today, a meaningful subset of the language can be learned
in a few hours of training. The more complex features of
the language should be reserved for later sessions after
the user has become comfortable with the basic features
of the language and wishes to generate more complicated
queries. This subset might include commands like PRINT
or LIST and IF or WHERE or their equivalents. These few
commands have extensive report and query capacities.
Addition of a few report functions such as MIN, MAX, SUM,
or AVERAGE extend the language even further. With this
limited subset, the user should be able to become comfortable with the language and carry out meaningful work.

3.6 Lack of Exposure to the Computer

Most end-users will have had little exposure to the computer. They are also likely to have formed misconceptions about what the computer can and cannot do. One of the difficulties in training novices lies in describing at the right level of detail the machine they are trying to control [Boulay R1].

3.6.1 Impact on Training

The end-user should be made aware through advance publicity, that no previous experience is necessary to learn to use a query language. Some means, such as interviews or questionnaires, should be employed to ascertain how much exposure the trainee has had to the computer and whether those experiences were positive or negative. If the trainees are not going to receive individualized instruction, consideration should be given to grouping them according to their experience.

It also must be determined whether the user needs to be taught the command language before being taught the query language itself. A trainee with no previous experience should receive an introduction to the computer in terminology that eliminates all technical jargon.
3.7 Poor Typing Ability

Another trait which is common among casual users is poor typing ability. Even the most rudimentary typing skills are frequently uncommon in office and particularly professional workers [Cuff 80]. Embarrassment caused by the lack of typing skills serves as an additional deterrent when end-users are considering training to use the computer. This trait, however, is not unique to casual users. Many professional programmers are also deficient in this regard. It affects productivity to some extent, but considering the amount of typing that a casual user will be required to do, the effect will be negligible.

3.7.1 Impact on Training

Advance publicity should emphasize that typing skills are not required lest this be used as an excuse for avoiding training. Perhaps, if casual users were aware that they are not alone in their lack of typing skills, it would alleviate some of the embarrassment.

Consideration should also be given during the training program to ensure (1) that users are given sufficient time to enter queries, and (2) that the features of the terminal are explained at the onset of training. It would also be helpful if the terminals that the users train on are similar to the ones that they will be using later.
3.8 Diverse Backgrounds

End-users involved in learning to use a query language will come from diverse backgrounds representing many facets of an organization. Some may have a scientific background, others may be accountants, financial advisers, etc. Each brings with him his own unique set of experiences which have formed the wealth of knowledge that comprise his background. Cognitive psychology attempts to understand how a person assimilates this knowledge. They have concluded that meaningful learning occurs best if the user connects new material with knowledge that already exists [Bran 79]. Mayer [81] describes this meaningful learning as a three-step process (depicted in Figure 3-2) based on: reception, availability, and activation.

![Diagram](image-url)

Figure 3-2: Meaningful Learning Process

Initially, (a) the learner pays attention to the new material by bringing it into short-term memory (reception). Secondly, (b) he must possess the
appropriate prerequisite concepts in long-term memory to permit assimilating the new information (availability); and finally, (c) these ideas must be transferred to short-term memory so that they can be combined with the new incoming information (activation). "If any of these conditions is not met, meaningful learning cannot occur, and the learner will be forced to memorize each piece of new information by rote as a separate item to be coded to memory" [Mayer 81].

By analyzing the findings of cognitive psychology and applying the premises of how we learn to the study of a query language, it becomes evident that a trainee's background (which will dictate what concepts are in long-term memory) can have an effect on how the learner will relate the newly presented material to pre-computer concepts.

3.8.1 Impact on Training

If the training program can be adapted to explaining concepts in terminology with which the user is already familiar, it is more likely that meaningful learning will occur. This presents no small problem to a training program and is not always feasible. The number of people in the training program will definitely affect the ability to provide the homogeneous grouping which the aforementioned would require. This problem could best be
handled by providing individualized instruction. If individualized instruction is not possible, the class size should be kept as small as possible. During on-line sessions a maximum of two people should be assigned to the same terminal to ensure that each trainee gets an opportunity to interact with the computer.

Users' backgrounds should be considered when pairing individuals for terminal sessions and when designing the on-line exercises used to explain the various commands of the language. Studies (Stevens et al.) have shown that trainees do not like to do preplanned exercises; that they would rather plunge into their own queries. If the training program could be structured around an application with which the users are familiar, the users would find more relevance in performing these exercises. This could be accomplished through an evolving training program. Initially, the training program could be structured around an application with which the first trainees are familiar. The second time the training program is given, it could be adapted to accommodate a new application. One of the advantages of the new software is that it makes it so easy to develop such prototype systems. Eventually, a repertoire of applications would exist from which the trainer could choose the best one to suit the mixture of backgrounds of a particular group of trainees.
3.9 Lack of Knowledge of Data Base Details

Most non-technical users will lack any knowledge of data base principles and ideally they should not be required to learn them. The amount that they will need to know will depend on the particular query language and the data model (2.6) on which it is based.

The relational data model will require the least amount of knowledge of data base principles. Most relational based query languages will require the users to know which data items are in which table and the linking items needed to join the tables. Some will, however, perform this navigation automatically.

The hierarchical data model query languages will require the users to understand the parent-child structure of the data base thus requiring them to know that they must go through the parent to get to the child.

The network data model is the most complex. The users will need to know what set relationships exist between the records. They will need to navigate through these sets checking for error conditions. This will necessitate operation on the data base record by record. In IQL, this is accomplished by five different FIND commands:

- FIND FIRST recordname RECORD OF setname
- FIND NEXT recordname RECORD OF setname
- FIND PRIOR recordname RECORD OF setname

57
- FIND LAST recordname RECORD OF setname
- FIND OWNER RECORD OF setname

Users must, therefore, know what set relationships are predefined so that they can choose access paths for navigating through the data base. Users are also responsible for checking error status codes in order to know if an end of a set has been reached.

3.9.1 Impact on Training

The user's lack of knowledge of data base principles may have little impact on the training program. A goal of many query languages is to shield the user as much as possible from the underlying physical structure of the data base. As stated previously, the amount of training in data base principles required will depend on the language itself and the data model on which it is based. If the data model is a network, the user should be appraised of the fact that he will have to commit to a longer, more-involved training period. Because of this complexity, many query languages are represented to the user as a relational data model even though the data may actually be stored as a network structure.

Most query languages will require that exact names of entities in the data base be used when writing a query. This information is usually available in the form
of a Data Dictionary. Users should be trained how to access the Data Dictionary so that they will know the correct names of entities in the data base.

3.10 Desirable Characteristics of Trainers

Having thus examined the various traits that casual users may exhibit, it is now appropriate to study what traits would be desirable for people chosen to train this class of users.

The trainers should be individuals who can understand the apprehensions some users have of interacting with the computer and who can be sympathetic without being patronizing. They should enjoy working with people and should be adept at communicating with them. The trainers, who are typically DP personnel, are often lacking in communication skills. It would be advisable, if this is the case, to have the trainers attend classes on improving these skills.

Another trait, which trainers should possess, is the patience to assist users in solving their problems while avoiding the inclination to solve the problems for them, which at the time might seem to be the most efficient approach. Trainers need to bear in mind that a goal of the training program is to encourage the users to become increasingly self-sufficient.
Trainers must also realize the importance of providing quick responses to questions the users may have after they finish the training program and begin to use the query language. The users need to know that the trainers have a continuing interest in them and want to help them succeed in applying what they have learned in the training program.

It is obvious that the trainers should be familiar with the products they are teaching, but what is not so obvious and usually neglected, is that they should also gain some background in the functional areas of their students. Hopefully, through the use of questionnaires or some other means, the trainers have become familiar with the background of the students. Using this information, the trainers should take time to gain some knowledge in the appropriate functional areas so that they can design the training exercises properly.

3.11 Summary

The variety of traits and characteristics a casual user may exhibit stresses the need for flexibility in a training program. If it is to be successful, the training methodology must fulfill the needs of the particular user or users being instructed.

Having analyzed the characteristics of query
languages and the human factor element of the
human-computer interface, these studies will now be
incorporated into a prescriptive methodology for training
non-technical people in the use of a query language.
CHAPTER 4

PRESCRIPTIVE TRAINING METHODOLOGY

A general procedure will now be proposed to systematically develop a training course to train non-technical users to use a query product. This methodology should prove applicable to many types of organizations. These include training programs organized by DP departments, autonomous departments, or service organizations. This procedure involves a step-by-step iterative process which is illustrated in the work flow diagram presented in Figure 4-1. An explanation of each of the steps contained within the methodology follows the diagram.
Figure 4-1: Work Flow Diagram
Select & Educate Trainers

Develop Method Presenting Products

End-User Manual Needed

Yes

Compile End-User Manual

No

Identify Target Groups

Determine Class Participants

Schedule Class

Plan Training Exercises

Figure 4-1, continued
Figure 4-1, continued
Figure 4-1, continued
Figure 4-1, continued
Provide Continued Support 4.13

All Groups Trained 4.14

Publicize Training Program 4.14.2

Figure 4-1, concluded
4.1 Assess Potential Trainee Population

Initially, prospective target groups should be identified who might benefit from the proposed training. One means of identifying potential trainees would be to develop and circulate advance publicity in which end-users are appraised of the fact that consideration is being given to developing a training program to suit their individual computer needs. This publicity could take the form of a questionnaire which seeks to determine the amount of exposure the end-users currently have had with the computer i.e. what types of applications are currently computerized or are in the process of being developed and what applications they would like to see developed. This questionnaire should also attempt to determine the backgrounds of potential users and how many people they, at this time, foresee benefitting from this type of training. The publicity should cite the advantages that can be gained by being able to develop ad-hoc queries in light of time saved and increased productivity. This questionnaire should give the organizers of the training program an inflated idea of potential trainees for the training program. The estimate will be inflated because there are many end-users who would benefit from such a training program and initially perceive this to be so, but when the time
4.1 Assess Potential Trainee Population

Initially, prospective target groups should be identified who might benefit from the proposed training. One means of identifying potential trainees would be to develop and circulate advance publicity in which end-users are appraised of the fact that consideration is being given to developing a training program to suit their individual computer needs. This publicity could take the form of a questionnaire which seeks to determine the amount of exposure the end-users currently have had with the computer i.e. what types of applications are currently computerized or are in the process of being developed and what applications they would like to see developed. This questionnaire should also attempt to determine the backgrounds of potential users and how many people they, at this time, foresee benefitting from this type of training. The publicity should cite the advantages that can be gained by being able to develop ad-hoc queries in light of time saved and increased productivity. This questionnaire should give the organizers of the training program an inflated idea of potential trainees for the training program. The estimate will be inflated because there are many end-users who would benefit from such a training program and initially perceive this to be so, but when the time
comes for training are unwilling to spare the time or personnel.

Insight can also be gained, from this questionnaire, as to the types of applications that these end-users will require, and the types of data bases that will be involved. This factor is significant. Too often, in the past, DP personnel have decided what users should have without consulting them. The DP personnel are then surprised when the users fail to show up for training.

4.2 Become Familiar with Available Data Bases

The training program developer should become familiar with currently installed data bases. The developer should be cognizant of the availability of the these data bases with regard to concurrent usage and be aware of any restrictions which have been placed on their access. They also need to be aware of which query facilities can be used in conjunction with which data bases.

4.2.1 Determine Whether Need to Develop New Data Bases

In some instances, it will be evident from the questionnaire, that end-users wish to access information which is not currently available on any existing data bases. If a new data base is required, a decision must be made as to whether this need is justified. Should it
be determined that a new data base has to be developed, training for those users requiring the new data base should not be scheduled until the data base development project has been completed.

4.3 Identify Products for Training

Once data bases have been analyzed, the product or products to be used in the training program must be identified. There may or may not be appropriate query facilities already installed and it may be necessary or desirable to acquire new ones. The following steps should be followed in analyzing these needs.

4.3.1 Analyze Installed Products

The developer of the training program should become familiar with the features of any already installed query facilities.

4.3.2 Investigate Option for New Product Acquisition

Should the option exist to acquire a new query product, analysis should proceed as follows:

4.3.2.1 Survey Available Query Products

A thorough study of all products on the market that are compatible with the organization's hardware/software configuration should be undertaken. In analyzing these products, the developer should pay particular attention
to the characteristics of query facilities as described in Chapter 2.

4.3.2.2 Acquire and Install Product/Products

Should the decision be made to acquire a new product, developers of the training program must allow sufficient time to acquire and install the product or products.

4.3.3 Categorize Products

The installed products, determined to be appropriate for training end-users, should be categorized as to which types of applications and for which users each is appropriate.

4.4 Select and Educate Trainers

Qualified trainers will constitute a critical element of the training program. It will be necessary to have qualified trainers available to teach and support the end-users for each product which is being offered in a training program.

The trainers should be intimately familiar with the product they are teaching. They should be aware of inconsistencies in the product so that they can adjust the training program to compensate for them. In selecting trainers, one should not overlook certain
desirable characteristics of trainers as mentioned in 3.10.

4.5 Develop Basic Method for Presenting Each Product

A general method of presentation should be developed for every product that is being offered. This presentation should strive towards presenting the trainees with an understandable conceptual model of the product (3.4.1). The syntax of the product will have a significant impact on the difficulty of the task (2.3). Some products will require little explanation. Others, which require the user to remember numerous syntactic details, will require ingenuity on the part of the trainer to devise methods which will soften the interaction between the user and the structure of the product. One means, previously mentioned (3.4.1), is the use of a template whereby the user selects and modifies the commands needed for his particular query. This greatly reduces the amount of information that the user must commit to memory.

The general nature of the course will become more specific when it is later adapted to the needs of a particular group. Certain general concepts should be independent of the particular trainees, however. These include the determination of the subset of commands
requisite to being able to understand and meaningfully use the product.

4.6 Compile End-Users Manual, Where Applicable

It must be determined whether it is necessary to compile an end-users manual. The purpose of this manual would be to present as simply as possible the information that the users will need to know to be able to productively use the query package. In many instances, the capabilities of the product are so extensive that many complex commands are included in the product’s users manual. These only tend to confuse the non-technical user. These users would benefit, at least initially, from having a simplified manual written in language they can understand. They may never require a more complex manual than this one. Care should be taken to insure that the information is presented in a layman’s terms avoiding all technical jargon.

4.7 Identify Target Group to be Trained

A follow-up should be made of the results of the questionnaire. Users who expressed a desire to learn a query product should be contacted. This follow-up could take the form of a personal interview in which the developer of the training program could become more
familiar with the users background and and the applications for which the user foresees needing a query product. In order to assure that the users derive the maximum benefit from the training program, the trainer needs to know about the users themselves as well as the functional areas with which they are involved.

This follow-up could also be used to determine what times would be convenient for the prospective trainees to attend class.

4.8 Determine Class Participants

A review should be made of the people who have agreed to a training program in order to arrive at the composition of a particular class. Consideration should be given to the backgrounds of the participants in order to arrive at a homogeneous grouping. In some instances, individuals from the same department may be interested in taking classes. Even though they would form a homogeneous group, it may be necessary to schedule them in separate classes because they cannot all be absent from their office at the same time.

Class size is an important factor in this type of training program. It cannot be overemphasized that conventional teaching methods are not appropriate for teaching this class of users. One-on-one instruction is
the ideal but, if this is not feasible, class size should be limited. It is unrealistic to think that a trainer can handle a class larger than 20-30 students during the instructional segment of the training. Additional trainers should be available during on-line sessions to insure that no trainer is responsible for more than four clusters of terminal users. No more than two students should be assigned to these terminal clusters. This is necessary to insure that the trainees are given the individualized attention they will need. The trainees will undoubtedly have questions as they first interact with the computer. Their questions need to be given prompt attention or the resultant frustration could cause the user to reject the computer entirely.

4.9 Schedule Class at Trainees' Convenience

In scheduling classes, a great deal of flexibility is required on the part of the developer of the program. This type of training is not conducive to the situation where class dates and locations are published in advance and users sign up for them. Circumstances may necessitate that the training be scheduled outside the normal working hours.

The length of time necessary to train users in a workable subset of a query facility will vary with the
particular product (2.3.5, 2.4.5). It will usually be beneficial to break training programs up into half-day sessions to allow the users to absorb what they have learned and to experiment with the system before learning more commands.

The locations of classes may also vary. In some instances, advantages may be gained by training the users in their own environment, utilizing the equipment that they will later use. This is advisable, however, only if the user can guarantee that the training session will be free of interruptions such as phone calls, visitors, etc.

If the class is larger than a few people, and the trainees wish to be trained at their site, they will need to provide a room that will accommodate any equipment that is needed for the class. They will have to be able to supply sufficient computer terminals so that no more than two people share a terminal during the on-line exercises.

An excellent tool for training groups of users to use a computer product is a terminal which can be hooked up to a large screen. As the trainer keys in commands on the terminal, they are displayed on the screen. This allows the trainees to see firsthand the actual responses from the computer. This also gives the trainer the opportunity to demonstrate the various aspects of the
product and to respond to the users' questions by actually showing them what would happen on the screen. This technique also makes it easier to adapt training programs to the users backgrounds. It eliminates the need for preparing transparencies or examples to accommodate all the different backgrounds of prospective students. The prototype systems can be developed and stored in the computer.

4.10 Plan Training Exercises around Trainees' Backgrounds

It will be necessary to provide examples of how the various commands of a product operate. If the trainees can identify these examples, as being applicable to the types of queries they would like to formulate, their interest will be enhanced. If they perceive that they will be able to adapt the on-line exercises to their own applications, they will have much more incentive to perform them. By having prior knowledge of the users applications, the trainer can develop simplified prototype systems along the lines that the user will be developing. This tends to personalize the training program for the trainees. They will be more attentive to a training program that has been designed specifically for them.
4.11 Conduct Training Program

Based on the mesh of characteristics of the query product and the traits of the trainees, modifications must be made to the training curriculum. An initial aspect to be considered is the amount of exposure that class participants have had to the computer.

4.11.1 Determine Amount of Exposure to the Computer

The trainees should be reassured at the onset of training, that no previous exposure is necessary in order to be able to successfully learn a query product (3.6.1). Non-technical users can have varying degrees of exposure to the computer. A particular class, however, by design, should all have similar amounts of experience. Two cases will be considered (1) those who have had little or no exposure to the computer, and (2) those who have had limited exposure to the computer. Excluded from this discussion will be those who have had extensive exposure to the computer because they would not be considered non-technical or casual users and would therefore not require the individualized treatment of this type of training program.

4.11.1.1 Little or No Exposure to the Computer

Eliminating technical jargon, explain the components of a computer system making analogies to concept with
which the user is already familiar i.e. files of
information as being analogous to information stored in a
filing cabinet.

4.11.1.2 Limited Exposure to the Computer

Review briefly, the components of the computer
system.

4.11.2 Explain Features of Terminal

In both cases, explain the features of the terminal
pointing out out special purpose keys and giving the
trainees hands-on experience so that they understand
their use.

4.11.3 Determine Need for Training in Command Language

After providing a general explanation of computers,
it will be necessary to consider whether the basic
features of the command language need to be taught.

Three cases can occur:

1. The user does not have to be familiar with the
   command language in order to use the query
   facility.

2. The user does have be familiar with the
   command language but already is.

3. The user does have to be familiar with the
   command language and is not.
4.11.3.1 Case 1:
Requires no further action in this step.

4.11.3.2 Case 2:
Review, briefly, the subset of commands that the user will need to know to utilize the query facility.

4.11.3.3 Case 3:
- Determine the subset of commands which must be taught so that the user can gain access to the system and utilize the query facility.
- Provide on-line training and experience in using these commands.

4.11.4 Data Base Principles
Depending upon the particular data model on which the query product is based, explain in as simplified terms as possible, the minimum amount of data base principles that the user will need to know to use the query facility (2.6, 3.9).

4.11.5 Error Recovery
Acquaint the trainees with various avenues available to them should an error occur. If on-line HELP facilities exist, instruct the trainees in their use, giving them on-line experience in handling some predetermined error situations. If the only recourse is off-line documentation, explain its structure and how to
use it to locate the response to an error message. Make the documentation available to the trainees so that they can become familiar with and use the documentation during training.

4.11.6 Data Dictionary

Provide training in the procedure necessary to access the Data Dictionary associated with the query product so that they will know the correct names of the data items (3.9). Again, provide the trainees with the opportunity for hands-on experience. The importance of providing hands-on experience at each step of the actual training cannot be overemphasized.

4.11.7 Subset of Commands

Explain a basic subset of commands of the query product, emphasizing an understanding of the principles of the product and how it works (2.8, 4.5). This subset of commands should be broken down into integral groups so that the trainee is given hands-on experience before proceeding to learn the next set of commands. As a rule of thumb, these integral subsets should not exceed five commands. Some commands can naturally be grouped together such as various formatting commands or functional commands such as MIN, MAX, etc.
4.11.8 Monitor Trainee's progress

Throughout the training session, provide the opportunity for the students to ask questions. This is especially critical during the period when they are experimenting with the computer.

4.12 Evaluate Training Program

Especially important in training programs is the need to determine if the users are grasping what is being presented to them. They should be able to perform the following functions after completing the training program:

- translate an English request into the query facility
- comprehend an existing query
- find the errors in an existing query

An evaluation can be made by analyzing the ease with which the trainees are able to perform the on-line exercises. This can facilitated by analyzing "PHOTO" (1.3) outputs of the terminal sessions. The trainers should be able to determine which commands the users found most difficult to use and to ascertain whether the errors made were syntactic or due to a lack of understanding of the principles of the product. Based on the results of the evaluation and after soliciting
constructive criticism from the trainees, the course should be modified accordingly.

4.13 Provide Continued Support for Trainees

A training program of this nature does not cease at the end of the formal training session. Continued support and assistance must be provided to the trainees after they return to their jobs and begin to productively use the query facility (3.10). They will need a "safety net" for some time until they are comfortable with the system and the product. They need to feel that they have a recourse should they attempt to write a query and have difficulty. This necessitates that there be user service personnel to provide this needed support on a continuing basis after the training program ends.

4.14 Determine Whether all Target Groups Have Been Trained

Coupled with providing continued support for past trainees, it will be necessary to determine if all those who have indicated a desire to take training have been taught.

4.14.1

If there are still groups to be taught, return to step 4.8 to determine the next class participants.
4.14.2 Publicize Training Program

If all groups have been taught, publicity should be circulated similar to the original questionnaire publicizing the training program, explaining what products are available and what prerequisites, if any, are required before taking the course. The only thing that might distinguish this publicity from the initial publicity, aimed at assessing the potential trainee population, would be that the candid comments of previous trainees, who are now using the query facility productively, could now be included.

Based on the responses to the various means of publicizing the course, return to Step 1 of the methodology to determine if there are any more potential trainees.
1. Assess Potential Trainee Population
2. Become Familiar with Available Databases
3. Identify Products for Training
4. Select and Educate Trainers
5. Develop Basic Method for Presenting Each Product
7. Identify Target Group to be Trained
8. Determine Class Participants
9. Schedule Class at Trainees' Convenience
10. Plan Training Exercises around Trainees' Backgrounds
11. Conduct Training Program
12. Evaluate Training Program
13. Provide Continued Support for Trainees
14. Determine Whether All Groups Have Been Trained

Figure 4-2: Prescriptive Methodology
CHAPTER 5

RESULTS, CONCLUSIONS, AND FURTHER RESEARCH

A step by step implementation of the prescriptive methodology was conducted as described in Appendix III. The evaluation form, distributed to the trainees at the end of the training program, is duplicated on the following pages. Included with each question is a count of the number of trainees (out of 6) who gave each response. A discussion of the evaluation form responses and the PHOTO sessions evaluation is included in the succeeding evaluation of the methodology.
I Instructional Period:

1. Were the sessions:

   (6)
   Too Long <-----------------> Too Short
   X  X  X  X  X

2. Did you feel that the number of commands covered during each session was:

   (6)
   Too Many <-----------------> Too Few
   X  X  X  X  X

3. Did you find the IOL End-User's Guide:

   (1)  (4)  (1)
   No Value <-----------------> Very Helpful
   X  X  X  X  X

4. Did you get enough personalized attention?

   (6)
   Yes ______  No ______

5. Did you find the training program manual:

   (1)  (2)  (3)
   No Value <-----------------> Very Helpful
   X  X  X  X  X

Figure 5-1: Tabulated Evaluation Form
II On-line Exercises

1. Were the number of exercise sessions:
   \[(4) (2)\]
   Too Many \[\rightarrow\] Too Few
   X X X X X

2. Did you find the on-line exercises:
   \[(1)(1)(4)\]
   Too Difficult \[\rightarrow\] Too Easy
   X X X X X

3. Would you have preferred to work with a partner?
   \[(3)\]
   Yes ______ No_______ Don't Know ______

4. Were your questions answered:
   \[(6)\]
   Promptly Yes_______ No_______
   Satisfactorily Yes_______ No_______
   (1 left blank)

Figure 5-1, continued
III Sessions in General:

1. Were there any commands or query statements in particular that caused you difficulty? If so, which ones?
   - FIND KEY (3)
   - SORT (1)
   - Ordering of query statements (1)

2. Would you have preferred to have had one all day class instead of two half-day sessions?
   - (2) (4)
   - Yes _____  No _____

3. Do you feel you have enough basic knowledge to start using SQL?
   - (6)
   - Yes _____  No _____

Figure 5-1, concluded
5.1 Results and Conclusions

A step by step discussion of the proposed methodology, noting any points which were reinforced or disproved based upon the aforementioned implementation follows.

5.1.1 Assess Potential Trainee Population

Since the potential trainee population was in effect pre-identified, this step in the methodology was already accomplished. It is, nonetheless, an important step to be considered. Unless there are people who would benefit from a training program of this kind, there is little point in proceeding further with the methodology.

The author's conviction that the usefulness of the inventory system developed for the Lehigh University Computing Center (LUCC) would be greatly enhanced if the end-users could learn to develop their own ad-hoc queries was actually the primary impetus that led to the development of this thesis.

5.1.2 Become Familiar with Available Data Bases

This step was facilitated in the aforementioned implementation because there was only one data base with which to become familiar. This would not always be the case, and is still seen as a critical step at this point. Unless the developers of the training program are aware
of what data bases exist, they will not know whether the potential trainees would be able to put to use the proposed training to which they would be devoting their time and energy.

5.1.3 Identify Products for Training

Prior to developing a training program, it is obvious that there must be a product to be taught. This was predetermined to be IOL in the named implementation. Some steps are intuitively necessary. The only question that arises is at what point in the methodology they should be executed. Nothing in the implementation indicated that this step was misplaced.

5.1.4 Select and Educate Trainers

If the trainer does not understand the products they are teaching, they cannot expect the trainees to comprehend what is presented to them. The trainees need to feel confident that their questions will be answered. If they perceive that the trainers do not understand what they are teaching it will undermine the trainees' confidence that they will be able to understand the material. The evaluation form tabulations indicated that all the trainees felt that their questions were answered promptly and all but one respondent (who left the
question blank) felt that their questions were answered satisfactorily.

5.1.5 Develop Basic Method for Presenting Each Product

In developing a basic method for presenting IQL, different techniques were used to attempt to impart an understanding of the conceptual model of IQL.

One such technique was the use of a query template whereby the trainees could modify a sample query to suit their present query needs. It was felt that this aid would assist the users so that they would not have to remember as many details. Whether they did not understand the concept, chose not to use it, or did not feel the need to use it, none of the trainees utilized this training aid.

The concept of environments was introduced in the IQL training, as is evident from many of the slides in the IQL training manual, to assist the trainees in understanding when each command is valid. Even with this emphasis, one trainee indicated that confusion existed as to when certain commands could be used.

The format appropriate for presenting IQL could differ significantly from the method to be used in training in other query languages. It is important to consider each product for which there will be a training program and give serious thought to how it should be
5.1.6 Compile End-Users Manual, Where Applicable

An End-Users manual (Appendix VIII) as well as training manuals (Appendices IV and V) were developed for this training program. Based on the tabulated results of the evaluation forms and observations of the trainees during the on-line sessions, these manuals were very helpful to the trainees. Five out of six trainees found the End-Users Guide very helpful. The one trainee who indicated that it was of little value was the most experienced user. This trainee added a comment that he did not use the End-User's Guide during the training sessions but probably would later.

Similar responses were given to the question concerning the training program manual. Five out of six trainees found the training program manual very helpful. The one trainee who indicated that it was of little value was the trainee who ranked second in amount of previous computer experience.

These responses were corroborated by the observation of the trainer that the more experienced users responded to errors by using the on-line HELP facilities while the inexperienced users relied more on the printed documentation, especially the training program manual. On-line HELP facilities were explained during the
lecture, however, they should have either been covered more thoroughly or more opportunity should have been given for on-line practice in their use. The more experienced user was more comfortable with the system and not afraid to make an error. The inexperienced user, on the other hand, wanted to be certain before entering any information into the computer. These observations support the importance of providing end-user manuals for this class of users.

5.1.7 Identify Target Groups to be Trained

The importance of this step varies with the size of the potential trainee population identified in the initial step of the methodology. If the potential trainee population is large, it will be necessary to classify them according to experience and homogeneity of backgrounds to arrive at workable subsets of trainees to be scheduled into particular classes. This was not the case in the aforementioned implementation, since only one training session was planned and the trainee group was pre-identified.

5.1.8 Determine Class Participants

In determining participants of a particular class, ensuring homogeneity of trainees' background, keeping class size small, and ensuring that no more than two
people shared a terminal during the on-line exercises were stressed. The actual implementation supported these requirements as explained below.

The backgrounds of all the trainees were not homogeneous. Two of the trainees had a significant amount of previous exposure to the computer. This caused some problems during the training sessions in that the trainee with the least amount of experience felt pressured because she was not able to perform the exercises as quickly as the others. This trainee required much more individualized attention during the on-line exercises. She was also the only trainee who indicated on the evaluation form that she felt that the exercises tended to be too difficult. If the entire class had been composed of trainees with backgrounds similar to hers, the exercises could have been made simpler and some of the more complex features of the language could have been omitted.

Homogeneity of backgrounds is more comprehensive than just ensuring that all participants have similar amounts of computer exposure. The trainee, in this instance, who lacked previous computer exposure, also lacked an understanding of data bases and queries thereon. She was not able to abstract and formulate the types of questions appropriate for obtaining information
from a data base. This would not necessarily follow for all non-technical users. Many are well versed in formulating the questions that would typically be used to query a data base. They, heretofore, have merely lacked a means of obtaining this information without going through an intermediary.

Decisions had to be made in determining how many people should share a terminal during the on-line sessions. When queried after the training, responses were evenly split as to whether the trainees would have preferred to work with a partner or work alone. Three trainees preferred to work with a partner and three did not. The three who indicated a preference for working with a partner were the ones who alternately had worked with a partner because of a malfunction of one of the terminals. The fact that half of the trainees preferred to work alone supports the premise that no more than two people should have to share a terminal during the on-line exercises.

The trainees all felt that their questions were answered promptly and satisfactorily. If the class size had been larger, this might not have been true. Assisting six trainees at the terminal sessions kept the trainer quite busy answering questions. Considering the fact that most of these questions were coming from four
of the trainees, additional class members would have necessitated having an assistant to the trainer during the on-line exercises.

5.1.9 Schedule Class at Trainees' Convenience

This step is closely linked with determination of class participants. Originally, it was placed after the "Planning the Training Exercises Around the Trainees' Background" step, but because one cannot be certain as to the class participants until a class has actually been scheduled, the ordering of the steps was changed. Even with a class of six trainees, scheduling problems arose when trying to arrive at a time that would be convenient for all trainees. During the training sessions, the Operations Manager's office was essentially unmanned. Were it not for the willingness of the Operations Manager to be of assistance in the trainer's endeavor, it would have been necessary to conduct two different training sessions to accommodate her entire staff. These problems are real and need to be considered when scheduling classes.

5.1.10 Plan Training Exercises around Trainees' Backgrounds

The training exercises in this implementation were based on an inventory application. It was interesting to
note that one trainee commented that the exercises should have been of a more practical nature. This trainee was the only one who did not have a specific interest in an inventory application. Her interest in attending the training program was in learning the language per se. This supports the contention that the exercises should be based on an application with which the trainees are familiar.

5.1.11 Conduct Training Program

Although this is an obvious step in the methodology, questions arise as to how to conduct the actual training sessions.

Responses to questions on the evaluations form dealing with the instructional period indicated that all the trainees were comfortable with the length of the sessions and with the number of commands covered during each session. Each also felt that they had received enough personalized attention.

Breaking the training program down into two segments, one for training in the command language and one for training in IQL was found to be helpful. The trainees who were already familiar with the command language were not willing to spend a half-day reviewing it.

The IQL training program was divided into half-day
sessions. Four out of six trainees responded on the evaluation form that they preferred two half-day sessions to one all-day session. One of the two who preferred one all-day session was the trainee with the least experience. It is the trainer's opinion that this trainee could not have absorbed an entire day's worth of material. Perhaps it was that she wanted to get it over more quickly because she was having difficulty.

5.1.12 Evaluate Training Program

The evaluation step is a critical one in the methodology. By studying what has been done, one is able to identify problem areas and correct them. By analyzing the evaluation form tabulations and the output of the PHOTO sessions, it was possible to isolate the types of errors made by the trainees and based on their frequency identify the cause. When the same type of error was made by several trainees, it was an indication that the training program should be modified to clarify these concepts. The PHOTO sessions provided a valuable resource in this respect.

It was originally the intention of the author to score the errors made by the trainees while performing the on-line exercises. By classifying and quantifying the errors, it was hoped to determine a measure of how well the trainees had grasped the material. Based on
observations made by the trainer during the on-line exercises, this method would have produced inconclusive results. The more experienced users asked very few questions. They preferred to experiment with the system and recover from their errors. The inexperienced user, on the other hand, asked many more questions and because of the help and guidance they received made fewer mistakes. This is not to say, however, that the latter trainees had a better grasp of the material. In reality, the opposite was true.

In order to provide an accurate measure, the trainees would have had to perform the exercises without any assistance. The results could then have been scored by classifying and counting the errors made by each trainee. It was judged more important, in this instance, to provide the trainees with the needed assistance.

In determining what commands should be taught in this introductory course, the trainer was uncertain whether to include certain procedures which she feared might cause the trainees difficulty. By analyzing the PHITO sessions, it became clear that the trainer's concerns were well founded. TOBI is more procedural than many of the other query languages on the market. The user must understand the concept of sorting a file and how to link records from one dictionary to another. It
performs these functions by means of a "SORT" query statement and a "FIND KEY" query statement respectively. These were the two statements that caused the trainees the most difficulty as evidenced by both the PHOTO sessions and responses on the evaluation form to the question which asked "Were there any commands or query statements in particular that caused you difficulty?"

Three of the six respondents mentioned the "FIND KEY" statement and one mentioned the "SORT" statement. This signaled to the trainer that more time and examples should be devoted to these concepts in future training programs.

By analyzing the PHOTO sessions and noting the types of questions asked during the on-line exercises, it became clear that the wording of some of the exercise sessions caused confusion among the trainees. The following exercise demonstrates this point.

Create and store a query that will list all "TUC" equipment that is under maintenance contract (Service = "M") giving the Vendor-No, Model-Name, Serial-No, Cust-No, and Cust-Name.

It appears that the exercise would have been more appropriate if worded:

Create and store a query that will list all "TUC" equipment that is under maintenance contract (Equip-Service = "M") giving the vendor number, model name, serial number, customer number, and...
customer's name.

The primary cause of confusion appeared to be the names of the items. The vendor number, for example, is a data item that exists in three dictionaries:

Vendor Dictionary as Vendor-No
Equipment Dictionary as Equip-Vendor-No
Model Dictionary as Model-Vendor-No

When it was listed in the exercise as "Vendor-No", the trainees assumed that this was the exact label of the data item. By expressing it as "vendor number", they would realize that they would have to locate the data item in the appropriate dictionary and find the correct label. The correct label for this query would be Equip-Vendor-No not Vendor-No. Since four out of six trainees made this error, on the spot corrections were made to the exercises, and the training manual (Appendix V) reflects these revisions.

The fact that adjustments of this nature should be made before conducting any future training programs demonstrates the importance of the evaluation step.

One of the reasons for evaluating the training program was to make a determination whether IQL was a suitable language for this class of user. Based on the results of the training program, even though IQL has some problems, it still proved beneficial to these end-users.
in extending the usefulness of their inventory systems. All of the trainees indicated that they felt that they had enough basic knowledge to begin using IQL and one trainee commented that knowledge of IQL would be of great use to her.

5.1.13 Provide Continued Support for Trainees

As a result of this implementation, it became clear to the trainer that the support provided for the trainees should not be passive. Even though the trainees were told to feel free to call if they had any questions, the trainer was not contacted with any questions or problems. It cannot be assumed that if the trainees do not call with questions, that the trainees have none. After giving the trainees an opportunity to experiment with the system and having had no response from them, the trainer contacted the trainees to see if they were having any problems. The trainees in the Operations Manager's department had written queries to augment the inventory system. In addition, the trainee with the least experience, who had commented on the evaluation sheet that she did not think she would ever have a need to use IQL, had written a query to augment a tape handling system that uses IQL as a basis for reporting.

When designing the training program, a decision had been made as to which commands to include. There were a
few commands that, should time permit, the trainer had intended to cover. Based on the results of the class, it was deemed inappropriate to cover them during the regular sessions. While visiting with the three members of the Operations Manager's staff, their questions suggested that knowledge of one of these commands would be helpful. The opportunity was thus given to explain this command and give them an insert to the End-Users Guide describing it. Had the trainer merely waited for the trainees to call, she would never have been aware of the difficulty the trainees were having.

5.1.14 Determine Whether All Target Groups Have Been Trained

If all groups have not been trained, they need to be considered for training. Also, depending upon the number that need to be trained and the staff available to train them, development of publicity to attract a new potential trainee population could be done concurrently.

Originally this step in the methodology stated that when all groups had been trained, the developer should return to Step #7 to determine new target groups. After due consideration, it was decided that it would be better instead to return at this point to the beginning of the methodology. Some of the beginning steps of the methodology may not need to be repeated but they should
at least be considered. For example, new data bases may have been installed or new products acquired for which a method of presentation would have to be determined. If all has remained static in the organization in this respect, these steps would in effect be completed, but the developers of the training program would then be certain that their information was current.

Since the author planned on making only one implementation, this step signaled the end of the methodology. At the completion of the training program, one trainee indicated that he knew of another user who would have benefited from the training program. Satisfied users can provide the best means of publicity and this resource should not be overlooked when publicizing the training program.

5.1.15 Summary

The results of the implementation suggested that the methodology is complete; additional steps are not required and none of the current steps are superfluous. A question did arise with the ordering of some of the steps; but, as a whole, the methodology provided a sound preparation for the conduct of the training program.
5.2 Further Research

The IQL implementation provided an initial test of the methodology. As indicated, some points were substantiated based on the results of the implementation. Additional controlled experiments could be run examining each step of the methodology. Specific examples follow which would be appropriate for particular steps.

5.2.1 Assess Potential Trainee Population

Different means of publicizing a training program could be developed. For example, questionnaires could be devised that would awaken the interest of non-technical personnel and, at the same time, convey the most amount of information to the training program developer about the prospective trainees. More investigation would be necessary before attempting to select the most effective means of publicizing a training program.

5.2.2 Identify Products for Training

Additional research could help to determine and classify which query languages are appropriate for different classes of users. This research effort would be dynamic because products of this nature are constantly being introduced or improved.
5.2.3 Develop Basic Method for Presenting Each Product

Different methods and aids could be researched that could be employed to aid the non-technical user in understanding how a query language works.

One trainee commented on the evaluation form that it would have been helpful to see more examples of each command. This request would be much easier to accomplish through the use of an "Advent" screen type facility (4.9) than by increasing the number of examples written on the blackboard or by preparing additional transparencies. An additional implementation using an "Advent" screen type facility could determine its usefulness as an additional visual aid.

5.2.4 Select and Educate Trainers

Since the non-technical person is emerging as a computer user, research would be useful as to what characteristics those who are to train these individuals should possess. Characteristics were suggested in 3.10 but further research could be done in defining these characteristics.

5.2.5 Compile End-Users Manual, Where Applicable

Development of appropriate procedures for compiling end-user manuals would be helpful. These manuals should express as simply as possible, without insulting the
intelligence of the reader, the important features of the language. Analyses could be made of different end-user manuals in order to determine which manuals are most effective.

5.2.6 Evaluate Training Program

Additional research could also be done to develop procedures for evaluating training programs. The evaluation form used in the TOL implementation could be refined to provide better insights into the strengths and weaknesses of the training program.

5.2.7 Provide Continued Support for Trainees

"Information Centers" have been installed in numerous organizations as a means of providing continued support for end-users. This is a relatively new concept and could be the focus of further research.

5.2.8 Determine Whether All Target Groups Have Been Trained

Methods of publicizing a training program by incorporating the comments of previous satisfied trainees could be researched. One possibility would be to publish a newsletter where end-users describe the increased productivity that they have achieved from using a query language.
5.2.9 Summary

Much work lies to be done in the area of training non-technical users to interact with a computer. Their involvement with computers is definitely becoming more predominant. This methodology has provided a beginning.
BIBLIOGRAPHY

[Beeler 81] Beeler, Jeffrey.
Training of Non-Tech End Users Seen Vital
to High Productivity.

[Boyce 74] Boyce, R. F., D. D. Chamberlain and W. F.
King III.
Specifying Queries as Relational
Expressions.
In J. W. Klimbie and K. L. Koffeman,
editors, Data Base Management, pages

King, R. A. Lorie, J. W. Mehl, T. G.
Price, M. Schkolnick, P. Griffiths
Selinger, D. R. Slutz, B. W. Wade, R. A.
Yost.
Support for Repetitive Transactions and Ad
Hoc Queries R.
ACM Transactions on Database Systems

[CODASYL 79] CODASYL End-User Facilities Committee
Status Report.
Information and Management 2:137-163,
1979.

[Codd 74] Codd, E. F.
Seven Steps to Rendezvous with the Casual
User.
In Data Base Management, pages 179-200.

[Crane 82] Crane, Janet.
The Changing Role of the DP Manager.

[Cuff 78] Cuff, R.N.
Database Query Systems for the Casual
User.
Technical Report FES-MMS-CAS-79,
Department of E.E., University of

[Cuff 80] Cuff, Rodney N.
On Casual Users.

[Datapro 81] Datapro Directory of Software
Datapro Research Corporation, 1981.

[Date 81] Date, C.J.
*An Introduction to Database Systems.*
Addison-Wesley, 1981.

[Denny 66] Denny, J.P.,
Effects of Anxiety and Intelligence on Concept Formulation.

The Black Box inside the Glass Box: Presenting Computer Concepts to Novices.

[Dunn 68] Dunn, R.F.
Anxiety and Verbal Concept Learning.

Behavioral Aspects of Text Editors.

[Engle 81] Engle, William R.
High Level Languages.

[Francie 80] Francie, Charlene, W. Thomas Lin.
IN DEPTH: The Gravy Train Derails.

[Frank 82] Frank, Ronald A.
Productivity - Let the Users Program.

[Gottfried 81] Gottfried, Ira S.
Understanding the New-Breed DPer.


[Gruenberger 81]

Gruenberger, Fred.
Making Friends with User-Friendly Systems.

[Harris 77]

Harris, L. R.
User-Oriented Data Base Query with the ROBOT Natural Language Query System.

[Joyce 76]

Joyce, J. N., and N. N. Oliver.
REGIS - A Relational Information System with Graphics and Statistics.

[Kennedy 74]

Kennedy, T. C. S.
The Design of Interactive Procedures for Man-Machine Communications.

[Kennedy 75]

Kennedy, T. C. S.
Some Behavioral Factors Affecting the Training of Naive Users of an Interactive Computer System.

[Lachman 79]

Lachman, Roy, Janet L. Lachman, Earl C. Butterfield.
*Cognitive Psychology and Information Processing: An Introduction.*

[Ling 80]

Ling, Robert F.
General Considerations of the Design of an Interactive System for Data Analysis.

[Lochovsky 78]

Lochovsky, Frederick H.
*Data Base Management System User* 113
Performance.


Mayer, Richard E.
Comprehension as Affected by Structure of Problem Representation.

Mayer, Richard E.
Advance Organizers That Compensate for the Organization of Text.

Mayer, Richard E.
A Psychology of Learning BASIC.

Mayer, Richard E.

Mccracken, Daniel.
Systems Productivity: Harvesting the Rewards.

McLeod, Dennis and Monte Meldman.
RISS - A Generalized Minicomputer Relational Data Base Management System.

Miller, L. A.
Programming by Non-Programmers.

Miller, Lance A. and John C. Thomas, Jr.
Behavioral Issues in the Use of Interactive Systems.

Mollen, David C. and Van Baskshi.
How to Support Company End Users.
[Moran 81] Moran, Thomas P.
An Applied Psychology of the User.

[NCSS 78] NOMAD Report Writer I Workbook

[NCSS 79] Your NCSS Computer System THE BASICS

[Nickerson 81] Nickerson, Raymond S.
Why Interactive Computer Systems are
Sometimes Not Used by People Who Might
Benefit From Them.
Int. Journal of Man-Machine Studies

[Plath 76] Plath, W.J.
REQUEST: A Natural Language
Question-Answering System.
IBM Journal of Research and Development

[Plum 77] Plum, Thomas.
Fooling the User of a Programming
Language.
Software Practice and Experience

[Pope 81] Pope, Maureen L., Mildred Shaw.
Personal Construct Psychology in Education
and Learning.
Int. Journal of Man-Machine Studies

[Read 81] Read, Nigel S. and Douglas L. Harman.
Assuring MTS Success.

[Reind 81] Reind, P., Van de Riet, Anthony
I. Wasserman, Martin L. Kerstein, and
Wiebren de Jonge.
High-Level Programming Features for
Improving the Efficiency of a
Relational Database System.
ACM Transactions on Database Systems

[Reisner 81] Reisner, Phyllis.


ORACLE, A Possible New Breed of DBMS.

[Yao 79] Yao, S. Bing.
ACM Transactions on Database Systems

[Zloof 75] Zloof, Moshe M.
QUERY-BY-EXAMPLE.

[Zloof 76] Zloof, Moshe M.
QUERY-BY-EXAMPLE, Operations on Hierarchical Data Bases.
APPENDIX I

PHOTO SESSION EVALUATION

The PHOTO sessions of the on-line exercises performed by the trainees during the training program served as a tool in evaluating the training program and subsequently the methodology from which the training program was developed.

Prior to conducting the training sessions, it was deemed appropriate to develop a scoring method to be used in evaluating the PHOTO sessions. Specific areas determined worthy of evaluation included:

I.1 On-line Exercises

Initial exercise sessions lead the trainees step by step through an exercise. Later sessions require that they develop their own procedures. The PHOTO sessions will illustrate the steps the trainees took to generate a query. They may indicate that certain exercises were too difficult or were poorly worded causing the trainees to have difficulty understanding them.

I.2 Response to Errors

The PHOTO sessions will also show how the trainees responded to errors, i.e. whether they made use of the on-line HELP facilities and whether they were able to understand the system generated error messages.
I.3 Course Content

The PHOTO sessions will give an indication as to whether the amount of information that the trainees could absorb was either overestimated or underestimated. Judgments had to be made in designing the training program as to which commands should be covered, the number of commands to teach in a particular session and how to the group the commands being taught. How well the trainees were able to complete the exercises should indicate whether these judgments were accurate.

I.4 Value of Template as Training Aid

An IQL template "IQLSAMPLE" was introduced during instruction as a training aid. The PHOTO sessions will show whether the trainees made use of this aid and the manner in which they used it.

I.5 Human-Computer Interaction

The PHOTO sessions should also give an indication of how well the trainees were able to interact with the system and IQL. This should reveal whether enough attention was given during training to the characteristics of the user as developed in Chapter 3.
I.6 Error Classification and Quantification

The errors that the trainees made while performing the on-line exercises can be classified and quantified as follows.

Each question which requires development of a query could be evaluated for correctness. Points could be assigned to these exercises as follows:

3 completely correct
2 syntactic errors i.e. spelling
1 minor logic errors such as mismatch of item-name and dictionary
0 severe logic error

These points could be tabulated across all exercises, for individual exercises, for individual questions within exercises, and for individual trainees. A predetermined weight could be assigned to each question.

I.7 Summary

These sessions must be evaluated in conjunction with the number and types of questions asked during the on-line exercises in order to ascertain the strengths and weaknesses of the training program.

Considering the wealth of information available through these PHOTO sessions, it becomes difficult to
decide on which issues to focus. This pilot implementation of the methodology may make these issues clearer. It would be the objective of future tests to single out particular issues and quantify them. A multitude of experimental tests could be run varying the elements of the training program as well as the class composition.
APPENDIX II

TABLE OF CURRENTLY AVAILABLE PRODUCTS

Legend of Column Headings:

1 - Query Language
2 - Report Generator
3 - Graphics Generator
4 - Application Generator
5 - High-Level Programming
6 - On-Line?
7 - Suitable for End-Users?
<table>
<thead>
<tr>
<th>Product</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Vendor</th>
<th>Machine Type</th>
<th>6</th>
<th>7</th>
<th>Data Base</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADAM</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>San Francisco Intelligent Machine Corp.</td>
<td>Own Machine</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADMIN 11</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>ADMIN11 Corp.</td>
<td>PDP/11</td>
<td>X</td>
<td>X</td>
<td>Its own files</td>
</tr>
<tr>
<td>ADPAC</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Adapac Computer Languages Corp.</td>
<td>IBM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADRS (APL-based)</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>T. D. S. B.</td>
<td>IBM</td>
<td>370</td>
<td>X</td>
<td>Its own files</td>
</tr>
<tr>
<td>APL</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>Various</td>
<td>Various</td>
<td>X</td>
<td>X</td>
<td>DL/I; IMS Applications</td>
</tr>
<tr>
<td>APL-DI</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>TBA</td>
<td>370</td>
<td>X</td>
<td>X</td>
<td>For IMS Applications</td>
</tr>
<tr>
<td>APL-PLUS</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>STSC</td>
<td>Various</td>
<td>X</td>
<td>X</td>
<td>Queries IMS</td>
</tr>
<tr>
<td>ADF</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>TBM</td>
<td>370/OS</td>
<td>X</td>
<td></td>
<td>Its own files</td>
</tr>
<tr>
<td>ASI INQUIRY</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>APP Software</td>
<td>370</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASI-ST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Applications Software Inc.</td>
<td>IBM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASRES</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Univ. of Georgia</td>
<td>360</td>
<td>X</td>
<td></td>
<td>Its own files</td>
</tr>
<tr>
<td>BASIS</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>Ratelle</td>
<td>Various</td>
<td>X</td>
<td>X</td>
<td>Its own Info. Retr. System</td>
</tr>
<tr>
<td>CAPS</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TCL Series</td>
<td>2900</td>
<td>X</td>
<td>X</td>
<td>Its own Info. Retr. System</td>
</tr>
<tr>
<td>COBOL Program Generator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Black/Prime</td>
<td>Prime</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>Vendor</td>
<td>Machine Type</td>
<td>6</td>
<td>7</td>
<td>Data Base</td>
</tr>
<tr>
<td>---------------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---------------------</td>
<td>---------------</td>
<td>---</td>
<td>---</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>COMPOSE-II</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Compusource</td>
<td>Data General</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONTEXT C-705</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>CONTEXT System</td>
<td>370</td>
<td>X</td>
<td></td>
<td>Its own files</td>
</tr>
<tr>
<td>CS4</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>Databaskonsult Sweden</td>
<td>DEC/IBM</td>
<td></td>
<td></td>
<td>Associative records</td>
</tr>
<tr>
<td>DATA ANALYZER</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>Prog.Products</td>
<td>370</td>
<td>X</td>
<td>X</td>
<td>Its own files</td>
</tr>
<tr>
<td>DATATRIEVE-11</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>DEC</td>
<td>PDP/11</td>
<td></td>
<td></td>
<td>DEC's files and data base</td>
</tr>
<tr>
<td>DMS</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>TAM</td>
<td>370</td>
<td>X</td>
<td></td>
<td>DL/I CICS and its files</td>
</tr>
<tr>
<td>DMS</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>TAM</td>
<td>8100</td>
<td>X</td>
<td></td>
<td>Its own files</td>
</tr>
<tr>
<td>DIRECT</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ranchohio</td>
<td>Honeywell</td>
<td></td>
<td></td>
<td>Its own files</td>
</tr>
<tr>
<td>Display Info.</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>TAM</td>
<td>S/38</td>
<td>X</td>
<td></td>
<td>For S/38 DR</td>
</tr>
<tr>
<td>Facilities</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Fm.I. Ltd.</td>
<td>Various</td>
<td>X</td>
<td>X</td>
<td>Its own files</td>
</tr>
<tr>
<td>DRS-11</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>Synoco Ltd.</td>
<td>DEC</td>
<td>X</td>
<td></td>
<td>Its own files</td>
</tr>
<tr>
<td>EASYTRIEVE</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>Panasonic</td>
<td>Various</td>
<td>X</td>
<td>X</td>
<td>IMS, TOTAL, ADABAS</td>
</tr>
<tr>
<td>EQUUS</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>TAM</td>
<td>S/3, S/32</td>
<td>X</td>
<td></td>
<td>Its own files</td>
</tr>
<tr>
<td>EXTRACTO</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Notipro Ltd.</td>
<td>370</td>
<td>X</td>
<td></td>
<td>Its own files</td>
</tr>
<tr>
<td>FLFXTRACT</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Faulkner</td>
<td>Various</td>
<td>X</td>
<td></td>
<td>Its own files</td>
</tr>
<tr>
<td>Product</td>
<td>Vendor</td>
<td>Machine Type</td>
<td>Data Base</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------</td>
<td>--------------</td>
<td>---------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FOCUS</td>
<td>Info. Builders</td>
<td>370</td>
<td>X</td>
<td>Its own files</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gen. Info.</td>
<td>IBM</td>
<td>370</td>
<td>X</td>
<td>Its own DB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Systems</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GENASYS</td>
<td>Generation</td>
<td>IBM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sciences</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GIS</td>
<td>TBM</td>
<td>Various</td>
<td>X</td>
<td>DL/IMS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HARVEST</td>
<td>T.D.S.R.</td>
<td>DEC/IBM</td>
<td>X</td>
<td>Its own files</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IBM 3790</td>
<td>TBM</td>
<td>370</td>
<td>X</td>
<td>Its own files</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IBM BRADS</td>
<td>TBM</td>
<td>5110/20</td>
<td>X</td>
<td>Its own files</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCESS</td>
<td>Univac</td>
<td>Univac</td>
<td>X</td>
<td>Its own files</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IDEA</td>
<td>Data General</td>
<td>Data</td>
<td>X</td>
<td>Its own files</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IDEAL</td>
<td>ADR</td>
<td>370</td>
<td>X</td>
<td>DATACOM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HIST DATA</td>
<td>Int. Inf. Sys.</td>
<td>DEC</td>
<td>X</td>
<td>Its own files</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOGGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IXDOC</td>
<td>A.G.</td>
<td>370</td>
<td>X</td>
<td>Its own files</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMS FASTOW</td>
<td>TBM</td>
<td>370</td>
<td>X</td>
<td>IMS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INFORMAT</td>
<td>R.P.</td>
<td>NOVA</td>
<td>X</td>
<td>Its own files</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INQUIRY</td>
<td>Informatics</td>
<td>370</td>
<td>X</td>
<td>IMS and</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>own files</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>Vendor</td>
<td>Machine Type</td>
<td>6</td>
<td>7</td>
<td>Data Base</td>
</tr>
<tr>
<td>-------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---------------------------------------------</td>
<td>--------------</td>
<td>---</td>
<td>---</td>
<td>---------------------------</td>
</tr>
<tr>
<td>INSCI</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Information Science</td>
<td>Various</td>
<td>X</td>
<td></td>
<td>Its own files</td>
</tr>
<tr>
<td>INSYTE</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Remote Comp.</td>
<td>Burroughs</td>
<td>X</td>
<td>X</td>
<td>Its own files</td>
</tr>
<tr>
<td>INTELLECT</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Artificial Intelligence</td>
<td>Man. Group</td>
<td></td>
<td></td>
<td>Various</td>
</tr>
<tr>
<td>IQ</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Information Science</td>
<td>Various</td>
<td>X</td>
<td>X</td>
<td>IDS, IMS, TOTAL</td>
</tr>
<tr>
<td>IRS</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Sigma Data</td>
<td>370</td>
<td>X</td>
<td></td>
<td>Its own files</td>
</tr>
<tr>
<td>ISA/OAS</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>ISA Corp.</td>
<td>370</td>
<td>X</td>
<td></td>
<td>Its own files</td>
</tr>
<tr>
<td>JPDLIS</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Univ. of Georgia</td>
<td>Univac</td>
<td>X</td>
<td></td>
<td>Its own files</td>
</tr>
<tr>
<td>LITRS</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Univ. of Georgia</td>
<td>370</td>
<td>X</td>
<td>X</td>
<td>TOTAL, DL/I and files</td>
</tr>
<tr>
<td>MANTIS</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>CTNCOM</td>
<td>Various</td>
<td></td>
<td></td>
<td>TOTAL, DL/I and files</td>
</tr>
<tr>
<td>MAPPER</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>Univac</td>
<td>1100</td>
<td>X</td>
<td>X</td>
<td>Its own files</td>
</tr>
<tr>
<td>MARK IV</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Informatics</td>
<td>Various</td>
<td>X</td>
<td>X</td>
<td>DL/I</td>
</tr>
<tr>
<td>MAXIMUS</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Maximus</td>
<td>Data Gen.</td>
<td></td>
<td></td>
<td>Honeywell, 60/66, Eclips</td>
</tr>
<tr>
<td>MDS II</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Honeywell</td>
<td>60/66</td>
<td>X</td>
<td></td>
<td>Its own files</td>
</tr>
<tr>
<td>MICRODATA</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>Microdata</td>
<td>Reality</td>
<td>X</td>
<td></td>
<td>Its own files</td>
</tr>
<tr>
<td>REALITY</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Infopac</td>
<td>370</td>
<td>X</td>
<td></td>
<td>Its own files</td>
</tr>
<tr>
<td>Product</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>Vendor</td>
<td>Machine Type</td>
<td>6</td>
<td>7</td>
<td>Data Base</td>
</tr>
<tr>
<td>-------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>--------------------------------</td>
<td>---------------</td>
<td>---</td>
<td>---</td>
<td>--------------------------</td>
</tr>
<tr>
<td>NCR TOTAL 1</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>NCR</td>
<td>X X</td>
<td></td>
<td></td>
<td>TOTAL</td>
</tr>
<tr>
<td>NOMAD</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>NCSS</td>
<td>X X</td>
<td></td>
<td></td>
<td>Its own relational DB</td>
</tr>
<tr>
<td>OADIS</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>Culliname</td>
<td>Various</td>
<td>X</td>
<td>X</td>
<td>IDMS</td>
</tr>
<tr>
<td>ON-LINE ENGLISH</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Culliname</td>
<td>370</td>
<td>X</td>
<td>X</td>
<td>TDMS and its own files</td>
</tr>
<tr>
<td>ORACLE</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Distribution Man. Systems</td>
<td>PDP/11</td>
<td>X</td>
<td>X</td>
<td>Its own file</td>
</tr>
<tr>
<td>PRO</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BUSINESS FDP</td>
<td></td>
<td>X</td>
<td></td>
<td>Its own files</td>
</tr>
<tr>
<td>SQL SEARCH</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OL Systems</td>
<td>370</td>
<td>X</td>
<td>X</td>
<td>Its own files</td>
</tr>
<tr>
<td>QUERY-BY-EXAMPLE</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TAM</td>
<td>370/4300</td>
<td>X</td>
<td>X</td>
<td>Its own relational DB</td>
</tr>
<tr>
<td>QUERY S/3</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>TAM</td>
<td>S/3</td>
<td>X</td>
<td>X</td>
<td>S/3 files</td>
</tr>
<tr>
<td>JUWICK QUERY</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>CACI</td>
<td>Various</td>
<td>X</td>
<td>X</td>
<td>Its own files</td>
</tr>
<tr>
<td>RAMIS II</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>Mathematica</td>
<td>370</td>
<td>X</td>
<td></td>
<td>Its own DB</td>
</tr>
<tr>
<td>RECALL</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Data Man.</td>
<td>370</td>
<td>X</td>
<td></td>
<td>Its own files</td>
</tr>
<tr>
<td>REQUEST</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>SY. Automation</td>
<td>Various</td>
<td>X</td>
<td></td>
<td>Its own files</td>
</tr>
<tr>
<td>ROROT</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Artificial Intelligence</td>
<td>IBM,</td>
<td></td>
<td></td>
<td>Honeywell</td>
</tr>
<tr>
<td>RPG II</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TAM</td>
<td>Various</td>
<td>X</td>
<td></td>
<td>Conventional files</td>
</tr>
<tr>
<td>Product</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>Vendor</td>
<td>Machine Type</td>
<td>6</td>
<td>7</td>
<td>Data Base</td>
</tr>
<tr>
<td>--------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>----------</td>
<td>--------------</td>
<td>---</td>
<td>---</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>RPG III</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>TB</td>
<td></td>
<td>S/38 X</td>
<td></td>
<td></td>
<td>For S/38 files and DB files</td>
</tr>
<tr>
<td>S/34 UTILITIES</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>TB</td>
<td></td>
<td></td>
<td>S/34 X</td>
<td></td>
<td></td>
<td>Uses S/34 files</td>
</tr>
<tr>
<td>S/38 DRAWS</td>
<td>X</td>
<td>X</td>
<td></td>
<td>TB</td>
<td></td>
<td>Thorne</td>
<td>S/38 X</td>
<td></td>
<td></td>
<td>For S/38 DB Defines its own files</td>
</tr>
<tr>
<td>S/12</td>
<td>X</td>
<td>X</td>
<td></td>
<td>TB</td>
<td></td>
<td></td>
<td>370 X</td>
<td></td>
<td></td>
<td>Its own relational DB</td>
</tr>
<tr>
<td>S/38</td>
<td>X</td>
<td>X</td>
<td></td>
<td>TB</td>
<td></td>
<td></td>
<td>370 X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S/38 Query</td>
<td>X</td>
<td>X</td>
<td></td>
<td>TB</td>
<td></td>
<td></td>
<td>S/38 X</td>
<td></td>
<td></td>
<td>Its own information</td>
</tr>
<tr>
<td>T-AŚK</td>
<td>X</td>
<td>X</td>
<td></td>
<td>Cincom</td>
<td></td>
<td></td>
<td>370 X</td>
<td></td>
<td></td>
<td>TOTAL</td>
</tr>
<tr>
<td>TOS/1700</td>
<td>X</td>
<td>X</td>
<td></td>
<td>Med. Systems</td>
<td>Burroughs</td>
<td>X</td>
<td>Its own files</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UCC/FCS</td>
<td>X</td>
<td>X</td>
<td></td>
<td>U.C.C. Corp.</td>
<td>Various</td>
<td>X</td>
<td>Its own files</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER/11</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Northcounty</td>
<td>PDP/11</td>
<td>X</td>
<td>Uses its own DB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WORK TEN</td>
<td></td>
<td></td>
<td></td>
<td>National Computing Ind.</td>
<td>IBM</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX III

IMPLEMENTATION OF PRESCRIPTIVE METHODOLOGY

This appendix describes a step by step annotated implementation of the proposed methodology which later provided a basis for evaluating the methodology itself.

III.1 Assess Potential Trainee Population

The prospective target group, identified as potentially benefitting from training in a query language was determined to be the Lehigh University Computing Center (LUCC) Operations Manager’s administrative staff. They were already using an inventory application which uses IQL as a basis for much of its reporting.

III.2 Become Familiar with Available Data Bases

The data base available to the trainee population was the aforementioned inventory data base. The author, who was to be the trainer, designed and implemented this inventory system, and was thoroughly knowledgeable of it.

III.3 Identify Products for Training

The product to be used in the training program was IQL since the purpose of training these particular users was to enhance the usefulness and flexibility of an already installed system.

131
III.4 Select and Educate Trainers

The author was selected as the trainer and since the author was the inventory system designer, the basic trainer education was already completed. Nevertheless, the trainer found a refresher on the capabilities of IOL, prior to beginning the training program, was most helpful.

III.5 Develop Basic Method for Presenting Each Product

In order to use IOL, an end-user must first be familiar with the basic features of the DEC 20 TOPS-20 command language. A separate training program was developed to accommodate those trainees who were not familiar with the command language. This training program (Appendix IV) contains an outline of the course, copies of selected slides used during the training, the on-line exercises, and an alphabetic summary of all commands taught in the course. This program was developed to take one half-day of instruction. Its purpose was to get the trainees acquainted with the computer and the command language. A subset of commands was chosen that would allow the user to meaningfully interact with the system.

Trainees who have familiarity with the command language can skip the TOPS-20 training program and begin
with the IQL training program. The IQL training program (Appendix V) contains the session outlines, copies of selected slides to be used during the course, and the on-line exercises. Appendices to the manual include an alphabetic summary of commands, a data dictionary of the inventory data base which was used as a basis for the on-line exercises, and a glossary of terms used in the training program. Its purpose was to provide a reference for the end-users during the training program. A review of the command language was included at the beginning of this program. The IQL training program was divided into two half-day sessions. It was felt that this would give the trainees an opportunity to absorb what was taught the first day and experiment with IQL before learning more complex commands.

The template approach (3.4.1) was deemed appropriate for IQL training, because the use of a template reduces the amount of syntactic information that the trainee must commit to memory. Instead, the trainees are able to reference the template query "IOLSAMPLE" whenever they want to write a query and modify it to suit their present query needs.

At this time, along with developing the training manuals, an evaluation form (Appendix VI) was also developed to be used in the evaluation of the training
Since people generally balk at filling out evaluation forms, this form was structured so that a minimal amount of response was required of the trainees but opportunity was given for elaboration on any point.

The evaluation form served a twofold purpose. Initially, it was used to evaluate the training program per se. Certain pre-judgments had been made by the trainer in designing the training program; such as the number of commands to be taught in a particular session, when to have on-line exercises, etc. Results of these evaluations would indicate whether these judgments were accurate or should be revised.

The PHOTD sessions, coupled with the insights gained from the types of questions asked, and the evaluation forms completed by the trainees provided a good basis for evaluating the training program itself and the methodology used to develop the training program.

In analyzing the training program itself, the author was interested in determining whether IQL itself was user-friendly enough to be used by the casual user. The trainer was also concerned with determining whether the structure of the course was balanced correctly with respect to the number of commands taught in a session. The outline for the training program was based on the premise that, in the beginning, very few commands should
be introduced at one time, whereas later, the trainees should be able to see the similarities between some commands and be able to grasp more in one session.

Last, but not least, training users in the IQL language may provide insights into which characteristics, be they those pertaining to the language itself or those that distinguish this class of user, are most crucial to training a non-technical user to use a query language.

III.6 Compile End-Users Manual, where Applicable

Since only a subset of IQL features were being presented to the trainees, it was felt that it would be advisable to compile an end-users manual including just these commands. DEC's "Interactive Query Language User's Guide" was used as a basis for this manual providing continuity should the user later want to explore the more complex features of IQL and refer to the complete User's Guide. This end-user's manual (Appendix VII) was used in conjunction with the training program.

III.7 Identify Target Group to be Trained

Since it was beyond the scope of this thesis to make this training program available to the entire Lehigh University user community, the target group for training was in effect pre-identified.
III.8 Determine Class Participants

The LUCC Operations Manager was consulted to see if any individuals outside her immediate staff would benefit from being able to ad-hoc query an inventory data base. Her suggestions were followed and the class participants were determined. Appendix VIII contains a profile of the training program participants.

III.9 Schedule Class at Trainees' Convenience

After consulting with all the class participants, the training program was scheduled for three morning sessions. Only two trainees were scheduled for the TOPS-20 Command Language training program.

III.10 Plan Training Exercises around Trainees' Backgrounds

The majority of the trainees were interested in learning IQL, either to supplant an existing inventory system or as an aid in developing a similar inventory system. Only one trainee lacked experience with inventory systems and her purpose in taking the training was to learn about query languages to broaden her professional skills.

The training exercises were, therefore, developed around the existing inventory data base. A training data
base was created which formed a subset of the actual inventory database. It was therefore familiar to the trainees but of a more manageable size for performing the on-line exercises.

These exercises, annotated to explain the purpose of each, along with the expected output are included as Appendix IX. The expected output was used as a basis for scoring the PHOT0 session output automatically generated when the trainees performed the on-line exercises. The number of errors made before getting the exercises correct served as an indication of how well the trainees were absorbing the training material.

III.11 Conduct Training Program

The training program was conducted in a classroom which had terminal facilities in an adjacent room. These facilities were chosen because they provided a well lit room with a blackboard and facility for use of an overhead projector. Having the terminals in an adjacent room made it convenient to perform the on-line exercises and afforded each trainee the opportunity to work at a terminal individually.

The sessions were held very much in accordance with the developed training program. Each trainee was provided with a notebook containing the training program
manual as well as an End-User's Guide. They were free to write in these manuals as they were theirs to keep.

It was originally intended that each trainee would work at his own terminal, but due to a malfunction of one terminal, one trainee (the Operations Manager) worked as a partner alternately with the two members of her staff. This provided the opportunity to query the trainees as to their preference of working alone or with a partner.

Transparencies were made of many of the slides depicted in the training manual. Examples of the commands were demonstrated by using the blackboard as a visual aid.

III.12 Evaluate Training Program

The trainees were asked to fill out an evaluation form (Appendix VI) at the end of the IQL training program. These evaluation forms, along with the PHOTO output were the basis for the evaluation of the methodology and are discussed in Chapter 5.

III.13 Provide Continued Support for Trainees

The background of the trainees was diverse. Since the author's affiliation with the university was a temporary one, having more experienced people in this training program would insure that there would be some
on-site continued support for the non-technical trainees. If these more experienced users saw the benefits of the query facility and began using it to develop systems, resident experts would emerge, thus providing continued support after the author-trainee had left the university. In the interim, the trainees were assured that they should not hesitate to call if they had any problems or questions concerning their use of IQL.

III.14 Determine Whether All Target Groups Have Been Trained

Since it was the intention of the author to only conduct one training program, all target groups had therefore been trained.

III.15 Summary

This completed the implementation of the methodology. The evaluation itself is discussed in Chapter 5.
APPENDIX IV

TOPS-20 TRAINING PROGRAM
TOPS-20 COMMAND LANGUAGE

TRAINING PROGRAM
This training program is modeled after the training program developed by National CSS, Inc. for introducing new users to the National CSS computer system [NCSS 79].
# Table of Contents

## 1. SESSION OUTLINE

1.1 On-line Exercise 1  
1.2 On-line Exercise 2  
1.3 On-line Exercise 3

## I. SUMMARY OF COMMANDS

I.1 Tops-20 Commands  
I.2 Edit Commands  
I.3 Special Characters

## II. GLOSSARY
SESSION OUTLINE

1. Features of computer
2. How to access computer
   a. LOGIN
   b. LOGOUT
   c. Special keys
      i. DEL
      ii. "C
      iii. "U
      iv. "H
3. On-line Help Facilities
   a. ?
   b. $
   c. HELP
4. Concept of environments
5. Creating Files
   a. File naming conventions
   b. EDIT
      i. Edit Positioning commands
         1. (.
         2. p
      ii. Edit Exit commands
         1. e
         2. eu
         3. eq
6. On-line exercise 1
7. Modifying files:
   a. Edit Information changing commands
      i. d
      ii. f
      iii. i
      iv. r
      v. s
8. On-line exercise 2
9. Managing Files:
   a. TYPE
   b. INFORMATION DISK
   c. INFORMATION OUTPUT
   d. DIRECTORY
   e. Manipulating files:
      i. RENAME
      ii. COPY
      iii. DELETE/UNDELETE
      iv. PRINT
      v. EXPUNGE
10. Interruptions:
    a. ATTACH
    b. "Q
    c. "T
d. ^C ^C
e. ^D
f. ^R
11. Command Files
12. On-line Exercise 3
13. Review
14. Evaluation
COMPUTER SYSTEM HAS FACILITIES FOR:

ENTERING INFORMATION INTO SYSTEM
STORING INFORMATION
PROCESSING INFORMATION
DISPLAYING RESULTS OF PROCESSING
ENTER INFORMATION - INPUT

PROCESS

GET RESULTS - OUTPUT

STORAGE
PROCEDURE FOR ACCESSING THE SYSTEM:

1. Turn on power for terminal and coupler.

2. Be sure LOCAL/REMOTE switch on your terminal is in REMOTE position.

3. Be sure DUPLEX switch is on FULL.


5. When you hear a high-pitched tone, place receiver in coupler. An indicator on coupler lights up to verify that you telephone call was successfully completed.

6. Hit the RETURN key.

7. In response to

LEHIGH UNIVERSITY, TOPS-20 Monitor

@

Type LOGIN userid

8. The system will prompt you for the password. Enter password, which for security purposes, will not be visible on screen.

9. The system responds with

@ This symbol indicates that you are in the TOPS-20 operating system environment.

10. When your work is completed, you notify the system that you have finished by typing

@LOGOUT

11. Hang up the telephone and turn off the power for both your terminal and the coupler.
DEL ÒR RUBOUT
CTRL U
CTRL C
CTRL H
TOPS-20 Environment

Edit Environment

Input Environment
File not found, Creating new file
0100

$
EDIT filename.filetype
PRINT COMMANDS

p.       prints current line
p*       prints first line
p*       prints last line
pn;m     prints beginning line # to ending line #
s        prints previous line (pressing the 'Escape' key)
linefeed prints next line (pressing the 'Linefeed' key)
ON-LINE EXERCISE 1

Creating a new file

1. Log in, and when you are connected to the system, create a new file called DEPT.DAT. Include the following items in this file:

   JOHNS JOSEPH   INDUSTRIAL ENGINEERING 58
   WILLIAMS SUSAN MECHANICAL ENGINEERING 20
   MICHAELS JAMES MATHEMATICS 35
   WILSON ANDREW ENGLISH 19
   DAVIS MATTHEW PHYSICS 21
   ROGERS DENNIS ELECTRICAL ENGINEERING 15

2. After the last line of input, return to the Edit environment. Verify by printing all the lines.

3. Store the file DEPT.DAT on your disk without line numbers.

4. Logout.
Input Environment
File not found, Creating new file
0100
## EDIT COMMANDS

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>d</code></td>
<td>delete</td>
</tr>
<tr>
<td><code>i</code></td>
<td>insert</td>
</tr>
<tr>
<td><code>r</code></td>
<td>replace</td>
</tr>
<tr>
<td><code>f</code></td>
<td>find</td>
</tr>
<tr>
<td><code>s</code></td>
<td>substitute</td>
</tr>
</tbody>
</table>

- delete: `dnim`
- insert: `in,increment`
- replace: `rn,increment`
- find: `fstring$
- substitute: `soldstring$newstring$nim`
ON-LINE EXERCISE 2

Correcting and Modifying the File

1. Bring the DEPT.DAT file into the editor.

2. For practice in positioning yourself in a file:
   a. Print the first line of the file.
   b. Print lines 200-300.
   c. Print the last line of the file.
   d. Print line 500.
   e. Print the previous line.
   f. Print the next line.
   g. Print the current line.

3. If you have any typing errors to correct, make the desired changes.

4. Make the following revisions to the file:
   a. Change Susan William's department to CHEMISTRY.
   b. Replace the line for JAMES MICHAEL'S with the following:
      SMITH DAVID          PHILOSOPHY
      61
   c. Change the spelling of DENNIS ROGER'S first name to DENTIS.
   d. Delete line 300.
   e. Insert the following line as the first line in your file:
      GOLD DAVID          COMPUTER SCIENCE
      58
   f. Print from the beginning of the file to line 400.
   g. Change all buildings numbered 58 to 38.
h. Add the following lines immediately after the line for SUSAN WILLIAMS:

STONE MARY        HISTORY  35
BERNHARD CLARE, FRENCH  22
ADAMS, DIANE, MATHEMATICS  35

5. Verify all revisions by printing the entire file.

6. Store the file and leave the Edit environment.

7. Logout.
TOPS-20 Environment
@ Edit

Edit Environment
* print insert find
delete replace substitute

Input Environment
File not found, Creating new file
0100

$
TOPS-20 COMMANDS

TYPE filename.filetype
PRINT filename.filetype/options
COPY oldfilename.oldfiletype newfilename.newfiletype
RENAME oldfilename.oldfiletype newfilename.newfiletype
DELETE filename.filetype
UNDELETE filename.filetype
EXPUNGE
DIRECTORY
INFORMATION DISK
INFORMATION OUTPUT
ON-LINE EXERCISE 3

Managing Files

1. Log in, and when you are connected to the system, create a new file BLDG.SEQ containing the following information:

005 WHITAKER LABORATORY
006 MUDD BUILDING
007 SINCLAIR LABORATORY
014 CHRISTMAS-SAUCON HALL
019 PACKARD LABORATORY

2. List all the files in your directory.

3. List the contents of BLDG.SEQ on your terminal screen.

4. Change the name of DEPT.DAT to DEPT.LOG.

5. Print the file DEPT.LOG on the printer.

6. Check to see if it is printing or if it is waiting to be printed.

7. Make a duplicate file of BLDG.SEQ and call it BLDG2.SEQ.

8. Delete BLDG.SEQ from your directory.

9. OOPS, I told you to delete the wrong file; get it back and delete BLDG2.SEQ instead.

10. List the files that are now in your directory.

11. See how much space you have left in your directory.

12. Free up the space for files you have deleted.

13. Check again to see how much space you have left.

14. Logout.

162
TOPS-20 Environment

Edit  Copy  Delete  Directory
Type   Rename  Undelete  Information
Print  

Edit Environment
* print insert find
delete replace substitute

Input Environment
File not found, Creating new file
0100

163
ENCLOSURE I

SUMMARY OF COMMANDS
TOPS-20 COMMANDS

ATTACH
Reconnects system following certain system crashes or when telephone connection is interrupted.

COPY
Create a duplicate file.

DELETE
Delete file from directory.

DIRECTORY
Lists files stored in directory.

EDIT
Transfers control to EDIT environment to create or modify files.

EXPUNGE
Frees up space held by deleted files.

INFORMATION DISK
Determines amount of space left in directory.

INFORMATION OUTPUT
Determines status of printer queue.

LOGIN
Gains access to system.

LOGOUT
Ends access to system.

PRINT
Prints a file on the off-line printer.
Options:
/DELETE - deletes file after printing
/FORMS:UNLINE - prints on unlined paper

RENAME
Changes filename and/or filetype of file.

TYPE
Displays contents of file at terminal.

UNDELETE
Recovers a file that has been deleted but not expunged.
EDIT COMMANDS

**d**
Delete one or more lines from the file.
\[\text{dn:m}\]

**f**
Find the occurrence of a string of characters within the foldstrings

**i**
Insert one or more lines into a file.
\[\text{in}, l\text{in}!\#\text{ of lines to insert}\]

**.**
Move pointer to specified line.
\[\text{.n}\]

**r**
Replace one or more lines in the file by deleting and inserting one or more lines.
\[\text{rn:m}, l\]

**p**
Move pointer to specified line and print it.
\[\text{pn:m}\]

**s**
Substitute a given string of characters for another string within one or more lines in the file.
\[\text{soldstrings$newstring$nn:m}\]

**eu**
Exit from Edit and save file unnumbered.

**e**
Exit from Edit and save file with line numbers.

**eq**
Exit from Edit and quit without saving changes.
**SPECIAL CHARACTERS**

TOPS-20 operating system for the DEC-20 recognizes certain special characters as commands to perform certain functions.

<table>
<thead>
<tr>
<th>Code</th>
<th>Function Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTRL/C</td>
<td>Suspend execution of program currently running.</td>
</tr>
<tr>
<td>CTRL/U</td>
<td>Delete the current line.</td>
</tr>
<tr>
<td>DEL or RUBOUT</td>
<td>Delete the last character entered.</td>
</tr>
<tr>
<td>CTRL/R</td>
<td>Redisplay current line being entered.</td>
</tr>
<tr>
<td>CTRL/O</td>
<td>Suppress displaying output of currently executing program.</td>
</tr>
<tr>
<td>CTRL/S</td>
<td>Stop displaying of output at terminal if in PAGE mode.</td>
</tr>
<tr>
<td>CTRL/Q</td>
<td>Continue displaying of output at terminal if in PAGE mode.</td>
</tr>
<tr>
<td>CTRL/T</td>
<td>Display current status of the user.</td>
</tr>
<tr>
<td>CTRL/H</td>
<td>When an error is made in entering a command, redisplay command line up to erroneous field.</td>
</tr>
</tbody>
</table>

167
**GLOSSARY**

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Terminal</strong></td>
<td>Communications device used to control user's operations to send and receive information from the computer.</td>
</tr>
<tr>
<td><strong>Coupler</strong></td>
<td>Communications device that transmits signals from the terminal to the computer over telephone lines.</td>
</tr>
<tr>
<td><strong>CPU</strong></td>
<td>Processing component of computer.</td>
</tr>
<tr>
<td><strong>Login</strong></td>
<td>Procedures that establishes communication with the computer.</td>
</tr>
<tr>
<td><strong>Environments</strong></td>
<td>Portion of computer that has control.</td>
</tr>
<tr>
<td><strong>Command</strong></td>
<td>Computer language instruction to perform a task.</td>
</tr>
<tr>
<td><strong>Delimiter</strong></td>
<td>Character placed before or after one or more characters in order to set them apart as a group, for example a slash (/).</td>
</tr>
<tr>
<td><strong>File</strong></td>
<td>Collection of records or lines processed as a unit.</td>
</tr>
<tr>
<td><strong>Filename</strong></td>
<td>Descriptive name used as an identifier (limited to 6 characters)</td>
</tr>
<tr>
<td><strong>Filetype</strong></td>
<td>Descriptive name used as a second identifier (limited to 3 characters)</td>
</tr>
<tr>
<td><strong>Record</strong></td>
<td>One item or line of input.</td>
</tr>
<tr>
<td><strong>String</strong></td>
<td>Series of one or more characters.</td>
</tr>
<tr>
<td><strong>Off-line Device</strong></td>
<td>Device that is accessed interactively by the system as it is available.</td>
</tr>
<tr>
<td><strong>Dictionary</strong></td>
<td>Table stored by IOL that describes a data file or data base.</td>
</tr>
</tbody>
</table>
APPENDIX V

IQL TRAINING PROGRAM
IQL TRAINING PROGRAM
The concept of environments as depicted in this training program was adapted from National CSS, Inc. training program literature [NCSS 79].
# Table of Contents

1. IQL SESSION OUTLINE - DAY 1 173
   1.1 On-line Exercise 1 181
   1.2 On-line Exercise 2 186
   1.3 On-line Exercise 3 190
   1.4 On-line Exercise 4 193

2. IQL SESSION OUTLINE - DAY 2 195
   2.1 On-line Exercise 1 199
   2.2 On-line Exercise 2 202

I COMMANDS 204
   I.1 IQL Query Statements 205
   I.2 IQL Assistance Commands 207
   I.3 TOPS-20 Commands 208
   I.4 Edit Commands 209
   I.5 Special Characters 210

II DATA DICTIONARY 211
   II.1 Building 212
   II.2 Customer 213
   II.3 Equipment 214
   II.4 Model 217
   II.5 Vendor 218

III Glossary 219
1. Explanation and purpose of IQL.
2. Review command language.
4. Relational database concepts.
5. Introduce commands:
   a. IQL
   b. EXIT
   c. DICTIONARIES
   d. ITEMS
6. On-line exercise 1
7. Introduce new commands:
   a. WRITE
   b. EDIT
   c. OPEN
   d. PRINT
   e. HEADING
   f. RUN
8. On-line exercise 2
9. Review
10. Introduce new commands:
    a. IF
    b. SORT
    c. STORE
    d. REPLACE
    e. DELETE
    f. QUERIES
11. On-line exercise 3
12. Review
13. Introduce new commands:
    a. OPEN (more than one dictionary)
    b. FIND KEYn
14. On-line exercise 4
15. Review
<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QCjobS.TMP</td>
<td>Current query area</td>
</tr>
<tr>
<td>QCjobS.QMP</td>
<td>Previous query area</td>
</tr>
<tr>
<td>QCjobE.LPT</td>
<td>Print formatted report</td>
</tr>
<tr>
<td>name.INQ</td>
<td>Analyzed query</td>
</tr>
<tr>
<td>OPQRYS.SEQ</td>
<td>One or more non-analyzed queries</td>
</tr>
<tr>
<td>ISAMF6.IDX</td>
<td>Dummy file required by IQL</td>
</tr>
<tr>
<td>ISAMF7.IDX</td>
<td>Dummy file required by IQL</td>
</tr>
<tr>
<td>QPDICT.SEQ</td>
<td>One or more dictionaries</td>
</tr>
</tbody>
</table>
TOPS-20 Environment

@ Edit  Copy  Delete  Directory
Type  Rename  Undelete  Information
Print  

Edit Environment
**print  insert  find  
delete  replace  substitute

Input Environment
File not found, Creating new file 0100

$
IQL

EXIT

DICTIONARIES

ITEMS
Dictionaries in your directory:

<table>
<thead>
<tr>
<th>Dict</th>
<th>File</th>
<th>File-in</th>
<th>Rec</th>
<th>Blk</th>
<th>Key</th>
<th>KY</th>
<th>RD</th>
<th>CP</th>
<th>RW</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUILDING</td>
<td>IS</td>
<td>Dsk7</td>
<td>BLDG</td>
<td>TDX</td>
<td>29</td>
<td>10</td>
<td>1</td>
<td>3</td>
<td>NU</td>
</tr>
<tr>
<td>VENDOR</td>
<td>IS</td>
<td>Dsk7</td>
<td>VENDOR</td>
<td>TDX</td>
<td>29</td>
<td>10</td>
<td>1</td>
<td>3</td>
<td>AU</td>
</tr>
<tr>
<td>EQUIPMENT</td>
<td>IS</td>
<td>Dsk7</td>
<td>EQUIP</td>
<td>TDX</td>
<td>112</td>
<td>10</td>
<td>1</td>
<td>15</td>
<td>AU</td>
</tr>
<tr>
<td>CUSTOMER</td>
<td>IS</td>
<td>Dsk7</td>
<td>CUST</td>
<td>TDX</td>
<td>52</td>
<td>10</td>
<td>1</td>
<td>4</td>
<td>NU</td>
</tr>
<tr>
<td>MODEL</td>
<td>IS</td>
<td>Dsk7</td>
<td>MODEL</td>
<td>TDX</td>
<td>53</td>
<td>10</td>
<td>1</td>
<td>5</td>
<td>AS</td>
</tr>
</tbody>
</table>

(End list of dictionaries)
<QA>items customer

<table>
<thead>
<tr>
<th>Dict</th>
<th>File</th>
<th>File-in</th>
<th>Rec</th>
<th>Blk</th>
<th>Key</th>
<th>KY</th>
<th>RD</th>
<th>CP</th>
<th>PW</th>
<th>PW</th>
<th>PW</th>
<th>PW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Type</td>
<td>Name</td>
<td>Direct</td>
<td>Len</td>
<td>Fac</td>
<td>Loc</td>
<td>LN</td>
<td>TP</td>
<td>PW</td>
<td>PW</td>
<td>PW</td>
<td>PW</td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td>---------</td>
<td>--------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>CUSTOMER IS Dsk7 CUST IDX</td>
<td>52 10 1 4 NU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>Top</th>
<th>Bottom</th>
<th>1st</th>
<th>#</th>
<th>Y</th>
<th>C</th>
<th>Picture</th>
<th>Scan</th>
</tr>
</thead>
<tbody>
<tr>
<td>DD CUST-FIRST-NAME</td>
<td>FIRST NAME</td>
<td>5 10 A 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD CUST-LAST-NAME</td>
<td>LAST NAME</td>
<td>15 15 A 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD CUST-REC</td>
<td>LAST NAME</td>
<td>1 52 A 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD CUST-BLDG-NO</td>
<td>BLDG</td>
<td>30 3 N 0 999</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD CUST-NO</td>
<td>NO</td>
<td>1 4 N 0 999</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD CUST-ADDR</td>
<td>ADDRESS</td>
<td>30 7 A 0 XXX-XXXX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD CUST-ROOM</td>
<td>ROOM</td>
<td>33 4 A 0 XXX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD CUST-DEPT</td>
<td>DEPARTMENT</td>
<td>37 15 A 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD CUST-NAME</td>
<td>NAME</td>
<td>5 25 A 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(End list of items)
ON-LINE EXERCISE 1

1. Enter the IQL environment.

2. Find out what dictionaries are available to your directory.

3. Find out what data items are in the Building dictionary.

4. Leave the IQL environment.

5. Logout.
FOUR STEPS OF REPORT WRITING

1. Write down the request for the report.

2. Is the necessary data available? Find out in which dictionaries the data is located.

3. Sketch the physical layout of the report.

4. Translate the English request into IQL statements [NCSS 78].
BASIC REPORT STATEMENTS

OPEN
PRINT
HEADING
IF
SORT
ON-LINE EXERCISE 2

1. Enter the IQL environment.

2. Create a query that will print the contents of the Building dictionary.

3. Generate this report using the RUN command.

4. Find out what data items are in the Customer dictionary.

5. Print the names and departments of all customers.

6. Leave the IQL environment.

7. Print the reports IQL created on the printer using the delete option.

8. Logout.
**IQLSAMPLE**

**HEADING** "\* (What title do you want for this report?).

**OPEN** -------- (What tables are required to satisfy this request?).

**IF** -------- (What selection criteria, if any?).

**SORT BY** ------- (What sequence?).

**PRINT** ------- (What data items do you want to see?).
NAMING AND STORING A QUERY

STORE
RESTORE
DELETE
QUERIES
ON-LINE EXERCISE 3

1. Enter the IQL environment.

2. Generate a listing of all customers in sequence by last name and then first name. Along with their names, print their department and building number.

3. Modify this query so that it also prints the customer’s room no.

4. Run the modified query.

5. Store this query as a non-analyzed query called CUSTLIST.

6. Generate a query that will list the equipment number, customer number, building number and room of all ‘DEC’ (Vendor-No of ‘DEC’) equipment.

7. Store this query as DECLIST.

8. Change the CUSTLIST query so that it prints only the customers’ first and last names.

9. Make these revisions permanent.

10. List all the queries available to your directory.

11. Delete the CUSTLIST query.

12. Leave the IQL environment.

13. Delete the print files created during this session.

14. Logout
LOGIN ATTACH

TOPS-20 Environment

IQL Environment

IQL Environment

<QA> WRITE STORE DICTIONARIES
EDIT REPLACE ITEMS
RUN DELETE QUERIES

Edit Environment

Input Environment

0100 $
POOLING INFORMATION FROM DIFFERENT DICTIONARIES

OPEN DICTIONARY1 DICTIONARY2 DICTIONARY3

FIND KEY2 =

FIND KEY3 =
ON-LINE EXERCISE 4

1. After logging into the system, enter the IQL environment.

2. Modify the DECLIST query to print the owner’s name as well as his customer number. (This will require having two dictionaries OPEN at one time).

3. Replace the non-analyzed query DECLIST with this modified query.

4. Run the modified query without listing it.

5. Create and store a query that will list all "TUC" equipment that is under maintenance contract (Equip-Service = 'M') giving the vendor number, model name serial no, customer number and customer name. Call this query TUCLIST.

6. Modify this query to be in sequence by the vendor number and descending model name.

7. Make the change permanent.

8. Run the query.

9. Leave the IQL environment.

10. Logout.
IQL SESSION OUTLINE - DAY 2

1. Review previous day's commands
2. Introduce function commands:
   a. TALLY
   b. TOTAL
   c. MAXIMUM
   d. MINIMUM
   e. AVERAGE
   f. SAVE
   g. EXECUTE
3. Online exercise
4. Review
5. Introduce Formatting commands:
   a. DATE
   b. HSPACE
   c. VSPACE
   d. DISPLAY
   e. LMARGIN
   f. RMARGIN
6. Online exercise
7. Review
8. Evaluation
REPORT FUNCTIONS

TALLY
TOTAL
MAXIMUM
MINIMUM
AVERAGE
DIFFERENCE BETWEEN RUN AND EXECUTE

IQL Environment
<QA> RUN

(only if error in query)

Analysis Level
(Analyzes statements in nonanalyzed query)

(automatic)

EXECUTE

(automatic)

Execution Level
(Follows statement analysis; passes one or more data files to prepare reports)
ON-LINE EXERCISE 1

1. After gaining access to the system, enter the IQL environment.

2. List all the queries available to your directory.

3. Find out how many 'DEC01' pieces of equipment there are.

4. What is the highest customer number that has been used?

5. In order to save computer time, issue the command that will analyze the TUCLIST query.

6. Generate the report using the command that will use the analyzed query.

7. What is the total monthly maintenance charge for LUCC owned equipment (Equip-Cust-No = 1) that is under maintenance (Equip-Service = 'W')?

8. Keep this query in the file of non-analyzed queries as LUCCRATE.

9. How many pieces of equipment are there in the equipment table of each model type?

10. Leave the IQL environment.

11. List all the files in your directory with a filetype of .INQ.

12. Delete all unneeded print files.

13. Logout
FORMATTING STATEMENTS

LMARGIN   HSPACE   DATE
RMARGIN   VSPACE   DISPLAY
ON-LINE EXERCISE 2

1. After gaining access to the system, enter the IQL environment.

2. Modify the TUCLIST query to create only the print file and not display on the terminal.

3. Make this change permanent.

4. Create a report called 'Vendor Listing' that lists the number and name of all vendors. Double space this report and leave 10 spaces between columns. Start the report in column 5.

5. Find the total maintenance rate for each customer.

6. Leave the IQL environment.

7. Logout.
LOGOUT

ATTACH

TOPS-20 Environment
@IQL

IQL Environment
<QA> WRITE STORE DICTIONARIES SAVE EXECUTE
EDIT REPLACE ITEMS EXECUTE
RUN DELETE QUERIES

Edit Environment
EU

Input Environment
0100 $
ENCLOSURE I.

COMMANDS
<table>
<thead>
<tr>
<th>IQL QUERY STATEMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AVERAGE</strong></td>
</tr>
<tr>
<td><strong>DATE</strong></td>
</tr>
<tr>
<td><strong>DISPLAY</strong></td>
</tr>
<tr>
<td><strong>FIND KEY</strong></td>
</tr>
<tr>
<td><strong>HEADING</strong></td>
</tr>
<tr>
<td><strong>HSPACE</strong></td>
</tr>
<tr>
<td><strong>IF</strong></td>
</tr>
<tr>
<td><strong>LMARGIN</strong></td>
</tr>
<tr>
<td><strong>MAXIMUM</strong></td>
</tr>
<tr>
<td><strong>MINIMUM</strong></td>
</tr>
<tr>
<td><strong>OPEN</strong></td>
</tr>
<tr>
<td><strong>PRINT</strong></td>
</tr>
<tr>
<td><strong>RMARGIN</strong></td>
</tr>
<tr>
<td><strong>SORT</strong></td>
</tr>
<tr>
<td><strong>TALLY</strong></td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
</tr>
</tbody>
</table>
VSPACE Changes the vertical spacing of reports
IQL ASSISTANCE COMMANDS

DELETE
Removes stored nonanalyzed query(s) from the file of stored queries.

DICTIONARIES
Provides a terminal display of the file information for each dictionary in the directory.

EDIT
Transfers control to the edit environment to change either the current query area or a stored nonanalyzed query.

EXECUTE
Generates a report from an analyzed query file that have been saved using the SAVE command.

EXIT
Returns control from the IQL environment to the TOPS-20 operating system.

ITEMS
Formats and displays the contents of a dictionary.

QUERY
Displays a list of all nonanalyzed queries stored by IQL in the file of stored queries.

REPLACE
Deletes the named query from the file of stored queries and writes the contents of the current query area into the file of stored queries.

RUN
Processes a nonanalyzed query and generates a report.

SAVE
Analyzes and saves in an analyzed query file a stored nonanalyzed query. You can subsequently generate a report from an analyzed query file with the EXECUTE command.

STORE
Stores the current query area into the file of nonanalyzed queries.

WRITE
Allows you to begin writing a new query in the current query area.
TOPS-20 COMMANDS

ATTACH
  Reconnects system following certain system crashes or when telephone connection is interrupted.

COPY
  Create a duplicate file.

DELETE
  Delete file from directory.

DIRECTORY
  Lists files stored in directory.

EDIT
  Transfers control to EDIT environment to create or modify files.

EXPUNGE
  Frees up space held by deleted files.

INFORMATION DISK
  Determines amount of space left in directory.

INFORMATION OUTPUT
  Determines status of printer queue.

LOGIN
  Gains access to system.

LOGOUT
  Ends access to system.

PRINT
  Prints a file on the off-line printer.
  Options:
  /DELETE - deletes file after printing
  /FORMS:UNLINE - prints on unlined paper

RENAME
  Changes filename and/or filetype of file.

TYPE
  Displays contents of file at terminal.

UNDELETE
  Recovers a file that has been deleted but not expunged.
EDIT COMMANDS

d  Delete one or more lines from the file.
   d n:m

f  Find the occurrence of a string of characters within the file.
   foldstrings

i  Insert one or more lines into a file.
   i n,m

.  Move pointer to specified line.
   .n

r  Replace one or more lines in the file by deleting and inserting one or more lines.
   r n:m,i

p  Move pointer to specified line and print it.
   p n:m

s  Substitute a given string of characters for another string within one or more lines in the file.
   s oldstrings$ newstrings$n:m

eu  Exit from Edit and save file unnumbered.

e  Exit from Edit and save file with line numbers.

eq  Exit from Edit and quit without saving changes.
SPECIAL CHARACTERS

TOPS-20 operating system for the DEC-20 recognizes certain special characters as commands to perform certain functions.

CTRL/C Suspend execution of program currently running.
CTRL/U Delete the current line.
DEL or RUBOUT Delete the last character entered.
CTRL/R Redisplay current line being entered.
CTRL/D Suppress displaying output of currently executing program.
CTRL/S Stop displaying of output at terminal if in PAGE mode.
CTRL/Q Continue displaying of output at terminal if in PAGE mode.
CTRL/T Display current status of the user.
CTRL/H When an error is made in entering a command, redisplays command line up to erroneous field.
ENCLOSURE II

DATA DICTIONARY
### BUILDING DICTIONARY

<table>
<thead>
<tr>
<th>Item</th>
<th>Top Title</th>
<th>Bottom Title</th>
<th>1st #</th>
<th>T S Printing</th>
</tr>
</thead>
<tbody>
<tr>
<td>DD*BLDG-REC</td>
<td></td>
<td></td>
<td>1</td>
<td>29 A 0</td>
</tr>
<tr>
<td>DD BLDG-NO   BLDG</td>
<td>NO</td>
<td></td>
<td>1</td>
<td>3 N 0 999</td>
</tr>
<tr>
<td>DD BLDG-NAME</td>
<td>BLDG</td>
<td>NAME</td>
<td>4</td>
<td>25 A 0</td>
</tr>
</tbody>
</table>
CUSTOMER DICTIONARY

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cust-Addr</td>
<td>Customer's address comprised of Bldg-No and Room</td>
</tr>
<tr>
<td>Cust-Bldg-No</td>
<td>Link to Building Dictionary Building no of customer's address</td>
</tr>
<tr>
<td>Cust-Dept</td>
<td>Customer's department</td>
</tr>
<tr>
<td>Cust-First-Name</td>
<td>Customer's first name</td>
</tr>
<tr>
<td>Cust-Last-Name</td>
<td>Customer's last name</td>
</tr>
<tr>
<td>Cust-Name</td>
<td>Customer's first and last name with blanks in between</td>
</tr>
<tr>
<td>Cust-No</td>
<td>Unique key of Customer Dictionary</td>
</tr>
<tr>
<td>Cust-Rec</td>
<td>Represents entire Customer Dictionary</td>
</tr>
<tr>
<td>Cust-Room</td>
<td>Room number of customer's address</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item ID</th>
<th>Name</th>
<th>Top Title</th>
<th>Bottom Title</th>
<th>1st #</th>
<th>T S</th>
<th>Printing</th>
</tr>
</thead>
<tbody>
<tr>
<td>DD</td>
<td>CUST-FIRST-NAME</td>
<td>FIRST NAME</td>
<td>5 10 A 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD</td>
<td>CUST-LAST-NAME</td>
<td>LAST NAME</td>
<td>15 15 A 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD*</td>
<td>CUST-REC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD</td>
<td>CUST-BLDG-NO</td>
<td>BLDG</td>
<td>1 52 A 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD</td>
<td>CUST-NO</td>
<td>CUST NO</td>
<td>30 3 N 0 999</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD</td>
<td>CUST-ADDR</td>
<td>ADDRESS</td>
<td>30 7 A 0 XXX-XXXX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD</td>
<td>CUST-ROOM</td>
<td>ROOM</td>
<td>33 4 A 0 XXXX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD</td>
<td>CUST-DEPT</td>
<td>DEPARTMENT</td>
<td>37 15 A 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD</td>
<td>CUST-NAME</td>
<td>NAME</td>
<td>5 25 A 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

213
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equip-Account-No</td>
<td>Chargeable account-no in format XXXXXX-XXX</td>
</tr>
<tr>
<td>Equip-Acct-Posi</td>
<td>First position of Account-No</td>
</tr>
<tr>
<td>Equip-Bldg-No</td>
<td>Link to Building Dictionary Building where equipment housed</td>
</tr>
<tr>
<td>Equip-Cust-No</td>
<td>Link to Customer Dictionary Cust-No of owner of this piece of equipment</td>
</tr>
<tr>
<td>Equip-Date Recvd</td>
<td>Date equipment received (YYMMDD)</td>
</tr>
<tr>
<td>Equip-Date-Recvd-MMDD</td>
<td>MMDD portion of date received. Used in conjunction with Equip-Date-Recvd-YY to print a date in format MMDDYY. (Print Equip-Date-Recvd-MMDD, 0, Equip-Date-Recvd-YY)</td>
</tr>
<tr>
<td>Equip-Date-Recvd-YY</td>
<td>YY portion of date received. Used in conjunction with Equip-Date-Recvd-MMDD to print a date in format MMDDYY. (Print Equip-Date-Recvd-MMDD, 0, Equip-Date-Recvd-YY)</td>
</tr>
<tr>
<td>Equip-Location</td>
<td>Location of equipment; comprised of Bldg-No, Room, and Position.</td>
</tr>
<tr>
<td>Equip-Maint-Vndr</td>
<td>Code of vendor servicing this piece of equipment</td>
</tr>
<tr>
<td>Equip-Model-No</td>
<td>Link to Model Dictionary</td>
</tr>
<tr>
<td>Equip-No</td>
<td>Unique key of Equipment Dictionary</td>
</tr>
<tr>
<td>Equip-Pos</td>
<td>Position in room where equipment located</td>
</tr>
<tr>
<td>Equip-Options</td>
<td>Options applicable to this piece of equipment</td>
</tr>
<tr>
<td>Equip-Price</td>
<td>Original cost of equipment in whole dollars</td>
</tr>
</tbody>
</table>
**Equip-Rec**

Represents entire Equipment Dictionary
Not used in writing queries

**Equip-Room**

Room in which equipment housed

**Equip-Serial-No**

Manufacturer's serial no

**Equip-Service**

Code designating service:
- W Warranty
- M Maintenance Contract
- X No Service

**Equip-Status**

Subdivision of LUCC equipment
All customer owned equipment has a Status of 'C'
LUCC equipment is subdivided as follows:
- R Rental
- S Spare
- A Admin
- L All remaining LUCC equipment

**Equip-Svc-End-Date**

Ending date of service (YYMMDD)

**Equip-Svc-End-Date-MMDD**

MMDD portion of Service End Date.
Used in conjunction with Equip-Svc-End-Date-YY to print a date in the format MMDDYY. (Print Equip-Svc-End-Date-MMDD, 0, Equip-Svc-End-Date-YY)

**Equip-Svc-End-Date-YY**

YY portion of Service End Date. Used in conjunction with Equip-Svc-End-Date-MMDD to print a date in the format MMDDYY. (Print Equip-Svc-End-Date-MMDD, 0, Equip-Svc-End-Date-YY)

**Equip-Vendor-No**

Link to Vendor Dictionary
Manufacturer of this piece of equipment
<table>
<thead>
<tr>
<th>Item ID</th>
<th>Name</th>
<th>Top Title</th>
<th>Bottom Title</th>
<th>1st #</th>
<th>TS</th>
<th>Printing</th>
</tr>
</thead>
<tbody>
<tr>
<td>DO</td>
<td>EQUIP-REC</td>
<td></td>
<td></td>
<td>1</td>
<td>112</td>
<td>A 0</td>
</tr>
<tr>
<td>DO</td>
<td>EQUIP-BLDG-NO</td>
<td>BLDG NO</td>
<td></td>
<td>40</td>
<td>3</td>
<td>N 0 99</td>
</tr>
<tr>
<td>DO</td>
<td>EQUIP-SERIAL-NO</td>
<td>SERIAL NO</td>
<td></td>
<td>6</td>
<td>10</td>
<td>A 0</td>
</tr>
<tr>
<td>DO</td>
<td>EQUIP-SVC-END-DATE-YY</td>
<td></td>
<td></td>
<td>22</td>
<td>2</td>
<td>N 0 99</td>
</tr>
<tr>
<td>DO</td>
<td>EQUIP-SVC-END-DATE-MMDD</td>
<td>SVC END DATE</td>
<td></td>
<td>24</td>
<td>4</td>
<td>N 0 99/99/</td>
</tr>
<tr>
<td>DO</td>
<td>EQUIP-DATE-RECV-YY</td>
<td>DATE RECV</td>
<td></td>
<td>28</td>
<td>2</td>
<td>N 0 99</td>
</tr>
<tr>
<td>DO</td>
<td>EQUIP-DATE-RECV-MMDD</td>
<td>DATE RECV</td>
<td></td>
<td>30</td>
<td>4</td>
<td>N 0 99/99/</td>
</tr>
<tr>
<td>DO</td>
<td>EQUIPMENT NO</td>
<td></td>
<td></td>
<td>1</td>
<td>15</td>
<td>A 0</td>
</tr>
<tr>
<td>DO</td>
<td>MODEL NO</td>
<td>MODEL</td>
<td></td>
<td>1</td>
<td>5</td>
<td>A 0</td>
</tr>
<tr>
<td>DO</td>
<td>VENDOR NO</td>
<td>VENDOR</td>
<td></td>
<td>1</td>
<td>3</td>
<td>A 0</td>
</tr>
<tr>
<td>DO</td>
<td>CUST NO</td>
<td>CUST</td>
<td></td>
<td>16</td>
<td>4</td>
<td>N 0 9999</td>
</tr>
<tr>
<td>DO</td>
<td>STATUS</td>
<td>STATUS</td>
<td></td>
<td>20</td>
<td>1</td>
<td>A 0 X</td>
</tr>
<tr>
<td>DO</td>
<td>SERVICE</td>
<td>SERVICE</td>
<td></td>
<td>21</td>
<td>1</td>
<td>A 0 X</td>
</tr>
<tr>
<td>DO</td>
<td>SVC-END DATE</td>
<td>SVC END</td>
<td></td>
<td>22</td>
<td>6</td>
<td>N 0 99/99/99</td>
</tr>
<tr>
<td>DO</td>
<td>DATE-RECV DATE</td>
<td>DATE RECV</td>
<td></td>
<td>28</td>
<td>6</td>
<td>N 0 99/99/99</td>
</tr>
<tr>
<td>DO</td>
<td>PRICE</td>
<td>PRICE</td>
<td></td>
<td>34</td>
<td>6</td>
<td>N 0 $ZZZZZ9.</td>
</tr>
<tr>
<td>DO</td>
<td>LOCATION</td>
<td>LOCATION</td>
<td></td>
<td>40</td>
<td>9</td>
<td>A 0</td>
</tr>
<tr>
<td>DO</td>
<td>ROOM</td>
<td>ROOM</td>
<td></td>
<td>43</td>
<td>4</td>
<td>A 0 XXXX</td>
</tr>
<tr>
<td>DO</td>
<td>POS</td>
<td>POS</td>
<td></td>
<td>47</td>
<td>2</td>
<td>A 0 XX</td>
</tr>
<tr>
<td>DO</td>
<td>ACCOUNT NO</td>
<td>ACCOUNT-NO</td>
<td></td>
<td>49</td>
<td>10</td>
<td>A 0</td>
</tr>
<tr>
<td>DO</td>
<td>ACCT-POS-1</td>
<td></td>
<td></td>
<td>49</td>
<td>1</td>
<td>A 0 X</td>
</tr>
<tr>
<td>DO</td>
<td>MAINT VENDOR</td>
<td>MAINT VND</td>
<td></td>
<td>59</td>
<td>3</td>
<td>A 0 XXX</td>
</tr>
<tr>
<td>DO</td>
<td>OPTIONS</td>
<td>OPTIONS</td>
<td></td>
<td>62</td>
<td>50</td>
<td>A 0</td>
</tr>
</tbody>
</table>
## MODEL DICTIONARY

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Top</th>
<th>Bottom</th>
<th>1st</th>
<th>#</th>
<th>T</th>
<th>S</th>
<th>Printing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model-Curr-Maint-Rate</td>
<td>Current monthly maintenance rate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model-Desc</td>
<td>Description of model</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model-Name</td>
<td>Name of Model</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model-New-Maint-Rate</td>
<td>Pending monthly maintenance rate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model-No</td>
<td>Unique key of Model Dictionary</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>First three characters are the code of the manufacturer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model-Rec</td>
<td>Represents the entire Model Dictionary. Not used in writing queries.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model-Vendor-No</td>
<td>Link to Vendor Dictionary</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Alphabetic code to designate a manufacturer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>Title of Item</th>
<th>1st</th>
<th>#</th>
<th>T</th>
<th>S</th>
<th>Printing</th>
</tr>
</thead>
<tbody>
<tr>
<td>DD MODEL-REC</td>
<td>*</td>
<td>53</td>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD MODEL-NO MODEL</td>
<td>NO</td>
<td>5</td>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD MODEL-NAME</td>
<td>MODEL</td>
<td>10</td>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD MODEL-VENDOR-NO</td>
<td>VENDOR</td>
<td>3</td>
<td>A</td>
<td>0</td>
<td>XXX</td>
<td></td>
</tr>
<tr>
<td>DD MODEL-CURR-MAINT-RATE</td>
<td>CURR</td>
<td>6</td>
<td>N</td>
<td>2</td>
<td>$ZZZ9.99</td>
<td></td>
</tr>
<tr>
<td>DD MODEL-NEW-MAINT-RATE</td>
<td>RATE</td>
<td>22</td>
<td>N</td>
<td>2</td>
<td>$ZZZ9.99</td>
<td></td>
</tr>
<tr>
<td>DD MODEL-DESC</td>
<td>DESC</td>
<td>25</td>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

217
# VENDOR DICTIONARY

<table>
<thead>
<tr>
<th>Item</th>
<th>Top ID</th>
<th>Name</th>
<th>Bottom Title</th>
<th>1st #</th>
<th>T</th>
<th>S</th>
<th>Printing</th>
</tr>
</thead>
<tbody>
<tr>
<td>VENDOR-REC</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>29</td>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>VENDOR-NO</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VENDOR-NME</td>
<td>VENDOR</td>
<td>NO</td>
<td></td>
<td>1</td>
<td>3</td>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>VENDOR-NME</td>
<td>VENDOR</td>
<td>NAME</td>
<td></td>
<td>4</td>
<td>25</td>
<td>A</td>
<td>0</td>
</tr>
</tbody>
</table>
Enclosure III

Glossary

Terminal: Communications device used to control user's operations to send and receive information from the computer.

Coupler: Communications device that transmits signals from the terminal to the computer over telephone lines.

Baud Rate: Speed at which terminal sends and receives messages. 300 baud = 30 char/sec; 1200 baud = 120 char/sec

CPU: Processing component of computer.

Login: Procedures that establishes communication with the computer.

Environments: Portion of computer that has control.

Command: Computer language instruction to perform a task.

Delimiter: Character placed before or after one or more characters in order to set them apart as a group, for example a slash (/).

File: Collection of records or lines processed as a unit.

Filename: Descriptive name used as an identifier (limited to 6 characters)

Filetype: Descriptive name used as a second identifier (limited to 3 characters)

Record: One item or line of input.

String: Series of one or more characters.

Off-line Device: Device that is accessed interactively by the system as it is available.

IQL: Interactive Query Language
Dictionary  Table stored by IQL that describes a data file or data base.

Alphavariate  An alphanumeric working item created by IQL while generating a report. Names of alphavariates must start with A, can be up to thirty characters and can contain a - z, 0 - 9 and -.

Assistance Level  The initial command level in IQL.

Constant  A pure number used anywhere in an IQL command. Constants can contain decimal points or omit them.

Current record  If a file is positioned at a particular record, that record is called the current record.

Data Item  An item in a record read by IQL from a data file or data base. IQL lists data-item names under the ITEM NAME column title of the dictionary display.

Edit Level  IQL enters the edit level when issue either the WRITE or EDIT assistance command to write or edit a query file.

Help Word  A help word is a key word that you submit to IQL for a display of explanatory help text.

Literal  A literal is any string of characters enclosed in quotes and used by IQL exactly as you specified.

Query  A query is a collection of source statements written like English language sentences that IQL uses to generate one or more reports.
Please circle the desired response. Feel free to elaborate on any point you see fit. You may use the back of the page if necessary.

The evaluation is divided into three parts: questions concerning (I) the instructional periods, (II) the on-line exercises, and (III) the sessions in general.

I Instructional Period:

1. Were the sessions:
   Too Long <----------------> Too Short
   X X X X X

2. Did you feel that the number of commands covered during each session was:
   Too Many <----------------> Too Few
   X X X X X

3. Did you find the IOL End-User’s Guide:
   No Value <----------------> Very Helpful
   X X X X X

4. Did you get enough personalized attention?
   Yes ______ No ______

5. Did you find the training program manual:
   No Value <----------------> Very Helpful
   X X X X X
II On-line Exercises

1. Were the number of exercise sessions:
   Too Many <-------------------> Too Few
   X   X   X   X   X   X

2. Did you find the on-line exercises:
   Too Difficult <-------------------> Too Easy
   X   X   X   X   X   X

3. Would you have preferred to work with a partner?
   Yes _____  No _____  Don't Know _____

4. Were your questions answered:
   Promptly
   Yes _____  No _____
   Satisfactorily
   Yes _____  No _____
III Sessions in General:

1. Were there any commands or query statements in particular that caused you difficulty? If so, which ones?

2. Would you have preferred to have had one all day class instead of two half-day sessions? Yes _____ No _____

3. Do you feel you have enough basic knowledge to start using IQL? Yes _____ No _____

4. What suggestions or comments do you have regarding the training program?
APPENDIX VII
PROFILE OF TRAINING PROGRAM PARTICIPANTS

The training program participants are all employees of the Lehigh University Computing Center (LUCC) and reflect a variety of user types. Three of the trainees have extensive computer exposure but are unfamiliar with IQL or the data base system on which the training exercises are based. One of these trainees has previous computer exposure but not with the DEC 20. Two of the other three trainees have much less computer exposure, but are intimately familiar with the inventory data base. The final trainee has the least amount of computer exposure and is only superficially familiar with the inventory data base system.

A short profile of each of the participants follows:

Carol Rauch - Operations Manager:

Previous experience as a user consultant assisting users with programming problems
Extensive familiarity with computer hardware but limited programming exposure in recent years
Familiarity with TOPS-20 Command language
Intimate familiarity with Inventory system
Knowledge of how to enter and exit IQL environment and execute analyzed queries

Kathy Horwath - Operations Administrative Assistant:

Limited computer exposure
Familiarity with TOPS-20 Command Language
Familiarity with Inventory data base

Knowledge of how to enter and exit IQL environment and execute analyzed queries

Monica Morganello - Operations Secretary:

Very limited computer exposure

Limited experience with TOPS-20 command language for data entry

Surface familiarity with Inventory data base

Becoming familiar with tape handling system that uses IQL for reporting

Tim Foley - Technical Consultant:

Extensive computer experience

No previous exposure to the Inventory system

Interest in the course lies in the possibility of developing an accounting system utilizing IQL for its reporting. The types of queries he would need to generate are very similar in nature to those of the inventory system.

Kevin Weiner - User Consultant:

Extensive computer experience

No previous exposure to the Inventory system

Interest in the course also lies in the possibility of utilizing IQL for an accounting system.

Mati Vaze - Systems Programmer:

Previous computer exposure on PDP/11 and IBM 4331

Unfamiliar with TOPS-20 Command Language
No previous exposure to the Inventory system

Interest in course lies in gaining working knowledge of DEC 20 and familiarity with IQL.
IQL

END-USERS

GUIDE

September 1982
This manual contains a subset of the features of IQL which are appropriate for an end-user utilizing IQL as a query language. "Interactive Query Language User's Guide" was used as a guideline in developing this manual and it should be referenced should it be desirable to become acquainted with all the features of IQL.
# Table of Contents

1. **RUNNING IQL**  
   1.1 Explaining On-line HELP facilities 232  
   1.2 Explaining Message Formats 233  
   1.3 Using Assistance Commands 234  
   1.4 Creating, Editing, and Saving a Query 234  
   1.5 Using a Query File to Generate a Report 235  
   1.6 Maintaining Query Files 235  
   1.7 Assistance Command Formats 235  
   1.8 Assistance Commands 236

2. **Creating and Editing a Query**  
   2.1 Formatting Defaults 253  
   2.2 Ordering Query Statements 253  
   2.3 Commenting in a Query 253  
   2.4 Query Statement Formats 254  
   2.5 Query Statements 256

I. **IQL Error Messages**  

231
CHAPTER 1
RUNNING IQL

You enter the IQL environment from the Operating System by typing IQL. You terminate the command with a carriage return which permits the Operating System to execute IQL. IQL displays a <OA> prompt and expects you to input an assistance command.

1.1 Explaining On-line HELP facilities

IQL has on-line help facilities consistent with the TOPs-20 operating system. There are of three different formats:

1. ? A question mark entered after partially typing in a command will cause IQL to list the possible commands that begin with those letters and wait for you to finish the command.

For instance:

<QA>d? Type an assistance mode command one of the following:

DEFINE DELETE DICTIONARIES DUMP

<QA>d

2. $ Hitting the escape key after partially completing a command will cause IQL to finish the command and prompt for any required information.

For instance:

<QA>queries (available)

Queries stored in your directory:
3. HELP ? will cause IQL to display the names of all available HELP messages.

HELP name will cause the system to give an explanation of the command in question.

These on-line HELP facilities apply only when you are in the query assistance environment (signified by the <QA> prompt). If you are entering a query via the IQL editor, error messages will not appear until you RUN or EXECUTE the query.

1.2 Explaining Message Formats

IQL displays messages as follows:

1. (message)

IQL displays progress reports or information messages and encloses them in parentheses. You are not required to take any action.

For instance, IQL displays:

(IQL25 DELETED)

The message informs you that query IQL25 has been deleted from the file of nonanalyzed queries.

2. % message

Messages starting with a percent sign are warning messages.

For instance, IQL might display:

% CANNOT FIND THIS DICTIONARY

The message informs you that IQL cannot locate the dictionary that you specified in an assistance command. Repeat the command with a valid dictionary or use the DICTIONARY
1.3 Using Assistance Commands

To enter an assistance command, you need only supply enough of the command to uniquely identify it. (For instance, DIC is enough to identify the DICTIONARIES command.)

1.4 Creating, Editing, and Saving a Query

When you first write a query with either the EDIT or WRITE command, IQL enters the information into the current query area. IQL uses the current query area as a work space. The current query area contains the source query statements that you enter following the EDIT or WRITE assistance command. The source query statements are described in Chapter 2. You can store the current query area into a file with either the STORE or SAVE command. With the STORE command, IQL writes the nonanalyzed current query area into a permanent disk file. With the SAVE command, IQL analyzes the source query statements from either the current query area or a nonanalyzed query file and writes the analyzed query into a permanent disk file.

You can edit a nonanalyzed query file but you cannot edit an analyzed query. You should, therefore, be certain a query is stored or replaced before saving it.
1.5 Using a Query File to Generate a Report

IQL first analyzes the query instructions and then performs them. Use the RUN command to analyze the query and to generate the report(s) from a nonanalyzed query. The nonanalyzed query can be located in either the current query area or in a nonanalyzed query file.

Use the EXECUTE command to generate a report from an analyzed query. Before using the EXECUTE command, you must analyze and save the nonanalyzed query with the SAVE command. Generating a report from an analyzed query is faster than generating a report from a nonanalyzed query.

1.6 Maintaining Query Files

IQL retains nonanalyzed queries in a single file and analyzed queries in separate files. Use the QUERIES command to list the nonanalyzed queries. Use the DELETE command to remove a nonanalyzed query and the REPLACE command to replace a stored nonanalyzed query with the contents of the current query area.

1.7 Assistance Command Formats

The assistance commands use the following command conventions:

1. First character location - The command begins in the first character location after the <QA> prompt.
2. **Space** - One or more spaces separate words in a query sentence.

3. **lower case and upper case characters** - A parameter identified by lower case characters indicates that you are to supply a variable as indicated by the variable name.

   A parameter identified by upper case characters indicates that you are to supply the exact characters as they are shown in the text. Usually, an upper case parameter is an option to the command and provides you with additional functions to the command.

4. **Square brackets [ ]** - A parameter enclosed in square brackets indicates an optional parameter. Do not use the square brackets in the assistance commands you write. When you do not supply the parameter, the system applies a default value as explained in the parameter description.

   A parameter not enclosed in square brackets indicates that the system requires the parameter.

5. **Boldface characters** - The characters of the command that are in boldface in the FORMAT sections represent the abbreviation that can be used for that command.

### 1.8 Assistance Commands

The remainder of the chapter presents the assistance commands in alphabetic order.
DELETE

FUNCTION:
The DELETE command removes the stored nonanalyzed query(s) from the file of stored queries.

FORMAT:
DELETE query-name [... query-name]

EXAMPLES:
<QA> DELETE IQL01 IQL30
(IQL01 DELETED)
% IQL30 NOT FOUND TO BE DELETED

<QA>
IQL deleted IQL01, but could not find a query with the name IQL30. IQL therefore displays a warning message.
DICTIONARIES

FUNCTION:
The DICTIONARIES command provides a terminal display of the file information for each dictionary in the directory.

FORMAT:
DICTIONARIES

DISCUSSION:
The only column of concern to you is the initial column DICT NAME.

DICT NAME
The DICT NAME column lists the dictionary names.

EXAMPLE:

<QA> DICTIONARIES

Dictionaries in your directory:

<table>
<thead>
<tr>
<th>Dict Name</th>
<th>File Type</th>
<th>File Name</th>
<th>Direct Type</th>
<th>Rec Len</th>
<th>Blk Fac</th>
<th>Key Loc</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUILDING</td>
<td>IS Dsk7</td>
<td>BLDG TDX</td>
<td></td>
<td>29</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>VENDOR</td>
<td>IS Dsk7</td>
<td>VENDOR TDX</td>
<td></td>
<td>29</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>EQUIPMENT</td>
<td>IS Dsk7</td>
<td>EQUIP TDX</td>
<td></td>
<td>112</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>CUSTOMER</td>
<td>IS Dsk7</td>
<td>CUST TDX</td>
<td></td>
<td>52</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>MODEL</td>
<td>IS Dsk7</td>
<td>MODEL TDX</td>
<td></td>
<td>53</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>PROGRAMS</td>
<td>SQ Dsk7</td>
<td>PROG SEQ</td>
<td></td>
<td>57</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(End list of dictionaries)

As shown in the example, the directory contains the dictionaries: BUILDING, VENDOR, EQUIPMENT, CUSTOMER, MODEL, AND PROGRAMS.
EDIT

FUNCTION:
The EDIT command transfers control to a text editor to edit or change either the current query area or a stored nonanalyzed query in the file of stored queries.

FORMAT:
EDIT [query-name] [LIST]

DEFAULTS:
If you omit a query name, IQL edits the current query. If you omit the LIST option, IQL does not list the query before entering the editor.

DISCUSSION:
If you furnish a query name, IQL moves the contents of the related query to the current query area before entering the editor.

Any TOPS-20 EDIT commands are valid in the IQL editor. There is a distinction between the IQL editor and the TOPS-20 editor. You cannot EDIT a query that does not exist. In TOPS-20 editor, if the file does not exist the system will assume that it is a new file and put you in the INPUT environment. EDIT in IQL will give an error message if you try to edit a query that does not exist. You must use the command WRITE to create a new query.

In the TOPS-20 editor, when you type 'E' to exit EDIT, the file you are editing is permanently saved. This is not so with the IQL editor. Typing E or EU to leave EDIT only saves a 'temporary' file. If you EXIT IQL and later enter IQL again the query will be gone. You must STORE or REPLACE the query to make it a permanent query.
EXAMPLES:

EDIT IQL14

**IQL14

Edit: QC012S.TMP.3

(Any valid EDIT commands are appropriate)

*EU

[QC012S.TMP.4]

IQL1, in this instance, located IQL14 in the file of nonanalyzed queries and brought it into the current query area as a temporary file QC012S.TMP. Changes were made using EDIT environment commands and the current query area was modified. EU was used to exit the EDIT environment.

These changes had no effect on the stored nonanalyzed query. To make these changes permanent, you would have to REPLACE IQL14. If you use the EXECUTE command to generate reports, you must also SAVE IQL14.
EXECUTE

FUNCTION:
The EXECUTE command generates a report from an analyzed query file.

FORMAT:
EXECUTE query-name

DISCUSSION:
You must specify an analyzed query that has been saved using the SAVF command. Each analyzed query file uses the extension of .INO. Refer to the SAVF command for more information on analyzed query files.

If you expect to use a query frequently without changing it, you can save computer time by storing the query, saving it, and using the EXECUTE command each time you want a report. With the EXECUTE command, IQL bypasses the analyze stage and produces the report.

EXAMPLE:

<QA>EXECUTE IQL50

(Report goes here if DISPLAY is on)

(End query phase; print file is QL112ELPT)

A print file is created named QL112E.LPT. If you want to print it, you must EXIT the IQL environment and issue the TOPS-20 PRINT command.
EXIT

FUNCTION:
The EXIT command returns control from IQL to the Operating System and displays an exit message.

FORMAT:
EXIT

DISCUSSION:
When you terminate the IQL session, IQL deletes all the working files it creates, including the current query area. To preserve the current query area before you terminate IQL, use the STORE command.

EXAMPLES:

<QA>EXIT
(End of IQL session)
ITEMS

FUNCTION:
The ITEMS command formats and displays the contents of a dictionary.

FORMAT:

ITEMS dictionary

DISCUSSION:
IQL formats the dictionary contents for a 72-character terminal display line. IQL displays the information in two sections. The first section is a description of the dictionary and the data file. Refer to the DICTIONARY command for an explanation of the first section. The second section of the display is a description of the dictionary contents labeled with column headings.

The columns that are of concern to you are: ITEM NAME, TOP TITLE, BOTTOM TITLE, NO. CHAR, TY, SC, and PRINTING PICTURE and these will be explained below.

ITEM NAME:
The item name is the name of the data item to be used in a query.

TOP TITLE:
The top title is the top column heading which will appear above a column when you use the item name in a query.

BOTTOM TITLE:
The bottom title is the bottom column heading which will appear above a column when you use the item name in a query.

NO. CHAR: The no. char. is the length of the item. For an alphabetic or alphanumeric item, the length of the item is expressed in characters. For a numeric item, the length is expressed in digits, exclusive of sign.

TY: TY refers to the item type; A for alphabetic or alphanumeric and N for numeric.
SC: SC tells the number of decimal positions in the data item.

PRINTING PICTURE:
The printing picture is a group of characters that IQL uses to format the item value for displaying or printing so that you can read it conveniently.

EXAMPLES:
<QA>ITEMS CUSTOMER

<table>
<thead>
<tr>
<th>Dict Name</th>
<th>File Type</th>
<th>File-in Name</th>
<th>Direct</th>
<th>Rec Len</th>
<th>Blk Fac</th>
<th>Key Loc</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSTOMER</td>
<td>IS Dsk7</td>
<td>CUST</td>
<td>TDN</td>
<td>52</td>
<td>10</td>
<td>1</td>
</tr>
</tbody>
</table>

Item ID Name | Top Title | Bottom Title | 1st # | T S | Printing Picture
-------------|-----------|--------------|-------|----|-------------------
DD CUST-FIRST-NAME | FIRST NAME | 5' 10 A 0 |       |     |                   
DD CUST-LAST-NAME | LAST NAME  | 15 15 A 0   |       |     |                   
DD*CUST-REC | | | | | |
DD CUST-BLDG-NO | BLDG       | NO 30 3 N 0 999 |       |     |                   
DD CUST-NO CUST | NO         | 1 4 N 0 9999 |       |     |                   
DD CUST-ADDR | ADDRESS    | 30 7 A 0 XXX-XXXX |       |     |                   
DD CUST-ROOM | ROOM       | NO 33 4 A 0 XXXX |       |     |                   
DD CUST-DEPT | DEPARTMENT | 37 15 A 0 |       |     |                   
DD CUST-NAME | NAME       | 5 25 A 0 |       |     |                   

(End list of items)

This command displayed the names of the data items in the Customer Dictionary. For example CUST-NO is a 4 digit zero decimal numeric data item with the report heading CUST//NO.
QUERIES

FUNCTION:
The QUERIES command displays a list of all nonanalyzed queries stored by IQL in the file of stored queries.

FORMAT:

QUERIES

EXAMPLE:

<QA>queries
Queries stored in your directory:

XAVERAGE XDATE XDISPLAY
XFIND XHEADER XHSPACE
XIF XLMARGIN XMAXIMUM
XMINIMUM XO-EN XTALLY
XPRINT XRMARGIN XSORT
XTOTAL XVSPACE XOPEN2

(End list of stored queries)
REPLACE

FUNCTION:
The REPLACE command deletes the named query from the file of stored queries and writes the contents of the current query area into the file of stored queries.

FORMAT:
REPLACE [query-name]

DEFAULTS:
If you omit the query name from the command line but you name the current query area, IQL replaces the stored nonanalyzed query with the contents of the current query area. If you do not name the current query area and do not supply a query name with the command, IQL prompts you for a query name.

DISCUSSION:
The contents of the current query area are not affected by the REPLACE command.

EXAMPLES:

<QA>REPLACE IQL50
(IQL50 replaced)
RUN

FUNCTION:
The RUN command processes a nonanalyzed query and generates a report.

FORMAT:
RUN [query-name] [NOLIST]

DEFAULTS:
If you omit the query name, IQL runs the query located in the current query area. If you furnish a query name, IQL moves that query from the stored nonanalyzed query into the current query area, analyzes the query, and generates a report.

DISCUSSION:
The query in the current query area is not affected by IQL when you use the RUN command to generate a report from the current query area.

If IQL finds an error in the query during the analysis, IQL issues the appropriate error messages and returns to the assistance level so that you can correct the errors (by using the EDIT command).

EXAMPLES:

<QA>RUN
(Listing of query goes here)
(Report goes here if DISPLAY is on)
(End query phase; print file is QL012ELPT)

The print file QL012ELPT has been created. If you want to print it, EXIT the IQL environment and print it.
SAVE

FUNCTION:
The SAVE command analyzes and saves in an analyzed query file a stored nonanalyzed query. You can subsequently generate a report from an analyzed query file with the EXECUTE command.

FORMAT:
SAVE query-name [NOLIST] [RUN]

DEFAULTS:
IQL analyzes the nonanalyzed query in the file of stored queries and writes the analyzed query into a file named name.INQ. The filename is the first six characters of the query name.

If you use the NOLIST option, IQL does not list the query contents. If you omit the NOLIST option, IQL lists the query contents.

If you use the RUN option, IQL analyzes the query, generates the report, and saves the analyzed query into a disk file.

DISCUSSION:
IQL writes the analyzed query in a disk file with a filename consisting of the first six characters of the query name combined with the extension .INQ. Since some versions of the system utilities do not fully process filenames with dashes (-) in them, you should avoid using dashes in the first six characters of the query name.

You should always store or replace a modified nonanalyzed query prior to saving it. Otherwise, it is possible that a discrepancy could exist between the nonanalyzed file and the saved file.

If a .INQ file already exists with target name, IQL will load that nonanalyzed query into the current area, analyze it and save it. You will thus lose the current query you were working with unless you stored it before issuing the save command.
EXAMPLES:

<QA>SAVE IQL50 NOLIST
**IQL50
STORE

FUNCTION:
The STORE command stores the current query area into the file of nonanalyzed queries.

FORMAT:

STORE [query-name]

DEFAULT:
If you omit a query name, IQL uses the name of the current query area. If you did not name the current query area, IQL prompts you for a query name.

DISCUSSION:
IQL writes the nonanalyzed query into the file of stored queries. You can retrieve a stored nonanalyzed query at any time with the LIST, EDIT, RUN, and SAVE assistance commands.

If you attempt to store a query and a query already exists by that name, IQL notifies you of the conflict and refuses to store the query. Neither the current query area nor the existing stored query is affected.

To generate a report from a stored query, use the RUN command with the query-name option. Refer to the RUN assistance command for more information.

EXAMPLES:

<QA>STORE IQL50
(IQL50 stored)

IQL50 is now stored in the file of nonanalyzed queries QPQRY5.SEQ. It is permanently stored until you either REPLACE or DELETE it.
STORE

FUNCTION:
The STORE command stores the current query area into the file of nonanalyzed queries.

FORMAT:

STORE [query-name]

DEFAULT:
If you omit a query name, IQL uses the name of the current query area. If you did not name the current query area, IQL prompts you for a query name.

DISCUSSION:
IQL writes the nonanalyzed query into the file of stored queries. You can retrieve a stored nonanalyzed query at any time with the LIST, EDIT, RUN, and SAVE assistance commands.

If you attempt to store a query and a query already exists by that name, IQL notifies you of the conflict and refuses to store the query. Neither the current query area nor the existing stored query is affected.

To generate a report from a stored query, use the RUN command with the query-name option. Refer to the RUN assistance command for more information.

EXAMPLES:

<QA>STORE IQL50
(IQL50 stored)

IQL50 is now stored in the file of nonanalyzed queries QPORYS.SEQ. It is permanently stored until you either REPLACE or DELETE it.
WRITE

FUNCTION:
The WRITE assistance command allows you to begin writing a new query in the current query area. IQL places you into the insert level of the editor.

FORMAT:
WRITE [query-name]

DEFAULT:
If you omit a query name, IQL does not name the current query area. If you later attempt to save or to store the current query area and omit the query name from the assistance command, IQL prompts you for a query name.

DISCUSSION:
The WRITE assistance command destroys the previous contents of the current query area.

EXAMPLES:

<QA> write

%File not found, Creating New file
Input: QC012S.TMP.1
00200  OPEN MODEL
00300  PRINT MODEL-NAME MODEL-NAME
00400  $ *
*EU

[QC012S.TMP.1]

IQL created a temporary file QC012S.TMP. You can RUN this query, but if you want to make it permanent, you must STORE it and possibly SAVE it.
CHAPTER 2
CREATING AND EDITING A QUERY

A query is a collection of query statements that IQL reads to generate one or more reports. The query statements direct IQL to do the following:

1. Read data files
2. Select records or data
3. Calculate information
4. Summarize the data
5. Format reports
6. Resequence the order of records

With the RUN assistance command, you instruct IQL to read a query, analyze it, and generate the report(s). You can store the query into a file with the REPLACE and STORE assistance commands. If you use the SAVE assistance command to create an analyzed query file, you should use the EXECUTE assistance command to generate the report(s). If you use the REPLACE, or STORE assistance command to create a nonanalyzed query file, and have not saved it also as an analyzed query file, you must use the RUN assistance command to generate the report(s). Refer to Chapter 2 for more information on these assistance commands.
2.1 Formatting Defaults

IQL use the following defaults in formatting reports if you do not override these defaults in a query:

1. Single spacing vertically
2. Three spaces between elements horizontally
3. Fifty-nine printed lines (including heading) per page
4. Date of run as mm/dd/yy at top left of each page
5. Page number at top right of each page
6. Left margin at 1
7. Right margin at 72

2.2 Ordering Query Statements

IQL determines which file you wish to process from the dictionaries you named in the OPEN statement. Any statements that IQL finds before the OPEN statement are executed once at the start of the query. The query runs much faster if you put overall format statements at the beginning of the query.

2.3 Commenting in a Query

Any statement that starts with an asterisk (*) is a comment. The comment continues until IQL encounters a period.
2.4 Query Statement Formats

The query statements use the following statement conventions:

1. First character location - The statement begins in the first character location of the line.

2. Space - One or more spaces separate words in a query.

3. Lower case and upper case characters - An argument identified by lower case characters implies that you are to supply a variable as indicated by the variable name.

   An argument identified by upper case characters indicates that you are to supply the exact characters as they are shown in the text.

4. Square brackets [ ] - An argument enclosed in square brackets indicates an optional argument. Do not use the square brackets in the query statements you write. When you do not supply the argument, the system applies a default value as explained in the argument description.

5. Special characters - An argument containing special characters, such as the equal sign, quotation marks, parentheses, and arithmetic symbols, requires the special characters as shown in the statement format. Parentheses must be preceded and followed by a space.

6. Alphanumeric values - Enter an alphanumeric value as a string of characters including quoted spaces. If you do not furnish all the characters to fill the field, the system left justifies the value and inserts spaces to the right of the value.

If the string contains spaces, enclose the string in either single quotes (' ) or double quotes (" ). The system understands either quote mark, but be sure you end the quoted space with the same quote mark with which you
started. If you use quotes in the value, set off the quoted value with the other quote mark. For instance:

'SMITHSONIAN INSTITUTION'

'BOB "CHIP" JONES'

"ROBERT C. SMITH"

To enter all spaces as the value for an alphanumeric field, enter the value as ' " '.

7. Numeric Values - Enter a numeric value as a string of special symbols and digits. You can use the following special symbols:

comma ,

dollar symbol $

decimal point .

signs + and -

The system ignores dollar signs and commas in the numeric value and accepts signs in the leading or trailing position of the value. If you do not enter a sign, the system assumes a positive value.

The system uses the decimal point that you supply to vertically align the value, but the system does not actually store the decimal point. The system knows from the scale in the dictionary where the decimal point is located in the numeric field.

If you do not furnish enough digits to fill the field, the system provides leading and trailing zeroes (after the decimal point) to fill the field.

To furnish an all zero value for a numeric item, simply enter the single digit 0.

8. Continuation - You can continue a query
2.5 Query Statements

The remainder of the chapter presents the query statements in alphabetical order.
AVERAGE

FUNCTION:
The AVERAGE statement calculates the average of a numeric item in one of two ways:

1. Over an entire report.
2. Within a specific value of one or more controlling items (breaks).

FORMAT:

AVERAGE item [BY item [ ... ]]

DEFAULT:
If you do not specify a break item (that is, BY item), the average is calculated over all records that pass through the AVERAGE statement.

DISCUSSION:
The format and use of the AVERAGE statement are exactly the same as the other summary statements MAXIMUM, MINIMUM, TALLY, and TOTAL.

The AVERAGE statement is a stand-alone statement; it is not necessary for you to use TOTAL or TALLY in order to use AVERAGE.

If you specify a break item, IQL calculates the average and prints it when IQL detects a change in the item.

**You can specify more than one break item as long as the file has been sorted in the order you specify the breaks.**

Since averaging is an arithmetic operation, the item you average must be numeric. However the item(s) controlling the break can be any type.

IQL prints the average line on the report as an average title on the left and the average value on the right. IQL constructs the average title for an item from the top and bottom dictionary titles and from the word AVG. IQL constructs the title for a break item from the top and bottom dictionary titles of the break item and constructs the overall title for all the break items with the word overall.
**EXAMPLES:**

The following report calculates the average monthly maintenance rate for each vendor. It also calculates an average maintenance rate for all the models. It is not necessary to sort the dictionary into sequence by Vendor-No because it is already in that sequence.

**OPEN MODEL**

**AVERAGE MODEL-CURR-MAINT-RATE BY MODEL-VENDOR-NO.**

**AVERAGE MODEL-CURR-MAINT-RATE.**

06/10/82

<table>
<thead>
<tr>
<th>VENDOR NO</th>
<th>AJ CURR RATE</th>
<th>AVG: $4.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>VENDOR NO</td>
<td>CAL CURR RATE</td>
<td>AVG: $80.00</td>
</tr>
<tr>
<td>VENDOR NO</td>
<td>DEC CURR RATE</td>
<td>AVG: $187.00</td>
</tr>
<tr>
<td>VENDOR NO</td>
<td>IBM CURR RATE</td>
<td>AVG: $30.75</td>
</tr>
<tr>
<td>VENDOR NO</td>
<td>ICC CURR RATE</td>
<td>AVG: $4.00</td>
</tr>
<tr>
<td>VENDOR NO</td>
<td>ITT CURR RATE</td>
<td>AVG: $18.50</td>
</tr>
<tr>
<td>VENDOR NO</td>
<td>PCO CURR RATE</td>
<td>AVG: $4.00</td>
</tr>
<tr>
<td>VENDOR NO</td>
<td>TCP CURR RATE</td>
<td>AVG: $1.00</td>
</tr>
<tr>
<td>VENDOR NO</td>
<td>TEC CURR RATE</td>
<td>AVG: $0.00</td>
</tr>
<tr>
<td>VENDOR NO</td>
<td>TUC CURR RATE</td>
<td>AVG: $4.00</td>
</tr>
<tr>
<td>OVERALL CURR RATE</td>
<td>AVG: $59.23</td>
<td></td>
</tr>
</tbody>
</table>

(End query phase; print file is OL029ELPT)
DATE

FUNCTION:
The DATE statement turns report dating on or off or gives a report a specific date value to print in page headings.

FORMATS:

1. DATE [ON]
   DATE [OFF]

2. DATE value

DEFAULT:
If you do not use DATE in a query, report dating is automatically turned on - IQL puts the date of the run in report headings.

DISCUSSION:
Value can be an item, constant, or literal. You can turn report dating on or off, or change the value of the report date at any point.

If you use DATE OFF, IQL omits the date from the page heading.

If you use DATE ON, report dating is turned on and IQL places the date the report is run, formatted as mm/dd/yy, at the top left of each page heading.

If you furnish a value in DATE, report dating is turned on, and the value is printed at the top left of the page heading. The value can be a constant, a literal, or an item value. If it is a constant, it must contain six digits and it is printed with slashes inserted after the second and fourth digits. If it is a value, it is edited per the picture in the dictionary entry for that item.

EXAMPLES:
The same report heading is shown reflecting the use of the different formats of the DATE statement.
1. DATE ON (default)

OPEN VENDOR.
PRINT VENDOR=NO VENDOR-NAME.

06/09/82

PAGE 1

VENDOR VENDOR
NO NAME
AJ ANDERSON/JACOBSON
CAL CALCOMP, INC.
DEC DIGITAL EQ. CORP.

2. DATE OFF

DATE OFF.
OPEN VENDOR.
PRINT VENDOR=NO VENDOR-NAME.

PAGE 1

VENDOR VENDOR
NO NAME
AJ ANDERSON/JACOBSON
CAL CALCOMP, INC.
DEC DIGITAL EQ. CORP.

3. DATE 063082

DATE 063082.
OPEN VENDOR.
PRINT VENDOR=NO VENDOR-NAME.

06/30/82

PAGE 1

VENDOR VENDOR
NO NAME
AJ ANDERSON/JACOBSON
CAL CALCOMP, INC.
DEC DIGITAL EQ. CORP.
DISPLAY

FUNCTION:
The DISPLAY statement turns on or off an automatic terminal display of reports and displays messages on the terminal while IOL runs the query.

FORMATS:

1. DISPLAY [ON]
   DISPLAY [OFF]

2. DISPLAY value [... value]

DEFAULT:
If you do not use DISPLAY in a query, IQL automatically displays all reports at the terminal as well as creating a print file.

DISCUSSION:
If you do not want the report to be displayed on the terminal, use DISPLAY OFF.

Value can be an item, literal, or constant.

IQL sends single-spaced lines with no column titles to the terminal. IOL displays the item values in the display line form left to right as they occur in the DISPLAY statement. Unless you specify otherwise (as described below), IOL places three spaces horizontally between item values.

If you use an integer in the display statement, IQL places that many spaces between the item values for the remainder of the display line or until IQL encounters another such integer in the DISPLAY statement. The use of a spacing integer overrides any other horizontal spacing.

If you omit a spacing value and use the HSPACE statement in a query, IOL reserves as many spaces between items as you specify in the HSPACE statement.

Before IQL displays the line, IQL edits the item values as specified by the picture in the dictionary (or any override picture you furnish with the PICTURE statement).
Generally, you use the DISPLAY statement for short prompts or progress report messages.

EXAMPLES:

1. DISPLAY ON (default)
OPEN VENDOR.
PRINT VENDOR-NO VENDOR-NAME.

06/10/82

VENDOR VENDOR
NO NAME

AJ ANDERSON/JACOBSON
CAL CALCOMP, INC.
DEC DIGITAL EQ. CORP.
HAZ HAZELTINE INC.
IBM INTERNATIONAL BUS. MACH.
ICC ICC
ITT ITT
PCO PRECISION COMPONENTS
SOR SORBUS, INC.
TCP TELEX COMPUTER PRODUCTS
TEC TECHTRAN INDUSTRIES, INC.
TSP TIME SHARING PERIPHERALS
TUC TUCK

(End query phase; print file is OL129ELPT)

2. DISPLAY OFF (Report is not displayed on terminal)
DISPLAY OFF.
OPEN VENDOR.
PRINT VENDOR-NO VENDOR-NAME.

(End query phase; print file is OL429ELPT)
FIND KEY

FUNCTION:
The FIND KEY statement reads ISAM files randomly based on either the value of a data item or one or more specific key values. The FIND KEY statements permits reading on either full or partial keys.

FORMATS:

1. FIND KEY = literal [...literal]
   FIND KEYn = literal THRU literal [...literal THRU literal]
   FIND KEY = literal THRU EOF [...literal THRU EOF]

2. FIND KEY = item
   FIND KEYn = item

DISCUSSION:
If you omit the FIND KEY statement and you process an ISAM file, IQL reads the ISAM file sequentially. This is useful for searching items that are not key items.

If you use FIND KEY = literal, IQL advances the file directly to the record that contains the value of the literal as the full or partial (leading portion) of the key. IQL provides an end-of-file response if it does not find the record.

If you use a range of literals (literal THRU literal or literal THRU EOF), IQL starts with the first record whose key is in the range and each time IQL executes the FIND statement, IQL reads the next record in the range. If you use several such literal ranges, IQL goes to the first record in the next range when IQL exhausts the last range. It is not necessary that the ranges be in any specific order of key.

If you FIND KEY = item, IQL looks for a record that has as a key the value of the indicated item. If it does not find such a record, it returns an all-spaces record and continues. The name on the
right of the = sign is always an item in the first or primary dictionary in the OPEN statement.

To read a primary file, use the FIND KEY or FIND Key1 statement. To read a secondary file, use the FIND KEY2 statement; and to read a tertiary file, use the FIND KEY3 statement.

EXAMPLES:

In the following query, we want to list equipment so the equipment dictionary is the first or primary dictionary. For each piece of equipment we want to print the Model-Name instead of just the Model-No. Model-No is in the equipment dictionary and is our link to the Model Dictionary. We also want to print the customer's name along with his customer number. Cust-No is in the Equipment Dictionary and is our link to the Customer Dictionary.

OPEN EQUIPMENT MODEL CUSTOMER.
IF EQUIP-MODEL-NO = 'DEC04'
FIND KEY2 = EQUIP-MODEL-NO
FIND KEY2 = EQUIP-CUST-NO
PRINT EQUIP-VENDOR-NO MODEL-NAME EQUIP-SERIAL-NO
   CUST-NO CUST-NAME.

06/10/82

VENDOR    MODEL   SERIAL-NO    CUST-NO    NAME
DEC      PDP 11/34A  AG21996    0001    LUCC

(End query phase; print file is 0L529ELPT)
HEADING

FUNCTION:
The HEADING statement furnishes text for the title of the report and turns automatic page headings on or off.

FORMATS:

1. HEADING ON
   HEADING OFF

2. HEADING value [..., value ]

DEFAULT:
If you omit the HEADING statement in a query, IQL automatically turns on headings, and the heading text in the center remains blank.

DISCUSSION:
Value can be an item, constant, or literal; a constant or literal must be enclosed in parentheses. You can turn headings on or off at any point and as often as necessary.

If you turn heading on or off, IQL prints the body of the report with or without the heading as soon as IQL advances to a new page.

A page heading consists of the report date on the top left, one or more lines of report heading text in the center, and a page number on the top right. You can turn on or off any or all of these ingredients of the page heading with the DATE, HEADING, and PAGE statements.

IF you use Format 2, headings are turned on.

The text in Format 2 is exactly the same as the text for a PRINT statement, with the exception that IQL starts a new line if the literal contains a double slash (//). Note that since you can use items in the heading, part or all of the heading can come from the current record or variable values.

Each line of heading text is centered at the top of each page. The center is halfway between the current left and right margins.
If you are creating only one report, the query runs faster if you put the HEADING statement before the OPEN statement so that IOL only executes the HEADING statement once.

Headings, once set, remain set until you change them.

EXAMPLES:

We want to create a report with a title of BUILDING LISTING. In the first example, we want the title to print on the same line as the date and page number. In the second example, we want the title printed on the line after the date and page number.

1.
HEADING 'BUILDING LISTING'.
OPEN BUILDING,
PRINT BLDG-NO BLDG-NAME.

06/10/82            BUILDING LISTING          PAGE 1

BLDG       BLDG
NO         NAME

005   WHITAKER LABORATORY
006   MUDD BUILDING
007   SINCLAIR LABORATORY
014   CHRISTMAS-SAUCON HALL
019   PACKARD LABORATORY
023   EDUCATION BLDGS & CLINIC
027   ALUMNI BUILDING
030   LINDERMAN LIBRARY
031   WILLIAMS HALL
035   DROWN HALL

(End query phase; print file is OL829ELPT)
2.
HEADING 'BUILDING LISTING'.
OPEN BUILDING.
PRINT BLDG-NO BLDG-NAME.

06/10/82

BUILDING LISTING

<table>
<thead>
<tr>
<th>BLDG NO</th>
<th>BLDG NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>005</td>
<td>WHITAKER LABORATORY</td>
</tr>
<tr>
<td>006</td>
<td>MUDD BUILDING</td>
</tr>
<tr>
<td>007</td>
<td>SINCLAIR LABORATORY</td>
</tr>
<tr>
<td>014</td>
<td>CHRISTMAS-SAUCON HALL</td>
</tr>
<tr>
<td>019</td>
<td>PACKARD LABORATORY</td>
</tr>
<tr>
<td>023</td>
<td>EDUCATION BLDGS &amp; CLINIC</td>
</tr>
<tr>
<td>027</td>
<td>ALUMNI BUILDING</td>
</tr>
<tr>
<td>030</td>
<td>LINDERMAN LIBRARY</td>
</tr>
<tr>
<td>031</td>
<td>WILLIAMS HALL</td>
</tr>
<tr>
<td>035</td>
<td>DROWN HALL</td>
</tr>
</tbody>
</table>

(End query phase; print file is 06929ELPT)
FUNCTION:
The HSPACE statement sets the default horizontal spacing increment between items when IQL sets up a print or display line.

FORMAT:
HSPACE integer

DEFAULT:
If you omit the HSPACE statement in a query, IQL places three spaces between items in a print or display line.

DISCUSSION:
You can use HSPACE any place in a query, and as often as necessary.

If you use specific horizontal spacing in a PRINT statement, it overrides the current HSPACE setting. The HSPACE value remains unchanged.

If you are only using one HSPACE for all reports, place the HSPACE statement before the OPEN statement so it is executed only once. IQL runs faster if the HSPACE statement follows the OPEN statement.

EXAMPLES:
Instead of the default of 3 spaces between columns, we want to generate a listing of Vendor-No and Vendor-Name with 6 spaces between columns. The first example shows the report without the HSPACE statement and the second with the HSPACE statement.
1. HEADING '//BUILDING LISTING'.
OPEN BUILDING.
PRINT BLDG-NO BLDG-NAME.

<table>
<thead>
<tr>
<th>BLDG NO</th>
<th>BLDG NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>005</td>
<td>WHITAKER LABORATORY</td>
</tr>
<tr>
<td>006</td>
<td>MUDD BUILDING</td>
</tr>
<tr>
<td>007</td>
<td>SINCLAIR LABORATORY</td>
</tr>
<tr>
<td>014</td>
<td>CHRISTMAS-SAUCON HALL</td>
</tr>
<tr>
<td>019</td>
<td>PACKARD LABORATORY</td>
</tr>
<tr>
<td>023</td>
<td>EDUCATION BLDGS &amp; CLINIC</td>
</tr>
<tr>
<td>027</td>
<td>ALUMNI BUILDING</td>
</tr>
<tr>
<td>030</td>
<td>LINHENDERMAN LIBRARY</td>
</tr>
<tr>
<td>031</td>
<td>WILLIAMS HALL</td>
</tr>
<tr>
<td>035</td>
<td>DROWN HALL</td>
</tr>
</tbody>
</table>

(End query phase; print file is OLA29ELPT)

2. HSPACE 6.
OPEN BUILDING.
PRINT BLDG-NO BLDG-NAME.

<table>
<thead>
<tr>
<th>BLDG NO</th>
<th>BLDG NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>005</td>
<td>WHITAKER LABORATORY</td>
</tr>
<tr>
<td>006</td>
<td>MUDD BUILDING</td>
</tr>
<tr>
<td>007</td>
<td>SINCLAIR LABORATORY</td>
</tr>
<tr>
<td>014</td>
<td>CHRISTMAS-SAUCON HALL</td>
</tr>
<tr>
<td>019</td>
<td>PACKARD LABORATORY</td>
</tr>
<tr>
<td>023</td>
<td>EDUCATION BLDGS &amp; CLINIC</td>
</tr>
<tr>
<td>027</td>
<td>ALUMNI BUILDING</td>
</tr>
<tr>
<td>030</td>
<td>LINHENDERMAN LIBRARY</td>
</tr>
<tr>
<td>031</td>
<td>WILLIAMS HALL</td>
</tr>
<tr>
<td>035</td>
<td>DROWN HALL</td>
</tr>
</tbody>
</table>

(End query phase; print file is OLB29ELPT)

269
IF

FUNCTION:
The IF statement specifies a condition for IQL, to test. Depending on whether the condition is satisfied or not, the query statements that follow are performed. Control of the IF ends when IQL encounters a period.

FORMATS:

1. IF value1 relation value2 [AND ...] [ELSE ...]
   IF value1 relation value2 [OR ...] [ELSE ...]

2. IF FIRSTIME ... [ELSEF ...]
   IF LASTIME ... [ELSE ...]
   IF EDFn ... [ELSE ...]
   IF NEWGROUP ... [ELSE ...]

DISCUSSION:
Value1 and value2 can be one or more items, constants, or literals separated by a comma.

You can write relationships in any of the following ways:

<table>
<thead>
<tr>
<th>Relation</th>
<th>Accepted Words/Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>equal</td>
<td>FO IS = EQUALS</td>
</tr>
<tr>
<td>not equal</td>
<td>NO NE &lt;&gt; NEQ</td>
</tr>
<tr>
<td>less than</td>
<td>LS LT &lt; LESS</td>
</tr>
<tr>
<td>greater than</td>
<td>GR GT &gt; GREATER</td>
</tr>
<tr>
<td>less than or equal to</td>
<td>LEO LF &lt;= LQ</td>
</tr>
<tr>
<td>greater than or equal to</td>
<td>GFO GF &gt;= GQ</td>
</tr>
</tbody>
</table>

Note that you can use NOT to reverse a relationship. For instance: NOT EO is the same as NE; and NOT LEQ is the same as GR.
You can combine simple relationships to make a complex relationship by joining them with AND or OR clauses. You can use parentheses to clarify complex relationships. IQL evaluates all OR clauses before any AND clauses. You cannot nest IF statements; that is, IQL does not permit the construct of IF... IF... ELSE... ELSE.

If you use the ELSE clause, IQL performs the statement(s) that follow the ELSE clause if the condition preceding the clause is not satisfied.

If you are comparing two numbers, the decimal points are lined up and any necessary leading or trailing zeros are supplied before the comparison is made.

If you are comparing two alphanumeric items or literals, any necessary trailing spaces are added to the shorter item before the comparison is made.

EXAMPLES:

In the following example, we want to select only those equipment that are under maintenance contract (Service = 'M'). We want to list for this equipment, the Equipment number, customer number of the owner and the date the maintenance contract will expire.

OPEN EQUIPMENT CUSTOMER.
IF EQUIP-SERVICE = 'M'
FIND KEY2 = EQUIP-CUST-NO
PRINT EQUIP-NO EQUIP-CUST-NO CUST-NAME EQUIP-SVC-END-DATE
06/10/82 PAGE 1

EQUIPMENT NO CUST NO NAME SVC-END DATE
AJ 01-15460 0029 GERALD SMITH 82/06/30
AJ 01-6162 0001 LUCC 99/06/30
DECO5-AG01199 0058 ROBERT WEI 82/06/30
DECO6-AG10100 0007 EMORY ZIMMERS 82/06/30

(End query phase; print file is OLE29ELPT)
LMARGIN

FUNCTION:
The LMARGIN statement sets the left margin for printing reports.

FORMAT:
LMARGIN integer

DEFAULTS:
If you omit the LMARGIN statement in a query, IQL sets the left margin to 1.

DISCUSSION:
You can use LMARGIN any place in a query, and as often as necessary.

LMARGIN affects page headings, summary lines, and detail print lines.

EXAMPLES:
Instead of having our report start in column 1, we want it to start in column 10. The first example shows the report without the LMARGIN statement and the second with it.

1. HEADING "///BUILDING LISTING".
OPEN BUILDING.
PRINT BLDG-NO BLDG-NAME.

06/10/82 PAGE 1

BUILDING LISTING

BLDG  BLDG
NO   NAME

005  WHITAKER LABORATORY
006  MUDD BUILDING
007  SINCLAIR LABORATORY
014  CHRISTMAS-SAUCON HALL
019  PACKARD LABORATORY
023  EDUCATION BLDGS & CLINIC

(End query phase; print file is OLA29ELPT)
2.

HEADING 'BUILDING LISTING'.
LMARGIN 10.
OPEN BUILDING.
PRINT BLDG-NO BLDG-NAME.

06/10/82

BUILDING LISTING

<table>
<thead>
<tr>
<th>BLDG NO</th>
<th>BLDG NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>005</td>
<td>WHITAKER LABORATORY</td>
</tr>
<tr>
<td>006</td>
<td>MUDD BUILDING</td>
</tr>
<tr>
<td>007</td>
<td>SINCLAIR LABORATORY</td>
</tr>
<tr>
<td>014</td>
<td>CHRISTMAS-SAUCON HALL</td>
</tr>
<tr>
<td>019</td>
<td>PACKARD LABORATORY</td>
</tr>
<tr>
<td>023</td>
<td>EDUCATION BLDGS &amp; CLINIC</td>
</tr>
<tr>
<td>027</td>
<td>ALUMNI BUILDING</td>
</tr>
<tr>
<td>030</td>
<td>LINDERMANN LIBRARY</td>
</tr>
<tr>
<td>031</td>
<td>WILLIAMS HALL</td>
</tr>
<tr>
<td>035</td>
<td>DROWN HALL</td>
</tr>
</tbody>
</table>

(End query phase; print file is OLF29ELPT)
MAXIMUM

FUNCTION:
The MAXIMUM statement calculates the maximum value of a numeric item in one of two ways:

1. Over an entire report.
2. Within a specific value of one or more controlling items (breaks).

FORMAT:
MAXIMUM item [BY item [... item]]

DEFAULTS:
If you do not specify a break item (that is, BY item), the maximum is calculated over all records that pass through the MAXIMUM statement.

Unless you specify otherwise, each maximum statement both calculates the specified maximum and prints it at the proper time.

DISCUSSION:
The format and use of the MAXIMUM statement are exactly the same as the other summary statements AVERAGE, MINIMUM, TALLY, and TOTAL.

If you specify a break item, the maximum is calculated and printed on the change in the indicated item. If you specify more than one break item, MAXIMUM breaks on a change in any of the break items.

Since taking a maximum is an arithmetic operation, the item for which you get the maximum must be numeric. However, the item controlling the break can be any type.

IQL prints the maximum line on the report as a maximum title on the left and the maximum value on the right. IQL constructs the title for an item from the top and bottom dictionary titles and from the word MAX:. IQL constructs the title for a break item from the top and bottom dictionary titles of the break item and constructs the overall title for all the break items with the words OVERALL.
EXAMPLES:

We want to find the maximum rate that is being charged for monthly maintenance.

OPEN MODEL.
MAXIMUM MODEL CURR MAINT RATE.

06/10/82

OVERALL CURR RATE MAX: $658.00

(End query phase; print file is OLG29ELPT)
FUNCTION:
The MINIMUM statement calculates the minimum value of a numeric item in one of the two ways:

1. Over an entire report.
2. Within a specific value of one or more controlling items (breaks).

FORMAT:

MINIMUM item [BY item [... item]]

DEFAULT:
If you do not specify a break item (that is, BY item), the minimum is calculated over all records that pass through the MINIMUM statement.

Unless you specify otherwise, each MINIMUM statement both calculates the specified minimums and prints them at the proper time.

DISCUSSION:
The format and use of the MINIMUM statement are exactly the same as the other summary statements AVERAGE, MAXIMUM, TALLY, and TOTAL.

If you specify a break item, the minimum is calculated and printed on the change in the indicated item. If you specify more than one break item, MINIMUM breaks on a change in any of the break items.

Since taking a minimum is an arithmetic operation, the item you take the minimum of must be numeric. However, the break item can be any type.

IQL prints the minimum line on the report as a minimum title on the left and the minimum value on the right. IQL constructs the title for an item from the top and bottom dictionary titles and from the word MIN:. IQL constructs the title for a break item from the top and bottom dictionary titles of the break item and constructs the overall title for all the break items with the word OVERALL.
EXAMPLES:

We want to find the smallest maintenance rate (other than zero) being charged for monthly maintenance.

OPEN MODEL.
IF MODEL-CURR-MAINT-RATE NE 0
MINIMUM MODEL-CURR-MAINT-RATE.

06/10/82

OVERALL CURR RATE MIN: $ 1.00

(End query phase; print file is OLH29ELPT)
OPEN

FUNCTION:
The OPEN statement informs IQL which dictionaries and files to use in a query.

FORMAT:
OPEN dictionary [dictionary] [dictionary]

DISCUSSION:
Only one OPEN statement can be used in a query.

You can use from one to three dictionary names in an OPEN statement, depending on how many dictionaries you wish to process in the query.

The file described by the first dictionary in the OPEN statement is called the primary file. This file controls the current phase of the query. If it is a sequential file, IQL automatically reads it. If the primary file is an indexed sequential (ISAM) file and you do not do your own reading with a FIND KEY statement, IQL automatically reads it sequentially.

If you use more than one dictionary name in OPEN statement, the files described by the second and third dictionaries are secondary files. You must do your own reading of these files by appropriate use of FIND item or FIND KEYn. Normally secondary files are read under control of information you get from the primary file.

When IQL comes to the last statement in the current stage or it encounters a GO TO NR, IQL goes directly to the statement just after the OPEN. If it is reading sequentially as described above, it reads the next record before carrying out the statement.

Any statements that come before the OPEN statement are carried out only once at the beginning of the query and before any records are read. The query runs faster if you put one-time formatting statements at the beginning of the query when you are not going to change formats during the run.
EXAMPLES:

In the first example, we want to access the Model Dictionary. In the second example, we want to access the Equipment, Model, and Customer dictionaries. (To use information from the second and third named dictionaries, we must use the FIND KEY: statement).

1.
OPEN MODEL.
PRINT MODEL=NO MODEL=NAME.

06/10/82

MODEL
NO  MODEL
AJ 01  ADAC242
CAL01  1012
DECO1  LA36
DECO2  LA34
...
...

2.
OPEN EQUIPMENT MODEL CUSTOMER.
HEADING 'EQUIPMENT IN BUILDING', EQUIP-BLDG-NO.
IF EQUIP-BLDG-NO = 7
FIND KEY2 = EQUIP-MODEL-NO
FIND KEY3 = EQUIP-CUST-NO
PRINT EQUIP-VENDOR-NO MODEL=NAME CUST-NAME.

06/10/82

EQUIPMENT IN BUILDING 007

VENDOR
NO  MODEL  NAME
DEC  LA36  KEN  KLIER
DEC  LA34  GARY  SIMMONS
DEC  PDP 11/04  ROBERT  WEI
TEC  TECHT 8410  KEN  KLIER
TEC  TECHT 817  GARY  SIMMONS

(End query phase; print file is OL329ELPT)
PRINT

FUNCTION:
The PRINT statement specifies a print line in a report.

FORMATS:
PRINT value [,...., value]

DEFAULTS:
If you do not specify otherwise, IQL turns titling on, uses three spaces horizontally between item values, and single spaces vertically.

DISCUSSION:
Value can be an item, literal, or integer. Item values are laid out in the print line from left to right as they occur in your PRINT statement.

Item values are edited according to their picture in the dictionary (or any override picture you use in your query) before being put in the print line.

If you use an integer in the statement, the integer overrides standard spacing for the PRINT statement. IQL sets the integer to the horizontal spacing increment for the remainder of the print statement or until you use another such spacing integer. For instance, if you use HSPACE 2 and the PRINT statement:

PRINT NAME, STREET, 5, PHONE, TITLE, 1, YEAR

IQL provides two spaces between NAME and STREET, five spaces between STREET and PHONE, five spaces between PHONE and TITLE, and one space between TITLE and YEAR. If you do not use a spacing integer and use HSPACE in your query, as many spaces are left between item values as you designate with HSPACE.

If you print line is too long for the space between your current left and right margins, IQL truncates the print line on the right and continues. No error message is issued.

If titling is on, IQL prints two lines of column titles at the top of each new page. In laying out space for each item horizontally, IQL uses the longer of the edited item length or the length of
the column title. Column titles for alphanumeric items are left justified. Column titles for numeric items are right justified.

If titling is on, and the current print line is from a different PRINT statement than the preceding line, IQL prints titles before printing the body of the current line.

If titling is off, IQL prints only the current line. In laying out this line, IQL ignores the title lengths. Note that if you print a line, turn titling off, and print the line again, there can be a difference in alignment since the length of the column titles can affect the first line but not the second line. Vertical spacing is done just before the print line (and title if any) is printed.

EXAMPLES:

We want to print the names and addresses of all customers.

OPEN CUSTOMER.
PRINT CUST-NO CUST-NAME CUST-BLDG-NO CUST-ROOM.

06/10/82 PAGE 1

<table>
<thead>
<tr>
<th>CUST NO</th>
<th>NAME</th>
<th>BLDG NO</th>
<th>ROOM NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>LUCC</td>
<td>019</td>
<td>154</td>
</tr>
<tr>
<td>0006</td>
<td>GARY SIMMONS</td>
<td>007</td>
<td>305B</td>
</tr>
<tr>
<td>0007</td>
<td>EMORY ZIMMERS</td>
<td>019</td>
<td>222</td>
</tr>
<tr>
<td>0010</td>
<td>BERRY RICHARDS</td>
<td>030</td>
<td>100</td>
</tr>
<tr>
<td>0013</td>
<td>RICHARD HERTZBERG</td>
<td>005</td>
<td>454</td>
</tr>
<tr>
<td>0019</td>
<td>GEORGE SIH</td>
<td>019</td>
<td>451A</td>
</tr>
<tr>
<td>0021</td>
<td>STAN JOHNSON</td>
<td>019</td>
<td>551</td>
</tr>
<tr>
<td>0023</td>
<td>KEN KLER</td>
<td>007</td>
<td>315</td>
</tr>
<tr>
<td>0026</td>
<td>RAYMOND BELL</td>
<td>023</td>
<td>524</td>
</tr>
<tr>
<td>0028</td>
<td>LEE TUSCHER</td>
<td>023</td>
<td>520</td>
</tr>
<tr>
<td>0029</td>
<td>GERALD SMITH</td>
<td>030</td>
<td>7</td>
</tr>
<tr>
<td>0040</td>
<td>FRANK LUH</td>
<td>035</td>
<td>203</td>
</tr>
<tr>
<td>0048</td>
<td>ARTHUR HUMPHREY</td>
<td>027</td>
<td>204F</td>
</tr>
<tr>
<td>0051</td>
<td>ALISTAIR MAC PHERSON</td>
<td>019</td>
<td>558</td>
</tr>
<tr>
<td>0055</td>
<td>DONALD TALHELM</td>
<td>019</td>
<td>304</td>
</tr>
<tr>
<td>0058</td>
<td>ROBERT WEI</td>
<td>007</td>
<td>327B</td>
</tr>
</tbody>
</table>

(End query phase; print file is OL229ELPT)
RMARGIN

FUNCTION:
The RMARGIN statement sets the right margin for printing reports.

FORMAT:
RMARGIN integer

DEFAULTS:
If you do not use RMARGIN in your query, IQL uses a right margin of 72.

DISCUSSION:
You can use RMARGIN any place in your query, and as often as necessary.
RMARGIN affects page headings, summary lines, and detail print lines.

EXAMPLES:
We are going to create a report that will be too wide to fit on the screen but will fit on paper if we print it off-line. We should include the DISPLAY OFF statement in our query because the report will not be very readable at the terminal because it will fold over.

RMARGIN 132.
DISPLAY OFF.
OPEN EQUIPMENT.
IF EQUIP-OPTIONS NE ' '
PRINT EQUIP-NO EQUIP-CUST-NO FONTP-STATUS EQUIP-OPTIONS.

(End query phase; print file is OL229ELPT)
SORT

FUNCTION:
The SORT statement sorts the primary input file on one or more keys. The keys can be data items or variables and can be sorted in ascending or descending order.

FORMAT:

\[
\text{SORT BY [ASCENDING] value } , \ldots [\text{ASCENDING}] \text{ value}
\]

\[
\text{SORT BY [DESCENDING] value } , \ldots [\text{DESCENDING}] \text{ value}
\]

DEFAULT:

IF you do not specify ASCENDING or DESCENDING, IQL sorts on an ascending order.

DISCUSSION:

Value can be an item or a variable. There is no maximum number of sort keys that can be used other than that the number of characters in all sort keys - taken together - cannot exceed the total length of the record.

You can sort on calculated variable values. However, the calculated values do not pass through the SORT statement.

EXAMPLES:

Suppose we want to sort items from the Equipment Dictionary in order by Model-Name (which is in the Model Dictionary). The following sequence is required.

1. OPEN EQUIPMENT MODEL.
   
   FIND KEY2 = EQUIP-MODEL-NO.
   
   (The link in the primary dictionary to the secondary)

2. SORT BY EQUIP-VENDOR-NO MODEL-NAME.
   
   Because the previous FIND KEY was issued, Model-Name is available to the sort for sequencing. The only data items that are actually kept in the sorted file, however, are those from the Equipment Dictionary.
3. If we want to print information from the Model Dictionary, it will be necessary to issue another FIND KEY statement after the SORT statement.

FIND KEY2 = EQUIP-MODEL-NO.

4. We can now print the information we want.

PRINT EQUIP-VENDOR-NO MODEL-NAME ...

The query will now be run to demonstrate the above.

OPEN EQUIPMENT MODEL.
FIND KEY2 = EQUIP-MODEL-NO.
SORT BY EQUIP-VENDOR-NO MODEL-NAME.
FIND KEY2 = EQUIP-MODEL-NO.
PRINT EQUIP-VENDOR-NO MODEL-NAME EQUIP-SERIAL-NO.

06/10/82 PAGE 1

<table>
<thead>
<tr>
<th>VENDOR NO</th>
<th>MODEL</th>
<th>SERIAL-NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>AJ</td>
<td>ADAC242</td>
<td>15460</td>
</tr>
<tr>
<td>AJ</td>
<td>ADAC242</td>
<td>6162</td>
</tr>
<tr>
<td>DEC</td>
<td>LA120</td>
<td>PN09657</td>
</tr>
<tr>
<td>DEC</td>
<td>LA120</td>
<td>WF24140</td>
</tr>
<tr>
<td>DEC</td>
<td>LA34</td>
<td>08457</td>
</tr>
<tr>
<td>DEC</td>
<td>LA34</td>
<td>16345</td>
</tr>
<tr>
<td>DEC</td>
<td>LA34</td>
<td>32663</td>
</tr>
</tbody>
</table>

(End query phase; print file is OLG30ELPT)
TALLY

FUNCTION:
The TALLY statement counts the occurrences of an item and enters the total in one of two ways:

1. Over an entire report.
2. Within a specific value of one or more controlling items (breaks).

FORMAT:

TALLY item [BY item [...]]

DEFAULT:
If you do not specify a break item (that is, BY item), the tally is calculated over all records that pass through the TALLY statement.

Unless you specify otherwise, each TALLY statement calculates the specified tally and prints it at the proper time.

DISCUSSION:
The format and use of the TALLY statement are exactly the same as the other summary statements AVERAGE, MAXIMUM, MINIMUM, and TOTAL.

If you specify a break item, the tally is calculated and printed on the change in the indicated item. If you specify more than one break item, TALLY breaks on a change in any of the indicated break items.

You can tally any type of item or break items, since TALLY merely runs an internal counter that increments by one each time IQL encounters the TALLY statement.

IQL prints the tally line on the report as a tally title on the left and the tally value on the right. IQL constructs the title for an item from the top and bottom dictionary titles and from the word TALLY:. IQL constructs the title for a break item from the top and bottom dictionary titles of the break item and constructs the overall title for all the break items with the words OVERALL.
EXAMPLES:

Suppose we want to find out how many terminals we have in each building.

HEADING '// EQUIPMENT COUNT BY BUILDING'.
OPEN EQUIPMENT.
SORT BY EQUIP-BLDG-NO.
TALLY EQUIP-NO BY EQUIP-BLDG-NO.

06/10/82

<table>
<thead>
<tr>
<th>BLDG NO</th>
<th>EQUIPMENT NO</th>
<th>TALLY</th>
</tr>
</thead>
<tbody>
<tr>
<td>005</td>
<td></td>
<td></td>
</tr>
<tr>
<td>006</td>
<td></td>
<td></td>
</tr>
<tr>
<td>007</td>
<td></td>
<td></td>
</tr>
<tr>
<td>011</td>
<td></td>
<td></td>
</tr>
<tr>
<td>014</td>
<td></td>
<td></td>
</tr>
<tr>
<td>019</td>
<td></td>
<td></td>
</tr>
<tr>
<td>023</td>
<td></td>
<td></td>
</tr>
<tr>
<td>027</td>
<td></td>
<td></td>
</tr>
<tr>
<td>030</td>
<td></td>
<td></td>
</tr>
<tr>
<td>031</td>
<td></td>
<td></td>
</tr>
<tr>
<td>035</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(End query phase; print file is 0L529ELPT)
TOTAL

FUNCTION:
The TOTAL statement calculates the total of a numeric item in one of two ways:

1. Over an entire report.
2. Within a specific value of one or more controlling items (breaks).

FORMAT:
TOTAL [BY item [... item]]

DEFAULTS:
If you do not specify a break item (that is, BY item), the total is calculated over all records that pass through the TOTAL statement.

Unless you specify otherwise, each TOTAL statement calculates the specified total and prints it at the proper time.

DISCUSSION:
The format and use of the TOTAL statement are exactly the same as the other summary statements AVERAGE, MAXIMUM, MINIMUM, and TALLY.

If you specify a break item, the total is calculated and printed when IQL sees a change in the indicated item. If you specify more than one break item, TOTAL breaks if it sees a change in any of the indicated items.

You can specify more than one break item as long as each is minor to one preceding it. A total is calculated for each such break item. When a change in any break item occurs, a change is assumed to have occurred in each break item to the right of it in the TOTAL statement. The purpose of this is to prevent inadvertent overlap of minor break items over major ones.

Since totaling is an arithmetic operation, the item you total must be numeric. However the item(s) controlling the break can be any type.
IQL prints the total line on the report as a total title on the left and the total value on the right. IQL constructs the title for an item from the top and bottom dictionary titles and from the word TOTAL:. IQL constructs the title for a break item from the top and bottom dictionary titles of the break item and constructs the overall title for all the break items with the word OVERALL.

EXAMPLES:

We want to find out the total maintenance charge for each customer for equipment they have under maintenance contract. We also want to list the customer's name. Since the Equipment Dictionary is in sequence by Equip-No, we must first SORT it into sequence by Cust-No.

*OPEN EQUIPMENT CUSTOMER MODEL.*
*SORT BY EQUIP-CUST-NO.*
*FIND KEY2 = EQUIP-CUST-NO.*
*FIND KEY3 = EQUIP-MODEL-Nm.*
*TOTAL MODEL-CURR-MAINT-RATE BY EQUIP-CUST-NO.*
*IF FIRSTIME OR NEWGROUP EQUIP-CUST-NO*
*PRINT EQUIP-CUST-NO CERT-NAME.*

06/10/82

<table>
<thead>
<tr>
<th>CUST NO</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>LUCC</td>
</tr>
<tr>
<td>CUST NO 0001 CURR RATE TOTAL: $2146.00</td>
<td></td>
</tr>
<tr>
<td>0006</td>
<td>GARY SIMMONS</td>
</tr>
<tr>
<td>CUST NO 0006 CURR RATE TOTAL: $18.00</td>
<td></td>
</tr>
<tr>
<td>0007</td>
<td>EMORY ZIMMERS</td>
</tr>
</tbody>
</table>

(End query phase; print file is OJ30ELPT)
VSPACE

FUNCTION:
The VSPACE statement changes the vertical spacing of reports.

FORMAT:
VSPACE integer

DEFAULT:
If you do not use VSPACE in a query, IQL single spaces reports; that is, it acts as if you had used VSPACE 1.

DISCUSSION:
IQL physically vertical spaces just before it prints a line or the titles that correspond with the line.

You can use VSPACE at any point and repeatedly.

VSPACE only affects reports produced by PRINT; it does not affect output from DISPLAY.

The integer in VSPACE can be any reasonable value.

EXAMPLE:

VSPACE 3.
OPEN VENDOR.
PRINT VENDOR=NO VENDOR-NAME.

06/10/82

VENDOR VENDOR
NO NAME

AJ ANDERSON/JACOBSON

CAL CALCOMP, INC.

.
.
.

(End query phase; print file is OL729ELPT)
ENCLOSURE I

IQL ERROR MESSAGES

Error messages from IQL are preceded by either a question mark (?) which indicates an error that cannot be handled by IQL or a percent sign (%), which indicates either a warning message or an information message. When IQL encounters an (?) error, IQL displays the error message, terminates IQL, and returns you to the TOPS-20 environment. You will lose the current query area. When IQL encounters a (%) error, IQL displays the warning or information message and continues.

Following are explanations of the most common IQL (%) error messages:

%THERE IS NO QUERY TO STORE
There has to be a current query area before you can issue the STORE command.

%THERE IS NO QUERY TO REPLACE WITH
There has to be a current query area before you can issue the REPLACE command.

%THERE IS NO CURRENT QUERY TO RUN
When you issue the RUN command, there must either be a current query area or you must name a query that has been stored in the file of nonanalyzed queries.

%THERE IS NO CURRENT QUERY
You issued the EDIT command for a query that does not exist. You must use the WRITE command instead.

%query-name IS ALREADY STORED. PLEASE DELETE IT OR STORE IT UNDER A NEW NAME.
Once you have stored a query, you must use the REPLACE command to update it.

%item-name NOT DEFINED
%UNDEFINED DATA ITEM OR ELSE AN ALPHA VAR SPELLED INCORRECTLY You probably misspelled a data item or you have named a data item that is not in the dictionaries you have opened. You also could have neglected to precede and follow parentheses or logical operators with a space.

%NO ENDING PAREN.

%DO MANY RIGHT PAREN. You have unbalanced parentheses. Possibly you forgot to precede and follow the parenthesis symbol with a space.

%NO ENDING QUOTE You have an uneven number of quotation marks.

%DICTIONARY NOT FOUND FOR FILE The dictionary you named in the OPEN statement is not a valid dictionary name. Use the DICTIONARIES command to find the names of valid dictionaries.

%VERB EXPECTED

%IF STATEMENT HAS NO VERB PHRASE An action command such as PRINT, SORT, TALLY, TOTAL, AVERAGE, MAXIMUM, MINIMUM must follow an IF.

%INVALID RELATIONAL OPERATOR

%INVALID OPERATOR See the IF statement for the list of valid operators.

%VERB EXPECTS A DATA ITEM LIST Action statements such as PRINT, SORT, etc. must act on some data items.
APPENDIX IX

TRAINING EXERCISES
IX.1 TOPS-20 Training Exercises

ON-LINE EXERCISE 1

This exercise will be preceded by an explanation of the features of the terminal itself, pointing out the special purpose keys. To do this exercise, the trainees will have to know how to gain access to and to exit from the system. They will also need to know the procedure for creating and storing a file.

The trainees will need to understand the concept of the environments; TOPS-20, Edit, and Input, and to know which commands are used to enter and exit from each.

The most common error that novice users make is to neglect to exit the input environment. The commands they think they are issuing are actually becoming lines in the file they are creating.

The trainees should be able to accomplish this exercise with a modicum of errors.
ON-LINE EXERCISE 1

Creating a new file

1. Log in, and when you are connected to the system, create a new file called DEPT.DAT. Include the following items in this file:

   JOHNS JOSEPH       INDUSTRIAL ENGINEERING 58
   WILLIAMS SUSAN     MECHANICAL ENGINEERING 20
   MICHAELS JAMES     MATHEMATICS          35
   WILSON ANDREW      ENGLISH              19
   DAVIS MATTHEW      PHYSICS              21
   ROGERS DENNIS      ELECTRICAL ENGINEERING 15

2. After the last line of input, return to the Edit environment. Verify by printing all the lines.

3. Store the file DEPT.DAT on your disk without line numbers.

4. Logout.
LEHIGH UNIVERSITY, TOPS-20 Monitor 4(33)-3
@LOGIN NEF
Job 31 on TTY2 15-Jun-82 13:43:44
End of LOGIN.CMD.6

@EDIT DEPT.DAT

File not found, Creating New file
Input: DEPT.DAT.1
00100  JOHNS JOSEPH  INDUSTRIAL ENGINEERING  58
00200  WILLIAMS SUSAN  MECHANICAL ENGINEERING  20
00300  MICHAELS JMAES  MATHEMATICS    35
00400  WILSON ANDREW  ENGLISH         19
00500  DAVIS MATTHEW  PHYSICS          21
00600  ROGERS DENNIS  ELECTRICAL ENGINEERING  15
00700

2.

00700  $
*P:*
00100  JOHNS JOSEPH  INDUSTRIAL ENGINEERING  58
00200  WILLIAMS SUSAN  MECHANICAL ENGINEERING  20
00300  MICHAELS JMAES  MATHEMATICS    35
00400  WILSON ANDREW  ENGLISH         19
00500  DAVIS MATTHEW  PHYSICS          21
00600  ROGERS DENNIS  ELECTRICAL ENGINEERING  15
*

3.

*EU

[DEPT.DAT.1]
@

4.

@LOGOUT

Killed Job 31, User NEF, Account , TTY3,
15-Jun-82 13:41:56, Used 0:02:48 in 0:58:18
ON-LINE EXERCISE 2

This exercise deals with correcting and modifying a file that they have previously created. A main concept to be instilled in this introduction to the computer is that, the trainees are not expected to remember each command and its format. They only need to be aware that there is a command to do what they want and to know where to look to find out its exact format.
ON-LINE EXERCISE 2

Correcting and Modifying the File

1. Bring the DEPT.DAT file into the editor.

2. For practice in positioning yourself in a file:
   a. Print the first line of the file.
   b. Print lines 200:300
   c. Print the last line of the file.
   d. Print line 500
   e. Print the previous line.
   f. Print the next line.
   g. Print the current line.

3. If you have any typing errors to correct, make the desired changes.

4. Make the following revisions to the file:
   a. Change Susan William's department to CHEMISTRY.
   b. Replace the line for JAMES MICHAEL'S with the following:

   SMITH DAVID    PHILOSOPHY

   c. Change the spelling of DENNIS ROGER'S first name to DENIS.
   d. Delete line 300.
   e. Insert the following line as the first line in your file:

   GOLD DAVID    COMPUTER SCIENCE

   f. Print from the beginning of the file to line 400.
   g. Change all buildings numbered 58 to 38.
h. Add the following lines immediately after the line for SUSAN WILLIAMS:

STONE MARY HISTORY 35
BERNHARD CLARENCE FRENCH 22
ADAMS DIANE MATHEMATICS 35

5. Verify all revisions by printing the entire file.

6. Store the file and leave the Edit environment.

7. Logout.
1.

```plaintext
@EDIT DEPT.DAT
Edit: DEPT.DAT,1
*
2.a.
*P*
00100 JOHNS JOSEPH INDUSTRIAL ENGINEERING 58
2.b.
*P200:300
00200 WILLIAMS SUSAN MECHANICAL ENGINEERING 20
00300 MICHAELS JAMES MATHEMATICS 35
2.c.
*P*
00600 ROGERS DENNIS ELECTRICAL ENGINEERING 15
2.d.
*P500
00500 DAVIS MATTHEW PHYSICS 21
2.e.
*S*
00400 WILSON ANDREW ENGLISH 19
2.f.
*
00500 DAVIS MATTHEW PHYSICS 21
2.g.
*P*
00500 DAVIS MATTHEW PHYSICS 21
3.
*FJAMES*$

%Search fails
*P*
00100 JOHNS JOSEPH INDUSTRIAL ENGINEERING 58
*FJMAESS$
```

299
00300  MICHAELS JAMES MATHEMATICS 35
*JSJAMES$JAMES$.
00300  MICHAELS JAMES MATHEMATICS 35

4.a.
*FSUSANS

*Search fails
*P*.
00100  JOHNS JOSEPH INDUSTRIAL ENGINEERING 58
*F
00200  WILLIAMS SUSAN MECHANICAL ENGINEERING 20
*R*.
00200  WILLIAMS SUSAN CHEMISTRY 20
1 Lines (00200/1) deleted

4.b.
*FJAMES$
00300  MICHAELS JAMES MATHEMATICS 35
*R*.
00300  SMITH DAVID PHILOSOPHY 61
1 Lines (00300/1) deleted

4.c.
*FDENNIS$
00600  ROGERS DENNIS ELECTRICAL ENGINEERING 15
*SDENIS$DENIS $.
00600  ROGERS DENIS ELECTRICAL ENGINEERING 15
*SDENIS$DENIS $.
00600  ROGERS DENIS ELECTRICAL ENGINEERING 15

4.d.
*D300
1 Lines (00300/1) deleted

4.e.
*IS0
00050  GOLD DAVID COMPUTER SCIENCE 50

4.f.
*P*:
1400

300
GOLD DAVID  COMPUTER SCIENCE  58
JOHNS JOSEPH  INDUSTRIAL ENGINEERING  58
WILLIAMS SUSAN  CHEMISTRY  20
WILSON ANDREW  ENGLISH  19

*58$38$:*:

GOLD DAVID  COMPUTER SCIENCE  38
JOHNS JOSEPH  INDUSTRIAL ENGINEERING  38

WILLIAMS SUSAN  CHEMISTRY  20
STONE MARY  HISTORY  35
LIENHARD CLARE  FRENCH  22
ADAMS DIANE  MATHEMATICS  35

Page 1

GOLD DAVID  COMPUTER SCIENCE  38
JOHNS JOSEPH  INDUSTRIAL ENGINEERING  38
WILLIAMS SUSAN  CHEMISTRY  20
STONE MARY  HISTORY  35
LIENHARD CLARE  FRENCH  22
ADAMS DIANE  MATHEMATICS  35
WILSON ANDREW  ENGLISH  19
DAVIS MATTHEW  PHYSICS  21
ROGERS DENIS  ELECTRICAL ENGINEERING  15

*EU

[DEPT.DAT.2]

@LOGOUT
ON-LINE EXERCISE 3

This exercise will require that the trainees be able to distinguish TOPS-20 commands from Edit commands and know that they must be in the correct environment to perform each.

Several commands are introduced here. Again, the emphasis is placed on knowing that a command exists to do what they want to do and knowing where to find this information.
ON-LINE EXERCISE 3

Managing Files

1. Log in, and when you are connected to the system, create a new file RLDG.SEO containing the following information:

005 WHITAKER LABORATORY
006 MUDD BUILDING
007 SINCLAIR LABORATORY
014 CHRISTMAS-SAUCON HALL
019 PACKARD LABORATORY

2. List all the files in your directory.

3. List the contents of RLDG.SEO on your terminal screen.

4. Change the name of DEPT.NAT to DEPT.LOG.

5. Print the file DEPT.LOG on the printer.

6. Check to see if it is printing or if it is waiting to be printed.

7. Make a duplicate file of RLDG.SEO and call it BLDG2.SEO.

8. Delete BLDG.SEO from your directory.

9. OOPS, I told you the wrong file to delete, get it back and delete RLDG2.SEO instead.

10. List the files that are now in your directory.

11. See how much space you have left in your directory.

12. Free up the space for files you have deleted.

13. Check again to see how much space you have left.

14. Logout.
@LOGIN NEF
   Job 31 on TTY2 15-Jun-82 13:43:44
   End of LOGIN.CMD.6

@EDIT BLDG.SEQ

%No such filename, Creating New file
Input: BLDG.SEQ.1
00100 005 WHITACKER LABORATORY
00200 006 MUDD LABORATORY
00300 007 SINCLAIR LABORATORY
00400 014 CHRISTMAS-SAUCON HALL
00500 019 PACKARD LABORATORY
$
*EU

[BLDG.SEQ.2]

2.

@DIR
   PS:<NEF>
   BLDG.SEQ.3
   SEQ.6
   DEPT.DAT.2
   QAT.1
   LOGIN.CMD.6

   Total of 5 files

3.

@TYPE BLDG.SEQ
005 WHITACKER LABORATORY
006 MUDD BUILDING
007 SINCLAIR LABORATORY
014 CHRISTMAS-SAUCON HALL
019 PACKARD LABORATORY

4.

@RENAME DEPT.DAT DEPT.LOG
   DEPT.DAT.2 => DEPT.LOG.1 [OK]

5.

@PRINT DEPT.LOG
[Job DEPT Queued, Request-ID 585, Limit 9]

6.

@INFORMATION OUTPUT

Printer Queue:

<table>
<thead>
<tr>
<th>Job Name</th>
<th>Req#</th>
<th>Limit</th>
<th>User</th>
<th>On Unit:</th>
</tr>
</thead>
<tbody>
<tr>
<td>PASCAL</td>
<td>562</td>
<td>2232</td>
<td>-LUGRU</td>
<td>/Forms:THESIS</td>
</tr>
</tbody>
</table>

Started at 10:35:53, printed 309 of 2232 pages

JANET 15 72 JEWOOD
TEST 563 18 SBC
FMT69 567 18 -LUGRU
ASM69 568 18 -LUGRU
ECE142 578 9 -LUGRU
DGEN 582 9 IR.RCLYONS
REPORT 584 9 -LUGRU
DEPT 585 9 NEF
MS 572 45 -LUGRU
ADDED 576 27 TALLYBUILD
TOPMAN 564 72 NEF
RETFIL 565 72 IR.MJLENTO
ECE53 581 36 -LUGRU
ASM69 569 81 -LUGRU
ECE52 580 45 -LUGRU
ECE30 579 54 -LUGRU
REPORT 583 54 -LUGRU
IQLMAN 571 108 NEF
USERS 561 198 NEF
RETRV 570 234 IR.MKWILSON
RETRV 575 234 IR.RCLYONS
RETRV 577 234 IR.MKWILSON

There are 23 Jobs in the Queue (1 in Progress)

7.

@COPY BLDG.DASEQ BLDG2.SEQ
BLDG.SEQ.6 => BLDG2.SEQ.1 [OK]

8.

@DELETE BLDG.SEQ
BLDG.SEQ.6 [OK]

9.

305
@UNDELETE BLDG.SEQ
BLDG.SEQ.2 [OK]
BLDG.SEQ.3 [OK]
BLDG.SEQ.6 [OK]

10.

@DIR

PS:<NEF>'
BLDG.QEQ.3
  .SEQ.2,3,6
BLDG2.SEQ.1
DEPT.LOG.1
  .QAT.1
LOGIN.CMD.6

Total of 9 files

11.

@INFORMATION DISK
PS:<NEF>
273 Pages assigned, 256 in use, 13 deleted
500 Working pages, 500 Permanent pages allowed
47117 Pages free on PS:

12.

@EXPUNGE
PS:<NEF> [13 pages freed]

13.

@INFORMATION DISK
PS:<NEF>
260 Pages assigned
500 Working pages, 500 Permanent pages allowed
47133 Pages free on PS:

14.

@LOGOUT
IX.2 IQL Training Exercises - Day 1

ON-LINE EXERCISE 1

The purpose of this exercise is to get the trainees comfortable with the procedure for entering the IQL environment and leaving it. It also requires them to know which commands are used to list all the available dictionaries and the data items in a particular dictionary. The trainees will need to know how to do this if and when they forget a data item name or its spelling.
ON-LINE EXERCISE 1

1. Enter the IQL environment.

2. Find out what dictionaries are available to your directory.

3. Find out what data items are in the Building dictionary.

4. Leave the IQL environment.

5. Logout.
1.
@IQL
<QA>

2.

<QA> DICTIONARIES
Dictionaries in your directory:

<table>
<thead>
<tr>
<th>Dict Name</th>
<th>File Type</th>
<th>File-in Name</th>
<th>Direct</th>
<th>Rec</th>
<th>Blk</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUILDING</td>
<td>IS Dsk7</td>
<td>BLDG TNX</td>
<td></td>
<td>29</td>
<td>10</td>
</tr>
<tr>
<td>VENDOR</td>
<td>IS Dsk7</td>
<td>VENDOR TNX</td>
<td></td>
<td>29</td>
<td>10</td>
</tr>
<tr>
<td>EQUIPMENT</td>
<td>IS Dsk7</td>
<td>EQUIPMENT TNX</td>
<td></td>
<td>112</td>
<td>10</td>
</tr>
<tr>
<td>CUSTOMER</td>
<td>IS Dsk7</td>
<td>CUST TNX</td>
<td></td>
<td>52</td>
<td>10</td>
</tr>
<tr>
<td>MODEL</td>
<td>IS Dsk7</td>
<td>MODEL TNX</td>
<td></td>
<td>53</td>
<td>10</td>
</tr>
</tbody>
</table>

(End list of dictionaries)

3.

<QA> ITEMS BUILDING

<table>
<thead>
<tr>
<th>Dict Name</th>
<th>File Type</th>
<th>File-in Name</th>
<th>Direct</th>
<th>Rec</th>
<th>Blk</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUILDING</td>
<td>IS Dsk7</td>
<td>BLDG TNX</td>
<td></td>
<td>29</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item ID</th>
<th>Top Title</th>
<th>Bottom Title</th>
<th>1st #</th>
<th>T</th>
<th>S</th>
<th>Print</th>
</tr>
</thead>
<tbody>
<tr>
<td>DD*BLDG=REC</td>
<td>1</td>
<td>29</td>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD BLDG=NO</td>
<td>1</td>
<td>3</td>
<td>N</td>
<td>0</td>
<td>999</td>
<td></td>
</tr>
<tr>
<td>DD BLDG=NAME</td>
<td>4</td>
<td>25</td>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(End list of items)

4.
<QA>EXIT
   (End of IQL session)
EXIT
@
5.
@LOGOUT
ON-LINE EXERCISE 2

This exercise will require that the trainees understand the procedure for generating a simple query and running the associated report. They will have to create queries from two different dictionaries. The trainees will also need to understand the naming convention used by IQL for print files so that they are able to print them.

This exercise leads the trainees step by step. One error that may occur is that they may forget to use the WRITE command instead of EDTT as would be appropriate in the TOPS-20 Editor.

The number of errors evident by analyzing the "PHOTO" sessions coupled with the number of questions asked will give an indication as to whether the trainees are afraid or embarrassed to ask questions. It will also give an indication of how long they may have had to wait before getting their questions answered. Two approaches can be taken with regard to assistance during the on-line sessions. One is that the instructor wait until a question is asked and a second approach would be for the instructor to oversee the exercises monitoring the trainees' progress. Some trainees would be intimidated by having someone looking over their shoulder while they are working. A possible compromise would be to give the
trainees a few minutes alone, and then if questions are not being raised, make the rounds of all the terminals to inquire as to their progress.

In any event, the training program is not a static one. If it appears that the trainees have not grasped the concepts needed to do this exercise, a review with more elaboration is to be conducted before proceeding to the next section.
ON-LINE EXERCISE 2

1. Enter the IQL environment.

2. Create a query that will print the contents of the Building dictionary.

3. Generate this report using the RUN command.

4. Find out what data items are in the Customer dictionary.

5. Print the names and departments of all customers.

6. Leave the IQL environment.

7. Print the reports IQL created on the printer using the delete option.

8. Logout.
1.
@IQL
<QA>

2.

<QA>WRITE

%File not found, Creating New file
Input: QC010S.TMP.1
00100 OPEN BUILDING.
00200 PRINT BLDG-NO BLDG-NAME.
00300 $  
*EU

[QC010S.TMP.1]

3.

<QA>RUN

OPEN BUILDING.
PRINT BLDG-NO BLDG-NAME.

06/11/82

BLDG BLDG
NO NAME

005 WHITAKER LABORATORY
006 MUDD BUILDING
007 SINCLAIR LABORATORY
014 CHRISTMAS-SAUCON HALL
019 PACKARD LABORATORY
023 EDUCATION BLDGS & CLINIC
027 ALUMNI BUILDING
030 LINDERMAN LIBRARY
031 WILLIAMS HALL
035 DROWN HALL

(End query phase; print file is QC010ELPT)

<QA>

314
4.

<QA> ITEMS CUSTOMER

<table>
<thead>
<tr>
<th>Dict</th>
<th>File</th>
<th>File-in</th>
<th>Direct</th>
<th>Rec</th>
<th>Blk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Type</td>
<td>Name</td>
<td>Direct</td>
<td>Len</td>
<td>Fac</td>
</tr>
</tbody>
</table>

CUSTOMER IS Dsk7 CUST INX

<table>
<thead>
<tr>
<th>Item</th>
<th>Top</th>
<th>Bottom</th>
<th>1st</th>
<th>#</th>
<th>T</th>
<th>S</th>
<th>Printing</th>
</tr>
</thead>
<tbody>
<tr>
<td>DD CUST-FIRST-NAME</td>
<td>FIRST NAME</td>
<td>5</td>
<td>10</td>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD CUST-LAST-NAME</td>
<td>LAST NAME</td>
<td>15</td>
<td>15</td>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD CUST-REC</td>
<td></td>
<td>1</td>
<td>52</td>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD CUST-BLDG-NO</td>
<td>BLDG</td>
<td>30</td>
<td>3</td>
<td>N</td>
<td>0</td>
<td>999</td>
<td></td>
</tr>
<tr>
<td>DD CUST-NO CUST</td>
<td>NO</td>
<td>1</td>
<td>4</td>
<td>N</td>
<td>0</td>
<td>9999</td>
<td></td>
</tr>
<tr>
<td>DD CUST-ADDR</td>
<td>ADDRESS</td>
<td>30</td>
<td>7</td>
<td>A</td>
<td>0</td>
<td>XXX-XXXX</td>
<td></td>
</tr>
<tr>
<td>DD CUST-ROOM</td>
<td>ROOM</td>
<td>33</td>
<td>4</td>
<td>A</td>
<td>0</td>
<td>XXXX</td>
<td></td>
</tr>
<tr>
<td>DD CUST-DEPT</td>
<td>DEPARTMENT</td>
<td>37</td>
<td>15</td>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD CUST-NAME</td>
<td>NAME</td>
<td>5</td>
<td>25</td>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(End list of items)

5.

<QA> WRITE

% File not found, Creating New file
Input: QC010S_TMP.3
00100 OPEN CUSTOMER.
00200 PRINT CUST-NAME CUST-DEPT.
00300 $
[QC010S.TMP.3]

<QA>RUN

OPEN CUSTOMER.
PRINT.CUST-NAME CUST-DEPT.

06/11/82

NAME              DEPARTMENT
GARY              SIMMONS        COMPUTING CTR
EMORY             ZIMMERS        CSCR
BERRY             RICHARDS       LINDERMAN LIB
RICHARD           HERTZBERG     MET/MTRL ENGR
GEORGE            SIH           MECH ENGR/MCHNS
STAN              JOHNSON        MECH ENGR/MCHNS
KEN               KLEIER         CHEMISTRY
RAYMOND           BELL           HUMAN DEVELOP.
LEE               TUSCHER        EDUCATION
GERALD            SMITH          CAM
FRANK             LUH            ACCOUNTING/LAW
ARTHUR            HUMPHREY      PROVOST OFFICE
ALISTAIR          MAC PHERSON    MECH ENGR/MCHNS
DONALD            TALHELM       E.E.
ROBERT            WEI            MECH ENGR/MCHNS

(End query phase; print file is QL110E.LPT)

<QA>

6.

<QA>EXIT
(End of IQL session)

EXIT
@

7.

@PRINT QL110E.LPT/DELETE
[Job QL110E Queued, Request-ID 5A8, Limit 9]
@PRINT QL010E.LPT/DELETE
[Job QL010E Queued, Request-ID 5A9, Limit 9]
ON-LINE EXERCISE 3

This exercise introduces the SORT statement and the IF statement. It also introduces the concepts of storing a query for later use. It will be necessary for the trainees to understand the basic difference between the TOPS-20 editor exit command EU, and the IQL editor exit command EU.

This exercise requires the trainees to think on his own as to which procedures they should use and where the information they want is located. They will need to remember which data items are in which dictionary or know how to find this information.

Having completed this exercise, the trainees should have a basic grasp of IQL and be able to generate many queries. The analysis of the 'PHOTO' sessions should indicate whether the amount of information that the trainees can absorb has been underestimated or overestimated. It should also give an indication as to whether the trainees should have been given more practice in simple queries before expecting them to know how to manipulate queries with such commands as STORE, REPLACE, and DELETE.
ON-LINE EXERCISE 3

1. Enter the IQL environment.

2. Generate a listing of all customers in sequence by last name and then first name. Along with their names, print their department and building number.

3. Modify this query so that it also prints the customer's room no.

4. Run the modified query.

5. Store this query as a non-analyzed query called CUSTLIST.

6. Generate a query that will list the equipment number, customer number, building number, and room number of all 'DEC' (vendor number of 'DEC') equipment.

7. Store this query as DECLIST.

8. Change the CUSTLIST query so that it prints only the customers' first and last names.

9. Make these revisions permanent.

10. List all the queries available to your directory.

11. Delete the CUSTLIST query.

12. Leave the IQL environment.

13. Delete the print files created during this session.

14. Logout.
1.
@IQL
<QA>
2.
<QA>WRITE

%File not found, Creating New file
Input: QC010S.TMP.5

00100 OPEN CUSTOMER.
00200 SORT BY CUST-LAST-NAME CUST-FIRST-NAME.
00300 PRINT CUST-LAST-NAME CUST-FIRST-NAME
00400 CUST-DEPT CUST-BLDG-NO.
00500 $ *EU

[QC010S.TMP.5]

<QA>RUN

OPEN CUSTOMER.
SORT BY CUST-LAST-NAME CUST-FIRST-NAME.
PRINT CUST-LAST-NAME CUST-FIRST-NAME
CUST-DEPT CUST-BLDG-NO.

06/11/82

<table>
<thead>
<tr>
<th>LAST NAME</th>
<th>FIRST NAME</th>
<th>DEPARTMENT</th>
<th>BLDG NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>BELL</td>
<td>RAYMOND</td>
<td>HUMAN DEVELOP.</td>
<td>023</td>
</tr>
<tr>
<td>HERTZBERG</td>
<td>RICHARD</td>
<td>MET/MTRL ENGR</td>
<td>005</td>
</tr>
<tr>
<td>HUMPHREY</td>
<td>ARTHUR</td>
<td>PROFVOST OFFICE</td>
<td>027</td>
</tr>
<tr>
<td>JOHNSON</td>
<td>STAN</td>
<td>MECH ENGR/MCHNS</td>
<td>019</td>
</tr>
<tr>
<td>KLIER</td>
<td>KEN</td>
<td>CHEMISTRY</td>
<td>007</td>
</tr>
<tr>
<td>LUCC</td>
<td>FRANK</td>
<td>COMPUTING CTR</td>
<td>019</td>
</tr>
<tr>
<td>LUH</td>
<td></td>
<td>ACCOUNTING/LAW</td>
<td>035</td>
</tr>
<tr>
<td>MAC PHERSON</td>
<td>ALISTAIR</td>
<td>MECH ENGR/MCHNS</td>
<td>019</td>
</tr>
<tr>
<td>RICHARDS</td>
<td>BERRY</td>
<td>LINDERMAN LIB</td>
<td>030</td>
</tr>
<tr>
<td>SIH</td>
<td>GEORGE</td>
<td>MECH ENGR/MCHNS</td>
<td>019</td>
</tr>
<tr>
<td>SIMMONS</td>
<td>GARY</td>
<td>CSCR</td>
<td>007</td>
</tr>
<tr>
<td>SMITH</td>
<td>GERALD</td>
<td>CAM</td>
<td>030</td>
</tr>
<tr>
<td>TALHELM</td>
<td>DONALD</td>
<td>F.E.</td>
<td>019</td>
</tr>
<tr>
<td>TUSCHER</td>
<td>LEE</td>
<td>EDUCATION</td>
<td>023</td>
</tr>
<tr>
<td>WEI</td>
<td>ROBERT</td>
<td>MECH ENGR/MCHNS</td>
<td>007</td>
</tr>
</tbody>
</table>
<QA>EDIT
Edit: QC010S.TMP.5
*FPRINTS
00300 PRINT CUST-LAST-NAME CUST-FIRST-NAME
00400 CUST-DEPT CUST-BLDG-NO.
R.
00300 CUST-DEPT CUST-BLDG-NO CUST-ROOM.
1 Lines (00300/1) deleted
*EU
[QC010S.TMP.6]

<QA> RUN
OPEN CUSTOMER.
SORT BY CUST-LAST-NAME CUST-FIRST-NAME.
PRINT CUST-LAST-NAME CUST-FIRST-NAME
CUST-DEPT CUST-BLDG-NO CUST-ROOM.

06/11/82

<table>
<thead>
<tr>
<th>LAST NAME</th>
<th>FIRST NAME</th>
<th>DEPARTMENT</th>
<th>BLDG NO</th>
<th>ROOM NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>BELL</td>
<td>RAYMOND</td>
<td>HUMAN DEVELOPM.</td>
<td>023</td>
<td>524</td>
</tr>
<tr>
<td>HERTZBERG</td>
<td>RICHARD</td>
<td>MTL/MTRL ENGR</td>
<td>005</td>
<td>454</td>
</tr>
<tr>
<td>HUMPHREY</td>
<td>ARTHUR</td>
<td>PROVOST OFFICE</td>
<td>027</td>
<td>204F</td>
</tr>
<tr>
<td>JOHNSON</td>
<td>STAN</td>
<td>MECH ENGR/MCHNS</td>
<td>019</td>
<td>551</td>
</tr>
<tr>
<td>KLIER</td>
<td>KEN</td>
<td>CHEMISTRY</td>
<td>007</td>
<td>315</td>
</tr>
<tr>
<td>LUCC</td>
<td>FRANK</td>
<td>COMPUTING CTR</td>
<td>019</td>
<td>154</td>
</tr>
<tr>
<td>LUH</td>
<td>FRANK</td>
<td>ACCOUNTING/LAW</td>
<td>035</td>
<td>203</td>
</tr>
<tr>
<td>MAC PHERSON</td>
<td>ALISTAIR</td>
<td>MECH ENGR/MCHNS</td>
<td>019</td>
<td>558</td>
</tr>
<tr>
<td>RICHARDS</td>
<td>BERRY</td>
<td>LINDERMAN LIB</td>
<td>030</td>
<td>100</td>
</tr>
<tr>
<td>SIH</td>
<td>GEORGE</td>
<td>MECH ENGR/MCHNS</td>
<td>019</td>
<td>451A</td>
</tr>
<tr>
<td>SIMMONS</td>
<td>GARY</td>
<td>CSCP</td>
<td>007</td>
<td>305B</td>
</tr>
</tbody>
</table>
<QA>
5.

<QA>STORE CUSTLIST
(CUSTLIST stored)

<QA>
6.

<QA>WRITE

%file not found, Creating new file
Input: QC010S.TMP.8

00100 OPEN EQUIPMENT.
00200 IF EQUIP-VENDOR-NO = 'DEC'
00300 PRINT EQUIP-NO EQUIP-CUST-NO
00400 EQUIP-BLDG-NO EQUIP-ROOM.
00500 $ *EU

[QC010S.TMP.8]

<QA>OPEN EQUIPMENT.
IF EQUIP-VENDOR-NO = 'DEC'
PRINT EQUIP-NO EQUIP-CUST-NO
EQUIP-BLDG-NO EQUIP-ROOM.

06/11/82

<table>
<thead>
<tr>
<th>EQUIPMENT NO</th>
<th>CUST NO</th>
<th>BLDG NO</th>
<th>ROOM NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEC01-15303</td>
<td>0001</td>
<td>019</td>
<td>200</td>
</tr>
<tr>
<td>DEC01-15411</td>
<td>0001</td>
<td>006</td>
<td>489</td>
</tr>
<tr>
<td>DEC01-28785</td>
<td>0023</td>
<td>007</td>
<td></td>
</tr>
<tr>
<td>DEC01-33912</td>
<td>0029</td>
<td>030</td>
<td>7</td>
</tr>
<tr>
<td>DEC01-54359</td>
<td>0001</td>
<td>014</td>
<td>B10</td>
</tr>
</tbody>
</table>

PAGE 1
(End query phase; print file is 0L410ELPT)

<QA>

7.

<QA>STORE DECLIST
(DECLIST stored)

<QA>

8.

<QA>EDIT CUSTLIST
**CUSTLIST

Edit: QC010S.TMP.9
*FPRINT$
00300 PRINT CUST-LAST-NAME CUST-FIRST-NAME CUST-DEPT
*SCUST-DEPT**.
00300 PRINT CUST-LAST-NAME CUST-FIRST-NAME

00400 CUST-BLDG-NO CUST-ROOM
*D.
1 Lines (00400/1) deleted
*EU

[QC010S.TMP.10]
<QA> RUN  
 **CUSTLIST  

OPEN CUSTOMER.  
SORT BY CUST-LAST-NAME CUST-FIRST-NAME.  
PRINT CUST-LAST-NAME CUST-FIRST-NAME  

06/11/82  

LAST NAME   FIRST NAME  
BELL         RAYMOND  
HERTZBERG    RICHARD  
HUMPHREY     ARTHUR  
JOHNSON      STAN  
KLIER        KEN  
LUCC         FRANK  
LUH          ALISTAIR  
MAC PHERSON  BERRY  
RICHARDS     GARY  
SIH          GEORGE  
SIMMONS      GERALD  
SMITH        DONALD  
TALHELM      LEE  
TUSCHER      ROBERT  
WEI          EMORY  

(End query phase; print file is OL510ELPT)  

<QA>  

9.  

<QA> REPLACE CUSTLIST  
(CUSTLIST replaced)  

<QA>  

10.  

<QA> QUERIES  
Queries stored in your directory:  

IQLSAMPLE   CUSTLIST   DECLIST  

324
(End list of stored queries)

11.

<QA>DELETE CUSTLIST
(CUSTLIST deleted)

<QA>

12.

<QA>EXIT
(End of IQL session)

EXIT
@

13.

@DIR Q*.LPT

PS;<NEF>
QL010E.LPT.1
QL110E.LPT.1
QL210E.LPT.2
QL310E.LPT.2
QL410E.LPT.1
QL510E.LPT.2

Total of 6 files
@DELETE Q*.LPT
QL010E.LPT.1 [OK]
QL110E.LPT.1 [OK]
QL210E.LPT.2 [OK]
QL310E.LPT.2 [OK]
QL410E.LPT.1 [OK]
QL510E.LPT.2 [OK]
ON-LINE EXERCISE 4

Exercise 4 requires that the trainees understand when and how they are to extract information from two or three dictionaries during the same query.

Step #6 may be too complicated for the casual user. It requires that they understand how the SORT statement works and that it signifies a stage in a query. Having just discussed this during the instructional session, they may not have difficulty, but it is doubtful that they will remember the sequence of steps necessary to sort on a data item in a secondary or tertiary dictionary. It is for this reason, that an example of this procedure is included in the End-Users manual.

Scoring this exercise based on the 'PHOTO' sessions alone will be difficult. What will be more significant will be the amount of instructor assistance that is required while they are doing this exercise. An analysis of this exercise may indicate that this concept should be left out of an introductory class.

This concludes the first day's exercise sessions.
ON-LINE EXERCISE 4

1. After logging into the system, enter the IQL environment.

2. Modify the DECLIST query to print the owner’s name as well as his Cust-No. (This will require having two dictionaries open at one time).

3. Replace the non-analyzed query DECLIST with this modified query.

4. Run the modified query without listing it.

5. Create and store a query that will list all 'TUC' equipment that is under maintenance contract (Equip-Service = 'M') giving the vendor number, model name, serial number, customer number, and customer name. Call this query TUCLIST.

6. Modify this query to be in sequence by vendor number and descending model name.

7. Make the change permanent.

8. Run the query.

9. Leave the IQL environment.

10. Logout.
<QA>

**DECLIST

Edit: QC010S.TMP.12
00100 OPEN EQUIPMENT.
*R.
00100 OPEN EQUIPMENT CUSTOMER.
1 Lines (00100/1) deleted
00200 IF EQUIP-VENDOR-NO = 'DEC'
00300 PRINT EQUIP-NO EQUIP-CUST-NO
*R.
00300 PRINT EQUIP-NO EQUIP-CUST-NO CUST-NAME
1 Lines (00300/1) deleted
*$
00200 IF EQUIP-VENDOR-NO = 'DEC'
*I.
00250 FIND KEY2 = EQUIP-CUST-NO
*EU

[QC010S.TMP.13]

<QA>RUN
**DECLIST

OPEN EQUIPMENT CUSTOMER.
IF EQUIP-VENDOR-NO = 'DEC'
FIND KEY2 = EQUIP-CUST-NO
PRINT EQUIP-NO EQUIP-CUST-NO CUST-NAME
EQUIP-BLDG-NO EQUIP-ROOM.

06/15/82

<table>
<thead>
<tr>
<th>EQUIPMENT</th>
<th>CUST</th>
<th>NAME</th>
<th>BLDG</th>
<th>ROOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>NO</td>
<td>NAME</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>DEC01-15303</td>
<td>0001</td>
<td>L.UCC</td>
<td>019</td>
<td>200</td>
</tr>
<tr>
<td>DEC01-15411</td>
<td>0001</td>
<td>L.UCC</td>
<td>006</td>
<td>489</td>
</tr>
<tr>
<td>DEC01-28785</td>
<td>0023</td>
<td>K.LIER</td>
<td>007</td>
<td></td>
</tr>
<tr>
<td>DEC01-33912</td>
<td>0029</td>
<td>SMITH</td>
<td>030</td>
<td>7</td>
</tr>
<tr>
<td>DEC01-54359</td>
<td>0001</td>
<td>L.UCC</td>
<td>014</td>
<td>B10</td>
</tr>
<tr>
<td>DEC01-54366</td>
<td>0001</td>
<td>L.UCC</td>
<td>031</td>
<td>013</td>
</tr>
</tbody>
</table>
(End query phase; print file is OL010ELPT)

<QA>

3.

<QA>REPLACE DECLIST
(DECLIST replaced)

<QA>

4.

<QA>RUN DECLIST NDLIST
**DECLIST

06/15/82

EQUIPMENT NO CUST NO NAME BLDG ROOM NO NO

329
DEC01-15303  0001  LUCC  019 200
DEC01-15411  0001  LUCC  006 489
DEC01-28785  0023  KEN  KLIER  007
DEC01-33912  0029  GERALD  SMITH  030 7
DEC01-54359  0001  LUCC  014 B10
DEC01-54366  0001  LUCC  031 013
DEC01-54838  0028  LEE  TUSCHER  023 520
DEC01-73007  0001  LUCC  035 208
DEC01-75509  0001  LUCC  019 154
DEC01-84323  0001  LUCC  011 A3
DEC01-92404  0001  LUCC  027
DEC01-92413  0001  LUCC  035 208
DEC01-92417  0001  LUCC  014 B10
DEC01-A0420  0001  LUCC  030
DEC01-PN23677  0026  RAYMOND  RELL  023 520
DEC01-PN32461  0001  LUCC  035 208
DEC01-PN32853  0001  LUCC  014 B10
DEC01-WF65445  0001  LUCC  035 208
DEC01-WF05760  0001  LUCC  027
DEC02-08457  0013  RICHARD  HERTZBERG  005 454
DEC02-16345  0006  GARY  SIMMONS  007 305B
DEC02-32663  0010  BERRY  RICHARDS  030 100
DEC02-34170  0001  LUCC  027
DEC02-38012  0019  GEORGE  SIH  019 451A
DEC02-39764  0001  LUCC  019 154
DEC02-39969  0001  LUCC  019 154
DEC03-11465  0001  LUCC  031 013
DEC03-11535  0001  LUCC  014 B10
DEC03-PN09657  0007  EMORY  ZIMMERS  019 222
DEC03-WF24140  0040  FRANK  LUH  035 203
DEC04-AG21996  0001  LUCC  019 118
DEC05-AG01199  0058  ROBERT  WEI  007 326
DEC06-AG10100  0007  EMORY  ZIMMERS  019 122
DEC08-AG14453  0048  ARTHUR  HUMPHREY  005 561

(End query phase; print file is OL110ELPT)

<QA>

5.

<QA>WRITE

%File not found, Creating New file
Input: QC010S.TMP.16
00100  OPEN EQUIPMENT CUSTOMER MODEL.
00200  IF EQUIP-VENDOR-NO = 'LUCC' AND
00300    EQUIP-SERVICE = 'M'
00400    FIND KEY2 = EQUIP-CUST-NO

330
FIND KEY3 = EQUIP-MODEL-NO
PRINT EQUIP-VENDOR-NO MODEL-NAME EQUIP-SERIAL-NO EQUIP-CUST-NO CUST-NAME.
$ *EU

[QC010S.TMP.16]

<QA>RUN

OPEN EQUIPMENT CUSTOMER MODEL.
IF EQUIP-VENDOR-NO = 'TUC' AND EQUIP-SERVICE = 'M'
FIND KEY2 = EQUIP-CUST-NO
FIND KEY3 = EQUIP-MODEL-NO
PRINT EQUIP-VENDOR-NO MODEL-NAME EQUIP-SERIAL-NO EQUIP-CUST-NO CUST-NAME.

06/11/82

<table>
<thead>
<tr>
<th>VENDOR NO</th>
<th>MODEL</th>
<th>SERIAL-NO</th>
<th>CUST NO</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>TUC</td>
<td>1630/2400</td>
<td>424</td>
<td>0001</td>
<td>LUCC</td>
</tr>
<tr>
<td>TUC</td>
<td>1650/2400</td>
<td>9700</td>
<td>0001</td>
<td>LUCC</td>
</tr>
</tbody>
</table>

(End query phase; print file is 0L210ELPT)

<QA>STORE TUCLIST
(TUCLIST stored)

<QA>

6.

<QA>EDIT TUCLIST
**TUCLIST

Edit: QC010S.TMP.17
00100 OPEN EQUIPMENT CUSTOMER MODEL.
00200 IF EQUIP-VENDOR-NO = 'TUC' AND EQUIP-SERVICE = 'M'
00300 *I..10
00310 FIND KEY3 = EQUIP-MODEL-NO
00320 SORT BY EQUIP-VENDOR-NO DESCENDING MODEL-NAME.
00330 $ *EU

[QC010S.TMP.18]
7.

```sql
REPLACE TUCLIST
(TUCLIST replaced)
```

8.

```sql
RUN
**TUCLIST

OPEN EQUIPMENT CUSTOMER MODEL.
IF EQUIP-VENDOR-NO = 'TUC' AND
EQUIP-SERVICE = 'M'
FIND KEY3 = EQUIP-MODEL-NO
SORT BY EQUIP-VENDOR-NO DESCENDING MODEL-NAME.
FIND KEY2 = EQUIP-CUST-NO
FIND KEY3 = EQUIP-MODEL-NO
PRINT EQUIP-VENDOR-NO MODEL-NAME EQUIP-SERIAL-NO
   EQUIP-CUST-NO CUST-NAME.

06/11/82
```

<table>
<thead>
<tr>
<th>VENDOR NO</th>
<th>MODEL</th>
<th>SERIAL-NO</th>
<th>CUST NO</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>TUC</td>
<td>1650/2400</td>
<td>970n</td>
<td>0001</td>
<td>LUCC</td>
</tr>
<tr>
<td>TUC</td>
<td>1630/2400</td>
<td>424</td>
<td>0001</td>
<td>LUCC</td>
</tr>
</tbody>
</table>

(End query phase; print file is 0L310ELPT)

9.

```sql
EXIT
(End of IQL session)
```

EXIT
ON-LINE EXERCISE 1

This exercise tests the trainees' grasp of the report functions, TALLY, MAXIMUM, MINIMUM and TOTAL. It also introduces the commands for dealing with analyzed queries, SAVE and EXECUTE. Discussion of these commands was postponed until the second day so that there would be less likelihood of confusing STORE with SAVE and RUN with EXECUTE.

The trainees will need to remember the procedure for accessing more than one dictionary in a query. They will also have to understand that prior to using report functions, they must make certain that the dictionary is sorted in the same sequence as the break items they specify in the report function statements.

This exercise will also show how much they have retained from the previous session.
ON-LINE EXERCISE 1

1. After gaining access to the system, enter the IQL environment.

2. List all the queries available to your directory.

3. Find out how many 'DFC01' pieces of equipment there are.

4. What is the highest customer number that has been used?

5. In order to save computer time, issue the command that will analyze the TUCLIST query.

6. Generate the report using the command that will use the analyzed query.

7. What is the total monthly maintenance charge for LUCC owned equipment (Equip-Cust-No = 1) that is under maintenance (Equip-Service = 'M')?

8. Keep this query in the file of non-analyzed queries as LUCCRATE.

9. How many pieces of equipment are there in the equipment table of each model type?

10. Leave the IQL environment.

11. List all the files in your directory with a filetype of .INQ.

12. Delete all unneeded print files.

13. Logout
1.
@IQI

<QA>

2.

<QA>QUERIES
Queries stored in your directory:

IQLSAMPLE               DECLIST               TUCLIST

(End list of stored queries)

<QA>

3.

<QA>WRITE

%File not found, Creating new file
Input: QC010S,TMP.20
00100 OPEN EQUIPMENT.
00200 IF EQUIP-MODEL-NO = 'DEC01'
00300 TALLY EQUIP-NO.
00400 $ *EU

[QC010S,TMP.20]

<QA>RUN

OPEN EQUIPMENT.
IF EQUIP-MODEL-NO = 'DEC01'
TALLY EQUIP-NO.

06/11/82

OVERALL EQUIPMENT NO TALLY: 19

(End query phase; print file is OL510ELPT)

<QA>

4.

<QA>WRITE

335
%File not found, Creating New file
Input: QC010S.TMP.23
00100 OPEN CUSTOMER.
00200 MAXIMUM CUST-NO.
00300 $
*EU

[QC010S.TMP.23]

<QA> RUN

OPEN CUSTOMER.
MAXIMUM CUST-NO.

06/11/82  PAGE 1

OVERALL CUST NO MAX: 0058

(End query phase; print file is OL610ELPT)

<QA>

5.

<QA> SAVE TUCLIST
**TUCLIST

OPEN EQUIPMENT CUSTOMER MODEL.
IF EQUIP-VENDOR-NO = 'TUC' AND EQUIP-SERVICE = 'M'
FIND KEY3 = EQUIP-MODEL-NO
SORT BY EQUIP-VENDOR-NO DESCENDING MODEL-NAME.
FIND KEY2 = EQUIP-CUST-NO
FIND KEY3 = EQUIP-MODEL-NO
PRINT EQUIP-VENDOR-NO MODEL-NAME EQUIP-SERIAL-NO
  EQUIP-CUST-NO CUST-NAME.

<QA>

6.

<QA> EXECUTE TUCLIST

06/11/82  PAGE 1

VENDOR NO  MODEL  SERIAL-NO  CUST NO  NAME

336
(End query phase; print file is OL710ELPT)

<QA>

7.

<QA>WRITE

<QA>WRITE

%File not found, Creating New file
Input: QC010S.TMP.28
00100  OPEN EQUIPMENT MODEL.
00200  IF EQUIP-CUST-NO = 1 AND EQUIP-SERVICE = 'M'
00300  FIND KEY2 = EQUIP-MODEL-NO
00400  TOTAL MODEL-CURR-MAINT-RATE.
00500  $*
EU

[QC010S.TMP.28]

<QA>

<QA>RUN

**LUCCLIST

OPEN EQUIPMENT MODEL.
IF EQUIP-CUST-NO = 1 AND EQUIP-SERVICE = 'M'
FIND KEY2 = EQUIP-MODEL-NO
TOTAL MODEL-CURR-MAINT-RATE.

06/11/82

OVERALL CURR RATE     TOTAL: $2087.00

(End query phase; print file is OL910ELPT)

<QA>

8.

<QA>STORE LUCCLIST

(LUCCLIST stored)

<QA>
9.

<QA>WRITE

%File not found, Creating New file
Input: QC010S.TMP.30
00100 OPEN EQUIPMENT.
00200 TALLY EQUIP-NO BY EQUIP-MODEL-NO.
00300 $
*EU

[QC010S.TMP.30]

<QA>RUN

OPEN EQUIPMENT.
TALLY EQUIP-NO BY EQUIP-MODEL-NO.

06/11/82 PAGE 1

MODEL NO AJ 01 EQUIPMENT NO TALLY: 6
MODEL NO CAL01 EQUIPMENT NO TALLY: 2
MODEL NO DECO1 EQUIPMENT NO TALLY: 19
MODEL NO DECO2 EQUIPMENT NO TALLY: 7
MODEL NO DECO3 EQUIPMENT NO TALLY: 4
MODEL NO DECO4 EQUIPMENT NO TALLY: 1
MODEL NO DECO5 EQUIPMENT NO TALLY: 1
MODEL NO DECO6 EQUIPMENT NO TALLY: 1
MODEL NO DECO8 EQUIPMENT NO TALLY: 1
MODEL NO IBM01 EQUIPMENT NO TALLY: 18
MODEL NO IBM02 EQUIPMENT NO TALLY: 4
MODEL NO IBM03 EQUIPMENT NO TALLY: 2
MODEL NO IBM04 EQUIPMENT NO TALLY: 1
MODEL NO ICC01 EQUIPMENT NO TALLY: 2
MODEL NO ICC02 EQUIPMENT NO TALLY: 2
MODEL NO IT02 EQUIPMENT NO TALLY: 11
MODEL NO PC001 EQUIPMENT NO TALLY: 3
MODEL NO TCP01 EQUIPMENT NO TALLY: 3
MODEL NO TEC01 EQUIPMENT NO TALLY: 1
MODEL NO TEC02 EQUIPMENT NO TALLY: 1
MODEL NO TEC03 EQUIPMENT NO TALLY: 1
MODEL NO TEC04 EQUIPMENT NO TALLY: 2
MODEL NO TUC03 EQUIPMENT NO TALLY: 3
MODEL NO TUC04 EQUIPMENT NO TALLY: 3
MODEL NO TUC05 EQUIPMENT NO TALLY: 3

(End query phase; print file is OLA10ELPT)

<QA>
10.

<QA>EXIT
   (End of IQL session)

EXIT

@

11.

@DIR *.INQ

   PS:<NEF>
   TUCIS.INQ.1

@

12.

@DELETE Q*.LPT
   QL010E.LPT.2 [OK]
   QL110E.LPT.2 [OK]
   QL210E.LPT.3 [OK]
   QL310E.LPT.4 [OK]
   QL410E.LPT.2 [OK]
   QL510E.LPT.3 [OK]
   QL610E.LPT.1 [OK]
   QL710E.LPT.2 [OK]
   QL810E.LPT.1 [OK]
   QL910E.LPT.1 [OK]
   QLA10E.LPT.1 [OK]
ON-LINE EXERCISE 2

This exercise is much simpler than the previous ones. It gives the trainees practice in using the formatting statements. The only complexity lies in remembering that to do Step #5, that they must first sort the equipment dictionary into sequence by Cust-No. They must also remember how to access multiple dictionaries.
1. After gaining access to the system, enter the IQL environment.

2. Modify the TUCLIST query to create only the print file and not display on the terminal.

3. Make this change permanent.

4. Create a report called 'Vendor Listing' that lists the number and name of all vendors. Double space this report and leave 10 spaces between columns. Start the report in column 5.

5. Find the total maintenance rate for each customer.

6. Leave the IQL environment.

7. Logout.
1.
AQQL

2.

<QA>EDIT TUCLIST
**TUCLIST

Edit: QC010S.TMP.32
*150
*00050 DISPLAY OFF.
*%
No such line(s)
*EU

[QC010S.TMP.33]

<QA>RUN
**TUCLIST

DISPLAY OFF.
OPEN EQUIPMENT CUSTOMER MODEL.
 IF EQUIP-VENDOR-NO = 'TUC' AND EQUIP-SERVICE = 'M'
  FIND KEY3 = EQUIP-MODEL-NO
  SORT BY EQUIP-VENDOR-NO DESCENDING MODEL-NAME.
  FIND KEY2 = EQUIP-CUST-NO
  FIND KEY3 = EQUIP-MODEL-NO
  PRINT EQUIP-VENDOR-NO MODFL.-NAME EQUIP-SERIAL-NO
    EQUIP-CUST-NO CUST-NAME.

(End query phase; print file is OL010ELOPT)

3.

<QA>REPLACE TUCLIST
(TUCLIST replaced)

4.

<QA>WRITE
%%File not found, Creating New file
Input: QC010S.TMP.35
00100  HEADING 'VENDOR LISTING'
00200  VSPACE 2.
00300  HSPACE 10.
00400  LMARGIN 5.
00500  OPEN VENDOR.
00600  PRINT VENDOR-NO VENDOR-NAME.
00700  $  
*EU

[QC010S.TMP.35]

<QA>RUN

HEADING 'VENDOR LISTING'.
VSPACE 2.
HSPACE 10.
LMARGIN 5.
OPEN VENDOR.
PRINT VENDOR-NO VENDOR-NAME.

<table>
<thead>
<tr>
<th>06/11/82</th>
<th>VENDOR LISTING</th>
<th>PAGE 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>VENDOR NO</td>
<td>VENDOR NAME</td>
<td></td>
</tr>
<tr>
<td>AJ</td>
<td>ANDERSON/JACOBSON</td>
<td></td>
</tr>
<tr>
<td>CAL</td>
<td>CALCOMP, INC.</td>
<td></td>
</tr>
<tr>
<td>DEC</td>
<td>DIGITAL EO. CORP.</td>
<td></td>
</tr>
<tr>
<td>HAZ</td>
<td>HAZELTINE INC.</td>
<td></td>
</tr>
<tr>
<td>IBM</td>
<td>INTERNATIONAL BUS. MACH.</td>
<td></td>
</tr>
<tr>
<td>ICC</td>
<td>ICC</td>
<td></td>
</tr>
<tr>
<td>ITT</td>
<td>ITT</td>
<td></td>
</tr>
<tr>
<td>PCO</td>
<td>PRECISION COMPONENTS</td>
<td></td>
</tr>
<tr>
<td>SOR</td>
<td>SORBUS, INC.</td>
<td></td>
</tr>
<tr>
<td>TCP</td>
<td>TELEX COMPUTER PRODUCTS</td>
<td></td>
</tr>
<tr>
<td>TEC</td>
<td>TECHTRAN INDUSTRIES, INC.</td>
<td></td>
</tr>
</tbody>
</table>

343
TSP
TIME SHARING PERIPHERALS
TUC
TUCK

(End query phase; print file is OL110ELPT)

<QA>

5.

<QA>WRITE

% File not found, Creating New file
Input: QC010S.TMP.37
00100 OPEN EQUIPMENT MODEL CUSTOMER.
00200 FIND KEY2 = EQUIP-MODEL-NO.
00300 FIND KEY3 = EQUIP-CUST-NO.
00400 TOTAL MODEL-CURR-MAINT-RATE BY EQUIP-CUST-NO.
00500 $  
*1100 00150 SORT BY EQUIP-CUST-NO.
*EU

[QC010S.TMP.37]

<QA>RUN

OPEN EQUIPMENT MODEL CUSTOMER.
SORT BY EQUIP-CUST-NO.
FIND KEY2 = EQUIP-MODEL-NO.
FIND KEY3 = EQUIP-CUST-NO.
TOTAL MODEL-CURR-MAINT-RATE BY EQUIP-CUST-NO.

06/11/82

<table>
<thead>
<tr>
<th>CUST NO</th>
<th>CURR RATE</th>
<th>TOTAL:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>$2146.00</td>
<td></td>
</tr>
<tr>
<td>0006</td>
<td>$18.00</td>
<td></td>
</tr>
<tr>
<td>0007</td>
<td>$175.75</td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td>$18.00</td>
<td></td>
</tr>
<tr>
<td>0013</td>
<td>$18.00</td>
<td></td>
</tr>
<tr>
<td>0019</td>
<td>$18.00</td>
<td></td>
</tr>
<tr>
<td>0021</td>
<td>$0.00</td>
<td></td>
</tr>
<tr>
<td>0023</td>
<td>$19.00</td>
<td></td>
</tr>
<tr>
<td>0026</td>
<td>$19.00</td>
<td></td>
</tr>
<tr>
<td>0028</td>
<td>$19.00</td>
<td></td>
</tr>
<tr>
<td>0029</td>
<td>$23.00</td>
<td></td>
</tr>
<tr>
<td>0040</td>
<td>$31.00</td>
<td></td>
</tr>
<tr>
<td>0048</td>
<td>$445.00</td>
<td></td>
</tr>
</tbody>
</table>

\[344\]
CUST NO 0051 CURR RATE        TOTAL: $ 0.00
CUST NO 0058 CURR RATE        TOTAL: $ 42.74

(End query phase; print file is OL210ELPT)

<QA>

6.

<QA>EXIT
    (End of IQL session)

EXIT

345
VITA

PERSONAL
Name: Nancy E. (Robison) Friedel
Birthdate: September 4, 1944
Birthplace: Butler, Pennsylvania
Parents: John & Catherine Robison
Husband: Donald C. Friedel
Children: Michael & Heather

EDUCATION
B.S. St. Francis College, Loretto, PA
Graduated Cum Laude (1966)
Major: Mathematics
Minors: Physics, Education

AWARDS & HONORS
Who's Who Among Students in American Colleges and Universities (1966)
Delta Epsilon Sigma National Catholic Honor Fraternity

PROFESSIONAL SOCIETIES
American Institute of Industrial Engineers
Association for Computing Machinery

PROFESSIONAL EXPERIENCE
Lehigh University
Bethlehem, PA
Research Assistant, 6/81-12/81
Teaching Assistant, 1/80-6/81

Pentamation Enterorises, Inc.
Bethlehem, PA, 9/76-12/79
Systems Representative

Self-Employed, 1/69-6/70
Contract Programmer

ALCOA
Cleveland, OH, 10/67-7/68
Pittsburgh, PA, 6/66-10/67
Programmer