The Usage of Query-By-Example in a Microcomputer Environment

By

Shun-Jane Lin

B.S., Chung-Hsing University, Taiwan, R.O.C.,1983

A MASTER'S REPORT

Submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

Kansas State University

Manhattan, Kansas

1986

Approved By:

R A McBride

Major Professor

TABLE OF CONTENTS

Chapter 1. Introduction 1
1.0 Overview 1
1.1 Relational Query Language 1
1.2 The Features of QBE 2
1.3 Motivation for Porting QBE 3
1.4 Report Organization 4

Chapter 2. Review of Literature 5
2.0 Introduction 5
2.1 The QBE Language 5
2.1.1 Screen Object 6
2.1.2 The Components of QBE Language 7
2.2 The Facilities of QBE 10
2.2.1 Retrieval Operations 11
2.2.2 Insertion, Deletion, and Update operations . 15
2.2.3 Table Operations 17
2.3 History of QBE 19
2.4 Extensions of QBE 19

Chapter 3. System Implementation 22
3.0 Introduction 22
3.1 dBASE-II 22
3.2 The QBE Translator 24
3.3 Parsing a Simple Query 33
3.4 Limitations 35

Chapter 4. Conclusion 35

References 40

Appendicies

A. User Manual 45
B. Logging Queries 65
C. The comparsion of Two Version of QBE 67
D. The Hierarchical Diagram for the QBE system..... 69
E. Program Listing 77

THIS BOOK CONTAINS NUMEROUS PAGES WITH DIAGRAMS THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE.

THIS IS AS RECEIVED FROM CUSTOMER.

List of Figures

Figure 2.1 General Table 6
Figure 2.2 Condition Table 7
Figure 2.3 The Operations of QBE Language 9
Figure 2.4 Simple Retrieval 11
Figure 2.5 Simple Retrieval with Ordering 11
Figure 2.6 Simple Retrieval with Multiple Prints 12
Figure 2.7 Retrieval of Table Name 12
Figure 2.8 Retrieval of Column Heading 12
Figure 2.9 Qualified Retrieval using Links 13
Figure 2.10 Retrieval Using the Condition Box 13
Figure 2.11 Retrieval of Collected Output from Multiple Tables 14
Figure 2.12 Simple Retrieval Using a Built-in Function 14
Figure 2.13 Simple Insertion 15
Figure 2.14 Simple Deletion 16
Figure 2.15 Simple Update 16
Figure 2.16 Query-Dependent Insertion 16
Figure 2.17 Query-Dependent Deletion 16
Figure 2.18 Query-Dependent Update 17
Figure 2.19 Table Creation 18
Figure 2.20 Table Deletion 20
Figure 3.1 The Process Diagram of QBE Translator 24
Figure 3.2 The Process of Parsing The Table Name Entry 26
Figure 3.3 An Example Query 27
Figure 3.4 Sfile1 28

Figure 3.5 Condition 28

Figure 3.6 The Algorithm of Stage 4 30

Figure 3.7 An Example Query 32

Figure 3.8 An Example Query 33

Figure 3.9 A Simple Query 33

Figure 3.10 Sfile1..... 34

ACKNOWLEDGEMENTS

I wish to thank Dr. Richard McBride for his help and guidance in completing this project. And a special thanks to my parents for love, encouragement and financial support.

Chapter 1

Introduction

1.0 OVERVIEW

Query-By-Example (QBE) is a high level, non-procedural data base language which provides users with facilities to query, update, and create a data base. All requests from a user are specified in QBE by filling in two-dimensional skeleton tables on the screen; this technique is known as two-dimensional programming. This unique and novel method allows the end user to express, with very few entries, the equivalent of a lengthy application program in a conventional language.

This report contains a description of an implementation of QBE in a microcomputer environment. QBE's language and facilities are also described through illustrative examples.

1.1 RELATIONAL QUERY LANGUAGE

The relational data base model was introduced by E. F. Codd in 1970 [COD70]. The central element of this model, a 'relation', can easily be mapped into a user's mental frame. Various high level data manipulation languages have been used to support this model in order to make these languages easy to use. Basically these languages can be divided into two major classes :

1. Languages based on relational algebra; and
2. Languages based on relational calculus which can be divided further into tuple relational calculus and domain relational calculus. The domain relational calculus language differs from the tuple relational calculus in that its variables range over domains rather than relations [ULL83].

In an algebraic language, the queries are expressed by applying specialized operators to relations. However, in a calculus-based language, the queries describe a desired set of tuples by specifying a predicate that the tuples must satisfy. Although both language classes are considered to be high level, it is often said that relational calculus-based languages are of a higher level than the algebraic languages. This is due to the fact that a program in a algebra-based language specifies the execution order of operations while a program in the calculus leaves it to the compiler or interpreter to determine the most efficient order of evaluation [ULL83].

A list of some existing languages, most of which are partially implemented or in use, is given for the reader's reference:

PRTV -- based on relational algebra [TOD76];

ALPHA -- based on relational calculus [COD71];

SQL2 SQUARE -- based on a mixture of relational calculus and algebra [CHA76,BOY73];

QUEL -- based on relational tuple calculus [STO75];

Query-By-Example -- based on domain relational calculus;

CUPID -- based on relational algebra [MCD75]; and

FORAL -- based on the binary relational model [SEN77].

1.2 THE FEATURES OF QBE

As mentioned in the previous section, QBE is a non-procedural language. The advantage of a non-procedural language is that a user need not possess the knowledge, required to specify a sequence of operations to be performed on the database,

in order to obtain a desired result. The QBE user is only required to have that knowledge of the data base structure which is needed to specify the data that is desired from the system. The translation of non-procedural QBE statements into a series of procedural language statements is done by an interpreter in the underlying system.

The syntax of QBE is very simple, yet it permits a wide variety of complex transactions to be expressed. The basic principle behind QBE is the simplicity of its user interface. This interface allows information to be specified by the user concerning desired information contained in the data base. The simplicity is provided by displaying a skeleton table in which user can enter a query by providing an example of a possible answer. To use QBE, a user only needs to have knowledge of the simple concepts of an example element, i.e. a variable, and a constant element. Further details of Query-by-Example concepts will be introduced in Chapter 2 and Appendix A through a collection of illustrations of queries and their answers.

1.3 MOTIVATION FOR PORTING QBE

Computers have become cheaper and widely available in the last decade. Because of this, interactive computing is in use in many businesses, and home computers have become commonplace. The people who own or use computers are often not professional programmers, so there is a need for a powerful tool which is easy to use and requires a minimum of user training. QBE is an appropriate system that meets such requirement since it is non-procedural and has a simple user interface.

QBE is available from IBM as an installed user program running on the VM/CMS operating system[IBM78]. QBE has been implemented on an Apple II microcomputer[COM80]; however, no details of this implementation were available to aid in this project. The version of QBE that was developed at KSU utilizes the facilities of dBASE-II[ASH81]. The user's queries are translated into a series of dBASE-II command statements in order to access the information in the data base.

The dBASE-II software can run on CP/M, MS-DOS, or CROMIX operating systems. Since the Zenith 150 computer systems run MS-DOS and are widely available on campus, it was chosen as the microcomputer to which QBE would be ported. However, because of the characteristics of the Zenith 150, the implemented QBE system also runs on any IBM compatible computer system.

1.4 REPORT ORGANIZATION

The intent of this project is to implement Query-by-Example for use on a microcomputer. Since the implementation of QBE was a cooperative effort, this report concentrates on those facilities in QBE that were implemented by the author, namely, the update, delete ,and insert facilities.

The purpose of this project and the general features of QBE have been discussed in this chapter. Chapter 2 provides a closer look at QBE's language and facilities. Details of how QBE statements are translated and executed using the facilities of dBASE-II are presented in chapter 3. Finally, chapter 4 summarizes the results of this part of the project and discusses possible enhancements.

Chapter 2

Review of Literature

2.0 INTRODUCTION

Before going into the implementation details of the QBE project, an overview of the QBE language is provided in this chapter. The basic components or objects of the language are discussed. Also, the facilities of QBE are introduced through a set of examples which illustrate the operations of data retrieval and manipulation. Since this chapter only briefly introduces the language, more details concerning the use of QBE on the Zenith 150 microcomputer are presented in the user manual (Appendix A). The history of QBE and languages based on it are also discussed in this chapter.

2.1 THE QBE LANGUAGE AS PROPOSED BY IBM

IBM's version of QBE is a language with a two-dimensional, graphic input format. A query in QBE is expressed through the use of a skeleton table. Two questions might be asked by a user who wants to start using the system. The first one is - what kinds of tables are provided by the system? The second question is - what can be put in each type of table? In order to answer these questions, this section describes tables or screen objects, and the elements of the QBE language.

2.1.1 Screen Objects

There are three types of screen objects in the QBE language: general tables, result tables and condition boxes. Each type of screen object has a purpose which is described below.

A general table is used in the operations of retrieval, change and creation of a relation in the data base. Unless a user desires to create a new table, each table must be associated with an existing relation in the data base. Any general table may contain four types of entries (see Figure 2.1): a table name entry, field name entries, row operator entries and data field entries. The table name entry is the place for a user to enter the name of the table that is to be dealt with. Each field name entry specifies a field name for a table (field names can also be generated automatically by the system). A row operator entry gives the QBE operation to be performed. There are four row operators provided by QBE: print(P.), delete(D.), insert(I.), and update(U.). A data field entry specifies a portion of the user's query, i.e., the data field entries describe what data are to be manipulated.

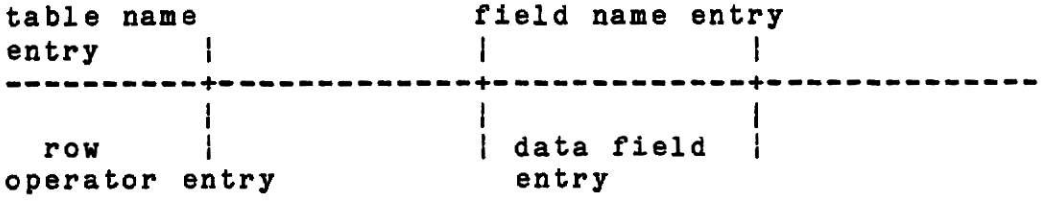

Figure 2.1 General Table

A result table is used to specify a format when the user

wants to print out or display a report. A result table also can be used when the user wants to store a new table whose contents are derived from existing tables. The data types of the fields of the new table are inherited from the relation which the data is taken. A result table may contain the same four types of entries which are used in a general table.

A condition box is used to specify conditions for complex query operations. The condition box only contains one type of entry, namely, a condition-entry (see Figure 2.2). A condition entry consists of elements (example elements or constant elements which are explained shortly) and relational operators. The evaluation of a condition should result in a boolean value. Each condition entry is specified on a separate line and results of all these entries are logically ANDed together; that is, the query will operate only if all of the conditions are true.

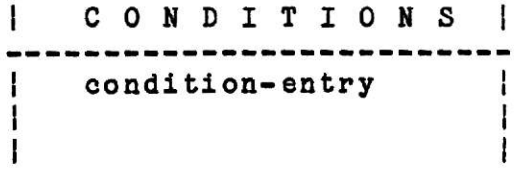

Figure 2.2 Condition Box

2.1.2 The Components of QBE Language

The QBE language consists of elements, operators, and expressions.

Elements are used to define and qualify data. There are two types of elements used in the QBE language: constant elements and example elements. Constant elements are used in specifying selection criteria; that is, to qualify or limit the results to

values which match a relation on the constant element. A constant element requires no special notation.

Example elements are used to: link data among different tables or different rows in the same table, specify conditions, and map data from one table to another. Example elements do not exist in the data base; they merely represent an example of a possible answer to a query. Example elements must begin with a prefixed underscore and be followed by one or more letters (e.g., _A, _RED).

QBE operators are used to define processing and to specify conditions. The types of operators provided by the QBE language as follows:

- System operators which define processing;
- Built-in functions, which determine the average, maximum, sum, count, minimum, and uniqueness (which eliminates duplicates) of some data items;
- Relational operators which permit comparisons to be made and conditions to be defined;
- Arithmetic operations;
- Logical operators which link conditions together.

All of the operators available in QBE are explained in Figure 2.3.

Type	Operators	Meanings
System operators	P.	print
	I.	Insert
	U.	update
	D.	delete
	AO.	in acending order (used with print operator)
	DO.	in decending order (used with print operator)
Built-in functions	CNT.	count data item
	SUM.	sum data item
	MAX.	find the maximum value
	MIN.	find the minimum value
	AVG.	calculate the average
	UNQ.	eliminate (all but one) identical data items
Relational comparisons	>	greater than
	<	less than
	>=	greater than or equal to
	<=	less than or equal to
	<>	not equal to
Arithmetic	*	multiplication
	/	division
	-	subtraction
	+	addition
Logical	|	or
	&	and

Figure 2.3 The operators of QBE language.

Expressions are constructed out of a combination of elements and operators. They are used to qualify or limit the data that the user wants to deal with. There are two types of expressions which are allowed in QBE language:

- Arithmetic expressions, which combine example and constant elements by using arithmetic operators; and
- Logical expressions, which combine example and constant elements by using logical operators.

2.2 THE FACILITIES OF QBE

There are four system operators, P., I., U., D., available in QBE. These four operators allow a user to create, query, and update a data base. 'P.' stands for print; it is the operator used to retrieve information from the data base. The 'I.' (insert) operator allows the information to be added to the data base. 'U.' is the operator which is used to update or change information values. 'D.' stands for deletion; it is the operator used to delete or drop information from the data base.

The use of the print operator does not have any side effects upon the data base. Whenever an insert, delete, or update operator is used in a table skeleton, a corresponding change is made in the data base. In the following text, several examples using these operations are given in order to illustrate the basic concepts concerning the usage of these operators. A detailed syntactic and semantic description of QBE language is covered in the user manual (Appendix A).

2.2.1 Retrieval Operations

An example of a simple retrieval is shown in Figure 2.4. The user fills in the name of the table in the table name field; in this case, the name is PART. Next, the user may either fill in the column heading or let the system generate them automatically. The user next expresses a query by entering the print operator(P.) and an example element (which is prefixed with underscore) to print out all PART colors.

PART	PNAME	COLOR	WEIGHT	
		P._C		

Figure 2.4 Simple Retrieval

To retrieve the data with ordering. The user can enter the print operator along with a system operator(AO. or DO.). The system prints out the data depending upon the request. The example shown in Figure 2.5 prints out all part names in alphabetical order. 'AO.' stands for ascending order, and 'DO' stands for decending order.

PART	PNAME	COLOR	WEIGHT	
	P.AO._C			

Figure 2.5 Simple Retrieval with Ordering

An example of a simple retrieval with multiple fields being printed is shown in Figure 2.6. The user can print out the entire PART table by entering the print operator(P.) in the first row

operator field.

PART	PNAME	COLOR	WEIGHT	
P.				

Figure 2.6 Simple Retrieval with Multiple Prints

To list the available table names in the data base, the user enters a print operator and an example element in the table name field (see Figure 2.7).

P._NAME				

Figure 2.7 Retrieval of Table Name

To see the column headings of a table, the user first fills in the table name to be worked with. This is followed by print operator. The example in Figure 2.8 shows how QBE can be used to print out the attributes of the PART table.

PART P.				

Figure 2.8 Retrieval of Column Heading

Qualified retrieval using links is shown in the example in Figure 2.9. In order to print the names of all the red parts made by the manufacturer YKK, the user uses two tables, the PART table and the MAKE table. The same example element is used in both tables, indicating that in order for a part to meet the criteria

specified by this query it must be red, and that same item should also be made by manufacturer YKK. Only if these condition are met simultaneously does the item qualify as a solution.

PART	PNAME	COLOR	WEIGHT	
	P._P	RED		

MAKE	MNAME	PNAME	QTY	
	YKK	_P		

Figure 2.9 Qualified Retrieval using Links

A condition box can be used to express one or more desired conditions that would be difficult to express in the tables. An example is shown in Figure 2.10, in which the names of all parts whose weight is heavier than the total weight of bolt and nut, but also lighter than 10 pounds are printed. The user can use example elements (in this case, _W1, _W2, _W3) and specify conditions by using example elements combined with relational operators in a condition box.

PART	PNAME	COLOR	WEIGHT	
	P. BOLT NUT		_W1 _W2 _W3	

C O N D I T I O N
_W1 > _W2 + _W3 _W1 < 10

Figure 2.10 Retrieval using the Condition Box

The user can combine together output that was retrieved from multiple tables. This is done by making use of a result table. An example of this is shown in Figure 2.11, where each manufacturer along with its corresponding buyer is printed out. Since the output is a new table, the user must use a third table. The third table is filled in with examples mapped from two existing tables which satisfy the stipulation of the query.

BUY	PNAME	BUYER		
	_P	_BUY		

MAKE	MNAME	PNAME	QTY	
	_N	_P		

RESULT	MNAME	BUYER		
	P._N	P._BUY		

Figure 2.11 Retrieval of Collected Output from Multiple Tables

QBE provides a set of built-in functions (as discussed in the previous section). These built-in functions can be used in specifying the desired answer. For example, the user can find the total number of manufacturers by entering a print operator along with a built-in function name (see Figure 2.12).

MAKE	MNAME	PNAME	QTY	
	P.CNT.UNQ.ALL.			

Figure 2.12 Simple Retrieval using a Built-in Function

2.2.2 Insertion, Deletion, and Update Operations

Insertions(I.), deletions(D.) , and updates(U.) are done in the same style as the query operation. The following examples are categorized into two parts: simple insertions, deletions, and updates (the term 'simple' applies to operations that involve constant elements only), and those that are query-dependent i.e., certain information has to be extracted from the data base in order to accomplish a query.

One tuple may be added into an existing table by placing an insert operator(I.) in the row operator field, and the information in each column for the table. Figure 2.13 demonstrates how to insert part name 'WASHER' (that has a color of 'RED', a weight of 5, and is made in the city of 'LONDON') into the PART table. The user specifies the new tuples as in the example shown in Figure 2.13.

PART	PNAME	COLOR	WEIGHT	CITY
I.	WASHER	RED	5	LONDON

Figure 2.13 Simple Insertion

The user may delete information from a table by entering the delete operator(D.) in the row operator field and the condition for this operation. The example shown in Figure 2.14, causes QBE to delete all of the information about parts made in the city 'PARIS'.

PART	PNAME	COLOR	WEIGHT	CITY
D.				PARIS

Figure 2.14 Simple Deletion

The user can modify the information in a relation by entering the update operator(U.) followed by a new value in the column that is to be changed. As the result of the example shown in Figure 2.15. QBE updates the color of all parts named bulb to be blue.

PART	PNAME	COLOR	WEIGHT	CITY
	BULB	U.BLUE		

Figure 2.15 Simple Update

The following is an example of a query-dependent insertion. A user can add tuples whose values are derived from other tuples in the data base. The example shown in Figure 2.16 causes QBE to find the colors of bulb, and then insert the colors with the information for all tuples with weight '4' and the part name 'BOX' into the PART table.

PART	PNAME	COLOR	WEIGHT	CITY
I.	BOX	_COLOR	4	
	BULB	_COLOR		

Figure 2.16 Query-Dependent Insertion

In the example shown below, QBE determines the number of products manufactured by YKK. QBE then deletes all of the

information about any other company which manufactures more products than YKK.

MAKE	MNAME	PNAME	QTY	
D.			>_Q	
D.	YKK		>_Q	

Figure 2.17 Query-Dependent Deletion

Sometimes it is desirable to update an entry with a value based upon its previous value. Suppose the weight of part name 'washer' must be increased by 5. Figure 2.18 demonstrates how this can be accomplished; QBE will find the old value of the part WASHER and increase it by 5. Finally, the weight of the part WASHER is updated to the new value.

PART	PNAME	COLOR	WEIGHT	CITY
	WASHER		_WT	
	WASHER		U._WT+5	

Figure 2.18 Query-Dependent Update

2.2.3 Table Operation

In the QBE language, the creation of a table is done in the same style as the previous operations.

Suppose it is necessary to create a new table with table name STU and column headings: NAME, ADDR, GPA, DEPT (see Figure 2.19). Starting with a blank skeleton on the screen, the user fills in the headings by inserting the field names. For each field name, the following row attributes have to be specified:

- TYPE specifies the data entry type, such as float, char, fixed, etc.,

- LENGTH specifies the length of that field,
- KEY specifies the fields that are considered to form the primary key of that table,
- DOMAIN specifies the name of the underlying domain of the field, i.e., the value set from which the field's data elements are drawn,
- INVERSION specifies whether or not an index is to be built based on the field (QBE assumes that every field is part of the key and that every field is to be indexed, unless informed to the contrary).

I. STU I.	NAME	ADDR	GPA	DEPT
KEY I.	Y	Y	N	N
TYPE I.	CHAR	CHAR	FLOAT	CHAR
DOMAIN I.	NAMES	ADDS	GPAS	DEPTS
LENGTH I.	20	30		10
INVERSION I.	Y	N	N	N

Figure 2.19 Table Creation

The operator U. can be used to rename a table name. For example, to update the table name from STU TO STU1, the user just enters 'STU U.STU1' in the table name field.

The user can remove a table from the data base by entering in the table name field the delete operator followed by the name of the table to be removed. For example, to remove the table named PART from the data base, the user just enters 'D. PART' into the table name field (as in the following example).

D. PART	PNAME	COLOR	WEIGHT	CITY

Figure 2.20 Table deletion

2.3 HISTORY OF QBE

Initially, QBE was intended as a query language for relational data bases. Since then it has been developed and extended to form a complete data base manipulation language with facilitites to update, and create a data base. Because most of the operations used for querying the data base are also employed in updating and creating a data base, the name of QBE language was not altered.

The first implementation of Query-by-Example was done by K. E. Niebuhr and S. E. Smith[NIE77]. Afterward, M. M. Zloof developed it into a commercial product at the IBM Yorktown Heights Research Laboratory, and the product was released in 1978.

2.4 EXTENSION OF QBE

Since QBE was introduced, two programming languages based on QBE have been developed for business and office automation. Two extended versions of QBE are the System for Business Automation (SBA)[ZLO77] and Office Procedures by Example (OBE) [ZLO81], [ZLO82].

SBA is a system in which business applications are described