Query by Example

by MOSHÉ M. ZLOOF
IBM T. J. Watson Research Center
Yorktown Heights, New York

INTRODUCTION

In the last few years we have witnessed a trend to appeal to the non-professional user who has little or virtually no computer or mathematical background.

The 'Query by Example' Language is an attempt in that direction. It operates on a relational Model of data as was introduced by Codd [1-5].

In this paper we deal only with normalized relations [1]. A relation is normalized if each of its domains is simple, i.e., no domain is itself a relation.

A normalized relation can be viewed as a table of *n* columns and a varying number of rows as illustrated in Figure I. Three properties of normalized relations are noteworthy to mention:

1. ALL rows of the table are distinct.
2. The ordering of the rows is immaterial.
3. The ordering of the columns is immaterial provided each has a distinct name.

EMP	NAME	SALARY	MANAGER	DEPARTMENT
	ANDERSON	8K	SMITH	TOY
	MORGAN	10K	LEE	COSMETICS
	.			
	.			
	.			

Figure 1—Employee relation

In THE LANGUAGE FACILITIES we introduce the concepts of the Language. ADDITIONAL EXAMPLES deals with additional examples. In GROUPING we introduce the concept of grouping. CONCLUSION deals with conclusions and remarks.

In addition a *sample data base* is available in the Appendix so that hopefully the user will refer to it in the course of learning the concepts of the Language.

THE LANGUAGE FACILITIES

In this section we introduce the Query by Example components. The concepts are described primarily through illustrations of queries and their answers, each illustration followed by a discussion to point out major features. The illustrations get progressively more complex until the whole scope of the Language is covered. In so doing, a user dealing with "simple" queries needs to study the system *only* to that point of complexity which is compatible with the level of sophistication required within the domain of those queries.

Furthermore, although the introduction of the concepts through illustrative examples reduces somewhat from the rigor of mathematical formulation through definitions, it is—in our opinion—more appealing to the casual user, which is one of the major aspects of Query by Example.

Most of the queries are drawn from the following tables (relations), which are part of a department store data base.

EMP (NAME, SAL, MGR, DEPT)
SALES (DEPT, ITEM)
SUPPLY (SUPPLIER, ITEM)
TYPE (ITEM, COLOR, SIZE)

—The EMP Table specifies the name, salary, manager and department of each employee.
—The SALES Table is a listing of the items sold by departments.
—The SUPPLY Table is a listing of the items supplied by suppliers.
—The TYPE Table describes each item by color and size.

At this point we are assuming that these tables are made available to the user upon calling them by name. In a subsequent paper, the creation, deletion, insertion and updating of these tables will be discussed in detail.

In this system the user basically formulates his query by *filling in* the appropriate table rows with an example of a possible answer. In fact for a large class of "simple" queries the user need only distinguish between the following two entities:

1. The 'example element' (variable) which must be underlined and
2. The 'constant element' which should not be underlined.

In addition the function 'P.' stands for 'print'. The user inserts a 'P.' before any data he wishes to be outputted.

Examples:

Q1. Print the red items:

The user fills in the TYPE Table in the following manner.

TYPE	ITEM	COLOR	SIZE
	P.PEN	RED	

Since the query is concerned with red items, RED is a 'constant element' and is, therefore, not underlined. On the other hand, the underlined element PEN referred to as an 'example element' is entered as an example of a possible answer. Actually a pen may not necessarily be an element of the data base and can be substituted by DRESS, WATER or a variable X without altering the meaning of the query.

One of the reasons we are using an example element instead of a variable is that it gives us the freedom to use an entity which is partly variable and partly constant (see GROUPING). The SIZE Column can either remain blank or can be filled with an example element as well.

Considering the sample data base at the end of the paper the answer to this query is:

ITEM
LIPSTICK
PENCIL

For those users interested in the mathematical formulation of the queries, each query will be reformulated in predicate calculus

Q1. $\{x{:}\exists y(x, \text{RED}, y) \in \text{TYPE}\}$

Q2. What colors of ink are available?

TYPE	ITEM	COLOR	SIZE
	INK	P.BLACK	

In this case the 'P.' is in the color column since we want a listing of the colors of ink. BLACK is the example element.

ANS:

COLOR
GREEN
BLUE

Q2. $\{x{:}\exists y(\text{INK}, x, y) \in \text{TYPE}\}$

Q3. Find the department(s) that sells an item(s) supplied by the supplier Parker.

Here the user fills in both the SALES and the SUPPLY Tables as follows.

SALES	DEPT	ITEM
	P.TOY	ROD

SUPPLY	ITEM	SUPPLIER
	ROD	PARKER

ANS:

DEPT
HOUSEHOLD
TOY
STATIONARY
HARDWARE

Note: The example element ROD (linking variable) is included in both tables, implying if an item is sold by the department in question that *same* item has to be supplied by Parker. (Pretty much the same way one would scan the data base manually to find the answers.)

Q3. $\{x{:}\exists y((x, y) \in \text{SALES} \wedge (y, \text{PARKER}) \in \text{SUPPLY})\}$

Q4. Find the supplier(s) that supplies an item(s) sold by the TOY Department.

SALES	DEPT	ITEM
	TOY	PEN

SUPPLY	ITEM	SUPPLIER
	PEN	P.GM

ANS:

SUPPLIER
PARKER
BIC
REVLON

Q4. $\{x{:}\exists y((\text{TOY}, y) \in \text{SALES} \wedge (y, x) \in \text{SUPPLY})\}$

Q5. List the names, salaries, and managers of employees in the TOY Department.

EMP	NAME	SAL	MGR	DEPT
	P.JONES	P.10K	P.SMITH	TOY

ANS:

NAME	SAL	MGR
ANDERSON	6K	MURPHY
NELSON	6K	MURPHY
HENRY	9K	SMITH

Here the multiple output was achieved simply by inserting P. in the NAME, SAL, and MGR columns. The only constant element is TOY.

At this point we should mention that as long as an example element is not used for linkage purposes one can write just the function P. leaving blank space in place of the element. Thus in Q5. one can dispose of JONES, 10K, and SMITH.

Q5. $\{(x, y, z):(x, y, z, \text{TOY}) \in \text{EMP}\}$

The following additional types of operators and functions are part of the system:

numeric comparisons: $= \neq < \leq > \geq$
negation operator: $\neg$
the operators JOIN, ALL & ALL D. (explained later) and built in functions; SUM, COUNT, AVE, MAX, MIN, etc.

Q6. Print out a list of all the departments, the items they sell and the suppliers that supply these items.

In this case we must first join the SALES Table with the SUPPLY Table on the common attribute ITEM and then apply the function P. as follows.

SALES	DEPT	ITEM
	TOY	PEN

SUPPLY	ITEM	SUPPLIER
	PEN	BIC

JOIN: SALES/SUPPLY	DEPT	ITEM	SUPPLIER
	P.TOY	P.PEN	P.BIC

The JOIN operator specifies joining the SALES and the SUPPLY Tables. The example element PEN appears in both tables to indicate a natural join on the common attribute ITEM.

ANS:

DEPT	ITEM	SUPPLIER
STATIONARY	DISH	DUPONT
HOUSEHOLD	PEN	PARKER
.	.	.
.	.	.
.	.	.
COSMETICS	LIPSTICK	REVLON
TOY	PEN	PARKER

Q6. $\{(x, y, z):\exists u(x, y) \in \text{SALES} \wedge (u, z) \in \text{SUPPLY} \wedge y = u\}$

Note: if the example element PEN does not appear in both tables, i.e., $y \neq u$, the join of these tables clearly becomes a Cartesian product.

Q7. Find the name(s) of any employee(s) who earns more than his (their) manager(s).

EMP	NAME	SAL	MGR	DEPT
	P.JONES	>10K	PETER	
	PETER	10K		

ANS:

NAME
LEWIS
HOFFMAN

If PETER is an example of such a manager and if PETER earns 10K (as an example) then JONES is an example of an employee who earns more than 10K (indicated by the > operator) and, therefore, more than his manager. It should be noted that the *order* of the rows is immaterial.

Q7. $\{x:\exists y \exists z \exists u \exists w \exists l \exists m((x, y, z, u) \in \text{EMP} \wedge (z, w, l, m) \in \text{EMP} \wedge y > w)\}$

Q8. Find the department(s) that sells Pens *and* Pencils.

SALES	DEPT	ITEM
	P.TOY	PEN
	TOY	PENCIL

ANS:

DEPT
STATIONARY
TOY

Here, in order to account for the AND, the same example element TOY is used in both rows since the same department has to sell both items.

Q8. $\{x:(x, \text{PEN}) \in \text{DEPT} \wedge (x, \text{PENCIL}) \in \text{DEPT}\}$

Q9. Find the department(s) that sells Pens *or* Pencils.

SALES	DEPT	ITEM
	P.TOY	PEN
	P.HARDWARE	PENCIL

ANS:

DEPT
HOUSEHOLD
STATIONARY
TOY

Here two different example elements are used to account for the OR since a department that sells pens does not necessarily have to sell pencils.

Q9. $\{x:(x, \text{PEN}) \in \text{DEPT} \vee (x, \text{PENCIL}) \in \text{DEPT}\}$

Q10. Find the department(s) that sells *all* the items supplied by the supplier Parker.

SALES	DEPT	ITEM
	P.HOUSEHOLD	[ALL PEN •]

SUPPLY	ITEM	SUPPLIER
	ALL PEN	PARKER

ANS:

DEPT
STATIONARY
TOY

ALL PEN is defined to be the set of all the items supplied by Parker. The dot ('•') under ALL PEN in the SALES Table indicates that the department(s) in question *may* sell more than all the items supplied by Parker. On the other hand, if we wish to indicate that the department(s) in question has to strictly sell more than ALL PEN, it will be written as

$$\begin{bmatrix} \text{ALL } \underline{\text{PEN}} \\ \underline{\text{PENCIL}} \\ \bullet \end{bmatrix} \quad (*)$$

The brackets in the ITEM column have no meaning other than grouping the dot with ALL PEN.

Q10. $\{x{:}\forall y((y,\ \text{PARKER})\in\text{SUPPLY}\rightarrow(x,\ y)\in\text{SALES})\}$

(*) Formally $\text{ALL } \underline{\text{PEN}} \subseteq \begin{bmatrix} \text{ALL } \underline{\text{PEN}} \\ \bullet \end{bmatrix}$

and $\text{ALL } \underline{\text{PEN}} \subset \begin{bmatrix} \text{ALL } \underline{\text{PEN}} \\ \underline{\text{PENCIL}} \\ \bullet \end{bmatrix}$

Q11. Find the department(s) such that *all* their items are supplied by Parker.

SALES	DEPT	ITEM
	P.HOUSEHOLD	ALL PEN

SUPPLY	ITEM	SUPPLIER
	[ALL PEN •]	PARKER

ANS:

DEPT
HARDWARE
TOY

Here the dot is under the ITEM column in the SUPPLY Table meaning that the Supplier Parker may supply more than all the items sold by the department in question.

Q11. $\{x{:}\forall y((x,\ y)\in\text{SALES}\rightarrow(y,\ \text{PARKER})\in\text{SUPPLY})\}$

Q12. Find the department(s) which sell *only all* the items supplied by Parker.

SALES	DEPT	ITEM
	P.HOUSEHOLD	ALL PEN

SUPPLY	ITEM	SUPPLIER
	ALL PEN	PARKER

ANS:

DEPT
TOY

Here the sets on both sides have to be equal thus there is no dot.

Q12. $\{x{:}\forall y((x,\ y)\in\text{SALES}\leftrightarrow(y,\ \text{PARKER})\in\text{SUPPLY})\}$

When a function such as P. SUM. COUNT. etc., precedes the operator ALL, the set ALL X (where X is any example element) retains its duplicate elements (*). This is necessary for the many instances when the duplicate elements are to be included in the count. This is illustrated in the next query.

Q13. Find the total salaries of the employees in the TOY Department.

EMP	NAME	SAL	MGR	DEPT
		P.SUM.ALL 10K		TOY

ANS:

SAL SUM =
21K

In this case the elements 6, 9, and the duplicate element 6 are summed. On the other hand if one wishes to exclude duplicate elements, the operator ALL D. is used where the 'D.' stands for differen‘ or distinct. This is again illustrated in the next example.

(*) Actually ALL X becomes a multi-set or a 'Bag' (in computer science terminology) where mapped duplicate elements are retained.

Q13. SUM $*\{x{:}\exists y\exists z(y,\ x,\ z,\ \text{TOY})\in\text{EMP}\}$

where the asterisk indicates that it is an operation on a multi-set.

Q14. How many colors of pencils are there?

TYPE	ITEM	COLOR	SIZE
	PENCIL	P.COUNT.ALL D.GREEN	

ANS:

COLOR COUNT =	
2	(namely: red, blue)

Had we used the operators P.COUNT. ALL GREEN, the color blue would have been counted twice and the answer would have been '3'.

Q14. COUNT$\{x{:}\ y(\text{PENCIL},\ x,\ y)\in\text{TYPE}\}$

Q15. Among all departments with total salaries greater than 22K, find those departments which sell pens.

EMP	NAME	SAL	MGR	DEPT
		(SUM.ALL 10K) >22K		TOY

SALES	DEPT	ITEM
	P.TOY	PEN

ANS:

DEPT
STATIONARY

Q15. $\{x{:}(x, \text{PEN}) \in \text{SALES} \wedge \text{SUM}$

$$*\{y{:}\exists z \exists u (z, y, u, x) \in \text{EMP}\} > 22k\}$$

Note: Again the asterisk indicates the summation is performed over a multi-set.

Q16. Find item(s) that come in colors other than green.

TYPE	ITEM	COLOR	SIZE
	P. ROD	¬ GREEN	

ANS: The whole column of items except PEN will be printed. INK will be printed even though it comes in green, because it also comes in blue, thus satisfying the stipulation in the query.

Q16. $\{x{:}\exists y \exists z (y \neq \text{GREEN} \wedge (x, y, z) \in \text{TYPE})\}$

Q17. List all the items except the ones which come in green.

TYPE	ITEM	COLOR	SIZE
	ROD	GREEN	
	P. ¬ ROD		

ANS: The whole column of items except PEN and INK will be printed.

Unlike Q16, Q17 requires the elimination of any item that comes in green, even if the same item comes in other colors. In other words, the green items are sorted out and subtracted from the set of all items, leaving the complement set of non-green items. This complement set in our sample data base is the set of all the items except PEN and INK.

Q17. $\{x{:}\forall y \forall z ((x, y, z) \in \text{TYPE} \rightarrow y \neq \text{GREEN})\}$

We must point out that if the data to satisfy the query are insufficient, the system prints 'NONE' in the appropriate column.

In addition, the system can be used as a verifier by completing the applicable columns with constant elements. If the element relations presented is positive, the system verifies that by printing the same constant elements. Otherwise, 'NONE' is printed in the column where the relation fails.

ADDITIONAL EXAMPLES

In this section we will formulate a collection of queries taken from various papers [5, 6, 7, 9] to illustrate major differences. No new features are introduced in this section.

Consider the following data base:

SUPPLY (SUPPLIER, PART NAME, JOB NAME)
PART (PART NAME, TYPE)
JOB (JOB NAME, LOCATION)

Q18. Find the names of suppliers who supply a job located in New York with all parts of Type A.

SUPPLY	SUPPLIER	PART NAME	JOB NAME
	P. ACME	[ALL. ROD]	BULB

PART	PART NAME	TYPE
	ALL ROD	A

JOB	JOB NAME	LOCATION
	BULB	NEW YORK

Consider the following data base:

EMP (NAME, SAL, MGR, DEPT)
SALES (DEPT, ITEM, VOL)
SUPPLY (COMP, DEPT, ITEM, VOL)
LOC (DEPT, FLOOR)
CLASS (ITEM, TYPE)

Q19. Find companies, each of which supplies every item of type A to some department on the second floor.

LOC	DEPT	FLOOR
	TOY	2

CLASS	ITEM	TYPE
	ALL PEN	A

SUPPLY	COMP	DEPT	ITEM	VOL
	P.PARKER	TOY	[ALL PEN •]	

Note: We can start formulating the query in any table, since the order is immaterial.

Consider the following data base:

EMP (MAN #, NAME, JOB CODE, SAL, DEPT #)
DEPT (DEPT #, NAME, MGR)

Q20. Find the information contained in the department record concerning departments having more than 20 employees whose job code is 802.

EMP	MAN #	NAME	JOB CODE	SAL	DEPT #
		(COUNT.ALL JIM)>20	802		30

DEPT	DEPT #	NAME	MGR
	P. 30	P.JONES	P. SMITH

GROUPING

In THE LANGUAGE FACILITIES we mentioned that the reason we chose to underline an element to make it a variable is to enable us to have an entity that is partly variable and partly constant.

Example: the number 56$\underline{0}$ is read 56X, and the name $\underline{\text{JIM}}$ is read JXY, where X and Y are variables.

This concept of creating a variable by underlining is extended to a second line to group equivalence classes of a set.

Example:

DEPT
$\underline{\underline{780}}$

where the second line indicates grouping by departments.

For ease of reference we change the EMP Table to EMP (NAME, DEPT).

Q21. Count the employees by departments.

EMP	NAME	DEPT
	P. COUNT. ALL $\underline{\text{JIM}}$	P. $\underline{\underline{\text{TOY}}}$

ANS:

NAME COUNT=	DEPT
2	HOUSEHOLD
3	TOY
3	COSMETICS
2	STATIONARY

Q21. $\{\text{COUNT}\{(x, y_1) \in \text{EMP}\},$

$\text{COUNT}\{(x, y_2) \in \text{EMP}\} \ldots,$

$\text{COUNT}\{(x, y_n) \in \text{EMP}\} : \{y_1 \ldots y_n\} = \text{DEPT} \wedge$

$: i \neq j \rightarrow y_i \neq y_j\}$

Q22. Count the employees by departments that have the same first letter on the left.

EMP	NAME	DEPT
	P. COUNT. ALL $\underline{\text{JIM}}$	P. $\underline{\underline{\text{T}}}\underline{\text{OY}}$

The answer is the same as in the case of Q21. However, if there are two departments with the same first letter, their employees will be counted together.

Q23. Count employees by departments and managers.

EMP	NAME	DEPT	MGR
	P. COUNT ALL $\underline{\text{JIM}}$	P. $\underline{\underline{\text{TOY}}}$	P. $\underline{\underline{\text{SMITH}}}$

CONCLUSION

In this paper we presented the data access portion of the Query by Example Language. We conclude that the *unique* features of this language are as follows:

1. The user has the perception of manual table manipulation.
2. The user has a pre-established *frame of reference*, i.e., the tables.
3. The user can easily pre-identify the relations to be used, resulting in an early reduction in the scope of the data base.
4. As opposed to linear-type languages where the user is constrained to one degree of freedom, here the user has multi-degrees of freedom in that the sequence of filling in the tables and the rows within the tables is immaterial. This implies that given a data base the system does not constrain the user's thinking process in any way while he/she is formulating the query. Take Q7 as an example. If the user's thinking process wishes to first choose a manager and then compare his/her salary to the salary of his/her employees, the query would be the same whatever the row order is, thus the system is capable of capturing the different ways different users approach the problem.
5. The sequence of the following steps is also immaterial.
   a) filling in the constant elements,
   b) linking the variables,
   c) specifying the output by the P. function (projection), and
   d) grouping.
6. It follows from 4 and 5 that Query by Example allows the user to divide the query into decoupled segments, making it declarative and highly non-procedural. In contrast, most linear-type and other languages require the user to first specify the information to be outputted and then structure the query accordingly.
7. Due to the decoupling features inherent in Query by Example, it can handle rather complicated queries without relinquishing its simplicity. This is in contrast to other languages where a lengthy and complicated query has to be artificially divided into multiple steps and then taken one at a time.

REMARKS

1. "Relational Completeness" and arithmetic operations will be covered in subsequent papers.

2. Papers related to Query by Example are listed in the References 10 and 11.

ACKNOWLEDGMENTS

To Peter de Jong I am most grateful for his helpful suggestions and general encouragement throughout the development of this work.

For their helpful discussions, I wish to thank John Gould and John Thomas of the Behavioral Science Group who are currently testing the system on subjects through a display. Also, I wish to thank Ashok Chandra, Ron Fagin, Pat Goldberg, and Irving Wladawsky of IBM Research Yorktown for their useful discussions.

REFERENCES

1. Codd, E. F., "A Relational Model of Data for Large Shared Data Banks," *Comm. ACM*, Vol. 13, No. 6, June 1970, pp. 377-387.
2. Codd, E. F., "Further Normalization of the Data Base Relational Model," *Courant Computer Science Symposia*, Vol. 6, *Data Base Systems*, Prentice-Hall, New York, May 1971.
3. Codd, E. F., "Relational Completeness of Data Base Sublanguages," *Courant Computer Science Symposia*, Vol. 6, *Data Base Systems*, Prentice-Hall, New York, May 1971.
4. Codd, E. F., "Normalized Data Base Structure: A Brief Tutorial," *Proc. 1971 ACM SIGFIDET Workshop on Data Description, Access and Control*, San Diego, November 1971.
5. Codd, E. F., "A Data Base Sublanguage Founded on the Relational Calculus," *Proc. 1971 ACM SIGFIDET Workshop on Data Description, Access and Control*, San Diego, November 1971.
6. Boyce, R. F., D. D. Chamberlin, W. F. King III, and M. M. Hammer, "Specifying Queries as Relational Expressions," *Proceedings of ACM SIGPLAN/SIGIR Interface Meeting on Programming Languages and Information Retrieval*, Gaithersburg, Maryland, November 1973.
7. Astrahan, M. M., E. B. Altman, P. L. Fehder and M. F. Senko, "Concepts of a Data Independent Accessing Model," *Proc. 1972 ACM SIGFIDET Conference*, Denver, Colorado, November 29-30, 1972.
8. *Interactive Query Facility (IFQ) for IMS/360*, Publication No. GH 20-1074, IBM Corporation, White Plains, New York.
9. Chamberlin, D. D., and R. F. Boyce, *SEQUEL: A Structured English Query Language*, IBM Report, No. RJ1394.
10. Zloof, M. M., *Query by Example: The Invocation and Definition of Tables and Forms*, IBM Research Report, No. RC5115, February 1975.
11. Zloof, M. M., *Query by Example: Operations on the Transitive Closure*, IBM Report in preparation.

APPENDIX

SAMPLE DATA BASE

EMP	NAME	SALARY	MGR	DEPT
	JONES	8K	SMITH	HOUSEHOLD
	ANDERSON	6K	MURPHY	TOY
	MORGAN	10K	LEE	COSMETICS
	LEWIS	12K	LONG	STATIONARY
	NELSON	6K	MURPHY	TOY
	HOFFMAN	16K	MORGAN	COSMETICS
	LONG	7K	MORGAN	COSMETICS
	MURPHY	8K	SMITH	HOUSEHOLD
	SMITH	12K	HOFFMAN	STATIONARY
	HENRY	9K	SMITH	TOY

SALES	DEPARTMENT	ITEM
	STATIONARY	DISH
	HOUSEHOLD	PEN
	STATIONARY	PENCIL
	COSMETICS	LIPSTICK
	TOY	PEN
	TOY	PENCIL
	TOY	INK
	COSMETICS	PERFUME
	STATIONARY	INK
	HOUSEHOLD	DISH
	STATIONARY	PEN
	HARDWARE	INK

SUPPLY	ITEM	SUPPLIER
	PEN	PARKER
	PENCIL	BIC
	INK	PARKER
	PERFUME	REVLON
	INK	BIC
	DISH	DUPONT
	LIPSTICK	REVLON
	DISH	BIC
	PEN	REVLON
	PENCIL	PARKER

TYPE	ITEM	COLOR	SIZE
	DISH	WHITE	M
	LIPSTICK	RED	L
	PERFUME	WHITE	L
	PEN	GREEN	S
	PENCIL	BLUE	M
	INK	GREEN	L
	INK	BLUE	S
	PENCIL	RED	L
	PENCIL	BLUE	L