CHOOSING A DATABASE QUERY LANGUAGE

Matthias Jarke and Yannis Vassiliou

November 1982
Revised April 1984

Center for Research on Information Systems
Computer Applications and Information Systems
Graduate School of Business Administration
New York University

Working Paper Series

CRIS #68

GBA #84-39(CR)

CHOOSING A DATABASE QUERY LANGUAGE

ABSTRACT

A methodology is presented for selecting query languages suitable for certain user types. The method is based on a trend model of query language development on the dimensions of functional capabilities and usability. Expected developments are exemplified by the description of "second generation" database query languages. From the trend model are derived: a classification scheme for query languages; a criterion hierarchy for query language evaluation; a comprehensive classification scheme of query language users and their requirements; and recommendations for allocating language classes to user types. The method integrates the results of existing human factors studies and provides a structured framework for future research.

CR CATEGORIES AND SUBJECT DESCRIPTORS: D.3.2 [Language Specifications]: Very High-Level Languages, H.2.3 [Database Management]: Languages, H.3.3 [Information Storage and Retrieval]: Query Formulation, H.1.2 [User/Machine Systems]: Human Factors.

GENERAL TERM: Query Languages

ADDITIONAL KEYWORDS AND PHRASES: databases, human factors, language evaluation, user classification.

1. INTRODUCTION

Over the last decade, the focus of computer system use in organizations has shifted from number-crunching and mass data processing to the management of data as a strategic resource. Database management systems provide a consistent view of the organization, give a variety of users appropriate and secure access to the data, and offer efficient file management support for application programs. A central interaction mode of users with a database system is through a query language.

A _query language_ (QL) is defined as a high-level computer language for the retrieval of data held in databases or files [BCS 81]. It is usually interactive, on-line, and able to support queries which are ad-hoc (not predefined). Interaction via a QL tends to be of limited complexity, and the displayed answer is usually relatively short. Finally, it is often assumed that the principal users of QLs have limited technical expertise.

The number and variety of query languages available or under development have grown so rapidly that a framework for evaluating query languages in terms of their functionality and usability by different types of users is needed. The goal of this paper is to propose a methodology for selecting a query language; a goal which has not been attempted in previous surveys and categorizations of QLs [BCS 81, LaPi 76,77,80, LeBl 79, LeSa 74, LoTs 81, McMc 82, RTG 82, StRo 77].

In order to reach this goal, well-structured taxonomies of query languages and language users are developed. The proposed methodology provides a framework for contrasting both taxonomies and combining them with existing technical and human factors research to recommend the type of language to select for a known user class. However, as the existing human factors research does not suffice for conclusive judgements, the main contribution of this paper is the proposed framework and a structured set of hypotheses for future studies. The usefulness of our methodology has been demonstrated by its application in the development of evaluation schemes for a major empirical study comparing natural vs. formal query languages [TURNE 82, VASSI 83, JARKE 84].

The paper is organized as follows. Section 2 presents an analysis of trends which seem to govern the development of QLs. Recent experimental systems are reviewed and contrasted with more traditional query language systems to illustrate the expected developments. The analysis leads to a two-level taxonomy of QLs.

A user taxonomy is developed in Section 3. Section 4 is a summary of recent human factors research in the area of query languages. A specialized cost-benefit method for selecting query languages which are most appropriate for a certain class of users is introduced in Section 5.

In Section 6, a hierarchy of evaluation criteria for query languages is developed that also provides a structure for determining user requirements profiles. These criteria are used, in Section 7, for a preliminary evaluation of query languages classes and user requirements, based on existing technical studies and human factors research; unfortunately, there are still major gaps in that area. For the same reason, some of the specific recommendations derived by matching language and user class profiles have the character of research hypotheses, focusing future studies, rather than definite results. Section 9 offers a summary and conclusions.

2. TRENDS IN QUERY LANGUAGE DEVELOPMENT

The purpose of this section is to develop an understanding of some trends underlying QL development, and to apply this knowledge to the construction of an evaluation-oriented classification scheme for query languages.

2.1. A Model of Query Language Development

A review of existing QLs [VaJa 84] revealed that their development stems from two different sources: from the need for simple end user interfaces, and from theoretical conceptions of programming language or database research. The relationship between these areas is not yet fully understood but is the subject of increasing current research.

The development of QLs can be illustrated by a two-dimensional representation with the axes denoting the functional capabilities of a system and its usability. Both terms will be operationalized in Section 6 below. Roughly speaking, functional capabilities refer to what one can do with the system while usability refers to the effort of actually doing it. Clearly, usability can only be valued with a specific user group in mind.

Figure 1 summarizes our QL development trend analysis. One group of developers originating from the disciplines of programming languages and database theory concentrates on the syntactic form and semantic meaning of database interactions. Languages steming from these disciplines are characterized by the full specification of an operation by a command or a sequence of commands. Starting from formal, mathematically oriented language concepts, this group of developers has moved to "English-like" keyword languages and finally restricted natural languages. The overall trend has been toward more "user-friendliness" while preserving general functional capabilities.

The second group of language developers started from the ergonomic analysis of the interaction of computer-naive end users with computer systems. While simple systems use function keys or line-by-line prompting, more complex systems involve the use of menu selection or graphical interaction with the database. These developments represent a trend toward more functional capabilities while remaining novice-oriented.

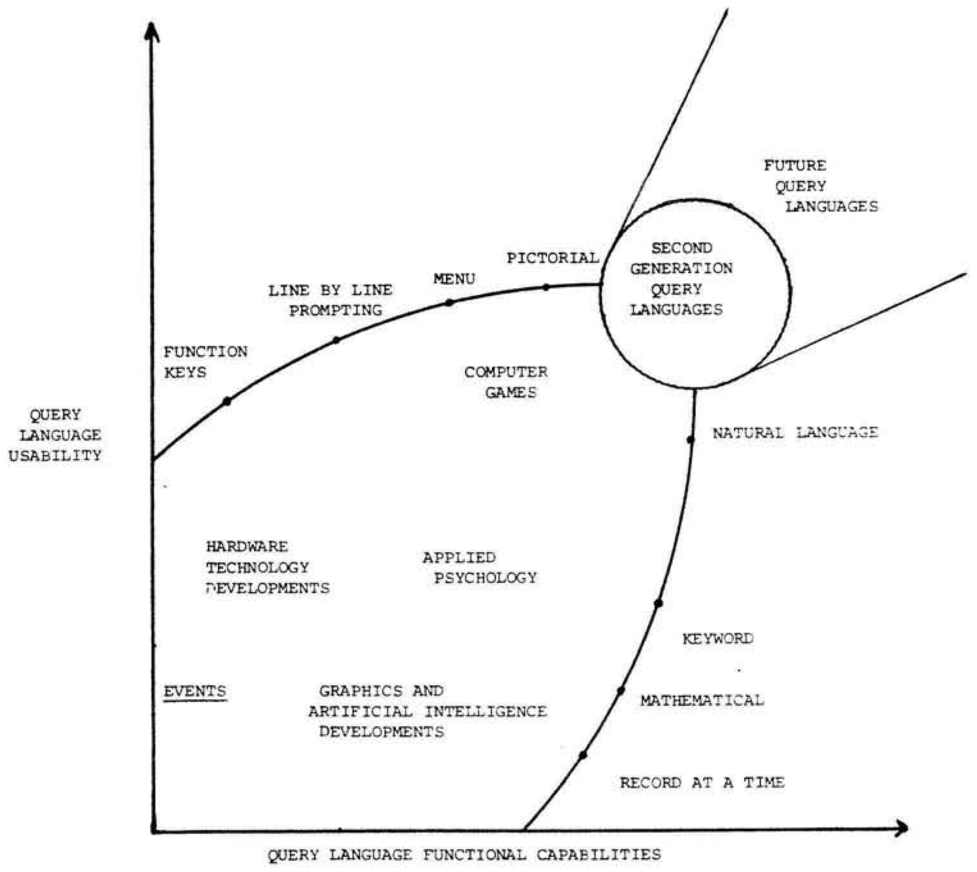


Figure 1: Query Language Development Trends

These two query language development approaches, differing in their underlying philosophy, have occurred fairly independently; one serving the more, the other the less sophisticated user. The languages that have been developed will be referred to as first generation QLs. Recent developments, however, are leading to an overlap of the usage area for both language groups. The challenge is to integrate both approaches into functionally powerful query languages for relatively unsophisticated users, the second generation QLs.

The emergence of second generation query languages is not coincidental. Rather, it has followed and has been greatly assisted by developments in many related areas. Notably:

(a) Hardware Technology Developments

Microprocessor technology and new devices such as videodiscs, content addressible memories, holographic memories, and optical storage devices allow for increased capabilities in storing and accessing data in several forms. Additionally, voice recognizers and synthesizers, eye-tracking and pointing devices lead to increased use of multi-media interactions.

(b) Developments in Graphics and Artificial Intelligence

The ability to display information in the most natural and dense form, that of an image, coupled with high-resolution display devices and the use of color, greatly contributes to the immediate comprehension of query output and the direct representation and manipulation of objects of interest [NeSp 79, FoVD 82, MOORH 76]. An important prerequisite for interactive graphics is the increased availability of high-speed communication lines and local intelligence. Research in artificial intelligence - particularly in natural language processing, expert systems, and robotics - assists in query formulation and user feedback [GaPa 80].

(c) Developments in Applied Psychology and Related Sciences

There is a new interest in "bridging the gap" between researchers and practitioners concerned with the human factors aspects of query languages, and their colleagues who are primarily concerned with the technology of query language design. Because of the tremendous possibilities for interaction with second generation query languages, the need for scientific methods to deal with the complex considerations of 'convenience', 'friendliness', and 'effectiveness' of QLs becomes apparent. Several psychological studies of QLs have been recently reported [BrSh 78, JARKE 84, REISN 75,81, SHNEI 78, SmWe83, VASSI83, WeSt 81] and considerable interest is evidenced by new professional meetings and special issues of publications focusing on human factors in computer systems [MORAN 81, HUMAN 82, VASSI 84].

(d) Success of Computer Games

Computer games have revolutionized the way people perceive interfaces to computer systems. The pleasant and simple interaction has helped remove many psychological barriers that humans have when faced with a computer system. Naturally, the success of computer games influences designers of QLs [MALON 82].

Future query language systems are expected to make increasingly sophisticated use of these developments.

2.2. Query Language Taxonomy

The trend analysis leads to a two-level classification scheme or taxonomy [VaJa 82] of query languages (Figure 2). The upper level explores the differences between first and second generation languages. It is called the senses level because the use of more senses in interaction is one of the main distinguishing factors between the two classes.

The lower level taxonomy focuses on the methods that have been used in existing QLs. The classification on the methods level applies to first generation languages only, because to date there are too few second generation systems to justify a clear subdivision of methods.

The first generation query languages are classified in two groups of four classes each: function-key, line-by-line prompting, menu selection, and graphic or pictorial for the ergonomically-oriented languages; record-at-a-time, mathematical, linear keyword, and restricted natural languages for the programming language oriented developments.

The first three language classes are typically exemplified by custom-made languages for specific applications.

The use of function keys is a limited but effective method of interaction for inexperienced users. By the press of a special key on the keyboard, a previously prepared transaction or report is processed.

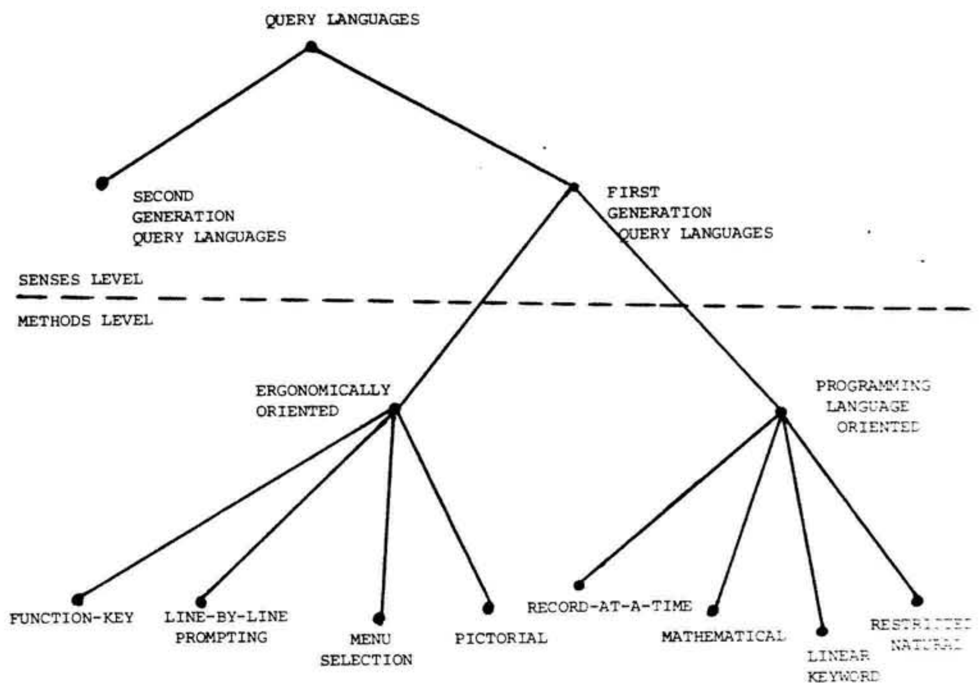


Figure 2: Query Language Taxonomy Tree

Line-by-line prompting, also called parameterized interaction [LeBl 79], is a simple system-driven dialogue. In the typical case, the user will be prompted to enter (line-at-a-time) the name of the object of interest, a field name, a comparison value, etc. The query is built-up from the user's responses.

A more sophisticated system-driven dialogue is menu selection. Here, users are required to point to their choice from a menu of options offered by the system. Menus are structured hierarchically; the choice of an option may cause the presentation of a new menu [ElNu 80].

In graphic or pictorial query languages the user can manipulate visual symbols to formulate queries. The entities and relationships in the database are represented by specific geometric shapes [McDON 75, TSICH 76, SENKO 78, ChFu 79]. This group could be considered an early version of second generation languages which did not fully succeed because the necessary hardware and understanding of user needs were not yet available.

This concludes the discussion of ergonomically oriented languages. Next are four language types evolving from the realm of programming language and database theory.

Conventional file management systems and many early database systems use a record-at-a-time logic for data retrieval. This approach is mentioned for completeness and because most QLs still use it for modification operations.

The introduction of the database concept, especially of the relational model [CODD 70], led to set-oriented data retrieval. Some query languages use the precise notation of the mathematical formalism for short and succinct expression of powerful operations. Examples include ALPHA [CODD 71], PASCAL/R [SCHMI 77], and ISBL [TODD 76]. These languages are especially suited as target languages for very high level user interfaces. Languages that use the position of the command operators and operands to convey meaning [BOYCE 75] are also included in this group.

The majority of query languages available today fall into the category of linear keyword. These languages use statements similar to a programming language like COBOL but more English-like. The commands have a definite syntax, and only words from a specific reserved list can be used. Some typical examples of linear keyword languages are SQL [ASTRA 76] and QUEL [STONE 75].

The restricted natural language mode has attracted interest in recent years. The intention is that the user can employ native natural language (e.g. English, German, French) for the interaction with the database. At least one such QL system is commercially available [AIC 82, HARRI 77] and several others are under development in research laboratories [BaBo 83, CODD 74, HENDR 78, HOEPP 83, LEHMA 78, PLATH 76, THOMP 83, WALTZ 78, WOODS 72]. Some natural language systems will engage in a dialogue with the user to resolve any ambiguity in requests [CODD 78]. Nevertheless, the natural language communication in all such state-of-the art QL systems is still far from close to person-person communication; this is the reason for the prefix "restricted".

In summary, the first generation QLs provide a very restricted interactive environment. The user has a limited hardware interface (terminal, keyboard), and a relatively artificial conceptual model of the data and their organization. The user also has a formal query language syntax (the rules of the game), and uses experience and mastery of the system to accomplish a task (how to play and win the game).

The user's visual ability while interacting with the database is limited; the objects of interest are rarely displayed directly. Rather, they are represented by formatted text, thereby not giving the user any iconic clues (what the data looks like) or spatial clues (where the data is) to help the querying process. Furthermore, the user does not employ fully the senses and cognition. For instance, in query formulation neither voice, touch, hearing, or gesture are used. Finally, the interaction is "static". A first generation query system shows little or no "intelligence" in deducing answers from incomplete, yet obvious representations of user intentions. The user may still perform a task but with limited productivity and at the possible expense of more stress, less interest, and less pleasure.

2.3. Second Generation Query Languages

Second generation QL systems attempt to incrementally utilize the human's instincts and senses. One subclass of these QL systems are referred to as "direct manipulation systems" [SHNEI 84]. Shneiderman identifies their basic features: object of interest visibility, rapid reversible actions, and replacement of command language syntax by direct manipulation of objects. Another subclass is that of "intelligent" QL systems which use advanced artificial intelligence techniques (e.g., knowledge bases) to bring man-machine interaction closer to communications between humans. The following examples indicate the directions taken in second generation language development.

A prototype system based on the principle of spatial database management, called SDMS, has been developed and implemented by Herot at Computer Corporation of America [Herot 81, 82]. The advantages of this query system include the ability to locate objects of interest by browsing and zooming and, the use of icons, color, highlighting, and arrangement. SDMS supports multiple data types: video and videodisc images, illustrations, text, and icons, which are direct representations of the underlying computer system functions. An example of the latter [HEROT 82] is an icon of a clock with the correct time from the computer system.

The environment of Cedar [BROWN 81] offers another example of a second generation QL. The language was developed at Xerox Parc as a derivative of the Smalltalk [GoRo 81] family. It is object-oriented and views data primarily through an entity-based browser with the help of dynamic windows. Window managers, in particular, are becoming standard in powerful small computer systems (often termed workstations) and it is projected that windows will be a principal component in future query languages. There is no need any more for keeping three separate screens as in SDMS; all details a user may desire to view simultaneously are kept in separate windows on the same screen.

The Architecture Machine Group at MIT [ScHu 82] is experimenting with a voice and gesture interactive system called "Put-That-There". The scenario calls for the database user to issue commands by intermixing voice, gesture (e.g. pointing), and eye-positioning at the desirable object. In the present prototype, the user needs a large screen, wears a headset microphone, has a watchband on the wrist for the gesture recognizer, and sits in a media room.

Advisory Systems [ScSl 84] support a natural language user by the application of stored knowledge that is acquired by experience. For example, the story-understanding system IPP [LEBOW 80] acquires its experience by reading stories in its domain of knowledge. Additionally, such systems also try to infer the user's intentions [CARBO 79]. In these respects, advisory systems go beyond conventional natural language front-ends and can therefore be called second-generation. Learning by experience rather than relying on a given set of stored rules also distinguishes these systems from simpler deductive front-ends to databases [JaVa 84] in which the knowledge has to be programmed.

A commercially available second generation QL is Query-By-Example [ZLOOF 77], which is based on the relational model of data. Relations are represented directly on the screen and the user moves the cursor freely along the rows and columns of the tables. Query formulation is done through the use of examples, often considered a natural education process [ThGo 75]. The major contributions and the success secrets of QBE are the "by-example" principle, the two-dimensional data representation, and the stepwise learning feature. The latter means that a novice can perform something interesting in a very short time, yet the system provides a great deal more power for the expert user.

Table 1 compares the features of second generation and first generation languages. A more detailed comparison and evaluation of both language types is given in [VaJa 84].

A word of caution is needed. Second generation QLs provide a new burden of responsibility for the application developer. For instance, the appropriate icons to represent objects, the use of color and highlighting, and the 'natural' arrangements of the objects in the database greatly influence the success of the system. Additional skills may be needed for application designers, which are not found in the traditional systems analysis and design education.

	QUERY FORMULATION		OUTPUT PRESENTATION	
	Medium	Method	Medium	Method
FIRST GENERATION QUERY LANGUAGES	keyboard function keys	function key use line-by-line prompting menu-selection graphic keyword command restricted natural language	screen printer	lists tables forms text
SECOND GENERATION QUERY LANGUAGES	keyboard function keys picking devices touch screens voice recognizers gesture tracking devices eye-positioning tracking	keyword command menus windows gesture eye positioning zooming voice browsing by-example touch deduction	screen voice synthesizer color graphics video display printer plotter	lists tables forms templates icons color highlighting images sounds (voice) arrangement text

Table 1: Characteristics of First and Second Generation Languages

3. <u>USER CLASSIFICATION</u>

Many criteria for classifying users have been proposed [CODD 74, CUFF 80, LeBl 79, MORAN 81, SHNEI 80, YORMA 77, ZLOOF 78]. Even though the classifications have several common points, their relationships have hardly been studied. [SHNEI 80] uses a two-dimensional scheme classified by syntactic and semantic knowledge. The analysis in this section includes other criteria that have been proposed elsewhere. The criteria fall into four dichotomous classifications: familiarity with programming concepts, frequency of query language usage, knowledge about the application, and range of operations required.

<u>Familiarity with programming concepts</u> is a more general concept than the often-cited distinction between programmers and non-programmers, which may lead to different and at times inconsistent interpretations [CUFF 80, GrWa 78, MORAN 81]. "High" familiarity with programming concepts refers to a user who is not afraid of computers and has acquired logical or algorithmic problem-solving abilities.

The dimension <u>frequency of system usage</u> was first introduced by [LeBl 79]. It is demonstrated below that this is one of the most important dimensions by deriving from it many of the other dimensions appearing in the literature. Frequency of use determines directly the acceptable amount of training; the more one wants to use the system, the greater an initial investment is justified. The amount of training in turn determines the typical skill level after the training period.

In the authors' opinion the transient skill levels during the training phase are of interest for QL selection only if the frequency of use is so low that each use of the system requires relearning or if the turnover of users is extremely high. Thus, the distinction between "novice" user (task: learning) and "expert" (task: routine skill) made in [MORAN 81, SCHNE 84] can be reduced to the frequency of usage dimension. We therefore use the term "novice" not only for new users but also for other infrequent users with little programming knowledge.

In combination, the two dimensions discussed above determine the user's ability to technically interact with the system; or "syntactic knowledge" [SHNEI 80]. Table 2(a) shows the relationship between the two basic dimensions and the level of interaction skill. Three user types are derived. Note that the "skilled" user is one that has a "high" score for any of the two dimensions and a "low" score for the other.

The semantic dimensions are concerned with application knowledge and range of operations of the user. In the database context, application knowledge refers to the precision of the user's conceptual model about the structure and contents of the database. The other dimension, range of operations, describes how many different types of queries the user requires. Together, these two dimensions give a picture of the task structure (semantic knowledge) of the user (Table 2(b)). Four user types are derived from this classification.

Frequency of system usage	Familiarity with programming concepts: Low	Familiarity with programming concepts: High
Low	Low (novice user)	Medium (skilled user)
High	Medium (skilled user)	High (professional user)

Table 2(a): User Types - Interaction Capability (Syntactic Knowledge) as a Function of Familiarity with Programming Concepts and Frequency of System Usage

Range of operations	Application knowledge: General	Application knowledge: Detailed
Narrow	casual user	clerical user
Broad	managerial user	application specialist user

Table 2(b): User Types - Task Structure (Semantic Knowledge) as a Function of Application Knowledge and Range of Operations

4. HUMAN FACTORS RESEARCH IN QUERY LANGUAGES

One of the most important sources of information for determining the users' QL requirements are human factors experiments. Below, recent human factors research relevant in this context is briefly reviewed. The major results are displayed in Table 3. It is not the intention of this paper to critisize these experiments. However, the reader should note that many are not "controlled" experiments that deliver statistically sound and generalizable results. For a more detailed overview of some laboratory experiments, see [REISN 81] or [SHNEI 80].

Since the now classic experiments of [REISN 75] and [ThGo 75] a number of laboratory studies and field experiments of human factors in use of query languages have been reported. For the purposes of this paper, these studies are classified as either comparisons between languages that use different methods or as studies of usability of certain features within a language type.

The first group of experiments consists of comparisons of keyword versus second generation languages [ThGo 75, GrWa 78], keyword versus positional languages [REISN 75], and keyword versus restricted natural languages [JARKE 84, SHNEI 78, SmWe 83, VASSI83]. The reader is cautioned, however, that the majority of these experiments did not intend a general comparison of methods but rather specific comparisons of languages.

REFERENCE	TYPE OF EXPERIMENT	USER CLASSES	RESEARCH QUESTION	TASKS AND TESTS	MAJOR RESULTS
Brosey and Shneiderman, 1978	lab	programmers non-programmers	relational vs. hierarchical model	comprehension memorization problem-solving	programmers are better on relations than non-programmers; hierarchies are good for natural tree applications
Damerau, 1979	field	novice application specialists	productivity of TQA/REQUEST	problem-solving	65% accepted queries
Gould and Ascher, 1975	lab	novices	query formulation process (IQF)	composition	influence of task complexity and ambiguity
Greenblatt and Waxman, 1978	lab	novices	learnability of QBE vs. SQL	composition	formulation in QBE is faster
Harris, 1977	field	application specialists	productivity of ROBOT	problem-solving	80-90% accepted queries
Jarke et al., 1984	field	novice application specialists (advisors)	productivity of natural language vs. SQL	problem-solving query acceptance	SQL has higher success rate; natural language less effort to use
Krause, 1982	field	skilled application specialists	productivity of USL	problem-solving	more than 90% successful after adaptation
Lehmann et al., 1978	field	skilled application specialists	functions of USL	problem-solving	statistics on use of various functions
Lochovsky and Tsichritzis, 1977	lab	programmers, non-programmers	comparison of 3 data models embedded in APL	composition debugging	programmers are superior; relational model best for non-programmers
Reisner, 1975	lab	programmers, non-programmers	learnability of SQL vs. SQUARE	composition	programmers are superior; SQL is better than SQUARE for beginners
Reisner, 1977	lab	programmers, non-programmers	feature analysis of SQL	composition	recommended layered structure for novice and skilled user
Shneiderman, 1978	lab	novices	productivity of natural vs. SQL	query generation	natural language user generated more invalid queries
Small and Weldon, 1983	lab with simulated processor	novices	productivity of natural vs. SQL subset	interactive problem-solving	formulation in SQL is faster
Thomas, 1976	lab	novices	use of quantifiers	various non computerized tasks	universal quantification is difficult for novices
Thomas and Gould, 1975	lab	novices	learnability of QBE	composition	67% successful after short training
Turner et al., 1984	lab	novice application specialists (advisors)	learnability of natural language vs. SQL	composition	natural language and SQL about equal in error rates
Vassiliou et al., 1984	lab	novices	learnability of natural language vs. SQL	composition	natural language less verbose; manageable language subset used
Welty and Stemple, 1981	lab	programmers, non-programmers	learnability of TABLET vs. SQL (procedurality)	composition retention	programmers are superior; TABLET is better for hard queries
Woods et al., 1972	field	novice application specialists	usability of LSNLIS (LUNAR)	problem-solving	good success rate for application-specific system

Table 3: Human Factors Experiments with Query Languages and Features

The second group of experiments concentrates on the usability of certain languages or language features within a given method. One focal point of laboratory experiments has been the keyword language SQL [REISN 77, WeSt 81, WELTY 79, THOMA 76], another the influence of conceptual data models [BrSh 78, LoTs 77, LOCHO 76]. In addition, there have been a number of field studies concerned with the usability of restricted natural languages in various settings [DAMER 79, HARRI 77, JARKE 84, KRAUS 82, LEHMA 78, WOODS 72].

As for user types, most studies in the syntactic knowledge (interaction capability) dimension focus on the novice user. Virtually all laboratory experiments are learning and retention tests and therefore apply mainly to the infrequent users. In addition, most experimenters explicitly chose subjects with little knowledge of programming concepts, often contrasting them with another group having more programming background. All experiments of this design show an overall better performance for users with programming background [REISN 75, LoTs 77, WeSt 81, TURNE 84]. As an illustration of some of the results, experiments indicate that novices:

* have difficulties with explicit quantification [THOMA 76];

* perform better with a relational model of data than with a network or hierarchy when using a keyword language embedded in APL [LoTs 77];

* learn a second generation language (QBE) faster than a keyword language (SQL) of similar power [ThGo 75, GrWa 78];

* perform better on hard queries with a more procedural approach (TABLET vs. SQL) for problem-solving than a keyword language [WeSt 81];

* can be offered a (closed) subset in a layered language [REISN 77].

The semantic classification of experimental subjects is less clear. While the laboratory experiments mostly work with students whose semantic knowledge is difficult to establish, the thrust of the field experiments is toward the application specialist, less often toward the managerial user [KRAUS 80, HARRI 77, JARKE 84, DAMER 79].

The goal in this paper is to suggest classes of languages for known user types. The survey of the human factors research in query languages revealed that it is not possible to use directly human factors research results for the pressing problem of language selection. Experiments have been criticized for sloppiness, very few repetitions, and limited coverage of issues. Most importantly for the purposes of this paper, little emphasis has been given in considering language methods (classes) in connection with user types.

In the next section, the methodology followed in this paper is presented. The methodology proposes new directions for experimental research in the QL area and provides a homogeneous structured framework.

5. QUERY LANGUAGE SELECTION PROCEDURE

The approach to query language evaluation followed here can be understood as a specialized cost-benefit analysis method. It is a cost-benefit analysis in the sense that multiple evaluation criteria are based on a simple economic model of query language usage and that the tradeoff between costs and benefits depends on the user type. The method is specialized toward the specific task of evaluating query languages from the point of view of one user. This approach has the advantage of permitting specific recommendations with available data from technical and human factors research. On the other hand, one can only expect a pre-selection of "usable" language types from such a method. When confronted with the actual language selection problem, the user community must still consider financial costs, available hardware, compatibility with existing software systems, and common use by different user types.

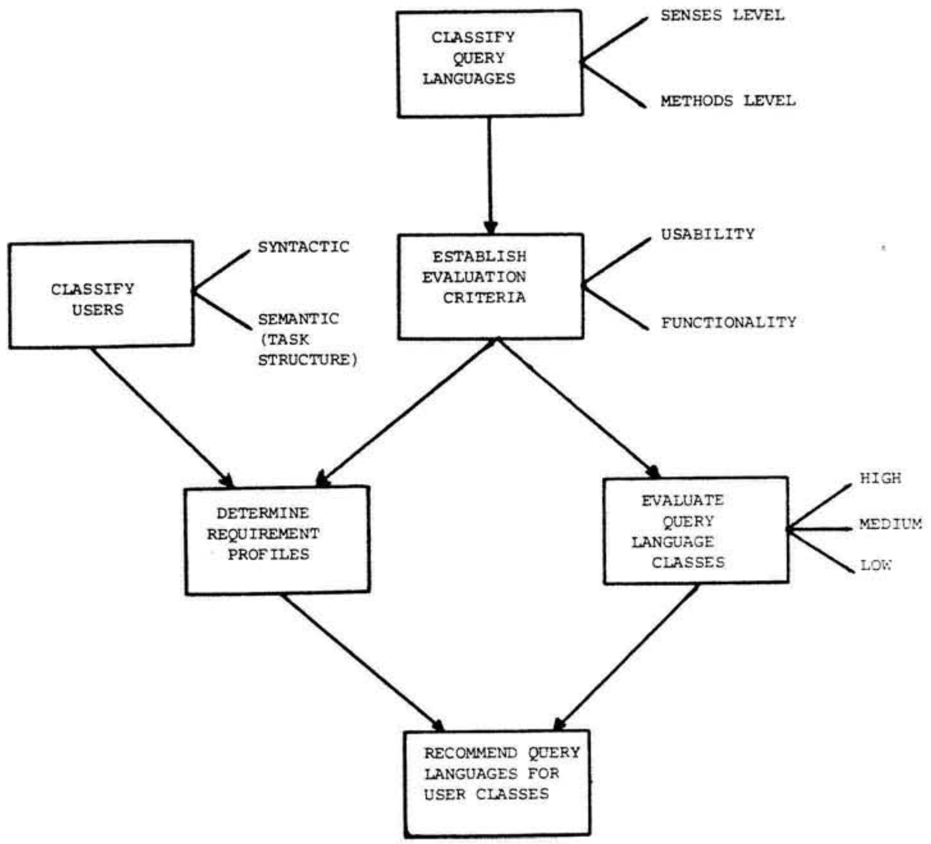
Furthermore, the fact that the query language is only a part, although a central one, of the total interface between the user and the computer system should not be overlooked. For instance, one problem arises for programming users whose QL may be embedded in a host programming language. Such users require compatibility and smooth data transfer between the two languages. [LaPi 80] differentiate four types of embedding which are listed here in ascending order of integration between host and query language: subroutine calls (DL/I[IBM 75], TOTAL-IQ [CINCO 78]); simple extension (COBOL/DML [CODAS 71]); procedural operators (C/QUEL [STONE 75], APL/EDBS [LoTs 77]); and full integration (PASCAL/R [SCHMI 77], ADAPLEX [SMITH 81]).

For interactive users, another interface problem may occur. In a hierarchy of software systems (abstract machines), an interactive QL is the outmost layer. Apart from the obvious overhead, there is a high risk of falling to a lower level system during the interaction. It is very likely that the interaction with such a lower level system (e.g., operating system) will not be consistent with the high-level QL. The availability of protection mechanisms for a smooth transition from one interaction level to another is a strong requirement for the success of advanced query language systems.

Having stated these limitations of the approach, the method of selecting query languages is now presented. Figure 3 provides an outline of the proposed method. The general problem of query language selection is characterized by a large and rapidly growing number of alternatives (query languages) and an even larger number of decision-makers (potential users). The first step in the evaluation scheme is to develop an abstraction mechanism that generates a limited number of classes. To achieve this, well-structured taxonomies of both query languages and users have been developed in Sections 2 and 3.

The next step is the development of a hierarchy of evaluation criteria. The goals for developing good evaluation criteria are:

* measuring all important costs and benefits of QL usage;
* discriminating clearly among both user and language types; and
* yielding simple criteria at the lowest level of the criteria hierarchy.


CLASSIFY QUERY LANGUAGES
SENSES LEVEL
METHODS LEVEL
CLASSIFY USERS
SYNTACTIC
SEMANTIC (TASK STRUCTURE)
ESTABLISH EVALUATION CRITERIA
USABILITY
FUNCTIONALITY
DETERMINE REQUIREMENT PROFILES
EVALUATE QUERY LANGUAGE CLASSES
HIGH
MEDIUM
LOW
RECOMMEND QUERY LANGUAGES FOR USER CLASSES

<u>Figure 3</u>: Evaluation Methodology for Query Languages

These goals are achieved by relating the evaluation criteria hierarchy to the dimensions of functional capabilities (benefit) and usability (cost) of the QL taxonomy (Section 6). Roughly speaking, the "costs" are determined by the initial training effort (the user's investment) and the effort of continuing work with the language (the production costs). The "benefits" derive from the usage of language functions such as result selection and composition, output presentation, and dynamic flexibility of interaction.

After establishing evaluation criteria, the method proceeds in Section 7 to apply them to both users (determine requirement profiles for each user type) and languages (characterize usability and functional capabilities of each language class). By matching user profiles with language profiles, the final step derives recommendations for the selection of query languages (Section 8). This "optimization step" is supported by results of human factors studies and by knowledge of technical restrictions.

The evaluation centers around several tables. For generating and filling these tables a simple Delphi-like method was used which also served as a validity check. The evaluation parameters in the tables (the column headings) were agreed upon after surveying the literature, extracting and generalizing important contributions, and reconciling differences for precise definitions. The tables were filled independently with the direct or indirect application of available human factors research results. In cases where empirical data was not available, experience and common-sense were relied upon. The common-sense approach may be misleading [MORAN 81]; often cited as armchair psychology. Therefore, in these cases the evaluations constitute <u>testable</u> <u>hypotheses</u>. The contribution in this paper, then, is the presentation of a homogenous structured framework within which the research hypotheses can be tested.

6. EVALUATION CRITERIA FOR QUERY LANGUAGES

In this section, evaluation criteria for QLs are developed. Like the QL taxonomy, the evaluation scheme is related to the trend analysis of QL development in that it relies on the two dimensions, usability and functional capabilities. However, these basic dimensions must be refined further in order to yield practical evaluation criteria.

The functional capabilities determine to a large degree the value of the information ("benefit") one can get from the system. In a QL context, this can be described by the language power and by the alternatives the user has for output presentation. On the other hand, the usability of a system (the effort or "cost" to work with it) is mainly related to the process of query formulation.

Figure 4 gives an overview of a criterion hierarchy derived from these considerations. All criteria are described below.

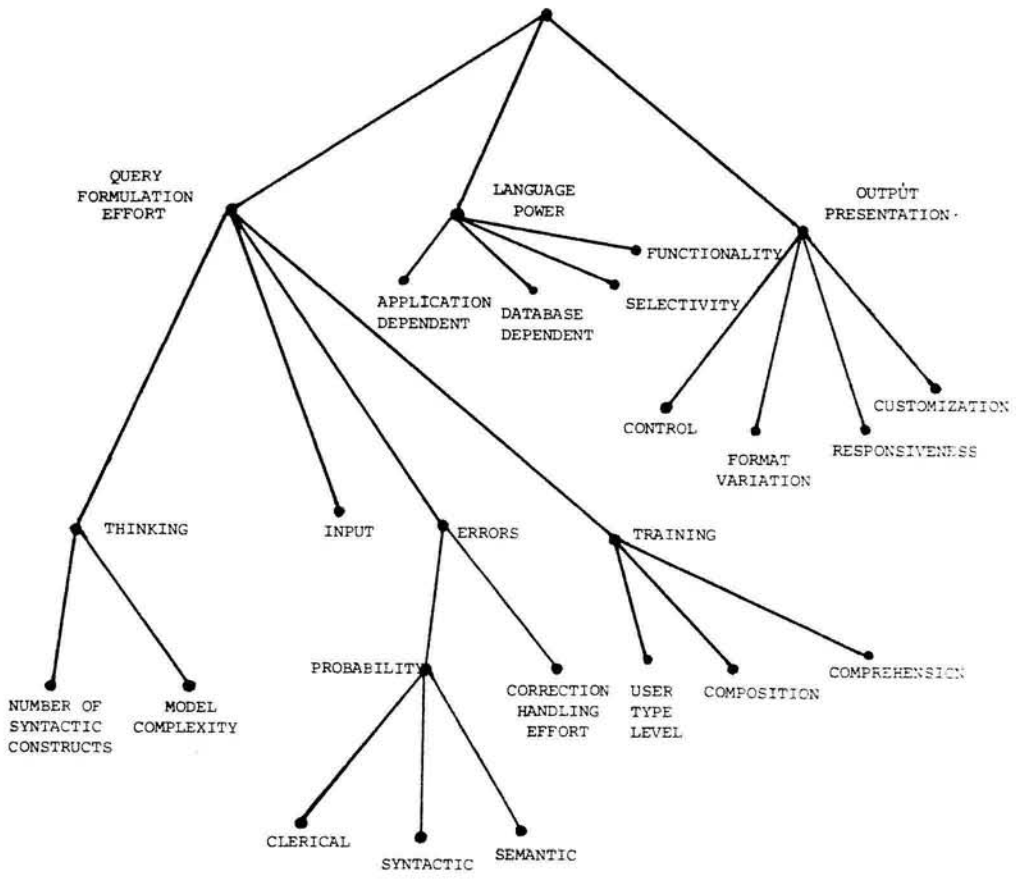


Figure 4: Query Language Evaluation Parameters

6.1. Usability Criteria

Query formulation effort describes the overall effort of the user to work with the system. This effort is the sum of the initial training effort to learn the QL system, and the repeated efforts to perform productive tasks on it. The relative importance of these two parts is determined by the frequency of system usage, both in the sense that repeated training may be necessary for infrequent users and in the sense that training time accounts for a larger percentage of overall effort for that group.

The analysis of suitability for performing tasks is based on the simple query language interaction model presented in Figure 5 (simplified from [STOHR 82]). It can be seen that productivity is determined by think time, input effort, and error handling time. While this model captures the effort of query formulation fairly well, it does not completely reflect the overall productivity in performing a task; productivity also improves through increased functional capabilities that reduce the number of queries necessary for solving a problem.

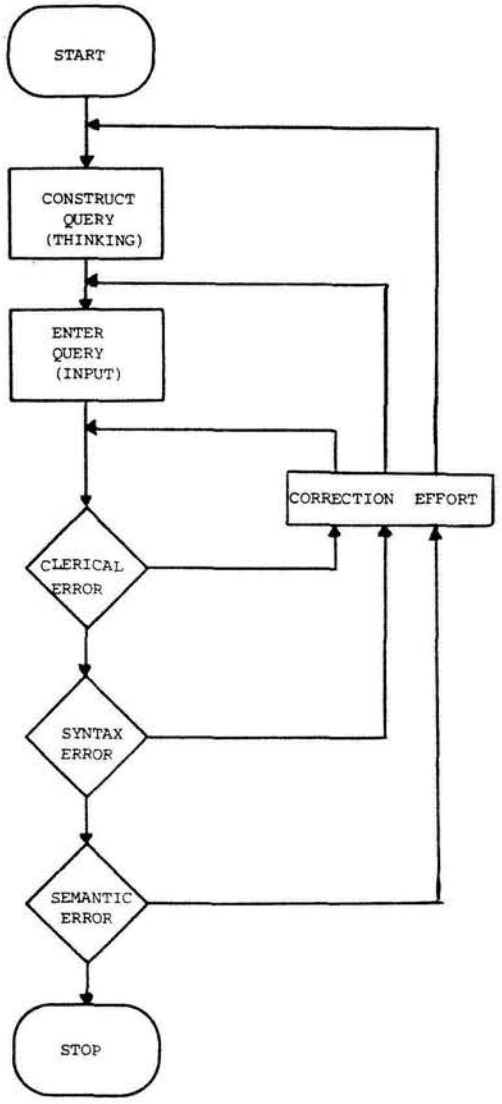


Figure 5: Query Language Interaction Model

<u>Thinking effort</u> includes the requirement that users remember <u>syntactic constructs</u>. Reisner [REISN 81] introduces the notion of a "model of the process of query writing" that users develop. This refers to the strategy the user adopts to express the request. The <u>complexity</u> of this model is part of the thinking effort.

<u>Input</u> refers to the amount of clerical effort required to express the request. When the interaction is via a keyboard, this may be measured by the number of keystrokes. Alternatively, with pointing devices a good measure of input is the number of pointed objects [LoTs 81].

Major system usage costs are induced by the efforts to handle <u>errors</u>. The overall effort can be described as a combination of the error <u>probability</u> and the <u>correction handling effort</u> after an error has occurred. Three main types of error may occur during query formulation: <u>clerical</u> (e.g. typos), <u>syntactic</u> (not following the correct syntax of the language), and <u>semantic</u> (formulation of a syntactically correct query which does not solve the task at hand).

The amount of <u>training</u> necessary for the user to perform useful tasks is a very important consideration for language acceptance. Training depends upon who learns (<u>user type level</u>) and what has to be learned (<u>composition</u> and <u>comprehension</u> of queries).

The <u>user type level</u> refers to the degree of expertise required before the user can utilize the language. Low level corresponds to novice user, medium level to skilled user, and high level corresponds to professional user.

Composition describes the degree of difficulty of learning how to formulate practically relevant queries. A facility that is easy to learn (composition) is not necessarily easy to use (thinking effort). A comparable example in programming is the language BASIC; it takes practically no time to learn, but writing a complex program in BASIC is not an easy task.

Comprehension refers to the amount of training required to understand a query formulated by another user.

6.2. Functional Capabilities Criteria

The second major parameter for query language evaluation is language power - how much a user can do with the language. The power of a QL can be described in terms of four parameters: application dependency, database dependency, selectivity, and functionality.

Application dependency is the sensitivity of a QL to the application domain; the degree to which the language has to change when it is used in a different application. Similarly, database dependency refers to the degree of language dependence on the underlying database model (e.g. network, relational).

Selectivity is the availability of operators that allow the user to specify as precisely as possible what data he wishes to retrieve.

Functionality refers to the number of different tasks for which the language can be used.

The third major criterion for QL comparison is the quality of output presentation. This is subdivided into control (ability of the user to control the pace at which the output is presented), format variation (the flexibility in selecting an output presentation format and/or redirecting output to alternative devices), responsiveness (how rapid and consistent is the system's response), and customization (the ability to have the best suited output for the application). These parameters mostly depend on the system rather than the language type. However, the philosophy behind certain language types leads to a more natural adaptation of an appropriate output feature than others.

7. EVALUATION OF LANGUAGE CLASSES AND USER REQUIREMENTS

Table 4 displays the results of applying the eighteen evaluation criteria at the leaves of the hierarchy to the nine language classes defined in Section 2.

In addition, the twelve user classes that result by combining interaction capability and task structure are related to the evaluation criteria in table 5.

In filling the table entries for usability criteria, empirical data (if available) from the human factors area was used. Functionality criteria evaluations were based on knowledge of technical restrictions of language classes.

As can be seen, the tables use a simple three-point scale (low, medium, high) for all criteria. A few finer distinctions are made using intermediate values (low-medium, medium-high). There are several reasons to restrict oneself to such a simple structure.

A finer scale would be problematic for several usability criteria in which empirical data for supporting the entries is missing or can only be applied indirectly. In such cases, the method for filling in the tables (described in Section 5) can be understood as a structured method for generating research hypotheses rather than definite answers.

Any scoring method like this requires a uniform representation of criterion values. Therefore, it seemed wise to choose a simple, uniform range of values for all criteria, even though more detailed information from empirical data was available for some of them.

7.1. <u>Query Language Evaluation</u>

Technical language specifications determine the values in table 4 which relate to functional capabilities. For the usability criteria, an effort was made to apply results from human factors research. The application of such results was mostly indirect. For instance, [ThGo 75] indicates short training requirements for the composition of correct QBE queries. A generalization of this result, places a "low" score for composition in second generation QLs (the class where QBE belongs).

QL PARAMETERS	# of Syntactic Constructs	Model Complexity		Clerical	Syntactic	Semantic	Correction Handling Effort	User Type Level	Composition	Comprehension	Application Dependence	Database Dependence	Selectivity	Functionality	Control	Format Variation	Responsiveness	Customization	Examples
				PROBABILITY OF ERROR															
	THINKING		INPUT	ERRORS				TRAINING											
QL TYPES	QUERY FORMULATION										LANGUAGE POWER				OUTPUT PRESENTATION				
FUNCTION-KEY USE	Low	Low	Low	Low	Low	Low	Low	Low	Low	Medium	High	Low	Low	Low	Low	Low	Low-Medium	High	Application Dependent
MENU-SELECTION	Low	Low	Low	Low	Low	Low-Medium	Low-Medium	Low-Medium	Low	Low-Medium	High	Low	Low-Medium	Low-Medium	Low-Medium	Low	High	High	Application Dependent
LINE-BY-LINE PROMPTING	Medium	Low	Medium	Medium	Low	Low-Medium	High	Medium	Medium	Medium	High	Low	Low-Medium	Low-High	Low	Low	Medium-High	Medium-High	Application Dependent
GRAPHIC OR PICTORIAL	Medium	Medium	Low-Medium	Low-Medium	Medium	Low-Medium	Medium	Medium	Medium-High	Medium	Low	Medium-High	Medium-High	Low-Medium	Low-High	Low-Medium	High	Medium	LSL, CUPID
RESTRICTED NATURAL	Low-High	Low	High	High	Low-Medium	High	High	Low-Medium	Low-Medium	Low	Medium-High	Low	Medium	Medium	Low	Low	Low	Medium	INTELLECT, USL, RENDEZVOUS
LINEAR KEYWORD	High	Medium-High	High	High	Medium-High	Medium-High	High	Medium	Medium-High	Low-Medium	Low	High	High	High	Medium	Medium	Low	Low-Medium	SQL, QUEL, NOMAD
RECORD AT A TIME	High	High	High	High	High	High	High	High	High	Medium	Low	High	High	High	Medium	Medium	Low	Low	
MATHEMATICAL	High	High	Medium	High	High	Medium-High	High	High	Medium-High	Medium-High	Low	High	High	High	Medium	Medium	Low	Low	ISBL, PASCAL/R
SECOND GENERATION	Medium	Low	Medium	Low-Medium	Low-Medium	Medium-High	Medium-High –	Low-Medium	Low	N/A	Medium	Low	Medium-High	Low-Medium	High	High	High	High	SDMS, VGQF QBE
OPTIMAL	Low	Low	Low	Low	Low	Low	Low	N/A	Low	Low	Low	Low	High	High	High	High	High	High	

Table 4: Taxonomy of Query Languages at the Methods Level

There are inherent dangers of misinterpretation, overgeneralization, and relience on experiments of dubious validity. Reiterating, these values should be taken as testable hypotheses in a homogeneous framework where the evaluation variables have been set out carefully.

The entries in table 4 give a more structured and precise description of the language classes introduced in Section 2. For example, it can be seen that the main advantages of second generation query languages are in the areas of output presentation, training, and query formulation. On the other hand, mathematical languages are strong in language power but awkward in query formulation and often in output presentation. Thus, these two language groups can be envisioned as complementary tools for different tasks. A more detailed description and analysis of Table 4 appears in [VaJa 84].

7.2. Determining User Requirement Profiles

An abstraction of the evaluation criteria to the basic categories of usability and functionality is made in table 5. At this level of abstraction, the user requirements are mostly inherent in the definition of a user class as given in this paper. For example, the range of operations determines the necessary selectivity and functionality. Similarly, the criterion "user type level" determines the prerequisites for working with a language directly in terms of interaction capability.

USER TYPE		EVALUATION CRITERIA		
INTERACTION CAPABILITY	TASK STRUCTURE	QUERY FORMULATION EFFORT	LANGUAGE POWER	OUTPUT PRESENTATION
NOVICE (LOW)	CASUAL	Low-Medium	Low	Low-Medium
	MANAGERIAL	Low	Medium	Medium-High
	CLERICAL	Low-Medium	Low-Medium	Low-Medium
	APPL. SPECIALIST	Medium	Medium	Medium-High
SKILLED (MEDIUM)	CASUAL	Low-Medium	Low-Medium	Low-Medium
	MANAGERIAL	Low	Medium-High	High
	CLERICAL	Low-Medium	Low-Medium	Medium
	APPL. SPECIALIST	Low-Medium	High	Medium-High
PROFESSIONAL (HIGH)	CASUAL	Medium	Low-Medium	Low-Medium
	MANAGERIAL	Low-Medium	Medium-High	High
	CLERICAL	Medium-High	High	Medium
	APPL. SPECIALIST	Medium-High	High	High

<u>Table 5</u>: Minimal Requirements by User Type

In other cases, the relationship is more indirect but still fairly obvious. For instance, users with only a general application knowledge will require a high quality of output presentation to understand the answers they get from the database.

A third possibility is that the user classification only determines the relative importance of evaluation criteria. The previously cited trade-off between training effort and day-to-day productivity is such a case.

In order to improve readability, highlights of the users' minimal requirement profiles are discussed together with recommendations in the next section.

8. RECOMMENDATIONS OF QUERY LANGUAGES FOR USER TYPES

In this section, languages are related to user types and recommendations for allocating language classes user types are made. Table 6 displays a summary of the recommendations. This table is a combination of tables 4 and 5. As an illustration of its development, consider the entries of table 6 corresponding to the novice-casual user. From table 5, the minimal requirements of this user class are determined. A consultation of table 4 shows that these requirements are met, at varying degrees, by menu selection, line-by-line prompting, second generation QLs and function keys; these constitute the entries in table 6.

Interaction capability (syntactic knowledge)	Task structure (semantic knowledge)			
	Casual User	Managerial User	Clerical User	Application Specialist User
Novice User	menu selection line-by-line prompting second generation function keys	second generation menu selection restricted natural	function keys menu selection line-by-line prompting	second generation restricted natural menu selection keyword
Skilled User	menu selection second generation function keys line-by-line prompting	second generation restricted natural menu selection	menu selection keyword function keys graphic/ pictorial	keyword restricted natural graphic/ pictorial
Professional User	(second generation)	keyword graphic/ pictorial second generation restricted natural	mathematical keyword function keys	mathematical second generation keyword positional

Table 6: Relating Language Methods to User Types

Results will be discussed for each user type and then summarized by language class.

The casual user is characterized as having only a general idea about structure and content of the database; the range of needed operations is also limited so that he or she may not require the full power of a query language [REISN 77]. Typical examples are the users of external databases like videotex [LoTs 84] or electronic funds transfer systems. Most casual users are not familiar with programming concepts (that is the reason why the lower left field of Table 6 is nearly empty) but their frequency of system usage may vary. The system must guide the infrequent casual user (novice), by offering simple menu choices or line-by-line prompts, while the more frequent user (skilled) may wish to adopt a more active role (second generation languages) or at least a faster sequence of actions (use of function keys).

The <u>managerial user</u> is probably the most demanding user type. Unwilling to "waste" time to acquire detailed knowledge of the database, he or she still wants to perform quite complicated and varied tasks, e.g., generating summary information of different types. Today, menu systems can be used for simple tasks and intermediaries must handle complex ones unless the manager has programming background and uses the database routinely (professional managerial user).

A more direct path to the database is the great promise of advanced language concepts such as second generation QLs or restricted natural language. Studies of usage of natural language show, however, that novice users may have problems with the syntactic restrictions of the language [JARKE 84] or the semantic restrictions of the database [BrSh 78]. For this reason, restricted natural language is positioned in third place for novice managerial users. Some field studies indicate that more frequent users of this type can adapt to the limitations [KRAUS 79].

Similar to the casual user, the <u>clerical user</u> (or parametric user [ZLOOF 78]) has to perform only a limited number of operations on the database, but may have detailed knowledge about the available data. The use of function keys or menus with little system guidance improves productivity for day-to-day tasks.

For the more computer-oriented or more frequent clerical user, keyword languages or even the more concise mathematical languages allow for powerful operations. Many studies exist for this user type in general (see, e.g., [EmNa 81, HUMAN 82]) but little has been published on tailoring query languages to clerical users.

As the range of operations becomes broader, the clerical user turns into an application specialist user (the synonymous "professional user" is assigned to another, syntactic, category). This type of user may have detailed knowledge of the database and requires multiple operations (data analysis, decision support) but often lacks programming background.

While conventional programming and query languages mainly support the professional (frequent and programming) application specialist, much of the recent research focuses on novice and skilled application specialists, who are expected to become a dominant group of computer users [SHNEI 84].

Studies of the use of natural language interfaces have so far been inconclusive. The authors hypothesize that natural language is competitive if restricted to a narrow domain [WOODS 72], relatively simple or tailored database structures [HARRI 77, DAMER 79], or frequent users who adapt to the limitations [LEHMA 78, KRAUS 82]. More novice users apparently have trouble if their knowledge of the data is limited and if the quality of the overall interface does not match that of the query language itself [JARKE 84]. With improved systems the somewhat optimistic preference for restricted natural language over keyword languages in the upper right field will become more realistic. For skilled users natural language offers concise formulation of some queries, but in general a keyword or mathematical language with its more powerful operators is preferable. The inclusion of second generation languages in the lower right box is due to their rapid reversible actions that support exploratory use of the database [SHNEI 82]. A promising research direction for natural language systems is their combined use with other media, such as menues [THOMP 83] or direct manipulation for those tasks where natural language is too ambiguous. Essentially, this enhancement make natural language interfaces to second generation QLs.

This section concludes with a summary of recommendations by language type. Interestingly, the pattern of query language development outlined in Section 2 repeats in the usage distribution of methods.

Formal query languages (keyword and mathematical) center around the lower right of Table 6, that is, around skilled or professional users with detailed database knowledge. Keyword languages have a more general scope than mathematical languages which in turn may be more efficient for specialized task structures such as application programming. Natural language can be thought of as an extension of this kind of (first generation) language intended for less sophisticated users.

Another line of development starts with choosing from a very limited set of functions prompted by the system and then gradually enriching the set of functions to accommodate more skilled or ambitious users.

From Table 6, it can be seen that the higher levels of both developments, second generation languages, restricted natural language, and, to a lesser degree, keyword languages and sophisticated menu selection, overlap in their usage. Currently, there is competition rather than cooperation, but the long-term trend should be towards integration.

9. CONCLUSIONS

In this paper, a methodology for choosing a database query language has been developed, and its application indicated. The method is based on a new interpretation model of the development of database query languages as evolving along two lines, one influenced by the areas of programming language and database theory, the other by human factors engineering. This observation immediately leads to a two-level classification of query languages: by the user senses employed, and by the language methods. It was demonstrated that this classification can serve as a tool for evaluating query languages in a structured manner.

The model can also be related to a comprehensive classification scheme of query language users from which most existing user categorizations can be derived.

Finally, the query language development model forms the base for a hierarchy of criteria to be used for evaluating language types and determining user requirements.

The language and user evaluations taken together lead to recommendations for assigning suitable language methods to user types and applications. However, the recommendations cannot be fully general at this time, since -- as the paper demonstrates, there are substantial gaps in query language research. Scientifically good human factors experiments are sparse. Furthermore, conclusive research to support the user profiles and language selection recommendations is not presently available. Perhaps the most important contribution of this paper, is the identification of a well-structured framework for testing the recommendations as research hypotheses. Until more empirical data is collected however, it is the authors' opinion that this work is a useful tool for query language developers and users.

Much remains to be done. As noted, few experiments in the field have been directed towards comparison at the general methods level, and experiments with natural language systems have not been conclusive. Psychological models of query formulation are only in the initial stage. There seems to be little research on database usage for clerical users and on the long-term performance of skilled users. Finally, second generation languages are in too early a stage of development for the differentiation of methods within this group to be clearly understandable. It is hoped that the framework developed in this paper contributes to structuring research questions to be answered.

Each single language type will have problems accommodating the variety of user types discussed in this paper. The authors envision that future query languages will employ multiple interaction modes in order to have a broader coverage and usability. For data consistency and system efficiency, such user interfaces must be integrated with well understood, formal query or database programming languages. Such systems will also provide facilities allowing users to customize the interaction to their own needs and preferences.

REFERENCES

[AIC 82]
Artificial Intelligence Corp., "Intellect Query System", Reference Manual, (1982).

[ASTRA 76]
Astrahan et al., "SYSTEM-R: Relational Approach to Database Management", ACM Transactions on Database Systems, Vol. 1,2, June, (1976).

[BaBo 83]
Bates, M., Bobrow,R.J., "A Transportable Natural Language Interface for Information Retrieval", in Proceedings 6th ACM-SIGIR Conference, June 1983.

[BCS 81]
British Computer Society, "QUERY LANGUAGES - A Unified Approach", Report of the British Computer Society Query Languages Group, Heyden Publ, Samet (ed.), (1981).

[BOYCE 75]
Boyce et al, "Specifying Queries as Relational Expressions: The SQUARE Data Sublanguage", Communications of the ACM, 18, (1975).

[BrSh 78]
Brosey, M., Shneiderman, B., "Two Experimental Comparisons of Relationl and Hierarchical Database Models", International Journal for Man-Machine Studies, Vol. 10, (1978).

[BROWN 81]
Brown, R.M., Cattell, R.G.G., Suzuki, N., "The Cedar DBMS: A Preliminary Report", in Proceedings ACM-SIGMOD Conference, Ann Arbor 1981, 205-211.

[CARBO 79]
Carbonell, J., Subjective Understanding: Computer Models of Belief Systems, Unpublished Ph.D. Dissertation, Yale University 1979.

[ChFu 79]
Chang, N.S., Fu, K.S., "Query-by-Pictorial Example", Proc. IEEE Computer Society Third Intl. COMPSAC '79, (1979).

[CINCO 78]
CINCOM Systems, Inc., "Total Information System, The Next Generation of Software", (1978).

[CODAS 71]
"Codasyl Data Base Task Group Report", in Cont. Data System Languages, (ACM), New York, (1971).

[CODD 70]
Codd, E.F., "A Relational Model of Data for Large Shared Databanks" Communications, ACM 13,6, (1970).

[CODD 71]
Codd, E.F., "A Data Base Sublanguage Founded on the Relational Calculus", Proceedings of ACM SIGFIDET Workshop on Data Description, Access and Control, San Diego, Codd and Dean (Ed.), (1979).

[CODD 74]
Codd, E.F., "Seven Steps to Rendezvous with the Casual User", Data Base Management, Klimbie and Koffeman (Eds), North-Holland, (1974), 179-199

[CODD 78]
Codd E.F., "How About Rec{ently?", Databases: Improving Usability and Responsiveness, Shneiderman (Ed.), Academic Press, 1978

[CUFF 80]
Cuff,R.N., "On Casual Users", Int. Journal Man-Machine Communications, 12, (1980), 163-187.

[DAMER 79]
Damerau, F.J., "The Transformational Question Answering (TQA) System Operating Statistics", IBM Research Report, RC 7739, (1979).

[ElNu 80]
Ellis,C.A., and Nutt,G.J., "Office Information Systems and Computer Science", ACM Computing Surveys, 12, (1980).

[EmNa 81]
D.W.Embley, G.Nagy, "Behavioural Aspects of Text Editors", ACM Computing Surveys 13 (1981), 33-70

[EnGr 75]
Engel,E., and Granda, R.E., "Guidelines for man/display interfaces", IBM Poughkeepsie Lab., Tech. report, TR 0002720, N.Y., December, (1975).

[FiNe 76]
Fields, C., Negroponte, N., "Using New Clues to Find Data", Proceedings of the International Converence on Very Large Data Bases (VLDB), (1976).

[FoVD 82]
Foley,J.D., Van Dam,A., "Fundamentals of Interactive Computer Graphics", Addison-Wesley, (1982).

[GaPa 80]
A.Gable, C.V.Page, "The Use of Artificial Intelligence Techniques in Computer-Assisted Instruction", International Journal of Man-Machine Studies 12 (1980), 259-282.

[GoAs 75]
Gould, J.D., Ascher, R., "Use of an IQF-like query language by non-programmers", IBM Thomas J. Watson Research Center Technical Report RC5279, Yorktown Heights 1975.

[GoRo 81]
Goldberg, A., Robson, Special Issue on Smalltalk, BYTE Magazine, August 1981.

[GrWa 78]
Greenblatt, D. and Waxman,J., "A Study of Three Database Query Languages", Databases: Improving Usability and Responsiveness, B. Shneiderman (ed), Academic Press, New York, (1978), 77-97.

[HARRI 77]
Harris, L.R., "User Oriented Database Query with the ROBOT natural Language Query System", International Journal of Man-Machine Studies, 9, (1979).

[HENDR 78]
Hendrix, G.G, E.D. Sacerdoti, D.Sagalowicz, J. Slocum, "Developing a Natural Language Interface to Complex Data", ACM Transactions on Database Systems, vol.3, (1978), 105-147.

[HEROT 81]
Herot, C.F., "Spatial Management of Data", ACM Transactions on Database Systems, 5, (1981), 493-513.

[HEROT 82]
Herot, C.F., "Graphical User Interfaces", in Proceedings of NYU Symposium on User Interfaces, New York, May, (1982).

[HOEPP 83]
Hoeppner, W., Christaller, T., Marburger, H., Morik, K., Nebel, B., O'Leary, M., Wahlster, W., "Beyond Domain-Independence: Experience with the Development of a German Language Access System to Highly Diverse Background Systems", in Proceedings Eighth International Joint Conference on Artificial Intelligence, Karlsruhe 1983, 588-594.

[HUMAN 82]
Human Factors in Computer Systems, Gaithersburg, MD, (1982).

[IBM 75]
"IBM Information Management System/Virtual Storage (IMS/VS), General Information Manual", G 20-1260-3, IBM Corp., (1975).

[JARKE 84]
M. Jarke, J.A. Turner, E.A. Stohr, Y. Vassiliou, N.H. White, K. Michielsen, "A Field Evaluation of Natural Language for Data Retrieval", IEEE Transactions on Software Engineering, forthcoming 1984.

[JaVa 84]
M. Jarke, Y. Vassiliou, "Coupling Expert Systems and Database Management Systems", in Artificial Intelligence Applications for Business, W. Reitman (ed.), Ablex, Norwood, NJ, (1984), 65-85.

[KRAUS 79]
J. Krause, "Preliminary Results of a User Study with the 'User Specialty Languages' System, and Consequences for the Architecture of Natural Language Interfaces", IBM Heidelberg Scientific Center TR 79.04.003, (1979)

[KRAUS 80]
Krause, J., "Natural Language Access to Information Systems: An Evaluation Study of its Acceptance by End Users", Information Systems 4 (1980), 297-318.

[KRAUS 82]
Krause, J., Mensch-Maschine-Kommunikation in natuerlicher Sprache, Niemeyer, Tuebingen 1982.

[LaPi 75]
Lacroix, M., Pirotte, A., "ILL: An English Structured Query Language for Relational Databases", Architecure and Models in DBMS's, Njissen (Ed.), North-Holland, (1975).

[LaPi 76]
Lacroix, M., Pirotte, A., "Example Queries in Relational Languages", Unpublished Technical Note, (1976).

[LaPi 77]
Lacroix, M., Pirotte, A., "Domain-Oriented Relational Languages", Proc. Intl. Conf. Very Large Data Bases, (1977), 370-378.

[LaPi 80]
Lacroix, M., Pirotte, A., "User Interfaces for Database Application Programming", MBLE Research Laboratory, Brussels, (1980).

[LEBOW 80]
Lebowitz, M., Generalization and Memory in an Integrated Understanding System, unpublished Ph.D. Dissertation, Yale University 1980.

[LeBl 79]
H.Lehmann, A.Blaser, "Query Languages in Data Base Systems", IBM Heidleberg Scientific Center TR 79.07.004, (1979)

[LEHMA 77]
Lehmann et al, "Language Facilities of USL German, Version III", IBM Heidelberg Scientific Center, TN. 77.04, (1977).

[LEHMA 78]
H.Lehmann, N.Ott, M.Zoeppritz, "User Experiments with Natural Language for Data Base Access", Proc. 7th Int. Conf. on Computational Linguistics, Bergen, (1978).

[LeSa 74]
Leavenworth, B.M., Sammet, J., "An Overview of Non-Procedural Languages", Sigplan Notices. 9, (1974).

[LOCHO 76]
Lochovsky, F.H., "Data Base Management System User Performance Variables", Ph.D. Thesis, University of Toronto, (1976).

[LoTs 77]]
Lochovsky, F.H., Tsichritzis, D.C., "User Performance Consideration in DBMS Selection", in Proceedings of ACM SIGMOD, (1977), p. 128-134.

[LoTs 81]
Lochovsky F.H., Tsichritzis, D.C., "Interactive Query Languages for External Databases", CSRG Technical Memo, University of Toronto, (1981).

[LoTs 84]
Lochovsky F.H., Tsichritzis, D.C., "Querying External Databases", Human Factors and Interactive Computer Systems, Y.Vassiliou (ed.), Ablex, Norwood, NJ, 1984.

[MALON 82]
Malone,T., "Heuristics for Designing Enjoyable User Interfaces: Lessons from Computer Games", Proceedings, Human Factors in Computer Systems, Gaithensburg, (1982), 63-68.

[McDON 75]
McDonald, N.H., "CUPID: A Graphic Oriented Facility for Support of Non-Programmer Interactions with a Database, Memo ERL-M563, Ph.d. Thesis, University of California, Berkeley, (1975).

[McDON 81]
McDonald, N.H. , J.P. McNally, "Video Graphic Query Facility Database Design", in Proceedings of the ACM SIGMOD/SIGSMALL Workshop on Databases for Small Systems, Orlando, Florida, October (1981).

[McMc 82]
McDonald, N.H., McNally, L.P., "Query Language Feature Analysis by Usability", unpublished paper, University of South Florida, (1982).

[MOORH 76]
Moorhead, W.G., "GXRAM-A Relational Data Base Interface for Graphics", Tech. Rep. RJ1735, IBM Research Lab, San Jose, (1976).

[MORAN 81]
T.P.Moran, "Guest Editor's Introduction: An Applied Psychology of the User", ACM Computing Surveys 13 (1981), 1-12

[NeSp 79]
Newman, W.S., Sproull, R.F., "Principles of Interactive Computer Graphics", McGraw-Hill, New York, (1979).

[PLATH 76]
Plath, W.J., "REQUEST: A Natural Language Question Answering System", IBM J. Res. Development, vol. 20, (1976).

[REISN 75]
Reisner, P., "Human Factors Evaluation of Two Data Base Query Languages: SQUARE and SEQUEL", Proceedings of NCC, Vol. 44, (1975).

[REISN 77]
Reisner, P., "Use of Psychological Experimentation as an aid to Development of Query Language", IEEE Transactions of Software Engineering, SE-3, 3, (1977), 218-229

[REISN 81]
P.Reisner, "Human Factors Studies of Database Query Languages: A Survey and Assessment", ACM Computing Surveys 13 (1981), 13-32

[RTG 82]
"Final Report of the ANSI/X3/SPARC DBS-SG Relational Database Task Group", M.Brodie and J.W.Schmidt (eds), SIGMOD Record 12, 4, (1982).

[ScSl 84]
Schank, R.C., Slade, S., "Advisory Systems", in Artificial Intelligence Applications for Business, W.Reitman (ed.), Ablex, Norwood, NJ, 1984, 249-265.

[ScHu 82]
Schmandt,C., Hulteen,E.A., "The Intelligent Voice-Interactive Interface", Proceedings of Human factors in Computer Systems, Gaithersburg, Md, (1982).

[SCHMI 77]
Schmidt, J.W., "Some High-level Language Constructs for Data of Type Relation", Transacions on Database Systems, 2, (1977), pp.247-261.

[SCHNE 84]
Schneider, M., "Ergonomic Considerations in the Design of Control Languages", Human Factors and Interactive Computer Systems, Y.Vassiliou (ed.), Ablex, Norwood, NJ, 1984.

[SENKO 78]
Senko, M.E., "DIAM II with FORAL LP: Making Pointed Queries with Light Pen", Information Processing 77, North-Holland, (1977).

[SHIPM 81]
Shipman, "The Functional Data Model and the Data language DAPLEX", ACM Transactions on Database Systems, 6,1, (1981).

[SHNEI 78]
Shneiderman, B., "Improving the Human Factor Aspect of Database Interactions", ACM Transactions on Database Systems, 3, (1978).

[SHNEI 80]
Shneiderman, B., Software Psychology, Cambridge/Mass., (1980).

[SHNEI 84]
Shneiderman, B., "The Future of Interactive Systems and the Emergence of Direct Manipulation", Human Factors and Interactive Computer Systems, Y.Vassiliou (ed.), Ablex, Norwood, NJ, 1984.

[SmWe 83]
Small, D.W., Weldon, L.J. "An Experimental Comparison of Natural and Structured Query Languages", Human Factors 25, 3 (1983), 253-263.

[SMITH 81]
Smith J.M, Fox S., Landers T., "Reference Manual for ADAPLEX", Tech. Report CCA-81-02, January (1981).

[STOHR 82]
E.A.Stohr, J.A.Turner, Y.Vassiliou, N.H.White, "Research in Natural Language Retrieval Systems", 15th Ann.Hawaii Int.Conf. on System Sciences, Hawaii 1982

[STONE 75]
Stonebraker, M.R., et al, "The Design and Implementation of INGRES", ACM Transaction on Database Systems, vol. 1, no. 3, September, (1976).

[StRo 77]
Stonebraker, M.R., Rowe, L.A., "Observations on Data Manipulation Languages and Their Embedding in General Purpose Programming Languages", Proc. ACM Intl. Conf. Very Large Data Bases, (1977), 128-143.

[ThGo 75]
Thomas, J.C. and Gould, J.D., "A Psychological Study of Query by Example", Proceedings of NCC, Vol. 44, (1975).

[THOMA 76]
Thomas, J.C., "Quantifiers and Question-Asking", IBM Research Report, RC 5866, T.J.Watson Research Laboratory, Yorktown Heights, N.Y., (1976).

[THOMP 83]
Thompson, C.W., Ross, K.M., Tennant, H.R., Saenz, R.M., "Building Usable Menu-Based Natural Language Interfaces to Databases", in Proceedings 9th VLDB Conference, Florence 1983, 43-55.

[TODD 76]
Todd, S.J.P., "The Peterlee Relational Test Vehicle - A System Overview", IBM Systems Journal, 15, (1976).

[TSICH 76]
Tsichritzis, D.C., "LSL: A Link and Selector Language", Proceedings of the ACM SIGMOD, Washington, (1976).

[TURNE 84]
Turner,J., Jarke,M., Stohr,T, Vassiliou,Y., and White,N., "Using Restricted Natural Language for Data Retrieval - A Plan for Field Evaluation", Human Factors and Interactive Computer Systems, Y.Vassiliou (ed.), Ablex, Norwood, NJ, 1984.

[VASSI 83]
Vassiliou, Y., Jarke, M., Stohr, E.A., Turner, J.A., White, N.H., "Natural Language for Database Queries: A Laboratory Study", MIS Quarterly 7, 4 (1983), 47-61.

[VaJa 84]
Vassiliou Y., Jarke M., "Query Languages - a Taxonomy", Human Factors and Interactive Computer Systems, Y.Vassiliou (ed.), Ablex, Norwood, NJ, 1984.

[WALTZ 78]
Waltz,D.L., "An English Language Question Answering System for a Large Relational Database", Communications of the ACM, 21, (1978).

[WELTY 79]
Welty, C., "A Comparison of a Procedural and a Non-Procedural Query Language: Syntactic Metrics and Human Factors", Ph.D. Dissertation, Univ. Of Mass at Amherst, (1979).

[WeSt 81]
Welty, C., Stemple, D.W., "Human Factors Comparison of a Procedural and a Non-Procedural Query Language", ACM Trasactions on Database Systems, (1981).

[WOODS 77]
Woods, W.A., "Lunar Rocks in Natural English: Explorations in Natural Language Question Answering", Linguistic Structures Processing, Zampolli (ed.), North-Holland, (1977).

[WOODS 72]
Woods, W.A., Kaplan, R.M., and Nash-Webber, B., "The Lunar Sciences Natural Language Informaion System", Bolt Beranek and Newman, Cambridge, Mass, June, (1972).

[YORMA 77]
Yormark, B., "The ANSI/X3/SPARC/SGDBMS Architecture", The Ansi/Sparc DBMS Model, Jardine, (ed.), North-Holland, Amsterdam, (1977), 521.

[ZLOOF 77]
Zloof, M., "Query By Example: A Data Base Language", IBM Systems Journal, 4, (1977).

[ZLOOF 78]
Zloof, M., "Design Aspects of the Query-by-Example Data Base Management Language", in Databases: Improving Usability and Responsiveness, Ben Shneiderman, (ed.), (1978).