QUERY LANGUAGES - A TAXONOMY

Yannis Vassiliou and Matthias Jarke

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

Working Paper Series

CRIS #34

GBA #82-35 (CR)

This paper was originally presented at:

NYU Symposium on User Interfaces
New York, NY, May, 1982

1. INTRODUCTION

A hypothetical futuristic scenario for querying a database begins with a user sitting on the couch, sipping some coffee and looking at the fancy display of her personal computer - or, is it her television set? She scratches her head and instructs the personal computer to display the monthly sales for her company, categorized by department. Just like her best friend who always understands her intentions, the personal computer presents an answer in her favorite format. To break the monotony, she issues her next request practicing her French - the reply by simulated voice is in the same language, a little bad in the accent.

But let us deal with the reality of the present. The Architecture Machine Group at MIT [ScHu 82] is experimenting with a voice and gesture interactive system called "Put-That-There". This real scenario calls for the database user to issue her commands by voice and gesture (e.g. pointing) and by positioning her eyes at the desirable object in a large screen. The user wears a headset microphone, has a watchband on her wrist for the gesture recognizer, is wired as if to be electrocuted and sits in a media room.

A third scenario is required to describe the most realistic circumstances for interactively querying a database. The user sits in front of her CRT terminal, makes sure she has the right description of the database - which field appears where, what relationships exist between the data - and starts typing the

ABSTRACT

A taxonomy of database query languages is presented based on an analysis of existing languages and current trends in research and practice. The categorization of query languages is hierarchical. At the senses level, languages are distinguished according to the clues and senses employed by the user. At a more detailed level, the methods level, languages are categorized according to the basic concepts, methodologies and conceptual user models which are employed. In addition to this categorization, the taxonomy includes schemata of evaluation criteria to be used in language comparisons. Using the results of human factors studies, the evaluation schemata are applied to typical user profiles to derive usability recommendations. The user profiles are part of a comprehensive classification scheme of query language users also developed in this paper.

ACKNOWLEDGMENTS

We are grateful to our colleague Margrethe Olson for her comments and suggestions on the first draft. We would also like to thank Martha Hart Deese for doing an excellent job typing the tables in this paper.

query on the keyboard. Is it FIND or GET I have to use here, do I put the GROUP BY before the WHERE? Then appears the message: "ILLEGAL COMMAND". After two more tries she gets the answer, but is this what she wanted?

What all of these scenarios have in common is a natural identification of our hypothetical user as a "novice"; an infrequent user with little or no programming knowledge. The novice user most likely wants to stay away from a programming language, pays little attention to precision and logic, has little sense of algorithmic thinking, and expects the system to respond to his or her personal style. Our preference for focusing on the novice user in our scenarios should not be taken as an intention to ignore all other types of users. Rather, it should be taken as a realization that the population of novice users is increasing at least proportionally with the growth of computer equipment and information systems. Regardless, other types of system users can benefit if the design of their database interaction languages takes into consideration the human factors principles of a language system designed for novices.

The three scenarios give an indication of where some hypothesize we are going, what direction we are taking, and where we are today with database query languages.

The number and variety of query languages available or under development has become so confusing that a framework is needed for classifying and evaluating query languages in terms

of their usability by different type of users. Our goal in this paper is to establish such a framework and to derive recommendations from it. In this respect, our work goes beyond previous surveys and categorizations that have appeared in the literature ([BCS 81], [LaPi 76,77,80], [LeBl 79], [LeSa 74], [LoTs 81], [McMc 82], [StRo 77]).

Our taxonomy is a two-level hierarchy grounded in the observation of two converging trends which seem to govern the recent development of query languages; one originating from formal language theory and the other from human factors engineering. Section 2 expands on this idea after giving the necessary definitions and narrowing the scope of the paper to a manageable task. In Section 3, describing the senses level, languages are classified by the senses employed for interaction with the database. Some recent experimental systems are reviewed to illustrate the type of developments to be expected at this level. These systems are contrasted with more conventional query language systems in a uniform evaluation scheme.

Section 4 describes the low level of the hierarchy, the methods level. Languages are classified by the methods of query formulation. Each method is evaluated in a (qualitative) cost-benefit scheme based on the effort for using the system (cost) and the functional power of the language (benefit).

In Section 5, a unified approach for classifying query language users is developed. The results of Section 4 and evidence from human factors experiments are used to derive recommendations concerning the usability of language methods by user type. Section 6 offers our conclusions and directions for future research.

2. BASIC CONCEPTS AND METHODOLOGY

In recent years, the focus of computer systems has shifted from number-crunching and mass data processing to the management of data as a strategic resource in the organization. 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 categorization of languages for database interactions is presented in the appendix (Table 1). A central interaction mode of users with a database system is through a query language.

A Query Language (QL) cannot be defined with precision. Roughly, it is a high-level computer language for the retrieval of data held in databases and files [BCS 81]. In this paper we focus on languages for querying databases rather than arbitrary collections of files. A query language is usually assumed to be interactive, on-line and able to support queries which are ad-hoc (not predefined). Interaction via a QL tends to be of

low complexity and the displayed answer tends to be limited to a few lines. It is often assumed that the principal users of QLs are the ones' that do not have technical expertise. The important performance factor is query development time as contrasted with system efficiency. By construction, query languages are usually of limited efficiency in run time. Successful QLs are those that set pragmatic goals in terms of usage and usability, rather than those that claim to be all-purpose tools for all types of users. For instance, the production of 10000 mailing labels in special formats would rarely be attempted in a QL.

The taxonomy is based on our observation that current query languages have developed from two sources. One group of developers, originating from the the areas of programming languages and database theory concentrates on the syntactic form and semantic meaning of database interactions. Starting from formal, mathematically oriented language concepts, this group has moved to "English-like" keyword languages and finally restricted natural languages. The overall trend has been towards more "user-friendliness". The second group of language developers started from the ergonomic analysis of interaction technology of novice users with computer systems. Beginning with simple 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 "functionality" while remaining novice-oriented.

In conventional query languages ("first generation QLs"), these two areas have developed fairly independently; one serving the more, the other the less sophisticated user. Recent developments, however, lead to an overlap of the usage area for both language groups. The challenge is to integrate both approaches to functionally powerful query languages for relatively unsophisticated users. We call these systems "second generation QLs".

The upper level of our taxonomy explores in more detail the differences between first and second generation QLs. It is called senses level because the use of more senses in interaction is one of the main advantages in second generation QLs. The lower level taxonomy focuses on the methods that have been used in existing, mostly first generation QLs.

Each of the categories on both taxonomy levels have strengths and weaknesses which must be traded off during the process of choosing a query language. First, the important evaluation parameters are established. Next, the values of these parameters are determined for each language category. Finally, the multi-dimensional evaluation can be related to the user and task profile of the application to determine the most suitable query language.

Our evaluation of query languages centers around several tables. For generating and filling these tables we used a Delphi-like method serving also as a validity check of our findings. First we made the taxonomies of query languages. The

evaluation parameters in the tables were agreed upon after surveying the literature, extracting and generalizing the important contributions, and reconciling our differences for precise definitions. We filled-in the table entries independently and had agreement in approximately 90% of the entries. The only case where we had the exact opposite entries in a column was caused by an ambiguous definition of the column heading; the column was subsequently dropped.

Our purpose in this paper is not to suggest that a particular language is the 'best' in terms of user performance [LOCHO 76, LoTs 77] or coverage of features. Rather, we believe the important question is: what factors make a QL a better language for a certain class of users and applications. We therefore survey psychologically oriented studies of query languages and relate their major results and conclusions to our taxonomy. Although these results should be taken with caution, we are advocates of psychologically oriented experimentation with query languages. Quoting from [SHNEI 78], '...each result is like a small tile contributing to an emerging mosaic of user behavior...'. One purpose of a taxonomy for query languages is to identify areas where further human factors experiments are most urgently needed.

In the next section we begin our hierarchical taxonomy by presenting the highest level; languages are categorized according to the different clues and user senses employed during the interaction of the user with the system.

3. THE SENSES LEVEL

The first generation query languages provide a very restricted interaction 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 his experience and mastery of the system to accomplish his 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. Furthermore, the user does not employ fully his senses and cognition. For instance, in query formulation he is not using voice, touch, hearing, or gesture. Additionally, his knowledge of spatiality is not utilized. The user may still perform his task but with limited productivity and at the expense of more stress, less interest, and less pleasure.

Considerations such as these have led to second generation QL systems which attempt to incrementally utilize the human's instincts and senses. Shneiderman [SHNEI 82] refers to these systems as "direct manipulation systems". He states that their basic features are: object of interest visibility, rapid reversible actions, and replacement of command language syntax by direct manipulation of objects. In this section we review some of these QL systems and present a categorization in relation to first generation QLs.

The emergence of the 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

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. Finally, the emergence of microprocessor technology assists in the proliferation of telecommunication networks and distributed processing.

(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]. Research in artificial intelligence - particularly in natural language processing, expert systems and robotics - assists in query formulation and user feedback [GaPa 80]. In general, the second generation QL systems appear more "intelligent" to the user.

(c)- Developments in Applied Psychology and Related Sciences

It seems there is a new interest in bridging the gap between the researchers and practitioners mostly 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 condiderations of 'convenience', 'friendliness', and 'effectiveness' of QLs becomes apparent. Several psychological studies of QLs have been recently reported [BrSh 78, REISN 75,81, SHNEI 78, WeSt 81] and considerable interest has been generated in human factors of query languages as evidenced by new professional meetings and special issues of academic computer science and professional publications focusing on human factors.

(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 while faced with a computer system. Naturally, the success of computer games influences designers of QLs [MALON 82].

It is hard to accurately identify the person(s) responsible for the emergence of second generation query languages. It seems though, that Negroponte and Fields deserve major credit with the presentation at the Conference on Very Large Data Bases [FiNe 76]. They emphasized that, traditional QLs use only symbolic clues like attribute names, attribute values and predefined data relationships in order to retrieve data. According to [FiNe 76], the next step is to go beyond symbolic clues to other, more natural clues, such as spatial (where the data is) and iconic (how the data look like). Browsing or navigating through the database takes a new dimension; very different from the one described by Bachman in his Turing Award lecture where the programmer is portrayed as a navigator finding his way through a network structured database. The use of spatial knowledge seems especially significant among all the interesting ideas in [FiNe 76]. There is ample documentation in psychological research substantiating the power of spatial organization and the ability of humans to remember by location and appearance [FiNe 76].

A prototype system based on the principle of spatial database management, called SDMS, has been developed and implemented by Chris Herot at CCA [Herot 81, 82]. It is currently running on INGRES [STONE 75] with planned extensions to other database systems [CCA 77]. The advantages of the system include the ability to locate objects of interest by browsing and zooming, and the use of icons, color, highlighting and arrangement. SDMS supports a multiplicity of 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.

Another system based on the same principles and on a multi-media user interface to databases is that of McDonald [McDON 81, 82a, 82b]. The purpose of this project is to develop a query capability for a database by making use of videodisc technology and interactive computer graphics. Motion and still pictures of data are presented to the user, together with browsing capabilities and spatial information. The user has the option of using the keyboard, a joystick, or a touch-panel. A very interesting feature of the system is the flexibility the user has in personalizing the method of querying the database.

McDonald's system requires the user to interact with the machine via three screens. This simultaneous viewing capability may have some disadvantages considering inevitable head movement and possible user confusion. In several early experiences in using the system in a laboratory setting, McDonald reports difficulties providing input for multi-media interactions. She also reports that touch-panels, keyboard, and joystick follow this order in user preference. The observation that joysticks are not very convenient to use contradicts the experiences reported in [HEROT 82] and the findings of [EnGr 75].

CHARACTERISTICS	QUERY FORMULATION		OUTPUT PRESENTATION	
LANGUAGES	Medium	Method	Medium	Format
FIRST GENERATION QUERY LANGUAGES	keyboard function keys	keyword commands restricted natural language menu selection line-by-line prompting use of function keys graphic or pictorial	screen printer	Lists Tables Forms Text
SECOND GENERATION QUERY LANGUAGES	keyboard function keys picking devices touch-sensitive screens voice-recognizers gesture-tracking eye-positioning tracking	keyword commands menus gesture eye-positioning zooming voice browsing by-example form-filling touch	screen voice synthesizer color graphics video-display printer	Lists Tables Forms Templates Icons Color Highlighting Images Sounds or voice Arrangement Text
QUERY BY EXAMPLE	keyboard function keys picking devices	by-example form-filling	screen printer	Templates Tables
VIDEO GRAPHIC QUERY FACILITY	keyboard function keys touch-sensitive screens picking devices	menus browsing pointing touch	screens (multiple) printer video-display color graphics	Tables Icons Color Arrangement
SPATIAL DATA MANAGEMENT SYSTEM	keyboard function keys touch-sensitive screens picking devices	keyword commands zooming browsing pointing touch	screens (multiple) printer video-display color-graphics	Lists Tables Forms Icons Color Highlighting Images Arrangement Text
PUT-THAT-THERE	picking devices voice-recognizer gesture tracking (eye-positioning tracking)	gesture voice (eye-positioning) browsing	screen voice synthesizer color graphics video display	Icons Color Images Voice Arrangement

Table 2(a): Taxonomy of Query Languages at the Senses Level

LANGUAGES	QUERY LANGUAGE DYNAMICS								
	Interaction Choice	Gamesmanship	Database Model Dependence	Error Probability	Training	User Feedback	User Interaction Control	Functionality	User Output Control
FIRST GENERATION QUERY LANGUAGES	Low	Low	Medium-High	Low-High	Medium-High	Low-Medium	Low-High	High	Low-Medium
SECOND GENERATION QUERY LANGUAGES	Low-High	Medium-High	Low-Medium	Low-High	Low-Medium	Medium-High	Medium-High	Low-High	Low-High
QUERY BY EXAMPLE	Low	Medium	Medium	Medium	Medium	Medium	Medium	High	Low
VIDEO GRAPHIC QUERY FACILITY	Low-Medium	Medium	Low	Medium-High	Low-Medium	Medium-High	High	Low	Low-Medium
SPATIAL DATA MANAGEMENT SYSTEM	High	High	Low	Medium-High	Low-Medium	Medium-High	High	Medium	Medium-High
PUT-THAT-THERE	Medium	High	Low	High	Low	Medium	High	Low	Low-Medium

Table 2(b): Taxonomy of Query Languages at the Senses Level

The only 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; a natural education process. We believe that the major contributions and the success secrets of QBE are the "by-example" principle, the two-dimensional data representation, and the stepwise learnability feature. The latter refers to the fact that a novice can perform something interesting in a very short time, yet the system provides a lot more power for the expert user. Several psychological studies focusing on QBE will be discussed in Section 5.

In Table 2 we present an evaluation and comparison of second generation QLs with first generation query languages. In addition, we present the evaluation of example second generation QL systems.

The first part of Table 2 examines languages in terms of the query formulation and the output presentation interaction parameters. For query formulation we first consider the different media that are available to the language user. For instance, the user of a second generation QL may have in addition to the regular keyboard some picking devices (joystick, mouse), a touch sensitive screen, a voice recognizer, etc. A more important distinction though between the two generation languages with respect to query formulation is the methodology,

the clues and senses which the user employs to extract information from the database. Second generation QLs make much greater use of senses and cognition. Output presentation is further subdivided into the medium on which the system informs the user of the action result, and the format of the system's reply. The latter refers to the types of objects, their appearance, and their arrangement that may facilitate comprehension of the interaction.

The second part of Table 2 is more subjective dealing with the dynamics of user interaction for the evaluation of the languages. Interaction choice refers to the flexibility the user has in choosing the style of interaction; whether the interaction medium can be switched to one which is more appropriate for the task at hand. When the interaction is interesting and the challenge of interaction has visibly important results, we say that the QL system provides high gamesmanship.

Most first generation QLs highly depend on the underlying database model. It is essential that the user knows whether the underlying database system and model is relational [CODD 71], network [CODAS 71], or hierarchical [IBM 75] for the language to operate. The underlying database model is generally transparent to the second generation language user.

The user can make clerical, syntactic or semantic errors during the interaction and the probability of something going wrong is not substantially different for the two generation QLs.

Finer distinctions of errors are discussed in the next section. On the other hand, the amount of training required for the acquisition of the ability to complete an interesting task seems to be lower with second generation QLs.

With the direct manipulation of objects in second generation query languages, the amount of user feedback from the system is very high; the action here is instantly visible. Most users find it important to have the feeling of being in control during the interaction with the system; we find no substantial difference here between the two generation QLs.

Where second generation QLs, at least in their current status, have less capabilities is on functionality. First generation QLs seem to have an advantage here since they can cover a broader and more diverse set of functions.

Different requests may have different output format requirements. The amount of user flexibility in controlling the format of the output is captured by the user output control parameter.

In summary, second generation QLs have several advantages over traditional first generation query languages: the methods that the user employs in query formulation and comprehension of the reply, the amount of training required, the system feedback, and the gamesmanship of the interaction. A disadvantage may be the limited language coverage.

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. These new skills are not found today in the traditional systems analysis and design education.

Furthermore, the fact that the query language is only a part, although a central one, of the "total" interface should not be overlooked. In a layer of several computer systems and languages the second generation QL is the outmost layer. Apart from the obvious overhead, there is the great risk of falling to a lower level system during the interaction. It is very likely that the interaction with such a lower level system will not be consistent with the second generation QL. We believe that protection mechanisms for a smooth transition from one interaction level to another is a strong requirement for the success of second generation query language systems.

4. THE METHODS LEVEL

The previous section presented a taxonomy of query languages according to the clues and the user senses that are employed for the interaction. In this section, we categorize first generation query languages according to method type and

evaluate them with respect to interaction and user performance parameters. Table 3 summarizes our findings. The table entries and highligts of our comparison are now explained.

We classify the first generation query languages in eight groups: function-key, line-by-line prompting, menu selection, graphic or pictorial, restricted natural language, linear keyword, positional keyword, and mathematical keyword. In addition, we relate the results from this taxonomy to our findings in the senses level by grouping the second generation QLs (last row of Table 3).

The use of function keys is a limited but very 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. Line-by-line prompting, also called parameterized ineraction [LeBl 79], is a very 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, the user is required to point to his choice from a menu of options offered by the system. Menus are structured hierarchically; the choice of an option may cause the availability of a new menu [ElNu 80]. These three language methodologies are usually custom-made for specific applications. They may be based on a lower level query language.

QL PARAMETERS / QL TYPES	# of Syntactic Constructs	Model Complexity	INPUT	Clerical	Syntactic	Semantic	Correction Handling Effort	User Type Level	Composition	Comprehension	Application Dependence	Database Dependence	Selectivity	Functionality	Control	Format Variation	Responsiveness	Customization	Examples
	THINKING			PROBABILITY OF ERROR				TRAINING			LANGUAGE POWER				OUTPUT PRESENTATION				
	QUERY FORMULATION			ERRORS															
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 (QBE)
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, DL/1, NOMAD
POSITIONAL KEYWORD	High	High	Medium-High	High	High	High	High	Medium-High	High	Medium-High	Low	High	High	High	Medium	Medium	Low	Low	SQUARE
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

Table 3: Taxonomy of Query Languages at the Methods Level

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].

The restricted natural language mode has attracted interest in recent years. The intention is that the user can employ his 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 at the research laboratory [LEHMA 77, 78, PLATH 76, CODD 74, 78, WALTZ 78, WILKS 77]. Some natural language systems will engage in a dialogue with the user to resolve any ambiguity in requests [CODD 78]. We should point out that the natural language communication in all such state-of-the art QL systems is still far from close to person-person communication; the justification of the prefix "restricted". This may not be significant for some semantically restricted applications and for some types of users [TURNE 82, STOHR 82]. It may be argued that natural language is a second generation QL. Even though natural language may become a dominant QL in the future, it does not meet our criteria for second generation QLs; the user still has limited use of methods, instincts and senses to interact with the database.

The majority of query languages fall into the category of linear keyword. These languages utilize statements similar to a programming language such as FORTRAN, 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 [DENNY 77], DL/1 [IBM 75], and QUEL [STONE 75].

Some keyword languages use the position of the command operators and operands to convey meaning [BOYCE 75]. Other keyword languages use the precise notation of the mathematical formalism to replace wordy English-like expressions [SCHMI 77,TODD 76]. The succinctness of mathematical symbols allow for short expression of powerful operations.

We have included in the Appendix a list of example QL language systems as well as the methodologies they employ, their procedurality type and the underlying database model and system. Representative QLs are also presented in the last column of Table 3.

The query language types, as defined above, are evaluated in terms of three basic interaction parameters: query formulation, language power, and output presentation.

Query formulation describes the overall effort of the user to work with the system. It is further divided into the thinking effort required before the user starts expressing the request, the amount of input for the query expression (e.g. number of keystrokes), the type of errors in query formulation and their handling, and finally, the training effort needed for the interaction.

Most query languages impose a strict protocol and require the user to remember syntactic constructs. This is especially characteristic of keyword-oriented languages. 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 complexity of this model is higher for keyword oriented languages than for languages which are system-driven dialogues (e.g. menu selection).

Input refers to the amount of clerical effort in expressing the request. When the interaction is via a keyboard this may be measured with the number of keystrokes. Alternatively, with pointing devices a good measure is the number of pointed objects [LoTs 81]. The clear winners in this category are function keys and menu selection.

Three main types of error may occur during query formulation: clerical (e.g. typos), syntactic (not following the correct syntax of the language), and semantic (formulation of a syntactically correct query which does not solve the task at hand). We consider here the probability that such errors will occur for different language types. Furthermore, we are interested in the amount of effort required to handle an error after it has been detected. Our findings indicate that use of fuction keys rarely results in any type of errors, and if an error occurs, the correction effort is small (press another function key). On the other hand, the challenging interface of

a second generation QL may result in semantic errors (e.g. follow the wrong path while browsing) for which the handling effort is high. This is not the case with clerical or syntactic errors in second genaration QLs; these errors are immediately detected and there is a rapid reversible action to correct them.

The amount of training necessary for the user to reach a level where he can perform useful tasks is a very important consideration for language acceptance. We have subdivided training into three categories: user type level, composition, and comprehension.

The user type level refers to the degree of expertise the user must have before he can use the language. Low level corresponds to novice user, medium level to skilled user and high level corresponds to professional user (see also Section 5 for detailed definitions). System-driven dialogue languages have an edge here; since the user is guided during the interaction, technical expertise is not necessary. We also note that the users of restricted natural language must be skilled, at least with regard to the application domain.

Composition is the task of formulating a query. A facility that is easy to learn facility is not necessarily easy to use. The analogue is with the programming language BASIC; it takes almost no time to learn, but writing a complex program in BASIC is not an easy task. Second generation query languages require very little training for using the available facilities. Our experience [TURNE 82] is that the "restrictions" part in natural

language necessitates training for query composition. The user must be taught to use only the allowable unambiguous English constructs.

Comprehension refers to the amount of training required to understand a query formulated by another user. It is very easy to understand an unambiguous natural language statement. We also conjecture that a user understands easily a linear keyword query; this is explained from the simple syntax and appropriate keyword selection in such languages. On the other hand, a user with little mathematical background will find it hard to comprehend a query in a mathematical keyword language.

The second major parameter for query language evaluation is language power - how much a user can do with the language and what factors this depends on. This is where the user of function keys and menu selection has to pay the price for simplicity in query formulation. Although such languages can be designed to be powerful, they can never reach the expressive power of a keyword-type QL. After all, there are upper bounds on the number of function keys available and on the number of alternatives that may be selected before the menu path becomes a labyrinth.

System-driven query languages depend heavily on the application domain while keyword oriented languages are more general purpose. In addition, the terminology of the application (vocabulary) must be defined before a restricted natural language is utilized.

As discussed in Section 4, the usage of most first generation QLs depends on the underlying database system and model; this is especially true with keyword oriented languages.

Selectivity is the ability of the language user to specify as precisely as possible what data he wishes to retrieve [LoTs 81]. The desirable high selectivity is usually found in keyword type languages and second genaration QLs.

Functionality refers to the number of different tasks the language can be used for. As we pointed out in section 4, this seems to be the major user performance advantage of first generation QLs (except function keys and menu selection) over second generation query languages.

Our third major criterion for QL comparison is 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. Regardless, the philosophy behind certain language groups leads to a more natural adaptation of a desirable output feature in a system. For instance, customization is easier in menu selection than with a positional keyword language and system responsiveness is highly visible in second generation QLs.

The contribution of this section is the taxonomy of QLs, and the presentation of usability and interaction features according to which query language designs can be evaluated. The reader may look at Table 3 as a cost-benefit analysis. The constant cost is the amount of training necessary before the language can be used. Variable costs are the efforts involved in formulating queries and handling errors. The user benefit derives from the language power and the output presentation features.

Not all entries in Table 3 are subjective. For instance, the fact that the functionality of keyword-oriented languages is high in comparison to the use of function keys or line-by-line prompting is very hard to argue against. On the other hand, for the few subjective entries in Table 3, such as user type level and user model of query writing complexity, we remind the reader that the methodology we followed (described in section 2) does not have strong scientific validity. The "common sense" approach may be misleading as pointed out in [MORAN 81]. Our conjectures can be tested using what Moran calls the features approach. This refers to experiments to measure the impact of language features on user performance.

5. USER TYPES AND QUERY LANGUAGES

In this section, we relate our taxonomy of query languages to the different types of users. We develop a unified classification scheme of query language users and recommend which methods would be most appropriate for the different user types. Where possible, we substantiate our position by existing human factors studies or point out areas where further studies are needed.

Many criteria for classifying users have been proposed [CODD 74, CUFF 80, LeBl 79, MORAN 81, SHNEI 80, YORMA 77, ZLOOF 78] but their relationships have hardly been studied. [SHNEI 80] uses a two-dimensional scheme classified by syntactic and semantic knowledge. We base our analysis on his approach but expand it to include other criteria that have been proposed elsewhere. An analysis of these criteria reveals that most of them can be derived from a system with the four (binary) dimensions, namely, familiarity with programming concepts, frequency of query language usage, knowledge about the application, and range of different operations required.

Familiarity with programming concepts seems a better wording 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]. We assign "high" familiarity with programming concepts to a user who is not afraid of computers and has acquired logical or algorithmic problem-solving abilities.

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

Table 4(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	Detailed
Narrow	casual user	clerical user
Broad	managerial user	application specialist user

Table 4(b): User types —

Task structure (semantic knowledge) as a function of application knowledge and range of operations.

The dimension frequency of system usage was first introduced by [LeBl 79]. We demonstrate 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 initial investment is justified. The amount of training in turn determines the typical skill level after the training period. We feel that the transient skill levels during the training phase are of interest for query language 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. In this way, the distinction between "novice" user (task: learning) and "expert" (task: routine skill) made in [MORAN 81] 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 so far determine the user's ability to technically interact with the system; his or her "syntactic knowledge" [SHNEI 80]. Table 4(a) shows the relationship between the two basic dimensions and the level of interaction skill.

The semantic dimensions are concerned with application knowledge and range of operations of the user. In the database context, application knowledge measures the precision of the conceptual model the user has about the structure and contents

of the database. This conceptual model can be anything from very detailed to a general idea. The other dimension, range of operations, describes how many different types of queries the user wants to ask in the language. Together, these two dimensions give a good picture of the task structure of the user, another semantic criterion (Table 4(b)).

We now proceed to relate the twelve user types defined by combinations of syntactic and semantic knowledge to the language methods discussed in the previous section. Developed using the technique described in the introduction and the results of human factors studies, Table 5 gives an ordered list of suitable language methods for each user type. Before discussing Table 5 in detail, however, we give a general overview of human factors research on the methods level.

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

The first group of experiments consists of comparisons of keyword vs. second generation languages [ThGo 75, GrWa 78], keyword vs. positional languages [REISN 75], and keyword vs. restricted natural languages [SHNEI 78, SmWe 77, TURNE 82]. The reader is cautioned, however, that not all of these experiments intended a general comparison of methods but rather specific

comparisons of languages. Nevertheless, they give some hints and directions for future research.

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 78]. 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, KRAUS 79, LEHMA 78, TURNE 82, WOODS 72].

As for user types, most studies in the syntactic knowledge dimension focus toward the novice user. Virtually all laboratory experiments are learning and retention tests. Therefore, as mentioned above, the results apply mainly to the infrequent user. 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.

The semantic classification of experimental users 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.

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 5: Relating language methods to user types

Reference	Type of Experiment	User Classes	Research Question	Tasks and Tests	Major Results
BrSh 78	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
DAMER 79	field	novice application specialists	productivity of TQA/REQUEST	problem-solving	65% successful
GoAs 75	lab	novices	query formulation process (IQF)	composition	influence of task complexity and ambiguity
GrWa 78	lab	novices	learnability of QBE vs. SQL	composition	formulation in QBE is faster
HARRI 77	field	application specialists	productivity of ROBOT	problem-solving	80-90% successful
KRAUS 79	field	skilled application specialists	productivity of USL	problem-solving	more than 90% successful after adaptation
LEHMA 78	field	skilled application specialists	functions of USL	problem-solving	statistics on use of various functions
LoTs 77	lab	programmers, non-programmers	comparison of 3 data models embedded in APL	composition debugging	programmers are superior; relational model best for non-programmers
REISN 75	lab	programmers, non-programmers	learnability of SQL vs. SQUARE	composition	programmers are superior; SQL is better than SQUARE for beginners
REISN 77	lab	programmers, non-programmers	feature analysis of SQL	composition	recommended layered structure for novice and skilled user
SHNEI 78	lab	novices	productivity of natural vs. SQL	query generation	natural language user generated more invalid queries
SmWe 77	lab with simulated processor	novices	productivity of natural vs. SQL subset	interactive problem-solving	formulation in SQL is faster
THOMA 76	lab	novices	use of quantifiers	various non-computerized tasks	universal quantification is difficult for novices
ThGo 75	lab	novices	learnability of QBE	composition	67% successful after short training
TURNE 82	lab, field	novice application specialists (advisors)	learnability and productivity of USL vs. SQL	composition, problem-solving	(preliminary): SQL has a higher success rate
WeSt 81	lab	programmers, non-programmers	learnability of TABLET vs. SQL (procedurality)	composition retention	programmers are superior; TABLET is better for hard queries
WOODS 72	field	novice application specialists	useability of LSNLIS (LUNAR)	problem-solving	good success rate for application-specific system

Table 6: Human Factors Experiments with Query Languages and Features

The major results of studies referred to in this paper are displayed in Table 6. For a more detailed overview of laboratory experiments, see [REISN 81] or [SHNEI 80].

We now turn to discussing the specific recommendations given in Table 5 for assigning language methods to user types. Our procedure will be to follow the columns of the table and then summarize the results by language type.

The casual user is characterized as having only a general idea about structure and content of the database but whose range of needed operations is also limited so that he may not use the full power of a query language [REISN 77]. Typical examples are the users of external databases like videotex [LoTs 82] or electronic funds transfer systems. Casual users would only by chance be familiar with programming concepts (that is the reason why the lower left field of Table 5 is nearly empty) but may vary in their frequency of system usage. 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 managerial user is probably the most demanding user type. Unwilling to "waste" time to acquire detailed knowledge of the database, he 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 happens to have 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 [TURNE 82] or the semantic restrictions of the database [BrSh 78]. For this reason, restricted natural language is only 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 clerical user (or parametric user [ZLOOF 78]) has to perform only a limited number of operations on the database, but his integration in the organization gives him detailed knowledge about the available data. The use of function keys or menus with little system guidance improves day-to-day productivity. 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 (we avoid the synonymous "professional user" because we assigned this title

already to a syntactic category). This type of user can be expected to have detailed knowledge of the database and wants to do many different 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, most of the recent research focuses on novice and skilled users who are expected to be a dominant group of computer users in the near future. For example, human factors studies 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 78];

* 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 in a keyword language [WeSt 81];

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

Studies of the use of natural language interfaces have so far been inconclusive, probably due to different user types and scope of application. We 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 79]. More novice users apparently get into trouble if their knowledge of the database is limited and if the quality of the overall interface (compare section 2) does not match that of the query language itself [TURNE 82]. With improved systems our 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 will be preferable. The inclusion of second generation languages in the lower right box is due to their rapid reversable actions that support exploratory use of the database [SHNEI 82].

We have reviewed in some detail the usability of language methods for each user type. We conclude this section by summarizing the results by language type. An interesting pattern can be observed here.

The pattern of query language development outlined in Section 2 can be observed in the usage distribution of methods. Formal query languages (keyword, mathematical, and positional) center around the lower right of Table 5, that is, around skilled or professional users with detailed database knowledge. Keyword languages have more general scope than mathematical languages which in turn may be more efficient for specialized task structures such as application programming. What these languages have in common is the idea of full specification of an operation by a command or a sequence of commands in a command language. Natural language can be thought of as an extension of this kind of (first generation) language to be used by less sophisticated users.

The other (technical) 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 accomodate more skilled or ambitious users. In ascending degree of sophistication, this group of methods includes function keys, line-by-line prompting, menus, and finally second generation languages.

From Table 5, 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.

6. CONCLUSIONS

The taxonomy developed in this paper is based on a new interpretation of the development of database query languages as being influenced by the areas of programming languages, database management, and human factors engineering. This observation 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. In addition, we developed a comprehensive classification scheme of query language users from which most existing user categorizations can be derived.

The language and user taxonomies taken together permitted us to make specific recommendations for relating language methods to user types and applications.

Much remains to be done. We have pointed out that few experiments in the field have been directed towards comparison at the general methods level, and that 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 for the differentiation of methods within this group to be clearly understandable.

Each single language type will have problems accommodating the variety of user types discussed in this paper. We envision future query languages to employ multiple interaction modes in order to have a broader coverage and usability. In addition, we believe that new languages will 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).

[BANER 77]
Banerjee,N., "CONDOR: Communication in Natural Language with a Dialogue Oriented Retrieval Systems", Computational Linguistics in Medicine, North-Holland, (1977).

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

[BENNE xx]
Bennet,B.W., "Evaluation of the DMS-1100 Query Language Processor QLP", Database Journal,7,No.3.

[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).

[CCA 77]
Model 204 User Language Reference Manual, Computer Corp. of America, Cambridge, Mass, (1977).

[CHANG 76]
Chang, C.L., "DEDUCE: A Deductive Query Language for Relational Databases", Pattern Recoginition and Artificial Intelligence, Chen (Ed.), Academic Press, (1976), 108-134

[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 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 72]
Codd, E.F., "Relational Completeness of Data Sublanguages", Data Base Systems, Courant Computer Science Symposium, Rustin (Ed.), Prentice-Hall, 1972

[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 Revently?", 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).

[DATE 80]
Date, C.J., "An Introduction to the Unified Database Language", Proc. ACM International Conf. on Very Large Data Bases, (1980), 15-32.

[DeHe 76]
Deheneffe, C., Hennebert, H., "NUL: A Navigational User's Language for a Network Structured Data Base", Proc. ACM SIGMOD, (1976), 135-142.

[DENNY 77]
Denny, G.H., "An Introduction to SQL, A Structured Query Language", Tech. Rep. RA93, IBM Reseach Lab, San Jose, (1977).

[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", Int.J.Man-Machine Studies 12 (1980), 259-282

[GMD 75]
GMD, "Datenbankensysteme - Erfahrungsberichte - Heft 2 ADABAS", St. Augustin, (1975).

[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).

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

[HASEM 77]
Haseman, W.D., "GPLAN: An Operational DSS", ACM SIBDP Database, Fall, (1977).

[HENDR 78]
Hendrix et al, "Developing a Natural Language Interface to Complex DAta", ACM Transactions on Database Systems, 3, (1978).

[HEROT 81]
Herot, C.F., "Spatial Managements of Data", ACM Transactions on Database Systems, 5, (1989), 493-513.

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

[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).

[IBM xx]
IBM, "Interactive Query Facility (IQF)", General Information Manual, Publ. No. GB21-9903, IBM Corp., Data Processing Division

[IDBS 79]
International Data Base Systems, Inc., "HARVEST Query Language/Report Writer", (1979).

[IDMS 77]
IDMS, "Concepts and Facilities", Cullinane Corporation, (1977).

[KAMEN 78]
Kameny et al, "EUFID: The End User Friendly Interface to Data Manatgement Systems", Proceedings 4th Int., Conference on Very Large Data Bases, IEEE, (1978).

[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)

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

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

[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).

[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 82]
Lochovsky F.H., Tsichritzis, D.C., "Querying External Databases", NYU Symposium on User Interfaces, New York, (1982).

[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", proceedings of the ACM SIGMOD/SIGSMALL Workshop on Databases for Small Systems, Orlando, Florida, October (1981).

[McDON 82a]
McDonald, N.H., "Video Graphic Query Facility", to appear in proceedings of the Second International Conference on Databases - Improving Usability and Responsiveness, Jerusalem, Israel, June, (1982).

[McDON 82b]
McDonald, N.H., "Multi-Media Approach to User Interface", Proceedings of the NYU Symposium on User Interfaces, New York, May, (1982).

[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

[MRI 73]
System 2000 Reference Manual, MRI System Corp, Austin, Texas, (1977).

[MYLOP 76]
Mylopoulos, J., et al, "TORUS: A Step Towards Bridging the Gap Between Data Bases and the Casual User", Information Systems, 2, (1976), 49-64.

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

[NICKE 81]
R.S.Nickerson, "Why Interactive Computer Systems are Sometimes not Used by People who Might Benefit from Them", Int.J.Man-Machine Studies 15 (1981), 469-483

[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

[SCHAU 76]
Schauer,U., "Ein System zur interaktiven Bearbeitung unfagreicher Messdaten", Hesselmeir and Spruth, (eds.), Springer, Berlin, (1976).

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

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

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

[SHNEI 82]
B.Shneiderman, "The Future of Interactive Systems and the Emergence of Direct Manipulation", Proceedings of the NYU Symposium on User Interfaces", New York, May, (1982).

[SmWe 77]
Small,D.W., and Weldon,L.J., "The Efficiency of Retrieving Information From Computers Using Natural and Structured Query Languages", Rep. SAI-78-655-WA, Science Applications, September, (1977).

[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., 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).

[THOMA 77]
Thomas, J.C., "Psychological Issues in Data Base Management", in Proceedings of Third Intern. Conf. on Very Large Data Bases, Tokyo, (1977).

[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 82]
Turner,J., Jarke,M., Stohr,T, Vassiliou,Y., and White,N., "Using Restricted Natural Language for Data Retrieval - A Field Evaluation", Proceedings on NYU Symposium on User Interfaces, New York, May, (1982).

[VONGO 74]
Von Gohren, G.L., "User Experience with Integrated Data Store (IDS)", Data Base Management Systems, Jardine (Ed.), North-Holland, (1974).

[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).

[WINOG 71]
Winograd, T, "Procedures as a Representation of Data in a Computer Program for Understanding Natural Language", TR-84, MIT, (1971).

[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).

APPENDIX

LANGUAGE ENVIRONMENT	Functional Range of Language			Additional Facilities
	Retrieval Only	Retrieval/ Update	Includes Database Administration	
Self-contained	Intellect USL	MDQS	SQL, QBE, QUEL	built-in functions grouping/sorting report generation
Embedded or integrated in a programming language		TOTAL-IQ C/QUEL APL/EDBS COBOL/DML PASCAL-R	DL/I COBOL/DDL	usually provided by host language

Table 1: Examples for some types of database languages

EXPLANATION

A self-contained language provides all the necessary capabilities for performing database interactions. On the other hand, the syntactic forms of a query language can be embedded in a programming language like, PC/I, COBOL, or PASCAL. There are four techniques for embedding (LaPi 80): subroutine calls (DL/I, TOTAL-IQ), simple extension (COBOL/DML), non-procedural operators (C/QUEL, APL/EDBS), and integrated (PASCAL-R). Some query languages are retrieval-only; this is typical of restricted natural language systems. More commonly, update capabilities are provided. Data administration includes the ability to create and modify database descriptions, define integrity constraints and impose access control. More language examples are given in Table 7.

QUERY LANGUAGES

LANGUAGE	DB MODEL	DB SYSTEM	METHOD	TYPE	REFERENCE
ADACOM	hybrid	ADABAS	keyword	navigation	[GMD 75]
ADASCRIPT	hybrid	ADABAS	keyword	navigation	[GMD 75]
ALPHA	relational	-	keyword	calculus	[CODD 71,72]
ASI/INQUIRY	hierarchical	IMS	keyword	non-proc.	[IBM 75]
CONDOR	natural	-	natural	linguistic	[BANER 77]
CUPID	relational	INGRES	pictorial	-	[McDON 75]
DAPLEX	hybrid	INGRES	keyword	non-proc.	[SHIPM 81]
DATA DISPLAY	network	IDMS	keyword	navigation	[IDMS 77]
DEDUCE	relational	-	keyword	calculus	[CHANG 76]
DRUID	network	IDMS	keyword	navigation	[IDMS 77]
EQBE	relational	IDAMS	fill-in	non-proc.	[SCHAU 76]
EUFID	network	special	natural	AI	[KAMEN 78]
FORAL-LP	hybrid	DIAM/II	pictorial	-	[SENKO 78]
GENIE	network	IDMS	keyword	navigation	[IDMS 77]
GIS	hierarchical	IMS	keyword	navigation	[IBM 75]
GPLAN	network	special	keyword	navigation	[HASEM 77]
HARVEST	network	SEED	keyword	navigation	[IDBS 79]
IDSQ	hybrid	IDS	keyword	navigation	[VONGO 74]
ILL	relational	-	keyword	non-proc.	[LaPi 75]
IMMEDIATE	hierarchical	SYS-2000	keyword	non-proc.	[MRI 73]
INTELLECT	natural	ADABAS	natural	linguistic	[AIC 82]
ISBL	relational	PRTV	mathem.	algebra	[TODD 76]
IQF	hierarchical	IMS	keyword	non-proc.	[IBM xx]
LADDER	natural	IDA/FAM	natural	AI	[HENDR 78]
LINUS	relational	MRDS	keyword	calculus	
LSL	network	LSL	keyword	navigation	[TSICH 76]
LUNAR	natural	special	natural	AI	[WOODS 72,77]
MODEL 204	network	MODEL 204	keyword	navigation	[CCA 77]
NATURAL	hybrid	ADABAS	keyword	navigation	[GMD 75]
NOMAD		NOMAD	keyword	non-proc.	
NUL	network	-	keyword	navigation	[DeHe 76]
PLANES/JETS	natural	ALPHA/	natural	AI	[WALTZ 78]
PASCAL-R	relational	PASCAL-R	mathem.	calculus	[SCHMI 77]
QBE	relational	QBE	fill-in	calculus	[ZLOOF 77]
QBPE	relational	-	pictorial	-	[ChFu 79]
QLP	network	DMS/1100	keyword	non-proc.	[BENNE xx]
QUEL	relational	INGRES	keyword	calculus	[STONE 75]
QUERY	hybrid	IMAGE	keyword	non-proc.	
RAMIS			keyword	non-proc.	
RENDEZVOUS	natural	special	natural	AI	[CODD 74,78]
REQUEST	network	TOTAL	keyword	navigation	[CINCO 78]
SHRDLU	natural	special	keyword	AI	[WINOG 71]
SQL	relational	SYSTEM R	keyword	mapping	[ASTRA 76]
SQUARE	relational	-	posit.	mapping	[BOYCE 75]
TQA/REQUEST	natural	special	natural	linguistic	[PLATH 76]
TORUS	natural	MINIZ	natural	AI	[MYLOP 76]
TOTAL-IQ	network	TOTAL	keyword	non-proc.	[CINCO 78]
UDL	hybrid	-	keyword	non-proc.	[DATE 80]
USL	natural	SYSTEM R	natural	linguistic	[LEHMA 77,78]