This page is intentionally left blank.

Chapter 10

Information Retrieval

The value of knowledge lies not in its accumulation, but in its utilization.

E. Green

Information retrieval is carried out by application of queries to databases where the result is expected to increase the users' knowledge. Database applications which are not considered information retrieval are systems which provide regular operational service, for instance, systems which schedule activities, manage inventories, or prepare bills or payrolls. Applications of databases in areas such as airline reservations and skill registries are close to information-retrieval applications. These and many other operational systems overlap with information-retrieval systems to an extent that much material in this chapter will also be relevant in the context of operational systems. In information retrieval the ability of systems to cope with a wide variety of ad hoc queries is stressed.

The intent of this chapter is not to cover the field of information retrieval in a comprehensive manner; more has already been written about information retrieval than about the subject of database itself, and several excellent books on this subject exist. It is, however, necessary to relate the concepts found in information retrieval to the objectives found in database design. When issues in the area of information retrieval are discussed, there is often an implied understanding regarding the data-organization concepts which are valid for the information-retrieval problem. While the data organization chosen for an application may be quite appropriate, it can be

puzzling to the uninitiated why one data-organization method was favored over the others. An explicit evaluation of the approach selected becomes important when an information-retrieval system is being considered in an application which is not a mirror image of the task foreseen during the original implementation of the information-retrieval programs. In order to develop a basis for comparison of information-retrieval objectives, we will categorize information retrieval into three areas:

1 Fact finding

2 Statistical inference

3 Deductive inference

In Sec. 10-2 these approaches will be discussed in detail. It should be understood, however, that a single system may have to support all three areas to some extent. Section 10-1 will abstract the data requirements for information retrieval in terms of the database system choices presented earlier. The formulation of queries to actually retrieve information is the subject of Sec. 10-3.

When information retrieval is discussed, the updating of the files is not considered to present a major problem. This greatly simplifies the comparison of the approaches. In practice, however, the updating problem is of concern for any information-retrieval system. If the information content is entirely static, dissemination of the material in printed form or on microfilm could very well provide a better alternative than the use of computers.

10-1 DATA CONSTELLATIONS

In order to obtain information from a system, a constellation of related data elements has to be extracted from a database and presented to the user in a meaningful form. The data may consist of a list of homogeneous values, relating to different entities; they may consist of a list of heterogeneous values describing a single entity, or a matrix of both.

One list could present the fuel consumption of various automobiles and another one the weight, horsepower, load capacity, etc., of one specific car. A single attribute may also record changing values over time, say, the temperature of a patient.

Whenever the data become voluminous, data-reduction algorithms become desirable.

The average fuel consumption, the trend of the temperature, and such provide information abstracted from a set of basic data in a compact form. These abstractions have always to be viewed critically. With an average value the standard deviation should be presented, with a trend the linearity.

A database system should permit rapid access to abstractions and summarizations, often supporting multiple views, while still permitting access to the underlying source data, so that questions can be resolved by *zooming in* to less abstract levels.

We will review how appropriate *constellations* of data can be obtained from the database structures presented in the preceding chapters. Three types of database structure suffice to distinguish databases from the point of view of information retrieval.

- Relational data structures
- Hierarchical data structures
- Network data structures

Each of the model structures may be implemented in various ways. In fact, the same software system might be configured to provide any of the three types.

In a *relational structure* the data are stored in tabular form. Operations combine and select from tuples, using fetch arguments selected from the query and from intermediate tuples. All binding occurs at query-processing time. The result is presented as a relation in the user's workspace.

In a *hierarchical structure* the search arguments define currency indicators. Fetch arguments define entry points into the database. Get-next operations are confined to the hierarchical structure. The result presented is a broom or a record composed of one segment of each segment type. Frequently network database structures are used to present hierarchically structured results.

In a *network structure* there are nearly arbitrary connections which bind data records to successor and predecessor records. The retrieval process enters the structure using a fetch argument and then navigates through the structure using get-next, get-owner, or get-member operations, collecting data values on the way. These data values are delivered incrementally to the query-processing process, and are used to control the progress of the query. The result is not made available as a unit, since its structure is not predictable. Figure 10-1 sketches the alternatives.

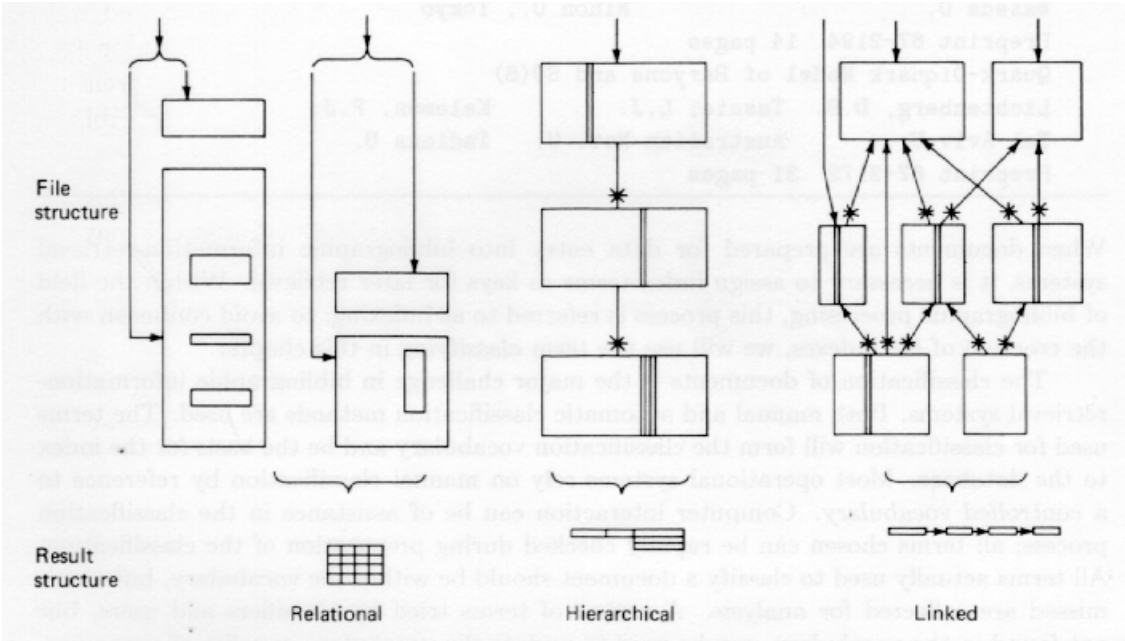


Figure 10-1 Database structure and data retrieval.

The manner in which the database systems present the results will obviously influence how the user perceives the information-retrieval process. It is still possible to interpose programs between the user and the primary database-manipulation facilities which put a different facade on the database structure.

‡‡‡‡‡ **Bibliographic Data** A short note on a class of information-retrieval systems which do not involve data-processing is in order. These are the bibliographic systems which maintain data about collections of publications. They typically have records composed of authors, keywords, abstract, and a reference. Figure 4-21 presented the internal structure of one such system. The data stored is preprocessed and a single record will have information value, if correctly retrieved, without further processing. The records may be complex: each element can contain multiple entries and require multiple indexes or linkages for access. The content may be full text or just abstracted material and references to more detailed data. The search arguments are also often complex.

Bibliographic systems are frequently and correctly called *information-storage systems*, since what they contain is already potential information. The contents is selected, prepared, and classified with the expectation of eventual retrieval and use. The entities stored are complete and independent.

Example 10-1 Use of a Bibliographic Information-retrieval System

A query to the SPIRES I system, (Parker[67]) may be

```
FIND TITLE "Model" AND TITLE "Quark" NOT AUTHOR "Gupta"
        AND DATE BETWEEN "Jul.67 & Nov.67"
```

The response to this query was

count = 2

Exchange Current and Decays of Decuplet Baryons in the Quark Model
Kobayashi, Tsunehior Konno, Kimiaki
Waseda U. Nihon U., Tokyo
Preprint 67-2194 14 pages

Quark-Diquark Model of Baryons and SU(6)
Lichtenberg, D.B. Tassie, L.J. Kelemen, P.J.
Tel-Aviv U. Australian Nat. U. Indiana U.
Preprint 67-2172 31 pages

When documents are prepared for data entry into bibliographic information-retrieval systems, it is necessary to assign index terms as keys for later retrieval. Within the field of bibliographic processing, this process is referred to as indexing; to avoid confusion with the creation of file indexes, we will use the term *classifying* in this chapter.

The classification of documents is the major challenge in bibliographic information-retrieval systems. Both manual and automatic classification methods are used. The terms used for classification will form the classification vocabulary and be the basis for the index to the database. Most operational systems rely on manual classification by reference to a *controlled vocabulary*. Computer interaction can be of assistance in the classification process; all terms chosen can be rapidly checked during preparation of the classification. All terms actually used to classify a document should be within the vocabulary, but terms missed are collected for analysis. A review of terms tried by classifiers and users, but not found in the vocabulary, can be used to update the vocabulary or a list of synonyms. The frequency of use of terms, singly or in combination, can provide further input to vocabulary control. ‡‡‡‡‡

10-2 CATEGORIES OF INFORMATION RETRIEVAL

In this section we will present the alternatives in information-retrieval requirements which the users may bring to a database system. We use the three categories: *fact retrieval*, *statistical inference*, and *deductive inference*.

10-2-1 Fact Retrieval

Fact retrieval is the traditional and popular view of an information-retrieval operation. The query is entered into the computer system, and the computer system looks through its data collection and selects an answer. In essence, the answer is stored somewhere in the data bank, and the problem is to find the answer. The task of the system is to find a data constellation identified by a set of keys. Fact retrieval is the primary use of bibliographic data banks or data banks containing material in textual form.

Many business inquiry systems also fall into the fact-retrieval category. A credit verification system will respond `yes` or `no`, or maybe turn on a green or red light in response to the query about the customer's creditworthiness. This answer, the actual information, is prestored in the database. It is not expected that at this point significant computational efforts will be carried out to determine if the customer is indeed capable of settling debts.

Demands on the Database A fact retrieval-system is characterized by extensive use of the fetch operation and by the use of indexed or direct access. One or few relations will participate in a query. Since the answers are already bound to a particular query type, it is also possible to bind the database structure to match the queries so that rapid responses are obtained.

10-2-2 Statistical Inference

When the size of a response to a query is so large that the user cannot comprehend it, techniques of data reduction must be employed. The techniques required may range from simple cross-tabulations to extensive statistical processing in order to provide meaningful data. The process of obtaining these results is more complex than the simple request-response sequence. Many intermediate data elements are obtained from the database to be incorporated into the result.

Frequently requests are made iteratively until it is possible to obtain a satisfactory formulation for the query. When cross tabulations are generated, many of the possible combinations may not produce results of interest. When statistical approaches are used for the selection of relevant factors, finding an appropriate analysis technique may require several trials. It is often desirable to produce results in an intermediary form in order to help the user formulate the data relationships and analysis concepts. Examples of outputs to aid in concept formation are graphical outputs of trendlines, histograms of different subsets of the data variables of interest, and presentation of the data in scattergrams to get impressions of possible correlations between variables.

Example 10-2 Query to Present a Data Relationship

A query to help formulate the question,
"`Can aspirin treatment in patients with arthritis affect uric acid ?`"
can be entered into a system which supports statistical data analysis (TOD)
as follows:

```
PROGRAM?  scatterplot
ELEMENT FOR HORIZONTAL AXIS = Ur_Acid
ELEMENT FOR VERTICAL AXIS   = ASA
SUBSET?  5   /* A subfile from a previously defined catalog is chosen */
Rheumatoid Arthritis  249 Patients
```

The response is shown in Fig. 10-2. This initial query shows some correlation and helps a user conceptualize a relationship. The researcher may now look at the same data for all records in the file, or for more specific patient groups.

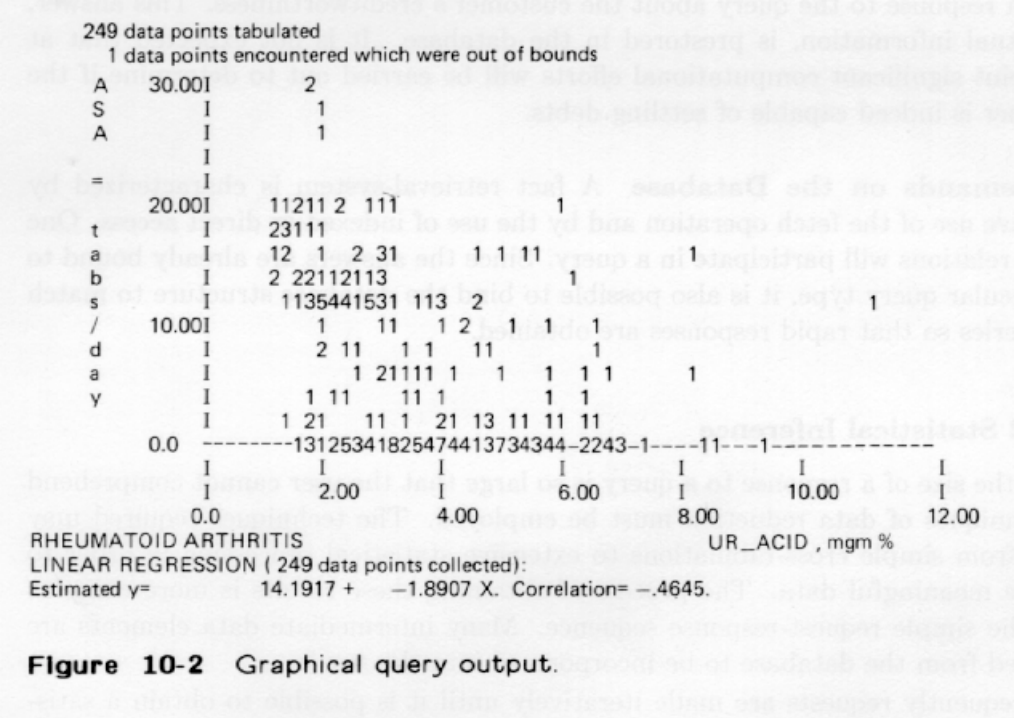


Figure 10-2 Graphical query output.

Retrieval of a few attributes of a large subset of tuples is not all restricted to the scientific environment. In management-oriented database systems, the selection of data describing a part of the enterprise and comparing the performance of these parts with averages based on the entire enterprise, or on industrial standards, is the preferred mode of operation. Graphical output can again aid in concept formation, and simple statistics can provide baseline data so that unusual events or trends can be recognized. This type of processing may produce exception reports that trigger action to determine and correct, if necessary, unexpected patterns. In this manner data can become information.

Demands on the Database Statistical inference implies the retrieval of large quantities of well-structured data. Use of the get-next operation becomes predominant.

Since processing programs can only understand files in tabular form, the output presentation of the database system must make the data constellations available in an array or as linear structures. Subset selection and retrieval has to be effective in order to process subsets proportionally faster than the entire data collection. The use of transposed files can be very beneficial.

It is difficult to present with common technology more than three-dimensional situations. This increases the processing requirements on the system: data constellations have to be presented in various combinations in order to develop an understanding of relationships among the data.

To avoid reprocessing large volumes of data, it may be desirable to collect intermediate results in workspaces. When a ring structure is used, the header records provide an opportunity to collect totals and counts for the detail values which are kept in the members of the ring. The availability of actual summary data can significantly reduce reprocessing effort.

In many of these applications, it is desirable that the database is *not* updated during analysis in order to avoid inconsistencies. A subset of the database can be copied into a temporary workspace to escape interference. Data structures in a user workspace can be bound increasingly tight as the analysis progresses.

10-2-3 Deductive Inference

In the previous two approaches to information retrieval, the results were directly related to the query; in the case of fact retrieval and in the case of statistical inference multiple entities are selected based on search arguments. A question of the type: `Why does aspirin reduce uric acid concentrations?` cannot be answered within such databases. There is no single tuple in the system which has the answer, although a relationship may exist between the two search arguments, via many intermediate facts and relationships stored in the database. The construction of systems which contain statements describing basic logical relationships and which will process queries about complex relationships is one of the most interesting applications involving database technology.

The problem can be seen as the building of a bridge between search arguments and the goal. When the database containing possible relationships becomes moderately large, the number of possible combinations becomes immense. Techniques of artificial intelligence have to be applied to reduce the search possibilities to a manageable number.

Data Structures to Express Relationships A database used for deductive inference requires a finer semantic categorization of relationships than was used when database models were constructed. An example of such a refinement is the categorization of the relationship represented by an ownership connection. The particular meaning of the relationship of a person entity to children, jobs, skills, or diseases can be made specific. The same relationships in the opposite direction would be named: parent, employee, expert, or patient. A deductive query-processing program has to be aware of the meaning of the relationship in order to follow the appropriate path.

If the database is unbound, the search algorithm will collect tuples with keys matching the query and then select further tuples based on values found in those tuples. When the database is bound, the query will find successors by following pointers.

In order to organize relationships for deductive inference, the relations are often decomposed into binary form. Now the implied relationships kept otherwise within a multiattribute tuple can be treated identically to relationships between tuples. The binary relations form triads of relationship name (or pathname), key, and goal. An example of such a structure was shown as LEAP in Sec. 4-7-5. The query will also be composed into binary relations.

An example of a network database for deductive inference is shown in Fig. 10-3, using a partial model of glaucoma. The records represent elements of the disease, and the relationships given are cause and effect, and similarity. The records are generated from a disease description provided by physicians. Only a part of the model and only one of the weights which can be applied to the relationships is shown in Fig. 10-3

Searches begin at the entry points corresponding to known data about a patient. More data can be requested; the analysis program has information about the cost of the tests which have to be administered to obtain data and can optimize the diagnostic effectiveness. The records are written by a SNOBOL program; the information retrieval is performed by a FORTRAN program.

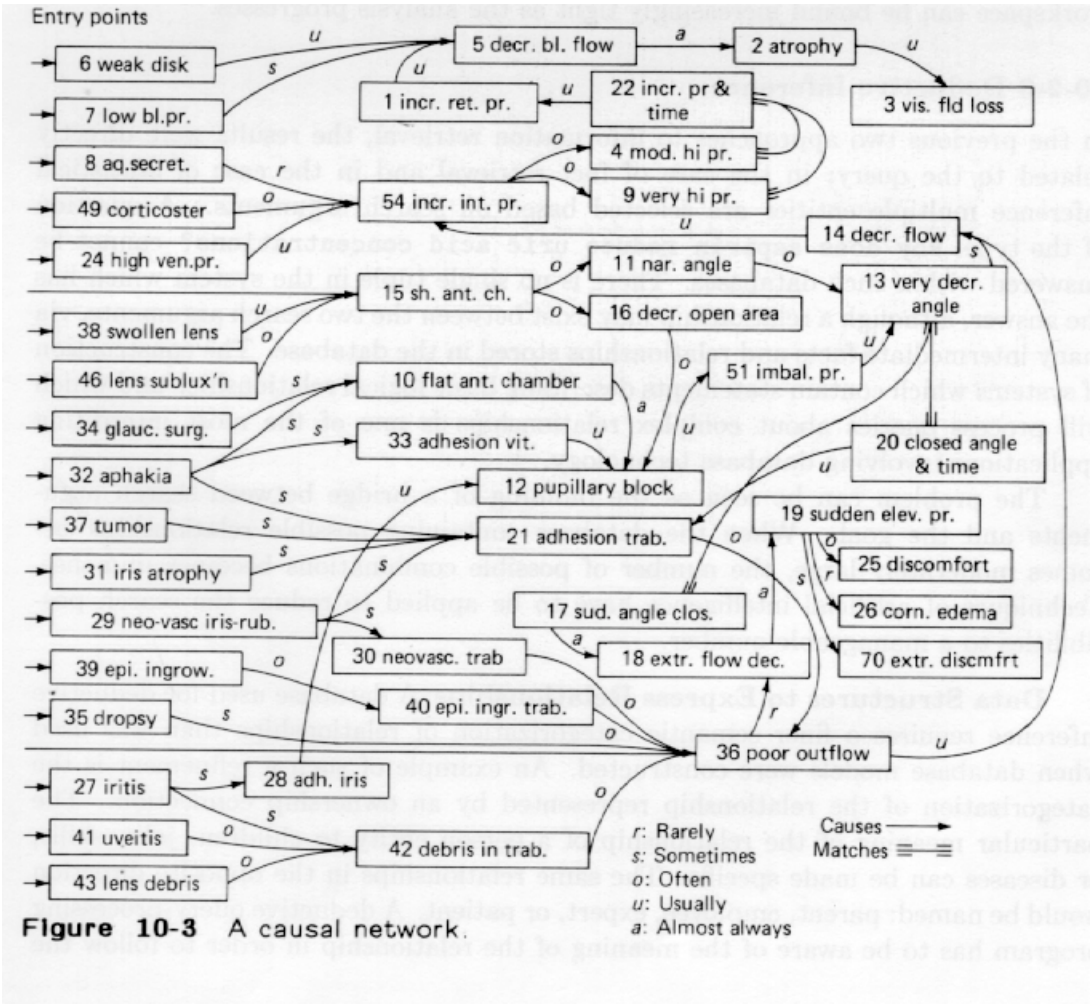


Figure 10-3 A causal network.

Demands on the Database The query process, possibly aided by information from a schema, has to evaluate the probability of success in following a particular path. The database may be constructed using *rules*, representing functions, or *frames*, representing entities, or a combination of both. It may also be profitable to provide a weight or strength measure between the frames so that concepts of being `a little bit happy`, being `very sick`, etc., can be expressed.

There is typically a preprocessing pass where the symbolically specified relationships are converted into a bound structure with chains or links. During execution the branches of the network can be rapidly traversed.

Many experimental systems have used relatively small databases. For these support of virtual memory systems has been adequate. Efficient tracing of linkages is possible if sufficient pages can be kept available so that the workspace can be maintained without heavy paging requirements. It is not yet clear how database systems can support the requirements posed by deductive inference systems using more than several thousand tuples.

10-2-4 Summary

The categories of database services available can now be evaluated relative to their effectiveness in information retrieval. When desirable features occur in different database approaches, combinations of database methods can be considered to yield a compromise solution.

The relational approach, by deferring binding to the time of query execution, is the most flexible. However, the amount of processing required may be excessive for on-line operation with large databases. When statistical inference over many tuples is part of the query process, the retrieval costs may be high in any case. For deductive inference semantic rules are required to reduce the number of relationships to be explored. These rules are in effect a form of early binding. Without such rules an unbound database will be very costly to explore.

Hierarchical structures lend themselves to rapid fact finding. For statistical processing data elements will be taken only from one level at a time, so that only structures with good access to siblings support this type of information retrieval well. Since a hierarchy is well defined and bound, deductive techniques are not required or supported by this approach.

Networks provide fact-finding capability using complex but bound structures. Statistical processing will be limited to hierarchical single level data subsets. Deductive goal seeking can be carried out by navigation in a network structure. The navigator may be a combination of an intelligent program which interacts with a knowledgeable user.

10-3 QUERY FORMULATION

Query formulation is the process performed by the user in order to communicate through the information-retrieval system with the database. The design of a query-formulation language will be affected by these three components:

- The users' needs and background
- The information-retrieval system
- The database

Users may range from casual users wishing to retrieve simple facts (weather predictions, automobile traffic, etc.) to specialists who spend many hours with the system and understand its capabilities and contents. For the former group convenience and clarity are important; for the latter group, efficiency and availability of tools to manipulate retrieved data are of consequence. There may also be intermediaries between the end users and the system: assistants who formulate queries to obtain data for decision makers, programmers who write programs to carry out more complex analyses of database contents than have been provided within the information retrieval system, people who collect data and maintain the database contents, and, finally, a database administrator, the only one who can *make the trees* on which the data are placed. In order to support this variety of users, a variety of approaches to query formulation can be considered.

The availability of a schema is an important aid in the process of query formulation, as sketched in Fig. 10-4. The definition of the attributes, availability of indexes or search paths, and the partitioning effectiveness of query arguments is best resolved before the database itself is accessed. In a hierarchical structure, aggregate values in header records can also provide data to aid in the optimization of processing during query execution.

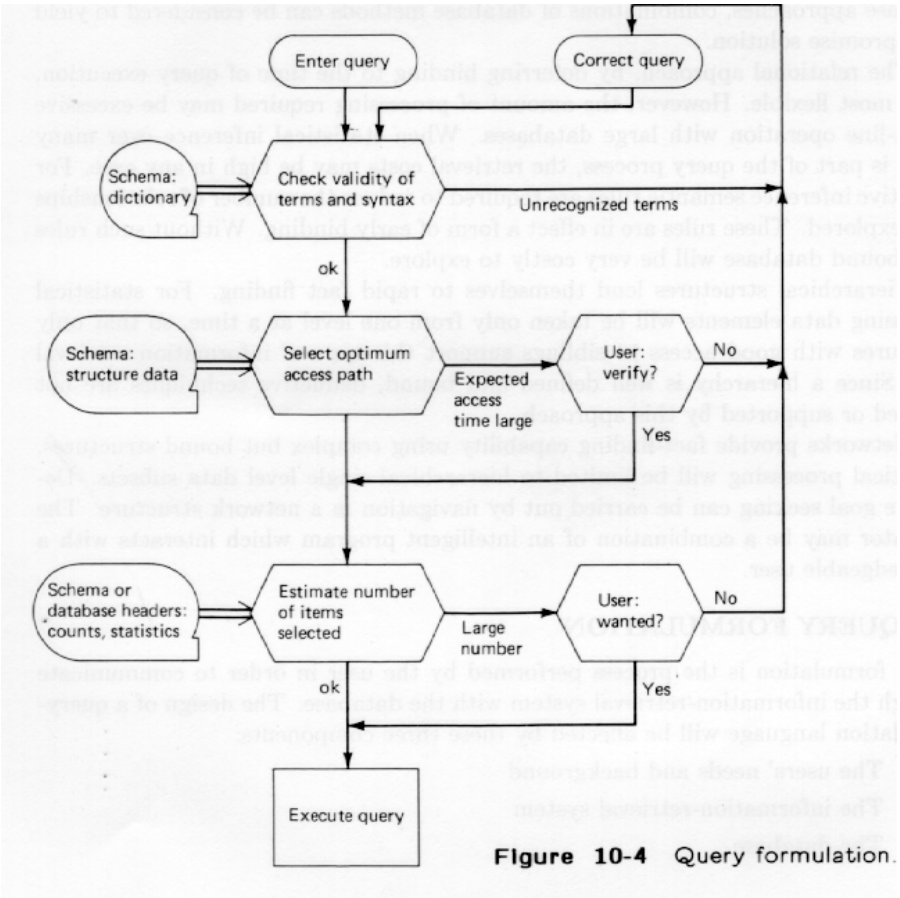


Figure 10-4 Query formulation.

10-3-1 Interrogative Query Formulation

Interrogative systems presuppose the least knowledge. The ability to pick up a telephone may be the only requirement posed. Successive questions posed by the computer are answered in simple yes-no or multiple-choice mode, or by the entry of commonly known identifying terms or names of objects. A `HELP` function is provided in case the user fails to understand the terms presented by the system. This approach can be relatively slow but is satisfactory if not carried out for extensive periods.

Examples of such devices are credit inquiry system, using special telephones, and serve-yourself banking terminals.

When a higher rate of data flow is required, display terminals can interrogate a user effectively by permitting several choices in one interaction. The user may indicate the chosen item by means of a lightpen or a touchscreen. One selection can lead to presentation of a more specific set of choices. The hierarchical nature of this query process is supported well by hierarchical file systems.

Given a patient file as illustrated in Fig. 4-25, the choices would be presented in the sequence: `Select drug, laboratory, other data?  Patient name?  Date?  Problem?  Drug type?  Frequency?  ...` . Figure 10-5 shows some sample screens.

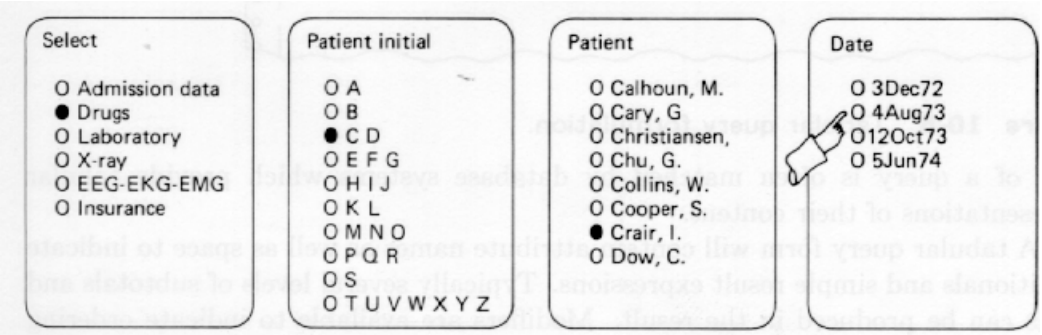
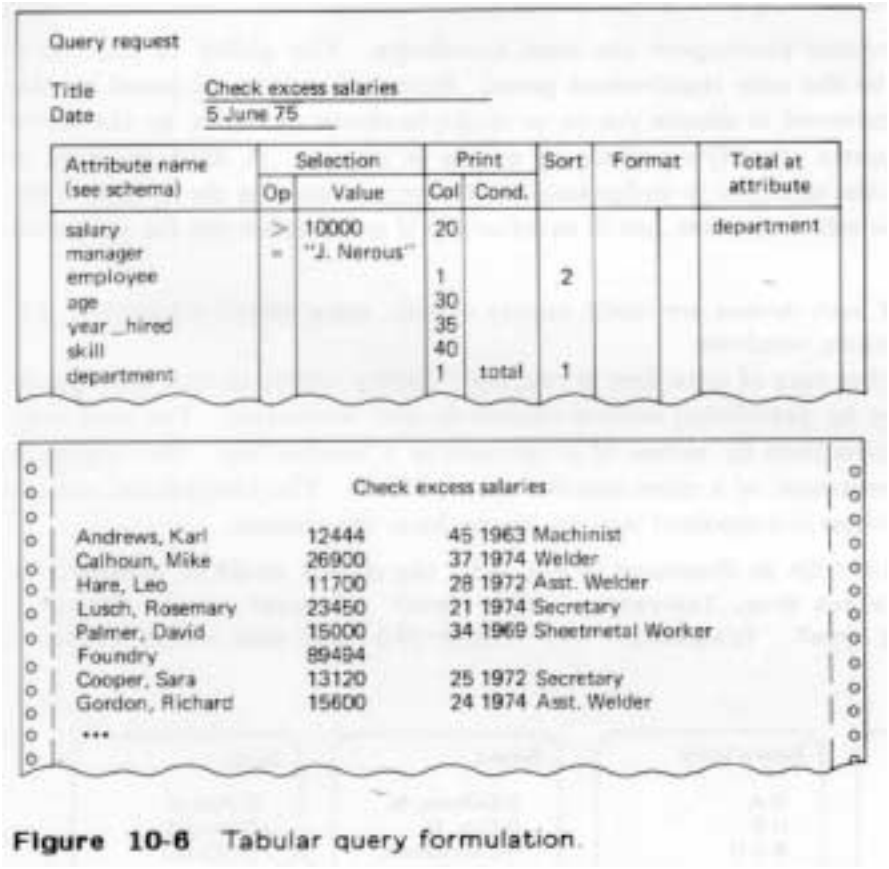


Figure 10-5 Screen selection.

The fanout on a display screen can provide a few dozen choices, so that when following a hierarchy after a few screen selections a great deal of data can be accessed. A high rate of communication capability is required to support this type of query formulation, since screen selection can be accomplished in a fraction of a second.

10-3-2 Tabular Queries

When access to the system is limited, preparation of the query using a printed questionnaire is another alternative. Directions are provided specifying the choice of parameters to be entered to describe the query. If the user has adequate knowledge about the database, a query can be formulated in a short time and then be presented to the system. Many report generators work very effectively in this mode. The



Query request

Title: Check excess salaries
Date: 5 June 75

Attribute name (see schema)	Selection Op	Selection Value	Print Col	Print Cond.	Sort	Format	Total at attribute
salary	>	10000	20				department
manager	=	"J. Nerous"					
employee			1		2		
age			30				
year_hired			35				
skill			40				
department			1	total	1		

Check excess salaries

Andrews, Karl	12444	45	1963	Machinist
Calhoun, Mike	26900	37	1974	Welder
Hare, Leo	11700	28	1972	Asst. Welder
Lusch, Rosemary	23450	21	1974	Secretary
Palmer, David	15000	34	1969	Sheetmetal Worker
Foundry	89494			
Cooper, Sara	13120	25	1972	Secretary
Gordon, Richard	15600	24	1974	Asst. Welder
***				

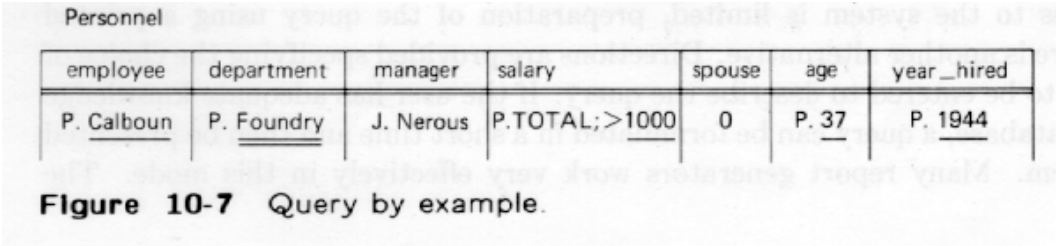
Figure 10-6 Tabular query formulation.

form of a query is often matched by database systems which provide tabular representations of their content.

A tabular query form will contain attribute names as well as space to indicate conditionals and simple result expressions. Typically several levels of subtotals and totals can be produced in the result. Modifiers are available to indicate ordering and grouping. Generalized computation tends to be awkward; an example is shown as Fig. 10-6.

When more complex computations or selection criteria are needed, the rigid tabular structure becomes inadequate. To overcome such problems statements and expressions as seen in computer languages are introduced. The formulation of queries then requires increased expertise.

To avoid the need for knowledge of the schema contents, a technique of specifying queries by example has been developed [Zloof[77]]. The same query as above formulated by example is shown in Fig. 10-7. Associated with the example entries are notations indicating selection and printing.



Personnel

employee	department	manager	salary	spouse	age	year_hired
P. Calboun	P. Foundry	J. Nerous	P.TOTAL;>1000	0	P. 37	P. 1944

Figure 10-7 Query by example.

Interaction is used to construct queries by example. First the database or file name is requested. This permits the system to list the schema elements on the display terminal. Those schema elements which are not desired as part of the query or the responses are ignored.

Here the `spouse` is ignored, and this heading can be deleted. Empty columns of the table can also be requested and named. Into the columns values for the query constants can be placed. Here `J.Nerous` and `>10000` provide constants. Underlining indicates that the values are samples, and the `P` prefix indicates that the column is to be printed. An extra underline indicates grouping for the generation of `TOTAL`s.

Several relations can be included in a query. A join is generated when the same underlined sample term appears in both relations. If the supervised employees are also to be listed in the evaluation, `Calhoun` would be placed in the `supervisor` column of `supervision`; such a use is equivalent to defining a tuple variable. The system is oriented toward a relational database but has been explored for other environments.

This query approach has been shown to have surprising power. Database schemas can also be specified by use of examples. Specifications by example are an effective means to describe output formats to computer systems. When conditions become complex, the coding rules required may diminish the initial simplicity.

10-3-3 Interactive Query Statements

For general query processing interactive methods provide the greatest flexibility. The user formulates a query using a query language, and the query statement is parsed and executed. The response to one query often leads the user to formulate further queries.

Many languages to state queries have been developed. They range from very formal, calculus-oriented languages to relatively simple languages which provide statements similar to those found in procedural languages. These languages are always influenced by the underlying database structure, by the degree of on-line interaction supported, and by the communication devices used. While techniques from procedural languages are used to parse and execute query statements, no common, system-independent query language has yet been accepted.

A similar query is specified in a display-oriented language (ASI-INQUIRY) as in Example 10-3.

Example 10-3 Query to Produce a Table on a Display Terminal

```
FIND ALL PUT employee, salary TOTAL, age, year_hired, skill
        WHEN salary ¿ 10,000
         AND manager = "J.Nerous"
         AND department = foundry
```

This statement will fill the display screen for inspection by the user. Entering `CONTINUE` will present another screen full of data, until the query result is completed. Entering `SHOWTOTAL` will present the total for the attributes indicated. A new query can then be entered for the next department.

This, and similar, languages can interface with hierarchical structures. The problems in formulating queries appropriately when data and selection elements appear on different levels have been discussed in Sec. 9-4.

In order to perform complex operations without repeated extensive accesses to the database, some query languages allow the creation of subset files, which can then be sorted and used as input to statistical processes.

In GIS the same query as Example 10-3, without the totals, can be written as in Example 10-4, using `Work` as an intermediate file.

Example 10-4 Query Using a Work File

```
SAVEX  excess_salary;
QUERY  Personnel;      /* file */
LOCATE departments;     higher hierarchical level */
WHEN   salary ¿ 10000;
AND    manager = "J.Nerous";
HOLD   Work←employee, salary, age, year_hired, skill, department;
SORT   Work←department,employee;
LIST   employee, salary, age, year_hired, skill;
```

The query is filed under the name `excess_salary` and can be retrieved later.

Query Language Operators Most query languages provide the capability to combine search arguments with boolean operators: `AND` and `OR`. A request which includes a `NOT` operator has a very low partitioning power for files, and hence its use is often limited. Comparative operators other than `=`, namely, {`<`, $\leq$, $\neq$, $\geq$, `>`}, also have low partitioning power. When comparative operators appear in expressions, a restricted subset of the file is defined. Complex expressions should not be executed piecemeal but should be analyzed to consider the available access facilities to determine how the file can be best partitioned. It is, for example, obviously desirable to combine the two subexpressions in the query

`salary > 10000` $\wedge$ `salary < 12000`

before applying either expression by itself to the file. The use of comparative operators favors indexing over direct access, since serial searches are required.

When the database is in text form, the conditionals in query expressions may become more complex. The query processor has to define words as character sequences between delimiters as blanks, and symbols as {. , ; " '}. The words which appear in text have many variants. Capabilities for obtaining synonyms of words may be needed; it is also desirable to recognize words which have the same stem. Boolean combinations of words which are related may be limited to appear in the same sentence or same paragraph.

An example of a search through a file containing the "What's New" columns from the Wall Street Journal using a `TEXT` option of GIS is shown in Fig. 10-8. The symbol ($) indicates truncation and the symbols (+1), etc., indicate the permissible word distance. The labels L1, L2, and L3 provide references to query expressions. The figure shows also one of the articles obtained.

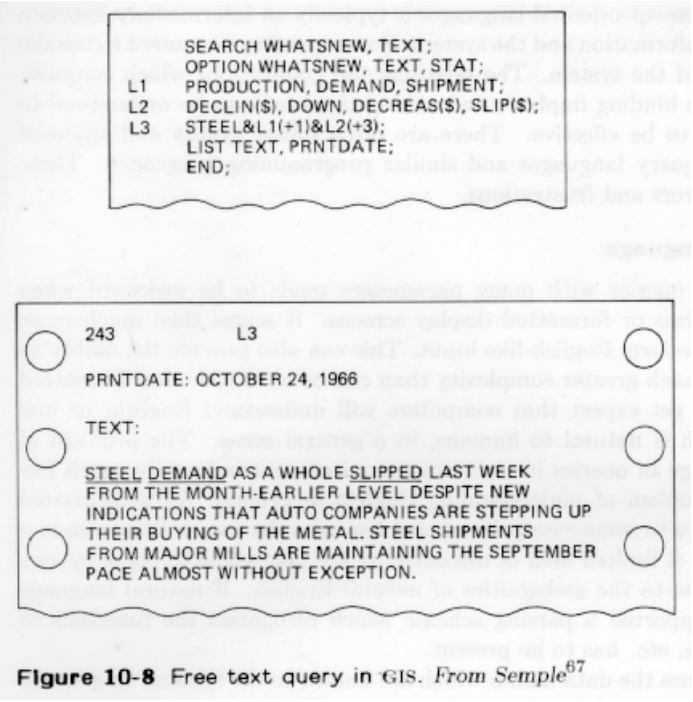


Figure 10-8 Free text query in GIS. *From Semple*[67]

Term Weights If, in addition to `STEEL`, other products which contain combinations of metals to are of interest, the terms in the expression can be assigned *weights* to indicate relevance. A formulation which will retrieve a variety of references is shown in Example 10-5.

Example 10-5 Specification of Term Weights

Terms of interest in the query are assigned measures of importance:

```
WEIGHT: Steel(5),Aluminum(3),Copper(3),Tin(3),Metal(2),ferrous(2)
```

and the query can include the conditional expression

```
TOTAL_WEIGHT ≥ 5 ?
```

The notion of a weight as a measure of interest can also be used in queries applied to formatted data; the query into excess salaries might be formulated as in Example 10-6 to select cases for further investigation. The `TOTAL_WEIGHT` is computed as $\sum$ `sign(value + basis)/scale`. Using the data from Fig. 10-6 the search expression `IF TOTAL_WEIGHT > 0 THEN PRINT employee` will select employees `Calhoun` and `Lusch`.

Example 10-6 Attribute Weight Computation Table

attribute	basis	scale	sign
salary	-10000	/1000	+
age	-20	/10	-
year_hired	-1975		-

The user of statement-oriented languages is typically an intermediary between the requester of the information and the system. Some training is required to use the available facilities and the system. The relations and connections which comprise the database and the binding implemented among them have to be understood to allow this approach to be effective. There are often minor syntax and keyword differences between query languages and similar programming languages. These differences lead to errors and frustrations.

10-3-4 Natural Language

The specification of queries with many parameters tends to be awkward when not supported by forms or formatted display screens. It seems then much more desirable to allow free-form English-like input. This can also provide the ability to have expressions of much greater complexity than can be obtained using formatted queries. We cannot yet expect that computers will understand English, or any other language which is natural to humans, in a general sense. The problem of using natural language in queries in a database environment is happily much less difficult than the problem of understanding English in general. In a formatted database which relates to some enterprise or problem area the user will operate in a well-defined context. A limited area of discourse makes it possible to parse queries without confusion due to the ambiguities of natural English. If natural language queries are being supported a parsing scheme which recognizes the functions of verbs, nouns, adverbs, etc. has to be present.

The schema defines the data names which are known to the system. In general, computer systems distinguish rigorously between attribute names and attribute values, even though both may have overlapping vocabularies. The structure of the internal schema representation used in SOCRATE as illustrated in Fig. 8-7 shows a dictionary which encompasses both attribute names and values from defined domains. Here it is possible to decide which query terms are values and which ones are attribute names. The database dictionary, in addition to the terms used in the schema, should include the words for the basic actions to be performed: `FIND, GET`, etc., as well as those words that describe common operations: `IS, HAS, EQUALS`.

Within limited fields success has been achieved in English-like communication without parsing the query according to grammatical rules. The query is tested only for word matches. If the general parsing problem is avoided, it is not fair to talk about a natural-language or English query capability. The actual effectiveness of pseudo-natural-language systems has to be carefully evaluated, since the examples presented in the literature provided by the supplier are obviously chosen to convey the strength rather than the weakness of the given approach. Pseudo-English can lead to expectations which are eventually frustrated.

Statements in pseudo-English have to be translated into a form which matches the structure of the database. Where the database has a hierarchical structure, one can expect that the query will be decomposed according to rules which generate a hierarchical subtree. When a full or partial match is found, the broom of the node found provides data elements which are relevant to the query. In network data structures interaction is required. Pseudo-English can cope well with the limited path choices, but screen selection will provide faster interaction.

Most systems today which support natural English translate the statements to a relational query format. Relational queries have a high degree of independence from the underlying database structure, unless there are many relations in the database. In that case a universal relation view, as defined in Sec. 9-1-2, can provide another intermediary step. The relational query is subsequently analyzed to permit efficient execution as described in Secs. 9-2-7 and 9-3-2, or it may be mapped into a hierarchical or network system. If the semantics of a databases network structure can be exploited for query processing, the apparent intelligence of the natural-language understanding processes can be greatly enhanced.

Example 10-7 shows the capability of a powerful translator from English to a formal propositional language [Kellogg[71]].

Example 10-7 Processing a Natural-Language Query

`Are all large western cities smoggy?`

is transformed to

`RETURN(TOTAL(IMPLICATION(IS large,(IS western,city)),smoggy))`

for database processing. The evaluation of the expression begins by finding a relationship of type `IS` and selecting tuples with attribute value `western` for objects of domain `city`. A subsequent `IS` operation can be applied to this result relation, etc. We have encountered relational queries in Sec. 9-2 and will not show further examples here.

The relational approach provides the sharpest definition of queries. Not all users will be comfortable with the mathematical formalism implied. If we wish to allow queries to a relational database in either interactive, tabular, or English form, a query translator will translate from those forms to the relational form. It may be instructive as part of the learning process to present the result of the translation from the original statements. The formal form will define the query more thoroughly. The user may eventually want to use or modify the statements in this rigorous form.

10-3-5 Summary

Much work remains to be done in the area of query processing. It is necessary to understand the needs of the user and the human factors in information search in order to find the proper balance between flexibility and expressiveness, ease of use and tedium. There are few examples where operational databases containing important information have been combined with fast-access techniques, adequate interaction capability, and good query languages. The pseudo-English used in query systems has often been more unnatural than computer languages. In hierarchical systems the need to specify accurately the level for the conditional and for the response attributes has been obscured by attempts at query-language simplification. In network approaches the path-choice problem may need interaction during the query processing for information retrieval.

The existence of valuable data in the database is of paramount importance. The best conceivable system will remain a toy if it does not warrant usage, and some primitive systems have seen intensive use because the data they contained was valuable to the community of users.

10-4 DYNAMICS OF INFORMATION-RETRIEVAL SYSTEMS

The design of information retrieval using databases has to match three components. For each of those components a number of choices have been presented.

Some of the choices are strongly associated with the data contents and the tools applicable to manage such data. The interface choice may be made on the basis of both technical suitability and their appropriateness in regard to users' involvement with the system and their sophistication. The underlying *database system* choices determine the the manner in which the data structure is presented to the interface, but also the performance for classes of usage.

Table 10-1 presents a summary. At any specific time, for one specific user, the choice from each column of the menu may be clear. The system designer cannot select just one item from each of three courses on the menu; there are often multiple types of users, and even one category of users may need to move from one approach to another as their objectives change over time.

Table 10-1 Components of an Information-retrieval System

Retrieval methodology	User interfaces	Database organization
	Interrogative	
Fact retrieval	Interactive	Relational
Statistical inference	Tabular	Hierarchical
Deductive inference	Procedural	Linked
	Pseudo English	
	Natural English	

Example 10-8 Phases of Database Usage

Data exploration The initial task is the acquisition of simple facts of immediate interest, i.e., how much oil reserves exist here or there, how much oil is consumed in summer or winter. The correctness of data elements is individually verified and provides a basis for further development of the database. Subsequently a researcher wants to develop an understanding regarding the distribution mechanism and the balance of available resources. Aggregations of resources and consumers are derived and presented to aid in concept formation. Interesting relationships lead to hypotheses which can be verified using statistical techniques on the data.

Evaluation of Hypotheses When a quantitative understanding of the problems has been achieved, alternate solutions can be evaluated. Now relationships between supply and demand, preferences, psychological habits, capital spending, etc., may have to be explored using techniques of deductive inference. Throughout this process the users will have changed in their understanding and expectations of the database and its capabilities.

Planning and Execution In the end there will be a proposal for action and the collected data have to be presented for financial analysis. Here tabulation and graphics to present projections will be useful.

10-4-1 User Dynamics

The change of users' needs over time is illustrated in Example 10-8 using a case of current interest, analysis of energy problems. The example demonstrates how user needs will shift over time in a single application. Other, parallel applications will show differing patterns at differing time frames.

It may be impossible to build single systems that will satisfy the full range of requirements. It is desirable, however, to be able to specify at a conceptual level which choices were made in the design and implementation of the information-retrieval system. The definition of choices at a high architectural level will considerably simplify the lower-level implementation decisions regarding file system support processes, update capability, and cost versus performance trade-offs.

10-4-2 System Dynamics

By definition one specific data value can provide information to a user only once, unless the user has forgotten data received earlier. An information-retrieval system hence remains valuable only if there is continuing change in data contents, data-analysis capability, or users. In order to operate an adequate information-retrieval system, the adequacy of the output in terms of providing useful information has to be monitored, and a mechanism has to be established to enhance both data organization and content as required.

Utilization statistics are one measure of database usage. Attributes which are frequently accessed can be provided with indexes to increase access speeds. Relationships which are frequently investigated can be bound as stored relations or can be materialized through pointer connections. Users who interact with the database system will adapt faster to the system than the systems will adapt to them, and hence exclusive reliance on utilization statistics is not adequate.

The fact that a certain type of retrieval has been used only once in the last year may be due to the fact that the system took more than 24 hours to generate the response, discouraging the user from any further attempts.

Figure 10-9 illustrates the monitoring process for the information-retrieval system. Automatic adaptation of systems to changes in usage has been considered. The potential damage if an infrequent query, asked in an emergency, cannot be answered within an acceptable interval, makes automation risky. Aggregate performance should only be optimized subject to constraints in response time for important queries and update transactions.

In practice the database system will not be judged only by its intrinsic quality but also by its response to the needs of users. Inadequate, unreliable communications systems connected to a good database-management system frustrate users. Existence of inadequate, erroneous data in the database will be equally frustrating. There exists the chicken-and-egg problem that a database system will not produce information without a large volume of high-quality data, and at the same time, it is difficult to convince users of the importance of collecting and verifying large amounts of data unless the system provides useful information in a responsive manner.

Over the lifetime of a database system, the costs of data acquisition and maintenance, will greatly exceed the cost of the database system itself. Procedures to test the content of the database for correctness, integrity, and logical completeness are all part of an effective continuing operation. Their development is part of the system design process and their quality will determine the effectiveness of the system over the long haul. The chapters which follow will discuss these aspects of database systems.

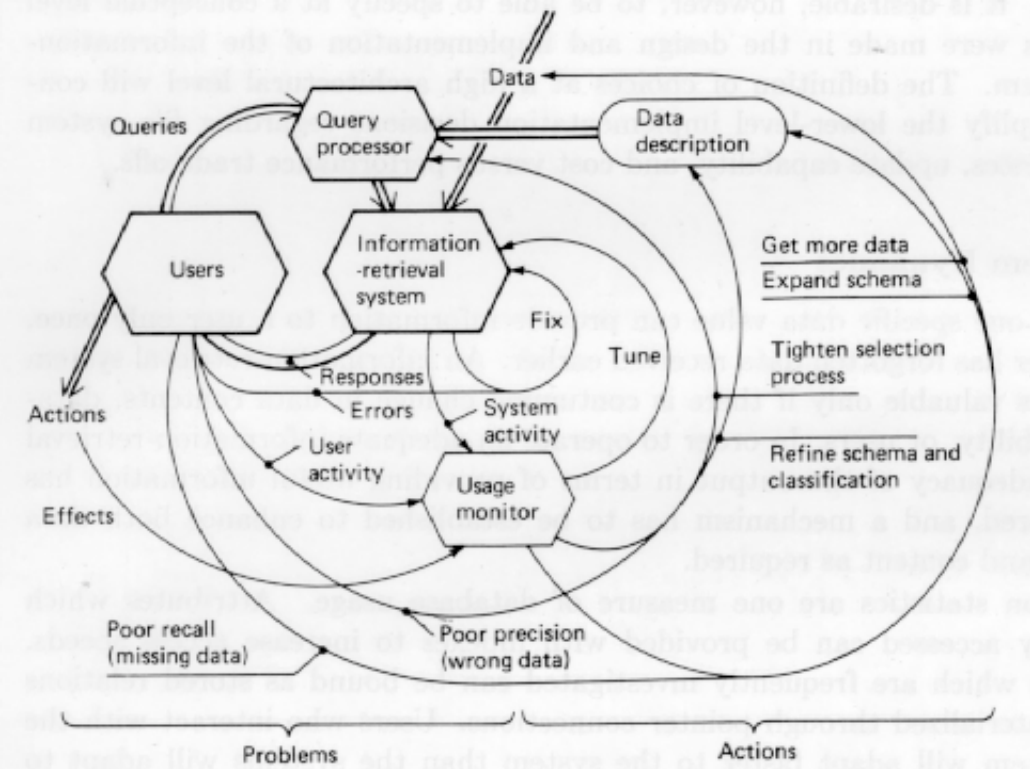


Figure 10-9 Monitoring of an information retrieval system.

BACKGROUND AND REFERENCES

The area of information retrieval is nearly as old as the field of computing itself. As soon as reasonably large external memories were available, people used these memories to store data for data retrieval. An issue of the *Communications of the ACM* (vol.5 no.1, *Jan.* 1962) surveyed the state of the field at that time. The field has been followed for the American Society for Information Systems (ASIS) in Cuadra[66:75] and Williams[76-81]. The literature on information retrieval, although the application of its concepts is critically dependent on database technology, does not provide a comprehensive view of database issues. An early comprehensive view is by Langefors[61]. Nearly every *Proceedings of the National Computer Conferences*, published by AFIPS, contains several papers describing information-retrieval systems. The ACM special interest group on information retrieval (ACM-SIGIR) has published *Proceedings of the Symposia on Information Storage and Retrieval* in 1969 and 1971. Another ACM publication is the *Proceedings of the SIGPLAN-SIGIR Interface meeting on Programming Languages and Information Retrieval* in 1973 (Nance[75]). A source of related material is the *Journal of the Association for Computational Linguistics*.

Much information retrieval work has been oriented toward the management of bibliographic and textual material. Examples of bibliographic systems are SMART (Salton[75]), LEADER (Kasarda[72]), and SPIRES (Martin in Walker[71]). A state-of-the-art summary from a workshop in Palo Alto (Jan. 1971) has been edited by Walker[71], and an AFIPS seminar oriented toward "computer types" at the 1972 FJCC is presented by Clifton[73]. A comprehensive reference is Lancaster[79]; the subject has also been covered by Kent[71] and Meadow[73]. Kochen[74] describes many societal issues of information retrieval.

Some systems TRAMP(Ash[68]), CONVERSE32 (Kellogg[71]) used LEAP-type relations while RDF (Levien[67]) is based on a complex network of relationships. Salton[62] considers queries in tree structures, and Schroeder in Kerr[75] describes an information-retrieval system using trees. Craig[67] describes DEACON, a system using ring structures. Reisner[75] and Rothnie[75] compare relational query alternatives. Several retrieval systems based on the relational calculus are found in Appendix B. Procedural versus nonprocedural queries are evaluated by Welty[81].

Statistical inference is supported by systems described by Rattenbury[74] (OSIRIS), Weyl[75] and Wiederhold[75] (TOD), Joyce[76] (REGIS), and Chan[81] (SUBJECT). Automation of statistical inference is explored by Blum[80] (RX).

Important work in the area of deductive inference processing has been published by Levien[67], Green[68], and Kellogg[71]. Simmons[70] surveyed the state of the art in intelligent query systems. Biss[71] and Minker[78] present later developments. Related papers appear in Gallaire[78,81]. Representations of data in such systems is the topic of papers in Findler[79]. A recent experiment by Dahl[82] uses PROLOG.

Senko[73] and Ghosh[74] treat the problem of finding the access path to data, beginning from the schema entries, in a formal model. Chang[78] uses a schema describing the connections among relations for query processing. The structural connections are exploited by ElMasri[81] to simplify queries. Davidson[80] uses lexical database relations to help parse the queries.

King in Nance[75] proposes an interaction protocol for casual users in business. Dana[72] presents a report generator. Query by example has been developed by Zloof[77]; evaluations have been made by Thomas[75] and by Reisner[77]. Semple[67] and Yu[82] discuss access using weights. Access to a database via a graphic color-display interface is demonstrated by Herot[80].

Cheatham[62] presented a formal approach to English queries, while Zadeh[72] and Chang[78] consider the handling of *fuzzy* queries. How interaction can make pseudo-English effective is shown by Rubinoff[68]. Hammer[80K] and King[81Q] use additional semantic knowledge to help with query processing.

Codd in Shneiderman[78] presents a methodology for interactive natural language query handling. English in a limited context is used by Woods[73] on moonrock data, and by Shortliffe[73] for antibiotic selection. Cooper[76] presents a detailed example of an algorithm to parse English-language queries. Work on an adaptable processor (REL) is described by The use of English as a query language is questioned by Montgomery[72]. Recent progress is due to Harris[77], Sagalowicz[77], Waltz[78], and Hendrix[78]. We see natural language systems now becoming commercially available; some entries are listed in Appendix B.

User expectations are described by Nichols and by Sibley in Jardine[74]. Sibley[74] and Ghosh in Furtado[79] describe experience with very large national systems, and Lancaster[69] evaluates MEDLARS.

EXERCISES

1 Locate a description of an information retrieval system, and

a Describe its objective

b Describe the data structure used

c Describe the database structure used

d Describe the query method

e Discuss the appropriateness of the choices made

f Give suggestion for improvements.

2 Which category of information retrieval can benefit fron the use of deriving data on entry or on query, as described in Sec. 8-3-4?

3 Formulate a query based on the `TEXT` option of GIS to specify the result shown in Examples 10-1 and 10-4. Comment on the suitability of weights.

4 Devise a DBTG-type structure for the data of Example 10-2.