The Headquarters, United States Air Force Command and Control System (473L) is not a command and control system in the sense of the SAGE System. It is, rather, a highly sophisticated planning and management system. 473L has been designed and programmed by the International Business Machines Corporation (IBM) under contract with the Electronic Systems Division (ESD) of the Air Force Systems Command, and under the operational guidance of the Systems Division of the U.S. Air Force Command Post, Headquarters USAF, in the Pentagon.

System 473L has one basic function, to assist the members of the Air Staff of Headquarters, USAF, and higher authority in the planning for, and the monitoring of, military operations involving U.S. Air Force personnel and resources. The Air Staff Officer is therefore the "user" of 473L. It is he who has stated the requirement for data in the system and for programs to manipulate this data. With this as the primary mission of 473L, all systems design has been oriented toward the user, the Air Staff Officer. It must be noted that this Air Staff Officer is not necessarily computer oriented, usually has no computer background, and does not have time to undertake any extensive training in the utilization of a computer system. This then leads to the fundamental concept of 473L. The system must permit the military staff officer requiring data within the system to interact with the computer and its data files, directly, without the assistance of a systems specialist, with a minimum of training. In 473L, the staff officer has this capability.

The heart of any command and control system is its data files. They must be complete, current, and accessible. In automating a command and control system a nucleus of mandatory programs must first be provided. These are a system control program package, a file maintenance program package, and a data retrieval and manipulation program package. Once this nucleus is completed the system may grow to solve the specific problems of the user.

From the standpoint of the user, only the means of retrieving and manipulating data is important. In the 473L System this is provided by the Query Language (QL). The phrase "Query Language" is used to identify the method of communication between the user or system's operator and the data base or system's files. This method of communication is appropriately called a language because it possesses the necessary characteristics of a language, i.e., it has a vocabulary, a grammar, a syntax, and a punctuation set.

The particular QL discussed in this paper represents a second generation of the 473L Query Language. The entire 473L System has been designed and implemented using the "incremental growth" concept (Figure 1, page 3). The original QL was implemented on an IBM 1401 system over two years ago during the "Operational Training Capability" (OTC) phase. The entire OTC system was upgraded to an IBM 1410 in June 1964. The second generation of 473L is scheduled for the end of the summer of 1964 with the installation of a Librascope L-3055 system in the Command Post. This second phase of

development is called the "Initial Operational Capability" (IOC). The development will now continue through a third phase, the "Complete Operational Capability" (COC) which is scheduled for completion in mid-1965.

During the time the Query Language has been in actual operation in the OTC phase of 473L, it and the users were under constant evaluation. This provided invaluable knowledge and experience to the designers of the second generation language. Four of the primary areas in which this experience influenced the design of the present Query Language are: (Figure 2, page 5)*

1. FILE STRUCTURE:

The files in the OTC system were set up in parallel and the language was designed to operate solely on parallel files. Although this made short data retrievals extremely rapid, it created a very time-consuming file maintenance procedure. A parallel file is one where all values for individual properties of the file are stored together, as opposed to a serial file where individual values for all properties of the file are stored together. In the ICC system files are serial-parallel.

2. LANGUAGE SIMPLICITY:

The OTC language made extensive use of special symbols, i.e., asterisk, slash, brackets, etc., to communicate meaningful information to the computer. Experience proved that these special symbols not only made the language more difficult to learn, but also reduced one's ability to remain proficient in the language. In the current QL almost all special symbols have been

*Figure 1 has been deleted.

PRIMARY AREAS OF IMPROVEMENT GAINED THROUGH OTC EXPERIENCE

- FILE STRUCTURE
- LANGUAGE SIMPLICITY
- USER'S LANGUAGE REQUIREMENTS
- COMMON PROGRAMMING FUNCTIONS

Figure 2

removed and punctuation is utilized in a normal English manner. In addition the organization of the statements has been changed to a more English-like structure, thus more closely approaching the ideal man-machine relationship.

3. USER'S LANGUAGE REQUIREMENTS

In OTC we learned that there are in reality three classes of users. The Air Staff Officer, who requires a simple straightforward language; the data control personnel who, through daily use, readily masters the language; and the system programmer to whom the degree of difficulty is secondary to the power of the language. The IOC language is therefore designed to ensure that language complexity is a function of the complexity of the requested action. This means that the Staff officer, desiring a straightforward retrieval, must as a minimum know only a subset of the total Query Language.

4. COMMON PROGRAMMING FUNCTIONS:

During the time that 473L has been operational, many data manipulation functions have been identified as being common to all operational capabilities. That is, they are utilized consistently in the processing of stored data by system operators and system programmers. Examples of this are searching one data file and using retrieved data to search a second file; searching a file to accumulate totals of data within the file, etc. Whenever possible, these common functions have been included in the capability of the current Query Language.

Earlier it was stated that a data retrieval package was a mandatory part of the nucleus of a command and control system. This statement, perhaps, requires some justification. There is a commonality in command and control systems, regardless of the level of command, which sets them apart from most other information processing and retrieval systems. This common factor is the inherent instability of the data files. Any command and control system requires an extremely large data base. This data base is highly dynamic because of the nature of command and control itself. Data must be current, therefore all files are constantly being updated. In addition the requirement for particular data is constantly changing, thereby requiring files to be added, expanded, restructured, and deleted on a continuous basis.

This data base instability therefore generates two requirements in a command and control system which must be satisfied. There must be some tool available to the system programmer which permits him to access data without tying his program to the data files as they exist at that point in time. Secondly, there must be a means of retrieving and processing data from the instant it is loaded into the system data base, without waiting for a new program to be designed, coded, and debugged. These two factors generated the requirement for the Query Language, and the Query Language meets both requirements.

To establish a basis for discussing the various elements of the QL, a sample fictitious file has been constructed (Figure 11, page 31). This sample has been simplified and although 473L file structures per se will not be explained in detail, a few features must, however, be discussed.

The 473L file structure is essentially a serial-parallel type. It is serial in the sense that all properties, or attributes pertaining to one entry, are stored together in the file, and it is parallel in the sense that an entry may have a primary section and a secondary section which are stored separately with each carrying an address connective to be used by the CL in maintaining their relationship. The sample illustrates only a primary section.

Within an entry, values may be carried in four ways:

1. As a single value where only one value is possible for an attribute in an entry. This requires a fixed space for each entry.

2. As multi-values in which each entry would carry a fixed number of values in an entry. This also requires a fixed space in the entry.

3. As multi-values in which each entry carries a variable number of values in an entry. This requires only the space necessary for the actual number of values possessed by that particular entry.

4. As remarks or free text on an entry basis.

In the sample force status file it is apparent that the attributes of the file are: (Figure 3, page 9)

COMMAND	. . .	Air Force Command
MDS	. . .	Model Design Series of Aircraft
UNIT	. . .	Military Unit Designation
AFLD NAME	. . .	Airfield Name
LAT LONG	. . .	Latitude and Longitude of Airfield

SAMPLE SYSTEM FILE

FILE NAME	-	FORCE STATUS
ATTRIBUTES	-	COMMAND
		MDS
		UNIT
		AFLD NAME
		LAT LONG
		ACFT POS
		ACFT RDY
		CREWS FMD
		FUEL CAP
		ELEV
		FUEL TYPE
		RUNWAY LENGTH

Figure 3

ACFT POS	. . .	Number of Aircraft Possessed
ACFT RDY	. . .	Number of Aircraft Combat Ready
CREWS FMD	. . .	Number of Crews Formed
FUEL CAP	. . .	Fuel Storage Capacity of the Airfield
ELEV	. . .	Elevation of the Airfield
FUEL TYPE	. . .	Various Fuel Types Available
RUNWAY LENGTH	. . .	Various Lengths of Runways

This sample file will be used in all the sample QL statements discussed.

The QL itself consists of a vocabulary and a grammar. The vocabulary consists of two parts, a fixed part and a dynamic part. The fixed part consists of unique words which describe and control the retrieval process. A complete list of these unique words, symbols, and punctuation is contained in Appendix A of this paper. The dynamic part of the vocabulary contains words that describe the data to be retrieved; that is, the file indicators, attribute names, etc., of the data base. This portion of the language is automatically changed by the File Maintenance program any time a file is added to the system or modified.

The grammar of the QL is implemented by a syntax and a punctuation set. The syntax refers to the arrangement of words into statements which are meaningful to the system. The syntactical relationships are specified by the punctuation and the unique words. A simple, uniform, English-like syntax is utilized to make the language easy to learn and retain. The punctuation set is also simple and uniform. Each punctuation mark is easily identified and only as many marks as are required to clearly separate the elements are used.

There are seven basic statement elements of the 473L Que y Language (Figure 4, page 12). In order of presentation and common usage the seven elements are the Program Indicator, File Indicator, Qualifier Conjunction, Qualifier, Selector Conjunction, Output Director, and the Output Selector. In certain QL statements some of these elements are omitted and in very complex Query Statements, the elements may be difficult to recognize. Careful analysis, however, will reduce any QL statement to these elements.

1. PROGRAM INDICATOR:

This is the first word of a QL statement. It directs the system control program to use the QL program. It also provides a logical English language beginning for the statement. The word RETRIEVE is a program indicator.

— RETRIEVE —

2. FILE INDICATOR:

This identifies the file from which data is to be retrieved and always follows the program indicator. In the sample file previously dis ussed the file indicator is FORCE STATUS.

— FILE NAME —

3. QUALIFIER CONJUNCTION:

The word WITH follows the file indicator. It serves as a conjunction between the file indicator and the qualifier. It makes the statem ent more readable and precludes the need for punctuation to identify the beginning of the qualifier.

— WITH —

BASIC QL STATEMENT ELEMENTS

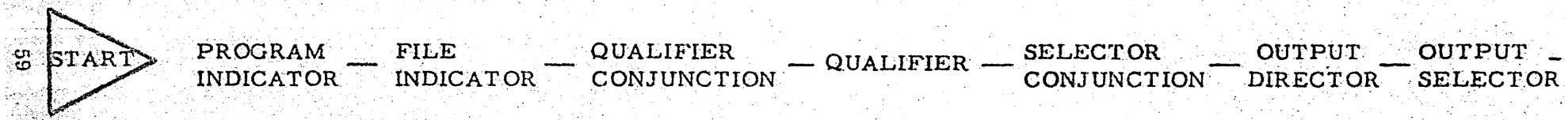


Figure 4

4. QUALIFIER:

This is the element of the statement which describes the specific nature of the data to be retrieved. A qualifier consists of a set of one or more modifiers, each of which is normally composed of an attribute, a comparator, and a value. An attribute is a characteristic of the file; a value is one of the states an attribute may assume; and a comparator defines the logical or mathematical relationship between the attribute and the value. RUNWAY LENGTH is an attribute of the sample FORCE STATUS file and 5000 feet could be a value for RUNWAY LENGTH. The expression "RUNWAY LENGTH >5000" is therefore a valid modifier. Another modifier could be "COMMAND = TAC." Placing these two together as "COMMAND = TAC, RUNWAY LENGTH>5000" forms a modifier set that describes certain entries in the file more specifically. The modifiers in a set are separated by commas and are logically additive; that is, an entry must meet the requirements of all the modifiers in a set to qualify. A simple qualifier contains only one modifier set. A compound qualifier may be constructed by combining several alternative modifier sets. This could be: "COMMAND = SAC, ACFT POS>10; COMMAND = TAC, RUNWAY LENGTH>5000." The semicolon (;) defines the end of one modifier set and the beginning of the next. It also specifies a logical OR relationship between the sets. Data may qualify by meeting either of the modifier set's criteria.

— QUALIFIER —

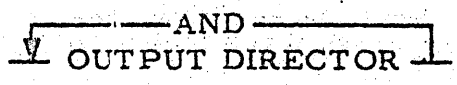
5. OUTPUT DIRECTOR CONJUNCTION:

The word THEN always follows the qualifier. It serves to make the statement more meaningful from an English language view and acts as the separator between the qualifier and the output director.

— THEN —

6. OUTPUT DIRECTOR:

This is normally a single word that specifies the output device desired and the format in which the retrieved data is to be presented.



7. OUTPUT SELECTOR:

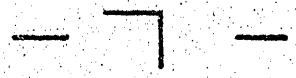
The last part of a QL statement is the output selector. It contains the names of the attributes that are to be presented in the output and specifies the detail arrangement of the output within the general format specified by the output director.

— SELECTOR —

An operator who desires to reuse a Query Statement may at this point in the Query Statement direct that it be added to a SAVE table without processing, by adding the word SAVE at the end of the statement. He may also append remarks to the statement at this time.

— SAVE — REMARKS —

A Query Statement is terminated by an End of Message symbol which is available on the 473L equipment.

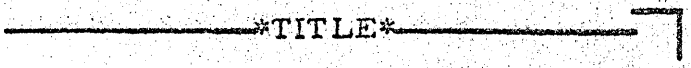


OUTPUT OPTIONS:

There are two optional features available which the operator may use in conjunction with the OUTPUT DIRECTOR and the OUTPUT SELECTOR to order his data and to title his report. The sequence of attributes in the output is controlled by the order in which they are listed in the statement, and the values are normally listed in the sequence in which they are retrieved from the file. To develop a more useful output the operator may specify that the data is to be sorted. The expression for SORT is either INCR, increasing (i.e., A to Z alphabetically or in increasing numerical order) or DECR, decreasing, which is the converse. Sorting instructions can only appear in the output selector portion of the statement and are expressed in the following form.

Sorting may be performed on a number of different attributes in one statement. The attribute listed first will be the primary sort, the attribute listed second is the secondary sort, etc.

The other option in the output is the ability to give an QL retrieval a title by enclosing the desired title between asterisks. The title must be entered following the OUTPUT SELECTOR but preceding the terminal punctuation as follows:



The statement format used in all QL statements is achieved by organizing all of the preceding basic elements as shown in Figure 5, page 17.

FUNCTIONS:

In addition to the basic elements and options previously discussed, there are a number of special functions in the QL that may be expressed in the qualifier and/or selector portions of a statement (Figure 6, page 18). These functions provide the ability to qualify on or to generate and select data based on data criteria not explicitly stored in the file. The functions are: Great Circle Distance (GCD) that computes the distance between two geographic points; SUM - which accumulates the sum of several values of one existing attribute; and the MIN/MAX function which selects the minimum or maximum value among several attributes on an entry basis.

To illustrate the flexibility and power of these various special functions of the Query Language, a more detailed look at GCD, SUM, and MIN/MAX is warranted.

QL STATEMENT USING BASIC ELEMENTS

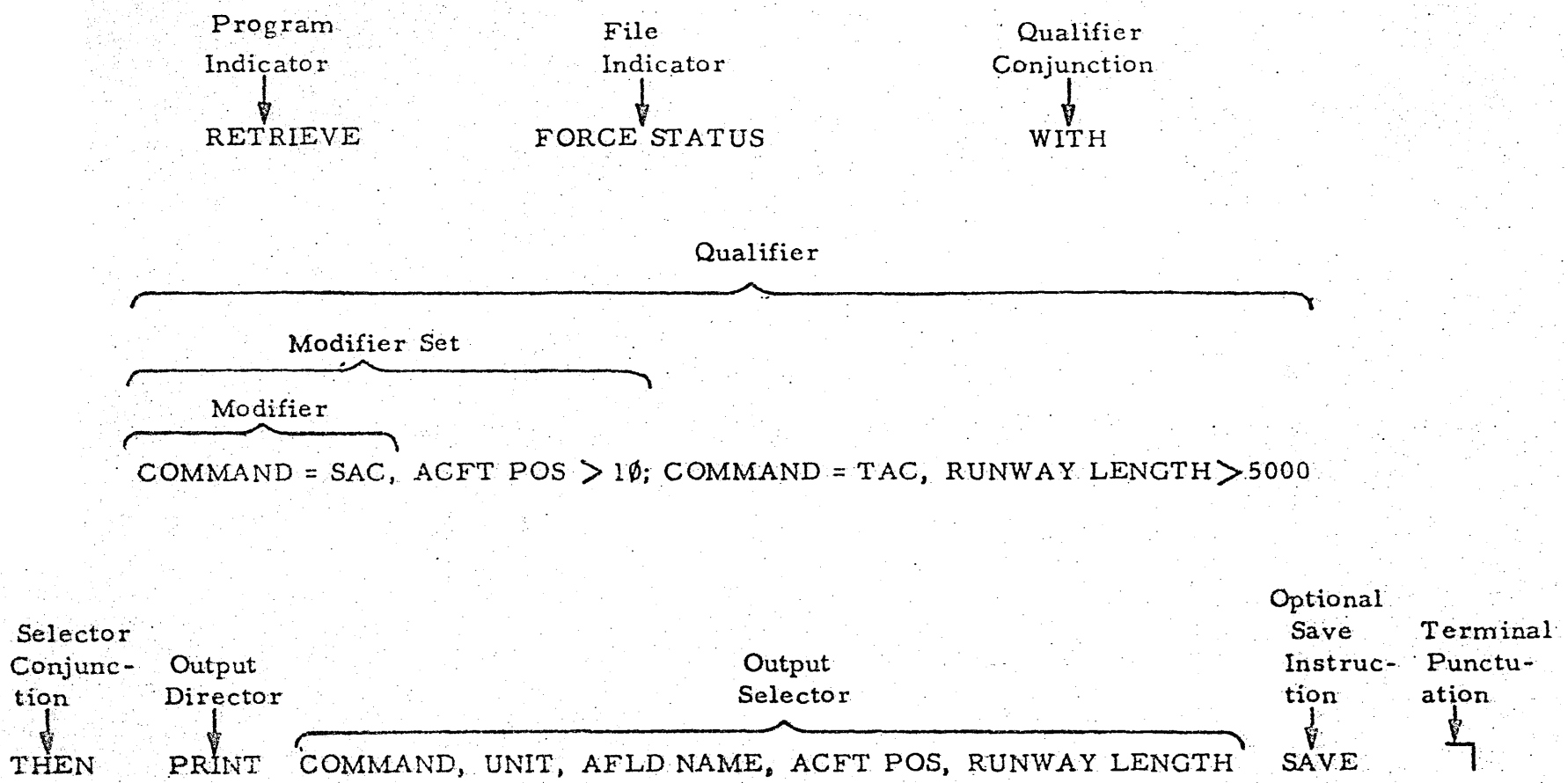


Figure 5

QL FUNCTIONS

GCD — GREAT CIRCLE DISTANCE COMPUTATION

SUM — SUM COMPUTATION

MIN/MAX — MIN/MAX COMPUTATION

Figure 6

GREAT CIRCLE DISTANCE

The Great Circle Distance function (GCD) (Figure 7, page 20) can be used in two separate ways to qualify data to be retrieved: as a modi .er and/or as a value of a modifier. When used as a modifier, the GCD funct on replaces the attribute, comparator, and value and is called a computed attribute. It is expressed by using term GCD followed by the common geographic point, comparator, and value enclosed with parenthesis. This is illustrated in qualifier format 1 in Figure 7. A typical example is GCD (DEN VA< 500).

When used only as the value of a modifier, it is called a omputed va.ue. It is expressed by using the term GCD followed by two geographic points separated by a comma and enclosed within parenthesis. This is illustrated in the value portion of qualifier format 2 in Figure 7. A typical example is GCD(DENVA, SHAW BEACH).

Both forms, computed attribute and computed value, may be used in one modifier. When used in this fashion a combination of the indiv dual forms previously discussed are used. This is illustrated in qualifier format 3 in Figure 7.

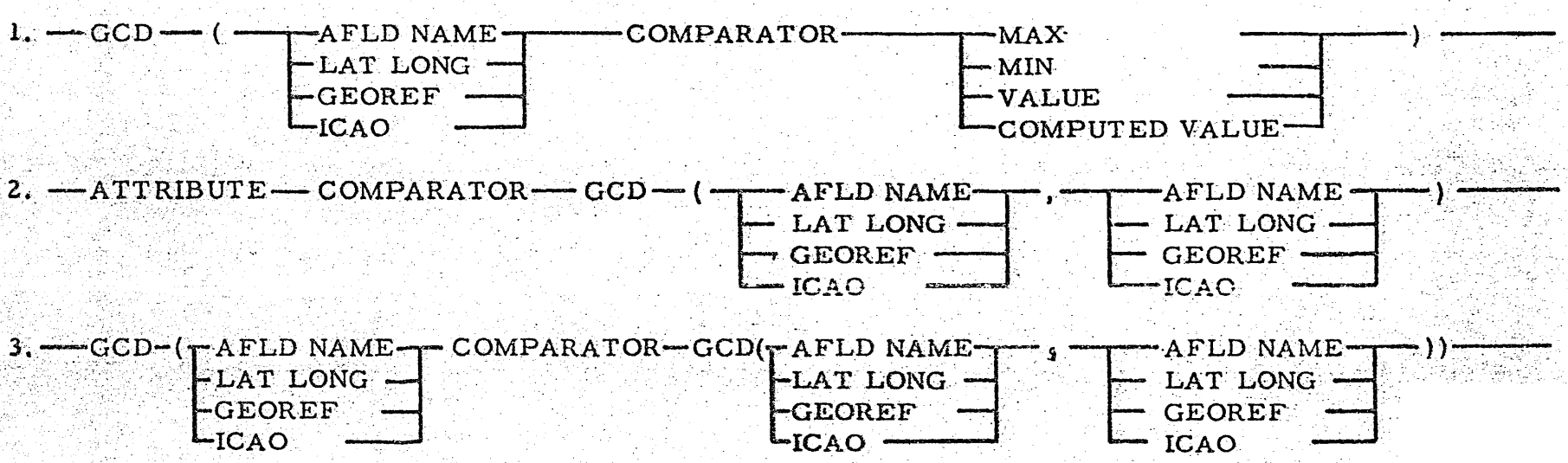
Using the sample file attribute names and values, GCD as a computed attribute, and computed value in the same modifier could actually appear as follows in a statement.

GCD (DENVA < GCD (DENVA, SHAW BEACH)

This modifier would have the following meaning when translated to English:

GCD
GREAT CIRCLE DISTANCE FUNCTION

QUALIFIER FORMATS



SELECTOR FORMATS

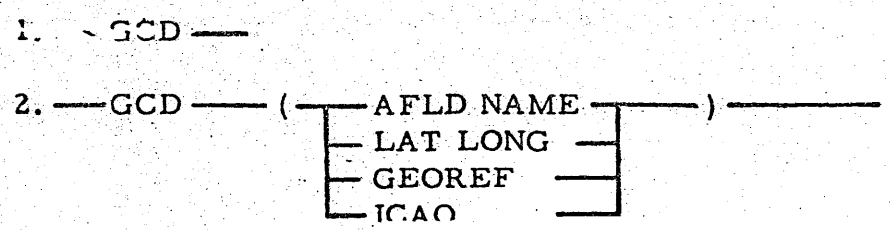


Figure 7

Qualify all entries within a distance from DENVA equal to or less than the distance between DENVA and SHAW BEACH.

When used in the selector portion of a statement, the GCD function can only be used as a computed attribute and must be expressed in one of two different forms. The short form of GCD would be expressed by using just the term GCD.

The short form is used to select the qualified results of a GCD function used as a computed attribute in the qualifier portion of the statement. It eliminates the need for the operator to express the complete GCD function twice if he desires to qualify using the GCD function, and to select the qualified GCD's.

When the operator desires to obtain the GCD from any specific point for all qualified airfields, the long form, or complete expression, can be used. It is expressed using the term GCD followed by the geographic point enclosed within parenthesis. This illustrated in Selector Format 2 in Figure 7. A typical example is GCD (DENVA).

In all GCD functions illustrated, except the short form, the terms AFLD NAME, LAT LONG, GEOREF, and ICAO were used. These terms represent the various ways of defining a geographic point that can be interpreted by the QL program.

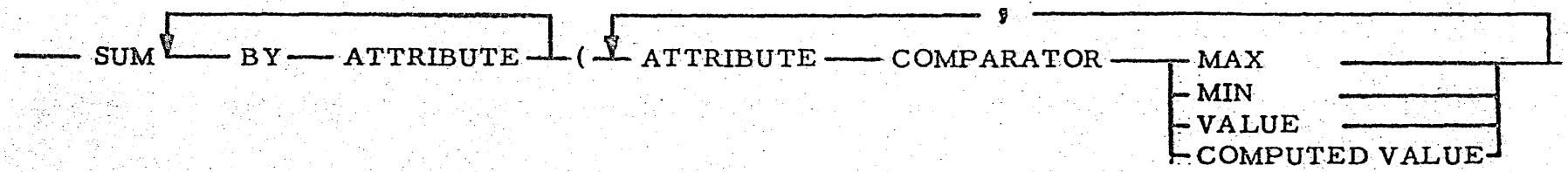
SUM

The SUM function is used to obtain the sum (i.e., total) of the values of an attribute for use as a qualifying attribute. (Figure 8, page 22.) It can be per-

SUM

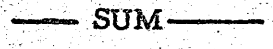
SUM FUNCTION

QUALIFIER FORMAT



SELECTOR FORMATS

1.



2.

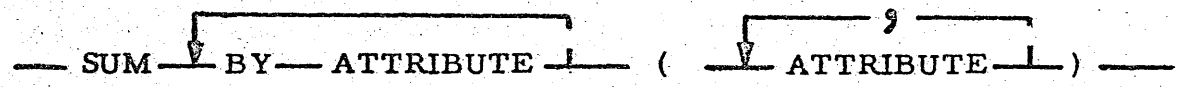


Figure 8

formed for any numeric attributes that are properly specified in the statement.

The SUM function can be used in either the qualifier or selector portion of a statement. The general form in the qualifier portion of a statement is expressed using the term SUM followed by the attributes used to control the function with the attributes to be summed and their respective comparators and values enclosed within parentheses. This is illustrated in the qualifier format in Figure 8, page 22. A typical example is SUM BY COMMAND (ACFT RDY $>$ 14).

The attributes appearing before the left parenthesis of the SUM function are called SUM control attributes. They are used to order the resultant SUMS. The attributes appearing within the parentheses are called the summed attributes and may be any attributes with numeric values. Summed attributes in the SUM function when used in the qualifier must be followed by a comparator and a value. It can take two different forms in the selector portion of the statement, the short form and the long form.

The short form is used in the selector when the SUM function was used in the qualifier and the resultant sums are desired as output. Just the term SUM is used to select the short form.

The long form is used when it is desired to select summed attributes that are not contained in the SUM function in the qualifier or when it was not used in the qualifier. The form is the same as that required in the qualifier except comparators and values are omitted. This is illustrated in selector format 2 in Figure 8. A typical example is SUM BYCOMMAND (ACFT RDY, ACFT POS).

For all functions appearing in a statement, the ' UM control attributes appearing in the first SUM function must be the same for all SUM functions in the statement.

MIN/MAX

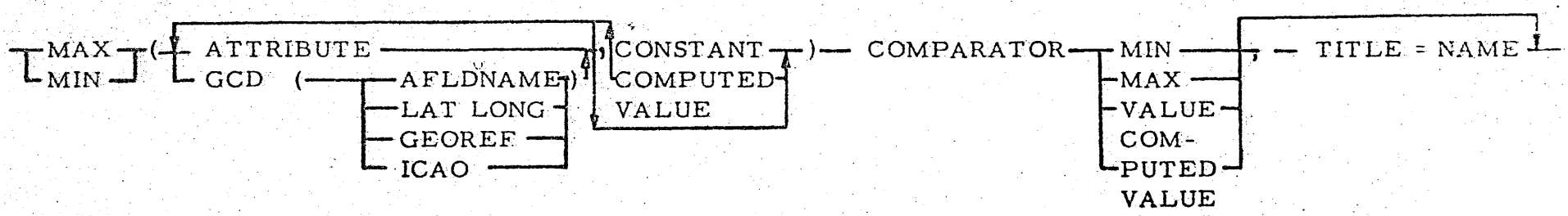
It is often desirable to specify in the qualifier that the largest or smallest value of any one of several attributes should exceed a given value. This function is provided in the QL by the computed attribute MIN/MAX (Figure 9, page 25).

The MIN/MAX function is used in the qualifier to determine which of several attributes has the numerically smallest (or largest) value for each otherwise qualified entry, and whether this value is equal to (or greater or less than) a specified value. A constant may also be specified in place of one of the attributes. The general form of this function in the qualifier is expressed by using the term MIN or MAX followed by the attributes, computed attributes, computed values and constant enclosed in parentheses and followed by the comparator, value, and title. This is illustrated in the qualifier format in Figure 9. A typical example is MIN (ACFT RDY, CREWS FMD) > 4, TITLE = SYSTEMS RDY.

If a MIN/MAX function in the qualifier is to be selected for inclusion in the output, it is possible to assign a unique title to the resulting list of values. If a title is not specified, the word MIN or MAX (as appropriate) will be used. A title for output purposes should be specified if more than one such function

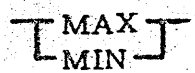
MIN/MAX
MINIMUM/MAXIMUM FUNCTION

QUALIFIER FORMAT

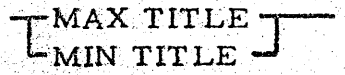


SELECTOR FORMAT

1.



2.



3.

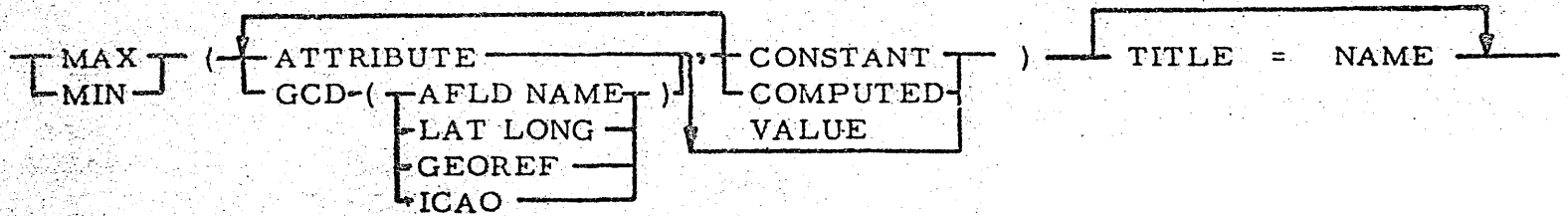


Figure 9

is to be included in the output. As is illustrated in the above example, a title may be specified for a MIN/MAX function in the qualifier by entering it immediately after the function. The word TITLE must immediately follow the value of the MIN/MAX function, separated by a comma, and followed by an equal sign and the actual title.

The rules for using MIN/MAX in the qualifier all apply in the selector, except that a comparator and a value must not be used. Again, it is possible to assign a title to the resulting list of values. There are several ways of expressing a MIN or MAX function in the selector.

The words MIN and/or MAX are used when requesting MIN or MAX functions not titled in the qualifier to be selected in the selector. This is illustrated in Selector format 1 in Figure 9.

The titles assigned are used when MIN or MAX functions titled in the qualifier are to be selected in the selector. This is illustrated in Selector format 2 in Figure 9.

The long form is used when MIN or MAX functions are requested only in the selector. This is illustrated in selector format 3 in Figure 9. A typical example is MIN(ACFT RDY, CREWS FMD) TITLE = SYSTEMS RDY.

COMPLEX QUERIES

An additional feature of the QL that was not previously discussed, the complex query, provides the user with a powerful tool. A complex query provides the ability to develop a statement that is composed of several sub-

ordinate queries. Each subordinate query is separated by a colon (:) and can address the same or a different system file. The complex query retains data retrieved by a prior subordinate query and automatically inserts the desired data in succeeding subordinate queries as values of modifiers. Essentially it provides the ability to use data from other files to qualify and select data from one file. Since only one sample file is given in this paper the full power of the complex query cannot be properly illustrated; however, an example is given using only one file.

Assume it is desired to retrieve all SAC units from the sample file that possess at least as many aircraft as the 413ATW at DENVA. This could be performed by using two separate statements by first retrieving the number of aircraft possessed for the 413ATW then using the retrieved value in the modifier of another statement to qualify those units meeting that requirement. However, one complex query can perform the same task.

```
RETRIEVE FORCE STATUS WITH COMMAND = SAC, UNIT =
413ATW THEN RETAIN ACFT POS: RETRIEVE FORCE STATUS
WITH COMMAND = SAC, ACFT POS> [R1, ACFT POS, OR]
THEN LIST UNIT, ACFT POS¬
```

The above complex query would retrieve the value for ACFT POS for the 413ATW in the first subordinate query and automatically insert it as a value to the ACFT POS modifier of the second subordinate query. The second subordinate query would retrieve all units meeting the desired requirements. The colon (:) is used to separate the subordinate queries, and the information enclosed within brackets directs the QL in locating the data to be used. The

R1 denotes the working file, the attribute ACFT POS denotes the attribute values to be used, and the OR denotes the relationship of those values when using them in the statement.

All previous discussions have related to the IOC Query Language. It is felt that some brief mention should be made of additional major features being planned for in the COC Query Language. These include a means of expressing mathematical computations that can be used in either the qualifier or selector portions of a statement. This will be performed by our planned COMPUTE function. The capability to extract information from several files to generate a new file that can then be queried will be performed by our planned COMBINE function.

This paper could not treat all areas within the scope of the IOC QL without becoming unduly long. Appendix B contains several detailed examples of QL statements and their results. Figure 10, pages 29 and 30, is a complete QL chart that shows all elements of the 473L Query Language and their relationship to one another. It is hoped that sufficient discussion has been given to demonstrate the requirement in a command and control system for a QL and to demonstrate the versatility of the 473L Query Language.

The authors of this paper acknowledge that the development of the IOC Query Language was only made possible through the team effort of everyone who has contributed or worked in this area of 473L. The Program Design Concept and Operational Specification, which were developed by this team and are unclassified technical documents published as part of 473L contract, were used in preparing this paper.

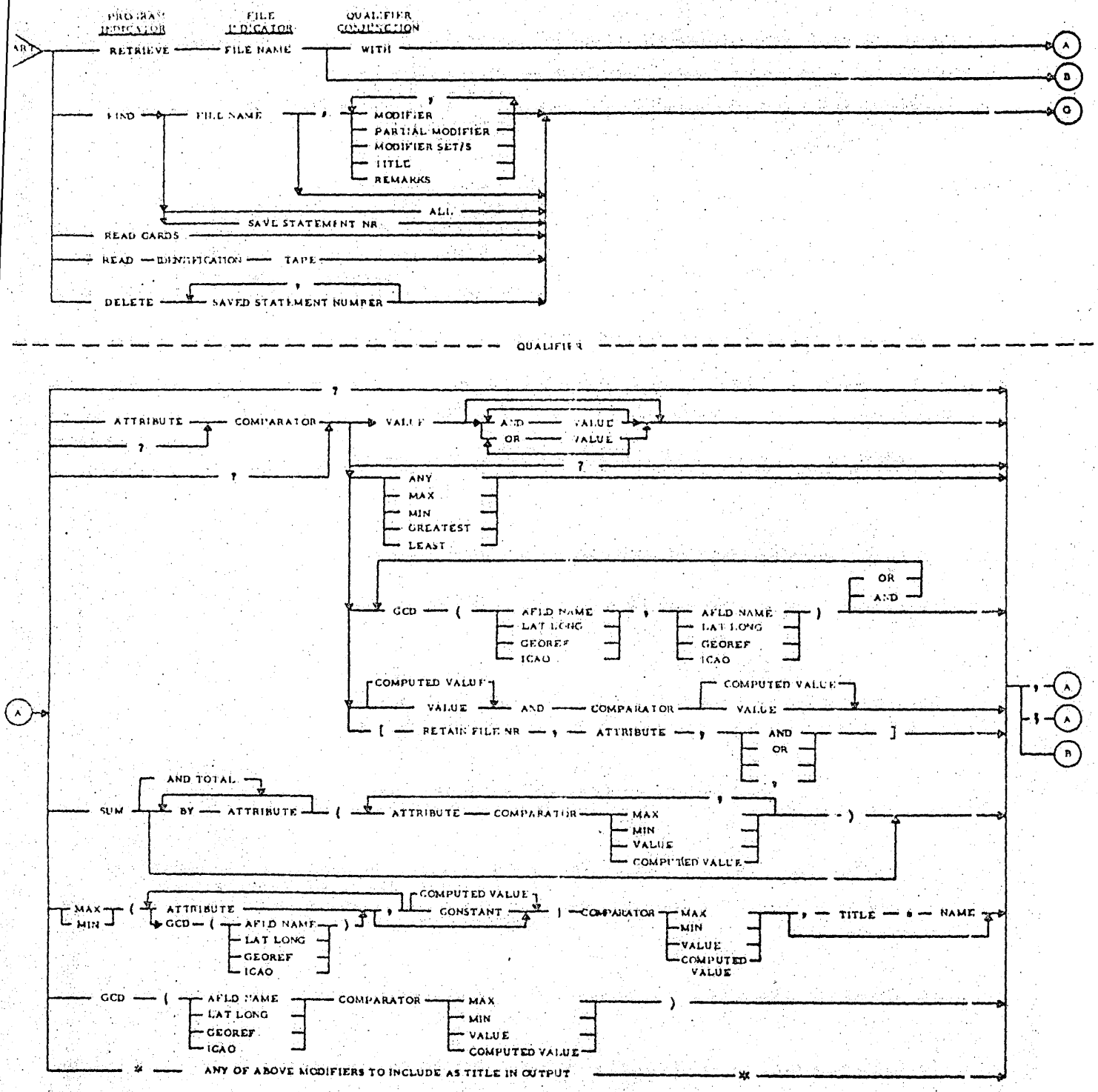


Figure 10. Query Language Element Relationship
(Sheet 1 of 2)

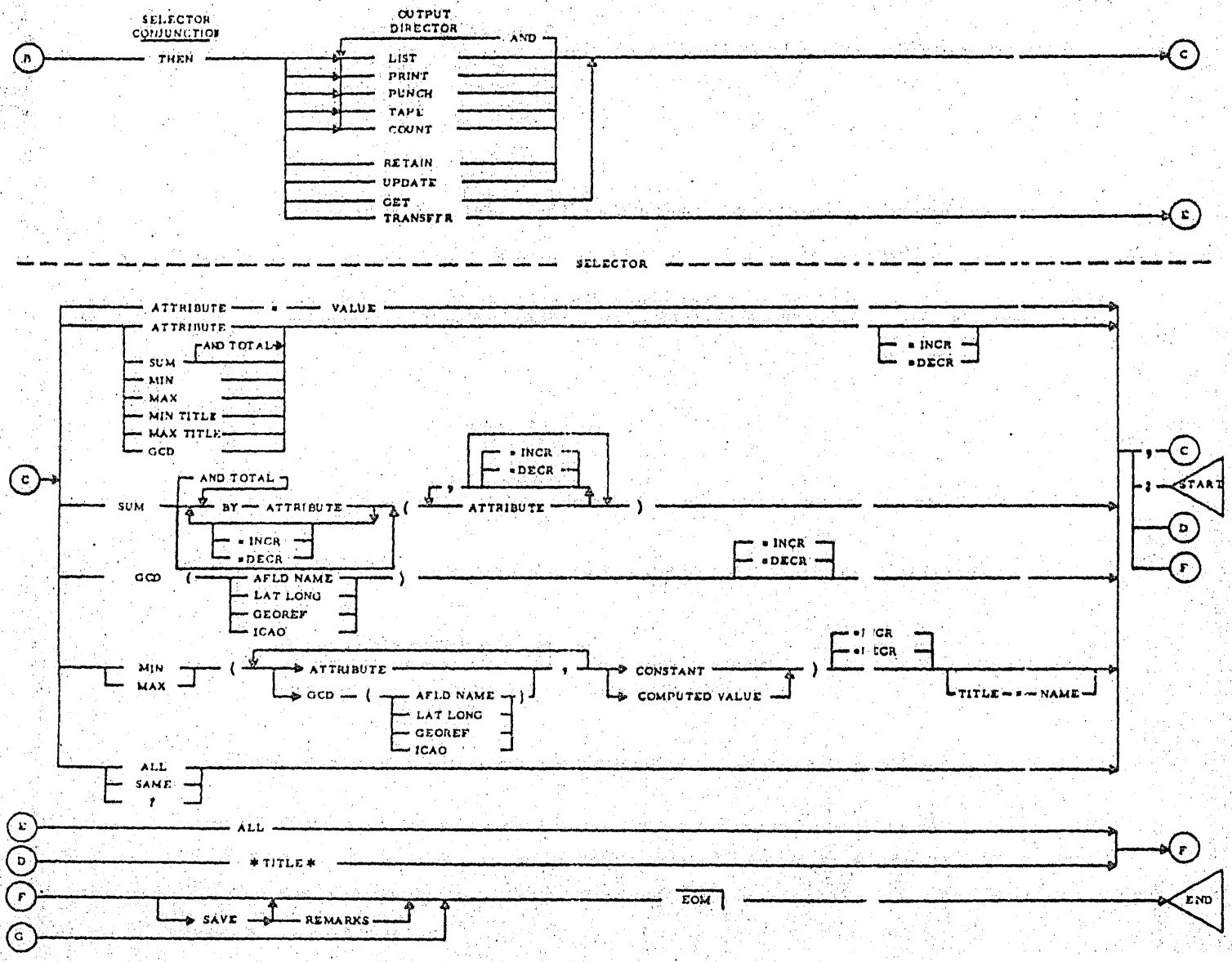


Figure 10. Query Language Element Relationship (Sheet 2 of 2)

FILE NAME
FORCE STATUS
ATTRIBUTE NAMES

FILE ENTRIES

ENTRY	COMMAND	MDS	UNIT	AFLD NAME	LAT LONG	ACFT POS	ACFT RDY	CREWS FMD	FUEL CAP (KGAL)	ELEV	FUEL TYPE	RUNWAY LENGTH
(1)	SAC	B47A	453BW	PODUNK	0000N0000W	020	014	012	28	0200	JP4 JP9 JP5 AV3	5000 2000
(2)	MATS	C119A	413ATW	DENVA	0835S0020E	031	028	014	32	0001	JP3 AV3	8000 4000
(3)	TAC	F104A	2340TFW	RATTLE	0714S00530W	002	002	001	05	0064	JP4 AV4	4000 5000
(4)	MATS	C124A	414ATW	DRY GULCH	0518N0032E	017	016	015	14	0218	JP5 AV4 AV3	8000 6000
(5)	SAC	B47A	453BW	DUNCAN	0649N0716W	016	012	012	16	9999	JP4 JP5	7050 5500
(6)	TAC	F104A	358FSW	SHAW BEACH	4317N07118W	010	004	005	38	0001	JP3 AV3	6500 7500
(7)	SAC	B47A	453BW	BURNFLY	5600N11842E	007	003	001	99	0000	JP3 JP4 AV3 AV4	5050 8500
(8)	MATS	C124A	2345ATW	DRYDEN	4317S11842E	020	014	002	00	0014		2000 4000
(9)	TAC	F104A	2340TFW	CLARK	4320N07058W	010	003	003	01	0093	JP3 JP4	5500 7000
(10)	SAC	B47A	92BW	SHAW BEACH	4317N07118W	008	004	002	19	0001	JP3 AV3	6500 7500
(11)	TAC	F104A	358FSW	COALVILLE	8840N02317W	030	018	004	22	0413	JP9	4900 3500
(12)	SAC	B47A	2359BW	BEVENS	8840N02000E	011	006	006	20	0012	JP4 AV5	7700 8800

Note: Numbers shown in parenthesis to the left of each entry were inserted to aid in referencing [illegible] statements in this [illegible].

Figure 11. Sample System File

APPENDIX A

Punctuation and Unique Words

Punctuation and Unique Words

The punctuation set used in QL defines the syntactical relationships in the format and separates the various elements. The punctuation set also includes the comparator symbols and the marks used to denote computational functions. Certain unique words are used to perform certain functions. The following summarizes the punctuation and unique words and defines their functions.

Punctuation Mark	Symbol	Meaning
Apostrophe	'	Defines the beginning and end of an insert to a QL statement; used in error correction and in completing incomplete saved statements.
Asterisk	*	Encloses titles.
Brackets	[ ]	Encloses retained values in complex QL statements.
Colon	:	Separates subordinate queries in complex QL statements.
Comma	,	1. Separates modifiers in the qualifier. 2. Separates attributes in the output selector. 3. Separates locations in GCD expressions, summed attributes in SUM expressions and attributes in MIN/MAX expressions. 4. Separates retained attribute name from Working File Indicator and OR or AND in retained values. 5. Separates tape reel numbers when addressing a tape file. 6. Separates File Indicator and statement characteristics in the FIND procedure.

Punctuation Mark	Symbol	Meaning
Comparators Equal to Not equal to Equal to or greater than Equal to or less than	 = ≠ ≥ ≤	Separates attribute and value in a modifier and indicates relationship between them.
EOM (End of Message)	¬	Denotes end of QL statement
Lozenge	◻	Marks end of QL statement that is displayed after being retrieved by FIND or is included in an Error Message. Inserted by program.
Parentheses	( )	1. Encloses the attribute names on which a computation or other data processing operation is to be performed. 2. Encloses the tape reel numbers when addressing a tape file. 3. A left parenthesis and underline; e.g., (____, are used to define the space in which typing should begin in certain procedures.
Question Mark	?	1. Defines the location where portions of a saved incomplete QL statement have been omitted. 2. Precedes Error Messages. Inserted by program.
Semicolon	;	Separates modifier sets in compound qualifiers.

Unique Words	Meaning
ALL	Used as a collective selector to retrieve values of all attributes for the qualified entries.
AND	1. Denotes logical "and" relationship among multiple values of an attribute. 2. Connects certain other unique words, such as output directors.
ANY	Used as a value to qualify on any of several specific values in certain attributes.
BY	Identifies SUM control attributes in the SUM expression.
CARDS	Indicates the source of input (used only with READ).
DECR (Decreasing)	Used as a sorting instruction to sort values in decreasing order.
GCD (Great Circle Distance)	The name of a computed attribute or value; directs such computation.
GET	A special purpose output director used by other system programs to retrieve data without immediate output. (Program use only.)
GREATEST	Used as a value to qualify on the largest of a multi-valued attribute in an entry.
H (Horizontal)	Used as a format indicating suffix with output directors to indicate that, in the output, attribute names should be in a horizontal row.
INCR (Increasing)	Used as a sorting instruction to sort values in increasing sort.

Unique Words	Meaning
LEAST	Used as a value to qualify on the smallest value of a multi-value attribute within an entry.
LIST	An output director used to direct the output to the ET display.
MAX (Maximum)	1. The name of a computed attribute; used to determine which of several attributes in an entry has the largest value. 2. Used as a value to determine the otherwise qualified entry having the largest value for an attribute.
MIN (Minimum)	1. The name of a computed attribute; used to determine which of several attributes in an entry has the smallest value. 2. Used as a value to determine the otherwise qualified entry having the smallest value for an attribute.
OR	Denotes logical "either/or" relationship amoung multiple values of an attribute.
PRINT	An output director used to direct the output to the line printer.
PUNCH	An output director used to direct the output to the card punch.
READ	A special program indicator used to initiate read-in of QL statements from the card reader or tape unit.
RETAIN	A special purpose output director used in complex QL statements to indicate that the retrieved values of the selected attributes are to be retained in working storage for later used by other statements.

Unique Words	Meaning
RETRIEVE	The primary program indicator for the QL used to identify an input as QL and to initiate the QL program.
RN (Retained File Number; e.g., R3)	Used in retained values to indicate from which working file the values are to be obtained for insertion in the statement.
SAME	Used as a collective selector to retrieve values of all attributes listed in the qualifier.
SAVE	A special word used to indicate that the QL statement is to be stored for later use.
SUM	The name of a computed attribute; used to direct the summing of all qualified values of the indicated attribute(s).
TAPE	1. An output director used to direct the output to magnetic tape. 2. Used with READ to indicate the source of input.
THEN	Output director conjunction.
TITLE	Used when assigning titles to computed attributes for output purposes.
TOTAL	Directs that a total sum be generated with subordinate sums.
TRANSFER	A special purpose output director used to initiate the bulk transfer of data from one file to another (program use only).
UPDATE	A special purpose output director used to change values stored in the data base.

Unique Words	Meaning
V (Vertical)	Used as a format indicating suffix with output directors to indicate that in the output the attribute names should be in a vertical column.
WITH	The qualifier conjunction; used to separate the file indicator from the qualifier.

Note: Certain of the above unique words will not be defined beyond what is given under Meaning in this section.

APPENDIX B

Example Query Statement

EXAMPLE STATEMENTS

To illustrate the versatility of the QL, a number of example statements are provided. The initial statements will be composed in a simple form and will become progressively more and more complex. The example statements will be arranged in three parts: the requirement, the QL statement required to fulfill the requirement, and the appropriate comments concerning the statement. All example statements pertain to the sample system file discussed earlier.

Requirement: To retrieve the Unit Designation and Airfield Name for all units belonging to SAC.

RETRIEVE FORCE STATUS WITH COMMAND=SAC THEN LIST AFLDNAME, UNIT ⌐

The above statement contained only a simple qualifier. Values for the attributes AFLDNAME and UNIT from entries 1, 5, 7, 10, and 12 would qualify and have been retrieved.

Requirement: To retrieve all units in SAC or TAC that possess 10 or more aircraft and the number of aircraft that each possessed.

RETRIEVE FORCE STATUS WITH COMMAND=SAC OR TAC, ACFT POS>1Ø THEN LIST UNIT, ACFT POS ⌐

The above statement illustrated OR'd values in a qualifier. Values for the attributes UNIT and ACFT POS from entries 1, 5, 6, 9, 11 and 12 would qualify and have been retrieved.

Requirement: To retrieve all units in SAC that are located at airfields that possess JP9 fuel and all units in TAC that are located at airfields that possess JP4 fuel.

RETRIEVE FORCE STATUS WITH COMMAND=SAC, FUEL TYPE= JP9; COMMAND=TAC, FUEL TYPE=JP4 THEN LIST UNIT ¬|

The above statement illustrates a compound modifier. Entries 1, 3 and 9 would qualify and have been retrieved.

Note the importance of compound qualifiers. The example prior to this illustrated OR'd values and perhaps would lead one to construct this QL statement in the following manner:

RETRIEVE FORCE STATUS WITH COMMAND=SAC OR TAC, FUEL TYPE=JP9 OR JP4 THEN LIST UNIT ¬|

This statement would have qualified entries 5, 11 and 12 in addition to entries 1, 3 and 9 that qualified in the previous statement. Close analysis of the first statement shows that for an entry to qualify it must belong to SAC and possess JP9 fuel or belong to TAC and possess JP4 fuel. The second statement shows that an entry will qualify if it belongs to SAC or TAC and possesses JP4 or JP9 thus producing the wrong results.

Requirement: To retrieve the total and individual number of aircraft possessed by TAC and the total and individual number of aircraft possessed by MATS.

RETRIEVE FORCE STATUS WITH COMMAND=TAC OR MATS THEN LIST SUM BY COMMAND (ACFT POS), ACFT POS, UNIT ¬|

The above statement illustrates the use of the SUM function in the selector. MATS entries 2, 4, 8 and TAC entries 3, 6, 9, 1; would have qualified and been selected. However, the SUM function would have also generated two SUMS, one for TAC and one for MATS, in addition to the attributes COMMAND and ACFT POS. This would appear in the output as follows:

COMMAND	SUM ACFT POS	ACFT POS	UNIT
TAC	52	2	2340TFW
		10	358FSW
		10	2340TFW
		30	358FSW
MATS	68	31	413ATW
		17	414ATW
		20	2345ATW

Requirement: To retrieve a list of all commands that possess more than 60 aircraft.

RETRIEVE FORCE STATUS WITH SUM BY COMMAND (ACFT POS > 60) THEN LIST, SUM, ACFT POS, UNIT ⌐┘

The above statement illustrates the SUM function in the qualifier and the short form SUM in the selector. In this example, the entries 1, 5, 7, 10 and 12 that belong to SAC would qualify as a group since the total of ACFT POS for these entries totals 62, allowing them to qualify. Entries 2, 4 and 8 belong to MATS and would also have qualified since the total ACFT POS when summed is 68. Entries 3, 6, 9 and 11 would not qualify since they totaled 52 which was less than the total specified in the qualifier 60. The output for this statement would appear as follows:

COMMAND	SUM ACFT POS	ACFT POS	UNIT
SAC	62	20	453BW
		16	453BW
		7	453BW
		8	92BW
		11	2359BW
MATS	68	31	413ATW
		17	414ATW
		20	2345ATW

The short SUM expression in the selector selected the summed attribute ACFT POS that was used in the SUM expression in the qualifier.

Requirement: To retrieve a list of all airfield names, units and number of aircraft possessed and crews formed that have a number of crews formed or Aircraft Ready, whichever is greatest, equal to or greater than five (5).

RETRIEVE FORCE STATUS WITH MAX (ACFT RDY, CREWS FMD) >5, TITLE=MAX FUNCTION THEN LIST UNIT, MAX FUNCTION, ACFT RDY, CREWS FMD ⌐

The above statement illustrates the MIN/MAX function. Entries 1, 2, 4, 5, 6, 8, 11 and 12 would qualify. The output would appear as follows:

AFLD NAME	UNIT	MAX FUNCTION	ACFT RDY	CREWS FMD
PODUNK	453BW	14	14	12
DENVA	413ATW	28	28	14
DRYGULCH	414ATW	16	16	15
DUNCAN	453BW	12	12	12
SHAW BEACH	358FSW	5	4	5
DRYDEN	2345ATW	14	14	2
COALVILLE	358FSW	18	18	4
BEVENS	2359BW	6	6	6

In reviewing the above example, the entry pertaining to SHAW BEACH is significant, since it is the only entry with more crews formed than aircraft ready. It illustrates the case in which the crews formed attribute, rather than the aircraft ready attribute qualified the entry when the MAT function was processed.

The example also illustrates the MIN/MAX function and the TITLE feature that is available with this function.

Requirement: To retrieve a list of all airfields that have a Runway Length equal to or greater than 8000 ft. and their distance in nautical miles from DENVA airfield.

RETRIEVE FORCE STATUS WITH RUNWAY LENGTH>8000 THEN LIST AFLD NAME, GCD (DENVA), RUNWAY LENGTH ⌐

This illustrates the GCD function in the selecto . Only ent ies 2, 4, 7, and 12 would qualify since they are the only entries that possess a Runway Length equal to or greater than 8000 feet. In addition to the list of airfield names selected, the GCD function would have been performed on each of the qualifying entries to compute their distance fr n DENVA and the resultant computed distances listed. The output woul l appear as follows:

AFLD NAME	GCD DENVA (MILES)	RUNWAY LENGTH
DENVA	Ø	8ØØØ
DRY GULCH	5Ø5	8ØØØ
BURNTLY	1Ø32	85ØØ
BEVENS	732	88ØØ

The actual GCD values shown in this illustration are only used to aid the discussion and are not the actual GCD values that would have been computed based on the actual LAT LONG coordinates in the sar ple file.

To illustrate GCD in the qualifier as a compute attributed and computed value, using the previously generated GCD's as a basis, assume that a requirement exists to retrieve a list of airfields that have a Runway Length equal to or greater than 8000 feet and that are a distance from DENVA equal to or greater than the distance between DENVA and BEVENS. The statement would appear as follows:

```
RETRIEVE FORCE STATUS WITH RUNWAY LENGTH >8ØØØ, GCD
(DENVA>GCD (DENVA, BEVENS), THEN LIST AFLD NAME, GCD,
RUNWAY LENGTH ¬
```

The resultant output of this statement would appear as follows:

AFLDNAME	GCD DENVA (MILES)	RUNWAY LENGTH
BURNTLY	1Ø32	85ØØ
BEVENS	732	88ØØ

USE OF UNIQUE WORDS

To supplement our discussion of simple and compound modifiers and the use of the various functions available in the QL, it is necessary to illustrate briefly a number of unique words that were previously defined.

Qualifier

a. MAX and MIN as values

```
RETRIEVE FORCE STATUS WITH ACFT POS = MAX THEN
LIST UNIT, ACFT POS ¬
```

This statement will retrieve only the entry in the file with the maximum number of aircraft possessed. Use of MIN would have retrieved only the entries in the file with the minimum number of aircraft possessed.

b. GREATEST and LEAST

```
RETRIEVE FORCE STATUS WITH RUNWAY LENGTH =
GREATEST THEN LIST AFLDNAME, RUNWAY LENGTH ¬
```

This statement will retrieve every entry in the file that possessed at least one runway. Only the longest runway length for each entry, however, would have been selected.

c. ANY as a value

RETRIEVE FORCE STATUS WITH FUEL 'YPE = ANY' THEN LIST AFLDNAME, FUEL TYPE ¬

This statement will retrieve all entries from the file that possessed at least one type of fuel.

Selector

a. ALL as a selector

RETRIEVE FORCE STATUS WITH ACFT POS>20 THEN LIST ALL ¬

This statement will qualify all entries in the file that possess 20 or more aircraft and list all attributes for the qualifying ntries.

b. SAME as a selector

RETRIEVE FORCE STATUS WITH COMM ND=SAC,ACFT RDY>10 THEN LIST SAME ¬

This statement will qualify only entries belongin to SAC tha: possess 10 or more aircraft. In the output, only the attributes used for qualifying, COMMAND and ACFT RDY, would be selected.

d. INCR and DECR in selector

RETRIEVE FORCE STATUS THEN LIST COMMAND=INCR AFLDNAME=DECR ¬

This statement will retrieve the COMMAND and AFLDNAME attributes for the entire file since no qualifier is used. In addition, the resultant output would be sorted alphabetically in increasing sort for command with AFLDNAME sorted in decreasing order for each COMMAND.