# SPECOL—A computer enquiry language for the non-programmer

*By* Bernard Smith*

**This paper describes a new Everyman's Enquiry language for use with Computers. The language is usable on almost any file merely by the changing of a few look-up tables and other data in a small reserved area of the program. Examples are given showing the language in use on three types of file; files differing in both form and content. The paper concludes with a summary of the advantages that are foreseen in the general field of direct customer to computer access.**

(First received December 1967, and in revised form March 1968)

SPECOL is a Special Customer Oriented Language which may be learned in a few minutes of study and may be and is being used by people who have no experience of conventional computer programming or of computer techniques. Users moreover need have no knowledge of the medium (disk, drum, magnetic tape, etc.) on which their data are stored.

The language was written in the United States for IBM 7010 and System 360 Computers. The basic compiler program is written in the machine codes of both computers and at present requires less than 25,000 characters of storage. It could probably be written for other ranges of computer in the matter of a few months. The language has been used successfully on numerous files, some of them exceedingly large—over several millions of records. Especially important is the speed and ease with which questions can be put to the computer and the quick turn round in jobs that can be expected. A search through a million records of a typical file, written 800 bits to the inch and blocked in 40-line blocks, takes about 15 minutes on an IBM 7010. Time on System 360 depends on the particular configuration in use and on the amount of parallel processing going on at the time.

## General concepts

The first aim of the language is that, within the vocabulary allowed, it should be as natural to the user as possible. This has been done by allowing the user to make his own selection of nouns for the language and thereafter by keeping other terms to a minimum. The principal connectives in the language are AND, OR, and NOT. These plus a few simple conventions enable quite complex questions to be put to the computer and make the use of other words unnecessary.

The second aim of the language is that it should be usable on any file, irrespective of its country of origin, content, or structure. This has been achieved merely by *viewing* information as falling naturally into a series of sets or nested sub-sets of data, which are then addressed and combined according to logical laws. Instead of re-organising the basic data into structures and then designing a language to match these structures, it is possible to leave the data as it is and manipulate it using only ANDs, ORs, and NOTs. The user, although he may not be aware of it, is following some elementary Set Theory and the language itself is in fact a complete Algebra of Sets. Users of the simplest files need not be aware of all conventions in the language, but in any case it is a very easy step from the simplest type of SPECOL to the full SPECOL that can be used on the most complex of files.

## Basic features of the language

The *subject matter* of the files on which the language may be used is virtually limitless, ranging perhaps from scientific and technical indexes to historical or geographical data, political, business, management files, marketing data, inventories, police and personnel records, car registrations, hotel guides and even menus for the housewife. It is possible, though this would be at some cost in efficiency, to use the language entirely on natural text.

*Variable features of files*, that is, features that may differ between one file and another (data identification, field names and other characteristics) are catered for in SPECOL in a small Control section of the program. The information necessary to complete this part of the program is worked out with the customer, usually in the matter of a few minutes, when he first decides he wants to use SPECOL on his file.

*Files* may consist of single-line records, multiple line records, or even records which can only be described in terms of paragraphs and sentences, i.e. in terms of sets and sub-sets of lines. Lines may be of any length, though for any one file will be fixed at a maximum. The form and content of lines may differ depending on some identification (usually a tag or some other criterion) positioned in a known place of the line. Searches may be made within or between these lines or subsets of lines.

* 28 *Queens Court, Queens Road, Cheltenham, Glos.* [*Mr. Smith is a Civil Servant employed in a Government Research Establishment. The SPECOL project has the support of the Treasury and several other Government Departments.*] © Crown Copyright

For example a user may specify whether he wants to find the two words Nuclear and Reactor (*a*) next to each other, (*b*) within one sentence, (*c*) within one paragraph or (*d*) merely anywhere in a record.

*Records* may be generally of any length, i.e. any number of lines. If the output from any individual record is likely to be large, say more than 20,000 characters, it is possible, with some computer configurations, that special provisions may need to be made.

*Fields* in a record may be of any length, though for any one particular field its length must be fixed. Fields may be assigned any name beginning with a letter. The Field Names form part of the Control section referred to above.

*Data items* (Search data) are indicated quite simply in parentheses following the relevant Field Name. For example, if we want all Captain and Major SMITHS in WASHINGTON who weigh less than 180 lb, and are in the age range 25 to 30, but are not taller than 6 feet, we could write

RANK(MAJOR OR CAPT) AND NAME(SMITH)
AND PLCE (WASHINGTON) AND WGHT (<180)
AND AGES (25 TO 30) AND NOT HGHT (> 6)

If we are merely testing for the existence or non-existence of data in a field we may omit the parentheses. For example, if we are looking for records which have names but not ranks, we could write simply:

NAME AND NOT RANK

Again (assuming, of course, that our data is in our file to begin with) if we want a list of towns in the UK or US that have a population of more than 5,000 but no doctor, we can write:

CTRY (UK OR US) AND TOWN
AND POPN (> 5,000)
NOT PRFN (DOCTOR)

At present, SPECOL statements are punched on standard punched cards, but could just as easily be entered remotely from a computer console or terminal. Statements may be punched anywhere within the 80 positions of a card. If a statement requires more than one card, this may be indicated by a space (blank coding) followed by a continuation symbol. Blank codings in a statement are ignored by the compiler except in three instances:

(1) There must *not* be a space within either a field name or a data item. If a space is genuinely required, it must be punched as a slash, e.g. PLCE (NEW/YORK).

(2) There must be at least one space before and after the connective OR and the connective TO when used within parentheses,

e.g. NAME (JONES OR SMITH)
*TIME (1200 TO 1400 OR 1800).

(3) There must be a space before a continuation symbol.

In all other cases, spacing is entirely at the discretion of the user. He may if he wishes use it for tabulating his statements. This is especially effective in long requests which contain similar statements. The main advantage of the feature, however, is that unless one is tabulating one does not have to worry about accidentally tapping the Space Bar.

An especially useful feature in the language is the so-called *universal character* or *dot* feature. A dot within a data item indicates that any character is permissible in that position. If the dot is also the last character in the data item, it indicates that any character is permissible from that position up to the end of the field. For example, the expression

NAME (SM.T.)

would retrieve SMITH, SMYTHE, SMITHSON, SMITHSONIAN, etc. There are numerous advantages to this capability. The following are the more important.

(1) It caters for errors and variants, e.g. if a man's name may be HANSEN or HANSON, we may write HANS.N and cater for both.

(2) It caters for hyphenated words, sometimes a problem in retrieval, e.g. Multi-access, Stand-alone, Time-sharing.

(3) It enables one to expand the specific into the general. For example if ACS is the notation for American Cargo Ships, and APS for American Passenger Ships, then A.S will stand for all American Ships,

| | | |
|---|---|---|
| MBS9 | = | Men's Black Shoes size 9 |
| MCS8 | = | Men's Brown Shoes size 8 |
| M.S. | = | Men's Shoes. |

(4) The final dot limits the length of data to be compared, thus saving punching or writing-out time. It also reduces the need to have variable length fields in a record, since a field may now be padded out to a maximum agreed length without fear of using additional computer time. Fields are automatically reduced in the computer process to the length of the data item concerned.

There are the usual capabilities in the language for extracting values: less than (<), greater than (>), equal to (no symbol), and anything but (—), as well as for indicating alternatives and ranges. There is also a capability of searching the last positions of data items. For example, if one is not sure of a person's

* Note the mixture of TO and OR connectives. The TO has the closest relationship in such cases; the expression being interpreted as TIME (1200 TO 1400) or TIME (1800). There may be up to 20 ORs or TOs in any one set of parentheses and up to 20 ORs or ANDs between Field Names.

name but vaguely remembers the end of it, one may write:

```
NAME (@ ITH)
NAME (@ IT. OR @ N.ON)
```

The symbol @ indicates 'Search the end of the data item for . . .'.

A field area may be *scanned* in all positions by using the Query symbol (?). For instance, to scan a Remarks field for the word TRAVEL or configuration S.IT.SON, we may write:

```
RMKS (? TRAVEL OR S.IT.SON)
```

Another useful feature is the capability of ignoring leading blanks or zeros in a field. Often items in a field may begin with insignificant blanks or zeros or a mixture of both. These can be ignored for matching purposes by prefixing the data item with a colon ( : )

```
e.g. NUMB (: 123 OR :4567)
     DLRS(: < 500.00)
     ADRS(: 123. OR :7890.)
```

There is an added subtlety here. If there is no dot following the item, this means the item is to be right-adjusted: that is right up against the right end of the field. If there is a dot, this means that the data may 'float'. In the last example, both bb001234 and bb789012 would be matches. Note that the colon may also be used in conjunction with other SPECOL symbols <, >, and dot etc.

A further feature of the language is the ability to use fictitious field names to initiate specially written routines. For instance, in SPECOL the field name DAOW is used to calculate the Day of the Week for any day in the 20th Century. On recognising the Field Name in a request the program will automatically jump to a routine and do the necessary work. In this way, ages might be calculated from birthdates, interest from principal, periods of activity from up and down times, and so on.

A judicious combination of the features described above should, it is believed, enable a user to reach any required part of his file, as well as perform some of his specially required chores. Among the latter, the most useful so far incorporated is the ability to make independent over-all counts of further classes and sub-classes of items occurring in the selected records. The whole range of the SPECOL functions is available for this purpose. An example of the feature is given in the second hypothetical question described below.

Instructions to print out the results of a search may be specified quite simply and with considerable flexibility.

```
Examples: (1) CTRY/TOWN/POPN
          (2) 5/NAME///PRFN/SEXX STAT
```

It is merely necessary to write down the name of the field(s) from which one wants data. A single slash between Field Names indicates one space required; two slashes, two spaces and so on. No slash means no space required between the fields (e.g. SEXX STAT above); a figure before the slash is yet another way of indicating spaces.

## Model files and some hypothetical questions

Three levels of file complexity are discussed below. The hypothetical questions on the files illustrate how SPECOL deals with them.

### (1) Files with single-line records

These are the simplest files. Each line is of the same length and identically structured. Let us assume a file of Vehicle Registrations of which the following are *some* of the fields and field names,

| | | | |
|---|---|---|---|
| REGN | Registration | MFGD | Manufacturing date |
| MAKE | Make | EXPD | Expiry date |
| COLR | Colour | NAME | Name of owner |
| BODY | Body | ADRS | Address |
| | | TOWN | Town |

Now let us suppose we want the owner of a Blue or Gray Hillman Saloon dated about 1946 to 1948 with a Registration number which ends in 25; or 1 something 35. The following SPECOL program would find and output the relevant records.

```
MODE  1 SPECOL 1.1
TYPA  COLR (BLUE OR GR.Y) AND MAKE
        (HILL.)
AND   BODY (SAL.) AND MFGD (46. TO 48.)
AND   REGN (@ 25 OR @ 1.35)
PNTA  REGN/COLR/MAKE/BODY/MFGD
AND   NAME
AND   ADRS/TOWN
```

*OUTPUT*

```
JAD 2125 BLUE HILLMAN SALOON 460601
JOHN  K  SMITH
41  NORTHWEST AVE    TEWKESBURY
DAD  1835 GREY HILLMAN SALOON 470901
JACK   P    JONES
59  WESSEX  ST    STRATFORD
```

The first statement in all SPECOL programs is a Mode and Title statement. The Mode No. is concerned with output and with this simple type of file will always be 1, indicating that output is to come from the one and only line of the selected record. The Mode No. is followed by an arbitrary Job Name. This serves as a cross reference between the computer log and the output results. It is entirely up to the user what he puts in this space. The computer log contains details of number of records searched and found, amount of output etc.

The second statement begins TYPA which introduces the Search section, and again with this type of file will always be TYPA. Additional statements in the section always begin with AND or NOT. Each statement refers to a group or set of data, i.e. this set of conditions must be satisfied before the program proceeds to the next statement.

Note the dot in GR.Y in case the word was spelled GRAY or GREY. HILLMAN, SALOON, and the dates are abbreviated to save time. The end of the Registration number is found by using the @ sign.

The *OUTPUT* section begins with PNTA and each new statement causes output to begin on a new line.

As another example, the following program will find cars registered in Birmingham, Bristol or Manchester, dated before 1957, but excluding Yellow Rolls Royces or Black Bentleys. Out of the selected records it will also count (1) the number of 1945 to 1950 cars, (2) the number of 1951 to 1956 cars; (3) cars coloured BLUE or RED (one class), (4) FORDS and (5) AUSTINS.

```
 MODE   1 SPECOL 1.2
 TYPA   TOWN (BIRM. OR BRIST. OR MANCH.)
 AND    MFGD ( < 57.)
 NOT    COLR (YELL.) AND MAKE (ROLLS.)
          OR ¢
        COLR (BLACK) AND MAKE (BENT.)
 PNTA   TOWN
 AND    REGN
 AND    10/COLR/MAKE
OVCNT   MFGD(45. TO 50.) (51. TO 56.) COLR
          (BLUE OR RED)  ¢
        MAKE (FORD) (AUSTIN)
```

*OUTPUT*

```
 BIRMINGHAM
 AOC 193
                GREEN        AUSTIN
 BRISTOL
 PH 2147
                BLACK        FORD
```

*COUNT TOTALS*

```
 350 MFGD 45. TO 50.
 800   „  51. TO 56.
 420 COLR  BLUE OR RED
 150 MAKE  FORD
 210   „   AUSTIN
```

Note the use of OR *between* Fields. Alternatives must appear in the same statement though it is permissible to continue statements over more than one request line by using the ¢ Continuation symbol. It is usually a good idea to factor out the common data (i.e. TOWN and MFGD) and write this first. The NOT of the NOT statement extends over the entire statement. This is true only when the NOT is the first term of a statement.

Note the 10/ in the third output statement which causes the output to be indented 10 positions.

The counts are self explanatory, although in fact other by-product counts not explained here are also produced automatically by the program.

**(2) Files with Headers and Trailers**

A slightly more complex file is one that consists of records containing a Heading line (called a Header) and subordinate lines (called Trailers). The trailers are all of the same type and structure:

| |
|---|
| Header |
| Trailer |
| Trailer |
| Trailer |

Let us consider a file of World Computer Centres and their personnel, from which we will select the following fields and field names;

*Heading field*

| | | |
|---|---|---|
| CTRY | Country | |
| CENT | Centre | |
| SVCA | Services offered: | Analysis |
| SVCO | | Optical scanning |
| SVCR | | Remote access |

*Trailer fields*

| | | | |
|---|---|---|---|
| NAME | Full Name | TITL | Title |
| FNME | First Name | STAT | Status |
| INIT | Initial | SLRY | Salary |
| SNME | Surname | JOBD | Job Description |

Even in using this file, if the search and output statements concern only Heading data, then the procedures are exactly as shown in the previous examples. If, however, the program in some way also concerns trailers, it becomes necessary to use some additional conventions. Firstly there are additional MODE Numbers. MODE 1 indicates that only Heading data is required for output; MODE 2 that only those trailers containing matched data are required; and MODE 3 indicates that *all* trailers of a selected record may be required for output even though some of the trailers do not contain any part of the match. Secondly there are additional search and output terms. TYPB introduces the Trailer Search section and PNTB the Trailer output section. Let us assume that we want a list of Married Engineers or Single System Analysts working at UK Computer centres which have Remote Access facilities as well as Optical Scanning. The following program would suffice;

```
MODE    2 SPECOL 2.1
```

```
TYPA    CTRY (UK)
AND     SVCR AND SVCO
TYPB    STAT(M) AND JOBD(ENGR) OR
          STAT(S) AND JOBD(SYSA)
PNTA    CTRY/CENT/SCVR SVCO
PNTB    STAT/JOBD/NAME
```

*OUTPUT*

```
UK    LONDON UNIVERSITY RO
           M ENGR JAMES P SMITH
           S SYSA WAT   A PERUSAL
           S SYSA ANN   A LYZER
UK  BRISTOL IBM         RO
           S SYSA  IMAN X PERT
           M ENGR  JOHN T JONES
```

Mode 2 has caused only those lines containing matched data to be output. Parentheses are not necessary with SVCR and SVCO since any data in these fields means a match. Note that what follows the OR in the TYPB statement is a complete alternative to everything that precedes it in that statement. In the output no space is required between SVCR and SVCO, hence the omission of the slash. Trailer output is automatically indented the length of the Heading unless otherwise indicated.

For a second example let us write a program to list all unmarried programmers with salaries between £3,000 and £4,000 per year, and let us assume in order for the output to be sorted later into Name order, that the output is required in single lines:

```
MODE     2 SPECOL 2.2.
TYPB     NOT STAT(M) AND JOBD (PROG)
AND      SLRY ( : 3000 TO  : 4000)
PNTA+    CTRY/CENT
PNTB     STAT/JOBD/SLRY/NAME
```

*OUTPUT*

```
UK LONDON  IBM  S  PROG  3500
  RICHARD S  JONES
UK LONDON  IBM  S  PROG  3200
  HAZEL    T  GREEN
UK LONDON  IBM  S  PROG  3000
  JOHN     P  WHITE
FR  PARIS    GE   S  PROG  3300
  RENE     S  DUPONT
```

Note that no Search is required of Heading fields so the first Command Term is TYPB. Since the NOT is *not* the first term of a statement it applies only to the immediately following field. SLRY is a right-adjusted field, hence a colon will cause any leading blanks or zeros to be ignored. The + after PNTA causes Header and Trailer output to appear on the same line. Heading data is repeated as necessary. Output may now be sorted by Name.

A common request with this type of file is that of stipulating the presence of one condition in one trailer, another condition in a second trailer and so on, all conditions to be satisfied for the record to be required. This is sometimes called *Cross-Trailer-working*, and so the SPECOL term for this in TYPBX.

Let us assume we require from a file the name of the centre that employs three people one named MISS HELPHER, the second MR. COUNTWELL and the third MR. PERT. We are not interested if only two of the three are present.

```
MODE    2 SPECOL 2.3
TYPBX   SNME (HELPHER) AND TITL (MISS)
AND     SNME (COUNTWELL) AND TITL (MR)
AND     SNME (PERT) AND TITL (MR)
PNTA    CTRY/CENT
PNTB    7/TITL/NAME/JOBD
```

*OUTPUT*

```
UK BRISTOL  IBM
  MISS TRULY A HELPHER    SEC
  MR   WILL  I COUNTWELL  ACCT
  MR   IMAN  X PERT       SYSA
```

All three persons are found so the record is selected. Note that only part of the name (SNME) was searched for, but that the whole name (NAME) was output.

**(3) Files with Headers plus Trailers of different types and with Trailers also grouped into sets or paragraphs**

This is a rather more complex type of file but nevertheless one that is fairly common. Let us take as an example, an index to technical journals. In this, the Heading line will contain general information about the journal, price, publisher, reference, etc. Each article in the journal will be assigned a serial letter and this will be the set or paragraph letter. For each article there will be one or more Trailers and each Trailer will be tagged with a letter to indicate its contents:

| | | |
|---|---|---|
| | H | Header |
| Article A | A | Author |
| | S | Subject |
| | G | Gist |
| Article B | A | Author |
| | S | Subject |
| | K | Keywords |

The following might be some of the fields:

*Heading fields*

| | | | |
|---|---|---|---|
| HDTE | Date of Issue | HPRD | Period |
| HTLE | Title | HCTY | Country |

*Trailer fields*

| | | | |
|---|---|---|---|
| ANAM | Author's Full Name | KWRD | Keywords |
| ASNM | Author's Surname | PNAM | Name of Person in Text |
| ATWN | Author's Town | PSNM | Surname of Person in Text |
| SUBJ | Subject | | |
| GIST | Gist | PTWN | Person's Town mentioned in Text |

Let us assume that we require a journal containing articles by Bull of London and O'Flynn of Dublin in the same issue:

```
MODE    3  SPECOL 4.1
TYPBX   ASNM(BULL) and ATWN(LONDON)
AND     ASNM(O'FLYNN) AND ATWN
          (DUBLIN)
PNTA    (ALL)
PNTB    0/ANAM/SUBJ
```

*OUTPUT*

```
 H   641005      SPORTING LIFE   MTHY  UK
BA   BULL        JOHN              LONDON
BS                  FISHING IN THE THAMES
EA1  O'FLYNN MIKE                  DUBLIN
EA2  O'GRADY SEAN                   CORK
ES                  IRISH WILDLIFE
```

In TYPBX working, a single SPECOL statement now refers to a single paragraph or set—in this case, one article. In addition, consecutive but different Field Names with the same first letter refer to the *same* sentence or trailer (BULL and LONDON must appear in the same sentence). Note that a whole Header may be output by using PNTA (ALL). The two articles extracted are lettered B and E. O'FLYNN and O'GRADY are co-authors. Although Author and Subject Field Names appear in the same PNTB statement, output is in different lines. Column alignment, however, is preserved. Mode 3 was used since S trailers were required in the output but were not mentioned in the Search. Paragraph and Tag letters and suffixes are output automatically.

As a last example let us take a browsing type of query —the requirement for an article on Physics written in the summer of 1963 by someone with a name ending in INGTON or INKTON. The article had something to do with ALPHA and BETA particles or PHOTONS and possibly contained the keyword QUANTUM. The author was not English or American:

```
MODE   3 SPECOL 4.2
TYPA   HDTE (6305.  TO 6309.)
TYPB   SUBJ (PHYSICS) AND ASNM (@ IN.TON)
AND    NOT ANTL (UK OR US)
AND    KWRD (QUANTUM) OR GIST ₵
         (?ALPHA OR BETA OR PHOTON)
PNTA   HDTE/HTLE
PNTB   7/ANAM/ATWN
AND    SUBJ
AND    GIST
```

*OUTPUT*

```
630615  NUCLEAR PHYSICS
        CA  JAMES P PARKINGTON       CORK
        CS1 WAVES, PARTICLES OR WAVICLES
        CG1 REVIEW OF RUTHERFORD'S
              EARLY FINDINGS
        CG2 ON ALPHA AND BETA PARTICLES
```

Summer 1963 is defined as (6305. TO 6309.). INGTON or INKTON are covered by the dot feature and a Word-End search. The Query (?) denotes Search all positions of the field.

**Advantages to be gained in SPECOL type working**

It is not possible in a single article of this length to describe all the features either at present in SPECOL or shortly to be incorporated. What it is hoped the paper will do is stimulate more interest in this field and illustrate at the same time the easy way in which information in a computer can be addressed and manipulated directly by the layman.

The fundamental idea of processing information in terms of sets and subsets of data will it is hoped gain some ground. Just as the concept of co-ordinate Geometry with its $x$, $y$, $z$ axes did so much to help mathematics so should the concept of sets and subsets of data become an essential part of data processing.

The article will also it is hoped bring home more clearly some of the advantages of direct customer to computer access. Not least among these is the dispensing with the services of a middleman in getting a job started, and the tiresome negotiation this often entails. A customer now has only himself to blame if he gets the wrong answer. Other savings are just as important. Programs may now be written in a matter of minutes rather than weeks; professional programmers will be spared writing programs that are 80% the same as someone else's; operations staff will save time avoiding the huge sorts and lists they are now so often called on to do; and customers will save time by not having to plough through large lists and prints which are normally the output from the general or 'canned' type of program.

Another advantage is the big step that the language takes towards on-line working. Conversational programming would seem to be just around the corner. Languages of this type will moreover not come in or be accepted overnight. They will develop and grow, and SPECOL has been designed with this very much in mind.

Perhaps the most fundamental advantage of SPECOL is the ease and simplicity with which it may be used by the ordinary person. Although the language itself is supported by most rigorous Set Theory proofs and of necessity follows well defined laws, it has still been possible to produce a language that contains only a few conventions and esoteric symbols. The language has remarkably few parameter or format constraints and the syntax, if indeed such a term is warranted to describe its detailed rules of use, can not be said to differ much from that of ordinary language. That this can be demonstrated now augurs well for the future of multi-access and remote enquiry systems on which business and management are coming more and more to depend.

**The Computer Journal**

Published Quarterly by

The British Computer Society, 23 Dorset Square, LONDON, N.W.1, England.

*The Computer Journal* is registered at Stationers' Hall, London (certificate No. 24690, June 1966). The contents may not be reproduced, either wholly or in part, without permission.

Subscription price per volume £4 10s. (U.S. $11.00 at $2.40). Single Copies £1 5s. (U.S. $3.00 at $2.40).

All inquiries should be sent to the Secretary at the above address.

*Communications:* Papers submitted for publication should be sent to E. N. Mutch, The University Mathematical Laboratory, Corn Exchange Street, Cambridge. Intending authors should first apply for *Notes on the Submission of Papers*, as the onus of preparing papers in a form suitable for sending to press lies in the first place with the authors.