Article

Personal bibliographic systems (PBS) for the PC: a generic survey of features

Dirk Schoonbaert

Assistant Librarian, Prince Leopold Institute for Tropical Medicine, Nationalestraat 155, B-2000 Antwerpen, Belgium
E-mail: bib@itg.be

Abstract: *Personal bibliographic systems (PBS) manage the input, storage, retrieval and output of bibliographic references, allowing for a number of different document types to accommodate for journal articles, books, book chapters, dissertations, reports, unpublished documents, etc. Unlike general purpose database management packages they are made to measure for bibliographic information, featuring amongst other things a variety of import profiles for records downloaded from the major commercial databases and automatic generation of dozens of different output styles, including those used by the most popular journals. This paper concentrates on low-cost mainstream bibliographic software for personal use or smaller libraries. Full-scale integrated library automation systems and online public access catalogue (OPAC) software, which specialise in finding a specific title within a vast amount of book-type references, are not discussed. This paper does not review or compare individual products but focuses on the various characteristics which, when available, make a PBS attractive. Special attention is paid to structural flexibility, retrieval options, input, display and output formats, and interface- and management-related issues. A product-independent table of desirable features is included as an appendix.*

1. Introduction

In the last decade, personal bibliographic systems (PBS) have established themselves as a popular and highly prolific PC software category, both in personal environments and in smaller libraries. PBS manage the input, storage, retrieval and more or less flexible output of a limited number of bibliographic references (say up to 50 000 records) allowing for a number of different document types to accommodate for journal articles, books, book chapters, dissertations, reports, unpublished documents and so on. Although several PBS have become obsolete by now, many others are flowering in their n^{th} version. Stigleman (1994) lists about 50 different PBS, excluding dozens of discontinued programs. Full-size integrated (turnkey) library automation systems and online public access catalogue (OPAC) software, which specialises in finding a specific title within a vast amount of book-type references, will not be discussed in this paper.

PBS vary strongly in concept: some are rather primitive, based on general purpose database management systems such as dBase IV which allow for the basic necessities of storage, retrieval and output of records but are not well suited to deal with the specific properties of bibliographic information. For example, they only admit fixed length fields or a fixed number of occurrences for each field. Other PBS are far more evolved and present highly sophisticated features such as automatic report generation, featuring dozens of different output styles, seamless integration of bibliographic references with wordprocessing programs and so on.

Although they are all basically designed for bibliographic information, several are by now also offering adequate full-text retrieval, for example using sophisticated proximity operators. In the more traditional approach, terms or sets can be combined using the Boolean operators (AND, OR, NOT) but many more recently developed systems (or program versions) use alternative procedures, often based on inventive artificial intelligence algorithms. Sieverts *et al.* (1991a, 1991b; 1992a, 1992b, 1992c; 1993; 1994) call the whole range 'Information Storage and Retrieval (ISR)' software and distinguish between 'classical retrieval systems', 'end-user software', 'indexing programs', 'full-text retrieval programs', 'personal information managers', 'hypertext programs' and 'relevance ranking programs'. Furthermore, individual products may belong to several of these categories at the same time. It is clear that each package has its own strong and weak points: with every PBS I have tested so far, I have been impressed by certain features but always found it a pity that other

"Therefore it is certainly an advantage if you are allowed to change both individual field characteristics and document type structures"

aspects were not dealt with as ingeniously as in some rival systems. It is obvious that none of them has all the aces. I therefore fully agree with Sieverts *et al.* (1994) when they conclude:

'To end, we must once again emphasise that there is no single ISR package that:

- can be learnt in depth by anyone in a few hours, and
- can import an endless amount of textual and bibliographic data, of any length and in any format, and
- permits you to perform fast, extensive, flexible and precise searches in this limitless amount, and
- can output data in any detailed form or style you want, and
- is suited to be used for any type of retrieval application you can think of.

But, fortunately, our experience is that many ISR packages will go quite far on some of these points. You have to know yourself and your needs, or those of your users, to decide about your priorities, and then to try and find what's best for you!'

I shall discuss in a generic way a number of characteristics which individual PBS either do or do not feature. Most of them are available in at least one of the following low-cost (< £500) mainstream systems with which I am familiar: *BiB/SEARCH*, *CDS-ISIS*, *Pro-Cite* and *Reference Manager*, which all belong to the rather traditional 'classical retrieval systems' and 'end-user software' categories (Sieverts *et al.* 1992a). I encountered some additional features in packages which are not strictly PBS (for example some CDROM interfaces) but which would do credit to any PBS system. As both the advantages and the disadvantages of individual features are not always immediately obvious, this discussion may prove helpful when shopping for a PBS (or selecting CDROM interfaces; see also Válas (1994)). I shall start with the very heart of these systems, i.e. the structural possibilities and limits of the database file itself, as these define more than any other issue whether a PBS is adequate for the user's basic needs. Further on the actual database functions such as input, retrieval and output will be dealt with. Finally, a number of interface- and management-related issues will be discussed. A product-independent table of desirable features is included as an appendix. For a discussion of the major PBS themselves I refer to a number of excellent review papers (Sieverts *et al.* 1991a, 1991b; 1992a, 1992b, 1992c; 1993; 1994; Stigleman 1992, 1993, 1994, 1996).

2. Database limitations and structural flexibility

2.1. Capacity

Not all PBS can accommodate an unlimited amount of information. There are limitations to the number of databases that can be maintained, as well as to the number of records, fields per record, characters per field or total amount of bytes per record or per database. To what extent such limitations constitute a problem depends on what exactly you want to achieve with a PBS. This goal determines what capacity items you must look for. If you want to keep a number of autonomous databases, it's not much use buying a PBS that only allows for one database, whatever its otherwise breathtaking features. If you intend to catalogue over 40 000 books, a software package that allows no more than 32 000 records just won't do. Conversely, there's no need to look for a PBS with virtually unlimited capacity when you want to catalogue only your personal reprint collection, which may never surpass 5000 items.

2.2. Document types

When dealing with bibliographic references, it is important that the system can handle highly structured data. This does not imply that it is not possible to find relevant information in non-structured texts: this is often done admirably by highly sophisticated full-text storage and retrieval software. But in order to retrieve or reproduce specific bibliographic elements, it is important that these different pieces of information are kept in separate (sub)fields, grouped in adequate document types.

Most PBS offer a number of standard document types such as journal article, book, book chapter, dissertation and so on. Some include quite a variety of standard formats including video tapes, electronic files and other non-book materials. If you feel these answer your needs perfectly there's not much point in looking for further structural flexibility. However, you may feel the need to modify these formats or add extra ones. This may not be that important for a personal reprint database, where you can put divergent materials within the constraints of existing formats. The field labels or output formats may not be fully representative of the data included but for personal purposes they will do. In a library situation, however, it may be necessary to change these formats. If you want to incorporate a great variety of materials in your databases, for example including unpublished documents or annual reports, it's no good

if your system only allows for the standard book and journal article formats. Even if an abundant array of formats are available, it may be necessary say to create new fields or adjust some field tags. Therefore it is certainly an advantage if you are allowed to change both individual field characteristics and document type structures.

In the most flexible PBS practically all fields and record types can be defined freely, which allows you to accommodate totally different kinds of information (even including non-bibliographic data). For example, I have been using the same PBS to maintain some 20 bibliographic databases but also video catalogues, musical LP/CD and track databases, interlibrary loan administration, transaction logging information for database usage, and even expert system-like databases (for example a tropical medicine database, using different field tags for each specific type of information such as disease name, aetiology, epidemiology, disease vector, pathology and treatment). Of course, this is not what you buy a PBS for in the first place but if it fulfils your bibliographic needs, it is certainly an asset if the same system can also deal with such other media materials or managerial information. Of course, flexibility of structure can only be fully enjoyed if it is matched by an equal flexibility of display and output formats. It doesn't make much sense to rename or create fields when they cannot be displayed or printed in an adequate order.

2.3. Fields

Unlike general purpose databases dealing with postal addresses or numerical data using fixed field lengths, bibliographic databases should provide for information with highly variable sizes. Fields can be just a few bytes long, in which case it would be a waste to reserve a fixed number of bytes for each potential occurrence of each field for each record in the database. On the other hand, field contents such as corporate sources or abstracts can occasionally get quite long. In most systems there is a limit to field lengths but this limit can be several thousand bytes so it poses no practical problems. Even though megabytes are getting less expensive every day it is obvious that variable field length is an important feature for bibliographic data, both as a prevention of space waste (economy) and as a virtual absence of limits, allowing a high degree of accuracy (for example full corporate source names and serial titles).

Not all systems are equally user-friendly in exploiting their field-structures. Many use short alphabetic tags of one or two characters. These are often mnemonic so it is easy to remember that the title field is coded with 't' or 'TI'. This makes it easy to do field-specific searches. This is far less evident when fields are identified with numerical tags (cf. MARC codes), which are not so easy to remember or guess for experienced users, let alone for novice users. When working with fields and field tags there should be an easy way to get a survey of these fields and their major characteristics (for example via pop-up or drag-down lists). Furthermore, this basic field indication may be complemented by the interface, for instance by providing full field names for display purposes automatically (for example 't' is transformed into 'Title').

Some systems allow for individual subfields. This can be useful to give initials more autonomy or to work with more than one publication date (for example one for retrieval purposes, another for actual output); or, conversely, to put related information (for example imprint) within one and the same field. In most instances, however, this can equally be resolved by using separate fields for each individual type of information.

2.4. Characters

For bibliographic purposes it is important that all necessary characters are adequately supported by the system. Next to the standard alphanumeric ASCII characters there is also need for the extended ASCII set, including the French accented characters and other diacritics. In the early days this was a problem in many Anglo-Saxon PBS but at present most support the extended ASCII set (256 characters), so the most prevalent foreign texts can be accommodated for. In the future the more exotic characters (for example Japanese) may also be included if current 8-bit character sets are ever replaced by 16-bit sets.

Another advantageous feature is the capacity to foreground or highlight certain parts of the text, so that these strings can be displayed or printed in bold, italic, underlined, super- and subscript, independent from the actual display or output formats or the type of field to which they belong.

2.5. Record numbering and linking

It may be helpful to have an explicit permanent number for each record so it can still be identified uniquely when intermediate records are deleted or the database is rearranged. Useful options include the automatic generation of such numbers and the possibility to modify single record numbers or renumber a complete database. Setting the automatic number increments to a higher value than one allows you later to reclaim the unused intermediate numbers to place newer records in the vicinity of specific older ones, thus manipulating, say, the browsing order. Automatic generation of the date of record creation and last modification may

> *"Of course, flexibility of structure can only be fully enjoyed if it is matched by an equal flexibility of display and output formats"*

> *"It is important that all necessary characters are adequately supported by the system"*

also be useful.

Some systems allow the linking of separate records. In this way, pointers to hierarchically related records can be included so a lot of bibliographic information of the parent record need not be duplicated within each individual child record, thus guaranteeing economy and consistency. This is generally possible in the relational database category but can also be found in other PBS types.

In the last few years multimedia capabilities have gained in popularity. Ever more PBS can now handle images, either through limited links to external files and viewing software or by full integration, including internal viewing software. Managing sound files appears to be less common in PBS. The capability to interpret and produce barcodes can be useful in library environments.

3. Database selection

This is generally the first option you are presented with when starting the system. A good PBS makes it easy to access the right database. Therefore it is important that all available databases are listed in an adequate way. Most systems display this choice via a menu: the database of choice can then be selected by typing either a number or a highlighted character from the name, or by manipulating the cursor keys or clicking on a menu item with the mouse. Having access to alternative database-directories is also helpful. In this way a hierarchical structure can be maintained in which each type of database has its specific subdirectory, so related databases are displayed within the same menu and other, irrelevant ones are excluded. This may also be feasible within one default database directory but then the system should offer ways to customise the presentation of databases, i.e. in another order than the alphabetical or chronological default. Another advantage is the ability to define a default database that is opened automatically at system start-up. These database selection issues may sound rather farfetched but some popular systems don't even list the available databases, so you can't open a database when you don't know its exact name and location. This can only be helpful if you want to avoid others peeking into your databases.

4. Entering new records (input)

4.1. Manual input

Typing references into a database is a tedious job. All help to lighten this chore, avoid unnecessary duplication of effort and minimise spelling mistakes is more than welcome. Generally, manual input is achieved using an electronic input sheet, offering a number of fields to be filled in. Preferably, only those fields that are relevant to the specific document type are presented. Having different input sheets for different document types (one of which should be the system default) avoids your being confronted with a lot of irrelevant fields during manual input. Also, it should be easy to customise the records you have entered using this default input sheet so that less common fields can still be added during manual input (provided they are defined in the field tables).

Input modules generally make use of some kind of text-editor. These differ greatly in versatility and power. When using a simple line editor a line can no longer be modified once the next line is reached. Full screen editors are far more flexible and allow jumping from one line to other lines above or below. Good editors can make life much easier. A common example is the capability to replace or copy strings or full field contents to other fields automatically within the same record (copy, cut and paste). A comparable issue is the creation of new records using duplication of existing records, for example for articles in the same journal issue, chapters in the same book, books in the same series and so on. Another useful feature is the option to enter default values for specific fields (for example journal or book title) or add field occurrences automatically (for example specific keywords) to all new records created during an input session. It is not always easy to enter foreign characters or diacritics when you don't know their ASCII value by heart. Some programs have special modules which can remap your keyboard (for example using <Alt> or <Ctrl> combinations) or display pop-up lists with foreign characters.

Index-assisted input, where the strings typed are compared to information that is already available within the indexes, has two major advantages. Firstly, it guarantees alphabetic consistency because the indexes act as authority files. Secondly, if automatic truncation is incorporated, this can shorten the amount of typing dramatically. For example, you may only need to enter '`tr r t m`' to get the rather intricate string 'Transactions of the Royal Society of Tropical Medicine and Hygiene' in your journal title field. It gets even better if the index extract also includes the number of actual hits, thus showing the comparative popularity of each suggested item. This is helpful when choosing between several alphabetically related terms. Yet it should equally be possible to overrule this feature or neutralise it temporarily, otherwise no new index terms can be added. Input specification tables, defining what type of data (alphabetic, numeric, alphanumeric, etc.) is allowed in what field, may optimise data integrity but decrease flexibility.

4.2. Importing external records

People don't always start their databases from scratch. Ever

more they get their records from external databases. If they use a printout as a basis then they have to enter these records manually. If however these records are already available in an electronic format, either through downloading or via scanning coupled with OCR, manual input can be avoided provided the PBS has adequate provisions to import such files. It is obvious that import files should conform to the internal structure of the PBS. The more complex this structure, the more difficult it is to handle acceptable input files. Many systems provide specially made profiles to convert records downloaded from specific sources such as popular online or CDROM databases. It is generally far more difficult to prepare files which are not in a format already provided for by the PBS producer. As far as data structure is concerned, some PBS are quite severe while others are more forgiving. Evidently, the more lenient a system is, the less adequately structured the imported records may turn out to be.

Next to the PBS-specific conversion modules there exist a number of generic conversion programs, such as Fangorn or Headform, which can convert practically any structured data into a popular standard format. Of course, even with the best conversion programs it will not always be possible to translate records fully from one PBS or database host to the other. For one thing, the respective interpretations of certain fields or document types may not be 100% compatible (for example article authors vs. book authors vs. book editors). Even using popular standards as an intermediate format may not always be the answer. The Medline format, for instance, is quite useless when dealing with non-journal article records. To limit the damage it is a good thing if during import unrecognised fields or unauthorised field contents are placed in a separate field (for example notes) so that these contents can afterwards be salvaged manually. The same principle holds for rejected records, which ideally should be stored in a separate file. Just as with manual input, it may be helpful if default field contents (for example availability indicators or additional keywords) can be generated during automated import. Some PBS offer automatic generation of keywords from the title or abstract field (optionally coupled to a stoplist).

Automatic duplication detection is a time-saving feature. This is achieved either during import, comparing each individual new record of the uploaded file to the records already present in the database, or at a later time, acting on the whole database as a batch procedure.

Uploading records from other databases produced with the same PBS can be considered a special case of electronic importing. On a record per record basis this will generally not cause serious problems, yet there may be mutually incompatible field or document type definitions between these two databases. Merging complete databases is a more complex matter as it may be necessary to avoid duplicate records, keep original record numbers or maintain a specific ranging order. Creating new databases may involve defining all record structures and parameters from scratch, but generally this is greatly facilitated by copying existing formats. Deleting records can also be important for database integrity, though this does not necessarily free the disk space occupied by the deleted records. The more versatile PBS keep track of deleted records and offer an 'undelete' command to salvage erroneously deleted records.

5. Modifying records (edit)

Many helpful features for creating records are equally relevant when editing them: line vs. screen editor; using only relevant (i.e. non-empty) fields vs. the complete datasheet; various copy, cut and paste capabilities and so on. Other important features are the maximum number of records that can be modified simultaneously using global editing (i.e. when a vast number of records or even the entire database are changed in one go) and the corresponding qualitative capacities, such as full-text modifications vs. only within specific fields; case-specific vs. case-independent; and so on. Some systems allow many types of external (and often very powerful) editors to be used instead of the standard one included with the system.

6. Searching the database (retrieval)

6.1. Indexes

Indexes are essential to guarantee fast retrieval. Single term searches should give almost instantaneous results. More complex searches will give different response times in different index-based PBS. These can of course be compared with each other but no definition exists of what exactly is meant by fast. Evidently, the type of PC (processor, clock speed, RAM) used is a highly influential factor. A disadvantage of using indexes is that they may take up a lot of disk space (for example 50% or more compared to the database file itself — there are vast differences between individual systems).

Some PBS generate one general index, including all relevant fields, while others have field-specific indexes so you can for example limit the search to title words or keywords, discarding less relevant information from for example abstracts or author addresses. Often, when field-specific indexes are supported, these are limited in number or cannot be defined by the user. Field-specific indexes are not only an advantage when searching but also when the input module allows some kind of authority control (for example for index-assisted manual entry of new records). Being able to view part of the field-specific index (preferably showing the number of hits for each item) is essential for quality

"Many helpful features for creating records are equally relevant when editing them"

"An essential refinement is the ability to limit the search to specific fields"

control of your databases. They make it feasible to track down typos and alphabetically related notations systematically. Some systems can also generate field-specific lists in descending order of frequency, so you can easily spot the most popular authors, keywords or journals in a database.

Not all fields are indexed in the same way. Most PBS offer more than one indexing technique, making a distinction between, amongst others, 'text' and 'non-text' fields. 'Text' fields, such as title or abstract, do not constitute a standard entity so each word is indexed individually. In 'non-text' fields the full contents are indexed as a coherent unit, so standard combinations such as journal or series titles or corporate authors are represented as one index entry. In this way the composite search terms will be found in one step and need not be retrieved by combining their constituent parts. Some systems have a limited maximum length for index entries (even when the corresponding field contents have no such limits). For 'text' fields this is generally not a handicap but 'non-text' index entries often need several dozens of characters to identify them uniquely (for example long journal or series titles, including subsections). Customisable stoplists can keep less meaningful words out of the indexes. It's even better when separate stoplists for each specific database or database type can be maintained. A third mode of indexing avoids irrelevant terms by only selecting strings marked explicitly from within the field. Other characteristics (repeatable vs. non-repeatable fields; alphanumeric vs. numeric data; decimal vs. integer ranging of numericals; author vs. non-author fields; etc.) can also influence the way fields are indexed or sorted within the index.

Indexes are generally maintained using one of two basic methods: 'real-time' updating brings the indexes up-to-date immediately after modifying the database file so they are fully reliable at any time. The disadvantage is that this can take some time, so that each time a record is created or changed the database cannot be accessed for a short while. This may take only a few seconds but when you are used to millisecond PC performance this will easily start to be a nuisance.

The second method consists of batch updating: the indexes are not updated until the user decides it is time to do so. No time is wasted while creating or editing records. The disadvantages are that the modified information is not incorporated in the index, so the system should provide a way to overcome this setback. Some programs offer sequential searching of that part of the database which differs from the index, i.e. the new or modified records. Updating the indexes may not always consist of just adding a small amount of information to the existing index but may necessitate the generation of a completely new index. This can take quite some time, which implies that the database must be indexed when it is not being used (for example overnight). Some PBS offer a choice between both immediate index updating and batch updating.

The alternative to using indexes is sequential full-text retrieval, in which the system searches the actual database file character by character. It is obvious that for a large database this is not the optimal mode of searching. As the actual database file is searched instead of the selective index files, sequential searching may in some cases get more accurate results but unless the database is rather small, response times will generally become prohibitively long.

As both types of retrieval have their advantages and disadvantages, the ideal is of course a combination of both: using indexes for default fast retrieval and sequential searching for information that is normally not indexed or has not yet been reindexed.

6.2. Modes of searching

Retrieval is basically a dialogue between the user and the PBS: you propose words or strings and the PBS tells you how often (and preferably in which fields) these words or strings are to be found within the database. Apart from the sequential vs. indexed retrieval issue, there is also a big difference in the way in which the various possibilities are presented or can be activated: using formal commands has the advantage that you can indicate precisely what you want and often get it in one step. The disadvantage, of course, is that you first must master the command language and know the available operators, field tags and so on. This is generally not the case with menu-based systems, which offer you a choice between carefully explained alternatives at each step. The disadvantage is that they cannot always offer the same power and speed as formal commands. Because of their step-by-step approach, they can become tedious for experienced users. The choice between both types is often a matter of personal preference. When more than one person uses the same PBS (for example both experienced staff and novice patrons in a library) the best solution is a system offering a choice between both possibilities.

Whether command- or menu-oriented, most traditional systems keep a kind of survey of requests in which each search formulation is executed on the complete database and the results constitute an autonomous set. In this way you can return to previously executed searches without the need to reactivate them. A comparable feature is backtracking, which allows you to go back one or several steps to previous screens, commands or menus. Records marked during display should be kept as autonomous sets which can afterwards be combined with other sets. It can also be useful to drop specific sets from the current session survey in order to get a clearer view of the search strategy used so far or to save active RAM memory. A good PBS should also be able

to save specific sets permanently so that they can be recalled in later sessions. This should not only hold for sets of records but also for sets of commands or search histories, so the same combination of commands can be used in any session. This is essential for SDI services (selective dissemination of information).
Other systems are fundamentally limiting: they assume that each new operation is meant to limit the current set. This is fine as long as you want to narrow your search but does not permit widening it or returning to previous steps. In this case the option to 'select all records' or 'select none' is essential to start new searches. For the rest of this section I shall deal only with the 'autonomous sets' based category.

6.3. Refining retrieval

Retrieval options can be rather primitive but a plethora of extra possibilities exist: some rather evident, others pretty ingenious. A basic requirement, however, is the capability to combine individual search terms and previously defined sets (for example using the Boolean operators AND, OR, NOT) or to limit them (for example to specific languages or publication years). This may involve quite intricate combinations with various sets of nested parentheses. A low maximum search expression length can be a basic limitation for building complex searches.

Another essential refinement is the ability to limit the search to specific fields. A basic necessity is a readily available survey of relevant field tags. Consultation of field-specific indexes showing the number of actual postings for each index entry is helpful and points out alphabetically related items, including their comparative popularity. Just seeing this information is helpful but being able to activate search terms directly from these index extracts is still better. Some systems allow you to choose just one index term, others allow several, either activating each index term as a separate search set or combining them in an OR relation, or doing both. These facilities can be optional or the automatic system default, or a combination of both. In the best case default retrieval fields can be customised (i.e. which fields are searched and in what order).

As explained above, not all fields are necessarily indexed in the same way. It certainly is an advantage if for some fields you can choose between 'text' and 'non-text' indexing and retrieval. Depending on your preferences you can select records either by searching for the full field contents (for example 'American Journal of Tropical Medicine and Hygiene') which gives you the unambiguous result in one go, or by combining some of the constituent parts (for example 'American' and 'Tropical'), which is more helpful when you don't know the exact title. However adequate the indexes may be, optional sequential searching (of the whole database or a specific set) can be a valuable extra option, even if just for case-specific verification.

Using lots of different fields can be interesting to guarantee highly sophisticated output formats and ultraspecific retrieval (for example only first authors), but sometimes it is more interesting to combine several related fields. For example, when you're looking for all publications by a specific individual it is not important whether he's a first author, a secondary author or a book editor. Some PBS offer the possibility to combine several fields automatically under one field group tag (for example `'author: author1 + author2 + author3'`).

Proximity operators are more narrowing than the Boolean AND relation: they demand that two words must appear in the same field, the same sentence or a specified (exact or maximum) number of words apart, in either specified or indiscriminate order. Comparative searching allows you to select records by specifying that the contents of a field are equal to, bigger or smaller than a certain value (for example publication date). Interval searching is especially valid for numerical data (values, years, for example `'PY=1985-1990'`) but can also be useful for alphabetical data such as a range of author names: for example `'AU=smith-smythers'` yields all authors, starting from the first 'Smith' up to the last 'Smythers'.

A special retrieval technique is the use of lateral searching: while viewing records, specific words (for example title- or keywords) can be marked which are then posted as new search terms. A more elegant option is when you can jump directly to the other hyperlinked records in which this search term features. In this way you can navigate around the database. This is a very attractive feature when you want to look at, for instance, all available titles within a specific series, when this series was not even a search term in the first place. However, if this cannot be combined with traditional retrieval techniques, it will not be easy for example to select all French language book chapters authored by a specific individual and published between 1980 and 1990.

6.4. Alphabet-related devices

It is obvious that while searching for specific words or strings, a number of alphabetically related items will be missed. This can be overcome by index browsing, which alerts you to the closest alternatives. An easier way is implicit truncation. Most systems offer a way to activate (or de-activate) right-hand truncation: for example `'immun*'` yields 'immunity', 'immunodeficiency', 'immunology' and so on. Left-hand truncation is less common and, if available, often only on a 'non-text' field basis: for example `'*diseases'` yields 'infectious diseases', 'sexually trans-

"Using lots of different fields can be interesting to guarantee highly sophisticated output formats and ultra-specific retrieval"

"Hits should be highlighted so it is immediately clear why a record has been selected"

mitted diseases' and so on. An extremely powerful variant of automatic truncation of 'non-text' fields is referred to as 'embedded wildcards'. Only the first letter(s) of some of the constituents are needed to find (amongst others) the correct item: for example '`j=j e m`' (in which '`j=`' stands for the journal title field tag) finds 'journal of electron microscopy', 'journal of emergency medicine' and 'journal of experimental medicine'. Full left truncation, in which '`*ology`' yields 'epidemiology', 'immunology' and 'parasitology' is quite rare in index-based systems. In sequential searches, however, this is no more difficult than right-hand truncation. A third possibility is internal truncation or masking, where one or more characters within a word are replaced by a wildcard: for example '`h*matology`' yields both 'hematology' and 'haematology'. Truncation or masking symbols can replace either an exact number or just any number of characters, ranging from zero to a dozen or more.

Some retrieval systems distinguish between upper and lower case, others do not. The ideal, of course, is that you can choose whether to search in a case-specific way or not. Just as it is interesting to combine several fields to one field tag, it is also helpful if you can specify a number of characters (including diacritics and foreign characters) to be searched together by default (yet search them individually when necessary): for example '`a`' yields 'a,à,á,ä,â,å' and the respective uppercase characters. Likewise, recognising a variety of date forms (for instance '5/7/96' vs. 05-07-1996) as being equivalent but also interpreting full month or season names enhances retrieval results.

6.5. Subject-related devices

Just as field indexes or other authority files (for example journal abbreviation lists) alert for alphabetically related terms, a thesaurus shows related subject terms (synonyms, related terms, higher level terms, lower level terms). These may function as optional suggestions or as an obligatory vocabulary police, rejecting all terms not defined in the thesaurus. Some powerful thesauri can 'explode' — activate all lower level terms automatically when choosing a higher level term. For example, using the explicit search term '`Africa`' may yield but a small portion of the relevant results: however, using the explode feature recall is much higher because records including 'Nigeria', 'Rwanda', 'Zimbabwe' and so on are also selected, even though the word 'Africa' itself is not present in these records. If an explicit thesaurus is lacking, the explode function can sometimes be emulated by defining a set of trigger terms: these activate a number of self-specified search terms automatically. These can be maintained using for example a synonym list or a number of individually saved search strategies. Synonym lists need not necessarily include meaningful alternatives. They may include soundalikes and lookalikes: words that sound like the target term but may differ considerably when written down, and vice versa. Some PBS allow for more than one level of keywords: each primary keyword can be combined with a number of secondary keywords acting like subheadings.

6.6. Sophisticated retrieval techniques

The retrieval techniques discussed so far are relatively straightforward and logical. In order to guarantee useful results, the search terms should conform to certain conditions, for example authors should conform alphabetically to an author index and keywords should conform to a hierarchic thesaurus or synonym list. Certain highly sophisticated techniques try to overcome these one-dimensional limitations in such a way that natural language can be adequately interpreted so the search expression should not be entered with the rigid Boolean operators in mind. These techniques are often based on artificial intelligence, using relevance ranking or fuzzy set searching. This may imply special computer algorithms that combine the search terms automatically in an OR relation and count actual postings and compare relative positions. Others go one step further and find out which other words often appear in close proximity with the actual search terms, and suggest or include these as extra search terms. Soundalikes and lookalikes may also be activated on an artificial intelligence basis instead of an explicit (customisable) list.

These less common techniques can be quite useful when dealing with unstructured texts. When searching for structured bibliographic records, however, it is not unwise to keep Boolean operators and field-specific indexes as a basis. This basis may then be supplemented with these extra features. The ones I have encountered so far (in CDROM retrieval software only) do not always lead to fully orthodox results.

A final retrieval issue is multiple database searching. Some of the big online hosts offer the possibility to access several databases at the same time, thus reducing the need to search individual databases consecutively. They also have routines to trace duplicates and remove them from the hit selection. Of course, this is only as good as the degree to which these databases resemble each other and use the same field indications and/or keyword systems. So far I have not yet met this feature in PC-based PBS but it is already included in some CDROM packages.

7. Display

The elegance and attraction of a PBS depends to a great extent on the ways the database can be browsed and the search results displayed. The least flexible PBS offer one fixed format per record type, showing all fields included in the data definition tables whether or not they are actually used in the record and always in the same sequence. This 'electronic formula' can be improved upon in several ways: first of all, empty fields should not unnecessarily take up screen space and push some of the more relevant fields to a following screen. Having the choice between different presentation formats is another advantage and the options should be indicated clearly. Toggling between several formats is an advantage, especially if the options include a customisable shortlist, showing up to 25 records per screen, each reduced to one line. A sophisticated display option is the combination of both full record details and a shortlist, using split screens. Ideally you can define which fields are displayed and in what order. This default can then be overridden at any time.

While viewing, a number of scrolling possibilities should be allowed: go to the top or the bottom of a record (especially when more than one screen per record is involved); go to the next or the previous record; or go to the beginning or the end of the current set. Hits should be highlighted so it is immediately clear why a record has been selected. In some systems this highlighting is deactivated when previous sets are combined and the search terms themselves are not used explicitly in the final search expression. During record display, selection and deselection of specific items should be possible.

Some PBS can only display the most recently created set, so in order to view a previous set the corresponding search needs redoing. The option of displaying just any set *ad hoc*, using mouse, cursor or set number to select them, is certainly more elegant. Another aspect is the sequence in which the records are displayed. Most systems tend to display the records in the order in which they are stored physically in the database. Commonly, records can be sorted in a number of ways before they are printed or downloaded. This is not equally obvious when displaying them. However, this is an agreeable option and adds to clarity if records can be viewed in a certain order, for example alphabetical (by author, or any other meaningful alphanumerical field) or in reverse chronological order (for example by publication date). Some systems offer several ways of arranging records before display but they are based on a limited sortkey which only creates an approximate order. Ranging can be quite refined, though, including several consecutive sort levels (for example year of publication, then authors, then title).

8. Printing and downloading (output)

8.1. Record selection

A first question to be addressed is that of record selection: do you need to print or download a complete search set or can you specify a subset using either fixed record numbers or relative positions within that set, or indicate ranges (idem) or select records during display (marked records)? Limitation of the number of records that can be printed or downloaded can be a nuisance but if they can be defined and customised by the systems manager, such limits can be advantageous for keeping public access output under control. Compared to printing, downloading has a few extra parameters to take into account. How much freedom do you get to specify output station, path and filename (including extensions)? When using the name of an already existing file, does the system alert you and allow the choice between appending the new records to the old file, overwriting the old file or renaming the new file?

8.2. Record formatting

One of the essential advantages of putting bibliographic information in a number of separate fields is that you can reconstruct these records in many different ways. The advantage is obvious when one considers the many divergent bibliographic styles that are being used by different scientific journals. With a good PBS, reformatting a complete bibliography is generally a matter of less than a minute. The most popular formats (for example *ANSI*, *Harvard*, *Vancouver* and *Science*) should be provided with the program. If you are allowed to customise these formats or create new ones then the possibilities are virtually unlimited, and if you can construct new document types freely this is an absolute must in order to be able to export them in a meaningful way.

For both the largely inflexible and the fully customisable systems there exist a whole range of levels of detail: some allow certain fields to be printed without allowing any change to the field contents or their respective order. Others allow fields to be selected in any chosen order or allow conditional relations between fields (for example 'if A then B, else C'). This can result in highly sophisticated nesting of instructions. Some PBS allow formatting within individual fields (for example how many words or characters for each field). This is especially important for adequate author for-

"One of the essential advantages of putting bibliographic information in a number of separate fields is that you can reconstruct these records in many different ways"

"A good PBS should have clear and easily accessible help functions"

matting or shortlists (for example one line for each reference, restricting the length of each of the fields included to fit into columns). Some PBS go so far as to select what type of characters or how many of them should be printed: for example only upper case letters, only numericals, change all lower case letters to upper case, or remove all interpunction. Some formatting issues such as absolute positioning use autonomous instead of relative values (for example start printing field 'a' at 'position 25').

Some fields tend to inspire more attention than others. Special provisions for author fields include inversion of initials, removal or inclusion of blanks or periods, using '&' before the last author, conditional selection of the number of authors reproduced (for example when more than six authors, print only the first three plus '*et al.*'), and so on. Some PBS feature special modules which manage a number of alternatives for each journal name, for example full name, *Index Medicus* abbreviation, other abbreviations (for example including periods), codes and so on. In this way the appropriate format can be used no matter which form is actually used in the record's journal field. Comparable routines are also offered for date fields, thus providing for instance different ranges and formats for year, month and day values.

8.3. Bibliography formatting

Other issues concern the overall outlook of the formatted bibliography, such as ranging the references: sorting on several levels (freely choosing the fields involved), combining different sort directions (for example ascending vs. descending: 'A -> Z' for one field and 'Z -> A' for another; using customisable stoplists to ignore leading articles) and page formatting (defining margins, columns (including word wrapping), indenting and numbering references (starting from 1, or using the fixed record numbers), providing customisable bibliography headings and so on). Again, foreign characters and diacritics can cause trouble if they are not specially provided for. In some PBS you can define a sort order integrating both standard and special characters. As with display, it can be useful to highlight the hit terms but this is generally more easy when printing than when downloading

Of course, the more options available, the more confusing the choice becomes. When all formatting instructions are integrated within the same output style each variant calls for a separate combination so the number of different styles quickly becomes confusing. Some PBS group the relevant parameters and output formats in a number of subsequent levels (for example basic record format; additional features; sorting order; page format; physical output device driver; and so on), so each level offers a limited amount of alternatives, but as each item can be combined with each item from each other level, so a few possibilities on each level lead to a vast array of different output formats. No matter in what fashion they are presented, the worst case scenario — which actually exists in one of the most popular PBS — is when you don't choose an output format because you can't know if any exist, as there is no listing of formats and you can only use them when you have learned their exact names (and positions) by heart.

Output does not always imply printing or downloading lists of bibliographic records. If index surveys can easily be produced then generating subject or author lists, featuring record numbers or full bibliographic descriptions linked with each keyword or author, becomes child's play. More essential is the possibility of exporting a complete database in a generally accepted interchange format (for example comma delimited), thus allowing the transfer of the database to another, newer and more versatile PBS.

A last feature related to output is the capability of some systems to combine the database with wordprocessing programs: using pointers in the text file, the system selects the corresponding references in the database automatically and formats them into adequate bibliographies for the manuscript, in a style appropriate for the specific journal to which the manuscript is submitted. There are different approaches to this kind of integration but even when these capabilities are rather poor, using the clipboard or task switching features of Windows will get you quite far (Stigleman 1996).

9. Interface- and management-related issues

9.1. DOS vs. Windows

Many PBS for the PC started as DOS-based software and have by now been upgraded to Windows versions (Stigleman 1996). In Windows applications, the general layout and many of the functions tend to resemble each other (for example using <Alt> menus; help and exit procedures). Windows features fast manipulation and activation of functions using mouse and buttons, resizing the work space and using different fonts for screen display. Whether you prefer these items to straightforward DOS-based applications (using cursor and function keys) is more a matter of personal taste than a true revolution welcomed by all. A real advantage of Windows over DOS is the ability to leave a Windows application temporarily and meanwhile execute other jobs without the need to exit the PBS. In this way one can speak of a modest form of multi-tasking. As disadvantages, one could say that Windows screens tend to be overloaded and multiple options are not always self-explanatory. Functions are sometimes hidden behind other menu items, necessitating an annoying succession of menu levels — if you find them at all. Next to DOS and

Windows, Macintosh programs or hybrids are also common. When using different platforms it is a great advantage when all PBS versions can use the same database file without the need for conversion. The same is true for subsequent software releases (i.e. different versions).

9.2. Modes, language and help

A preference for either menu-based or command-based modes is often a matter of taste. Having a choice between several modes is certainly an asset when dealing with a mixed public: expert users can use a no-nonsense command mode which gives them results quickly and accurately while novice users can select a menu-based interface (often more limited) which carefully explains the options at each step.

If your patrons represent various nationalities, a multilingual interface is quite an asset. Again, these come in various formats: the more sophisticated systems offer a choice between several dialogue languages, featuring a full interface for each of these languages. Others come in separate versions, one for each language. Still others are basically unilingual but offer the possibility to include multilingual help messages. In this case, it is a boon if you can edit these messages by yourself.

A good PBS should have clear and easily accessible help functions. Either these should be present on the screen continuously in shorthand form or some shortcut indication should remain visible on the screen. In Windows interfaces this tends to be standardised while in DOS-based PBS all kinds of different help modules exist.

Two other options can be viewed as extensions of the help functions. Indication of the progress being made during time-consuming actions (for example while searching or downloading) can reassure users that the system has not blocked, or make them decide to cancel the operation — if this option is available.

Being able to visit DOS without leaving the active database can also be a helpful feature. In this way (if the basic DOS memory allows it) one can activate functions which are not available in the PBS software itself, for example by using simple DOS batch files to activate multilingual help-messages. In Windows interfaces, this is of course a standard possibility. It is also an advantage if your system is based on one of the more popular programming languages, as built-in routines are often available or can be built by the user.

From an ergonomic viewpoint, being able to change the screen colours (both fore- and background) is not a superfluous luxury, especially if you have to spend long hours accessing your databases or when hit terms or otherwise foregrounded words or fields are highlighted and you have only a monochrome display unit.

9.3. Safety and security

Some systems are sensitive to specific strings or keyboard combinations. In the best case the retrieval session is closed. In the worst case an automatic deinstallation program is activated. Prompting for confirmation when partially deleting sets of records or search histories, changing databases or leaving the system can prevent accidental loss of the results of considerable efforts. Unexpected power failure generally does not corrupt the datafile: if it does, installing an uninterrupted power supply (UPS) may be the only safe solution.

Password protection can be important if more than one user has access to the system or you want to prevent others from tampering with your personal databases or circumvent a dedicated PBS installation and roam freely around your PC or network, possibly changing parameters or deleting whole databases or systems. Even with PC-based PBS password protection can be quite sophisticated, including various levels of authorisation. It is quite feasible that although many people use the same database, certain information (for example specific fields) may only be accessed by a specific public, part of which is also allowed to enter new records or change existing ones. Still other functions, such as creating or deleting entire databases, should be accessible to the system administrators only.

9.4. Feedback

The automatic generation of usage statistics is also an interesting feature in multi-user environments such as libraries. In this way you get to know what category of patrons consulted what databases; when (date, hour); for how long; and how many searches were performed or records displayed, printed and downloaded for each. This is not to suggest that a relatively simple PBS should in this respect do a better job than some full-scale integrated library systems featuring highly sophisticated transaction logging modules, but it certainly is an advantage if your low-cost PBS can do something workable in this direction.

9.5. Networking

Obviously, not all PC-based PBS are network products. In the best case, the system is fully network-compatible (preferably at no substantial extra cost), including full file- and record locking: several users can then access the same

> *"Potential users should define their own list of preferences and priorities and then select the real life PBS that offers the best compromise"*

database simultaneously but editing specific records is limited automatically to one user at a time, to prevent data corruption. Other systems only work in standalone situations or provide networking facilities with non-DOS versions only (for example UNIX). In practice, however, many single-user systems can be used in PC networks. Conflicts and data loss can be prevented by installing the DOS 'Share' program but this implies that only one user at a time can access a specific database. Also, this may constitute copyright violation (some programs allow this type of networking, many others explicitly do not) and offers no guarantee whatsoever that the system will not crash unexpectedly. So if you want to offer your databases to more than one user simultaneously, choosing a PBS with full networking capabilities is a must.

10. Conclusion

Personal bibliographic systems (PBS) constitute a highly prolific PC software, using an approach basically different from general purpose DBMS and traditional electronic library catalogues. Within their own niche, PBS come in all sorts of varieties and differ strongly in concept. The various individual features reviewed in this article together offer an enormous potential of attractive characteristics. When looking at individual systems, however, it is obvious that none has all the aces. Moreover, the usefulness of the separate features will be evaluated differently by individual users, either for personal or for professional (for example in a library) reasons. Instead of advertising one specific PBS, I believe it is far more useful to point out the major pros and cons (or even the very existence) of the various features that may be encountered in this type of software. Potential users should define their own list of preferences and priorities and then select the real life PBS that offers the best compromise. Such systems are reviewed adequately in the surveys included in the bibliography.

There are three final aspects that one should bear in mind when shopping for a PBS:

(1) is your PC powerful enough to exploit the possibilities of the system fully (and if not, do scaled-down versions exist, which are amongst other things less hungry for memory and disk space)? Cheap read-only versions may be very useful when you want to distribute your databases offering recipients full search and output capabilities.

(2) is the provider sufficiently reliable and supporting? How long has the company and its system been in this branch of the software business (cf. program version number)? Are demo versions available? How useful are the printed or electronic manuals? What kind of support (nearby representatives? toll-free telephone number?) or training can you get at what price?

(3) can you afford it? Is the package worth the investment, both in money and in time and effort (learning, customising, etc.), compared to others? Simpler and cheaper systems might fulfil your needs just as well — but then again, they might not.

Acknowledgements

I thank my colleagues Gilbert Roelants and Veerle Demedts for their useful comments on the qualities of several PBS and for their critical reading of the manuscript.

References

SIEVERTS, E.G. and M. HOFSTEDE (1991a) Software for information storage and retrieval tested, evaluated and compared. Part 1: General introduction, *The Electronic Library*, **9**(3), 145-153.

SIEVERTS, E.G., M. HOFSTEDE, P.H. HAAK, P. NIEUWENHUYSEN, G.A.M. SCHEEPSMA, L. VEEGER and G.C. VIS (1991b) Software for information storage and retrieval tested, evaluated and compared. Part 2: Classical retrieval systems, *The Electronic Library*, **9**(6), 301-316.

SIEVERTS, E.G., J. FIGDOR, S. BAKKER and M. HOFSTEDE (1992a) Software for information storage and retrieval tested, evaluated and compared. Part 3: End-user software, *The Electronic Library*, **10**(1), 5-18.

SIEVERTS, E.G., M. HOFSTEDE, B. OUDE GROENIGER (1992b) Software for information storage and retrieval tested, evaluated and compared. Part 4: Indexing and full-text retrieval programs, *The Electronic Library*, **10**(4), 195-207.

SIEVERTS E.G., M. HOFSTEDE, G. LOBBESTAEL, B. OUDE GROENIGER, F. PROVOST and P. SIKOVA (1992c) Software for information storage and retrieval tested, evaluated and compared. Part 5: Personal information managers, hypertext and relevance ranking programs, *The Electronic Library*, **10**(6), 339-355.

SIEVERTS, E.G., M. HOFSTEDE, A. NIEUWLAND, C. GROENEVELD, B. DE ZWART (1993) Software for information storage and retrieval tested, evaluated and compared. Part 6: Various additional programs, *The Electronic Library*, **11**(2), 73-89.

SIEVERTS, E.G. and M. HOFSTEDE (1994) Software for information storage and retrieval tested, evaluated and compared. Part 7: What to choose, or the purpose of it all, *The Electronic Library*, 12(1), 21-27.

STIGLEMAN, S. (1992) Bibliographic formatting software: a buying guide, *Database*, **15**(1), 15-27.

STIGLEMAN, S. (1993) Bibliographic formatting software: an update, *Database*, **16**(1), 24-37.

STIGLEMAN, S. (1994) Bibliographic formatting software: an updated buying guide for 1994, *Database*, **17**(6), 53-65.

STIGLEMAN, S. (1996) Bibliography programs do Windows, *Database*, **19**(2), 57-66.

VÁLAS, G. (1994) Comparison of some widespread CDROM information retrieval software packages, *Online & CDROM Review*, **18**(4), 211-226..

Appendix : a generic survey of PBS features

The desirability and (dis)advantages of each of these features is discussed in the main text.

Structural flexibility

- Maximum number of databases in the system
- Maximum number of records per database
- Maximum number of bytes per record
- Structured fields vs. unstructured text
- Maximum number of different document types
- Free definition of document types (all vs. some vs. specific)
- Maximum number of fields per record
- Free field definition (all vs. some vs. specific)
- Maximum field length (idem)
- Variable field length (idem)
- Repeatable fields (idem):
- limited vs. unlimited occurrences
- Subfields (free vs. fixed; all vs. some vs. specific)
- Mnemonic field names and tags (idem)
- Free field names and tags (idem)
- Full ASCII set, incl. diacritics
- Highlighting: bold, italic, underline, superscript, subscript
- Record (identification) number (free vs. fixed)
- Renumber records (individual vs. all); change renumbering increments
- Automatic generation of dates (creation; last modification)
- Linking records (e.g. parent-child)
- Incorporate multimedia: images, sound (links to external files vs. fully integrated)
- Incorporate barcodes

Database selection

- Database menu
- Fixed vs. customisable database presentation (e.g. order of databases)
- Easy database selection from menu
- Gateway to alternative directories
- Define default database

Input

- Electronic input sheets
  - — fixed vs. customisable
  - — system-wide vs. document type specific
- Line editor vs. full-text editor
- Accents & diacritics module
- Copy, cut & paste capabilities
- Duplication of field contents (all vs. some vs. specific fields)
- Define default field contents (idem)
- Duplication of existing records
- Index-assisted entry (all vs. some vs. specific fields)
- Authority control (idem; automatic vs. optional)
- Input specification tables
- Internal conversion profiles (fixed vs. customisable)
- Automatic duplicate detection
- Automatic alert for new field contents (e.g. index terms)
- Merging databases (keeping original record numbers)
- Create new databases
- Delete individual records or sets
- Undelete deleted records

Editing

- Line editor vs. screen editor
- Internal editor vs. free choice of external editor
- Copy, cut & paste capabilities
- Global editing:
  - — Maximum number of records per session
  - — Field-specific vs. full-text
  - — Case-specific vs. case-independent

Indexes

- Real-time indexing vs. batch indexing
- Individual indexes for all fields vs. basic index
- Individual indexes for some fields (fixed vs. free)
- Maximum index term length
- Individual words ('text') vs. full field contents ('non-text') vs. selective indexing
- Numeric vs. alphanumeric indexing
- Integer vs. decimal ranging of numericals
- Indexing of numericals: decimal vs. integer ranging
- Support personal name fields (author formatting)
- Stoplists
  - — fixed vs. customisable
  - — system-wide vs. database-specific
- Alphabetical index survey available (per field)
- Frequency index survey available (per field)

Retrieval

- Sequential vs. indexed searching as default
- Sequential searching as additional option
- Response times for single searches
- Response times for combined searches
- Autonomous sets vs. limiting retrieval
- Individual set creation
- Search history survey
- Backtracking
- Marked records as individual sets
- Older sets can be re-used
- Older sets can be dropped
- Save sets
- Save search formulations

- Survey of saved sets and search formulations
- Boolean operators
  - and
  - or
  - not
  - others
- Combine several operators (use nested parentheses)
- Maximum length of search formulation
- Proximity searching
  - within the same field
  - within the same paragraph
  - a maximal number of words apart
  - a specific number of words apart
- Comparative searching
  - equals
  - is bigger than (or equals)
  - is smaller than (or equals)
- Interval searching
  - numerical
  - alphabetical
- Field-specific retrieval
- Retrieval using index-extracts
- Idem, incl. display of the number of hits
- Idem: number of index-terms that can be selected simultaneously
- Truncation right (exact vs. minimal)
- Truncation left (idem)
- Masking (idem)
- Case-independent vs. exact match
- Combine several fields with one retrieval code
- Combine several diacritics with one retrieval character
- Lateral searching
- Navigation ('hypertext')
- Structured keywords (several levels)
- Hierarchic thesaurus
- Explode capabilities
- Trigger terms (fixed vs. customisable)
- Synonyms list (fixed vs. customisable)
- Natural language interpretation
- Relevance ranking
- Soundalikes or lookalikes
- Multiple database searching (incl. automatic deduplication)

Display

- Shortlist vs. full record display
  - separate screens
  - split screens
  - toggle
- Scrollable record display
- Display only non-empty fields
- Field selection (fixed vs. customisable)
- Field order (fixed vs. customisable)
- Change fields or sort order *ad hoc*
- Range records for display
- Use different fonts, WYSIWYG, etc.
- Foregrounding (bold, underlined, italics, etc.)
- Highlight hit terms
- Mark hits for (de)selection

Printing and downloading

- Selection of records to be printed or downloaded
  - complete set
  - records marked during display
  - specific (record)numbers
  - range of (record)numbers
- Printing & downloading limits (fixed vs. customisable)
- Free choice of downloading station, path and filename
- Append to vs. overwrite existing files
- Different output formats (fixed vs. customisable)
- Select
  - fields
  - number of words
  - number of characters
  - type of characters
  - include (fixed) record number
- Specific field formatting features
  - author formatting (e.g. invert initials, (conditional) selection of author numbers, etc.)
  - journal formatting (e.g. full title vs. abbreviations; blanks vs. periods, etc.)
  - date formatting
- Conditional instructions (if ... then ... else ...)
- Nesting of instructions
- Ranging records (free choice of fields vs. fixed; one or more levels)
- Ranging diacritics (fixed vs. customisable)
- Numbering references (independent vs. fixed record number)
- Indention of records (fixed vs. customisable)
- Positioning of characters
  - horizontal vs. vertical
  - absolute vs. relative
- Page formatting
  - margins (horizontal, vertical)
  - columns, wrapping
  - heading fixed vs. customisable
- Easy selection of available formats and parameters
- (Multiple) device drivers (screen vs. printer type vs. ASCII file vs. wordprocessing file, etc.)
- Output to general interchange format
- Automatic bibliography generation from wordprocessing

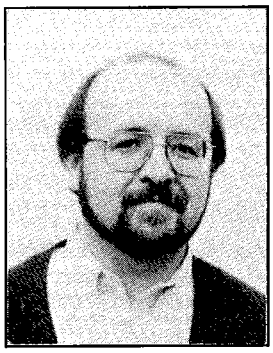
Interface related issues

- Operating system
  - DOS
  - Windows
  - Macintosh

— other
- Navigation devices
  - — cursor
  - — function keys
  - — (<Alt> or drop-down/pop-up) menus
  - — mouse
- Multilingual interface
  - — multiple language versions
  - — fully integrated
- (Multiple) mode(s)
  - — easy - menu
  - — expert - command mode
  - — query by form vs. intuitive navigation
- Help functions
  - — separate vs. integrated
  - — general vs. context-specific
  - — DOS escape
  - — compatible programming language
  - — display the progress of time-consuming actions (retrieval, printing, downloading)
  - — possibility to cancel time-consuming actions (retrieval, printing, downloading)
  - — customisable screen colours
- Security
  - — unconventional key combinations for delete or exit operations
  - — prompt for confirmation for delete or exit operations
  - — sensitivity to specific key combinations
  - — sensitivity to power failure
  - — password protection:
    - — at one level
    - — at several levels
- Usage feedback (statistics)
  - — date
  - — hour
  - — total time
  - — user(group)s
  - — database
  - — number of records retrieved
  - — number of records displayed
  - — number of records printed
  - — number of records downloaded
- Networking
  - — full networkability, incl. record-locking
  - — limited capacities (e.g. file-locking)
  - — strictly single-user

Miscellaneous

- Cost (software price vs. 'learning' time investment)
- Support (default vs. maintenance contract)
- Training (idem)
- Manuals
- Upgrading policy
- Scaled down versions
- Demo versions
- Cheap read-only version for file distribution



The author

Dirk Schoonbaert

Dirk Schoonbaert graduated from the University of Antwerp (Germanic Philology 1980, and Library and Documentation Science 1985) and has worked in several scientific libraries. Since 1986 he has been Assistant Librarian at the Prince Leopold Institute of Tropical Medicine in Antwerp. His professional interests include the electronic aspects of information management and retrieval.