Pybliographer — The Future

Frédéric Gobry and Peter Schulte-Stracke

August 11, 2002

This document introduces the next development steps for **Pybliographer**. It gives a task list as well as more detailed dicussions of related issues and problems.

Contents

To Do List

For the next release

The Graphical User Interface

- ∗ Refactoring the `Gnome/Index.py` code. [PTR]
  - Documenting the new API.
  - Providing for multiple index views; switching them.
  - ℘ Better data structures for widget interface – but see also future gtktreeview & associates.
- ℜ Incremental updates. [PTR]
- ℜ Combined author/title field in Index [PTR]
  - A better column title is needed.
  - The formatting should be adaptable.
  - The index must be sorted.
- ℜ Mark entries
  - Make subsets (lists or sequences) out of marked entreis; associate them with a name ...

Miscellaneous

- A progressbar (rather simple)
- ℘ A check for duplicates. (But this may need a lot of work and better fit into a revamped *Import* module.)

Bugfixes

- Stable Working Directory for open.

For Release 2.0

The Graphical User Interface

- Moving to GTK 2.

Generalities

- Moving to Unicode.
- Elementary database substructure.

1. Introduction and Generalities

1.1. The Way of Pybliographer

Highlights of Pybliographer Release 2

- The Index(-view) has been reworked, it now consumes less resources, starts and redraws much faster.
- The Import facilities have been completely rewritten with a view to providing a *import framework*, thus much reducing the expense involved in writing additional input filters, but also vastly (?) improving, at the same time, the level of support and comfort for every user.

1.2. An overview of user requirements

As a bibliographic database manager, *Pybliographer* places itself at the intersection of various expectations, traditions, and requirements, each of which developed originally independent, and often in ignorance of the other, and which are still shaping the field according to their own peculiarities, although they are getting into closer contact recently.

It should prove useful to start with a view of these ancestral developments:

In 1876 Charles A. CUTTER thus explained the aim of the catalogue:[1]

- To enable a person to find a book of which either
  - the author is known.
  - the title is known.
  - the subject is known.
- To show what the library has
  - by a given author.
  - in a given subject.

[1] as quoted in (1, p.196)

  - in a given kind of literature.

- To assist in a choice of a book
  - as to its edition (bibliographically).
  - as to its character (literary or topical).

These are still important principles, today perhaps even more than most of the time.

1.3. The legacy of the card catalogue – MARC[2]

Originally inventories for the use of the librarians only, since about one century, the library catalogues have become accessible to the public and henceforth set the standard of bibliographical description.

At about the same time began the development of rules for the bibliographical description,[3] to enable the sharing of catalogue entries between libraries[4] and to enable a catalogue to be build and used *compatibly* without annoyance by many people (which is a form of sharing, too).

The actual rules used have always been a compromise between precision and cost, oscillating between the ideal of capturing the whole title page (of course, only the title page[5]) and the need of coping (manually) with all this. Over time, the descriptions tended to grow in size. There are two reasons for this:

1. the more titles exist, the more it becomes difficult to distinguish between similar ones,

[2] Although MARC stands for Machine Readable Cataloguing, it is used here loosely but legitimately and stands for the whole of classical librarian's cataloguing, of which the actual MARC standard has been, in a way the epitome and culmination of centuries of work. Of course, it has begun to change ...

[3] The *Preußische Instruktionen* and the *ALA Rules* were among the first. Their successors are the *RAK* and the AACR2, both are in turn influenced (the RAK more so) by the international standardisation efforts of the *ISBD*.

[4] Either by one library providing the catalogue cards f or the majority of the libraries (in the US), or by building union catalogues for the purpose of sharing the holdings, too (the inter-library lending service in Germany).

[5] see Robert Musil's *Mann ohne Eigenschaften* and the talk between General Stumm von Borgwehr and the director of the Royal Library ...

2. more information is made centrally available, mostly to share the cost of providing it. Such was the case in Germany, where subject headings appeared in the national bibliographical database, thus moving something which has traditionally been the responsibility of the individual institution onwards to the *Deutsche Bibliothek*. To do so, additional categories needed to be defined, rules set up, an authority file maintained, as it was done earlier for the nucleus of the bibliographical description.

It should be noted, that traditionally only books have been catalogued, together with other physical objects, that might land in a library (MARC being quite comprehensive in its scope of materials allowed). That reflects the point of view of the librarian, who is the custodian of these said objects, buys, lends, ranges them. The contents are not his business. In that his view differs from the usual patron's view, as exemplified by the BibTEX and reference manager software (see 1.4 and ??). This orientation is slowly changing under the influence of Internet resources and integration with patron's software (section 1.8).[6]

Perhaps the most important heritage from this development is the idea of *entry points*: try to establish for everything that you catalogue attributes that are well-defined and precise enough and make the database search-able for them. For a card catalogue this implies defining not only under which *form* an author is filed, but in addition under which author a book is filed (if there is a choice). The first class of normalisations is done via *authority files* the importance of which, under the conditions of on-line catalogues, has only increased, because it is more difficult to find entries which differ slightly (1) while the so called question of the main entry has no longer the importance it used to have.

[6] MARC intended, indeed, to support so called *analytical* cataloguing as well, if only as a secondary task. Even the description of archival resources is not, or so one thought, out of its scope. This reflects the facts that American libraries hold archival collections to a far greater degree than European ones, and that these are far simpler than the classical European archival fonds; but remarkably this proved to be a failure – a separate standard (EAD) evolved, this time based on XML. Perhaps the reason was the absence of any form of hierarchy in MARC databases.

1.4. Taking the fast track – BibTeX

Very early the scholarly (and technical) document preparation has been supported with programs to augment document sources with correctly formatted references (*Refer* and *BibTeX* are two similar programs for the major document processing environments).

These are, in a way, the opposite of the MARC applications: the categories are more often than not only loosely defined, they are slanted towards the production of reference lists, almost at the exclusion of other uses, the formats are implicitly defined and easy to extend, and the defining program is restricted to the production of reference lists – the building and maintaining of its own data base, e.g., is left to other means.

In spite of (or perhaps because of) these deficiencies these have been very successful programs. Their restrictions, that follow a well proved tradition after all, were intended and adequate at the time of their creation. They have been addressed by the next class of applications which developed, characteristically, in the highly competitive marketplace of the *personal computer*, as opposed to the more traditional scientific Unix/workstation environment, that TEX and BibTEX (and earlier troff and Refer) thrived in.

1.5. Enters the Personal Computer – Reference Managers

With the advent of the Personal Computer a new class of users came to the fore. While the majority of the scientific/technical writers (who had little choice, anyway) stayed with their TEX applications, the new users were mostly attracted to the perceived simplicity of the word processors, and eschewed the command line and simple editors of Unix. Almost by necessity, the new applications stressed those aspects of the job, that the older tradition neglected; word processor users didn't care for the typographical refinements of TEX, nor the frugality and intellectual self restriction of the command line. They wanted simplicity and ease of use, ... and over the time, they succeeded. The graphical user interface comes no longer as an afterthought, it is at the centre of the development effort (often to the detriment of the application proper).

The features of a commercial product are given in appendix B.

For us it is important to pay attention to the *organising* of the references, by providing suitable instruments, e.g., the *virtual folders* of Biblioscape.

1.6. The spell of the Internet – Database & Document Access

Although the Internet precedes the personal Computer by some years,[7] the impact of the Internet was but slowly felt. We can distinguish the following uses:

Database access

Document access

Electronic documents posed new challenges to the librarian's profession. Ephemeral, they were often felt not to warrant the expense of traditional cataloguing (out of which considerations the *Dublin Core* standard evolved), hierarchical, they stretched the card-bound structure of MARC to the limits, ever changing, they challenged the concepts of bibliographical unit, of work, and edition (see 6 and 1.8).

1.7. The view from the fringes – exotic becomes normal

1.8. The standards evolve

The major development has been not the Dublin Core but the development of an object model that underlies the cataloguing process.

1.9. Other lists of requirements

The following is a list of user requirements as given in a talk in the technical university of Chemnitz[8]

[7] Of course, it all depends. Arguably the first personal computer was the Alto at *Xerox Parc* from the early 1970, while the switch to the *IP/TCP* protocol family did not occur but a couple of years later.

[8] `http://archiv.tu-chemnitz.de/pub/2001/0013/data/anforderungen.htm`

- Flexible access via various categories, viz. author, title, publication year, journal, and combinations thereof.
- Import and export in various formats, viz. BibTeX, Medline, Refer, RFC1807, [MARC, MAB, ...]
- Supports working groups, allows combinations of databases.
- Allows to merge data from various sources, and to normalise it (e.g., with respect to differing ways to abbreviate a journal title ...)
- Adapts to the various formatting requirements of different journals, easy development of new styles.
- Extensible by local fields, to hold notes, order information, annotations, excerpts, summaries, etc.
- Simple use during document preparation: allows to search the references from the editor/word processor (by author, title, e.g.), thus eschews the use of labels – the latter are to be used and provided only internally. $\mapsto$ 4.2
- Allow confidential data in shared databases (e.g., annotations of theses).
- N.B.: These requirements transcend BibTeX's abilities; thus it should be used only as an export format.

Other points that were mentioned on the mailing list:

- Pybliographer starts too slow; with larger databases one has no indication of progress, and must wait for a long time.
- Pybliographer holds the whole database in memory, and uses a lot of it! (For comparison: Allegro claims about 600 B (of secondary storage) per entry.)

1.9.1. The opportunity for a flexible bibliographic format.

The following excerpts from a recent googling(5) I'd like to add: The opportunity for a flexible bibliographic format – Designing a bibliographic format for highly varied levels of effort:

> It should be possible to create a bibliographic format which permits people to use exactly the level of detail they wish, rather than requiring or preventing less or more. This would reduce the current undesirable proliferation in bibliographic formats.

The problem: People create bibliographic records with enormously varying levels of effort. Further, the distribution of effort across different aspects of bibliographic information also varies. Current bibliographic formats tend to be useful only over a relatively narrow range. They establish minimums by requirements on information and its form. They establish maximums by limitations in the expressiveness of the format. They do not gracefully handle attempts to get by with less work, or to add more work to further refine the record. This is approximated by optional fields, but the format and meaning of a fields content is usually inflexible. Thus there is an undesirable proliferation of formats, a new one created whenever someone wants a new tradeoff in ease versus precision. The new format is generally as inflexible as those which preceded it, and the problem thus continues. [. . .]

The opportunity: Create a bibliographic format which can gracefully handle a wide range of cataloguing intensity. One which could handle the complete spectrum from Bib/Refer to MARC. And do it in a way which permits work to be spent only and exactly where one wishes. Such a format would be a 'lingua franca', as any other bibliographic format could be converted into it without losing, or requiring additional, information.[. . .]

1.10. Summary of Requirements

- The oldest stems from the librarian's duty to keep inventories of his collection in form of *catalogues* – and the associated application software.
- Very early the scholarly (and technical) document preparation has been supported with programs to augment document sources with correctly formatted references (*Refer* and *BibTeX* are two similar programs for the major document processing environments).
- These programs concerning themselves, however, only with this last step, other tools developed over time to help with organising and searching the databases these programs require. Such *reference managers* also aided in the odious task of actually creating (i.e., typing) the needed entries.
- With the growth of Internet usage, it became possible and more or less usual to access databases of bibliographical interest directly. Both the nature of the data (format, extent, correctness, completeness) and of the means of access (protocols, navigation, access restrictions, formatting) vary enormously. Nevertheless, it has become desirable to easily integrate the results of these searches into one database – adding a new complexity to the tasks.
- In addition, more and more the exchange of documents needs to be integrated, too.
- The World Wide Web may also serve as an imperfect but easily usable and ubiquitous way of providing *access* to a database.
- Finally, the traditional library catalogues have also added support for additional administrative functions (both on the acquisitions as on the lending side), many of which are quite natural extensions of their primary function.

2. The Graphical User Interface

In this chapter we discuss the user interface, in our case based on the Gtk 2.0 toolkit. First we give an overview of the user interface elements, then we look at the rewuirements of individual widgets.

See also: For the integration of Pybliographer into Word Processors and Browsers, see section 4.2; for using Pybliographer from the commandline or scripts, see chapter 7.

2.1. Major elements of the user interface

5.6

2.2. Displaying and Using Indices

The usual way of perusing a bibliography or a catalogue is by means of a listing, either according to the author's name, or according to the title, a keyword, or any other suitable piece of information.

With online catalogues, one usually has access to the underlying (technical) *indices*, if only to compensate for the limitations of usual search facilities and the missing overview possible with today's display devices.

It is best to have multiple indices (Indexviews), so as to make best use of the structure of the data (e.g., the *Person or Author Index* would usually show the works of the author upon selection (as a tree view) not the authority record of the author, a title index might in a similar way allow subordinate entries for editions or translations, etc.

Sometimes special indices are required. A manuscript collection might want to enter and find listed the first few words of a manuscript or letter, a specialist might need the names of translators, or scribes, or printers. So ther must be a degree of configurability in addition to that needed to cater for the differing tastes and traditions.

2.3. Accelerator Key Assignments

Accelerator keys allow easy selection of functions, in particular easy switching of views, which is in paricular desireable when doing data entry and editing. These are activities which repeatedly and predictably need a great

number of screens to proceed, switching them with the mouse is particularly unpleasant. Consitent assignment of *shortcuts* is needed.

3. Organising and Manipulating References

One aspect of Pybliographer is its use to *organise references* – that lays the accent onto the varieties of uses that these references could be put to, and the concomitant variety of possible organisation we have to cope with. This de-emphasises the bibliographical description, as a rule, but stresses our ability to annotate and link.

As a rule, our data is kept in one database, organised into *folders* (compare Biblioscape B.1).

A variety of *Indices* are kept to allow fast access. q

4. Searching, Selecting, and Formatting Data

A *search* is an operation that is performed against one or more *databases* – usually locally represented by Pybliographer and remotely by database servers known as *connections* – and that results uniformly in a *result set*.

As a rule, such a result set is if necessary *imported* as such into the Pybliographer database, and sooner or later *selected* from.

A *selection* is any subset of the database, usually the result of manually inspecting (*scanning*) some result set, that is presented to a processing option: either to be set apart for further processing at a later time, or to be processed by the output routines in order to be presented or exported. [XXX Are there other uses?]

Notes:

- Results sets can vary considerably in size. Efficiant means of handling them are important.
- Various situations are to be considered and equally suppported
  - One looks up price/holdings/availability information for an already known title. That should result in the new information being added to the existing entry (and perhaps the latter being a bit improved). Possible actions include posting a note to pick up the book at the next visit to the library, or to make a copy from a journal, or to order it from the bookseller or the ILL service.
  - One browses several data bases and cumulates the results. Later one reduces the set and adds it to the data base.
  - One adds, perhaps on a regular schedule, records from an external source. Evidently this situation calls for other mechanisms and tools than the previous one, in spite of its superficial similarity.
  - On builds a bibliography or similar documentation database – involves most of the above situations and adds requirements for stringent administrative control.

4.1. A generic query service

The exact possibilities that a database provides for searching are as manifold as are the data. Paradoxically this makes it more easy to provide one integrated query service – any attempt to equip every database and every service with its own query interface cannot but falter in view of the enormeous programming investment that would be needed.

It's not clear, perhaps, how the individual services can communicate their capabilities upwards towards the query service, but it should be evident that *some* communication would be helpful – even if one fully acknowledges that the assessment of a database service must usually include information that is not available to the program. (Example: cataloguing rules – RAK, AACR, ...)

4.2. Integration with editors, wordprocessors, and browsers

Integration with editors like Emacs, word processors like Abiword, and bowsers like Mozilla enhances ease of use, broadens its field of application and reduces user's keystrokes and thus the opportunity for errors on his side.

We distinguish the following use cases:

Adding a citation When writing, the user wants to add a reference to certain document, of which he might remember no more than some words from the title, or the author, or even less, but as a rule, not the exact citation key that BibTEX e.g., would use to identify it.

The user should refer to the document by selecting from a result set or shortlist.

Inserting citations Quite often the case arises that a document must be cited in, say, an email without wanting the whole (BibTEX) machinery to be put in action.

A simply formatted citation could be offered via *Drag and Drop*.

Annotating When reading, the user wants to add a note.

He is helped with creating and maintaining annotations to documents (registering them with Pybliographer); annotations can be large and are not suitable for storage *in* the bibliographical record, but a link could. On the other hand it is conceivable to point back from the annotation to the document, so it could stand on its own.

Viewing documents An electronic document could be requested.

Define appropiate viewers and requestors (mainly for eprints – the rest is more simple).

Extracting metadata An electronic document is perused and the wish is that it be catalogued.

A limited bibliographical description can be build from the metadata that is stored and communicated with the document. *Importantly:* If downloaded, the local file address should be stored as well.

4.2.1. Emacsen

4.2.2. Other editors

4.2.3. Mozilla

4.2.4. Other browsers

4.2.5. Kword

4.2.6. Abiword

4.2.7. Open Office

4.2.8. Other word processors

4.2.9. Other applications

4.3. Formatting references and bibliographies

4.4. Other applications

[Connecting to OPACS, etc.]

5. Interfacing to External Ressources

5.1. Connecting to external databases

Pybliographer uses *connections* to external ressources for the following purposes:

- To look up a catalogue (for immediate consumption, so to say), i.e., as a client program.
- To obtain, as a result of a query, additional data for import.
- To obtain additional documents that are referred to from the data base, i.e., via an *URL* or an *Eprints* link.
- To perform operations on the remote system, i.e., to request books or copies by means of an *OPAC*.
- ℘ To cooperate with other Pybliographers, sharing and updating data.

5.2. Writing Import Filters

Import filters are used not only when a *file* is imported, in a narrow sense, but also when opening a file, and, perhaps more importantly, when processing the results of a *query*.

An *Import Module* provides the needed support. Typically, there is one such module for every major data format, say MARC or BibTEX.

A major part of such a module is a Reader class, which subclasses (is derived from, and specialises) the class `ImportReader` in the module `Pyblio.Import`.

In fact, most Reader modules do not derive directly from `InportReader`, but from an intermediary, as `TaggedReader` for data structured as a sequence of tagged fields, like in the MARC format, `TextReader` for data that comes as 'unstructured' text, like taken from the references of an article, `XmlReader` for XML formatted text.

In this way, we have a *framework* for import modules, one that makes writing or modifying an import reader much more simple. This is important, because requirements and interface continue to evolve.

The format specific modules are not the end of the story, yet. For one, a format like MARC is in fact so vast that it will probably never be completely implemented and used, neither by a library, nor by Pybliographer. It is, however, to be expected that one time or the other, someone would be better off if he could make use of that exotic feature, and would be happy to add support. In addition, one may encounter a situation in which the standard processing is inadequate and better be modified. Thus it is useful to be able to have an easy way of adapting to one's very special needs, be it in terms of adding or of overriding standard behaviour, i.e., methods.

There is yet another point to be made. A framework is also a way of sharing the implementation effort. That means that adding code to the common base is more attractive and this in turn makes the use of Pybliographer more attractive. So everone profits.

5.2.1. Framework Services

The base and intermediate classes, the framework, provide the following services and features:

Input The source data is accepted by various means: files, iterators, in-core objects, and made available in a standardised way to the processing.

User interfacing The user interface, always requiring a big programming effort, is structured by the base classes (togerther, of course, with the Gui Component), thus sharing a lot of work.

Data handling In the past, the handling of the input data was almost completely left to the format specific code, resulting in an enormous duplication of code and often poor exploitation of the source data. With the new design, most of the semantic data handling will be done by functions in specialised modules, sharply distinguishing between the parsing (syntactic) and database aspects of the task.

Duplication checking Like the user interface, the duplication check is an example of a pragramming task, that is not easily undertaken within the confines of one particular application, but only if it can be done wholesale.

On- and offline correction It is highly desirable to be able to correct imported data in a systematic way, the framework will provide for that, both during inital processing, dubbed on-line, as well as at a later time.

Customisation There are ways to adapt the import processing which can be done easily in a generic way. It is, e.g., simple to exclude all data from certain categories (tags) from further processing and therefore `TaggedReader` provides a parameter to specify such tags – it can even be done interactively from an options dialogue. No programming needed.

Data management aspects The data imported will be kept logically identified as long as it is desired, without the need to keep separate files for it. That remedies a major problem with the work organisation in the past and it is necessary in any way for a data base oriented Pybliographer.

5.2.2. Class hierarchy

The fundamental class is `ImportReader` in Module `Import.py`. It implements an `Iterator` interface, viz. `first()` and `next()` methods.

```
class ImportReader (Iterator.Iterator):

    """Base class for all import reader classes.

    Support for input methods, encodings, database and GUI connections,
    """

### The following need no change as a rule:
    def first (self):
        return self.next()

    def next (self, entry=None, data=None):
        """Process the next entry from the input source.
        entry and data argument are provided for easy testing."""
```

```
        e = entry or self.next_entry()
        x = data or self.read_next()
        if self.options.has_key('preserve_input'):
            e.lines = x
        self.entry = e
        return self.parse(x)
```

Subclasses will need to provide, among others, implementations of `read_next` and `parse`.

A typical implementation of `parse` from `TaggedReader` follows:

```
    def parse (x):

        self.begin_record(x)

## it may be convenient, if the read_next routine already assembles
## continuation lines. Let's assume this

        for i in x:
            tag = i[0:self.tagcol]
            data = i[tagcol:]

            if discardrx and discardrx.match(tag):
                continue

            if _cache.has_key(tag):
                _cache[tag] (self, tag, data)
            else:
                methname = 'do_'+str(tag)

                if hasattr(self, methname):
                    method = getattr(self, methname)
                else :
                    method = self.do_tag

                method(self, tag, data)
```

```
            _cache[tag] = method

        self.end_record(x)
        return self.entry
```

So in this case, for a tag 'XYZ', a subclass implemented method 'do_XYZ (self, tag, data)' would be called, if it exists. If not, 'do_tag' would be called and as a rule, distinguish according to the tag, as directed by parameters of the class, whether to ignore, or to store (in a generic way) the data.

5.2.3. Class parameters

Control objects

control=None specifies the associated control object.

Input Three possibilities are provided for. The data parameter is particularly useful for testing. *One and only one must be selected from the following*

file=*filename* if input is from a file,

data=*data object* if input is from an incore object,

iter=*iterator* if an iterator is specified.

5.3. Writing Export Filters

5.4. Pybliographer as Web Server

5.5. Requesting remote operations

5.6. Requesting documents

Given an entry that refers an electronic document, it is an easy idea to consider requesting it by mouse-click, so to say.

6. Bibliographic Data: types, formats, schemata

6.1. Bibliographical Objects

See the IFLA report.

6.2. Persons

6.3. Works

6.4. Subjects

6.5. Database Organisation

7. Scripting and Configuration

8. Technical Questions

8.1. Architectural issues

8.2. Compatability issues

0016 Remain compatible with BibTeX

8.3. Charset and markup issues

0001 Establish Unicode internally and for storage

A. Programming Tasks

Some other things to do XXXX

- Sort order according to German standards
- Handling German *Umlaute* – ä ö ü –

B. Features of commercial products

B.1. Biblioscape

Organize references with folders, dynamic folders, etc.

Folder	Add references into folders. One reference can be put into multiple folders without creating a duplicate. Organize references into folders with drag and drop.
Dynamic folder	Organize and save queries into a tree structure. All references meeting the search criteria will be listed under a dynamic folder.
Indexed search	Return search results in a couple of seconds no matter how big the database is. One line search works the same way as most Internet search engines. Supports logical searches, fuzzy searches, etc.
Import filter	Bibliographic data from any data sources can be imported with a proper import filter. User can create new or edit existing import filters.
Output style	References can be displayed any any style like MLA, APA, etc. A large number of styles are provided for different journals. Users can also create new ones.
Cross linking	Link a reference to other references in the same database. You can define a relationship for the links, like "Supportive", "Contradict". You can also add comments for each link.
Navigation view	A reference can be displayed in an organizational chart where each node can lead to related records.
Formatted preview	Display a reference as formatted text according to the active output style.
Live preview	Display data fields of the selected reference in a grid without opening it. Changes made to the data will be saved to database as you move to another record.

Graphics and OLE	If a reference has associated graphics and OLE objects, add them to the Document field. The document field can be used to store the full text of a reference.
Field lookup	List all unique values along with the number of occurrences of a data field. All data fields with possible repeated values can be shown in lookup view. These include Author, Keyword, Publisher, Language, Country, Subject, etc.
Recycle bin	All deleted references are put into the Recycle bin. You can recover them from the Recycle bin or remove them permanently from the Recycle bin.
Advanced search	Query any data field with a visual query builder.
Find and Replace	Search for a word or phrase and limit the search to a data field or all data. The same is true for the Replace operation.
Sorting	Sort a column by clicking on the header. Click again and the column will sort in reverse order. User can also define multi-level sorting.
Filtering	Define a filtering criteria with a visual filter builder and apply the filter to any dataset.
Term list	Users can keep frequently used phrases in a term list. Terms can be organized in the list by category.
Move field	The content of a data field can be moved from one field to another.
Global edit	The content of a data field can be changed at once for all selected references.
Eliminate duplicates	Duplicate records can be found and removed. Fuzzy search is supported for finding duplicates.
Analyze references	Data fields in the reference table can be analyzed for data distribution.
SQL commands	Users who are familiar with SQL can query the database directly with SQL commands.

Report	A built-in database report writer will a print data report, including a subject bibliography grouped by keyword, author, year, subject, etc.

Format papers to generate citations and bibliography

Format a paper	Convert the temporary citations of a document into formatted citations and a bibliography.
Unformat a paper	Convert a Biblioscape formatted paper back to unformatted form (with temporary citations) so citations can be added or deleted before the final formatting.
Word support	Full integration with Microsoft Word, Biblioscape menus and toolbar can be added to the Word menu and toolbar system.
WordPerfect support	Full integration with Corel WordPerfect, Biblioscape menus and toolbar can be added to the WordPerfect menu and toolbar system.
Other word processors	Biblioscape methods for word processors integration are published and open to all word processors that support DDE.
HTML support	Biblioscape can generate formatted papers in HTML format. A hyperlink can be created automatically between an in-text citation and its reference in a bibliography.
Natural citation	Use words or phrases to uniquely identify a reference in a temporary citation instead of using a Reference ID. If references are moved into another database, temporary citations don't need to be changed.
Cite while you write	Use BiblioSidekick to display references in a small, always on top windows. While in a word processor like Word or WordPerfect, just drag and drop the selected reference in the place where you want to cite it.
BiblioWord	A full featured word processor inside Biblioscape. Just drag selected references from a panel on the right when you want to cite. BiblioWord supports live spelling check, thesaurus, tables, graphics, OLE, multi-level undo, etc.

Access the Internet to capture bibliographic data, Web pages

Remote databases — Access thousands of remote bibliographic databases on the Web with an integrated Web browser. These sites include university sites, commercial databases, and government sites. Most of them are free.

Capture references — Search web based bibliographic databases from inside Biblioscape, click a button to capture search results into a Biblioscape database with the right import filter. New import filters can be created by users.

Capture Web pages — Research on the Web with the Biblioscape integrated browser, capture a web page into a Biblioscape References table or Notes table. All words in the Web page will be indexed for future search. Graphics and links are captured along with the page.

Resources — A directory of bibliographic resources on the Web. Each entry listed has an associated import filter. The local Resources list can be expanded and edited by the user.

Web directory — Biblioscape Web site lists a collection of sites valuable to researchers. Web sites are organized by subject. Bibliographic databases are the main part of the listing. Although other types of Web resources are also listed.

Z39.50 — Most Z39.50 enabled bibliographic databases also have a Web interface, Biblioscape's integrated Web browser can be used to search such sites and capture search results directly into a database.

Link to a note — Easily create a link between a note and a Web site.

Take notes and link them to references, tasks, web sites, etc.

Tree structure — Organize notes in a tree structure. Note's position in the tree can be rearranged by drag and drop.

Indexed search — Find your note fast with indexed search. Each word in your Notes database is indexed for super fast search. The search words are colored in red on the hit page. Indexed search supports logical operators, wildcards, fuzzy search, etc.

Advanced search	Limit your search to a data field like Date Created, Keywords, etc. Build complex searches with a visual query builder.
Format text	The text in your note can be formatted with all the standard options, including fonts, color, background color, superscript, subscript, paragraph alignment, bullet list, number list, etc.
Link	Each note can be linked to other notes, references, tasks, catalog items, Web URLs, local files, etc. Double clicking on a link will take you to the linked item.
Web capture	Notes can be used to organize captured Web pages. All the graphics and hyperlinks of captured web pages can be properly displayed.
Table support	You can insert tables in your notes. Additional rows can be added and deleted.
Find and Replace	Standard Find and Replace tools for finding and replacing text in your notes.
Graphics and OLE	Graphics can be added to your notes. OLE is also supported. Therefore, you can add chemical structure drawings, spreadsheets, CAD drawings, etc. in your notes.
Table view	The notes can also be displayed in a table besides the default tree view. Notes can be sorted and grouped in a table.
Keyword lookup	Each note can have associated keywords. These keywords can be displayed in a lookup list along with its number of occurrences. Double clicking on a keyword will retrieve all related notes.
Spelling and thesaurus	A powerful spelling checker is included. Additional dictionaries can be downloaded for all major European languages. A thesaurus is also included to help the user to find the right words during writing.
Icons	Each note can be assigned a different icon to distinguish it from other notes.
Export	Each note can be exported to a file in RTF or HTML format.

Manage tasks and organize your research ToDo list

Sort tasks:	Click on the column header to sort tasks, click again to sort in reverse order.
Group tasks	Group tasks by Priority, Status, Date Created, etc.]
Task progress	Track the progress of a task by marking its percentage completed.
Task creation	Create tasks inside References module, and add selected references into the Description field of the new task.
Link to a note	Create a link between a selected task and a note.
Advanced search	Search tasks with a visual query builder.

Draw a chart to present your ideas

Flow chart	Draw a flow chart with an easy to use chart editor.
Knowledge map	Draw a chart and link a chart object to other modules. For example, double clicking on a chart object will open a group of references, tasks, notes, etc. A SQL query can be associated with each chart object. A knowledge map can be built with such associated queries.
Tree structure	Organize your charts in a tree structure. The position of each chart in the tree can be rearranged by drag and drop.
Link to a note	Create a link between a chart and a note.
Zoom Display	options like zoom in and zoom out, actual size, and fit to screen are supported.
Icon	Each chart can have an icon associated and displayed.
Shape and color	The shape and color of each chart object can be customized. The label text can be displayed in different fonts and colors.
Connectors	Chart objects can be connected with a flexible connector which can be curved. A connector can have its own label, font, color, size, different sources and destination arrows, and link points.

Manage a library without a steep learning curve

Catalog	Manage library collection data into 56 data fields, organized into several groups including Bibliographic, Holding, Request, Order, Serial, and General.
Serials	Manage serials and related activities including tracking, routing, etc.
Circulation history	Search, sort, and group circulation data. Display circulation activities by borrower, status, subject, etc.
Check Out	Check out books for library patrons, add notes, easily change due dates.
Check In	Check in books returned by borrowers. Automatically reminds librarian about Hold status.
Renew	Renew books for borrowers, add a note. Find renewed items by ID or title.
Hold	Put a hold on a checked out book. Show a reminder when that book is returned.
Interlibrary Loan	Manage interlibrary loan requests, track loan status, log shippings, etc.
Borrowers	Manage borrower's information (address, phone, fax, email, et]c.)
Lenders	Manage lender's information (contact's name, phone, fax, email, notes, etc.)
Suppliers	Manage supplier's information (address, phone, fax, email, notes, etc.)
Sort	Click on any column header to sort then click again to sort in reverse order.
Group	Group data by drag and drop. Data can be grouped at multi-levels by any data field.
Field chooser	Choose which data fields to include in the data grid by drag and drop.

Report and print — Build or customize data reports with a powerful report builder. Users can create new reports with a wizard. New reports can be easily added to the menu system. Reports can be previewed, printed, or saved as a files.

Web enable your bibliographic database with one click

Web publishing — Publish databases on the Web with BiblioWeb server. No other web server required. Runs on any Windows 95, 98, Me, NT4, 2000 machine.

Indexed search — Search references with a powerful search engine. Enter search commands like you do with a Web search engine. Supports search keywords AND, NOT, OR, LIKE, NEAR, Wildcards, etc.

Advanced search — Limit searches to certain fields. Build complex queries with up to 3 conditions.

Add references — Users with a Write account can add new references to the database using a web browser.

Edit and delete — Users can edit or delete their own references over the web.

Import — Import references over the Web with the right import filter, so you don't need to enter references one by one.

Hyperlinks — Search results are displayed with hyperlinks. Clicking on the hyperlink will trigger a new search for related items.

Style — Marked references can be displayed in any of the output styles that exist in Biblioscape.

Export — Marked references can be exported in several formats to be easily imported into other programs.

Format papers — Users can even format a paper over the Internet. Temporary citations in a document will be converted to formatted citations and bibliographies.

User forum — Includes a user forum application, so you can host a web based forum without extra cost.

References

[1] HALLER, Klaus: *Katalogkunde : Eine Einführung in die Formal- und Sacherschließung*. 3., erw. Aufl. München : Saur, 1998. – 91 avf 1332(3)+1

[2] HYLTON, Jeremy A.: *Identifying and Merging Related Bibliographical Records*. Cambridge, Mass., MIT, Diplomarbeit, Juni 1996. – MIT-LCS-TR-678

[3] IFLA STUDY GROUP ON THE FUNCTIONAL REQUIREMENTS FOR BIBLIOGRAPHIC RECORDS: Functional Requirements for Bibliographical Records : Final Report / International Federation of Library Associations and Institutions. 1998 (19). – UBCIM Publications New Series. – ISBN 3–598–11382–X

[4] MAXWELL, Robert L. ; MAXWELL, Margaret F.: *Maxwell's Handbook for AACR2 : Explaining and Illustrating the Anglo-American Cataloging Rules and the 1993 Amendments*. rev. Edition of Handbook for AACR2 / by Margaret Maxwell, 1989. Chicago : American Library Association, 1997. – ISBN 0–8389–0704–0

[5] MCHARITY. *The opportunity for a flexible bibliographic format*. 18 Dezember 1996

[6] SCHNEIDER, Wolfram: *Ein verteiltes Bibliotheks-Informationssystem auf Basis des Z39.50 Protokolls*, Technische Universität Berlin / Konrad-Zuse-Zentrum für Informationstechnik, Diplomarbeit, Juli 1999