Functionality profile

Axiell ALM Netherlands BV

Contents

Introduction

Axiell ALM Netherlands specializes in database management software for museums, libraries and archives.
In our products we make a distinction between core software, applications, databases and tools. In Adlib terms these may be described as follows:

- **Core software** basically consists of executables and dll's which implement all functionality shared by the different applications, like searching the databases, data entry in records, printing, and much more.
- **Applications** are collections of definition files, specifying database structures and application dependent interface elements like screens (data entry forms) and the lists of user-accessible data sources and access points and such.
- **Databases** are files containing records of related data entered by the user. Applications link to these databases, making them accessible to the core software and therefore practicably usable.
- **Tools** are utilitary software to aid in support and maintenance tasks like importing and exporting data, or to customize or build entire application and database structures.

The Adlib core software is regularly being updated with new functionality, and is freely available to customers with a maintenance contract.
Adlib model applications provide an extensive base for any museum, library or archive to fulfill most registration needs. However, you are free to adapt your application with tools like Adlib Designer, for instance to add or remove fields or to design new print formats.
So, the distinction between core software and applications allows the user to be able to update one's Adlib core software (with which new or improved functionality is introduced and bugs will have been fixed), whilst leaving the applications exactly as they are. On the other hand, it also allows you to customize your applications, and still profit from improvements in the core software.

To quickly inform you about the capabilities of Adlib software and associated files, this document currently provides a succinct overview of the internal structure of Adlib data files and the way data is managed and made accessible to applications; and a following chapter summarizes the most relevant Adlib core software (adlwin.exe) functionality visible to the user.

Note that the actual implementation, operation and interface of this functionality is described in the appropriate manuals and in the Adlib Designer Help.

1 Data structures and accessibility

1.1 Data files

Adlib software and applications accept databases in three different formats: the Adlib proprietary .CBF format, Adlib SQL Server format and Adlib Oracle format.

1.1.1 Adlib CBF

■ Databases

The Adlib proprietary .CBF file is a so-called flat file: a minimally structured collection of data records in Adlib format. Because it is a flat file, it doesn’t contain structural information.

An Adlib .CBF database is the top-most physical data container, which contains topically related data records. So, Adlib applications always use multiple database files, e.g. one to store person and institution name records in, one to store museum object records in, one to store book and serial bibliographical data in, etc. All these databases are linked to each other in multiple and various ways. This setup ensures that there is no data redundancy in the databases. For instance, a person’s name only needs to be registered once: all other databases in which this name if referred to, only a pointer (the linked record number) is saved, not the name data itself.

A database can be divided into virtual subfiles, known as datasets, defined by their maximum record number range. These are not separate physical files, only a subdivision of one database file. This allows the user to search subfiles separately, or to search a combination of subfiles. You can also screen off parts of a database from certain users by this division. Generally, datasets are used to group together records that logically belong together. For example, a library catalogue database can be divided into a books dataset, an articles dataset, a serials dataset, and an audio-visual materials dataset.

Each smallest piece of data is stored in so-called fields, in the database file. In the database, a field is identified by its unique tag, consisting of two characters.
Unlike (even minimally normalized) relational databases, Adlib databases allow for a variable number of field repetitions, so-called (field) occurrences. This makes it possible to register for instance, multiple authors of a book, multiple synonyms for object names, multiple checked in issue numbers of serials, etc. Each repetition will be stored in its own field instance. Repeatable fields occur often in Adlib model applications.

■ Indexes

Fields which are often searched on, need indexes to speed up searching. Adlib indexes are 2-dimensional tables composed of "pages"* which together hold either actual data (copied from a database for this purpose) or reference numbers to the original data in a linked database, both from one or more specific fields, plus the record number of the record in which the indexed value occurs, and possibly a so-called domain** value as well. An indexed value appears as many times in the index as it occurs in the relevant field(s) in records.
Each index is a separate file, and their contents is automatically kept up-to-date by the Adlib core software.

* The subdivision of a table into pages is only meant to increase searching speed and limit memory use; when using an index you can ignore this subdivision.

** Domains are used to more specifically describe a field value, usally field values from authority files; a name in a persons and institutions database, for example, may be a supplier name, an employee name, an author name, etc.: these name types are considered domains. Domains in indexes allow searching and input validation by name type or term type.

There are different types of indexes: term indexes contain whole, textual field values; integer indexes contain (record) numbers; free text indexes index long text fields and contain references to single words indexed in the *wordlist* index***; date indexes contain dates (in one of several formats), and logical indexes contain single characters (indicating TRUE) and empty spaces (indicating FALSE).

There are two special indexes:

- The first is *priref*, which stands for "primary reference". Adlib uses this index to list all the record numbers (= the primary references of each record) from your database together with the location of each record in the database file.
- *** Another special index is the *wordlist*. This index contains all words that "occur" in all free text indexes of all databases in an Adlib folder. Adlib uses this index to save memory when indexing words; thanks to this list, when compiling free text indexes, Adlib can suffice with referring to a serial number in the wordlist instead of including whole words. So only the wordlist index contains the actual words, the free text indexes refer only to the wordlist. This process takes place behind the scenes, so you don't need to take it into account when setting up free text indexes.

■ Supplementary data-related files

Other physical files which contain data-related information, are *.lck* (record lock) files which contain record numbers of current locked (already in edit mode) records, *.ptr* (pointer) files which contain record numbers of a user-selected and saved record seletion and/or Adlib search statements saved by the user, and *.cnt* (count) files contain the most recently assigned values relevant for autonumbering fields. Each file pertains to one database.

1.1.2 Data relations

A defining characteristic for databases is the way in which data relates to other data. Relevant aspects of Adlib databases are the following:

■ Linked fields

Linked fields are fields containing information from a different database. The Adlib core software retrieves the linked data from a specified database and places it in standard database fields. That means this data can be processed on detailed display screens in the same way as data in standard fields. If the retrieved data is displayed in fields that can be modified by the user, then Adlib can write the modified data back to the linked file if specified so.

The database in which Adlib must search for data is referred to as the linked file. The current database, to which data is linked, is referred to as the primary file.

So for a link there is a tag in the primary file, which will contain the key data (e.g. a person’s name) that a user fills in. This is called the linked field (even though it is located in the primary file). Adlib will use this key data to search the linked file for related or substitutional data, via an index.

The user may also be allowed to modify key data in the linked file, and have Adlib automatically adopt the modification for each reference in the primary file. For this, it must be known in the primary file in which record in the linked file the linked data is located. To this end, a so-called link reference field (a forward reference) has been defined for the linked field in the primary file, in which Adlib specifies the record number of the linked record. (If there is no link reference field, linked fields only hold key data, and are not automatically updated if the user changes that key data in the linked file: the linked information occurs both in the primary file and the linked file.) Once the link has been made, the user can edit the key data in the linked file and automatically "update" the linked field to this record, in all primary records.

To avoid data redundancy, most linked fields in Adlib databases are linked on reference.

■ Internal links

Internal links define hierarchical relations between terms in one and the same database, usually an authority file like the *Thesaurus* or *Persons and institutions*.
Internal links are typically used for searching and validating terms entered in linked fields in a primary database, making it possible to automatically have non-preferred terms substituted by preferred terms, to always use the right spelling when entering existing key-words, and to search for narrower, broader, and equivalent terms. In the authority files themselves the internal links are used to automatically create the correct reciprocal (mirrored) records when the user defines a hierarchical relation in one term record, and to validate new terms against earlier saved terms to make sure your authority files won't get corrupted.

The *Term* field in the *Thesaurus* and the *Name* field in the *Persons and institutions* database that hold the main keywords in authority records are not linked fields, but normal fields. All the other relational fields in such a database are linked fields, linked to the term or name field.

Instead of normal internal links, a database may have internal links on reference. This is necessary when terms or names in an authority file are not unique. An internal link on reference only stores the record number of the linked record in a link reference tag (the forward reference of an internally linked field), and fills the linked field with a value from the linked term or name record only when the user opens or uses an authority record with such a linked field, or exports it; but each term appears only once in the authority file, which saves you space on your hard disk.

■ Reverse links

Reverse links (also called backward references) are a general implementation of many-to-many relationships between records. Reverse links are implemented if databases that are linked to each other by means of linked fields with a link reference, need to update each other when in records linked terms are filled in, edited or removed, for instance when adding or removing a record, or when editing, adding or deleting a keyword in either database.
A reverse link is specified in both databases that are linked to each other. A reverse link can only be provided in a database if there exists a link reference tag for the field linked to in the other database, and if there is an index on that link reference field; and the tag provided for a reverse link in one database must be exactly the same as the link reference tag of the field linked to in the other database. So, a reverse

link points to the link reference tag of the field linked to, so that the record number of the record from which is linked is stored in (an occurrence of) the link reference tag of the linked record in the other database.

■ Feedback links

If changes in authority records, like making a preferred term in the *Thesaurus* a non-preferred term, or like deleting a name in *Persons and institutions*, need to automatically update all records in other databases that are linked to the concerning term or name, then feedback links are needed.
Feedback links are references in an authority file definition, to databases that link to this authority file. The difference with reverse links is that feedback links are used in files that have no linked fields to other databases. Typically, in Adlib applications this would apply to the *Thesaurus* and *Persons and institutions*: catalogues and other primary databases have linked fields to these authority files to validate the input of terms or names, but there are no fields in those authority files that are linked to the catalogues, because usually it is not needed to include data from catalogues in the authority records. In most existing Adlib applications, feedback links are not being used.

■ Domains

So-called domains are used to validate data by logical subsets within a thesaurus that physically only consists of one dataset. Such subsets are known as domains. The advantage of domains is that in one authority file (like a thesaurus) many types of terms can be stored, while terms are validated against specific domains.
In the authority file structure there is no association between a term or name and its domain: both fields merely occur in the same record. Only in an index on these terms this association can exist because of the domain field. In such an index, the term field value will be stored with its domain name as: `domain::key`.
A single term in a thesaurus or other authority file may be a member of multiple domains. For example, a person's name can be a writer, a customer, and a publisher. In the term index on this field, each entered `domain::key` combination will be present separately.

1.1.3 Adlib SQL Server and Adlib Oracle

Microsoft SQL Server and Oracle represent relational databases. A relational database consists of a collection of tables with in them usually a (very) limited number of field columns.
Field occurrences (field repetitions) that occur often in an Adlib .CBF file, have no similar equivalent in even a minimally normalized relational database.

The joining of Adlib and SQL technologies has led to the possibility to approach (Adlib specific) relational databases in standard Adlib software. After all, we want to keep using largely the same software, the same user interface and the same maintenance programs (like Adlib Designer), whilst we do want to offer our customers the advantages of SQL Server or Oracle but not the disadvantages.

Adlib SQL Server and Adlib Oracle databases are therefore always converted .CBF files (the Adlib proprietary database format), in which all original information and data is preserved. After that conversion, the Adlib databases will have been converted to as many tables in one relational database. Also, each index file, wordlist, pointer file, lock file and count file will have received its own table. For the conversion to the new format, pointer files are structured into XML, which allows for greater exchangeability, and better readability at table level. A whole data record in a .CBF file will have been inserted into one table cell as an XML document, accompanied by the record number in another column. This XML document is hexadecimally encoded though*, since in table cells (in SQL Server 2000) only printable characters may occur.
(* In upcoming versions of Adlib, for use with SQL Server 2005, these XML documents will no longer be hexadecimally encoded.) Note that with the storage as a hexadecimally encoded XML document the records won't be less quickly retrievable, because fast searching of records still takes place with the aid of indexes.
In an Adlib SQL Server or Oracle database, the name of a table indicates its contents. A single name like *collect*, is a former Adlib database. A double name, in the format *databasename_indexname*, indicates an individual index. If the second half of such a double name literally reads "*pointerfiles*", then it concerns a table in which the pointer files for the relevant SQL main table (ex-Adlib database) are stored.

So, in an SQL database the contents of a record are saved in one field. The record contents in this field are structured in XML, but may not be readable as such. The priref, the creation date and modification date are in separate columns, but are also attributes of the XML *<record>* element (the latter is convenient for a possible export of XML records), namely as follows:

```
<record priref="nnnn" creation="yyyy-mm-ddThh:mm:ss" modi-
fication="yyyy-mm-ddThh:mm:ss">

   <field tag="tt" occ="nn">data for field</field>
   <field>.....</field>
    ....

</record>
```

An index table only has two columns: *priref* and *term*. Per index key there is one row in such a table. The *term* field contains the key on which you can search (although the *term* field may contain linked record numbers).
Special tables are *recordlocks* and <*database table*>.pointerfiles:

- There is one *recordlocks* table for the entire SQL database. Each row in this table describes one record lock: the lock id, the name of the database table to which the lock applies, the record number of the locked record, and the time of locking.
- Per database table there is one *pointerfiles* table. Each pointer file in it is described in its own row of the table. The available columns are: *owner* (maker of the pointer file), *title*, *number*, *selection-statement* (the search statement or selection), *hitcount* (number of results), *modification* (date on which the pointer file result was updated last), and *data* (contains this pointer file as an XML document).

Because of the conversion of Adlib .CBF records to XML documents, the extendibility of XML has been utilized to allow Adlib database fields to become multi-lingual here. Underneath an XML field occurrence node, language nodes may occur to contain translated values for this field occurrence.

1.2 Structure files

Adlib proprietary application, screen and database structure files specify application-specific GUI's (graphical user-interfaces) and define the structure, links, internal procedures, and limits of the physical data files: they specify the characteristics of every single variable Adlib object, from individual fields, access points, databases, screen layout, user access rights, internal procedures, to available print formats and much more.
These files are readable only to Adlib core software*, and can be edited or created only in Adlib tools like Adlib Designer. See the Designer Help for complete documentation about all options and settings.

> * There *is* one way to generate complete XML documentation of all structure files though, via the *Documentation* options in Designer (and moving the relevant stylesheets in the installation first). This would allow third-party software to have read-access to all settings in the structure files.

Functionality that is not offered by the multitude of options in these structure files, can possibly be added by writing programs in Adlib's own programming language ADAPL. Such programs, called adapls, can be associated with other Adlib objects to perform data manipulation, validation and/or to create additional print formats. For more information, again see the Designer Help.

Application structures are specified in .PBK files, screens (data forms) in .fmt files, and database structures in .INF files. These structure files apply to all three database formats, and only a few settings in these files will need to change when preparing an application for Adlib SQL Server or Adlib Oracle databases; structure files always remain physically separate from the database files.

The Adlib core software applies these physical structure files to build and display an interface in which data can be presented or entered in a very user-friendly way, and to access and manage all the physical data files in the background. The user won't notice any difference between database formats (except for an indication in the status bar).

1.3 Accessing Adlib databases

Depending on the database format used, there are different ways in which Adlib databases can be accessed:

- CBF databases are of course best accessed using Adlib software. Adlib model applications are run by the adlwin or adloan executables, which together operate as stand-alone systems or as distributed systems through a local network: databases may reside on a server, while the applications run on desktops.
  CBF databases can be accessed (for retrieval as well as input) through the internet too, by means of the Adlib wwwopac executable and a web application, for instance an Adlib Internet Server web application, but custom web applications can be built as well. Moreover, there are some standard techniques available for retrieving data, like OAI (Open Archives Initiative) and SRW/SRU (Search/Retrieve Web service).
- Adlib SQL Server and Adlib Oracle databases are also best accessed (for retrieval as well as input) using Adlib software, either in a local network or via the internet. However, SQL Server and Oracle being third party technology, they can be accessed in the multitude of ways those systems can be accessed in general. But, as explained above, Adlib SQL Server and Adlib Oracle databases are differently structured from typical relational databases, and for data entry and editing it is simply necessary to have software that knows how to read Adlib structure files and apply them. We strongly advice against making manual changes to data in one of these database types, as would be possible using third party software. So, without using Adlib software, data *retrieval* would be the only real access option.

1.3.1 Retrieval and presentation of data

When using Adlib software, retrieval of data records is simple because of the user-friendly GUI. A minimum of knowledge about the structure of the accessed databases is required to find records. Being able to select a database from a list, and entering a value the user is looking for in a certain field, is enough. However, extensive combined queries in Adlib's own search language may also be utilized. Although this language will look familiar instantly to programmers, it is not nearly as complex as SQL (Structured Query Language, the default interface for relational databases), because in an Adlib search statement you never need to express the links between data.

However, if you are going to uses third-party software and SQL to search Adlib SQL Server or Adlib Oracle database, you need to know in detail how Adlib uses indexes to retrieve records normally:

- When a *Record number* access point (index) is searched, just the *priref* index is used to find a record number and the location of the record in the database.
- When a term-indexed field is searched, e.g. through an access point, Adlib first looks up the search key in the relevant field index to retrieve the record number(s) of the records in which the value occurs. Then those record numbers will be looked up in the priref index to find the actual locations of the records.
- When an index on a link reference field is searched, e.g. through an access point or when validating input for a linked field, there are three indexes to take into account. In the linked database there is a term index on the relevant field, which holds the actual keys (terms or names) and their record number in that database. Adlib uses the database definitions to find out about this index and then uses it to search for the search key and retrieve the accompanying record number.
  Now, for the current database there is an index which lists the record numbers of the mentioned linked terms or names (since a field linked by reference stores only the linked record number of the term it displays), accompanied by the (local) record numbers of the records in the current database in which the relevant field(s) link to said term or name. Adlib uses this index and the retrieved linked record number to find the relevant local record numbers.
  Then finally, the locations of those local record numbers are available through the priref index.

Note that in an Adlib SQL database the priref index is not applied: the local record number is enough to directly locate the actual record in the table.

With SQL and/or the Microsoft SQL Server Enterprise Manager, for an Adlib SQL Server database, or the Oracle Enterprise Manager Console for an Adlib Oracle database, you will be able to search index tables and word lists (for instance for integrity checks and possibly repairs), but not the records themselves currently because they are hexadecimally encoded, as mentioned earlier.

In the SQL Server Enterprise Manager for instance, you can open a table and through the right-click pop-up menu *Return all rows* (to retrieve all records or index key words), *Return top…* (retrieves the number of rows that you specify), or *Query* (to enter an SQL query).

Each row in a main table contains one record. The *priref* column contains the record number, *creation* the creation date and *modification* the last modification date of the record. The *data* column contains the hexadecimally (indicated by 0x) encoded records in their entirety. (Note that in the SQL Server Enterprise Manager the *data* column only indicates that its content is binary, and doesn't display the actual contents like in the SQL Express Manager 2005, for example.) In principle you can (programmatically) convert the complete hexadecimal string to ASCII. Every two alphanumerical characters in here represent one ASCII character. *3C* for example, is equal to decimal 60, and the 60th ASCII character is "<", the start of an XML tag. And hex *72* for instance, translates to "r". So every XML record starts with *<record*…

Thus, should you desire to build your own application as the interface for this data, instead of just using your Adlib applications, then you can translate every retrieved record to XML and process it further easily. To be able to actually find certain records, prior to any translation to XML, you will have to use the indexes which exist for many fields.

The presentation of retrieved data through Adlib software in its own GUI is fixed (although editable) in Adlib application and screen structure files. These specify per data source which fields are displayed in search result record lists or on individual screen tabs in detailed display of a record.
The presentation of data output to Microsoft Word documents or to the printer is highly flexible and more easily controlable by the user. As can be read in the next chapter, there are several ways to create output formats, even by users themselves. A great number of often used output formats comes standard with Adlib though.
The presentation of retrieved Adlib SQL Server or Adlib Oracle data through third-party software (on screen or in print) is usually programmers work, since SQL is required to create views or reports and programming skills are needed to create procedures to further process the result.

2 Core software functionality

2.1 Functionality in adlwin.exe-based applications

The so-called *adlwin* executable file provides the functionality shared by the following applications/modules, which can be purchased separately and/or in different combinations:

- Adlib Museum;
- Adlib Library;
- Adlib Archive;
- Adlib Serials;
- Adlib Library Acquisitions;
- Adlib Library Loans Management;
- Adlib Library OPAC;
- Adlib Museum OPAC;
- Adlib Museum Reproduction Orders.

2.1.1 Searching

There are three ways of searching:

- The *Search wizard* is an easy way of searching for data. Step-by-step the user is first asked to choose a data source from a list, then to select an access point (an index on a field) in a list, and subsequently to type a value to search on. (Numbers and dates are searched by range, and thus require two values.) From a list of found index values, the user must then choose one or all keys to search records with.

- Search forms (called *Query by forms*) are a more extensive way to search for records, and is available only for selected data sources. The user is presented with an empty data form which holds a number of often searched-on fields: this form allows you to enter search values in one or more fields and perform a combined search with them.

- The *Expert search system* provides the possibility to use a powerful query language to enter custom search statements. This allows you to quickly execute complex, Boolean combined searches, using any field in the data dictionary.

■ Truncation

Depending on the search method, search values may be (right) truncated implicitly or explicitly, and depending on the index definition there is the possible alternative of left truncation as well. (A truncated search is a search on a part of a word, to find all terms containing that partial word.)

■ Term substitution

If you use an access point that is linked to an authority file (such as the thesaurus), you may have used a non-preferred term as the search key. Adlib will replace your search key with the associated preferred term from the authority file, and search on it, instead of on the key you entered.
Hierarchically equivalent terms for search keys will automatically be included in the search.

■ Generic searching

From a list of found keys in the *Search wizard*, you can perform a generic search, being a search on any narrower keys as well.

■ Boolean combining

The result of a search, or single search statements in the *Expert search system*, can be combined with a new search or other single search statements to further limit or expand the search result. This can be done using any of the following Boolean operators:

- **AND**: both conditions must be met.
- **OR**: at least one of both conditions must be met.
- **AND NOT (NAND)**: the first condition must be met, while the second is not.
- **WHEN**: both conditions must be met in the same occurrence (a repeatable instance) of a field group.
- **WHEN NOT**: the first condition must be met, while the second is not, when applied to the same occurrence of a field group.

Multiple combined search statements can be nested using brackets, also determining the order in which they must be executed.

■ Relational operators

Relational operators available in the *Expert search system* are (descriptional):

- **equals**: search a field for values matching the search value(s) (strings, terms, dates or numerical values), or partly matching if the search value is truncated.
- **greater than**: search a field for numbers or dates greater than the search value.

- **greater than or equal to**: search a field for numbers or dates greater than, or with the same value as the search value.
- **smaller than**: search a field for numbers or dates smaller than the search value.
- **smaller than or equal to**: search a field for numbers or dates smaller than, or with the same value as the search value.
- **contains**: search a field (or entire records) for a (partial) word, left and right truncated at the same time.
- **hierarchically narrower (children)**: search a field in, or linked to, a hierarchical database like a thesaurus, on a search term and all narrower terms that may be defined for it.
- **hierarchically generic**: search a field in, or linked to, a hierarchical database like a thesaurus, on all one level broader terms of the search term and all narrower terms underneath them up to the lowest level, that may be defined for them.
- **hierarchically related**: search a field in, or linked to, a hierarchical database like a thesaurus, on the term itself and all related terms on the same level.
- **hierarchically broader (parents)**: search a field in, or linked to, a hierarchical database like a thesaurus, on a search term and all broader terms that may be defined for it.
- **hierarchically broadest (top term)**: search a field in, or linked to, a hierarchical database like a thesaurus, on the top most broader term(s) of all broader terms of the entered term, that may be defined for that entered term, or the term itself if broader terms aren’t present.

■ Sets and pointer files

During your session in a database, search language statements can be used in following search statements by referring to them as sets. A result set may also be saved on hard disk as a pointer file: such a file contains record numbers of the concerning search result and (if applicable) the search statement itself. These pointer files can be opened at any later time to view the records it refers to and/or to quickly re-execute the same search statement or to combine it with a new search statement.

2.1.2 Display of the search result

■ List screen

The result of a search is presented in a brief display screen which shows a small selection of data (including a thumbnail of any linked

image) from every retrieved database record. A search result can be sorted on one or more fields, using only the first field occurrence or using all occurrences separately, and depending on the type of field(s): alphabetically, on date, or numerical, all ascending or descending. You can even set a custom adapl program to do some pre-sort data processing of the search result.
Besides the *Brief display* screen there are two more screen tabs, *Thumbnails* and *Filmstrip* which present the same search result in a different way, more focused on the images linked to the found records.

■ Browsing

From the list screen, the user can display and/or edit records in detail. From a record in detailed display you can also easily browse to next or previous records, or the first or last record, in the search result.

■ Detailed display

The detailed display of a record is presented as a selection of tabbed data forms on which topically related data is conveniently grouped. Any linked images are shown in a separate, resizeable and zoomable image viewer window in front of (some of) the detail screens. A selection of EXIF metadata about the image file is available from within the image viewer too.
In detailed display, underlined data acts as hyperlinks to the detailed display of the relevant linked record, or to opening the relevant linked document or URL in an appropriate third-party software application.

■ The hierarchy browser

In the detailed presentation of records from hierarchically built databases like an archive database, the *Thesaurus*, or *Persons and institutions*, you can display the tree structure of all records that are linked internally to the currently displayed record through broader and narrower terms.

2.1.3 Data entry

■ New records

New record can be entered at almost any point during the user's session in Adlib, after choosing a data source for the new record.
Note that Adlib offers an extensive access rights system to block individual users from doing things they are not allowed to. The creation of new records, editing, or adding terms and names to authority files are just a few examples of functionality which can be shielded.

■ Editing records

Usually, any record in detailed display can be edited. And through linked fields in the current record you can even view, create or edit records in the linked database.

■ Default field values

The application manager may have set default values or automatically calculated values for some of the fields of a record; it depends on the setup if such predefined values can be changed by the user. The record number is always assigned automatically by the Adlib software.

■ Mandatory fields

There may also be mandatory fields. The user has to fill in a value in these fields before the record can be saved.

■ Field entry conditions

Different fields have different entry conditions: some accept several lines of text, others only a few words, individual letters, or an integer number, a floating point number, a date (in one of seven European, ISO, American and Julian formats), a time, an ISSN, ISBN, or a Boolean value. Or a field offers a drop-down list of preset values from which the user may choose. Further, there are fields which only accept a path to a file or to an image, or a URL to a website. And rich text fields allow the user to apply font layout types to format the text. It is also possible that an adapl (a small additional, and customizable, program in Adlib's own programming language) is used to further restrict or reformat input.
Some fields are read-only and cannot be changed by the user. Usually these fields are automatically filled with data from a linked database, when the user fills in some key field. For instance, after entering a name in some catalogue record, accompanying data like address and phone number may be retrieved automatically from another database and displayed here in read-only fields.
And then there are so-called conditional fields: these are screen fields which are conditionally hidden or set to read-only. That means a screen field may be visible or not, or may be editable or not, dependent on the value(s) in certain other fields.

■ Validation of data

Since Adlib utilizes different databases for different areas of interest, links have been created between fields and databases that must share data. A prominent example of this functionality is the use of a *Thesaurus* and a *Persons and institutions* database as authority files. In these databases you register preferred terms, respectively names in their proper format and spelling and in an appropriated, so-called domain – domains are subsets of a database, only containing terms which have

a similar meaning. A number of fields in other databases, like in the catalogues, have been linked to one these authority files, making them validated fields: this means that when the user fills in such a field, only terms or names present in the authority files (and possibly from a certain domain) are acceptable input, or the user must explicitly add the new term or name to the relevant authority file. This validation can be requested by the user but is always performed automatically as well when the user leaves the field. For an empty or new value, the validation process immediately offers the user a choice of existing terms or names in the relevant authority file from which to choose, and also the option to quickly add the new term or name to the authority file. Adlib further supports the use of multiple thesauri in a single application; and with the global update function you may update specified fields in all marked records of the current database with the aid of a different thesaurus.
Properties and Help are available for pratically every field and screen, to help you fill in proper values.

■ Repeatable fields

Some fields are repeatable, to allow the user to register e.g. more than one author of a book or multiple object names for one object. The user can simply add extra instances of such fields, called occurrences, to the active field.

■ Linking images and other files

Fields that accept a link to an image or other file needn't be filled in manually. Through standard Windows Explorer functionality the user can search the system or network for the image or file to link to; the path to it will then automatically be copied to the relevant field.
Links to existing images can even be created by dragging an image file to the Adlib image viewer.
Adlib supports Windows Image Acquistion as well. If your application has been set up for this, the user can create and link images in a single procedure, by using a scanner, camera or other still-image device.

■ Copying data

Records can be copied to base new records on existing data. And besides the standard Windows cut, copy and paste functionality with which you may edit individual field contents, Adlib also implements its own clipboard, to copy the contents of multiple fields to the corresponding fields in another record.

■ Deriving records

With deriving you can copy records from other, so-called 'friendly' databases or datasets to the current one. This is useful when you are describing documents or objects that contain similar data. (Note that a customization is needed to enable this functionality in a standard application.)

■ Checking spelling

You have the possibility to perform a spelling check on text in non-linked text fields. The spell-checker automatically checks the contents of all these fields in the currently opened screen.

■ Entering special characters

If an Adlib application employs a Unicode database, then the user is allowed to enter special characters in text fields, for instance characters from languages like Hebrew, Chinese, and Greek.

■ Multi-lingual data entry

Adlib applications and databases can be customized to allow for multi-lingual data entry and searching. It depends on the implementation how the user can switch between entry languages.

■ Search and replace

After marking records, you can perform a search-and-replace on these records, where you enter a character string to be found in certain fields (which you can select yourself), which must be replaced by another character string which you have entered. Options include searching on the whole field contents, on whole words, on partial words, matching the type case, and with or without confirmation for each replacement.

■ Automatic creation of reciprocal records

In hierarchical Adlib databases, like the *Thesaurus*, a number of fields are internally linked to each other, which allows you to create relations between terms in one and the same record, for instance the relation between a preferred term and a non-preferred term, or the narrower (more specific) terms for the currently specified preferred term. When you save such a record, mirrored records for all the related terms in this record are created by Adlib automatically (if they don’t exist yet), so that every narrower, broader, equivalent and other such terms get their own record in which the reciprocal relations are specified.

2.1.4 Using record selections

In brief or detailed display of records, records can be marked to include them in a temporary selection of records. Several operations in Adlib can be performed on a single record as well as on selections of records. This includes deleting records, search-and-replace of data in fields, printing, exporting and a global (thesaurus) update of selected fields. And the pointers to records in a selection can be written to a pointer file, to be able to recall the current record selection at a later time.

2.1.5 Printing

There are three ways of printing selected records from within Adlib, namely:

- through custom or standard adapl programs, called output formats, created especially for printing records. Being programs, these output formats offer great freedom and flexibility in data manipulation.
- through custom or standard Word templates. Word templates are easy to create and offer all the layout functionality MS Word has to offer. You can create templates for labels, for standard letters and for record lists. It’s up to you to decide which database fields and fixed text should be included in a template. Images can be printed as well. Printing this way results in Word documents which can be printed, saved or automatically sent via e-mail. Output formats and Word templates can even be used together to get the best of both worlds.
- through an interactively created report via the *Print wizard*. With this wizard, the user doesn’t need to program adapls or create Word templates to make a new print format. Available options in the *Print wizard* include:
    - adapting earlier created print formats;
    - selecting the fields to be printed;
    - setting the order of the fields to be printed;
    - per field, choosing whether only the first field occurrence or all occurrences must be printed;
    - per field, if empty fields must be printed;
    - per field, setting the column width in which the field contents will be printed;
    - setting of record printing in columns or rows;
    - choosing portrait or landscape page format;
    - setting header and footer texts, which may contain variables like the printed page number or the current date;
    - setting a record separator;
    - setting font styles for headers, footers, field labels and the data itself. Custom font styles can be saved;

- saving the current print format.

After finishing the *Print wizard* (also when you have selected an output format), the standard Windows *Print* window will open, so that you can select a printer and change other standard settings (number of copies and whether to print to file, to e-mail or as MS-DOS text) before actually printing.

When you just want to print an image linked to a record, you can use the image viewer in Adlib.

2.1.6 Import and export

Selected records can be exported to exchange files and Adlib records in exchange files can be imported into selected data sources in your Adlib application. In Adlib Designer you'll find the most extensive implementation of import and export functionality, but in Adlib applications you'll be able to do some common import and export tasks too.

You can use earlier created jobs (settings for an import or export procedure) and save newly created ones.

The *Export wizard* offers a choice of the exchange file formats *Adlib tagged file*, *CSV (comma separated values) file*, and *XML file*. You can also select the fields which you want to export.

The *Import wizard* offers a choice of the exchange file formats *Adlib tagged file* and *XML file*. While importing, Adlib can check whether a record already occurs in the database and update it, and assign new record numbers or import the existing ones.

2.1.7 SDI

SDI stands for Selective Dissemination of Information. This means that you can keep your customers (for instance library visitors or literature researchers) abreast with information that is relevant to them. In practice, it means that you can create and save interest profiles based on search queries. Based on these profiles, the Adlib system is automatically searched for information, which is then presented in a user-friendly format and layout specified by you through an adapl or stylesheet. You could use this to send your customers regular e-mails or printouts about new additions that are relevant to them.
A number of options allow you to specify start and end date, set the frequency with which the SDI procedure must be executed, and limit the search results, and other related configuration settings.

2.1.8 Data protection through access rights.

As mentioned earlier, data can be protected by using access rights in various ways. Users may need to log in before being able to use Adlib, or selected users may not have read, write or full access to certain

data sources. Even selected records may be hidden from certain groups of users.
And standard Adlib core software functionality may be protected too, for example to shield the use of pointer files, or particular search methods, or exporting, importing, deriving, or access to output formats, and much more.

2.1.9 Interface adjustability (by user)

■ Multi-lingual presentation

Adlib has a multi-lingual graphical user interface, meaning that the user can easily switch between at least five languages (English, Dutch, German, French and Greek) to present all menu's, screen texts and other system texts in. (This interface language is completely separate from the language in which you wish to enter data.)

■ Screens can be hidden

A record in detailed display is often presented on many screens, because there are just so many fields that could be used for registering. However, if there are certain screens which you never use, you may hide them from view. Adlib will remember your settings for the next time you log in. Hidden screens can be displayed again at any moment in detailed display of a record.

■ Customizable toolbars

You can adjust the arrangement of the toolbar to your liking (to a certain extent): you may change the order of the sub toolbars with respect to each other, you can shorten or lengthen sub toolbars (so that fewer or more buttons in them are directly visible), and you may place sub toolbars underneath each other in new toolbar lines.

2.1.10 Barcode scanner input

The output of up to four (barcode) scanner devices connected to your computer (through the serial ports) can automatically be imported into an Adlib record in edit mode. You can provide a regular expression to which borrower numbers must comply. This is because with the scanner you can scan borrower numbers from library cards as well as copy numbers from e.g. books. But the scanner doesn't differentiate between such numbers. So with the regular expression you specify the format of the borrower number; all borrower numbers which do not comply to this format, will be considered copy numbers by Adlib.