User friendly interface for maintaining BIBTEX databases

See Mu Kim* with Michael J. Dinneen†
Department of Computer Science
University of Auckland
Private Bag 92019
Auckland, New Zealand

Abstract

We describe a new bibliographic database tool, BIBTECKER, that helps users manage their BIBTEX databases. The main features of BIBTECKER are a GUI interface, a built-in entry syntax checker, user extendable type fields, comment maintenance, verbose mode and command line parameters. We also compare BIBTECKER with existing BIBTEX tools and mention plausible improvements.

1 Introduction

Authors who use LATEX to produce new scientific or mathematical documents usually have their own BIBTEX databases for references (see [1] and [2]). The problem of using LATEX with BIBTEX databases is that users have to manage the databases themselves since the standard LATEX system does not provide any kind of database management tool. Manual maintenance of large BIBTEX databases requires enough patience and error-free typing skills. Therefore, it is a good idea to have software tools that make the whole process less painful.

Currently, there are some applications which are designed to help users to maintain their databases such as Emacs BIBTEX mode or the bibtool package (see [3] and [7]). Firstly, the BIBTEX mode provided by Emacs has basic operations, insertion, deletion and modification as well as field operations relating to an entry. However, it is not commonly used among people who are not familiar with Emacs on UNIX because it is text-based rather than using a graphical user interface, and it cannot be used on many platforms. Secondly, the bibtool package does not have any basic operations like insertion, deletion and modification. However, it has some distinctive features that Emacs lacks such as entry syntax checking, key duplication checking, sorting and merging.

*`skim048@cs.auckland.ac.nz`
†`mjd@cs.auckland.ac.nz`

The primary aim of this project is to design and implement a user friendly graphical interface which is not only simple to use but also functionally powerful to use on any platform. The important features of this new BIBTEX tool are the following.

- It supplies a graphical user interface for insertion, deletion and modification of entries.
- It has a parser which checks the syntax of entries in a BIBTEX database file and also guides users with user friendly error messages.
- It has a verbose mode for the beginners.
- It supports command line input arguments.
- It provides easy and simple search functions.

The above main features will be explained in detail throughout the rest of this report. Section 2 introduces the definition of BIBTEX databases, the types of standard entries and fields. Section 3 gives a user's guide on the BIBTEX tool, BIBTECKER. Section 4 discusses design and implementation issues relating to BIBTECKER. In the last section, we conclude and mention a couple of future improvements for BIBTECKER.[1]

2 What are BIBTEX databases?

This section explains BIBTEX databases and the format of standard BIBTEX entry types, and shows how to use BIBTEX databases in a LATEX document.

2.1 Definition

A standard BIBTEX database for LATEX is a database file that contains various information about publications and works of reference[4]. Each publication in a BIBTEX database has a standard format called entry. An entry consists of an entry name, a key value and fields. One of standard entry types explained in Section 2.2 is used as an entry name. A key value identifies an entry in a BIBTEX database, so it is used to refer to a specific entry in a LATEX document. Each piece of information about the publication is stored in a different field of the entry.

2.2 Format of BIBTEX entry types

There are 15 standard entries, including the STRING macro, which are recognised by LATEX documents. Thus, all other types of entries with the exception of 15 standard entries are considered as comments in a BIBTEX database. Apart from the STRING macro, the 14 standard entries explained in Section 2.2.1 are used to record information about publications and works of reference.

[1]The current version of this software is freely available by contacting one of the authors.

Usually, the STRING macro is used to define an abbreviation in the field text in place of actual text. The abbreviation name consists of any combination of letters, numbers and symbols except "#%'(),={} characters and the format of actual_text follows the standard field value format mentioned in Section 2.2.2. It is important for users to remember that the STRING macro should be defined before it is actually used in a BIBTEX database.

The STRING macro has the following format.

```
@STRING{abbr_name = actual_text } or
@STRING(abbr_name = actual_text )
```

For example,

```
@STRING{ JGR = { Journal of Geophysical Research } } or
@STRING( JGR = { Journal of Geophysical Research } )
```

2.2.1 Standard entry format

The 14 standard entry types have the following format.

```
@ENTRY_NAME{KEY,
            FIELD1 = { },
            FIELD2 = { },
            FIELD3 = { },
            FIELD4 = { },
            ...
            }
or @ENTRY_NAME(KEY,
            FIELD1 = { },
            FIELD2 = { },
            FIELD3 = { },
            FIELD4 = { },
            ...
            )
```

For example,

```
@article{BH80,
      author = {Terry Beyer and Sandra Mitchell Hedetniemi},
      title = {Constant time generation of rooted trees},
      journal = SJC,
      volume = {9},
      year = {1980},
      pages = {706–712}
      }
```

The first word, prefixed with @, determines the entry type. All information in each entry type should be contained in parentheses({} or ()) and each field is separated by a comma. The KEY field is a unique key value which is used in a LATEX document by specifying the \cite{KEY} or \nocite{KEY} command. All entry names are not case sensitive for the user's convenience. Users can add, delete and rename any field but required fields. In general, each entry has different types of fields since references to different types of publications can contain different information. For example, a reference to a journal might include the volume and number of the journal, which is usually not meaningful for a book. The details of standard entries can be referred to in [4] or in the BIBTECKER itself.

2.2.2 Field format

The field format in each entry has the following form.

field_name = {any string} | "any string" | abbreviation
[# {any string } | # "any string" | # abbreviation]*

For example,
title = "Computer" # JGR # {in modern society} # 1999

Each field consists of field_name, an equals sign and field value. Like an abbreviation in the STRING macro, the field_name can be composed of any combination of characters except special symbols such as "#%'(),={}. field_name is not case sensitive like an entry name. The field's value can consist of more than one substring or abbreviation, but importantly each substring should be contained in {} or "". In addition, any two substrings including abbreviations can be combined by using a # character between them. There are three different types of field: required, optional and ignored fields.

Required field: Must be present in the entry. If omitted, an error message will be produced.

Optional field: Optional fields can be omitted, but if they are specified in the entry, they will be also displayed in the output.

Ignored field: The BIBTEX program ignores all fields that are not required or optional fields. These fields are called ignored fields. Any data in the ignored field is not displayed in the output unless the user writes a BIBTEX stylefile. Users can add any ignored field to an entry, but a newly inserted field name should be unique in the entry. This field can contain all relevant information such as an abstract of a paper or a comment.

The types of standard field can be found in [4].

2.2.3 How to use BIBTEX entries in a LATEX document

There are two different methods to specify a bibliography in a LATEX document. One way is to use the \bibitem command in a LATEX document, in which case users have to include corresponding information about each publication at the end of a LATEX document. The other way is to use a \cite or \nocite command, in which case users can make use of

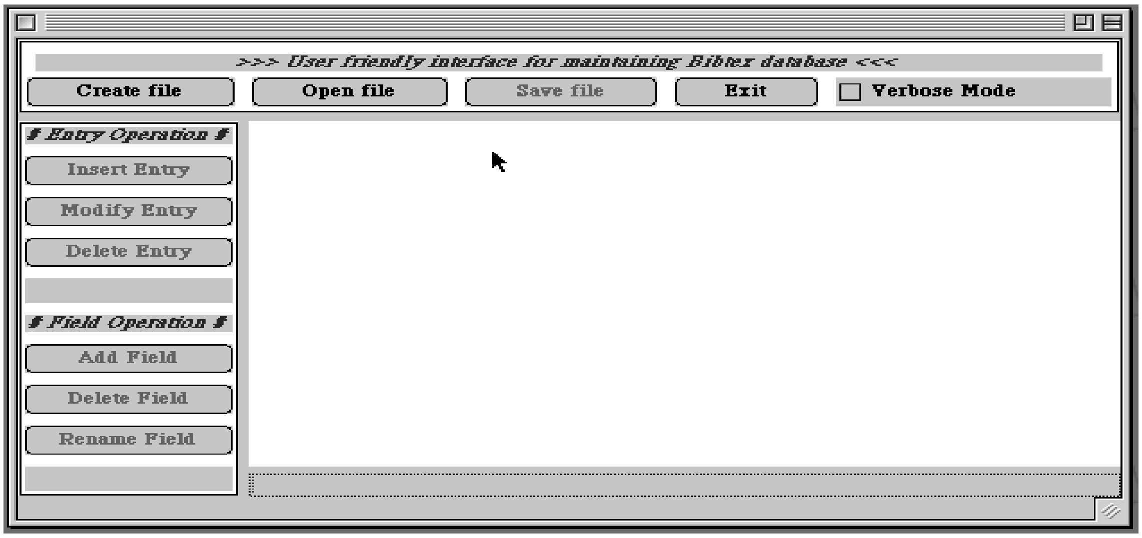

Figure 1: Initial window when BIBTECKER is loaded.

BIBTEX databases. The BIBTEX program is used to construct a bibliography for a LATEX document by searching one or more BIBTEX databases specified in the LATEX document. The great advantage of using BIBTEX databases is that if entries are included in a BIBTEX database only once, they can be available for future publications and can be used by other people who can access the same BIBTEX databases. An example for using the \cite{key} command is given in [8].

3 A user's guide

This section introduces a new BIBTEX tool, BIBTECKER, which provides users with user-friendly graphical interfaces to maintain BIBTEX databases. Most operations explained in this section are performed in temporary workspace. The explanation of this workspace is given in Section 4.2.

3.1 How to run BIBTECKER

Because BIBTECKER is implemented in Java[5], it can be used on any platform without recompiling the source code. The following explanation of how to invoke BIBTECKER applies to command-line versions of Java (eg. JDK under UNIX or DOS). Users can run BIBTECKER by typing the following:

%> java BIBTECKER

If BIBTECKER is loaded successfully, the main window is displayed on the screen (Figure 1). One of the main features of BIBTECKER is that users can specify input parameters from the command line.

%> java BIBTECKER [BIBTEX file_name] [-katsc search_value]

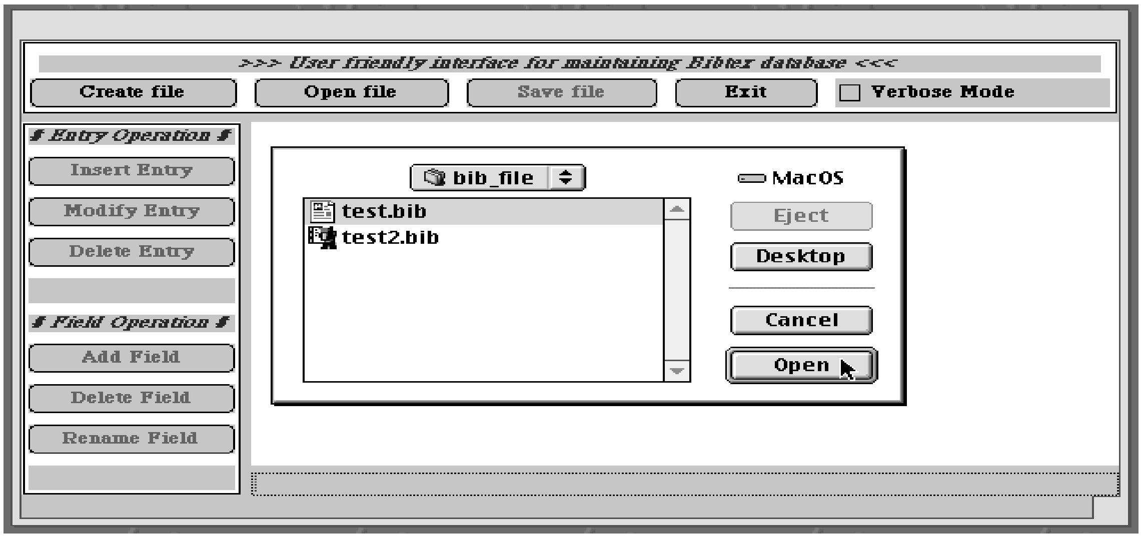

Figure 2: File dialog box when *Open file/Create file/Save file* button is pressed.

The first parameter, [BIBTEX file_name], is a BIBTEX database file that users want to open in BIBTECKER. It is optional, so it can be omitted. When [BIBTEX file_name] is present, users can also specify a search type via a command-line switch, and text to search for as the parameter search_value. There are 5 different search types, but users can only use one of 5 alternatives in a single invokation. Explanation of the searching types is as follows:

-k search_value: -k means search by key value.
When this option is used, BIBTECKER tries to find an entry which has the corresponding key value.

-a search_value: -a means search by author.
When this option is used, BIBTECKER tries to find the first entry which contains the search_value in an author field as a substring.

-t search_value: -t means search by title.
When this option is used, BIBTECKER tries to find the first entry which contains the search_value in a title field as a substring.

-s search_value: -s means search by abbreviation in the BIBTEX database's STRING macros. When this option is used, BIBTECKER tries to find the STRING macro which has the corresponding abbreviation.

-c search_value: -c means search the BIBTEX database's comment by substring.
When this option is used, BIBTECKER tries to find the first comment which contains the search_value in the comment

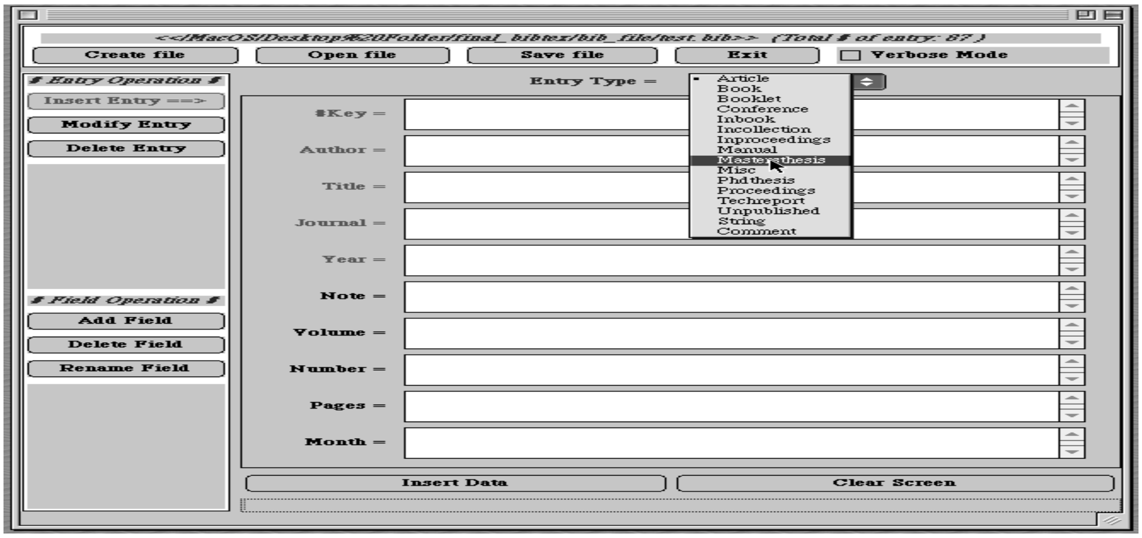

Figure 3: Insertion panel to insert new entries.

as a substring.

For the searching operation, any string except an empty string can be used as search_value and also search_value is not case sensitive. The -k and -s options are based on exact string matching, while the other options are based on substring matching.

3.2 File I/O buttons

BIBTECKER provides 3 different buttons relating to file I/O operations.

***Create file* button:** Used to create a new BIBTeX database file. When it is clicked, if there is any modified entry in the workspace that has not been saved explicitly into a BIBTeX database file, BIBTECKER will display a warning message (Figure 13) before creating a new file, otherwise it will show a file dialog box (Figure 2) for creating a new BIBTeX database file.

***Open file* button:** Used to open a new BIBTeX database file. When it is clicked, if there is any modified entry in the workspace that has not been saved explicitly into a BIBTeX database file, BIBTECKER will display a warning message (Figure 13) before opening a new file, otherwise it will show a file dialog box (Figure 2) for opening a new BIBTeX database file.

***Save file* button:** Used to save the contents of the current entries in the workspace onto a user's hard disk permanently. When it is clicked, BIBTECKER will display a file dialog box (Figure 2) for saving the current BIBTeX database file. Also users can create a new BIBTeX database file by specifying a different file name in a file dialog box.

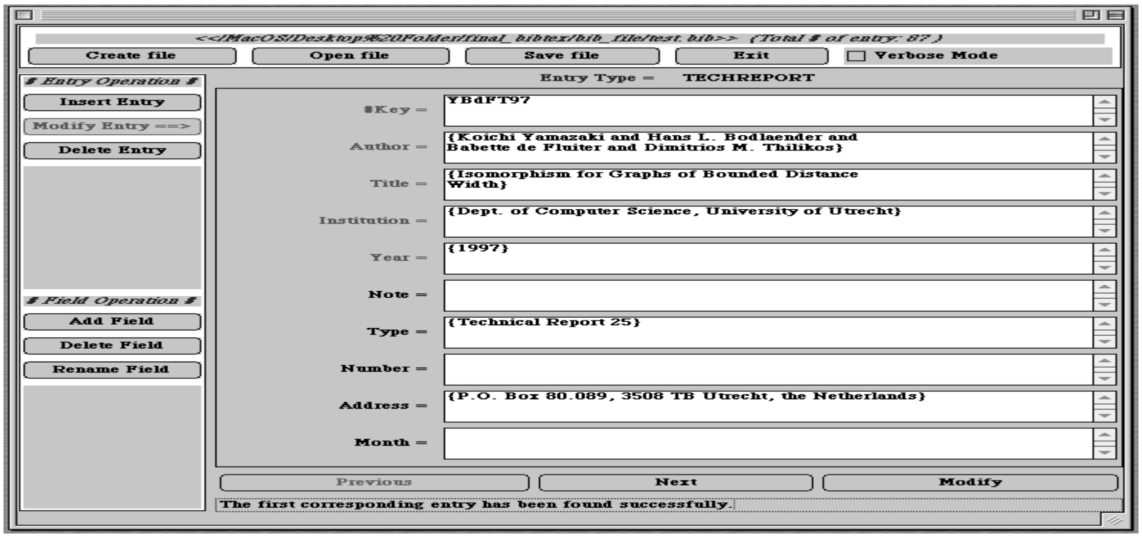

Figure 4: Modification panel to modify entries.

3.3 Basic entry operations, *Insert, Delete* and *Modify Entry*

***Insert Entry* button:** Used to insert new entries into the workspace. Users can select any kind of entry type by clicking the entry type list box on the top right of the insertion panel (Figure 3). Entry type list box has the COMMENT entry as well as other standard 15 entries including the STRING macro. With the COMMENT entry, users can add new comments to the end of a BIBTEX database file. Any kind of string can be used in the COMMENT entry, but when curly braces, {}, are used in the COMMENT entry, each opening curly brace, {, requires a corresponding closing curly brace, }. The following examples in the COMMENT entry will produce a syntax error message.

Example 1: This is a {comment

Example 2: This is a } comment {

Standard entries other than the COMMENT entry follow the format for standard entries given in Section 2.2. Whenever users select a different entry type in the entry type list box, corresponding entry fields will be displayed. By default, it displays required fields in red and optional fields in black. Using a different colour is very useful for distinguishing between required and optional fields in the entry because it is not easy for users to remember which field is a required and which field is an optional in the 15 standard entries. BIBTECKER allows users to do some field operations in the insertion panel (Figure 3), Thus users can insert new ignored fields to the entry or delete/rename any existing optional or ignored field from the entry. Those operations will be explained in Section 3.4 in detail. When users click the *Insert Data* button at the bottom of the insertion panel (Figure 3) after filling in

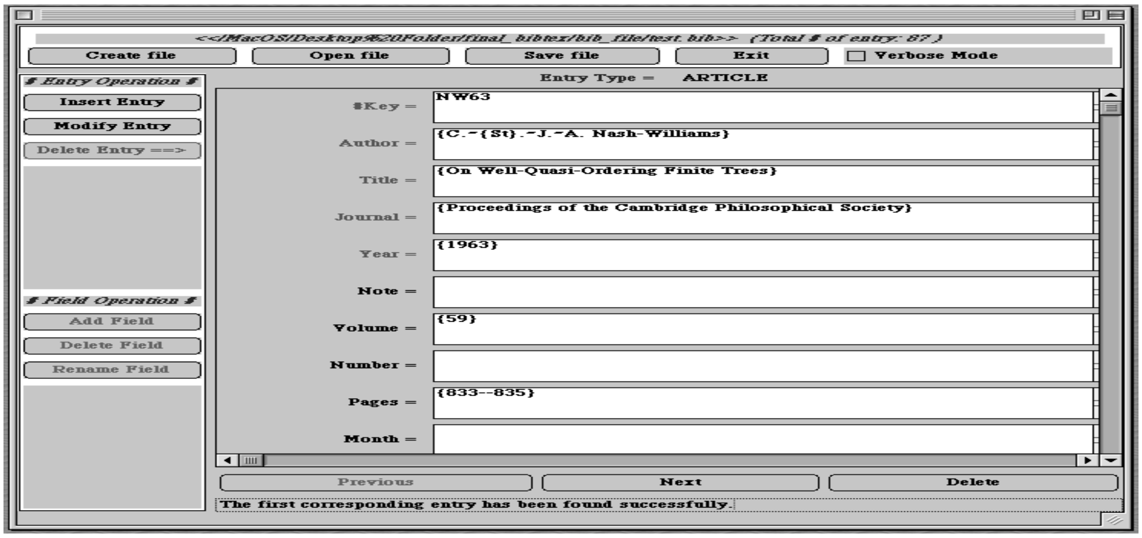

Figure 5: Deletion panel to delete entries.

each field value, the field syntax checker explained in Section 4.3 checks the syntax of each field. If the syntax of each field is correct, the current entry will be inserted into the end of the workspace and the fact that the workspace has been changed will be indicated by the text “<*Modified*>” being appended to the text in the title bar of the main window (Figure 13). Otherwise, if the syntax is incorrect, BIBTECKER will display a dialog box (Figure 11) containing syntax error messages. The *Clear Screen* button at the bottom of the insertion panel (Figure 3) is used to clear the field values in the current window, which means it does not actually delete any entries from the workspace.

***Modify Entry* button:** Used to modify entries from the workspace. When this button is clicked, BIBTECKER will show a query dialog box (Figure 12) that allows users to select a search type and search string. This dialog box has 5 different search types explained in Section 3.1. When BIBTECKER has found the first corresponding entry from the workspace, the modification panel displaying the contents of the entry will be opened (Figure 4). The modification panel has 3 buttons, *Previous, Next* and *Modify.*

***Previous* button:** Used to find the previous entry which matches the search type and value.

***Next* button:** Used to find the next entry which matches the search type and value.

***Modify* button:** Used to write the contents of the modified entry from the window back into the workspace.

The functionality of the *Modify* button is same as the *Insert Data* button in the insertion panel (Figure 3). Basic field operations in Section 3.4 are also allowed in the modification panel.

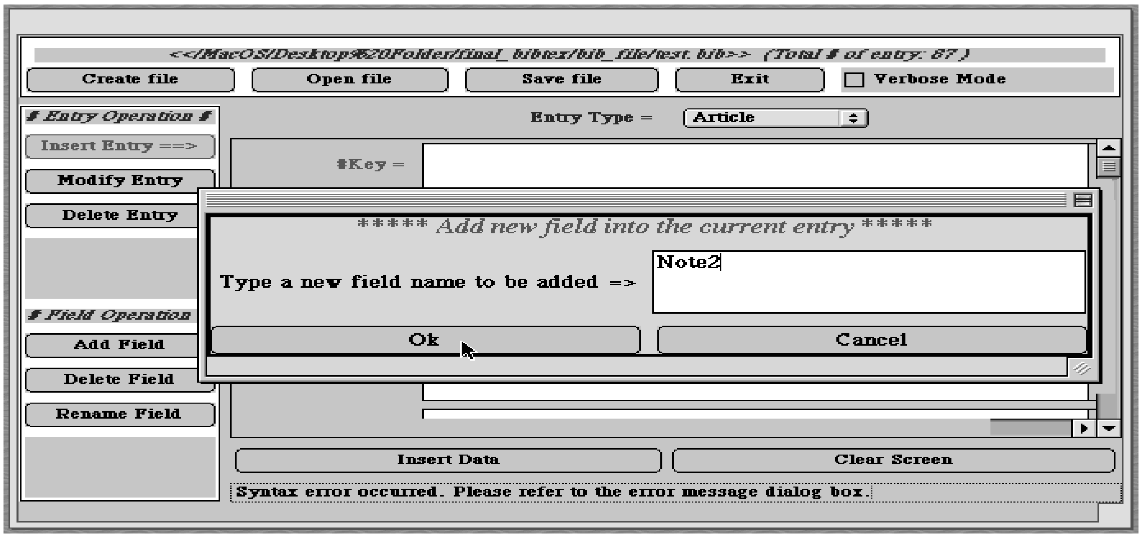

Figure 6: Add field dialog box to add a new field.

***Delete Entry* button:** Used to delete entries from the workspace. When this button is clicked, BIBTECKER will display a query dialog box (Figure 12) that allows users to select a search type and search string. When BIBTECKER has found the first corresponding entry from the workspace, the contents of the entry will be loaded onto the deletion panel (Figure 5). The deletion panel has 3 buttons, *Previous, Next* and *Delete.* The functionality of the *Previous* and *Next* buttons in the deletion panel is the same as those in the modification panel (Figure 4). The *Delete* button is used to delete the current entry from the workspace. Users cannot modify the contents of the entry. Field operations are not allowed in the deletion panel.

3.4 Basic field operations, *Add, Delete* and *Rename Field*

As mentioned in Section 2.2.2, BIBTECKER only displays required and optional fields for each entry. For users who want to add a new ignored field to an entry or delete an optional or ignored field from an entry or rename an optional or ignored field within an entry, BIBTECKER supplies 3 basic field operations.

***Add Field* button:** Used to add a new ignored field to the current entry. When it is clicked, the add field dialog box will be displayed (Figure 6). If the newly inserted field does not exist in the current entry, it will be added to the end of the entry. Otherwise, the add field dialog box will be displayed again (Figure 6).

***Delete Field* button:** Used to delete an optional or ignored field from the current entry. When it is clicked, the delete field dialog box will be displayed (Figure 7). Users can only delete optional or ignored fields from the current entry.

***Rename Field* button:** Used to rename an optional or ignored field in the current entry. When it is clicked, the rename field dialog box will be displayed (Figure 8).

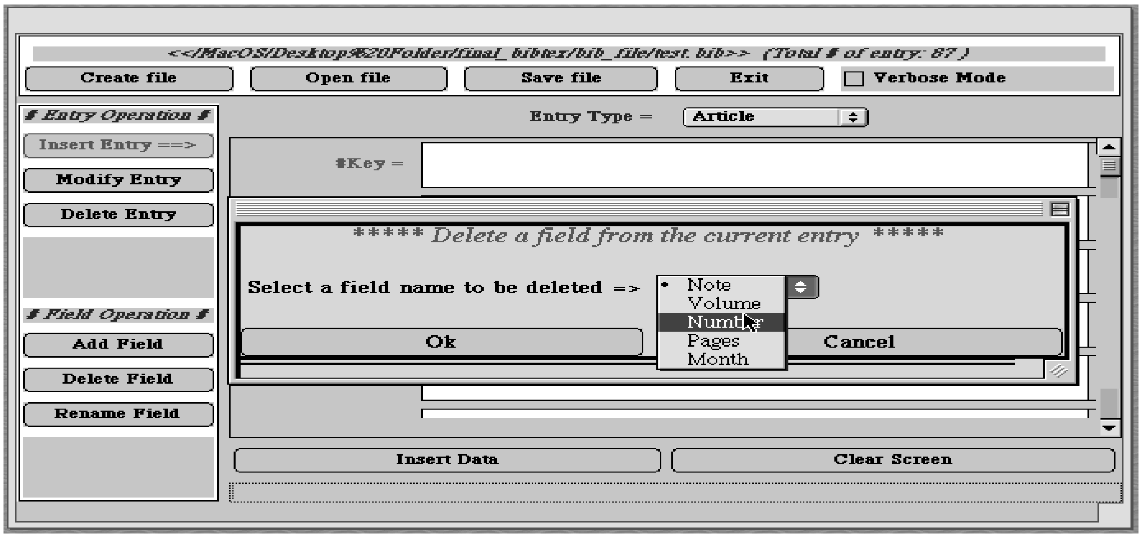

Figure 7: Delete field dialog box to delete a field.

Users can only rename optional or ignored fields from the current entry. Also, the newly renamed field must be unique in the entry.

For *Add Field* and *Rename Field* operations, the syntax of an abbreviation in the STRING macro explained in Section 2.2 is also applied to the syntax of the field name, so "#%'(),={} characters are not allowed in the field name.

3.5 Other functionalities

***Exit* button:** Used to exit BIBTECKER (Figure 1). When it is clicked, if modified entries in the workspace have not been saved explicitly, BIBTECKER will display a warning message (Figure 13) before exiting a program.

***Verbose mode*:** For users who are not familiar with BIBTECKER, verbose mode (Figure 1) is provided. This mode is off by default. Verbose mode is used to prevent unwanted accidents such as deleting or modifying the current entry. If verbose mode is on, it affects all button operations in BIBTECKER except when clicking on *Create file, Open file, Save file* and *Exit* button (Figure 1). Suppose users click the *Delete* button in the deletion panel accidently. If verbose mode is on, BIBTECKER will display a warning message dialog box (Figure 9) before deletion. Otherwise, the current entry will be deleted from the workspace without any warning.

***Message bar*:** Located at the bottom of the main window (Figure 9). Whenever any action is performed by the user, BIBTECKER will display a corresponding message in this message bar. According to this information, users will be informed whether the previous action was performed successfully or not.

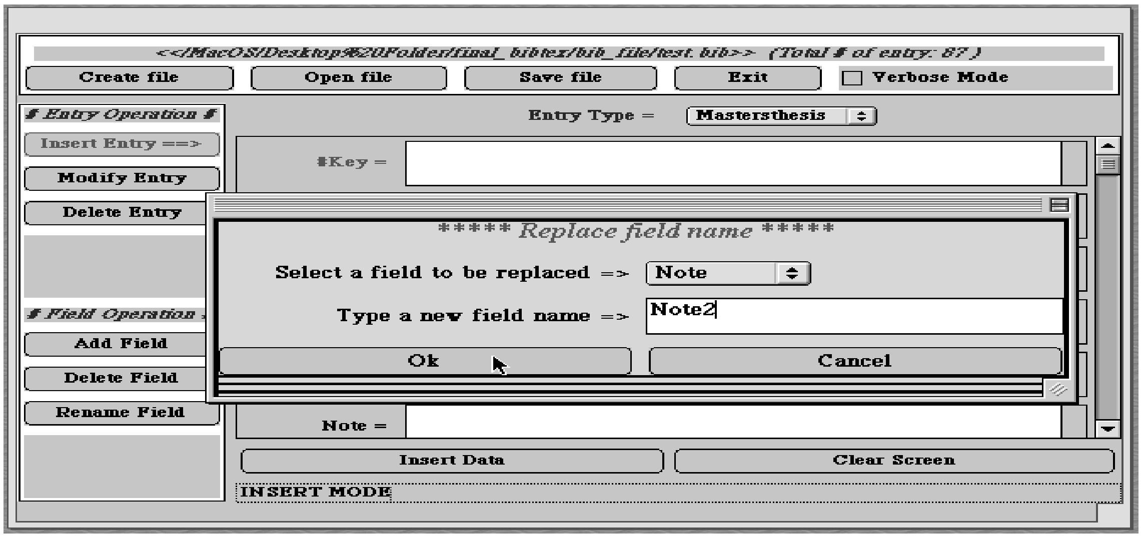

Figure 8: Rename field dialog box to rename a field.

3.6 Kinds of error messages

BIBTECKER provides users with helpful error messages. The error messages can be classified into file I/O, entry parsing and field value parsing errors.

3.6.1 File I/O error

This can occur when users open a BIBTEX database file by clicking *Open file* button on the main window or specify a filename as an input parameter. The opening parser in Section 4.3 is responsible for detecting file I/O errors.

3.6.2 Entry parsing error

This can occur when BIBTECKER parses an opening BIBTEX database file. In general, most users have their own BIBTEX database files which are maintained using an editor such as vi, notepad and so on. When users open those files in BIBTECKER, the opening parser in Section 4.3 checks the syntax of each entry in a BIBTEX database file and displays corresponding error messages (Figure 10) if it finds any syntax errors in the entry.

Error type 1
} character has been detected before { character in the entry.
Or) character has been detected before (character in the entry.
Solution
Check if the number of { and } or (and) are equal in the entry.
Possible error example
@article{key1, title = “project }}
Correct usage
@article{key1, title = “project”}

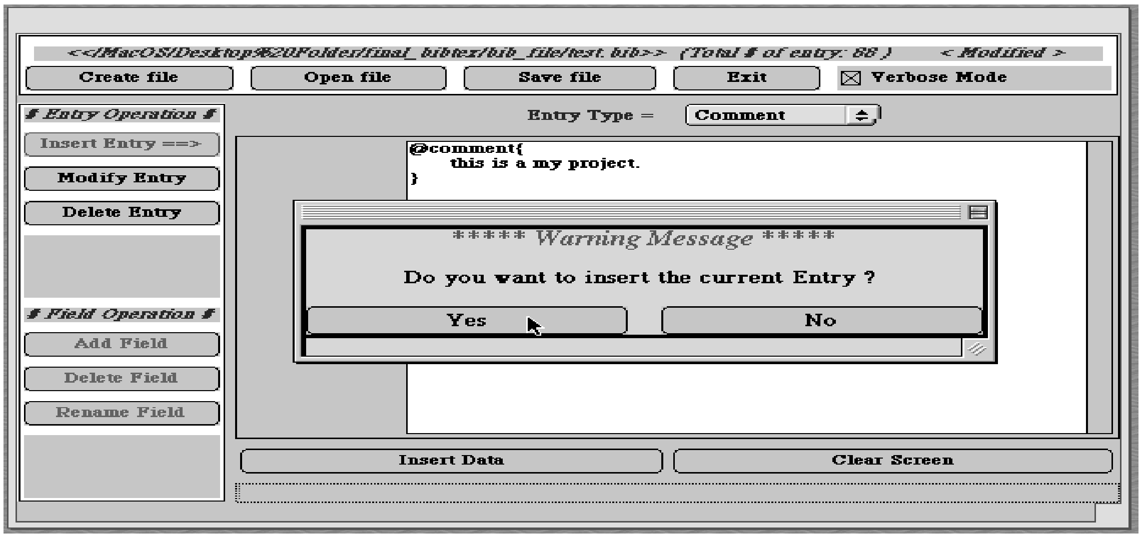

Figure 9: Warning message dialog box when the verbose mode is on.

Error type 2

Each entry should be contained in {} or ().
Or an unnecessary character, {, was detected in the entry.

Solution

Check if each entry starts with { and closes with }.

Possible error example

@article{key1,title = {project}

Correct usage

@article{key1, title = {project}}

Error type 3

The syntax of the STRING macro is not correct.

Solution

The STRING macro has the following format.
@STRING(abbreviation = real text)
or @STRING{abbreviation = real text}

Possible error example

@STRING VIP = "very important person"
@STRING {VIP = "very important person")
@STRING (VIP = "very important person"}

Correct usage

@STRING{VIP = "very important person"}
@STRING(VIP = "very important person")

Error type 4

An abbreviation of the STRING macro is not correct.

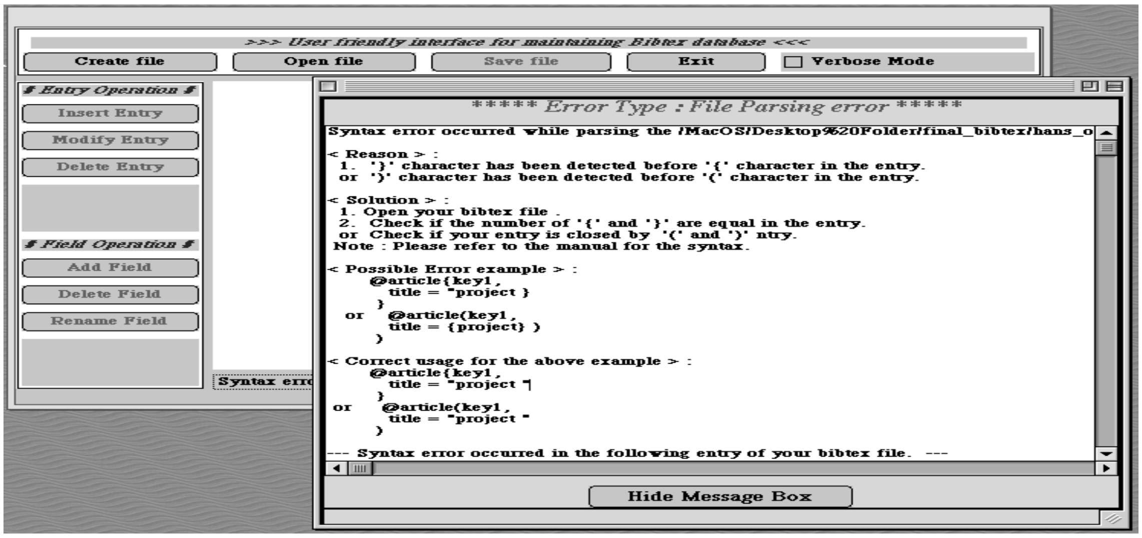

Figure 10: Entry parsing error messge while opening a BIBTEX file.

Solution

Check if you used the correct syntax of the STRING macro.
You cannot use "#%'(),={} characters in abbreviations.
Also space characters are not allowed in abbreviations.

Possible error example

@STRING { = {very important person}}
@STRING {VI P = {very important person}}
@STRING {VI%P = {very important person}}

Correct usage

@STRING{VIP = {very important person}}

Error type 5

The key value of your entry is not correct.
Or you missed out the key value in the entry.

Solution

Check if you used the correct key value in the entry.
"#%'(),={} characters are not allowed in the key value.
Also space characters are not allowed.

Possible error example

@Misc { }
@Misc {key%1}

Correct usage

@Misc{key1}

Error type 6

There is no = sign between field name and field value in the entry.

Solution

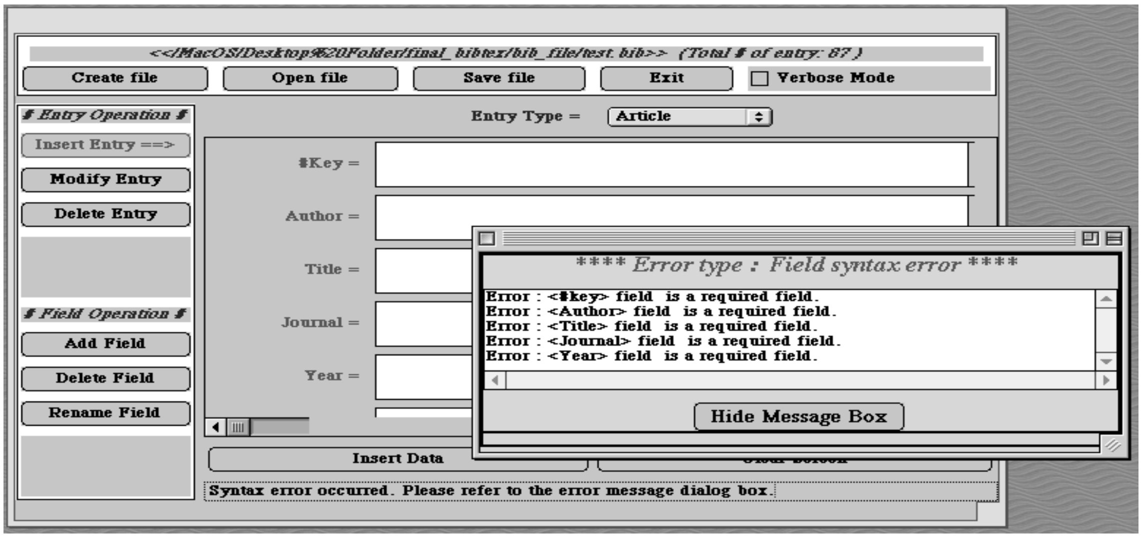

Figure 11: Syntax error messages during insertion or modification of an entry.

Check if you missed out = sign between field name and field value.

Possible error example

@Misc {key1, year 1999 }
or @Misc {key1, year }

Correct usage

@Misc{key1, year = 1999}

Error type 7

The field name in the entry is not correct.
Or you missed out the field name in the entry.

Solution

Check if you used the correct syntax for each field in the entry.
You can not use "#%'(),={} characters in the field name.
Also space characters are not allowed.

Possible error example

@misc{key1, = 1994}
or @misc{key1, %year = 1994}

Correct usage

@misc{key1, year = 1994}

3.6.3 Field value parsing error

This can occur when users insert a new entry into the workspace or modify an entry in the workspace. The field syntax checker in Section 4.3 is responsible for detecting field value parsing errors. If the field syntax checker finds syntax errors during the insertion or modification of the entry, it will display a syntax error message window (Figure 11).

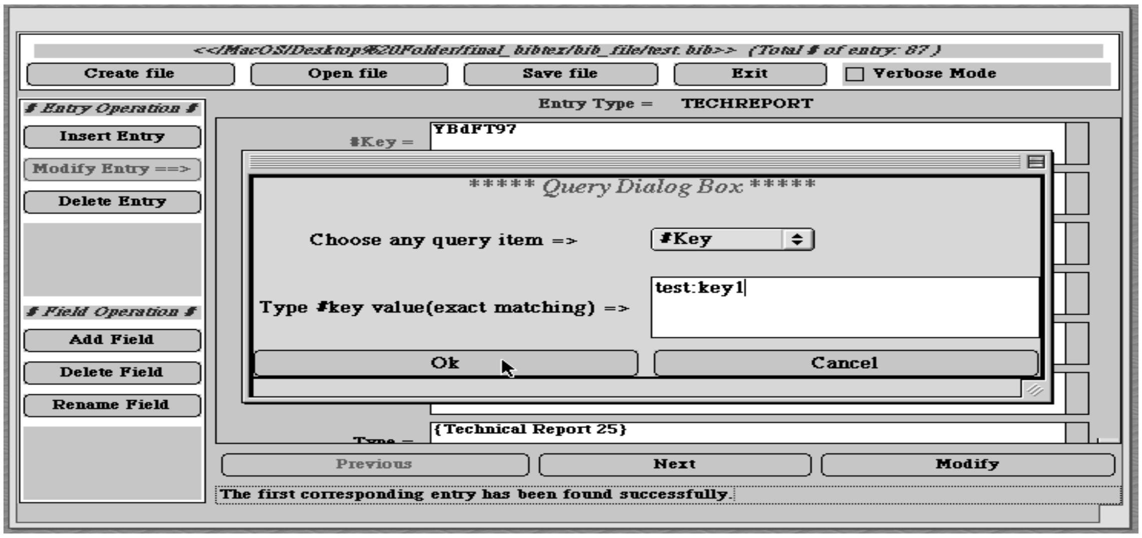

Figure 12: Search dialog box to find a corresponding entry.

Error type 1

<fieldName> field is a required field.
This can occur when the user did not type the required field value.

Error type 2

Key field value of each entry should be unique in a BIBTEX database file.
This can occur when the key value users entered is already used by another entry.

Error type 3

} was detected before { in the <fieldName> field.
For example, author = } see mu kim }
Correct usage: author = { see mu kim}

Error type 4

Mismatching {} or “” in the <fieldName> field.
For example, author = {{see mu kim }
Correct usage: author = { see mu kim}

Error type 5

A required closing character } or ” was not detected in the <fieldName> field.
For example, author = {kim see mu
    or author = “kim see mu
Correct usage: author = {kim see mu }
    or author = “kim see mu”

Error type 6

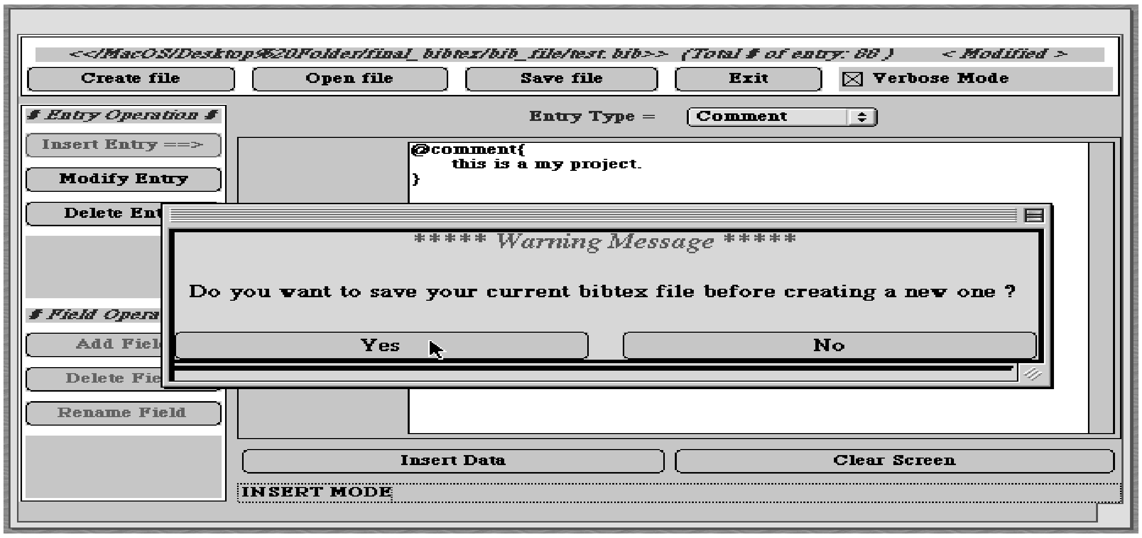

Figure 13: Warning message box before opening a different BIBTEX file.

Empty strings, {} and “”, are not allowed in the <fieldName> field.
For example, author = {} # {}
or author = “” # “”
Correct usage: author = {kim} # {see mu}
or author = “kim” # {see mu}

Error type 7

There is no string after # in the <fieldName> field.
For example, author = {kim} #
or author = “kim” #
Correct usage: author = {kim} # SEEMU

Eerror type 8

Illegal characters were detected after } or ” in the <fieldName> field.
For example, author = {kim } seemu
or author = “kim” seemu
Correct usage: author = {kim} # seemu
or author = “kim” # seemu

Error type 9

An illegal character, #, was detected in front of string in the <fieldName> field.
For example, author = # {kim }
Correct usage: author = SEE # {kim}

Error type 10

Another character, #, was detected after # in the <fieldName> field.
For example, author = {kim }# #

Correct usage: author = {kim} # SEE

Error type 11
"#%'(),={} characters(including spaces) are not allowed as a key value.

Error type 12
There is no # between string in the < fieldName > field.
For example, author = {kim} {see mu}
Correct usage: author = {kim} # {see mu}

4 Design and implementation issues

This section explains design and implementation issues, the concept of a workspace, the role of the opening parser/field syntax checker and system layout in BIBTECKER.

4.1 Java language for implementation of BIBTECKER

The main reason for using Java[6] for this project is that Java classes are platform independent, so users can easily install and use BIBTECKER on different machines without recompiling the source code. Another reason is that Java provides programmers with powerful graphics packages. Because BIBTECKER is based on user friendly graphical interfaces, users who do not have much knowledge about BIBTEX databases are able to make use of BIBTECKER very easily and comfortably.

4.2 Introduction and purpose of a workspace

BIBTECKER has a temporary memory space called the workspace for an open BIBTEX database file. When users open a BIBTEX file, all entries in the file are loaded into the workspace. Thus every entry operation such as insertion, deletion and modification can be performed in the workspace rather than on BIBTEX database files directly. This idea comes from a general modelling of standard database systems such as relational databases. In relational databases, all operations are usually performed in a rollback segment which is similar to the workspace in BIBTECKER. For example, when users insert a new record into a table in a relational database, this record will be temporarily stored in the rollback segment instead of the relational databases directly. Users can then save the contents of the rollback segment into the database permanently by using a command like submit. BIBTECKER has a similar temporary storage called the workspace. In BIBTECKER, users can use the *Save file* button to save the contents of the current workspace into a BIBTEX database file. Certainly, there are some advantages to using a workspace. Firstly, users can decide whether or not they will save the modified contents of the workspace into the BIBTEX database file or are able to save them into a different BIBTEX database file. In addition, BIBTECKER can avoid any unnecessary overhead of the file I/O operations especially in case that a BIBTEX database file is loaded from a remote machine.

4.3 Role of the opening parser/field syntax checker

In general, most people using LaTeX have their own BIBTeX databases maintained using an editor such as vi, BBedit or notepad. Therefore it is not easy for users to maintain the error-free BIBTeX databases. To make syntax checking easy, BIBTECKER provides a graphical user interface-based opening parser. When users open a new BIBTeX database file, the opening parser will check the syntax of each entry in the file. If the opening parser detects a syntax error, it will display an error message window (Figure 10) with useful information such as the position where the error occurred, possible reasons and solutions along with easy examples to follow. If no error is encountered, all entries in the BIBTeX database file will be loaded into the workspace. Currently, BIBTECKER's opening parser supports entry syntax checking for the following.

- Entry format syntax checking: It checks whether the entry has a correct entry format.
- Field name syntax checking: As with rules for abbreviations in the STRING macro, "#%'(),={} characters are not allowed in field names.
- Key value checking: It checks whether the key value is used correctly.

Some other syntax checking such as key duplication errors, field format errors or the required field missing error will be handled by the field syntax checker during the insertion and modification processes. Thus, all types of error checkings can be covered by the opening parser and the field syntax ckecker.

4.4 Layout of BIBTECKER

BIBTECKER has the following important components (Figure 14).

Workspace: This is a temporary memory area which contains all entries for the opening BIBTeX database file. Insertion, deletion and modification relating to each entry are performed in the workspace. All entries can be saved into a BIBTeX database file by using the *Save file* button on the main screen window.

Opening parser: While opening a BIBTeX database file, This checks syntax errors of each entry in the file.

Field syntax checker: When users insert a new entry to the workspace or modify the current entry from the workspace, if no syntax error is found, the insertion or modification will be processed. Otherwise, an error message window is displayed and remaining procedures will be stopped.

Query dialog window: This is the dialog box containing the searching functions. Simply by choosing one search type and putting the search value in the text field of the dialog box, users can search for a corresponding entry. If a corresponding entry is not found, the dialog box will be displayed again.

Insertion panel: Provides a user interface to insert a new entry into the workspace.

Deletion panel: Provides a user interface to delete the current entry from the workspace.

Modification panel: Provides a user interface to modify the current entry from the workspace.

5 Conclusions and possible improvements

This project introduced the new BIBTEX database management tool, BIBTECKER, which has the following advantageous features.

- It provides the following important capabilities: basic entry operations, entry and field syntax checking and various searching functions also provided by other tools such as Emacs or the bibtool package.
- It runs on any platform without recompiling the source code.
- Easy to use: Users do not need in-depth knowledge of BIBTEX databases.
- BIBTECKER offers an improved user interface through a user friendly graphical environment.
- It has powerful and varied functions such as attractive searching operations and the opening parser/syntax checker.
- It employs a concept frequently used in standard databases, called "workspace storage" in this paper.
- It provides user extendable type fields and comment maintenance.

BIBTECKER can easily be used by anyone for the purpose of management of BIBTEX databases.

However, BIBTECKER does not provide all possible functionality that a BIBTEX database management tool could provide. There are several changes that could be made to BIBTECKER to make it more powerful.

- Development of a multi-user system: Because the current version of BIBTECKER only works in single-user mode, one BIBTEX database cannot be shared among users, unlike standard database systems. Implementation of a multi-user system with support for handling shared BIBTEX databases would make BIBTECKER more powerful and flexible.
- Addition of sorting and merging functionality into BIBTECKER: It is possible to enhance BIBTECKER by supplying sorting and merging operations for entries in BIBTEX databases. These functions can be easily implemented by modifying the current BIBTECKER slightly.

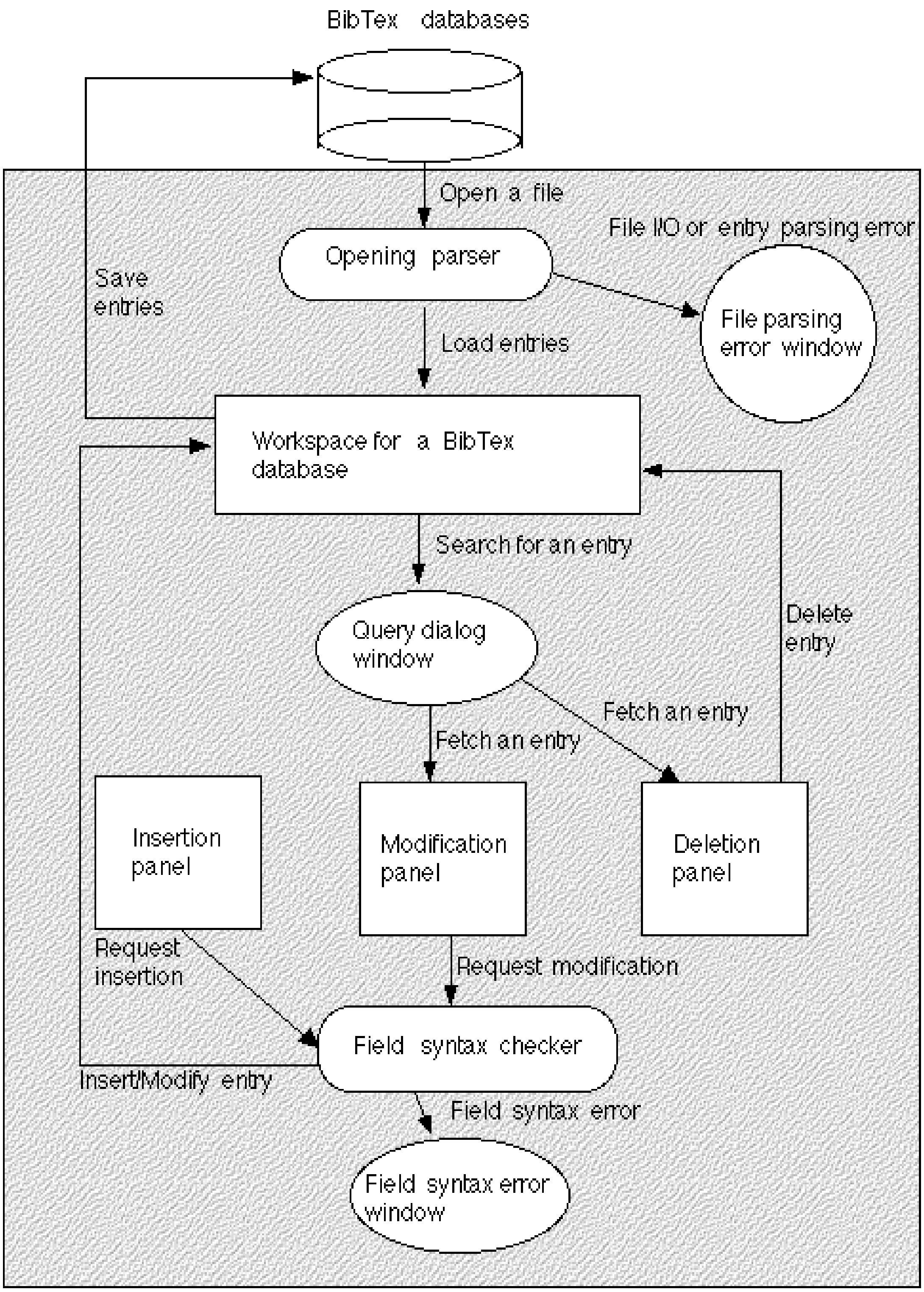

Figure 14: Layout of BIBTECKER.

- Insertion of the key value generator into BIBTECKER: In the current version of BIBTECKER, users have to specify a unique key value of an entry for the entry insertion. It would be nice to have an automatic key value generator that generated a unique key value for users. Therefore, users does not need to worry about the key duplication error in a BIBTEX database. This function can be easily implemented by modifying the current BIBTECKER slightly.

References

[1] Leslie Lamport. LATEX: *A Document Preparation System.* Addison-Wesley, 1985.

[2] Oren Patashnik. *BIBTEXing, Documentation of general BIBTEX users.* 1988.

[3] *GNU Emacs online manuals on UNIX.* eg: man emacs

[4] Helmut Kopka and Patrick W. Daly. *A guide to LATEX document preparation for beginners and advanced users.* Addison-Wesley.

[5] David Flanagan. *Java in a nutshell a Desktop Quick Reference.* O'Reilly.

[6] (in Korean) Ja Chul Koo. *Interesting Java Programming.* Hight hight, 1997.

[7] *BIBTEX bibliography tools* http://www.math.utah.edu/~beebe/software/bibtex-bibliography-tools.html

[8] *BIBTEX example* http://math.rutgers.edu/FTP_DIR/shelarch/example/index.html