

t|Database 2011

User Guide

Version 6.0.0

Professional Data Editing Software

Copyright © 2004-2010 TshwaneDJe Human Language Technology
<http://tshwanedje.com/>

Table of Contents

General Notes to the User Guide	1
Getting Started with tlDatabase: QuickStart Guide	2
Creating a New Database.....	2
Overview of the tlDatabase Interface.....	2
Entry List [1].....	3
QuickSearch Box.....	3
Tree View [2].....	4
Attributes (F1, F2) and Tools (F3, F4, F5) Window [3].....	4
Preview Area [4].....	4
Adding New Entries to the Database.....	4
Structuring Entries with the Tree View.....	5
Changing the Order of Elements.....	5
Marking Entries as 'Incomplete'.....	5
Saving Your Work.....	6
Restoring the Last-saved Version of an Entry ('Undo').....	6
Adding a Batch of New Entries, Or 'Import Wordlist/CSV'.....	6
Deleting Entries from the Database.....	6
Printing your Database.....	6
Making Regular Backups	7
Manual Backups.....	7
Automatic Backups.....	7
Offsite Backups.....	7
Changing the Interface Language (Localisation)	8
Built-in Localisation Editor.....	8
Keeping Your Software Up To Date	9
“Wide View” Mode and “High View” Mode	10
F12 Overlay Window (Larger Attribute Editing Window).....	11
Attributes (F1) and (F2)	12
Attributes (F1).....	12
“Incomplete” Checkbox.....	13
Attributes (F2) – Closed Lists with Multiple Selection.....	13
Sorted vs. Unsorted Multi-selection Lists.....	13
Editing Attributes	14
Using Text Formatting Within a Text Field.....	14
Markup Characters (Bold, Italics, Underline etc.).....	14
Smart Formatting Reversal (%b, %i vs. %B, %I - “soft” vs. “hard” bold/italics).....	15
Using Inline Elements for Text Formatting [Advanced].....	15
Configurable Keyboard Shortcuts.....	15
Inline Elements: Special <i>'Tagging'</i> Shortcut Keys [Advanced].....	17
Replace As You Type.....	17
Configuring Replace-As-You-Type [Advanced].....	17
Search (F3)	18
Overview.....	18
Search Options.....	18
“Case Sensitive”.....	18

“Whole Word Only”.....	18
“Regular Expression” Searches [Advanced].....	18
Field Breakdown.....	20
Search Filter.....	20
“Find” Tool.....	20
“Search and Replace” Tool.....	20
Searching F2 Lists with “Find” and “Search and Replace”.....	20
Format (F4).....	22
Alternative Sets of Labels for Lists.....	22
Expand Entities.....	22
Show Timestamp.....	23
Show Usernames.....	23
Filter (F5).....	24
Types of Filter Conditions.....	24
“Reveal” Filter.....	25
“Hide” Filter.....	26
Combining “Reveal” and “Hide”.....	26
Extracting or Printing a Subset of the Database.....	27
Tags.....	28
Overview.....	28
Tagging Commands.....	28
Tag Filter.....	29
Tagging in the “Find” Tool.....	29
Tagging a Range of Entries.....	29
Compare/Merge.....	30
Overview.....	30
Action Commands: “Add”, “Merge”, “Replace” and “Delete Left”.....	31
Settings.....	32
Batch Merge.....	32
Customising the Database Structure using the DTD [Advanced].....	34
Basics of Hierarchical Data Modelling in t1Database.....	34
Elements and Attributes.....	34
What is a DTD?.....	35
Element Types.....	35
Element Attributes.....	36
Element Child Relations.....	37
Element Child Relation Constraints.....	38
Attribute Lists.....	39
Multimedia (Audio and Images).....	42
DTD Templates.....	42
Customising the Alphabetic Sorting.....	43
Overview.....	43
Table-based Sorting.....	43
Configuring Table-based Sorting.....	43
Selecting Other Sort Plug-ins.....	44
Loading/Saving Sort Configurations.....	45
Styles System.....	46
Overview.....	46
Styles/formatting.....	46
Basic Formatting Options.....	46

Colours.....	47
Text Before/After Options.....	48
Group Style Properties for Multiple Elements or List Items.....	48
Automatic Numbering.....	49
Output (Display) Order.....	51
Element and Attribute Output (Display) Order.....	51
“Visible” Flag (Non-Printing Fields).....	52
Entities.....	52
Using Entities to Embed Labels within Other Fields and Further Customisation.....	52
Style Sets, Or ‘One Database, Many Dictionaries’.....	54
“Smart Styles” (Dynamically Customisable Styles) [Advanced].....	55
Importing Data.....	56
Import Wordlist / CSV (Comma Separated Values).....	56
Importing XML [Advanced].....	58
Post-Import Processing / Data Remodelling.....	58
Exporting Data.....	59
Copying To the Clipboard.....	59
Copy Entry Text.....	59
Copy Entry HTML.....	59
Exporting the Database, in Part or in Full.....	59
Text.....	59
RTF (Rich Text Format).....	59
HTML (Web Page).....	60
XML (eXtensible Markup Language).....	60
XML (Formatted).....	61
Network (ODBC / Relational Database) Support [Advanced].....	62
Configuring an ODBC Database.....	62
“Cached” ODBC (ODBC, Sped Up).....	62
Entry Locking.....	62
Locking the Database.....	63
Optimisation Tips.....	63
Commandline Options [Advanced].....	64
Watch Folders [Advanced].....	65
Lua Scripting [Advanced].....	66
Getting Started with t!Database Scripting.....	66
Lua Script Attributes.....	66
Stand-alone Scripts.....	66
API Reference Documentation, Tips, Samples and Further Information.....	67
Other Uses of Lua Within t!Database.....	67
“Smart Styles”.....	67
Lua Filters.....	67
Lua Sorting.....	68
User Management.....	69
Configuring User Logins.....	69
Deleting Users: “Delete” vs. “Purge”.....	69
Privileges System.....	69
Monitoring and Tracking Progress.....	70
User Progress Statistics.....	70

General Notes to the User Guide

tlDatabase is a software application for creating (XML-like) structured databases, and allowing them to be edited by end users a user-friendly way, i.e. the end-users do not need to work “in XML” in order to edit the database. (Setting up a database, however, may require some assistance from a user with more advanced technical abilities. TshwaneDJe can also provide type kind of assistance.)

tlDatabase is fully internationalized; thus all fields will accept **Unicode** characters (i.e. characters from any language). If you have problems displaying characters from a particular language, this is usually just a configuration problem (e.g. choosing the correct font).

tlDatabase was initially conceived as a “spin-off” from TLex, an industry-leading software environment for compiling dictionaries, used by many major publishers worldwide. Thus many examples in this User Guide stem from the world of dictionaries.

<p>Important Technical Note: The underlying database model of tlDatabase is XML-like, <i>not relational</i>.</p>

Terminology Used: The terms “element” and “attribute” stem from the terminology of the XML industry standard, on which tlDatabase is based.

Frequently Asked Questions (FAQ): For additional documentation, tips and tricks, answers to common queries, information on undocumented features, and other information for getting the most out of your software, it is suggested that you regularly check out the tlDatabase FAQ (“Frequently Asked Questions”) document, available online at: <http://tshwanedje.com/faq.html>. The FAQ also contains supplementary documentation (e.g. on new features) that has not yet made it into the User Guide.

Getting Started with tlDatabase: QuickStart Guide

Creating a New Database

To create a new database, open tlDatabase and click on “Start a new project” (or select the “File/New database” menu option). The following dialog will appear:

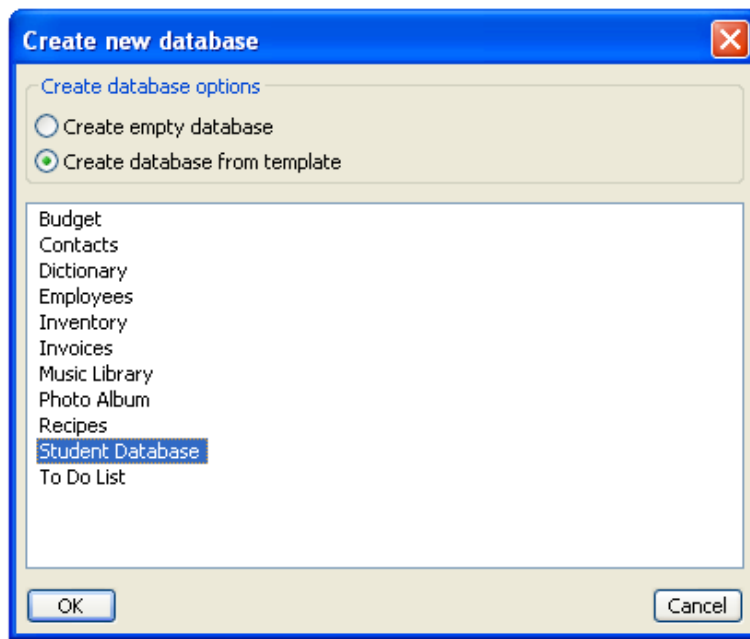


Figure 1: Creating a New Database

The simplest way to get started is to create a new database from one of the easy-to-use starter templates. More advanced users may create a blank database from scratch; this requires at least some basic knowledge of elements and attributes (see the User Guide section on customising the DTD).

Once you have selected the desired option, click “OK”. The tlDatabase editing environment will appear, allowing you to begin adding entries to your database. Before you proceed, select “File/Save” from the menu (or press Ctrl+S), and enter a filename under which to save your database. Thereafter, this database can be opened again by selecting the “File/Open: tlDatabase file” menu option and selecting this file. Thus one usually works with a tlDatabase database much like one works with ordinary documents in other applications, such as Microsoft Word.

All recent tlDatabase files that you have worked on are listed under the “File/Open recent” menu option, as well as on the “start page” (under “Tasks / Open recent”) that appears in the centre of the window when you open tlDatabase.

Overview of the tlDatabase Interface

Once you have created a new database, you will be presented with the main editing environment.

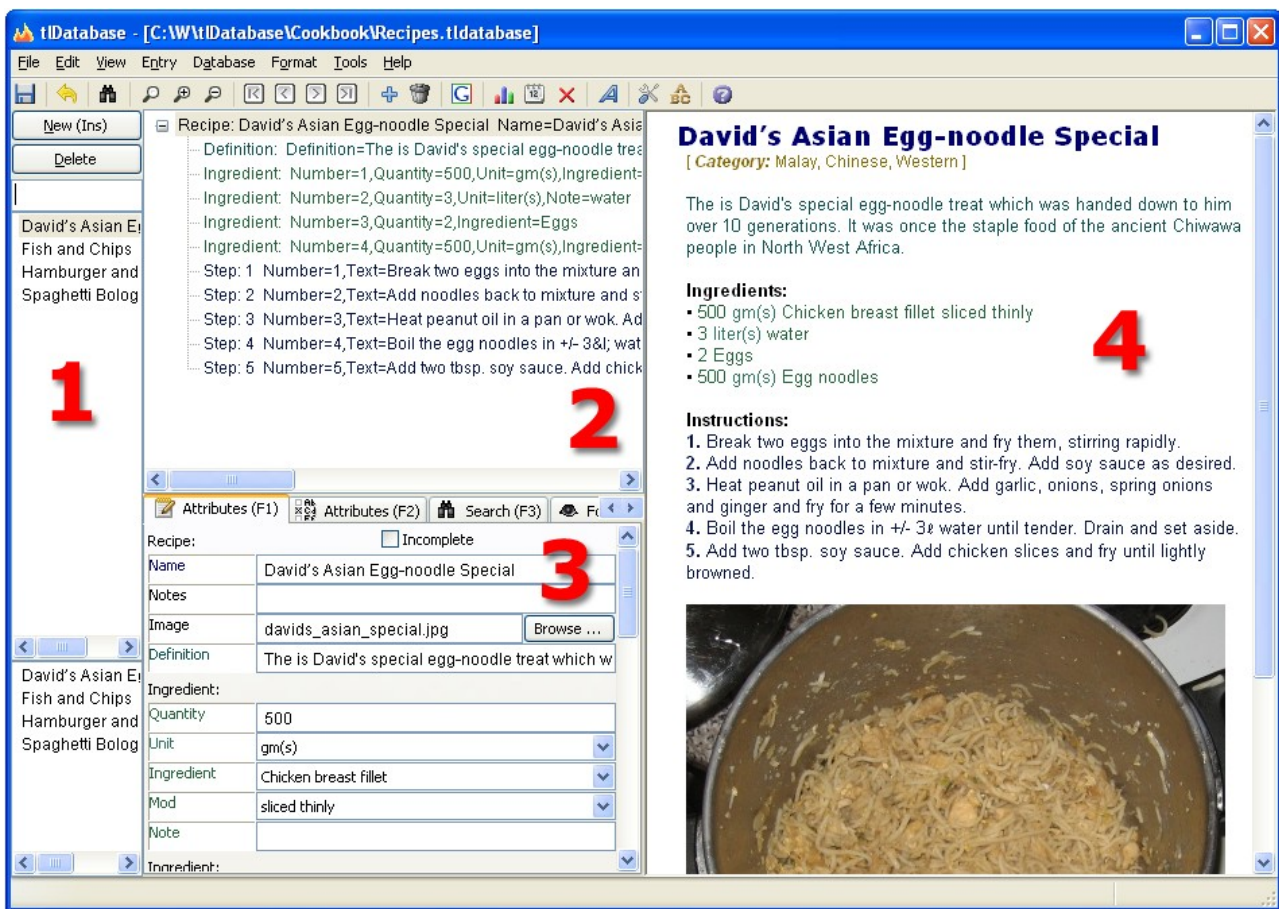


Figure 2: The tDatabase interface, showing the main editing areas

On the screenshot, the four main editing areas have been marked “1” to “4”. These are [1] the “Entry List”, [2] the “Tree View”, [3] the “Attributes and Tools window”, and [4] the “Preview Area”.

Most actual *editing* is done in [2] and [3].

Each of these windows will now be discussed briefly.

Entry List [1]

This is a scrollable list of all entries in your database. You can use this to select an entry to view or work on.

QuickSearch Box

Just above the Entry List is a “quick-search” box. When you start typing a word into this box, tDatabase will automatically select the closest matching entry in the list.

NB: The **shortcut key** “Esc” (escape) will always immediately take you to the quick-search box from almost anywhere in tDatabase.

Tree View [2]

Each entry in the database typically has some kind of hierarchical structure (for example, a recipe may consist of 'ingredients' and 'instructions', while an invoice may contain 'line items'). The Tree View is used to view or modify this hierarchical structure of an entry, e.g. to add new 'instruction' items to a recipe. Each node in the Tree View is called an **element**. Right-clicking on any element in the Tree View displays a menu with a list of editing options available for that element, such as adding child elements.

Attributes (F1, F2) and Tools (F3, F4, F5) Window [3]

This window consists of five sub-windows, which can be accessed quickly using the shortcut keys “F1” to “F5”. The first two sub-windows are used to edit the so-called **attributes** of the currently selected element in the Tree View, i.e. the actual values that are associated with the element. Thus the Tree View, which shows *elements*, is basically used to modify the *skeletal structure* of an article (e.g. adding a usage example at a certain point), while *attributes* flesh out that structure with actual *content* (e.g. the text of the usage example itself plus its citation). The “Attributes (F1)” window contains text boxes for editing text attributes of the currently selected element, while the “Attributes (F2)” window allows one to modify attributes that are constrained to selection from a defined list of values, such as a part of speech list (with the exception of single-selection lists, which also appear under “Attributes (F1)”).

“Search (F3)” activates a search function which enables text searches to be made in the entire database. “Format (F4)” allows certain settings to be modified that affect how the output will appear in the Preview Area and when exporting the data (for printing or electronic display). The filter function, “Filter (F5)”, is a kind of specialized search query that can be used to define criteria to work on or export a *subset* of the database.

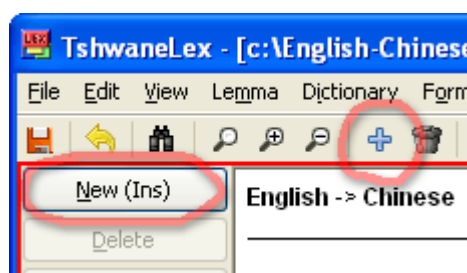
Preview Area [4]

This window displays an approximate representation of how the currently selected entry will appear in print (a so-called WYSIWYG (“what you see is what you get”) view). The Preview Area updates immediately as changes are made in the Tree View or Attributes (F1 and F2) and Format (F4) sub-windows.

Printing: Note that you can print the contents of the Preview Area (e.g. currently selected entry) by right-clicking in it and selecting “Print...” or “Print Preview...”.

Adding New Entries to the Database

The simplest way to create a new entry is to click on the “New (Ins)” button in the top left corner of the editing window, or the “+” sign in the toolbar (see image at right). **You will then be prompted for the headword/term of the entry to be added.** Enter the headword and click “OK” or press the “Enter” key. A new, empty entry will be added. To add word senses to the lemma, right-click on the “Lemma” element in the Tree View and click on “Add: Sense”. To add



a definition or a translation equivalent (TE) to the *sense*, right-click on the newly added “Sense” element in the Tree View and select “Add: Definition” or “Add: TE”, and then edit the definition or translation equivalent under “Attributes (F1)”. Alternatively, as a shortcut, one may left-click on the “Sense” element and then enter a definition or translation equivalent directly into the “Definition (NEW)” or “TE (NEW)” textboxes under “Attributes (F1)”.

You can also use the “Entry/Insert new entry” menu option to add new entries to the database, or use the “Insert” **shortcut key** on the keyboard.

Structuring Entries with the Tree View

Adding: Right-clicking on any element in the Tree View shows a list of all possible elements that may be added as a child of that element. (Underneath the separator, cascades of child elements are also available. For example, instead of first adding the child element “Sense” to a “Lemma”, followed by the addition of the child element “Example” to that “Sense”, one may select “Add: Sense::Example” to obtain the same result with a single instruction. These “multi-element cascades” are shown with double-colons “::”.)

Deleting: Elements can be deleted using the right-click “Delete” menu option, or by pressing “Delete” on the keyboard, to delete the currently selected element.

Moving: Right-click options to “Cut” (Ctrl+X) and “Paste” (Ctrl+V) elements are also available. A pair of scissors “[8<]” is shown in the Tree View during the process of cutting and pasting, and will remain visible in front of the last element pasted (to show what is currently on the clipboard).

The right-click “Cut” (Ctrl+X) and “Paste” (Ctrl+V) menu options can also be used to cut and paste elements between *different* entries.

Drag-and-Drop: It is also possible to move elements in the Tree View around by dragging them with the mouse cursor. **This is disabled by default**, but can be enabled with the “View/Tree View drag-and-drop” menu option.

Copying: A right-click menu option “Copy” (Ctrl+C) allows an element subtree to be duplicated anywhere else in the document via “Paste” (Ctrl+V).

Changing the Order of Elements

Elements on the same level in the tree hierarchy (“sibling elements”) can be moved up and down, using either the right-click “Move up” and “Move down” options, or their corresponding **keyboard shortcuts** “Ctrl+Up” and “Ctrl+Down” respectively.

Marking Entries as 'Incomplete'

Entries that require further attention may be marked as “incomplete”. The complete/incomplete status of each article can be toggled with the “Incomplete” checkbox under “Attributes (F1)”, or by pressing Ctrl+Shift+I. Incomplete entries are displayed with question marks next to them in the Preview Area.

Various import and export functions of tlDatabase make use of the “incomplete” setting. For example, when exporting data in preparation to be printed, incomplete entries are by default (this is a configurable option) excluded from the output, to help prevent you from inadvertently publishing unfinished entries.

Note that the “Filter (F5)” tool can easily be used to find and show all entries marked as incomplete at any time.

Saving Your Work

Newly created or modified entries are displayed with a “*” (or “!”) next to them in the Preview Area. This means “unsaved changes”, and appears on all new entries which have changes that have not yet been saved to disk. To save your changes, select “File/Save” (Ctrl+S). It is a good idea to do this regularly.

Restoring the Last-saved Version of an Entry ('Undo')

If, when working on an entry, you change your mind about the changes, it is possible to “restore” that entry to the *last saved version*. To do so, choose the “Edit/Restore” menu option (Ctrl+Shift+Z). This command will only have effect when there are unsaved changes in your entry, i.e. when a “*” appears next to that entry, and when the entry has been saved to disk at least once before.

Adding a Batch of New Entries, Or 'Import Wordlist/CSV'

It is possible to add many entries to the database at once from a wordlist (one word per line), or from a CSV (Comma Separated Values) file from spreadsheet software such as OpenOffice Calc or Microsoft Excel. To do this, use the “File/Import/Wordlist or CSV (Comma Separated Values)” menu option. See the chapter “Importing Data” for more information.

Deleting Entries from the Database

To remove an entry from the database, select that entry the Entry List, and click the “Delete” button near the top left of the editing window. You can also use the keyboard shortcut “Ctrl+Delete”, or you can use the dustbin (trashcan) icon in the toolbar.

Printing your Database

When you wish to print your database, the simplest is to export it to RTF (Rich Text Format) and open it in a word processor such as OpenOffice or Microsoft Word. Apply any final formatting changes (e.g. adding columns) in the word processor before printing. To export to RTF format, select the “File/Export/RTF (Rich Text Format)” option from the menu. An “Export options” dialog will appear (the meaning of the various options will be explained in the course of this User Guide). Click “OK”. You will be prompted to enter a filename for the output RTF file.

Note that by default, entries that have been marked as “Incomplete” (i.e. those that are displayed with question marks) will not be exported. This can be overridden, if desired, in the “Export options” dialog box, with a further option to hide the incomplete marker “?”.

Making Regular Backups

Manual Backups

For all database projects, it is **crucial** to make regular backups. t!Database provides features to help make it easy to quickly create a backup of your current database. To create a backup copy of your database at any time while working on it, select the “File/Create a backup” menu option. This will save a full “snapshot” of your database into a backup folder. (Note that, by default, this will typically be saved on the same computer as the file you are working on. This will thus not protect you if that computer is damaged or stolen; it is crucial to also make **offsite backups**, discussed below.)

The folder in which these backups are saved can be configured via the “Tools/Options” menu option.

Automatic Backups

In addition to the manual “File/Create a backup” menu option, t!Database also has an *automatic* backup system that automatically saves a backup copy of the currently open database into the backups folder at a configurable time interval. By default, this is set to one hour (“60” minutes). Whether or not to perform automatic backups, and the interval in minutes, can be configured under the “Tools/Options” menu (on the “settings” tab labelled “General”). Every ‘interval’ the previous backup is overwritten with the latest backup, but one backup each per twenty-four hours is kept.

All users are also very strongly encouraged to periodically make *offsite backups*, discussed next.

Offsite Backups

The backups created manually using “File/Create a backup” as well as the “automatic backups” will by default be saved to the same computer on which your database is stored (although you may configure this to save to a drive on another computer on the LAN (Local Area Network)). However, to protect against disastrous eventualities such as hard disk failures, theft, lightning, earthquakes, fires, floods, nuclear incidents and so on, it is crucial to also have a policy for creating *offsite backups* – that is, backups that are stored at least in a *different building* to the one in which the computer being used to compile the database is (and for serious projects, in a different city). This could be as simple as regularly writing a copy of your database file to a CD or flash disk and taking it to someone's home, or e-mailing a copy of the database to a colleague. You may wish to consider further protecting offsite backups by storing them in a safe.

It is recommended that you make an offsite backup at least once every two weeks.

Changing the Interface Language (Localisation)

The interface language can be selected via “Tools/Options”, under “Language”.

Built-in Localisation Editor

tlDatabase includes the tools to change and edit its own interface language. These can also be accessed via “Tools/Options” under “Language”. This tool is fairly straightforward, and can be used to create new interface languages from within the software itself, by clicking on the “New” button, then using “Save as” when the localisation editor appears. The localisation editor will display an interface showing the original English strings, and a box where you can enter the translated string. Each string is also rooted to a “key” value, which looks something like “MENU_FILE” - this gives a clue as to where it is used in the software, e.g. all “MENU_” strings appear in the application main menu.

The 'Apply' button can be used at any time while translating to apply the new translations immediately (for some of them, however, a restart of tlDatabase is still required).

When you click “Save”, the translated strings are saved in a “.lang” type file under a folder that will be something like “c:\Program Files\tlDatabase\Data\Catalogs”. This file can be distributed to other machines by just copying it into that folder. (It is in fact a simple text file.)

There are a few “special” characters and strings relating to the localisation:

& Appears in front of the character that will typically become a shortcut key to access the command, e.g. “&File” to make “Alt+F” open the “File” menu (in Windows, these may be underlined).

\t Indicates a “tab” character. This must typically appear in menu items between the command name and the shortcut key, e.g. “&Save\tCtrl+S” for MENU_FILE_SAVE. You enter a tab character by pressing “Ctrl+Tab”.

\n Newline.

%s Indicates that the software will, at run-time, substitute this with a string value, e.g. “Enter new text for label \"%s”.

%d Indicates that the software will, at run-time, substitute this with a numerical value.

Note that the relative ordering of %s and %d markers *within one string* must remain the same.

Keeping Your Software Up To Date

TshwaneDJe continually releases new “maintenance updates” for tlDatabase. These updates contain important and useful improvements and bugfixes, and it is strongly recommended that you keep your software up to date. You can check if there is a newer release available for your version of the software by using the “Help/Check for updates” menu option. This will open your web browser to a site that tells you if an update is available, and if so, tells you where and how to download it.

Maintenance updates (not to be confused with actual version *upgrades*) are free. When you install a maintenance update, re-activation is not required.

“Wide View” Mode and “High View” Mode

While working, it may be desirable to change the work area of the Attributes and Tools sub-windows (F1 to F5) to cover more horizontal space. With the “View/Wide Tools window layout” menu option (Ctrl+Alt+L), the Attributes and Tools sub-windows (F1 to F5) can be widened to cover most of the width of the tDatabase window. This is illustrated for the compilation of encyclopaedia entries in the screenshot below:

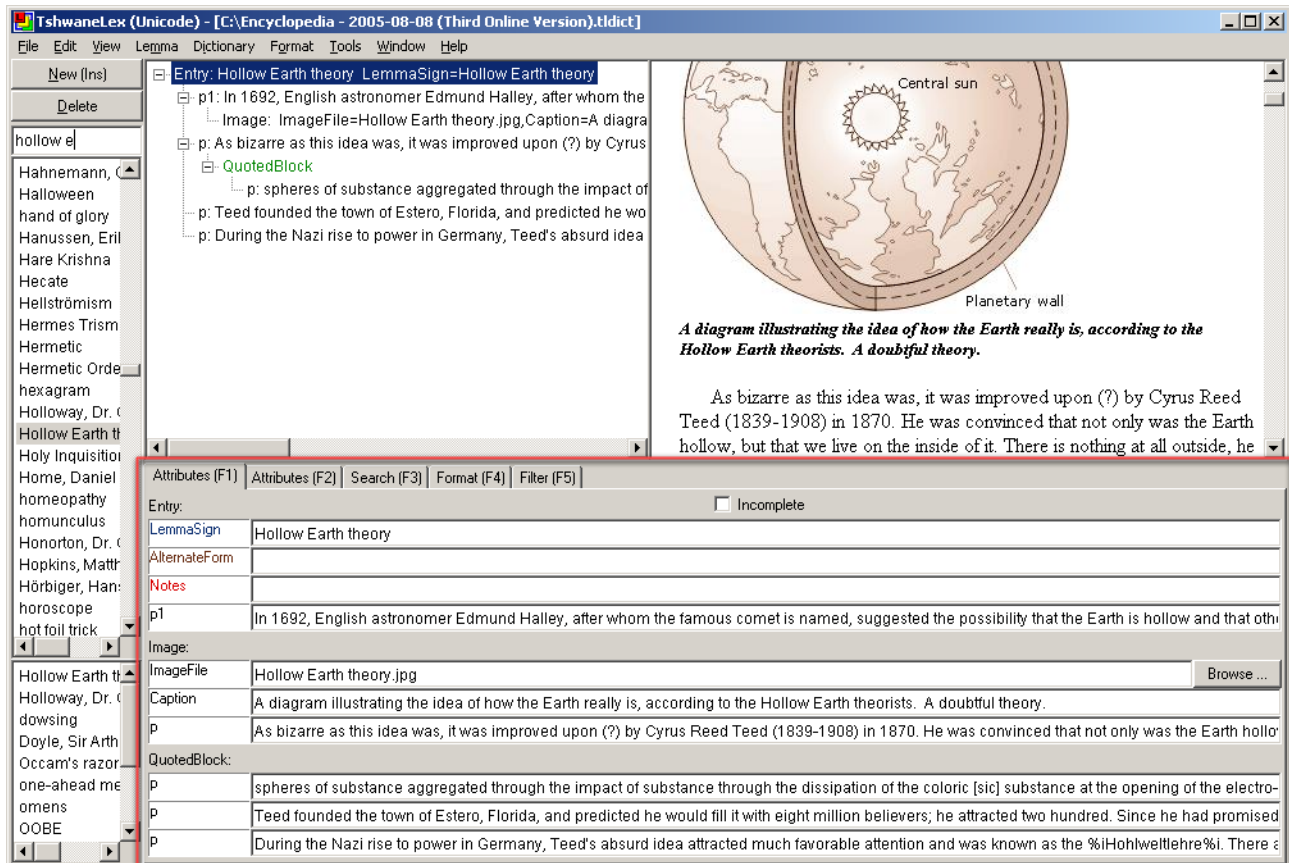


Figure 3: “Attributes (F1)” in wide view window layout (Ctrl+Alt+L) for James Randi’s Encyclopedia [Data online at: <http://randi.org/encyclopedia/>]

In other cases one may wish to see more “vertical” space, so as to for instance work with more input boxes under “Attributes (F1)” at a single glance. With the “View/Toggle Tree View” menu option (Ctrl+Alt+T), the Tree View can (momentarily) be hidden. In the screenshot below, for example, the high view window layout has been enabled for the compilation of an eleven-lingual AIDS terminology list.

female condom *A sheath which a woman inserts into her vagina before sexual intercourse to prevent pregnancy and as protection against sexually transmitted diseases.*

IsiZulu: ikhondomu labesifazane [female condom - isiZulu.mp3](#)
Ikhondomu esetshenziswa abesifazane uma beya ocansini ukuze bavikele ukukhulelwa futhi bazivikele ezizweni.

IsiXhosa: ikhondom yabafazi [female condom - isiXhosa.mp3](#)
Isigqumathelo athi umfazi asisebenzise xa alalanayo ukuzikhusela ukuba angamithi okanye angangenwa zizifo ezosulela ngokulalana.

SiSwati: ikhondomu yalabasikati [female condom - siSwati1.mp3](#)
Lishubhu lelifakwa ngulomsikati enhlunwini ngembi kwakuya emacasini kuvimbela kukhulelwa nekuvikela tifo.

SiSwati: ifemidomu [female condom - siSwati2.mp3](#)

IsiNdebele: ikhondomu yabengubo [female condom - isiNdebele.mp3](#)
Iraba efakwa esifazini ngaphambi kobana kuyiwe emsemeni, ngomnqopho wokukhondela ukuba sidisi nofana amalwelwe wemsemeni.

Setswana: khondomo ya basadi [female condom - Setswana.mp3](#)
Sebipo se mosadi a se dirisang ka nako ya thobalano go thibela pelegi le malwetse a a fetisiwang ka thobalano.

Sesotho sa Leboa: khondomo ya basadi [female condom - Sesotho sa Leboa.mp3](#)
Sethibo seo mosadi a se aparago ka bosading pele ga thobalano go thibela go ima le tshireletso kgahlanong le malwetši.

Sesotho: khondomo ya basadi [female condom - Sesotho.mp3](#)
Sesireletsi seo mosadi a se kenyang ka nako ya motaba ho thibela ho ima le ho itshireletsa kgahlano le mafu.

Tshivenda: khondomu ya vhasadzi [female condom - Tshivenda.mp3](#)
Tshitsireledzi tshine mufumakadzi a tshi ambara musi a tshi lalana na munna u thivhela u vnhifa muvhilini hu sa ŋoŋeho na malwadze.

Xitsonga: khondomu ya vavasati [female condom - Xitsonga.mp3](#)
Xisihhelelo lexi wansati a xi nghenisaka eka xirho xa yena xa mbeleko loko a nga si ya emasangwini ku sivela ku biha emirini kumbe mavabyi ya le masangwini.

Afrikaans: vrouekondoom [female condom - Afrikaans.mp3](#)
'n Skede wat 'n vrou in haar vagina plaas voor seksuele omgang om

Figure 4: “Attributes (F1)” in high view window layout (Ctrl+Alt+T) for an eleven-lingual AIDS terminology list [Data: © Department of Arts & Culture, South Africa]

Note that although the wide and high view window layouts were illustrated for the “Attributes (F1)” sub-window, these views are also available for all other sub-windows (F2 to F5).

F12 Overlay Window (Larger Attribute Editing Window)

In addition to the “wide” tools window layout, another option to gain more editing space for editing the content of text boxes is the **F12 overlay window**. While in an “Attributes (F1)” text box, pressing F12 will pop up a larger window for editing the given attribute. Once done, pressing F12 closes the window again. While the F12 window is open, you can select any “Attributes (F1)” window, and the F12 window will automatically switch to editing that attribute. This window also allows *newline characters* to be entered in a more intuitive way.

The F12 window also works in various text edit boxes elsewhere in tIDatabase. (This can be useful particularly when editing certain types of text values, such as a Lua script.)

Attributes (F1) and (F2)

Attributes (F1)

The majority of a database's contents are **typed** into freely editable text boxes under “Attributes (F1)”. In addition to these text boxes, one also edits *drop-down closed list* attribute values here. Drop-down lists allow *only one* possible value to be selected at a time. In the screenshot below, the value of the “Country” attribute in a contacts database is being selected from a drop-down list.

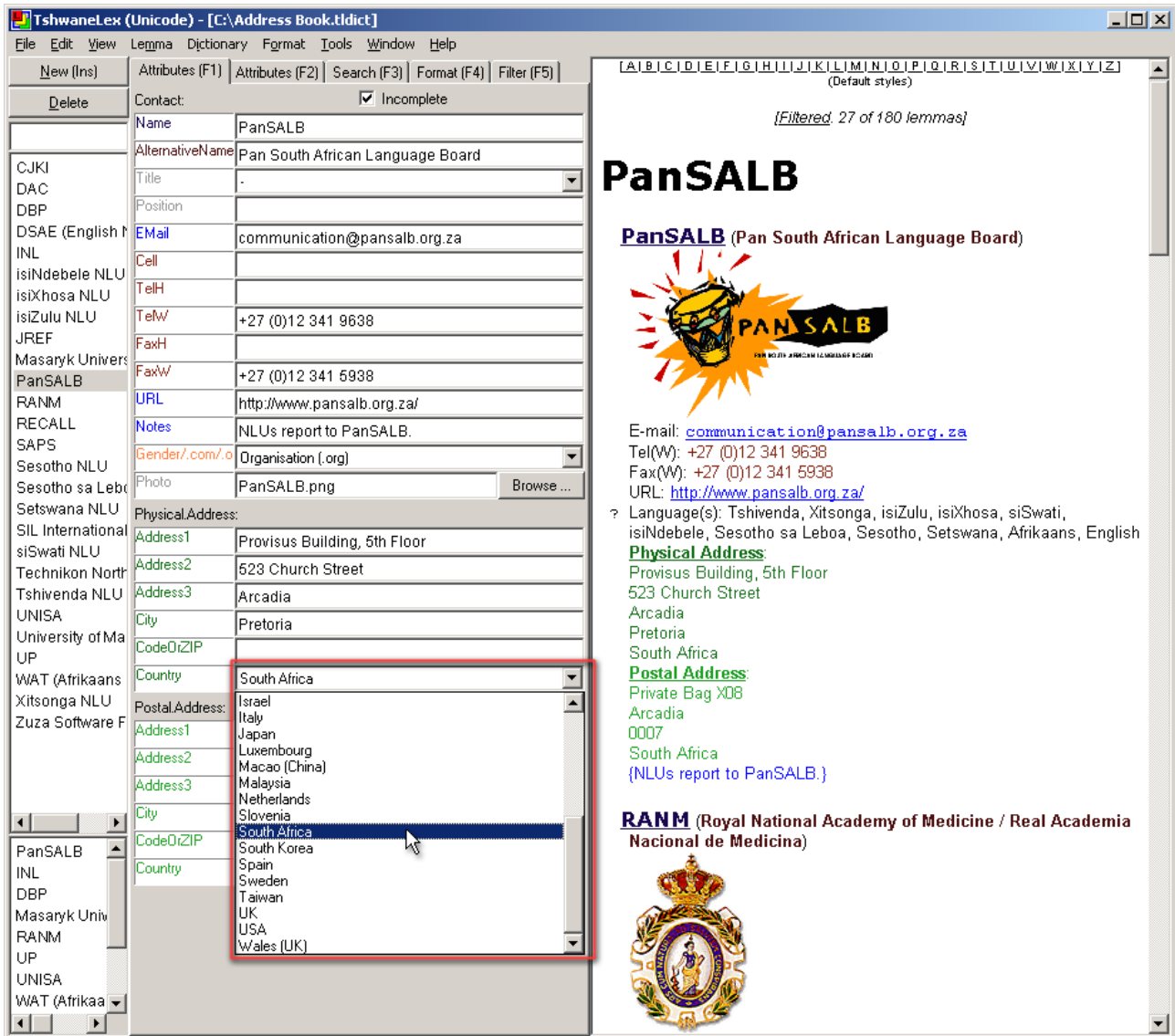


Figure 5: Selecting an attribute value from a drop-down list under “Attributes (F1)”, in an address book

The way in which such attribute lists are set up is discussed in the section on Attribute Lists, in the chapter on Customising the Database Structure using the DTD.

By default, the attribute labels under F1 are displayed in the same colour as the colours for those attributes in the article Preview. This can be disabled with the “View/Use colours in Attributes window” menu option.

“Incomplete” Checkbox

Note that in the previous screenshot, the Contact “PanSALB” has been marked “Incomplete”, hence the question mark which appears on the left of the entry in the Preview Area. See the section on 'Marking Entries as Incomplete' for more information.

Attributes (F2) – Closed Lists with Multiple Selection

The “Attributes (F2)” editing area displays attributes whose values can be entered by simply **ticking** one or more values from a centrally defined list, as shown in the screenshot below.

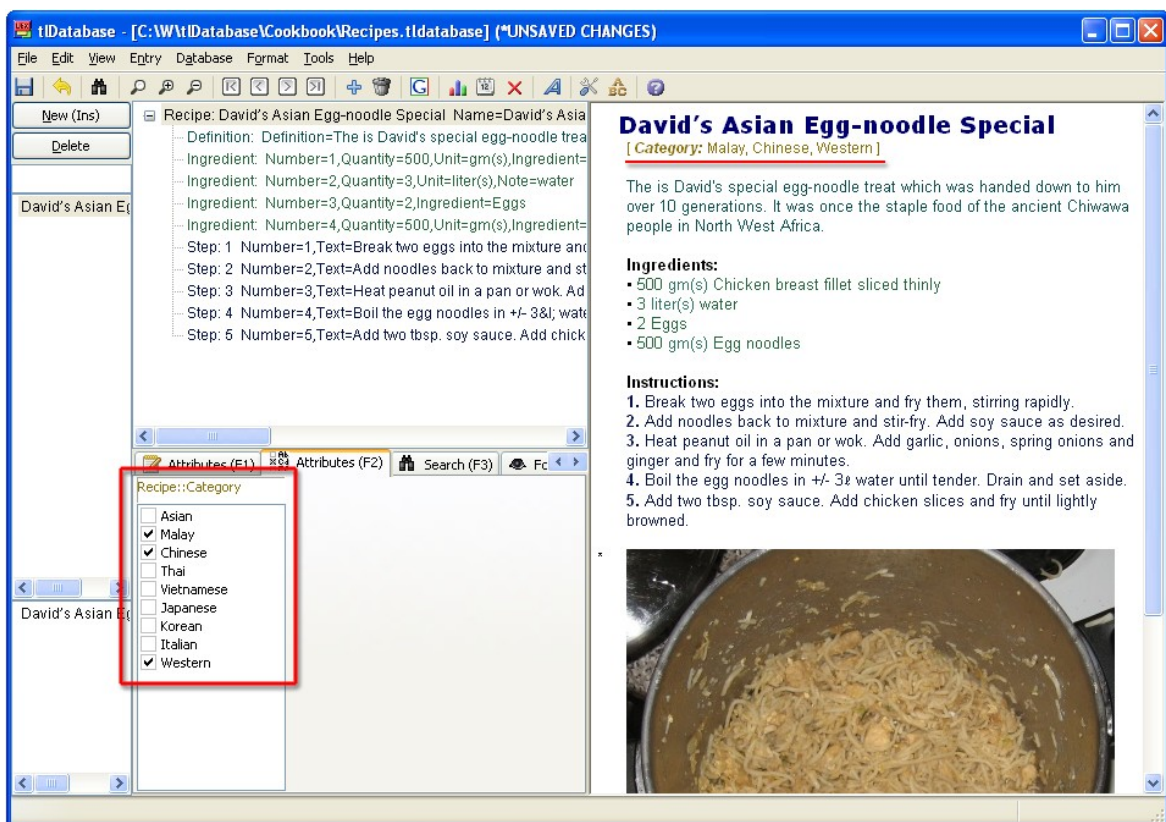


Figure 6: Selecting multiple attribute values from a closed list under “Attributes (F2)” (in this case recipe categories).

Sorted vs. Unsorted Multi-selection Lists

Under F2, two types of multi-selection attribute lists may appear: “multiple, unsorted” and “multiple, sorted”. “Unsorted” attribute list values will be displayed in the output in the same order as the user ticked those values; “sorted” will always be displayed in the output in the same order the list items appear in the list. The way in which these attribute lists need to be set up is discussed in the section on 'Attribute Lists' in the chapter on Customising the Database Structure using the DTD.

Editing Attributes

Using Text Formatting Within a Text Field

Note that general formatting issues are discussed in the chapter on the Styles System; this section deals with overriding formatting *within* attributes.

Markup Characters (Bold, Italics, Underline etc.)

tlDatabase provides special **formatting markup characters** (“percentage markup”) that may be used to specify formatting styles such as bold, italics and underline within a *subsection* of an attribute. These are specified by entering a “%” symbol followed by the markup character for the desired formatting style, e.g. “b” for “bold”. For example, if one wants the following to appear in the output:

This example demonstrates the use of **bold** and *italics* within an attribute.

Then one would enter the following:

This example demonstrates the use of %bbold%b and %iitalics%i within an attribute.

Following is a full list of the markup characters available in tlDatabase, as well as corresponding shortcut keys which may also be used to enter them:

Markup Character	Meaning	Shortcut Key
%b	Bold	Ctrl+B
%B	Bold	–
%i	<i>Italics</i>	Ctrl+I
%I	<i>Italics</i>	–
%u	<u>Underline</u>	Ctrl+U
%r	Superscript (raise)	–
%l	Subscript (lower)	–
%k	Strikethrough (strikeout)	–
%s	Small caps	–
%n	New line	–

In order to generate an actual “%” symbol in the output, use “%%”.

All of these markup characters, except for “%n” (New line), must appear in *pairs* that enclose the text being formatted. tlDatabase will display a warning if a markup character has not been “closed”. Instead of physically typing the %b, %i and %u markup characters on either side, one can also select (highlight) the section one wants to markup first, followed by pressing %b, %i or %u respectively.

Note that markup characters can also be used within the Styles System. See the chapter on the Styles System for more information.

Smart Formatting Reversal (%b, %i vs. %B, %I - “soft” vs. “hard” bold/italics)

For the markup characters “%i” and “%b”, if the text surrounding the marked-up text is *already* italics or bold, respectively, then the formatting will revert to non-italics (Roman) or non-bold. For example, if one has the following text in an attribute:

As in Orwell's %iAnimal Farm%i, however, some people are “more equal than others”

Then this would ordinarily be output as follows:

As in Orwell's *Animal Farm*, however, some people are “more equal than others”

However, if the entire attribute is being output in italics, due to the Styles, then it would be output as follows:

As in Orwell's Animal Farm, however, some people are “more equal than others”

If you wish to “force” the marked-up text to *always* be italics or bold, regardless of the style of the surrounding text, then use the uppercase versions of the markup characters, “%I” or “%B”, as in the example below:

%IHomo sapiens%I is a separate species from Neanderthals and other hominids

Using Inline Elements for Text Formatting [Advanced]

An alternative to percentage markup for formatting, that is a more closely and correctly XML-based solution, is to use “inline elements” (PCDATA, or “parsed character data”). In this case, formatting for bold and italics may take the form of XML element tags; thus for example, instead of “%i”, you might use “<i>” and “</i>” opening and closing tags for italics:

As in Orwell's <i>Animal Farm</i>, however, some people are “more equal than others”

Inline elements have certain advantages and are in many ways more powerful (and are more standard), but also have some drawbacks/limitations. They are explained in more detail in the online tlDatabase FAQ (Frequently Asked Questions: <http://tshwanedje.com/faq.html>).

Configurable Keyboard Shortcuts

In tlDatabase, keyboard shortcuts, also known as macros, can be used to type in characters (with or without diacritics), or symbols, that are not available on your keyboard. By extension, such shortcuts can also be used to type in entire text strings. To create those shortcuts, go to the “Tools/Options/Keyboard shortcuts (macros)” menu option.

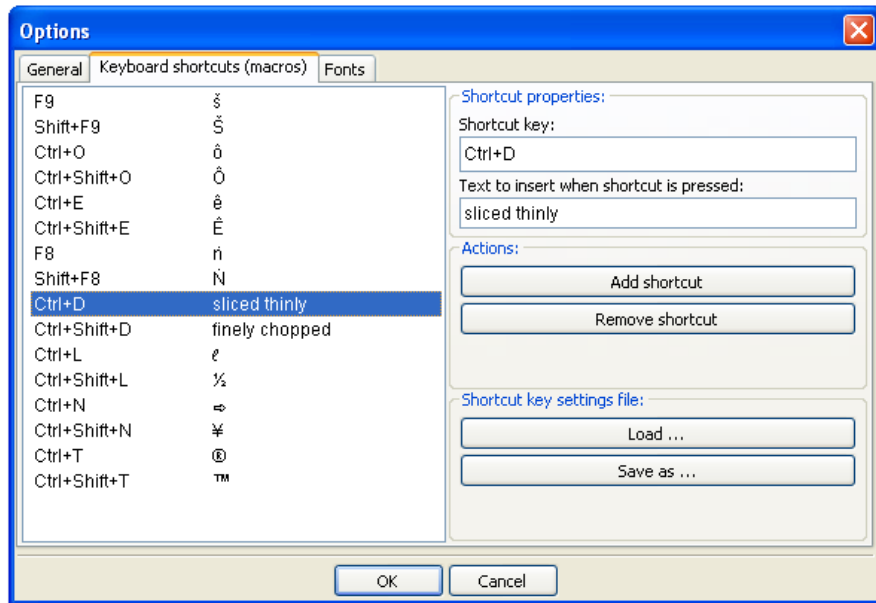
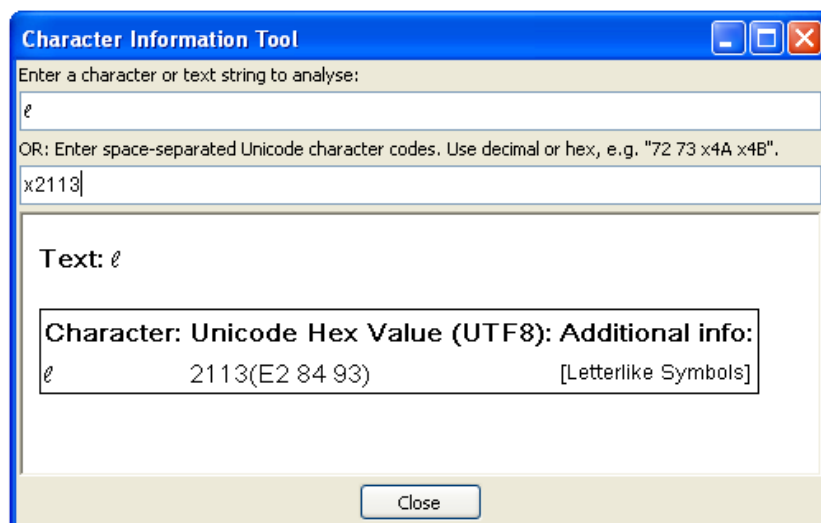


Figure 7: Configuring keyboard shortcuts (via Tools/Options)

From the above screenshot, one for example sees that when pressing the shortcut key “Ctrl+L” the “liter” symbol will be output. Other text strings that might occur frequently in your database (e.g. “sliced thinly” for a recipe database) can also be mapped to shortcut keys.

It is suggested that “special characters” such as the liter symbol are first selected elsewhere, e.g. in a word processor, using the “Insert/Special Character...” (OpenOffice.org) or “Insert/Symbol...” (Microsoft Word) menu option. It is important to choose a Unicode font in the word processor, in order for the special character to be “future proof”. One can then copy (Ctrl+C) the character from the word processor, and paste (Ctrl+V) it into the “Text to insert when shortcut is pressed” field of the dialog shown above.

[Advanced] If one knows the hexadecimal value of the character (in this case “2113”), one can also use the built-in “Tools/Character Information Tool” of t!Database to “generate” the character, as shown in the screenshot below. The desired character may then be copied from this dialog to the clipboard.



Note that, while in the “Shortcut key” input box of the keyboard shortcut configuration dialog, you have to *press the shortcut key* itself. Also, when choosing the shortcut key, try not to use a shortcut that is already in use for one of the t!Database commands (the shortcut key will not work).

New keyboard shortcuts can be added (“Add shortcut”), while existing ones can be removed (“Remove shortcut”).

It is also possible to save a set of keyboard shortcuts to a file (“Save as ...”), and to load an existing set into t!Database from a file (“Load ...”).

Inline Elements: Special *<i>*'Tagging'*</i>* Shortcut Keys [Advanced]

Special "tag" shortcut keys can be created under "Tools/Options/Keyboard shortcuts (macros)" that make tagging data with inline elements under "Attributes (F1)" far more convenient and user-friendly than typing out tags manually. This involves creating a shortcut key with the following format for "Text to insert when shortcut is pressed":

\$TAG\$:tagname

For example:

\$TAG\$:b

Pressing this shortcut key in an "Attributes (F1)" box will then automatically 'intelligently' output either an opening "" or closing "" tag as appropriate, or if some text is selected, surround the selected text with a pair of opening and closing tags.

Replace As You Type

In addition to the keyboard shortcuts, some “special characters” can also be typed in as pre-prepared “replace-as-you-type” combinations. For example, if you enter “(c)” it will automatically be replaced by the actual copyright symbol, “©”. If you enter “\+”, it will be replaced by the symbol “±”. Various other combinations exist, including combinations for entering phonetic symbols. To see a complete list of the available “replace-as-you-type” combinations, access the “Tools/Show replace-as-you-type help” menu command. (This can be printed by right-clicking in the help window and selecting “Print...”.)

By default, “replace-as-you-type” is enabled. This setting can be toggled with the “Tools/Enable replace-as-you-type” menu command.

Configuring Replace-As-You-Type [Advanced]

The default “DefaultReplaceAsYouType.rayt” configuration file can be found in the application data folder (typically “C:\Program Files\t!Database\Data”). It is possible to add your own replace-as-you-type combinations by creating a text file with the extension “.rayt” and saving it to this same folder. Such text files can be prepared with any basic text editor, such as Notepad. Unicode format text files are also supported. Each “replace-as-you-type” entry is entered on a new line simply as the string to be typed, followed by a comma, followed by the text to replace the string with, e.g. “(c),©”. After copying your file to the relevant data folder, restart the application for the changes to take effect. Any number of separate .rayt files may be placed in this folder; all will be loaded.

Search (F3)

Overview

The “Search (F3)” tool allows you to perform a full text search on the entire database.

Note that if a filter is currently applied, the search function will only search the subset of entries currently revealed by the filter (see the chapter on the “Filter (F5)” tool for more information).

Note there are also 'Find' and 'Replace' tools under the 'Edit' menu which are similar, though slightly different, to the 'Search (F3)' tool.

Search Options

“Case Sensitive”

When the “Case sensitive” option is ticked, the search function will regard the case of a letter (i.e. uppercase/lowercase) as important while searching, and will not match results where the case differs from the search string.

“Whole Word Only”

When the “Whole word only” option is ticked, the search function will only return articles where the search string occurs as a whole word, and not those where the search string appears as part of a word. This difference is illustrated in the screenshots below:

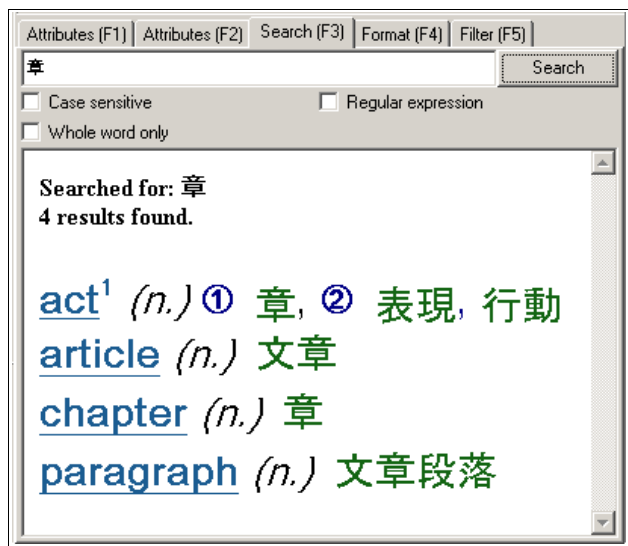


Figure 8: Searching with partial word matching in a bilingual English – Chinese dictionary
[Data: © Lorraine Liang]

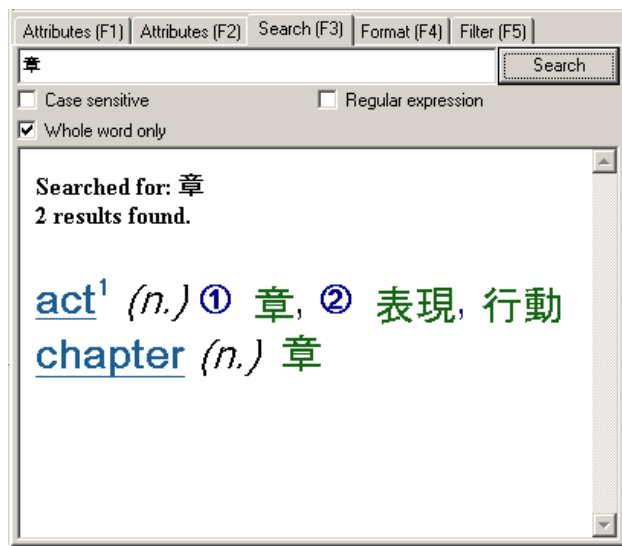


Figure 9: Searching with “Whole word only” matching in a bilingual English – Chinese dictionary
[Data: © Lorraine Liang]

“Regular Expression” Searches [Advanced]

Ticking the “Regular expression” option allows you to perform “regular expression” searches. This is a special “language”, with its own syntax, for creating more complex search queries.

The basic purpose of regular expressions is to allow you to create a single search query string that can match *multiple* possible strings in the database text. As a simple example, one can search for occurrences of either “gray” or “grey” (two alternative spellings of the same word) by using the search string “gray|grey”. The vertical line “|” is a regular expression construct that denotes “OR”.

Broadly speaking, regular expressions are an industry standard, and many detailed references are available on the World Wide Web, such as the following recommended Wikipedia article: http://en.wikipedia.org/wiki/Regular_expression. Thus, only a brief overview will be provided here. Note that there are different varieties of regular expressions; we use the so-called “Perl syntax”.

Following are a few more examples of regular expression constructs:

Single-character wildcard: “.” can be used to indicate “one of any character”, thus the search string “li.e”, will match “like”, “line”, “life”, “lime”, “lice”, etc. (but not “lie”).

Multi-character wildcard: “.*”

Beginning of field: “^” can be used to indicate “beginning of field”, thus allowing you to search for attributes that *start* with a given text string, e.g. “^to”.

End of field: “\$” can be used to indicate “end of field”, thus allowing you to search for attributes that *end* with a given text string, e.g. “cat\$”.

Empty field: “^\$” (meaning, match beginning of field followed immediately by end of field)

One or more: “+” denotes that the preceding character may be repeated one or more times, e.g. “lo+se” will find occurrences of both “lose” and “loose”.

Zero or more: “*” denotes the the preceding character may be repeated zero or more times.

Character group: [...], e.g. “gr[ea]y” will match both “grey” and “gray”.

Character range: [...-...], e.g. “[0-9]” will match any numerical digit from 0 to 9, and “[a-z]” will match any character in the 26-character English alphabet.

The regular expression search string “^[a-z][a-z][a-z]” would thus, for example, find all attribute values that are just a single three-letter word.

Inverse (not): [^...], e.g. “[^a]” will match any character other than “a”, “[^]” will match any character other than space, and “[^aeiou]” will match any character that isn't a vowel.

Beginning-of-word: \<

End-of-word: \>

If you wish to actually search for one of the characters that are used as special regular expression query characters, you can do so by “escaping” the character – this is done by preceding it with a “\” character. Thus “\.\$”, for example, will find all articles where an attribute actually ends with a “.”.

Field Breakdown

In the search results window, the displayed number of matches and matching entries is clickable; clicking on this will display a breakdown of the number occurrences in particular fields (along with corresponding filters).

Search Filter

In the search results window, clicking on “Filter +” will create a filter that reveals only the entries that contain a match for the search query. Clicking on “Filter -” will create the opposite filter, revealing only the entries that *do not* contain a match for the search query. See the chapter on the “Filter (F5)” tool for more information.

“Find” Tool

In addition to the search functionality available under F3, a full database “find” tool is accessible from the “Edit” menu. This can be invoked with “Ctrl+F”.

The “Find” tool includes the same basic options as Search (F3), namely “whole word only”, “case sensitive”, “regular expression”, and a “Fields” tab that allows you to restrict the search to particular attributes only. In addition to this, it also has the following four commands:

- “Tag all”: Tag all matching entries. This adds to any current selection of tagged entries.
- “Untag all”: Untag all matching entries. This does not clear existing tags from non-matching entries.
- “Filter +”: Create and apply a “Reveal” filter that displays only the entries containing matches.
- “Filter -”: Create and apply a “Hide” filter that displays all entries that do not contain matches.

See the relevant sections on tagging and filters for more information.

“Search and Replace” Tool

A full database “search and replace” tool is also accessible from the “Edit” menu. “Search and Replace” can be invoked with the shortcut key “Ctrl+H”. An alternate “Find” dialog can be invoked with “Ctrl+F”.

Additional advanced “search and replace” options allow you to do replacements without touching the user and date/time tracking fields - these should generally be used only if you know what you are doing.

Searching F2 Lists with “Find” and “Search and Replace”

You can search for multiple selections in an attribute by using a “,” (NB: without space) as the search string, e.g. “noun,verb” (NOT “noun, verb”). You can also do “intelligent” replaces; e.g. replacing “noun,verb” with just “noun” will clear the “verb” checkboxes. Replacing with “noun,pronoun” will clear the “verb” checkbox and tick the “pronoun” checkbox (providing it exists in the list).

Note that this will not automatically add new items to the list if the replacement string is not in the list; any desired strings must be added manually in the DTD editor.

Format (F4)

In the sub-window “Format (F4)”, eight different types of general formatting settings may be modified. These settings affect how the output will appear in the Preview Area and when exporting the data (for printing or electronic display). In each case the Preview Area updates immediately as changes are made. In the screenshot below the settings have been chosen in such a way that they contrast the settings used in the screenshots so far, and/or in the screenshots shown in the following chapters.

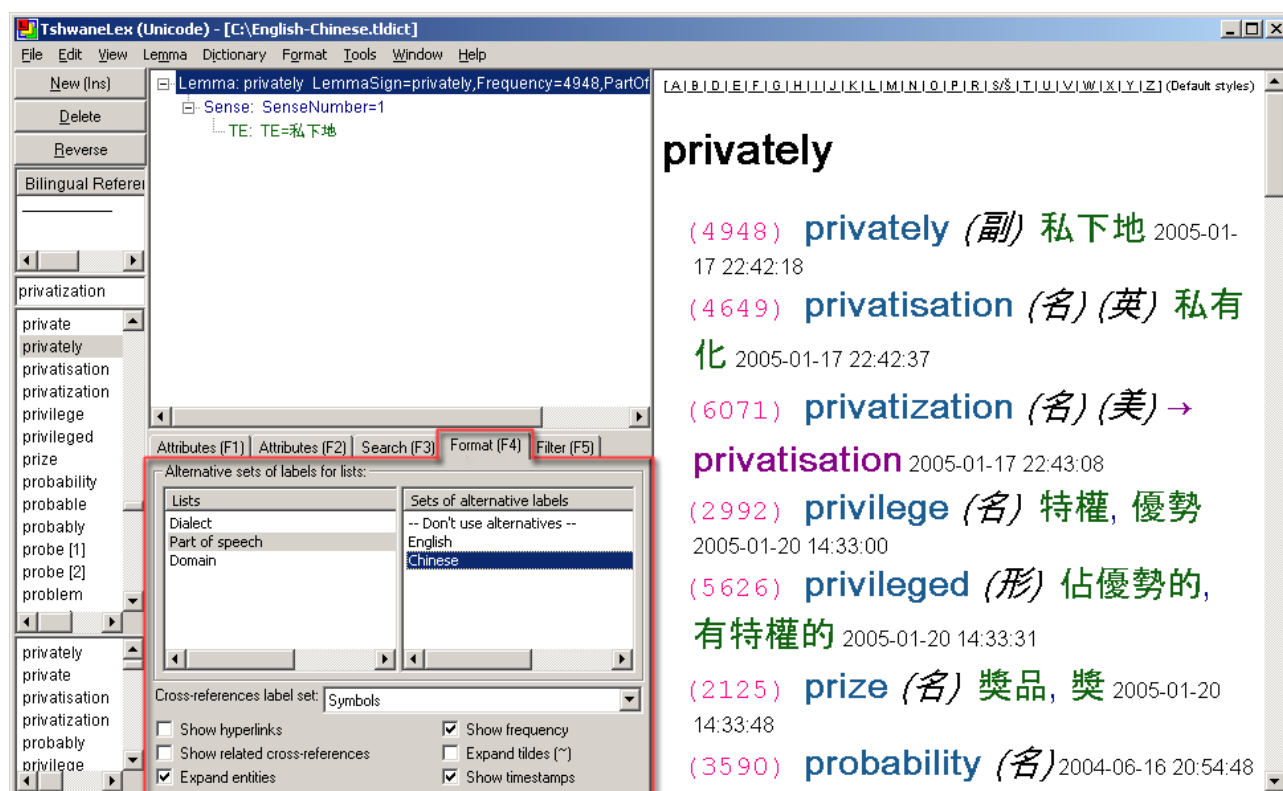


Figure 10: Manipulating general formatting settings under “Format (F4)”, here for a bilingual English – Chinese dictionary [Data: © Lorraine Liang]

Alternative Sets of Labels for Lists

The largest section of the “Format (F4)” sub-window is reserved for “Alternative sets of labels for lists”. As will be explained in the section on Attribute Lists, in the chapter on Customising the Database Structure using the DTD, each attribute list can have any number of alternate/variant lists. In the screenshot above, two alternative sets for the parts of speech have been prepared, one in English, the other one in Chinese, with the Chinese version being chosen. Cf. the first black sections in italics and between brackets in the Preview Area.

Expand Entities

Entities, discussed in the chapter on the Styles System, are automatically substituted with their associated values in the Preview Area when the “Expand entities” option is ticked. This is the default.

Show Timestamp

The last-saved date and time for each entry are automatically saved by tiDatabase. This information can be revealed by ticking the “Show timestamp” option. Timestamps are displayed at the end of each entry in the screenshot above.

Show Usernames

If 'User Management' is enabled on a database, 'created-by user' and 'last-saved-by user' information is automatically saved at each entry. This information can be revealed in the Preview Area by ticking the “Show usernames” option.

Filter (F5)

The “Filter (F5)” function allows you to display and work on only a *subset* of entries in your database, based on specified criteria. The filter acts as a kind of “mask” or “sieve” that “lets through” only those entries that conform to the specified conditions. The filter can also be thought of as a kind of “search” tool, as it frequently can be used to *find* entries that conform to the given conditions.

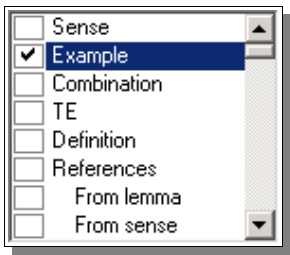
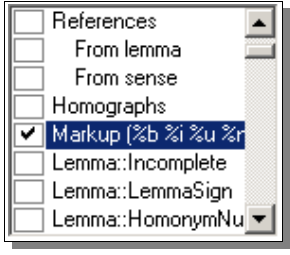
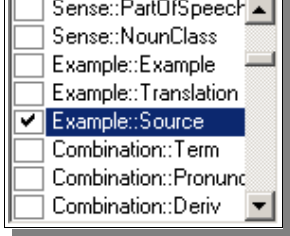
There are many different possible uses for the filter. For example, in a recipe database, one could specify a filter query to “show only those recipes which do not yet have images”. More complex filter queries can be defined.

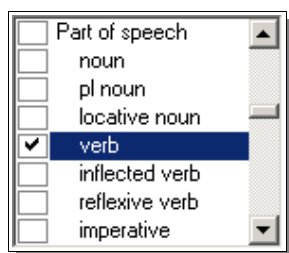
When a filter is applied, the Preview Area will display a message indicating this, and show how many entries, out of the total number of entries, have passed the filter, e.g. “[Filtered: 254 of 3289]”.

Other tools in tIDatabase, such as “Search (F3)” and the data exporters, are also “filter-aware”, and may accordingly work only on the subset of entries revealed by the currently applied filter.

Types of Filter Conditions

There are several types of filter conditions. These are all listed under “Filter (F5)”. Each condition has a checkbox to the left of it, which may be ticked in order to select that condition. The main types of filter conditions are as follows:

	<p>Element Filter: Checks for the <i>presence</i> of the given element type within the tree structure of an article, e.g. “Sense”, “Combination”, or “Example”.</p> <p>For the cross-references element (“References”), one may also further check for <i>only cross-references from a Lemma</i>, or <i>only cross-references from a Sense</i>.</p>
	<p>Markup Filter: Checks if any markup characters (e.g. “%b”) appear within an article.</p>
	<p>Attribute Filter: Checks if the given attribute value <i>appears</i> in an article <i>and has a value filled in</i>. Attribute filter conditions are listed in the form “ElementName::AttributeName”. For instance, ticking “Example::Source” would check for articles that contain an “Example” element <i>and</i> have a value filled in for the “Source” attribute of the “Example” element.</p>



List Filter: Checks if any list attributes in an article have the specified list item(s) selected.

Note that ticking the list *name* (e.g. “Part of speech”) instead of a particular item (e.g. “noun”) will check if *any* of the list items (in this case for “Part of speech”) are selected in an article.

Two sets of filter conditions may be specified – conditions for *inclusion* of entries (“Reveal”), and conditions for *exclusion* of entries (“Hide”). These may be combined to create more complex filters.

To apply a filter, press “F5” to select the Filter tool, specify the filter mode (e.g. “AND” or “OR”), tick the desired conditions, then click “Apply”. To unapply the filter again, click “Unapply”. To clear the currently ticked conditions and mode, click “Reset”.

“Reveal” Filter

The “Reveal” filter (previously called the “Include” filter) is used to reveal only entries that *conform* to the specified conditions. By selecting between “Any of (OR)” and “All of (AND)”, one can further specify whether only *one or more* of multiple specified conditions needs to be present, or whether multiple specified conditions should *all* be present, respectively, in order to include the entry. Some examples follow:

1. “Show only entries marked as incomplete”

- Under “Reveal”, select “Any of (OR)”
- Tick “Lemma::Incomplete”
- Click “Apply”

2. “Show all homonyms”

- Under “Reveal”, select “Any of (OR)”
- Tick “Lemma::HomonymNumber”
- Click “Apply”

3. “Show only nouns and verbs”

- Under “Reveal”, select “Any of (OR)”
- Tick “noun” under “Part of speech”
- Tick “verb” under “Part of speech”
- Click “Apply”

The “OR” option is used here as the article may have either the “noun” label or the “verb” label selected for the lemma to be revealed by the filter.

4. “Show all nouns that belong to the 'linguistics' domain”

- Under “Reveal”, select “All of (AND)”
- Tick “noun” under “Part of speech”
- Tick “linguistics” under “Domain label”
- Click “Apply”

Note that this example assumes that you have a “linguistics” domain label. The “AND” option is used because *both* conditions must be satisfied for an article to be shown (i.e. lemma “must be a noun” AND “must have a linguistics label”).

“Hide” Filter

The “Hide” filter (previously called the “Subtract” filter) can be used to reveal only those entries that do *not* conform to specified conditions. Phrased alternatively: The “Hide” filter can be used to *hide* all entries that *conform* to the specified conditions. By selecting between “Any of (OR)” and “All of (AND)”, one can further specify whether only *one or more* of multiple specified conditions needs to be present, or whether multiple specified conditions must *all* be present, respectively, for an entry to be hidden by the filter. Some examples follow:

1. “Show all entries that do not have usage examples”

- Under “Hide”, select “Any of (OR)”
- In the conditions list, tick “Example”
- Click “Apply”

2. “Show all entries that do not have the part of speech specified”

- Under “Hide”, select “Any of (OR)”
- In the conditions list, tick “Part of speech”
- Click “Apply”

Combining “Reveal” and “Hide”

The “Reveal” and “Hide” filters may be combined, allowing more complex filters to be defined. Some examples follow:

1. “Show all incomplete articles that have definitions but not usage examples”

- Under “Reveal”, select “All of (AND)”
- In the “Reveal” conditions list, tick “Lemma::Incomplete”
- In the “Reveal” conditions list, tick “Definition”
- Under “Hide”, select “Any of (OR)”
- In the “Hide” conditions list, tick “Example”
- Click “Apply”

The “Reveal” part of this filter will firstly select only the subset of articles that are marked as incomplete and that have definitions; the “Hide” part of this filter will then subtract from that all those lemmas that have usage examples.

2. [Note: In Bantu languages such as Kiswahili, nouns are grouped in singular/plural classes.]

In the screenshot below, all “complete” nouns that have been marked with the singular “class 9” or the corresponding plural “class 10”, and are “animate”, but for which the treatment does not have an “example sentence” or a “combination”, are being extracted. Out of the 6,147 lemmas in the database, this condition is fulfilled for only 46 of them, as shown at the top of the Preview Area.

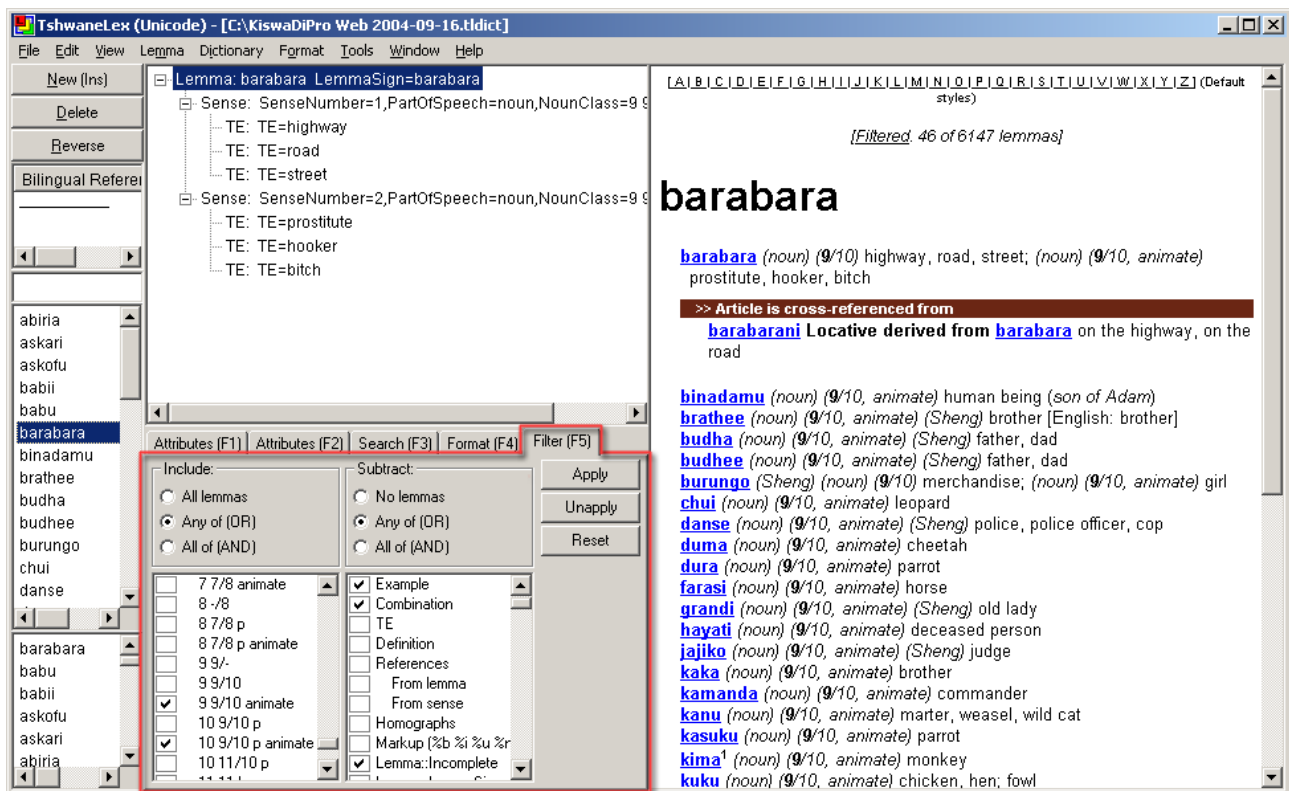


Figure 11: Combining the “Reveal” and “Hide” filters under “Filter (F5)” to extract a section from a bilingual Swahili – English dictionary [Data online at: <http://africanlanguages.com/swahili/>]

Extracting or Printing a Subset of the Database

The filter may be used to select and export only a subset of the database. When you export your data (whether to RTF, HTML or XML format), the “Export options” dialog includes an option to “Use filters”. Selecting this option will cause the current filter to be applied to the output.

As an example, one could choose to export only the entries in a recipes database marked “Asian” in order to export an Asian cookbook or recipe list.

By combining the filter with the use of multiple sets of styles, which may further hide or reveal different elements or attributes, multiple sub-databases can effectively be generated from a single database. See the chapter on the Styles System for more information.

Tags

Overview

Entries in your data can be “tagged”. This is a temporary 'flag' on an entry that merely marks that entry as being tagged. This allows you to then apply certain actions to the tagged subset of entries, such as filter on all tagged entries (see the section on filters for more information). Tagged entries display in the Preview Area with a red half-border to the left and top, and with a small image of a tag next to them.

Tags are not saved with your document, and are cleared when you close it.

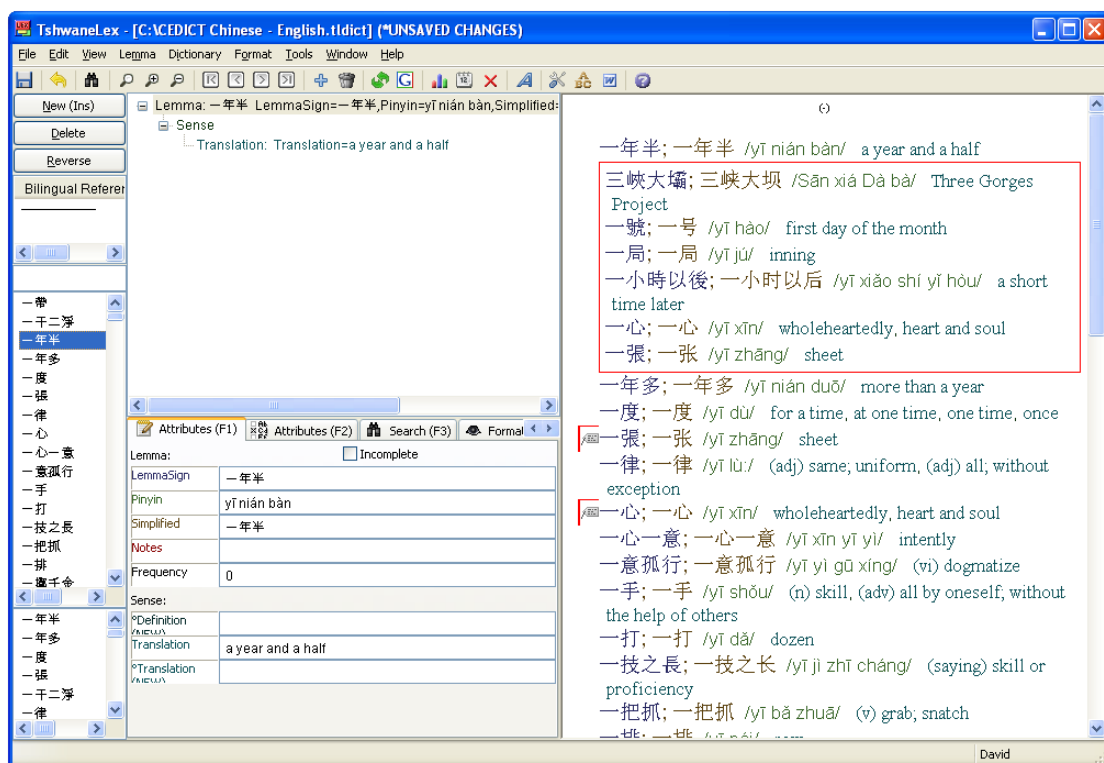


Figure 12: Tagged entries in a Chinese - English dictionary. The "Show tagged always" option is enabled, thereby showing all tagged entries with a red border in the Preview Area at all times. (The data is from the CEDICT project and is online at <http://dictionaryq.com/chinese/>.)

Tagging Commands

“Edit/Tag entry” (Ctrl+F2): Tag or untag the currently selected entry.

“Edit/Tag all”: Tag all entries in the Entry List. This is “filter-aware” - if a filter is currently applied, only the entries that pass the filter will be tagged. This allows for a new way to *combine* the results of several different filters – by tagging the matching entries of each filter, one can then finally filter on “tagged” entries.

“Edit/Clear all tags”: Clear all current tags.

“Edit/Filter tagged”: Create and apply a filter that displays only currently tagged entries.

“Edit/Show tagged always”: Display all tagged entries in the Preview Area persistently. One use for this would be if you want to continually view or refer to an entry while working on another entry.

Tag Filter

Under “Filter (F5)”, the “[Tagged]” filter condition allows you to apply either a “reveal” or “hide” filter for tagged entries.

Tagging in the “Find” Tool

In addition to the “Edit” menu commands for tagging, the “Find” tool allows you to automatically tag or untag all entries that match a given search query.

This could be useful if you would like to combine the results of multiple different search queries, and e.g. create a filter from that.

Tagging a Range of Entries

To tag any range of entries:

1. Select the first desired entry of the range in the entry list
2. Click “Edit/Tag entry”
3. Select the last desired entry of the range in the entry list
4. Click “Edit/Tag range”

If combined with a tag filter, this can then also be used to easily export a specific range of entries by selecting the 'Use filters' option at export time.

Compare/Merge

Overview

The Compare/Merge tool allows you to visually compare two databases, or two versions of a database, side by side. When there are differences, four options are provided for resolving those differences, namely “Add”, “Merge”, “Replace” and “Delete left”. This tool thus allows you to integrate changes made by other users back into the main database, or to integrate changes made on another computer (e.g., if you have worked at home, and now wish to add the changes made at home to the database at the office). It also allows you to merely check on the progress that has been made since a previous version of the database, or by another user. The Compare/Merge tool can be accessed via the “File/Compare/merge” menu option.

As an example, assume that you usually work on the main database file for a Kiswahili – English dictionary, called “Kiswahili.tlatabase”, while your colleague, Sarah, works on her own copy of this database, called “Sarah.tlatabase”. Once a week, Sarah e-mails you her file, and you merge her changes into the main dictionary database. You then send her a copy of the new, up-to-date combined database, and she proceeds with her work on that file. In order to use the Compare/Merge tool to achieve this, you would do the following:

- Open your main dictionary file “Kiswahili.tlatabase ” as you would usually do, e.g. using “File/Open: tlDatabase file” and selecting it. (Always open the main dictionary database file first, and then the file you want to compare against and whose changes you want to integrate.)
- Select the “File/Compare/merge dictionary” menu option.
- A file selection dialog will appear, labelled “Choose a file to compare against”. Select “Sarah.tlatabase ” and click “Open”.
- The Compare/Merge dialog will appear, as shown in Figure 13.

The Compare/Merge dialog is split into two halves – the left half displays a list of lemmas in the *main database* “Kiswahili.tlatabase ” that was opened first; the right half displays a list of lemmas in the database you are comparing *against*, namely Sarah.tlatabase. The filenames are indicated in the headings above each list of lemmas. It is important to note that any changes that you make using the Compare/Merge dialog will always be made to the *main* (i.e. left) database.

The title bar of the dialog displays a summary of the number of differences found. There are four possible cases, each of which is displayed in a different colour in the list of lemmas:

1. **“Same”**: The lemma exists in both dictionary documents, and is identical in both, i.e. no changes have been made. These are displayed in **black** text with a white background (see e.g. “pombe”, “profesa” or “-puana” in the screenshot).
2. **“Different”** The lemma exists in both dictionary documents, but the article is different, i.e. changes have been made. These are displayed in either **purple** or **red** (see e.g. “polisi [1]”, respectively “-punguza”, in the screenshot). If the lemma in the compared-against (right) side has a newer “last-edited” timestamp, then purple is used. Otherwise, red is used.
3. **“Only left”**: The lemma exists in the main (left) dictionary document, but does not exist at all in the compared-against (right) dictionary document. In this example, this would imply that either you have added a new lemma to the Kiswahili. tlatabase file, or Sarah has deleted an existing lemma from her Sarah.tldict file. These are displayed in **blue** (see e.g. “polisi [2]” in the screenshot).

4. **“Only right”**: The lemma does not exist in the main (left) dictionary document, but does exist in the compared-against (right) dictionary document. In this example, this would imply that either you have deleted an existing lemma from the Kiswahili.tlatabase file, or Sarah has added a new lemma to her Sarah.tlatabase file. These are displayed in **green** (see e.g. “-pokea”, “polepole” or “-pona” in the screenshot).

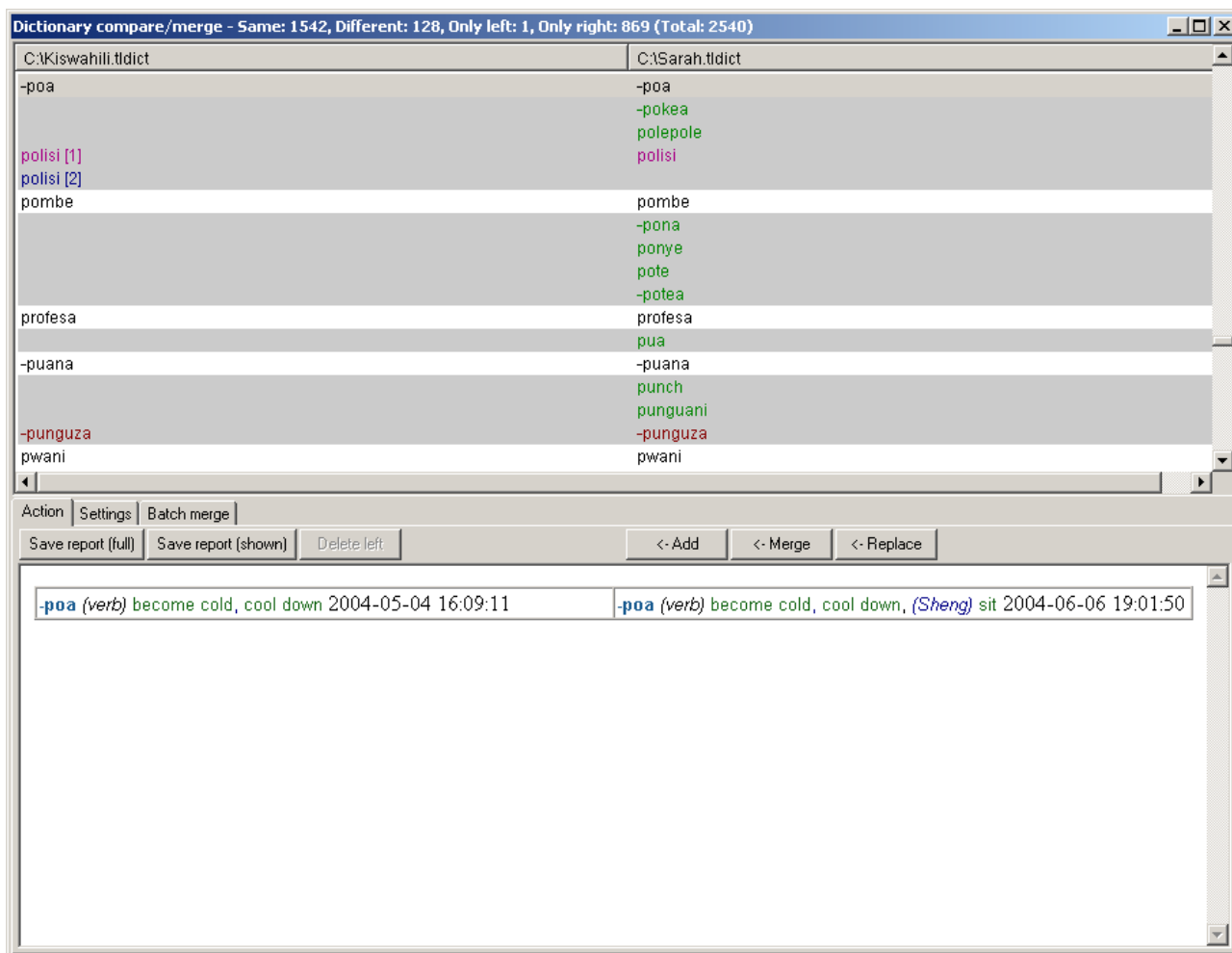


Figure 13: “Action” tab in the dictionary Compare/Merge dialog, here comparing/merging two versions of a bilingual Kiswahili – English dictionary database [Data online at: <http://africanlanguages.com/kdp/>]

Action Commands: “Add”, “Merge”, “Replace” and “Delete Left”

At this point you might then want to inspect Sarah’s changes in each case where there are differences, and use the “Add”, “Merge”, “Replace” or “Delete left” commands, which are available on the “Action” tab, to then transfer her changes to the main database. Keep in mind that these commands will always make changes to Kiswahili.tlatabase (i.e. the file opened first), and never to Sarah.tlatabase. The four commands work as follows:

Add: The article on the right is *added* to the main database, in its entirety, as a new lemma. Use this if the lemma is a new one or a new homonym added by Sarah.

Merge: All senses and other child elements of the article on the right are *added* to the existing article on the left (i.e. the existing article in the main database). This option only applies to lemmas that exist in both databases, but are different.

Replace: The version of the article on the right *replaces* the version of the article on the left (i.e. the existing article in the main database). This option only applies to articles that exist in both databases, but are different. Use this if Sarah has made *corrections or extensions* to an existing lemma.

Delete left: The article in the main (left) database is deleted. This option only applies to articles that exist in the main database only, i.e. those shown in blue. Use this if Sarah has deleted a lemma, and that lemma should also be deleted from the main database.

Note that it is possible to make changes to the main database while the Compare/Merge dialog is open. One may wish to resize the Compare/Merge dialog so that one can easily move from that window to the main dictionary database.

Settings

Typically one would not be interested in seeing, say, articles that are identical in both databases. It is possible to choose which of the four possible types of differences to display in the list. This can be done by clicking on the “Settings” tab, and using the four checkbox options, as shown below:

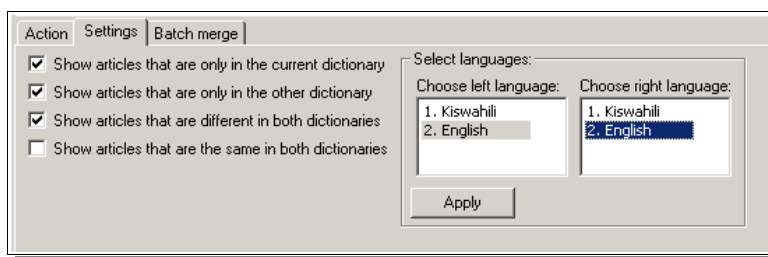


Figure 14: “Settings” tab in the dictionary Compare/Merge dialog, here comparing/merging two versions of a bilingual Kiswahili – English dictionary database [Data online at: <http://africanlanguages.com/kdp/>]

Batch Merge

If there are many changes in the compared-against database that you wish to merge into the main database, this can be done automatically, rather than adding/merging/replacing entries “one by one”. This function thus allows you to add updates to the main database “in bulk” from the compared-against dictionary. To do this, select the “Batch merge” tab, as shown in the screenshot below:

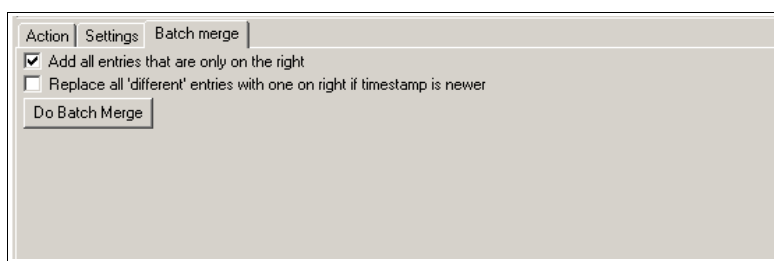


Figure 15: “Batch merge” tab in the Compare/Merge dialog

There are two options for “Batch merge”:

1. **“Add all entries that are only on the right”:** When this is selected, all articles that appear in the compared-against (right) document but not in the main (left) document will be added during the “batch merge”. These are all the entries marked in **green**.

2. **“Replace all 'different' entries with one on right if timestamp is newer”**: When this is selected, all articles that appear in both databases, but where the version on the right is *newer*, will be *replaced* by the newer version. These are all the entries marked in **purple**.

Once you have selected the desired options, click “Do Batch Merge” to go ahead with the merge.

<p>Important: It is strongly recommended that you make a backup of the main database (“File/Create a backup”) before doing a “batch merge”.</p>

Customising the Database Structure using the DTD [Advanced]

The preceding chapters presented a “gentle” overview of the use of the default features available in tlDatabase. Starting with the current chapter, more advanced features are discussed. The first of these describes the fully customisable and built-in DTD (Document Type Definition) editor of tlDatabase. This powerful tool, which is based on XML (eXtensible Markup Language) standards, allows you to customise the database structure for any project.

The tlDatabase DTD editor dialog can be accessed from the “Database/Customise DTD (data structure)” menu option.

Note: It is recommended that you make a **backup** of the database (“File/Create a backup”) before making changes to the DTD.

Basics of Hierarchical Data Modelling in tlDatabase

Many types of data can be modelled hierarchically. For example, a recipe may contain a list of ingredients and a list of instructions (or steps). An invoice may contain a list of line items. A dictionary entry may consist of several word senses, and each of those word senses may in turn contain subsenses. Any of the senses or subsenses may contain usage examples or subentries, which themselves may again in turn contain one or more word senses. When creating an entry, the user's task can be seen as consisting of two essentially separate but closely related sub-tasks: (1) To specify the basic skeletal, hierarchical *structure* of the entry, and (2) to flesh out that basic structure with *content*, such as the actual ingredients or instructions for a recipe. One can apply a “tree” metaphor here: the structure may be seen as being like the “branches” of a tree, while the content itself are the “leaves” on the branches.

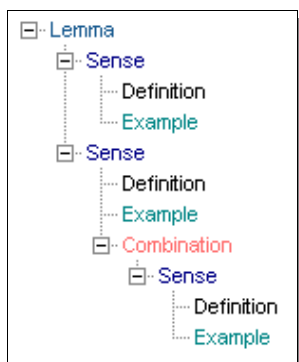


Figure 16: Empty basic tree structure (“tree without leaves” – elements only)

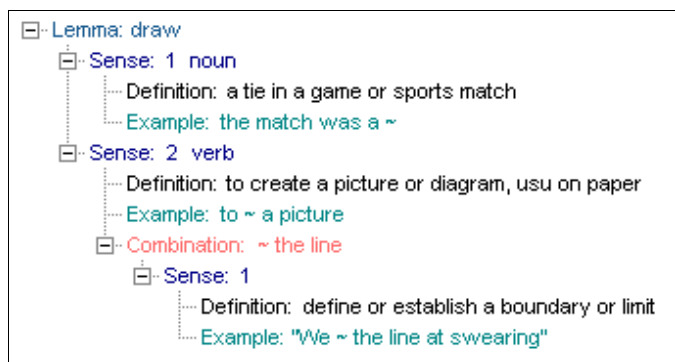


Figure 17: Basic tree structure with content filled in (“tree with leaves” – elements and attributes)

Elements and Attributes

In tlDatabase, the “root” and “branches” of the tree structure are called *elements*. The text *content* values, which provide the actual *data*, are called *attributes*. An element is essentially a *container* that stores, firstly, other child elements, and secondly, attributes. Different types of elements may be defined, and for each element type there is a particular set of attributes associated with that element. For example, for the “Lemma” element, one may fill in attributes such as the lemma sign itself, the pronunciation field, and an etymology field. For a usage example (“Example”) element, there may

be three attributes: the usage example itself, a translation (if a bilingual dictionary), and optionally the source (citation). For a “Definition” *element*, one may fill in the definition *attribute*.

The tree structure in tIDatabase is edited using the Tree View control (see the section on Structuring Entries with the Tree View Control, in the chapter Getting Started with tIDatabase: QuickStart Guide). To add an “Ingredient” to a recipe, for example, one right-clicks on the “Recipe” node in the Tree View and selects “Add: Ingredient”. The “leaves” of the tree, i.e. the contents, are filled in under the “Attributes (F1)” and “Attributes (F2)” tool windows. Thus one uses the Tree View primarily to edit the *structure* of an entry, and the “Attributes” windows primarily to edit the *content* of element attributes.

When one clicks on any element in the Tree View, the “Attributes (F1)” window shows the attributes associated with the type of element one has clicked on, and one may immediately proceed with entering or modifying the relevant contents. For example, if you click on an “Ingredient” element, one might expect to see attributes for that ingredient, such as the quantity (e.g. “500”, the unit (“grams”), the ingredient itself, and any ingredient modifications or notes (e.g. “finely chopped”).

These links between (1) internal article structure, (2) Tree View, and (3) boxes / fields to be filled in by the user, constitute one of the crucial design features to ensure smooth and sound compilation. Only the relevant attributes are seen at any given point, so that the screen does not get clogged, and so that the potential for errors is minimised as only attributes allowed by the DTD are editable.

What is a DTD?

A DTD (Document Type Definition), also known (loosely) as a “schema”, is a description of the structure of the entries of a particular database. Specifically, this is a description of the actual types of *elements* that the database may have, as well as the *attributes* that each of those elements may have. Every document needs a DTD, and this is usually mostly set up as completely as possible at the start of a project, with ideally only minor modifications later on. The DTD is then used while creating the database, and by enforcing conformance to a well-designed DTD, one can ensure that the final database follows a logical and consistent structure throughout.

When a new database is created in tIDatabase using a template, the template defines a basic default DTD. This may then be customised further by the user. Note, however, that a few DTD element and attribute types are fundamental to tIDatabase and cannot be removed, although all elements and attributes may be renamed.

A DTD primarily defines three basic things:

1. **Element types:** The types of elements that may appear in the database.
2. **Element attributes:** The attributes associated with each element type.
3. **Element child relations:** For each element type, one may specify which other element types may be attached as children, for example a “Recipe” may have “Ingredient” and “Instruction” elements attached to it.

Element Types

The DTD contains a *list of all the element types* that may appear in the database.

In the tDatabase DTD editor dialog, the full list of element types appears on the *left*, under the heading “Element types”. See the screenshot below:

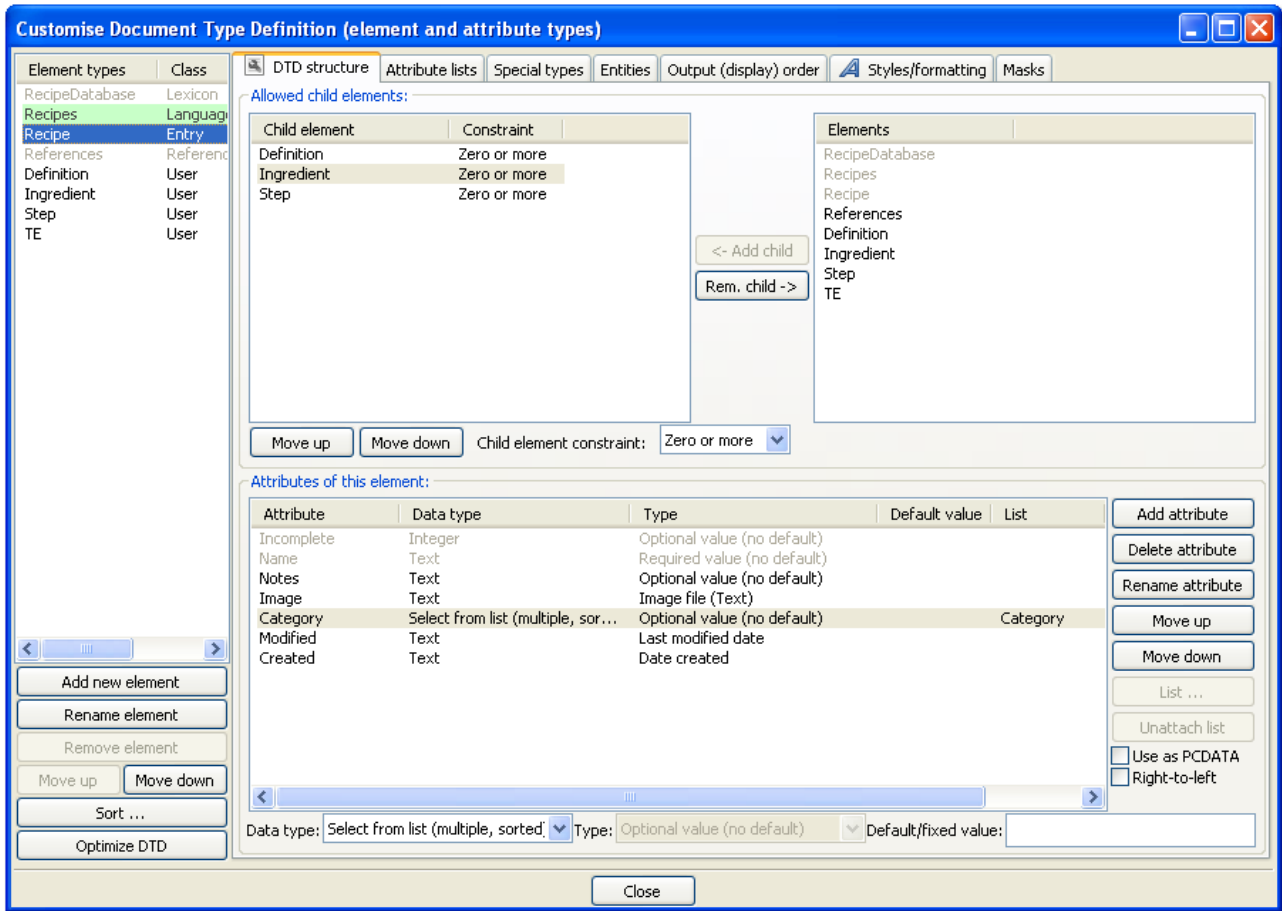


Figure 18: DTD Editor: Setting up the "Recipe" element

To create a new element type, click on the “Add new element” button below the element list, near the bottom left of the dialog.

Existing element types may be renamed or removed using the “Rename element” and “Remove element” buttons below the element list.

WARNING: If one removes an element type in the **DTD editor dialog**, *all existing elements of that type throughout the database* will be removed.

Element Attributes

Each *element* in the DTD has its *own set of attributes*.

When an element is selected in the list of elements on the left of the DTD editor dialog, the “Attributes of this element” window on the “DTD structure” tab, shown in the screenshot below, displays a list of the attributes for that element. For example, if “Recipe” is selected, all main “recipe” attributes are shown, such as “Name” for the recipe name. (These are the attributes that are shown under “Attributes (F1)” when a node of this element type is selected in the Tree View while editing the database.)

New attributes may be added using the “Add attribute” button to the right of the list of attributes. Existing attributes may be renamed using the “Rename attribute” button. The order that attributes appear under “Attributes (F1)” may also be modified by using the “Move up” and “Move down” buttons. (Note: This only affects the order that attributes appear under “Attributes (F1)”, and *not the output order* of attributes in the Preview Area or RTF, HTML, etc. exporters. The *output order* is configured separately, and will be explained later.)

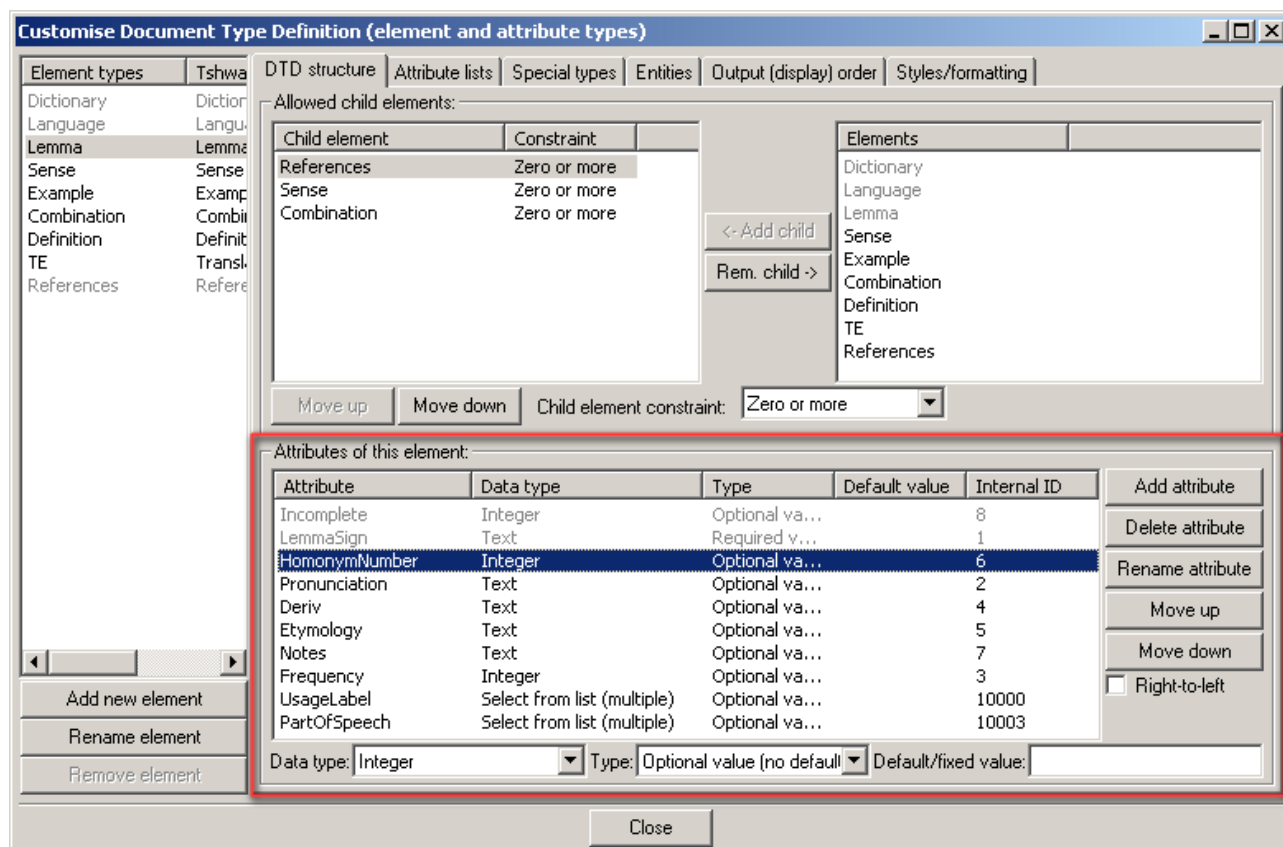


Figure 19: “DTD structure” tab, Attribute editing window: Editing the attributes of the “Lemma” element

Element Child Relations

The list of elements at the left of the DTD editor dialog is a “flat” list, i.e. it only describes *what* element types appear in the database, but does not describe how they relate to one another – that is, which element types are allowed to be attached to which other element types when building the hierarchical structure of each entry in the Tree View using the right-click “Add.” commands. Clearly, only certain elements may be added to others, for example, it will make little sense to add a “Recipe” element beneath an “Instruction” element, so this is not allowed. Thus, one needs to define element “child relations”, which describe the “allowed child element types” of each element, i.e. what “belongs to” what else. This is done in the “Allowed child elements” window of the “DTD structure” tab in the DTD editor dialog, shown in the screenshot below. This window contains two lists: the left one, with heading “Child element”, is a list of the currently allowed child elements for the selected element type. The list on the right is simply a list of *all* element types, from which one may choose to add allowed child elements.

For example, if one wishes to specify that a “Recipe” element may have an “Ingredient” element attached to it, then one would do the following:

1. Select “Recipe” from the list on the very left of the DTD editor.

2. Select “Ingredient” from the list of all elements at the right of the “Allowed child elements” window.
3. Click on the “<- Add child” button.

The “Ingredient” element will be added to the list of allowed child elements for “Recipe”. Subsequently there will be an “Add: Ingredient” command in the menu that appears when right-clicking on “Recipe” in the Tree View.

One can remove an element from the list of allowed child elements for the selected element by selecting them in the “Child element” list and clicking the “Rem. child ->” button.

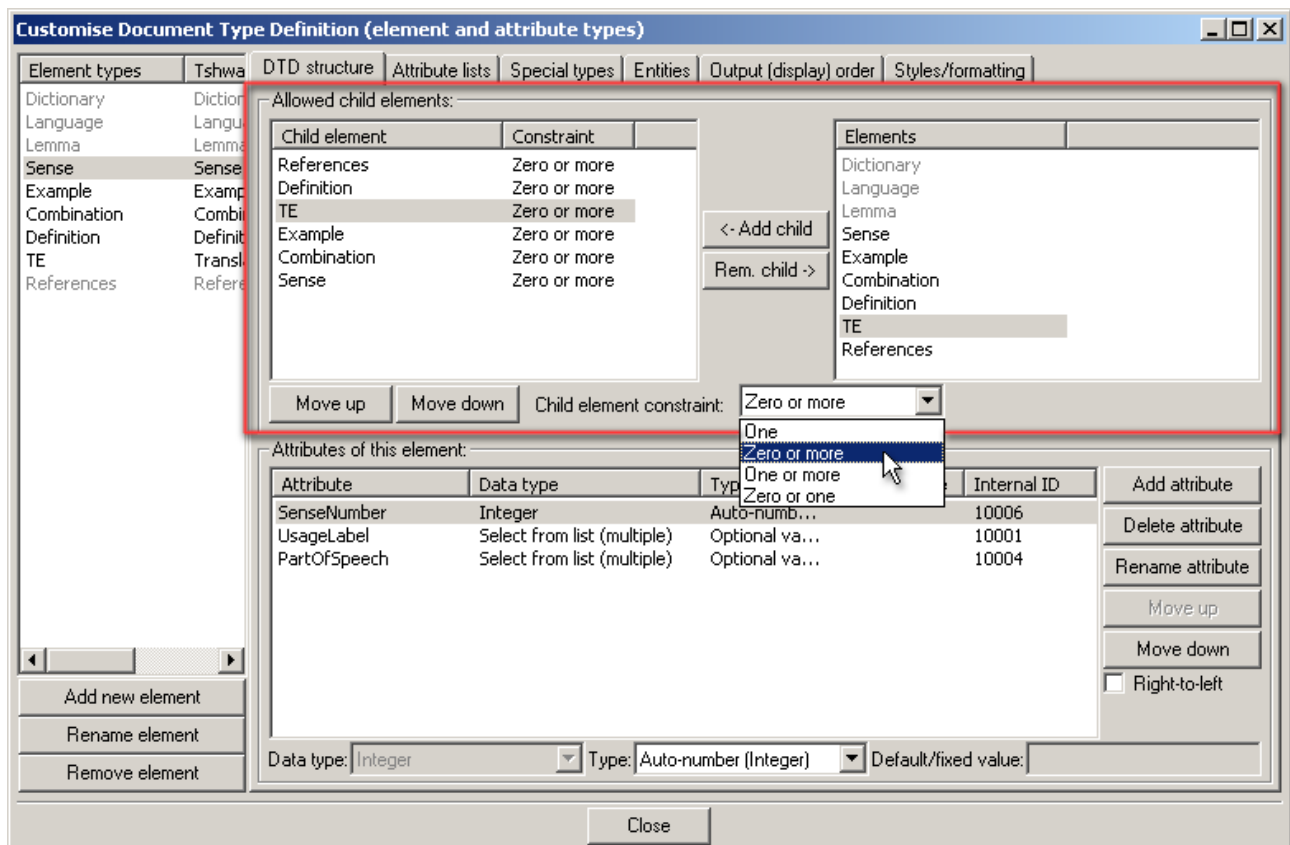


Figure 20: “DTD structure” tab, Child elements editing window:
Editing the list of allowed child elements of the “Sense” element

Element Child Relation Constraints

Each “parent->child” element relation may also have a **constraint** associated with it, such as “this element may have *only one* child of this type”. For example, if the “Subentry->Sense” child relation specified that no more than one “Sense” may be attached to a “Subentry”, then t1Database will not allow more than one “Sense” to be added to the “Subentry” element when editing in the Tree View, and this option will be “greyed out” in the right-click menu of the Tree View.

There are four possible constraint types, which may be selected using the “Child element constraint” drop-down list in the “Allowed child elements” window of the “DTD structure” tab:

1. **One:** One child of this type only.
2. **Zero or more:** Zero or more children of this type.
3. **One or more:** One or more children of this type (“at least one”).
4. **Zero or one:** Either zero or one child of this type, but no more than one.

The order that child elements appear in the Tree View may also be modified by using the “Move up” and “Move down” buttons. (Note: This only affects the order that elements appear in the Tree View, and *not the output order* of child elements in the Preview Area or RTF, HTML, etc. exporters. The *output order* is configured separately, and will be explained later.)

Attribute Lists

Attribute lists are closed lists of pre-prepared attributes, from which one can simply select those attributes that need to be attached to certain elements. Basically, there are two different list types – single-selection, and multi-selection lists. In the first the user can only select one item from the list (“one of”), in the second zero or more items may be selected. For the second type, a distinction is also made between “sorted” and “unsorted”. For the sorted type, the order of the output of selected list items will always be the same as the order of the items defined centrally for the list. For the unsorted type, the order of the output of selected list items will be the same as the order in which they are selected by the user. Also note that any field can be converted from a free text field to a closed list (and vice versa) at any time in tDatabase. Care should be taken if converting back and forth between multi-selection lists and other data types.

While editing the data, the “one of” attribute lists are found under “Format (F1)”, see e.g. Figure 5, while the two types of “multiple” attribute lists are found under “Format (F2)”, see e.g. Figure 7.

The second tab on the DTD editor dialog, the “Attribute lists” tab, is used for the management of the various lists in the database. This tab consists of four windows, as shown in the screenshot below:

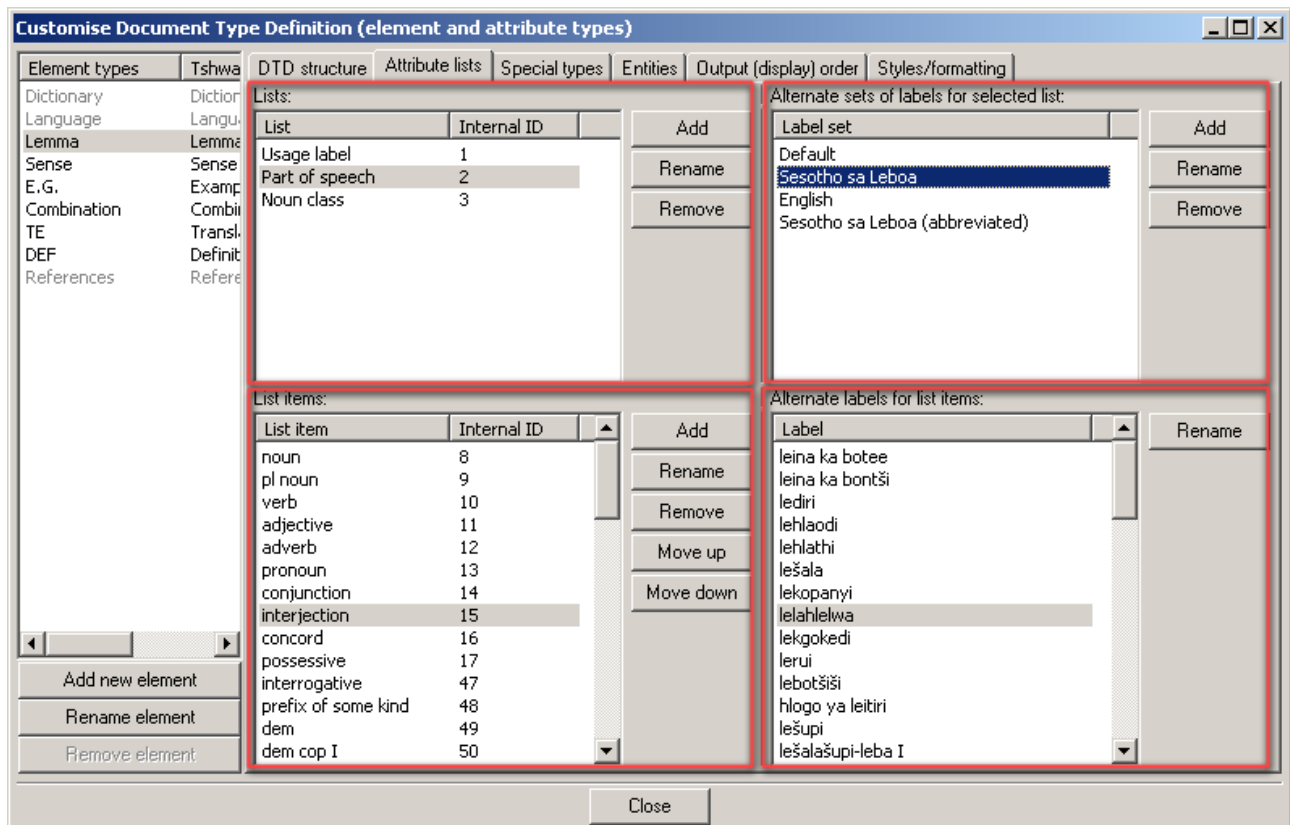


Figure 21: “Attribute lists” tab in the DTD editor dialog

The top-left window, called “Lists”, simply enumerates the names of the various attribute lists. Existing attribute lists can be renamed or removed, and new ones added. The database in the screenshot contains three lists, namely “Usage label”, “Part of speech” and “Noun class”. When you stand on the name of a particular list in the “Lists” window, the list items for that particular attribute list are revealed immediately underneath, in the bottom-left window labelled “List items”. Here too, existing items can be renamed or removed, and new ones added. The order of the list items can be changed with the “Move up” and “Move down” buttons. This order is also the order in which the items will appear under F1 and F2.

Once one has prepared the data under “Lists” and “List items”, one basically has enough to start using the attribute lists. What remains to be done, however, is to link those attribute lists to the elements on which these list attributes are allowed to appear. This is done on the “DTD structure” tab. For example, if one wishes to attach a “Part of speech” list to the “Subentry” element, then one will first need to create a new attribute for the “Subentry” element, e.g. “PartOfSpeech”, and, under “Data type”, select one of the list types. Say one opts for the type “Select from list (multiple)”, then a new dialog is presented, from which one can select the appropriate attribute list, as shown below:

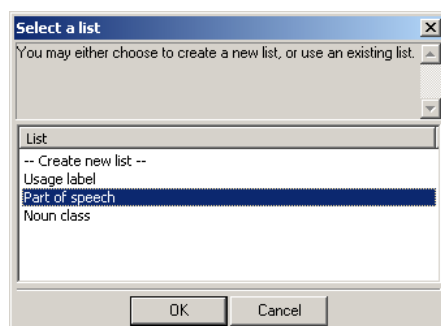


Figure 22: Attaching a list to an attribute in the DTD

Choosing the attribute list “Part of speech” and clicking “OK” will attach that attribute list to the “Subentry” element. Given a “multiple” type was chosen, this list will be available under F2. A single list can be reused as many times as one wants. So, the same “Part of speech” list can also be attached to for example the “Lemma” element and the “Sense” element. This is for instance the case in Figures 19 and 20 respectively.

The power of using attribute lists is obvious: they save time during the editing of the data (as items can simply be selected from drop-down menus (F1) and/or selected by ticking checkboxes (F2), and should thus never be typed in when compiling entries), they ensure consistency (e.g. ‘preposition’ will always be ‘prep.’, and not arbitrarily ‘prep’ or ‘prep.’ or ‘pp’ or ‘pp.’ or ‘preposition’, etc.), and spelling errors are avoided (as the lists only need to be prepared once). Moreover, all list attributes are available for the “Filter (F5)” tool, which means that one can easily extract all entries marked with particular list items – for example, all “Asian” recipes in a recipe/cookbook database.

However, in tDatabase the use of (closed) attribute lists has been taken one step further. Firstly, they can be expanded at any time, by simply returning to the DTD editor dialog, and modifying the existing lists. Secondly, each of the lists can have as many (customisable) “linked variant / alternative lists” as one wants, which effectively means that one can *dynamically customise the language of the metalanguage* with just a few clicks. If one prepares both a long and a short form of the usage labels, then one can use the former for a desktop edition and the latter for a pocket edition of the same data. If, for bilingual lexicography, one has prepared the part-of-speech lists in the source as well as the target language, then versions with different metalanguages can be output depending on the intended market. In an electronic or online environment, dynamic metalanguage customisation can even be realised in real time, allowing for truly instantaneous tailoring.

Names for parallel lists are given in the top-right window of the “Attribute lists” tab, which is called “Alternate sets of labels for selected list”. The alternative items themselves are entered underneath, in the bottom-right window labelled “Alternate labels for list items”. In Figure 21, for example, the Sesotho sa Leboa (Northern Sotho) equivalents for the English part-of-speech items are shown.

Once an attribute list and its alternates have been set up in the DTD, they immediately become available for compilation, as may be seen from the F2 window in the screenshot below (where “leina ka botee” is the Northern Sotho equivalent of “(singular) noun”):

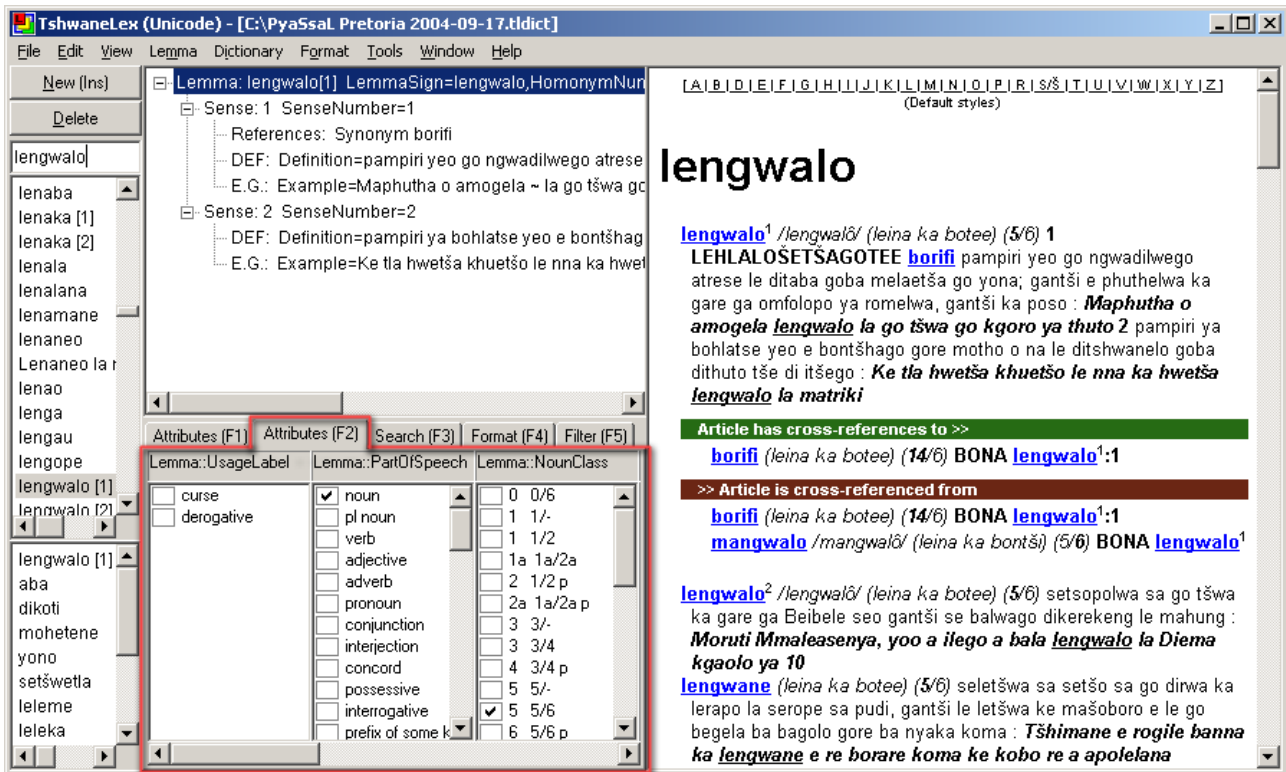


Figure 23: Selecting list items from attribute lists under “Format (F2)” for an explanatory Northern Sotho dictionary [Data online at: <http://africanlanguages.com/ps/>]

Swapping to an alternate label set for attribute lists may easily be done under the F4 window, as shown below:

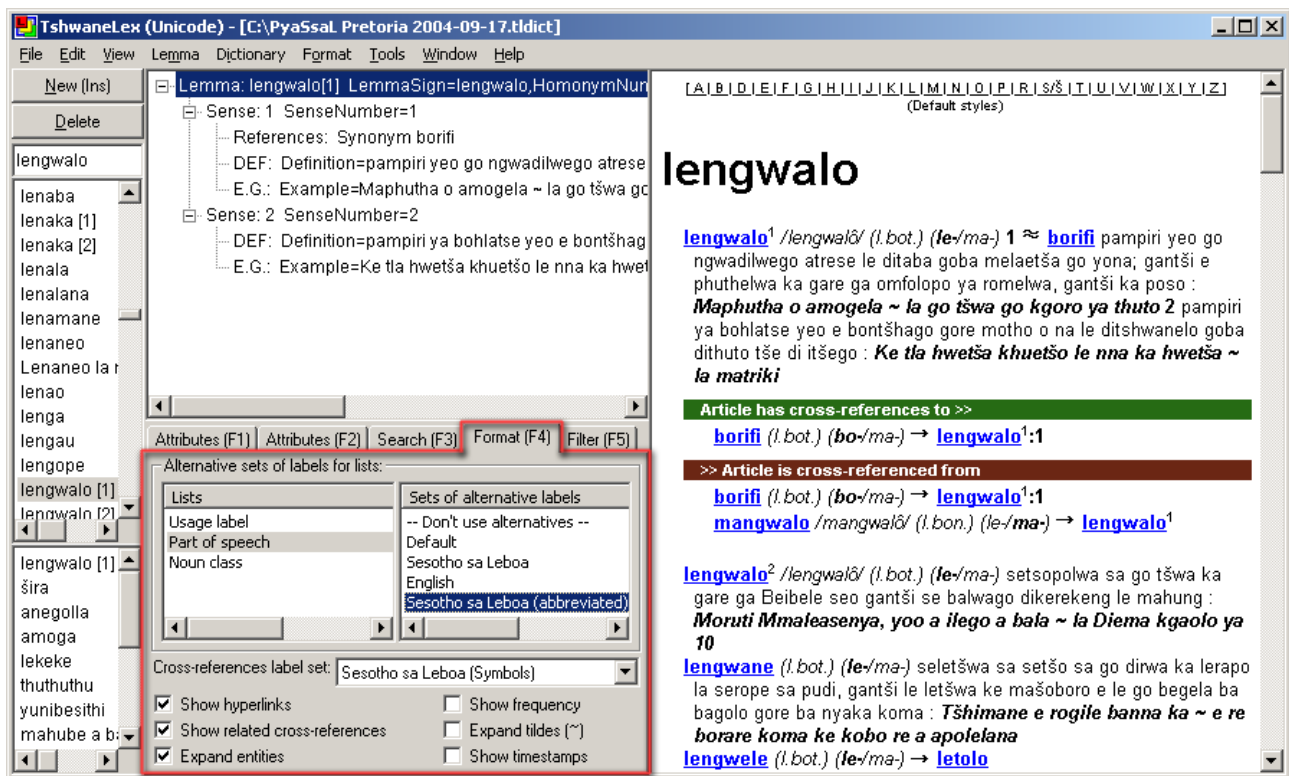


Figure 24: Selecting alternate label sets for attribute lists under “Format(F4)” for an explanatory Northern Sotho dictionary [Data online at: <http://africanlanguages.com/psl/>]

Changes take immediate effect throughout the entire database, as seen in the Preview Area. These settings are also used when exporting data.

Multimedia (Audio and Images)

Both sound recordings and images can be linked to any element in the DTD. To do so, it suffices to select the “Sound file (Text)” attribute type, respectively the “Image file (Text)” attribute type, on the “DTD structure” tab of the DTD editor dialog. For a screenshot of the relevant section of the “DTD structure” tab, see Figure 31.

All sound files and images are stored in a central place. This place is defined by filling in the “Sound/images path” input box of the “Dictionary/Properties” menu option. See Figure Error: Reference source not found for a screenshot of that dialog.

For an example of a database with image files, see the 'Cookbook' sample database.

DTD Templates

It is possible to save the DTD of a particular database with the “Database/Save DTD template” menu option. This DTD template can then be loaded into a new database (“File/New database”). See the “Load DTD template from file” section of Figure Error: Reference source not found in this regard.

Customising the Alphabetic Sorting

Overview

TshwaneLex automatically sorts the articles in your dictionary, freeing you from having to do so. However, many different methods of sorting exist, and often many even for the same language. In order for TshwaneLex to support any possible sorting method that may be desired, an extensibility mechanism was included whereby new “sorting plug-ins” can be created. Thus, support for any sorting method (e.g. by radical/stroke count or by pinyin romanised form for Chinese) can be added to TshwaneLex.

Table-based Sorting

The default sorting method/“plug-in” supported by TshwaneLex is a configurable four-pass table-based sorting system (based on the ISO 14651 standard) that can be used for most Latin-based alphabets. The four passes are used for checking different characteristics that may take precedence over one another when two strings are compared to determine their order in the dictionary.

For example, in pass one, the strings are typically first compared using the so-called “base alphabet”, that is, the underlying “alphabetic characters” of the letters, as if there were no diacritics.

Only if the underlying letters are the same, are the diacritics in the string then compared in a second pass. This allows all “related” characters to be sorted together, e.g. all “o” forms (“o”, “ö”, “ò”, “ó”, “ô”, etc.), meaning, for example, that the French lemma signs “roder” and “rôder” will follow one another in the dictionary, as the dictionary user would likely expect. Thus all “o” forms are, in this case, *primarily* sorted as if they were just “o”, and only *secondarily* sorted on the differences in diacritics.

If both the underlying alphabetic characters *and* the diacritics are the same, then a third pass checks for uppercase/lowercase differences (e.g. the French lemma signs “Moïse” vs. “moïse”).

Finally, if there are no case differences, the fourth pass compares all other “non-alphabetic” characters, e.g. spaces, dashes and other punctuation symbols (these characters in the fourth pass are known as “ignorables”).

Configuring Table-based Sorting

The specific alphabetic characters and diacritics that should be used during comparison, as well as the order of the letters, is naturally language-dependent. For example, in Estonian, “z” is sorted before “t”. Additionally, for some languages, certain diacritic characters *should* be sorted as separate alphabetic characters even when the underlying form is the same as another character. Again for Estonian, “ä”, “ö” and “ü” are separate letters that are sorted between “w” and “x”, and not together with “a”, “o” or “u” (see Figure 25 below). Thus, the entire system can be fully configured in “tables” – hence the name “table-based sorting”.

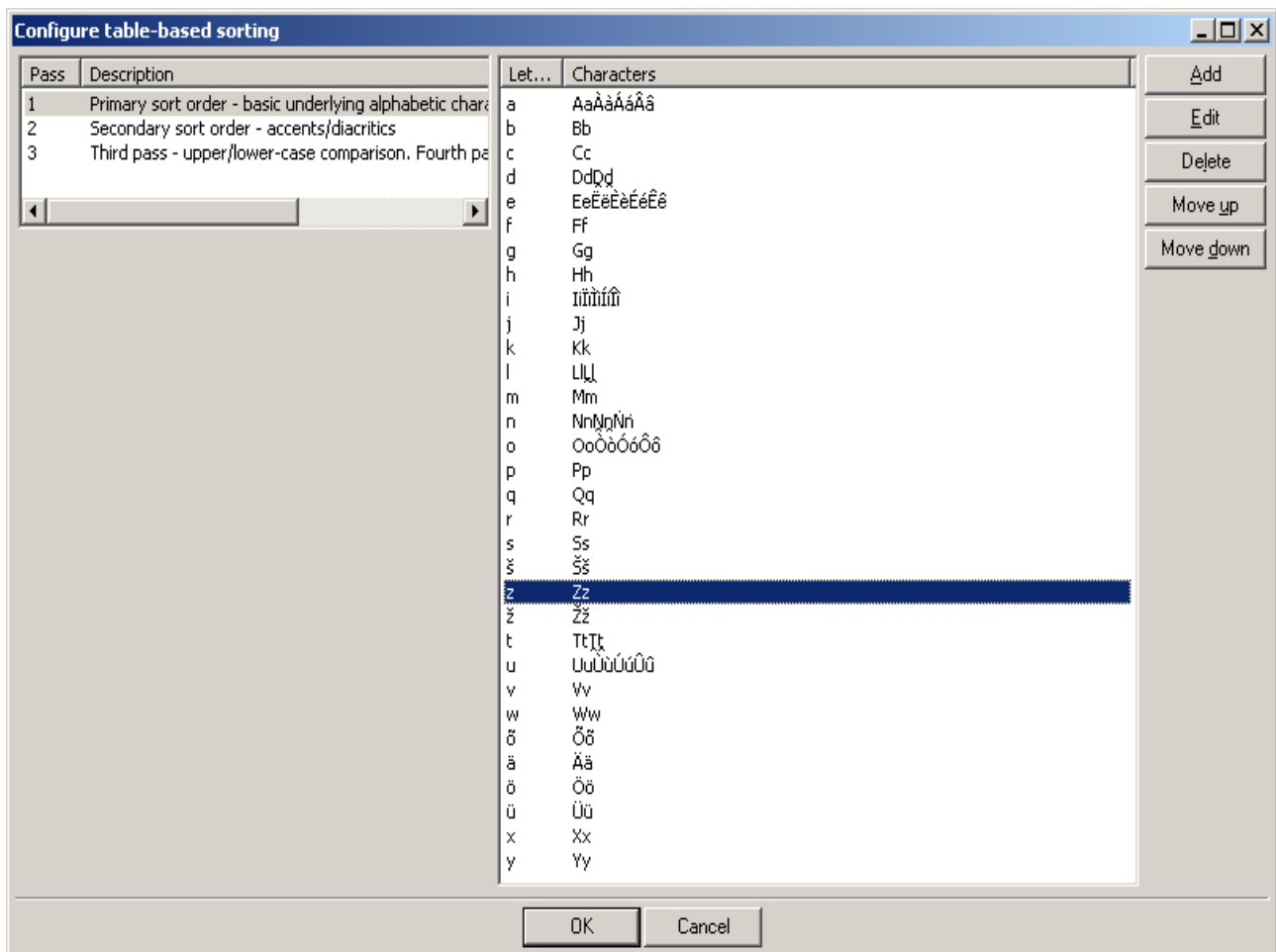


Figure 25: First table/pass of sorting configuration for the Estonian alphabet

In order to edit the tables that define the sorting behaviour for the table-based sorting system, select the menu option “Dictionary/Configure sorting”, then click on the table-based sorting method in the list under “Configured sorting methods”, and click on the “Configure ...” button. The configuration dialog for the table-based sorting will appear, allowing you to define alphabetic characters, their ordering, and the precedence of for instance diacritics or case differences.

Note: Table-based sorting is rather complex; a good way to familiarise yourself with the system is to study and depart from the existing default configuration.

Selecting Other Sort Plug-ins

The table-based sorting system is just one “sorting plug-in” in TshwaneLex. A different plug-in would be used to sort, for example, Chinese characters.

In order to make use of a different sort plug-in, or even to use a differently configured instance of the same table-based sorting plug-in, firstly open the “Configure sorting” dialog, shown in Figure 26.

The bottom half of the sorting configuration dialog displays a list of all sorting configurations which have been set up for this dictionary. The top half of the dialog displays which of these sorting configurations are actually *currently in use* by the language(s) in your dictionary. In the above screenshot, depicting an Estonian – Chinese bilingual dictionary, the Estonian side of the dictionary

has been configured to use the table-based sort plug-in, while the Chinese side of the dictionary has been configured to use the “Chinese - Radical / stroke count” plug-in.

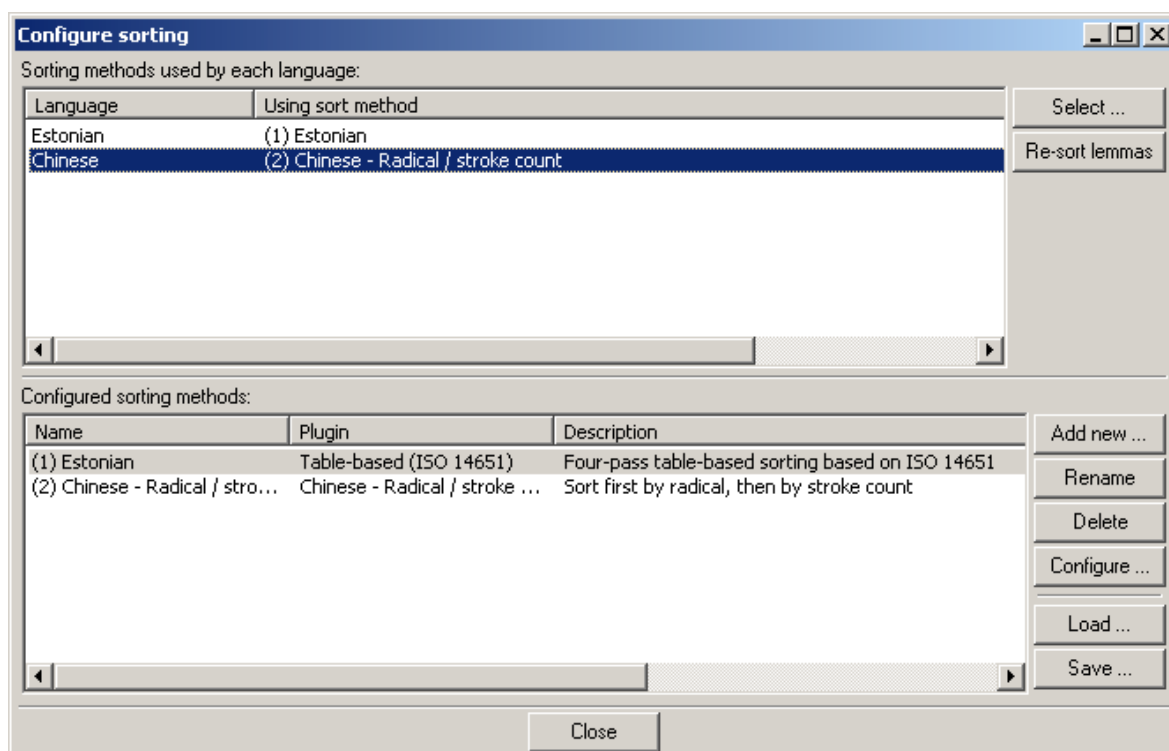


Figure 26: Sorting configuration dialog

The steps for adding and selecting the Chinese sorting method are as follows:

- Click on the “Add new ...” button.
- Select the desired plug-in to use from the list, namely “Chinese - Radical / stroke count”.
- Click “OK”.
- Optionally enter a name for this sorting configuration, e.g. “Chinese sort”.
- In the “Language” list in the top half of the dialog, click on the language you would like to have sorted with this newly selected plug-in, namely “Chinese” in this case.
- Click on the “Select ...” button to choose from the list of “configured sorting methods” in the bottom half of the dialog.
- Select the sorting configuration from the list that appears, here “Chinese sort”, and click “OK”.

The Chinese side of the dictionary will then automatically be resorted with the new sorting method.

Note that you *must* first configure a sorting method in the bottom half of the dialog *before* you can select it for use in the top half of the dialog.

Also note that, unlike the table-based sorting, the Chinese sorting plug-in has no further end-user configuration options; hence, the “Configure ...” button will be disabled for this sorting method.

Loading/Saving Sort Configurations

Once you have configured a sort method, you may want to re-use the configuration for other projects. The “Save ...” and “Load ...” buttons allow you to save the current configuration to a “TshwaneLex sorting configuration file”, or load a previously-saved configuration.

Styles System

Overview

The Styles System is used to “transform” the structured data of an entry into the corresponding output in the Preview Area (and when exporting to formats such as RTF, HTML, etc.):

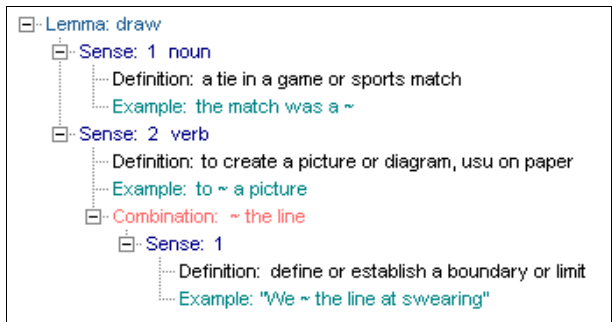


Figure 27: Structured data of an entry in the Tree View

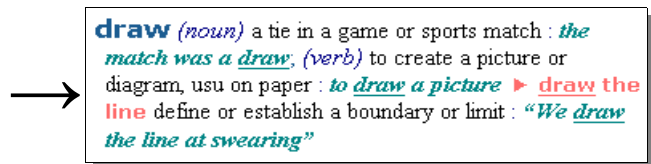


Figure 28: Corresponding formatted entry in the Preview

Every element and every attribute in can have its own style. The styles can be modified by opening the **Styles dialog**, accessible from the “Format/Styles...” menu option. The Styles dialog consists of three tabs: “Styles/formatting”, “Output (display) order” and “Entities”.

Styles/formatting

In addition to being available from the “Format/Styles...” menu option, the “Styles/formatting” tab in the Styles dialog can also be accessed by clicking the “Styles/formatting” tab at the top of the DTD editor dialog, and even by right-clicking on any element in the Tree View and choosing the “Modify styles/Edit style” menu option. The “Styles/formatting” tab (see the screenshot below) shows a list of all available styles on the left, under the header “Styles”. Element styles are highlighted in grey, and the attribute styles for each element are shown beneath it. Selecting an element or an attribute from this list enables one to modify its style properties. Note that when this dialog is called up from the Tree View, the dialog will open with the styles/formatting of the element one has clicked highlighted.

Basic Formatting Options

A “style” allows various formatting options (such as font face, font size, bold, italics, underline, superscript, small caps, etc.) to be chosen for each element or attribute. The style can be changed in one place, and the change will immediately take effect throughout the entire database for all occurrences of the corresponding element or attribute. These style changes are also applied immediately in the Preview Area, thus one may see the effects of style changes *while* one is changing the styles. (Resize and/or move the window of the Styles dialog to see (part of) the Preview Area.)

In this regard the tlDatabase Styles System thus functions in a similar way to the styles systems found in word processors such as OpenOffice.org and Microsoft Word. Further, when generating RTF output, corresponding styles are created in the word processor document, allowing the styles to be further manipulated, if necessary, in those packages.

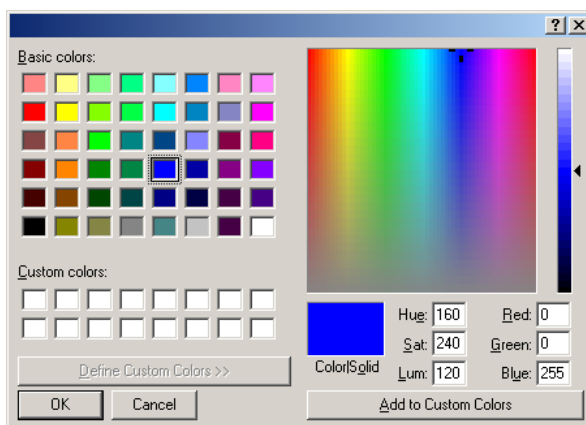


Figure 30: Colour chart for selecting foreground and background colours of elements and attributes

Text Before/After Options

In addition to formatting options such as bold and italics, and the selection of colours, common text/punctuation that should appear before or after specific elements or attributes can be configured with the style. This can be used, for example, to precede the recipe category list with the text “Categories:”, or to always place this field in square brackets. By using styles, this sort of text/punctuation may be changed throughout the entire database at once, simply by changing the style. The user is also freed from entering such text/punctuation manually, over and over. Using the text before/after options thus saves time, ensures consistency, avoids typing errors, and provides extra flexibility.

Note that for each style, there are two “Text before” and “Text after” options. The first of these, labelled “uses style formatting”, will be output *with* the formatting options of the style (e.g. **bold**, **magenta**) applied. The other options, labelled “before style formatting” and “after style formatting” will be output before/after the formatting options of the style are applied.

For example, in Figure 29 the text “%nE-mail: ” has been typed into the “Text before (before style formatting)” input box. This means that each “Email” attribute will start on a new line (%n), and will be preceded by the text “E-mail:” followed by a space. The formatting “Courier New” and “Underline” will not be applied on this “Text before” section, however. (Note that the font size is taken from the parent, “Contact”, in this case “10 pt”.) Again, see Figure 5 for the result in the Preview Area.

Note: All formatting markup characters (%n, %r, etc.) can be used within the Styles System. For a list of the different markup characters, see the section Using Text Formatting Within Attributes, in the chapter on Editing Attributes.

Group Style Properties for Multiple Elements or List Items

Group style properties are edited in the bottom right area of the Styles/formatting tab, under the heading “Style properties for groups of multiple child elements or multiple list items”. These options allow one to specify the separator character to use between elements when there is a “group” of elements, that is, when an element in the document tree contains multiple child elements of that type. For example, one may specify here that multiple translation equivalents (“TE” elements) under each “Sense” element should be separated by commas, or that multiple senses under a lemma should be separated by semi-colons.

In addition to specifying a separator character, the group style options also allow one to configure text that should appear before or after the entire group of child elements of the same type. This can be used, for example, to precede every phrasal-verb section of an article with “PHR V”.

The group style properties are also used to specify before/after text and separator characters for “list” type attributes that allow multiple list items to be selected from attribute lists, such as the part of speech attribute of the lemma and sense elements. Note that the group style properties have no effect for other non-list attribute types.

Automatic Numbering

A comprehensive “auto-numbering” system allows automatic numbering to be assigned to *any* element type, by means of an “Auto-number (Integer)” attribute type. So, for instance, to number the various senses of a polysemous lemma automatically, one will first have to create an “Auto-number (Integer)” attribute for the “Sense” element, as shown below:

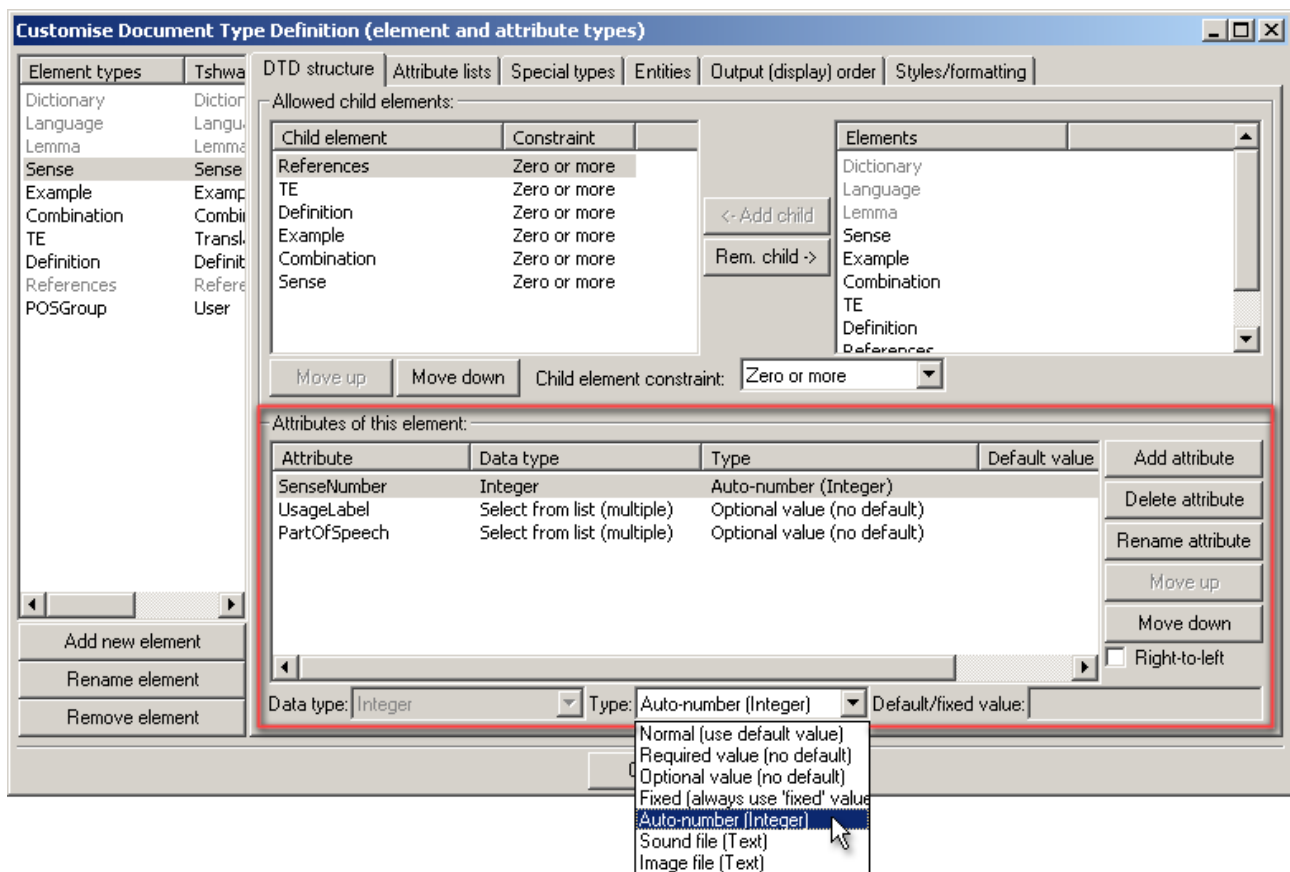


Figure 31: “DTD structure” tab, Attribute editing window: Selecting the “Auto-number (Integer)” attribute type, used for automatic numbering of any elements

The type of numbering scheme to be displayed in the output can be configured in the Styles System, via the “Numbering” button on the “Styles/formatting” tab.

So, for this example, if one first clicks on the “SenseNumber” attribute under the “Sense” element on the “Styles/formatting” tab, the following dialog will pop up:

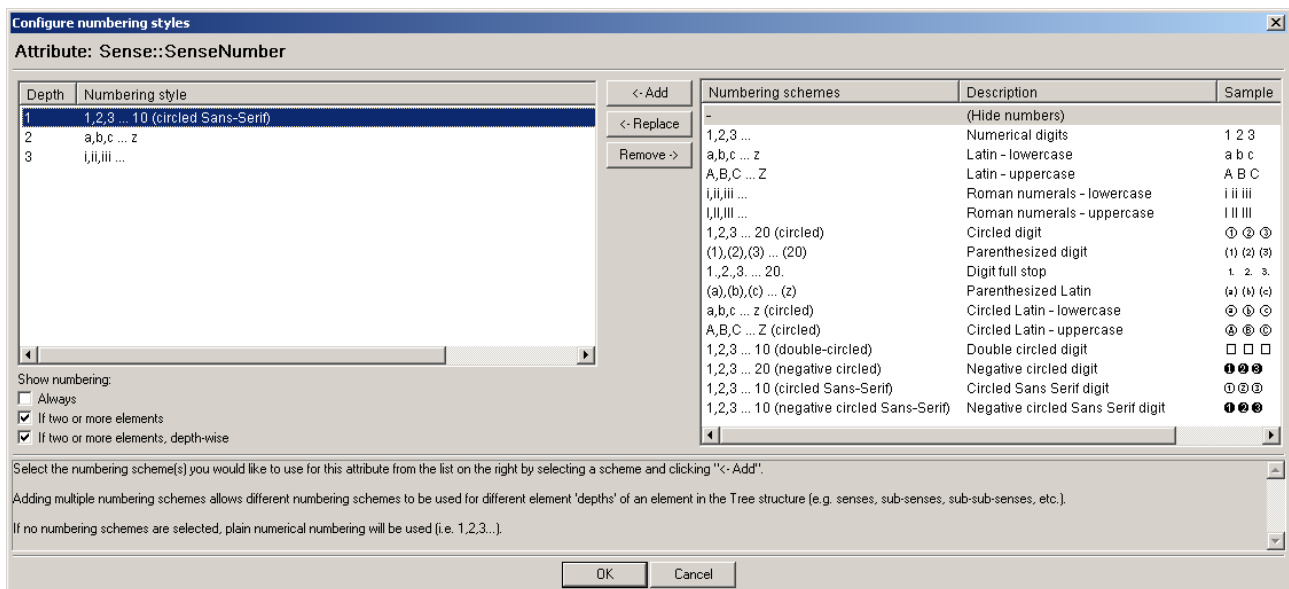


Figure 32: Numbering styles dialog, used to select numbering schemes for “Auto-number (Integer)” attribute types

The list on the right shows various numbering schemes, and clicking the “<- Add” button selects one of them. Selecting several (different) numbering schemes allows you to number the different Tree View levels of an element.

For example, if in a particular dictionary a sense can have subsenses, and these subsenses can again have subsenses, then one will need three levels of numbering. In the screenshot the “circled Sans-Serif digit” numbering scheme has been chosen for level one (see e.g. Figure 7), “Latin – lowercase” for level two, and “Roman numerals – lowercase” for level three. One can of course choose the same numbering scheme for all levels, but choosing different ones generally increases readability.

There are also options to always show the numbering (thus even if there is only one sense in a particular article, that sense will always be preceded with a number/letter “1”/“a”/“A”/“i”/...); or to only start numbering from the moment there are at least two items on the same level in a particular article; or to only start numbering when there are at least two items on different levels in a particular article.

tlDatabase automatically recalculates the numbering for “Auto-number” attributes whenever you add, move or remove their associated elements in the Tree View.

For homonyms, plain numerical digits are used by default (see the DTD in Figure 19). If one would rather have homonyms ‘numbered’ as, say, “a”, “b”, “c”, etc., then one can simply open the “Styles/formatting” tab in the Styles dialog, mark the “HomonymNumber” attribute, click on the “Numbering” button, and then select the “Latin – lowercase” numbering scheme. (Adding extra levels of numbering schemes will, of course, have no impact for homonyms.)

Note: For some numbering schemes to be displayed correctly, such as the “circled Sans-Serif digit”, one will need to choose a font for the auto-number attribute (on the “Styles/formatting” tab in the Styles dialog) that supports it. (A font that supports all but one, cf. Figure 32, is “Arial Unicode MS”.) In general, if you see **hollow squares** (□) it means that the selected font does not support the required Unicode characters. Please try to use a different font in that case.

Output (Display) Order

Element and Attribute Output (Display) Order

The *output (display) order* of elements and attributes in the Preview Area (and formatted output such as RTF, HTML, etc.) is configured *separately* to the DTD structure. This allows some separation of the underlying data structure and the final visual output. The output order of elements is configured using the “Output (display) order” tab in the Styles dialog (or the DTD editor dialog).

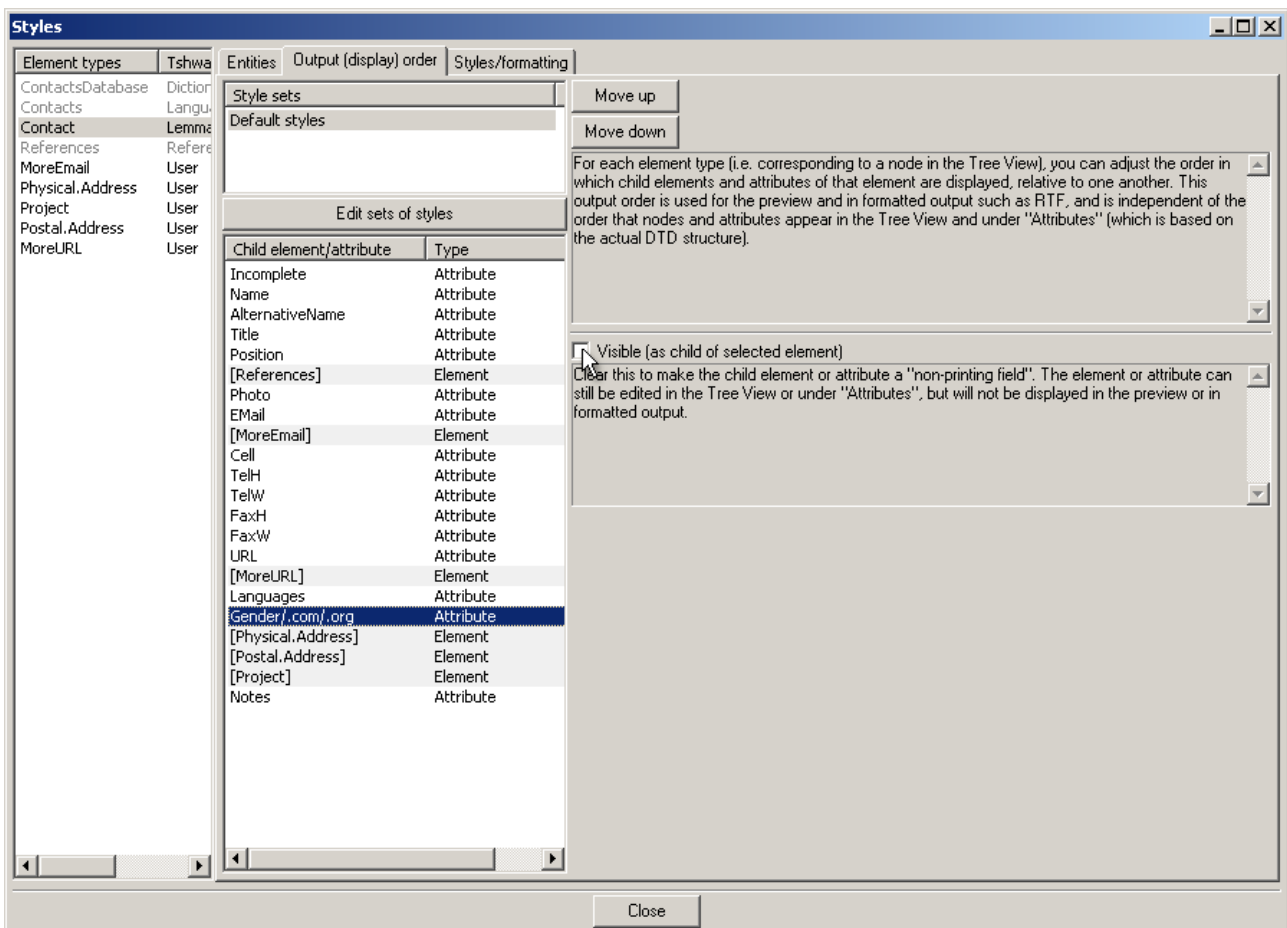


Figure 33: “Output (display) order” tab in the Styles dialog:
Configuring the output order of child elements and attributes of the “Contact” element

For each element type, one can configure the output order of (a) *child elements* of that element type, and (b) *attributes* of that type, all relative to one another. As an example, consider the “Contact” element type in the above screenshot, which has a number of attributes such as “Name”, “Photo” and “Notes”, as well as a number of child elements such as “References” and “Physical.Address”. Some of the contact attributes (such as “Name”) should appear *before* the “References” and “Physical.Address”, other contact attributes (such as “Photo”) should appear *after* the “References” but *before* “Physical.Address”, while the “Notes” attribute should even appear at the *very end* of the entry.

When the “Contact” element is selected (from the list on the very left of the “Output (display) order” window), the list of all child elements and attributes of the “Contact” element is shown, and the output order of these can be modified by using the “Move up” and “Move down” buttons. One can thus specify that for the “Contact” element, the name should be displayed before any child

references of the contact, that the photo should be included between the references and the child addresses, but that the notes attribute should be displayed at the very end of the entry.

Observe that, like the styles, changes made here are applied immediately in the Preview Area.

“Visible” Flag (Non-Printing Fields)

Each child element or attribute in the output order configuration may also be selected to be hidden in the Preview Area or output. This can be done by toggling the state of the “Visible” checkbox. This allows fields to be marked as “non-printing fields”, i.e. fields that should not appear in the dictionary output. In the above screenshot, for example, the “Gender/.com/.org” attribute has been marked as a non-printing field.

When a child element or attribute is hidden, it *will still* be shown in the Tree View or under “Attributes (F1)”, and it can still be edited – it is only disabled in the *output*. See in this regard Figure 5.

Note that it is also possible to quickly toggle an *element* type as visible/invisible by right-clicking on the node in the Tree View and selecting “Modify styles/Toggle visible” from the menu.

Entities

Entities can be used to define shortcuts to common text. These are then automatically substituted by the entity’s associated corresponding text value in the output wherever the entity occurs in the dictionary. Entities can be used to enter characters not usually easily accessible on a keyboard (e.g. you can enter “©”, and tlDatabase will automatically replace this when creating the output with the copyright symbol “©”). Entities are entered by typing an “&” character, followed by the character name, followed by a semi-colon. Examples of other entities are “£” for the pound symbol “£”, and “…” for the ellipsis character “...”.

In tlDatabase, you can view the list of entities, or add or remove entity types, using the “Entities” configuration dialog, available by selecting the “Entities” tab at the top of the DTD editor dialog (“Database/Customise DTD”).

By default, entities are automatically substituted with their associated values in the Preview Area. This can be toggled under “Format (F4)” with the “Expand entities” checkbox.

Using Entities to Embed Labels within Other Fields and Further Customisation

Entities can also be used for situations where labels need to be embedded within other fields, as is the case for, for example, the French ‘f’ (“feminine noun”) and ‘m’ (“masculine noun”) gender labels. In a bilingual English – French – English dictionary one may for instance find the following entry:

contrivance (*noun*) ① [*tool, machine*] appareil^m,
machine^f; ② [*scheme*] invention^f, combinaison^f

In an electronic environment, one would want users who click on the ‘f’ and ‘m’ gender labels to be sent to the correct meanings of those labels, rather than to random abbreviations. In existing electronic English – French – English dictionaries it is not uncommon to click on the ‘f’ and ‘m’

gender labels, and to be presented with an array of possibilities. Following up on each of them, may lead to:

- Clicking on ‘f’ may lead to:
 - **F, f** ... *nom masculin* ... F, f
 - **F** ... *abréviation* ... F, fr ... F
- Clicking on ‘m’ may lead to:
 - **M, m** ... *nom masculin* ... M, m
 - **me, m’** ... *pronom personnel* ... me; myself
 - **m** ... *abréviation* ... m
 - **m’** ► **me**
 - **M.** .. *abréviation* ... Mr

In other words, if one does not already know what these abbreviations stand for, one receives *no* guidance at all, with the first option for ‘f’ as “nom masculin” definitely confusing.

To avoid such problems, the ‘f’ and ‘m’ gender labels could be defined as entities (e.g. ‘&f;’ and ‘&m;’ respectively) which are replaced in the output with labels configured in a *single, central place*, i.e. on the “Entities” tab, as shown below:

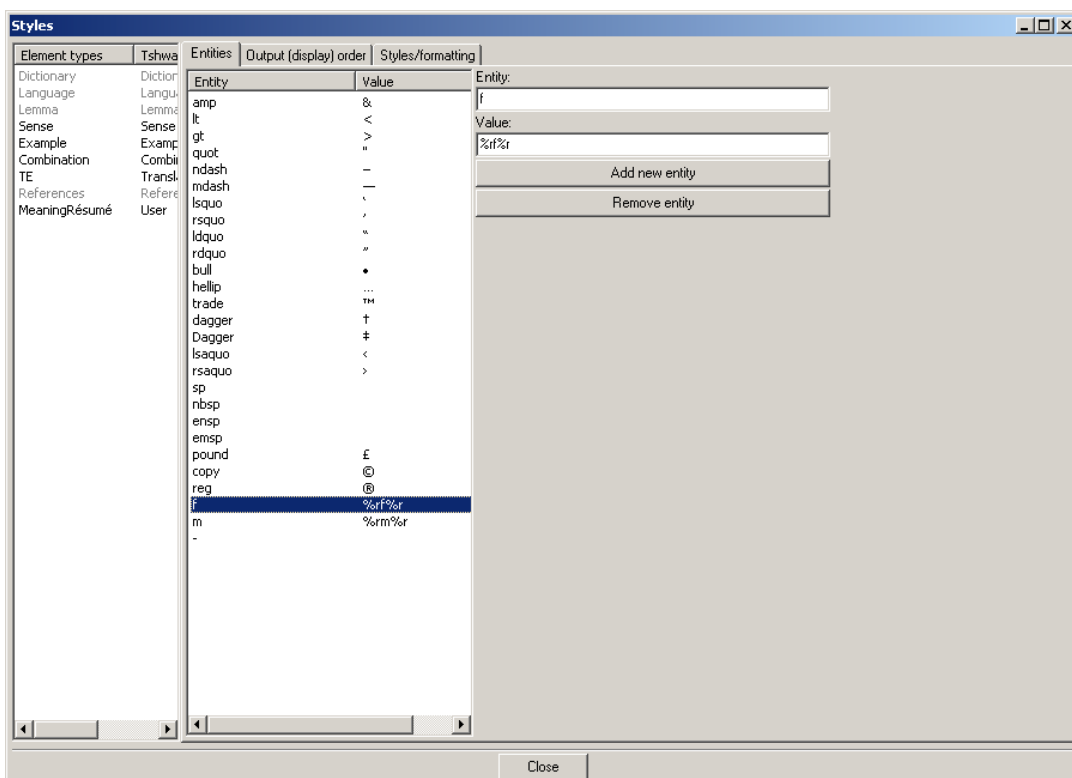


Figure 34: “Entities” tab in the Styles dialog: Adding new entities for the ‘intelligent’ use of gender labels, in this case the raised ‘f’ for “feminine” and the raised ‘m’ for “masculine” [with ‘%r’ for “superscript”]

In the next screenshot, the use of the ‘f’ and ‘m’ gender labels as entities is illustrated.

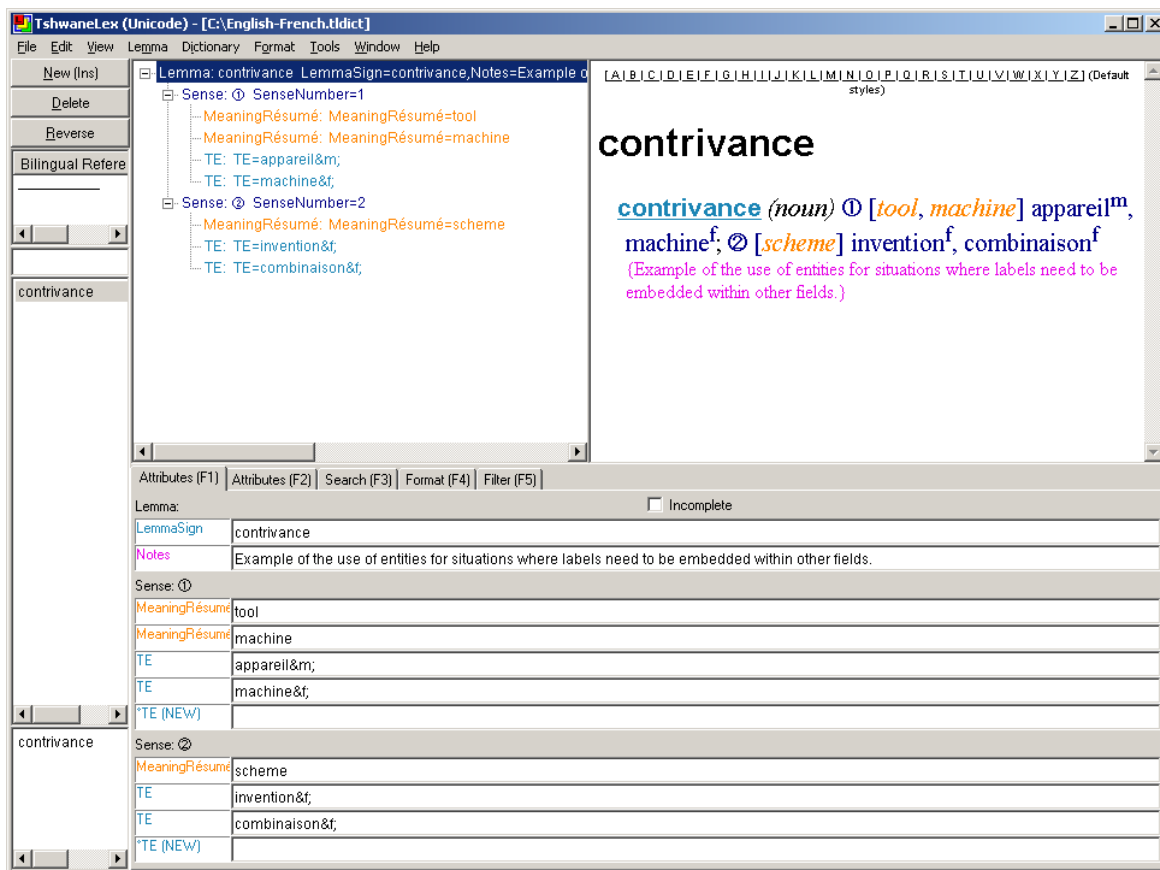


Figure 35: Example of the use of entities (here ‘&f;’ and ‘&m;’) for situations where labels need to be embedded within other fields

Using entities for such cases allows the software to be *aware* that these labels are part of the metalanguage, and to thus provide a more meaningful response should the user click on them. A powerful *extra mechanism for customisation* is moreover provided, as the language/appearance of the labels can, again, easily be adapted in one single place.

Style Sets, Or ‘One Database, Many Dictionaries’

Not only can the dictionary grammar for any project be flexibly configured and then kept under control with the customisable and multilayered DTD editor dialog, given that all elements and attributes are also linked to a comprehensive Styles System for generating the output (and preview), one single database can efficiently hold several dictionaries. This outcome has been taken one step further, and functionality has been added to tlDatabase with which it is possible to set up **multiple sets of styles** and to toggle between them. There is no upper limit as to the number of sets of styles that can be defined in tlDatabase.

Broadly speaking, this is achieved by doing two things. Firstly, by making use of multiple element “categories” to which the various data attributes are assigned by the user depending on which dictionary or dictionaries they should appear in, and secondly by defining a different set of styles for each ‘view’ of the database, i.e. for each dictionary. Certain element categories are made visible or invisible in each style, which thus effectively functions as a kind of ‘mask’ that filters and reveals only the portions of data to be shown for the current database. Additionally, this also allows a different ‘look’ to be defined for each sub-database.

In other words, the multiple sets of styles available in a *single* database, allow for the *simultaneous* compilation of a *multitude* of sub-databases.

Given that both the styles/formatting and the output (display) order can be manipulated for each “style set”, both the “Styles/formatting” and the “Output (display) order” tabs of the Styles editor (“Format/Styles...”) have a “Style sets” section in the top-left corner, as shown on the left screenshot below:

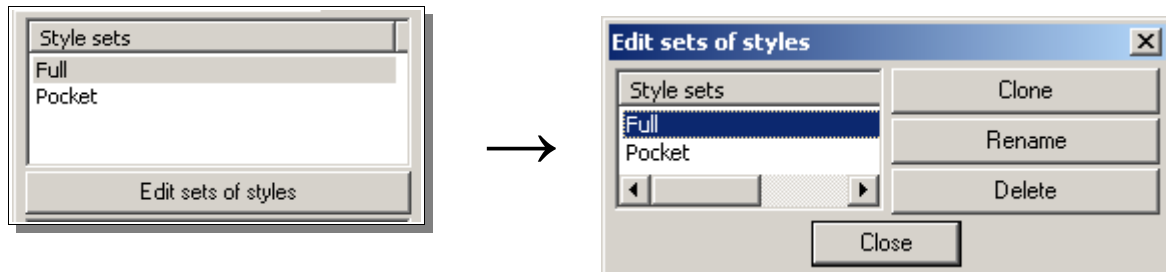


Figure 36: “Edit sets of styles” under the “Format/Styles...” menu option

Clicking on the “Edit sets of styles” button brings up a new dialog, shown on the right screenshot above, where a new style set can be configured starting from a clone of an existing style. Given a detailed-enough DTD, all the features discussed in the present chapter can be applied in creating multiple views.

With Ctrl+P (or “Format/Toggle current style set”) one can toggle between the different style sets. If one has prepared, say, five style sets, then one can ‘run through’ all of them by pressing Ctrl+P consecutively.

“Smart Styles” (Dynamically Customisable Styles) [Advanced]

tDatabase allows you to set up rules (of any possible complexity) that allow the visual appearance (style) of a field to change dynamically in different situations. As a simple example, in a dictionary, you might want the automatic punctuation before a usage example to change depending on the field that happens to precede it. Or, you might want to automatically generate a fullstop after the example only if it doesn't end on an exclamation or question mark.

All of these things can be achieved by attaching a “Lua modifier script” to a style. This can be done under the “Advanced” tab in the Styles dialog.

Importing Data

Import Wordlist / CSV (Comma Separated Values)

Many new entries can be added to the database at once using the “File/Import/Wordlist or CSV (Comma Separated Values)” menu option. This imports data in one or more columns (one entry per row) from a CSV file saved from spreadsheet software such as OpenOffice Calc or Microsoft Excel. Each column can be “mapped” to a certain attribute in the DTD while importing.

In order to assign attributes to columns, one must **double-click** on them in the left-side list of Element::Attribute names. The same attribute can be added more than once. To remove an assigned attribute, double-click on it in the right side list.

Below, an example is shown of the import of three columns of data from a CSV file for a dictionary database (“Lemma” is the term for an entry in dictionary-making):

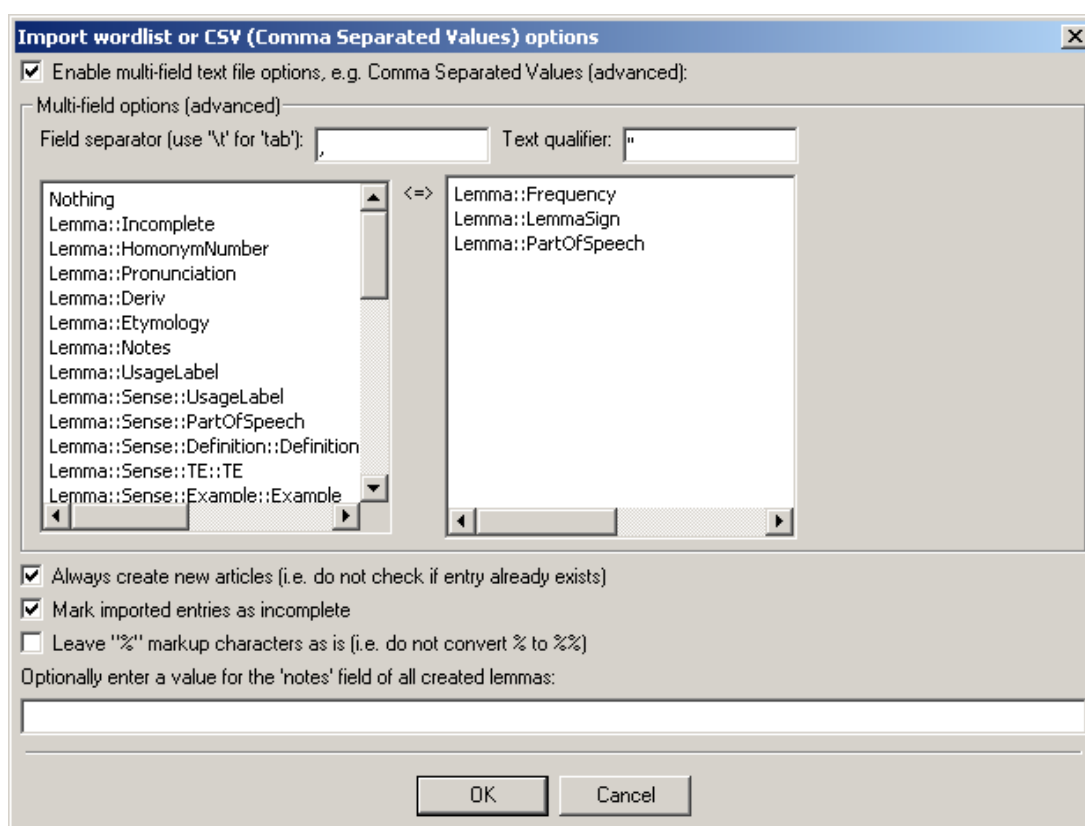


Figure 37: Import wordlist or CSV (Comma Separated Values) options dialog, here used to import three columns of data (frequencies, dictionary headwords, and parts of speech)

In the above example, a new entry will be created for every “lemma sign / part of speech pair” attribute (cf. the “Always create new entries” option), and all imported data will be marked as incomplete. Importing data can be repeated as often as one wishes, and at any stage.

Note: The “Import Wordlist/CSV” tool correctly handles **Unicode** files, and includes an option to allow for the use of t!Database **formatting markup characters** (%i, %k, etc.) in the source text/CSV data. For the latter, see the section Using Text Formatting Within Attributes, in the chapter on Editing Attributes. Also note that when importing several columns, the data can

immediately be placed in the correct fields of the **DTD**. For the latter, see the chapter on Customising the Database Structure using the DTD.

Note also that some software, such as older versions of Excel, are not able to save Unicode/UTF8 format CSV files.

Importing XML [Advanced]

Data in XML form can be imported into tlDatabase via the "File/Import/XML" menu option (provided, to a reasonable degree, that it is well-structured).

It is possible to both import “clean” (i.e. as a new, blank document), or do a “merge” import into an existing database. It is usually best to import data into a 'clean/empty' document, i.e. to select the XML import command when no database is open in tlDatabase.

Note that after importing XML, there would usually be no tlDatabase styles, thus all imported entries will usually be displayed in a default text style in black on a white background. You can use the "Format/Styles" menu option as usual to add styles once you are satisfied with the import.

tlDatabase has one or two basic 'expectations' of how the data should be structured in order to import the data in a meaningful way (i.e. in a way that allows tlDatabase to 'understand' what some of the key fields are). The following is an example of roughly the simplest XML document that can be thrown at the importer:

```
<DocumentRoot>
  <Entries>
    <Entry Name="cow">
    </Entry>
  </Entries>
</DocumentRoot>
```

Note that the element for main entries appears at the *third depth level* in the document. The names of the elements can be anything, although their structure is important (i.e. second-level element represents each 'section' within tlDatabase, and third level represents the list of entries within that section).

Note that you do not necessarily need a DTD attached to the data - if importing XML data with no DTD, tlDatabase will attempt to construct a DTD based on the elements/attributes it encounters. For well-structured data, this can often work well.

Post-Import Processing / Data Remodelling

For various reasons, it is seldom the case that freshly imported XML data can be used “as is”; one often has to do various kinds of clean-up, processing and data remodelling in order to make the data more useful. Depending on what one wants or needs to do, this can often be non-trivial, and involve e.g. specialised Lua scripts to manipulate the data. If you are in doubt, TshwaneDJe is highly experienced in dealing with these kinds of issues; contact the company to discuss your requirements.

Exporting Data

Copying To the Clipboard

Copy Entry Text

To copy the full entry text (of the entry you are working on) to the clipboard, in text-only format, use the “Edit/Copy entry text” menu option.

Copy Entry HTML

To copy the full text (of the entry you are working on) to the clipboard, in HTML format, use the “Edit/Copy entry HTML” menu option.

Exporting the Database, in Part or in Full

Under the “File/Export” menu option, various ways to export your data are available. These are:

Text

Use this to export the text, in text format, of your database. The general export options shown above are available.

A possible use for this type of export is with spellchecking purposes in mind. (Corrections will of course have to be made in the database itself.)

RTF (Rich Text Format)

This is the most common way to export data, as the text *formatting* (in black-and-white) is saved as well. See the section on Printing your Database, in the chapter Getting Started with t1Database: QuickStart Guide, for more information. Various format options, as illustrated below, are available:

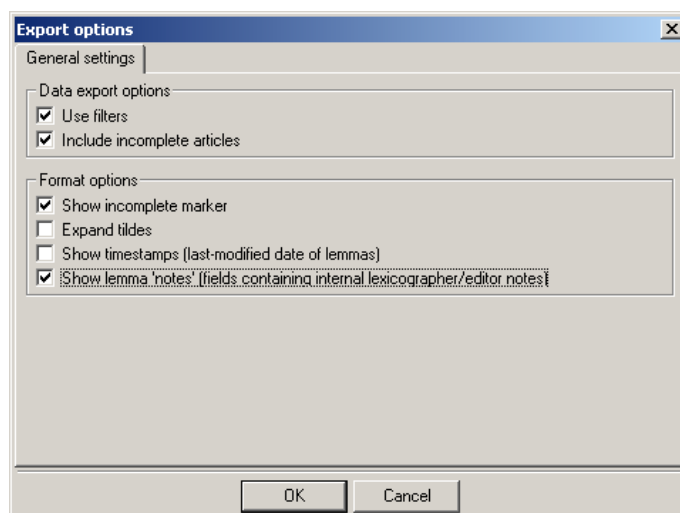


Figure 38: Export options dialog, General settings: Data export and format options

Note that if one has included images in the database, the RTF output and the image files must be stored in the same folder.

HTML (Web Page)

Use this to export the data, in HTML, when the intention is to place the result online as a web page, or as a set of web pages. In addition to all the general settings as discussed above, there are several HTML-specific options that are available.

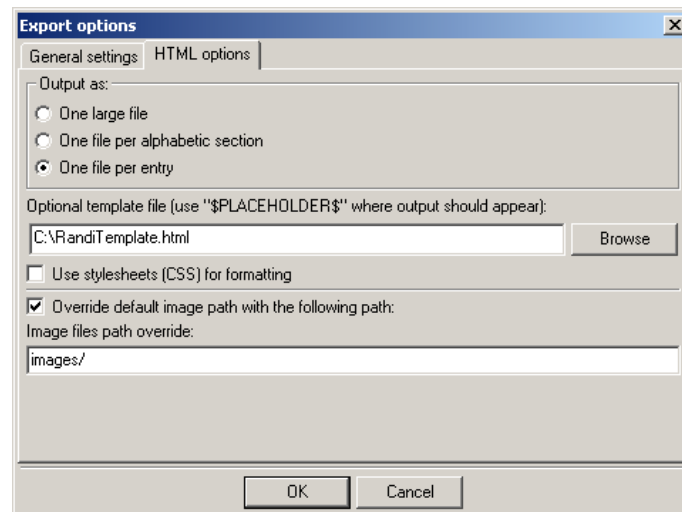


Figure 39: Export options dialog, HTML options

The first of the HTML-specific export options allows you to select to export the data either as one large HTML file, or to split the output up into alphabetic sections, or even into one file per article.

A “template HTML file” can optionally be specified, in which the output will be “dropped” at the “\$PLACEHOLDER\$” text.

Cascading Style Sheets (CSS) can also optionally be used.

Images are typically stored in an “images” folder online; use the relative path to it, to override the path stored under “Database/Properties”.

XML (eXtensible Markup Language)

Use this option if you want to export your data to the open, standard XML file format.

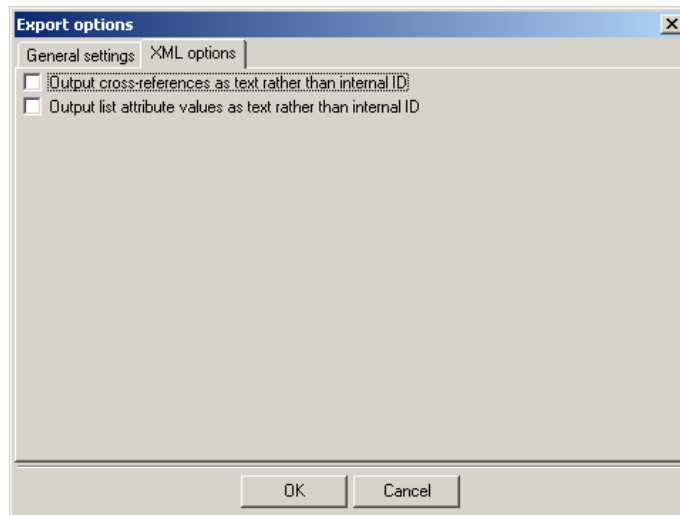


Figure 40: Export options dialog, XML options

XML (Formatted)

The plain “XML” exporter outputs your data in its *internal, structured* format. The “Formatted” XML exporter applies the basic Style and output order transformations to your data, and thus exports your data far more closely to how it appears in the *Preview Area*, but semantically tagged.

This exporter is most useful if preparing data for, for example, typesetting, as it can be brought into typesetting software and then easily styled, as each field is semantically marked up.

Note that, unlike the “plain” XML exporter, the “Formatted” XML output does *not* lend itself easily to being imported back into the software, as much of the *structural* information is lost.

Network (ODBC / Relational Database) Support [Advanced]

Configuring an ODBC Database

ODBC (Object Database Connectivity) is a standard interface for connecting applications to database server software, such as PostgreSQL or Microsoft SQL Server. To save a new or existing dictionary document to a database, select “File/Save as: ODBC database” from the menu. (Note that the empty database must already have been created, and the access rights and “ODBC data source” set up accordingly. In Windows, the data source is set up under “Control Panel/Administrative Tools/Data Sources (ODBC)”.) tlDatabase will prompt for two things before saving the database:

1. ODBC data source name: The name of the configured ODBC data source.
2. Table name prefix: All table names in the schema may optionally be given a prefix. This allows multiple tlDatabase documents to be stored in one actual relational database on the server (this is sometimes useful in cases where, for example, the number of databases available is limited by a Web hosting plan). Unless otherwise desired, it is safe to leave this on the default “tl_”.

Depending on the database type and ODBC driver, you may also be prompted for the database username/password.

If user management has not yet been set up for the project, tlDatabase will also display the user management configuration dialog, in which you may configure additional users. **NB: By default, a single “Admin” user is created with the password “Admin”.** See the chapter on User Management for more information. Note that for ODBC databases, you cannot disable tlDatabase user management.

Once you have initially saved the ODBC database, you can open and work on it largely as if it were an ordinary tlDatabase file.

“Cached” ODBC (ODBC, Sped Up)

It is generally far better to use the ODBC interface labelled 'ODBC (with local cache)' - this is basically the ordinary ODBC interface, but additionally transparently keeps a local cached copy of entries that haven't changed on the hard disk, allowing tlDatabase to dramatically speed up general work and activities like searching and filtering. The first time the entire database must be loaded will be slow; thereafter, it will be much faster, as it will only re-retrieve entries when they change.

Should you need to reset the cache, the cache folder lives in your 'temp' folder, typically something like 'c:\Documents and Settings\YourUsername\Local Settings\Temp\TshwaneDJe_Cache' - just close tlDatabase and delete the TshwaneDJe_Cache folder.

Entry Locking

The main difference between working on a standalone tlDatabase file and working on an ODBC database is the entry locking system in the latter. Entries are by default protected from casual editing, and displayed in the Preview with a padlock next to them. When you attempt to modify an entry, it will ask you if you are sure, and if you agree it will attempt to “unlock” the entry, 'checking it out' to your user. If successful, the padlock displays as unlocked and with a green tick. While you

have an entry checked out, no other users will be allowed to check it out. If another user has an entry checked out, it will be displayed with a red cross through the padlock, signifying that you may not unlock it. Other users are thereby prevented from simultaneously making conflicting changes.

When you save your changes, they are committed back to the database, and the saved entries are 'checked back in', allowing other users to modify them once more.

Locking the Database

Certain types of administrative tasks, such as editing the DTD, require that nobody else be logged on to the database at that time. In order to lock all users out, select the “Dictionary/Lock database” option. If any users are connected to the database, a dialog will be displayed showing which users are logged on, while those users will also be prompted to save their work and log out. Once all users are logged out, the dictionary will be locked and you may proceed to work on it. While the dictionary is locked, nobody else may log on, and only the user who locked the database may unlock it.

Do not forget to unlock the database once you are done with the required administrative tasks.

Optimisation Tips

Working on an ODBC database may feel sluggish, particularly if the server or network is not fast. This is because entries must first be loaded from the database before they can even be displayed, and this is done dynamically and on an ongoing basis in order to ensure that you are viewing the latest data. One thing that can help a lot if the system feels slow, is to select the “Format/Preview selection only” option. Also, make sure to use the “cached” ODBC (mentioned above).

NB: It is also crucial to ensure that database indexes are running optimally. In PostgreSQL, for example, one should periodically run an “ANALYZE” on the database – this can greatly speed up queries.

Commandline Options [Advanced]

The following options allow the application to be driven and controlled via the commandline.

`--nouserinteraction` : Disables user interaction for the session.

`--runluascript` : Automatically runs the specified Lua script as soon as the application has opened (and after opening any documents also specified on the commandline). You can specify multiple `--runluascript` items, and they will be executed in the order specified.

`--autoexit` : Causes the application to immediately exit after opening. If used in combination with other options, this allows the application to be automated to perform scripted commands, e.g. one could have a batch process that automatically opens some file, performs some processing (e.g. via Lua script), and then exits.

`[filename]` : You may optionally specify a document filename to be automatically opened. If you specify a `.xml` file, the built-in XML importer will be used.

Watch Folders [Advanced]

On a per-project (per-document) basis, you can configure one or more folders in which any XML files saved to that folder, will automatically be imported into that document as long as it remains open. There are numerous settings that control the watch folder behaviour; these can be configured, and are documented in more detail, under either 'Dictionary/Properties' or 'Termbase/Properties', under the 'Properties [Advanced]' tab.

WatchFolders

Watch the given folder(s) for incoming files to be automatically imported (semi-colon separated list)

MoveImportedFilesToFolder

If specified, move successfully imported files to this folder

MoveFailedFilesToFolder

If specified, move unsuccessfully imported files to this folder

OnSuccessfulImport

0 = Move to MoveImportedFilesToFolder folder, 1 = delete file

IgnoreFilesAlreadyImported

Don't import the same file again if it's already in the MoveImportedFilesToFolder folder

ImportFileTypes

File types to automatically import (semi-colon separated list)

CheckIntervalMilliseconds

Number of milliseconds (thousandths of a second) between polls of the watch folders. Thus 5000 = five second interval.

AutoSave

Automatically save document when new data has been imported

Lua Scripting [Advanced]

tlDatabase includes built-in programmability via integration of the open source Lua scripting language.

Getting Started with tlDatabase Scripting

The two main types of Lua script applications in tlDatabase are:

1. **Lua Script Attributes:** A Lua script attribute is just another attribute in the document, but is special in that instead of ordinary text or numerical data, it contains a piece of Lua code that is executed in order to generate the resulting output for that attribute value. This is similar in principle to 'formulas' in spreadsheet applications such as Microsoft Excel.
2. **Stand-alone Scripts:** Stand-alone (or external) scripts exist in the form of a text file, and are loaded and executed via the "Tools / Execute Lua Script" menu option in tlDatabase.

Lua Script Attributes

A good place to start learning about Lua script attributes is to have a look at the "Lua Scripting Attribute" sample file in the Samples folder. To view the sample's script attribute contents, go to "Database/Customise DTD", select the "Sense" element on the left, then select the "SenseNumLua" attribute under "Attributes of this element", click in the "Default/fixed value" box at the bottom right of the DTD editor, and press F12 to open the script in the overlay editor window. This sample demonstrates the use of a fixed Lua attribute value to generate automatic sense numbering in a manner currently not supported by the automatic numbering styles: Specifically, repeating the sense number of the parent at every subsense (e.g. "2a ... 2b ..."). The return value of the script is the value that is output in the Preview etc. (with the usual styles and so on applied, just as with any other attribute).

Note that for a Lua script attribute you can choose if the attribute value should be "fixed" or not. If "fixed" (as in the sample), the script is entered once - in the DTD editor - and the same script is always executed. If not fixed, a different script can be entered for the attribute on an individual entry/element basis.

To create a Lua DTD attribute, create an attribute as usual in the DTD editor, then select "Lua script" as the "Data type".

Note the use of the `gCurrentNode` global. This value always contains the node (i.e. element in the document tree) that the attribute on which the script is currently being executed belongs to. (Related to this is also a `gCurrentEntry` global, pointing to the owner entry.)

Note how all local variables are explicitly declared as such - in Lua, variables are global unless otherwise specified.

Stand-alone Scripts

Stand-alone (or external) scripts are basically text files containing Lua code. They can be executed via the "Tools / Execute Lua Script" menu option in tlDatabase. By default they use a ".lua" file

extension, though this is not necessary. Such a script can be used to perform pretty much any kind of manipulation on the database.

NB: The text file encoding for Lua scripts *must* always be 'UTF8, without signature'.

API Reference Documentation, Tips, Samples and Further Information

The tIDatabase API (Application Programming Interface) reference can be viewed via "Start menu / Programs / tIDatabase / TshwaneLua Scripting API". Good starting points are the "Detailed Description" sections of tcNode and tcDocument.

Note that we implement Lua 5.0. For general Lua information (e.g. syntax or standard Lua function reference), it is suggested that you make use of the Lua reference manual, available from the <http://www.lua.org/> website.

Always keep in mind the use of F12 to open the overlay editor window when editing scripts - this is virtually indispensable. It is also often useful to edit scripts in a text editor externally, and copy and paste them into tIDatabase.

In our API, indexes are always zero-based unless otherwise specified. Strings are always UTF8.

More information, plus sample scripts, are available at: <http://tshwanedje.com/tshwanelua/>. Several sample files demonstrating various uses of Lua are also available in the 'Samples' project folder.

Other Uses of Lua Within tIDatabase

“Smart Styles”

Lua scripts can be used to dynamically modify styles for particular fields as the output is generated, based on any possible programmable condition. See the section on “Smart Styles” in the “Styles” chapter (and the samples included with tIDatabase) for more information.

Lua Filters

Lua scripts can be created that implement new “Filter (F5)” conditions. These take the form of .lua files, and must be placed in the 'Plugins/Filters/' folder of the application (note that an application restart may be required when copying new scripts into this folder). All .lua files listed in this folder will appear under the list of Filter conditions under “Filter (F5)”, with “Lua” and the name of the file in square brackets. An example of a Lua filter, that is included with tIDatabase, is the “Lua[Modified]” filter, that filters through only entries that have unsaved changes. This is a simple script that looks as follows:

```
-- Return true if entry has modified flag set
if gCurrentEntry:HasChanged() then
    return 1;
end
return 0;
```

Lua Sorting

Custom alphabetic sorting algorithms can be implemented using the “Lua Script Sort” sort method under “Dictionary/Configure sorting” / “Add new ...”. The Lua script should implement a “main” function that takes two strings, as follows:

```
function main(s1, s2)
    -- (Return -1 if s1<s2, 1 if s1>s2, and 0 if s1=s2)
    ...
end
```

It is also possible to augment existing sorting methods in tlDatabase via Lua; the Lua sort algorithm can, for example, modify the incoming strings in some way before passing them on to another configured sort method. This can also be used to implement digraph sorting.

User Management

Configuring User Logins

The user management system allows you to configure user logons for each user that will have access to a database. The user management system can be accessed via the “Dictionary/User management” menu option.

NB: User Management can be enabled for both standalone .tldatabase files and for networked client/server ODBC databases – you do not need a relational database server to use this functionality.

When you first enable user management, tldatabase will create new “last modified by” and “created by” attributes in the DTD at the *entry* level. Once these are present, tldatabase will keep track of who added or modified entries. You can also manually configure attributes like this on sub-elements if you want the system to keep track of who created or modified subsections of entries (e.g. particular senses).

Notes: When logging on to tldatabase, user names are not case-sensitive, but passwords are case-sensitive.

NB: DO NOT forget your logon password(s). If you feel you might forget, write them down and store them in a safe place. Also make sure “Caps Lock” is not on when entering a password.

Deleting Users: “Delete” vs. “Purge”

If you want to remove a user from the system, by default, if you select the “Delete” command, the user's profile will not actually be removed from the system; rather, it will merely be *marked* as being deleted. Henceforth that user will no longer be able to log on. The primary reason for retaining the user profile is to be able to still display the information in the article history for where that user “last modified” or “created” articles. It also allows you to potentially undelete the user again in future.

If you select the “Purge” option, the user profile is well and truly deleted from the system. If you use this option, then for any entries created by or last modified by that user, the system will no longer know what user information to display. This can not be undone.

To summarise, the “Delete” command can thus be seen as a “soft” delete, while the “Purge” command is a “hard” delete.

It is recommended that you stick to using the “Delete” option.

Privileges System

In the user management dialog, you can select for each user what privileges they have. For example, you can untick “Export” to disallow certain users from being able to export the database to other formats, or untick “Edit DTD” to prevent certain members of the team from being able to modify the DTD (as this is something that should only be done by someone who knows what they are doing).

Monitoring and Tracking Progress

There are at least a few different ways to keep track of the progress of individual users. One is to use the “Created by” or “Last modified by” filters under “Filter (F5)” to reveal entries created or modified by a particular user or users. This could be combined with the “Search (F3)” tool to further narrow it down to particular date ranges. For example, if you filter on “Last modified by: David”, and then search for “2007-04”, it will show you all entries modified by the user David in April of 2007. This could also be taken down to the level of individual days, e.g. “2007-04-25”. See the FAQ (<http://tshwanedje.com/faq.html>) for more possibilities.

You can use the “Sort by” function under “Format (F4)” to sort the entries by the 'last modified date' (e.g. “Lemma::Modified”). This will re-index and re-sort the Lemma List according to the last modified date, showing you all the newest work at the bottom of the list. This can also optionally be combined with a user-specific filter, allowing you to see only the newest work of a particular user sorted this way.

User Progress Statistics

Another tool for tracking the progress of users is the **User Statistics** tool. This can be accessed via the “Database/User statistics” menu option. This tool displays charts over time, based on the “last-modified” date of entries, of the number of entries modified on particular days – both the totals, as well as the number of entries for specific users. The date range can also be changed. A chart from the User Statistics tool is displayed in the screenshot below.

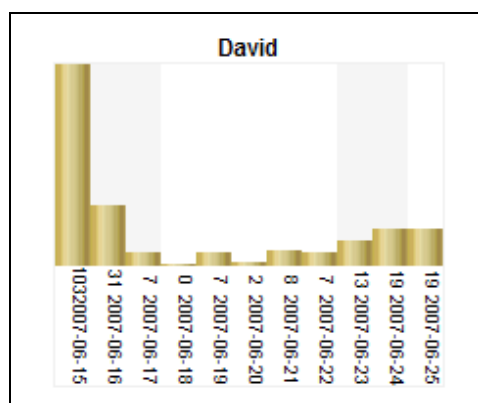


Figure 41: A 14-day “User Statistics” chart for a single user

Bars with a grey background denote weekends.