

e-Sword and Databases

Part 2

White Paper by Stephen Jovanovic
Copyright © 2013, all rights reserved

The Introductory bit that has to be Said

This is part two of e-Sword and Databases, and in this part I want to discuss how e-Sword reads the data, the information placed in the database.

If you remember, in the outset of Part 1, I said:

“Primarily and basically e-Sword is a **Database Reader**, in that it **only** reads data base files.”

This is very important to remember because a Database has only one primary function, and that is to gather data, and that data can be any form shape or size. It is an **all or nothing tool**, and it will only display what is put into it. Leave anything out, it wont display it. So in the *way* that e-Sword reads the data, **all** that information as to be placed in the database before e-Sword can read it.

Where e-Sword Puts the Data

When a database file is opened in e-Sword, the Fields and Records are placed in a certain order, and in a certain place.

In part one we looked at a Bible Database file, and in that file we have seen that it was made up of Tables, Fields, and Records. We learned that the Bible Table had the following Fields:

- Book
- Chapter
- Verse
- Scripture

In e-Sword, the Bible Tree displays both the Book and Chapter Fields. The Verse and Scripture Fields are Displayed in the main Bible Window. But the data in the Book Field has been changed to read a name, and the verses are displayed as part of a complete reference. But when the “Look up Scripture Reference Dialog is opened, there you see (from Left to right) the Book Field displaying the name of the books of the Bible, followed by the Chapters of the selected book, and then the verses of the Chapter selected.

How e-Sword Reads and Outputs the Data

1. Getting it all sorted

We also learned that in regards to the Book, Chapter, and Verse Fields, that have been given an

Integer (whole number) value. With regard to the Book Field, when e-Sword reads it, those numbers are “converted” to a String Output, in where they are given a name. So when e-Sword reads, i.e., the number 1, it outputs it to a String called, Genesis. A String is a length of text.

In the Bible View (Window) the following database information is displayed:

- The selected book of the Bible from the Bible Tree which has been displayed from the Book Field in the Bible Table of the Database.
- The selected chapter from the Bible Tree which has been displayed from the Chapter Field in the Bible Table of the Database.
- All the verses relating to a particular chapter from the Verse Field in the Bible Table of the Database are now displayed in the main Bible Window as part of the whole output of the Bible references.
- All the text in the Scripture Field related to the above Fields

So, when a Book and a Chapter of the Bible is selected, all the information pertaining **just to** that part of the database is displayed in the Bible window. Select another chapter in the Bible Tree, all previous information from the previous chapter is cleared, and the new information is displayed.

2. Making it Look Readable and Pretty

One thing you notice in e-Sword is that the data is displayed in a certain way that makes it readable and presentable. This is called, **Formatting**. But the problem with a Database is that it can't and it doesn't cause the data to be presented in such a way that makes it look readable and pretty. All a database can do is give information and nothing more. A Database program such as SQLite, MS Access, Oracle, Paradox, and etc., do not have text formatting capabilities, and for what they are used for neither should they have them. But we want to have a pleasant appearance to our eyes that makes it look good to read. And also for one to be able to see things like emphasis and expressions used in the text. Therefore, in order for that to happen extra data/information has to be put in the database.

But that's not all. In real life there are people who speak different languages, read in different languages, have different kinds of text which are not Roman/English kind of text. Some countries read Right to Left (Horizontally), and in some other countries read Up and Down (Vertically), and in all English and most European countries read Left to Right. And it's here that databases are not good at all, in fact are pretty much useless in trying to detect what kind of text and language the user has to have.

So, all of these things have to be put into the database file so that the user can read the information/data. And so we come to a crucial part of what has to be done to the data for it to be understandable.

A. Language Settings and UTF

SQLite uses UTF for reading language setting and fonts, and for Hebrew, Arabic, Chinese, Japanese, and other non-Roman text languages for the database file created for e-Sword.

What does all this mean? UTF stands for **Unicode Transformation Format**, and it comes in three main flavors.

1. UTF-8 which is, “Unicode Transformation Format – 8 bit,” and is the primary setting for all Roman text languages in e-Sword.
2. UTF-16 which is, “Unicode Transformation Format – 16 bit,” and is a large extension of UTF-8, and which there is a much larger character set than in UTF-8. There are two subcategories of UTF-16, which are:
 - a. UTF-16le (Unicode Transformation Format Little Endian). UTF-16le gives the capability of text to be read Right to Left and (perhaps) Up and Down.
 - b. UTF-16be (Unicode Transformation Format Big Endian)
3. UTF-32 which is Unicode Transformation Format – 32 bit, and is also known as UTF 4 or Unicode version 4. This is basically the mother of all Unicode, and its huge. It also has the Little and Big Endian capacity that UTF 16 has.

Basically, all UTF does is it read the Character Set of the language that has been set for the database file, and the direction of which the text is to be displayed; if the text is Left to Right or Right to Left.

For example, if the language has been set by the **reserved word** HEBREW, and the direction of the text has been set as “1” (for Right to Left {see Details Table for these setting}) then the UTF encoding will pick up this information, and it will read the glyphs, and display the text in e-Sword.

B. RTF Code and Settings

Rich Text Formatting (a Microsoft invention) also as to be entered into the database in order for the text to be displayed with a particular formatting. It is interesting to note that e-Sword editors are database file generators, and it inserts the RTF code for the format into the database file, and it reads that format. No RTF (or for that matter, UTF) code is stored inside e-Sword. All RTF code is placed in the Record of a particular Table of that database file.

Fonts, Font sizes, margins, page and paragraph details are inserted into the Record of a particular Table or the Database file. As where database files are in Unicode, they too are read by e-Sword.

OK, this actually ends this bit on e-Sword and Databases and of how they work. Although much of the information is quite technical, and maybe somewhat difficult to understand, I trust this has been an interesting learning curve for you.

If you have SQLite Manager or SQLite Browser, open an e-Sword resource file in it, and explore it. Look at the different Tables, Fields, and Records. Look at the settings and note the code used in a Record Entry, and you’ll see just how all this works in e-Sword. But before you play around with a file in SQLite, be sure to make a copy of it, so if you stuff it up, at least you will have a copy to replace it.