H2D
revista
humanidades
digitais

V3 N1 2021

Teaching Digital Editions of the Bible and Ancient Sources. A Reflection

Ensino de Edição Digital da Bíblia e Fontes Antigas. Uma Reflexão

https://doi.org/10.21814/h2d.3509

James Moore, Humboldt Universität zu Berlin, Germany

Como citar

Moore, J. (2021). Ensino de Edição Digital da Bíblia e Fontes Antigas. Uma Reflexão. *H2D|Revista De Humanidades Digitais*, *3*(1). https://doi.org/10.21814/h2d.3509

ISSN: 2184-562X

Teaching Digital Editions of the Bible and Ancient Sources

: A Reflection

Ensino de Edição Digital da Bíblia e Fontes Antigas. Uma Reflexão

https://doi.org/10.21814/h2d.3509

James Moore, Humboldt Universität zu Berlin, Germany

Abstract

This pedagogical article presents an example of how one can teach novice students to build and to work with their own digital editions of ancient sources. It discusses modern approaches to digital editions and promotes teaching SQL databases in ancient studies programs. It provides a detailed guide on how one can structure a student database that will be beneficial to students of all levels and disciplines.

Keywords

SQL Database; PostgreSQL; Bible; Ancient Studies; Digital Editions

Resumo

Este artigo pedagógico apresenta um exemplo de como os alunos podem aprender a construir e a trabalhar com as suas próprias edições digitais de fontes antigas. O artigo explora abordagens modernas às edições digitais e promove o ensino de bases de dados SQL em cursos de estudos antigos. Fornece, ainda, um guião detalhado para auxiliar os alunos a estruturar uma base de dados, que será vantajosa para todos os níveis de ensino e áreas disciplinares.

Palavras-chave

Base de dados SQL; PostgreSQL; Bíblia; Estudos Antigos; Edições Digitais

1. Introduction

In the second semester of 2020/1 at the Humboldt University in Berlin, Germany I taught a course titled Introduction to Digital Editions of the Bible and Texts from the Ancient World. The two primary objectives of the course were (1) to introduce students to relational databases, their construction, and their use in personal small group research in the fields of Bible and ancient studies and (2) to survey for the students additional digital tools and resources used in the development and publication of web-based digital editions. The primary learning outcome was the development of a working multilingual Bible manuscript database for each student's future research. Each database included a workspace in which the student may further develop multilingual/multi-manuscript comparisons, concordances, a multilingual lexicon, complete multilingual word-by-word grammatical analyses, and a biblical commentary. Secondary learning outcomes included the knowledge to expand the student's database into projects on prosopography, social network analysis, paleography, and bibliographic research. Additionally, the students were made aware of digital tools necessary for the modern study of biblical and ancient manuscripts, including types of digital photography, basic techniques for editing photographs (e.g. in Photoshop and Gimp), data storage and access, and indexing vs. storing photographs in their databases. The students were also made aware of full stack development, the API (Application Programming Interface) and UI (User Interface) necessary for a "live" digital edition website.

Central to my teaching philosophy is, and always has been, that *every course*, no matter how philosophical or abstract, must make clear the pragmatic job skills it teaches students. This course prepares students for cutting-edge research in the fields of Bible and ancient studies by providing technical skills necessary for post-graduate employment.

I do not have a degree in computer science. I am, however, one from the generation that straddled the cultural turn to the digital world, and my interests in computing led me to acquire self-taught knowledge of computing languages. Life choices led me to pursue Ancient Near Eastern Studies. Throughout my career as an undergraduate and graduate student I can recall many failed attempts of finding my own digital workflow. It was often more time-consuming to do work digitally rather than with pen and paper, especially working in right-to-left (rtl) languages. But times have changed, and today, working in databases is extremely time-efficient. I now cannot see a single disadvantage of doing most of my professional work in a database environment.

A very popular approach to digital editions in ancient studies is to prepare a **Database alongside XML files**. A common approach is to organize the ancient sources' metadata (information *about* the source) in a database along with *paths to external files* which hold the sources' textual information. It is popular to store digitized texts in an XML (Extensible Markup Language) file. XML format has been well received in ancient (and cultural) studies because it offers many advantages. One can “markup” (or sometimes called “tag”) any part of a text with any type of information (e.g. cultural, grammatical, commentative), and a digital community known as TEI (Text Encoding Initiative; https://tei-c.org/) has spent decades adopting and adapting guidelines for how to “markup” languages, both ancient and modern. While XML files marked up with TEI guidelines may be used to present an aesthetically pleasing digital publication, I have found two great disadvantages to using XML in personal or small group research. First, every structural item needs to be defined in an XML file. This means that every paragraph, line, type-face alteration, etc. must be written in code. This is an extremely lengthy process, and while it may be aided by using an editor application combined with a custom written CSS (Cascading Style Sheets) file, these are time-consuming and require technical skills to write. They also may need to change as research develops. Second, it is virtually impossible to extensively markup rtl languages, such as Hebrew or Aramaic, without a customized CSS file, and even then, difficulties abound. In short, XML is good when a project has a specific and well-defined goal or presentation in mind, but I personally find it more trouble than it is worth for multilingual and advanced research that is ever evolving.

While I informed students in my course of XML and about marking up files, I opted to teach students how to perform most necessary procedures for student (and early career) success within a single SQL relational database. On this point, I will begin to describe the course.

2. The Philosophical Foundation

I began the course by posing the questions: who pays for the internet? What should be free and to whom? This helped us to intellectually orient the course, not only within our context of a public university, but also as researchers who may find themselves on digitalization projects negotiating budgets. These questions allowed us to discuss throughout the course pragmatic issues regarding network access, the limits of open-source software, problems with platform compatibility, the source of digital labor, and other relevant conundrums.

I also asked students to reflect throughout the course on their own methodology, workflows, and processes by which they formulate research questions and complete projects as well as to reflect on their limits on time, labor, and capabilities. By the end of the course it was clear that simply thinking about research problems from the perspective of relational databases is a productive exercise for fine-tuning methodologies and objectives on any research project.

3. The Structure of the Course

My objective was to offer a course that did not require the students to pay for software, but designing such a course came with many challenges. I spent months

(1) deciding whether to introduce the students to all the open-source relational database software or just one, (2) deciding whether to only teach SQL coding (Structured Query Language—the computer language used by the most popular databases) or to incorporate a GUI (Graphic User Interface [application]) into the course, (3) looking for free or open-source cross-platform GUIs, and (4) deciding what I want the students to achieve or produce by the end of the course.

1. **SQL database software.** Of the many relational database options available, three that are based on SQL are commonly used: MySQL, MariaDB, and PostgreSQL. MySQL is open-source and freely available, but it is owned by Oracle. With corporate backing comes many advantages, and the software is widely available and well maintained, but due to an earlier experience I had with changing paywalls while I was using FileMaker, I have a bias and suspicion against MySQL's corporate backing. MariaDB splintered from MySQL when it was acquired by Oracle. It has a large following and would make a good option for a student research database. I, however, use PostgreSQL for my own research database, so I decided to teach it. PostgreSQL is free and open-source with a long history of development. It is the preferred database for many scientific research projects, and, therefore, it is well supported on many university campuses.

2. **GUI vs. CLI.** Database software can run from either a command line using SQL or from an application. For a productive workflow, the vast majority of data input is more efficient when using a GUI (Graphic User Interface) application than when using the CLI (Command Line Interface). That said, I believe a user should know how to perform all SQL functions and operations using the CLI (or at least how to look them up). Most importantly a grasp of CLI is necessary to perform advanced and targeted searches of the data and to produce useful view-tables in which relationally connected data can be brought together and queried. There are excellent free online tutorials and videos available to aid the students' learning of SQL through the CLI.

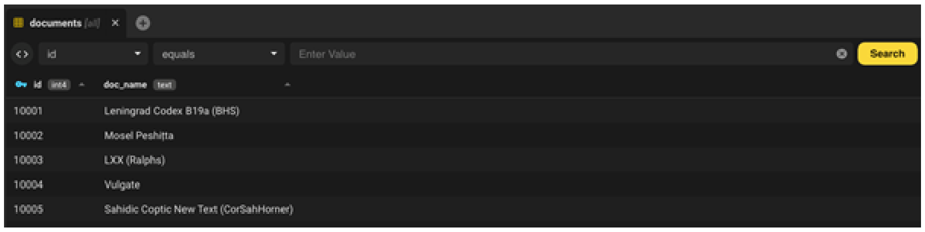
3. **Choosing a GUI.** The advances in recent years in GUIs that make data input and querying easy, is what allows students and scholars to efficiently perform their research in an SQL database environment. In the past, database software was generally only useful after a custom (web) GUI was developed by software engineers. Older general-purpose GUIs, such as PGadmin (https://www.pgadmin.org/), were/are very useful for administrators monitoring the databases and server-side activity, including allocating user permissions to (parts of) databases. Now, however, many companies make available GUIs that are friendly (to variable degrees) for data input and querying. These have opened the world of SQL databases to the intermediate computer user for everyday use. Just like selecting any type of application, however, there are advantages and disadvantages to the different options available. For my teaching purposes, the biggest barriers were cross-platform compatibility (i.e. does the GUI work on Windows

and mac?) and price. I use Postico (https://eggerapps.at/postico/) for my own research database, but this GUI is not free (although it is extremely reasonably priced) and is only available on mac. Some GUIs provide a free trial period but restrict features. I was looking for a GUI that (a) had a friendly graphic user interface that telescoped or teleported the user through the databases' foreign key relationships (i.e. the relationships among tables), (b) had a simple search field interface, (c) provided a graphical interface for importing and exporting CSV (Comma Separated Values) files - the universal spreadsheet file format - and (d) could handle rtl Unicode characters. The application Beekeeper Studio https://www.beekeeperstudio.io/ is an excellent choice, though it does not provide a graphical interface for importing CSV files, and its ability to teleport the user through the foreign key relationships is limited to tables within the same schema (a schema is basically a folder into which tables may be grouped). The first drawback is remedied by the GUI Table Plus (https://tableplus.com/); the free version is greatly restricted, especially on Windows, but offers a useful CSV import feature. The second drawback to Beekeeper Studio, however, may force one to slightly alter the structure of their database, but despite this, it is an excellent teaching aid. Besides, I anticipated that the students would produce databases that would be altered or rebuilt as they move forward in their education and careers. Lastly, for the purposes of backing up their databases or for restoring databases from the class' models, they used PGadmin.

4. **The students' databases.** I anticipated that the students would begin with a variety of levels in computing. In reality they began with virtually no knowledge of databases or coding. As one student expressed, "I had to come to terms with the label 'digital native' that I had been assigned." As with any new course, I had greater ambitions than were realistic, so early in the semester I had decided to focus on helping the students develop a working Bible manuscript database and showed them how to incorporate other ancient sources rather than walk them through the process of incorporating non-biblical sources. The course was offered in the theological faculty, so my choice to focus on Bible manuscripts seemed appropriate.

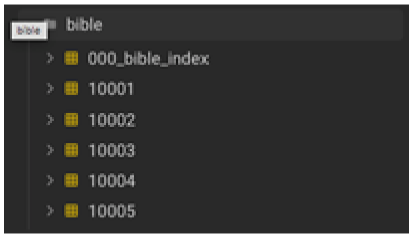
The students' databases included the following components:

- "documents" table - Foundational to the students' databases is a documents table in which biblical manuscripts are listed and assigned an id number (Figure 1).
- "bible" schema - Each Bible manuscript's id number is assigned a table in the bible schema (fig. 2). For manuscripts like the Leningrad Codex, which is freely available in Unicode, the students formatted a CSV file in Excel or Numbers and imported it into the corresponding manuscript table. For manuscripts which are not yet digitized, the students were able to manually input the original text and translation.



id int4	doc_name text
10001	Leningrad Codex B19a (BHS)
10002	Mosel Peshitta
10003	LXX (Ralphs)
10004	Vulgate
10005	Sahidic Coptic New Text (CorSahHorner)

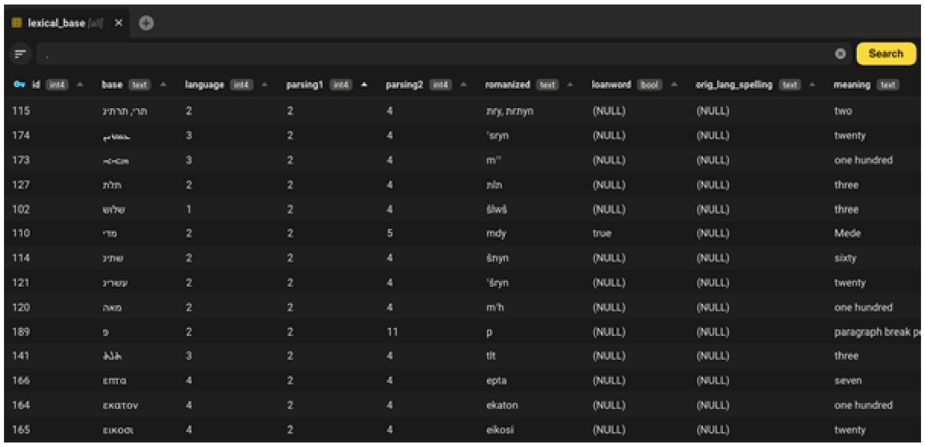
Figure 1: List of Manuscripts



```
bible
  000_bible_index
  10001
  10002
  10003
  10004
  10005
```

Figure 2: Bible Schema

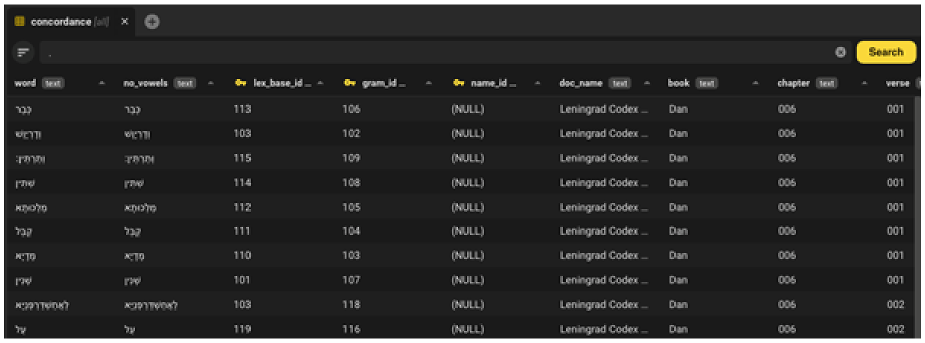
- "lexicon" schema - Three tables are central to this schema, a “lexicon_base” table (Figure 3), a “concordance” table (Figure 4), and a “grammar” table (Figure 5). The lexical_base table serves as the students’ own lexicon, which will grow over time. It has columns for a lexeme, meaning, notes, bibliography, and other linguistic features. It can be easily tailored for a full dictionary-type project, and at least one student expressed interest in developing their database in this direction in the future. The concordance table, the rows of which are automatically generated from a simple line of code, contains every individual word found in all manuscripts. Here the students will link each word to an entry in the lexicon. This process too can be mostly automated by creating a view-table specific to the genre or language of the newly entered text and then updating the concordance table, where the words of the new entries agree with those already assigned lexical id numbers. The words which the students wish to study in grammatical detail can be sent from the concordance table to the grammar table. There they can assign a complete grammatical analysis to each word. There is a place for students to leave general comments on any individual word on the concordance table or grammatical comments on individual words on the grammar table.
- “commentaries” schema - The commentary is a place for students to make notes or compile bibliographies on any book, chapter, or verse in the Bible (Figure 6).



lexical_base [all]

id int4	base text	language int4	parsing1 int4	parsing2 int4	romanized text	loanword bool	orig_lang_spelling text	meaning text
115	תרי, תרתין	2	2	4	ṯry, ṯrtyn	(NULL)	(NULL)	two
174	ܥܣܪܝܢ	3	2	4	ʿsryn	(NULL)	(NULL)	twenty
173	ܡܐܐ	3	2	4	mʾʾ	(NULL)	(NULL)	one hundred
127	תלת	2	2	4	ṯlṯ	(NULL)	(NULL)	three
102	שלוש	1	2	4	šlwš	(NULL)	(NULL)	three
110	מדי	2	2	5	mdy	true	(NULL)	Mede
114	שתין	2	2	4	štyn	(NULL)	(NULL)	sixty
121	עשרין	2	2	4	ʿśryn	(NULL)	(NULL)	twenty
120	מאה	2	2	4	mʾh	(NULL)	(NULL)	one hundred
189	פ	2	2	11	p	(NULL)	(NULL)	paragraph break p
141	ܬܠܬ	3	2	4	tlt	(NULL)	(NULL)	three
166	επτα	4	2	4	epta	(NULL)	(NULL)	seven
164	εκατον	4	2	4	ekaton	(NULL)	(NULL)	one hundred
165	εικοσι	4	2	4	eikosi	(NULL)	(NULL)	twenty

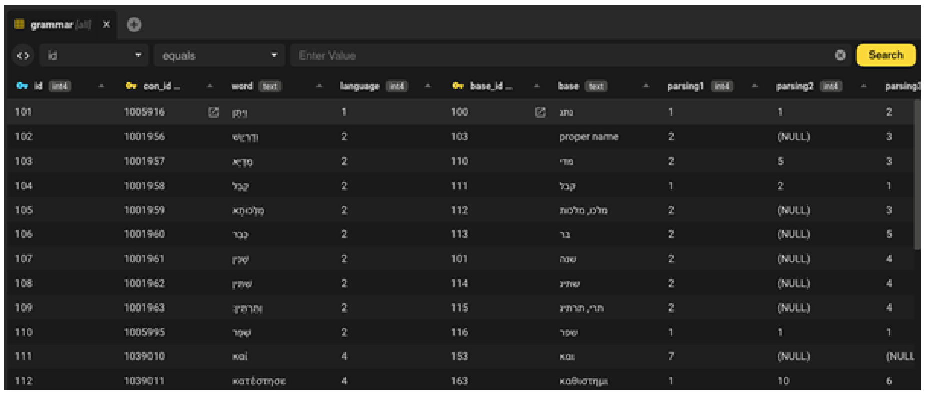
Figure 3: Student Lexicon



concordance [all]

word text	no_vowels text	lex_base_id ...	gram_id ...	name_id ...	doc_name text	book text	chapter text	verse
גְּבַר	גְּבַר	113	106	(NULL)	Leningrad Codex ...	Dan	006	001
וְדָרְיָוֶשׁ	וְדָרְיָוֶשׁ	103	102	(NULL)	Leningrad Codex ...	Dan	006	001
וְתַרְתֵּין׃	וְתַרְתֵּין׃	115	109	(NULL)	Leningrad Codex ...	Dan	006	001
שִׁתִּין	שִׁתִּין	114	108	(NULL)	Leningrad Codex ...	Dan	006	001
מַלְכוּתָא	מַלְכוּתָא	112	105	(NULL)	Leningrad Codex ...	Dan	006	001
קַבֵּל	קַבֵּל	111	104	(NULL)	Leningrad Codex ...	Dan	006	001
מָדָיָא	מָדָיָא	110	103	(NULL)	Leningrad Codex ...	Dan	006	001
שְׁנִין	שְׁנִין	101	107	(NULL)	Leningrad Codex ...	Dan	006	001
לַאֲחַשְׁדַּרְפְּנַיָּא	לַאֲחַשְׁדַּרְפְּנַיָּא	103	118	(NULL)	Leningrad Codex ...	Dan	006	002
עַל	עַל	119	116	(NULL)	Leningrad Codex ...	Dan	006	002

Figure 4: Concordance Table

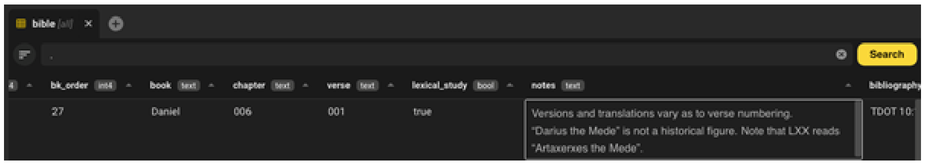


grammar [all]

id ▾ equals ▾ Enter Value

id int4	con_id ...	word text	language int4	base_id ...	base text	parsing1 int4	parsing2 int4	parsing3
101	1005916	יִנְתֵּן	1	100	נתן	1	1	2
102	1001956	וְדָרְיָוֶשׁ	2	103	proper name	2	(NULL)	3
103	1001957	מָדָיָא	2	110	מדי	2	5	3
104	1001958	קַבֵּל	2	111	קבל	1	2	1
105	1001959	מַלְכוּתָא	2	112	מלכו, מלכות	2	(NULL)	3
106	1001960	גְּבַר	2	113	בר	2	(NULL)	5
107	1001961	שְׁנִין	2	101	שנה	2	(NULL)	4
108	1001962	שִׁתִּין	2	114	שתין	2	(NULL)	4
109	1001963	וְתַרְתֵּין׃	2	115	תרי, תרתין	2	(NULL)	4
110	1005995	שְׁפַר	2	116	שפר	1	1	1
111	1039010	καὶ	4	153	καί	7	(NULL)	(NULL
112	1039011	κατέστησε	4	163	καθίστημι	1	10	6

Figure 5: Grammar Parsing Table



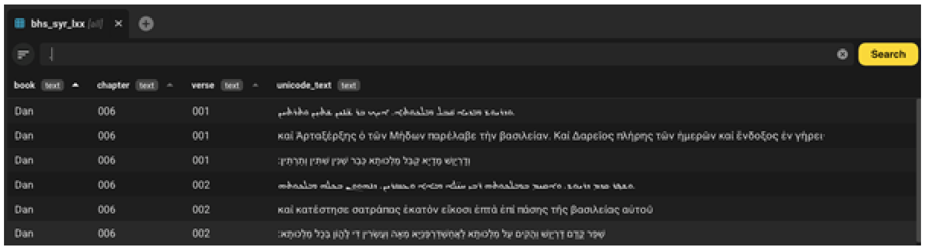
4	bk_order int4	book text	chapter text	verse text	lexical_study bool	notes text	bibliography
	27	Daniel	006	001	true	Versions and translations vary as to verse numbering. "Darius the Mede" is not a historical figure. Note that LXX reads "Artaxerxes the Mede".	TDOT 10:

Figure 6: Student Bible Commentary

3.1. View-tables

A powerful feature of relational databases is that data from any part of the database can be brought together with any other part in a view-table. These are tables that draw selected information from any one or more tables in the database into a single read-only table. The data can then be extensively queried.

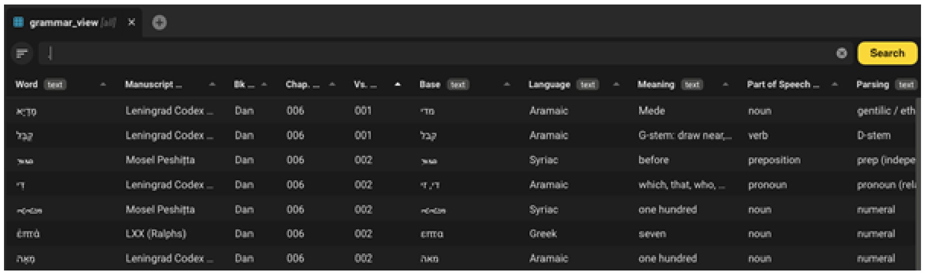
By week seven, the students had a basic and working database on their own machines. In order to test the databases while we were developing them, students were required to read and parse verses along the way as though they were in a language learning class. For our course we focused on Biblical Aramaic and read sections of Daniel and Ezra in Aramaic, Syriac, and Greek. We prepared the texts of the Leningrad Codex, the Peshitta, and the LXX in their respective tables. Then I instructed the students on how to create a view-table that orders data from the individual Leningrad Codex, the Peshitta, and the LXX tables by verse. The result was an interlinear digital Bible view-table (Figure 7). The table is searchable, and students are able to find strings of letters with or without vowels in Syriac or Aramaic or with or without accents and breathing marks in Greek. In a second window they could work on their concordance of the assigned verses, and in a third window they could grammatically label each word.



book text	chapter text	verse text	unicode_text text
Dan	006	001	[illegible]
Dan	006	001	καὶ Ἀρταξέρξης ὁ τῶν Μήδων παρέλαβε τὴν βασιλείαν. Καὶ Δαρεῖος πλήρης τῶν ἡμερῶν καὶ ἔνδοξος ἐν γήρει·
Dan	006	001	וְדָרְיָוֶשׁ מָדָאָה קַבֵּל מַלְכוּתָא כְּבַר שְׁנִין שִׁתִּין וְתַרְתֵּין׃
Dan	006	002	[illegible]
Dan	006	002	καὶ κατέστησε σατράπας ἑκατὸν εἴκοσι ἑπτὰ ἐπὶ πάσης τῆς βασιλείας αὐτοῦ
Dan	006	002	שְׁפַר קֳדָם דָּרְיָוֶשׁ וַהֲקֵים עַל מַלְכוּתָא לַאֲחַשְׁדַּרְפְּנַיָּא מְאָה וְעֶשְׂרִין דִּי לֶהֱוֺן בְּכָל מַלְכוּתָא׃

Figure 7: Manuscripts in Parallel View-Table

As an assignment one week they were asked to create a second and more complicated view-table. This would transform their grammatical table, which relies on 16 daughter tables, into an easy-to-read and searchable parsing chart (Figure 8). They could then simply share their parsings of assigned verses in CSV format for comparison (or for grading if the instructor desires). Obviously, they can also search individual words by any grammatical combinations they wish.

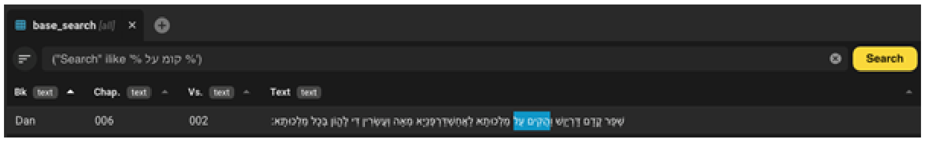


grammar_view [all]

Word	Manuscript ...	Bk ...	Chap. ...	Vs. ...	Base	Language	Meaning	Part of Speech ...	Parsing
מָדָיָא	Leningrad Codex ...	Dan	006	001	מדי	Aramaic	Mede	noun	gentilic / eth
קַבֵּל	Leningrad Codex ...	Dan	006	001	קבל	Aramaic	G-stem: draw near,...	verb	D-stem
ܩܕܡ	Mosel Peshitta	Dan	006	002	ܩܕܡ	Syriac	before	preposition	prep (indepe
דִּי	Leningrad Codex ...	Dan	006	002	די ,ד	Aramaic	which, that, who, ...	pronoun	pronoun (rel
ܡܐܐ	Mosel Peshitta	Dan	006	002	ܡܐܐ	Syriac	one hundred	noun	numeral
ἑπτά	LXX (Ralphs)	Dan	006	002	επτα	Greek	seven	noun	numeral
מְאָה	Leningrad Codex ...	Dan	006	002	מאה	Aramaic	one hundred	noun	numeral

Figure 8: Advanced Grammar View-Table

I further showed the students how they could reverse engineer the manuscripts in a view-table using the concordance, so that they may search by lexical bases rather than exact spellings. This allows them to find adjacent syntactical features, such as some idiomatic constructions. For example, in this view-table they could type in "" and get back every verse in the database, no matter how the verbal root "" is conjugated or spelled.



base_search [all]

("Search" ilike '% קום על %')

Bk	Chap.	Vs.	Text
Dan	006	002	שפר קדם דריוש והקים על מלכותא לאחשדרפניא מאה ועשרין די להון בכל מלכותא:

Figure 9: View-Table. Reverse Engineered from Lexical Bases for Advanced Searches

In only a semester with other tasks to complete, we stopped our build at this point, and worked on fine-tuning it and developing a workflow. Besides the benefit of learning a computer language and a little code, using an SQL database is significantly easier and less time-consuming than marking up XML files. I nonetheless showed the class how PostgreSQL contains an XML data type that will maintain the integrity of such lines of code should the students develop their databases along those lines. Certainly, to achieve more complex syntactical searches, XML tagging would be of value, but apart from highly specified project work, one must weigh the time spent producing XML files against the value of their usability.

While students were fine-tuning their databases and workflow by regularly preparing verses for class, I continued to use 30–50% of the class time to discuss how to incorporate bibliographies (such as from Zotero) and photographs into their databases. We also discussed the types of digital photography used in ancient studies (e.g., Hi-def scans, IR, multispectral, RTI) and the pros and cons of these formats for personal, group, or large project research. We ended the course with a discussion of full stack development, in particular the PERN

(PostgreSQL, Express, React, and Node.js) stack. The goal of these modular discussions was to inform students of the major working components of most modern research projects.

4. Teaching Conditions

I was fortunate to test this course under fairly ideal conditions. The class size was extremely small, three very good students, which allowed for a true workshop environment to develop. The class could function well with around 10 students or with 11+ students, if teaching assistants and/or workshop hours were added to it. That I taught this class online during COVID pandemic restrictions had, in some measure, benefits over the traditional classroom setting. I was able to easily record the meetings so that students could review them later, and students were able to share their screens to present their work or troubleshoot. Student feedback showed that recorded sessions were key to their success, especially during the database build. In this regard, there is no advantage to holding a face-to-face class in a brick-and-mortar setting for most lessons. That said, there would be an advantage to face-to-face troubleshooting workshops 2–4 times throughout the semester.

Factors that must be considered when teaching a digital course are the university's network and security restrictions. One student was attending from a university in our consortium and did not have credentials for the VPN on which my university's database servers are kept. After various attempts to find a workaround that would easily allow students to access a model database built on my university's resources, the solution proved too complicated for novice students in the end. It was simply not possible to allow them to create a network database, which I could easily help troubleshoot, as I had originally planned. Instead, students created their databases on their own machines. The drawback to this is that they were initially bogged down with learning how to install, spin up, and connect to their own databases, which can be overwhelming to the novice.

I found it useful to create a backup copy of the database each week and to create a new database for the next week. After each class session, I would create a database dump, date it, and make it available to students. I would then prepare the steps for the next week's work. This allowed me to have on hand copies of each stage of the build from each week. These versions of the database came in handy more than once throughout the course. When a student would make a significant error, they were able to simply restore their database from the latest class session and continue from there.

5. Results

The course was a great success and produced Bible and ancient studies students who can easily join, adapt to, or perhaps propose a project that is producing digital editions. It also helped the students develop a digital work ethic. If they continue to develop and use their databases throughout their education and into their professional careers, they will have a repository of their cumulative work and knowledge that is easy to query, adapt, and share.

I think in the end a clear argument can be made that a digital editions course should be included in biblical and ancient studies programs as a foundational methodologies course. In my view, it should be offered to undergraduate students and mandatory for graduate students. The only prerequisite is that students need a working knowledge of one ancient language—ideally one in which the teacher is proficient. Students informed me that during the three most difficult weeks, early in the build, they spent between 5–8 hours working on their databases. This fell in the middle of the semester and did not disrupt final examination preparation. I did not require a final project or paper, but theoretically one could be assigned.

While I designed the course for text-based educational programs, a version could be offered that prioritizes archaeology and artifact analysis. *The point to be made is that a course on digital editions provides students with foundational skills in the medium of the modern era.* These skills make students more valuable candidates in their future careers, both in and outside the academy. Academic programs spend one or more semesters teaching students field-specific methodologies, which are normally antiquated and hardly retained. Why not use some of that time to teach students the skills that are necessary for cutting-edge research in biblical and ancient studies and which translate to a variety of positions in the workforce?

I am fortunate to work in a faculty that supports experimental learning and innovative research, but hopefully institutions and hiring committees will discover the value of teaching modern digital skills to students as part of their core curriculum and as an integrated part of research in the modern humanities.

Submitted 2021-07-06 | Published 2021-10-31