Chapter 3
Slicing and dicing the clause cube[36]

3.1 Introduction

This chapter suggests a way in which data warehousing concepts, such as slicing and dicing, may be used to reveal various perspectives on the linguistic data stored in a threedimensional clause cube. After a short recapitulation of the concepts discussed in Chapter 2 regarding the concept and creation of a clause cube, various ways of processing the captured information will be illustrated using a micro-text. Slicing is such an analytical technique which reveals various dimensions of data and their relationships to other dimensions. By using this data warehousing facility the clause cube can be viewed or manipulated to reveal, for example, phrases and clauses, syntactic structures, semantic role frames, or a twodimensional representation of a particular clause's multidimensional analysis in table format.

These techniques will then be applied to the Gen. 1:1-2:3 clause cube, rendered in Addendum A. The source code is available for perusal in Addendum B (see the included CD). An executable program file,"Gen1Version15.exe", is also available on the CD. This program may be run in order to see in detail all the functionalities that are referred to in this chapter.

[36] This chapter is a revised version of a paper read at the SASNES2004 conference, Rand Afrikaans University (now University of Johannesburg), August 2004, ("Slicing and dicing cyber cubes of Biblical Hebrew clauses" (Kroeze, 2004b)), and of an article accepted for publication in Handbook of Research on Text and Web Mining Technologies ("Slicing and dicing a linguistic data cube" by Kroeze, Bothma & Matthee, 2008), edited by Min Song, and to be published by Idea Group Inc, USA.

3.2 Using a data cube to integrate complex sets of linguistic data[37]

The clauses constituting a text can be analysed linguistically in various ways depending on the chosen perspective of a specific researcher. These different analytical perspectives regarding a collection of clauses can be integrated into a paper-based or word-processing medium as a series of twodimensional tables, where each table represents one clause and its multidimensional analysis.

This concept can be explained with a simplified grammatical paradigm and a very small micro-text consisting of only three sentences (e.g. Gen. 1:1a, 4c and 5a)[38]:

- bre$it bara elohim et ha$amayim ve'et ha'arets (in the beginning God created the heaven and the earth)
- vayavdel elohim ben ha'or uven haxo$ex (and God separated the light and the darkness)
- vayikra elohim la'or yom (and God called the light day)[39]

An interlinear multidimensional analysis of this text can be done as a series of tables (see Figure 3.1).

---

[37] This section is a short summary of the main ideas that were discussed in Chapter 2 and Kroeze (2004a). Including the most salient points here enables readers to study the chapter as an independent unit.

[38] These clauses were chosen because all of them have four phrases and because they represent different syntactic structures. Many of the other clauses have less than four phrases which would imply empty cells. Only one clause in Gen. 1:1-2:3 has five phrases.

[39] See Section 3.4 for a discussion of the phonetic transcription used.

	Phrase 1	**Phrase 2**	**Phrase 3**	**Phrase 4**
Phonetic transcription	bre$it	bara	elohim	et ha$amayim ve'et ha'arets
Literal translation	in the beginning	he created	God	the heaven and the earth
Word groups	PP	VP	NP	NP
Syntactic function	Adjunct	Main verb	Subject	Object
Semantic function	Time	Action	Agent	Product

	Phrase 1	**Phrase 2**	**Phrase 3**	**Phrase 4**
Phonetic transcription	vayavdel	elohim	ben ha'or	uven haxo$ex
Literal translation	and he separated	God	between the light	and between the darkness
Word groups	VP	NP	PP	PP
Syntactic function	Main verb	Subject	Complement	Complement
Semantic function	Action	Agent	Patient	Source

	Phrase 1	**Phrase 2**	**Phrase 3**	**Phrase 4**
Phonetic transcription	vayikra	elohim	la'or	Yom
Literal translation	and he called	God	to the light	Day
Word groups	VP	NP	PP	NP
Syntactic function	Main verb	Subject	IndObj	Complement
Semantic function	Action	Agent	Patient	Product

Figure 3.1. A series of twodimensional tables, each containing a multidimensional linguistic analysis of one clause.

The linguistic modules[40] that are represented here were chosen only to illustrate the concept of an integrated structure of linguistic data, as well as the manipulation thereof, and should not be regarded as comprehensive. In more detailed analyses additional layers of analyses, such as morphology, transliteration[41] and pragmatics could be added. A data cube provides a way in which the results of various divergent research projects may be integrated.

Although such series of tables can be regarded as a database, if it is electronically available, these tables are not combined into a single coherent data structure and they do not allow for flexible analytical operations. Knowing the advanced *ad hoc* query possibilities that are facilitated by database management systems on highly structured data, the ability to perform similar operations on implicitly structured linguistic data becomes attractive. Such queries would be facilitated if all the separate tables could be combined into one complex data structure. This is an example of document processing that "needs database processing for storing and manipulating data" (Kroenke, 2004: 464).

The obvious suggestion for solving this problem would be to use a relational database to capture linguistic data, but there are some prohibiting factors. There are many differences among the structures of clauses and the result will be a very sparse database (containing many empty fields) if one were to create attributes for all possible syntactic and semantic fields. Even in the event that this could work, an extra field will be needed to capture the word-order position for every phrase. Furthermore, relational database management systems are restricted to two dimensions: "The table in an RDBMS can only ever represent multidimensional data in two dimensions" (Connolly and Begg, 2005: 1209).

---

[40] Cf. Van der Merwe (2002: 89). The term *module* is preferred here to refer to the different layers of linguistic analysis, because *level* is used in data cube terminology to refer to the members of a hierarchical dimension (cf. Ponniah, 2001: 360-362).

[41] A transliteration is a precise rendering of text written in one alphabet by means of another alphabet. The transcription given in this thesis is a rough phonetic rendering which cannot be used to mechanically reconstruct the Hebrew text.

Closer inspection of the above-mentioned twodimensional clause tables reveals that they actually represent multidimensional data. The various rows of each table do not represent separate records (as is typical of a twodimensional relational database), but deeper modules of analysis, which are related to the data in the first row. A collection of interlinear tables is in fact a twodimensional representation of three- (or multi-)dimensional linguistic data structures. Each table represents one twodimensional "slice" of this threedimensional structure, and the whole collection is a stack of these slices. This insight holds the key to solving the problem of capturing and processing this data.

If the data is essentially multidimensional, the ideal computerised data structure with which to capture it would be a multidimensional database. This type of data structure already exists and is usually employed in businesses' data warehouses to enable multidimensional on-line analytical processing (MOLAP) (cf. Connolly and Begg, 2005: 1209; Ponniah, 2001: 365). Data cubes are used to capture threedimensional data structures and hyper cubes[42] for multidimensional data structures. They are based on threedimensional or multidimensional arrays.

Before the implementation of these concepts in terms of programming is discussed, it should first be made clear how the linguistic data referred to above can indeed be regarded as threedimensional. The knowledge that is represented by a collection of interlinear tables can be conceptualised threedimensionally as a cube similar to the famous "magic" cube toy (Rubik's cube). The cube is subdivided into rows and columns on three dimensions. The sizes of these dimensions, however, do not have to be the same and will be determined by requirements of the unique data set. Each sub-cube is a data-container and can store one piece of information. The information cube therefore consists of a cluster of clauses and their analyses. The horizontal dimension is divided into rows representing the various clauses - each row being a unique record or clause. The vertical dimension is divided into columns and represents the various phrases in the clauses. The depth dimension represents the

[42] Cf. Kroenke (2004: 553).

various modules of analysis, for example, phonetic rendering, literal translation, word groups, syntactic functions and semantic functions.

The linguistic data captured in the twodimensional tables of the micro-text above can thus be stored in a threedimensional data-structure in the following way (see Figure 3.2) – cf. Figure 2.6, which is repeated here for easy reference:

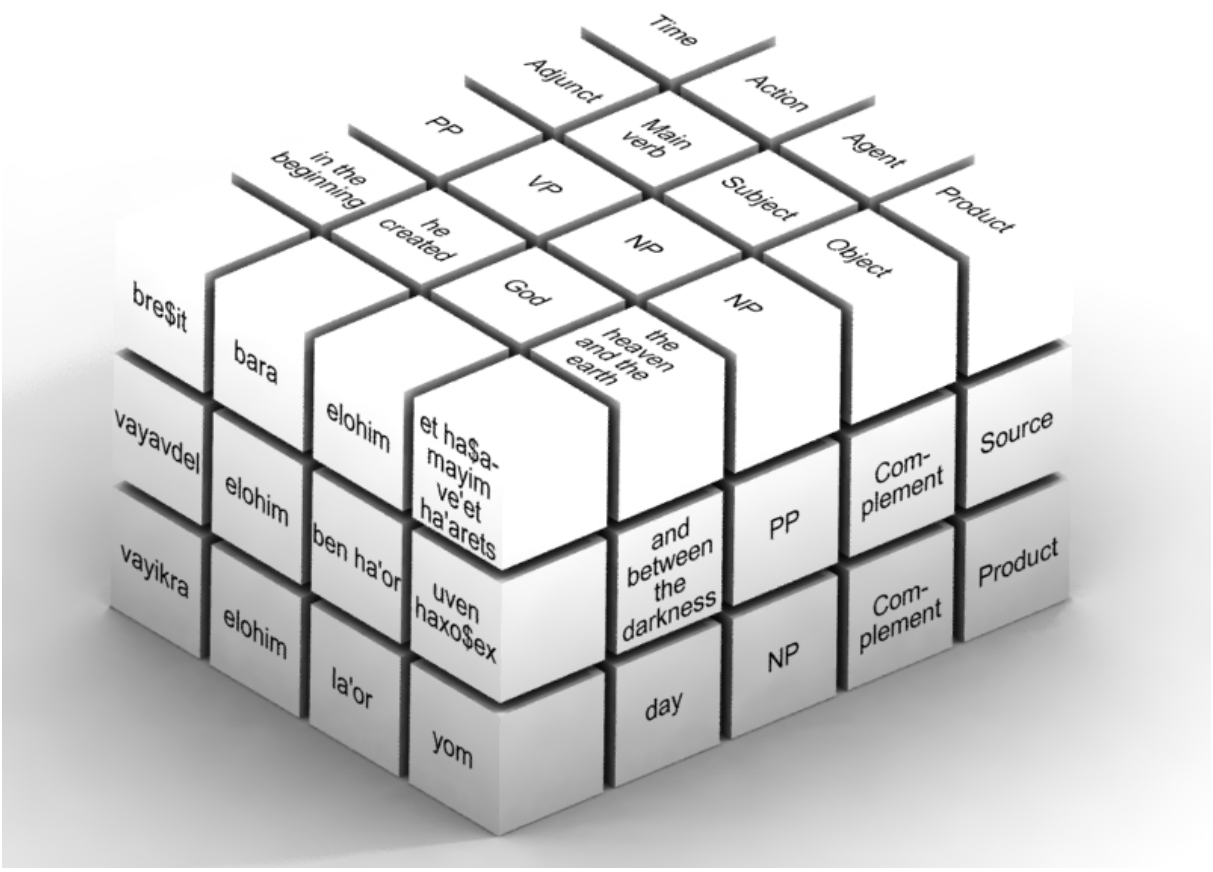

Figure 3.2. A threedimensional clause cube.

Such a clausal data cube can be implemented on a computer using a threedimensional array.[43] A threedimensional array is a stack of twodimensional data variables.

[43] Compare Chapter 2 and/or Kroeze (2004a) for a detailed discussion on the design and implementation of a clause cube.

Some programming languages, such as Visual Basic 6, also allow the use of multidimensional arrays (with four or more dimensions), which could represent a hyper cube of clauses, but due to huge space implications for the computer's memory[44] and the difficulty to visualize four or more dimensions, this thesis deals with three dimensions only. Since it is possible to declare the exact number of rows, columns and depth members of a threedimensional array, enough members can be created on the depth dimension to store all modules of clause analyses.

3.3 Processing the information in a clause cube

Combining repetition control structures such as nested loops with threedimensional arrays makes it possible to process the stored information in an efficient manner. Using three- or multidimensional tables to represent abstract data is not only a tool to store information, but also an important intermediate step in creating computerized visualizations of this information (cf. Card et al., 1999: 17, 23).[45] Koutsoukis et al. (1999: 8) differentiate between manipulation and viewing functions performed on multidimensional data. Slicing, rotating and nesting are viewing functions, while drilling-down and rolling-up are manipulation functions. A slice is a twodimensional layer of the data and implies that the dimension which is being sliced, is dropped. To rotate, dice[46] or pivot the cube means to reveal another perspective or view that consists of a different combination of dimensions. Nesting is "to display values from one dimension within another dimension" (Koutsoukis et al., 1999: 8). Drilling-down is the revelation of more detailed data, linked to a specific cell, on the deeper levels of a hierarchical dimension, while rolling-up (or drilling-up, consolidation, aggregation) refers to summarised data on the higher levels of a hierarchical dimension. In this

---

[44] "As the number of dimensions increases, the number of the cube's cells increases exponentially" (Connolly and Begg, 2005: 1209).

[45] Compare Chapter 7.

[46] Some authors use "slicing-and-dicing" as one concept, while others – like Koutsoukis et al. (1999: 8) here – regard dicing as a synonym for rotation. This chapter uses dicing to indicate the retrieval of subsections of a slice of data.

way the threedimensional array facilitates actions that are typical of data warehousing and on-line analytical processing (OLAP).

In this chapter rotation, slicing and dicing, as well as simple searching functions, will be discussed in more detail. Nesting is probably not applicable to linguistic data, and rolling-up and drilling-down can only be explained by means of hierarchical analyses, such as syntactic tree diagrams. These more complex operations, including searches on more than one parameter and fuzzy searches, as well as the ordering and filtering of the sub-arrays of the clause cube, fall outside the scope of this chapter. Some of these will be explored in Chapter 6.

3.3.1 Rotation

Rotation can be regarded as a computerized version of the human ability to reflect on problem domains from various perspectives. "Different external views can be achieved … by applying rotational transformations to a multidimensional array" (Glasgow & Malton, 1994: 24).

Viewing the clause cube from the front reveals the phonetic representation of the individual sentences of the text. Retrieving these elements can be used to display the phonetic rendering of the text (compare Figures 3.2 and 3.3).

bre$it	bara	elohim	et ha$amayim ve'et ha'arets
Vayavdel	elohim	ben ha'or	uven haxo$ex
Vayikra	elohim	la'or	yom

Figure 3.3. Information revealed on the front side of the clause cube.

If the cube is rotated to show the top side, the first clause's multi-modular analysis is revealed (compare Figures 3.2 and 3.4). The upside down order is due to the structure and rotation of the cube. A more logical order can be obtained by dicing the

separate nuggets of information by means of array processing and presenting it in the required order, or by slicing the cube from the bottom (see below).

Time	Action	Agent	Product
Adjunct	Main verb	Subject	Object
PP	VP	NP	NP
in the beginning	he created	God	the heaven and the earth
bre$it	bara	elohim	et ha$amayim ve'et ha'arets

Figure 3.4. Information revealed on the top side of the clause cube.

Similarly, rotating the cube to display the original bottom side as the front side will reveal the last clause's multi-modular analysis (see Figure 3.5). This time the information is presented in an expected, logical order.

Vayikra	elohim	la'or	yom
and he called	God	to the light	day
VP	NP	PP	NP
Main verb	Subject	IndObj	Complement
Action	Agent	Patient	Product

Figure 3.5. Information revealed on the bottom side of the clause cube.

Looking at the original right side of the cube does not, however, reveal any meaningful perspective (unless the researcher wants to focus, for some reason, on the last constituent of each clause, for example in a study on word order) (compare Figures 3.2 and 3.6).

et ha$amayim ve'et ha'arets	the heaven and the earth	NP	Object	Product
uven haxo$ex	and between the darkness	PP	Complement	Source
yom	day	NP	Complement	Product

Figure 3.6. Information revealed on the right side of the clause cube.

The original left side is similar, but reveals data about the first element of each clause. This information could be used for studies in pragmatics on fronting of clausal elements serving as a topic or focus (see Figures 3.7).

Time	Adjunct	PP	in the beginning	bre$it
Action	Main verb	VP	and he separated	vayavdel
Action	Main verb	VP	and he called	vayikra

Figure 3.7. Information revealed on the left side of the clause cube.

The original back side is again very meaningful, from a semantic perspective, because it reveals the combinations of semantic functions per clause. This information can be used in a study on semantic frameworks, for example, to construct an ontological dictionary such as WORDNET or WORDNET++ (cf. Dehne et al., 2001), and to create a conceptual data model by the COLOR-X method (cf. Dehne et al., 2000). Rotating the cube from its original position in a clockwise manner to see the original back side reveals the semantic role frameworks of the clauses (see Figure 3.8), with, however, the hind part foremost.

Product	Agent	Action	Time
Source	Patient	Agent	Action
Product	Patient	Agent	Action

Figure 3.8. Information revealed by rotating the clause cube 180 degrees in a clockwise manner.

Rotating it head over heels reveals the same information but in a different, upside down, order (see Figure 3.9). The correct order can be revealed by slicing (see below).

Action	Agent	Patient	Product
Action	Agent	Patient	Source
Time	Action	Agent	Product

Figure 3.9. Information revealed by rotating the clause cube 180 degrees head over heels.

3.3.2 Slicing

Rotation is a relatively easy way to demonstrate some of the various perspectives that a researcher can glean from a multidimensional data set. However, the last two examples illustrate the fact that rotation can be confusing because the ordering of constituents differ due to the fact that top can become bottom, left can become right, et cetera, depending on the manner in which the cube is spun. Slicing is better in this regard because a meaningful, easy-to-understand plane can be chosen and all the records can be viewed in the same order.

The clause cube shown in Figure 3.2 could, for example, be sliced from the top to show the three clauses' multi-modular analyses, which brings one back to where this

chapter started, namely the twodimensional representation[47] of multi-modular clausal data (although in a different order of presentation when left in the default data cube ordering) (see Figures 3.10 – 3.12).

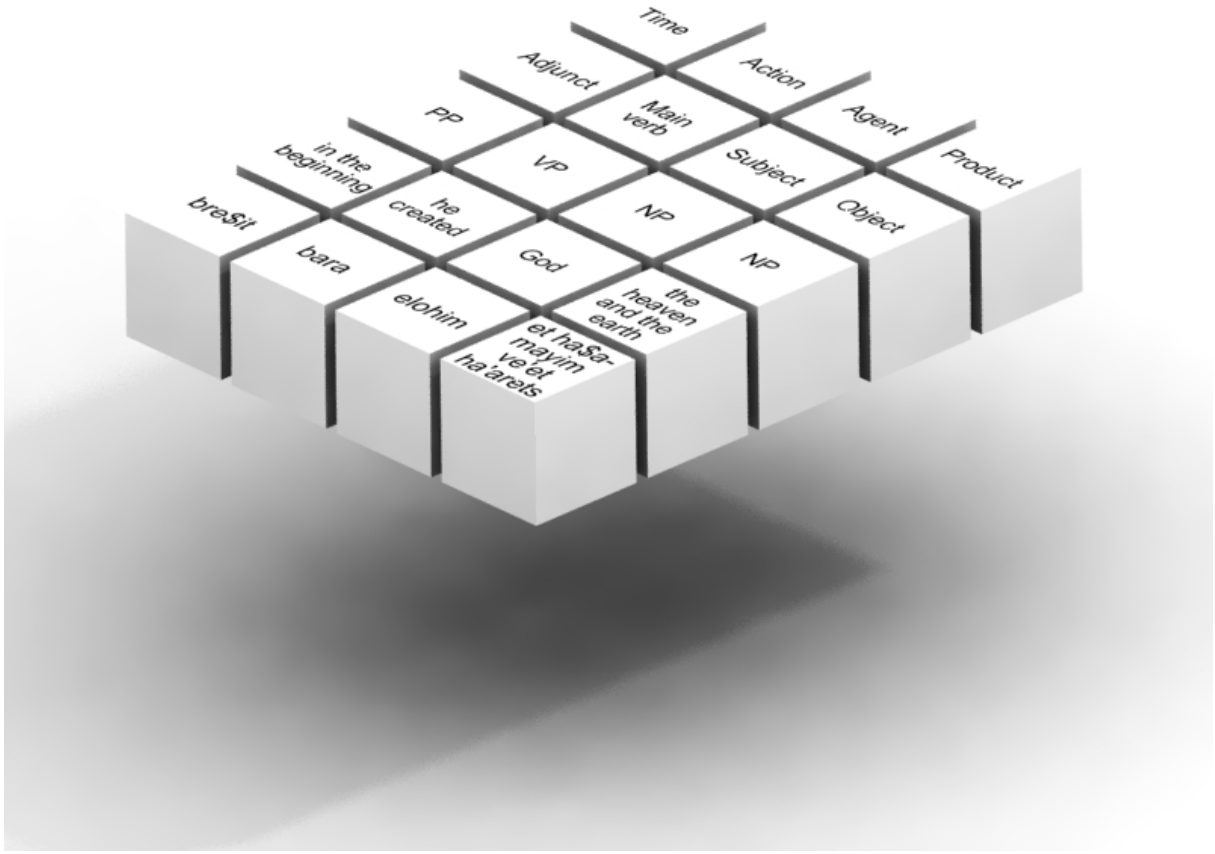

Figure 3.10. The top slice of the data cube, revealing the multi-modular analysis of the first clause.

[47] Cf. Kroenke (2005: 178-179), who discusses twodimensional projections of three dimensions of student data.

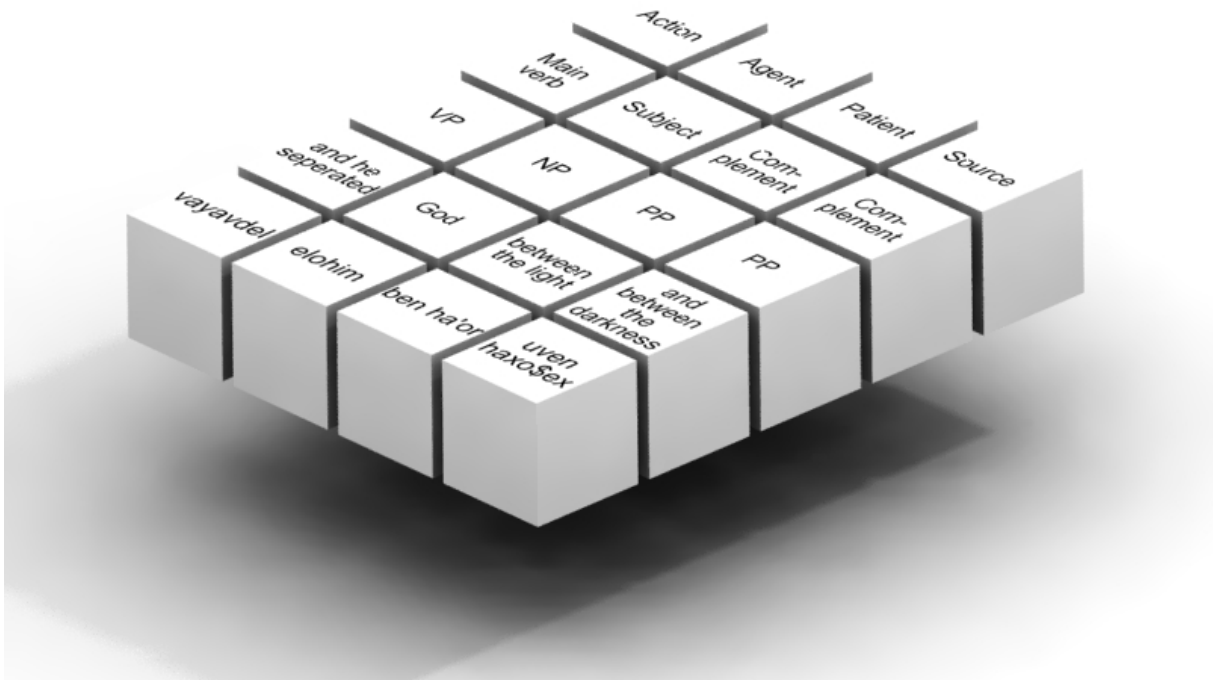

Figure 3.11. The middle slice of the data cube, revealing the multi-modular analysis of the second clause.

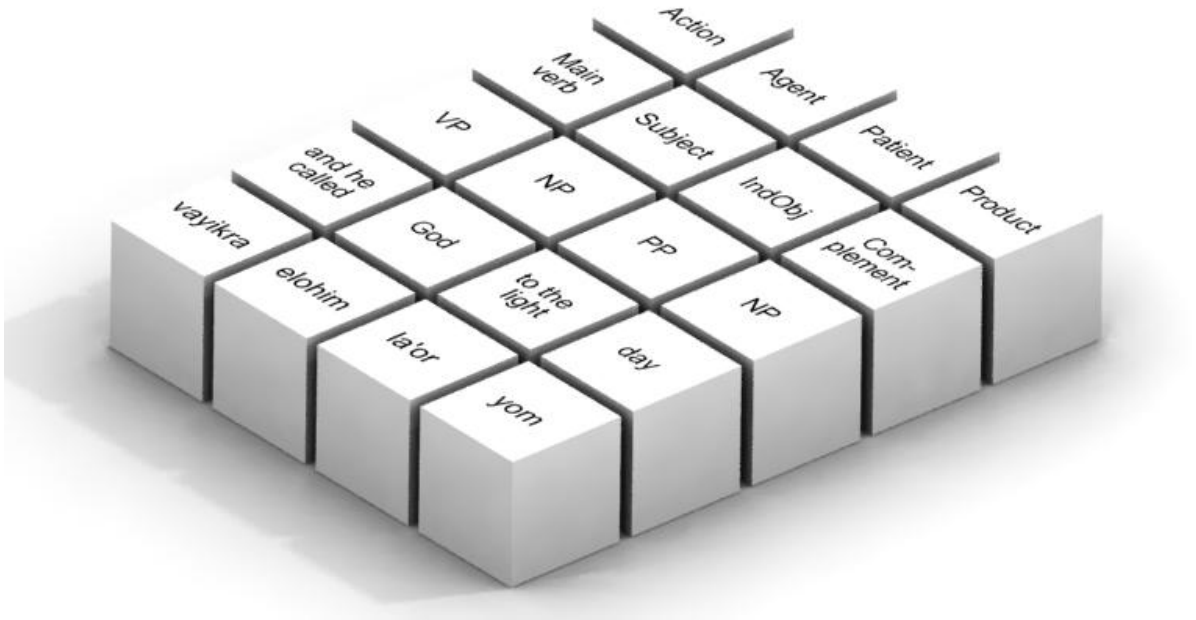

Figure 3.12. The bottom slice of the data cube, revealing the multi-modular analysis of the third clause.

A slice is a "twodimensional plane of the cube" (Ponniah, 2001: 362). The designer of the graphical interface for the output of a slicing or dicing operation actually has the freedom to place data elements wherever they will appear in a most user-friendly way. They do not have to be displayed in a fixed and rigid order that represents their position in the data cube.

It would be very easy to change the order of the rows in these slices to a more user-friendly version of the display to show the phonetic rendering in the top row and the semantic functions in the bottom row. As indicated above, this option is only one of many possibilities offered by the clause cube.

Another advantage of slicing is that it can reveal the elements inside the cube that cannot be seen by rotating it (like the multidimensional analysis of the second clause of Figure 3.11 revealed above). In larger cubes containing hundreds or thousands of clausal analyses, a large number of constituents will be hidden inside the cube. The more members each dimension has, the more data will be out of direct sight.

Slicing can also be used to reveal a specific, required perspective that is hidden inside the cube. Say, for example, a researcher wants to see all the syntactic frameworks of the micro-text. Even in the simple 4x3x5 cube of Figure 3.2 this perspective cannot be acquired by looking at the six outer sides of the cube. One can only see the syntactic frameworks of the first and last clauses, which would not be satisfactory had the clause cube contained many clauses. However, this perspective can be obtained by slicing off the first three planes of the front side and looking at the fourth layer to reveal the syntactic frameworks of all the clauses in the cube (see Figure 3.13).

Adjunct	Main verb	Subject	Object
Main verb	Subject	Complement	Complement
Main verb	Subject	IndObj	Complement

Figure 3.13. Information revealed by slicing off the first three planes from the front side of the clause cube.

Similarly, slicing off four layers from the front will reveal the semantic function frameworks of all clauses in the cube (see Figure 3.14).

Time	Action	Agent	Product
Action	Agent	Patient	Source
Action	Agent	Patient	Product

Figure 3.14. Information revealed by slicing off the first four planes from the front side of the clause cube.

Slicing off the first two layers will reveal all the combinations of word groups, which may be relevant for a morpho-syntactic study (see Figure 3.15).

PP	VP	NP	NP
VP	NP	PP	PP
VP	NP	PP	NP

Figure 3.15. Information revealed by slicing off the first two planes from the front side of the clause cube.

Slicing off only the first layer reveals the literal translation of the text (see Figure 3.16).

in the beginning	He created	God	the heaven and the earth
and he separated	God	between the light	and between the darkness
and he called	God	to the light	day

Figure 3.16. Information revealed by slicing off the first plane from the front side of the clause cube.

It should already be clear by now that a multidimensional data structure provides much more versatility in data viewing and manipulation functions than mere twodimensional tables. Slicing is not only more flexible and satisfactory than rotating, but is also closer to the manner in which a computer processes a threedimensional array. There is, of course, not a real cube that can be rotated inside the computer's memory.[48] But there are millions of memory spaces that can be numbered and filled and called up in any required order. Any slice can be acquired relatively easily by using a repetition control structure (for-loop) containing the specific number that represents the required slice as a constant index in the array reference.[49]

Slicing can also be used as an alternative to rotation: slicing off and viewing the external layer on every side of the cube is the equivalent of rotating the cube. This is exactly how "rotation" is implemented in a threedimensional array in the computer's memory. Valuable slicing options in this problem space are slicing the cube from the front to reveal the Hebrew text (phonetically), literal translation, word group combinations, syntactic frameworks and semantic frameworks; and slicing the cube from the top to reveal multi-modular analyses of subsequent clauses. Slicing from the sides may be valuable in studies on word order and pragmatics.

3.3.3 Dicing

In this thesis the term *dicing* is used to indicate the subdivision of data slices into smaller pieces. Dicing can be used to retrieve very specific required data. One could, for example, retrieve only syntactic functions and their related semantic functions in order to study the mapping of these linguistic modules. In the micro-text above one would probably discover that the semantic function of patient may either be mapped on the syntactic function of complement or indirect object. Dicing may also be used to reorder a set of related data into a logical order on the user interface in order to

---

[48] A cube is a "*conceptual* representation of multidimensional data .... A MOLAP system stores data in an MDBMS, using propriety matrix and array technology to simulate this multidimensional cube" (Rob & Coronel, 2004: 587).

[49] "[T]he dimension(s) that are held constant in a cube are called **slices**" (Kroenke, 2004: 554).

present user-friendly information.[50] In fact, slicing is actually also acquired by means of iterative sets of dicing. Dicing requires knowledge of the structure of the data cube (implemented as a threedimensional array) because there is very strict correspondence between the array index and the clause number, phrase number and language module.

3.3.4 Searching

Simple search functions can be used to look up clauses or phrases. If a specific clause's array index (which acts as a candidate key) is known, one can use it to search for the clause; for example, if one knows that one wants to look at the fiftieth clause in the databank, one could use a loop to display array elements Clause (50,1,1) – Clause (50,5,6). One can also search for examples of specific elements, such as rare syntactic or semantic functions. When a function has to search through the whole multidimensional array to find all possible matches, execution of the program must be paused after each hit to allow the user to study a relevant example before moving on to the next one. Although this may not be an optimised solution, one should remember that the research environment differs from the production environment. The functionality could be made more elegant and efficient for such a purpose.

[50] The elements in the sub-arrays of one dimension can be ordered according to a specific parameter to reveal interesting patterns (Choong et al., 2003). Reordered representations can be used to easily spot syntactic constructions containing peculiar combinations such as the so-called "double accusative" (one construction containing two distinct complements) (cf. Gesenius et al., 1976: 370).

3.4 Application: slicing and dicing the Genesis 1:1-2:3 clause cube[51]

The principles discussed above were applied to the Hebrew text of Genesis 1:1-2:3. The program was created in VB6. The databank was included as a module in the program and consists of a clause cube comprising of the analyses of all clauses containing a main verb in Genesis 1:1-2:3, as done by the author (see Chapter 2 and Addendum A). The linguistic modules that were analysed are:

- Phonetic transcription of phrases[52] (compare Addendum C)
- Literal translation of phrases
- Identification of phrase types (compare Addendum D)
- Syntactic functions of phrases (compare Addendum E)
- Semantic functions of phrases (based on Dik, 1997a, 1997b; compare Addendum F).

These analyses were done by the author, based on his personalised and tacit knowledge of Biblical Hebrew grammar, summarised in Addenda C – F (cf. Kroeze, 2000a and 2000b). Not everybody will necessarily agree with these categories and analyses; however, the analysis itself is not the main focus of this thesis. The primary goal is to illustrate how integrated existing knowledge can be retrieved in various informative ways.

---

[51] This section again builds on the clause cube concepts and implementation discussed in Chapter 2 and Kroeze (2004a). The essential ideas of the construction and structure of the clause cube are repeated here to facilitate the understanding of the analytic operations performed on the data.

[52] It should be possible to use Hebrew characters by means of Unicode because Visual Basic 6 uses Unicode to represent character strings. A phonetic transcription, however, makes this study more accessible for a wider audience. The same ideas could be applied in any language, and knowledge of Hebrew writing should not be a prerequisite for participating in the academic debate on the validity of this concept. An alternative could be to use the Westminster or Michigan-Claremont transliteration (see Groves, 1989: 65).

Embedded clauses have been indicated as a unit in the main clause and separately analysed in a subsequent row.[53] Embedded phrases containing an infinitive or participle have not been analysed in more detail.

The number of columns on the vertical dimension had to be enlarged to five to facilitate the analysis of a clause with five phrases in the rest of the data set (Gen. 1:17a-18a). No clause in the data set had more than five phrases. Provision was also made to capture the unique verse number of each clause, e.g. Gen01v01a, as a user-friendly, primary key.

The viewing and manipulation processes performed on the Genesis 1:1-2:3 clause cube reveal that it is not only possible to view the stored data in a typical interlinear manner, but that any meaningful perspective on the data can be acquired relatively easily. Once the data has been captured in a data structure that represents its natural multidimensionality,[54] basically any query can be answered by using array-processing functions.[55]

Below follow a few examples (screen shots) of the perspectives that are facilitated by slicing the Genesis 1:1-2:3 clause cube (see Figure 3.17-3.20).

---

[53] Instead, a fourth dimension could have been used to capture and represent data of embedded clauses. However, it has been decided to view them as separate clauses, in order to keep the conceptualisation simpler and to minimise sparsity (empty elements in the multidimensional array).

[54] "Multidimensional structures are best visualized as cubes of data, and cubes within cubes of data" (Connolly and Begg, 2005: 1209).

[55] A well-designed data cube "obviates the need for multi-table joins and provides quick and direct access to arrays of data, thus significantly speeding up execution of multidimensional queries" (Connolly and Begg, 2005: 1211).

Figure 3.17 shows a slice of the cube that reveals the multi-modular analysis of Gen. 1:17a-18a (one clause spanning two verses). The user can scroll forward or backward through the stack of twodimensional analyses to study any clause's multi-modular analysis. If the clause number is known it can be used to display that clause directly. The verse number can also be used to access data directly. Vertical scroll bars are activated in some cells to enable the user to see all the text recorded when the window is too small to show all at once.

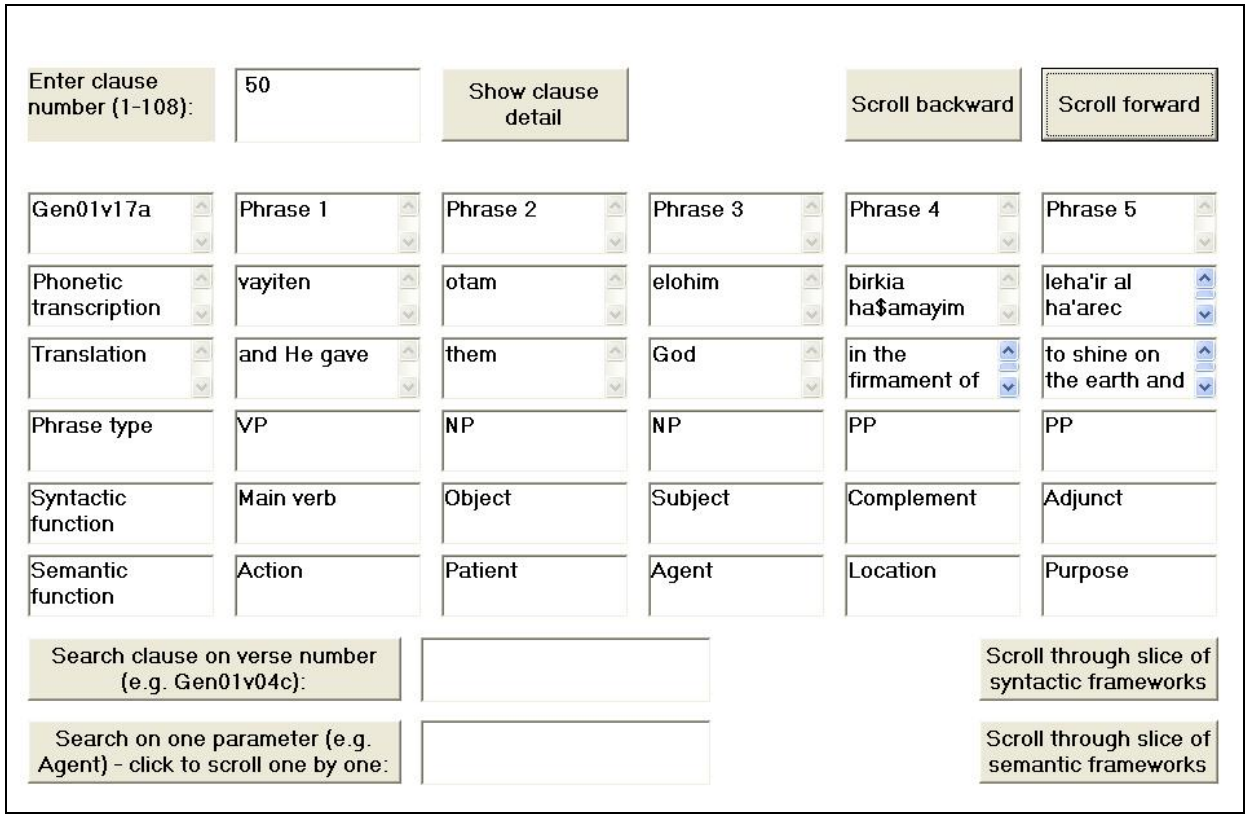

Figure 3.17. A slice of the Genesis 1:1-2:3 clause cube that reveals the multi-modular analysis of Gen. 1:17a-18a (one clause spanning two verses).

Figure 3.18 shows that the clause cube can be searched on a specific parameter. For example, if one wants to find an example of the semantic function of *reason* a search through the threedimensional array will find and display the multi-modular analysis of Gen. 2: 3b, indicating embedded clauses in verses 3c-3d. Reason is an embedded predication, in this case an embedded clause cluster (ECC), indicated by square brackets in the phonetic transcription and literal translation.[56] The embedded predications are then analysed in more detail as separately numbered clauses. The clause number (106 in this example) is displayed in a textbox at the top of the screen; one can also type any clause number (1-108) in the same textbox and click on the "Show clause detail" button to view the required clause's analysis.

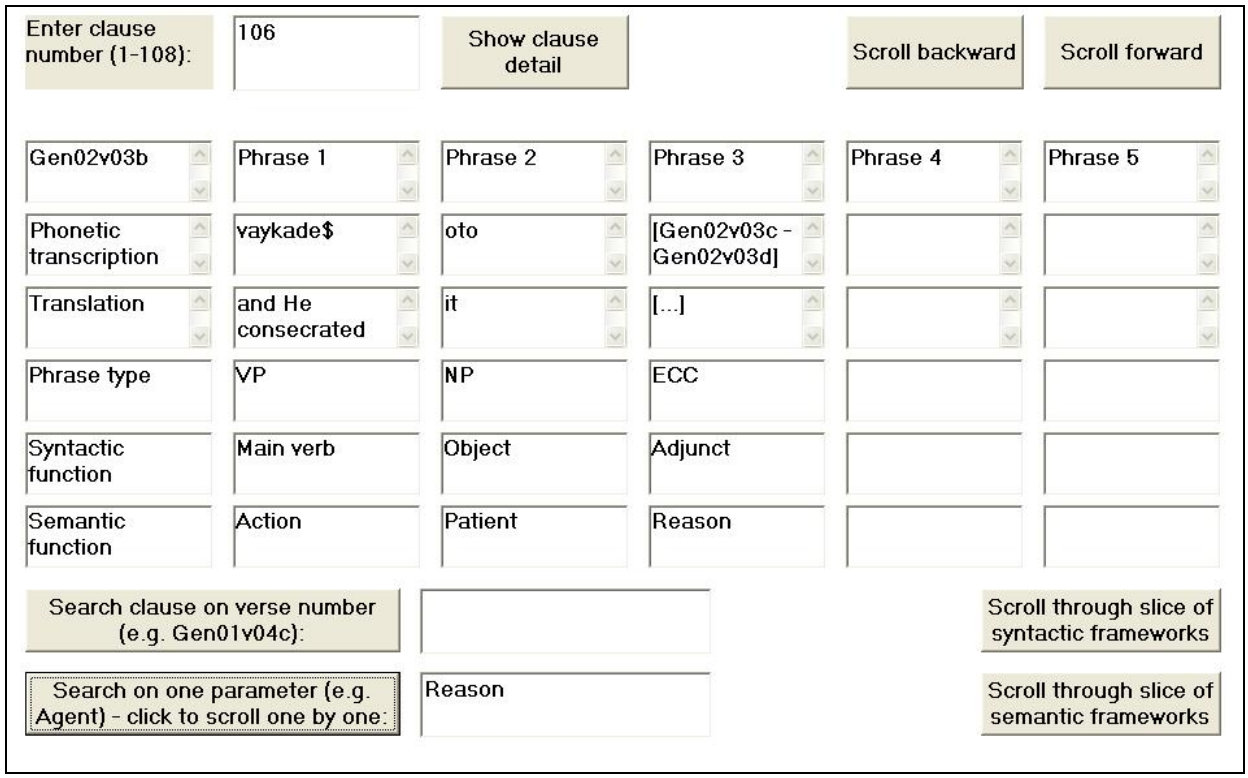

Figure 3.18. The clause cube searched on a specific parameter.

[56] As discussed in Chapter 6, there are still some inconsistencies in the databank, for example the level of detail of embedded clauses and embedded clause clusters rendered in the various language modules, which should be addressed in follow-up work.

In Figure 3.19 the "Scroll through slice of syntactic frameworks" button is used to scroll through the slice that reveals the syntactic structures of all 108 clauses in the cube (six per screen). See Addendum E for a detailed discussion of the syntactic theory used.[57]

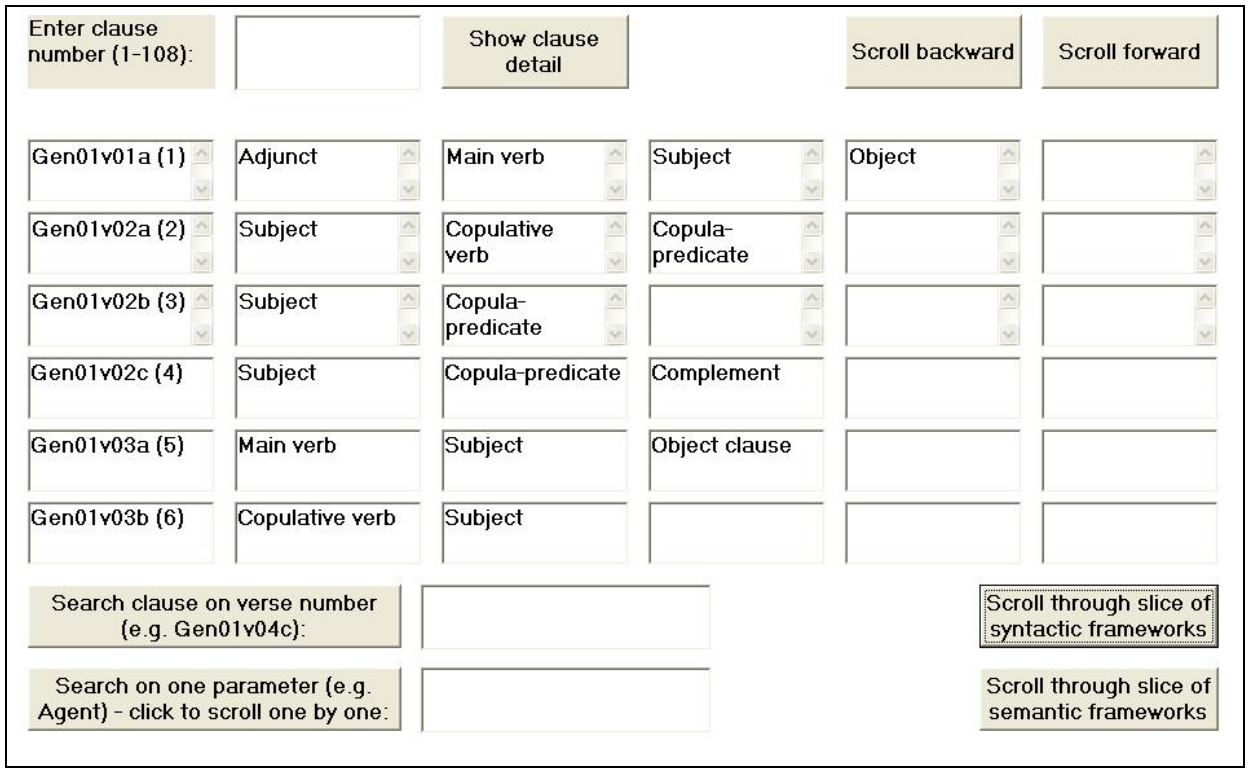

Figure 3.19. Using the "Scroll through slice of syntactic frameworks" button.

[57] Copula-predicate is a synonym for the complement of a copula (*i.a.* a copulative verb) (Du Plessis, 1982: 85-86). The copula is often omitted in Biblical Hebrew. The whole predicate then consists of the copula-predicate, which may be expressed by a noun phrase, adjective phrase, participle phrase, adverb phrase or preposition phrase.

In Figure 3.20 the "Scroll through slice of semantic frameworks" button is used to scroll through the slice that reveals the combinations of semantic functions in all 108 clauses in the cube (six per screen).

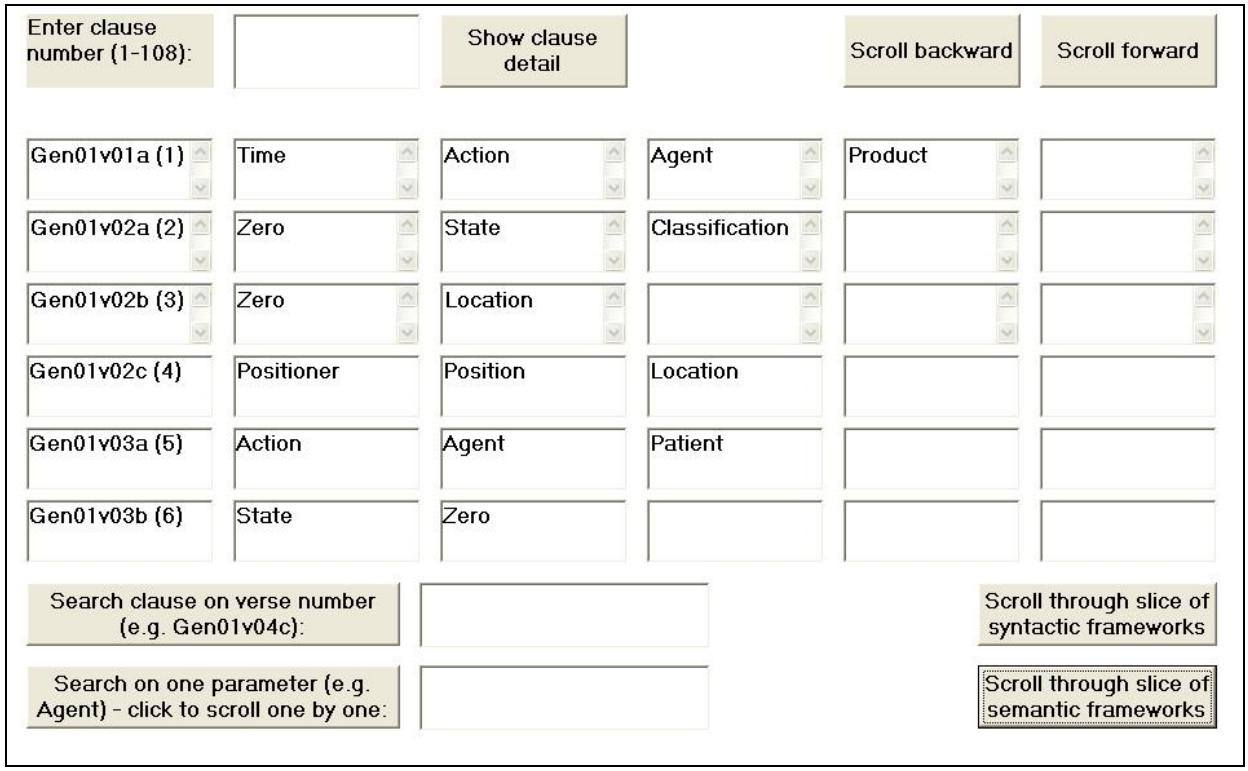

Figure 3.20. Using the "Scroll through slice of semantic frameworks" button.

3.5 A comparison of data-cube concepts and clause-cube concepts

Many of the ideas and concepts used in this project have been borrowed and adapted from the theories regarding data warehouses, data marts and online analytical processing. Although the same basic ideas and technologies are used, it is, however, not exactly the same thing. In Table 3.1 below some salient business data-warehousing concepts (cf. Kudyba & Hoptroff, 2002: 5; Bellatreche, Karlapalem & Mohania, 2002: 25; Rajagopalan & Krovi, 2002: 77; Nazem & Shin, 2002: 108; Davidson, 2002: 114, 117, 123-127, 132-133; Cavero, Marcos, Piattini & Sanchez, 2002: 185-188; Viktor & Du Plooy, 2002: 198; Abramovicz, Kalczynski & Wecel, 2002: 207-210; Jung & Winter, 2002: 221; Gopalkrishnan & Karlapalem, 2002: 243, 255; Ng & Levene, 2002: 285-286; Data warehouse, 2007; Data mart, 2007; Online

analytical processing, 2007) will be compared with their adapted meanings in this thesis.

Concept	Business realm	Linguistics realm
Definition	Data warehousing is the processing of consolidating related business data, thus revealing patterns about specific business events and objects related to time periods, in order to facilitate strategic decisions.	In a data warehouse of linguistic data (e.g. a clause cube) the analyses of various language modules are consolidated in one data structure in order to facilitate the exploration of patterns in and across the interrelated levels.
Subject-oriented	All data related to a specific business event is collected and consolidated.	All (required) data related to the units of a specific text is collected and consolidated.
Time variant	The data is processed to reveal patterns related to periods of time.	The data is processed to reveal hidden patterns in the linguistic data, but these are **not** related to time.
Non-volatile	The data is not updated dynamically but static and read-only.	After the construction of the clause cube the data is usually not updated, but CRUD facilities may be provided to correct analyses or add more data.
Integrated	Data is gleaned from all related business operations	Linguistic units are analysed on various linguistic modules, or existing analyses could be integrated from various sources.
Architecture	A data mart is a subset of a	If various data structures were

	data warehouse. Relational or normalised approach uses tables that are joined using primary and foreign keys. Multidimensional approach uses a separate data structure to capture pre-joined data.	used to capture information from various linguistic modules, these could be regarded as data marts. In this project, however, only one multidimensional data structure is used to pre-join related data.
Storage	A data warehouse stores huge amounts of business data that has been reformatted to enhance analysis and retrieval. Data should be stored in a format that enables flexible, advanced processing and querying.	The clause cube could store huge amounts of linguistic data that has been (re)formatted to enhance analysis and retrieval. In this experiment a short text was used, however, analysed on five levels. The linguistic data is grouped per phrase, but this could be broken down to smaller units to enhance flexibility.
Advantages	OLAP facilitates the discovery of patterns in business data to prompt strategic decisions and improve customer relationship management.	"LOLAP" (linguistic OLAP) facilitates the discovery of patterns in linguistic data to test or prompt linguistic hypotheses.
Disadvantages	The preparation of data and the building of a data warehouse are time-consuming and expensive. It is difficult to	The preparation of data and the building of a clause cube are time-consuming, especially if the construction of the clause

[58] As an alternative one could consider automatic annotation of texts. However, "[e]xtracting more advanced types of semantic information, for example, types of events (to say nothing about

	integrate data from various sources in different formats and platforms.	cube is to be done manually.[58] Integrating existing data would be challenging since the underlying linguistic theories differ considerably.

Table 3.1. A comparison of data-warehousing concepts in the business and linguistic realms.

Using data-warehousing concepts as metaphors for a multidimensional linguistic databank may have implications that "extend beyond technology design questions" because it may limit researchers' creative thinking by superimposing expectations and roles that are typical of the business environment on a humanities realm; therefore, it may be worthwhile to explore additional perspectives such as *data libraries* to complement strategies to integrate linguistic data (cf. Davidson, 2002: 115, 133). The quality of the information gleaned from integrated databanks and the results of data mining these repositories also depend on the extent to which "the social context of the work of data capturing" is taken into account (Viktor & Du Plooy, 2002: 203). Unfortunately, these issues and research opportunities fall outside the scope of this thesis.

3.6 Conclusion

A multidimensional clause cube can facilitate the linguistic analysis with which any exegetical process should commence, which in turn can benefit a multidimensional approach to biblical exegesis (cf. Van der Merwe, 2002: 94). It also facilitates a format in which the biblical text is processed for readers, that is "succinctly enough to

---

determining semantic arguments, 'case roles' in AI terminology), is not quite within the current information extraction capabilities, though work in this direction is ongoing" (Java et al., 2007:51).

be handled by the short-term memory", thus enhancing the success of the communication process (ibid.).

The Genesis 1:1-2:3 clause cube illustrated that linguistic data stored in a data cube can be viewed and manipulated with multidimensional array processing to answer a vast number of queries about the data and relationships between data on various linguistic levels. This implies that linguistic data have been transformed into information, which can again be used to facilitate knowledge acquisition and sharing.

In the following chapter a more elegant solution for the permanent storage of the databank will be investigated, using XML technology. This databank will have to satisfy the requirement of being round-tripped, i.e. imported into the VB6 program and exported to an external storage medium (XML file). It will also have to facilitate advanced processing and visualisation of the networks of linguistic data.