Chapter 2
Towards a multidimensional linguistic database of Biblical Hebrew[9]

2.1 Introduction

Biblical Hebrew clauses can be and have been studied from many different perspectives. These perspectives or layers mirror the "modules" of the human mental language machine (cf. Van der Merwe, 2002: 89). Over the past forty years much of this knowledge has been captured in various computer software systems and databases.[10] Van der Merwe (2002: 96-97) refers to some of these products. The most basic layer is the digital representation of the Hebrew text, which can be called the transliteration layer. The second layer is the phonological layer, followed by the morphological, morpho-syntactic and syntactic layers.[11] More advanced layers such as the semantic and pragmatic layers have received less attention, but it is very probable that knowledge bases and expert systems that deal with these layers will, more and more, become available. Compare Link (1995) who proposes an algebraic perspective on the semantic analysis of human language and the computerized version of Dik's functional grammar for English, French and Dutch (Dik, 1992). Van der Merwe (2002: 94) suggests the use of the notions *topic* and *focus* to mark-up pragmatic functions in Biblical Hebrew (BH).

---

[9] This chapter is a revised and extended version of a paper read at the AIBI VII conference, Leuven, July 2004 ("Processing Hebrew clauses using threedimensional arrays"), and of an article published in Journal of Northwest Semitic Languages, 2004, vol. 30, no. 2, pp. 99-120.

[10] Cf. Talstra (1989: 4). In 1987 ten machine-readable versions of the Masoretic Text and various Bible concordance programs already existed (Hughes, 1987: 343-384; 498-545).

[11] Sowa (2000: 182) refers to morphological, syntactic and semantic parsing as *stages* in analyzing a natural language sentence, saying: "Each of the three stages in sentence processing depends on a repository of linguistic knowledge". The proposed multidimensional linguistic database of Hebrew clauses can be regarded as such a repository, which integrates various layers of analysis. Also cf. Hughes (1987: 497).

From these suggestions it is already clear that there are two main approaches in creating computerized biblical information systems. According to Talstra (1989: 2), the ideal linguistic database should be created by programs applying imitated rules, "otherwise a database of biblical texts will consist only of an echo of a personal, subjective knowledge and contain linguistic information not being produced by rules but by arbitrary personal choice".[12] Ultimately, however, this is an unattainable goal, because subjectivity will also influence the formulation of the linguistic rules that are to be imitated. Even Talstra & Postma (1989: 20) had to admit that it is impossible to formulate and refine rules that will attain a correct analysis in all cases. Therefore, there should also be a place for systems that capture the tacit knowledge that exists in the heads of experienced exegetes. Database solutions that capture existing linguistic data can fill this gap. Chiaramella (1986: 129) also refers to the "strong discussion about the best way to store knowledge" (either data structures or procedural objects) and says that "successful experiments have been made for both". This thesis follows the second route by proposing a database that integrates multi-modular clausal analyses.

2.2 The need for integration

Having available a number of electronic aids for the study of the Hebrew Bible is wonderful, but also overwhelming and even frustrating, due to the fact that various tools have to be used to study different layers and to get various perspectives. Therefore, systems have been suggested or have been developed to display multi-layer analyses of Hebrew clauses, integrating the various dimensions of clausal analysis in an interlinear table format on one screen.

---

[12] "Experimental results in cognitive psychology suggest that humans apply model-based reasoning for problem solving in a variety of domains. Consequently, a formalism that captures the representations and processes associated with model-based reasoning would facilitate the implementation of computational reasoning systems in such problem solving domains" (Glasgow & Malton, 1994: 31).

The Lexham Hebrew-English Interlinear Bible, for example, shows the Hebrew text, transliteration, lemma, lemma transliteration, lexical value and literal English translation of each word in a grid format (Van der Merwe, 2005). See Figure 2.1.

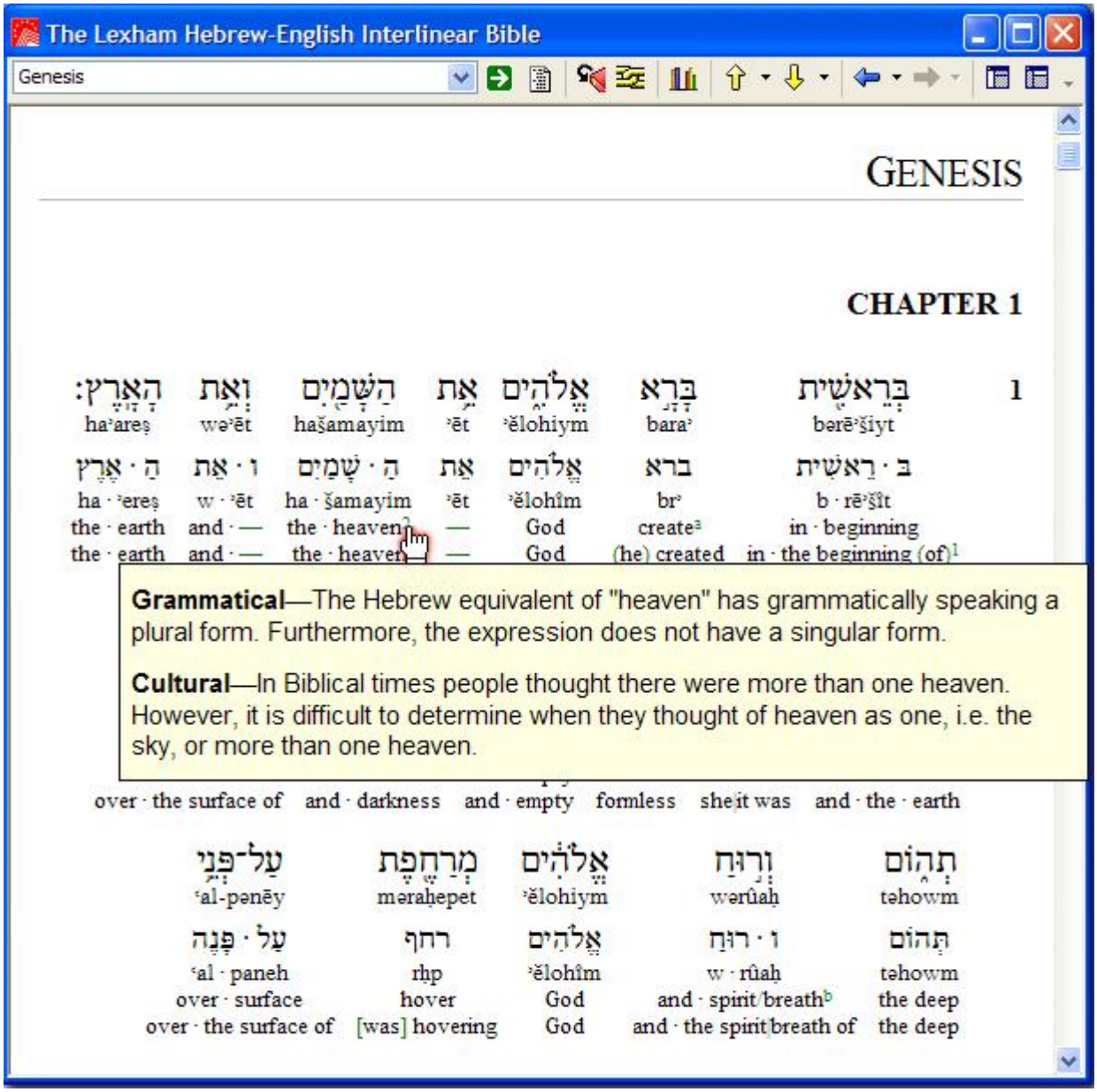

Figure 2.1. An excerpt of the Lexham Hebrew-English Interlinear Bible (http://www.logos.com/products/details/2055).

Also compare Kroeze (2002) for a proposal to include syntactic and semantic functions (in Jonah) in an interlinear web format (see Figure 2.2).

Jonah 1:1a Audio וַיְהִי דְבַר־יְהוָה אֶל־יוֹנָה בֶּן־אֲמִתַּי לֵאמֹר

Heb	... קוּם לְפָנַי	לֵאמֹר	אֲמִתַּי	בֶּן־	יוֹנָה	אֶל־	יְהוָה	דְבַר־	וַיְהִי
Pron	***qum*** ... *lefa**nay***	*lemor*	*amit**tai***	*ven*	*yo**nah***	*el*	*yah**veh***	*dvar*	*vay**hi***
Morph	See below: 2a, 2b, 2c-d	-לְ prep. + אמר qal inf. cs.	proper noun	בֵּן noun m. s. cs. st.	proper noun	preposition	proper noun	דָּבָר noun m. s. cs. st.	*vav* cons. + היה qal impf. 3 m. s.
Lit	See below	*by saying*	*Amittai*	*(the) son of*	*Jonah*	*to*	*Yahweh*	*(the) word of*	*and it was*
WG	-	PP	NP (cs.p.)		PP		NP (cs.p.)		conj. + verb
Syn f	3 object clauses	adjunct	copula-complement				subject		conj. + cop. vb.
Sem f	patient	manner	receiver				zero		- + state vb.
Id tr	*And the word of the Lord came to Jonah, the son of Amittai, thus: ...* *> The Lord spoke to Jonah, the son of Amittai: " .... "*								

Figure 2.2. An interlinear analysis of Jonah 1:1a in an HTML table format (Kroeze, 2002).

Van der Merwe (2002) suggests the use of hypertext as one possible solution to integrate various perspectives or exegetical approaches. However, it could be very difficult or even impossible to integrate all available analyses due to the huge differences in the authors' assumptions and points of departure. Compare Andersen & Forbes (2002) who demonstrate the various divergent approaches, even on elementary layers such as morphology or parts of speech. A possible solution is to show the various analyses in a parallel manner leaving the final decision to the user. Also compare De Troyer's (2002) plea for integrated biblical tools, which implies that such a semi-integrated tool could be very useful to scholars.

Interlinear tables, similar to the illustrations in Figures 2.1 and 2.2, resemble the tables found in relational databases that capture data about entities, and this gives birth to the wish to be able to do *ad hoc* queries on the stored data. A database allows easy access to the data and the possibility of adding new data easily (Tov,

1989: 90). This is not possible with flat files[13] or text files. RDBMSs'[14] structural and data independency feature facilitates these requirements. In order to work dynamically with the stored data it is important to use a proper database management system, which facilitates the use of linked files and complicated search and sorting functions (Niewoudt, 1989: 102).

2.3 A clause cube as the ideal data structure

Unfortunately, interlinear tables cannot simply be transformed into relational database tables, because there is a separate table for each record (or clause) and the rows do not represent unique records.[15] A closer inspection of an interlinear table reveals that the rows actually represent various dimensions or layers of data-analysis that are strongly linked to the elements in the upper row. This type of interlinear table is in fact a twodimensional subset of three- (or multi-)dimensional linguistic data structures. According to Koutsoukis et al. (1999: 7) a stack of twodimensional spreadsheets (rows and columns) is a threedimensional cube.[16] This can be conceptualised as a data structure that consists of a set of sub-cubes arranged according to rows, columns and depth layers (see Figure 2.3).

[13] Although flat files may have rows and columns and thus look like relational database tables, they do not support relational operators such a joins, projects and selects (Hughes, 1987: 497).

[14] Relational database management systems.

[15] Chiaramella (1986: 122) identified the problem of representing text in relational, hierarchical and network database management systems: "Nothing currently exist [sic] for efficient description of texts within database systems".

[16] Compare Pietersma's (2002: 351) discussion of an interlinear Greek-Hebrew text. According to Pietersma an interlinear text is twodimensional because it has a vertical and horizontal dimension.

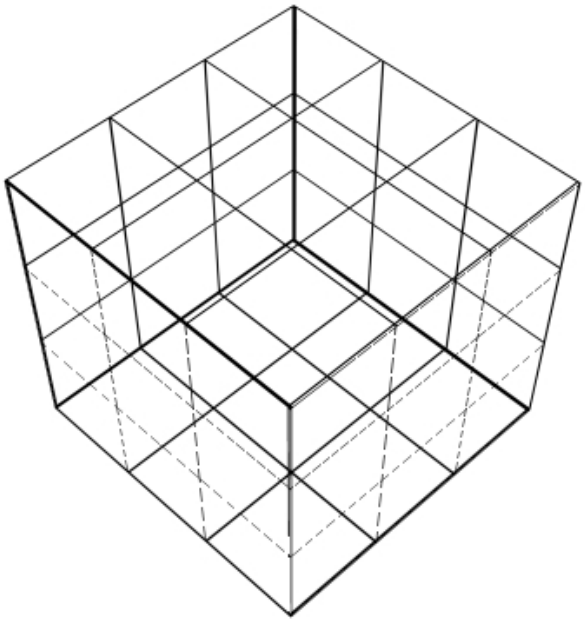

Figure 2.3. A threedimensional data structure that consists of a set of 27 sub-cubes arranged according to three rows, three columns and three depth layers.

The linguistic knowledge that is represented by a collection of interlinear tables can therefore be rendered threedimensionally as a clause cube consisting of a cluster of phrases and their analyses. To explain this concept a very simplified example will first be used below. Three simple English sentences will be used, and the reader should note that these are not real data from the Hebrew Bible. The word *cube* suggests that the lengths of the height, width and depth are equal, which is indeed the case in this simplified example (see Figure 2.4). However, as will be explained later on, this is usually not the case when working with real data. The data-cube concept has been borrowed from data warehousing terminology where the various dimensions may also have different sizes.

The horizontal dimension is divided into rows representing the various clauses - each row is a unique record or clause. The vertical dimension (columns) represents the various word groups or phrases in the clauses. Having attributes in this dimension called phrase 1, phrase 2, phrase 3, etc., at first does not seem very informative, especially if one is used to the descriptive attributes typical of relational databases.[17] However, "it is crucial to preserve the document structure (books, chapters, verses, half-verses, words) of the data, to allow access in terms of traditional categories"

[17] Compare the discussion on the choice of the phrase as primary structuring unit in Chapter 4 (4.4).

(Talstra, 2002: 4). And this method seems to be the most straightforward way to preserve word order.[18] Yet, the combination of these obvious elements on the horizontal and vertical dimensions with the layers on the depth dimension is indeed very illuminating. The depth dimension represents the various modules of analysis, for example, graphemes, syntactic functions and semantic functions. The unique intersections of the members of the various dimensions are the cells, and the contents of the cells the measures (Chau et al., 2002: 216). As in business data "the dimensions provide a 'natural way' to capture the existing real-world information structure" (Koutsoukis et al., 1999: 11).

[18] In the application of this principle to complex examples embedded phrases and clauses will be indicated by square brackets where they occur. These embedded constituents will then be analysed separately.

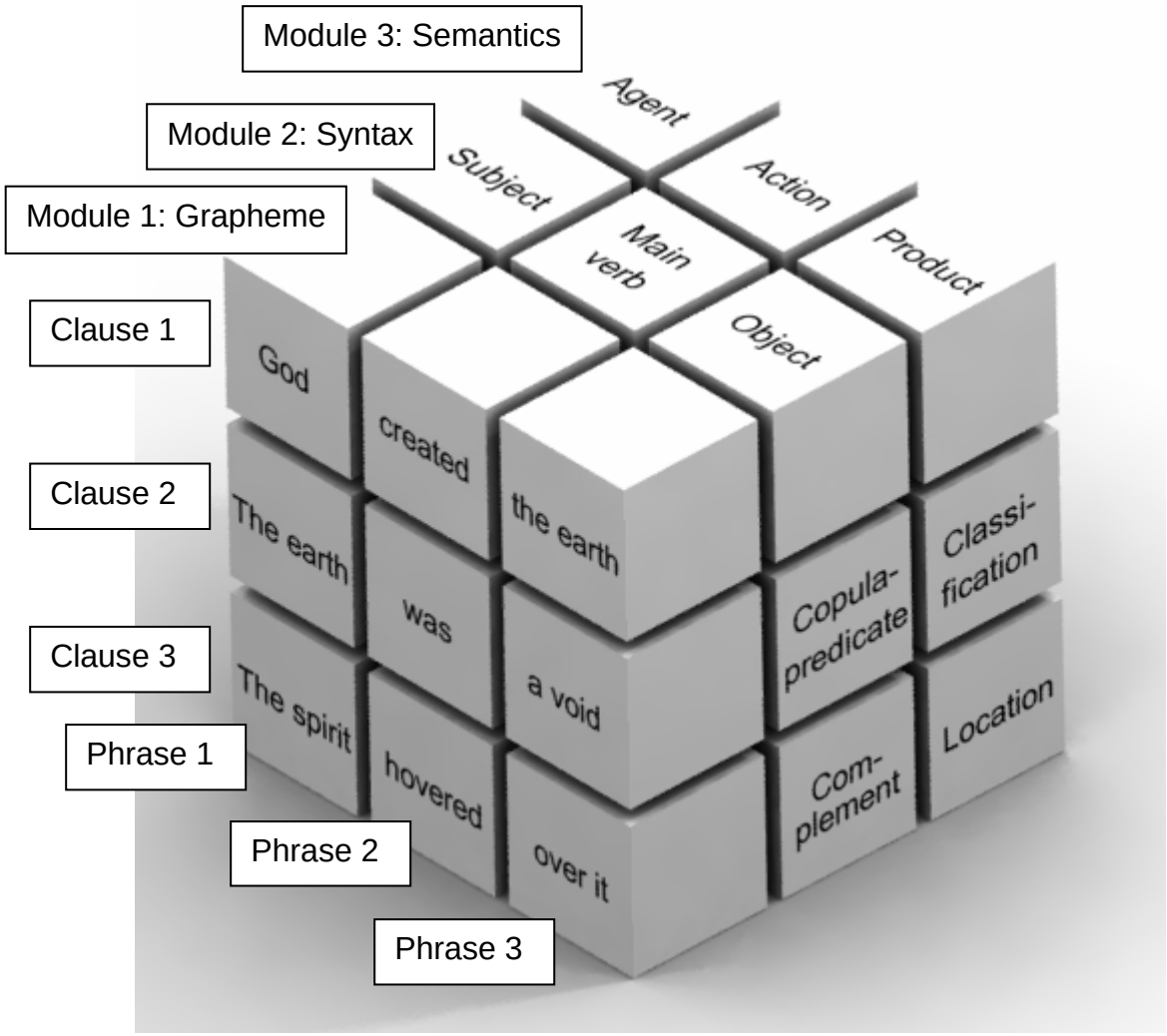

Figure 2.4. The knowledge that is represented by a collection of interlinear tables, rendered threedimensionally as a clause cube consisting of layers of clauses and analyses stacked onto each other.[19]

When one applies this concept a micro-text consisting of three real clauses from the Hebrew Bible (Gen. 1:1a, 4c and 5a)[20] one has to use three rows to represent three

[19] In order to enhance legibility, the clause cube, drawn with Blender 2.3 is shown here in the orthographic/orthonormal view, where distant objects' sizes are not rendered as smaller as in intuitive perspective viewing (Roosendaal & Selleri, 2004: 48).

clauses, four columns to represent the number of phrases in each clause, and five depth levels to represent five layers of analysis. (Although this structure will be sufficient for this very small corpus, it should be enlarged by adding more rows, columns and depth levels to cater for more possibilities in a larger corpus and to add a unique identifier for every clause.) The three clauses are:

- 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)

If each clause is analysed in the suggested way, one needs 20 cells (individual data containers) to capture all of the relevant data (see Figures 2.5a – 2.5c). Each clause in this minute corpus consists of four phrases, each of which is analysed on five levels. These levels are, from bottom to top: phonetics[21], translation, word groups, syntax, semantics.

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

[21] The decision to render the Hebrew text phonetically was made to facilitate accessibility and implies that the word order will be presented from left to right. If the Hebrew text were presented in the Hebrew alphabet the word order should have been shown from right to left.

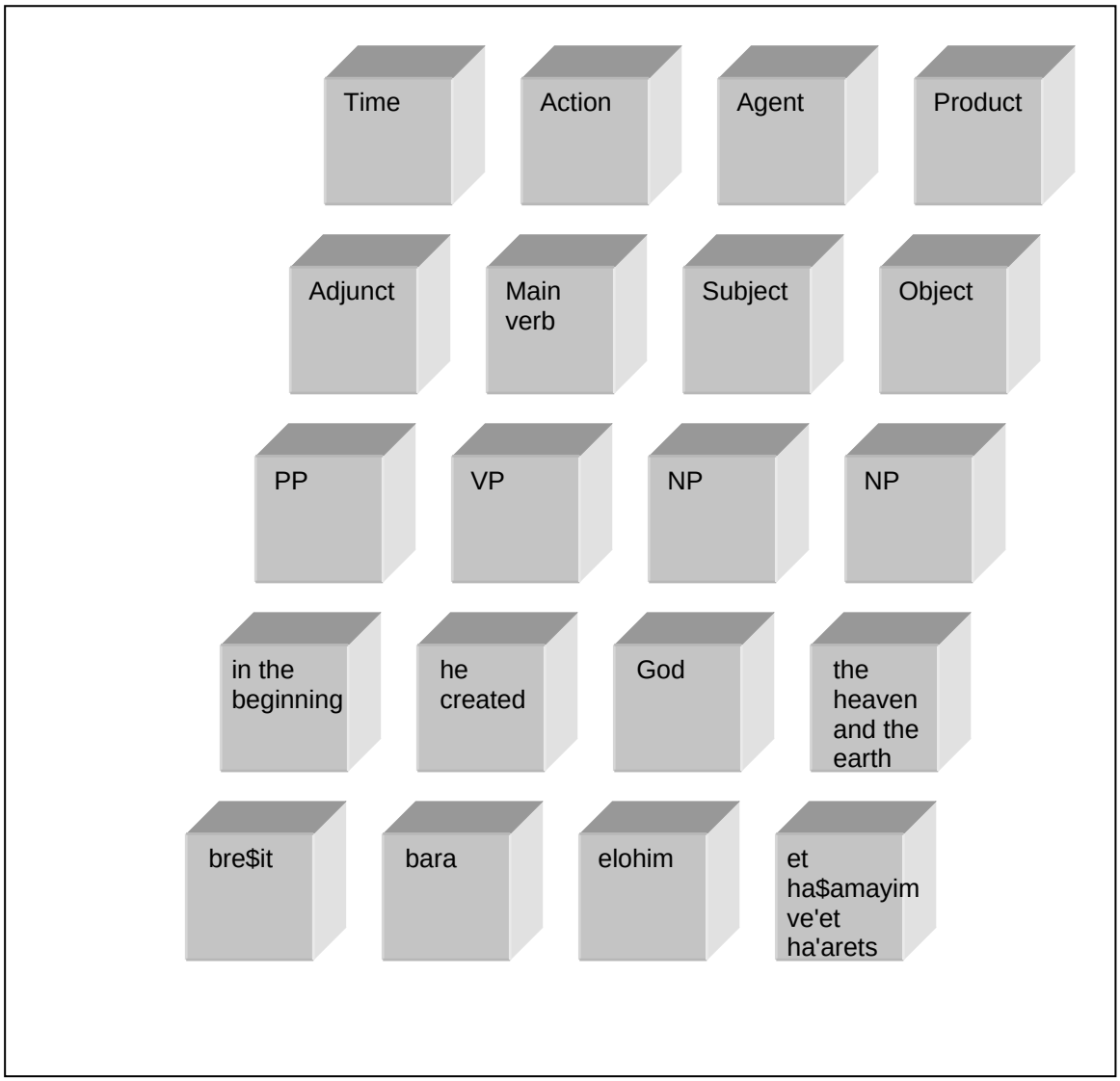

Figure 2.5a. Gen. 1:1a analysed according to phrases and linguistic levels.

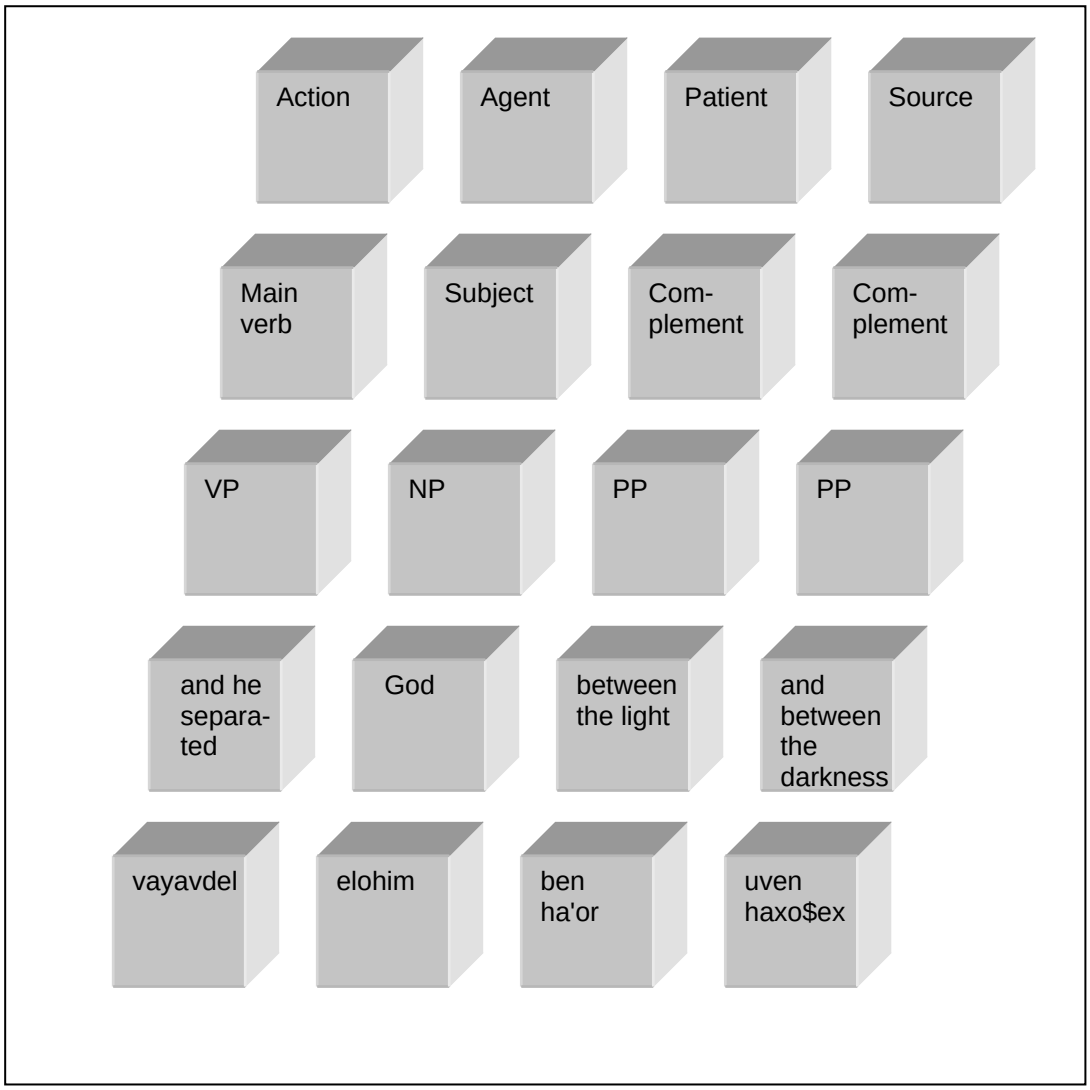

Figure 2.5b. Gen. 1:4c analysed according to phrases and linguistic levels.

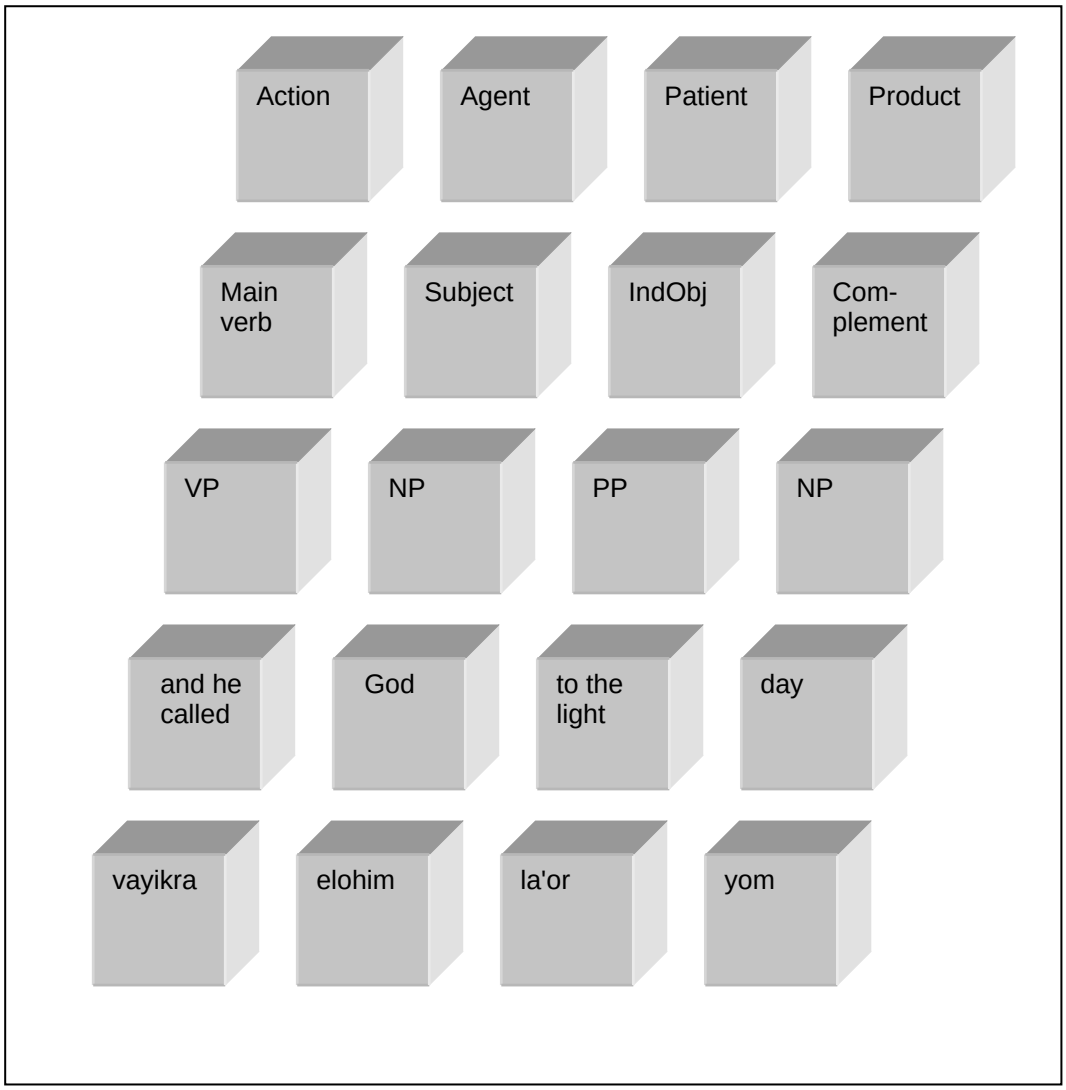

Figure 2.5c. Gen. 1:5a analysed according to phrases and linguistic levels.

If these individual cells are lined up to form a clause cube, the data structure may be visualised as a threedimensional clause cube (the word *cube* is still used although the length, width and depth have different sizes). See Figure 2.6a.

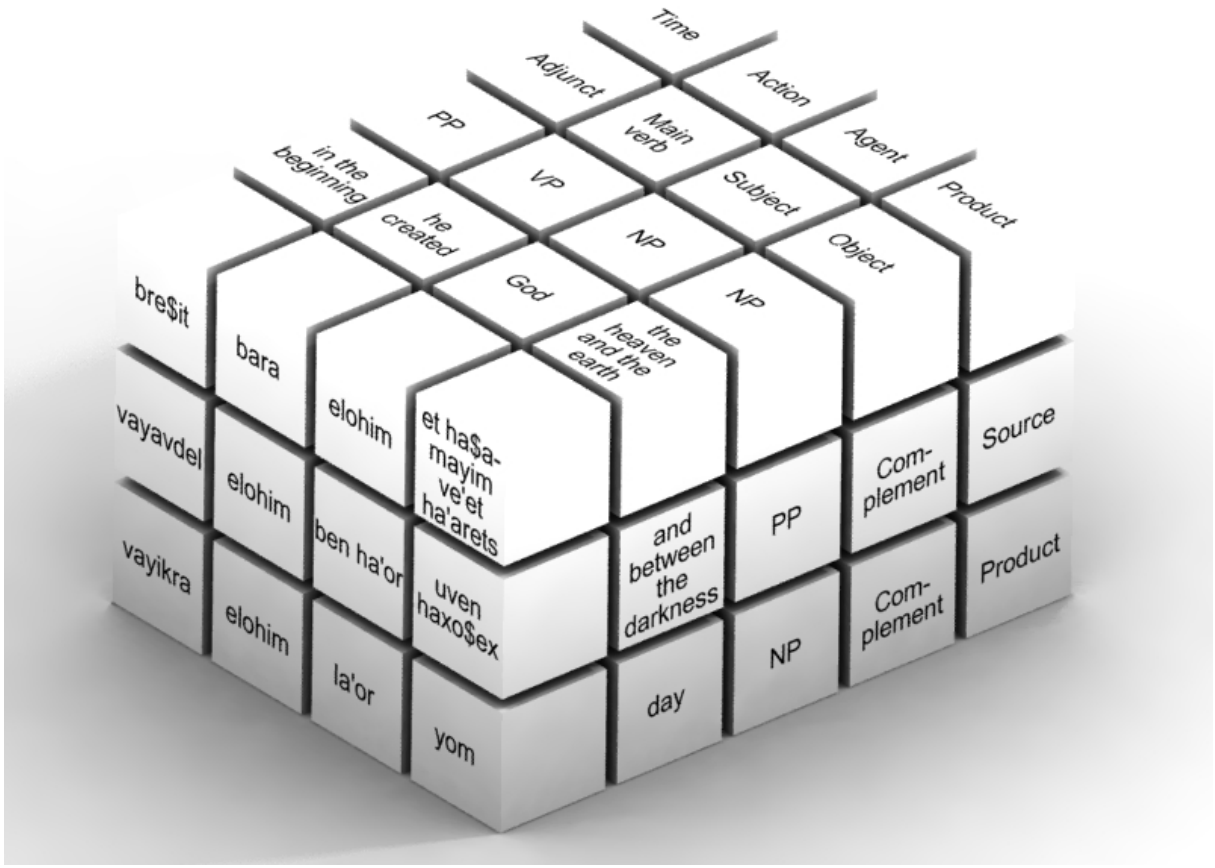

Figure 2.6. A clause cube (orthographic view) containing real linguistic data of three BH clauses, Gen. 1:1a, 4c and 5a.

Of course, looking at the whole cube from the outside only shows the contents of the cells placed on the outside. More data is hidden inside the cube. If one could slice off the top and middle rows, one would be able to see these hidden data (see Figures 2.7a – 2.7b). This concept will be discussed in more detail in Chapter 3.

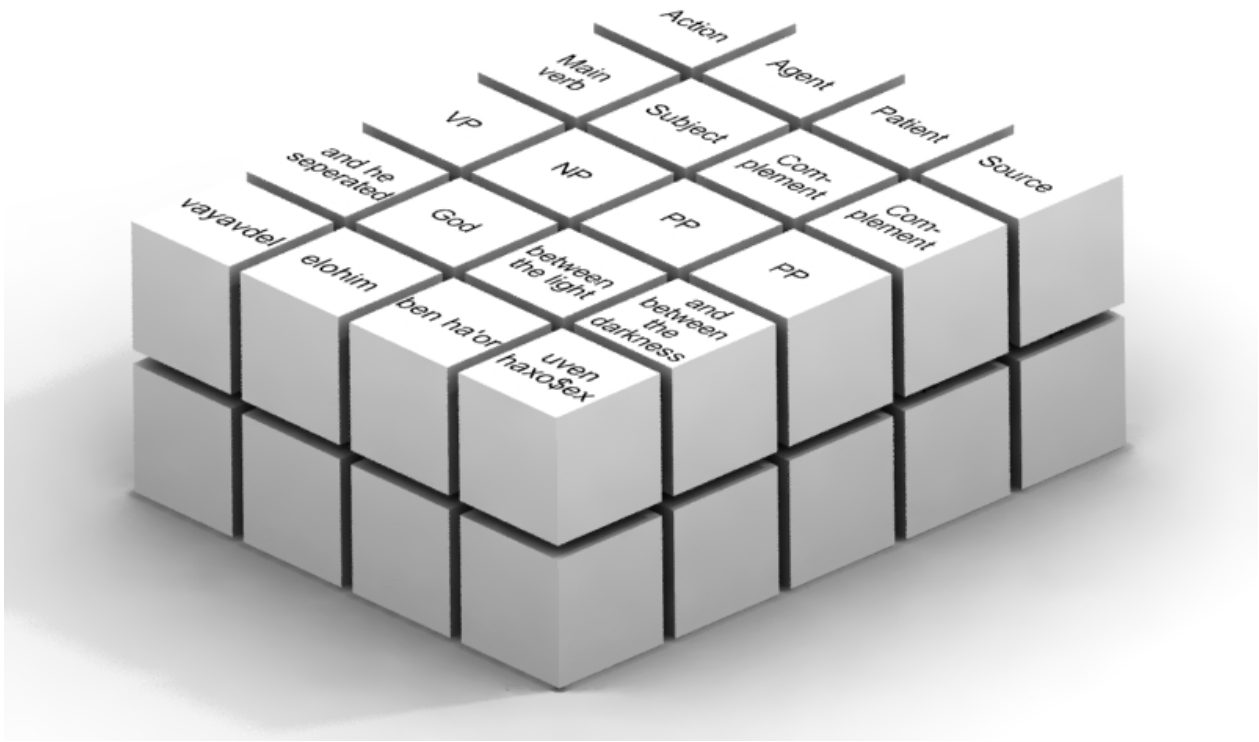

Figure 2.7a. Revealing data contained in the second slice of the cube by removing the top slice.

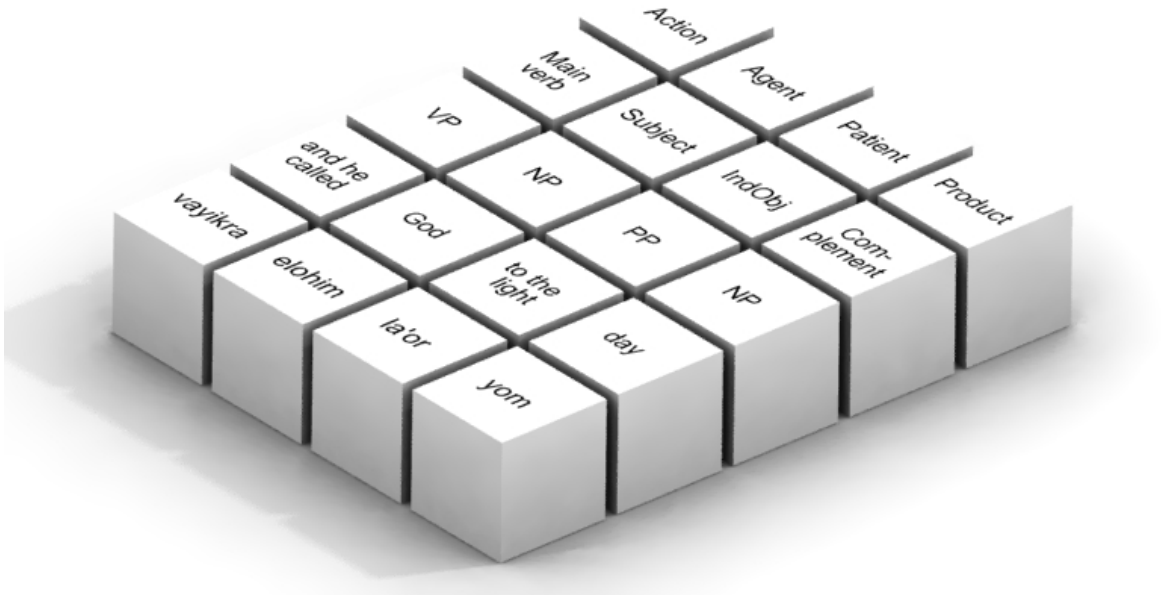

Figure 2.7b. Revealing data contained in the bottom slice of the cube by removing the top and middle slices.

2.4 Implementing the clause cube in cyber space

Such a clause cube can be implemented on a computer using a threedimensional array, which can be called a cyber cube or data cube.[22] An array can be used as a knowledge representation scheme, which models entities and the relations between them in a certain problem domain, and array functions are used to generate, inspect and transform these representations (Glasgow & Malton, 1994: 8). Arrays have probably already been used in many biblical information systems, for example to sort sets of lemmatised language into sets of "identically parsed items" (Hughes, 1987: 502).

A data cube can easily be created in many computer languages by declaring a multidimensional array, for example, in Visual Basic 6[23] a data cube with 3 rows, 3 columns and 3 layers is declared by the following statement: "Public Clause(3,3,3) As String".[24]

The code to create such an exemplary clause cube using a threedimensional array in Visual Basic 6 may be found in Addendum A (on the included CD). It captures linguistic data describing the first 108 clauses in the Hebrew Bible (Genesis 1:1-2:3). Only the first clause is shown here (see Figure 2.8). The first dimension represents the 108 rows of clauses, the second dimension the phrases with a maximum of five per clause,[25] and the third dimension represents the layers of analysis:

- Module 1: Clause number
- Module 2: Transcription
- Module 3: Translation
- Module 4: Phrase type
- Module 5: Syntactic function
- Module 6: Semantic function

One extra layer is added on the depth dimension to those shown in Figures 2.5 – 2.7 to record the verse number, for example, Gen01v01a, as a unique identifier for each clause. This identifier, or primary key, will be recorded in the first column of the first

---

[22] A data cube is a multidimensional, electronic data structure, often used in data warehousing to facilitate multidimensional views of data.

[23] Visual Basic was chosen as programming language for this experiment because it allows the use of threedimensional arrays and easy creation of executable files. More advanced features, such as four or more dimensions in arrays, as well as extensive connectivity to database management systems (Anderson, 2003: 59, 116) could be used in more detailed and complex versions of the clause cube.

[24] Such an array can be visualised as a "cube of side length m subdivided into m^3 unit cubes" (cf. Banchoff, 1996: 15).

[25] Although the first clause only has four phrases, the size of this dimension is declared as five to accommodate clauses with five phrases in the rest of the data set. Empty columns are not populated.

row on the first layer, implying that empty cells will exist on the second to fifth columns of the first row on the first layer. These could have been used to record additional information about the clause, but are left empty in this project. Therefore an empty element, indicated by a hyphen in places Clause(1,2,1), Clause(1,3,1), Clause(1,4,1) and Clause(1,5,1) of the array, implies a hierarchical dependency on the clause number recorded in Clause(1,1,1). Repeating the same verse number reference in all these places would have been redundant. It should be noted that this extra layer represents a theoretical sub-level which is difficult to implement in a natural way using a threedimensional array. The array structure suggests that this element, Clause(1,1,1), is closely linked to the first word group on all the other layers. This, however, is not the case, because the information stored in Clause(1,1,1) actually is metadata that pertains to the whole clause.

These modules, of course, do not represent all possible levels of analysis. More levels could have been added to represent both lower and higher levels of linguistic analysis. On a more basic level, morphological analysis is possible, for example to indicate the various morphemes of each word (bre$it = preposition be + noun re$it) and to list the conjugational and declensional characteristics of verbs and nouns. Links to various available dictionaries could also be included, and on a higher level, pragmatic functions could be added. These, and other, possibilities are ignored in this study, because the primary goal is not to provide a complete linguistic analysis, but to show how various analyses could be integrated in a multidimensional, computerised data structure.[26]

---

[26] Not only grammatical, but also literary analyses of texts are multidimensional. According to McGann (2003: 14-15) there exists an indefinite and dynamic number of perspectives on textuality, “an array of interpretations”. He says: “Since interpretive agency is a continuously evolving variable, and since the object of interpretation is a codependent function of that unfolding interpretive action, this field of textual relations must be understood as n-dimensional.” In this project the grammatical analyses of the text of Genesis 1:1-2:3 is also understood as a multidimensional array of interpretations, some of which are conceptualised, quite literally, as a clause cube.

Below follows a part of the code that creates the threedimensional array and populates it with the selected layers of linguistic data (Figure 2.8).

```
Option Explicit
Public Clause(1 To 108, 1 To 5, 1 To 6) As String
Sub Main()
Clause(1, 1, 1) = "Gen01v01a"
Clause(1, 1, 2) = "bre$it"
Clause(1, 1, 3) = "in the beginning"
Clause(1, 1, 4) = "PP"
Clause(1, 1, 5) = "Adjunct"
Clause(1, 1, 6) = "Time"
Clause(1, 2, 1) = "-"
Clause(1, 2, 2) = "bara"
Clause(1, 2, 3) = "he created"
Clause(1, 2, 4) = "VP"
Clause(1, 2, 5) = "Main verb"
Clause(1, 2, 6) = "Action"
Clause(1, 3, 1) = "-"
Clause(1, 3, 2) = "elohim"
Clause(1, 3, 3) = "God"
Clause(1, 3, 4) = "NP"
Clause(1, 3, 5) = "Subject"
Clause(1, 3, 6) = "Agent"
Clause(1, 4, 1) = "-"
Clause(1, 4, 2) = "et ha$amayim ve'et ha'arets"
Clause(1, 4, 3) = "the heaven and the earth"
Clause(1, 4, 4) = "NP"
Clause(1, 4, 5) = "Object"
Clause(1, 4, 6) = "Product"
…
End Sub
```

Figure 2.8. A part of the code that creates a threedimensional array and populates it with the selected layers of linguistic data.

Data cubes are usually used to implement multidimensional databases or data warehouses[27] to enable users to "explore and analyse a collection of data from many different perspectives, usually considering three factors (dimensions) at a time" (Kay, 2004). According to Kay (2004) "we can think of a 3-D data cube as being a set of similarly structured 2-D tables stacked on top of one another." In our case the data cube consists of the various interlinear clause tables all linked together in one data structure in order to enhance the analytical possibilities. Such a data warehouse is a database solution that can capture and integrate linguistic data from various sources.

One of the benefits of multidimensional arrays is the use of indexes referring to the specific position of a piece of data. These indexes can be used to extract subsets of the data very efficiently and quickly (cf. Kay, 2004). Therefore, multidimensional arrays form the basis for multidimensional online analytical processing tools (OLAP). The possibility to do *ad hoc* queries is one of the essential characteristics of OLAP (Karayannidis & Sellis, 2003: 157). In business data cubes are used for multidimensional queries, for example, how many of a certain product was sold in a specific period in a specific place? (See, for example, Marchand et al., 2004: 3.)

Some programming languages, such as Visual Basic 6, even allow for the use of multidimensional arrays, which could represent a hypercube of clauses.[28] Such a 4-D cube consists of a series of 3-D data cubes (Kay, 2004). In our application a fourth to sixth dimension could be used to break down clause constituents hierarchically into their smallest parts,[29] for example, the NP *et-ha$amayim ve'et ha'arets* (Gen. 1:1) consisting of 2 NPs and a conjunction, with the 2 NPs each consisting of an object

---

[27] A data warehouse is a multidimensional analytic database that "links otherwise disparate data items" and "allows for customised user views of the data" (Koutsoukis et al., 1999: 3).

[28] In geometry a hypercube is a basic fourdimensional structure having 16 corners and "consisting" of (bounded by) 8 cubes. A cube, which is a basic threedimensional structure, has 8 corners and "consists" of 6 squares. A square, which is a basic twodimensional object, has 4 corners and "consists" of 4 lines or segments. A line is the segment between two points, a basic onedimensional object with no corners. A point is a zerodimensional object (Banchoff, 1996: 9).

[29] Glasgow & Malton (1994: 31) found that "an array representation scheme provides an effective and efficient means for spatial reasoning" and suggests that more research should be done to test its applicability to other domains including hierarchical worlds.

marker and NP, which again consists of an article and a noun. The higher level attributes, which represent summarized values, are called aggregates, while the lower level attributes are called grouping attributes (Lee et al., 2003: 124). This level of detail, however, falls outside the scope of this thesis.

Although it is very easy to add another dimension (for example, "Public Clause(3,3,3,3) As String") or even more dimensions in cyber space it becomes more difficult to visualize this type of data structures. Multidimensional arrays have another downside. With every dimension added the number of memory slots needed increases exponentially, for example, a 3x3 table needs 9 spaces, a 3x3x3 data cube needs 27, and a 3x3x3x3 hypercube needs 81.[30] The more dimensions the hypercube has, the sparser it becomes: more and more cells are empty and this wastes memory and processing time. Although compression techniques do exist to manage the problem of sparsity, they tend to destroy the multidimensional data structure's natural indexing (Kay, 2004).[31]

An alternative to adding more dimensions for hierarchical data, as suggested above, could be to add more members on the depth dimension, allowing measures to occupy more than one cell in each member of the array (cf. Glasgow & Malton, 1994: 7, 13). With reference to linguistic data the typical hierarchical Chomskyan tree structure of the syntactic structure of a clause could be represented by such an array structure (see Figure 2.9).

---

[30] Cf. Banchoff (1996: 15).

[31] To solve this problem Karayannidis & Sellis (2003: 156-157) proposed a chunk-based storage manager for OLAP data cubes that is both space conservative and uses a location-based data-addressing scheme. This system is also able to capture hierarchical data.

<table>
<tr><td colspan="7">NP ((1,1,1), (1,1,2), (1,1,3), (1,1,4), (1,1,5), (1,1,6), (1,1,7))</td></tr>
<tr><td colspan="3">NP ((1,2,1), (1,2,2), (1,2,3))</td><td>Particle (1,2,4)</td><td colspan="3">NP ((1,2,5), (1,2,6), (1,2,7))</td></tr>
<tr><td>Particle (1,3,1)</td><td colspan="2">NP ((1,3,2), (1,3,3))</td><td>Particle (1,3,4)</td><td>Particle (1,3,5)</td><td colspan="2">NP ((1,3,6), (1,3,7))</td></tr>
<tr><td>Obj. marker (1,4,1)</td><td>Article (1,4,2)</td><td>Noun (1,4,3)</td><td>Conjunction (1,4,4)</td><td>Obj. marker (1,4,5)</td><td>Article (1,4,6)</td><td>Noun (1,4,7)</td></tr>
<tr><td>et (1,5,1)</td><td>ha- (1,5,2)</td><td>$amayim (1,5,3)</td><td>ve- (1,5,4)</td><td>'et (1,5,5)</td><td>ha- (1,5,6)</td><td>'arets (1,5,7)</td></tr>
</table>

Figure 2.9. A representation of a hierarchical syntactic structure using various members of the same dimension and by allowing measures to occupy more than one cell of a member.

Due to huge space implications in the computer's memory and the difficulty of visualizing four or more dimensions, this research is restricted to three dimensions. Because it is possible to exactly declare the required number of rows, columns and depth layers of a threedimensional array, enough members can be created on the depth dimension to store all modules of clausal analysis.[32]

Other kinds of technology exist to implement multidimensional databases, such as relational online analytical processing systems (ROLAP), which are collections of cuboids or twodimensional relational tables and do not suffer as much from the sparsity problem, but they do not have implicit indexes (Kay, 2004). Although this technology can be researched to evaluate its suitability for solving the problem of this project, it is expected that, due to the rigorous table structures that are inherent in relational databases, this option does not lend itself as well as multidimensional arrays to capture and extract clausal data. According to Koutsoukis et al. (1999: 6)

[32] An alternative approach is followed by Koutsoukis et al. (1999: 12) who combine sparse dimensions (year and season) "to create a 'conjoint dimension'".

"MDDBs[33] are better suited for OLAP-type applications because of their structure and embedded functionality".[34]

One difference between a business data cube and a clausal data cube is that first-mentioned contains data that have already been processed and aggregated (Kay, 2004) while a clause cube contains the basic raw data. However, Karayannidis & Sellis (2003: 157) argue that, in order to support *ad hoc* queries, users should be able to drill down "to the most detailed data in order to compute a result from scratch". It could, therefore, contain hierarchical data consisting of both raw and aggregated data. Compare Chau et al. (2002: 214): "The contents of a data warehouse may be a replica of part of some source data or they may be the results of preprocessed queries or both". A clause cube that contains hierarchical data, such as syntactic tree-structure information, will be similar to such a business data cube. Another important similarity between a business data cube and a clausal data cube is that both types of data are stable, if one assumes that the process of analysis and tagging has been finalised. It does not get updated or changed like data in an online transaction processing system. The data cube concept was developed to focus on powerful analysis of business data, rather than on the fast and efficient capturing of transaction data. These characteristics support the hunch that this technology is very suitable for the storing and analysis of clausal data.

2.5 Building and using a multidimensional database for Biblical Hebrew

To build a clause cube one could integrate the results of various computerized clausal analysis systems. The process that one should follow is similar to the steps used for building a data warehouse, i.e. (Chau et al., 2002: 216):

---

[33] Multidimensional database management systems.

[34] Compare Cheung et al. (2001: 2) for a summary of the advantages and disadvantages of both ROLAP (relational online analytical processing) and MOLAP (multidimensional online analytical processing) – they propose a combination of the two approaches. For an alternative solution compare Chun et al. (2004).

- Extraction of data from existing databases and flat files
- Cleaning and integration of data
- Loading of data in the data cube or hypercube
- Transformation of data into a format that is suitable for a graphical user interface

Once a proper multidimensional data warehouse has been designed and created it can be populated using data from existing marked-up products: hypertext into hypercube! Products using the mark-up language XML[35] are especially suitable for this purpose because the XML tags can be used to convert free text into a database. "Unlike HTML, XML is meant for storing data, not displaying it" (Holzner, 2004: 40). Using XML to convert existing texts into data sources for a Biblical Hebrew linguistic data warehouse will necessitate cooperation, even more than when using HTML to tag hypertext (see Bulkeley, 2002: 649), especially if the various sources are to be integrated properly.

Combining nested loops with threedimensional arrays makes it possible to process the stored information in an efficient way. For example, it becomes possible to slice-and-dice the data cube of clauses to reveal various dimensions. Slicing the cube from the front reveals the Hebrew text, syntactic frameworks, semantic frameworks, etc. Slicing the cube from the top reveals the multi-layer analyses of subsequent clauses. One can also drill down into the cube to reveal other information that is linked to a specific cell. These aspects will be discussed in detail in the next chapter.

2.6 Conclusion

This experiment with a threedimensional data structure indicated that a threedimensional array could be used to represent inherently multidimensional linguistic data regarding Biblical Hebrew clauses. Various layers of linguistic

[35] XML (eXtensible Mark-up Language) can be regarded as a subset of SGML (Standard Generalized Mark-up Language) (DeRose, 1997: 186, 233, 235).

knowledge can be integrated by stacking various modules of analysis onto each other. On a linguistic level the corresponding elements are linked by means of the phrases constituting the clauses. On a programmatic level they are linked by means of the indexes which are inherent and essential to the array structure.

Storing linguistic data in such a threedimensional data cube should enable ways in which this data could be used in an efficient way. The captured data can be viewed and manipulated in various ways, for example to create stacks of twodimensional interlinear tables showing required aspects of clauses' data. In this way the threedimensional array facilitates actions that are typical of online analytical data processing and data warehousing. These issues will be discussed in more detail in the following chapter.

An array, as such, cannot be stored permanently on a hard disk. Therefore, a VB6 program module is used in the initial phases of this project to declare and populate the array (see Addendum A). In follow-up phases XML will be discussed as a suitable mark-up technology that can be used to permanently store the linguistic data in a separate databank (cf. Chapter 4). This will not only facilitate the recovery of the data for advanced processing in the current VB6 project (cf. Chapters 5 and 6), but it will also enable the re-use of the data on other platforms, for example in a Java program that graphically visualises the links between the layers of linguistic data (cf. Chapter 7).