Bayan: An Arabic Text Database Management System

Roger King*
Ali Morfeq
Department of Computer Science
University of Colorado
Boulder, Colorado 80309

Abstract

Most existing databases lack features which allow for the convenient manipulation of text. It is even more difficult to use them if the text language is not based on the Roman alphabet The Arabic language is a very good example of this case Many projects have attempted to use conventional database systems for Arabic data manipulation(including text data), but because of Arabic's many differences with English, these projects have met with limited success In the Bayan project, the approach has been different Instead of simply trying to adopt an environment to Arabic, the properties of the Arabic language were the starting point and everything was designed to meet the needs of Arabic, thus avoiding the shortcomings of other projects A text database management system was designed to overcome the shortcomings of conventional database management systems in manipulating text data Bayan's data model is based on an object-oriented approach which helps the extensibility of the system for future use In Bayan, we designed the database with the Arabic text properties in mind We designed it to support the way Arabic words are derived, classified, and constructed Furthermore, linguistic algorithms (for word generation and morphological decomposition of words) were designed, leading to a formalization of rules of Arabic language writing and sentence construction A user interface was designed on top of this environment A new representation of the Arabic characters was designed, a complete Arabic keyboard layout was created, and a window-based Arabic user interface was also designed

1. Introduction

Text is one of the complex data types involved in the growing research field of office automation Many research systems have been designed to manage complex data types such as text, images, and voice [VLDB83] The primary function of text management systems is to provide for the storage and manipulation of documents An additional function of these systems is to provide a means for conveying the meaning of the text that they manage The implementation of the former function is usually addressed in the area of database research, while the latter function is addressed in the area of artificial intelligence research

Database management systems (DBMS) can provide useful services for the manipulation and storage of data These services include storage mangement, concurrency control, query processing, security, and recovery in the event of a system crash Also, relationships between documents can be established and maintained using a DBMS However, the structure of text data is different from the data types usually found in conventional database systems The structure of text is different in that it has no fixed length It contains a variable number of words or characters This structure property makes it very difficult to use conventional methods employed in DBMSs for performing queries, except in the case of full text scanning [FALO85], where every word is indexed

Text databases are typically very large The access methods used most often on them are insertion and retrieval Updates and deletions are not performed often in a text database Rather, versioning is used more often in order to maintain a history of text modifications for the user Various methods of performing text retrieval have been suggested These methods include the use of inverted indexes and signature files Inverted indexes do not scale up well for use in the potentially large documents a database may be required to store The signature file method attempts to address this problem [FALO84] In the signature file method, a set of keys is chosen to describe the text They are then encoded into a signature by which the text can be identified

Various database models have been used to implement text mangement systems In [STON83] the relational model was used for document processing systems A network-based approach was used in the TEXTNET project [TRIG86] The Bayan Arabic text database management system employs an object-oriented model An interested reader can consult [SALT89] [FLAO88] [WOEL86] [GONN87] for background discussions about the design of text processing and management, [HOAR85] for a discussion of a standard architecture for document interchange , and [FALO84] [FALO85a] [PEEL85] for

* This researcher was supported by ONR under contract N00014-88-K-0559, and by AT&T under contract number OCG0545B

discussions on different Access methods for text DBMSs

In the next section we will give a brief background discussion of the properties of the Arabic language Section 3 will give a description of the Bayan data model A description of the user and database interfaces is given in the 4th section In section 5, we will give a brief description of some of the current and future work that is being done in building Bayan's environment.

2. Background

Arabic language processing has been hampered because, for the most part, people have simply attempted to adopt English-based computer systems for Arabic such as [TAYL86] This is very difficult and, to some extent, futile because the nature of the Arabic language is much different than that of English Arabic, for example, is written from right to left. The Arabic sentence structure and grammar are very different from that in English Therefore, existing systems ("adopted English systems") cannot adequately process Arabic language data In Bayan, on the other hand, the Arabic properties are served first This new environment will try to examine the problems in working with the Arabic language by using Arabic text databases as an application of that environment

Bayan addresses those problems that have led to the shortcomings in the current Arabic programs In the next subsection, an overview of the properties of Arabic will be given so the reader may appreciate some of the possible problems in dealing with Arabic using an English-based system Some aspects of Arabic word processing requirements can be found in [BECK87] [AHME86]

The Arabic Language

Arabic is the official language of 22 countries stretching from eastern Asia to northwest Africa. Furthermore, there are other languages which use a modified Arabic alphabet (such as Persian and Urdu) The importance of Arabic is perhaps obvious to the reader

2.1. Properties of Arabic Script

The following subsections describe the general properties of the Arabic script, including characters, digits and writing methods

Arabic characters

There are 28 characters in the Arabic alphabet The sounds of 12 of them do not map directly to sounds that are part of the English language The following are some important properties of the alphabet

- Arabic may only be written in cursive script
- Arabic is written from right to left.
- In writing, each character is connected to the preceding or following characters, except six characters, called "stubborn characters" These six stubborn characters are { و ر ز د ذ ا }
- Each character, in general, takes on a different shape depending on whether it appears in the beginning, middle or end of a word or if it is standing by itself (for example, after a "stubborn character" at the end of a word)
- The Arabic alphabet contains no vowels as such Instead, diacritical marks, called "tashkeel", are used to represent vowel sounds They are written above or below the consonants There are 13 "tashkeel" Four "tashkeel" are basic and nine are derived by combining one or more "tashkeel" The "Tashkeel" are essential to understanding the sentence properly, without the tashkeel, a sentence may easily be misunderstood This is because the placement of the "tashkeel" is determined by the rules of grammar, that is, the "tashkeel" tells the reader if a given word is a verb, subject, adjective and so on
- Each character in Arabic could be pronounced in 13 different ways, depending on the "tashkeel" written with it

Arabic numerals

Although the numerals of the English language are referred to as Arabic numerals, they differ completely from the numerals used throughout the Arabic world

There are ten digits that are used in Arabic These digits are { ١ ٨ ٧ ٦ ٥ ٤ ٣ ٢ ١ ٠ } The numbers are written and read from left to right (not right to left, like Arabic words) This implies, obviously, that the least significant digit in a number is the right most digit

2.2. Existing Standards

There have been many attempts to simply modify English computer systems to accommodate Arabic Such attempts have ranged from modifying input and output (I/O) device drivers to using graphics to draw the Arabic script within application programs All of these projects have failed in their attempt to represent all of the aspects of the Arabic language For example, many of them do not have any means of representing the "tashkeel".

One of the proposed standards for representing Arabic characters in computers may be found in [ALSA86] That work contains a proposed representation of a 7-bit Arabic character set (without Tashkeel) and an 8-bit set for both Arabic (without Tashkeel) and English Still these sets are not able to meet all the requirements of satisfactorily representing the Arabic language The source of these shortcomings, again, is because these character sets were intended to work with English machines

In general, the proposed standards lack the following

- The ability to represent all of the Arabic "tashkeel", especially the ones that are used as a combination of "tashkeel" (for example, "ّ")
- In general, they represent "tashkeel" as characters added to a word Two words may have the same characters but differ in meaning because of their

different "tashkeel" When considering "tashkeel" as characters added to a word, this difference cannot be taken into consideration

- They arrange some of the characters with special markings as unique characters while they are simply the same character combined with different shape characters For example, (ﺄ ﺇ ﺃ) are all the same character, called "hamza" The system must consider all of them as one character and not as different characters
- Finally, these standards are not accepted by all Arab countries and, therefore, until now there is no uniformity on this matter

Most of the proposed standards are simply derivations from [ALSA86]. Some simply change the ordering of the characters while others remove or add some extra characters There are some application programs which which tried to solve some of the problems related to the "tashkeel" by creating a character set which included each of the "tashkeel" However, since there are only 127 character entries in the 7-bit extended character sets, they were forced to ignore some of the "tashkeel" in their implementation

3. Bayan's Data Model

Many systems have tried to manipulate English text using different data models In [STON83] a text-based database is presented using the relation model In [TRIG86] a network-based approach for text handling is presented In [PEEL85] a method for representing documents is presented

In our system, we use an object-oriented model Every entity within this model is modeled as an object Each object has a number of attributes that hold the values and properties of different object characteristics Methods are associated with each object to perform different operations on the object Manipulation of the object can only be accomplished through the use of these methods We chose this data model because it gives us a higher level of abstraction of the entities in the database than other models This suits our requirements for a flexible, extensible, and modular data model A more detailed discussion of object-oriented databases and their usage can be found in [KING86],[HUDS88], and [MANO86]

By choosing an object-oriented model for our design we gave ourselves the ability to divide the problems of representing text data into smaller problems which can be solved one by one These solutions can then be used to produce a more complete solution which can be extended with new features in one part without affecting the structure of other parts of the data model This is a new way to look at text data, as objects rather than strings of characters This will allow for the reuse of text in many documents We think that this can be a very useful idea even in English based text systems

The main components of Bayan's Object manager are shown in [Figure 3 1] The application program uses the object manager interface The object manager then performs the requested operations on the objects All operations on objects should go through the object manager The object manager also manages the object buffer The object buffer is where the objects are kept to allow further operations to be performed on them In figure 3 1 we can see that some objects are in the object buffer, while others are still in the object store The object manager contains all the information about the different classes The class information includes the structure of each class object and the different methods of access

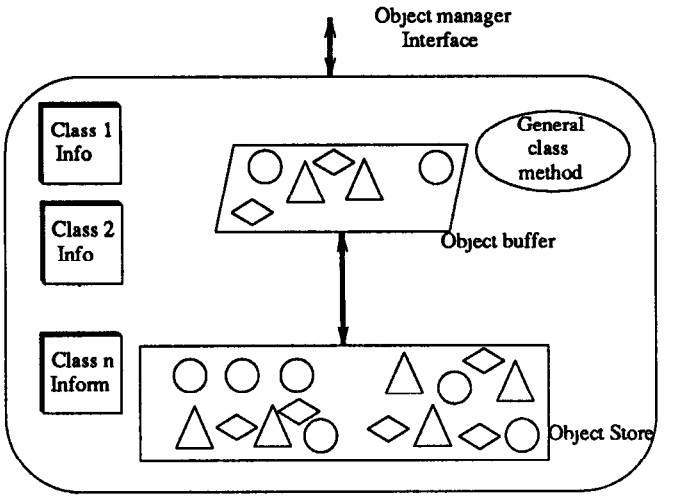

Figure 3 1
Object Manager
components

3.1. Standard Class Methods

There are some standard methods that can be used with all objects in the database These methods are necessary for creating, updating, deleting, and storing objects in the database Some of the standard methods are the following

- **Create_Object:** This method will take an object identifier and create an instance of a given class This method will only create an instance in the memory buffer To store this object, the application should use the Save_Object method An optional automatic save operation can be set With the automatic save operation activated, every time an object is created the object will be created in the buffer and in the object store The same applies if an attribute is changed This feature is used to accommodate the need to use the 'what if' concept The user can make changes in the memory buffer to check the results of a change If he wants to make those changes permanent he can commit them to the object store using

the Save_Object method

- **Save_Object**· This method will save a given object to the object store This is done automatically if the Automatic_Save option is requested
- **Get_Object:** This method will search and place a given object into a buffer memory for access by the application program If the object is in the buffer then it will inform the application where the object is If the object is not in the memory buffer, then it will read it into the buffer and inform the application where the object is The buffer manager uses the Least Recent Used (LRU) method for buffering objects
- **Modify_Attribute:** This method will modify the contents of an attribute The attribute name and the new value must be given as arguments to the method
- **Change_Access_levels:** This method can be used to change the authorization code for accessing an object This method takes authorization codes similar to the protection values used with files in the operating system This includes self, group, and world access codes Only appropriately authorized people can access a given object This method can be used for concurrency control by changing the access code to the appropriate values to lock or unlock an object
- **Delete_Object** This method will remove an object from the database Because of the nature of text data , it is very seldom that an object is deleted from a database Versioning is used extensively in text databases to keep track of changes to the data Bayan at the time being allows the deletion of objects from the object store Versioning will added to Bayan

3.2. Bayan's object classes

Bayan has three main classes of objects that will cover the required representation of text objects in general and in Arabic text in particular These classes are the Arabic_Document class, Text_Unit Class, and Arabic_Word class Objects of the first class can be composite objects This means that they can have other objects as one or more of their attributes Objects of the second class may not have other objects as attributes In general, objects in Bayan may contain other objects (composite objects), links to another object's attributes (derived attributes), and methods to access these objects

Objects of the third class (Arabic_Word) are used extensively by the query manager The Query manager is discussed in the following sections

[Figure 3 2] shows the block diagram of the relationships between the different objects which compose a book Note that all objects are in the same object store (similar to the objects shown in [figure 3 1]) We chose different levels to display the objects to make the figure easier to understand The arrow means that the originating object is used as an element in the target object

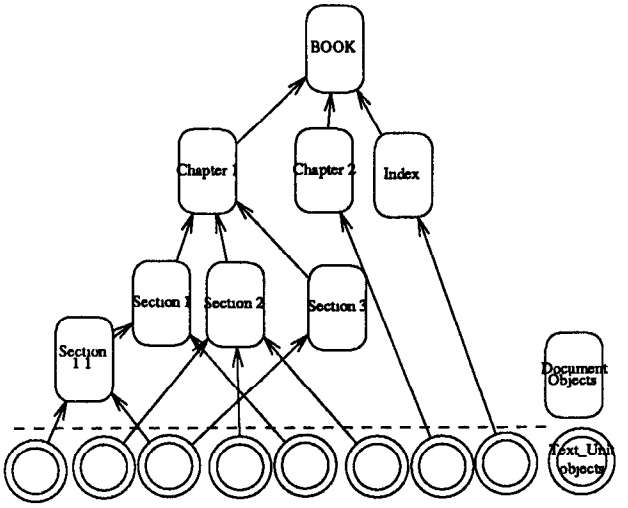

Figure 3.2
Example of how to build a book in bayan.

Arabic_Document class

The first of the classes in the Arabic database is the Arabic_Document class This class will contain the elements that will compose the document These elements can be other documents or a number of Text_Unit objects The following is a description of this class The description will not contain the standard methods supplied by the object manager Only those methods that are special to the Arabic_Document class are shown

```
CLASS Arabic_Document
{
  ATTRIBUTES
     DATE_TYPE        creation_date_time,
     DATE_TYPE        access_date_time,
     AUTHOR_ID_TYPE   authors_id,
     ACCESS_TYPE      access_permissions,
     STATUS_TYPE      Document_status,
     OBJECT_ID_TYPE   Elements[ ],
  METHODS
     LIST_ALL_ELEMENTS( ),
     ADD_AN_ELEMENT(Position, Element_obj_id),
     DELETE_ELEMENT(Position, Element_obj_id),
     OUTPUT_TEXT_BODY( ),
}
```

The attributes are specified such that the attribute type is written first followed by the attribute name itself

For example, 'creation_date_time' attribute will have the type 'DATE_TYPE' which is defined somewhere else While in the case of the Elements attribute, it will be of the type 'OBJECT_ID_TYPE' which means that this attribute value will contain a list of Bayan objects identifiers

The LIST_ALL_ELEMENTS method is used to list all the elements that compose a document. It will return a list of Text_Unit objects that compose that object If the Elements attribute contains an Arabic_Document Object then this method will be applied again to that document to get its Text_Unit objects By doing this recursively we will have a list of all Text_Unit objects contained in a document and can use this to print or process that document

The OUTPUT_TEXT_BODY method will take the LIST_ALL_ELEMENTS method one more step It will take every Text_Unit object and output its text body This can be used to print or display a document

Text_Unit class

This class of objects is what we can think of as a "Text_Unit" This means that the objects in this class will contain the atomic text body This class will contain the actual information we are trying to manage

Large raw text data will be divided into small Text_Unit objects Then documents can be composed of these objects Documents can contain Text_Unit objects and other document objects For example, A book can be divided in the following way Each paragraph is a Text_Unit object, a group of these Text_Unit objects can be grouped to form a small document object which represents subsections These documents (representing subsections) can be grouped to form documents representing sections These documents (representing sections) can be grouped to form another document representing a chapter These documents (representing chapters) can be grouped further to form a book Tables and Figures can be thought of as another Text_Unit object and can be included in a document A picture, on the other hand, can be thought of as a Text_Unit (if it is a simple picture) or as a document (if it is a complex one) composed of smaller documents and Text_Units Figure 3.2 shows an example of how a document can be constructed to form a book The structure of the Text_Unit class is as follows

```
CLASS Text_Unit {
    ATTRIBUTES
        DATE_TYPE        access_date_time,
        AUTHOR_ID_TYPE   authors_id,
        ACCESS_TYPE      access_permissions,
        Arabic_Text_TYPE text_body,
    METHODS.
        OUTPUT_TEXT_BODY( ),
        EDIT( ),
}
```

The 'text_body' attribute will contain the actual text data the system is managing The 'OUTPUT_TEXT_BODY' method is the one used to output the body of text in the object, while the 'EDIT' method will be used to invoke an editor to edit the text body. Bayan has a built-in Arabic text editor The Arabic_Text_TYPE consists of special structures to hold the Arabic text We designed it this way to test the different methods of storing text (internal representations) Currently we are using the coding method shown in the next section, to represent Arabic characters

Arabic_Word class:

This class will contain Arabic word objects and all the information about them It will contain information about the classifications of words according to the rules of the Arabic language These classification can be used to determine the linguistic type of a word Derived words are stored as root words with the proper derivation methods associated with them The structure of the Arabic_Word class is as follows

```
CLASS Arabic_Word
    {
    ATTRIBUTES
        WORD_TYPE        Word_class,
        ARABIC_CHAR      Root[ ],
        DERIVATION_METHOD derivations[ ],
    METHODS
        IS_LEGAL(Derivation_rule) ;
        ADD_RULE(Derivation_rule),
        MATCH( Word_To_Look_For),
        MAKE_DERIVED_WORD( ),
        LIST_ALL_DERIVED( ),
    }
```

The attribute 'Word_class' is where the word is classified according to the Arabic language Classification may include root, verb, noun, tools, or special word The 'Root' attribute will contain the actual word in Arabic It is defined as a list of Arabic characters (see section 5 1) The 'derivation' attribute is the one which will hold all the information about the possible derivations from the root word The derivation rules are explained in the following sections

The method 'IS_LEGAL' will determine if a specific derivation rule may be applied to a root or not The method 'ADD_RULE' will add a new derivation rule to an object The object method 'MATCH' will match a given word to its root or one of its derived words It will return TRUE if the word can be matched The object method 'MAKE_DERIVED_WORD' will derive a new word from the root using the specified derivation rule Finally, the 'LIST_ALL_DERIVED' method will output all possible derived words from the root word using the stores derivation rules in the object store

4. The Bayan environment

The Bayan environment attempts to solve the problems described in section 2 by attempting to fully represent

Arabic script In particular, representing the different forms of "tashkeel" is given high priority Bayan also includes a database of Arabic words, roots and derivations rules This environment could be a base for future Arabic natural language processing research Figure 4 1 shows the major components of the environment The application interface will give application programs the ability to communicate with Bayan's DBMS It will be responsible for getting information to and from the application program The application program can only use this interface to access the Arabic Text data of the system Each component is discussed in the following sections

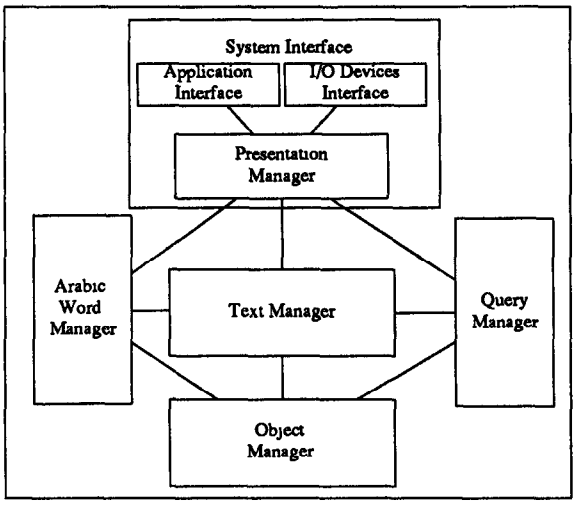

Fig 4 1
Bayan Major Components

5. I/O interface

The I/O interface includes drivers to write and type Arabic script according to the Arabic language rules. We will discuss a character set, keyboard layout, and our contextual analysis algorithm. The contextual analysis algorithm is the one which makes sure that all Arabic characters are displayed correctly with the proper shape

The Arabic character set

A 16-bit coded method is to be used in the Bayan system Arabic characters are encoded in such a way that each character is represented by one and only one code That is, the "tashkeel" do not affect the value of the character Instead, the "tashkeel" are treated as a description or a property of the character They can be treated in the same way underlining, highlighting and other screen display information are treated A character will be represented by a byte while attributes (such as underlining, highlighting and so on) are represented by another byte It was decided to represent Arabic characters as in Figure 5 1 In this figure, the right hand side byte is used to represent the characters while the left hand side byte is used to describe the properties of the character (i e the "tashkeel")

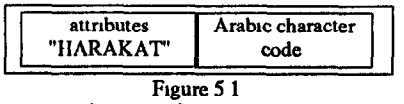

Figure 5 1
A proposed representation of an Arabic Character

It may be possible to make the code-byte values a modification of the proposed standard found in [ALSA86] in order to allow some compatibility with the proposed standard However, the character forms that represent the "tashkeel" are removed as, it is argued here, it is more efficient not to include the "tashkeel" in the character set, rather, they should be considered as attributes to each character Furthermore, it is also proposed here, that each character should be stored in one code only

Output drivers must be able to determine the proper shape of each character (depending on where it occurs in the word) along with the correct positioning of the "tashkeel" This is done by building a contextual analyzer which knows how Arabic script is written

An Arabic keyboard

Since there are only 28 characters in the Arabic alphabet, it is possible to use an English keyboard and simply modify it to resemble the Arabic typewriter keyboard The Bayan's keyboard is shown in Figure 5.2 Note that the "tashkeel" have keys on the Bayan keyboard Arabic typewriters typically do not have "tashkeel" (the tashkeel are added by hand)

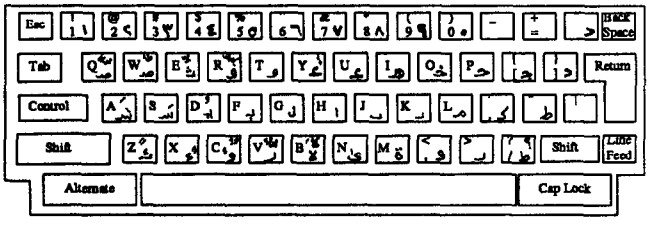

Figure 5 2
Bayan's Keyboard Layout

Device drivers should be able to map the scanning code from the keyboard to the appropriate Arabic code and attribute This keyboard layout is a collection of different proposed layouts and it overlaps with many of them The default input method is to enter the Arabic character followed by its "tashkeel" However, an editor has been added in order to allow the user to first type characters without entering the "tashkeel" and to go back and add the proper

"tashkeel" later if so desired This feature was added because some users prefer to write a word first and then go back and add any desired "tashkeel"

Contextual analysis

In Bayan, a contextual analyzer was built The role of this analyzer is to determine the shape of the Arabic character according to its position in a word The analyzer takes into consideration whether the character appears at the beginning, middle or end of a word It will represent the character properly and also place the "tashkeel" in their proper place

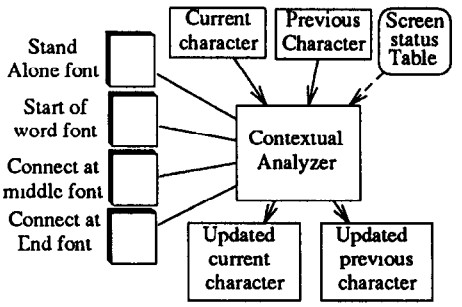

Figure 5 3
Bayan's Contextual analyzer

The block diagram of the contextual analyzer is shown in Figure 5 3 It uses four fonts for the output of Arabic characters There is a separate font for each possible position or shape of a character These fonts are · "stand-alone font", "start-of-word font", "connect-at-middle font", and "connect-at-end font" The analyzer will take the current and previous character codes and their shapes as its input and will return the correct shapes of the previous and current character This means that the calling procedure to the contextual analyzer should check if the previous character changes shape after the return from the contextual analyzer call The algorithm to determine the shape of the character at the current position can be found in [MORF89]

The contextual algorithm will take the current and previous characters as inputs The contextual analysis algorithm will examine the previous and current characters If the previous character can not connect to the current character (e g space, comma, a stubborn character,), then it will make the current character's shape to be 'START_POSITION_SHAPE' and it will exit Next, the algorithm will check if the current character is a space, a punctuation mark, or a digit, in which case, it will set the shape of the previous character to be the proper ending shape depending on the shape of its previous character(i e the second to the last character) If both of the above tests fail, the algorithm will change the current character to connect to the previous character depending on the shape of its previous character

```
Data structures and constants.
   identifier
        START_POSITION,
        MID_CONNECT,
        END_CONNECT,
        STAND_ALONE, /* identifiers for
                    character shape
                    it can be assigned
                     the values from 0-3
                  */
   Arab_Char· Structure of {
      Code  Byte,      /*Arabic character code*/
      Tashkeel  Byte,  /*Tashkeel code*/
      Shape  int,      /*character shape number*/
                  /* it must be one of the above shapes */
   }
```

The identifiers are used to represent the different fonts used in our system For example, START_POSITION is the font that contains the shape of all characters when they are at the starting position of a word

In the following table, we show an example of how contextual analysis determines the shape of the Arabic character displayed The first column shows the current input character The second column shows what is being displayed as a result of the input so far The third column contains comments of what has been done The black rectangle is the cursor.

Input character	Ouput on screen	Comments
س	س	start of word shape
ل	سل	connect the second to the first letter
ا	سلا	a ligature is a must, so go back and join both into one new char
م	سلا م	start of word shape because the previous can not connect to next
	سلام	space cause the last letter to have ending shape

6. Text Manager

This part of Bayan will be responsible for managing a document object's input and output It prepares the Arabic_Document objects to be used by other parts of the system The text manager will take a Text_Unit object or an Arabic_document and prepare the text body using the word manager and pass the resulting text to the presentation manager to be used by the calling procedure

The Text manager will also cooperate with the Query manager to find the required text items, which will also

involve calling the word manager to get the root of the word(s) that it is looking for A more detailed discussion can be found under the 'Query manager section' The text manager will also take a plain text and prepare it to be used by the query manager for storing the word object's references within documents for later use [Figure 6 1] shows the major parts of the Text manager

The Natural Arabic Text Coder/Decoder

This part of the Text manager is used to take Arabic text and split it into pieces. each piece consists of an Arabic word If a word can not be found, it will be flagged as unknown

It is also used to add text units to the system after the creation of its text body The same routine can be used to import existing text from other systems into Bayan. Care should be taken when importing data from other systems since the representation of the data may differ

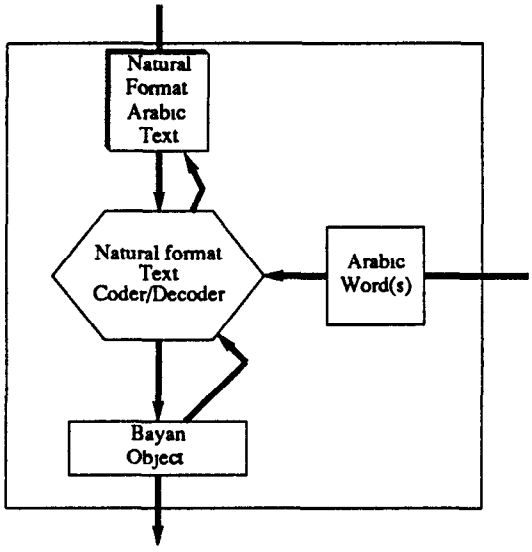

Figure 6 1
Text manager
Components

7. The Arabic word manager

The word manager uses Arabic_word objects extensively It is is responsible for deriving words from their roots and reporting the resulting word to the requesting procedure It must know the rules for word derivations in Arabic as well as have the ability to learn new rules. Every Arabic_Word object will have the derivation rule information in the "derivation" attribute The Arabic word manager will be able to use that information to derive new words The Arabic word manager will be able to receive new derivation rules from the user and add that to the Arabic_Word objects This feature will allow the user to add additional rules later if needed This is done because of the number of the derivation rules and derived words in the Arabic language Figure 7 1 shows the major components of the word manager

This manager allows a database administrator (DBA) to enter what is known as the teaching mode In the teaching mode, the learning manager is called and then the DBA can add new words to the vocabulary of the system, add new derivation rules and specify which words it applies to In the teaching mode, the DBA can modify existing derivation rules for the roots of an Arabic word

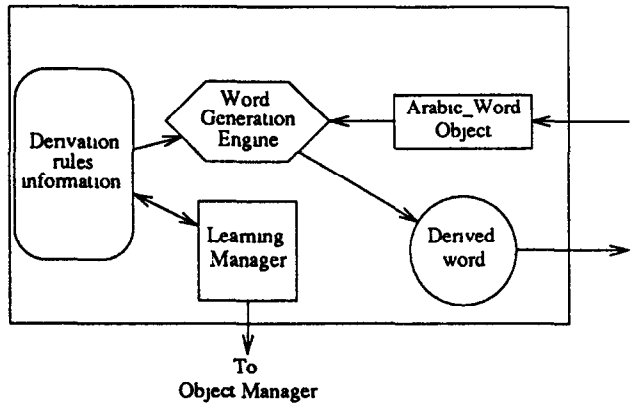

Figure 7 1
Word manager components

The learning manager

This part of the word manager can learn new words It takes information about a word and adds it to the object manager. It is important that this manager only be used by someone proficient in Arabic as incorrect information can be harmful to the database since Bayan stores words as objects rather than actual characters of a word

Derivation rules are defined for each word by use of the learning manager This is the only way to add a word to the vocabulary of the system Arabic words are added to the database objects through this option after obtaining the required information to form a complete word object and to check if the word already exists

The learning manager uses an interactive window-based interface that prompts the user to supply the required information This process guides the DBA by requesting the needed information before accepting the new word The DBA is responsible for the correctness of the information in the system

The learning manager can also accept new derivation rules. Rules can be specified by the DBA using the proper format Rules can be added to the knowledgebase of the learning manager and are used by the word generation engine.

The word generation engine

The word generation engine interprets derivation rules and executes commands encoded in the derivation rules It is like a small computer executing a limited instruction set It will take a root as an input and produce the newly generated word (if any) as output If there is an error, the same root will be returned with a flag indicating the presence of an error Errors can range from a null root to the wrong number of characters in the root and so on

Derivation rules

All the rules are the same for each root class of words Therefore, they need to be stored only once, afterwards, the word generation manager can generate the desired word from its root It should be noted that rules are completely different for three character roots, four character roots and five character roots

In general, a derivation is performed by adding characters and/or changing the "tashkeel" of the word The encoding of the derivation is as follows

<digit>
concatenate character number <digit> of the root adding the following "tashkeel", where each <digit> will be followed by the proper "tashkeel".

<character>
concatenate <character> where character represents an Arabic character with its "tashkeel"

An example of what the derivation rules look like is the following

مْ ا ْ ؤ ْ ِ ا ِ ن

مْ ا ْ ؤ ْ ِ ة

Arabic derivation rules, however, do not only involve the concatenation of characters at the end of a word Characters may also be added to the middle, beginning or end of a word to derive a new word

An example:

Given the following:

Root = نَصَرَ

Rule = مْ ا ْ ؤ ْ ُ ونَ

Result = مَنْصُورُونَ

The steps for this result is shown in the following table

Step	Result	Comments
1	مَ	the new word starts with the arabic character 'Meem'
2	مَنْ	letter number (1) of the root word to be added
3	مَنْصْ	letter number (2) of the root word to be added
4	مَنْصُو	letter (waow) to be added
5	مَنْصُورْ	letter number (3) of the root word to be added
6	مَنْصُورُو	letter (waow) to be added
7	مَنْصُورُونَ	letter (noon) to be added
8		return the new word

8. Query manager

This part of the Bayan system receives requests to search for a word in a document(s) or simply answering a question about a word An example of a question can be What are all the derivatives of the following word's root? The result is returned to the calling procedure

The Query manager will be able to take words and search for all their occurrences in the documents in the database It can search for exact words, for root derivatives of a word, or words of similar meanings It returns a list of object identifiers (Documents or Text_Unit) which have the desired word(s) in them This information can then be used by the presentation manager to display the body of the text

The query manager uses a morphological analysis algorithm (which under development at this time) which will be able to parse Arabic words and returns the components which make up a word These components include root words and different affixes

8.1. Morphological analysis of Arabic

A morphological analyzer for Arabic words is in the process of being built It provides information concerning Arabic words such as the recognition of different parts of a word, the root of a word, and affixes attached to words

The following information was required for constructing the morphological analysis algorithm for Arabic words

(1) The roots of words along with all the derivation rules that can be applied to them,

(2) The derived word and root links (which can be performed by a lookup table or by a smart algorithm) which takes a word and returns its root,

(3) A list of all possible affixes that can be attached to a word and that are not covered by the derivation rules, and

(4) A list of words that are not derived from roots, such as tools (e g , "from", "to", "on" and so on)

We are constructing a grammar which can be used by a parser to parse Arabic words Constructing the grammar will require a considerable amount of work to collect and study most , if not all, the cases of the derivation and classification of Arabic words

8.2. The Search

When the Query manager determines which word to look for (using its own algorithm plus the morphological analyzer), it will search in a search structure similar to the one found in [SALT89] to find the occurrences of the word in the database documents The Bayan search structure is shown in [Figure 8 1]

At the Query word-level, an entry of an Arabic word (root) is stored with a pointer to a list of documents-Text_Unit level At the document pointer level, there is a counter of how many documents a word appears in The structure stores pointers to the lowest level which is called the Document-Unit_text level At the Document Unit_text level, it will contain the Document object identification and Text_Unit object Identification Those identification numbers can be used for accessing the documents at the Document level or at the lower level of the Text_Unit objects

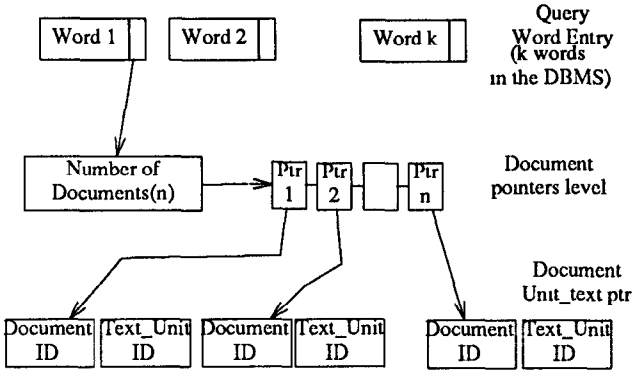

Figure 8 1
Bayan-Query manager
search structure

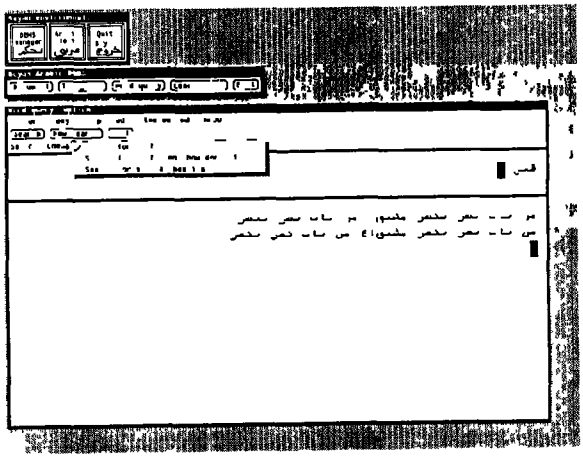

Figure 8 2
A prototype of Bayan session

When a word comes to the query manager, it searchs for the word at the query_word entry level If it does not find it, it will report a failure in the search If it finds the word, it follows the pointer to the next search level which contains a variable number of pointers to the document Text_Unit level The query manager follows the first pointer and retrieves the document id and the Text_Unit id, and returns them to the requesting process Upon the request of the user, the query manager will be able to go through all the pointers and return the proper identification of the Documents and the Text_Unit objects until all pointers are explored

An additional feature that can be added to the search structure is a synonym pointer which will point to word search entries which give the same meaning as the original word Figure 8 2 shows a prototype session of Bayan with the query manager invoked This prototype was built to display English messages to give English speaking readers the ability to understand it The final system will be fully in Arabic

9. Bayan's user interface

Bayan's user interface was built in a SUN workstation environment Machine dependency was kept to a minimum The existing interface will allow the use of Arabic text in windows, it will also perform all proper input and output operations on that window Furthermore, Arabic text may be edited in these windows

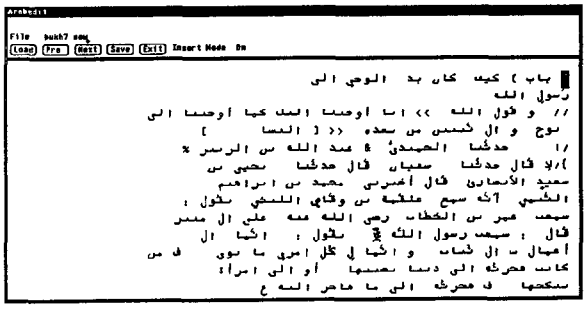

Figure 9 1
Arabic window in Bayan

Figure 9 1 shows an Arabic window which displays Arabic text information Note that the text has displayed the proper "tashkeel" for every word of text The figure shows the main options allowed by the system Currently the Text_Unit option is activated The Text_Unit object called "Bukh7 new" is displayed The other options are document, query, learn and exit options The document option, for example, allows the user to manipulate and create documents

10. Scenario

A user will create a Text_Unit object using Bayan's editor Figure 9 1 shows an editing session of a Text_Unit object called 'bukh7 new' External files can also be imported to Bayan's editor using the "Load" option An option can be activated by clicking on the button which represents the command The updated Text_Unit can then be stored in the database using the "Store" option The user, when done, can ask Bayan to incorporate the new or modified Text_Unit object into its final database Bayan will check the Text_Unit object for new words, index the words, or it can activate the learning manager if a word is not known to Bayan

The user can specify the information which Bayan's learning manager prompts him for This will include the specification of which class of Arabic words this word belongs to The user will also verify the derivation method used to derive this word from its root Additional information may be requested to determine affixes added to the word If the user made a mistake in the word, he can go back and modify the Text_Unit object using the editor, then ask Bayan to continue its operation Bayan will add this information to its database The user can query that Text_Unit object by words that appear in it

Documents can be built using the "Document option" which can be selected from the main menu The user will be required to enter the elements which make up documents These elements may include Text_Unit objects and/or other documents

The user can make queries about words within document objects The result of the query will be a list of the document objects in which the word appears The user can then specify that he wants to see the Text_Unit object level and the Text_Unit objects will be displayed At the Text_Unit object level the user can browse or modify these objects If he modifies a Text_Unit object he would need to ask Bayan to update its database as explained above

Queries can also be made about Arabic words, their roots, or their derivation methods Queries can ask Bayan to show examples of words derived from the same root, or even words of similar meanings

11. Conclusion

Bayan is a text database management system which was designed to overcome the shortcomings of conventional database management systems in manipulating text data in general and Arabic text in particular Bayan is a starting step for more advanced research regarding Arabic It points to the possibility of more advanced studies about the language, such as natural language processing based on Bayan's environment For now, though, many persistent problems have been addressed, such as the design of an Arabic character set including the "tashkeel", an Arabic keyboard, word-generation algorithms and Arabic word generation methods from the roots of words At the present time, the Arabic text database, which utilizes the representation and algorithms designed in Bayan, is being built In addition, a built-in morphological word analyzer is also under construction Bayan should be able, at the end, to handle large amount of text data written in Arabic and give the user a way to query this data

Acknowledgments:

We wish to thank the database group for interesting discussions of the issues, especially Athman Bouguettaya, William McIver Jr , Pam Drew, Kequn Zhao, and all other people whose discussions were very useful We would like also to thank Jim Martin for a very useful discussion of the issues of natural language processing

Refrences:

[AHME86] Ahmed S Ahmed, *'About retrieving the Quran in Othmani script using computers'*, 9th national computer conference in Saudi Arabia 1986

[ALSA86] Al-Saleh Mohammed, *'Standards for Arabic character in information'*, 9th national conference in Saudi Arabia, 1986

[BECK87] Joseph D Becker, *'Arabic word processing'*, Communication of the ACM, July 1987

[BOOT86] L Booth, et al, *'Arabization of an automated library system'*, 9th computer conference in Saudi Arabia, 1986

[CHRI84] S Christodoulakis, J Vanderbroek,et al Proceedings of the tenth International conference of VLDB, 1984

[FALO85] Christos Faloutsos , *'Access methods for text /'*, Computing Surveys, Vol 17, No 1, March ,1985

[FALO85a] Chris Faloutsos and Stavros Christodoulakis, *'Signature Files An Access Method for Documents and its Analytical Performance Evaluation /'*, ACM Transaction on Office Information Systems,Vol 2, No 4, October, 1984

[FALO88] Christos Faloutsos and Ralph Chan, *'Fast Text Access Method For optical and large magnetic disks Design and performance comparison/'*, Proceedings of 14th VLDB conference,1988

[GONN87] Gaston H Gonnet, Frank Wm Tompa, *'Mind your grammar a new approach to modelling text '*, Proceeding of the 13th VLDB conference, Brighton , 1987

[HASS87] Mohamed Hassoun, *'Conception et relisation d'une base de donnees pour les traitements automatique de la langue arabe'*, 10th computer conference in Saudi Arabic, 1987

[HUDS88] Scott Hudson and Roger King, *'The Cactis Project Database Support for Software Engineering'*, IEEE Transactions on Software Engineering, June, 1988

[HUDS89] Scott Hudson and Roger King, *'Cactis A self-adaptive, Concurrent Implementation of an Object Oriented DBMS'*, ACM Transaction on database systems, 1989

[King86] Roger King, et al , *'CACTIS A database system for specifying Functionally-defined data'*, Proc 1986 Workshop on object oriented database systems, Spet 1986

[MANO86] Frank Manola and Umeshwar Dayal, *'PDM An object-oriented Data Model'*, IEEE Transaction on software Engineering , 1986

[MORF89] Ali Morfeq, Roger King, and Aqil Azmi, *Bayan A Text Database Managment System which support a Full Representation of the Arabic Language*, the Quarterly Bulletin of IEEE Data Engineering, Decemeber, 1989

[PEEL85] Arno Peels, Norbert Janssen, and Wop Nawijn, *'Document Architecture and Text formatting '*, ACM Transaction on Office Information Systems, Vol 3, No 4, October, 1985

[SALT89] Gerard Salton, *'Automatic Text Processing, The Transformation, Analysis, and Retrieval of information by computer '*, Addison-Wesley Publishing company 1989

[STON83] Michael Stonebraker, et al , *'Document Processing in a relational Database System '*, ACM Transactions on office information systems, Vol 1, No 2 , April, 1983

[TAYL86] Murat Taylı, et al , *'Intelligent Arabic workstation'*, 9th computer conference in Saudi Arabia, 1986

[TRIG86] Randall Trigg and Mark Weiser, *'TEXTNET A network-based approach to text handling '*, ACM Transaction on office information systems, Vol 4, No 1, January, 1986

[VLDB83] *Panel on complex Data objects text, voice, images Can DBMS manage them?* Proceedings of Very Large Databases, 1983

[WOEL86] *'An object oriented Approach to Multimedia Databases'*, Communication of ACM 1986