THE LANGUAGE DATA REPOSITORY:

MACHINE READABLE STORAGE FOR SPOKEN LANGUAGE DATA

A Senior Honors Thesis

By

MICHAEL NEAL AUDENAERT

Submitted to the Office of Honors Programs
& Academic Scholarships
Texas A&M University
In partial fulfillment of the requirements of the

UNIVERSITY UNDERGRADUATE
RESEARCH FELLOWS

April 2000

Group: Computer Science

THE LANGUAGE DATA REPOSITORY:

MACHINE READABLE STORAGE FOR SPOKEN LANGUAGE DATA

A Senior Honors Thesis

By

MICHAEL NEAL AUDENAERT

Submitted to the Office of Honors Programs
& Academic Scholarships
Texas A&M University
In partial fulfillment of the requirements
For the Designation of

UNIVERSITY UNDERGRADUATE
RESEARCH FELLOW

Approved as to style and content by:

Lisa Ann Lane	Dick B. Simmons	Edward A. Funkhouser
(Fellows Co-Advisor)	(Fellows Co-Advisor)	(Executive Director)

April 2000

Group: Computer Science

ABSTRACT

The Language Data Repository:

Machine Readable Storage For Spoken

Language Data. (April 2000)

Michael Neal Audenaert
Department of Computer Science
Texas A&M University

Fellows Co-Advisors:
Dr. Lisa Ann Lane, Department of English
Dr. Dick B. Simmons, Department of Computer Science

The Language Data Repository project is working to develop a software architecture capable of storing the transcripts and recordings of spoken language data and capable of hosting software tools to aid in the analysis of that data. The proposed software architecture can be used by multiple people to store linguistic data from multiple languages on either local machines or non-local machines that can be accessed via a network by multiple users simultaneously. The primary user community for the LDR software comes from a targeted subset of linguists conducting research on language groups with no officially established or standardized writing system. These linguistic field workers are typically involved in activities such as: learning these "unwritten" languages, developing orthographic systems, beginning literacy programs, and producing written texts in the new orthographic system (e.g., Bible translations and traditional stories). The secondary user community consists of linguists who need a reliable method of storing spoken language data and the transcripts of those data, regardless of the existence of an established or standardized written code for that language. Such a software system offers two main improvements over current, paper-based methods of recording transcripts of linguistic data. First, by utilizing machine readable storage, it will enable linguists to use computational tools to aid in linguistic analysis by increasing the ability to quickly and accurately test and evaluate linguistic hypotheses of the rules governing the linguistic systems. Secondly, a standardized method of recording data in a machine

readable format will enhance linguists' ability to document their research and share their results with a greater number of colleagues than previously possible. A benefit to this increase in the distribution of primary data to other colleagues is the ability for more people to test various hypotheses simultaneously.

TABLE OF CONTENTS

Page

LIST OF FIGURES

1 INTRODUCTION

The Language Data Repository (LDR) project can be viewed as an attempt to develop a computational system capable of modeling any human language. The proposal of such a system implies that human language can be view as an abstract concept, and that all human languages can be represented in terms of this abstract concept. This is a significant claim about the universality of human language systems (i.e., "Universal Grammar" (UG)). Despite the theoretical difficulty of substantiating UG claims, practical approaches to conducting field research would indicate that such a software system is plausible. Organizations such as SIL International (formerly the Summer Institute of Linguistics) train linguists to go to field locations in order to collect data from speakers of languages which have no established or standardized writing system, many of which have never been studied or documented before, for the purpose of learning and documenting the language. Such training programs prescribe types of data to collect and methods of analysis that presumably will allow the researcher to describe any spoken language (and perhaps non-verbal languages such as sign languages, though the LDR will not initially be designed to document and assist in the analysis of such languages). The LDR project hopes to accomplish its objectives by analyzing the field methods that these organizations use and by developing a software architecture capable of representing the data collected by these researchers.

One question that arises from is "why should we design such a piece of software?" One might claim the potential relevance of the LDR to the concepts of UG as well as to other topics of linguistic theory merit the investigation of such a system. The central motivation behind the LDR project is to provide a practical tool that linguists can use to catalog language data collected from "unwritten" languages and to harness the power of modern computers to aid in the linguistic analysis of that data. Any benefits that the development of the LDR may have to theoretical linguistics is essentially a side effect of the development process, although in the long run, it is possible that these side effects may prove to be more useful than the system itself. Therefore, we are developing the LDR system for the practical

[1] This thesis follows the style and format of *Language*.

benefits it may have for those linguists engaged in field research and for the side benefits that it may offer to more general questions in linguistics.

Although the LDR system is being designed as a tool for the collection of linguistic data from "unwritten" languages, there is nothing inherently different about collecting data from unwritten languages and languages with established orthographies. It is anticipated that the LDR will have uses for the general needs of linguists conducting field research for which they will need to store and analyze data taken from languages which have writing systems. An example of this would be Norma Mendoza-Denton's research about Chicanas in a California High School (Mendoza-Denton 1997). It is expected that the system will have uses far beyond that of research involving unwritten languages.

1.1 Current Methods and the Benefits of Machine Readable Storage

The data from field research is currently collected as written transcripts on paper and on computer media, or as recorded data on video and audio cassettes and archived. Even for relatively small projects the volume of data that must be cataloged is tremendous. While this method of recording and cataloging data has proven effective, it is extremely tedious and there are at least two easily identifiable problems. First, any linguistic analysis requires a tremendous amount of "grunt work" to find the relevant data from among the collected data and it is impossible to analyze all the data by hand when developing and testing linguistic hypothesis. Secondly, it is often difficult to the point of being impossible for another linguist to access and analyze the raw data of a colleague's research. If a linguist can obtain physical access to the data, the process of gaining an understanding of the data is frequently prohibitive due to the necessary period of familiarization with the data and the cataloguing techniques employed on individual bases.

A software system such as LDR offers two main improvements over current methods of recording, cataloguing and archiving linguistic data. First, by utilizing a standardized machine readable storage, it will enable linguists to use a uniform set of computational tools to aid in linguistic analysis and in the testing and evaluation of linguistic hypotheses. Secondly, a standardized method of recording data in a machine readable format will

enhance the ability of linguists to document their research and share the knowledge they gained.

1.2 Related Software

There are a number of other software tools that fall into the category of "related software," many of which were developed by an organization called JAARS, which is closely affiliated with SIL. Most of these programs are developed for specific tasks such as speech analysis and morphological parsing. There are a few programs that have been developed for a purpose similar to that of LDR. Among these are the SIL/JAARS program LinguaLinks and its predecessor Shoebox, and a number of other less well known systems, many of them DOS based or simple database interfaces. SIL is currently working in conjunction with New Tribes Missions on another tool called FieldWorks that is intended to replace LinguaLinks. The first version of this program is expected to be released in November 2000. In my discussions with both field workers and members of the development teams for some of these programs, I have become convinced that there is no tool in existence today that meets the needs of the user community. Most field workers that I spoke with indicated that they preferred to use traditional collection and storage techniques because the benefits offered by the tools did not offset their complexity. I believe that LDR's support for collaborative research via its client-server features and its ability to adjust to the changing needs of the user community through the concept of plug-in tools provide substantial conceptual improvements over other systems.

2 METHODS

This year my research has emphasized refining the LDR concept, defining the requirements of the system and beginning to outline the computational implementations of those requirements. The project is currently about mid way through the proof of concept phase (cf. Section 3.1.1). Due to time constraints, I have limited the scope of the linguistic data that I will consider to words. In addition to providing the most reasonable place to start building the linguistic data class library from a computer science perspective, words are a natural starting point from a linguistic perspective since they are some of the first data that will be collected in the field. This allows for the exploration of the issues surrounding machine based representation of phonological data and data storage without introducing the complexities of syntax that would be associated with utterances or the complexities of rule based data that would be associated with morphemes.

My specific accomplishments this year include:

- Determining a viable research and development schedule.
- Describing the basic requirements of the system.
- Developing a format for the system's internal representation of written transcriptions of spoken data that corresponds to current paper-based transcription methods.
- Developing a preliminary data model for the LDR system and instantiating that model in an Oracle database.
- Developing preliminary object models for some of the data to be represented by the system.

2.1 Concept Development

The concept that evolved into the LDR began as a C structure that would represent all possible language data and store it in a flat file system. Due to the linguistic complexity involved, this design quickly proved inadequate. Preliminary considerations of the

requirements for the LDR resulted in a number of important concepts (e.g. the need to have allow for access to a central data server). During the first months of my formal research, these preliminary considerations quickly evolved into the current system concept. By late 1999, the system concept had moved from the initial thoughts scribbled on paper to a well thought out, stable concept that has since undergone only minor revisions. This concept describes a system that will be able to support the fundamental requirements for the system architecture and be flexible enough to adapt to the changing needs of the user community.

2.2 The Development Schedule

One of the first tasks that needed to be dealt with in developing the LDR system was to determine, from a very high level perspective, what needed to be accomplished in the development process and how the basic development tasks should be ordered. A four stage development schedule was produced to meet this need (cf. Section 3.1). This project schedule outlines the basic tasks to be accomplished and has helped to place my research into the context of the overall product development schedule. The project development is currently in Stage One.

2.3 Requirements

The first step in the LDR development process is to begin gathering the requirements for the system. Unfortunately, due to time constraints, I have not been able to document a complete listing of the requirements for the LDR system. I have, however, gained an understanding of the requirements necessary to begin the design process. Specifically, I have determined the needs for the general system structure, the representation of phonological data and some of the data storage requirements. The requirements and their motivations are described in the results section (cf. Section 3.3). As the system matures, a more formal documentation of the requirements will be necessary, but the requirements outlined here are intended to be sufficient, for the current scope of the project.

There are a number of aspects of the user community which will greatly increase the complexity of the requirements gathering process. Among these difficulties is the dynamic

nature of research methods. Active research results in continuous modifications and improvements to currently existing research methods. As problems and limitations with existing techniques are found, new techniques are explored to solve the problems and improve the research process. The system needs to account for this continual change in the user requirements. Another difficulty is the fact that there are a number of competing theories in linguistics. The development of the LDR system must be careful to be as theory neutral as possible, and when it is not possible, to provide a means for supporting as many different theories as possible. These difficulties are lessened by a the fact that the user community, especially certain subsets of the user community, has developed a fairly complete, well documented understanding of what is needed in the data collection stages and the supporting methodology. Decades of research in "unwritten" languages and teaching new members of the community have produced a well defined understanding of the activity that the LDR system will support and enhanced the ability of the user community to clearly communicate what the system needs to accomplish.

The requirements gathering process will draw heavily from both the academic side of linguistic knowledge and research and from the practical side of field work. In developing the requirements outlined below, I relied on linguistic textbooks, interviews (both formal and informal) with various professors, members of SIL, and on my own experiences with field research in Papua New Guinea. I also began to describe the research process using integrated definition (IDEF0) modeling techniques. IDEF0 is a modeling technique that represents activities and the inputs, controls outputs and mechanisms involved in that activity. IDEFO also provides a useful tool for modeling functional requirements of complex systems which facilitates communication with the user community. These models were helpful in the early stages of the research and served as a guide for my interviews with Bob Hauser, a member of SIL. Due to time, equipment constraints, and lack of access to individuals who have been trained in the particular areas of research I was modeling, I did not continue the development of this model. While the original model was very helpful in the preliminary stages, its continued usefulness is questionable since the system architecture is not directly manipulated by the user. IDEF0 modeling may prove more useful in the

development of the analytical tools hosted by the architecture. IDEF1x modeling may also prove useful in developing and refining the system databases, particularly if used in conjunction with IDEF0 modeling. Unfortunately, the time required for this modeling process plus the cost of development tools is prohibitive given the currently available resources.

3 RESULTS

My work this on this project has been very successful and I have developed a stable concept of the system, documented the fundamental requirements and begun to explore some of the design issues for implementing the user requirements. From the design aspect of my research, my work in developing a method of representing phonological characters has yielded the most tangible results. I have produced a character (the `PhonChar` class) for the system specific representation of phonological data that satisfies the user requirements mentioned below (cf. section 3.3.2) and have outlined a very preliminary character for the representation of distinctive features as used to identify and describe different phonetic segments (cf. Appendix A). A prototype `PhonChar`[2] class and a corresponding `PhonString` class have been implemented. Preliminary tests using a very basic mapping of `PhonChar` character values to Unicode characters indicate that this class is functioning as intended and will satisfy the needs of the system. My work in developing an overall object model has proceeded somewhat more slowly, due to the prioritization of completing the phonological representation of the data strings and my lack of experience with object oriented design. Recently, however, I have had some success in developing a conceptual model for many of the core areas of the system including a class library (i.e. a software template for storing and acting upon data.) to be extended by persistent objects, a structure for objects that will need exist in a distributed environment, and data models for some of the data to be stored by the system. Unfortunately, at the present time, I have not been able to implement and test these data models. This stage will be left for future research and development.

3.1 The Development Schedule

This schedule lays out a general path for the development of the LDR. Due to the present impossibility of predicting the amount of time I or others will be able to devote to the project in the future, it is unrealistic to set this schedule into a specific time frame.

[2] I will use the convention of displaying the names of classes and objects in a fixed width font and will distinguish objects and classes by the fact that a class name will *have an initial* capital letter whereas an object will not.

Instead it is indented to place the current research in the context of the entire project development cycle.

3.1.1 Stage One: Requirements Gathering and Proof of Concept

In this initial stage of the LDR development, the emphasis will be on gaining a better understanding of the problem area, the significant technical challenges that we will encounter (e.g. the need for revision control systems, rule based data representation etc.), and the requirements for successfully solving the problem. Key objectives of this stage include:

- Development of a stable set of user requirements.
- Demonstration of the ability to represent and store linguistic data.
- Demonstration of the ability to dynamically add tools to the architecture (specifically tools for data collection).

The ability to represent and store data and to develop and incorporate tools that act upon the collected data form the core of the LDR system. Before it is reasonable to begin work on other areas of the system, we must first demonstrate that the system will be able to perform these core tasks. Major deliverables at the end of this stage will include requirements and specification documents, a database design, a basic user interface specification, and a class library for the storage of linguistic data objects.. Prototyping and modeling techniques will be used heavily during this phase of the system design to facilitate the interaction of system developers and members of the user community in order to help define the user's requirements.

3.1.2 Stage Two: Demonstration and Validation

The goal of Stage Two is to demonstrate the ability of the LDR architecture to solve all major technical challenges facing the system. At the end of Stage Two, it should be apparent that LDR will meet the objectives set forth in the user's requirements document. Key objectives of this stage include:

- Demonstration of the client-server architecture.
- Demonstration of the multi-user/multi-language features including user profiling.
- Finalization of the database and system manager.

The second stage of the project does not attempt to find the optimal solution for all of the technical challenges, but instead seeks to demonstrate that all of the user requirements can be solved within the LDR framework. Stage Two also provides the opportunity to verify that the user's requirement are accurately reflected in the requirements documents and are understood by the system developers. The major deliverable at the end of Stage Two is a reasonable alpha prototype of the LDR system. This prototype does not need to be bug free or even to fully meet the user's requirements, but it should satisfy all major requirements.

3.1.3 Stage Three: System Optimization and Finalization

In the third stage, the algorithms are optimized for efficiency and maintainability, known bugs are fixed, and a fully functional beta version is developed and released for field testing. The results of the beta tests will then be used to prepare the production version of the system for release. Key objectives of this stage include:

- Optimization of client-server architecture, and user interface.
- Demonstration that the system is sufficient for scientific documentation of linguistic research.
- Demonstration that the system fully meets all of the user requirements.
- Finalize user and developer documentation.

In this phase, the algorithms will be reviewed to ensure optimal performance, the code will complete a final test and evaluation process and the documentation will be finalized. in preparation its first public release.

3.1.4 Stage Four: System Release and maintenance

Stage Four involves setting up the central data repository and distributing the LDR software to field workers via the Internet under the terms of the GNU public license. This stage will allow for the distribution of two versions of the system, one, intended to be used in distributed environments, will be distributed in the form of a client module, a server module, and a database module. This distribution will require slightly more expertise to set up and will need more thorough installation documentation. The other module, intended to be used in non-distributed environments, will have the client, server and database modules integrated into one system. Also included in Stage Four is user support and software maintenance.

3.2 The System Concept

The LDR system, as portrayed in Figure 1, is divided into two major components, the system architecture and plug-in tools. The architecture will be developed to provide a representation of language data and to host the tools. Tools will provide the primary functionality of the system by manipulating the data at the request of the user.

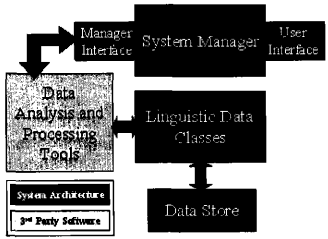


Figure 1: The LDR System Concept

The architecture is built around a set of classes designed to represent "spoken" linguistic data. When instantiated, these objects create a machine readable representation of "spoken" linguistic data. For long term data storage, these objects are stored in the system data store (cf. Section 3.4.4). The data store is the abstract designation for the method by which the system will maintain persistent objects. Conceptually the data store may be implemented using a database management system (DBMS) or any other data storage method. The system manager and the data class library

will be designed without specific design considerations being given to the underlying implementation of the data store. This abstraction of the implementation of the data storage allows for the design of a system that can be easily adapted to use a number of different data store implementations (e.g. existing linguistic corpora). Given the under-specification of the data store implementation it is inaccurate to refer to it as a database.

Data analysis and processing tools provide the functionality of the LDR system by manipulating the data stored in the database through the use of the linguistic data objects. These tools, developed to work with the LDR system manager, can be loaded into the system at any time. This allows these tools to be developed by third party software development teams; teams which may have much more expertise in particular areas of computational linguistics than the original development team. Once loaded into to the system manager, users may then interact with these tools through the user interface provided by the system manager. By providing access to the user interface through the system manager, the LDR system can provide a standardized look and feel across tools developed by many individual development teams, thereby greatly reducing the learning curve for each new tool.

3.2.1 The Linguistic Data Classes

The linguistic data classes, as mentioned above, allow for a persistent, machine readable representation of language data. They provide a method of representing data elements collected from speakers of the target language (e.g. individual words, utterances and phrases, dialogues, monologues and other data), hypothesized rule based data as determined through linguistic analysis (e.g. phonemes and morphemes), and the people involved in the research process and ethnolinguistic information about these individuals (e.g. the age, gender, and social role of both the person who collects the data and the person who provides the data). These classes form the heart of the LDR.

Since the LDR will be maximally useful if it allows its users to represent all, or at the very least, the vast majority of the data they collect, the design of this element of the architecture will be of the utmost importance. If the implementation of the data classes omits a critical

piece of data that a user needs to collect, then the system fails to be as useful as intended. On the other hand, a well designed, complete implementation will allow for the development of data analysis tools that may far exceed the vision of the original design team.

Given the tremendous amount of data that may need to be stored by the system, the efficiency of the these classes is not an inconsequential issue. The reality of fieldwork is that most computers used in the field are not top of the line mainframes, but often slower than ideal laptops. Given the available hardware and data collection situation, the efficiency of data storage and retrieval must be a very high priority of the system.

3.2.2 The System Manager

The system manager provides a means for loading the plug-in tools into the system, providing widgets to aid in the display and entry of linguistic data, and maintaining a common look and feel for the user interfaces implemented by the various tools. It is also responsible for user identification, for reading the initialization and user preferences files, and for establishing a connection to the data server when the system is being operated in a distributed environment.

3.2.3 The Plug-in Tools

The functionality of the LDR system comes from the ability of the architecture to host tools developed to conform to the interface standards of the architecture. Some of these tools may be developed by the architecture development team, however, by developing a system that can support tools developed by third parties, the LDR system will be able to provide computational support that the original development team does not develop. As time and use reveal deficiencies in the original architecture, new releases can improve the architecture without sacrificing the viability of the tools that operate on that architecture. This ensures that the LDR system will be flexible enough to continue to meet the changing requirements of the user community long into the future.

3.2.4 Support for Collaborative Research

As technology advances, networked computer systems are beginning to appear, even in remote field locations. While most field sites do not have access to the Internet or wide area nets (WANs), local area nets (LANs) are beginning to appear in areas where research teams are co-located. It is reasonable to assume that in the near future even the most remote field sites will have access the internet. In order to support collaborative research either over LANs or the Internet, the system will be developed with a client-server architecture. The current plans call for the system manager and the analysis tools to form the client portion of the architecture. The data classes and the database form the server portion. This division will be evaluated for possible efficiency improvements as the system is developed. For environments in which a distributed computing is not practical, the system will provide a non-distributed implementation.

3.3 Requirements

The requirements listed here represent needs of the initial proof of concept prototype system. While insufficient for the design of the final LDR, they allow for the development of a prototype capable of demonstrating the many of the features of the final LDR. The requirements have been divided into three major groups, the system architecture requirements, the phonological representation requirements and the data storage requirements. The system architecture requirements describe the over all structure of the system. The phonological representation requirements describe the needs for the representation of phonological data from a users perspective. The data element requirements out line the data representation needs of the prototype LDR.

3.3.1 System Requirements

The core of the proposed LDR system, the system architecture, provides the structure for collecting language data (linguistic data objects) and a platform (system manager) from which other software tools may operate on the collected data to aid in the analysis process. Three general requirements outline the core functionality to be provided by the architecture :

- The system must support many different users, working together and independently, both located centrally and disparately.
- The system must make the work of these separate users available to other users while accurately attributing the work to the individual researchers who collected and catalogued the data.
- The system must allow users to record data collected from any human language.

Linguistic field research involves both solitary researchers as well as teams of linguists collaborating on a single research project. The individuals on the teams may be working in close proximity or widely separated by either time, distance or both. For the LDR system to be successful it must support team research as well as the work of individual researchers working alone. In addition to the flexibility of the LDR, the ability to make the work of a researcher working in the field available to a linguist working in a university or industrial setting will be of significant benefit to the linguistic community as a whole. While the technology to connect remote field sites and academic or industrial centers through the Internet or other networking protocols is not currently available, it is feasible to take the data and results collected and developed in the field and store them in an area accessible to a network. It is also very easy to imagine a day when even the most remote areas of the world will be "wired" through the use of radio, satellite, or some other technology. Accordingly, the architecture will be developed so that any client can access any data server provided that the client can identify the intended data server on the network.

A user will interact with the LDR architecture through the use of "plug-in" tools that operate on the data represented by the architecture. These tools will be integrated into the architecture at runtime through the use of a system manager that will enforce a common user interface across the tools. The system manager will also provide other support necessary for the integration of tools and the display of linguistic data. In summary, the LDR provides three basic components, a means of storing linguistic data, tools to operate on that data, and a system manager to support the integration of these tools.

3.3.2 Phonological Representation Requirements

Representing phonological data in a written format consists of transcribing the linguistic data phonetically and phonemically using a standard phonetic symbol set, such as the International Phonetic Alphabet (IPA cf. Appendix C). For linguistic codes with established writing systems or for situations in which the field worker will develop a writing system, the data can also be stored in an orthographic form. Phonetic and phonemic representations of data will be referred to as transcriptions. Two over arching requirements govern the development of the phonological aspects of the LDR system. They are:

- The LDR data representation features will provide a method for representing phonetic, phonemic and orthographic data in both machine readable and human readable form.
- The LDR system will support transcription methods analogous to current paper based transcription methods. This will involve the ability to enter phonological characters directly from the keyboard, and to modify the mapping of phonological characters to key-stroke sequences. Additionally, the system will retain information to map phonological characters to information regarding place and manner of articulation and distinctive features (DF). This mapping will be modifiable by the user.

The ease of entry of phonological data is critical to the success of the LDR system. While it is unlikely that machine storage will ever support data entry that is as easy or flexible as paper based methods of transcription, if the system fails to at least approximate paper based methods, the system will be unusable because users cannot afford to take the time to enter the data.

In addition to these written transcriptions, audio and video data storage provides an added understanding of the data element. The LDR data representation features will provide a method for capturing these data in both machine readable and human readable formats.

The system will support the representation of four basic sets of information for the phonological transcription of words (Clark and Yallop 1990; Roca and Johnson 1999; Akmajian, et al. 1998):

- The ordering of phones.
- The division of those phones into syllables (to include information about the role of phones as onset, nucleus, and coda).
- The tone associated with those syllables (rising, falling, level-low, level-mid, level-high, rising-falling, falling-rising, or no tone).
- The stress associated with those syllables (primary, secondary, tertiary or no stress).

Since not all phonological data entered into the system will contain each of these elements, the data storage method will be capable of representing phones and any, all, or none of the other elements depending upon the data entered by the user.

The LDR will support the development of an orthography by providing the user with a means of graphically defining the characters of the orthography, mapping those characters to the keyboard, and relating those characters to the associated phonemes.

In addition to allowing users to define an orthography, the system will provide means of updating currently stored data in the event that the orthography is changed (most likely this will occur when the user decides to use a different orthographic symbol to represent a particular phone or phoneme).

3.3.3 Data Storage Requirements

The system will provide a means for storing spoken data collected from any human language to include the following data elements (cf. Akmajian, et al. 1998):

- words
- utterances
- dialogues
- monologues
- phonemes, phonological rules, and phonotactic constraints
- morphemes, morphophonological rules and morphotactic constraints

In addition to spoken language data, the system will provide a means for storing ethnographic information to include the following (cf. Bernard 1995):

- people who collect or provide language data
- languages spoken by people
- ethnolinguistic groups to which people belong and the roles they play within that group

Each of these basic categories of data is explained in more detail below.

3.3.3.1 Data Elements

The most abstract type of data represented by the system is the data element itself. A data element is any utterance from a language that is elicited from a speaker of the language (i.e. an informant) or a theoretical construct based on that data (e.g. an underlying form or a phoneme). Every data element will be uniquely identifiable by the system and will provide a means for identifying it with the language from which it was collected and with its associated ethnolinguistic information. Additionally, every data element will be capable of saving itself to a data store, restoring itself from the data store, and removing itself from the data store. Every type of data element will provide specific, documented methods for modifying the data represented by that element. The system will also implement a revision control system to allow for tracking of changes made to any particular data element. This revision control system promotes the scientific documentation of data and helps to prevent erroneous assumptions on the part of one linguist from corrupting the original and unmodified data.

This requirement as described herein, is presently underspecified. Notably the information that is entailed by ethnolinguistic data needs to be further specified. Again, current time constraints prevent the development of a stable set of requirements for ethnolinguistic data at this time.

3.3.3.2 Words

The LDR will provide a means for representing words, defined as "the smallest free forms found in language" (O'Grady 1997). These words will extend the requirements of the data element and will be composed of a set of surface forms or phonetic transcriptions, a set of underlying forms or phonemic transcriptions, and a set of orthographic transcriptions. The system will allow the storage of multiple phonological transcriptions of data elements collected by multiple linguists from multiple informants on multiple occasions. It will also allow the association of surface and underlying forms for accurately modeling linguistic data and for supporting the analytical process. This association will support a one to many mapping of both underlying forms to surface forms as well as a one to many mapping of surface forms to underlying forms. The latter of these two mappings is a significant break from the traditional view of the relationship between surface and underlying forms of a word (cf. Goldsmith 1990; Jakobson and Halle 1956; and Kenstowicz 1994) and is discussed in more detail below. A word will also support the storage of:

- semantic information related to the word which includes multiple definitions and glosses of that word into another language
- lexical information related to the word which includes syntagmatic functions and other taxonomic information

The exact nature of the semantic and lexical data to be supported by the system has yet to be determined, but will be determined during the final design and implementation process.

In order to support the transcription and analysis of phonological data in the field situation, a "word" will support the representation of many different surface and underlying

forms and the relative mapping and association of these forms to other linguistic data related to the word. The need for multiple surface forms may be generated by a number of factors which include the suppositions that:

- A single linguist will likely transcribe a word multiple times before coming to a conclusion about which is the most appropriate transcription.
- Multiple collaborating linguists collecting the same data element may hear that data element differently or may transcribe the same utterance differently.
- The data element may actually have different phonetic and phonological forms depending on the ethnolinguistic background of the informant or the ethnolinguistic situation.

The need for multiple underlying forms stems from the fact that the system is intended to support the analysis of linguistic data, not merely the modeling of it. As mentioned above, the association between surface and underlying forms needs to mappable both from underlying to surface forms and from surface forms to underlying forms. The LDR system attempts to support the approach of first gaining an understanding of the possible underlying form from an analysis of the surface form. During the analysis, it is likely that multiple underlying forms will be posited for any given surface representation of the word. Over the course of analysis and through the collection of more data, a more satisfactory underlying form will be determined and the others discarded. Ideally, this hypothesized underlying form will be agreed upon by the linguists involved in collecting the particular data element. In this final underlying form, the data element could be represented as having multiple surface realizations of one underlying form. This single to many association eliminates the need for links from the surface forms to the underlying form. However, since the system must support the intermediate stages as well as the ideal stages of phonological analysis, and since the world is far from ideal, the links from the surface to the underlying forms are necessary.

3.3.3.3 People

The system will provide a means of representing the people involved in the linguistic research process. Each person may have a first, middle, and last name, a title, a full name, an age, and a gender. The system will allow every person to be associated with the languages that the person speaks natively and non-natively, with the ethnolinguistic groups that they are members of, and with the ethnolinguistic roles which they inhabit in those groups. Each person may be associated with a number of different addresses which can be specified by a mailing address or physical location. The relationship between the person and the address will specify the type of address (e.g. mailing, field address, permanent address etc.). Each person will be further described as being either an informant, i.e., the person from whom language data is collected, or as being a user of the system, i.e., someone who uses the system to either to support field work or to access data collected by another user..

In addition to the attributes common to each person, each user will also be described by a login name and a password. Each user will be associated with the languages and the ethnolinguistic groups that the user studies, the language data that user has collected, and the theoretical data that the user has proposed. The system will also maintain information regarding any linguistic organizations with which that user is affiliated.

In addition to the attributes common to each person, each informant will also be associated with a place of residence and associated with the data collected from that informant, as well as the ethnolinguistic circumstances of the data collection situation. The system will also maintain information associating users and the informants with whom they work. This relationship will contain information about the alias by which the user identifies the informant and the length of time the user has worked with the informant.

3.3.3.4 Languages

The system will provide a means of describing various language groups. Each group will have a name, a description, and the information regarding the location of the speakers of the language group and the approximate number of speakers.

3.3.3.5 Ethnolinguistic Groups

For each language group the system will provide a means of representing and cataloguing the various ethnolinguistic groups of that language. Each group will be described by a name unique to that language and have a textual description for the documenting additional ethnographic details. For each group, the system will provide a means of representing the various roles that may be played by members of that group (e.g. chief, mid-wife, etc.). Similar to the ethnolinguistic group, the role inhabited by the individual members of the group will be defined by a name unique to that group and have a textual description for the documenting of additional ethnographic details.

3.4 Data Representation[3]

3.4.1 Phonological Representation

The LDR system defines two system specific characters: a `PhonChar` character to describe the LDR system's internal phonological alphabet, and a `DFChar` character to allow for the representation of distinctive features for each phone. The specification for the distinctive features comprising this character has been derived from Lass 1984 and is detailed in Appendix A. Due to the large size of the `DFChar`, data is not stored in this format, but rather as `PhonChar` characters. These smaller characters may be converted into `DFChar` characters as necessary for data analysis. The PhonChar character type will be used for the representation of phonological data by the LDR.

3.4.1.1 The Phonological Character Bit-Field

The `PhonChar` character serves as the system's primary format for the storage of phonological data. This system specific character allows for the representation of both phones and super-segmental features of phonological data (i.e. syllable structure, stress and tone). This method of representing phonological data is necessary due to the limits of

[3] I use the term "object model" to refer to a model of a class (or small set of classes) and the objects that class instantiates. I use the term "class library" to refer to a model of a collections of classes and the relationships between them. Thus the persistence library and personnel classes sections provide class libraries and the word and transcription classes sections provide object models

existing character sets. The ASCII character set provides no support for phonological characters. The Unicode character set, while it does allow for the representation of characters, does not provide for representation of super-segmental information.

Since it is likely that the super-segmental data would not be stored for the majority of characters, it may be possible to reduce this character to a one byte, ASCII-like character by encoding the super-segmental components of the data as separate characters. While this method of representing phonological data would allow for a much more compact storage of data (nearly half the space in the ideal case), it has two drawbacks: First, by allowing only two hundred and fifty-five phones, diacritics, and super-segmental data, it would allow the user very little flexibility in defining new characters. Second, the need to determine whether a character is super-segmental or segmental would require a significant amount of added complexity. Due to the need for processing large volumes of phonological data during linguistic analysis, preliminary analysis suggests that this added complexity would offset the benefits of more compact storage. The efficiency considerations will be more thoroughly investigated at a later development stage.

The tools will be responsible for the display of phonological characters and for the translation of characters into appropriate `phonChar` and `phonString` objects. To aid in this process, the LDR system will provide a character map to translate Unicode characters into their appropriate phonChar counterparts. When the tool receives a Unicode input character from the user interface, or passes a `PhonChar` character to the interface, that character will pass through a PhonChar character map. This character map associates the system specific PhonChar character to a more general Unicode character representation that can be displayed by the user interface. This character map will be modifiable by the user. Users will be able to select a Unicode character to be displayed for each `PhonChar` character. This enables users to employ the phonetic alphabet that most suits their preferences and experience (e.g. the American Standard alphabet rather than IPA). In addition to this character map, future versions of the system may implement a string map to define a standard format for Unicode strings representing super-segmental information and to map those Unicode strings to `PhonString` objects.

Figure 2 shows the bitwise representation of the PhonChar character. The nine bits, a ... i, allow for the assignment of 512 different phones to PhonChar. The assignment of characters to these values is based primarily on manner of articulation and secondarily on place of articulation. The full specification for the character assignment is found in Appendix B. The five most significant bits are reserved for the storage of stress (str), tone (tn) and syllable feet (syl ft) information. Since the stress and tone fields are intended to have values only for syllable boundaries, the stress and syllable foot fields can overlap. The LDR architecture define a constant for each str, tn and syl_ft value as shown in hexadecimal notation in Figure 2.

PhonChar

str/ft			tn				a	b	c	d	e	f	g	h	i	

tn

			name	value
0	0	0	nil	0x0000
0	0	1	rising	0x0800
0	1	0	falling	0x1000
0	1	1	rising-falling	0x1800
1	0	0	falling-rising	0x2000
1	0	1	level-low	0x2800
1	1	0	level-mid	0x3000
1	1	1	level-high	0x3800

str/ft

0	0	nil	0x0000
0	1	primary	0x4000
1	0	secondary	0x8000
1	1	tertiary	0xC000

0	0	nil	0x0000
0	1	onset	0x4000
1	0	nucleus	0x8000
1	1	coda	0xC000

Figure 2: The PhonChar Bit-Field

3.4.2 The Persistence Class Library

The persistence class library to be implemented by the system is a modified version of the example library developed by Reese (1997) and distributed by the Centers for Imaginary Environments or CIE (http://www.imaginary.com). The CIE supports several open source software development projects. By extending this library, the system hides the implementation of the database specific features of the data classes behind several layers of abstraction. This feature of the system provides a number of benefits including the following:

- Changes made to either the database or to data classes require only a minimal subset of the code to be changed
- Additional classes can be added to the system to allow for the use of alternative data storage implementations including other existing data collection programs
- Data elements can be developed at an abstract level without a detailed consideration of the structure of the database greatly easing the development and design process

Due to the importance of the efficiency and robustness issues surrounding the development of data classes, this element of the design introduces one of the greatest risk factors into the system development process. The ability of this design element to meet these requirements is critical to the system. However, effective testing of this element will require that a large corpus containing all types of data represented by the system be available. This requirement for testing is not practical until the beta test phase; a phase far too late in the process to be making the changes that would enable correcting problems in this design. Since major design flaws in this area at the beta test stage will result in a failure of the project, steps must be taken now to identify potential problems and to redesign the persistence library should it not meet system requirements. These steps will include a rigorous unit testing with large volumes of simulated and actual data and a detailed analysis and optimization of the efficiency of this design.

Figure 3 represents the currently proposed version of the class library as it is extended to provide support for data elements in a distributed environment. In a version of the system intended for a non-distributed environment, the remote interfaces would be omitted and the server-side classes renamed appropriately. Below is a brief discussion of the major components of the class library in the context of this diagram. For a more extensive description of this design please refer to Reese (1997).

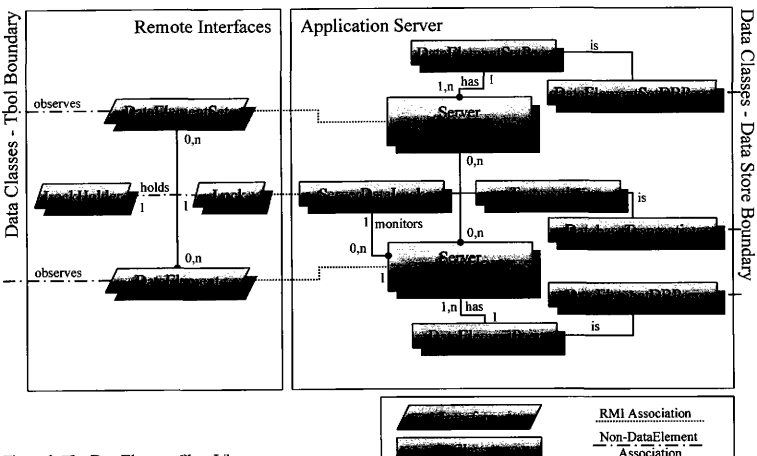


Figure 3: The DataElement Class Library

3.4.2.1 The `ServerDataElement` Class

The `ServerDataElement` class is the abstract class that all language data classes (e.g. `Word`, `SurfaceForm`, `UnderlyingForm` etc.) will extend. This class implements the identification and some revision control features for every subclass. Each subclass is responsible for maintaining the data for which that class is designed to represent. Subclasses will allow for the modification of the data represented by the class. These modifications will be allowed only if they are attempted by an object that implements `RemoteLockHolder` and that the `RemoteLockHolder` has a lock on the object that it is trying to modify or that the object is unlocked. Significant public methods of this class are:

- **`static public ServerDataElement getServerDataElement`** – this method returns a ServerDataElement when provided with a transaction and either the data to be used in the creation of the object or the id of an object to retrieve from the database.
- **`public synchronized void remove`** – flags the object to be removed from the data store provided that an appropriate lock is held on the object.
- **`public void restore`** – this method restores the object from the data store using the value of the id field when provided with a transaction

This class has a corresponding interface for use in a distributed environment and a corresponding "set" class to handle sets of ServerDataElement objects. The "set" class also has an interface for distributed computing.

3.4.2.2 The Peer Classes

The Peer classes implement methods for the insertion of data into the data store, the removal of data from the data store, the restoration of data from the data store and the updating of the data store with modified data. The DBPeer, or database peer classes provide the code to that actually interfaces with the database. To allow the system to interact with a data store other than the database, or with multiple databases designs, the peer class for each data store needs to be built and the other classes needs to be designed to allow for the

specification of which data store to use. The peer class would then select an appropriate data store specific peer class to provide the means for the actual modification of the data store.

3.4.2.3 The `Transaction` Class

A transaction object manages the actual process of writing data to the data store. This class allows multiple data elements to be updated concurrently or not at all. This prevents errors that occur part of the way through the process of saving changes to the database from corrupting the original database. It provides the public methods to abort, commit, restore, and save. The `Transaction` class has corresponding classes for each of the data stores that the system will use.

3.4.2.4 The `RemoteLockHolder` Interface

This interface is implemented by any client tool that needs to modify data elements and provides a single public abstract class, `setLock`. Any client that wishes to modify an object will pass itself to that object in order to obtain a lock or verify that it holds an existing lock. An object implementing `RemoteLockHolder` can call the `save()` method of a lock that it holds to trigger its Transaction to save the data associated with that transaction.

3.4.3 Data Element Object Models

As I have indicated, the focus of my work in terms of data storage centers around the identification of the data pertaining to the original collection and analysis of words. The requirements for the system state that a word will be represented by surface, underlying and orthographic forms. The word object will also need to maintain ethnolinguistic information concerning the environment from which it was collected. While I have not engaged in the analysis of the requirements for ethnolinguistic data, my research to date indicates that the personnel data described below, in Section 3.4.3.1, will serve to capture a large percentage of the data that will need to be represented. Below, in Section 3.4.3.1- Section 3.4.3.3, I describe preliminary object models for the representation of words, surface forms,

underlying forms, and personnel data. Additional details of these objects will need to be worked out during the future stages of this project.

3.4.3.1 The Personnel Classes

Representing the people involved in the collection and analysis of linguistic data is critical for three reasons:

- Properly attributing linguists for their work.
- Tracking ethnolinguistic data.
- Identifying informants who may be providing exceptionally good information or exceptionally poor information.

Figure 4 displays a class library for the collection of personnel data relevant to linguistic field research. At the core of this library is the `Person` class. This class is extended by the `User` and `Informant` classes. Every person will be uniquely identifiable by a 4 byte integer assigned by the system and will have attributes such as name, age, gender. The `User` class extends `Person`, adding attributes for a login name and a password. Users will also maintain a have an attribute for storing the addresses at which they can be contacted. The `Informant` class extends the `Person` and adds attributes for the representation of a physical address. Each user also instantiates an `InformantSet` object describing the informants that user works with. Similarly, each informant instantiates the `UserSet` class to describe the users that informant has worked with.

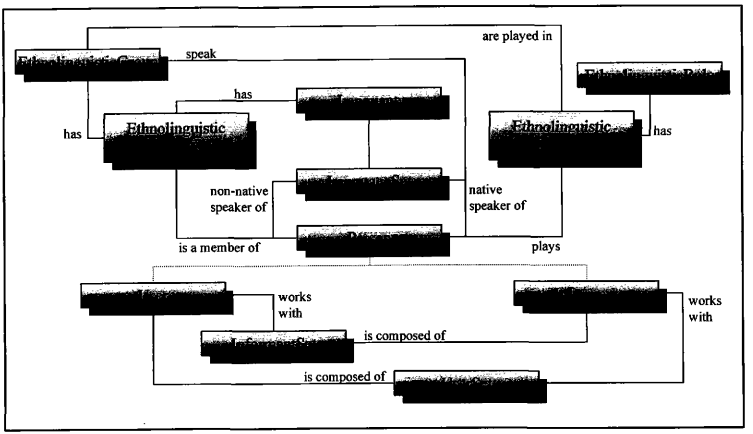
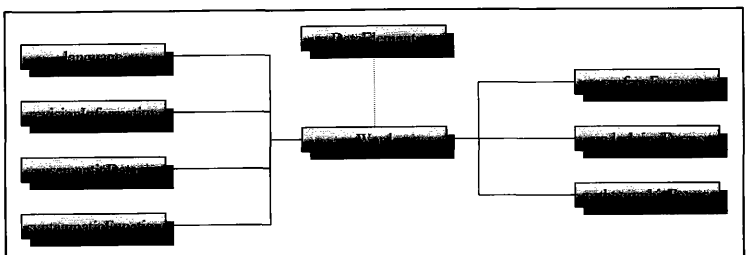


Figure 4: The Personnel Class Library

In addition to providing a way to model people, this library also provides a way to model ethnolinguistic groupings of those people. Each person has attributes describing the sets of languages they speak as a native speaker and as a non-native speaker, the ethnolinguistic groups they are a member of, and the ethnolinguistic roles they play in those groups. Each `Language` is described by a name, and has attributes for a brief description of the language, the number of speakers of the language, and the geographical location of the speakers of the language. Each language also describes a set ethnolinguistic groups in which members participate. Like the `Language` class, the `EthnolinguisticGroup` class is described by a name, and a brief description. It also describes a set of languages commonly spoken by the members of the group, and the set ethnolinguistic roles that are inhabited by various members of the group.

3.4.3.2 The Word Class

The `Word` class extends the `DataElement` class, as described in Section 3.4.2. The object model in Figure 5 represents the objects instantiated by a `Word` object. This model is presented from the perspective of a server side `Word` object extending the ServerDataElement class. The other classes described in the `DataElement` class library will need to be extended by creating `WordSet`, `WordPeer`, `WordSetPeer`, `WordDBPeer`, and `WordSetDBPeer` classes. In a distributed environment, appropriate RMI interfaces will also be needed. Furthermore, the naming conventions corresponding to those presented in the `DataElement` class library will be adopted over the simplified names. The `surfaceForms`, `underlyingForms` and `orthographicForms` objects are instantiations of the `SurfaceFormSet`, `UnerlyingFormSet` and `OrthographicFormSet` classes respectively. These classes provide a means of storing collections of their respective transcriptions (cf. Section 3.4.3.3, below) and a means of relating the transcriptions to each other. Please refer to the requirements section, Section 3.3.3.2, for a more complete discussion of the transcriptions and the relations that exist between them. The word also maintains a copy of the `language` object representing the language the word is from, the `semanticData` pertinent to the word and `syntagmaticFunctions` that the word can perform (e.g. noun, verb, adjective, etc.).



Attributes

public RevisionInformation revisionInformation;

protected HashTable surfaceForms;
protected HashTable surfaceForms;
protected OrthographicForm orthographicForm;

protected Language language;
protected SemanticData semanticData
protected SyntagFunctionSet syntagFunctions;

Methods

public save()
public restore()
public remove()

public void addSurfaceForm(SurfaceForm s)
public void addUnderlyingForm(UnderlyingForm u)
public void setOrthographicForm(OrthographicFrom o)
public void setLanguage(Language l)
public void addSyntagFunction(SyntagFunction s)

Methods

public void removeSurfaceForm(long id)
public void removeUnderlyingForm(long id)
public void removeSyntagFunction(String name)
public SurfaceFormSet getSurfaceForms()
public UnderlyingFormSet getUnderlyingForms()
public OrhtographicForm getOrthographicForm()
public Language getLanguage()
public Boolean fromLanguage(Language l)

Attribute ——— Inheritence .....

Figure 5: The Word Object Model

3.4.3.3 The Transcription Classes

The transcriptions classes serve to represent the transcripts of actual spoken or written data (a written format is included to support the development of an orthography for a particular language, and to facilitate research among languages that do have a standardized written representation). This group of classes is comprised of the `SurfaceForm`, `UnderlyingForm` and `OrthographicForm` classes. As shown in figure 6 they, like the `Word` class, extend the `DataElement` class. The first division of this group of data distinguishes the phonological and orthographic transcriptions

3.4.3.3.1 The OrthographicForm

This class differs significantly phonological classes in that it needs no special type to represent the data but can instead use a standard string object. In addition to extending the functionality of the of the `DataElement` this class provides a method for associating the orthographic form with the linguist who proposed that form. This is intended to help identify the origin of conflicting orthographic forms of a single word arising from different analysis results, dialectical variations in orthography (such as can be found in classical Greek) and other sources. Future designs will likely include a link to an `OrhtographicSystem` and a link to the phonological form from which they were derived.

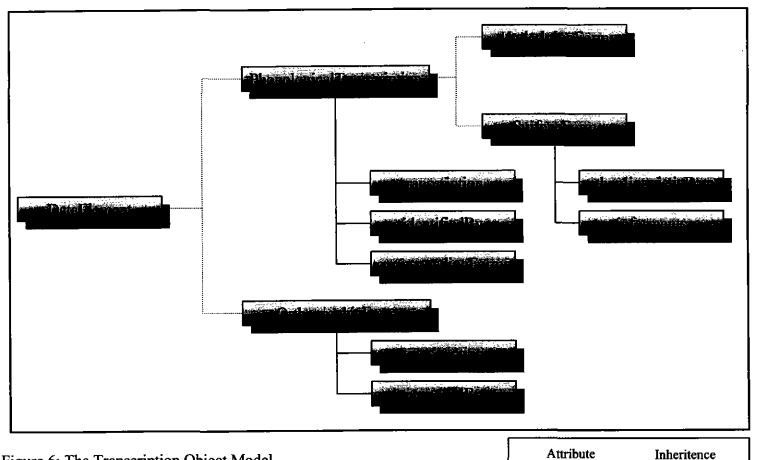


Figure 6: The Transcription Object Model

3.4.3.3.2 The Phonological Transcriptions

The major characteristic of the phonological characters is that they represent their transcription attribute as a PhonChar object, providing a means of representing a string of phones and the associated stress, tone and syllable structure. The PhonologicalCharacter, which serves as the base class for both SurfaceForm and UnderlyingForm transcriptions, maintains the PhonString transcription object, the UserSet identifiedBy object, and the PhonologicalTranscriptionSet correspondingForms object. The identifiedBy attribute is the set of linguists who participated in eliciting the data (in the case of surface forms) or proposed the data (in the case of underlying forms). The correspondingForms provides a means for associating surface forms with the underlying form from which they are derived, and vice versa (cf. section 3.3.3.2).

The SurfaceForm class extends the base class by providing for the representation of the informants (an instantiation of the InformantSet class) who provided that particular surface form. Tracking the informants from whom data is elicited provides a first step toward collecting the ethnolinguistic data needed to inform linguistic analysis. Other ethnolinguistic information will be represented by the ethnolinguisticData attribute (an instantiation of the EthnolinguistData class). The requirements for this data, which will be based largely on Bernard (1995), have yet to be determined but will likely include information such as the place in which the data was collected and the time period in which the data was collected.

The UnderlyingForm class does not currently implement any functionality not present in the PhonologicalTranscription class. I am currently working to determine what, if any, extra data needs to be modeled by this class. If there is none, I will remove this class from the design and rename the PhonologicalTranscription class to indicate that it provides the primary representation for transcriptions of phonemic data.

3.4.4 The Data Store

The system will use a database management system to provide the primary method for storing data (the data store) for the persistent objects. The implementation of the database is the lowest level component of the architecture and will be hidden from both the final users of the system and from the tools developed for the system. Since the tools will access the stored language data through the use of the linguistic data classes, there is no need for the tool developers to be aware of the structure of the database, or even of its existence. The database design must also take into consideration the issues of data storage completeness introduced in Section 2.1.3.1, Linguistic Data Classes.

The present research project plan foresees the release version of the software system to use an object-relational database management system, PostgreSQL. The motivation for choosing this database over other more well known database systems is largely due to cost considerations. Only in extremely rare circumstances can field researchers afford a commercial system such as Oracle. I believe that the cost of selecting a less well known DBMS is offset by the fact that PostgreSQL can be freely distributed with the system (cf. Appendix E). By choosing this DBMS, we are able to keep the cost of the final system to a minimal amount while making only minor sacrifices in usability. Future releases of the LDR may implement the data store using another DMBS, or other technology. This will allow the user to choose the type of data store they wish to use on their machines.

4 FUTURE RESEARCH

The research presented here is just the beginning of what will be needed to complete the LDR system. The future research and development needs of the LDR system can be broken into four basic categories corresponding to the four major components of the system: data classes, data store, system manager, and tools. Below I outline some of the future projects that can be started now that the concept for basic system has been laid out.

4.1 Data Classes

I have presented a basic outline for the persistent data to be represented by the system, and for the data associated with a word. The system design needs to be expanded to include the ability to represent utterances or phrases, dialogues, and monologues. Representation of this data will require the expansion of the phonological transcriptions. In addition to this "static" data, the LDR system will need to represent rule based data. One of the fundamental tenants of linguistics is that language is systematic and governed by predictable rules. In order to accurately represent human languages, the LDR system will need to be able to represent morphophonological rules, phonological rules, syntactic rules and more. The development of methods for these rule based data will require a significant research effort due to the complexity of the data and the lack of consensus among linguists as to how these rules should be represented conceptually. There is also a considerable amount of computational difficulty involved in describing these rules and efficiently applying them over appropriate subsets of the data stored in the system.

In addition to developing a more complete representation of data, future research in the LDR system will need to develop a more complete representation of ethnolinguistic data. Due to time constraints and the focus of research, I have just barely scratched the surface in my examination of the requirements for representation of ethnolinguistic information. Ethnolinguistic data is a critical part of the analysis process and must be well represented by the LDR system. Future work will need to conduct a through analysis of what data is needs to be represented and how to most efficiently and represent that data.

Other projects in this area of the system development include developing a robust distinctive feature character, analyzing and optimizing the design and developing a revision control system.

4.2 Database

As the needs for data representation are expanded, the system database needs to be extended to meet the storage requirements for this data. Before that can happen though, the prototype database, implemented in Oracle, needs to be ported to a more robust PostgreSQL implementation. The database development section will also be responsible for developing the database peer classes prescribed by the persistence library (cf. section 3.4.2) and for determining the need and usefulness for stored procedures and other database optimization features. Like the data classes, the database will need to be analyzed for efficiency considerations. The database will also need to support a revision control system.

4.3 System Manager

Work on the system manager is currently being conducted by Ryan Saunders. Future work includes the implementation of a scripting language to allow the user to set a variety of system features. This language will govern properties of the system manager ranging from keyboard and character maps to window layout and system colors. Other projects for the system manager development include support for operating in a distributed environment (e.g. specifying the data class server and the database server to be used and accessing those servers), developing methods for inter-process communication, implementing user profiling, enhancing the management of plug-in tools and design and optimization of the user interface layout.

4.4 Tools

The biggest need in the development of tools at this time is to determine what tools are needed, what those tools should do and how those tools should interact with the user. While the implementation of a few basic tools may be warranted, the main development of

tools needs to be delayed until the system manager has reached a more stable point in its development phase.

4.5 Challenges

The greatest challenge for the completion of the LDR project is a lack of personnel resources. Currently, man-power is needed in the areas of additional linguistic research, systems analysis, database design, and general software implementation and testing. I am currently investigating the possibility of recruiting a number of individuals to aide in the development of the system. Other challenges facing the project are acceptance of the user community and the difficulty of developing reliable documentation. User community acceptance is a standard problem with software development tasks in general and has proved especially difficult in systems similar to the LDR. Linguists who are comfortable with the techniques that they have been using for years and who may not be technically oriented, frequently are unwilling to endure the learning curve required to make a system such as LDR useful. There is also a considerable amount of skepticism among users about such systems. This obstacle may be somewhat lessened by the established training processes of the user community. If the system is truly useful, coursework can be added to the existing training program to familiarize individuals new to the field with the system. This familiarization process would allow for an evolutionary acceptance process. The challenge of producing reliable documentation is also a traditional problem for computer science that is compounded for the LDR system. Developing good documentation is difficult. The documentation required for the LDR system must support an unusual variety of users with dramatically different knowledge bases. These users range from technically sophisticated software development groups to technically illiterate end users. This challenge will be moderated by a rigorous documentation program integrated into the design and implementation program.

5 SUMMARY

The results of my research from colleagues, professors, and professionals in industry has been very positive. In this paper I have provided a foundational description of a software system that will aid linguists in the cataloguing and the analysis of language data. This system, unlike similar systems, will provide fundamental support for collaborative efforts through its user-profiling feature and its support for distributed environments. Additionally, the modular structure of the system supports the flexibility and extensibility needed by the user community. This flexibility and extensibility is a direct result of the use of "plug-in" tools that provide the primary functionality of the system. The structure of the persistence class library allows for improvements and refinements to be made to the design of various elements of the architecture with minimal difficulty. I have concluded from my research that this system provides many significant conceptual improvements over existing systems. Unfortunately, due to extremely limited resources, it is unlikely that we will be able to field a robust version of the system in the near future. Fortunately, it is possible to develop and field a preliminary system that will implement a sub-set of the data that the final system will need to implement. Once this system is released, it can be incrementally improved as new features are added.

REFERENCES

Akmajian, Adrian, Richard A. Demers, Ann K. Farmer and Robert M. Harnish. 1998. *Linguistics: An Introduction to Language and Communications. Fourth Edition.* Cambridge, MA: The M.I.T. Press.

Bernard, H. Russell. 1995. *Research Methods in Anthropology: Qualitative and Quantitative Approaches. Second Edition.* Walnut Creek, CA: AltaMira Press.

Clark, John and Colin Yallop. 1990. *An Introduction to Phonetic and Phonology. Second Edition.* Oxford: Blackwell.

Goldsmith, J. 1990 *Autosegmental and Metrical Phonology.* Oxford: Blackwell.

Jakobson, R., and M. Halle. 1956. *Fundamentals of Language.* The Hague: Mouton.

Kenstowicz, M. 1994. *Phonology in Generative Grammar.* Oxford: Blackwell.

Lass, Roger. 1984 *Phonology: An Introduction to Basic Concepts.* Cambridge, NY: Cambridge University Press.

Mendoza-Denton, Norma. 1997. Chicana/Mexicana Identity and Linguistic Variation: An Ethnographic and Sociolinguistic Study of Gang Affiliation in an Urban High School. Ph.D. Dissertation, Stanford University.

O'Grady, William, Michael Dobrovolsky and Mark Arnoff. 1997. *Contemporary Linguistics.* New York: St. Martins Press.

Reese, George. 1997. *Programming with JDBC and Java.* Sebastobol, CA: O'Reily & Associates, Inc.

Roca, Iggy and Wyn Johnson. 1999. A Workbook in Phonology. Oxford: Blackwell.

SUPPLEMENTAL SOURCES CONSULTED

Bloomfield, Leonard. [1920] 1984. *Language.* Chicago: The University of Chicago Press.

Carroll, John B (ed.). [1956] 1987. *Language, Thought, and Reality: Selected Writings of Benjamin Lee Whorf.* Cambridge, MA: The M.I.T. Press.

deSaussure, Ferdinand. 1959. *A Course in General Linguistics.* New York: McGraw-Hill Publishers

Gass, Susan M. and Larry Selinker. 1994. *Second Language Acquisition: An Introductory Course.* Hillsdale, NJ: Lawrence Earlbaum Associates.

Pullum, Geoffrey and William Ladusaw. 1986 *Phonetic Symbol Guide.* Chicago: The University of Chicago Press.

Sapir, Edward. [1921] 1949. Language: An Introduction to the Study of Speech. Fort Worth: Harcourt, Brace & Jovanovih.

APPENDIX A: THE DISTINCTIVE FEATURE CHARACTER BIT-FIELD

The LDR architecture will implement a distinctive feature character set to define a `DFChar` object. This character will be implemented as a bit-field with each bit representing one binary distinctive feature. The tables below show an extremely preliminary representation of the `DFChar` bit-field and the feature associated with each bit. This is provided only to give the reader an idea of how the DFChar will be structured. It is not intended to be a final specification of the bit field and much work will be required before this character type will be useful. Also shown are the constants defined by the `DFChar` class for the various features and the values of these constants. Below each table is a more complete description of the features contained in that table.

Bit	Feature	Constant	Value
0.	Obstruent	OBS	1
1.	Consonantal	CONS	2
2.	Syllabic	SYL	4
3.	UNUSED	UNUSED	8

Table 1: Primary Features

Obstruent – Obstruents show a minimal output of periodic acoustic energy. Stops, fricatives and affricatives would be obstruents whereas liquids, nasals and vowels would not be.

Consonantal – Indicates a stricture of full closure to close approximation

Syllabic – The presence of this bit indicates a phone that may serve as the nucleus of a syllable. Among other things the inclusion of this feature allow for the distinction of [j w] from both liquids and vowels.

Bit	Feature	Constant	Value
4.	Oral	ORAL	16
5.	Coronal	COR	32
6.	Anterior	ANT	64
7.	UNUSED	UNUSED	128

Table 2: Cavity Features –Primary Strictures

Oral –This distinguishes between constrictions occurring in the oral cavity and pharynx, from those in the larynx. Nasals, contrary to what might be expected, are [+ORAL] as they are produced distal to the glottis. This feature, among other things, allows us to distinguish unvoiced glottal stops and fricatives from their non-glottal counterparts.

Coronal – Refers to an elevation of the blade of the tongue. Dental, alveolar, retroflex and palato-alveolar consonants are [+COR].

Anterior – Provides a reference point for strictures based on the palato-alveolar region. Strictures anterior to the [ʃ] are defined as [+ANT]

Bit	Feature	Constant	Value
8.	High	HIGH	256
9.	Mid	MID	512
10.	Front	FRONT	1024
11.	Back	BACK	2048

Table 3: Cavity Features –Primary Strictures

High + Mid – High and Mid serve to define a four tier height system: [+HIGH, -MID] (high), [+HIGH, +MID] (high middle), [-HIGH, +MID] (low middle), [-HIGH, -MID] (low). The more common method of representing tongue body height in terms of a high/low displacement from a neutral tongue position allows for the representation of only three heights. Since examples of four height vowel systems can be found in dialects of both Danish and German (Lass 87) a simple high/low displacement system for tongue height will not be sufficient for the purposes of LDR. One of the drawbacks of using the proposed tongue height system is that the concept of a neutral reference point is lost. This could be remedied by the inclusion of a low feature, but that seems unnecessary at this point.

Front + Back – The use of front and back features allows for the location of the tongue body in three general areas: front, back and center.

Bit	Feature	Constant	Value
12.	Nasal	NASAL	4096
13.	Lateral	LATERAL	8192
14.	Round	ROUND	16384
15.	Inrounded	INROUND	32768

Table 4: Secondary Apparatus & Lip-Attitude

Nasal – Positive value indicates a lowered velum allowing air to pass through the nasal pharynx.

Lateral – Positive value indicates airflow along one or both sides of the tongue. According to Lass SPE restricts [+lateral] to [+coronal] but this does not account for velar laterals and ejectives.

Round – Defines the narrowing or lack of narrowing of the lip orifice.

Inrounded – Positive value indicates retraction and vertical lip compression. This feature is proposed by Lass to account for what he describes as two different types of rounding in Swedish.

Bit	Feature	Constant	Value
16.	Voice	VOICE	65536
17.	Constricted	CONST	131072
18.	Murmur	MURMUR	262144
19.	Strident	STRIDENT	524288

Table 5: Source Features

Voice – Positive value indicates that the glottis is vibrating to produce periodic output

Constricted – Voice plus laryngeal constriction, as seen in some forms of creaky voicing

Murmur – Another name for breathy voice. Vocal folds vibrate but the artynoid cartilages are apart and allows air to leek past without vibration.

Strident – High frequency noise (e.g. sibilants)

Bit	Feature	Constant	Value
20.	Aspirated	ASPIRATED	1048576
21.	Egressive	EGRES	2097152
22.	Glotalic	GLOT	4194304
23.	Velaric	VELAR	8388608

Table 6: Airstream

Aspirated – Indicates a consonant whose stricture is released before the onset of voice in a following segment.

Egressive – Positive value indicates airflow in the direction of the lips, negative value indicates airflow in the direction of the lungs.

Glotalic – Positive value indicates an airstream that is initiated by a movement of a glottal closure.

Velaric – Positive value indicates and airstream that is initiated by a movement of a velar closure.

Bit	Feature	Constant	Value
24.	Distributed	DISTRIB	16777216
25.	Continuant	CONTIN	33554432
26.	Delayed Release	DEL_REL	67108864
27.	UNUSED	UNUSED	134217728

Table 7: Miscellaneous

Distributed – Refers to the relative length of the stricture with [+distrib] being relatively long and [-distrib] being relatively short.

Continuant – Positive value indicates that airflow is not completely blocked at any point.

Delayed Release – Positive value indicates that the full release of a stop is delayed. Typically found in affricatives.

Bit	Feature	Constant	Value
28.	UNUSED	UNUSED	268435456
29.	UNUSED	UNUSED	536870912
30.	UNUSED	UNUSED	1073741824
31.	UNUSED	UNUSED	2147483648

Table 8: Unused

APPENDIX B: THE PHONCHAR CHARACTER MAP

002 stops

Symbol	PhonChar Value	Unicode Value	Keystroke	Name	Description
p	0	0070	p	Latin small letter P	bilabial stop voiceless
b	1	0062	b	Latin small letter B	bilabial stop voiced
t	2	0074	t	Latin small letter T	dental stop voiceless
d	3	0064	d	Latin small letter D	dental stop voiced
ʈ	4	0288	sh alt t	Latin small letter T with retroflex hook	retroflex stop voiceless
ɖ	5	0256	sh alt d	Latin small letter D with retroflex hook	retroflex stop voiced
c	6	0063	c	Latin small letter C	palatal stop voiceless
ɟ	7	025F	sh alt j	latin small letter dotless J with stroke	palatal stop voiced
k	8	006B	k	latin small letter K	velar stop voiceless
g	9	0261	g	latin small letter script G	velar stop voiced
q	A	0071	q	latin small letter q	uvular stop voiceless
G	B	0262	sh g	latin letter small capital G	uvular stop voiced
ʔ	C	0294	/	latin letter glottal stop	glottal stop voiceless
	D				
	E				
	F				

[4] The four digit hexadecimal value for the PhonChar character may be obtained by post-pending the one digit PhonChar Value onto the end of the three digit table value. (e.g. The value for 'g' would be 0029, the 002 for the stop table and 9 for the character.)

003 Nasals, Trills, Taps, Flaps

Symbol	PhonChar Value	Unicode Value	Keystroke	Name	Description
m	0	006D	m	latin small letter M	bilabial nasal
ɱ	1	0271	sh m	latin small letter M with hook	labiodental nasal
n	2	006E	n	latin small letter N	alveolar nasal
ɳ	3	0273	alt n	latin small letter N with retroflex hook	retroflex nasal
ɲ	4	0272	sh alt n	latin small letter N with left hook	palatal nasal
ŋ	5	014B	sh n	latin small letter N with hook	velar nasal
N	6	0274	alt ,	latin letter small capital N	uvular nasal
	7				
	8				
B	9	0299	sh alt b	latin letter small capital B	bilabial trill
r	A	0072	r	latin small letter R	alveolar trill
R	B	0280	sh [	latin letter small capital R	uvular trill
ɾ	C	027E	sh r	latin small letter R with fish hook	alveolar tap
ɽ	D	027D	sh]	latin small letter R with tail	retroflex tap
	E				
	F				

004 Fricatives

Symbol	PhonChar Value	Unicode Value	Keystroke	Name	Description
ɸ	0	0278	sh alt p		bilabial fricative voiceless
β	1	03B2	sh b		bilabial fricative voiced
f	2	0066	f		labiodental fricative voiceless
v	3	0076	v		labiodental fricative voiced
θ	4	03B8	sh t		dental fricative voiceless
ð	5	00F0	sh d		dental fricative voiced
s	6	0073	s		alveolar fricative voiceless
z	7	007A	z		alveolar fricative voiced
ʃ	8	0283	sh s		postalveolar fricative voiceless
ʒ	9	0292	sh z		postalveolar fricative voiced
ʂ	A	0282	alt s		retroflex fricative voiceless
ʐ	B	0290	alt z		retroflex fricative voiced
ç	C	00E7	sh c		palatal fricative voiceless
ʝ	D	029D	alt j		palatal fricative voiced
x	E	0078	x		velar fricative voiceless
ɣ	F	0263	alt f		velar fricative voiced

005 Fricatives

Symbol	PhonChar Value	Unicode Value	Keystroke	Name	Description
χ	0	03C7	sh x		uvular fricative voiceless
ʁ	1	0281	alt [		uvular fricative voiced
ħ	2	0127	sh alt h		pharangeal fricative voiceless
ʕ	3	0295	alt /		pharyngeal fricative voiced
h	4	0068	h		glottal fricative voiceless
ɦ	5	0266	alt h		glottal fricative voiced
	6				
	7				
	8				
	9				
	A				
	B				
	C				
	D				
	E				
	F				

006 laterals and approximants

Symbol	PhonChar Value	Unicode Value	Keystroke	Name	Description
ɬ	0	026C	alt l		alveolar lateral-fricative voiceless
ɮ	1		sh l		alveolar fricative lateral voiced
ʋ	2	028B	sh v		labiodental approximant
ɹ	3	0279	alt r		alveolar approximant
ɻ	4	027B	sh alt [		retroflex approximant
j	5	006A	j		palatal approximant
ɰ	6	0270	sh alt m		velar approximant
l	7	006C	l		alveolar lateral-approximant
ɭ	8	026D	sh alt l		retroflex lateral-approximant
ʎ	9	028E	alt y		palatal lateral-approximant
ʟ	A	027F	;		velar lateral-approximant
	B				
	C				
	D				
	E				
	F				

007 Implosives

Symbol	PhonChar Value	Unicode Value	Keystroke	Name	Description
ɓ	0	0253	[redacted]		bilabial implosive voiced
ɗ	1	0257	alt d		alveolar implosive voiced
ɠ	2	0260	alt g		velarr implosive vioced
ʛ	3	029B	alt q		
[redacted]	4	[redacted]	[redacted]		
[redacted]	5	[redacted]	[redacted]		
[redacted]	6	[redacted]	[redacted]		
[redacted]	7	[redacted]	[redacted]		
[redacted]	8	[redacted]	[redacted]		
[redacted]	9	[redacted]	[redacted]		
[redacted]	A	[redacted]	[redacted]		
[redacted]	B	[redacted]	[redacted]		
[redacted]	C	[redacted]	[redacted]		
[redacted]	D	[redacted]	[redacted]		
[redacted]	E	[redacted]	[redacted]		
[redacted]	F	[redacted]	[redacted]		

008 other characters

Symbol	PhonChar Value	Unicode Value	Keystroke	Name	Description
ʍ	0	028D	sh alt w		labial-velar fricative voiceless
w	1	0077	w		labial-velar fricative voiced
ɥ	2	0265	sh alt y		labial-palatal approximant
ʜ	3	029	sh k		epiglottal fricative voiceless
ʢ	4	02A2	sh alt /		epiglottal fricative voiced
ʡ	5	02A1	sh /		epiglottal plosive
ɕ	6	0255	sh alt x		alveolo-palatal fricative voiceless
ʑ	7	0291	sh alt z		alveolo-palatal fricative voiced
ɺ	8	027A	sh alt r		alveolar-latteral flap
ʃʝ	9	0267	sh alt h		simultanious S and x
[illegible]	A	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	B	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	C	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	D	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	E	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	F	[illegible]	[illegible]	[illegible]	[illegible]

009 Clicks

Symbol	PhonChar Value	Unicode Value	Keystroke	Name	Description
⊙	0	0298	alt 1		bilabial click
|	1	01C0	alt 2		dental click
!	2	01C3	alt 3		postalveolar click
ǂ	3	01C2	alt 4		palatoalveolar click
‖	4	01C1	alt 6		alveolarr-lateral click
	5				
	6				
	7				
	8				
	9				
	A				
	B				
	C				
	D				
	E				
	F				

00A Reserved Characters

Symbol	PhonChar Value	Unicode Value	Keystroke	Name	Description
i	0	0069	i		
y	1	0079	y		
I	2	026A	sh i		
Y	3	028F	sh y		
e	4	0065	e		
ø	5	00F8	sh o		
ɛ	6	025B	sh e		
œ	7		alt e		
æ	8	00E6	sh q		
a	9	0061	a		
Œ	A	0276	sh alt o		
ɨ	B	0268	alt i		
ʉ	C	0289	alt u		
ɞ	D	0258	sh alt f		
ɵ	E	0275	sh alt u		
ə	F	0259	alt e		

00B Reserved Characters

Symbol	PhonChar Value	Unicode Value	Keystroke	Name	Description
ɜ	0	025C	sh alt e		
ɢ	1	025E	alt x		
ɐ	2	0250	alt a		
ɯ	3	026F	alt m		
u	4	0075	u		
ʊ	5	028A	sh u		
ɣ	6	0264	sh f		
o	7	006F	o		
ʌ	8	028C	alt v		
ɔ	9	0254	alt c		
ɑ	A	0251	sh a		
ɒ	B	0252	sh alt a		
	C	0277	alt w	latin small letter closed omega	
	D				
	E				
	F				

010 Diacritics

Symbol	PhonChar Value	Unicode Value	Keystroke	Name	Description
d̥	0	0325	8		Voiceless
t̬	1	032C	alt 6		Voiced
tʰ	2	02B0	sh h		Aspirated
tʼ	3	0315	'		Ejective stop
t̤	4	0324	-		Breathy voiced
t̰	5	0330	0		creaky voiced
t̼	6	033C	alt '		Linguolabial
t̪	7	032A	5		Dental
t̺	8	033A	alt 5		Apical
t̻	9	033B	6		Laminal
ɔ̹	A	0339	alt 7		More rounded
ɔ̜	B	031C	7		Less rounded
u̟	C	031F	sh =		Advanced
a̠	D	0331	=		Retracted
ë	E	0308	sh -		Centralized
e̽	F	0330	sh '		Mid-Centralized

011 Diacritics

Symbol	PhonChar Value	Unicode Value	Keystroke	Name	Description
e̩	0	0329	`		Syllabic
e̯	1	032F	9		Non-Syllabic
e˞	2	02DE	sh alt j		Rhoticity
t^{w}	3	02B7	sh w		Labialized
t^{j}	4	02B2	sh j		Palatalized
$t^{ɣ}$	5	02E0	sh alt g		Velarized
$t^{ʕ}$	6	02C1	alt .		Pharyngealized
ɬ	7	0334	sh alt ;		Velarized or pharyngealized
i̝	8	031D	3		Raised
i̞	9	031E	4		Lowered
e̘	A	0319	2		Advancing Tongue Root
e̙	B	0318	1		Retracted Tongue Rood
ĩ	C	0342	sh 0		Nasalized
d^{n}	D		sh ,		Nasal release
d^{l}	E	02E1	sh .		Lateral Release
$d^{˥}$	F		sh \		no audible release

012 Diacritics

Symbol	PhonChar Value	Unicode Value	Keystroke	Name	Description
eː	0	02D0	sh alt .		Long
eˑ	1	02D1	sh .		half-long
ĕ	2	0306	sh 9		Extra-short
k‿p	3		sh alt -		Linking (absence of a break
k͡p	4	0361	sh alt =		Double articulations
	5				
	6				
	7				
	8				
	9				
	A				
	B				
	C				
	D				
	E				
	F				

APPENDIX C: THE INTERNATIONAL PHONETIC ALPHABET

(revised to 1993, corrected 1996)

Consonants (Pulmonic)

	Bilabial	Labiodental	Dental	Alveolar	Postalveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Glottal
Plosive	p b			t d		ʈ ɖ	c ɟ	k g	q ɢ		ʔ
Nasal	m	ɱ		n		ɳ	ɲ	ŋ	N		
Trill	ʙ			r					ʀ		
Tap or flap				ɾ		ɽ					
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ʝ	x ɣ	χ ʁ	ħ ʕ	h ɦ
Lateral fricative				ɬ ɮ							
Approximant		ʋ		ɹ		ɻ	j	ɰ			
Lateral approximant				l		ɭ	ʎ	ʟ			

Where symbols appear in pairs, the one to the right represents a voiced consonant. Shaded areas denote articulations judged impossible.

Consonants (Non-Pulmonic)

Clicks	Voiced implosives	Ejectives
ʘ Bilabial	ɓ Bilabial	' Examples:
ǀ Dental	ɗ Dental/alveolar	pʼ Bilabial
ǃ (Post)alveolar	ʄ Palatal	tʼ Dental/alveolar
ǂ Palatoalveolar	ɠ Velar	kʼ Velar
ǁ Alveolar lateral	ʛ Uvular	sʼ Alveolar fricative

Other Symbols

ʍ Voiceless labial-velar fricative
ɕ ʑ Alveolo-palatal fricatives
w Voiced labial-velar approximant
ɺ Alveolar lateral flap
ɥ Voiced labial-palatal approximant
ʜ Simultaneous ʃ and x
ʜ Voiceless epiglottal fricative
ʢ Voiced epiglottal fricative
ʡ Epiglottal plosive

Affricates and double articulations can be represented by two symbols joined by a tie bar if necessary.

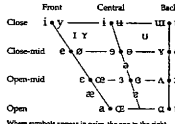
k͡p t͡s

Diacritics

Diacritics may be placed above a symbol with a descender, e.g. ŋ̊

̥	Voiceless	n̥ d̥	̤	Breathy voiced	b̤ a̤
̬	Voiced	s̬ t̬	̰	Creaky voiced	b̰ a̰
ʰ	Aspirated	tʰ dʰ	̼	Linguolabial	t̼ d̼
̹	More rounded	ɔ̹	ʷ	Labialized	tʷ dʷ
̜	Less rounded	ɔ̜	ʲ	Palatalized	tʲ dʲ
̟	Advanced	u̟	ˠ	Velarized	tˠ dˠ
̠	Retracted	e̠	ˤ	Pharyngealized	tˤ dˤ
̈	Centralized	ë	~	Velarized or pharyngealized	ɫ
̽	Mid-centralized	e̽	̝	Raised	e̝ (ɹ̝ = voiced alveolar fricative)
̩	Syllabic	n̩	̞	Lowered	e̞ (β̞ = voiced bilabial approximant)
̯	Non-syllabic	e̯	̘	Advanced tongue root	e̘
˞	Rhoticity	ɚ ɝ	̙	Retracted tongue root	e̙

Vowels



Where symbols appear in pairs, the one to the right represents a rounded vowel.

Suprasegmentals

ˈ Primary stress
ˌ Secondary stress
ˌfounəˈtɪʃən
ː Long eː
ˑ Half-long eˑ
˘ Extra-short ĕ
| Minor (foot) group
‖ Major (intonation) group
. Syllable break ɹi.ækt
‿ Linking (absence of a break)

Tones and Word Accents

Level		Contour	
˩̋ or ˥	Extra high	˨̋ or ˧˥	Rising
˦	High	˥̋	Falling
˧	Mid	˦˥	High rising
˨	Low	˧˩	Low rising
˩̏ or ˩	Extra low	˩̏˥	Rising-falling
↓	Downstep	↗	Global rise
↑	Upstep	↘	Global fall

APPENDIX D: THE SQL SOURCE CODE

```
REM table to store the address of a person, organization or other
      entity.
Create Table Address
  (id raw(4) Constraints add_raw PRIMARY KEY,
   line1 CHAR(1024),
   line2 CHAR(1024),
   line3 CHAR(1024),
   city CHAR(255),
   province CHAR(255),
   postalCode CHAR(64),
   country CHAR(128),
   type CHAR(64));

REM defines a data element. each element has an owner (person), a
      langauge
REM and a confidence value.
create table DataElement
  (id RAW(8) CONSTRAINT pk_de PRIMARY KEY,
   ownerID raw(4) CONSTRAINT fk_owner REFERENCES person(id),
   lgID raw(4) CONSTRAINT fk_lg REFERENCES Language(id),
   confidence raw(1));

REM stores the name, description and location of an
      EthnolinguisticGroup
Create Table EthnolinguisticGroup
  (name char(255) constraint elg_name PRIMARY KEY,
   description long,
   location char(1024));

REM defines the various ethnolinguistic roles invidividuals play with
      in a elg.
create table EthnolinguisticRole
  (elgRole CHAR(255),
   elgName CHAR(255) CONSTRAINT elgR_fk1 REFERENCES
      EthnolinguisticGroup(name) ON DELETE CASCADE,
   identifiedBy RAW(8) CONSTRAINT elgRoleIdentBy REFERENCES
      LDR_User(id),
   description long,
   CONSTRAINT elgRole PRIMARY KEY (elgRole, elgName));

REM defines an informant as a type of person.  Has place of residence
REM age and gender information
Create Table Informant
  (id RAW(4) constraint fk_id references Person(id) ON DELETE CASCADE,
   name CHAR(255),
   residence CHAR(1024),
   age NUMBER(3,0),
   gender CHAR(1),
   CONSTRAINT pk_inf PRIMARY KEY (id));
```

```
REM create a view of the table that does not include the informant's
      name, but does include the alias from person
Create View Inf As
  Select Informant.id, age, gender
  FROM Informant, Person
  WHERE Person.id = Informant.id;


REM defines a user of the LDR system as a type of person.  Has a
      password
REM age and gender.
Create Table LDR_User
  (id RAW(4) constraint fk_usr references Person(id) ON DELETE CASCADE,
   loginName CHAR(32) CONSTRAINT usr_loginName NOT NULL,
   password CHAR(32) constraint usr_password NOT NULL,
   age NUMBER(3,0),
   gender CHAR(1),
   CONSTRAINT uq_LDR_login UNIQUE (loginName),
   CONSTRAINT pk_LDR_User PRIMARY KEY (id));

REM create a view of the table that does not include the user's
      password
Create View Usr As
  Select l.id, loginName, p.full, p.first, p.middle, p. last, p.title,
      age, gender
  FROM LDR_User l, Person p
  WHERE l.id = p.id;


REM defines a language, identified by id, having a name, a description
      and
REM physical location.
Create Table Language
  (id RAW(4) Constraint lg_id PRIMARY KEY,
   name CHAR(255) Constraint lg_name NOT NULL,
   description LONG,
   location CHAR(255));

REM defines a name for Persons.  Allows specification of first, middle
      and
REM last names, as well as a full name and a title.
Create Table Name
  (id raw(4) constraint fk_name_id References Person(id) ON DELETE
      CASCADE,
   full char(255),
   first char(255),
   middle char(255),
   last char(255),
   title char(64),
   Constraint name_id PRIMARY KEY (id));
```

```
REM defines notes entity having a text value, an author a type (e.g.
      data
REM notes, personnel notes, etc.
Create Table Notes
  (id raw(8),
   text long constraint notes_txt NOT NULL,
   author raw(4) constraint fk_notes_aut references Person(id),
   type char(64) constraint notes_type NOT NULL,
   datawritten date,
   Constraint pk_notes PRIMARY KEY (id, author));

REM defines an organization, allows users to be affiliated with
      organizations
Create Table Organization
  (name CHAR(255) constraint org_name PRIMARY KEY,
   description long);

REM defines an orthographic form, a type of data element and a
      transcription
REM of a word.
create table OrthographicForm
  (id raw(8) CONSTRAINTS orthF_id REFERENCES DataElement(id) ON DELETE
      CASCADE,
   wordID raw(8) CONSTRAINT orthF_wordID REFERENCES Word(id) ON DELETE
      CASCADE,
   transcription LONG RAW CONSTRAINT orthF_trans NOT NULL,
   CONSTRAINT pk_orhtForm PRIMARY KEY (id));

REM defines a person giving an id and an alias (login name for user).
      The name
REM The name entity allows a name to be assigned to each person.
REM There are two types of Person entities supported by the database
      LDR_Users
REM and Informants
Create Table Person
  (id raw(4) CONSTRAINT pk_per_id PRIMARY KEY,
   isUser CHAR(1) CONSTRAINT per_isUsr NOT NULL,
   full char(255),
   first char(255),
   middle char(255),
   last char(255),
   title char(64));

REM defines a surface form, a type of data element and transcription of
      a word
create table SurfaceForm
  (id RAW(8) CONSTRAINT surF_id REFERENCES DataElement(id) ON DELETE
      CASCADE,
   wordID RAW(8) CONSTRAINT surF_wordID REFERENCES Word(id) ON DELETE
      CASCADE,
   transcription LONG RAW CONSTRAINT surf_trans NOT NULL,
   CONSTRAINT pk_surF PRIMARY KEY (id));

REM defines an entity to represent a syntagmatic function in a language
```

```
create table SyntagmaticFunction
  (name CHAR(255),
   lgID RAW(4) CONSTRAINT fk_sf_lg REFERENCES Language(id) ON DELETE
      CASCADE,
   description LONG,
   parent CHAR(255),
   CONSTRAINT pk_sf PRIMARY KEY (name, lgID),
   CONSTRAINT fk_sf_parent FOREIGN KEY (parent, lgID) REFERENCES
      SyntagmaticFunction(name, lgID));

REM defines a surface form, a type of data elment and a transcription
      of a word
create table UnderlyingForm
  (id RAW(8) CONSTRAINT undlF_id REFERENCES DataElement(id) ON DELETE
      CASCADE,
   wordID RAW(8) CONSTRAINT undlF_wordID REFERENCES Word(id) ON DELETE
      CASCADE,
   transcription LONG RAW CONSTRAINT undlF_trans NOT NULL,
   CONSTRAINT pk_undlF PRIMARY KEY (id));

REM defines word, a type of data element
create table Word
  (id raw(8) CONSTRAINT fk_word_id REFERENCES DataElement(id),
   CONSTRAINT pk_word PRIMARY KEY (id));

REM defines a relationship between a data element and a user
REM indicating that a particular data element was collected by a user
create table CollectedBy
  (deID RAW(8) CONSTRAINT fk_cb_deid REFERENCES DataElement(id) ON
      DELETE CASCADE,
   usrID RAW(4) CONSTRAINT fk_cb_usrid REFERENCES LDR_User(id) ON
      DELETE CASCADE,
   dateCollected DATE,
   location CHAR(1024),
   CONSTRAINT pk_cb PRIMARY KEY (deID, usrID));

REM defines a relation from informant to a data element indicating
REM which informant provided that data element
create table CollectedFrom
  (deID RAW(8) CONSTRAINT fk_cf_deid REFERENCES DataElement(id) ON
      DELETE CASCADE,
   infID RAW(4) CONSTRAINT fk_cf_inf REFERENCES Informant(id) ON DELETE
      CASCADE,
   dateCollected DATE,
   location CHAR(1024),
   CONSTRAINT pk_cf PRIMARY KEY (deID, infID));

REM defines a relation from elg to the languages spoken by that elg
REM also specifies the strata each language has in the elg
create table ELGSpeaksLG
  (lgID RAW(4) CONSTRAINT fk_elgspkslg_lg REFERENCES Language(id) ON
      DELETE CASCADE,
   elgName CHAR(255) CONSTRAINT fk_elgspkslg_elg REFERENCES
      EthnolinguisticGroup(name) ON DELETE CASCADE,
```

```
   strata CHAR(64),
   CONSTRAINT pk_elgspks PRIMARY KEY (lgID, elgName));

REM defines a relation between a word and its syntagmatic functions
create table HasSyntagFunc
  (wordID RAW(8) CONSTRAINT fk_hsf_word REFERENCES Word(id) ON DELETE
      CASCADE,
   name CHAR(255),
   lgID RAW(4),
   CONSTRAINT fk_hsf_synfunc FOREIGN KEY (name, lgID) REFERENCES
      SyntagmaticFunction(name, lgID) ON DELETE CASCADE,
   CONSTRAINT pk_hsf PRIMARY KEY (wordID, name, lgID));

REM defines a relation between a person and an elg that that person is
REM a member of
create table MemberOfELG
  (perID RAW(4) CONSTRAINT fk_memofelg_per REFERENCES Person(id) ON
      DELETE CASCADE,
   elgName CHAR(255) CONSTRAINT fk_memofelg_elgname REFERENCES
      EthnolinguisticGroup(name) ON DELETE CASCADE,
   CONSTRAINT pk_memofelg PRIMARY KEY (perID, elgName));

REM defines a relationship from Notes to DataElement
create table NotesRegardingDE
  (deID RAW(8) CONSTRAINT nrde_deid REFERENCES DataElement(id) ON
      DELETE CASCADE,
   noteID RAW(8),
   noteAut RAW(4),
   CONSTRAINT fk_nrde_note FOREIGN KEY (noteID, noteAut) REFERENCES
      Notes(id, author) ON DELETE CASCADE,
   CONSTRAINT pk_nrde PRIMARY KEY (deID, noteID, noteAut));

REM defines a relationship from Notes to a Person
create table NotesRegardingPer
  (perID RAW(4) CONSTRAINT nrp_perid REFERENCES Person(id) ON DELETE
      CASCADE,
   noteID RAW(8),
   noteAut RAW(4),
   CONSTRAINT fk_nrp_note FOREIGN KEY (noteID, noteAut) REFERENCES
      Notes(id, author) ON DELETE CASCADE,
   CONSTRAINT pk_nrp PRIMARY KEY (perID, noteID, noteAut));

REM defines a relation between organization and address
create table OrgHasAddress
  (orgName CHAR(255) CONSTRAINT fk_oha_org REFERENCES
      Organization(name),
   addID RAW(4) CONSTRAINT fk_oha_add REFERENCES Address(id),
   CONSTRAINT pk_oha PRIMARY KEY (orgName, addID));

REM defines a relation between person and address
create table PersonHasAddress
  (perID RAW(4) CONSTRAINT fk_pha_per REFERENCES Person(id),
   addID RAW(4) CONSTRAINT fk_pha_add REFERENCES Address(id),
   CONSTRAINT pk_pha PRIMARY KEY (perID, addID));
```

```
REM defines a ternary relation between person, elgRole and ELG
REM specifying a speakers ethnolinguistic role in an ELG.
create table PlaysRoleInELG
  (perID RAW(4) CONSTRAINT fk_elgRole_per REFERENCES Person(id) ON
      DELETE CASCADE,
   elgName CHAR(255),
   elgRole CHAR(255),
   CONSTRAINT fk_elgRole_role FOREIGN KEY (elgName, elgRole) REFERENCES
      EthnolinguisticRole(elgName, elgRole),
   CONSTRAINT pk_elgRole PRIMARY KEY (perID, elgName, elgRole));

REM defines a relation from LDR_User to a data element indicating
REM that the data element was proposed by the user
REM used for underlying, orthographic and other theoretically based
REM data
create table ProposedBy
  (deID RAW(8) CONSTRAINT fk_pb_deid REFERENCES DataElement(id) ON
      DELETE CASCADE,
   usrID RAW(4) CONSTRAINT fk_pb_usr REFERENCES LDR_User(id) ON DELETE
      CASCADE,
   dateProposed DATE,
   CONSTRAINT pk_pb PRIMARY KEY (deID, usrID));

REM defines a relation between person and the languages s/he speaks
create table Speaks
  (perID RAW(4) CONSTRAINT fk_spk_per REFERENCES Person(id) ON DELETE
      CASCADE,
   lgID RAW(4) CONSTRAINT fk_spk_lg REFERENCES Language(id) ON DELETE
      CASCADE,
   native CHAR(1) CONSTRAINT fk_spk_nat NOT NULL,
   CONSTRAINT pk_spk PRIMARY KEY (perID, lgID));

REM creates relation from DataElement to DataElement
REM indicates that a DataElement (deID) has been split or merged
REM to form a new DataElement (newID)
create table SplitOrMerged
  (deID RAW(8) CONSTRAINT som_deid REFERENCES DataElement(id) ON DELETE
      CASCADE,
   newID RAW(8) CONSTRAINT som_newid REFERENCES DataElement(id) ON
      DELETE CASCADE,
   revCode RAW(1) CONSTRAINT som_revcode NOT NULL,
   CONSTRAINT pk_som PRIMARY KEY (deID, newID, revCode));

REM defines a relation between ldr_user and ethnolinguisticGroup
REM relating users and the elg's they study
create table StudiesELG
  (usrID RAW(4) CONSTRAINT fk_selg_usr REFERENCES LDR_User(id) ON
      DELETE CASCADE,
   elgName CHAR(255) CONSTRAINT fk_selg_elgName REFERENCES
      EthnolinguisticGroup(name) ON DELETE CASCADE,
   CONSTRAINT pk_selg PRIMARY KEY (usrID, elgName));

REM defines a relation between ldr_user and language indicating that
```

```
REM a user studies or has studied a particular language
create table StudiesLG
  (usrID RAW(4) CONSTRAINT fk_slg_usr REFERENCES LDR_User(id) ON DELETE
      CASCADE,
   lgID  RAW(4) CONSTRAINT fk_slg_lg  REFERENCES Language(id) ON DELETE
      CASCADE,
   duration CHAR(64),
   CONSTRAINT pk_slg PRIMARY KEY (usrID, lgID));


REM defines a relation between organization and a user
create table UsrAffiliatedWith
  (orgName CHAR(255) CONSTRAINT fk_uaw_org REFERENCES
      Organization(name),
   usrID RAW(4) CONSTRAINT fk_uaw_usr REFERENCES LDR_User(id),
   CONSTRAINT pk_uaw PRIMARY KEY (orgName, usrID));

REM defines a relation between LDR_User and Informant describing
REM which informant works with which user
create table WorksWith
  (usrID RAW(4) CONSTRAINT fk_ww_usr REFERENCES LDR_User(id) ON DELETE
      CASCADE,
   infID RAW(4) CONSTRAINT fk_ww_inf REFERENCES Informant(id) ON DELETE
      CASCADE,
   alias CHAR(32) CONSTRAINT ww_alias NOT NULL,
   duration CHAR(4),
   CONSTRAINT pk_ww PRIMARY KEY (usrID, infID));
```

APPENDIX E: THE POSTGRESQL COPYRIGHT

VITA

Michael Neal Audenaert

110 Cavendish Cir. Madison, AL 35758
neala@tamu.edu

EDUCATION

Fall 1997-present	**Texas A&M University** – College Station, TX Major: Computer Science (GPR 3.59); Minor: Linguistics (GPR 3.6) Expected Graduation: May 2001
Spring 1998	**Blinn College** – Bryan, TX Emergency Medical Technician Certification

HONORS AND AWARDS

1997-2001	President's Endowed Scholarship	Amount: $12,000
1997-2001	National Merit Scholarship	Amount: $6,000
1997-present	Engineering Scholars Program	
Fall 1999-Spring 2000	University Undergraduate Research Fellows, Conducted independent research on the development of a software package to aid in the cataloguing and analysis of unwritten languages.	
Spring 2000	1st Place, Undergraduate Computer Science Poster Session, Student Research Week, Project: Language Data Repository: Machine Readable Storage for Spoken Language Data	
Fall 1999	1st Place, Engineering Scholars Program Poster Competition, Project: Language Data Repository: Machine Readable Storage for Spoken Language Data	
Spring 1998	New Medic of the Year, Texas A&M Emergency Care Team	
1997-1998	Thomas C. Lingeman Scholarship: Amount: $1,000	

PROFESSIONAL EXPERIENCE

Summer 1999	Summer Intern **Dynetics, Inc**. Huntsville, AL Enhanced THAAD Project Office IDEF0 model of the THAAD Battery Battle Management and C3I functional requirements. Compared the TPO THADD model with the US Army Air Defense Artillery School's model for Air Missile Defense Planning and Control Systems to show correspondence between the two models and to recommend modifications.

ACTIVITIES

Spring 2000	International Student Outreach, Grace Bible Church
Fall 1999	International Student Ministries Coordinator, Baptist Student Ministry
Fall 1997 - Present	Texas A&M Emergency Care Team, Events Coordinator (Fall 1998-Spring 1999), Operational Supervisor, Medic in Charge, CPR Instructor, First Aid Instructor, Webmaster (Fall 1998-Spring 1999)
Summer 1998	Conducted field research on the language and culture of the Yagarian, Bena Bena, and Bahenamo people groups in Papua New Guinea
Fall 1997 - Spring 1998	Student Conference on National Affairs, general committee member technology subcommittee

RESEARCH INTERSTS

Computational linguistics, unwritten languages, linguistic anthropology (indigenous people groups).