

A WEB-BASED DATABASE SYSTEM FOR SOIL,
WATER, AND FORAGE TESTING
INFORMATION MANAGEMENT

By

QIANG SU

Bachelor of Engineering
Tianjin University
Tianjin, P. R. China
1985

Master of Science
Vrije Universiteit Brussels
Brussels, Belgium
1994

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2000

Oklahoma State University Library

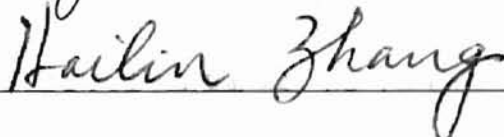
A WEB-BASED DATABASE SYSTEM FOR SOIL,
WATER, AND FORAGE TESTING
INFORMATION MANAGEMENT

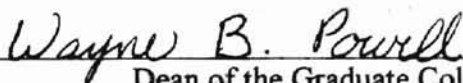
Thesis Approved:



Thesis Adviser







Dean of the Graduate College

PREFACE

In this thesis, a web-based database system is created for the Soil, Water, and Forage Analysis Laboratory at Oklahoma State University to manage test results of over 30,000 various samples and to interpret results to thousands of laboratory users annually.

HTML web pages are created as front end interfaces for data transfer and retrieval. The Oracle database manager on a UNIX machine is employed as the database server. OraPerl is used to implement the CGI programs that interact with the HTML web pages and the Oracle server. The two major functions of the OraPerl programs are to capture inputs from HTML pages and to send database manipulation requests to the Oracle server. Data transfer and retrieval are implemented as two separate web pages with their corresponding CGI programs. The uploading (data transfer) web page sends ASCII files containing test results to the web server and invokes the OraPerl program to upload the data from the ASCII files into corresponding Oracle tables. The uploading OraPerl program identifies the target table by the data format in an ASCII file, the modifies the input data, if necessary, to fit the data format of the target tables. In addition to uploading data into Oracle tables, the uploading CGI program generates reports for each set of test results uploaded. The reports are HTML documents that provide interpretations and recommendations based on the test results. The data retrieval web page receives customer identification number, password, then

data specification from a user, and invokes the OraPerl program to retrieve data from the Oracle table. The data retrieval CGI program formats the retrieved data into HTML documents that are sent back to the requesting browser for a customer either to view or to download.

This system significantly enhances analysis turnaround time, giving improvement of three to five days.

ACKNOWLEDGMENTS

I wish to express my sincere appreciation to my major advisor, Dr. G. E. Hedrick for his intelligent supervision, constructive guidance, inspiration and friendship. My sincere appreciation extends to my other committee members Dr. H. Zhang and Dr. J. P. Chandler, whose guidance, assistance, encouragement, and friendship are also invaluable.

I would also like to give my special appreciation to my husband, Jinquan Wu, for his precious suggestions for my research, love and understanding throughout this whole process. Thanks also go to my parents Mr. Zongxiong Su and Mrs. Lihua Shen for all love and support they have given me throughout my life. And, I would like to thank my brother Mr. Chen Su and his wife Mrs. Yumin Shi for their love, strong encouragement at times of difficulty, and confidence they have contributed to me.

Finally, I would like to express my gratitude to all of those who have helped by giving many valuable suggestions.

TABLE OF CONTENTS

Chapter		Page
I.	INTRODUCTION	1
II.	REVIEW OF THE LITERATURE	4
	ORACLE Database Management System	4
	Structured Query Language (SQL) and PL/SQL	8
	Practical Extraction and Report Language (PERL) and OraPERL	12
	Standard Generalized Markup Language (SGML) and Hyper Text Markup Language (HTML)	15
III.	DESIGN	21
	Overall SWFAL System Design	21
	Oracle Database Tables Design	22
	CGI Interface Design	23
	Web-page Design	27
	Security Design	28
IV.	IMPLEMENTATION	29
	Oracle Database Management	29
	Creating Tablespace	29
	Creating tables	29
	CGI Interface	31
	Structure of program -- upload.pl	31
	Structure of program -- result.pl	41
	Web-page	43
	Web-page of uploading data file	43
	Web-page of retrieval data	46
	Web-page of report	48
V.	SUMMARY AND FUTURE WORK	51
	Summary	51
	Future Work	52

BIBLIOGRAPHY	53
APPENDIX A: List of Acronyms	55
APPENDIX B: Maintenance of Generating Report Part in the System	56

LIST OF TABLES

Table	Page
1. Attributes of SWFAL tables	23

LIST OF FIGURES

Figure	Page
1. Tablespaces in Oracle	6
2. SQL, PL/SQL, OCI, Pro*C and OraPERL in Oracle	10
3. Architecture of the PL/SQL “Engine”	11
4. The client – server environment in the SWFAL system	22
5. Overall of the SWFAL system	25
6. The relation chart of CGI interface objects in SWFAL system	26
7. The general flow chart of upload.pl	32
8-1. The detailed flow chart of generate report	33
8-2. The detailed flow chart of generate report (continued)	34
8-3. The detailed flow chart of generate report (continued)	35
8-4. The detailed flow chart of generate report (continued)	36
8-5. The detailed flow chart of generate report (continued)	37
8-6. The detailed flow chart of generate report (continued)	38
8-7. The detailed flow chart of generate report (continued)	39
8-8. The detailed flow chart of generate report (continued)	40
9. The general flow chart of result.pl	41
10. The “file upload” web page	43
11. The “file upload” web page with file upload window	44

12. The “File Transfer” web page	45
13. The “dbresult” web page	46
14. The “records” web page	47
15. Soil test report	48
16. Water quality report	49
17. Forage analysis report	50

CHAPTER I

Introduction

The Soil, Water, and Forage Analytical Laboratory (SWFAL) at the Department of Plant and Soil Sciences, Oklahoma State University is part of the Cooperative Extension Service in Oklahoma. It performs analysis on water quality, soil fertility, and other water and soil-related tests to provide information to farmers, ranchers, and government agencies for enhancing agricultural production, minimizing the adverse impact on the environment, and protecting our natural resources. Conventionally, the test results are sent to customers by regular mail, so lab users must wait for about one week to receive their reports. This sometimes causes significant delays in information transfer. Both the customers and the laboratory managers strongly felt the need for a more convenient data management system. With the advent of World Wide Web (WWW) in association with the advancement in the technology of relational database management systems (RDBMS), it is possible to create a database management system that renders instant access of the test results to customers. The objective of this project is to create a set of Oracle tables to store the test results and a web page with its associated Common Gateway Interface (CGI) programs to upload and retrieve data into and from the Oracle tables.

The web-page system comprises two sub-components: one for uploading test results into an Oracle database on a UNIX server, the other for retrieving data from the Oracle server. Both the uploading and the retrieval components include a CGI program to access and manipulate the Oracle database and a Hyper Text Markup Language (HTML) program to receive input information from a user, send the inputs to the CGI program, and display output sent back by the CGI program. The CGI programs were implemented in OraPERL, which is an Oracle extension to the conventional Practical Extraction and Report Language (PERL). Four Oracle tables are created to store data from soil fertility test, forage test, soil and water salinity and soil texture test, respectively.

The uploading CGI program is designed to insert data into one table at a time (program run). Therefore, different tables of test data are saved in separate American Standard Code Information Interchange (ASCII) files for uploading. The uploading program identifies the target table of an uploaded file by the data format in the file. Besides inserting data into the oracle tables, the uploading program generates reports for each sample. The reports are HTML files, which are linked to the HTML file generated by the data-retrieval program. In the data-retrieval program, the target table from which data are retrieved is identified using the information passed to it from the HTML page that invokes the CGI program. The data from the database manipulation are formatted into an HTML program in the CGI program and are sent back to the browser for display. The main tasks of this project may be summarized as follows:

- Create four Oracle tables on a UNIX server to store test results;
- Design and implement a web page for uploading test results into the database tables;

- Design and implement a CGI program page for uploading test results into the database tables, and generating the report for each tested sample;
- Design and implement a web page for retrieving data from the database tables;
- Design and implement a CGI program page for retrieval data from the database tables.

This thesis is organized into five chapters. Chapter I is introduction. Chapter II briefly reviews the basic concepts and definitions of Oracle database management system, Structured Query Language (SQL) and PL/SQL, PERL and OraPERL, Standard Generalized Markup Language (SGML) and HTML. Chapter III gives the detail of design of the SWFAL system. Chapter IV discussed the implementation detail of the system. Chapter V summarizes the work of this project and projects some future work that can be done.

CHAPTER II

Review of the Literature

The SWFAL system consists of various computing techniques including Oracle Database Management System, SQL and PL/SQL, PERL and OraPERL, SGML and HTML. In this chapter, the author briefly overviews previous research results and the background information about all the above-mentioned technologies used to develop the system.

2.1 Oracle Database Management System`

Oracle is the most widely used database in the world. It runs on virtually every kind of computer, from PCs and Macintoshes, to minicomputers and giant mainframes, and it functions virtually identically on all these machines [KGLK, 1995]. Oracle is a Relational Database Management System (RDBMS). This concept is an extremely simple way of thinking and managing the data used in a business. It is nothing more than a collection of tables of data [RPCC, 1997].

A database model is a set of definitions describing how real-world data are conceptually represented as information in the computer. The relational model collects related information in a set of tables. A table either contains the attributes of an entity or the relationship between entities. This relational model used obeys in Oracle three relational rules of first normal form [RPCC, 1997]:

1. First normal form rule: a table must have no multi-valued columns.
2. Access rows by content only rule: we can only retrieve rows/columns by their content, and there is no order defined for the rows or columns.
3. The uniqueness rule: two rows in a table cannot be identical in all column values at once.

All RDBMS operations and SQL operators are built on the relational algebra theory that defines the set operations of union, intersection, difference, and product, as well as the native relational operations are project, select, join, and division. In Oracle, database structures are well-defined objects represented as tablespace and other schema objects that store the data of a database. Tablespace is the logical representation of the storage structure in an Oracle database. To some extent, a tablespace in an Oracle database system is analogous to the disk in an operating system. Physically, a tablespace usually consists or points to one or more operating system files. Inside Oracle, tablespaces are used to store Oracle objects, like tables, views, etc. The database operations of Oracle are actions to manipulate the data and their structures.

The first major schema object in ORACLE is **tablespace**; a database is divided into logical storage units called tablespace. A tablespace is used to group related logical structure together. Tablespaces commonly group all objects of an application to simplify certain administrative operations.

Figure 1 illustrates that each Oracle database is divided logically into one or more tablespaces. One or more data files are created explicitly for each tablespace to store the data of all logical structures in a tablespace physically. The combined size of a tablespace's data files is the total storage capacity of the tablespace (SYSTEM has 2M

while USERS has 4M). The combined storage capacity of a database's tablespaces is the total storage capacity of the database (6M as shown in Figure 1).

The second major schema object in Oracle is **table**, the basic unit of data storage in Oracle. The tables of a database hold all of the user-accessible data. Each table is defined with a table name and set of columns. Each column is given a column name, a data type, and a width. Once a table is created, valid rows of data can be inserted into it. The table's rows can then be queried, deleted, or updated.

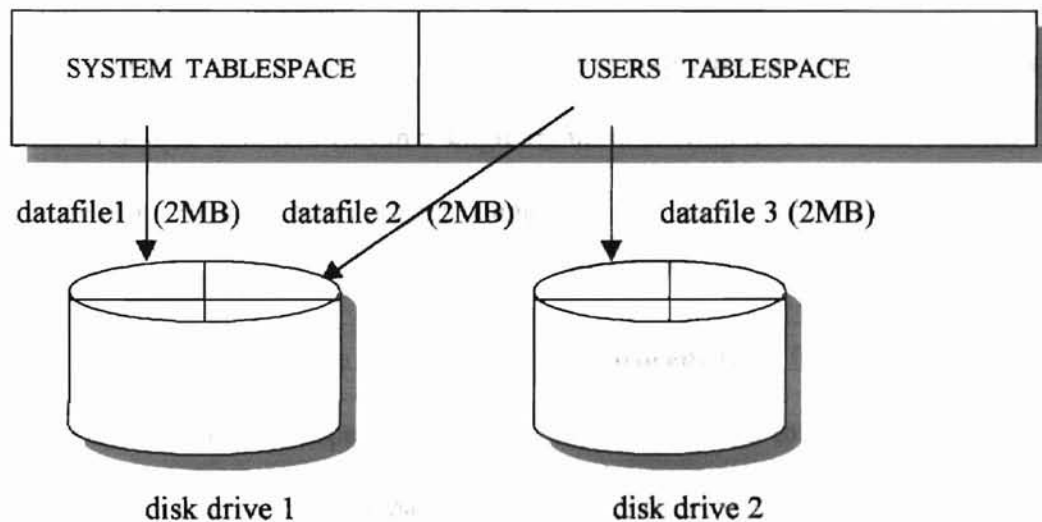


Figure 1. Tablespaces in Oracle

The third major schema object in Oracle is **view**, a customized presentation of the data in one or more tables. Views do not actually contain or store data. They derive their data from the tables on which they are based, referred to as the base tables. Like tables, views can be queried, updated, inserted into, and deleted from, but with some restrictions. All operations performed on a view actually affect the base tables of the view.

The view provides an additional level of table security by restricting access to a predetermined set of rows and columns of table. For instance, a view of table can be created so that the columns with sensitive data, such as SSN, are not included in the definition of the view. The view can hide data complexity. For example, a single view can be used to create a join, which is a display of related columns or rows in multiple tables. However, the view hides the fact that this data actually originates from several tables. Therefore the view can simplify commands for user, present the data in a different perspective from that of the base table, and store complex queries.

The fourth major schema object in Oracle is **sequence**; a sequence generates a serial list of unique numbers within a single numeric column of a database's tables. Sequences simplify application programming by automatically generating unique numerical "labels" for the rows of tables.

The fifth major schema object in Oracle is **procedure** or **function**, a set of SQL and PL/SQL statements grouped together as an executable unit to perform a specific task. ORACLE was the first company to release a product that used the SQL [OC, 1998]. Procedures and functions allow user to combine the simplicity and flexibility of SQL with the procedural functionality of a structured programming language. Using PL/SQL, such procedures and functions can be defined and stored in the database for reuse.

The sixth major schema object in Oracle is **package**, a method of encapsulating and storing related procedures, functions, and other package constructs together as a unit in the database. Packages provide database administrator or developer organizational benefits, as well as increased functionality and database performance.

The more detail about procedure/function/package will be discussed in next "SQL and PL/SQL" section.

The other major schema objects in Oracle are **synonym**, **index**, and **cluster**. The synonym is an alias for a table, view, sequence, or program unit. The index and cluster are optional structures associated with tables, which can be created to enhance the performance of data retrieval.

The data types in Oracle are listed below.

- CHAR: stores fixed length character strings.
- VARCHAR2: stores variable-length character strings.
- NUMBER: stores fixed and floating-point numbers.
- DATE: stores point-in-time values in a table.
- ROWID: unique ROWID corresponds to the physical address of a row's row piece. (row piece is unique identifier for rows in a given table.)
- LONG: store variable-length character data containing up to 2 GB
- RAW and LONG RAW: for data that is not to be interpreted by ORACLE.

Intended for binary data or byte strings.

2.2 Structured Query Language (SQL) and PL/SQL

SQL is the standard language used to communicate with a relational database. Industry accepted SQL standard, first introduced by the American National Standards Institute (ANSI) in 1986, current SQL92 by ANSI and the International Standard Organization (ISO), new standard SQL3 with enhancements in object-oriented data management is undergoing review now [BM, 1998]. Practically, there are numerous

SQL implementations that are released by various vendors, and SQL is portable to all RDBMS systems.

SQL is a very simple, yet powerful database access language, it is a non-procedural and interpretive language. With SQL, the user specifies what information is needed and the system determines how to retrieve it. Languages such as these are referred to as Fourth Generation Languages. Using SQL in Oracle, we can create tables in the database, store information in tables, select exactly the information needed from database, make changes to data and the structure of underlying tables, and combine and calculate data to generate the information needed [KGLK, 1995].

Oracle SQL commands have six classes as below [DCDH, 1997].

1. Data Manipulation Language statements (DML): *select, from, where.*
2. Data Definition Language statements (DDL): *create table, drop table, alter table, create view etc.*
3. Transaction Control statements
4. Session Control statements
5. System Control statements
6. Embedded SQL statements

There are several ways to access and manipulate database objects in Oracle.

Figure 2 shows SQL, PL/SQL, OCI, Pro*C and OraPERL [CGOCGC, 1997]. Although some Oracle tools (PL/SQL, Pro*C, OraPERL) and applications simplify or mask the use of SQL, all database operations are performed using SQL.

Oracle SQL has many extensions to the ANSI/ISO standard SQL language; one of them is PL/SQL, Procedural Language extension of SQL. PL/SQL is an application

development language. While SQL is the basic data access language in Oracle that all other Oracle products and tools use, including PL/SQL engine. PL/SQL allows all the DML statements, cursor operations and the transaction processing statement in SQL. PL/SQL is a superset of SQL, allowing one to specify both what to do and how to do it.

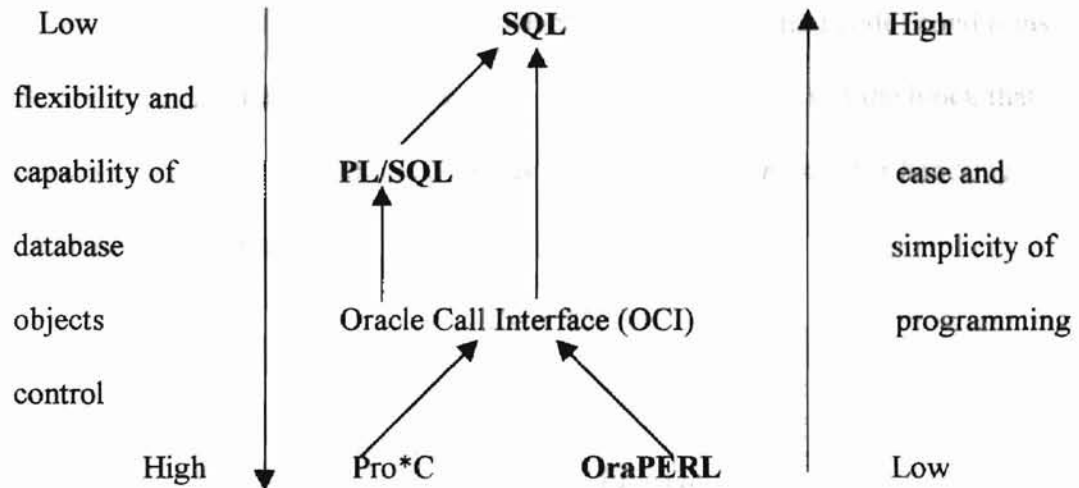


Figure 2. SQL, PL/SQL, OCI, Pro*C and OraPERL in Oracle.

PL/SQL has all procedural language features. The Advantages of PL/SQL are:

1. PL/SQL improves performance of database management because its code is stored and shared in an RDBMS server.
2. PL/SQL enhances productivity since it allows modularity and software reuse.
3. PL/SQL has portability because it is across all Oracle platforms.
4. PL/SQL has "seamless" integration with the RDBMS server through the procedural option:
 - Oracle allows several other options
 - Distributed and parallel servers, parallel query, context server, media server, web server.

The architecture of the PL/SQL "Engine" is described in Figure 3 [CGOCGC, 1997]. There are three sections, DECLARE, EXECUTABLE and EXCEPTION, to a PL/SQL block, the EXECUTABLE section is mandatory, the other two are optional. The first section is DECLARE, which contains variables, cursors, and constants. The second section is called EXECUTABLE, which contains the actual code (conditions and SQL statements) that the block executes. This is the only part of the block that must always be present. The third section is EXCEPTION, which is for handling runtime errors and warnings.

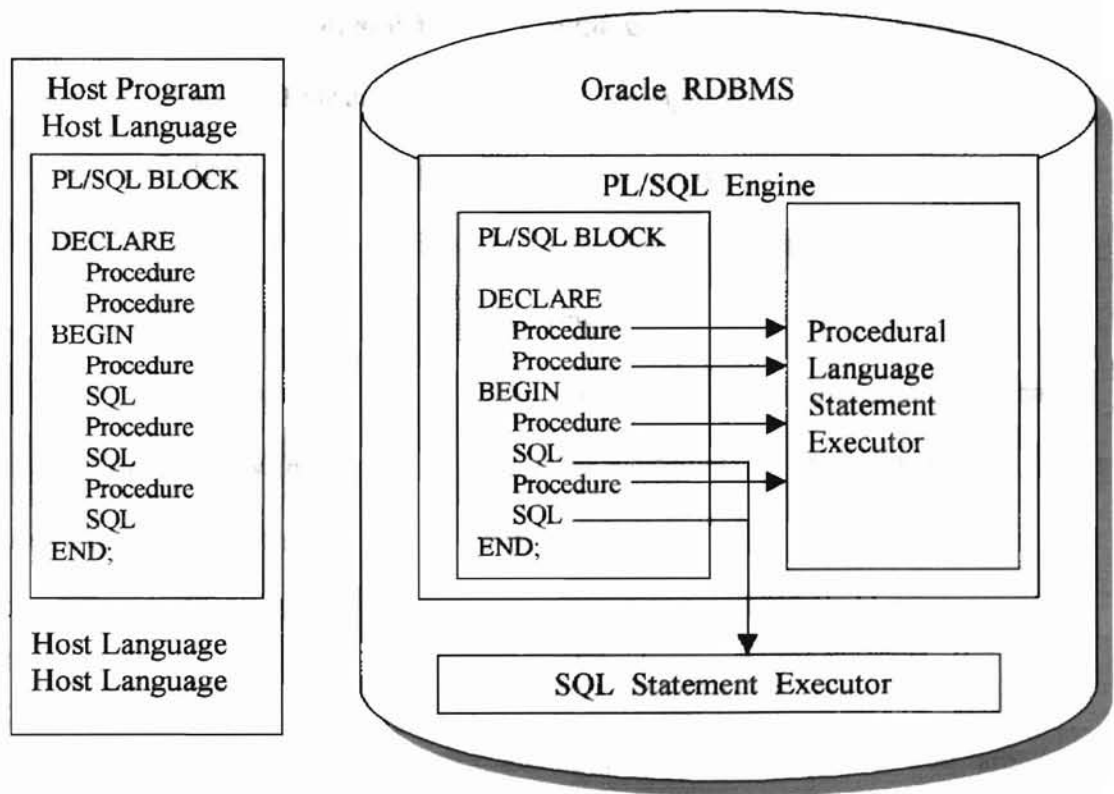


Figure 3. Architecture of the PL/SQL "Engine".

The types of program units in PL/SQL are the PL/SQL Block, Stored Procedures, Stored Functions and Stored Packages. The PL/SQL Block is an unnamed

PL/SQL procedure that groups SQL statements and PL/SQL constructs into a single program unit. The Stored Procedures means a PL/SQL block stored in the database and called by name from applications. The Stored Functions is identical to procedures except functions always return a single value to the caller while procedures do not. The Stored Packages is called a method to encapsulate and store related procedures, functions, variables, and other package constructs together as a unit in the database [KGLK, 1995].

2.3 Practical Extraction and Report Language (PERL) and OraPERL

PERL, Practical Extraction and Report Language, was originally created by Larry Wall in 1987 as a utilitarian scripting tool mainly for string manipulations [WLSR, 1991, EW, 1999]. Over the years, it has evolved into a sophisticated programming language capable of implementing object-oriented programming [SESSPN, 1999]. One of the greatest expansions of PERL is the addition of database-accessing functions. With over one million users worldwide, PERL has become the first choice of most Unix CGI scripts for WWW development, text processing, and Internet services [HJSR, 1998].

The database access functions were organized into PERL packages and added to the PERL system as modules, classes, and library files. Packages in PERL are declarations to partition global namespaces. The package names are added to global identifiers as prefix during compilation. The effect of a package declaration is terminated by a subsequent package statement or the end of the current scope, such as the end of a file. A module in PERL is a package defined in a file whose name is the

same as the package. The required file extension for a PERL module is ".pm". A PERL class is indeed a module that contains a special subroutine to create object. A library file contains a collection of loosely related functions designed for use by other programs. It lacks the rigorous semantics of a PERL module and has ".pl" the file name extension. The use statement followed by a module or file name makes the functions defined in the module or file available to a program.

PERL's database interfacing modules are organized into two groups: one is the database interface (DBI) group and the other is the database driver (DBD) group. DBI is a generic interface for many databases, which means that one can write a script that works with many different database engines without change. A DBD is defined for an individual database system and is usually developed by vendors of the database system. To use PERL program to access a database system, one must secure the database driver and installed into the PERL system.

OraPERL is an extension of PERL, which is PERL with built-in functions to access Oracle database using a PERL script. OraPERL is built by extending the PERL Interpreter through the sub-functions of user information, and built on top of the Oracle Call Interface (OCI) Driver Layer (see Figure 2). OCI itself is a client side program consisting of a library of C language routines to allow C (or other third generation language) programs to send SQL statements to a DBMS. OraPERL is also public domain software. The synopsis of major built-in functions and variables in OraPERL are described as below.

The main functions for database accesses are following [PC, 1995].

- *\$lda = &ora_login(\$system_id, \$name, \$password)*: Logs into the specified database with the system ID, username and password given. Default for \$system is \$ENV{'ORACLE_SID'}. Returns an *\$lda* for use with *&ora_open()*.
- *\$csr = &ora_open(\$lda, \$stmt [, \$cache])*: Associates the given an SQL statement with the database identified by *\$csr*. Executes it if it contains no substitution variables. Creates a fetched row cache of given size (default *\$ora_cache*). Returns a *\$csr* for use with *&ora_fetch()*.
- *&ora_bind(\$csr, \$var,...)*: If a SQL statement contains substitution variable, *&ora_bind()* is used to assign the actual value to them. This function binds the given values to the substitution variables in the SQL statement associated with *\$csr*, and executes the resulting statement.
- *\$n = &ora_fetch(\$csr [, \$strunc])*: This function is used in conjunction with a SQL SELECT statement to retrieve information from a database. It takes one mandatory parameter, a statement identifier (obtained from *&ora_open()*). It returns the number of fields available from the query.
- *@ary = &ora_fetch (\$csr[, \$strunc])*: Retrieves the (next) output data from the statement identified by *\$csr*. LONG data may be truncated if *\$strunc* is non-zero.
- *&ora_close(\$csr)*: Releases the cursor identified by *\$csr*.
- *&ora_logoff(\$lda)*: Logs out of the database identified by *\$lda*.
- *&ora_do(\$lda, \$stmt)*: Equivalent to *&ora_close(&ora_open(...))*.

OraPERL provides seven special variables; three of them are used to dictate the behavior of Oracle under certain conditions [OC, 1997]:

- *\$ora_cache*: This variable determines the default cache size used by the *&ora_open()* function for SELECT statements if an explicit cache size is not given.
- *\$ora_long*: Normally, OraPERL interrogates the database to determine the length of each field and allocates buffer space accordingly. This is not possible for fields of LONG or LONGRAW. To allocate space according to the maximum possible length (65535 bytes) would obviously be extremely wasteful of memory. Therefore, when *&ora_open()* determines that a field is a LONG type, it allocates the amount of space indicated by the *\$ora_long* variable. This is initially set to 80 (for compatibility with OraPERL products) but may be set within a program to whatever size is required.
- *\$ora_trunc*: Since OraPERL cannot determine exactly the maximum length of a LONG field, it is possible that the length indicated by *\$ora_long* is not sufficient to store the data fetched. In such a case, the optional second parameter to *&ora_fetch()* indicates whether the truncation should be allowed or should provoke an error. If this second parameter is not specified, the value of *\$ora_trunc* is used as a default. This only applies to LONG and LONGRAW data types. Truncation of a field of any other type is always considered an error.

Generally, with these functions and modules, establishment a connection to a database system with PERL need through five steps:

1. Create a database handle;
2. Setup a statement handle;
3. Catch the feed back or query reset the statement handle;
4. Free statement handle;

5. Free database handle.

2.4 Standard Generalized Markup Language (SGML) and Hyper Text Markup Language (HTML)

The Standard Generalized Markup Language (SGML) is an international standard for defining document structure [ISO 8879, 1986]. Since its adoption in 1986, it has gained widespread acceptance by industry, government, and independent agencies [SGML '94 Conference Proceedings, 1994], and it has entered the mainstream of business computing; Microsoft Word, WordPerfect, and Interleaf now have facilities to import and export SGML [RRCG, 1994].

There are two features that distinguish SGML from other text processing systems:

- It is based on descriptive markup that separates the logical components of the document as opposed to procedural markup that defines the physical appearance of the document.
- It is a meta-language system for document definition rather than a specific markup scheme for document processing.

Almost any kind of document structure can be defined using SGML [MRGKHG, 1997]. The structure of an SGML-coded document, and details of optional SGML features used in its preparation, are formally defined in a set of markup declarations that form a document type definition (DTD). These markup declarations describe a set of markup instructions, known as tags, which can be used to identify the start or end of the logical elements of the text. The start of each element is marked by a start-tag; an end-tag is normally used to indicate where the element ends.

Where necessary SGML markup tags can be qualified by attributes. Attributes are used within SGML to [BM, 1998]:

- identify specific tags uniquely;
- cross-refer to elements identified by unique identifiers;
- recall externally stored data;
- indicate the role of an associated element.

Attributes can also allow users to control the way in which text is presented to readers.

SGML also provides ways of declaring short cuts to document markup. The amount of markup that needs to be entered or transmitted can be reduced by [BM, 1998]:

- omitting markup tags that can be implied by the use of other tags;
- using special short forms of tags;
- using shorthand references to identify entities that contain markup tags.

These techniques make it possible to reduce to a minimum the amount of markup required in an SGML-coded file.

Users of SGML systems may hardly notice the difference between their existing word processors and SGML-based programs. Both may use the same sequences of key or button depression to enter and format the text. The main difference will be that the set of tags/buttons that are permitted/active at a particular point in an SGML document will be restricted to the set that is defined for the containing element in the document type definition. This will mean, for example, that it will no longer be possible to place a third-level heading directly under a first-level heading if the document type definition requires there to be an intervening second-level heading.

Word processors can import and format SGML-coded documents if they have a program that is capable of converting SGML markup into a form that can be understood by the formatting program used by the word processor. For generalized SGML documents this can be a difficult process. There is, however, one particular application of SGML that is specifically designed to make it as easy as possible to convert word-processed text into and out of SGML: the HyperText Markup Language (HTML) used on the WWW.

When Tim Berners-Lee created the first HTML browser at CERN, he was not designing an SGML system. Initially, he wanted a mechanism that would describe the processes going on within his browser in a form that could be safely transmitted over the Internet. As the SGML developers had found, the safest code set for the transmission of information between computer systems is that defined in ISO 646, which formally defines an International Reference Version (IRV) of the code set originally created by the ANSI as the ASCII. To delimit his markup instructions from the text Tim chose the same delimiters as SGML, the characters `<and>`. His initial markup instruction set included things like end of paragraph `<p>`, italic `<I>` and bold ``. To end italic and bold text strings Berners-Lee chose to use end delimiters of the same form as SGML, `</I>` and `` [BM, 1998].

While at first glance this initial coding scheme looked like SGML, it was apparent to those who knew SGML that there were fundamental differences between the concepts behind HTML and those behind SGML. In particular, the role of the `<p>` tag was fundamentally different. In HTML this tag initially only indicated the point at which a paragraph end was required. In SGML this tag indicates the start of a

paragraph, with a matching `</P>` tag identifying the end of the paragraph, where the actual paragraph break occurs [BM, 1998].

Another fundamental difference was that HTML originally had no control on when one should switch bold and italic on and off. As with many word processors, there was nothing to stop users from switching on bold in the middle of one paragraph and then switching on italic at the next. In such cases the first part of the second paragraph would be presented in italic. If bold was then switched off the text would continue to be presented in italic until a command to switch off italic was received.

The problem with this approach is that user could not be sure that all HTML document browsers would work in exactly the same way. Some might choose to automatically switch bold and italic off when they started a new paragraph. This led to the same document providing different results in different browsers.

The philosophical differences between HTML and SGML were resolved with Version 2.0 of HTML, when it was decided to use an SGML document type declaration to formalize HTML so that restrictions could be placed on where each of the HTML elements could be started and ended. The formal definition made it clear that it was no longer permissible to end. The formal definition made it clear that it was no longer permissible to extend formatting instructions over paragraph boundaries. It also introduced logic equivalents for formatting-related instructions. For example, emphasis (``) was introduced to replace italic (`<I>`) and `` was introduced to replace bold (``) [BM, 1998].

HTML is not as well controlled structurally as most SGML document type definitions. It is still more presentation-oriented than structurally ordered. Version 3.2

of the standard, which was formally agreed in January 1997, introduced some additional structural elements, including one that can be used to arbitrarily group sets of elements. It is likely that the trend of introducing a greater range of logical elements will develop over time [W3C, 1997].

HTML is very easy to map word-processing software. At its simplest level it can be looked at as an SGML representation of Rich Text Format (RTF). HTML is an ideal way of getting users of existing word processors to start to use generic markup tags because it can be introduced into existing word processors with very few changes to existing document creation processes.

HTML is an evolving language, and each new version is given a number. The versions of HTML include:

- HTML 2.0 (1994) had most of the elements we know and love, but missed some of the Netscape/Microsoft extension, and did not support tables, or ALIGN attributes [W3C, 1997].
- HTML 3 (late 1995) was an ambitious effort on the part of Dave Raggett to upgrade the features and utility of HTML. However, it was never completed or implemented, although many features were integrated in the next “official” version of HTML, known as HTML 3.2 [GI, 1998].
- HTML 3.2 (June, 1996) was the next official version, integrating support for TABLES, image, heading and other element ALIGN attributes, and a few other finicky details. HTML 3.2 is the current "universal" dialect: essentially all browsers understand HTML 3.2. IT, however, missed some of the Netscape/Microsoft extensions, such as FRAMEs, EMBED and APPLET [W3C, 1997, NB, 1998].

- HTML 4.0 (Dec.,1997) is the current official standard. It includes support for most of the proprietary extensions. It also support extra features (Internationalized documents, support for Cascading Style Sheets, extra TABLE, FORM, and Java Script enhancements), that are not universally supported [W3C, 1997, NB, 1998].

CHAPTER III

Design

This chapter deals with the design issues of the SWFAL system. The chapter starts with the overall design of the system, then includes design of the specific modules of the system. In the end of this chapter, the security of this system is discussed.

3.1 Overall SWFAL System Design

On a traditional network, using Microsoft Networking or Novell NetWare, the server knows which clients are logged in and connected at all times. This knowledge can be passed on to applications running on the server, so it can provide data caching and other services to that client as required [HJSR, 1998].

In the context of a desktop database application, the following diagram (Figure 4) broadly demonstrates how the persistent connection between the client and server makes it possible to provide cached dynamic and updateable record sets on the client.

SWFAL system is a web-page system. The client has suitable software (Common Gateway Interface: CGI) installed to negotiate with the data service software component (Oracle manager) on the server on an ongoing basis, through a connection to a particular data store on the server. The server's data service software provides all or part of the results to the client where it is cached locally, and the interface can then display the relevant data. The client cache is automatically filled working with the data.

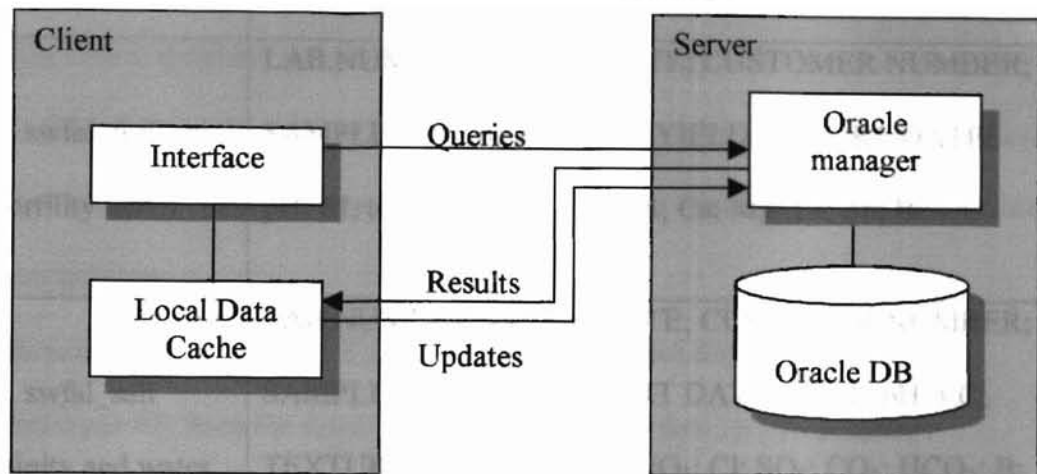


Figure 4. The client - server environment in the SWFAL system.

The client software can also pass data uploads to the Oracle manager on the server, and instruct it to update the original data. The flow chart of systems from the client to the server and back can be depicted shown in Figure 5.

3.2 Oracle Database Tables Design

The data collection for SWFAL project was analyzed and then database normalization techniques were used to design the database schema so as to avoid any replication and ensure data integrity.

The creation of tables for the project can be divided by four objects, which are fertility, water & salinity, forage, and texture tables. The name and attributes of each table are listed in the following.

Table Name	Attributes
swfal_fert (fertility table)	LAB.NUMBER; LOGIN DATE; CUSTOMER NUMBER; SAMPLE NUMBER; CROP; YIELD; REPORT DATE; pH; BI; top_N; sub_N; P; K; S; Ca; Mg; Fe; Zn; B.
swfal_sali (salinity and water quality table)	LAB.NUMBER; LOGIN DATE; CUSTOMER NUMBER; SAMPLE NUMBER; REPORT DATE; TEST; pH; EC; TEXTURE; Na; K; Ca; Mg; NO ₃ ; Cl; SO ₄ ; CO ₃ ; HCO ₃ ; B; TSS; PAR; SAR; EPP; ESP.
swfal_fora (forage table)	LAB.NUMBER; LOGIN DATE; CUSTOMER NUMBER; SAMPLE NUMBER; CROP; REPORT DATE; NITRATE; ADF; NDF; MOISTURE; PROTEIN; TDN; NEMAIN; NELACT; NEGAIN.
swfal_text (texture table)	LAB.NUMBER; LOGIN DATE; CUSTOMER NUMBER; SAMPLE NUMBER; REPORT DATE; TEXTURE; SAND; CLAY; SILT.

Table 1. Attributes of SWFAL tables.

3.3 CGI Interface Design

The SWFAL system uses popular browsers as a front end to provide interaction between the system and user. User inputs are accepted using HTML forms. These inputs are then passed on to the web server using Internet as medium for data transfer.

The web server then uses CGI protocol to relate these user inputs to programs that will query the Oracle database and perform calculations. Finally, CGI protocols are used to transfer the results of the calculations (report) back to the user. The output is displayed in the browser in various formats. The flow chart of CGI interfaces in SWFAL system is shown in Figure 6.

Functionality requirement of CGI program for upload data

1. Read input file from the form in html file that is invoked by this program.
2. Add the information of data file into Oracle tables.
3. Handle file, which is the message of upload successful or failed, upload from browser.
4. Generate four type reports according to input data file:
 - a. distinguish four report title names according to the uploading file.
 - b. get deferent user information for each sample from input.
 - c. print out the corresponding table's item names and values, which is the testing result part of report.
 - d. calculate and print out the interpretation and requirement part of report.

Functionality requirement CGI program for retrieval data

1. Get the information from the form in html file that is invoked by this program.
2. Add the host and user names of a visitor to the log file.
3. Get the database records according to the form information.
4. Error handle: when there is an error, print the error message and exit this program.

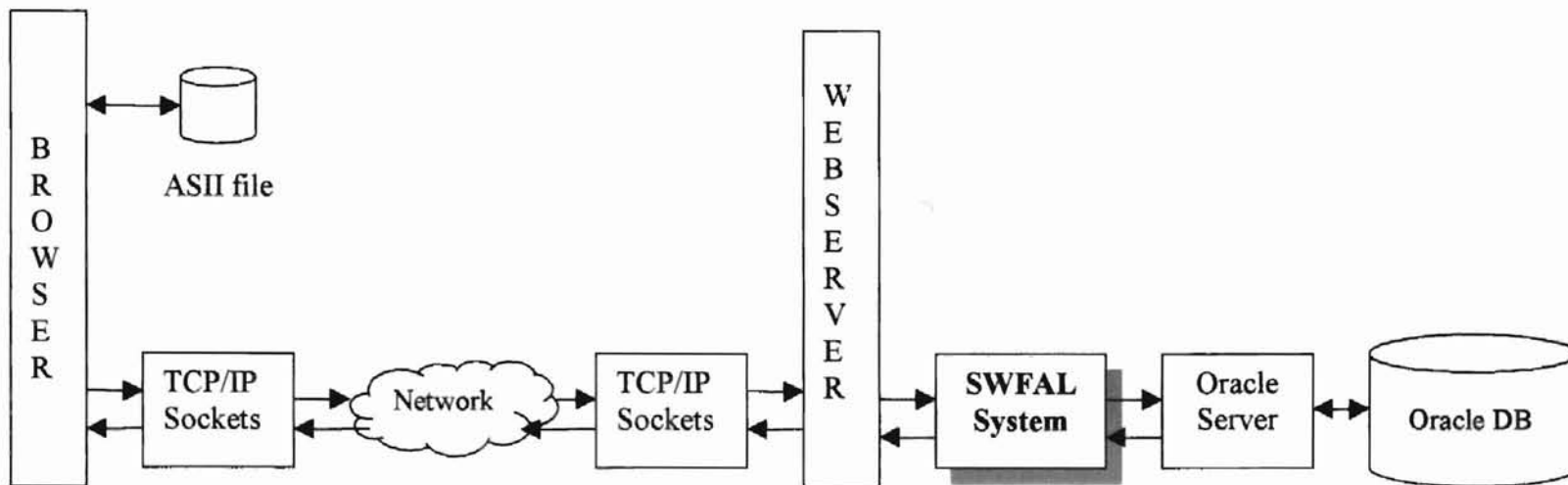


Figure 5. Overall of the SWFAL system

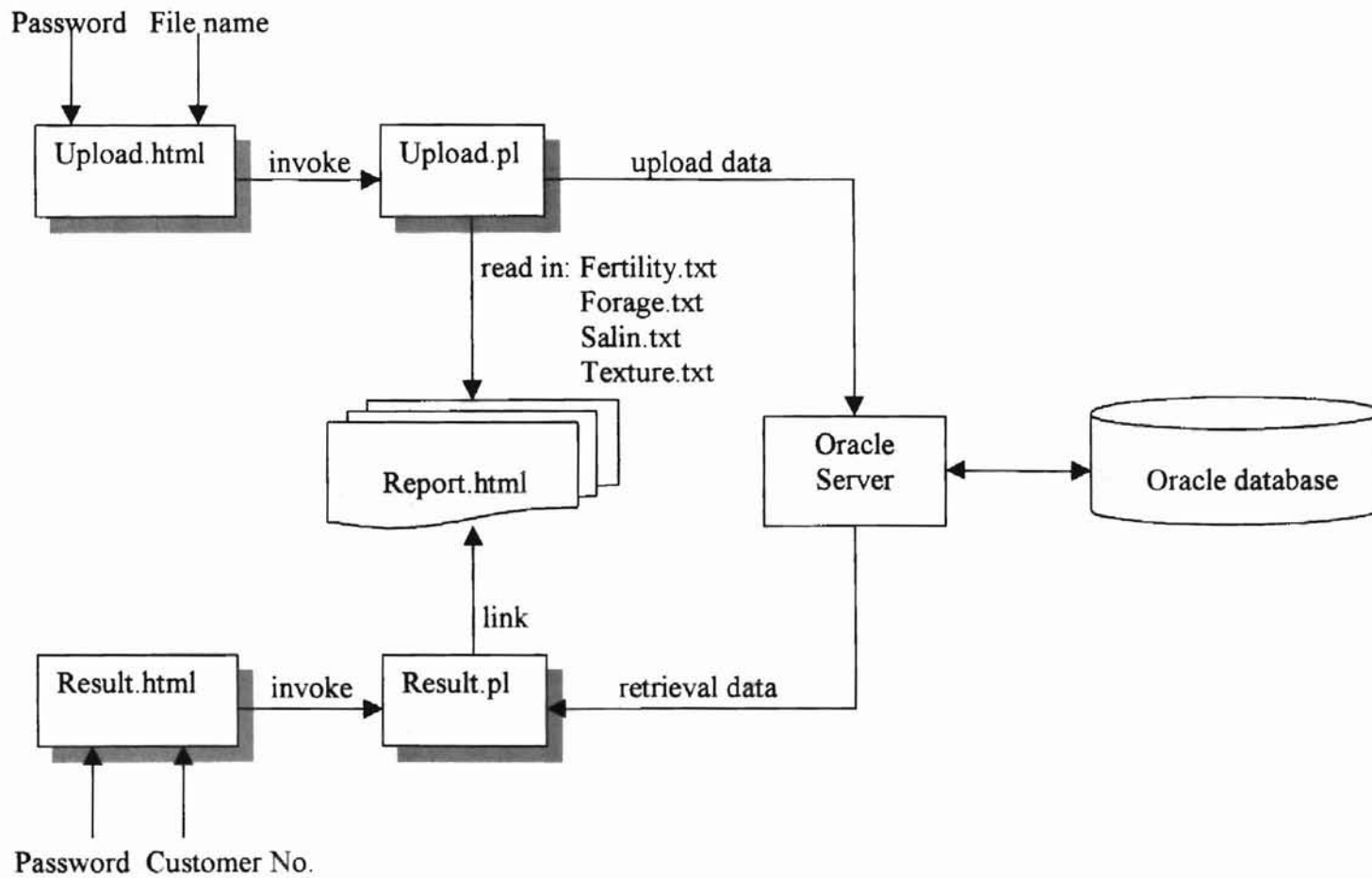


Figure 6. The relation chart of CGI interface objects in SWFAL system

3.4 Web-page Design

User interface design deals with designing HTML forms and tables, which are easy to use and provide maximum interactivity with the user. Care is taken to perform maximum data validation at the client end so as to reduce the dependence on the server. The forms designed for accepting the user inputs as follows:

Upload data web page

- Requested user input: Password, File name;
- Functional icon: Upload the file, Reset values;
- Functionality requirement:
 1. When upload file successful, show the “File Transfer” page, which includes file name, file size, location, local time, and the line(s) this file in the Oracle table.
 2. The “File Transfer” page has “go back” and “new search” icons that help user to do new upload.
 3. If upload file failed, show the error message.

Retrieval data web-page

- User input (requested): Password, Customer Number;
- Data type input (multiple choices): All the data; One sample number;
- Table specification (multiple choices): User defined from ... to; Most recent day;
Most recent 2 days; Most recent 3 days;
- Functional icon: Start search, Reset values;
- Functionality requirement:
 1. When retrieval data successful, show the “record” page, which lists all records of user selected type in Oracle database.

2. The all records listed in record page links all the corresponding types of reports, user can treat the Lab # as a functional icon that will go to the “report” page.
3. The “record” page has “go back” and “new search” icons that help user to do new retrieval.
4. The “record” page has a “spread sheet” icon that shows the spread sheet format of records.
5. The “record” page has an “interpretation” icon that links the other interfaces focus on the interpretations and nutrients needed.
6. If retrieval data failed, show the error message.

3.5 Security Design

The current application is quite robust and secure from network hackers, it relies on the security provided by the UNIX system and Oracle Database Security features to protect its data storage. On the accessing web page, the password file provides the security. Every time, when users uploading or accessing, the web page asks the passwords of users, only if it matches on a string in the password file, the next operation will take place, otherwise, a wrong password message will show up.

CHAPTER IV

Implementation

This chapter deals with the implement details of the SWFAL system. The chapter starts with implement Oracle data management, then includes the implement of CGI interface and web page.

4.1 Oracle Data Management

4.1.1 Creating Tablespaces

Oracle stores table data and other database management related data in data files. A *tablespace* is a group of disk files containing logically related database objects. For this project, an independent tablespace called SWFAL is created using the following SQL code.

```
CREATE TABLESPACE SOILSID  
DATAFILE: /usr/DISK4/oracle/soilsid1.dbf SIZE 50M ONLINE
```

4.1.2 Creating Tables

Oracle stores the data in the form of tables. The tables for the SWFAL are created using SQL scripts. An example of which is below:

```
CREATE TABLE swfal_fert  
sLabNum char(10) NOT NULL,  
sLogindate char(10), NOT NULL,
```


sCustomNum char(10), NOT NULL,
sSampleNum char(10), NOT NULL,
sCrop char(10), NOT NULL,
sYield char(10),
sYield char(10),
sReportDate char(10),
sPH char(10),
sBI char(10),
stop_N char(10),
ssub_N char(10),
sP char(10),
sK char(10),
sS char(10),
sCa char(10),
sMg char(10),
sFe char(10),
sZn char(10),
sB char(10)

Name of the table to be created is followed by the key words CREATE TABLE. The columns in the table are specified with the name of the column, the data type of the column along with the maximum precision for the column and the keyword NULL or NOT NULL, indicating whether the particular column is allowed to hold a null value or not.

The PRIMARY KEY keyword specifies which columns in the table uniquely identify a particular row in the table. Even if a table does not have a primary key, an index can be created on the table so as to make the data retrieval from the table faster.

Storage parameters for ignored in the above example, in which case Oracle uses its default storage characteristics.

4.2 CGI Interface

The implementation of CGI interface includes two programs – upload.pl and result.pl. To meet the functionality requirement upload.pl will invoke the upload file web page and generate the “File Transfer” page and the corresponding report html form. Also, result.pl will invoke the “dbresult” web page and generate the “records” page, which links report page, “spread sheet” page, and “interpretation” page.

4.2.1 Structure of upload.pl

The general structure of upload.pl is showed in Figure 7, the detail of generate report part is showed from Figure 8-1 to Figure 8-8, because it is more complex.

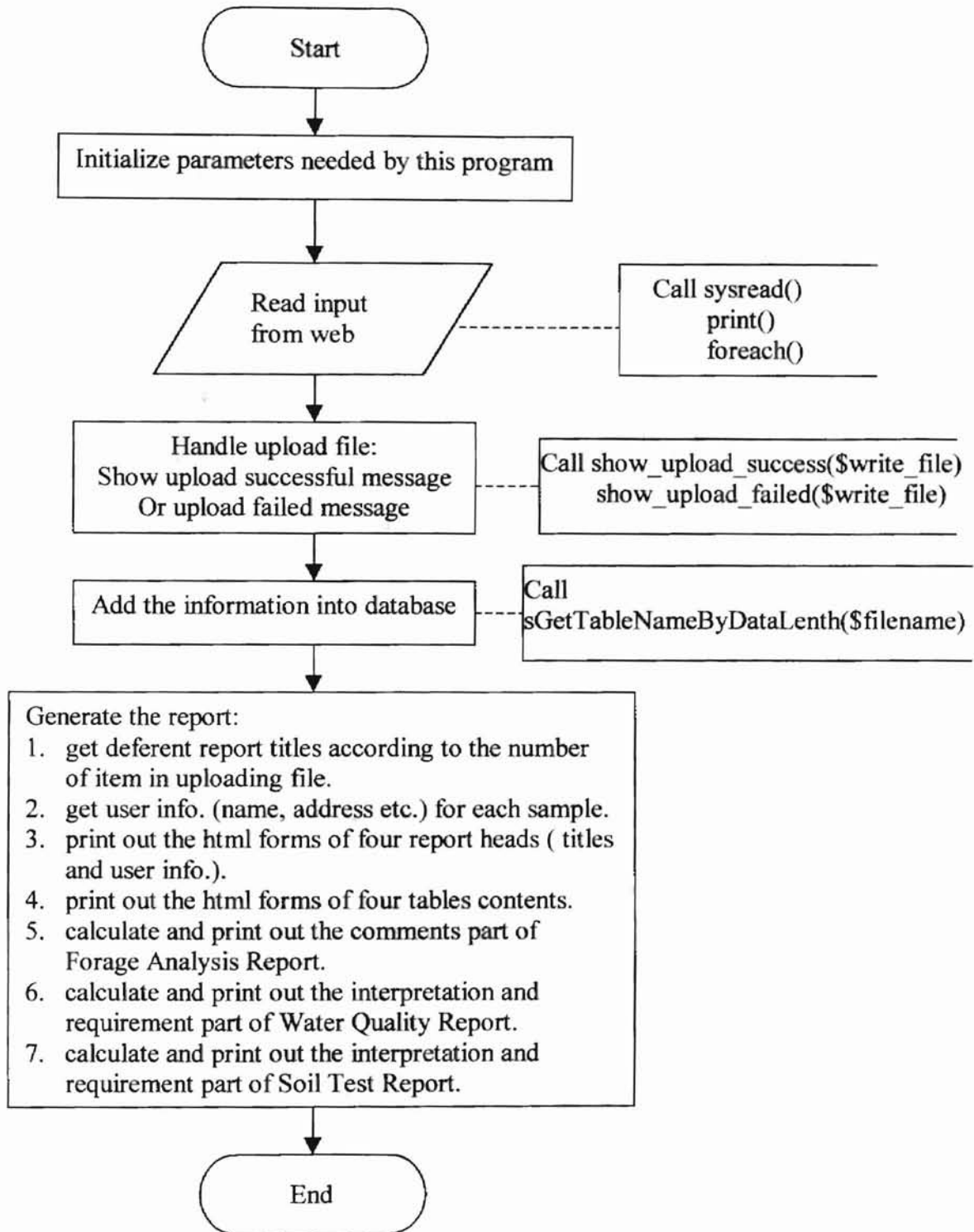


Figure 7. The general flow chart of upload.pl

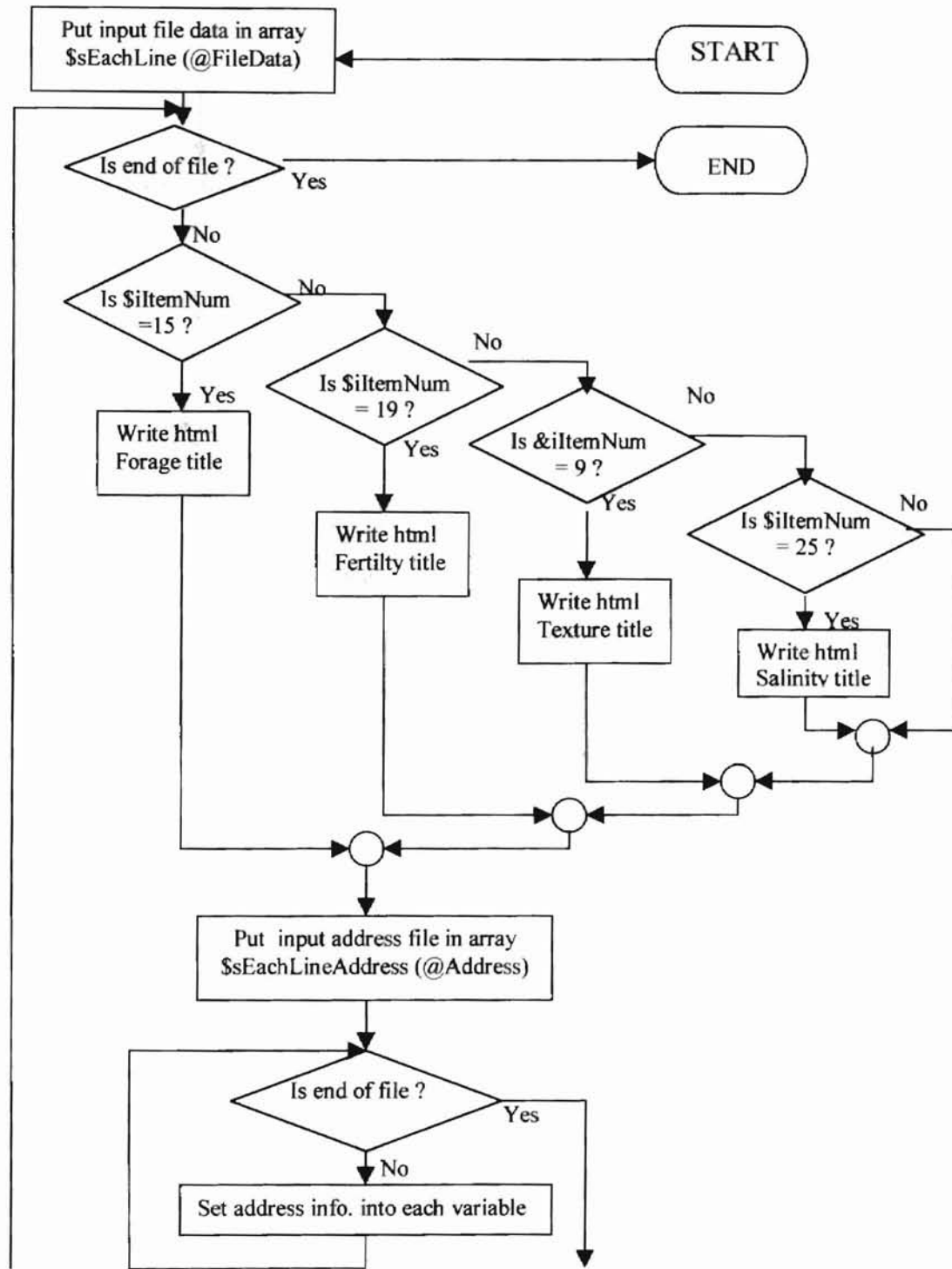


Figure 8-1. The detailed flow chart of generate report.

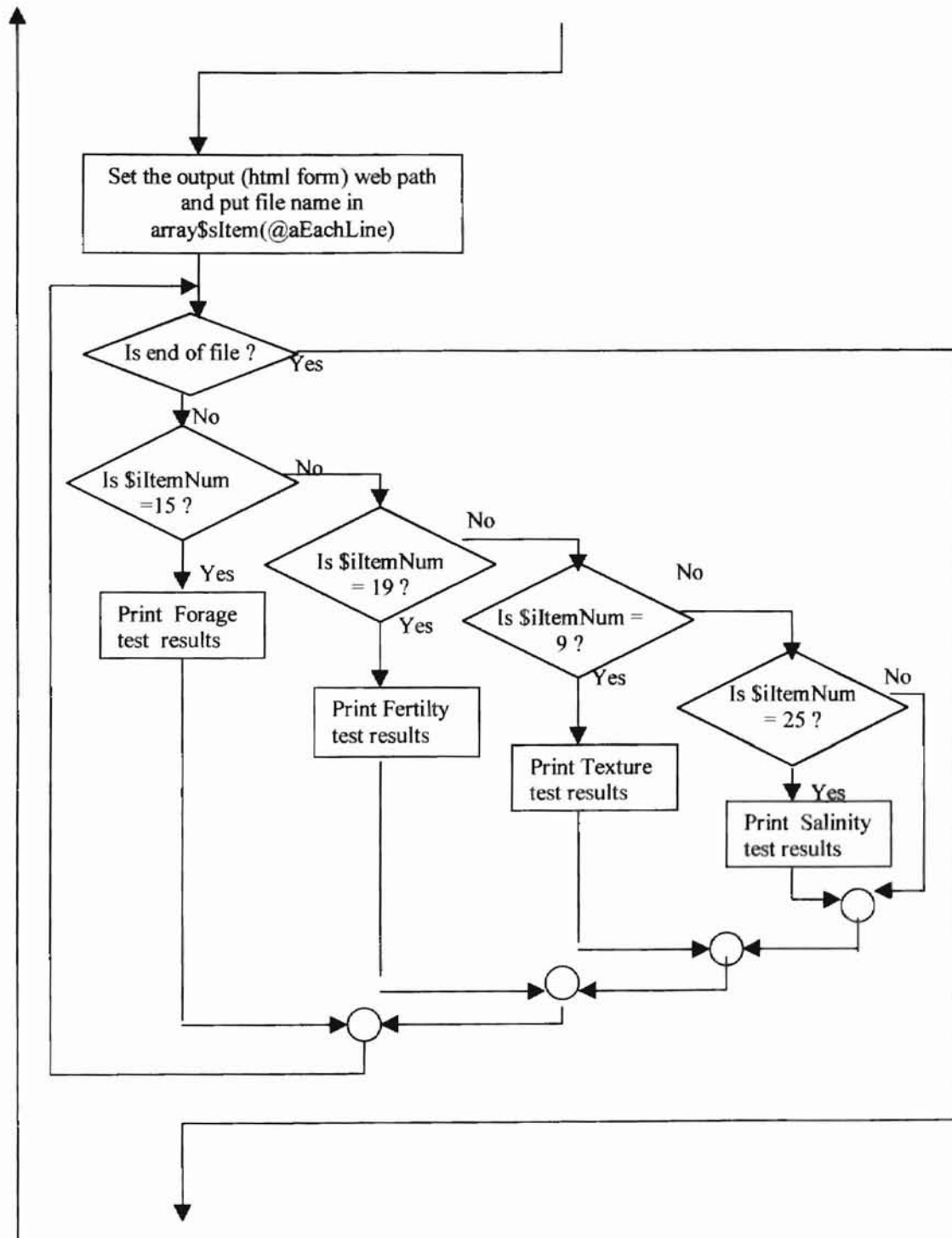


Figure 8-2. The detailed flow chart of generate report (continued).

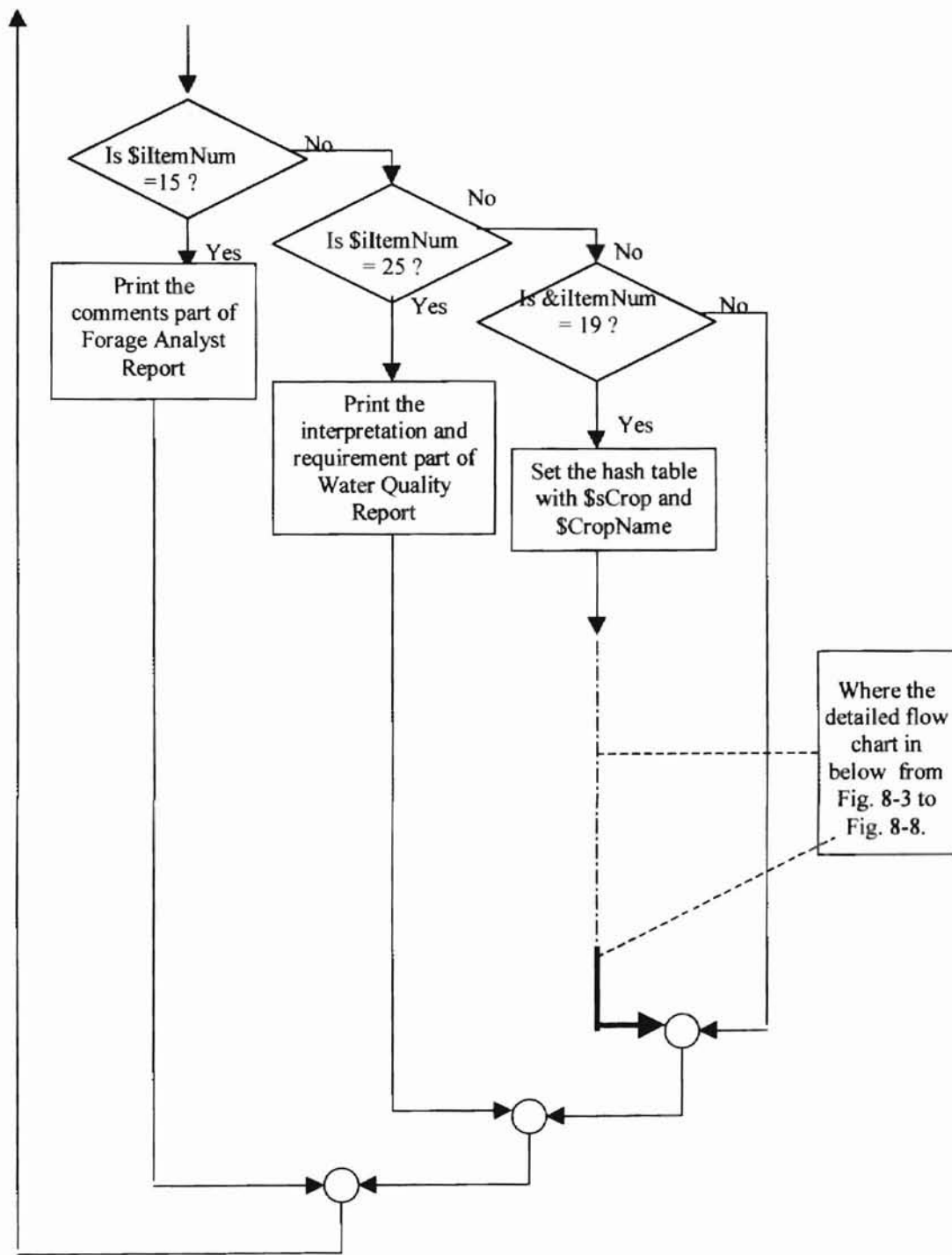


Figure 8-3. The detailed flow chart of generate report (continued).

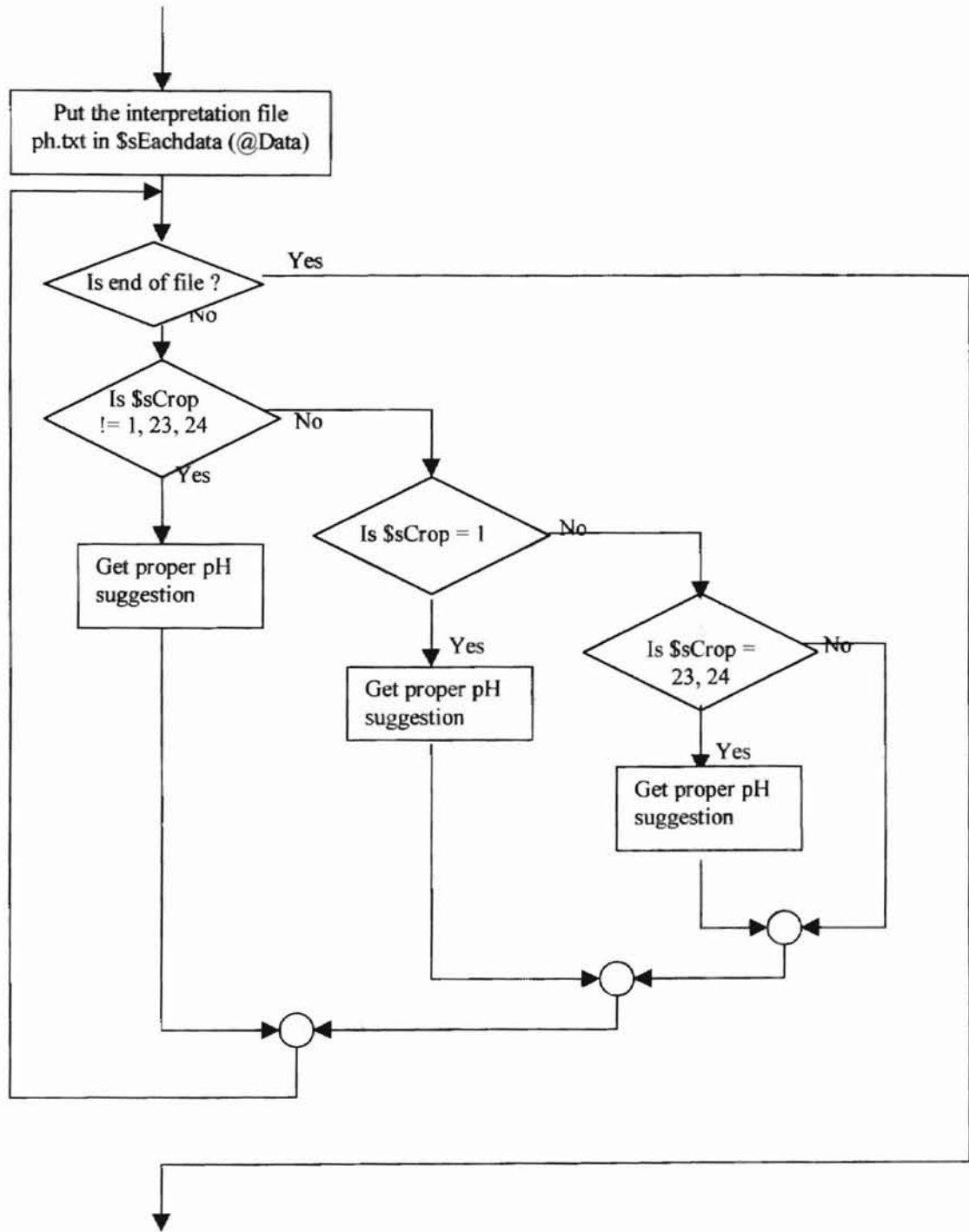


Figure 8-4. The detailed flow chart of generate report (continued).

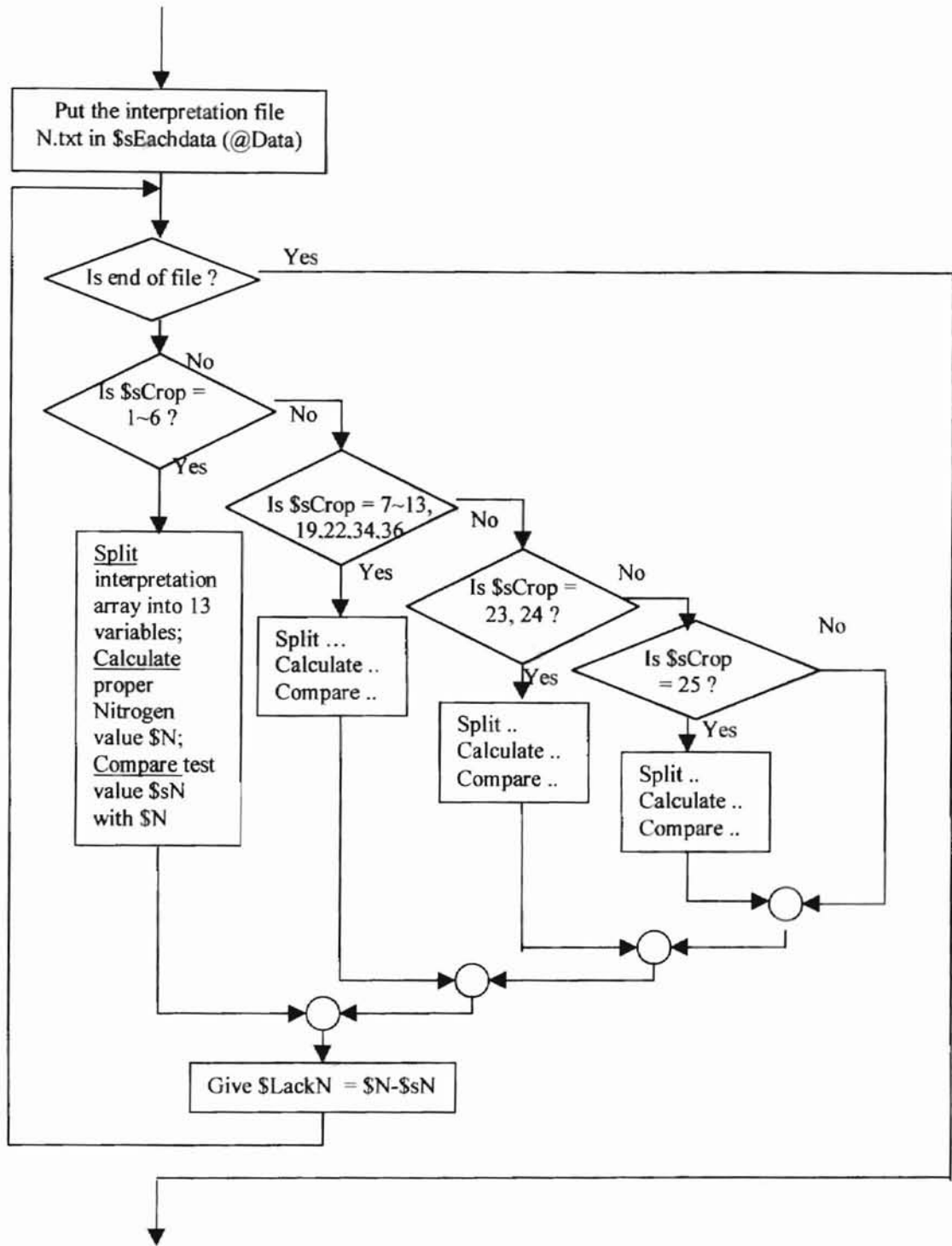


Figure 8-5. The detailed flow chart of generate report (continued).

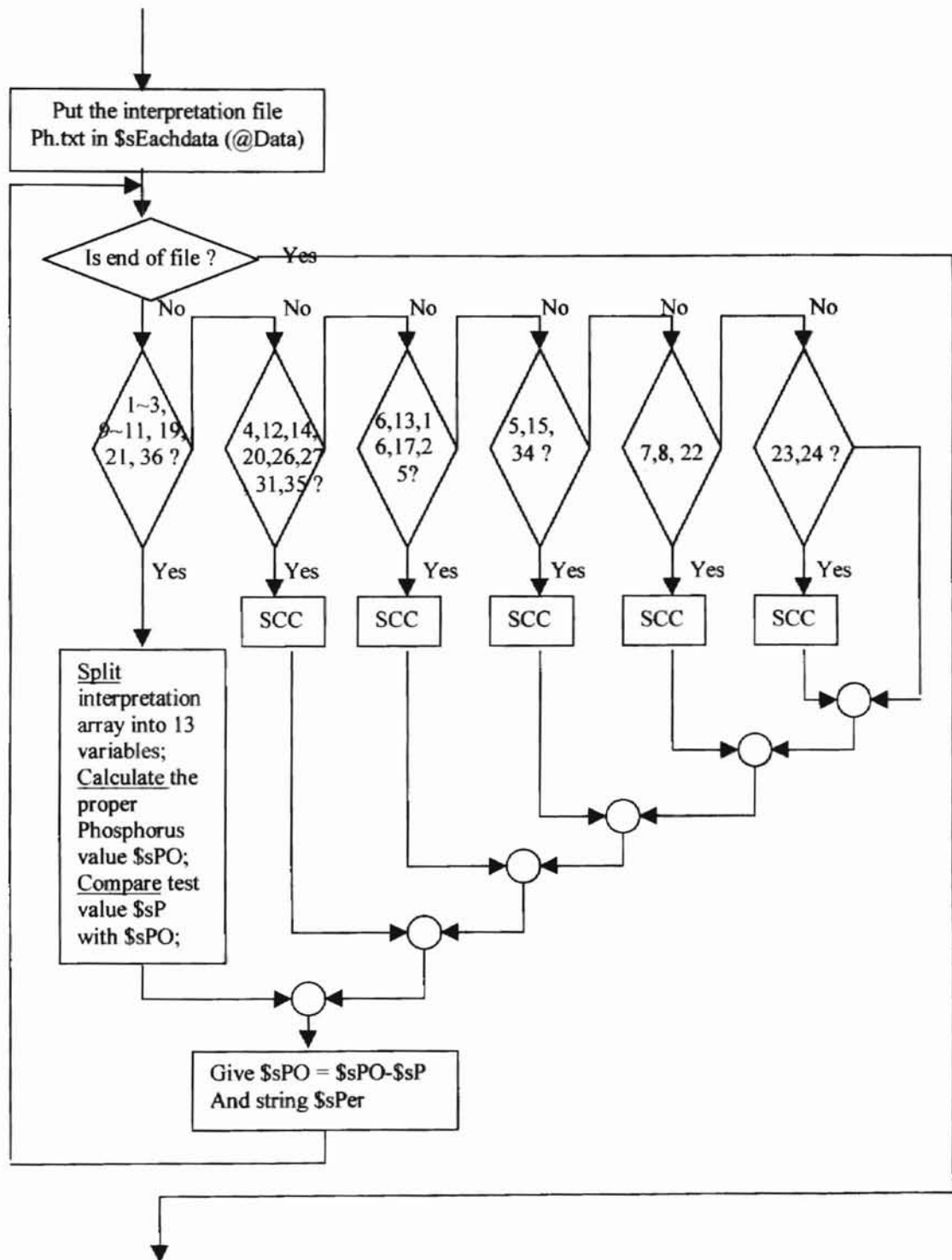


Figure 8-6. The detailed flow chart of generate report (continued).

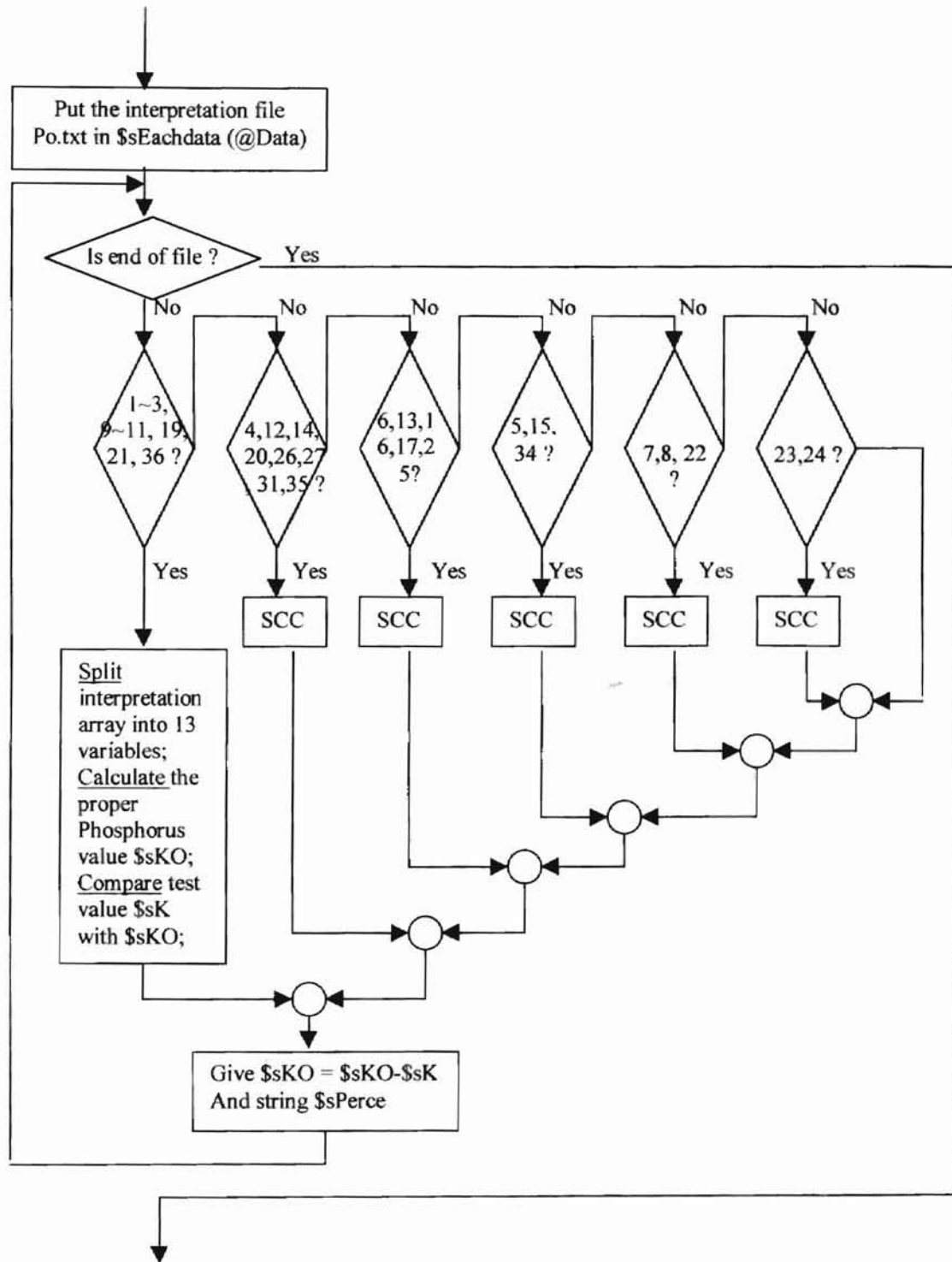


Figure 8-7. The detailed flow chart of generate report (continued).

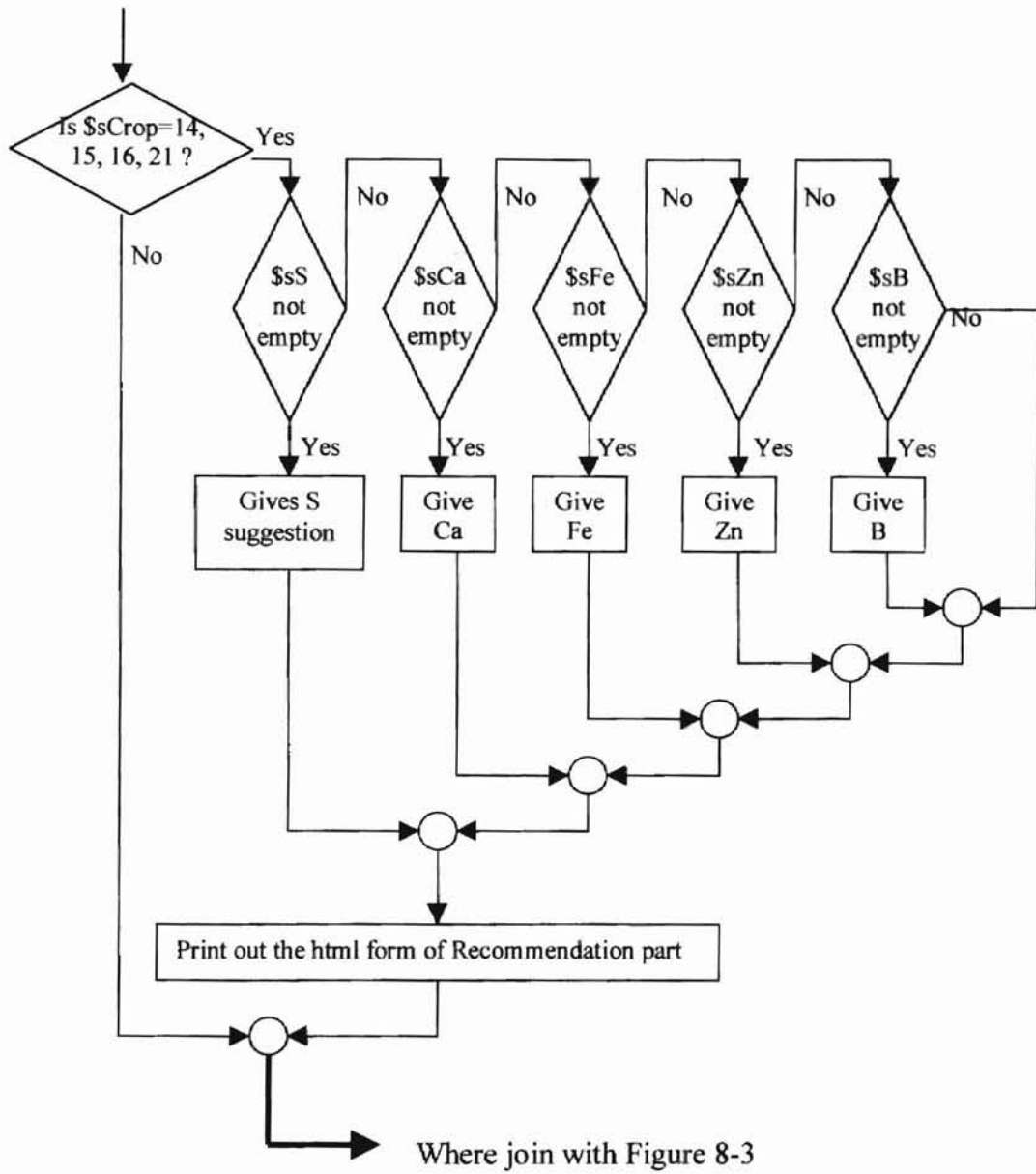


Figure 8-8. The detailed flow chart of generate report (continued).

4.2.2 Structure of result.pl

The structure of result.pl is showed in Figure 9, and it invoke the html form is show in below the Figure 9.

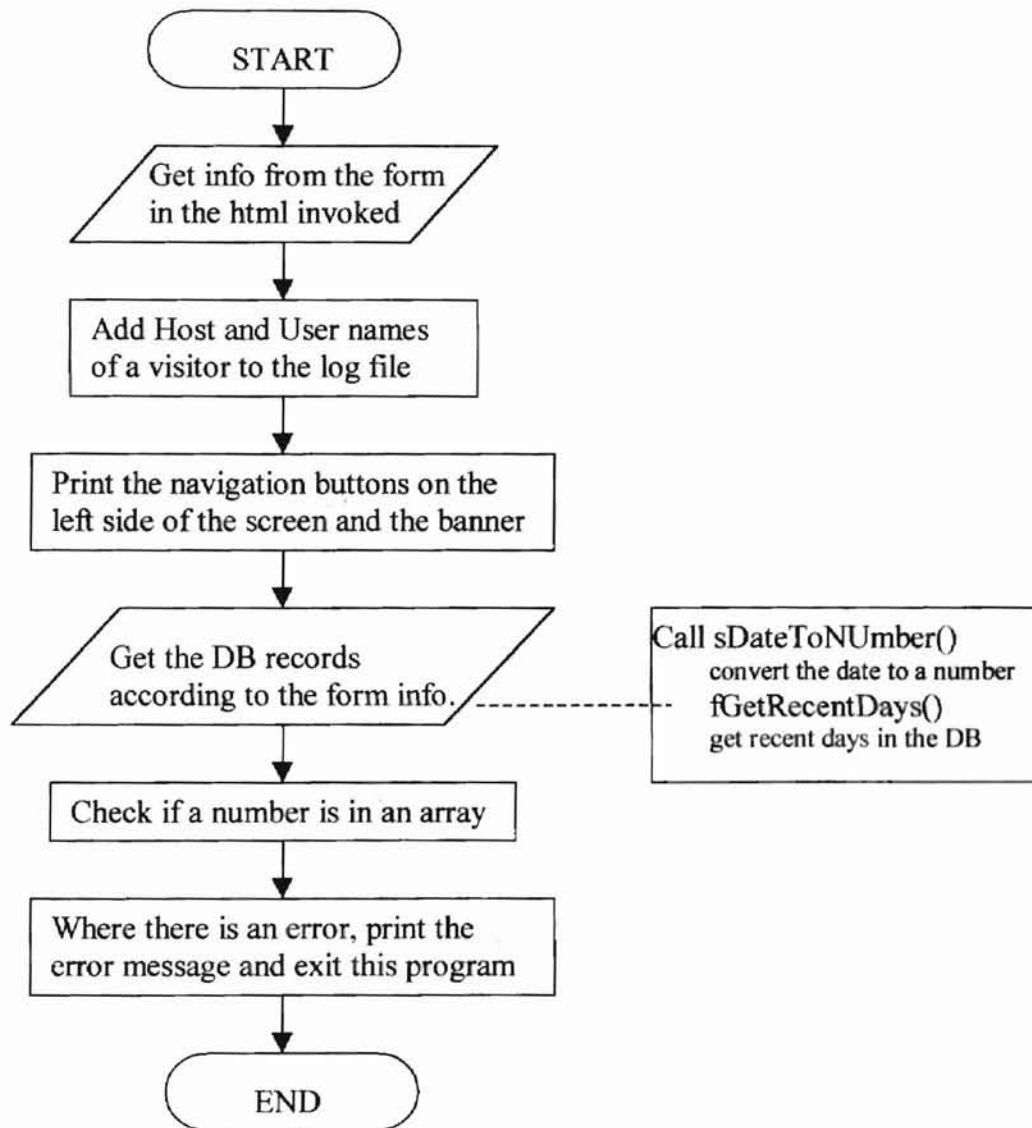


Figure 9. The general flow chart of result.pl

The HTML structure

```
<HTML>
  <HEAD></HEAD>
  <BODY>
    <TABLE 1>
      result.pl
      Navigation buttons
      SWFAL banner
      Result
      <table 1_1>
        <FORM>
          <TABLE 1_1_1>
            Type radio box
          </TABLE>
        </FORM>
        <table 1_1_2>
          Input values
          <TABLE 1_1_2_1>
            from to user defined days
          </TABLE>
        </table>
      </table>
    </TABLE>
  </HTML>
```

<table 2>

Database data

</table>

</BODY>

</HTML>

4.3 Web page

4.3.1 Web page of uploading data file

When user type the web sit of <http://silt.agr.okstate.edu/JackWu/swfal/upload.html>, the File upload web page will showed as Figure 10.

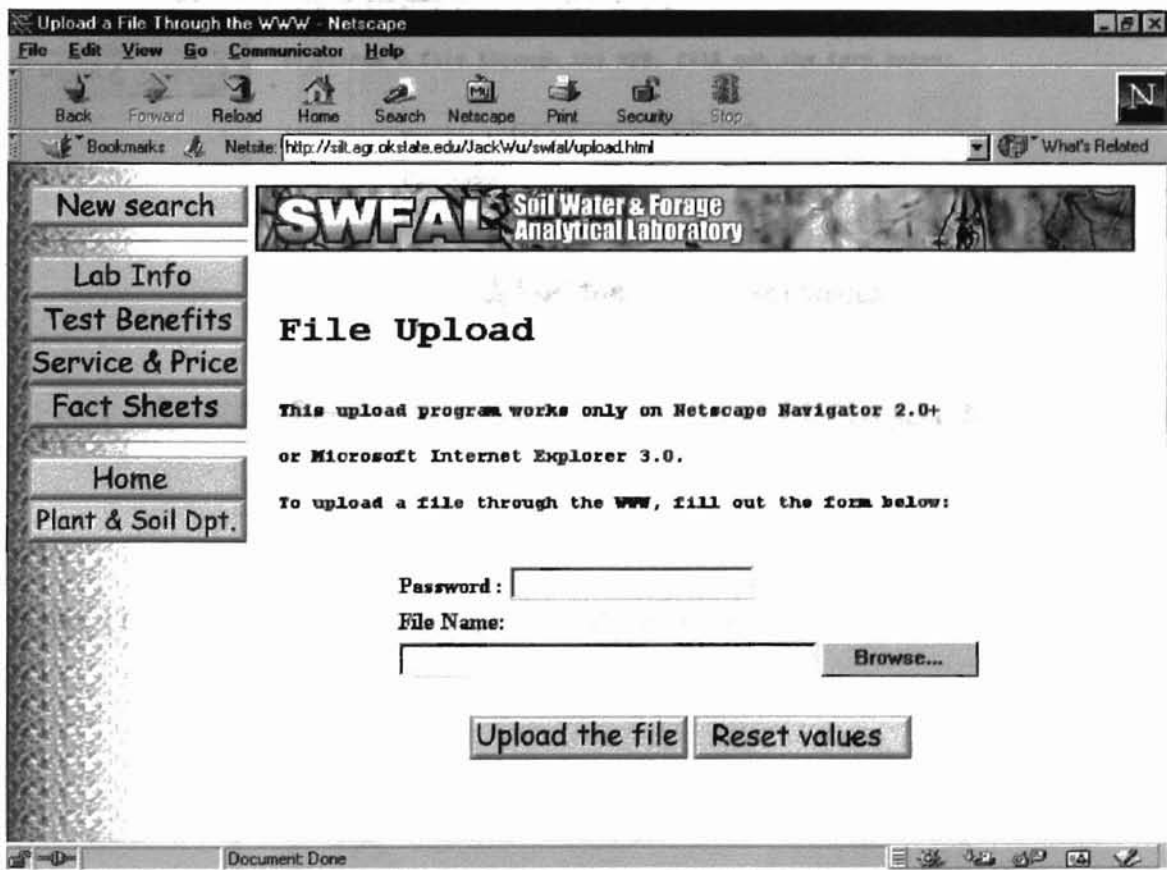


Figure 10. The "file upload" web page.

When you click the "Browse" icon, the File Upload window will show up as Figure 11.

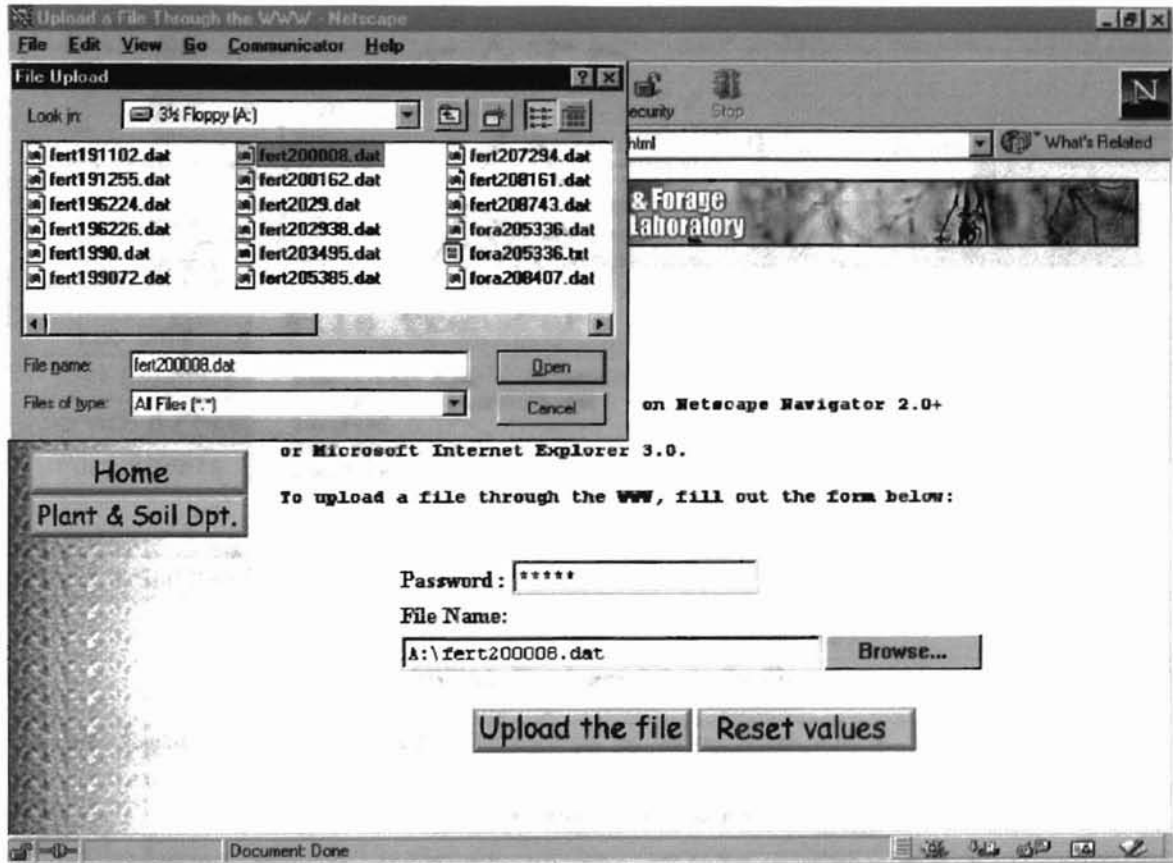


Figure 11. The "file upload" web page with file upload window.

When user finish opening the uploading file, filling the correct password, and clicking the “Upload the File” icon, the “File Transfer” web page will show up as Figure 12.

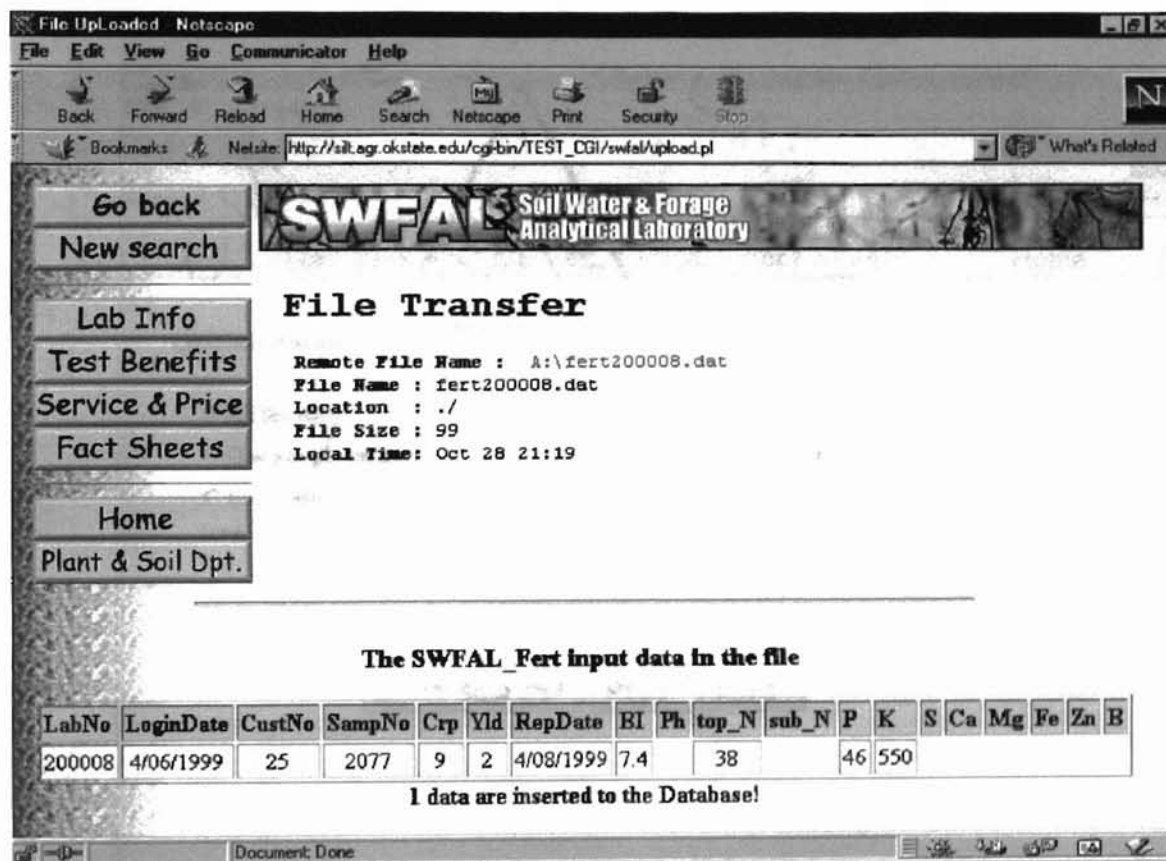


Figure 12. The “File Transfer” web page.

4.3.2 Web page for retrieval data

When user type the web sit of <http://silt.agr.okstate.edu/JackWu/swfal/dbresulr.html>, the “dbresult” web page will showed as Figure 10.

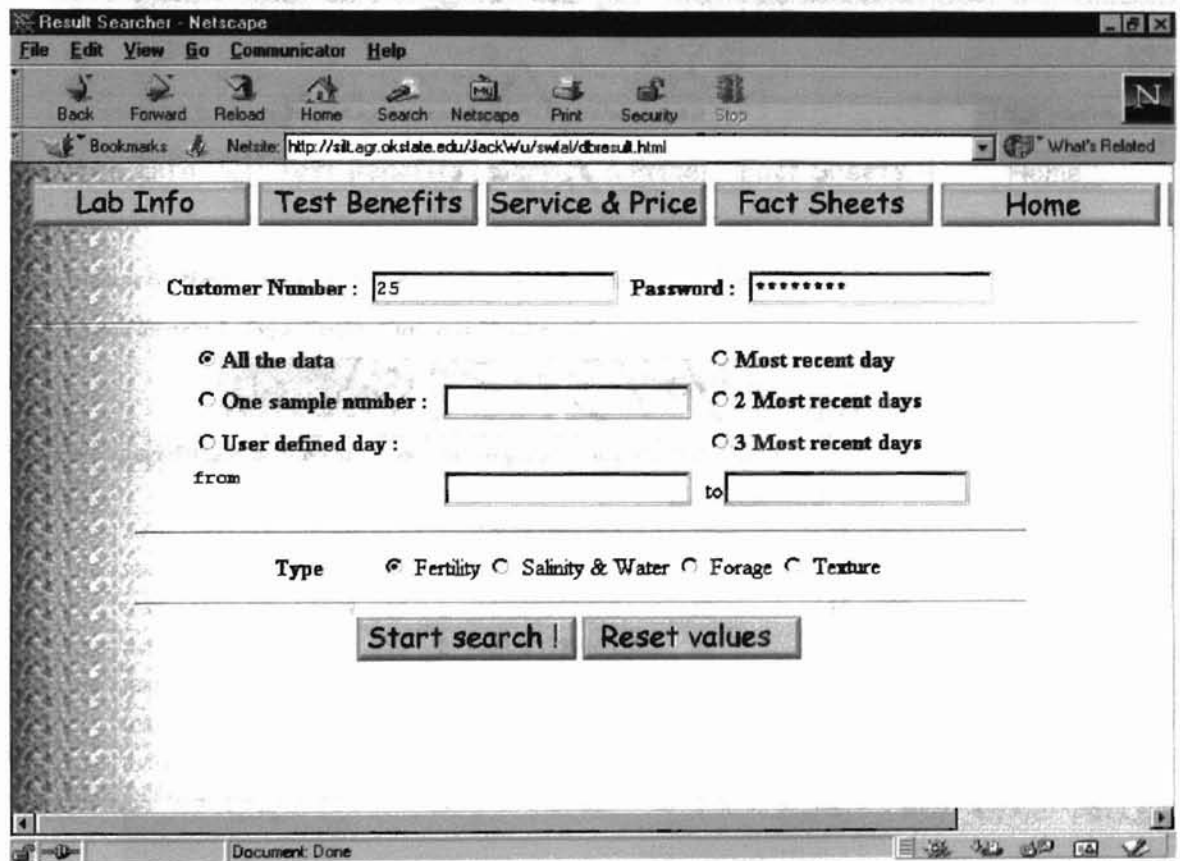


Figure 13. The “dbresult” web page.

When user finish filling Custom Number and correct password, selecting the Data type and Table type, and clicking “Start search!” icon, the “records” web page will show up as Figure 14.

Search Results - Netscape

File Edit View Go Communicator Help

Back Forward Reload Home Search Netscape Print Security Stop

Bookmarks Netsite: <http://sit.ag.okstate.edu/cgi-bin/TEST CGI/swfal/result.pl> What's Related

Lab Info Test Benefits Service & Price Fact Sheets Home

Attributes for Next Search: Fertility Salinity & Water Forage Texture

Attributes Searched: Type: Fertility Customer Number: 25

Do Back New Search Interpretation

Lab#	LoginDate	Cust#	Samp#	Crp	Yld	RepDate	pH	BI	top_N	sub_N	P	K	S	Ca	Mg	Fe	Zn	B
												← lbs./acre →						
216364	10/18/1999	25	2146	23		10/20/1999	6.3	7.3	10			89	383					
215784	10/07/1999	25	2138	13	2	10/08/1999	5	6.9	18			56	334					
215785	10/07/1999	25	2139	13	2	10/08/1999	7.5		12			16	293					
215786	10/07/1999	25	2140	13	2	10/08/1999	6.2	7.2	29			23	147					
215788	10/07/1999	25	2142	13	2	10/08/1999	5.8	7	13			29	318					
215791	10/07/1999	25	2144	13	2	10/08/1999	7.6		169			12	401					
215790	10/07/1999	25	2143	13	2	10/08/1999	5.4	7	38			26	187					

Document Done

Figure 14. The “records” web page

4.3.3 Web page for report

When user clicks the “Lab#” listed in “records” page, the corresponding report page will show up. Figure 15 to Figure 17 gives the sample of three types report.

OKLAHOMA COOPERATIVE EXTENSION SERVICE
SOIL, WATER & FORAGE ANALYTICAL LABORATORY
 Division of Agricultural Sciences and Natural Resources • Oklahoma State University
 Plant and Soil Sciences • 048 Agricultural Hall • Stillwater, OK 74078

SOIL TEST REPORT

GARVIN CTY EXT OFC	Name:	Lab ID.No: 215784
RM 7		CustCode: 25
COURTHOUSE	Location:	Sample No: 2138
PAULS VALLEY, OK 73075		Report Date: 10/08/1999


Samp#	Crp	Yld	pH	HI	top_N	sub_N	P	K	S	Ca	Mg	Fo	Zn	B
					lb/ac	lb/ac	lb/ac	lb/ac	lb/ac	lb/ac	lb/ac	ppm		
2138	13	2	5	6	9	18	56	334						

Interpretations & Requirements for		Bermudagrass	
- Test -	- Interpretation -	-- Requirement --	-- Comments --
pH	Deficient	1. tons ECCE/acre	
Nitrogen	Deficient	82 lbs/acre N	
Phosphorus	98 % Sufficient	7 lbs/acre P ₂ O ₅	
Potassium	Adequate	No K ₂ O	

Figure 15. Soil test report

Netcape
 File Edit View Go Communicator Help
 Back Forward Reload Home Search Netscape Print Security
 Bookmarks Netsite <http://all.ag.okstate.edu/lack/wu/wfsl/report/216471.html> What's Related

OKLAHOMA COOPERATIVE EXTENSION SERVICE



SOIL, WATER & FORAGE ANALYTICAL LABORATORY
 Division of Agricultural Sciences and Natural Resources • Oklahoma State University
 Plant and Soil Sciences • 048 Agricultural Hall • Stillwater, OK 74078

WATER QUALITY REPORT

GARVIN CITY EXT OFC **Name:** **Lab ID.No: 216471**
RM 7 **CustCode: 25**
COURTHOUSE **Location:** **Sample No: 2145**
PAULS VALLEY, OK 73075 **Report Date: 10/22/1999**

TEST RESULTS FOR Household Water Test


Analyses	Sample Results	EPA Drinking-Water Limits
Sodium (ppm)	40	Not Regulated
Calcium (ppm)	78	Not Regulated
Magnesium (ppm)	51	Not Regulated
Potassium (ppm)	2	Not Regulated
Nitrate-N (ppm)	<1	<10 ppm
Chloride (ppm)	30	<250 ppm

Document Done

Figure 16. Water quality report

Metscape
 File Edit View Go Communicator Help
 Back Forward Reload Home Search Netscape Print Security Future Web
 Backmarks Netsite <http://ill.ag.okstate.edu/ack/wu/swal/report/216512.html> What's Related

OKLAHOMA COOPERATIVE EXTENSION SERVICE



SOIL, WATER & FORAGE ANALYTICAL LABORATORY
 Division of Agricultural Sciences and Natural Resources • Oklahoma State University
 Plant and Soil Sciences • 048 Agricultural Hall • Stillwater, OK 74078

FORAGE ANALYSIS REPORT

WASHITA CTY EXT OFC Name: Lab LD.No: 216512
 CustCode: 75
125 W MAIN Location: Sample No: 6028
CORDELL, OK 73632 Report Date: 10/22/1999

Samp#	Cry	Nitrate	ADF	NDF	Moist	Protein	TDN	Etaint	Elact	Esain	RFV
		ppm	%								
6028	21		34.0	46.9	10.0	22.9	62.44	0.60	0.60	0.40	124

Comments:

Documents Done

Figure 17. Forage analysis report.

CHAPTER V

Summary and Future Work

5.1 Summary

To improve the agricultural extension services at Soil, Water, and Forage Analytical Laboratory at the Department of Plant and Soil Science, Oklahoma State University, the SWFAL web-based database system will be developed. The objective of this project is to create a set of Oracle tables to store the testing result and a web page with its associated Common Gateway Interface (CGI) programs to upload and retrieve data into and from the Oracle tables. The SWFAL web-based database system is implemented by:

- the four Oracle tables on a UNIX server to store testing results;
- a web page for uploading testing results into the database tables;
- a CGI program (upload.pl) for uploading testing results into the database tables and generating the reports for tested samples;
- a web page for retrieving data from the database tables;
- a CGI program (result.pl) for retrieving data from the database tables.

SWFAL system is a very helpful and advanced database information system. Since the system is available on WWW, it can be accessed from anywhere regardless of geographic location, it made the SWFAL lab. Manager uploading file easier and the users of SWFAL extension service accessing data faster and more convenient, it made

the reduction of paper use to protect the environment indirectly. The only system that is required is a client machine with Internet connection and a browser supporting graphics. These browsers are popularly available at minimal cost or even no cost.

5.2 Future Work

Currently the analysis reports are made for only three types of sample, which are the soil fertility, water quality, and forage types, the soil texture report is under development. The whole system needs tested data in boundary, on boundary, and out of boundary.

Bibliography

- [BM, 1998] Bryan, M., SGML and HTML Explained, Harlow, England: Addison-Wesley Longman, Inc., 1998.
- [CTTN, 1998] Christiansen, T., and Torkington, N., Perl Cookbook, CA: O'Reilly, 1998.
- [CGOCGC, 1997] Cheng, G., Ou, C., and Geoffrey, C., Access of Data using SQL-PL/SQL-OraPerl in Oracle, <http://www.npac.syr.edu/users/gcf/arldatabase/arloracle-data-accessfall97/>
- [DCDH, 1997] Date, C.J., and Darwen, H., A Guide to the SQL Standard, MA: Addison-Wesley Longman, Inc., 1997.
- [EW, 1999] EarthWeb Inc., The Perl Journal, <http://itknowledge.com/tpj/whatisperl.html>
- [GI, 1998] Graham, I., Version of HTML, <http://www.utoronto.ca/webdocs/HTMLdocs/NewHTML/html3.html>
- [HA, 1998] Homer, A., Professional ASP Techniques for Webmasters, UK: Wrox Press, 1998.
- [HASD, 1998] Homer, A., and Sussman, D., Professional MTS and MSMQ with VB and ASP, UK: Wrox Press, 1998.
- [HJSR, 1998] Hall, J.N., and Schwartz, R. L., Effective Perl Programming: Writing Better Programs with Perl, MA: Addison-Wesley Longman, Inc., 1998
- [ISO 8879, 1986] International Standard ISO 8879 Information Processing – Text and Office systems – Standard Generalized Markup Language (SGML), Switzerland: International Organization for Standardization, 1986
- [KGLK, 1995] Koch, G., and Loney, K., ORACLE: The Complete Reference, CA: McGraw-Hill, Inc., 1995.
- [OA, 1999] O'Reilly & Associates, Inc. Practical Extraction and Report Language, <http://www.perl.com/pub/doc/manual/html/pod/perl.html>

- [OC, 1998] Oracle Corporation, Support Web Center: Glossary, <http://www.oracle.com/support/library/glossary.html>
- [OC, 1997] Ou, C., Introduction to Several Topics Referenced in Web-Oracle Database, <http://osprey7.npac.syr.edu:3768/reference-docs/cwou/database/>
- [MRGKHG, 1997] Matzen, R. W., George, K. M., and Hedrick, G. E., A Formal Language Model for Parsing SGML, Systems and Software, 1997; 36: 147-166.
- [NB, 1998] Neil, B., The XML companion, Harlow, England: Addison-Wesley Longman, Inc., 1998.
- [PC, 1995] Price, C. C., Oraperl Quick Reference, http://www.rhic.bnl.gov/html/local/oracle/oraperl_quick_ref.html
- [RPCC, 1997] Rob, P., and Coronel, C., Database System: Design, Implementation, and Management, MA: Course Technology, 1997.
- [RRCG, 1994] Rada, R., and Carson, G., The New Media, Communications of the ACM, Volume 37(9) 23-25 (September 1994).
- [SESSPN, 1999] Siever, E., and Spainhour, S., and Patwardhan, N., Perl in a Nutshell, CA: O'Reilly, 1999.
- [SGML '94 Conference Proceedings, 1994] SGML 94 'Conference Proceedings, (November 11-15, Virginia), Graphics Communication Association, 1994.
- [WLSR, 1991] Wall, L., and Schwartz, R.L., Programming Perl, CA: O'Reilly, 1991.
- [W3C, 1997] W3C, HTML Home Page, <http://www.w3.org/Markup/>

APPENDIX A: List of Acronyms

ANSI:	American National Standards Institute
ASCII:	American Standard Code Information Interchange
CGI:	Common Gateway Interface
DBD:	Database Driver
DBI:	Database Interface
DDL:	Data Definition Language
DML:	Data Manipulation Language
DTD:	Document Type Definition
HTML:	Hyper Text Markup Language
ISO:	International Standard Organization
OCI:	Oracle Call Interface
OraPERL:	an extension of PERL to access Oracle database
PERL:	Practical Extraction and Report Language
PL/SQL:	Procedural Language extension of SQL
RDBMS:	Relational Database Management System
RTF:	Rich Text Format
SGML:	Standard Generalized Markup Language
SQL:	Structured Query Language
SWFAL:	Soil, Water, and Forage Analytical Laboratory
WWW:	World Wide Web

APPENDIX B: The maintenance of generating report part in the system

Since “generate report” is the biggest sub function in this program, it is necessary to give both item explanations and brief structure persodu code ordered by line number. Hopefully, it can make convenience for maintain and modify this program.

In Unix “silr.agr.okstate.edu” system, follow the path of “/usr/DISK3/suite_cgi / TEST_CGI/swfal”, there is the program “upload.pl”. Use “vi upload.pl” to edit program, if use “:set nu”, the each line number will be showed up. The explanation of “generate report” sub function and its brief structure itemize below.

The item explanation

I.	Report heads part	230 ~ 310
1.	Forage analysis report	232 ~ 253
2.	Soil test report	255 ~ 262
3.	Soil texture report	264 ~ 270
4.	Water quality report	272 ~ 310
II.	Get the address and uploading data file, and print the header	312 ~ 354
1.	Get the address	312 ~ 324
2.	Get the uploading data file	328 ~ 329
3.	Print out the header	330 ~ 354
III.	The content of reports	360 ~ 3062

1. The round up about report items (results)	360 ~ 516
a. Forage analysis report	379 ~ 442
b. Soil test report	443 ~ 472
c. Soil texture report	473 ~ 485
d. Water quality report	486 ~ 513
2. The interpretations and requirements part	517 ~ 3067
a. Forage analysis report	522 ~ 538
b. Water quality report	539 ~ 950
1) Irrigation water (\$sTest = 1)	545 ~ 680
2) Livestock water (\$sTest = 2)	681 ~ 720
3) Household water (\$sTest = 3)	721 ~ 832
4) Salinity management (\$sTest = 5) and Comprehensive salinity (\$sTest = 6)	833 ~ 950
c. Soil test report	954 ~ 3067
1) Crop name hash table	958 ~ 1065
2) The value of Ph	1068 ~ 1394
3) The value of Nitrogen	1400 ~ 2150
4) The value of Phosphorus and Potassium	2157 ~ 2453 2455 ~ 2806
5) For crop number = 14, 15, 16, 21	2811 ~ 3005
Get S recommendation	2811 ~ 2900
Get Ca recommendation	2904 ~ 2931
Get Fe recommendation	2935 ~ 2950

Get Zn recommendation	2955 ~ 2975
Get B recommendation	2979 ~ 3005
6) Check sufficient	3015 ~ 3062

The brief structure

```

213 sub get_report ()
    {
222     foreach $sEachLine(@FileData)
        { # I. Report headers
            # II. Get address for each sample, open uploading file, and print the
header
            # III. Report contents
360     foreach $iItem(@aEachLine)
        { # Round up report items
513     }
522     if ($iItemNum == 15)
        { # Forage analysis report
538     }
539     elsif ($iItemNum == 25)
        { # Water quality report
950     }
951     elsif ($iItemNum == 19)
952     { # Soil test report
3067     }

```

```
3085     } # end of foreach $$EachLine(@FileData)
3088 } # end of sub Get_report ()
```

2

VITA

Qiang Su

Candidate for the Degree of
Master of Science

Thesis: A WEB-BASED DATABASE SYSTEM FOR SOIL, WATER, AND FORAGE
TESTING INFORMATION MANAGEMENT

Major Field: Computer Science

Biographical: Born in Shanghai, China, the youngest child of Zongxiong Su and Lihua Shen.

Education: Graduated in July, 1980 from Tianjin 16th high school in Tianjin, China. Received a Bachelor of Engineering degree in Electronic Instrument Engineering from Tianjin University in July, 1985, Tianjin, China; Received a Master of Science degree in Application Information Technology from Vrije Universiteit Brussels, Brussels, Belgium; Completed the requirements of the Master of Science at Oklahoma State University, Stillwater in May, 2000.

Professional Experience: Assistant engineer from September 1985 to September 1991 in Tianjin Canon Copy Machine Company, Tianjin, China. Research assistant from October 1998 to present in Oklahoma State University, Stillwater, Oklahoma.