AN OBJECT-ORIENTED DATABASE RETRIEVAL SYSTEM FOR AQUATIC TOXICITY DATA FILES

By

KUMPERA KULPAIBOON

Bachelor of Science
Ramkhamhaeng University
Bangkok, Thailand
1983

Master of Business Administration
Oklahoma City University
Oklahoma City, Oklahoma
1987

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
December, 1993

AN OBJECT-ORIENTED DATABASE RETRIEVAL

SYSTEM FOR AQUATIC TOXICITY

DATA FILES

Thesis Approved:

Huizhu Lu

Thesis Advisor

J Chandler

[illegible]

Thomas C. Collins

Dean of the Graduate College

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to Dr. Huizhu Lu for her encouragement and advice throughout my graduate program. Without her close attention, patience, and guidance this research work wouldn't have been possible.

Many thanks also go to Dr. John P. Chandler, Dr. George E. Hedrick, and Dr. Sterling L. Burks for serving on my graduate committee. Their cooperation and supports were very helpful throughout the research work. A special thanks to Dr. Burks for providing AQUIRE data files and guidance on user interface section of the research.

My parents, Somphot and Jamnean Kulpaiboon, encouraged and supported me all the way. I sincerely appreciate their love, motivation, and faith in my abilities.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Database and Object-Oriented Paradigm

A database system generally is a computerized record keeping system. Its purpose is to maintain data with integrity and security and make them available to a user. The user can access its records with an accessing language. An accessing language at least includes retrieval, addition, deletion, and updating operations.

Object-oriented programming (OOP) is a new programming technique that allows programmers to reuse code components previously developed under a defined set of classes, objects and methods. The object-oriented programming provides a capability for incorporating symbolic representation of facts, data, and heuristic knowledge in a database model. Such a database system based on object-oriented concepts is called an object-oriented database system.

The object-oriented database system offers an ability to represent both declarative and procedural knowledge. The declarative knowledge is incorporated with the procedural knowledge to form a complete representation of real-world

entities. This provides a better way to present data in database application areas.

The object-oriented environment supports classes and hierarchy structures. Any class inherits methods from its superclass. As a result, applications may be written by adapting previously constructed code to the task here. A programmer selects a previously defined class that functionally resembling the intended application and creates a subclass. He or she needs only redefining significant differences by overriding an inappropriate method with a customized one.

AQUIRE Database

The Aquatic Toxicity Information Retrieval Database (AQUIRE) was established in 1981 by the United States Environmental Protection Agency Office of Pesticides and Toxic Substances. It provides a quick access to a comprehensive aquatic toxicity data. It contains over 105,300 records updated annually. The AQUIRE data files contain information up to March 1989.

AQUIRE contains reviewed scientific articles published both nationally and internationally on the toxicity of chemicals to aquatic organisms and plants. It also includes selected toxicity test results and related effects of laboratory and field aquatic toxicity chemical to freshwater and marine organisms (AQUIRE 1989).

The Objective of the Thesis

A DEC VAX 11/785 computer running VMS operating system using VAX FORTRAN language is the original retrieval system. However, most of the toxicity researchers use personal computers as electronic tools for their researches. Thus, the goal of this thesis is to construct a retrieval system to AQUIRE data files, with object-oriented programming (OOP), on a personal computer using DOS operating system.

The Organization of the Thesis

Followings are the organization of the Thesis. Chapter I, this chapter, introduces object-oriented database and original AQUIRE database system. The objective of the thesis is also presented in this chapter. Chapter II introduces object-oriented paradigm, object-oriented languages, and review of the existing object-oriented systems. Chapter III presents further details of the object-oriented system and its data model.

Chapter IV is the comparison between VAX FORTRAN and Intel 80386 data representations. The conversion methods are introduced in this chapter. Chapter V presents the designed components of the proposed retrieval system. Chapter VI is the conclusion of the thesis and suggestion for future study. Finally, the information to setup the retrieval system, user's manual, and sample retrieved output are presented in the Appendix A, B, and C.

CHAPTER II

LITERATURE REVIEW

Introduction

Recently, database systems have emerged in many new application areas such as CASE, CAD/CAM, office automation, and expert systems. These areas require advanced data modeling capabilities to handle their complex data representations. An object-oriented data model presents a feasible solution. Its model consists of conceptual entities as objects, collection of objects as classes, and organizing all classes in a hierarchical form.

The basic idea of an object-oriented database is to represent an item in the real world being modeled with a corresponding item in the database (Peterson 1987). This leads to improved maintainability and understandability of the systems, especially, the system with a great degree of complexity.

The object-oriented database is developed from the concepts of object-oriented programming to support complex applications. It reduces the semantic gap between a program and its supporting data. The important properties of an object are data encapsulation and inheritance. The object-

oriented programming encapsulates procedures into data they manage. These procedures are called methods. These exact form of data and methods are not directly accessible from outside the object, but with interaction between objects (Peterson 1987). An object may inherit attributes from other objects. It may mix, match, and even substitute those inherited data and methods.

Klaus Dittrich divided object-orientation into three levels: structurally object-oriented, operationally object-oriented, and behaviorally object-oriented (Dittrich 1986). A structural object-orientation is a model that defines data structures to represent entities of any complexity. An operational object-orientation is a model that includes operators to deal with complex objects. This level includes the structural object-orientation. Finally, a behavioral object-orientation includes features to define object type of any complexity with a set of specified operators. Thus, an instance can only be used by calling these operators.

Object-Oriented Language

The nature of object-oriented language changes the way programmers think about programming. Object-oriented programming focuses on the data to be manipulated rather than on procedures that do the manipulation. A programmer creates an object and a set of methods instead of a procedure to control the system operations.

There are many examples of object-oriented language such as Simula-67, Smalltalk, ADA, Clascal, LISP Flavors, and C++. Each language has a certain degree of supporting an object-oriented paradigm. For example, Smalltalk is more object-oriented than ADA because it supports an inheritance mechanism, but ADA does not. However, Ada has a stronger typing mechanism than Smalltalk has.

Simula-67. Simula-67 is considered the father of all object-oriented languages (Meyer 1987). It was the first language to introduce class as a language mechanism for encapsulating data. However, the term object-oriented and the explicit awareness of the idea came from Smalltalk (Booch 1986). There are object-oriented languages that follow the traditional path of Simula-67 such as ADA and Clascal. Some languages follow the concepts of LISP and Smalltalk such as Flavors and LOOPS. Some languages are extensions of other languages to support object-oriented concepts. For example, Object Pascal and C++ are extensions of Pascal and C respectively.

Smalltalk. Smalltalk is an emerging language for expert systems and AI work. In Smalltalk, everything is based on objects, methods, messages, classes, and inheritance (Coed and Yourdon 1991a). The programmer defines a class of objects. Objects in Smalltalk are analogous to pieces of data in other languages (Hu 1990). Smalltalk class

describes an object and the methods. For example, numbers are objects and each number is an instance of the class number and responds to messages such as plus, minus, and modulo. An integer is a subclass of the class number. It responds to the methods defined in its superclass, class number (Alexander 1985).

Object-C. Objective-C, developed by Brad Cox, is an object-oriented language resulted of combining Smalltalk with C (Booch 1986 and Christian 1993). The language is an extended C in the direction of Smalltalk, but less like C than C++. It is a primary programming language on the NeXT machine. The language introduces an object and message as a new data type and operation to those in C. The power of Objective-C lies in the predefined classes.

C++. C++ is based on C and is a superset of C++. It was introduced by Bjarne Stroustrup in 1983 (Christian 1993). It is a hybrid language that supports both the procedural and object-oriented paradigm. The language supports object-oriented concepts and high-level abstraction (Wiener 1988). It is similar to Modula-2 in its simplicity and support for modularity. Finally, C++ has a flexibility to deal with the hardware-software interface and low level system programming.

C++ is a strongly typed object-oriented language. Compiler checks and demands that all initialization and

assignment statements comply with the type defined at compile time. This increased type checking reduces errors in software system when compares to C. In C++, the programmer declares a class and defines the methods for the class with specific accessibility. These are public, protected, and private.

Related Works on Object-Oriented system

GemStone. GemStone is an object-oriented database server developed by Servio Logic Development Corporation. It is classified as a behaviorally object-oriented system (Peterson 1987). GemStone is designed to support a database management system for multiuser environment. GemStone supports an object identity no matter where the object is or what the object contains. It provides controlled shared access of common data, which is increasingly common to object-oriented databases. GemStone was initially implemented on a VAX-VMS machine. The principal concepts used in GemStone are objects, message, and class. These correspond to record, procedure call, and record type in a conventional database (Hu 1990).

POSTGRES. POSTGRES is a relational database with extensions to support object management developed at the University of California at Berkeley (Stonebraker 1986). It supports abstract data type mechanism. POSTGRES allows a

user to define a new data type, as a tuple in the relational model, along with procedures to manipulate it.

The approach to relational model with procedures as database objects give three properties (Peterson 1987). First, subobject is no longer need to be accessed in a database object, because the procedures are the mechanism to implement the part-of hierarchy. Second, the unpredictable types of subobjects can be modeled efficiently. Finally, an object can be stored once and indirectly referenced by a subobject.

OZ+. OZ+ is an object-oriented database system intended to use in modeling office activities. It is developed at the University of Toronto and implemented on Sun 3/50 system. It provides storage and retrieval of persistent data in multiuser environment (Weise 1989). OZ+ uses the concepts of objects, contents, rules, events, and message to manage object persistence and concurrence of the database system.

ODDESSY. ODDESSY is an object-oriented database system intended for computer-aided database design. It uses the concepts of objects, messages, and rules (Diderich 1989). ODDESSY defines two types of messages to specify information about an entity. First, a vertical modifier is used in data entry to define a relationship of an object with another.

Second, a horizontal modifier is used to describe property of an object.

ORION. ORION is an object-oriented database system developed at Microelectronics and Computer Technology Corporation. It is a single-user, multitasking database system that supports complex object called composite object (Kim 1989). The system is designed on Sun UNIX machine implemented Common Lisp. ORION consists of four subsystems: message handler subsystem, object subsystem, storage subsystem, and transaction subsystem. The message handler subsystem controls all messages including system defined functions, user defined methods, and access messages. The object subsystem provides control of high-level functions including class definitions, their inheritances, and database schema. The storage subsystem manages storage location. Finally, The transaction subsystem manages and controls multiple concurrent transactions.

System X-1. System X-1 adopts an object-oriented approach for knowledge representations and inferences (Leung and Won 1990). It includes both declarative and procedural representations. All system objects are grouped in the system class. The knowledge base is defined by a collection of classes, objects, and methods. End-user communicates with system defined objects or knowledge-engineer defined objects through user interface. Knowledge-engineer can define

knowledge through a conventional editor or menu driven interrogation.

DKOM. Distributed-Knowledge Object Modeling (DKOM) consists of a set of distributed knowledge objects (KO). Each knowledge object consists of a behavior part, a knowledge-base part, and a monitor part (Tokoro and Ishikawa 1984). Knowledge objects run in parallel and communicate with each other by mean of message passing. It responds to a request message according to its knowledge-base.

Next chapter, Chapter III, the author presents further details of the object-oriented system. The chapter begins with object-oriented data model, and follows with object, class, and their mechanisms.

CHAPTER III

OBJECT-ORIENTED SYSTEM

Object-Oriented Data Model

An object-oriented database system is a database system that provides all the traditional DBMS services and supports object-oriented concepts. Generally, it combines the capabilities of an object-oriented language and the storage management functions of a database system (Hu 1990). It is emerging to support the complex applications to reduce the semantic gap between complex applications and the data storage supporting those applications (Peterson 1987).

The objected-oriented term refers to a set of semantic modeling mechanisms for capturing the information of a real world application. The object-oriented model represents both declarative and procedural knowledge. Declarative knowledge is knowledge about facts and truths of the object. Procedure knowledge is knowledge about the functions that can be performed by the object. Both form a complete representation of a real world application. In object-oriented database system, the application is part of the system.

The standard for object-oriented data model has not been established yet. However, the least requirements to

classify an object-oriented data model are object and object identity, semantic primitives for the modeling of the structure of information, and object behavior and encapsulation (Rishe 1992). The first two requirements represent the semantic database models toward the information as real world entities. The last requirement presents abstraction mechanisms for the objects.

Object and Class

An object is an abstract software model of an entity in a problem domain. The object is defined with both data and operations that define behavior of the data. These objects send messages back and forth to each others. All object of a particular type belong to the same class. A class is a description of one or more objects with a uniform set of data and operations.

Relationships Between Classes

Classes' responsibilities can be identified by the relationships between classes. These relationships are is-kind-of, is-analogous-to, and is-part-of (Wirf-Brock, Wilkerson, and Weiner 1990). These relationships can be useful in identifying and assigning the responsibilities.

Is-kind-of Relationship. Is-kind-of is a share relationship, normally a parent-child relationship. Classes

that are a kind of some class share some responsibility. These are data and methods.

Is-analogous-to Relationship. The relationships between class can be analogous. Class A and class B may share the same responsibilities if class A has a relationship to another part of the system that is analogous to that borne by class B (Wirf-Brock, Wilkerson, and Weiner 1990). Several classes that share the same superclass are most likely to be analogous.

Is-part-of Relationship. A class can compose of many instance of other classes. When a class composes of instances of other classes, no inheritance of behavior is implied.

Object-Oriented Programming

An object-oriented programming (OOP) is a new step of the programming concept. It takes recent developments in programming language to their next logical step to increase clarity, modularity, and programming efficiency (Tello 1991). It provides a way to modularize programs that can reuse copies such modules on demand. This is one of the advantages over a procedural language. The program can be easily modified by simply adding or deleting modules (or object). There are three features of a language that makes it a truly object-oriented. These features are

encapsulation, polymorphism, and inheritance (Pappas and Murray 1990, Manola 1990).

Encapsulation

Encapsulation is a term for information hiding. Variables, constants, and their methods are hidden inside entities called objects (Duncan 1991). These data are communicated by means of messages. Each object is an instance of an object or class type. The purpose of encapsulation is that a particular object class's implementation can be modified without side effects elsewhere.

Polymorphism

Polymorphism is an object's ability to select the correct internal method based on the type of data received in a message (Duncan 1991). A print object may receive a message containing an integer, a real number, or an ASCII string. The print object may take an appropriate action depending on the incoming message without knowing the contents of the message.

Inheritance

Inheritance is the ability to create classes that will automatically model themselves on other classes. It is a language mechanism for deriving a new class of objects from

an existing class (Duncan 1991). When a class B has been defined so as to model itself on class A, class B is a child class of class A. It inherits data and method from class A. Class B need only contain the actual code and data for new or changed methods. Normally a class can have zero or more parent calls or child class.

The following chapters are details of the object-oriented retrieval system design. They begin with the data representations, designed components, and their definitions.

CHAPTER IV

DATA REPRESENTATIONS

AQUIRE data source tape is in ASCII character set using DEC VAX 11/785 processing unit with VMS 5.1 operating system and VAX FORTRAN as the implementation language. Retrieval system and other manipulation software are not included.

Files 1 to 9 describe their respective files 10 to 18. File 1 contains an introduction to AQUIRE data tape and description for data file 10. File 2 to 9 are in form of FORTRAN COMMON block descriptions for data files. The contained information shows the record format.

Files 10 to 18 are data files. File 10 is a text file, contains Chemical Abstract Service registry numbers. File 11 is the center of its database system. It contains pointers (record number or related field) to related records in file 12 to 18. File 16 contains a trailing character to fill 45 characters per record. Table 1 shows AQUIRE files names, record sizes and their contents.

TABLE 1

LIST OF AQUIRE DATA FILES

File Name	Logical Records	Logical Length	File Contents
1	58	80	TAPE_INTRO.TXT
2	66	80	AQTOX.INC
3	36	80	RETOX.INC
4	21	80	SPECIES.INC
5	32	80	STOCAS.INC
6	26	80	STOCCB.INC
7	22	80	STOCHR.INC
8	21	80	STOREM.INC
9	24	80	STOTHA.INC
10	5,392	136	CAS numbers and chemical name
11	105,394	160	Main AQUIRE data file
12	9,952	544	Citation information
13	2,815	68	Species information
14	117,103	24	CAS numbers used in each test
15	369,003	20	Concentration-Conf. int-BCF
16	117,100	45	Purity/Chem characteristics
17	260,816	72	Remarks data file
18	777,825	16	Temp-Hardness-Alk-D.O.-pH

note: Logical Length unit is bytes.

VAX FORTRAN Numeric Data Representation

VAX binary INTEGER numbers are stored in two's complement representation with the bytes stored in increasing order of significance. The least significant byte is the first byte. The most significant bit is the sign bit. The sign bit is zero for positive numbers and one for negative numbers. INTEGER and REAL number bits are labeled from the right, 0 through 15 or 0 through 31.

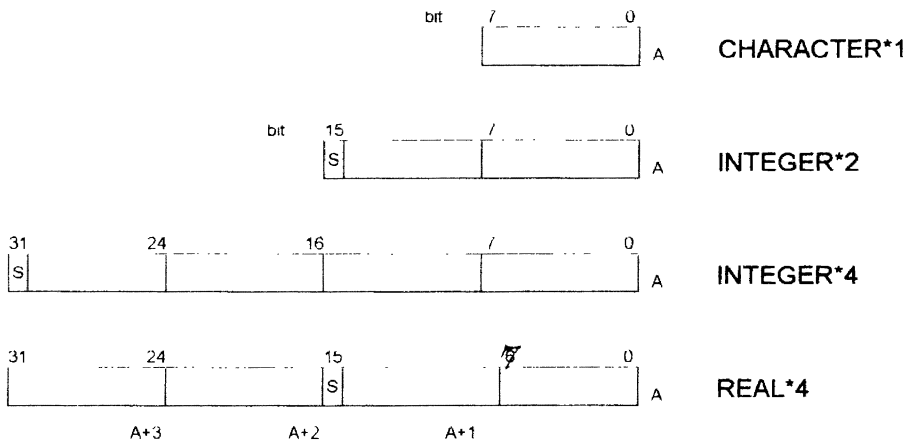


Figure 1. VAX FORTRAN Data Representation

Integer Number

INTEGER*2 are stored in two contiguous bytes aligned on an arbitrary byte boundary. Values are in the range -32,768 to 32,767.

INTEGER*4 are stored in four contiguous bytes aligned on an arbitrary byte boundary. Values are in the range -2,147,483,648 to 2,147,483,647.

The following is a table showing a bit address and its representation of VAX FORTRAN's integer and real number.

TABLE 2

VAX DATA REPRESENTATION

Data Type	Bit Address	Description
INTEGER*2	15 14:0	sign bit binary number
INTEGER*4	31 30:0	sign bit binary number
REAL*4	15 14:7 6:0 and 31:16	sign of fraction bit excess 128 binary exponent normalized 24-bit fraction with the redundant most significant bit not represent

Real Number

Floating-point data (REAL*4) is four contiguous bytes starting on an arbitrary byte boundary. Bits are labeled from right to left, 0 through 31. Bit 15 is the sign bit. The fraction bits increase in significance from 16 through 31 and 0 through 6. The fraction is normalized with the redundant most significant fraction bit not represented

(hidden bit normalization). This bit is assumed to be one unless the exponent is zero. The exponent zero represents either a real number value zero or a reserved operand.

The exponent, bit 14:7, is stored in binary excess of 128 notation. Binary exponents from -127 to 127 are represented by the equivalent binary 1 to 255.

The value data is in the approximate range 0.29*10**-38 to 1.7*10**38 with a precision of about seven decimal digits.

Intel 80386 Data Representation

Intel 80386/80486 processor breaks down a datum into bytes, which is the smallest addressable data, and stores them in memory low-order byte first. This representation maintains compatibility with its earlier processors, 8086 and 80286.

Assuming that the 32-bit hexadecimal value 100F755D is stored in memory, beginning at location A, the individual memory bytes are:

Address	A	A+1	A+2	A+3
Contents	5Dh	75h	0Fh	10h

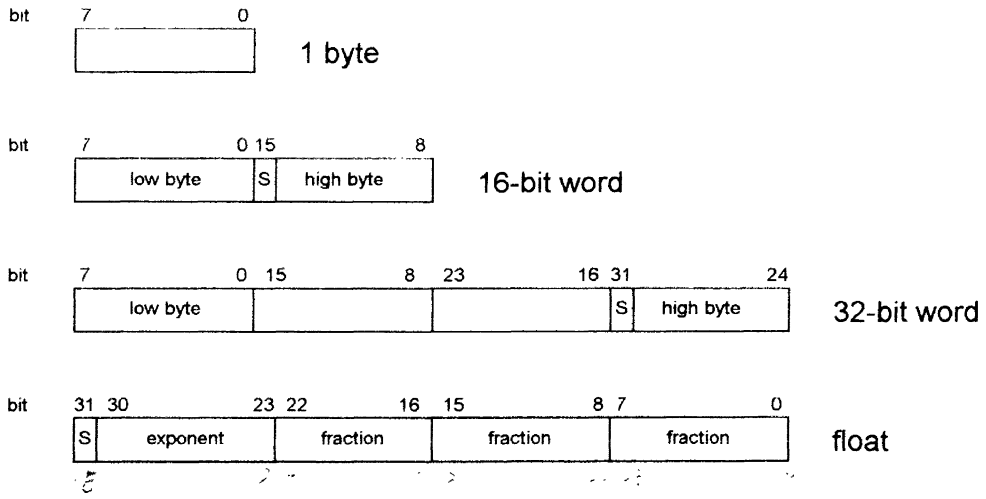
Integer Number

Integer and long-integer are stored in contiguous two or four bytes using the same convention as VAX's integers. A signed integer is stored in two's complement notion with the

byte stored in increasing order of significance. The most significant bit indicates a sign bit.

Real Number

A short real number is stored in four contiguous bytes. Bit 31 is a sign bit, zero for positive numbers and one for negative numbers. The exponent field, bit 30:23, is stored with the bias of 127 for short real numbers. For example, if the exponent represents a value of 2**3, the value in the exponent filed is 130. The values zero and all binary one's are reserved for representing special values and cannot be used to represent floating point numbers.



INTEL 80x86 DATA REPRESENTATION

Figure 2. Intel 80x86 Data Representation

The fraction part of the floating point number occupies 23 bits, bit 22:0. The fraction includes the implied 1 bit. Representative values range from ±1.18*10**-38 through ±3.4*10**38 with a precision of about seven decimal digits.

Data Conversion

The retrieval system has to read AQUIRE data and convert them from VAX's data to Intel's data format before any further use. The conversion section arranges data into a correct byte order and retrieves them with a corresponding data type.

Method: Integer Conversion

Begin. step 1. Copy two bytes of an integer field from an input record to a two bytes memory buffer.

step 2. Reverse bytes order in the buffer.

```
Intel_int[0] = Vax_int[1];
Intel_int[1] = Vax_int[0];
```

step 3. Return buffer into integer data type.

End.

Long integer conversion can be done with the same method as integer conversion, but with four bytes size applied. C language offers a union structure which will return a correct byte represents an integer data type. This can be done by copying the binary content to a defined union

structure then reading the integer from that union structure.

Method: Real Number Conversion

Begin. step 1. Copy four bytes of an integer field from an input record to a four bytes memory buffer.

step 2. Reverse bytes order in the buffer.

```
Intel_real[0] = Vax_real[2];
Intel_real[1] = Vax_real[3];
Intel_real[2] = Vax_real[0];
Intel_real[3] = Vax_real[1];
```

step 3. Return buffer into real number data type.

End.

C language offers a union structure which will return a correct byte representing a specified data type. The union structure works correctly with integer and long integer data types. However, with real number, data must be arranged manually before the union structure will return the correct value.

VAX floating point data keeps the sign bit and the exponent field in the low ordered word but Intel floating point data defines the sign bit and exponent field in the high ordered word. The above method will do the arranging data.

Character Data

Both VAX FORTRAN and Intel use ASCII character set for their character data type representation. Character and string can be directly used in the program.

CHAPTER V

DESIGNED COMPONENTS OF THE RETRIEVAL SYSTEM

Object-Oriented design is a process of decomposing a program into objects and defining the relation between them. It identifies those objects in the real world according to the domain of problems. Then, the objects are simulated in the computer according to their behavior.

The author chooses C++ as the implemented language, because C++ is an extension of C with object-oriented capability. It supports both procedural and object-oriented paradigms. Most of all, it supports the hardware-software interface and low level system programming, which is a necessary feature to deal with the original AQUIRE data files.

AQUIRE Data files

AQUIRE files consists of nine fixed-size record files. AQTOX is a center file that contains pointers to the other files. Figure 3 displays their records and their field names. Each block represents a record with its file name on the top and its field names inside each small block. An arrow-headed line represents a pointer which points from one

field to a record (record number or field of a record) of a specific file.

Each file contains one or more pointer fields which are referred to a related record in other files. Figure 4 shows these fields with their names and the number of possible occurrences. Record with more than one occurrence represents a specified number of records to combine into one related information. For example, field ZCCBPTR of AQTOX file contains a pointer to a specific record in STOCCB file which contains a pointer to the next record in itself. This next record contains a pointer to the next record, and so on, up to twelve records in STOCCB file.

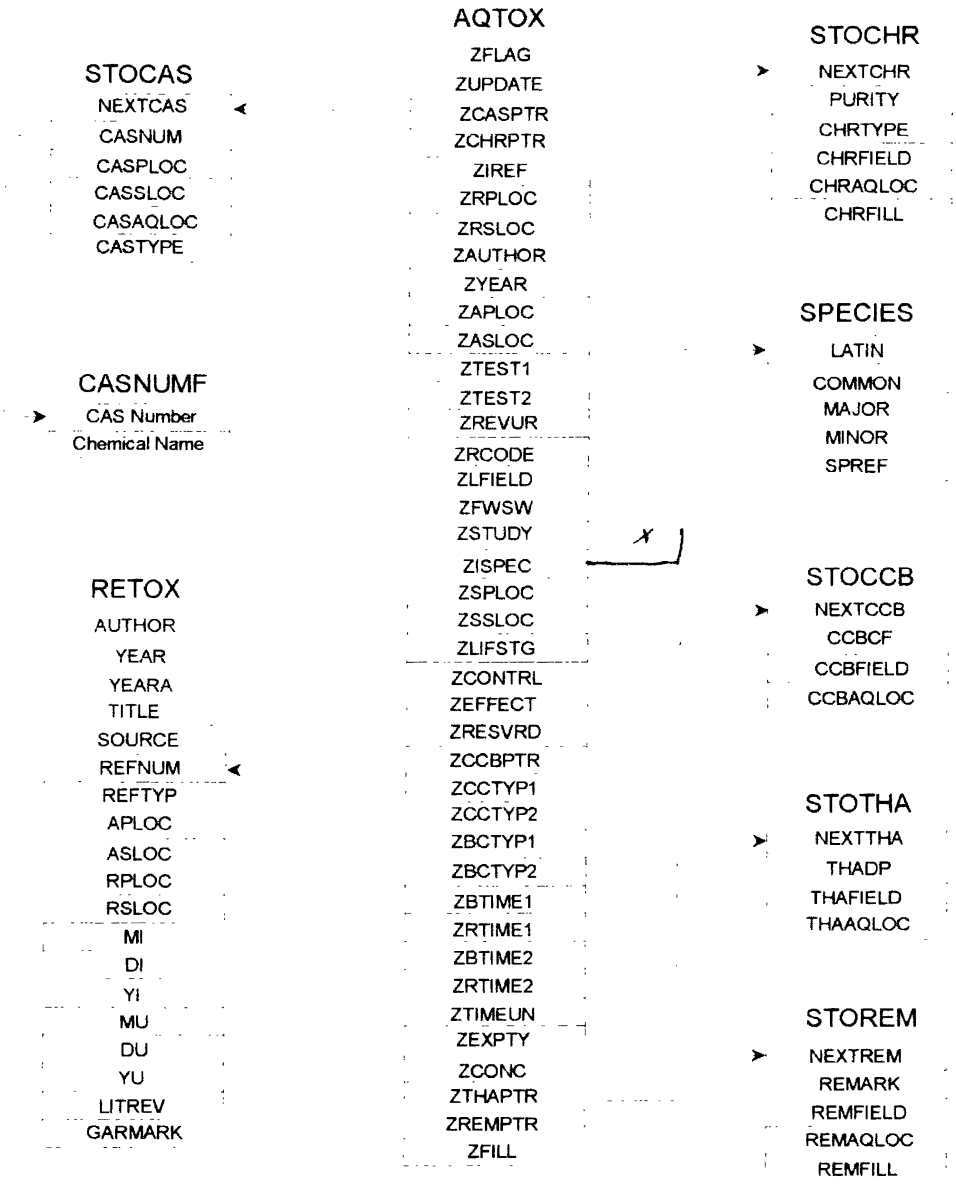


Figure 3. AQUIRE Record and Their Fields

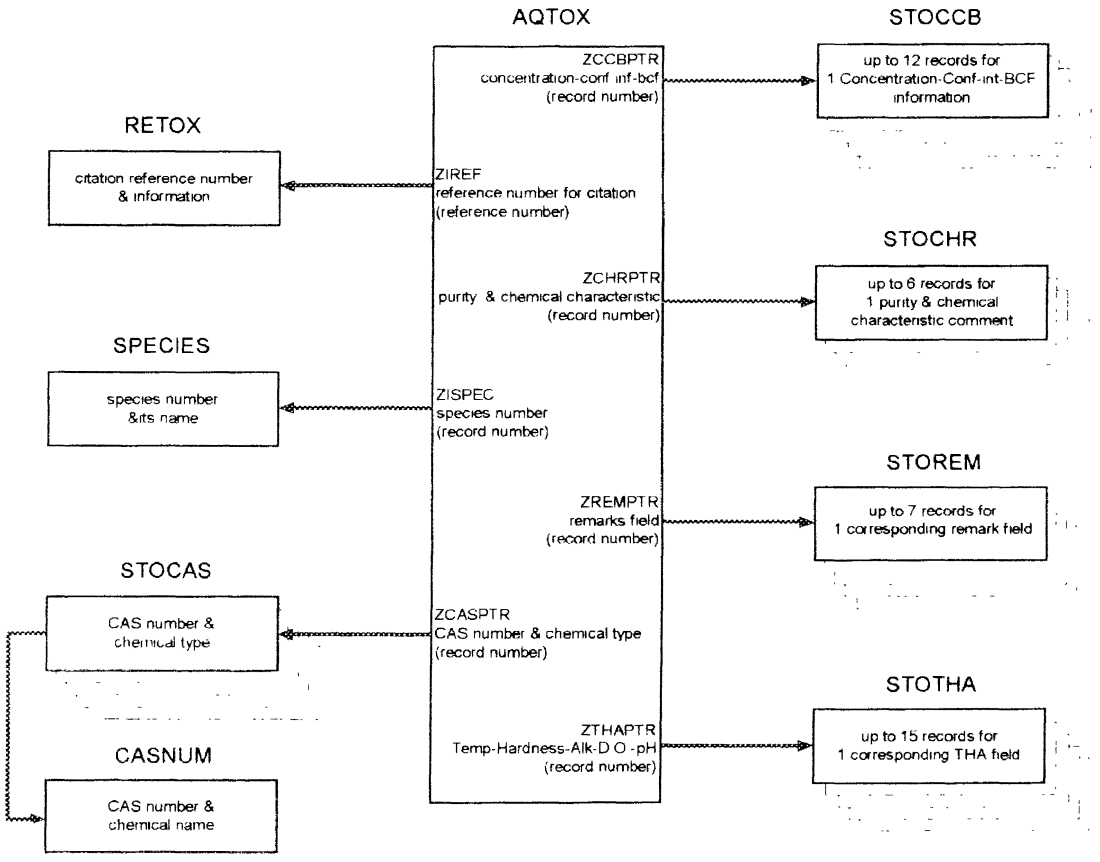


Figure 4. AQUIRE Files and Their Pointers

AQUIRE Dialog window System

The AQUIRE Dialog Window system mainly consists of two subsystems, both are object-oriented subsystems. One is a data retrieval, another is a user interface. The data retrieval subsystem controls access to database files. The user interface subsystem is designed with dialog windows and a menu system. It is a graphical user interface. A user will learn to use and understand its functions in a short time.

Data Retrieval Subsystem

The Data retrieval subsystem is the sole access to database files. This subsystem consists of eight class structures. Each class has specific services to the subsystem. The subsystem provides its client with data from the database. It converts the data from VAX FORTRAN data to Intel data representation before returning to its client.

The data retrieval subsystem maintains its own search method. It checks for current data before attempting a search. This will eliminate any unnecessary search. Figure 5 shows the designed hierarchy classes of the data retrieval subsystem.

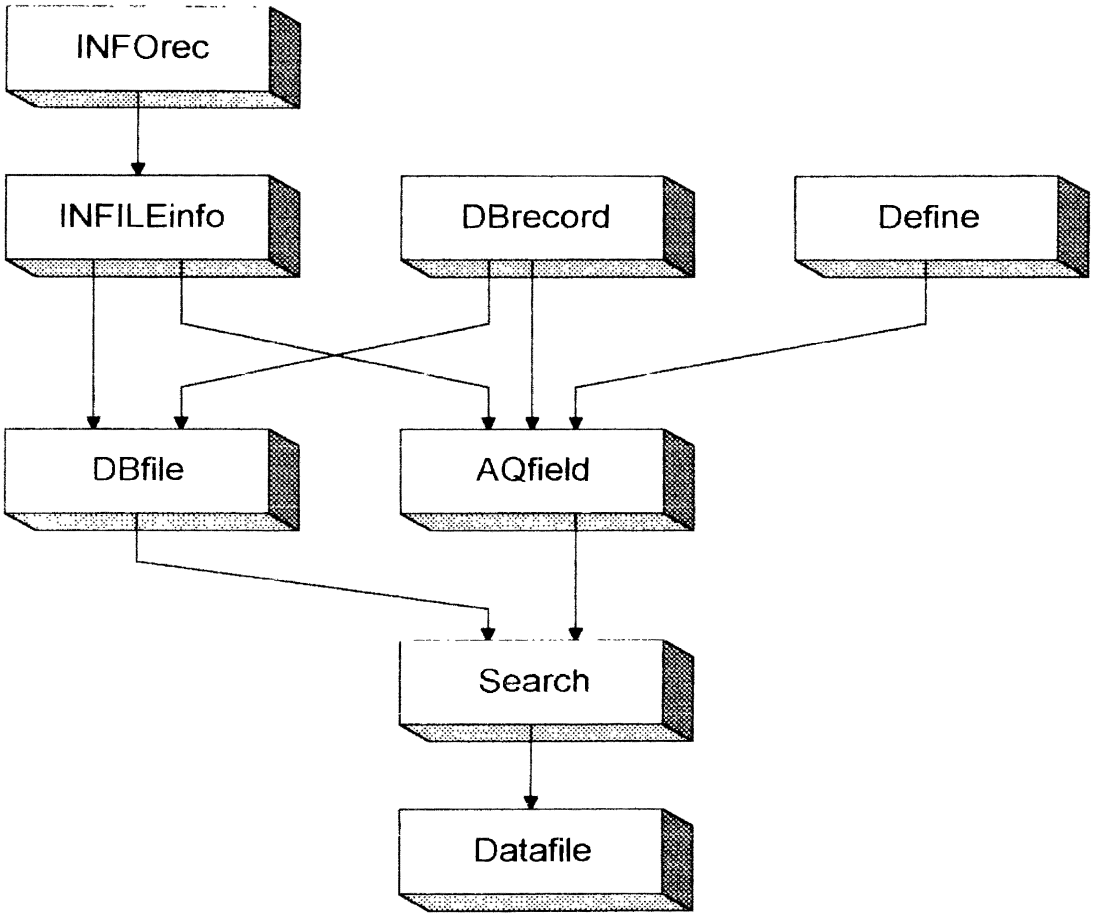


Figure 5. A Data Retrieval Subsystem

The class descriptions that constructs the data retrieval susbsytem are as follows.

Class DBrecord

This is a base class for all accessing database files. This class keeps information only about a current access record and its file name. It returns a field of record in binary format, a file name, a record size and its record number.

Parent Class. None

Child Class. DBfile, AQfield

Attribute. filename, dbrecord, field, dbrecnum, reclen

Service. setdbrec(), dbfile(), dbfield(), dbrecnum(), dbreclen()

Class AQfield

AQfield's main purpose is to perform a data conversion. It contains no member data. It gets a database field from its parent-class DBrecord and its corresponding information from its parent-class INFILEinfo. This information consists of field name, field type, field size, and field offset. Then, it converts the requested field of data into Intel data format. It also checks with it parent-class Define to define further meaning of the requested field.

Parent Class. DBrecord, INFILEinfo, Define

Child Class. Search

Attribute. None

Service. fieldvalue()

Class DBFile

DBfile controls all accessing functions to a database file. It obtains information of a retrieving record from its parent-class INFILEinfo. This information consists of file name and its record size. Then, it keeps a retrieved record in its buffer before calls its parent-class's member function setdbrec(). It also responsible to control open and close a database file.

Parent Class. DBrecord, INFILEinfo

Child Class. Search

Attribute. dbfile, flname, buffer, recsize, recread, reccount

Service. opendbf(), closedbf(), setbegin(), curfile(), iscurfile(), getrec(), reccount()

Class Search

This class inherits properties from its parent-class DBfile and AQfile to retrieve a record and extract any field of data from the record. Its solely responsible is to locate a specific record in a database file.

Parent Class. DBfile, AQfield

Child Class. Datafile

Attribute. setsearch()

Service. search()

Class Datafile

The responsibility of this class is to control open, close, and get a specific record from database files. It checks whether a database file is opened or not before it calls the search() functions. This confirms the data integrity and reduces the responsibility of other classes that require access to database files. This class is a connecting point between database files and a user interface subsystem.

Parent Class. Search

Child Class. None

Attribute. openfile(), closefile()

Service. getrecord()

Class INFOrec

This class maintains and updates information of a database file and database record. This is necessary information to access and extract a database file. This information consists of filename and its location, record name, record size, field name, field data type, and field offset.

Parent Class. None

Child Class. INFILEinfo

Attribute. headrec, inforec

Service. curinfo(), setflinfo(), addinfo(), newinfo(), filename(), recsize(), allfield(), fieldnum(), fieldname(), fieldtype(), fieldsize(), fieldoffset()

Class INFILEinfo

This is the only child-class of INFOrec. It reads a corresponding information file and saves the information in INFOrec. It also checks for presence of requested information before it attempts to read the information file.

Parent Class. INFOrec

Child Class. None

Attribute. infile, openinf(), closeinf(), readhead(), readrec()

Service. getinfname(), filename(), recsize(), allfield(), fieldnum(), fieldname(), fieldtype(), fieldsize(), fieldoffset()

Class DBFILEinfo

This class finds a match database file name of a given field name. It checks whether a file name or field name is valid or invalid. It contains no data. Actually, it is a collection of functions using information from class INFILEinfo.

Parent Class. INFILEinfo

Child Class. None

Attribute. noexteninf()

Service. dbfilename(), dbfilematch(), validfield(), validfile()

Class Define

This class gives a corresponding description of any field of record to a defined character code. It also checks for validity of some defined field contents.

Parent Class. AqtoxDef, castype_def, StoccbDef, StochrDef, StoremDef, Stothadef

Child Class. None

Attribute. fieldname()

Service. define(), valid()

User Interface Subsystem

This is a graphical user interface with menus and dialog windows. The menus give a user choices to select one or more dialog windows and commands to work with the database. With object-oriented designed interface, the user can direct access to or exit from any commands or dialog windows at any time.

Many class structures in user interface subsystem derives from Turbo Vision's class library. Turbo Vision is a user interface toolkit, C++ class library. It is a part of Borland C++ with Application Frameworks. It has components to construct dialog boxes, overlay windows, and build menu-based applications.

Figure 6 shows TDialogApp class, which is the main control of the user interface subsystem, and the communication between the data retrieval subsystem and the user interface subsystem.

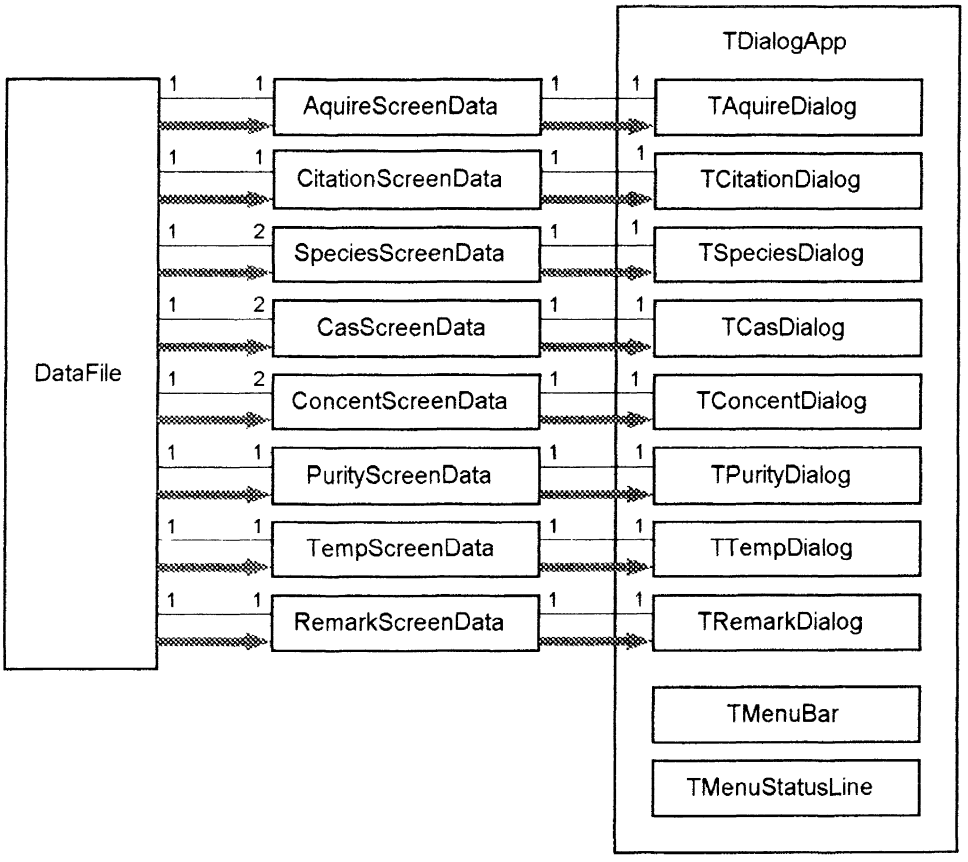


Figure 6. Dialog Window Interface Diagram

The class descriptions that constructs the user interface susbsytem are as follows.

Class AquireScreenData

This class keeps information for main Aquire dialog window. It accesses database file 11 through an instance of the class DataFile, AquireDataFile. It has a control function to check if it needs to update its information with one from the database.

Parent Class. None

Child Class. None

Attribute. fdFlag, preData, setZtime(), clearUnFlagField(), setNowData()

Service. nowData, setSearch(), getData()

Class CitationScreenData

This class keeps information for Citation dialog window. It accesses database file 12 through an instance of the class DataFile, CitationDataFile. It has a control function to check if it needs to update its information with one from the database.

Parent Class. None

Child Class. None

Attribute. fdFlag, preData, setDate(), clearUnflagField(), setNowData()

Service. nowData, setSearch(), getData()

Class SpeciesScreenData

This class keeps information for Species dialog window. It accesses database files 11 and 13 through two instances of the class DataFile, AquireDataFile and SpeciesDataFile. It has a control function to check if it needs to update its information with one from the database.

Parent Class. None

Child Class. None

Attribute. fdFlag, flFlag, preData, clearUnFlagField(), searchSpeciesFile, searchAquireFile(), searchZlifstg()

Service. nowData, setSearch(), getData()

Class CasScreenData

This class keeps information for CAS dialog window. It accesses database files 10 and 14 through two instances of the class DataFile, CasnumDataFile and CasDataFile. It has a control function to check if it needs to update its information with one from the database.

Parent Class. None

Child Class. None

Attribute. fdFlag, flFlag, preData, clearUnFlagField(), resetNowData(), searchCasFile(), searchCasnumFile()

Service. nowData, setSearch(), getData()

Class ConcentScreenData

This class keeps information for Concentration-Conf inf-BCF dialog window. It accesses database files 11 and 15 through two instances of the class DataFile, AquireDataFile and ConcentDataFile. AquireDataFile gives information of concentration type and BCF type. ConcentScreenData contains maximum of 12 records of file 15 to fill the dialog window.

Parent Class. None

Child Class. None

Attribute. None

Service. nowData(), getData()

Class PurityScreenData

This class keeps information for Purity and Chemical Characteristics dialog window. It accesses database file 16 through an instance of the class DataFile, PurityDataFile. PurityScreenData contains maximum of 6 records to fill the dialog window.

Parent Class. None

Child Class. None

Attribute. None

Service. nowData, getData()

Class TempScreenData

This class keeps information for Temp-Hardness-Alk-D.O.-pH dialog window. It accesses database file 18 through

an instance of the class DataFile, TempDataFile. TempScreenData contains maximum of 15 records to fill the dialog window.

Parent Class. None

Child Class. None

Attribute. None

Service. nowData, getData()

Class RemarkScreenData

This class keeps information for Remarks dialog window. It accesses database file 17 through an instance of the class DataFile, RemarkDataFile. RemarkScreenData contains maximum of 7 records to fill the dialog window.

Parent Class. None

Child Class. None

Attribute. None

Service. nowData, getData()

Class ControlOpenCloseWindow

This class controls open and close dialog window. It maintains an ID flag for each dialog window. All dialog windows have to signal this class before creating or removing a dialog window.

Parent Class. None

Child Class. None

Attribute. dwOpen

Service. setOpen(), setClose(), isOpen()

Class TMenuStatusLine

TMenuStatusLine inherits properties from its parent class Turbo Vision's TStatusLine. Its function is to display short help information for each menu command at the bottom of the screen.

Parent Class. TStatusLine

Child Class. None

Attribute. None

Service. hint()

Class TDialogApp

TDialogApp is a user environment class. It is the beginning of this window system application. It maintains all dialog windows which are user's interfaces. It opens and closes a dialog window on user's demand. These dialog windows serve both input and output to user.

Parent Class. TApplication

Child Class. None

Attribute. pAquireDialog, pCitationDialog, pSpeciesDialog, pCasDialog, pConcentDialog, pPurityDialog, pTempDialog, pRemarkDialog, Cascade(), outFileSelect(), dosshell(), initMenuBar(), initStatusLine()

Service. handelEvent()

Class TAquireDialog

TAquireDialog is a user interface dialog window. It displays data from AquireScreenData which are retrieved from AquireDataFile. It functions both user's input and output on the same screen.

Parent Class. TDialog

Child Class. None

Attribute. None

Service. handelEvent()

Class TCitationDialog

TCitationDialog is a user interface dialog window. It displays data from CitationScreenData which are retrieved from CitationDataFile. It functions both user's input and output on the same screen.

Parent Class. TDialog

Child Class. None

Attribute. None

Service. handelEvent()

Class TSpeciesDialog

TSpeciesDialog is a user interface dialog window. It displays data from SpeciesScreenData which are retrieved from AquireDataFile and SpeciesDataFile. It functions both user's input and output on the same screen.

Parent Class. TDialog

Child Class. None

Attribute. None

Service. handelEvent()

Class TCasDialog

TCasDialog is a user interface dialog window. It displays data from CasScreenData which are retrieved from CasnumDataFile and CasDataFile. It functions both user's input and output on the same screen.

Parent Class. TDialog

Child Class. None

Attribute. None

Service. handelEvent()

Class TConcentDialog

TConcentDialog is a user interface dialog window. It displays data from ConcentScreenData which are retrieved from AquireDataFile and ConcentDataFile. It functions both user's input and output on the same screen.

Parent Class. TDialog

Child Class. None

Attribute. None

Service. handelEvent()

Class TPurityDialog

TPurityDialog is a user interface dialog window. It displays data from PurityScreenData which are retrieved from PurityDataFile. It functions both user's input and output on the same screen.

Parent Class. TDialog

Child Class. None

Attribute. None

Service. handelEvent()

Class TTempDialog

TTempDialog is a user interface dialog window. It displays data from TempScreenData which are retrieved from TempDataFile. It functions both user's input and output on the same screen.

Parent Class. TDialog

Child Class. None

Attribute. None

Service. handelEvent()

Class TRemarkDialog

TRemarkDialog is a user interface dialog window. It displays data from RemarkScreenData which are retrieved from RemarkDataFile. It functions both user's input and output on the same screen.

Parent Class. TDialog

Child Class. None

Attribute. None

Service. handelEvent()

CHAPTER VI

SUMMARY AND CONCLUSIONS

An object-oriented data model is the result of combining object-oriented programming to database management. An object-oriented data model is now occupying a significant part of the database technology. It is expected to become predominant in tomorrow's database. However, there is not yet a defined standard procedure in designing object-oriented system such that in relational system. Different authors apply different approach and diagraming technique to define the data structure.

There are benefits in developing object-oriented software. One is to reduce the total life cycle of software cost. Another is to set up a software system that resists accidental corruptions. Object-oriented programming is much more easier to modify and extend than conventional-oriented programming. It offers an ability to manage large software projects or create prototypes quickly (Tesler 1986). In the future, the object-oriented database technology will possibly become commonly used data model as the relational model.

The AQWINDOW database retrieval system is designed with the object-oriented program structure. It consists of two subsystems: a data retrieval subsystem and a user interface subsystem. The data retrieval subsystem maintains information of the data files and retrieval methods. The user interface subsystem is a dialog window and menu driven system. It provides users services and queries the database.

The AQWINDOW is a working model of an object-oriented database retrieval system to AQUIRE database. The system can be easily modified to be a general purpose database retrieval system by overloading some class structures defined for AQUIRE data.

There are some restrictions to the ability of AQWINDOW The system takes only one search argument field at a time. The search argument must be in full. Finally the system does not have ability to correct spelling. The above shortcomings are suggested for future system modification. The future research should apply natural language to user interface subsystem. This will extend the retrievability and return more informative data than the defined set of command and dialog window.

REFERENCES

Alexander, J. H. and M. J. Freiling. 1985. Smalltalk-80 Aids Troubleshooting System Development. System and Software. Volume 4, Number 4 (April): 111-118.

AQUIRE 1989. U.S. Environmental Protection Agency, Duluth, Minnesota.

Baase, Sara. 1983. VAX-11 Assembly Language Programming. Englewood Cliffs, New Jersey: Prectice-Hall.

Brown, Linfield C. and Thomas O. Branwell, Jr. 1987. The Enhanced Stream Water Quality Models QUAL2E and QUAL2E-UNCAS: Documentation and User Model. Athens, GA: Environmental Research Laboratory.

Booch, Grady. 1986. Object-Oriented Development. IEEE Transactions on Software Engineering. Volume SE-12, Number 2 (Feburary): 211-221.

Christian, Kaare. 1993. Borland C++ Techniques & Utilities. Emeryville, CA: Ziff-Davis Press.

Coed, Peter and Edward Yourdon. 1991a. Object-Oriented Analysis. Englewood Cliffs, New Jersey: Yourdon Press Computing Series.

Coed, Peter and Edward Yourdon. 1991b. Object-Oriented Design. Englewood Cliffs, New Jersey: Yourdon Press Computing Series.

Coed, Peter and Jill Nicola. 1993. Object-Oriented Programming. Englewood Cliffs, New Jersey: Yourdon Press Computing series.

Cooke, Nancy M. and James E. McDonald. 1986. A Formal Methodology for Acquiring and Representing Expert Knowledge. Proceedings of the IEEE 74 (October).

Diederich, J. and J. Milton. 1989. Objects, Messages, and Rules in Database Design. In Object-Oriented Concepts, Databases, and Applications. ed. W. Kim and F. H. Lochovsky, 177-215. Reading, Massachusettes: Addison-Wesley Publishing Co.

Dittrich, Klaus R. 1986. Object-Oriented Database System: the Notion and the Issue. International Workshop on Object-Oriented Database System (September).

Duncan, Ray. 1991. Power Programming: A Look at Differences Between C and C++. PC Magazine (July): 444.

Davis, J. R. and P. M. Nanninga. 1985. GEOMYCIN: Towards a Geographic Expert System for Resource Management. Journal of Environmental Management 21: 377-390.

Georgeff, Michael P. and Amy L. Lansky. 1986. Procedural Knowledge. Proceedings of the IEEE 74 (October): 1383-1398.

Hu, David. 1990. Object-Oriented Environment in C++. Portland, Oregon: Management Information Source, Inc.

Kim, W., N. Ballou, and H. Chau. 1989. Features of the ORION Object-Oriented Database System. In Object-Oriented Concepts, Databases, and Applications. ed. W. Kim and F. H. Lochovsky, 309-337. Reading, Massachusettes: Addison-Wesley Publishing Co.

Leung, K.S. and M.H. Wong. 1990. An Expert System Shell Using Structured Knowledge. Computer (March): 38-47.

Manola, Frank. 1990. Object-Oriented Knowledge Bases. AI Expert (March): 26-36.

Meyer, Bertrand. 1987. Reusability: The Case for Object-Oriented Design. IEEE Software. (March): 50-64.

Nelson, Ross P. 1991. Microsoft's 80386/80486 Programming Guide. Redmond, Washington: Microsoft Press.

Papas, Chris H. and William H. Murray. 1990. Turbo C++ Professional Handbook. New York: McGraw-Hill, Inc.

Parker, Sandra C. and C. Ray Asfahl, eds. 1988. An Industrial Chemical Hazards Database With a Natural Language Interface: An Application of Artificial Intelligence. Computer & Industrial Engineering 15: 443-445.

Peterson, Robert W. 1987. Object-Oriented Database Design. AI Expert. (March): 26-31.

Rishe, Naphtali. 1992. Database Design The Semantic Modeling Approach. New York: McGraw-Hill, Inc.

Stern, A. M. and C. R. Walker. 1978. Hazard Assessment of Toxic Substances: Environment Fate Testing of Organic Chemicals and Ecological Effects Testing. In Estimating the Hazard of Chemical Substances to Aquatic Life. ASTM STP 657. John Cairns, Jr., K. L. Dickson, and A. W. Maki, eds. American Society for Testing and Materials: 81-131.

Stonebraker, Michael and Lawrance A. Rowe. 1986. The Design of POSTGRES. In Procedding 1986 ACM SIGMOD International Conference on the Management of Data. (June).

Tello, Ernest R. 1991. Object-Oriented Programming for Windows. New York: John Wiley & Sons, Inc.

Tesler, Larry. 1986. Programming Experiences. Bytes. (August): 195-205.

Tokoro, Mario and Yutaka Ishikawa. 1984. An Object-Oriented Approach to Knowledge Systems. Proceeding of the International Conferences on Fifth Generation Computer Systems.

Webster, Bruce F. 1989. The NeXT Book. Reading, Massachusetts: Addison-Wesley Publishing Co.

Weise S. P. and F. H. Lochovsky 1989. OZ+: an Object-Oriented Database System. In Object-Oriented Concepts, Databases, and Applications. ed. W. Kim and F. H. Lochovsky, 309-337. Reading, Massachusettes: Addison-Wesley Publishing Co.

Wiener, Richard S. and Lewis J. Pinson. 1988. An Introduction to Object-Oriented Programming and C++. Reading, Massachusettes: Addison-Wesley Publishing Co.

Wirfs-Brock, Rebecca, Brian Wilkerson, and Lauren Wiener. 1990. Designing Object-Oriented Software. Englewood Cliffs, New Jersey: Prentice Hall.

Zaniolo, Carlo and Ait-Kaci, Hassan eds. 1986. Object Oriented Database System and Knowledge Systems. In Expert Database Systems, Proceedings From the First International Workshop, ed. Larry Kerschberg, 49-65. University of South Carolina: The Benjamin/Cummings Company, Inc.

APPENDIXES

APPENDIX A

INFORMATION TO SETUP AND RUN THIS PROGRAM

This database retrieval system requires at least 7.23MB free hard disk space on a working computer system. The hard disk requirement is for install a subset of AQUIRE database and the retrieving program. The full database will approximately require ten times the hard disk space for this subset of AQUIRE database.

Setup and Run

step 1

Make two directories named DATA and PROGRAM. These two must be sub-directories of the same directory and must be at the same level. The directory DATA will contain AQUIRE data files and their information files. The directory PROGRAM will contain the retrieving program.

step 2

Copy data files and information files to directory DATA. These file names are listed in table 3. Copy program file AQWINDOW.EXE to directory PROGRAM.

step 3

Change current directory to PROGRAM. Type command AQWINDOW.EXE to run the program.

TABLE 3

LIST OF FILES AFTER SETUP

Directory	File Name	Size	File Description
DATA	10	733312	data file 10
	PART11	875823	partial data file 11
	PART12	811432	partial data file 12
	13	191420	data file 13
	PART14	814053	partial data file 14
	PART15	803886	partial data file 15
	PART16	826393	partial data file 16
	PART17	805224	partial data file 17
	PART18	804679	partial data file 18
	CASNUM.INF	55	information for file 10
	AQTOX.INF	956	information for file 11
	RETOX.INF	416	information for file 12
	SPECIES.INF	111	information for file 13
	STOCAS.INF	164	information for file 14
	STOCCB.INF	98	information for file 15
	STOCHR.INF	166	information for file 16
	STOREM.INF	122	information for file 17
	STOTHA.INF	96	information for file 18
PROGRAM	AQSQL.EXE	188954	command line program
	AQWINDOW.EXE	544184	window menu program

APPENDIX B

USER'S MANUAL

Program Information

This application is designed and tested on an IBM PC AT class computer with Intel 80386 microprocessor and DOS operating system.

The author writes this program with C++ object-oriented programming language. The object-oriented user interface allows the user to go to any level of command directly from any where in the application. The user can access a dialog window, menu command, or exit from the application at any time.

This program is compiled with Borland C++ version 3.1. The initial data retrieval system module compiled with Borland's Turbo C++ version 2 and later compiled with Borland C++ version 3.1 with some modifications. The user interface module is constructed by deriving classes from Turbo Vision's class library. Turbo Vision is a part of Borland C++ with Application Framework. It is a software development's tool kit.

This application supports both key board and mouse for user input. However, the user may use this application easier with keyboard and mouse than with keyboard alone.

Desktop

A Desktop is the whole computer screen. It is divided into three sections: menu bar, working area, and status line. A menu bar provides a user with choice of commands to access the database and to manage dialog windows. A working area is an area where all dialog window will be placed on. A status line shows some short keyboard shortcuts and information to menu items. Figure 7 shows an initial desktop with three screen sections.

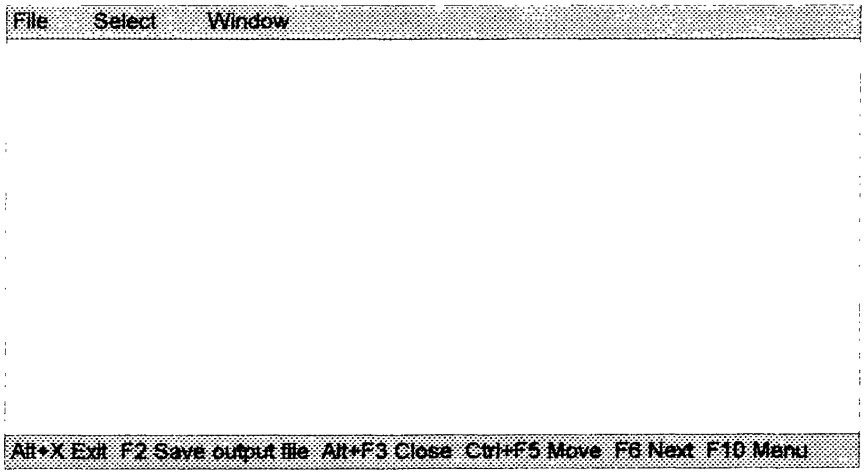


Figure 7. Initial Desktop

Menu Bar

The menu line on top of the screen is called a menu bar. It consists of three submenu titles: file submenu, select submenu, and window submenu. These are pull down submenus.

A pull down submenu can be activated either by highlighting the desired submenu then press Enter key or positioning a mouse's pointer at the submenu then click the mouse's left button. Each submenu also has a highlighted character, which, when combined with an Alt key will pull down that submenu. For example, a user can press Alt+F to pull down the file submenu. Submenus consists of menu items.

A menu item has a highlighted character that can be typed to select that item. An accelerator for a menu item is listed to the right of a menu item. The accelerator key can be used without activating the submenu. A menu item that followed by an ellipsis leads to a dialog box.

Table 3, shows all menu items and their shortcut keys of this menu system.

Working Area

The working area section occupies most of the desktop space. It only service is to place the pull down menus and dialog windows on.

Status Line

The status line section is a single line at the button of the screen that lists some keyboard shortcuts. It serves as a short information of the menu command. The status line command is automatically invoked when either a submenu or menu item is activated (highlighted). It displays a brief information about the submenus and menu items.

TABLE 4

LIST OF SUBMENU COMMANDS

Submenu	Menu Item	Accelerator Key
File	Select output file	F3
	Save to output file	F2
	DOS shell	
	Quit	Alt+X
Select	AQUIRE	Alt+A
	Citation	Alt+I
	Species	Alt+S
	CAS	Alt+C
	Concentration-Conf. Inf-BCF	Alt+O
	Purity/Chem Characteristic	Alt+H
	Temp-Hardness-Alk-D.O.-pH	Alt+T
	Remark	Alt+R
Window	Move	Ctrl+F5
	Next	F6
	Cascade	
	Close	Alt+F3

Dialog Window

A dialog window is an interface between a user and the database. Each window has a name labeled on top. The dialog window displays database fields with their appropriate labels. Some are searchable fields, some are information only fields. The dialog window also has one or more buttons. This button directs the user to either another related dialog window or another related record of the same database file.

This application consists of 8 dialog windows which are named as following.

1. Main AQUIRE Database
2. Citation Information
3. Species Information
4. Chemical Information
5. Concentration-Confidence interval-BCF Information
6. Purity/Chemical Characteristic Information
7. Temperature-Hardness-Alk-D.O.-pH Information
8. Remarks to AQUIRE Information

The Main AQUIRE Database dialog window is the center of the retrieval system. The user can directly switch to other dialog windows with related information at any time. The system will update the information in the dialog window as the user switches from one dialog window to another.

Search Field

The searchable field labels with a highlighted character. The user can moves from one field to another with Tab key on the keyboard. The searchable field accepts a case-insensitive string as a search argument. However, it does not check for a correct spelling. The search argument must be in full. The system compares the search argument with a field of record. It will not match a search argument with a partial field of record.

The system will look for a field of data matched to a search argument after the user types in the searched value and presses the Enter key. The search argument is limited to one field at a time. If the user enter more than one search field, the system will takes the first argument in its order as a search argument and discards the others.

While the system searches the database, the user cannot move the cursor or move from one field to another. If the system has found the matched data, it will update the dialog window with the new information. If it does not find a matched data, it will either clear the dialog window or return to the previous information. Then the user can resume the operation.

Button

A Button is another component of the dialog window. It either directs the user to another related record within

another file or within the same file. The user activates a button by either moves the cursor with Tab key to the button then press enter or positions the mouse's pointer on it then press the mouse button. Each button has a highlighted character label, the user can activate it using Alt key combines with the highlighted character.

If the button directs the user to another file, it will open a relevant dialog window and update its information with a related record from the database. If it directs the user to a related record within the same files, it will update its information with the related record.

The direct access button is equivalent to the menu item of the select menu. It sends the same command to the system, but it works as a shortcut key. The Button updates the information on a dialog window using the same method as the user enters a search argument field. This will eliminate a misspell or error typing. The user will quickly and easily move from one dialog window to another.

Menu Command

The menu system consists of three submenus: file submenu, select submenu, and window submenu. Each submenu consists of menu items. Followings are descriptions of each menu item.

File Submenu

The file submenu command consists of four commands. These commands are select output file, save to output file, DOS shell, and quit command.

Select Output File Command (F3). This command let a user to choose an alternate output file name. The current dialog window data will save to this output file whenever the user issues the save to output file command (F2). At the beginning of the application, the default file is set to "out.dat".

This command leads user to a dialog box. The dialog box displays the current selected output file name, cancel button, OK button, and file dialog button. The user has choices to cancel the command, select a new file name, or use the file dialog command. The file dialog command will display a list of existent file name that can be selected at user's convenience.

The user may choose any existent or non-existent file name. If the user selects an existent file name, the save to output file command (F2) will append the data to the selected output file. If the user selects (or enters) a non-existent file name, the save to output file command will create the selected file and write the data to that file. After the output file is created, the later saved data will append to the selected output file.

The user may change the output file any time during running the application. However, if the selected output file is currently a non-existent file, the application will not create the selected file until the user issues the save to output file (F2).

Save to Output File (F2). This command saves the current data of a dialog window to an output file. The default output file is "out.dat". It saves the data only of the current, active, dialog window. It does not save the data of all running dialog window. If the user wants to save the data of all dialog window, he or she has to save it one by one.

After the data is saved, the system will display an information dialog to confirm the operation. The user may either press Enc or Enter key to acknowledge the information.

This command does nothing if it is activated without any dialog window opened.

DOS Shell. This command will temporarily exit from the application. This command is designed for a user to access DOS (or others) command. The user may use this command to send an output file to a printer or open a text editor to edit an output file. The user may go back to the application by issuing "exit" command.

Quit (Alt+X). This command will terminate all functions then quit the application. The user may issue this command at any time to quit the application.

Select Submenu

This submenu consists of eight dialog window commands. A dialog window is the mean access to the database. Each dialog window can be independently opened or closed. Figure 8, shows the Select submenu with its menu items. Following are list of the dialog window commands with its shortcut key in the parenthesis.

1. AQUIRE (Alt+A)
2. Citation (Alt+I)
3. Species (Alt+S)
4. CAS (Alt+C)
5. Concentration-Conf. Int-BCF (Alt+O)
6. Purity/Chem Characteristic (Alt+H)
7. Temp-Hardness-Alk-D.O.-pH (Alt+T)
8. Remark (Alt+R)

Window Submenu

The window submenu consists of four window manipulation commands. These command are move, next, cascade, close.

Move (Ctrl+F5). This command moves a dialog window and place it any place on the working area. It is useful if the user wants to see the information from more than one dialog

window. The user activated this command then use an arrow key on the key board to move the dialog window to the desired location. The user sets down the dialog window by press enter key. The user may cancel the command by pressing the Esc key, the dialog window will set back to its previous location.

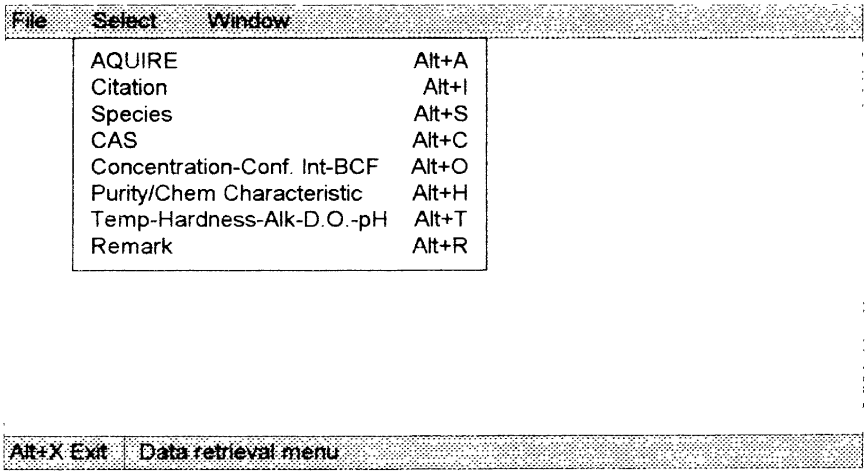


Figure 8. Select Submenu With Menu Items and Status Line

Next (F6). This command will make the next dialog window become active. The next dialog window is the dialog window in the order of creation. The active dialog window is the dialog window that placed in front of the others. The user may use this command to find specific dialog window or select a specific dialog window from the select submenu.

Cascade. This command will place all opened dialog windows at their original location. The original location is the location when a dialog window first created.

Close (Alt+F3). This command closes the active dialog window. Even though a dialog window is closed, its data is saved. When this dialog window is opened again it will display the original data, unless its related data of another dialog window is changed.

APPENDIX C

SAMPLE OUTPUTS FROM AQWINDOW

Followings are samples of retrieved information from AQWINDOW. The output data are saved into the default output file (data.out). Each example starts from a different search argument. Then, the author retrieves related data from other files through related dialog window.

Sample Output 1

Following is the retrieved data using citation's reference number 885 as an argument. The data following citation information are related chemical and species used in the test related to reference number 885.

Citation Information

```
          Author(s): Sanders,H.O.
    Publication Year: 1969 ( )
      Citation Title: Toxicity of Pesticides to the
                      Crustacean Gammarus lacustris
  Source of Citation: Tech. Paper No. 25, Bur. Sports Fish
                      Wildl., Fish Wildl. Serv., U.S.D.I.:
                      18 P. (Used with Reference 732)
    Reference Number: 885
      Reference Type: A
  This Record Number: 108
       Inserted Date: 9/20/82
        Updated Date: 0/0/0
```

Main AQUIRE Database

```
  This Record Number: 6
```

Modified Date: 1/31/85
First Author Name: SANDERS
Publication Year: 69
Citation Ref. #: 885
Test Number: 0
Total Test: 0
Reviewer: 7
Review Code: 2
Test Field: Lab
Test Media: Fresh water
Study Type:
Species Number: 6
Species Life Stage: 2 MO
Control: Indeter
Test Effect: LC50
Test Time 1: (max) 96 Hours
Test Time 2: NR
Exposure Type: S
Test Method: Unmeasured

Species Information

Latin Name: Gammarus lacustris
Common Name: Scud
Species Life Stage: 2 MO
Major Group Code: CR
Minor Group Code: AMPH
Species Ref. #: 31
Species Number: 6

Chemical Information

Chemical Name: (2,4-DICHLOROPHENOXY)ACETIC ACID, ESTER WITH 1,2-PROPANEDIOL, MONOBUTYL ETHER
CAS Number: 1320189
Chemical Type: TEST
This Record Number: 6

Concentration-Confidence interval_BCF Information

1st. Data

Concentration Type: F
BCF Type: NR
Concentration Value: 2100 to: NR
Confidence int. Value: 1700 to: 2500
BCF Value: NA to:

2nd. Data

Concentration Type: NR

```
            BCF Type: NR
  Concentration Value:              to:
confidence int. Value:              to:
            BCF Value:              to:

          Purity/Chemical Characteristic Information

          Chemical Characteristic          Purity Code

       1. EC                               NR
       2.
       3.
       4.
       5.
       6.

          Temperature-Hardness-Alk-D.O.-pH Information

                       value    from     to

       Temperature:    21.1     NR
          Hardness:    NR
              Alk.:    30.0     NR
              D.O.:    NR
               pH.:    7.1      NR

          Remarks to AQUIRE Information

      Remarks

   1.
   2. TDS = 88.0, MG = 3.1 PPM, CA = 7.1 PPM//
   3. ORGANISMS EXHIBITING 1ST SIGNS OF POISONING DID NOT
       SURVIVE
   4. O EFCT CONT/TRANSFER TO CLEAN H2O//
   5.
   6.
   7.
```

Sample Output 2

Following is the retrieved data using main AQUIRE database's record number 4567 as an argument. The data following main AQUIRE database are related citation, chemical, and species information used in the test related to main AQUIRE database's record number 4567.

Main AQUIRE Database

This Record Number: 4567
Modified Date: 1/31/85
First Author Name: KUMARAGURU
Publication Year: 81
Citation Ref. #: 5256
Test Number: 0
Total Test: 0
Reviewer: 6
Review Code: 2
Test Field: Lab
Test Media: Fresh water
Study Type:
Species Number: 4
Species Life Stage: 0.89 G ; 0.8 - 1.2 G
Control: Indeter
Test Effect: LC50
Test Time 1: (max) 384 Hours
Test Time 2: NR
Exposure Type: F
Test Method: Measured

Citation Information

Author(s): Kumaraguru,A.K. and F.W.Beamish
Publication Year: 1981 ( )
Citation Title: Lethal Toxicity of Permethrin (NRDC-143) to Rainbow Trout, Salmo gairdneri, in Relation to Body Weight and Water Temperature
Source of Citation: Water Res. 15(4):503-506
Reference Number: 5256
Reference Type: A
This Record Number: 199
Inserted Date: 9/20/82
Updated Date: 0/0/0

Species Information

Latin Name: Oncorhynchus mykiss
Common Name: Rainbow trout,donaldson trout
Species Life Stage: 0.89 G ; 0.8 - 1.2 G
Major Group Code: OS
Minor Group Code: SALM
Species Ref. #: 1
Species Number: 4

Chemical Information

Chemical Name: PERMETHRIN
CAS Number: 52645531

```
        Chemical Type: TEST
   This Record Number: 4561

       Concentration-Confidence interval_BCF Information

            1st. Data

   Concentration Type: F
            BCF Type: NR
  Concentration Value: 3.17          to: NR
Confidence int. Value: 2.79          to: 5.78
            BCF Value: NA            to:

            2nd. Data

   Concentration Type: NR
            BCF Type: NR
  Concentration Value:               to:
Confidence int. Value:               to:
            BCF Value:               to:
```

Sample Output 3

Following is the retrieved data using chemical name "cadmium chloride" as an argument search for chemical information dislog window. The data following chemical information are related citation and species used in the test related to cadmium chloride.

```
            Chemical Information

        Chemical Name: CADMIUM CHLORIDE
           CAS Number: 10108642
        Chemical Type: TEST
   This Record Number: 29

            Main AQUIRE Database

   This Record Number: 29
        Modified Date: 1/31/85
    First Author Name: QURESHI
     Publication Year: 80
      Citation Ref. #: 5288
          Test Number: 0
           Total Test: 0
             Reviewer: 2
```

Review Code: 2
Test Field: Lab
Test Media: Fresh water
Study Type:
Species Number: 532
Species Life Stage: NR
Control: Satisfy
Test Effect: LC50
Test Time 1: (max) 192 Hours
Test Time 2: NR
Exposure Type: S
Test Method: Unmeasured

Citation Information

Author(s): Qureshi,S.A., A.B.Saksena, and V.P.Singh
Publication Year: 1980 (A)
Citation Title: Acute Toxicity of Four Heavy Metals to Benthic Fish Food Organisms From the River Khan, Ujjain
Source of Citation: Int. J. Environ. Stud. 15(1):59-61
Reference Number: 5288
Reference Type: A
This Record Number: 633
Inserted Date: 9/20/82
Updated Date: 0/0/0

Species Information

Latin Name: Tubifex tubifex
Common Name: Tubificid worm
Species Life Stage: NR
Major Group Code: AN
Minor Group Code: OLIG
Species Ref. #: 10
Species Number: 532

Concentration-Confidence interval_BCF Information

1st. Data

Concentration Type: T
BCF Type: NR
Concentration Value: 320000/ to: NR
Confidence int. Value: to:
BCF Value: NA to:

2nd. Data

Concentration Type: NR

BCF Type:	NR	
Concentration Value:		to:
Confidence int. Value:		to:
BCF Value:		to:

Temperature-Hardness-Alk-D.O.-pH Information

	value	from	to
Temperature:	28	NR	
Hardness:	224	NR	
Alk.:	180	NR	
D.O.:	8	NR	
pH.:	8.5	NR	

VITA

Kumpera Kulpaiboon

Candidate for the Degree of

Master of Science

Thesis: AN OBJECT-ORIENTED DATABASE RETRIEVAL SYSTEM FOR AQUATIC TOXICITY DATA FILES

Major Field: Computer Science

Biographical:

Personal Data: Born in Bangkok, Thailand, March 2, 1960, the son of Somphot and Jamnean Kulpaiboon.

Education: Received Bachelor of Science Degree in Mathematic from Ramkhamhaeng University, Bangkok, Thailand in April, 1983; received Master of Business Administration Degree in Management Information System from Oklahoma City University, Oklahoma City, Oklahoma in December, 1987; completed requirements for the Master of Science degree at Oklahoma State University in December, 1993.

Professional Experience: Research Assistant, Department of Computer Science, Oklahoma State University, June, 1993, to July, 1993.