

**AnaEx: An Expert System Solution to Anaerobic  
Bacterial Identifications**

---

A Thesis

Presented to

The Faculty of the Department of Computer Science

University of Houston

---

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

---

By

Gerges A Gad, MT (ASCP)

December, 2001

# AnaEx: An Expert System Solution to Anaerobic Bacterial Identifications

---

---

Gerges A. Gad

**APPROVED:**

---

Dr. Robert Anderson, Advisor

---

Dr. Jaspal Subhlok

---

Dr. Kathy A. Johnson

---

Dean, College of Natural Sciences  
and Mathematics

## **Acknowledgements**

First and foremost, I offer my thanks and gratitude to God, without Whom I would not be able to do anything. My special thanks go to Dr. Robert Anderson, for his guidance and support throughout this study. I would also like to thank Dr. Jaspal Subhlok and Kathy A. Johnson of the School of Allied Health Sciences, University of Texas – Houston for serving on my thesis committee. A special thanks to Dr. Johnson for assisting me with database design concepts directed to Health Informatics. I also wish to acknowledge my manager, Diane Trippel, who supported me in my studies. Special thanks goes to Becky Johnson, MT(ASCP) of Memorial-Hermann Hospital Southwest's Microbiology Laboratory and Fran Schaffer, MT(ASCP) of Memorial-Hermann Hospital's Microbiology Laboratory for their assistance in the project. Last, but certainly not least, my deepest gratitude goes to my mother Julia and my sister Mariam for their constant encouragement and total support in my attainment of this goal. Finally, I dedicate this work in memory of my dearly departed father, Abdelmesih, who left this vain world to rest in the Paradise of Joy with the angels and saints.

**AnaEx: An Expert System Solution to Anaerobic  
Bacterial Identifications**

---

An Abstract of a Thesis

Presented to

The Faculty of the Department of Computer Science

University of Houston

---

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

---

By

Gerges A Gad, MT (ASCP)

December, 2001

## **Abstract**

Expert Systems are computer programs that use artificial intelligence to solve problems within a specialized domain that ordinarily requires human expertise. In the clinical laboratory setting, a very important aspect of helping physicians with diagnosis of infections is the identification of microorganisms that are causing these infections. Most outstanding of these infections are infections caused by anaerobic bacteria. These microorganisms are involved in virtually any type of bacterial infection in humans. They thrive in areas of the body that have low levels of oxygen. Their identification, although very important, is often difficult. This thesis addresses this problem by providing an expert system that takes into consideration various clinical factors about the microorganism, and presents identification as a possible solution.

# Table of Contents

0.	Introduction .....	1
1.	Artificial Intelligence .....	2
1.1	Artificial Intelligence .....	2
1.2	Expert Systems.....	4
1.3	Knowledge Engineering.....	10
2.	Task Domain.....	12
2.1	Microbiology.....	12
2.2	Taxonomy of Microorganisms.....	15
2.3	Anaerobic Bacteriology.....	20
2.4	Current Identification Methods of Anaerobic Bacteria .....	21
2.5	Limitations of Manual Methods .....	25
3.	AnaEx: A Solution.....	27
3.1	RapID ANA II.....	27
3.2	AnaEx.....	31
3.3	Benefits of Using Expert Systems.....	34
3.4	Limitations of Using Expert Systems .....	35
4.	System Design.....	37
4.1	Targeting a Decision .....	37
4.2	Dependency Diagrams .....	38
4.3	Decision Tables.....	46
4.4	Knowledge-Base Rules.....	50
5.	Using VP-Expert .....	53
5.1	VP-Expert.....	53
5.2	Prototype Design.....	56
5.3	Advantages of VP-Expert.....	57
5.4	Limitations of VP-Expert .....	64
6.	Using VB With Access.....	66
6.1	Visual Basic and Access .....	66
6.2	Database Design.....	68
6.3	User Interface Design .....	77
6.4	Limitations .....	82
6.5	Correlation to ES.....	83
7.	Test Results.....	90
8.	Conclusion .....	93
8.1	Synopsis.....	93
8.2	Future Work .....	94
8.3	Final Words.....	97
	Bibliography.....	99
	Appendix A .....	102
	Knowledge Base of AnaEx .....	102

## List of Figures

Figure 2.1 Structure of the Bacterial Cell .....	13
Figure 2.2 Cell wall of gram-positive bacteria .....	14
Figure 2.3 Cell wall of gram-negative bacteria .....	15
Figure 2.4 Gram-Positive Oral Species .....	18
Figure 2.5 Gram-Negative Oral Species .....	19
Figure 2.6 Shapes of Bacteria .....	25
Figure 4.1 Block Diagram of the area under study .....	38
Figure 4.2 Block Diagram of the Decision .....	39
Figure 4.3 Aerotolerance Test of the Decision Block Diagram (expanded) .....	40
Figure 4.4 Microcode Test of the Decision Block Diagram (expanded) .....	41
Figure 4.5 Dependency Diagram .....	45
Figure 4.6 Decision Table Plan for Final Rule Set .....	46
Figure 4.7 Completed Decision Table .....	48
Figure 4.8 Final Reduced Decision Table .....	50
Figure 5.1 Consultation Screen .....	58
Figure 5.2 Example of the "How?" Debug Feature .....	60
Figure 5.3 Example of the "Why?" Debug Feature .....	60
Figure 5.4 Text Trace of a Consultation Session .....	62
Figure 5.5 Partial Graphic Trace of a Consultation Session .....	63
Figure 6.1 SOM Diagram of the Reference Tables .....	72
Figure 6.2 SOM Diagram for the Knowledge Base Tables .....	73
Figure 6.3 SOM Diagram for the Specimen Table .....	74
Figure 6.4 Relationship Diagram for the AnaEx Database .....	76
Figure 6.5 An Example of the Data in the Knowledge Base Tables .....	77
Figure 6.6 First Form in AnaEx System .....	78
Figure 6.7 Dialog Boxes for the aerotolerance test .....	79
Figure 6.8 Dialog Boxes for Gram Stain and Specimen Source .....	80
Figure 6.9 Bio-Chemical Test Results Form .....	80
Figure 6.10 AnaEx Microorganism ID Report .....	81
Figure 7.1 Test Results .....	92

## List of Tables

Table 2.1 Fundamental physical differences between eukaryotes and prokaryotes .....	17
Table 3.1 Example of the RapID ANA II System Code Compendium Data .....	29
Table 5.1 Data Dictionary for compend.dbf .....	56
Table 6.1 Object Relationship for the AnaEx Database .....	75
Table 7.1 Test Results .....	91

# **Chapter 0**

## **0. Introduction**

Expert Systems are computer programs that use artificial intelligence to solve problems within a specialized domain that ordinarily requires human expertise. In the clinical laboratory setting, a very important aspect of helping physicians with diagnosis of infections is the identification of pathological microorganisms. Most outstanding of these infections are infections caused by anaerobic bacteria. These microorganisms are involved in virtually any type of bacterial infection in humans. They thrive in areas of the body that have low levels of oxygen. Their identification, although very important, is often difficult.

This thesis addresses this problem by providing an expert system that takes into consideration various clinical factors about the microorganism, and presents identification as a possible solution. The project involves developing an expert system using VP-Expert, an expert system shell. After this initial prototype design, the system is built using VB and Access, for some added enhancements. Testing of clinical data is performed on the system and compared to the manual method performed in the laboratory. A correlation between the two is observed and results recorded. A presentation to the clinical laboratory staff is offered, and results of the presentation are recorded.

# Chapter 1

## 1. Artificial Intelligence

### *1.1 Artificial Intelligence*

Expert Systems are computer programs that are derived from a branch of computer science research called **Artificial Intelligence (AI)**. AI's scientific goal is to understand intelligence by building computer programs that exhibit intelligent behavior. The term *intelligence* covers many cognitive skills, including the ability to solve problems, learn, and understand language; AI addresses all of these. It is concerned with the concepts and methods of symbolic inference, or reasoning, by a computer, and how the knowledge used to make those inferences will be represented inside the machine.

AI researchers are active in a variety of **domains**. These domains include: Formal Tasks (mathematics, games), Mundane Tasks (perception, robotics, natural language, common sense reasoning) and Expert Tasks (financial analysis, medical diagnostics, engineering, scientific analysis, and other areas).

Artificial intelligence can be viewed from a variety of perspectives. From the perspective of intelligence, artificial intelligence is making machines “intelligent”. It means they would be acting, as we would expect people to act.

From a *research* perspective, artificial intelligence is the study of how to make computers do things, which, at the moment, people do better [1]. AI began in the early 1960s. The first attempts were game playing (checkers), theorem

proving (a few simple theorems) and general problem solving (only very simple tasks). General problem solving was much more difficult than originally anticipated. Researchers were unable to tackle problems routinely handled by human experts.

From a *business* perspective, AI is a set of very powerful tools, and methodologies for using those tools to solve business problems. From a *programming* perspective, AI includes the study of symbolic programming, problem solving, and searching. Typically, AI programs focus on symbols rather than numeric processing.

AI programming languages include LISP and PROLOG. LISP (LISt Processor), developed in the 1950s, is the early programming language strongly associated with AI. LISP, based on lambda calculus, is a functional programming language with procedural extensions. LISP was specifically designed for processing heterogeneous lists - typically a list of symbols. Features of LISP that made it attractive to AI researchers included run-time type checking, higher order functions (functions that have other functions as parameters), automatic memory management (garbage collection) and an interactive environment. Because of its simple elegance and flexibility, most AI research programs are written in LISP, but commercial applications have now moved away from LISP.

The second language strongly associated with AI is PROLOG (Programming in Logic). It was developed in the 1970s. PROLOG, based on first-order logic, is declarative in nature and has facilities for explicitly limiting the search space. PROLOG consists of English-like statements that are facts

(assertions), rules (of inference), and questions. Programs written in PROLOG have behavior similar to rule-based systems written in LISP.

Object-oriented languages are a class of languages more recently used for AI programming. Important features of object-oriented languages include: concepts of objects and messages, and inheritance (object hierarchy where objects inherit the attributes of the more general class of objects). Other important aspects of the Object-Oriented Paradigm are: *encapsulation, abstraction, and polymorphism*. Examples of object-oriented languages are Smalltalk, Objective C, and C++. Object-oriented extensions to LISP (CLOS - Common LISP Object System) and PROLOG (L&O - Logic & Objects) are also used.

## ***1.2 Expert Systems***

AI programs that achieve expert-level competence in solving problems in task areas by bringing to bear a body of knowledge about specific tasks are called *knowledge-based systems (KBS) or expert systems (ES)*. Often, the term “expert systems” is reserved for programs whose knowledge base contains the knowledge used by human experts, in contrast to knowledge gathered from textbooks or non-experts. More often than not, the two terms, expert systems and knowledge-based systems, are used synonymously. Taken together, they represent the most widespread type of AI application. The area of human intellectual endeavor to be captured in an expert system is called the *task domain*. **Task** refers to some goal-

oriented, problem-solving activity. **Domain** refers to the area within which the task is being performed.

Definitions of expert systems vary. Some definitions are based on function, while others are based on structure. Some definitions have both functional and structural components. Many early definitions assume rule-based reasoning.

According to *Encyclopedia Britannica*, an expert system is defined as “a computer program that uses artificial intelligence to solve problems within a specialized domain that ordinarily requires human expertise” [14].

Dr. Edward Shortliffe at Stanford University developed one of the first programs popularly termed an “expert system” in the early 1970’s. It recommended the selection of antibiotics based on clinical data such as the site of infection and associated medical conditions. While not the first decision support program, it was the first to use symbolic knowledge in a rule-based format.

Over the past few decades, **knowledge engineering**, the computer programming methods used to create expert systems, have been incorporated into the standard software engineering repertoire of techniques.

Expert systems are now used routinely in a variety of industries. As stand-alone products, they are used by various industries to make rapid decisions. For example, credit card companies may use them to make decisions about extending credit for individual transactions by customers. When embedded inside another product, they are used in products such as grammar checkers and “wizards” in

popular spreadsheet and graphics packages. A related technology, *neural networks*, is used in devices such as refrigerators and air conditioners [20].

Knowledge-based expert systems, or simply expert systems, use human knowledge to solve problems that normally would require human intelligence. These expert systems represent the knowledge as data or rules within the computer. These rules and data can be called upon when needed to solve problems. Books and manuals have a tremendous amount of knowledge but a human has to read and interpret the knowledge for it to be used. Conventional computer programs perform tasks using conventional decision-making logic containing little knowledge other than the basic algorithm for solving that specific problem and the necessary boundary conditions. This program knowledge is often embedded as part of the programming code, so that as the knowledge changes, the program has to be changed and then rebuilt. Knowledge-based systems collect the small fragments of human know-how into a knowledge base, which is used to reason through a problem, using the knowledge that is appropriate. A different problem, within the domain of the knowledge base, can be solved using the same program without reprogramming.

Knowledge-based expert systems, or simply expert systems, use human knowledge to solve problems that normally would require human intelligence. These expert systems represent the knowledge as data or rules within the computer. These rules and data can be called upon when needed to solve problems. Books and manuals have a tremendous amount of knowledge but a human has to read and interpret the knowledge for it to be used. Conventional

computer programs perform tasks using conventional decision-making logic containing little knowledge other than the basic algorithm for solving that specific problem and the necessary boundary conditions. This program knowledge is often embedded as part of the programming code, so that as the knowledge changes, the program has to be changed and then rebuilt. Knowledge-based systems collect the small fragments of human know-how into a knowledge base, which is used to reason through a problem, using the knowledge that is appropriate. A different problem, within the domain of the knowledge base, can be solved using the same program without reprogramming. Expert systems usually contain two components: a *knowledge base* and an *inference engine* program, enabling it to infer conclusions.

The **knowledge base** is where the knowledge of one or more human experts in a specific field or task is stored. The knowledge base an expert uses is what he learned at school, from colleagues, and from years of experience. Presumably the more experience he has, the larger his store of knowledge. Knowledge allows him to interpret the information to formulate diagnosis, design, and analysis.

The knowledge base is set up as an “intelligent” database in that it can usually manipulate the stored information in a logical, natural, or methodical way. It can conduct searches based on predetermined rules of defined associations and relationships, as well as the more traditional data search techniques [15].

The knowledge base is usually made up of *factual knowledge* and sometimes even *heuristic knowledge*. **Factual knowledge** consists of information

that is commonly shared, usually found in textbooks or journals, and typically agreed upon by humans knowledgeable in a specific field or task. **Heuristic knowledge**, on the other hand, is experiential knowledge of performance; it is the knowledge behind “an educated guess.” In contrast to factual knowledge, heuristic knowledge is rarely discussed, and is largely individualistic. It is the knowledge of good practice, good judgment, and plausible reasoning in the field. It is the knowledge that underlies the “art of good guessing.”

**Knowledge representation** formalizes and organizes the knowledge. One widely used representation is the *production rule*, or simply rule. A **rule** consists of an IF part and a THEN part (also called a *condition* and an *action*). The “IF” part, lists a set of conditions in some logical combination.

An example would be:

IF	
	the animal is a bird    AND
	it does not fly        AND
	it swims                AND
	it is black and white
THEN	
	it is a penguin

Following this model, an expert system will receive *propositions*, or answers to a certain line of questions. It will compare the propositions to the facts

and rules registered in its knowledge base. Using the inference engine, it will evaluate the propositions against the rules and infer an answer.

The piece of knowledge represented by the production rule is relevant to the line of reasoning being developed if the IF part of the rule is satisfied; consequently, the THEN part can be concluded, or its problem-solving action taken. A rule may *fire* another rule, which in turn may seem more information from the user. It may also complete the session by concluding either a quandary or an answer. Expert systems whose knowledge is represented in rule form are called **rule-based systems**.

The **inference engine** of an expert system usually is setup up to mimic the reasoning, or problem solving ability that the human expert would use to arrive at a conclusion. The inference engine simulates the process of deducing the solution of some problem from the information and rules in the knowledge base and the facts obtained from the answers to a series of questions.

Though an expert system consists primarily of a knowledge base and an inference engine, a couple of other features are worth mentioning: *reasoning with uncertainty*, and *explanation of the line of reasoning*.

Knowledge is almost always incomplete and uncertain. To deal with uncertain knowledge, a rule may have associated with it a *confidence factor* or a weight. The set of methods for using uncertain knowledge in combination with uncertain data in the reasoning process is called **reasoning with uncertainty** [16].

An expert system uses uncertain or heuristic knowledge and its credibility is often in question. When an answer to a problem is questionable, we tend to

want to know the rationale. If the rationale seems plausible, we tend to believe the answer. So it is with expert systems. Most expert systems have the ability to answer questions of the form: “Why is the answer X?” Tracing the line of reasoning used by the inference engine can generate these explanations [16].

### ***1.3 Knowledge Engineering***

**Knowledge Engineering** is an applied part of the science of artificial intelligence, which, in turn, is a part of computer science. A **knowledge engineer** is a computer scientist who knows how to design and implement programs that incorporate artificial intelligence techniques.

Today there are two ways to build an expert system. They can be built from scratch, or built using a piece of development software known as an expert system tool or a shell. A knowledge engineer interviews and observes a human expert or a group of experts and learns what the experts know, and how they reason with their knowledge. The engineer then translates the knowledge into a computer-usable language, and designs an inference engine, a reasoning structure, that uses the knowledge appropriately. He also determines how to integrate the use of uncertain knowledge in the reasoning process, and what kinds of explanation would be useful to the end user.

Next, the inference engine and facilities for representing knowledge and for explaining are programmed. The domain knowledge is entered into the

program one piece at a time. It may be that the inference engine is not just right or the form of knowledge representation is awkward for the kind of knowledge needed for the task or some pieces of knowledge are wrong. All these are discovered and modified as the expert system gradually gains competence.

The discovery and accumulation of techniques of machine reasoning and knowledge representation is generally the work of artificial intelligence research. The discovery and accumulation of knowledge of a task domain is the province of domain experts. Using AI techniques and the knowledge of domain experts to create a working expert system is the domain of knowledge engineers [16].

## Chapter 2

### 2. Task Domain

#### 2.1 Microbiology

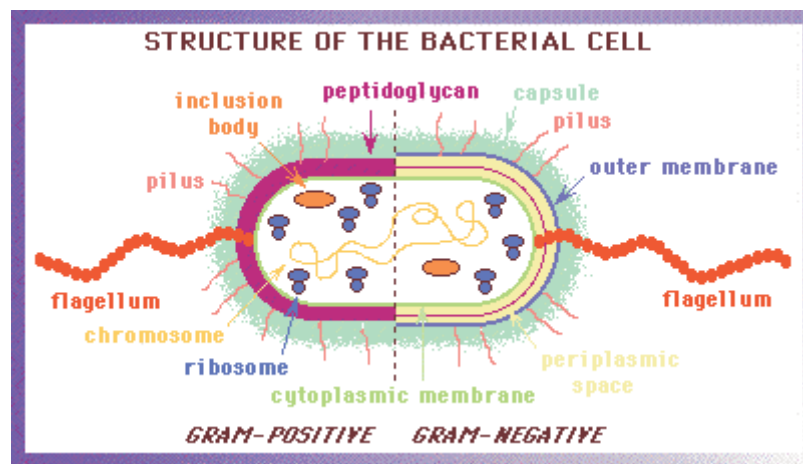
Microbiology is the study of **microbes** - *microscopic*, *unicellular*, and largely *undifferentiated* life forms. **Microscopic** objects are “invisible or indistinguishable without a microscope.” The period used on a typical typewriter or computer printer is about 0.1 mm in diameter. The average bacterium has a volume of 1.0 micrometer cubed (0.001 mm cubed). A bacterium is about 1/100 the diameter of a period, or 100 bacteria could be laid end to end across a period!

**Unicellular** denotes having or consisting of a single cell. With extremely rare exceptions (slime molds, lichen), microorganisms exist as single, undifferentiated cells. Microorganisms are **undifferentiated**, i.e. all cells in a population of microorganisms are identical to one another.

Microbes include the following: bacteria, fungi, viruses, algae, protozoa (single cell animals), and eggs and larval forms of parasitic worms.

Bacteria, along with blue-green algae, are very simple prokaryotic cells. That is, in contrast to eukaryotic cells, they have no *nucleus*; rather the genetic material is restricted to an area of the cytoplasm called the *nucleoid*. Prokaryotic cells also do not have cytoplasmic compartments such as *mitochondria* and *lysosomes* that are found in eukaryotes. The **cytoplasm** (gel-like material inside a cell that protects cell parts and helps move materials around the cell) contains

circular double stranded DNA (dsDNA), **ribosomes** (which make various proteins), **enzymes** (the catalysts of biochemical reactions which are responsible for bringing about almost all of the chemical reactions in living organisms), and a **cell membrane** (permeable membrane that has a variety of functions, including bringing chemicals and nutrients in and out of the cell) [22]. See Figure 2.1 [24].

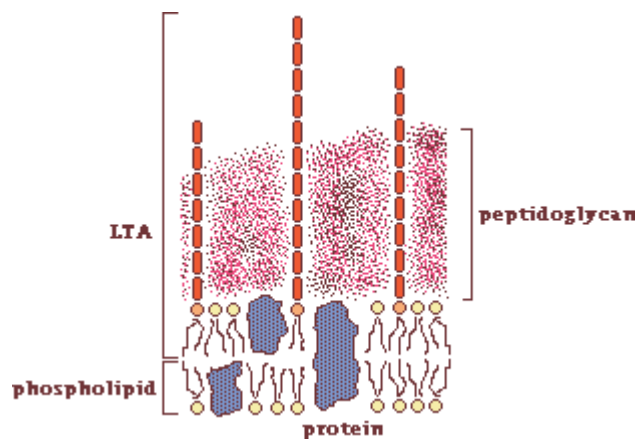


**Figure 2.1 Structure of the Bacterial Cell**

A structure that is found in prokaryotic but not in eukaryotic animal cells is the **cell wall** (starchy outer covering), which allows bacteria to resist *osmotic stress*. These cell walls differ in complexity, and divide bacteria into two major groups – gram-positive and gram-negative. This characteristic is based on the staining patterns caused by the biochemical differences of the surrounding cell wall. Gram-negative bacteria have cell walls with a high lipid component, whereas gram-positive bacteria have a higher peptidoglycan and lower lipid

content. These differences in the make-up of the cell wall are responsible for the gram staining properties. See Figures 2.2, 2.3 [24].

The possession of this cell wall, gives rise to the different antibiotic sensitivities of prokaryotic and eukaryotic cells. Prokaryotes and eukaryotes also differ in some important metabolic pathways, particularly in their energy metabolism. Many bacterial species can adopt an *anaerobic* existence, which makes bacteria better suited to survival in their environments.



**Figure 2.2 Cell wall of gram-positive bacteria**

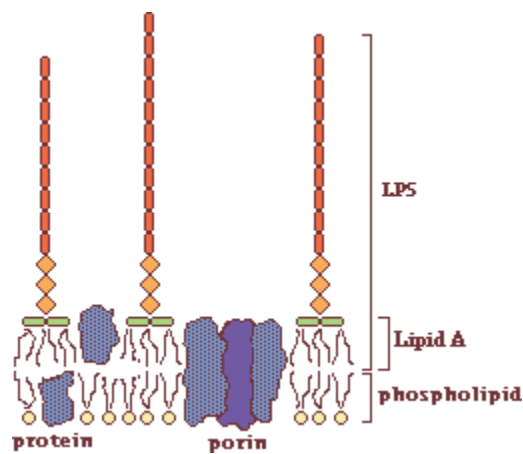


Figure 2.3 Cell wall of gram-negative bacteria

## 2.2 Taxonomy of Microorganisms

All living organisms are classified into one of five kingdoms. These kingdoms are:

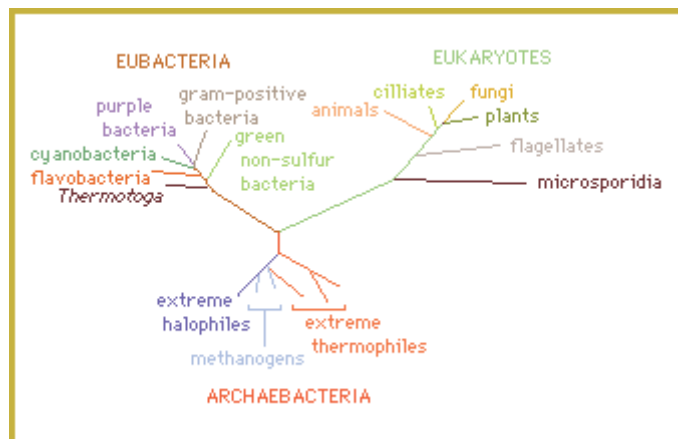
- **Animals:** Multi-cellular organisms that eat
- **Plants:** Multi-cellular organisms that employ photosynthesis
- **Fungi:** Multi-cellular organisms that absorb food, molecule by molecule
- **Protista:** All the rest of the eukaryotes
- **Monera:** Simplest living unicellular organisms – prokaryotes

Bacteria belong to the Monera kingdom. The Monera kingdom is divided into two phylums: *Cyanobacteria* and Bacteria. **Cyanobacteria**, commonly called blue-green algae, have about 1500 species that generally live in an aqueous

environment. They form the green scum that is found in warm stagnant ponds or lakes. They contain chlorophyll, are capable of photosynthesis, and produce oxygen.

Bacteria, on the other hand, do not contain chlorophyll. Some bacteria are capable of photosynthesis, but not by using the chlorophyll molecule and they do not produce oxygen.

Bacteria are only one of the members of the microorganism world. Using ribosomal-RNA (rRNA) sequence comparison, microbes can be broken into three major groups: Prokaryotes (Eubacteria), Eukaryotes, and Archaeobacteria. Figure 2.4 shows the evolutionary relationships of the major groups of microorganisms [24].



**Figure 2.4 Universal phylogenetic tree determined from rRNA sequence comparisons**

Prokaryotes and Eukaryotes differ physically from each other in many aspects. Table 2.1 lists those differences [24].

Eukaryotes	Prokaryotes
Many types of organelles	No true organelles
Nuclear membrane	No nuclear membrane
Complex phospholipids & sphingolipids	Simple phospholipids
Sterols	Absence of sterols
80S ribosomes	70S ribosomes
Absence of cell wall or made of cellulose	Peptidoglycan
Many chromosomes	Single circular chromosome
Usually diploid	Haploid
Relatively long-lived mRNA	Short-lived mRNA
Intervening sequences	Colinear
Long cell division time	Short cell division time

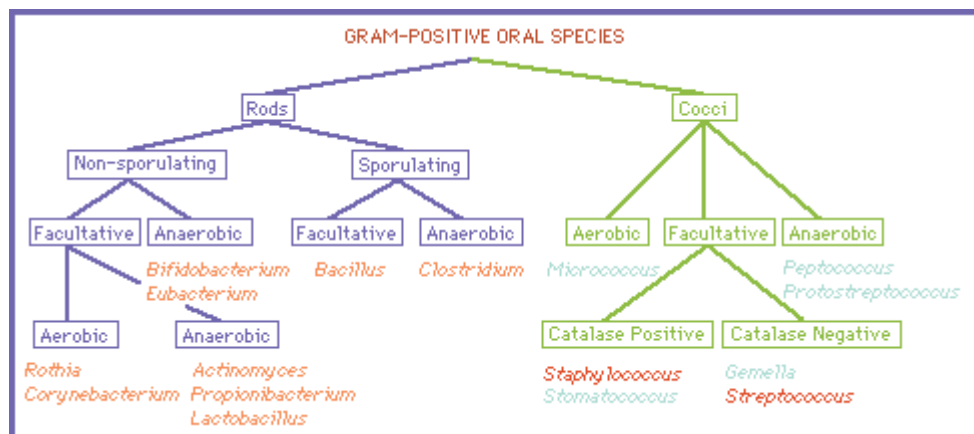
**Table 2.1 Fundamental physical differences between eukaryotes and prokaryotes**

Microbes, except for the viruses, are named and classified according to the strategy of an early botanist, Carolus Linnaeus: they are named according to a *genus-species* scheme. A genus is a closely related group of organisms, similar in appearance and other qualities. A species is a much closer related sub-category of a genus, and at least in multi-cellular organisms is considered to represent a group of organisms that can only mate and reproduce with each other. The genus-species name is usually either italicized or underlined, and the first letter of the genus is in uppercase. For example, a bacterium that causes pneumonia is named *Streptococcus pneumoniae*. *Vibrio cholera* is the bacterium that causes cholera.

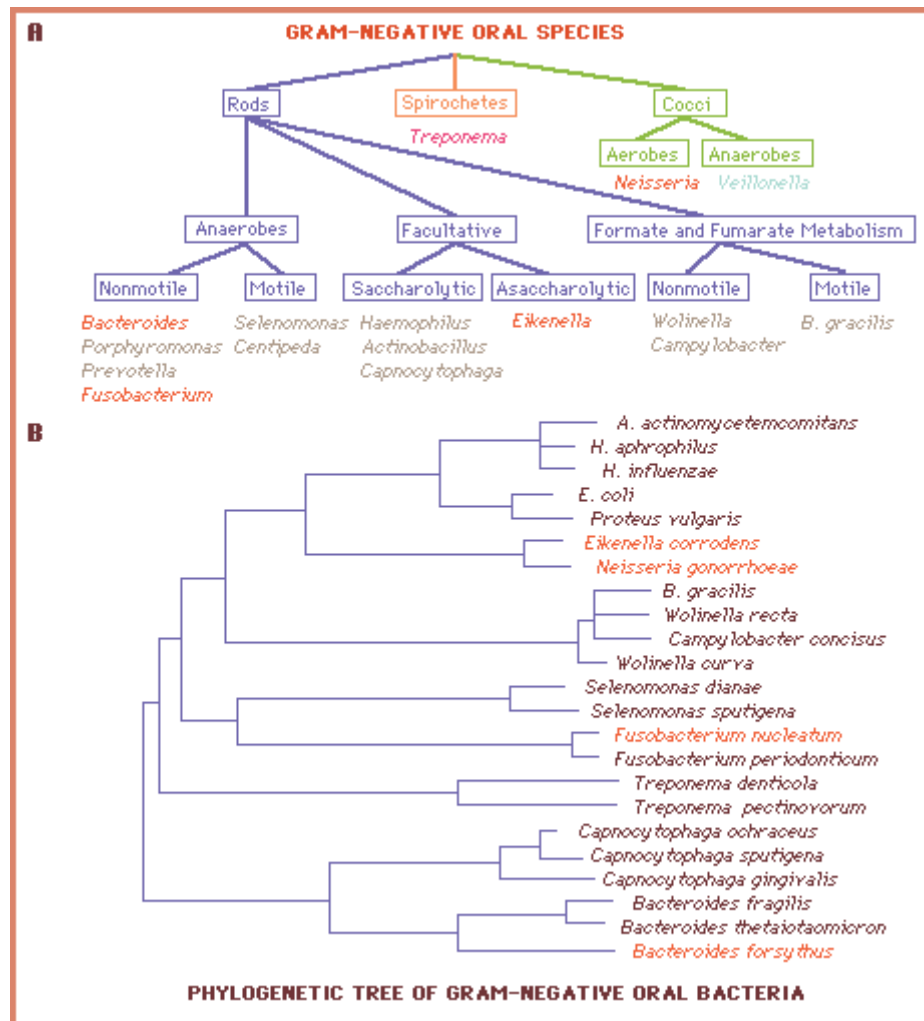
Sometimes the genus and species give clues as to the shape or arrangement of the microbe, or the disease it causes. For example, *Streptococcus pneumoniae* causes pneumonia and is a coccus, or sphere-shaped bacterium that is

arranged in chains (strep-); *Vibrio cholera* is a comma-shaped bacterium that causes cholera (Latin *vibrio*=comma) [22].

There are two main methods used in the classification, diagnosis, and epidemiology of microorganisms: Phenotypic and Genotypic. Phenotypic characteristics include: Morphological/physiological methods of single cells and colonies, and staining reactions. Important stains include: gram stain, and acid-fast stain. Gram Stain divides the Eubacteria into two major groups. Acid-fast stain identifies one particular type of Eubacteria, the acid-fast bacteria. As an example, Figures 2.5 and 2.6 give a breakdown of the gram-positive and gram-negative bacteria found in the oral cavity [24].



**Figure 2.4 Gram-Positive Oral Species**



**Figure 2.5 Gram-Negative Oral Species**

Among the physiological properties analyzed are: requirement for specific growth factors (carbon dioxide, oxygen, vitamins, cofactors), production of specific metabolic end products (alcohols, acids), and production of specific cell components (fatty acids, proteins, enzymes). Serotypic methods (antigen-antibody reactions) are another important phenotypic characteristic, and include such methods as: *immunofluorescence* and *immunoblotting* (Western blotting) [24].

Genotypic characteristics are more definitive methods of classification. They employ methods such as: genomic DNA fingerprinting, which is based on the size of *restriction endonuclease* fragments, *DNA homology*, *DNA hybridization* probes made from 16S rRNA gene sequences, and *Southern blotting* [24].

### ***2.3 Anaerobic Bacteriology***

Anaerobic bacteria are pathogens involved in virtually any type of bacterial infection in humans. They play a major role in most commonly encountered categories of infection, including skin and soft tissue *osteomyelitis*, and *pleuropulmonary*, *intraabdominal*, and female genital tract infections. They also cause rapidly progressive infection, with significant mortality [23].

Anaerobic bacteria differ from other bacteria in several ways. They thrive in areas of the body that have low levels of oxygen (such as the intestine). They are also found in decaying tissue, and in deep or dirty wounds, where other bacteria cannot live and where the body's defenses cannot easily reach. Anaerobic bacteria do not need oxygen to exist; in fact, some cannot survive in its presence. They tend to cause infections that form collections of pus (abscesses).

Most of the human *normal flora* is composed of anaerobic bacteria. This suggests that anaerobic infections might be of medical concern. Anaerobic infections can occur in a variety of body sites and involve many different types.

Most of the normal anaerobic flora is not pathogenic; rather, they are considered to be **opportunistic**. If given the opportunity, they can inflict serious and occasionally life-threatening disease. These types of infections most often occur due to trauma, injury or surgery. In general, a loss of natural barriers, that introduce these bacteria into normally sterile body sites, may result in infection.

The sites commonly involved in anaerobic infection include the following:

- intraabdominal infections
- pulmonary infections
- pelvic infections
- brain abscesses
- skin and soft tissue
- oral and dental infections
- bacteremia and endocarditis.

#### ***2.4 Current Identification Methods of Anaerobic Bacteria***

Most bacteriological identifications employ one of several manual or semi-automated methods. When a bacterium is suspected as a pathological cause of an illness, a specimen from the patient or study subject is collected. The specimen may be fluid such as blood or urine, or tissue sample such as spleen or kidney biopsy or it may be external tissue such as wound puss, or sputum. Most of the anaerobic bacteria would best be found in sterile internal organs and

cavities. Optimal specimens for anaerobic bacteria include: body fluids, tissue biopsy, blood and bone marrow.

The specimen is taken to the clinical laboratory as soon as it is collected, or it must be preserved in an anaerobic environment – such as a screw-cap vial – until taken to the lab, to allow the survival of anaerobes. In the lab, it is analyzed for suitability and then processed. It is inoculated onto anaerobic, as well as aerobic, media and incubated for growth. Anaerobic growth media includes (among others): Kenamysin-vancomycin-laked blood (KVLB) agar, and Thioglycollate (Thio) broth. The medium is placed in an anaerobic environment such as an anaerobic holding jar or bag.

The purpose of culturing the specimen on aerobic media is to determine if the bacteria is a true anaerobe, or a *facultative anaerobe*. A **facultative anaerobe** is an aerobic microorganism that can utilize fermentation when molecular oxygen is absent, and can utilize aerobic cellular respiration when O<sub>2</sub> is present. Examples of important facultative anaerobes include: *Bacillus anthracis* (which causes anthrax), *Corynebacterium diphtheriae* (which causes diphtheria), and *Escherichia coli* (which causes traveler's diarrhea, and urinary tract infections).

Relatively crude anaerobic techniques usually permit recovery of hardier anaerobes such as *Bacteroides fragilis* and *Clostridium perfringens*. Microbiologists may be lulled into thinking that the current techniques are adequate by the attitude that infection is seldom caused by the delicate anaerobes, which require specialized (expensive and labor-intensive) cultivation techniques [23].

Today's clinical laboratories are not what they used to be. A strong push to maximize the use of available space and to downsize, results in that some laboratories fail to identify or cultivate many anaerobes. There is also a misguided confidence that anaerobic infections can be dealt with because of the availability of antibiotics. This attitude may lead clinicians to avoid requesting anaerobe cultures and microbiologists to avoid using the optimum anaerobic techniques. When this approach involves a top drug with activity against virtually all anaerobes, it results in increased expense and the real risk of increasing resistance to these agents. In different parts of the world, already there is a resistance to virtually all of the most effective anti-anaerobic drugs [23].

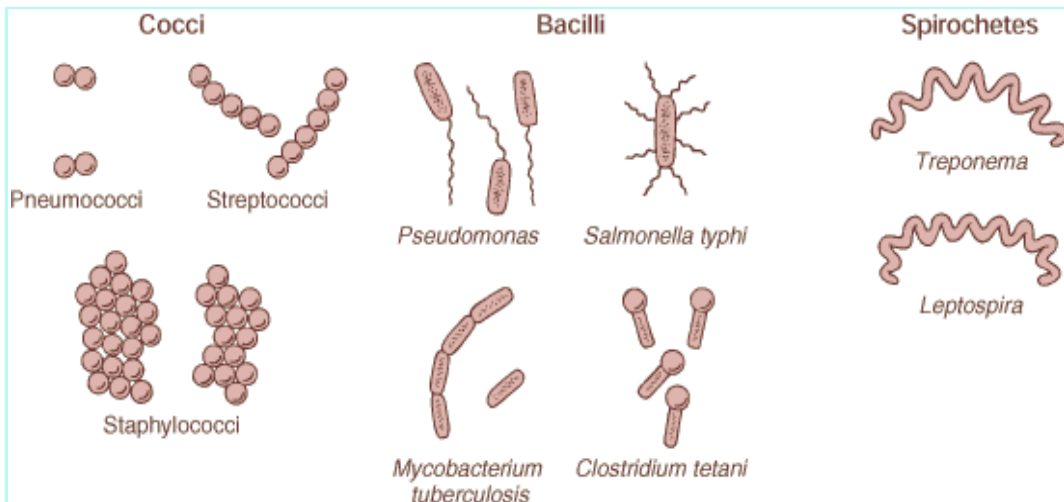
Progress in anaerobic bacteriology would not have been possible without the development of procedures for growing colonies on the surface of solid agar. Systems for accomplishing isolation and cultivation include various types of anaerobic chambers and jars. Here, evacuating the air with a vacuum pump and replacing it with an oxygen-free gas mixture from gas cylinders attains *anaerobiosis*. Other methods produce hydrogen and use it to displace oxygen [23].

When growth is observed on the media, clinical testing begins. Many factors are taken into consideration even at the very early stage of analysis to help identify or at least narrow down the identification. Among these preliminary observations are: the *growth media* that the bacteria grew on, the *temperature* it grew in, and the *colony morphology* – which includes shape, color and odor of the

colony. Another very important test that can narrow down bacteria into four major categories (and some other minor categories) is the Gram Stain.

Gram Stain sub-classifies bacteria based on two factors: color and shape. Bacteria that retain the “gram stain” are known as gram positive, while those that lose it are gram negative. There are three main shapes to which bacteria conform: cocci, bacilli and spirals (see Figure 2.7) [25]. Of these, cocci and bacilli are most significant. Bacteria that stain as gram-negative bacilli will not be mistaken for those of the gram-positive cocci category. Based on the gram stain, a microbiologist will know which path to follow to further identify the bacterium, hence the importance on the gram stain.

There are several biochemical tests that may be employed at this point of the analysis. These tests are more definitive than stains, but more expensive. They test the biochemical structure of the bacteria. Usually, a battery of tests is required to obtain an accurate resolution for the identification of the microorganism.



**Figure 2.6 Shapes of Bacteria**

## ***2.5 Limitations of Manual Methods***

Physicians use clinical lab reports as a main reference when diagnosing a patient. Pathologists rely on lab reports to do further research. An incorrect report will hinder the physician from prescribing the correct therapy and pathologist from drawing accurate analysis and conclusion. Therefore, an accurate and correct clinical laboratory report constitutes the core and fundamental element in medical treatment and pathological research.

The probability of making mistakes in identifying microorganisms in the clinical laboratory is high due to the following reasons. First, many microorganisms have similar shapes and colony morphology, and can grow in similar growth media and temperature. These factors complicate the identification of similar microorganism. Second, technologists, who conduct tests

and produce reports, draw conclusions on the basis of their knowledge and experiences. Possibilities exist for making errors due to the limited knowledge base and cognitive activities that human judgment is based on. Third, new microorganisms are continuously being identified. Microbiologists, who do not maintain knowledge of new advances in the research area, may overlook the new findings and mistake them for some species limited to their knowledge.

Medical technologists possess a vast quantity of knowledge about their domain, but even the most specialized scientists, such as medical microbiologists, cannot maintain a complete knowledge base of all the clinical data associated with all the identified bacteria. Even if the domain is narrowed down to “medically-significant anaerobic bacteria”, the “human” knowledge base will be found lacking. Added to this limitation, is the fact that most technologists will experience working with microorganisms that are considered more common or frequently isolated, but will have little or no experience with rare but significant -- and in many cases very critical -- pathogens.

## Chapter 3

### 3. AnaEx: A Solution

#### 3.1 *RapID ANA II*

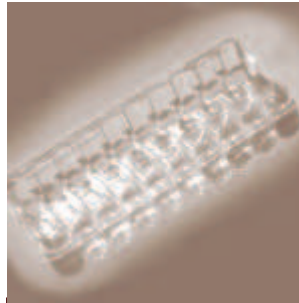
AnaEx (**An**arobic Bacteria Identification **Ex**pert System) is the name given to the Expert System that was developed to assist in identifying anaerobic bacteria. It is designed based on the RapID ANA II system developed by REMEL Inc., Lenexa, KS. The RapID ANA II System is a qualitative micro-method employing conventional and *chromogenic substrates* for the biochemical identification and differentiation of medically important organisms from human clinical specimens.

Through the exclusive use of RapID ANA II System results, an empirical database has been developed which defines organisms by their patterns of reactivity expressed as a series of positive percentages for each system test. This unique database provides the basis for a computer-assisted numerical approach to the identification of organisms addressed by the RapID ANA II System [28].

A numerical approach to the identification of microorganisms is a natural outcome of the availability of digital computers. In a numerical approach, the similarity between microorganisms is assessed mathematically and organisms are arranged into groups on the basis of shared characteristics (*phenons*). An operational taxonomic unit (OTU) or taxon, is composed of a collection of microorganisms with shared phenons and a single feature is neither essential to

group membership nor sufficient evidence to assign an organism to a particular taxon. The RapID ANA II System makes use of many different characteristics and therefore is able to accommodate well-defined groups as well as elements of group variations. [28]

Each RapID ANA II panel has ten reaction cavities molded into the periphery of a plastic disposable tray (see Figure 3.1). Reaction cavities contain dehydrated reactants. The tray allows the simultaneous inoculation of each cavity with a predetermined amount of inoculum. A suspension of test organism in RapID Inoculation Fluid is used as the test inoculum which re-hydrates and initiates test reactions. After incubation of the panel, each test cavity is examined for reactivity by noting the presence or absence of a color. In some cases, reagents must be added to the test cavities in order to provide a color change. The tests are then scored as “positive” or “negative”. The resulting pattern of positive and negative test scores is used as the basis of identification of the test isolate by comparison of test results to reactivity patterns stored in a database or through the use of a computer-generated Code Compendium [26].



**Figure 3.1 RapID ANA II Test Tray**

The RapID ANA II Code Compendium is divided into three major sections, to include gram-negative anaerobic rods, gram-positive rods, and gram-positive and gram-negative anaerobic cocci. Microcodes are listed in numerical order. Each entry consists of the microcode and its interpretation. It then lists: the organism choice, corresponding *probability*, *biotype*, and *contraindicated test results*, and *comments*. Numerical data is presented as a *probability percent* and the *biotype score*. Interpretations of the numerical values for the first choice are listed as: *probability level* and *biotype frequency*. An example of the RapID ANA II System Code Compendium data is shown in table 3.1.

Microcode		677306 Identification				B. uniformis (Presumptive)
		Probability		Biotype		
Name	Percentage	Level	Score	Frequency	Contraindicated Tests	
B. uniformis	96.3	Presumptive	1/3197	Rare	PYR	
B. ovatus	3.4				GLY, PYR	
B. thetaiotamicron	0.3				ARG, SER, PYR	

**Table 3.1 Example of the RapID ANA II System Code Compendium Data**

A brief explanation, taken from the compendium [28] of all these data elements follows. The complete explanation can be found in the compendium.

**Identification:** When the probability of the first choice is greater than 95.0% and the biotype score reflects normal variation from the ideal database pattern, identification to the species level is obtained.

**Choice:** The organisms in the database most applicable to the microcode are printed as identification choices in order of descending probability. A maximum of three choices is printed.

**Probability:** The Probability column displays the relative probabilities of each choice for the microcode entry. Probabilities are derived by calculation of a likelihood score for each taxon in the database. The probability level is an interpretation of the probability percent of the 1st choice.

**Biotype:** The biotype score is an expression of the “typicality” or “fit” of the test results compared to the ideal pattern for the taxon in the database. The *biofrequency* is printed as the number of strains of the 1st choice taxon that would have to be observed before the test pattern is encountered.

**Contraindicated Test Results:** If a test result is positive and the database percentage is  $\leq 25\%$  or if a test result is negative and the database percentage is  $\geq 75\%$  then the test is considered a contraindication against the choice.

### 3.2 *AnaEx*

RapID ANA II uses a set of 18 biochemical tests to identify anaerobes. AnaEx extends the logic used by RapID ANA II. A few important factors, which the system lacks, are included in AnaEx.

Foremost among the extension is the inclusion of the gram stain results in the system. As mentioned earlier, gram stain is one of the most pivotal tests in bacteriology. It is a quick, cost-efficient and reliable test that rapidly classifies the microorganism into one of several main categories. This test is so vital it allows physicians to begin treating a patient based on the gram stain results alone, even without waiting for a confirmatory identification of the pathogen. This is due to the fact that gram-stain-similar microorganisms behave in certain similar patterns. Certain antibiotics are *broad-spectrum* antibiotics and may be used to treat an entire group of microorganisms. Examples of broad-spectrum antibiotics are the *Aminopenicillins*, which target gram-negative bacteria.

Before the system even begins to analyze and identify a microorganism, it checks to see if it is an anaerobe. True anaerobes grow only on anaerobic media in anaerobic environments. Facultative anaerobes are aerobes that can tolerate small amounts of CO<sub>2</sub> and can grow in anaerobic environments. AnaEx tests for this growth behavior. It tests for growth on CDC ANA and CDC ANA KV agar plates. CDC Anaerobe 5% Sheep Blood Agar (CDC ANA) is an *enriched nonselective* culture medium particularly useful for the isolation and cultivation of

obligate anaerobes from clinical specimens. It supports the growth of a wide variety of *obligately anaerobic*, *facultatively anaerobic*, *microaerophilic*, and *aerobic* bacteria. CDC Anaerobe 5% Sheep Blood Agar with Kenamycin and Vacomycin (CDC ANA KV) is an *enriched selective* culture medium for the isolation of obligately anaerobic gram-negative bacilli from clinical specimens.

Growth on both anaerobic media strongly suggests the possibility of an anaerobic bacterium. No growth on either plate is a clear indication of no anaerobic presence. Nevertheless, this alone does not conclude that the microorganism isolated is an anaerobe. Facultative anaerobes must be ruled out. The next set of tests verifies growth on aerobic media in aerobic environments. They test for growth on SBA and CHOC agar plates. Sheep Blood Agar (SBA) is an *enriched nonselective* culture medium. Chocolate Agar (CHOC) is an *enrichment* culture medium, which enhances growth of weaker microorganisms that may not grow well on other media. Growth on either one of these indicates aerobic characteristics. Since true anaerobes cannot grow in aerobic environments, these microorganisms would be ruled out as aerobes. Of course, there may be more than one microorganism in the same specimen. Care must be taken to assure that the same microorganism is being tested on both types of media. Each type of colony must be analyzed separately.

After anaerobic characteristics are verified using the above set of tests, the gram stain is done. The system then asks for the specimen source. This optional variable can be very useful for diagnosis and treatment.

Next, comes a set of tests that are critical for the expert system. This is a set of 18-biochemical tests grouped in sets of three (see Table 3.1). Each group is scored and computes to a numerical value. The set of six values together compose a microcode. The microcode is the identifier index of the microorganism. The value assigned to each test depends on the position of test in the group. A negative test result is scored as a zero while a positive test results receives a score of one, two or four if it is in the first, second or third position of the group, respectively.

<b>Test</b>	<b>Value</b>	<b>Group</b>
URE	1	<i>ONE</i>
BLTS	2	
$\alpha$ ARA	4	
ONPG	1	<i>TWO</i>
$\alpha$ GLU	2	
$\beta$ GLU	4	
$\alpha$ GAL	1	<i>THREE</i>
$\alpha$ FUC	2	
NAG	4	
PO <sub>4</sub>	1	<i>FOUR</i>
LGY	2	
GLY	4	
PRO	1	<i>FIVE</i>
PAL	2	
ARG	4	
SER	1	<i>SIX</i>
PYR	2	
IND	4	

**Table 3.1 Rapid ANA II Biochemical Tests Setup**

### ***3.3 Benefits of Using Expert Systems***

Considering the previously discussed difficulties of identifying microorganisms using conventional methods, some sort of automated problem solver would seem logical to put in place. The preeminent solution in such a case would be an expert system. What benefits does an expert system have to offer in this case?

An expert system can contain, in its knowledge base, a much superior database than a human expert can. Even the most experienced technologist in a very specialized area, e.g. an anaerobic microbiologist, will not have complete knowledge of all the aspects of the domain. An expert system, however, can contain all the data and knowledge that is programmable and available to it. It will not “forget” nor “misunderstand” the knowledge.

Adding knowledge to an expert system is painless. Many bacteria are being identified constantly, several of which are medically important. Furthermore, as more is learned about the different microorganism, their classification may change, and thus their genus-species “name”. *Bordetella pertussis*, which causes whooping cough, was known as *Haemophilus pertussis* some years ago – a taxonomical change even at the genus level. A proficient scientist must keep up with all these changes. This requires learning and re-training, not always an easy task for humans. As a technologist becomes more experienced, and hence older, his or her learning ability tends to dwindle. Teaching an expert system new knowledge is not an effortless task either,

however, it is less problematic and much quicker than retraining a human expert. This would require adding some more data in the knowledge base and examining the established rules and possibly modifying them. A good expert system would even allow the end user to add more knowledge without having to reprogram or modify any rules. This is the approach taken when designing AnaEx. A user who is proficient in databases can easily add new knowledge to the system without the assistance of a programmer.

An expert system would provide reproducibility. If the same analysis were executed numerous times following the same path, the same results would be reproduced every time. A good expert system would be able to provide the reasoning for the solution obtained. It would do this by keeping a trace of the path used to obtain the solution. AnaEx accomplishes this by keeping all the relevant data in its database.

### ***3.4 Limitations of Using Expert Systems***

Expert Systems are always available. Unlike human experts, they do not take vacations or sick days, and do not require time to sleep. Theoretically, they are available 24 hours a day, 7 days a week. This theoretical possibility is not always accomplished because the system still requires some maintenance time, and may even require being off-line for some time. It may not become “sick” but

it may experience some problems and be “unproductive” for a certain period of time.

The existence of an AnaEx would not render the need for a “human” expert futile. The expert system acts more as an advisor rather than an absolute authority. Despite the fact that its knowledge base is larger than that of a human expert, it lacks the rationality of a human, which would allow a person to dismiss a correctly computed result that is inconsistent with other external factors, such as diagnosis and symptoms. Bacteria usually follow a known behavior, which was determined based on research studies. However, these studies do not necessarily reveal all possible patterns of behavior. Like humans, microorganisms do not necessarily have to conform to the identified patterns of behavior, and in some instances they do not. For example, all members of the *Fusobacterium* genus are lipase-positive, except for the species *F. nucleatum*. Bacteria know what they are doing. We are trying to figure them out!

## Chapter 4

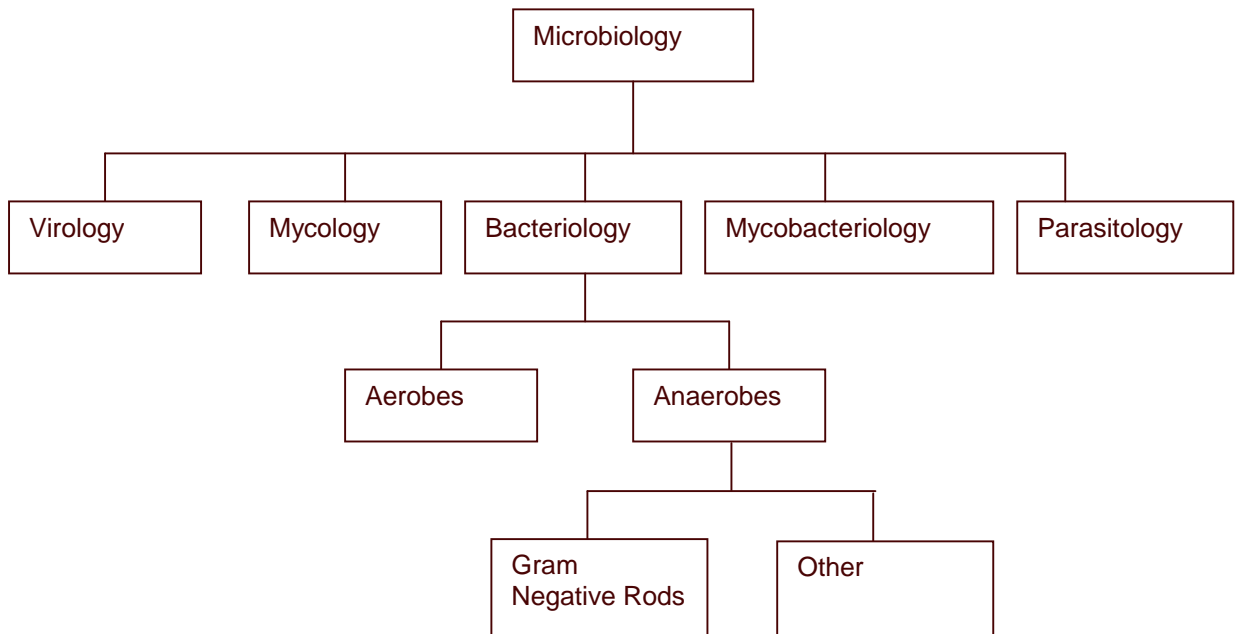
### 4. System Design

#### 4.1 *Targeting a Decision*

Microbiology is a broad discipline. As discussed earlier, it is composed of many different sub-fields. In order to build an effective expert system, the task domain needs to be narrowed down significantly. The intent is to build a stand-alone system that will assist technologists who perform microbiological analysis to identify anaerobic bacteria.

The scope of the system is to identify the organism, classifying it as an anaerobe. The decision process begins with the realization that the microorganism is bacteria, and not a virus or a parasite, for example. This assumes that the technologists already possess the knowledge to make this decision. Of course, there are very distinct methods that can easily distinguish bacteria from other types of organisms. Viruses for example grow very slowly, at an average rate of 6-8 weeks, before any growth is visible, and will only grow on special viral media, whereas bacteria grow within 48-72 hours. Once the organism is identified as bacteria, it is then subjected to certain tests to distinguish anaerobes from aerobes. If the organism is found to be an aerobe, the case is completed and the session ended. If it is truly an anaerobe, the session continues and the ultimate decision is to classify what kind of an anaerobe it is.

The block diagram in Figure 4.1 shows the scope focus of the project. The focus is on the bacteriology branch of microbiology. The project will be centered only on the anaerobic gram-negative rods (GNRs).

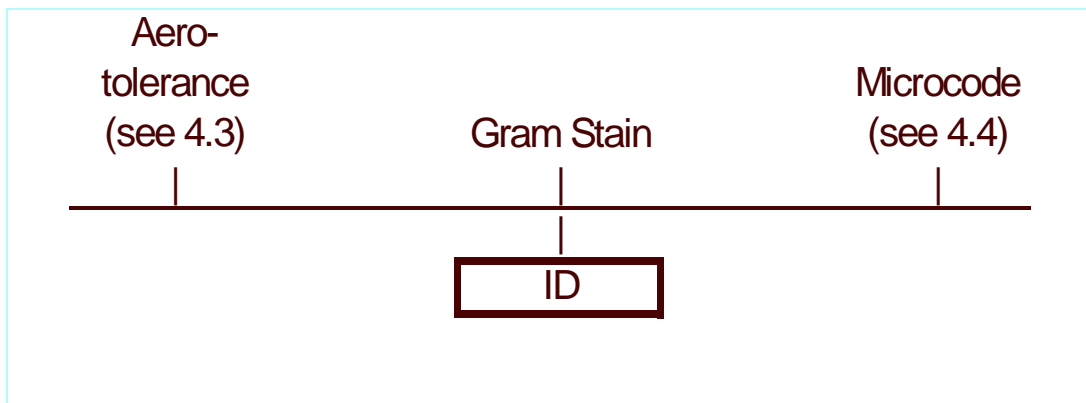


**Figure 4.1 Block Diagram of the area under study**

## **4.2 *Dependency Diagrams***

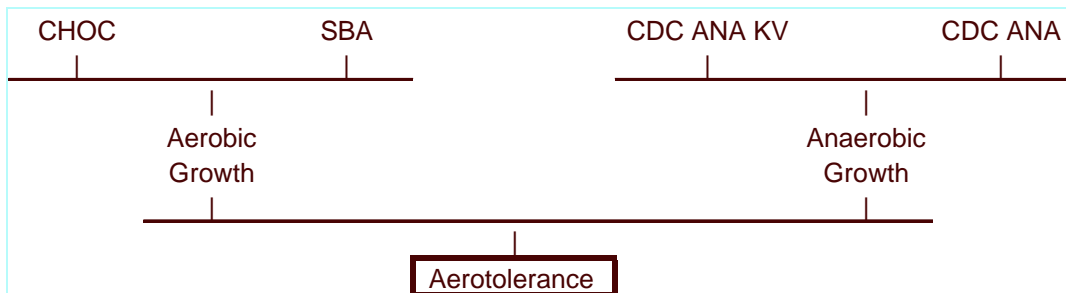
Using the block diagram, the focus of the project becomes clear, and the next step can be taken – creating a dependency diagram. A **dependency diagram** indicates the relationships (dependencies) among critical factors, input questions, rules, values and recommendations made by the KBS. It is a complete graphic statement of the KBS. It serves as the paper model for how to write or code the actual knowledge base.

To create the dependency diagram, it is important to realize what factors are taken into consideration to reach a decision. In the scope of this project, there are three major factors to be considered in identifying an anaerobe: aerotolerance, gram stain, and the microcode. A Block diagram for these factors is created to visualize the issue (see Figure 4.2).



**Figure 4.2 Block Diagram of the Decision**

The aerotolerance test is the first and determinative factor that would either continue or conclude a case. It assesses a microorganism's aerotolerance to anaerobic conditions. The result, anaerobic or aerobic, is based on the microorganism's ability to grow in both aerobic and anaerobic environments on aerobic and anaerobic growth media respectively (see Figure 4.3). Bacteria that are found to be aerobic will not be tested further, as it will not be identifiable.



**Figure 4.3 Aerotolerance Test of the Decision Block Diagram (expanded)**

The gram stain is straightforward: a gram stain is performed on the organism and the result is concluded. For the scope of this project, only gram-negative rods will be identified, so the result can be either “GNR”, or “other”. If the result is “other” the case is completed, and the microorganism is identified as “other genus”.

The final and conclusive factor in the equation is the microcode test. This is the test that will produce a code corresponding to a specific microorganism, thereby identifying it. This code is based on biochemical tests performed on the sample. The microcode itself is a composition of 6 codes, each of which is based on the results of three biochemical tests (see Figure 4.4). The microcode is calculated as follows:

$$\text{microcode} = (\text{code1} + \text{code2} + \text{code3} + \text{code4} + \text{code5} + \text{code6})$$

Each code in turn is calculated as follows:

```

code1 = ( (URE_VAL + BLTS_VAL + aARA_VAL) * 100000)

code2 = ( (ONPG_VAL + aGLU_VAL + BGLU_VAL) * 10000)

code3 = ( (aGAL_VAL + aFUC_VAL + NAG_VAL) * 1000)

code4 = ( (PO4_VAL + LGY_VAL + GLY_VAL) * 100)

code5 = ( (PRO_VAL + PAL_VAL + ARG_VAL) * 10)

code6 = ( (SER_VAL + PYR_VAL + IND_VAL) * 1)

```

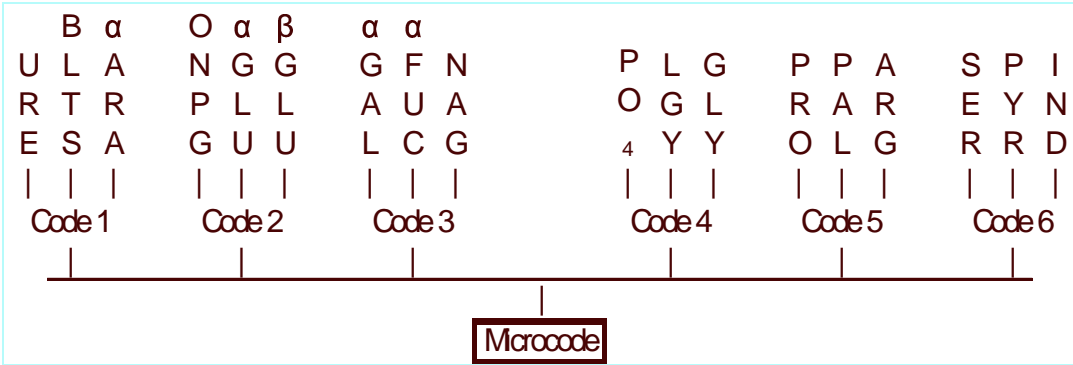


Figure 4.4 Microcode Test of the Decision Block Diagram (expanded)

Each code is based on the result of three biochemical tests in the group. A group can have a minimum score of zero and a maximum of seven. Each test is scored as zero if negative. If positive, it is scored as a one, two or four, depending on its position in the group. The test involved in code1 would be scored as follows:

```

IF   URE = POS

THEN  URE_VAL = 1

ELSE  URE_VAL = 0;

```

```
IF    BLTS = POS
THEN  BLTS_VAL = 2
ELSE  BLTS_VAL = 0;

IF    aARA = POS
THEN  aARA_VAL = 4
ELSE  aARA_VAL = 0;
```

The microcode result is a number. This number is indicative of one of two conditions: *found* or *not\_found*, indicating whether the microcode is found in the database/compendium or not. The final result, ID, is either a genus-species name of the bacteria (if the microcode was found) or one of three possible alternate results: *aerobe*, *other\_genus*, and *unknown*. “Aerobe” indicates that the microorganism is an aerobe, and thus cannot be identified by the system. If it is a true anaerobe, and the microcode was not found this may be due to an unidentified gram stain result. Since the system only identifies GRNs, it would produce the “other genus” result. The “unknown” result may be due to the fact that although this is a GNR anaerobic microorganism, the microcode obtained is not found. In many cases this will be due to an error in the test performance or reading, which would prompt the technologist to reevaluate the test. Alternatively, it may truly be a microorganism that the RapID ANA II system cannot identify.

To begin creating the dependency, it helps to turn the complete (expanded) block diagram (Figure 4.2) horizontally. This makes apparent the relationship between the block diagram and the result desired at the end of this analysis. The process begins by drawing boxes with triangles appended for all critical factors identified in the block diagram. In this case, these factors are aerotolerance, gram stain, and microcode. Because “gram stain” feeds directly into the final identification, its intermediate box and triangle can be deleted from the dependency diagram. On straight lines coming into the triangles, a word or phrase that best describes the item that will influence the outcome of the critical factor is written. This item will be called a “variable.” All possible values that the variable can have are written under the line. Also, names for the values that the critical factor itself can have are written under each box that represents a critical factor.

According to Dologite [29], this seemingly mundane modeling forces an evaluation of each piece of the KBS puzzle. The evaluation causes the modeling process to be repeated over several iterations (with journeys back to the block diagram) to model a correct solution. This is normal and actually desirable. This approach certainly seemed to be very valid in the development of this project. It was well worth the time and effort that was invested in the planning and designing of the project even before a single line of code was written. In the long run it saved a lot of time, effort and frustration. The act of putting lines and words on paper and seeing the result caused the critical and creative processes to flow.

After several passes, a credible dependency diagram evolved. The final abridged dependency diagram for the AnaEx system is shown in Figure 4.5.

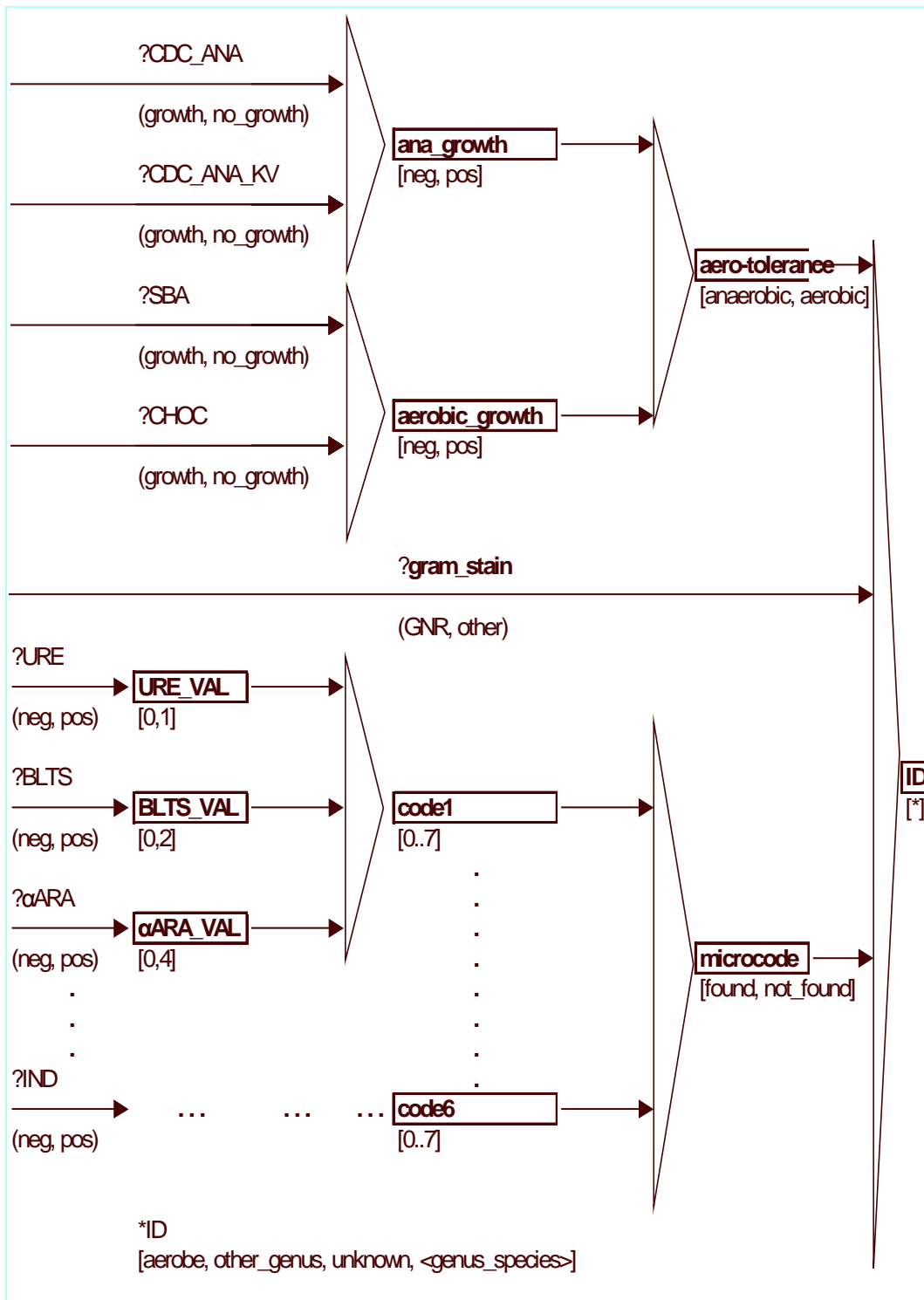


Figure 4.5 Dependency Diagram

### 4.3 Decision Tables

Creating a decision table, such as the one shown in Figure 4.7, for each triangle on the dependency diagram is the final major modeling step. The **decision table** is necessary to show the interrelationships of values to the outcome of any intermediate phase or final recommendation of the KBS.

Preparing a decision table is a straightforward process. The process begins by planning the number of rows necessary for the table. This can be determined by listing all the factors, or *conditions*, that come into the triangle under consideration. In Figure 4.6, the decision table is for the final rule set, which concerns three conditions, each of which can take a number of different values. Aerotolerance, the first of these conditions, can take only two values; it is either anaerobic or aerobic. Such information comes from the dependency diagram and is easily mapped onto the decision table plan.

Conditions	Values	Number of Values
<i>aero-tolerance</i>	[ <i>anaerobic</i> , <i>aerobic</i> ]	2
<i>gram_stain</i>	[ <i>GNR</i> , <i>other</i> ]	2
<i>microcode</i>	[ <i>found</i> , <i>not_found</i> ]	2
<b>2 X 2 X 2 = 8</b>		↑

Figure 4.6 Decision Table Plan for Final Rule Set

After all the conditions and values are listed, the number of rows for the decision table can be determined. In this case there will be 8 rows: two

possibilities for “aerotolerance,” *times* two possibilities for “gram\_stain,” *times* two possibilities for “microcode.” This calculated number represents all the possible combinations of conditions that can occur at this decision point of the KBS.

The eight rows are arranged as shown in the completed decision table in Figure 4.7. Creating this table begins by drawing an empty table shell consisting only of row numbers, and column labels. Column labels are condition names, and a heading is added for an extra column to identify the possible outcome of each combination of values. This modeling approach allows all combinations of the values of different conditions to be evaluated [29].

To simplify filling the values into each cell of the table, it is helpful to draw horizontal dividing lines between rows. The place to draw a dividing line is determined by the number of values from the earlier planning stage. For example, the first condition, aerotolerance, has only two possible values. So the first dividing line splits the rows in half (a *2 cut*) between rows 4 and 5. Each half is now a separate “whole” to be taken into consideration when dividing the rows for the second condition.

The second condition, gram\_stain, also has two possible values, so another 2 cut division is appropriate. This 2 cut must now be done twice, once over the top half of rows and once again over the bottom half of rows. The final condition, microcode, again requires a 2 cut and calls for treating each of four subdivisions as a whole. So the last horizontal lines split each pair of rows.

Values are next placed into this divided shell of empty cells. As evident from the aerotolerance column, the value “anaerobic” is repeated over half the cells and “aerobic” is repeated over the other half. Moving right one column, the two values for gram\_stain are repeated over both halves of the originally divided table. Moving right one more column, it remains to iterate the two values associated with the last condition over each pair of rows.

<b>Rule</b>	<b>aero-tolerance</b>	<b>gram_stain</b>	<b>microcode</b>	<b>ID</b>
1	anaerobic	GNR	found	<genus_species>
2	anaerobic	GNR	not_found	unknown
3	anaerobic	other	found	other_genus
4	anaerobic	other	not_found	other_genus
5	aerobic	GNR	found	aerobe
6	aerobic	GNR	not_found	aerobe
7	aerobic	other	found	aerobe
8	aerobic	other	not_found	aerobe

**Figure 4.7 Completed Decision Table**

With all the cells filled in, it now remains to evaluate each combination of values and write the outcome value in the last column. This requires scrutinizing each row, or example case, evaluating all the values listed, and having an expert determine what would be concluded from the evidence given. An evaluation of the first three rows could be read as follows:

1. If aerotolerance is anaerobic, and the gram stain is GNR, and the microcode is found (in the database or compendium) a genus-species identification of the microorganism is concluded. For

example, if the microcode is 45 (which is “found” in the database), the ID will be *Fusobacterium necrophorum*.

2. If aerotolerance is anaerobic, and the gram stain is GNR, and the microcode is not found (in the database or compendium) an identification of the microorganism cannot be concluded, and so the result is “unknown”.
3. If aerotolerance is anaerobic, and the gram stain is other (than GNR), and the microcode is found (in the database or compendium) a genus-species identification of the microorganism cannot be concluded.

To continue the evaluation of the remaining rows, it quickly becomes obvious that some conditions are meaningless in certain contexts, such as rule 3 stated above. This leads to the reducing of the completed decision table.

In row 3, the case evaluated is a microorganism that is not GNR. There is no reason to evaluate the microcode and consider whether it is found or not, since the system only identifies GRNs. In this case, an **indifference symbol**, the hyphen, is used to indicate that the condition does not have any bearing on the evaluation. Its effect is to collapse rules 3 and 4 into a single rule.

Likewise, once it is determined that the aerotolerance is aerobic, nothing else is relevant. The system only identifies anaerobic bacteria, and all aerobes are simply given the ID of “aerobe.” The result is that rules 5 to 8 can be collapsed into a single rule. The final reduced decision table is found in Figure 4.8. At the

end of the process, each line in the reduced decision table will become one rule in the knowledge base.

<b>Rule</b>	<b>aero-tolerance</b>	<b>gram_stain</b>	<b>microcode</b>	<b>ID</b>
1	anaerobic	GNR	found	<genus_species>
2	anaerobic	GNR	not_found	unknown
3	anaerobic	other	--	other_genus
4	aerobic	--	--	aerobe

**Figure 4.8 Final Reduced Decision Table**

The decision table format used here is a variation of traditional decision table structure and is adopted to facilitate rule-based modeling. In a traditional decision table, rules would be defined by the columns [29].

#### **4.4 Knowledge-Base Rules**

Writing rules for the knowledge base is an easy exercise at this point in the development process. Usually, every rule in the reduced decision table is converted into an IF-THEN rule.

Basically, a rule begins with the keyword IF followed by the conditions evaluated. A series of conditions can be linked by the logical operators “AND” and “OR”. “AND” means that conditions on both sides of the AND must be true in order for the rule to pass or “fire.” “OR” means that one or both conditions must be true.

If all the conditions in a rule are true, the “THEN” clause, or *conclusion*, of the rule is fired. It causes the variable named on the left side of the equal sign in the THEN clause to be assigned the value of the variable on the right side of the equal sign.

For example, consider the following simple rule:

RULE 2

IF ana\_growth = pos AND

aerobic\_growth = pos

THEN id = aerobe

BECAUSE "Growth on both aerobic AND anaerobic media";

This rule says, in effect, “IF, during a consultation, the value of the variable ‘ana\_growth’ is found to be ‘pos,’ AND the value of the variable ‘aerobic\_growth’ is found to ‘pos,’ THEN the variable ‘id’ should be assigned the value ‘aerobe.’ ” In this example, an optional keyword, BECAUSE, was used, to offer an explanation of the meaning of the conclusion.

Following the THEN conclusion, the “ELSE” keyword and alternate conclusions can optionally be added to a rule. When present in a rule, it says, in effect, “If the premise stated in the rule is known *not* to be true, do this.” For example, consider this rule:

```
RULE URE_VAL
```

```
IF URE = POS
```

```
THEN URE_VAL = 1
```

```
ELSE URE_VAL = 0;
```

This rule says, in effect, “IF, during a consultation, the value of the variable ‘URE’ is found to be ‘pos,’ THEN the variable ‘URE\_VAL’ should be assigned the value ‘1,’ ELSE, it should be assigned the value ‘0.’ ”

The complete KBS rules used in the VP-Expert system, for AnaEx, can be found in Appendix A.

## Chapter 5

### 5. Using VP-Expert

#### 5.1 *VP-Expert*

VP-Expert is an expert system shell developed by WordTech Systems, Inc. An expert system shell is basically an inference engine. It can be used to reason about many different knowledge bases. Just as a database manager has built-in templates and procedures for managing large amounts of data, expert system shells already have built into them the necessary search and managing routines for reasoning with the facts and rules that are written into the knowledge base.

The inference engine is the heart of any expert system shell. It is the mechanism used to solve problems, i.e. to find goals. The inference engine accomplishes this by managing and manipulating the RULE base. In order to reach its goal, VP-Expert's inference engine systematically searches for new values to assign to appropriate variables that are present in the knowledge base. Thus it has the capability to add to the known store of knowledge [30].

The *goal variables* for which the inference engine is searching are identified using FIND clauses. FIND clauses are found in an ACTIONS block. After a FIND clause is encountered, the inference engine begins to search for a value to be assigned to its associated goal variable. If the value assigned to that goal is not known, the inference engine searches the RULE base for a value to

assign to that goal. Initially, not all variable values, which might aid in the search, are known. Thus the search must systematically accumulate new knowledge by considering RULEs from the RULE base, which might yield helpful facts in the process of finding the goal. VP-Expert does this via a search method commonly called *backward chaining*.

The **backward chaining** method, as described by Dologite *et al* [29], and Friederich *et al* [30] works as follows. First a goal variable is identified in a FIND clause. If the value assigned to that goal is not known, the inference engine initiates a sequential search through the RULE base for the first RULE that might assign a value to that goal. In other words, starting with the first RULE listed in the RULE base, the inference engine looks at the conclusion of each RULE it comes to, to see if the goal variable for which it is searching is assigned a value there. When the inference engine finds such a RULE, it temporarily ceases moving forward through the RULE base. Instead it now looks at the premise of the same RULE. Here one of three cases is possible. If the premise evaluates to true, then the RULE fires and the search is ended. If the premise evaluates to false, the inference engine leaves that RULE and moves on to examine the conclusion of the next RULE. If, however, there is a relation in the premise that cannot be evaluated because a variable value is unknown, the inference engine initiates yet another search – now with this variable as the sub-goal. Thus the inference engine systematically searches out new facts to add to the store of knowledge.

This searching procedure is carried out in a recursive manner consistent with its search for the initial goal. Internally, VP-Expert keeps track of these searches in a *stack*. The search continues until some variable being searched for is found and assigned a value. After such a value is found, the inference engine, noting that a sub-goal has been reached, can return to and sequentially continue a previously initiated search. If the value was assigned to the goal variable, then the search is complete.

If during this process, a value other than UNKNOWN cannot be assigned to a variable being searched for, the inference engine searches the knowledge base for any ASK and CHOICES statements associated with that variable. The ASK statement can then prompt the user for any currently UNKNOWN information that is required for the search to continue. This occurs whenever such information does not exist, either directly or indirectly in the currently existing knowledge base.

The process of backwards chaining continues to find values for each target variable in turn, in a “*First In – Last Out*” (*FILO*) stack pattern. As each variable is *pop’d* off the stack, the system gets closer to finding the value for the ultimate target variable. As each variable’s value is known, the system continues to process the rules from the stopping point when the variable was *pushed* unto the stack.

## 5.2 *Prototype Design*

AnaEx was first developed using VP-Expert. In the implementation of AnaEx, there are three goal variables: microcode, ID, and message. The microcode goal variable tries to get a value for the microcode based on the six sub-codes, which are derived from 18 biochemical test results. If a microcode is calculated and found, an ID may be inferred based on the microcode. Finally, a message is sought for to report the identity of the microorganism along with any other helpful information.

A subset of the compendium data was inserted in a dbase file: *compend.dbf*. This file is used by AnaEx to look up the calculated microcode and try to make a match. The data elements of the dbase file are listed in table 5.1.

<b>Name</b>	<b>Description</b>
MICRO_CODE	Microcode
PRB_OVRLP	Probability Overlap
ID1	First ID
PRB_PRCNT1	Probability Percentage for ID1
PRB_LVL1	Probability Level for ID1
BIO_FREQ1	Bio Frequency for ID1
CNTRA_TST1	Contraindication Test for ID1
ID2	Second ID
PRB_PRCNT2	Probability Percentage for ID2
PRB_LVL2	Probability Level for ID2
BIO_FREQ2	Bio Frequency for ID2
CNTRA_TST2	Contraindication Test for ID2
ID3	Third ID
PRB_PRCNT3	Probability Percentage for ID3
PRB_LVL3	Probability Level for ID3
BIO_FREQ3	Bio Frequency for ID3
CNTRA_TST3	Contraindication Test for ID3
COMMENT	Comments

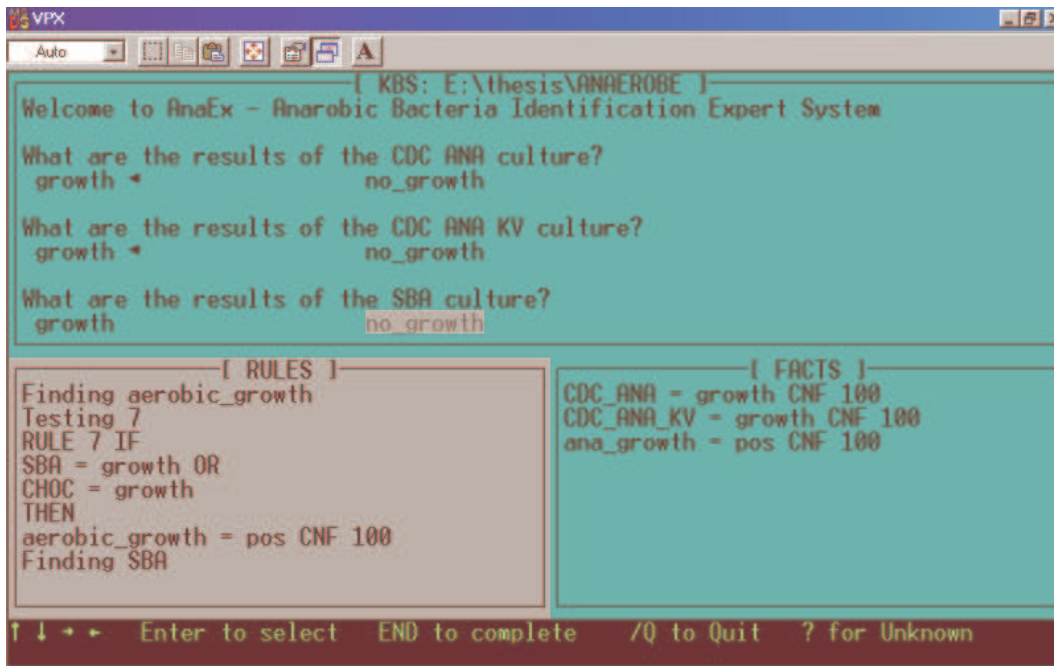
**Table 5.1 Data Dictionary for *compend.dbf***

Built into this data file is the possibility that a single microcode may map to a maximum of 3 different microorganisms, with the first one being most likely correct. This is the probability overlap factor, which was explained in Section 3.1.

### ***5.3 Advantages of VP-Expert***

VP-Expert is a good prototyping tool because it allowed for the rapid development of the expert system. The rules syntax is intuitive to design and develop.

VP-Expert provides good debugging tools. During development, the consultation screen is usually a three-window screen as shown in Figure 5.1. The screen consists of a reduced consultation window, a “rules” window, and a “facts” window.



**Figure 5.1 Consultation Screen**

The “rules” window is a *viewport* into the inference engine’s search strategy. It is a live scrolling record of how the inference engine performs its search strategy and what the inference engine is currently doing. Its path of reasoning is made transparent to the developer to observe while a consultation is actually in progress.

The “facts” window is a viewport into the system’s working memory (cache). It defines the values, both intermediate and final, derived during the course of the consultation. They are expressed in equations like: “ana\_growth = pos CNF 100.”

A confidence factor (CNF) is given to indicate the degree of certainty, or *confidence*, that a particular conclusion is valid. A **confidence factor** is a number

attached to a value that indicates the developer, expert, or user's degree of certainty in the value. In VP-Expert, zero indicates no confidence and 100 indicates total confidence or trust in the value. A confidence factor is not a statistical factor, although it often is referred to as a percentage. It may be based on subjective intuition or objective criteria to represent how sure one is that a value is correct [29].

There are some other debugging tools that are available in VP-Expert to assist the development of expert systems, the "How?" option being one of them. It describes *how* a value was assigned to a variable. An example can be seen in Figure 5.2. Users and developers often consider it essential to be able to examine and critique how a decision is determined. Another very useful option is the "Why?" option, which allows for the halting of the consultation at any point and asking *why* a question is being asked. Figure 5.3 illustrates an example of using the "Why?" feature.

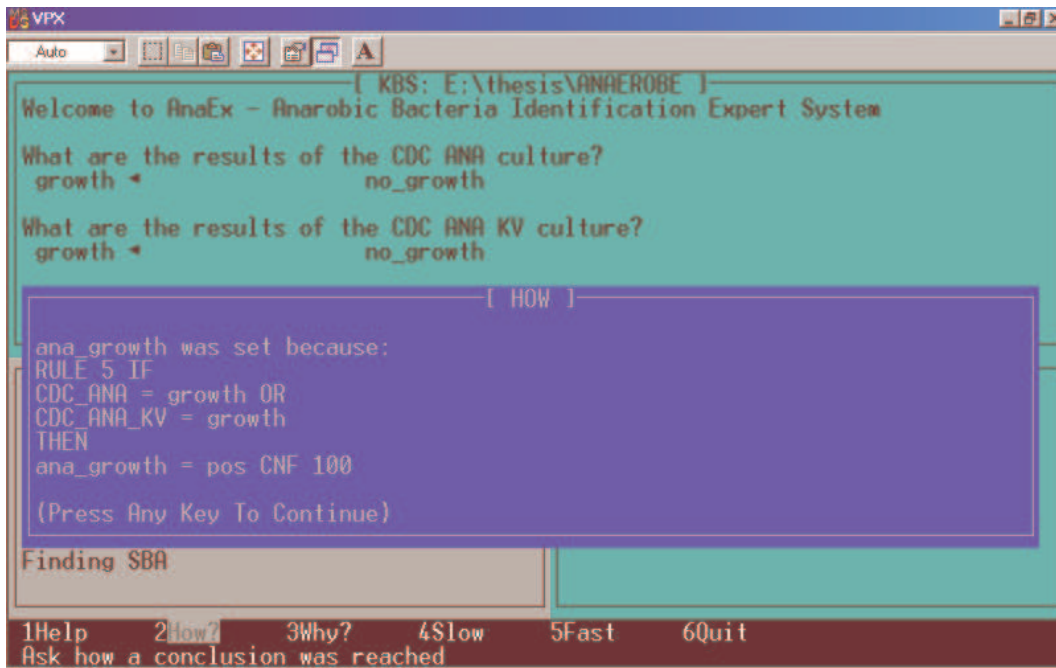


Figure 5.2 Example of the "How?" Debug Feature

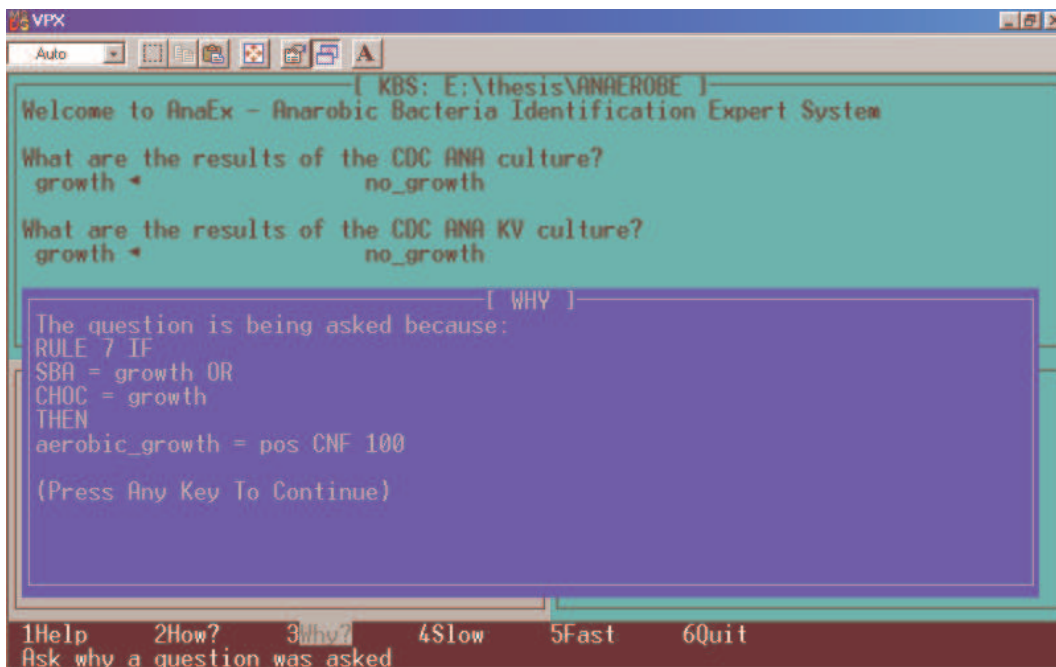


Figure 5.3 Example of the "Why?" Debug Feature

If a decision maker is using a KBS to explore various scenarios, it is valuable to be able to change a variable to see how that change could affect a solution. The “What if” feature answers questions such as “*What if ‘X’ was chosen instead of ‘Y’ at the last prompt?*”

Frequently, a developer has to deal with the problem of a knowledge base that runs, but seems to contain some logical errors. An aid in this situation is to perform a *trace* on the consultation to see what is going on behind the scenes. A trace saves the search pattern used by the inference engine, which can be viewed in two ways: as a text tree, or as a graphics tree. Examples of both are shown in Figures 5.4 and 5.5.

```

(= yes CNF 0 )
! microcode
! ! Testing NOMICROCODE2
! ! ! anaerobe
! ! ! ! Testing 3
! ! ! ! ! ana_growth
! ! ! ! ! ! Testing 5
! ! ! ! ! ! ! CDC_ANA
! ! ! ! ! ! ! ! (= growth CNF 100 )
! ! ! ! ! ! ! ! CDC_ANA_KV
! ! ! ! ! ! ! ! (= growth CNF 100 )
! ! ! ! ! ! (= pos CNF 100 )
! ! ! ! ! aerobic_growth
! ! ! ! ! ! Testing 7
! ! ! ! ! ! ! SBA
! ! ! ! ! ! ! ! (= no_growth CNF 100 )
! ! ! ! ! ! ! ! CHOC
! ! ! ! ! ! ! ! (= no_growth CNF 100 )
! ! ! ! ! ! Testing 8
! ! ! ! ! ! (= neg CNF 100 )
! ! ! ! ! gram_stain
! ! ! ! ! ! (= other CNF 100 )
! ! ! ! (= other_genus CNF 100 )
! ! ! ! (= pos CNF 100 )
! ! (= -1 CNF 100 )
! ! (= other_genus CNF 100 )
! message
! ! Testing MESSAGE_AEROBE
! ! Testing MESSAGE_OTHER
! ! (= other_genus CNF 100 )

```

**Figure 5.4 Text Trace of a Consultation Session**



**Figure 5.5 Partial Graphic Trace of a Consultation Session**

One simple way of constructing a small knowledge-based system is to represent the expert's knowledge in a table of examples. Once the example cases are recorded in a file (text, database, or spreadsheet), the file can be processed through an *induction algorithm*. This processing automatically converts the examples into a working minimal knowledge base [29]. In a less robust way, the *Induce* option in the VP-Expert system shell can induce a simple knowledge base from a table of examples. However, the induction algorithm does not allow for any optimization; every row will become one rule in the knowledge base.

#### ***5.4 Limitations of VP-Expert***

VP-Expert served as a good starting point for developing this expert system. However, it had some limitations.

Foremost among the limitations is its database connectivity. One of the powerful features of today's expert system shells is their ability to integrate with existing database and spreadsheet files. While VP-Expert is capable of connecting to database files, there were two major issues in this feature.

First, VP-Expert is only capable of connecting to older xBASE databases. It cannot connect to more advanced databases, such as SQLServer, Oracle, or even M.S. Access. That limits the type of database files that can be used and also limits the extensibility of the application.

The second factor is the ability to write values to the database and save them. VP-Expert claims to be capable of performing such tasks, however no documentation was found to explain how that is to be achieved. Without that important feature, the results of the consultations cannot be automatically saved for future reference. This concept is especially important in the field of medicine, where medical records and test results must be saved for up to ten years from the time of service.

The other major limitation in VP-Expert is the user interface. VP-Expert offers some options to provide a "user-friendly" interface. One option is the ability to create different windows with a consultation session – e.g. an instructions window. There is a feature that allows for controlling all the windows

– opening and closing them. Another feature is controlling the foreground and background color. The “facts” and “rules” windows may be eliminated from the consultation screen.

VP-Expert was developed in the early 1990’s. It is DOS-based. This does not allow for it to achieve the level of technological enhancements available to modern operating systems. It also makes it platform dependant. It can only run on an IBM-PC type of a computer. It is not easy to implement on the Internet.

Due of these and other limitations, AnaEx was prototyped using VP-Expert, but developed using different technologies, to try to address some of these restrictions.

## Chapter 6

### 6. Using VB With Access

#### 6.1 *Visual Basic and Access*

In an effort to bring enhancements to AnaEx, the prototype was used to develop the system using Visual Basic and Access. Visual Basic is a programming language. The “Visual” part refers to the method used to create the *graphical user interface (GUI)*. Rather than writing numerous lines of code to describe the appearance and location of interface elements, pre-built objects are simply added in place on the screen.

The “Basic” part refers to the **BASIC** (**B**eginners **A**ll-Purpose **S**ymbolic **I**nstruction **C**ode) language. Visual Basic has evolved from the original BASIC language and now contains several hundred statements, functions, and keywords, many of which relate directly to the Windows GUI. Useful applications can be created rapidly by learning just a few of the keywords, yet the power of the language allows for the achievement of anything that can be accomplished using any other Windows programming language.

There are several benefits to using VB. *Data access* features allow for the creation of databases, front-end applications, and scalable server-side components for most popular database formats, including Microsoft SQL Server and other enterprise-level databases. ActiveX technologies allow for the use of the functionality provided by other applications, such as Microsoft Word word

processor, Microsoft Excel spreadsheet, and other Windows applications. Internet capabilities make it easy to provide access to documents and applications across the internet or intranet from within applications, or to create internet server applications. The finished application is a true *.exe* file that uses a Visual Basic Virtual Machine that can be freely distributed [31].

Microsoft Access is a relational database. As a database system, Access has many good aspects and some weaknesses. It is important to clarify the “class” that Access falls into. Access is a desktop database package. It is not designed to compete with full database server systems such as Oracle or SQL Server, whose engines are superior in terms of speed and multi-user capabilities. This is usually the first perceived weakness: it does not provide good performance when run across a network and when more than a handful of people use it at once. However, if one considers what Access is designed as - a desktop database with limited multi-user capabilities - its performance capabilities are good.

Compared to other desktop database packages, Access has one huge advantage. It is likely that most industries are running Windows as an operating system and using Microsoft Office as an application base. Access integrates well with these packages and data transfer between Access and the other Office components is relatively easy. In addition, put side by side to the other desktop databases, Access is both rich in features and powerful. Access 2000 probably puts Access above any other desktop database available [32].

## 6.2 Database Design

Database systems are designed for managing large amounts of data, and they provide many important features that object oriented programming languages do not, e.g.: fast queries, sharing of objects among programs, sophisticated error handling for database operations, and permanent storage.

*Relational database systems (RDBMS)* and record oriented systems based on *B-Tree* or *Indexed Sequential Access Method (ISAM)* are the standard systems used in many software developments. In relational databases all data is stored in two-dimensional tables. These tables were originally called *relations*, which led to the name relational databases. An application will generally have a series of tables to hold its data. A database is nothing more or less than a collection of two-dimensional tables [33].

According to Elmasri & Navathe [34], the normalization process, as first proposed by Codd (1972), takes a relation schema through a series of tests to “certify” whether or not it belongs to a certain *normal form*. Initially, Codd proposed three normal forms, which he called *First, Second, and Third Normal Forms*. A stronger definition of 3NF was proposed later by Boyce and Codd and is known as Boyce-Codd Normal Form (BCNF). All these normal forms are based on the functional dependencies among the attributes of a relation. Later, a Fourth Normal Form (4NF) and a Fifth Normal Form (5NF) were proposed, based on the concepts of multi-valued dependencies and join dependencies, respectively.

**Normalization** of data can be looked on as a process during which unsatisfactory relation schemas are decomposed by breaking up their attributes into smaller relation schemas that possess desirable properties. One objective of the original normalization process is to ensure that the update anomalies do not occur.

A **relation** is defined as a *set of tuples*. By definition, all elements of a set are distinct; hence, all tuples in a relation must also be distinct. This means that no two tuples can have the same combination of values for all their attributes.

**First Normal Form** is now considered to be part of the formal definition of a relation; historically, it was defined to disallow multivalued attributes, composite attributes, and their combinations. It states that the domains of attributes must include only atomic (simple, indivisible) values and that the value of any attribute in a tuple must be a single value from the domain of that attribute [35].

**Second Normal Form** is based on the concept of fully functional dependency. A functional “ $X \rightarrow Y$ ” relationship is a fully functional dependency. Removal of any attribute A from X means that the dependency does not hold any more. A relation schema is in 2NF if every nonprime attribute in relation is fully functionally dependent on the primary key of the relation. It also can be restated as: a relation schema is in 2NF if every nonprime attribute in relation is not partially dependent on any key of the relation [35].

**Third Normal Form** is based on the concept of transitive dependency. A functional dependency “ $X \rightarrow Y$ ” in a relation is a transitive dependency if there is a

set of attributes  $Z$  that is not a subset of any key of the relation, and both  $X \rightarrow Z$  and  $Z \rightarrow Y$  hold. In other words, a relation is in 3NF if, whenever a functional dependency  $X \rightarrow A$  holds in the relation, either (a)  $X$  is a *superkey* of the relation, or (b)  $A$  is a prime attribute of the relation [35].

**Boyce-Codd Normal Form** is stricter than 3NF. A relation schema is a BCNF if, whenever a functional dependency  $X \rightarrow A$  holds in the relation, then  $X$  is a superkey of the relation. The only difference between BCNF and 3NF is that condition (b) of 3NF, which allows  $A$  to be prime if  $X$  is not a superkey, is absent from BCNF [35].

Adhering to the third normal form, while theoretically desirable, is not always practical. In theory, normalization is worth pursuing; however, many small tables may degrade performance or exceed open file and memory capacities. It may be more feasible to apply third normal form only to data that changes frequently.

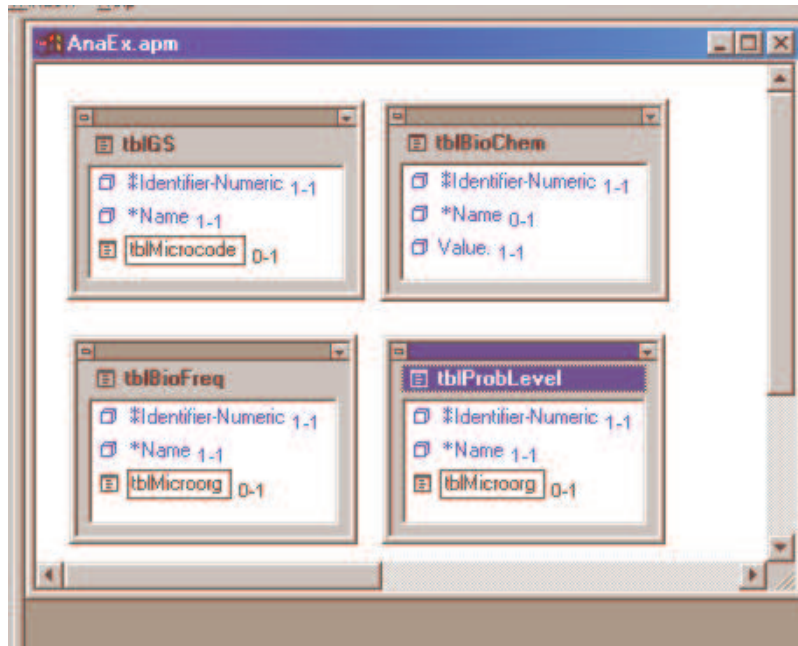
AnaEx's database design adheres, in the strict definition, to 2NF. Adhering to 3NF would have been costly, and cumbersome for this application. The database is designed for maximum efficiency. This efficiency is achieved by storing both a foreign key (microcodeID) and its value (microcode) in one of the tables. This makes lookups for that value in the table more prompt. The trade-off is the extra storage of the microcode value, and of course, the non-adherence to 3NF. In actuality, however, the database does adhere to BCNF with the exception of that anomaly. The microcodeID and microcode values will seldom if ever change in the database, and so, the decision taken about the design is valid.

The data is separated into three sections: reference, knowledge base and specimen data.

The reference section is composed of four look-up tables. These tables ease the process of inputting repeat data, as well as keeping the data consistent. If a data element changes in the future, there is no need to modify the entire database for all records that contained that data element. Rather it is a simple task of updating the element in the look-up table.

Figure 6.1 shows a *Semantic Object Model (SOM)* diagram of the reference tables. **Semantic** is an adjective that means “of or relating to meaning.” **Semantic Objects** are defined as “a named collection of attributes that sufficiently describes a distinct entity.” Semantic object modeling is concerned with capturing the meaning of the data being modeled, specifically what the data means to the user.

The four reference tables are: tblGS, tblBioFreq, tblProbLevel, and tblBioChem. Each look-up table has two entries, “ID” as primary key and “Name”. The table tblGS contains all the possible results for the gram stain test that the application is capable of processing. The tables, tblBioFreq and tblProbLevel, contain the bio-frequency and the probability level interpretations for the microcode respectively. Finally, tblBioChem, stores the names of all the biochemical tests involved in the system as well as the value assigned to them for a positive result.

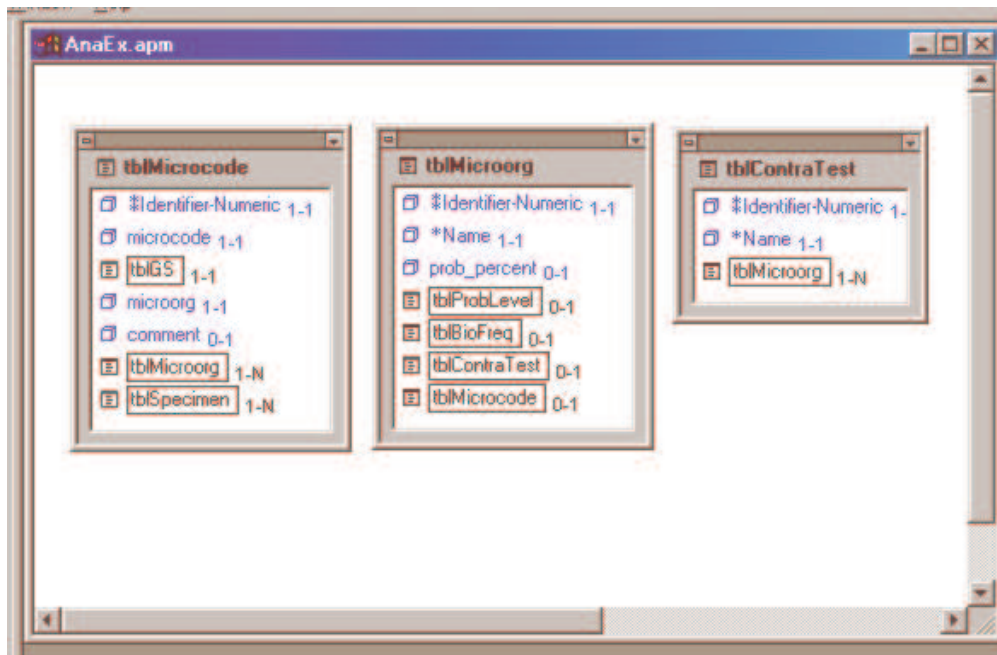


**Figure 6.1 SOM Diagram of the Reference Tables**

The next set of tables, and by far the most critical set, is the knowledge base tables. This set is comprised of three tables: tblMicrocode, tblMicroorg, and tblContraTest. The information about the microcode is stored in tblMicrocode. It contains the numeric microcode, the name of the microorganism, and any special notes. It is linked to the second table, tblMicroorg, by the tblMicroorg table attribute, to store all the relative information about each of the microorganisms that are associated with that microcode.

The tblMicroorg table contains mainly a numeric identifier, and the genus-species name of the microorganism. It also contains the probability percentage, and links to the probability level table (tblProbLevel), bio-frequency table (tblBioFreq), and the contraindicated tests table (tblContraTest). It is linked back to the microcode table by the tblMicrocode table attribute.

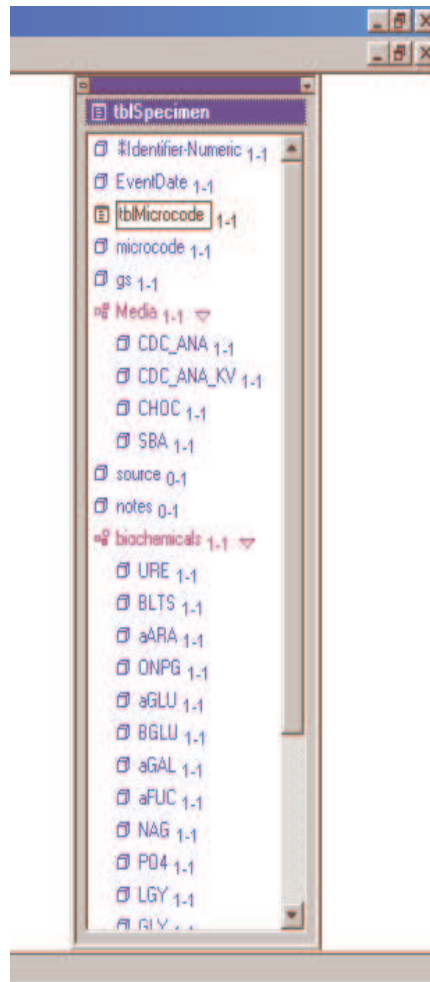
The last table, tblContraTest contains the name of the contraindicated tests for that microcode. These are the tests, which are considered contraindication against the choice. It is linked back to the microorg table by the tblMicroorg table attribute.



**Figure 6.2 SOM Diagram for the Knowledge Base Tables**

Lastly, the specimen data is contained in one table: tblSpecimen. This is the table that contains the historical records of the specimen. It is also used for tracing the decision. It contains a numerical identifier, and the date of the consultation. It is linked to the microcode table with the tblMicrocode table attribute, and contains the microcode itself, and the gram stain (gs) results. A *group* of attributes called “Media” stores the results of the media cultures, and

another group “biochemicals” stores the results of the biochemical tests. Two other optional attributes, source and notes, are available.



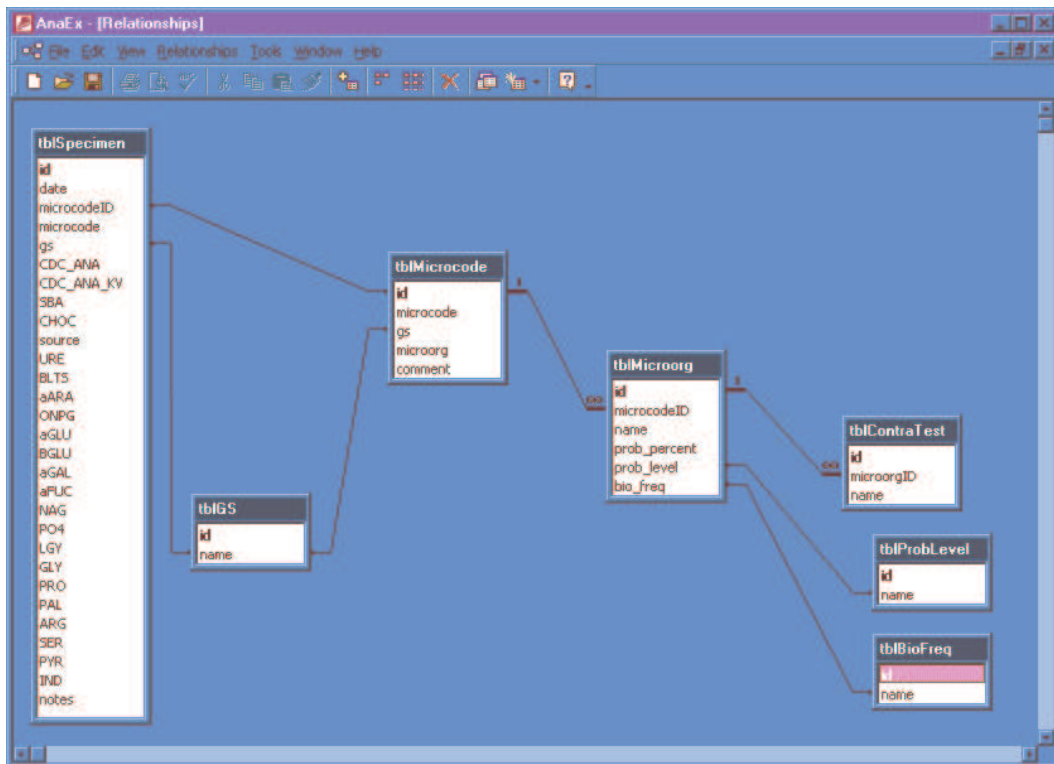
**Figure 6.3 SOM Diagram for the Specimen Table**

During the database design, the tables have been normalized to the 2NF standard. Care was taken to avoid redundancy and duplication. However, it was necessary to keep a good balance between normalization and the cost associated with it. Table 6.1 demonstrates the object relationship for the AnaEx database. All

*primary keys* are underlined and all *foreign keys* are italicized. Figure 6.4 illustrates the relationship diagram among the database tables.

<b>Reference Tables</b>	
tblGS	( <u>id</u> , name)
tblBioFreq	( <u>id</u> , name)
tblProbLevel	( <u>id</u> , name)
tblBioChem	( <u>id</u> , <i>biochem</i> , value)
<b>Knowledge Base Tables</b>	
tblMicrocode	( <u>id</u> , <i>microcode</i> , <i>gs</i> , <i>microorg</i> , comment)
tblMicroorg	( <u>id</u> , <i>microcodeID</i> , name, <i>prob_percent</i> , <i>prob_level</i> , <i>bio_freq</i> )
tblContraTest	( <u>id</u> , <i>microorgID</i> , name)
<b>Specimen Table</b>	
tblSpecimen	( <u>id</u> , date, <i>microcodeID</i> , <i>microcode</i> , <i>gs</i> , CDC_ANA, CDC_ANA_KV, SBA, CHOC, source, URE, BLTS, aARA, ONPG, aGLU, BGLU, aGAL, aFUC, NAG, PO4, LGY, GLY, PRO, PAL, ARG, SER, PYR, IND, notes)

**Table 6.1 Object Relationship for the AnaEx Database**



**Figure 6.4 Relationship Diagram for the AnaEx Database**

Figure 6.5 shows an example of the data in the knowledge base tables. In tblMicrocode, id 5 contains the microcode “45”, which is identified as the microorganism “*F. necrophorum*”. There is a comment that reads: “Cells may be pleomorphic. Isolated from necrotic lesions, abscesses, blood, and other clinical specimen.” This microcode actually maps to two microorganisms. The first one, and the more likely candidate, is *F. necrophorum*. It has a probability percentage of 96.7%, which is interpreted as “Adequate” for the identification to be acceptable. There are two tests which are considered contraindications against the choice: ARG and SER. The other microorganism is *F. nucleatum*. Its probability percentage is only 3.3% and the contra-indicatory test is SER.

id	microcode	gs	microorg	comment
7	-1	GPC	No Microorg identified	
2	0	GNR	Probability Overlap Among First Two	
3	0	GPC	Probability Overlap Among First Two	
4	44	GNR	Fusobacterium sp.	
5	45	GNR	F. necrophorum	Cells may be pleomor

id	name	prob_perce	prob_level	bio_freq
12	F. necrophorum	96.7	Adequate	
4	ARG			
5	SER			
* AutoNumber)				
13	F. nucleatum	3.3		
6	SER			
* AutoNumber)				
* AutoNumber)				

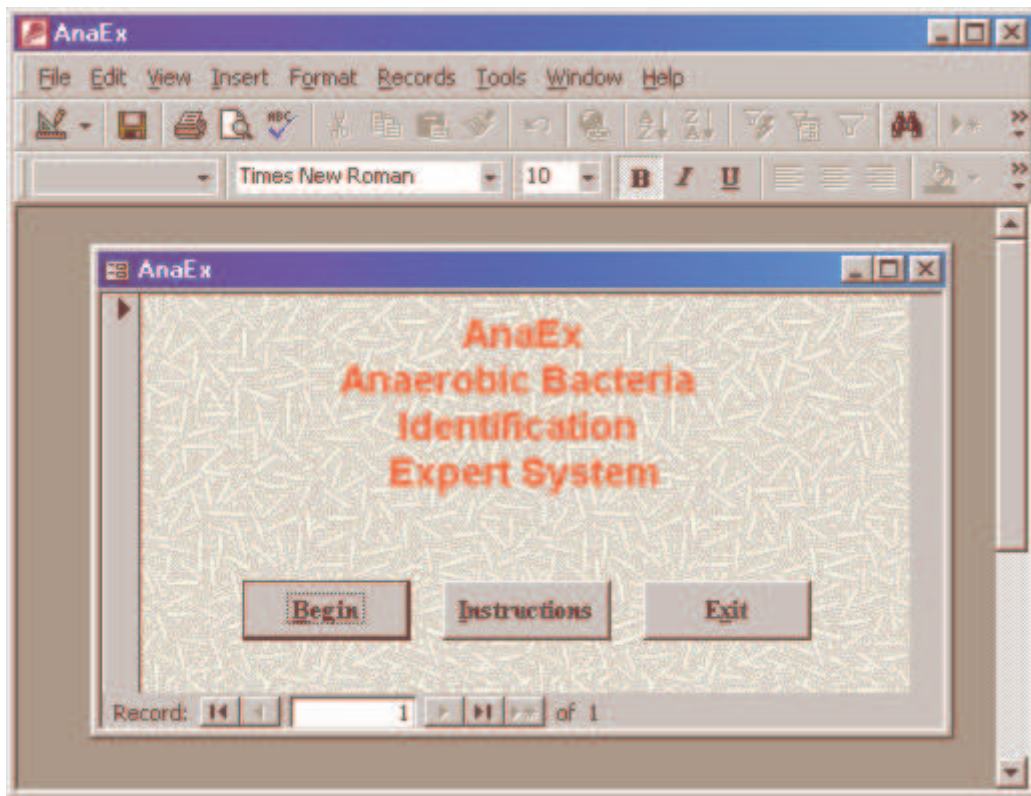
Figure 6.5 An Example of the Data in the Knowledge Base Tables

### 6.3 User Interface Design

One of the factors that led to the development of AnaEx in VB was VP-Expert's weak user interface. Special care was given to that aspect in this development. The UI built in VB is designed to be user-friendly, while still providing all the functionality of the system in a cohesive manner.

The system begins with a simple introductory form (Figure 6.6). This form contains three buttons, with the choices of "Begin" which would begin a new consultation session, "Instructions" which provides the user with some short

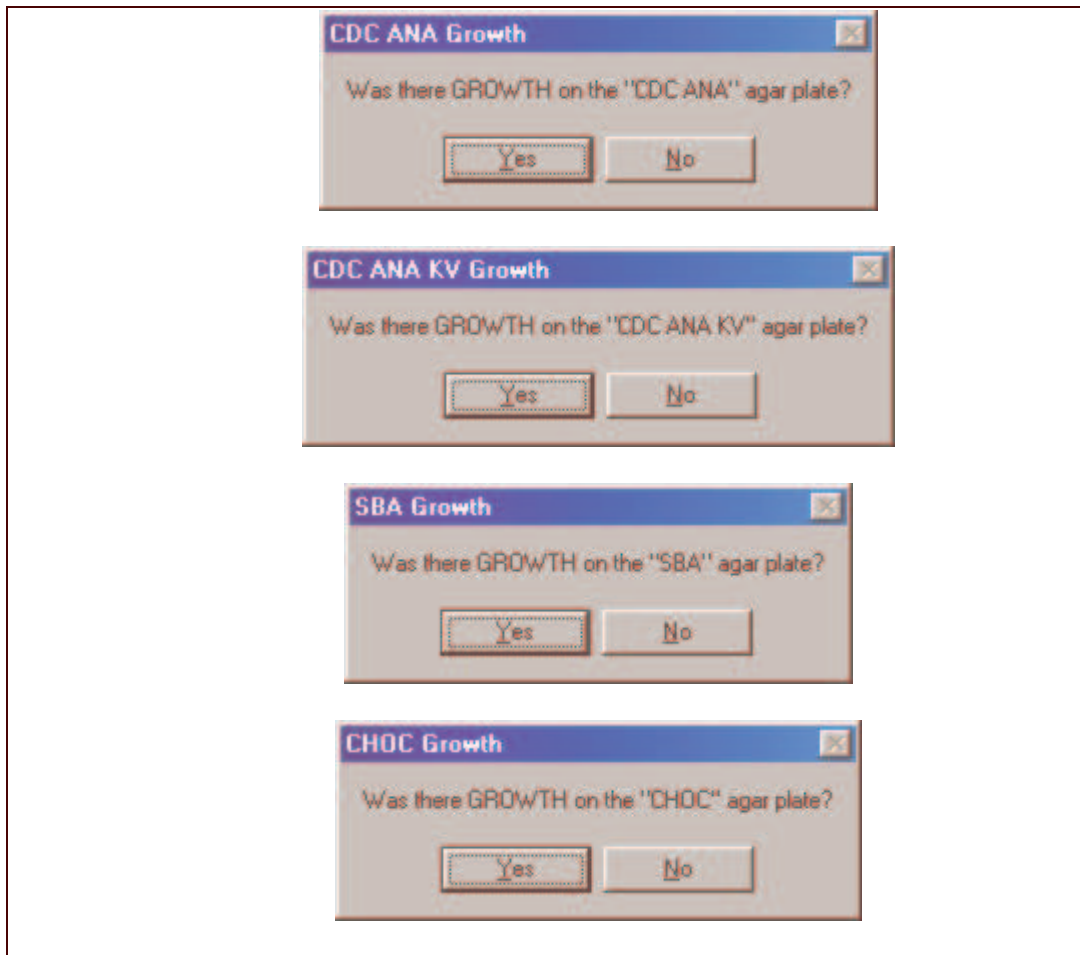
but helpful instructions on how the system works, and finally “Exit” to exit the system.



**Figure 6.6 First Form in AnaEx System**

When the user begins a consultation session, a request for the specimen ID is made. If the specimen ID is found in the specimen table, the user is informed that this specimen was already processed earlier, and is given options to either review the report for that specimen, or end the session. If the specimen ID is a new one, the expert system begins the process by asking a series of questions. It first asks questions to determine the aerotolerance of the microorganism. This is done by asking for the growth results on the two anaerobic media: “CDC ANA”

and “CDC ANA KV”. Next, the aerobic (SBA and CHOC) media are queried. Each one of these four questions comes up in a small dialog box (Figure 6.7).



**Figure 6.7 Dialog Boxes for the aerotolerance test**

If the microorganism is found to be an anaerobe, the system then inquires about the gram stain results and the specimen source (Figure 6.8). Subsequently, the system asks for the results of the biochemical tests in one complete form (Figure 6.9). On this form, it automatically calculates each code, and eventually, the microcode.

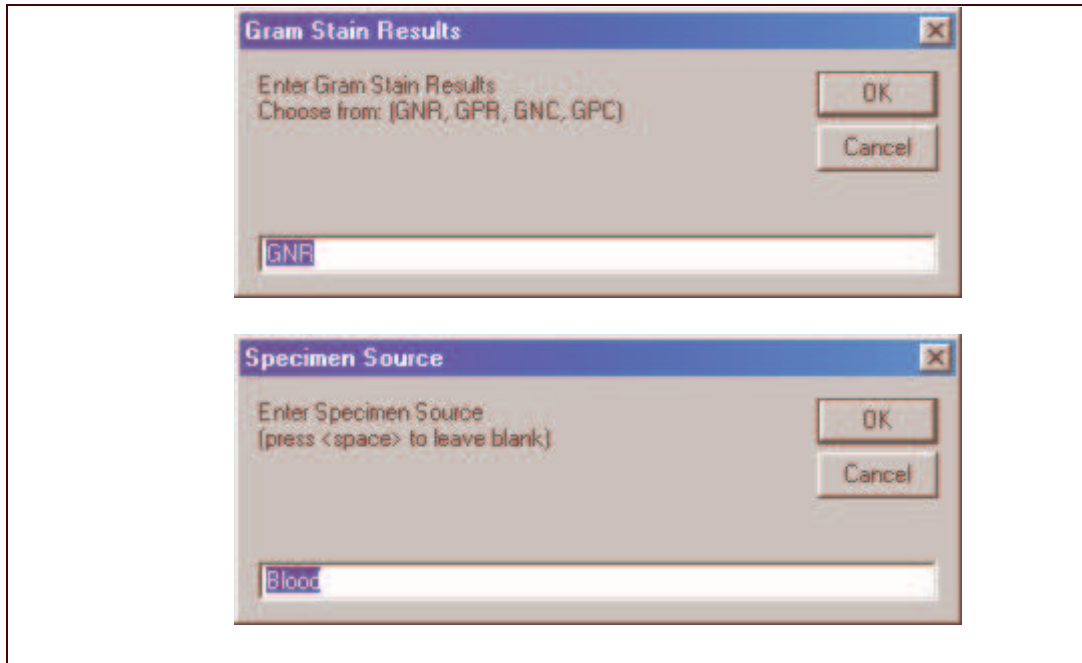


Figure 6.8 Dialog Boxes for Gram Stain and Specimen Source

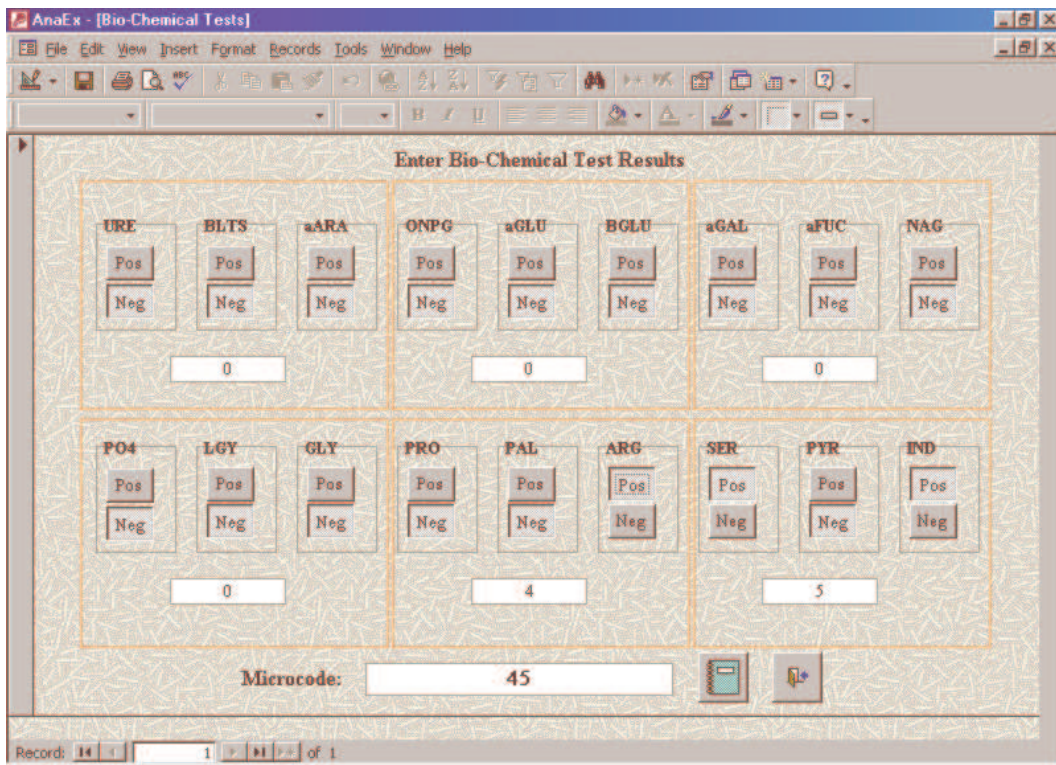


Figure 6.9 Bio-Chemical Test Results Form

After the microcode is calculated, the user clicks on the “report” icon. Before the report is generated, the user is asked to enter any specimen notes that would be saved as part of the record. This can be used for any special circumstances regarding the specimen, or any comments that the user wishes to document. Finally, a report is generated with all the relevant information required for the identification of the microorganism. A sample report may be seen in Figure 6.10.

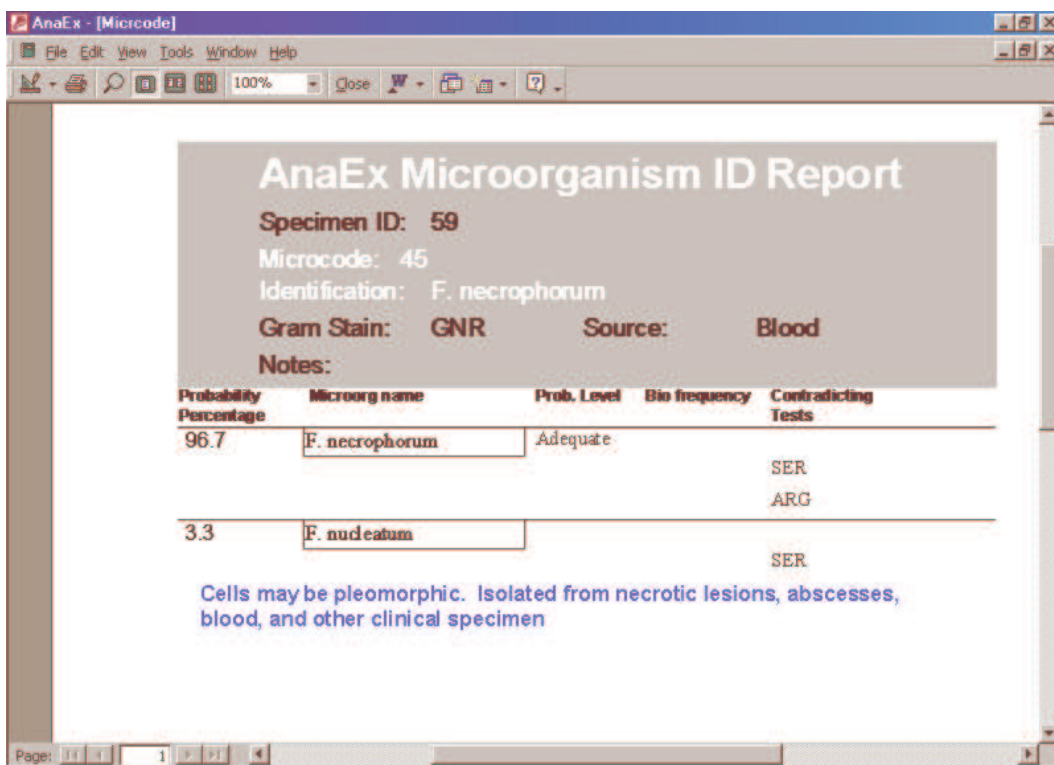


Figure 6.10 AnaEx Microorganism ID Report

## **6.4 Limitations**

Developing the expert system using VB and Access provided the enhancements that were lacking in the VP-Expert approach. It provided more robust database connectivity, and greatly enhanced the user interface. It also allowed for future enhancements using more advanced technologies, such as making the system web-enabled.

These enhancements, however, did not come freely. One of the powerful features of VP-Expert is the debugging tool it contains. The ability to trace a consultation and know what rule the expert system was evaluating, and what value each of the variables has is a very useful tool. Also the “What If?” tool is another helpful feature.

The VB implementation of AnaEx provided some of these tools, but not to the same extent as VP-Expert. The specimen table that saves all information about the session, and the report that can be reproduced using that table implement the tracing ability. It does not provide VP-Expert’s ability to halt the session at any point and request “Why?” and “How?” resolutions. It also lacks the “What If?” tool that would enable the changing of a variable to see how that change could affect a solution. That feature, however, can also be implemented by the use of the specimen database; a variable’s value may be modified and another report may be generated. This, of course, does not measure up to VP-Expert’s dynamic mechanism.

These features are indeed important, but they were not considered critical for the scope of this project. These features are more significant in research-type projects, where changing variables to access different solutions, is essential. The scope of this project is more concerned with the practical use of the system. This system would be used in a clinical setting where experimental research is being routinely done, but rather more customary yet crucial testing. It was deemed more imperative, given the nature of the medical field, to concentrate more on the importance of such concepts as saving the medical records and the factors leading to the solution. Due to such aspects as lawsuits and medical-legal cases, there is a much superior necessity for ensuring proper historical data banking, than for experimental data manipulation.

## **6.5 *Correlation to ES***

The AnaEx solution implemented using VB and Access does not resemble the traditional Expert System design or implementation. There are certain aspects that make this distinction.

First, the rules are not in the traditional rule format. Most of the rules either exist as complete subroutines or functions, are incorporated into subroutines along with other rules, or may even be separated into numerous subroutines. For example, RULE 1 in the original knowledge base was:

RULE 1

```
IF ana_growth = neg
THEN id = aerobe
BECAUSE "No growth on anaerobic media";
```

In the VB/Access implementation, it became:

```
Sub getAnaGrowth()
    Dim strMsg, strTitle

    If (Not anaGrowth) Then
        strMsg = "Aerobic organism - no growth on anaerobic media"
        strTitle = "Non-Anaerobic organism"

        MsgBox strMsg, vbExclamation, strTitle
    End If
End Sub
```

Actually, that is not the complete rule, since it calls the function “anaGrowth”, which evaluates the aerotolerance of the microorganism on anaerobic media. It asks the user about the results of the cultures and sets the values of the variables. All these tasks were performed separately in the VP-Expert implementation. The code for the “anaGrowth” function reads as:

Function anaGrowth() As Boolean

Dim strMsg, strTitle

strMsg = "Was there GROWTH on the ""CDC ANA"" agar plate?"

strTitle = "CDC ANA Growth"

If MsgBox(strMsg, vbYesNo, strTitle) = vbNo Then

myCDC\_ANA = False

Else

myCDC\_ANA = True

End If

strMsg = "Was there GROWTH on the ""CDC ANA KV"" agar plate?"

strTitle = "CDC ANA KV Growth"

If MsgBox(strMsg, vbYesNo, strTitle) = vbNo Then

myCDC\_ANA\_KV = False

Else

myCDC\_ANA\_KV = True

End If

If (myCDC\_ANA Or myCDC\_ANA\_KV) Then

anaGrowth = True

Else

anaGrowth = False

End If

End Function

Second, the number of “rules” or in this case, subroutines, is significantly reduced for such a considerable system, such as AnaEx, which is potentially capable of identifying any kind of anaerobic bacteria. The reason for that is that the core of the knowledge base is implemented on a higher level of abstraction: a database. A traditional rule for identifying the microorganism *F. necrophorum*, may look like:

```
RULE F_necrophorum
    IF      anaerobe = pos          AND
          gram_stain = GNR        AND
          microcode=45
    THEN id = F_necrophorum
```

This would have to be repeated for every ID in the system. AnaEx only contains a very small subset of the GNR microorganisms that can be potentially identified; it contains 130 microorganism ID’s. That would equate to 130 rules just for resolving the “last piece of the puzzle”. It also means that each time a new ID is added to the system, the programmer must write a new rule. The VB/Access solution avoids this problem by abstracting that level of detail to database tables that the system will query from and get the correct solution. The hypothetical 130 “traditional” rules are replaced by the following set of simple statements:

```
Set MYDB = CurrentDb

strRptName = "rptMicrocode"

strQryName = "qryMicrocode"

'Set condition

mysql = "(((tblMicrocode.microcode)=" & txtMicrocode.Value & ")
AND ((tblMicrocode.gs)="" & myGS & ""))"

'Open report

DoCmd.OpenReport strRptName, acPreview, strQryName, mysql
```

With this solution, there is no need to code any new rules. A competent user with basic knowledge of databases can easily add new ID's to the knowledge base without the assistance of a programmer.

So the question remains: what was the purpose of developing the system in the “traditional” format at the beginning? There are many advantages to that approach.

First, It was very helpful, and even imperative to learn the logic and science of expert systems. At a distance, the logic seems very intuitive, but when one tries to implement even a simple expert system, it is discovered very quickly, that the logic is more involved. A good grasp of how an expert system “thinks” is necessary for any type of ES implementation. That is best learned by the actual design and development of an expert system. When AnaEx was first being

reasoned, an attempt to build it using a solution such as VB/Access was tried. It became obvious that such a solution would not be accomplished without a solid understanding of expert systems and the actual implementation of one.

Second, the design phase of the original system proved to be important as well as fundamental for designing this and future expert system solutions. Building block diagrams, dependency diagrams, and decision tables is an essential step in the system design. It was well worth the time and effort that was invested in the planning and designing of the project even before a single line of code was written. In the long run it saved a lot of time, effort and frustration. Again, the best way to learn this logic is to actually use it to build a system with it. AnaEx would not have been successful if it was not well designed.

Lastly, the knowledge learned from the traditional expert system implementation is practical. It has set baseline logic for expert system design. Enhancements to AnaEx can now be made without reverting back to the traditional expert system shells. Enhancements can be applied at this level of the system. Any enhancement, however, would require a complete pass through the design phase, including block diagrams, dependency diagrams, decision tables, and any other design tool. This is not a setback in design, but rather a stepping-stone to better system design.

Overall, despite the fact that there was time spent on a system that became merely a proto-type and was not actually used in the end product, it is not to be considered a waste of time by any means. The knowledge and expertise gained from it is invaluable. It will assist in better system designs in the future, as well as

easier enhancements to the existing systems. It was a very important and necessary step to be taken.

## Chapter 7

### 7. Test Results

In order to assess the success of AnaEx, it had to be examined in the natural setting of its task domain: the laboratory. Since anaerobes are only occasionally isolated in the laboratory, finding numerous tests to conduct testing was challenging. A number of hospitals that use the RapID ANA II test kit were contacted. They were asked to provide test results for samples that they have run using the traditional methods in the lab. Only two hospital laboratories, Memorial-Hermann Southwest and Memorial-Hermann Hospital were able to provide such data.

They provided the biochemical test results of the RapID ANA II kits, as well as gram stain results. Some data contained the source of the specimen, but not all; this is an optional data element in the system. The laboratories were asked how they initially identify the aerotolerance of the specimen, and they indicated that they follow the same method implemented in AnaEx: growth on anaerobic media, and lack of growth on aerobic media.

A total of 62 test cases were obtained from the laboratories. Out of those, 9 cases resolved to anaerobes with microcodes that were not found in the compendium, and were not identifiable using the system, however, the laboratories obtained the same irresolvable results. The rest of the samples were identified using AnaEx, and resolved to the same identification obtained in the

laboratories, using the conventional analysis methods. The system was able to identify 85% of the samples tested (53 out of 62). The results correlated 100% to the laboratory results. Table 7.1 and Figure 7.1 list the project test results. A group of 10 non-anaerobic bacteria were also tested using AnaEx and the result was that they all were identified as “non-anaerobic” and the system terminated at the “failed” aerotolerance test.

<b>#of Specimens</b>	<b>ID</b>
9	Not Found
1	<i>B. distasonis</i>
17	<i>B. fragilis</i>
4	<i>B. fragilis</i> Group Isolate
1	<i>B. merdae</i> (T4-1)
2	<i>B. ovatus</i>
6	<i>B. thetaiotamicron</i>
4	<i>B. uniformis</i>
2	<i>B. vulgatus</i>
2	<i>Bacteroides</i> sp.
2	<i>Pr. bivia</i>
2	<i>Pr. corporis</i>
2	<i>Pr. denticola</i>
6	<i>Pr. loescheii</i>
1	<i>Provetella</i> sp.
1	<i>Pr. melaninogenica</i>

**Table 7.1 Test Results**

Test Results

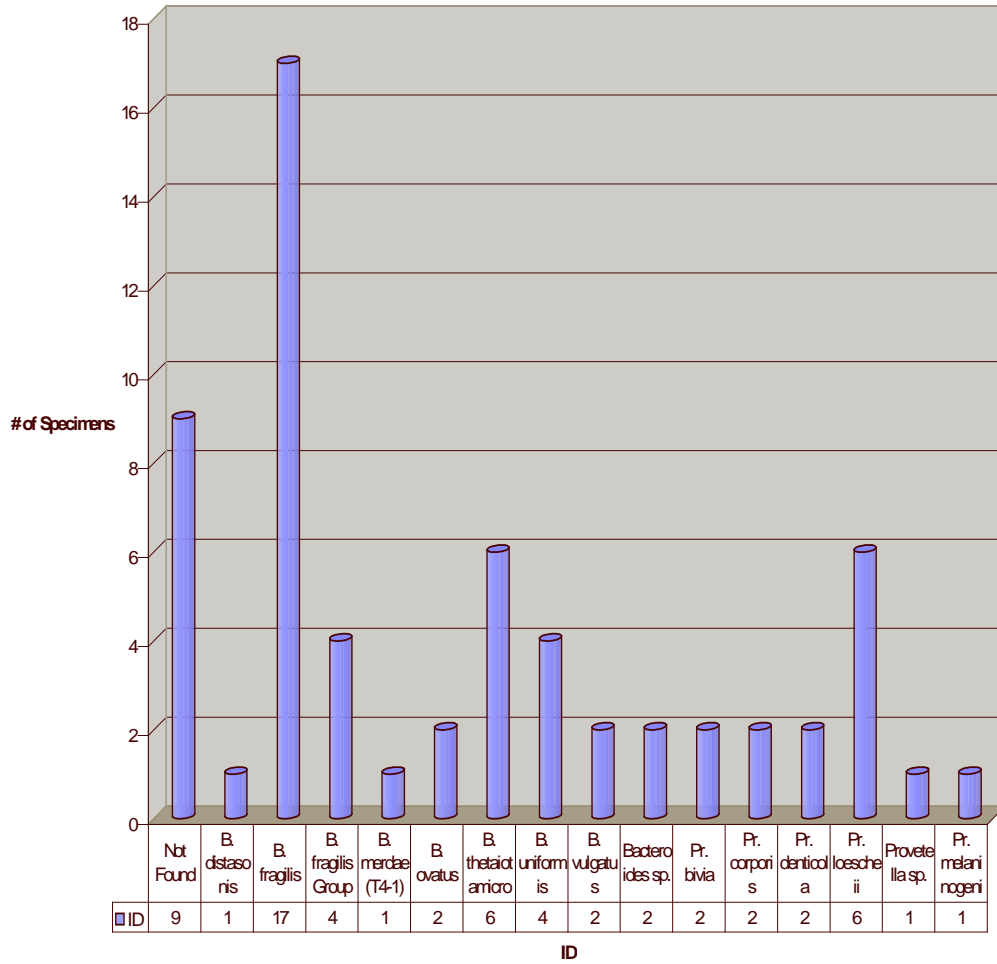


Figure 7.1 Test Results

## Chapter 8

### 8. Conclusion

#### 8.1 *Synopsis*

This thesis presents a solution to assist human experts with the identification of anaerobic bacteria. The solution presented utilizes the computer science field of artificial intelligence. This is realized by means of expert systems.

The system developed is not a complete stand-alone entity that can identify bacteria on its own. It still utilizes the human expert's intelligence. The identification it supplies is a suggestion that the human expert may accept or consider for further analysis, or completely reject. The human expert must evaluate the suggestion of the expert system against other data, such as the patient's symptoms, diagnosis, and medical history. These factors, along with others play a part in the definitive diagnosis of a disease.

AnaEx, the system discussed in this thesis, takes into consideration, three important clinical factors about the microorganism to be identified: aerotolerance, gram stain results, and the RapID ANA II microcode. Using these three components it is able to identify a microorganism, usually to the species level. It also supplies the user with the confidence levels for the solution, which are identified in the RapID ANA II system as "probability level" and "bio

frequency”. It also provides any contra-indicatory tests against the solution. All this data equips the technologist to better assess the validity of the solution.

The system stores all the information used to make the decision, as well as the specimen identifier, and the date of the analysis for future reference. This can be used as a means to “trace” the solution and can be provided in the event of the questioning of the decision. Other optionally requested and saved, nevertheless important, data elements are specimen source and any notes that the technologist wishes to save about the analysis.

The system was tested using data from two different area hospital laboratories, and the decisions made by AnaEx correlated 100% to the decisions made by the human experts, under the same testing conditions.

## **8.2 *Future Work***

The best way to assess how useful such a system would be in the industry is to present it to the users who would actually be using it. AnaEx was presented and demonstrated to the medical technologist of the Memorial-Hermann Hospital microbiology clinical laboratory staff.

The presentation was well received and the effort was commended. They were pleased that the field of computer science is sensitive to the needs of non-computerized industries, such as medical microbiology. They were enthusiastic

that such an intricate yet fascinating discipline, as artificial intelligence, would be used to solve thorny problems such as anaerobic bacteria identification.

The technologists offered several suggestions for improvement of the system. One of the simple suggestions offered was to make the RapID ANA test results screen (Figure 6.9) more intuitive. Since the tests are based on chromogenic (color) reactions, it would be easier for the technologist entering the results to simply pick the color of the test result and have the system interpret that to positive or negative, rather than manually interpreting the result themselves. For example, rather than choosing between “pos” and “neg” for URE, the technologist would choose between “red” and “yellow”. This would reduce the technologist’s time spent on the analysis, and prevent interpretation errors from occurring. Also, it was suggested that a clearer distinction about what “button” is being selected is made, or perhaps use a different approach of selecting, e.g. radio buttons.

Another suggestion was to include not only GRNs, but also all other types of bacteria that the system can potentially identify. This task would also be relatively painless, as it only involves the addition of data into the database. This task would need to be performed if this system is to be used in actual production, but for the sake of research, the system was restricted to a subset of GNR organisms.

One point mentioned by the users of RapID ANA II is that the IDS system performs poorly when used to identify GPC. The solution to identifying GPC organisms is to use gas chromatography. Hence, it was also suggested that if

AnaEx is to be a complete system, it should include a module to assess GPC's using gas chromatography.

One final and significant suggestion that was made involved the actual design of AnaEx. The laboratory does not use the RapID ANA II system for the identification of all anaerobes. Instead, they follow a certain "flow chart" that involves growth patterns and sensitivity to high potency antibiotic disks. The anaerobic GRNs are cultured on *Bacteroides Bile-esculin (BBE) Agar* and observed for growth and esculin hydrolysis. They are also cultured anaerobically in the presence of antibiotic disks and examined for zones of inhibition of growth. The antibiotics used are: *Kenamycin (1000 µg)*, *Vancomycin (5 µg)*, and *Colistin (10 µg)*. For example, a microorganism that is resistant to all three antibiotic disks and grows on BBE is identified as *B. fragilis*. This "pattern-matching" method of identification usually does not always supply identification down to the species level, except in very common cases such as *B. fragilis*. It usually presents identification only down to the genus level, such as *Porphyromonas sp.*, which is sensitive only to Vancomycin, and does not grow on BBE.

When questioned about the validity of supplying only genus-level identification of anaerobes, the technologists supplied three reasons. First, the physicians are not always interested in a species-level identification, and are satisfied with a genus-level identification, since most species in the same genus would exhibit similar behavior. Second, the physicians cannot wait for the complete identification of an anaerobe, and will actually begin antibiotic treatment with even an initial identification of "anaerobe isolated - identification

to follow”. Last, but certainly not least, and probably most important, is cost. Performing the RapID ANA II test costs six times as much in material costs, and three times as much in technologist time. Hence the BBE/Disks solution is employed for most anaerobes isolated, and the RapID ANA kit is reserved for more difficult or uncommon bacteria isolated. A great improvement to AnaEx would be to include this logic in the system to more closely mimic the steps utilized by the technologist in identifying the microorganisms.

Another area that can be improved is the database design. The database strictly adheres to 2NF. It actually adheres to BCNF with the exception of the presence of both the microcode and the microcodeID in the specimen table. One enhancement to the system would be to make the database BCNF compliant while still keeping it efficient. This may or may not be possible, but it is worth some research.

### ***8.3 Final Words***

This thesis presented a solution to a problem that is encountered in the clinical laboratory: identifying anaerobic bacteria. There has been much work done in the area of aerobic bacterial identification, but less work in the anaerobic area, because of the infrequent encounter of such microorganisms. This problem, the infrequency of exposure to anaerobic bacteria, hinders experience in the field. The project developed, AnaEx, presents a simple solution. When presented to

human experts in the field, it was received with enthusiasm, and much appreciation. Suggestions were offered to make it more useful and more powerful, and these suggestions are taken very seriously. Overall, the project was a success, and perhaps a pioneer in its task-domain. Using an expert system to assist technologists in identifying anaerobes was certainly stimulating.

## Bibliography

- [1] *Dictionary of Computing*. New York: Oxford University Press, 1986.
- [2] Bishop, Peter. *Fifth Generation Computers Concepts, Implementations & Uses*. Chichester, England: Ellis Horwood Ltd, 1986.
- [3] Brule, James F. *Artificial Intelligence: Theory, Logic and Application*. Blue Ridge Summit, PA: TAB Books, 1986.
- [4] Forsyth, Richard. *Expert Systems: Principles and Case Studies*. London: Chapman and Hall Computing, 1984.
- [5] Harmon, Paul and King, David. *Expert Systems: Artificial Intelligence in Business*. New York: Wiley, 1985.
- [6] Liebowitz, Jay. *Introduction to Expert Systems*. Santa Cruz, CA: Mitchell Publishing, Inc, 1988.
- [7] Michaelsen, Robert H., Michie, Donald and Boulanger, Albert. "The Technology of Expert Systems." *Byte* 10, no. 4 (April 1985).
- [8] Rich, Elaine and Knight, Kevin. *Artificial Intelligence*, Second Edition. New York: McGraw-Hill, 1991.
- [9] Waterman, Donald A. *A Guide to Expert Systems*. Reading, MA: Addison-Wesley, 1986.
- [10] Winston, Patrick H. and Prendergast, Karen A. *The AI Business: Commercial Use of Artificial Intelligence*. Cambridge, MA: The MIT Press, 1984.
- [11] Lenat, D.B. and R.V. Guha. *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Reading, MA: Addison-Wesley Publishing Co., Inc., 1990.
- [12] Nikkei AI. "General Review of Expert Systems in Use." *Nikkei AI Special Issue* (Winter 1992).
- [13] Shortliffe, E.H. *Computer-Based Medical Consultation: MYCIN*. New York, NY: American Elsevier, 1976.
- [14] Encyclopædia Britannica. 2000 ed.

- [15] Sweet, Steven. "Think About It: Artificial Intelligence & Expert Systems." *How Computers Work*. 3, no. 4, (November 1999).
- [16] Feigenbaum, E. A., P. McCorduck et al. *The Rise of the Expert Company*. New York, NY: Times Books, 1988.
- [17] Mangels, Jim. "Anaerobic Bacteriology: In Today's Cost-Effective Environment." *Anaerobic Bacteriology* 4, no. 1, (January 1996).
- [18] Isenberg, H.D. *Clinical Microbiology Procedures Handbook*. Washington, D.C.: American Society for Microbiology, 1992.
- [19] Finegold, S.M. and W.L. George. *Anaerobic Infections in Humans*. San Diego, CA: Academic Press, Inc., 1989.
- [20] Blekas, Kostas D. *Expert Systems in Medicine*. Athens, Greece: National Technical University of Athens, 1995; <http://www.cslab.ece.ntua.gr/~kblekas/medicine.html>
- [21] Department of Microbiology and Immunology. *Microbiology and Immunology On-Line*. Columbia, SC: Department of Microbiology and Immunology at the University of South Carolina School of Medicine, 2001; <http://www.med.sc.edu:85/book/bact-sta.htm>
- [22] *Microbiology: Introduction*. 1999; [http://mindquest.net/biology/microbiology/outlines/u\\_intro.html](http://mindquest.net/biology/microbiology/outlines/u_intro.html)
- [23] Gannon, Colleen K. "Anaerobic Bacteriology Simplified" *Focus On Microbiology Newsletter* (Spring 1996).
- [24] DiRienzo, Joseph M. *Organization of The Microbial World*. 1997; <http://jmdserver.dental.upenn.edu/CourseBook2.html>
- [25] Berkow, Robert. *The Merck Manual of Medical Information-Home Edition*. 2001; [http://www.merck.com/pubs/mmanual\\_home/](http://www.merck.com/pubs/mmanual_home/)
- [26] "RapID ANA II", *BACTInews*, 6, no. 2 (April 1998).
- [27] Remel Microbiology Products Website; <http://www.remelinc.com/>
- [28] Remel Microbiology Products. *RapID ANA II System Compendium*, Version: V5.93.
- [29] Dologite, D. G. *Developing Knowledge-Based Systems Using VP-Expert*. New York, NY: Macmillan Publishing Company, 1993.

- [30] Friederich, Sylvia and Gargano, Michael. *Expert Systems Design and Development Using VP-Expert*. New York, NY: John Wiley & Sons, 1989.
- [31] MSDN Library. Microsoft Inc, 2001; <http://msdn.microsoft.com/library/>
- [32] Wolf, Simon. *Database Design*. A3, 2001 [http://www.athree.com/db\\_basics/design.html](http://www.athree.com/db_basics/design.html)
- [33] Bartels, Dirk. *A Comparison between Relational and Object Oriented Database systems for Object Oriented Application Development*. San Mateo, CA: Poet Software, 2001
- [34] Elmasri, R., & Navathe, S. (1994). *Fundamentals of Database Systems*. 2<sup>nd</sup> ed. Redwood City, CA: The Benjamin/Cummings Publishing Co.
- [35] Kettemborough, Clifford. *Database Normalization – Definitions*. Graziadio School of Business Management, Pepperdine University; <http://luna.pepperdine.edu/~ckettemb/class/DBNorm.html>

# Appendix A

## *Knowledge Base of AnaEx*

```
!*****
*****
! ANAExpert - Anaerobic Bacterial Identification Expert System
!
! KB: Anaerobe.kbs
!
```

DESCRIPTION

```
!*****
*****
BKCOLOR = 3;           !set screen background color to light blue
!RUNTIME;             !eliminate rules and facts windows
EXECUTE;              !eliminate pressing "go" to run the consultation
ENDOFF;               !eliminate need to press END after a choice
```

SETTINGS

```
!*****
*****
```

ACTIONS

### ACTIONS

```
!***** OPENING WINDOW
```

```
    WOPEN 1,1,10,20,60,7           !define opening window 1
    ACTIVE 1                       !activate window 1
    DISPLAY "
```

AnaEx

Anaerobic Bacterial Identification Expert System

This consultation assists in identification of anaerobic bacteria, which has been isolated in the laboratory.

It asks a series of questions about:

- growth media
- gram stain results
- biochemical test results

It suggests the most likely identification of the microorganism.

Please press any key to begin the consultation.~

"

```
WCLOSE 1
```

```
!remove window 1
```

```

!***** PRINTER WINDOW
!   COLOR = 20           !set following text color to blinking red
!   DISPLAY "   CAUTION!"
!   COLOR = 4           !set following text to non-blinking red
!   DISPLAY "Please make sure your printer in ON."
!   DISPLAY "Press any key to continue.~"
!   CLS                 !clear the screen
!   COLOR = 0           !set following text to normal black

!***** INSTRUCTIONS WINDOW
!   WOPEN 1,1,1,6,77,4   !define instructions window 1
!   ACTIVE 1             !activate window 1
!   DISPLAY "           Instructions
!   Use the arrow keys to move the lightbar to a desired
!   answer choice then press the Enter key."

!***** CONSULTATION WINDOW
!   WOPEN 2,5,1,17,77,3 !define consultation window 2
!   ACTIVE 2             !activate window 2
!   FIND id
!   WCLOSE 1             !remove window 1
!   WCLOSE 2             !remove window 2

!***** RECOMMENDATION WINDOW
!   WOPEN 1,5,13,9,48,5 !define concluding recommendation window 1
!   WOPEN 2,6,14,7,46,7 !define window 2 (nested in window 1)
!   ACTIVE 2             !activate window 2
!   LOCATE 1,9           !specify row and column for next DISPLAY
!   DISPLAY "
!Based on the answers given, the recommended
!identification is: {id}.
!
!(Press any key to conclude the consultation.)~"
!;

      DISPLAY "Welcome to AnaEx - Anarobic Bacterial Identification Expert
System"
      DISPLAY ""
      FIND microcode
      FIND id
      FIND message
!   DISPLAY "Suggested ID: {id}"
;

```

```

!*****
*****
RULE MESSAGE_AEROBE
  IF id = aerobe
  THEN message = aerobe
  DISPLAY "Suggested ID: {id}";

RULE MESSAGE_OTHER
  IF id = other_genus
  THEN message = other_genus
  DISPLAY "Suggested ID: {id}";

RULE MESSAGE_UNKOWN
  IF id = unknown
  THEN message = found1
  DISPLAY "Microcode: {microcode}"
  DISPLAY "Suggested ID: {id}";

RULE MESSAGE_FOUND_1
  IF id <> unknown AND
     prob_overlap = undefined
  THEN message = found_1
  DISPLAY "Microcode: {microcode}"
  DISPLAY "Suggested ID: {id}"
  DISPLAY "Probability: {PRB_PRCNT1}%"
  ;

RULE MESSAGE_FOUND_MORE
  IF id <> unknown AND
     prob_overlap <> ?
  THEN message = found_more
  DISPLAY "Microcode: {microcode}"
  DISPLAY "Probability overlap among {prob_overlap} choices"
  DISPLAY "1. {ID1}          {PRB_PRCNT1}    {PRB_LVL1}"
  {BIO_FREQ1} {CNTRA_TST1}"
  DISPLAY "1. {ID2}          {PRB_PRCNT2}    {PRB_LVL2}"
  {BIO_FREQ2} {CNTRA_TST2}"
  reset message
  ;

RULE MESSAGE_OVERLAP_3
  IF prob_overlap = 3
  THEN message = overlap_3
  DISPLAY "3. {ID3}          {PRB_PRCNT3}    {PRB_LVL3}"
  {BIO_FREQ3} {CNTRA_TST3}"

```

```

        reset message
        ;

RULE MESSAGE_OVERLAP_COMMENT
    IF    prob_overlap > 1
    THEN  message = comment
          DISPLAY "{COMMENT}"
        ;

RULE ID
    IF    microcode <> ?
    THEN  id = unknown
          GET microcode = MICRO_CODE, compend, ALL
          id = (ID1)
          prob_overlap = (PRB_OVRLP)
        ;

RULE NOMICROCODE2
    IF    anaerobe = pos          AND
          id = other_genus
    THEN  microcode = -1
          id = other_genus;

RULE NOMICROCODE
    IF    anaerobe = neg
    THEN  microcode = -1
          id = aerobe;

RULE MICROCODE
    IF    code1 <> ?      AND
          code2 <> ?      AND
          code3 <> ?      AND
          code4 <> ?      AND
          code5 <> ?      AND
          code6 <> ?
    THEN  microcode = (code1 + code2 + code3 + code4 + code5 + code6)
        ;

RULE CODE1
    IF    anaerobe = pos          AND
          URE_VAL <> ?          AND
          BLTS_VAL <> ?          AND
          aARA_VAL <> ?
    THEN  code1 = ((URE_VAL + BLTS_VAL + aARA_VAL) * 100000);

```

RULE CODE2

```
IF   anaerobe = pos      AND
     ONPG_VAL <> ?      AND
     aGLU_VAL <> ?      AND
     BGLU_VAL <> ?
THEN  code2 = ( (ONPG_VAL + aGLU_VAL + BGLU_VAL) *
10000);
```

RULE CODE3

```
IF   anaerobe = pos      AND
     aGAL_VAL <> ?      AND
     aFUC_VAL <> ?      AND
     NAG_VAL <> ?
THEN  code3 = ( (aGAL_VAL + aFUC_VAL + NAG_VAL) * 1000);
```

RULE CODE4

```
IF   anaerobe = pos      AND
     PO4_VAL <> ?      AND
     LGY_VAL <> ?      AND
     GLY_VAL <> ?
THEN  code4 = ( (PO4_VAL + LGY_VAL + GLY_VAL) * 100);
```

RULE CODE5

```
IF   anaerobe = pos      AND
     PRO_VAL <> ?      AND
     PAL_VAL <> ?      AND
     ARG_VAL <> ?
THEN  code5 = ( (PRO_VAL + PAL_VAL + ARG_VAL) * 10);
```

RULE CODE6

```
IF   anaerobe = pos      AND
     SER_VAL <> ?      AND
     PYR_VAL <> ?      AND
     IND_VAL <> ?
THEN  code6 = ( (SER_VAL + PYR_VAL + IND_VAL) * 1);
```

RULE 1

```
IF   ana_growth = neg
THEN  id = aerobe
BECAUSE "No growth on anaerobic media";
```

RULE 2

```
IF   ana_growth = pos      AND
     aerobic_growth = pos
THEN  id = aerobe
```

BECAUSE "Growth on both aerobic AND anaerobic media";

RULE 3

```
IF  ana_growth = pos      AND
    aerobic_growth = neg  AND
    gram_stain = other
THEN id = other_genus
     anaerobe = pos;
```

RULE 4

```
IF  ana_growth = pos      AND
    aerobic_growth = neg  AND
    gram_stain = GNR
THEN anaerobe = pos
ELSE anaerobe = neg;
```

RULE 5

```
IF  CDC_ANA = growth  OR
    CDC_ANA_KV = growth
THEN ana_growth = pos;
```

RULE 6

```
IF  CDC_ANA = no_growth AND
    CDC_ANA_KV = no_growth
THEN ana_growth = neg;
```

RULE 7

```
IF  SBA = growth  OR
    CHOC = growth
THEN aerobic_growth = pos;
```

RULE 8

```
IF  SBA = no_growth  AND
    CHOC = no_growth
THEN aerobic_growth = neg;
```

RULE URE\_VAL

```
IF  URE = POS
THEN URE_VAL = 1
ELSE URE_VAL = 0;
```

RULE BLTS\_VAL

```
IF  BLTS = POS
THEN BLTS_VAL = 2
ELSE BLTS_VAL = 0;
```

```
RULE aARA_VAL
  IF    aARA = POS
  THEN  aARA_VAL = 4
  ELSE  aARA_VAL = 0;
```

```
RULE ONPG_VAL
  IF    ONPG = POS
  THEN  ONPG_VAL = 1
  ELSE  ONPG_VAL = 0;
```

```
RULE aGLU_VAL
  IF    aGLU = POS
  THEN  aGLU_VAL = 2
  ELSE  aGLU_VAL = 0;
```

```
RULE BGLU_VAL
  IF    BGLU = POS
  THEN  BGLU_VAL = 4
  ELSE  BGLU_VAL = 0;
```

```
RULE aGAL_VAL
  IF    aGAL = POS
  THEN  aGAL_VAL = 1
  ELSE  aGAL_VAL = 0;
```

```
RULE aFUC_VAL
  IF    aFUC = POS
  THEN  aFUC_VAL = 2
  ELSE  aFUC_VAL = 0;
```

```
RULE NAG_VAL
  IF    NAG = POS
  THEN  NAG_VAL = 4
  ELSE  NAG_VAL = 0;
```

```
RULE PO4_VAL
  IF    PO4 = POS
  THEN  PO4_VAL = 1
  ELSE  PO4_VAL = 0;
```

```
RULE LGY_VAL
  IF    LGY = POS
  THEN  LGY_VAL = 2
  ELSE  LGY_VAL = 0;
```

```
RULE GLY_VAL
  IF GLY = POS
  THEN GLY_VAL = 4
  ELSE GLY_VAL = 0;
```

```
RULE PRO_VAL
  IF PRO = POS
  THEN PRO_VAL = 1
  ELSE PRO_VAL = 0;
```

```
RULE PAL_VAL
  IF PAL = POS
  THEN PAL_VAL = 2
  ELSE PAL_VAL = 0;
```

```
RULE ARG_VAL
  IF ARG = POS
  THEN ARG_VAL = 4
  ELSE ARG_VAL = 0;
```

```
RULE SER_VAL
  IF SER = POS
  THEN SER_VAL = 1
  ELSE SER_VAL = 0;
```

```
RULE PYR_VAL
  IF PYR = POS
  THEN PYR_VAL = 2
  ELSE PYR_VAL = 0;
```

```
RULE IND_VAL
  IF IND = POS
  THEN IND_VAL = 4
  ELSE IND_VAL = 0;
```

```
!*****
*****
```

## QUESTIONS

```
ASK gram_stain: "What is the result of the Gram Stain?";
CHOICES gram_stain: GNR,other;
```

```
ASK CDC_ANA: "What are the results of the CDC ANA culture?";
CHOICES CDC_ANA: growth, no_growth;
```

ASK CDC\_ANA\_KV: "What are the results of the CDC ANA KV culture?";  
CHOICES CDC\_ANA\_KV: growth, no\_growth;

ASK SBA: "What are the results of the SBA culture?";  
CHOICES SBA: growth, no\_growth;

ASK CHOC: "What are the results of the CHOC culture?";  
CHOICES CHOC: growth, no\_growth;

ASK URE: "What is the result of URE?";  
CHOICES URE: neg,pos;

ASK BLTS: "What is the result of BLTS?";  
CHOICES BLTS: neg,pos;

ASK aARA: "What is the result of aARA?";  
CHOICES aARA: neg,pos;

ASK ONPG: "What is the result of ONPG?";  
CHOICES ONPG: neg,pos;

ASK aGLU: "What is the result of aGLU?";  
CHOICES aGLU: neg,pos;

ASK BGLU: "What is the result of BGLU?";  
CHOICES BGLU: neg,pos;

ASK aGAL: "What is the result of aGAL?";  
CHOICES aGAL: neg,pos;

ASK aFUC: "What is the result of aFUC?";  
CHOICES aFUC: neg,pos;

ASK NAG: "What is the result of NAG?";  
CHOICES NAG: neg,pos;

ASK PO4: "What is the result of PO4?";  
CHOICES PO4: neg,pos;

ASK LGY: "What is the result of LGY?";  
CHOICES LGY: neg,pos;

ASK GLY: "What is the result of GLY?";  
CHOICES GLY: neg,pos;

ASK PRO:	"What is the result of PRO?";
CHOICES PRO:	neg,pos;
ASK PAL:	"What is the result of PAL?";
CHOICES PAL:	neg,pos;
ASK ARG:	"What is the result of ARG?";
CHOICES ARG:	neg,pos;
ASK SER:	"What is the result of SER?";
CHOICES SER:	neg,pos;
ASK PYR:	"What is the result of PYR?";
CHOICES PYR:	neg,pos;
ASK IND:	"What is the result of IND?";
CHOICES IND:	neg,pos;