47p.
diagrams
illus.

21956

UNIDO Project No. XA/RAF/96/646

Investment, Technology and Information Programme for Selected English-speaking African Countries.

Component 4: Industrial Information and Networking

Authors:	Paul J. Rimmer.	Managing Director
	Roy Tynan.	Technical Director.

Organisation:	The Pixel Factory Limited
Date:	17th September 1997
Pixel Factory Document Reference:	UNIDO/97/03-FR

Table of Contents

Table of Contents

IRMS^{plus} System Overview

The IRMS^{plus} system is a World Wide Web enabled database system designed for efficient information storage and retrieval. IRMS^{plus} allows *controlled* database access both to UNIDO designated users and to anybody who has access to the World Wide Web.

The different categories of users suggested are;

• Administrator -	Has full access to all aspects of the system and is responsible for maintenance and upgrade etc.
• Manager -	Has full access to the information stored in the database and can designate access levels for other users.
• Field Operator -	Data input for their own country, plus data retrieval, searching, reporting etc.
• Public User -	Data retrieval, searching, reporting etc.

The main components of IRMS^{plus} are the user interface, the database management system, the web server and the network operating system.

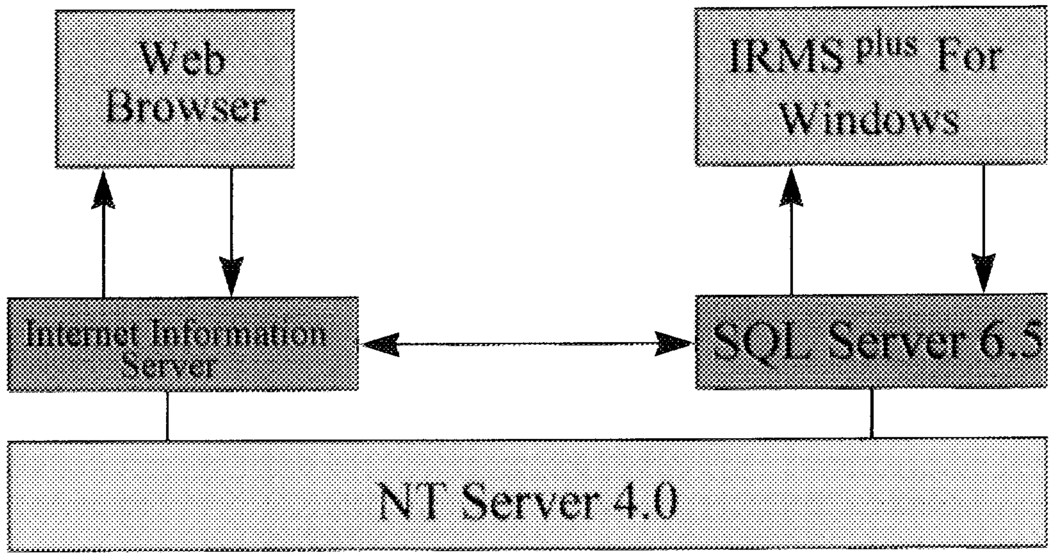


Figure 1 IRMS^{plus} system architecture

Figure 1 shows the main components of IRMS^{plus}, the system is designed to make use of several Microsoft products which have been designed to provide an integrated system running under the Windows NT Server network operating system.

IRMS^{plus} has two user interfaces, these are;

The IRMS^{plus} for Windows application, this is a custom designed piece of software which allows users to perform detailed data analysis and reporting routines on information stored in the IRMS^{plus} database.

Web Browser, this will allow users to connect to the IRMS^{plus} database via the Internet, thus Field Operatives will be able to enter their country specific data directly into the database and other Internet users will be able to view/search the database.

Section One: Core Technologies

1. The Microsoft Integrated Server Suite

The IRMS^{plus} system utilises a suite of Microsoft products as its enabling technologies. These products are;

• Windows NT Server 4.0	*Network operating system.*
• SQL Server 6.5	*Database management system.*
• Internet Information Server 3.0	*Web server.*

Microsoft NT Server

Due to the network operating system being a crucial part of IRMS^{plus} a detailed analysis follows.

1.1 System Overview.

NT Server is Microsoft's top-of-the-line, corporate-oriented operating system. Targeting the needs and demands of corporate MIS departments and hoping to win the race for a modern, stable, powerful network operating system. Microsoft spent at least four years developing a robust and feature-rich operating system intended to meet the rigorous demands of enterprise-wide corporate settings. Windows NT has now been through several versions, growing more powerful and useful for both servers and workstations as the software has matured and CPUs have become more powerful. With Windows NT 4, Microsoft offers a network operating system that they believe outclasses many other more mature systems.

Windows NT is a multithreaded, pre-emptive multitasking operating system with full 32-bit memory addressing. It supports DOS, Windows, Win32 GUI and character-based applications, POSIX-compliant and character-based OS/2 1.x applications, and it includes integrated networking, security, and administration tools. It can run on single or multiple CPUs without modification, and it supports Intel processors as well as DEC Alpha, MIPS, and Power PC chips.

A key feature of Windows NT is that it was designed as the future operating system from Microsoft. Two other important features are NT's internal client/server model, in which the internals of the operating system are divided into these two categories (client and server), and dynamic disk-caching, which uses disk space across multiple drives as available, noticeably enhancing performance.

1.2 Pre-emptive Multitasking

In systems that can run multiple programs at once, some means of keeping all the tasks running is needed.

Windows 3.1's method of multitasking was called co-operative multitasking. By contrast, Windows NT's pre-emptive multitasking scheduler pre-empts one program in favour of another, democratically allotting CPU cycles or even stealing them when necessary.

One result of pre-emptive multitasking is that the user no longer needs to consider the resources a certain task will consume, nor do they have to allow one task to finish processing before starting another.

1.3 Multithreading

All tasks performed by NT can be classified as processes. During any given period, NT is executing a wide variety of processes. These might include checking a user's password, keeping track of the system's clock, accessing data from a disk, doing a memory fetch from RAM, or doing a mathematical calculation.
In NT, processes can be broken down even further into threads. In fact, scheduling events in NT is actually based on the thread unit, not the process. Most processes contain only a single thread, but they can consist of multiple threads if the programmer chooses.

1.4 Implementation of Technology in NT Server

NT stands for New Technology, but what it offers the user isn't really all that new. Most of NT's features existed before its release and were available in other forms. NT 4 looks and behaves much like Windows 95, it has networking not unlike the best Novell has to offer, and internally it's much like UNIX.

The following additional key features make Windows NT an attractive operating system for new implementations:

- NT is written from the ground up as an operating system; it's not just a GUI laid on top of DOS.

- NT has the look and behaviour of Windows 95, so Windows 95 users don't have to be retrained to use NT.

- Scalable architecture means that NT can run on different types of computers--from single CPUs (both Intel x86 and RISC chips) to multiple processor-based systems (sometimes called symmetric multiprocessor systems).

- NT offers high reliability. Unlike DOS and DOS/Windows, NT incorporates a robust micro-kernel design that prevents a single misbehaving application from pulling down the whole system.

- Application compatibility means that NT can run a mix of any of these classes of applications: DOS, Windows 3.x, POSIX-compliant, MS OS/2 1.x character-based programs, and new 32-bit NT applications

- Complete support for Distributed Component Object Model (DCOM) and ActiveX is supported for data sharing between applications--even over the network.

- Because of file system compatibility, Windows NT can work with four types of file systems: FAT, HPFS, NTFS, and Windows 95's VFAT. FAT (DOS's file scheme) is widely used, HPFS is used by OS/2, and NTFS is the proprietary Windows NT file system.

- NT also provides file system enhancements. Aside from being able to convert FAT and HPFS partitions to NTFS, NT's file system offers advanced security features, supports long filenames (up to 256 characters), and provides automatic error correction if a bad sector is detected.

- NT has a built-in networking solution. NT supports industry-standard network protocols, with built-in drivers for NetBEUI, IPX/SPX, TCP/IP, and other transports..

- As for security, NT provides U.S. government C2-level security features. For most purposes, NT has essentially bullet-proof security that can prevent an unauthorised user from entering the system or otherwise gaining access to files on the hard disk. NTFS partitions can't be reached without a proper password, so files are protected..

1.5 NT compared to UNIX

Contrary to popular opinion, UNIX is no longer the inscrutable operating system that only engineers can use. In fact, much of Windows NT has its origins in a variation of UNIX called Mach, developed at Carnegie-Mellon University. In the past several years, UNIX has become standardised among its many vendors, giving it an even stronger foothold in the marketplace

NT and UNIX have much in common, which is good news. The multiprocessing, multitasking, and networking capabilities of NT's Object Manager and Process Manager owe much to UNIX's own time-tested architecture (as well as to VMS design concepts, as mentioned earlier).

1.5.1 Devices

A major similarity between NT and UNIX is in how they interact with devices attached to the computer. To design an operating system that's as flexible as possible, UNIX and NT both connect to the world through device drivers that appear as files. When either UNIX or NT wants to send data to or fetch data from a screen, keyboard, I/O port, memory, or disk file, the same internal approach is used: They simply route the process to a device that appears to the operating system as a sequential file

1.5.2 Memory Allocation

NT and UNIX use memory in similar ways, too. Both can access a large, "flat" memory space of many megabytes. NT and UNIX programmers don't have to deal with any of the problems inherent in a 16-bit PC application's need for segmented addressing schemes. NT typically offers a memory space of 4G per application. Actually, only 2G of that space is available for the application; the rest is for NT's use.

1.5.3 Other Points.

It is true that UNIX has some advanced features, such as distributed file services and parallel processing, that NT does not. Microsoft is in only the infant stages of developing support for NT. On the downside, however, UNIX is a behemoth, and it's not fully standardised.

1.6 NT Server System Architecture

The NT System Model

To fully understand NT's design, it is important to have an understanding of the basic model NT is based on. An operating system's functioning is very complex, requiring code built in many layers. Without a basic theoretical model to build on, an operating system's writers could easily become mired in the details of their own code. Decisions at critical points are harder to make without the model's guidance. Over the years, operating systems engineers have developed some basic theories (and resulting models) used for the design new operating systems; the three most common models are:

- Monolithic
- Layered
- Client/server

1.6.1 User and Kernel Modes

All three models have one thing in common: They each divide operating system tasks into at least two categories--user mode and kernel mode. The kernel is the innermost (core) part of the operating system. Code that runs in kernel mode has access to system hardware and system data. To protect the operating system and stored data (such as files), only certain code is allowed to run in kernel mode. All other code, such as that for applications, runs in user mode.

In most operating systems, for example, applications run in user mode, so they don't have direct access to system resources, such as the hard disk. The user-mode application must ask the kernel to access the hard disk. The application isn't permitted to write directly to the disk, because a mistake in the writing process could scramble other unrelated data files--including the operating system itself.

1.6.2 The Client/Server Model

NT's design draws heavily from the client/server model, treating applications as clients because they ask for services, such as having the operating system put a window on the screen or having data sent to a printer or written to a disk.

1.6.3 Environment Subsystems

NT uses environment subsystems, which intercept binary code requests from a CPU or operating system and translate them into instructions NT can successfully execute. An environment subsystem is really just a program, called a virtual machine, that makes an application act as though it's running on its own machine (or at least is in the environment it was written for).

When a program is executed in NT, the following happens:

1. NT tries to determine what type of environment the program is designed for. If the program type isn't recognised, an error message is generated, and nothing happens.

2. If NT does recognise the program type, it calls up the necessary environment subsystem.

3. NT loads the program into the environment and executes it.

As the program is running, the environment subsystem is busy translating different program types, including actual CPU instructions. However, because many applications that run on NT are actually written for and can execute on Intel-compatible processors, most of the translations are for API calls.

1.6.4 Security

In addition to the environment subsystems, there's another type of subsystem: an integral subsystem. NT has several integral subsystems, but the main one is the security subsystem. For example, a user who hasn't been granted rights might try to access the system. The security subsystem prevents this by requiring a password for each user. Also, each user can have a set of specific privileges controlling his or her level of access to the system.

1.6.5 The NT Executive

The Executive is the heart of NT's architecture, and it includes everything except the protected-mode subsystems and the actual hardware.

The operating system code can run in two modes, user (unprivileged) mode and kernel (privileged) mode. All the subsystems (such as the DOS, POSIX, and OS/2 environment subsystems) run in user mode. By contrast, the NT Executive runs in kernel mode, which means it has access to all of NT's critical internal data structures and procedures.

1.6.6 The System Services Module

When a subsystem asks to have a service performed (such as getting memory allocated for a program just launched), it must ask the System Services module to handle it. The System Services module then sends the message to the Virtual Memory Manager (VMM). The same procedure would take place in other services. When a program is launched, for example, NT must supply at least one thread for it.

1.6.7 Other Modules

These eight modules make up the remainder of the Executive:

- *Object Manager*
  Although it's not truly an object-oriented operating system, NT does use so-called "objects" as the basic operating element for interactions between user mode and kernel mode--for example, when a subsystem needs access to shared resources

- *Virtual Memory Manager*
  Virtual memory is simulated RAM memory. When RAM memory is low, NT uses hard-disk space to simulate what looks like RAM space to applications, but is actually the result of temporarily "swapping" data in RAM to the hard disk to free up RAM for the requested activity

- *Process Manager*
  The Process Manager's job is to create and terminate processes and threads when calling applications need them. If an application wants to create a new process, for example, the request is sent through the active server environment subsystem to the System Services module

- *Local Procedure Call Facility*
  The LPC Facility's job is to supply a communication link between two threads that belong to separate processes. (processes are composed of threads, at least one thread per process.) For example, the printing and editing functions of a word processor could be coded into the application as two separate threads.

- *Security Reference Monitor*
  The Object Manager often works with the Security Reference Monitor to make sure objects aren't accessed (accidentally or intentionally) by unauthorised users. A user can be an actual person (someone trying to access a file or a port, for instance), or it can be a process, thread, or some kind of event

- *I/O Manager*
  One of its features allows most I/O software drivers to be loaded on-the-fly without restarting Windows NT. So while NT is running, you can power up a network card, a new printer, an external CD-ROM drive, etc., then load the driver in NT. When an application asks for an I/O service, such as sending data to a printer, the message first goes through the environment subsystem running the application. It's then passed through System Services to the I/O Manager

- *System Kernel*
  The kernel is the heart of NT. Almost everything that happens in NT passes through the kernel in one way or another. Its main job is to schedule and

dispatch threads and processes, so it continually queues up data, sends it to the CPU(s), and routes it after processing. The kernel also handles interrupts from sources such as the keyboard or other physical devices and is responsible for managing exceptions, error conditions resulting from system violations such as divide-by-zero errors or attempts to write over protected memory areas

- *Hardware Abstraction Layer*
  The Hardware Abstraction Layer, or HAL is the final barrier between the system hardware (including the CPU, memory, I/O ports, keyboard, video, and so forth) and the rest of NT. The only parts of NT that communicate directly with the HAL are the kernel and the I/O drivers.

1.7 Overview of Windows NT Networking

1.7.1 The OSI Model

The OSI model divides a computer network's (either LAN or WAN) processes into seven layers. The bottom (or physical) layer specifies the network's wire and other physical attributes, and the top (or application) layer defines how the network interacts with the user. The model calls for each layer to communicate just with the layers immediately above and below it; this allows each layer to be well defined. Each layer's protocol is responsible for shielding the higher layers from knowing how the layers further down work.

Although the ISO has standardised protocols for each of the seven layers, most local area networks use older protocols that are not fully OSI-compliant.

1.7.2 Windows NTs Network Architecture

The layered architecture that Windows NT uses is modelled on OSI, the most popular of these layered architectures is TCP/IP.

1.7.2.1 TCP/IP

TCP/IP's network layer protocol, IP, uses both network and node address fields, so it can be routed easily. In fact, almost all routers on the market support TCP/IP.

Unlike IPX or NetBEUI, the term TCP/IP doesn't refer to a single protocol, but to a set of protocols (also called a protocol stack) that performs the functions of all seven layers of the OSI model.

The TCP/IP protocol stack is defined in a series of documents called RFCs (requests for comment) that are circulated on the Internet. The following are common TCP/IP protocols:

- IP: Internetwork protocol
- UDP: User datagram protocol
- TCP: Transmission control protocol
- SNMP: Simple network management protocol
- Telnet: A terminal emulation protocol
- FTP: File transfer protocol
- TFTP: Trivial file transfer protocol
- SMTP: Simple mail transfer protocol
- NFS: Network file system

The TCP/IP protocol suite isn't actually OSI-compliant. The Internet protocol suite really defines three layers above the data link and physical layers. TCP/IP is commonly used on Ethernet, Token Ring, FDDI, and serial telephone line networks (called SLIP for serial line IP).

1.7.2.2 Internet Protocol

The Internet protocol (IP) operates as a network layer protocol responsible for routing, addressing, and packet delivery. Like most network layer protocols, IP doesn't handle assured delivery, packet division and sequencing, or error correction. These functions should be handled in higher layer protocols, such as TCP.

1.7.2.3 Transmission Control Protocol

The Transmission Control Protocol, or TCP, is the most common higher layer protocol in the TCP/IP stack. TCP is a connection-oriented, assured-delivery protocol with error checking and packet division and sequencing. TCP assures a transmitter that the data he or she is sending gets to its destination.

2 Microsoft SQL Server

System Overview.

Microsoft SQL Server is a scalable high performance relational database management system. SQL server can manage large amounts of data in a multi-user distributed client/server environment. It offers a high degree of data availability, concurrency and integrity while delivering high performance.

Features

Microsoft SQL Server is based on a parallel database architecture that achieves high levels of performance and scalability.

- *True multithreaded kernel:* Microsoft SQL Server is tightly integrated with native Windows NT threading and scheduling services to provide:
- Higher transaction throughput and performance even for hundreds of concurrent users.
- Greater robustness and reliability because user tasks execute on separate threads and are protected from one another in case of fault.
- *Parallel architecture:* By executing internal database functions in parallel, the performance and scalability of the system is dramatically increased:
- Parallel data scanning allows SQL Server to perform certain queries up to 400% faster than before.
- Asynchronous read-ahead uses predictive algorithms to pre-fetch data into cache for increased performance.
- Parallel load and index operations improve performance and scalability for very large databases.
- *High system capacity:* SQL Server supports up to two gigabytes of memory and eight terabytes of disk storage.

2.1 Core Elements of SQL-Server

There are two main elements within SQL-Server that directly effect IRMS these are:

- *2.1.1 SQL Enterprise Manager*
  This provides powerful administration tools that can be used to manage multiple servers. With SQL Enterprise Manager you can configure, start, pause, and stop SQL Servers, monitor current server activity, and view the SQL Server error log. You can create and manage devices, databases, and database objects. You can manage security, including logins, database users, and permissions. SQL Enterprise Manager and the SQL Executive service provide a way to set alerts for various server events and to schedule server tasks. SQL Enterprise Manager also provides a graphical way to set up and manage replication, and enables you to execute and analyze queries, back up and restore databases, and generate SQL scripts.

- *2.1.2 ISQL/w*
  The ISQL/w utility provides a graphical way to query SQL Server, analyse the execution plan of a query, and view statistics information about an executed query. Multiple queries can be executed simultaneously with ISQL/w.

2.2 Configuring SQL-Server

Databases and transaction logs are stored in files called database devices. Before creating a database, a device in which to store the database must first be created. A device can store many databases, and a database can be stored in several devices.
In addition to database devices, you can create backup devices in which to store a database and its transaction log backups. Only the system administrator can create devices.

2.3 SQL-Server Administration and Security

Before a user can access a SQL Server, the system administrator (SA) must add the user's login ID to the server. The SA can also assign the user a password, a username, a default database, and a default language. The master database is the default database. If the user is not assigned a username in the default database, the user's login ID is used as the username. Only the SA can add SQL Server logins.

If the SQL Server is configured for integrated security, a login ID for each user is not necessary. Instead, use the SQL Security Manager utility to automatically map Windows NT-based usernames to SQL Server login IDs.

2.4 Creating a Database

The following steps are taken to add a database device

1. In the Server Manager window, select the server to add a device to.
2. From the Manage menu, choose Database Devices.
3. Select and name a New Device
4. Allocate the size (in M Bytes) for the device.

It is usual to allocate a database log device as well. This log device will hold a complete backup of all database activity i.e. creation of tables, fields, requests for data from the tables etc. The setup procedure for a log device is almost the same as for the database device.

2.5 Creating Tables within the Database

A table contains database data. When a table is created, you name its columns and supply a data-type for each column. Also usually specified are a primary key, foreign keys, whether the table has an identity column, and UNIQUE or CHECK constraints. SQL Server can have as many as 2 billion tables per database and 250 columns per table. The number of rows and the total size of a table are limited only by the storage space that is available.

2.6 Specifying an Index

- *What is an Index*

An Index is a database object that provides access to data in the rows of a table and is based on key values. Specifically, an index is a set of pointers that are logically ordered by the values of a key. Indexes provide quick access to data and can enforce uniqueness on the rows in a table. SQL Server supports clustered and non-clustered indexes. In a clustered index, data is stored in the same order as the index; in a non-clustered index, data is stored differently from the index.

- *Creating an Index*

1. From the Server Manager a server and the database are selected
2. The table then fields from the database is then selected
3 Under Index Attributes, the appropriate attributes for the index are then selected.

2.7 Triggers

A trigger is a stored procedure that is automatically invoked when a database changes. Triggers are frequently used to maintain the integrity of a database. A trigger executes automatically when a user attempts to modify data on a table specified in the trigger. Only one insert, one update, and one delete trigger can exist per table.

3 Microsoft Internet Information Server

Introduction

The Microsoft Internet Information Server is designed to deliver high speed, secure information publishing while also serving as a platform for developers and independent software vendors to extend the Internet's standard communication capabilities.

A server package needs several components to provide full-scale support for Internet publishing, including transport services, client applications, administrative tools, database and applications connectivity, and encrypted communication.

The Microsoft Internet Information Server is tightly integrated with Windows NT Advanced Server to provide an efficient, reliable, scalable, and secure platform for internal and external Internet administrators. Windows NT Advanced Server provides the administrator with consistent and easy-to-use graphical tools to perform all administrative tasks on Microsoft Internet Information Server and Windows NT Advanced Server.

A benefit of this tight integration is the ability to share applications and interfaces with existing and future Windows NT services, along with tools such as the Internet Service Manager, Control Panel applets, User Manager, Performance Monitor, and Event Viewer.

Along with sharing interfaces and tools, the Microsoft Internet Information Server can also use the services provided by Windows NT Advanced Server. For example, the Microsoft Internet Information Server can use Microsoft SQL Server to log server statistics and use the standard Windows NT Event Log to keep track of security and access information. Another feature is the ability to use the standard Remote Access Service (RAS) transport to provide Internet Information Server resources to remote workstations. RAS provides transparent access to all the features of Windows NT Advanced Server and Microsoft Internet Information Server, including the ability to do administration and application-to-application communication.

3.1 Security

In the context of the Internet, security refers to many different aspects of publishing information. It includes protecting the site itself, storing security information at the site, and transferring data between the server and the client.

Effective security is easy to implement and manage. It is critically important that user accounts and passwords are protected even if an Internet server is compromised by some outside influence. If a site is compromised, a required security measure is to protect user confidentiality by storing security information in an encrypted database on or off the server through an administrative domain structure. Controlling the security context assigned to an anonymous user also allows the administrator tight control over the degree to which a server is exposed when connected to the Internet.

Secure Sockets Layer (SSL) provides a security scheme for bulk-encrypting data between the server and its clients when private communication is required. This type of encryption is provided with the Microsoft Internet Information Server. There are many other encryption features present in the Windows NT security model that are available to the Microsoft Internet Information Server and the Microsoft Internet Explorer Web client. The Microsoft Internet Information Server completely integrates with the object-level and user-level security services provided by the Windows NT Advanced Server security model.

The Microsoft Internet Information Server also uses Windows NT security services for challenge/response authorisation of file access. This provides password authentication between the browser and the server based on Windows NT Advanced Server password authentication procedures and takes place completely in an encrypted channel. In addition, Internet Information Server includes SSL for encrypted communications.

Still another security feature of the Microsoft Internet Information is integrated Basic Authentication. This means that you can use the existing Windows NT Advanced Server security architecture and administration tools to assign permissions to specific users even over the Internet, and query for username and password without requiring new client software. You can also specify exactly which permissions (down to the file level) are granted for anonymous logins.

3.2 Administration

One of the poor points for the Internet servers today is the lack of tools and services to make administration and configuration easier. With the Microsoft Internet Information Server, you get all the services that are part of Windows NT Advanced Server, in seamless integration with existing and future services and applications.

Some of the options and services are:

- *Multiple Server Administration.*
  Monitor and administer many servers all at the same time from a desktop computer.

- *FTP and Gopher Services.*
  Because not everything on the Internet is a Web page, the Microsoft Internet Information Server includes FTP and Gopher services. These services can be run on the same computer or separate computers and are

completely integrated with the Internet Information Server administration (Internet Service Manager) and Windows NT Advanced Server security.

- *Bandwidth Throttling.*
  Meaning that a limit the amount of information that can be sent from the server at any one time can be implemented.

- *Virtual Server Support (or Multi-Homing).*
  This allows a single server to be configured to support as many TCP/IP addresses (within reason).

- *Virtual Directories.*
  Virtual directories allow Webmasters to distribute the physical storage of their published information while providing a single directory structure to external clients.

- *Flexible Logging.*
  Logging capability that allows log files to be automatically rotated based on the size of the log or how long the log has been in use: day, week, or month.

- *Remote Administration.*
  Remotely administering servers is a common feature of all Microsoft BackOffice applications. Providing this remote capability in the Microsoft Internet Information Server is a natural extension not only for internal networks but also for servers on the Internet.

3.3 Development Extensions

The Microsoft Internet Information Server includes a new programming interface called the Internet Server API (ISAPI). This is a programmatic interfaces for extending the capabilities of the Microsoft Internet Information Server. It has library wrappers for developers who use the old-style Windows Sockets interface, and special new functions to extend the Microsoft Win32 API, making it easier for Windows developers to incorporate Internet awareness into their applications.
Another big advancement in Internet server technology is the Internet Database Gateway. This package is included in the Microsoft Internet Information Server. It is a simple, yet powerful gateway for interfacing Web documents with database information. The interface is based on ODBC, so it will work with all major databases including Microsoft SQL-Server.

Section One: Conclusion

The three main technological areas have been discussed

1. Windows NT
2. SQL-Server
3. Internet Information Server

Figure 1 (at the start of this document) diagrammatically shows their relationship with each other.

It is beyond the scope of this document to now detail further the above three technologies, please consult various journals, books and Microsoft technical publications for more information.

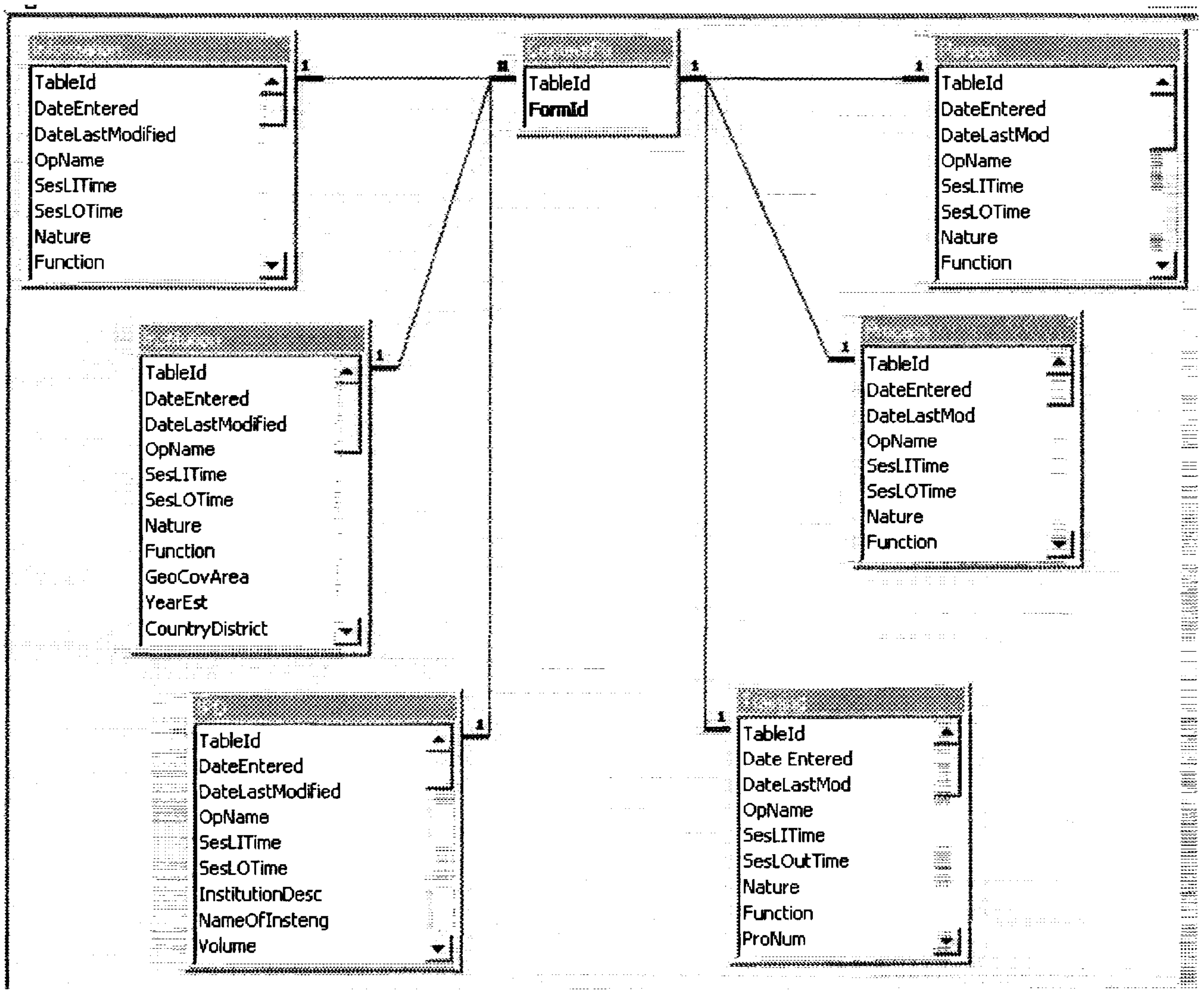
In section two, the actual technologies implemented in IRMS^{plus} are discussed.

Section Two: Technology Implementation in IRMS^{plus}.

1. Database Design

The database structure has been kept as simple as possible, Figure 2.1.1 below outlines the relational structure of the IRMS^{plus} database

Figure 2.1.1: Database Relationship



From Figure 2.1.1: The tables are related via a simple one-to-one relationship from a pivotal table named ContentPtr. This ContentPtr table uses the HTML Form Query String parameters to lookup and reference to the correct table. For example a web page may send a TableId value equal to 1, from the ContentPtr table this value has a reference to the Information table. Another referencing parameter sent from the Web pages is the Country Identifier. Although not shown in the above diagram, the Country Identifier Number and the Table ID constitute the primary search for Inserting information into the database, from the web. The IRMS for Windows application uses the TableID and Country Identifier key elements for its primary searching, but has a much more powerful search engine for interrogating the database, this is discussed later.

Initially the tables within the database were designing using Microsoft Access 97 (A97). From A97 the tables can be exported into SQL-Server via ODBC / DNS. There are limitations on the field types within the tables of the database that can be exported from A97 to SQL-Server, namely the AutoNumber field.

The AutoNumber field of A97 is usually designated a Primary Index, enabling fast searching and uniqueness of that record within a table. However, when an A97 table is exported to SQL-Server the concept of and Auto-Number field is not maintained. For this reason SQL-Server scripting has been used to create the database tables.

Note: The database device and log device have already been created as outlined in Section One- SQL-Server

Figure 2.1.2 below shows the actual SQL-Server script file that was used to create the *Information* table for **IRMS^{plus}**.

Using the SQL/w utility of SQL-Server the script of Figure 2.2 is executed creating the table in the database device. The same process is repeated for all tables, obviously changing the script to the specific table.

Figure 2.1.2: SQL-Server Script File

```
/****** Object:  Table dbo.Information    Script Date: 9/4/97 4:53:08 PM ******/
if exists (select * from sysobjects where id = object_id('dbo.Information') and sysstat & 0xf = 3)
        drop table dbo.Information
GO
/****** Object:  Table dbo.Information    Script Date: 9/4/97 4:53:08 PM ******/
CREATE TABLE dbo.Information (
        TableId int NULL ,
        DateEntered datetime NULL ,
        DateLastModified datetime NULL ,
        OpName varchar (50) NULL ,
        SesLITime datetime NULL ,
        SesLOTime datetime NULL ,
        Nature varchar (20) NULL ,
        Function varchar (20) NULL ,
        YearEst varchar (4) NULL ,
        CountryCode varchar (20) NULL ,
        SysNameEng varchar (40) NULL ,
        OrigNameNotEng varchar (40) NULL ,
        InstResLink varchar (10) NULL ,
        InstResName varchar (40) NULL ,
        TypeofSys varchar (20) NULL ,
        ServicesofSys varchar (30) NULL ,
        Hardware varchar (15) NULL ,
        Software varchar (15) NULL ,
        Indexing varchar (25) NULL ,
        UpdatedLoad varchar (10) NULL ,
        FileSize varchar (10) NULL ,
        Accessibility varchar (10) NULL ,
        PeriodCovered varchar (10) NULL ,
        SysDbLink varchar (20) NULL ,
        SysDbNames varchar (40) NULL ,
        OthePubs varchar (40) NULL ,
        TypeofStorage varchar (20) NULL ,
        Languages varchar (20) NULL ,
        NatureofInfo varchar (20) NULL ,
        GeoCoverage varchar (20) NULL ,
        MediaofDist varchar (20) NULL ,
        Availabilityo_Data varchar (20) NULL ,
        VendorLink varchar (20) NULL ,
        VendorName varchar (40) NULL ,
        Comments varchar (80) NULL ,
        MainDesc varchar (80) NULL ,
        PropDesc varchar (40) NULL ,
        Abstract varchar (80) NULL ,
        UnidoPCodes varchar (15) NULL ,
        UnidoSCodes varchar (15) NULL ,
        DirEmail varchar (25) NULL ,
        ConPerEmail varchar (25) NULL ,
        RecCount int IDENTITY (1, 1) NOT NULL ,
        Country varchar (40) NULL )  set identity_insert information on   GO
```

2. Web Site Architecture and Web Page Design

2.1 Web Site Architecture

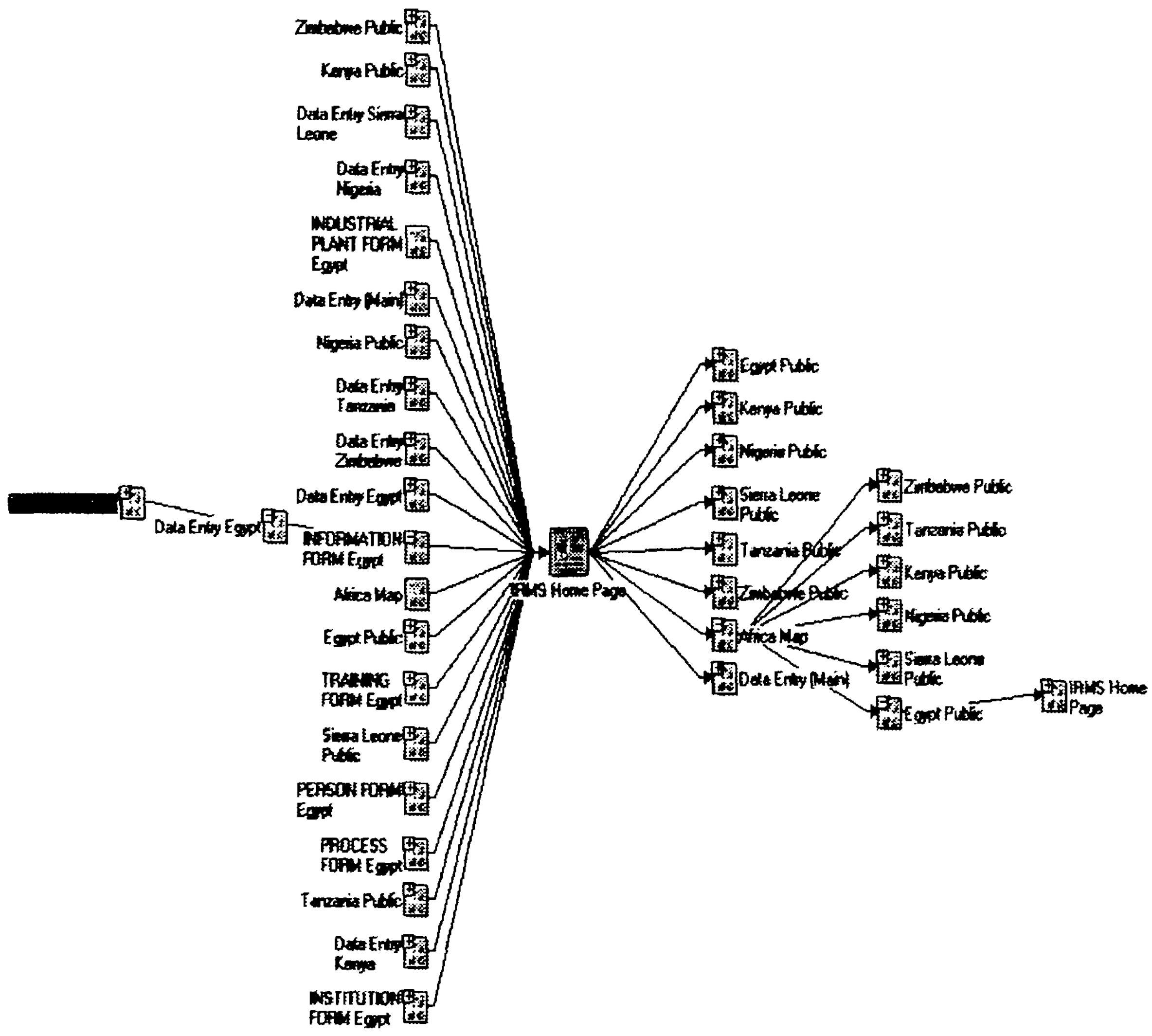
The web site is centralized around the IRMS Home page (default.htm). From this page the web site is essentially split into two areas.

1. Public Access
2. UNIDO Staff Access

Within the two areas of the web site it then splits again, this time into Country specific areas.
Depending up on the access right granted to the user i.e. Administrator, Manager, Field Operator and Public User "sub" areas of the web site are viewable.
A Public User will be able to browse the entire site but will not be allowed to enter any data or view critical information contained within the database. The three remaining groups will have various access rights determined by the UNIDO administration. It is anticipated that for example, a UNIDO Field operator will be able to enter data for their relevant country but not be allowed to delete any information for that country. Managers will be able to access, insert and delete information for a particular country. The Administrator as would be expected has full rights across the entire system.

Figure 2.2.1 shows the hierarchy of the web site, in particular for the Egypt section.

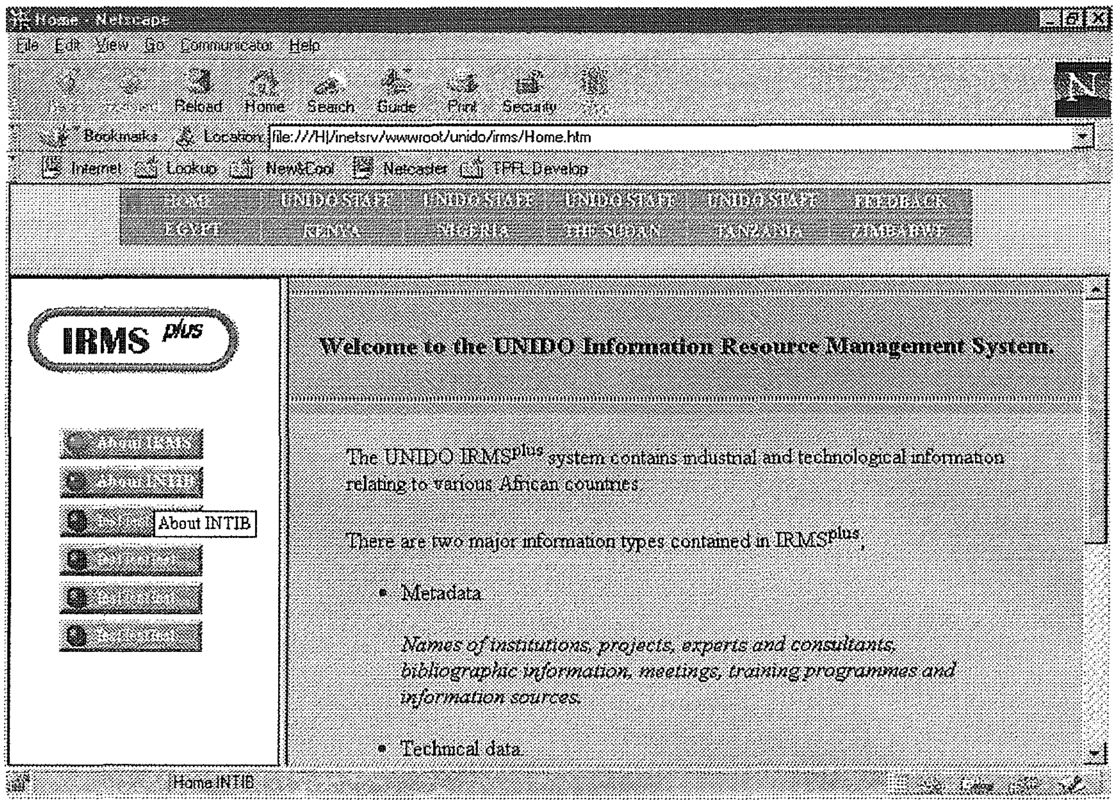
Figure 2.2.1 Web Site Architecture



2.2 Web Site Page Design

The IRMS Web Pages have been designed to implement Frame based navigation. Navigation around the web site is achieved by selecting the main topic from a top frame this creates a second frame on the left hand side. From the left sided frame further navigation takes place. The main content of the navigation is viewed in the centre window. Figure 2.2.2 shows a screen shot of the Netscape browser displaying an initial development web page, the two navigation frames can be clearly seen at the top and left with the content displayed in the centre.

Note: *The screen shot of Figure 2.2.2 was taken during development and does not resemble the final developed page style.*
Figure 2.2.2 Example IRMS Home page and Navigation


Home - Netscape
File Edit View Go Communicator Help
Reload Home Search Guide Print Security
Bookmarks Location: file:///H|/inetsrv/wwwroot/unido/irms/Home.htm
Internet Lookup New&Cool Netcaster TPFL Develop
HOME UNIDO STAFF UNIDO STAFF UNIDO STAFF UNIDO STAFF FEEDBACK
EGYPT KENYA NIGERIA THE SUDAN TANZANIA ZIMBABWE
IRMS plus
About IRMS
About INTIB
About INTIB
Home INTIB

Welcome to the UNIDO Information Resource Management System.

The UNIDO IRMS^{plus} system contains industrial and technological information relating to various African countries

There are two major information types contained in IRMS^{plus},

- Metadata

  Names of institutions, projects, experts and consultants, bibliographic information, meetings, training programmes and information sources.

- Technical data

It can be seen from Figure 2.2.2 that the left hand frame have what appear to be coloured lights in the buttons. These Light Emitting Diodes (LED) simulations are achieved by using JavaScript to animate the button on detecting movement into, over and out off the button by the mouse pointer (cursor). The source code for the HTML and JavaScript to animate the buttons is shown below in Figure 2.2.3

Essentially what the JavaScript does is sense that the mouse has moved into a particular area i.e. the area of the button. An event is triggered that JavaScript recognises, on this event JavaScript loads an image that contains the LED indicating selection e.g. colour Red. When the mouse moves out of a buttons area another different event is triggered, JavaScript then loads the "inactive" LED colour, Blue as in Figure 2.2.2.

Figure 2.2.3

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>

<head>

<title>Home Contents</title>
<base target="main">

<script>
//<!--
//*****************************************************
//
//          Javascript UNIDO Animated Buttons
//
//             BETA VERSION 5th SEPTEMBER 1997
//
//             © The Pixel Factory Limited
//
//******************************************************

// Used  to  track displayed page.
var selected_page=1;

// Necessary to declare these here for browsers with less than JavaScript 1.1.
title=null;
one=null;
two=null;
three=null;
four=null;
five=null;
intro=null;
//      **
function off_image(img) { }
//      **
function on_image(loc, img) {   }

// Loads pages, for browsers with less than JavaScript 1.1.
function letsGo(i)
//      **
// Alter to IRMS parent frame
 {parent.main.location = address[i]; }

// Create and define arrays.
//      **
address = new Array(6);
address[1] = "homei.htm";
address[2] = "home1.htm";
address[3] = "home1.htm";
address[4] = "home1.htm";
address[5] = "home1.htm";
address[6] = "home1.htm";

// -->
</script>
<script language="JavaScript1.1">
<!--

// Preload 3 images to memory.

menuImages = new Array(3)
for(i = 1; i < 4; i++)
   {
        menuImages[i] = new Image(24,24)
       menuImages[i].src = "button"+i+".gif"
   }
//
```

Continued. . .

```
// Preload 6 images to memory.
// titleImages = new Array(6)
//  for(i = 1; i < 7; i++)
//   {
//   titleImages[i] = new Image(50,50)
//   titleImages[i].src = "homec"+i+".gif"
//   }

// Loads pages, for browsers with JavaScript 1.1.
function letsGo(i)
   {
   selected_page=i;

   // Change color of clicked link image to red.
   if (i == 1)
      document.images['one'].src=menuImages[3].src;
   if (i == 2)
      document.images['two'].src=menuImages[3].src;
  if (i == 3)
   document.images['three'].src=menuImages[3].src;
 if (i == 4)
    document.images['four'].src=menuImages[3].src;
 if (i == 5)
    document.images['five'].src=menuImages[3].src;
 if (i == 6)
    document.images['intro'].src=menuImages[3].src;

   // Load selected page.
//      **
   parent.main.location = address[i];

   // Reset all images (except selected link).
   if (selected_page != 1)
      document.images['one'].src=menuImages[1].src;
   if (selected_page != 2)
      document.images['two'].src=menuImages[1].src;
  if (selected_page != 3)
      document.images['three'].src=menuImages[1].src;
   if (selected_page != 4)
     document.images['four'].src=menuImages[1].src;
   if (selected_page != 5)
     document.images['five'].src=menuImages[1].src;
   if (selected_page != 6)
      document.images['intro'].src=menuImages[1].src;
   }

// Called by onMouseOver -  change image to yellow if it's not currently selected.
  function on_image(loc, track)
   {
   if (selected_page != track)
      loc.src=menuImages[2].src;
   }
//   document.images['title'].src=titleImages[track].src;

// Called by onMouseOut - change image back to  white if it's not currently selected.
function off_image(loc, track, img)
   {
   if (selected_page != track)
      loc.src=menuImages[1].src;
      }
//      **
//document.images['title'].src=titleImages[selected_page].src;

// Deactivate Cloaking  -->
</script>
</head>
<body bgcolor="#FFFF80">
<br>
<p align="center"><img src="images/logo4.gif" width="156"
height="50"></p>
<br>
```

Continued. . .

```
<center>
<A HREF="javascript:letsGo(1)"
   onMouseOver="on_image(one, 1) ; self.status='Home Main' ; return true"
   onMouseOut="off_image(one, 1, 1) ; self.status='' ; return true">
   <IMG SRC="button3.gif" NAME="one" WIDTH=24 HEIGHT=24 BORDER=0 HSPACE=0><IMG
SRC="Homec1.gif"    ALT="About IRMSplus" BORDER=0 WIDTH=80 HEIGHT=24 HSPACE=0></a>
<BR>
<A HREF="javascript:letsGo(2)"
   onMouseOver="on_image(two, 2) ; self.status='Home INTIB' ; return true"
   onMouseOut="off_image(two, 2, 2) ; self.status='' ; return true">
   <IMG SRC="button1.gif" NAME="two" WIDTH=24 HEIGHT=24 BORDER=0 HSPACE=0><IMG
SRC="Homec2.gif"    ALT="About INTIB" BORDER=0 WIDTH=80 HEIGHT=24 HSPACE=0></a>
<BR>
<a href="javascript:letsGo(3)"
onmouseover="on_image(three, 3) ; self.status='Home Main' ; return true"
onmouseout="off_image(three, 3, 3) ; self.status='' ; return true"><img
src="button1.gif.gif" border="0" width="24" height="24" name="three"><img
src="test.gif" alt="Test" border="0" width="80" height="24"></a>
<br>
<a href="javascript:letsGo(4)"
onmouseover="on_image(four, 4) ; self.status='Home Main' ; return true"
onmouseout="off_image(four, 4, 4) ; self.status='' ; return true"><img
src="button1.gif.gif" border="0" width="24" height="24" name="four"><img
src="test.gif" alt="Test" border="0" width="80" height="24"></a>
<br>


<a href="javascript:letsGo(5)"
onmouseover="on_image(five, 5) ; self.status='Home Main' ; return true"
onmouseout="off_image(five, 5, 5) ; self.status='' ; return true"><img
src="button1.gif.gif" border="0" width="24" height="24" name="five"><img
src="test.gif" alt="Test" border="0" width="80" height="24"></a>
<br>
<a href="javascript:letsGo(6)"
onmouseover="on_image(intro, 6) ; self.status='Home Main' ; return true"
onmouseout="off_image(intro, 6, 6) ; self.status='' ; return true"><img
src="button1.gif.gif" border="0" width="24" height="24" name="intro"><img
src="test.gif" alt="TEST" border="0" width="80" height="24"></a>

</body>
</html>
```

------------------------------ end of Figure 2.2.3 ------------------------------------

There does appear to be a considerable amount of code just to animate a buttons colour, but the code is duplicated for each button, plus the code example of Figure 2.2.3 does include the complete fully functional navigation URLs as well.

Note: JavaScript is supported in the Netscape browser Version 2 and upwards it is also supported in Microsoft's IE 3.0. If a browser does not support JavaScript then the JavaScript is ignored and the buttons simply do not animate.

So far this section has high-lighted the development of the web site and pages, the next section discusses how information entered by the user is transferred across the Internet to the UNIDO database.

2.2 Hyper Text Markup Language : Forms and URLs

The basic language of a web page is Hyper Text Markup Language (HTML). Although the current version of HTML includes many additional features it still supports the core level of functionality. Additional support for technologies such as ActiveX and Java will not be covered here. The web pages for IRMS^{Plus} as mentioned in the previous section however will support JavaScript

Information requested by a user and data inputted by a user find their way around the Internet to and from destinations by what are referred to as Uniform Resource Locators (URL). A URL is a unique address of another computer with which a communication session is to be established. A full discussion of URL's and the World Wide Web are beyond the scope of this document, what will be described however, is the method of how information entered via the IRMS^{Plus} web site is sent to the UNIDO database and retrieved again. Consider code element shown in Figure 2.2.4 below.

Figure 2.2.4 Form based URL

```
<form METHOD="POST" name="http://www.unido.org/irms/info.asp">

Nature:<input type="text" size="20" name="T1">

Function:<input type="text"size="20" name="T2">

<input type="submit" name="B1" value="Submit Form">
<input type="reset" name="B2" value="Reset Form">

</form>
```

Figure 2.2.4 is explained below

- *<form method="POST" name="http://www.unido.org/irms/info.asp">*
  This informs the browser that the information is to be "posted" to the address (URL) given in the name="......." clause.

- *Nature:<input type="text" size="20" name="T1">*
  Nature is a title that appears on the web page for a text entry field called T1 of size 20 characters in length..

- *Function:<input type="text"size="20" name="T2">*
  Operates the same as Nature above, but with a different name i.e. T2

- *<input type="submit" name="B1" value="Submit Form">*
  This line of code draws a button on the web page and designates as an *"input"* type, meaning that when a user clicks on this button with their mouse the information within the *<form>*...and...*</form>* tags will be sent to the URL via *METHOD POST.*

- *<input type="reset" name="B2" value="Reset Form">*
  This command draws another button on the screen, this button is given the name "Reset Form", it does exactly that, it clears all user entered text from the text fields of the web page in question

Consider two of the lines again:

1. *<form METHOD="POST" name="http://www.unido.org/irms/info.asp">*

and

2. *Nature:<input type="text" size="20" name="T1">*

Line 1 as discussed states that information from this web page is to be sent (METHOD="POST") to a file named *info.asp*. This file is contained under the directory *irms*, this directory in turn comes under the web address (URL) *unido.org*.

Line 2 indicates a text field of size 20 characters, identified to the user as *Nature*. The key to transmission of information is the *name="... " tag*. In this case the name is *T1*.

The UNIDO web server will accept the request for METHOD POST and will search for the file *info.asp* On finding the file the web server "knows" that it can expect to receive two "inputs" to the file as well, one named *T1* and the other *T2*.

The file *info.asp* then processes the inputted names *T1* and *T2*, this is discussed in the next section.

3. Database Connectivity: Active Server Pages

Introduction

Current releases of NT service packs install an extension for IIS namely Active Service Pages (ASP). ASP is reliant on another Microsoft technology, Internet Server Application Programming Interface (ISAPI).. It is ISAPI that is responsible for interpreting and compiling ASP code into a dynamic HTML content generator.

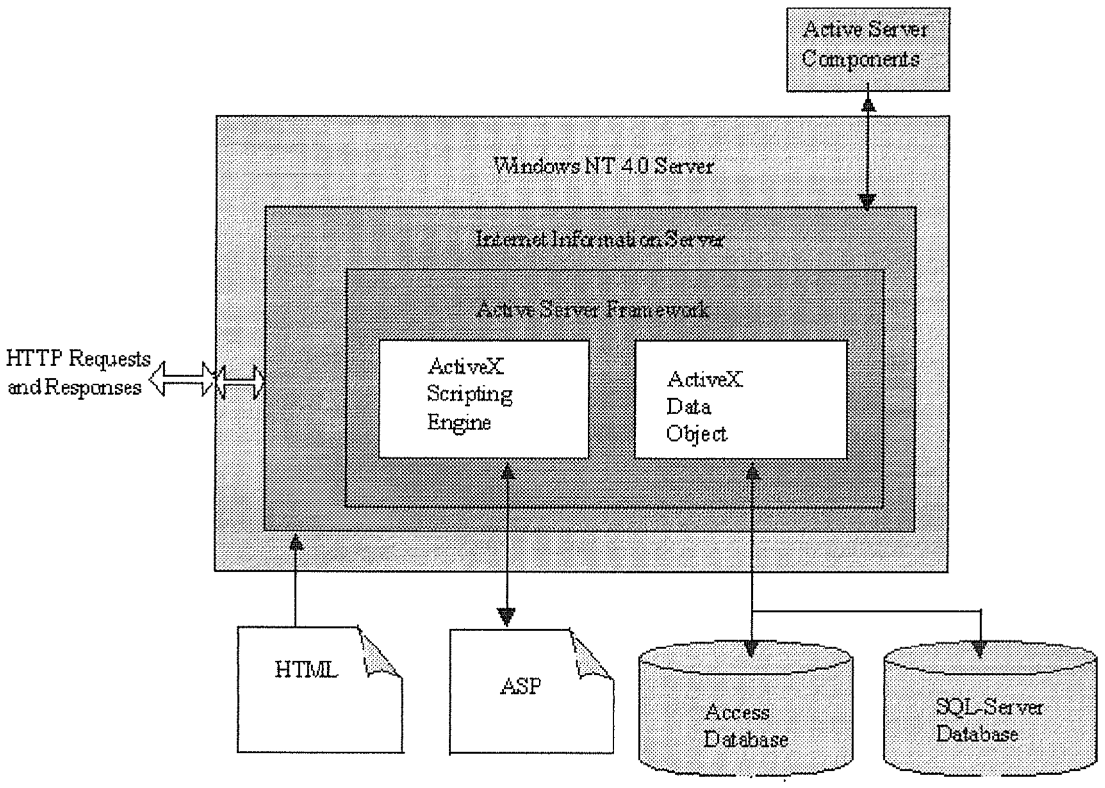
3.1 Dynamic HTML via ASP

With ASP Microsoft have collected the following technologies into one coherent system, ActiveX, Component Object Model and the related Distributed Component Object Model, Open Database Connectivity, OLE-DB and IIS. Probably the greatest strength of ASP is its ability to implement dynamic web database applications using ADO, the ActiveX version of the database object DAO.

3.2 The Active Server Programming Model

This model is extremely complex, Figure 2.3.1 below is a simplified view of the model

Figure 2.3.1 Active Server Programming Model



3.3 Active Server Components

Components are the single most important feature of ASP. Components are the core of all new Microsoft products from Excel through to Visual C++. Components give programmers easier maintenance and promote code reuse. Active Server Components expose operating system services to ASP

3.4 ActiveX Scripting Engine

The ActiveX Scripting Engine is responsible for processing the ASP script, essentially it's an OLE Automation wrapper for ActiveX Scripting language.

3.5 ActiveX Data Object (ADO)

ActiveX Data Objects (ADO) enables a client application to access and manipulate data in a database. ADO's primary benefits are ease of use, high speed, low memory overhead, and a small disk footprint. In ADO, the Recordset object is the main interface to data. An example of the script using the ADO object is shown below.

```
set conn = CreateObject("ADODB.Recordset")
conn.Open "SELECT * FROM
information","DATABASE=unido;UID=sa;PWD=;DSN=unido_dsn"
```

This generates a forward-only, read-only Recordset object. A slightly more functional Recordset can be generated as follows:

```
set conn = CreateObject("ADODB.Recordset")
conn.Open "SELECT * FROM
information","DATABASE=unido;UID=sa;PWD=;DSN=unido_dsn",
        adOpenKeyset, adLockBatchOptimistic
```

This creates a fully scrollable and batch-updatable Recordset.

In the above two examples we are using a DSN on the server called ***unido_dsn***. This DSN points at the *unido* database and then we select all records from the ***information*** table. Note also that we are connecting as the System Administrator, ***sa*** with no password.

3.6 Intrinsic Objects in ASP

Active Server Pages includes a number of built-in server and application building objects. These objects free developers from the tiresome work of writing code to access details about incoming requests from clients, managing the application state, handling cookies, and assembling the response. These intrinsic objects include the following:

- *Request and Response*

The request object provides access to any information passed into the script with the HTTP request. This includes information from cookies, forms, URL queries, and HTTP headers.

- *Application and Session*

These objects are designed to make state management easier, managing a state across a number of users and applications has typically been difficult in Web-based solutions. The session object is used to store information needed for a particular user-session.

- *Server*

The server object allows scripts to create instances of ActiveX components, and thus extend the Active Server Page environment with new capabilities. The server object provides access to methods and properties on the server

- *Base Components in ASP*

To help create web applications, Internet Information Server 3.0 also provides several base components.

3.7 ActiveX Data Objects component

Unlike the Internet Database Connector (IDC) another Microsoft web extension, the ADO component can be "driven" using any ActiveX scripting language from a single *.asp* file.

- *Content Linking component*

The Content Linking component will automatically generate and update tables of contents and navigational links to previous and following Web pages.

- *File System component*

This component provides access to reading in text files stored on the server. By providing file-system access, developers do not need to write their own code to open and close files on the file system.

- *Browser Capabilities component*

Using the Browser Capabilities component, *.asp* files can recognise the capabilities of a requesting browser and dynamically optimise the layout and content. This ensures that the developer does not have to create a series of duplicate pages for each browser.

- *Advertisement Rotator component*

The Advertisement Rotator component allows a list of different advertisements to be assigned relative display-priority percentages. Every time the *.asp* files are requested, the component can be used to display an ad based on the pre-set criteria.

3.8 Application Structure in ASP

An ASP application can be thought of as a complete application defined within the IIS directory structure. Within each site a separate directory structure exists enabling separate ASP applications consisting of an optional *global.asa* and the *.asp* files.

The "global.asa" File

This file provides the Application and Session objects both invoked via their *OnStart* and *OnEnd* events. The scope of the Application object is when a user requests a page from the applications directory for the first time, the *Application_OnStart* event is triggered executing any scripting code within the *OnStart* subroutine. The Application object remains in scope until all sessions have terminated or IIS is restarted, this invokes the *Application_OnEnd* event.
The Session object is invoked when a users browser requests an ASP page from the application directory, Table 2.3.1 highlights the Application and Session event model

Table 2.3.1 Application and Session Model

Application OnStart	The first time users request an *.asp* file from the application directory since IIS last started
Application OnEnd	When all sessions end or time-out
Session OnStart	Invoked by users when their browser requests an ASP page from the application directory. This may occur on the first request or after a previous session with the client browser has been abandoned.
Session OnEnd	Occurs when a session time-out occurs or the Abandon method has been invoked by your ASP code

Three points to bear in mind from Table 2.3.1 are:

1. The time-out value is set in minutes and defaults to 20, this is the minimum value at which it can be set.

2. The Session.Abandon method can be called from your ASP code to terminate a session.

Figure 2.3.2 below shows the skeleton of a *global.asa* file

Figure 2.3.2: Basic *global.asa* file

```
<script language=VBScript RUNAT=server>
Sub Application_OnStart
End Sub
</script>

<script language=VBScript RUNAT=server>
Sub Application_OnEnd
End Sub
</script>

<script language=VBScript RUNAT=server>
Sub Session_OnStart
End Sub
</script>

<script language=VBScript RUNAT=server>
Sub Session_OnEnd
End Sub
</script>
```

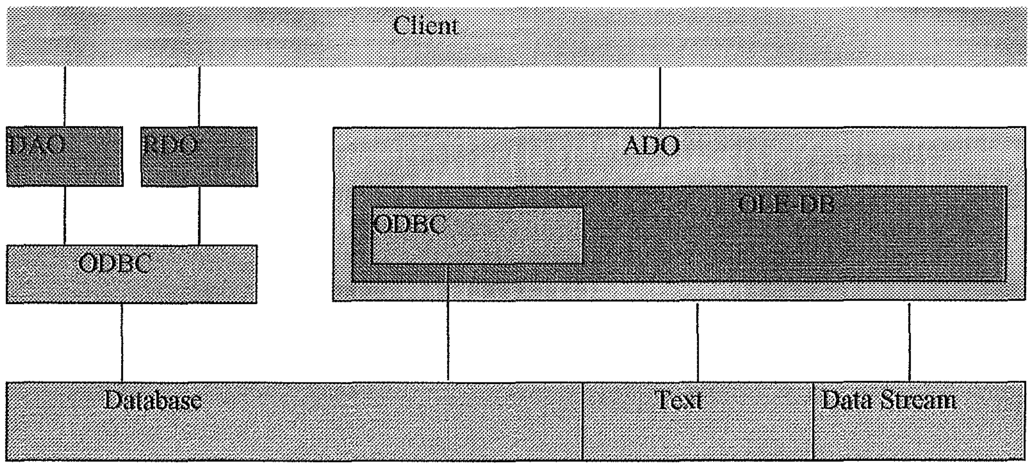
Points to notice from Figure 2.3.3 are:

The use of the scripting tag <script language=VBScript RUNAT=server>. This informs the server that the script to execute is VBScript. The RUNAT clause causes the script to execute on the server and not the client browser.

3.9 ADO and ASP

ADO is built around OLE-DB which is essentially an Object Orientated API built via C++ and provides a wrapper for Open DataBase Connectivity, ODBC.

Figure 2.3.4 ADO (DAO/RDO) Model



Note: ADO and RDO are references to the Visual Basic language are not discussed in this document

3.10 The ADO Object Model

There are three main objects within ADO they are, The Connection Object, The Command Object and The RecordSet Object.

3.10.1 The Connection Object

The Connection Object is responsible for maintaining information about the data provider from which the RecordSet Object will be created. It works in-conjunction with a Data Source Name (DSN) on the NT server.

3.10.2 The Command Object

This object is used to create a minimal cursor driven RecordSet Object but is designed to utilised advanced features such as passing parameters inside a query or even invoking stored procedures.

3.10.3 The RecordSet object

When a RecordSet Object is created, the CreateObject method of the Server Object is instantiate. All the properties of the recordset must be specified. Alternatively, create a recordset implicitly by using a Connection Object or the Command Object.

3.11 ADO Methods

- **CreateObject**: This enables the instantiation of the ADO objects e.g. Connection and RecordSet.
- **Open:** Creates a channel of communication with the server.
- **Requery:** Applied to the RecordSet Object created with the Open method, invokes the query that populated the recordset.
- **Update:** Used with the RecordSet Object, moves data from the data cache to the recordset.

3.12 ADO Properties

- **ActiveConnection:** Informs ADO where the data is and how to access it.
- **CursorType:** Determines how the data provider engine supplies data to the RecordSet Object.
- **LockType:** Resolves concurrency issues with the Open method of a RecordSet object.
- **Name:** The actual name of a Field Object or RecordSet Object.
- **Source:** An SQL text command to fetch data from the data provider.

Figure 2.3.5 shows a complete code example for an ASP application that creates a RecordSet object. This listing enters user details into a database then thanks them for their entry. It uses a DNS named *unido,* which is defined as an ODBC source connection to SQL-Server.

One other point to note is that the data sent to this *.asp* file comes from HTML *Form* based elements i.e. text fields, and are posted (METHOD="post") via a *Submit* button., hence the *Request.Form* method, this can be seen in Figure 2.3.6

Note 1:
A complete listing of all ASP files has not been included in this document due to the volume of paper required.
Note 2:
The example ASP listings in this document are not the actual listings implemented.

Following the next two listings is a short section on Structured Query Language (SQL). This is the actual method that ASP saves and retrieves information.

Figure 2.3.5: A Simple ASP Database Application, *detail.asp*

```
<%
'~~~~ open the database connection
set conn = Server.CreateObject("ADODB.Connection")
'~~~~ create a RecordSet
set rs = Server.CreateObject("ADODB.Recordset")
conn.open("DRIVER={SQL Server};SERVER=UNIDO;UID=sa;PWD=;DATABASE=unido")
'create the SQL string
sqlStr = "insert into information (nat,func,ye,cc,sne,on,ir,tos) "
        sqlStr = sqlStr & "values ('"& Request.Form("Nature") &"',"
        sqlStr = sqlStr & " '"& Request.Form("Function") &"',"
        sqlStr = sqlStr & " '"& Request.Form("YearEst") &"', "
        sqlStr = sqlStr & " '"& Request.Form("CountryCode") &"', "
        sqlStr = sqlStr & " '"& Request.Form("SysNameEng") &"', "
        sqlStr = sqlStr & " '"& Request.Form("OrigName") &"', "
        sqlStr = sqlStr & " '"& Request.Form("InstRes") &"', "
        sqlStr = sqlStr & " '"& Request.Form("TypeOfSys") &"')"
        set rs = conn.execute(sqlStr)
        conn.close

%>
<html>

<head>
<title> Reply Page</title>
</head>

<body bgcolor="ffffffff">

<center>
        Thank you for your details.<br>
</center>

</body>
</html>
```

Figure 2.3.6: Details.html

```
<html>
<head>
<title>Details </title>
</head>
<body bgcolor="ffffffff">
<center>
        <h1>Information Details</h1>
</center>
<center>
        <hr width="60%">
        <table width="60%">
                <td>
                <font size="+1">
                Please enter your details below.<br><br>
                <form METHOD="post" ACTION="http://unido_server.org/vasp/details.asp">
                        <font size="+0">
                        <strong>Nature<br></strong><input type="text" name="Nature"
                        size=30><br>
                        <strong>Function<br></strong><input type="text" name="Function"
                        size=30><br>
                        <br><hr width="50%"><br>
                        <strong>Year Established<br></strong><input type="text" name="YearEst"
                        size=50><br>
                        <strong>Country Code<br></strong><input type="text"
                        name="CountryCode" size=50><br>
                        <strong>System Name (Eng)<br></strong><input type="text"
                        name="SysNameEng" size=30><br>
                        <strong>Original Name (not Eng)<br></strong><input type="text"
                        name="OrigName" size=30><br>
                        <strong>Institute Responsible<br></strong><input type="text"
                        name="InstRes" size=8><br>
                        <strong>Type of System<br></strong><input type="text"
                        name="TypeOfSys" size=15><br>
                        <br><hr width="50%"><br>
                        <p>
                        <input type="submit" value="Send Details Now">
                        <input type="reset" value="Clear">
                </form>
        </table>
</center>
<br>
</body>
</html>
```

4. Structured Query Language

Fundamental to most if not all database concentric applications is the Structured Query Language (SQL)

Structured Query Language (SQL) is a way to communicate with a relational database that lets programmers and users define, query, modify, and control the data. Using SQL syntax, statements are construct that extract records according to criteria specified. Some of the most common SQL keywords are listed below in Figure 2.4.1.

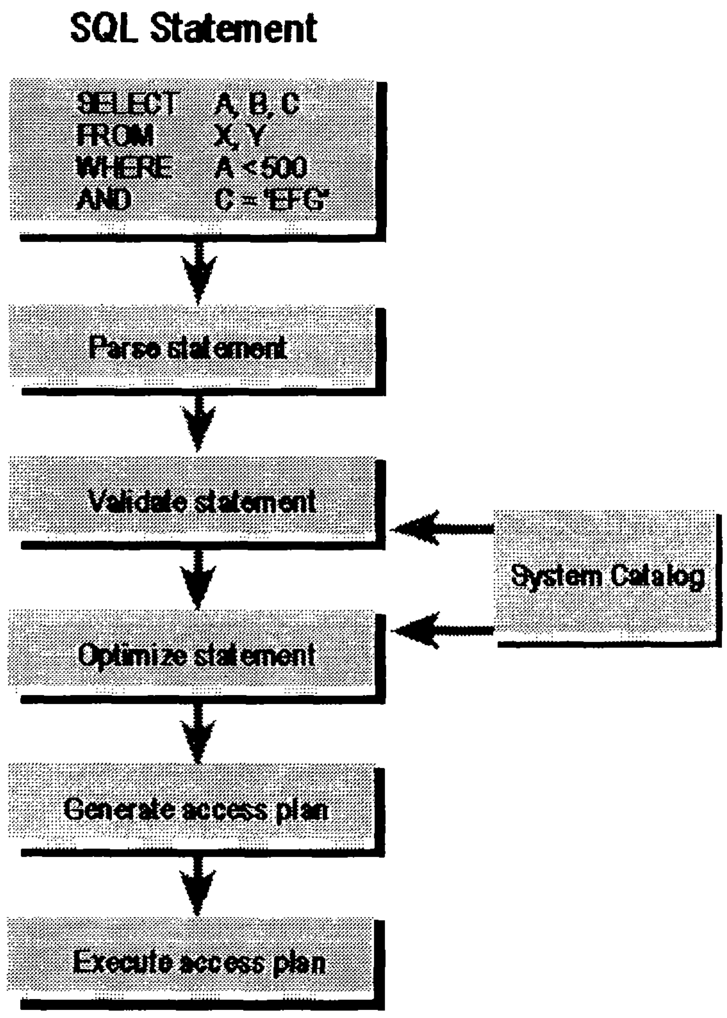
Figure 2.4.1 Common SQL Statements

SQL keyword	**Description**
SELECT	To identify which tables and columns in the data source are to be used
WHERE	To apply a filter which narrows the selection.
ORDER BY	To apply a sort order to the recordset.
INSERT	To add new records to a recordset.
DELETE	To delete records from a recordset.
UPDATE	To modify the fields of a record.

4.1 Processing a SQL Statement

The steps involved in a SQL statement process are common to all three techniques, although each technique performs them at different times. Figure 2.4.2 shows the steps involved in processing a SQL statement.

Figure 2.4.2: SQL Processing



To process a SQL statement, a DBMS performs the following five steps:

1. The DBMS first parses the SQL statement. It breaks the statement up into individual words, called tokens, makes sure that the statement has a valid verb, valid clauses, etc. Syntax errors and misspellings can be detected in this step.

2. The DBMS validates the statement. It checks the statement against the system catalogue. Do all the tables named in the statement exist in the database? Do all of the columns exist and are the column names unambiguous? Does the user have the required privileges to execute the statement? Certain semantic errors can be detected in this step.

3. The DBMS generates an access plan for the statement. The access plan is a binary representation of the steps that are required to carry out the statement; it is the DBMS equivalent of executable code.

4. The DBMS optimises the access plan. It explores various ways to carry out the access plan. Can an index be used to speed a search? Should the DBMS first apply a search condition to Table A and then join it to Table B, or should it begin with the join and use the search condition afterward? Can a sequential search through a table be avoided or reduced to a subset of the table? After exploring the alternatives, the DBMS chooses one of them.

5. The DBMS executes the statement by running the access plan.

The steps used to process a SQL statement vary in the amount of database access they require and the amount of time they take. Parsing a SQL statement does not require access to the database and typically can be done very quickly. Optimisation, on the other hand, is a very CPU-intensive process and requires access to the system catalogue. For a complex, multiple query, the optimiser may explore thousands of different ways of carrying out the same query. However, the cost of executing the query inefficiently is usually so high that the time spent in optimisation is more than regained in increased query execution speed. This is even more significant if the same optimised access plan can be used over and over to perform repetitive queries.

Note1: The actual construct of the SQL statements such as INNER / OUTER JOINS, SELECT INTO and other types of SQL statement have not been included in this document. These elements will be included in the final Windows Help file that will accompany the IRMS for Windows application.

The final section next, discusses the Windows 95 and NT database application, IRMS for Windows.

Note2: At the time of writing this document the final graphical interface of IRMS for Windows was still under development. Screen shots in the next section should be viewed as beta versions only.

5. IRMS for Windows

Introduction

In the previous sections discussion has centred around the Internet aspect of IRMS^{Plus}. Once information has been sent via the Internet to the database UNIDO staff must be able to view, alter, enter and generally analyse the information "off-line" i.e. not connected to the Internet.

The Pixel Factory Limited are designing a Windows 95 and Windows NT compliant database application that will interrogate the UNIDO- IRMS database to provide the features outlined above. In addition the application, IRMS for Windows (IFW) provides other facilities e.g. printing of reports, graphical representation of information and internal E-mailing.

5.1 Technology

Firstly, three languages were considered for IFW development.

1. Microsoft Visual Basic 5.0 Enterprise
2. Microsoft Visual C++ 5.0 Enterprise
3. Borland Delphi 3.0 Client/Server

The Pixel Factory Limited use the above three languages but eventually opted for number 3, Borland's Delphi.

Most modern computer languages have some form of inherent Object Orientation and component modelling capabilities. However, Delphi 3.0 is the more stream-lined of the three. Visual Basic although very popular and well supported is still to slow and has a very limited Object model. Visual C++ is extremely powerful and does have possibly the definitive Object Model for Personal Computer software developers. However, with the emergence of Delphi 3.0 in May 1997 Borland have not only produced a first class Object Orientated language but have also engineered a component reuse and development architecture. This is defined as RAD, Rapid Application Development.

5.2 Client / Server

Utilising Delphi's in-built database objects allows for a simplified connection to a remote database system (across a network) such as SQL-Server. The connection to the SQL-Server database is transparent as far as the user sees it. To a user, the database appears to exist within the Personal Computer. All a user will see is an initial request to logon to the database by supplying a User Name and a Password.

5.3 IRMS for Windows

The SQL-Server logon process is the first interaction with IFW a user will come across. The initial logon screen ("splash screen") will request their User Name and Password. The User Name and Password will be verified via SQL-Server. It is SQL-Server that determines the database access rights of the user. Access rights are setup by the system administrator.

Figure 2.5.1 shows a typical welcome screen , Figure 2.5.2 shows the selection screen

Figure 2.5.1 The Welcome Screen

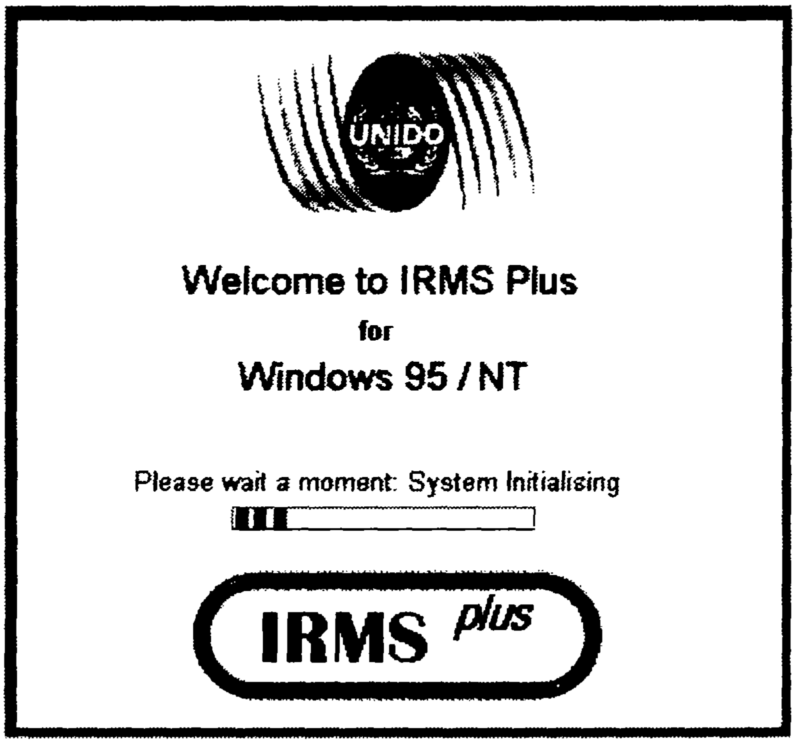
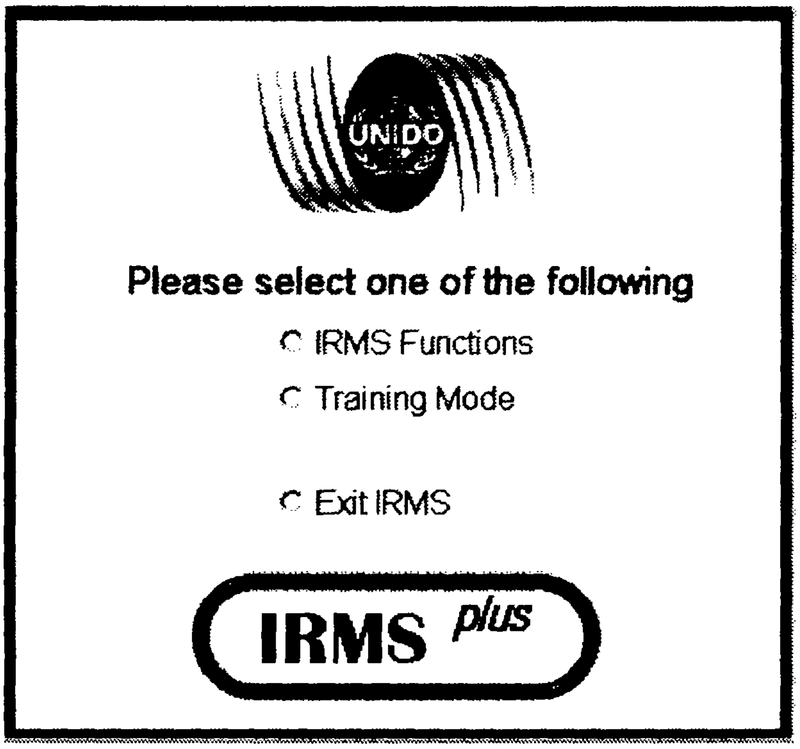


Figure 2.5.2 The Selection Screen

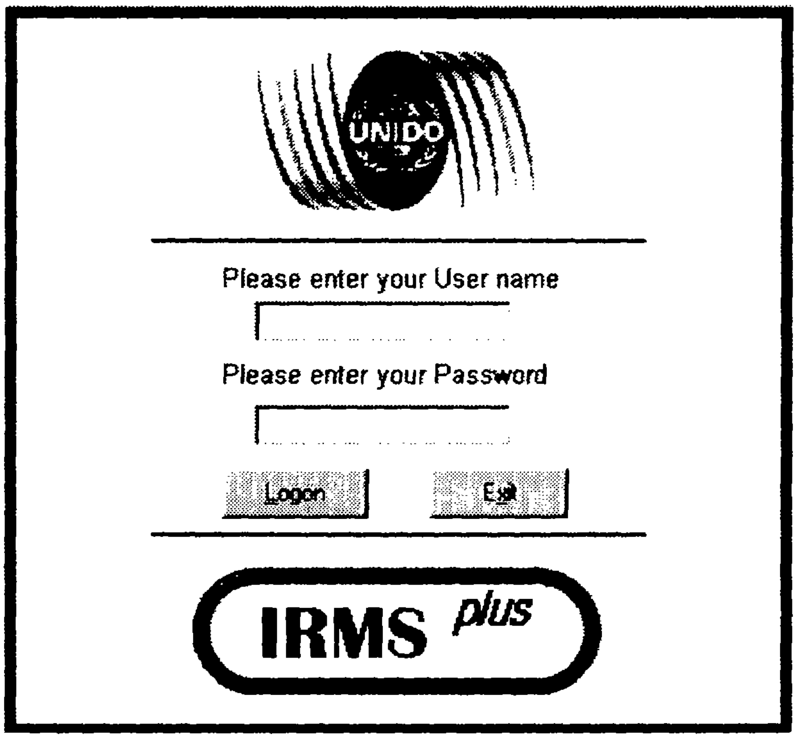

Please select one of the following

IRMS Functions

Training Mode

Exit IRMS

Figure 2.5.3 Logon Screen


Please enter your User name

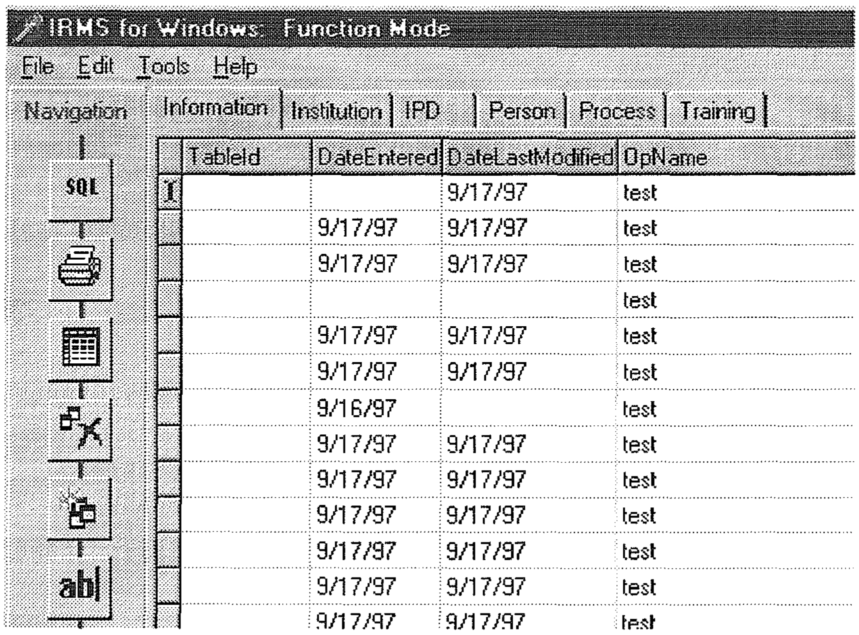
Please enter your Password

Figure 2.5.2 shows the screen where a user selects what appropriate function of IFW. This allows a user to enter into the main IRMS function area or practice in the Training area.

From either selection the main screen of IFW is the same however, in training mode a user will work with the training database not the actual IRMS database.

5.4 The Main Screen

Figure 2.5.4 The Main Screen



TableId	DateEntered	DateLastModified	OpName
		9/17/97	test
	9/17/97	9/17/97	test
	9/17/97	9/17/97	test
			test
	9/17/97	9/17/97	test
	9/17/97	9/17/97	test
	9/16/97		test
	9/17/97	9/17/97	test
	9/17/97	9/17/97	test
	9/17/97	9/17/97	test
	9/17/97	9/17/97	test
	9/17/97	9/17/97	test
	9/17/97	9/17/97	test

Figure 2.5.4 shows the main screen after the Selection screen. Information is presented initially in a "view" only mode with the *Information* database table first interrogated . To select data entry either the New Record button

or the Edit button must be pressed.

The first twenty records will be displayed, the user has the option of then viewing more records if required.

The main menu bar will provide the usual windows functions e.g. File Open/Close, Print, Exit, Copy, Cut and Paste, Tools and Help. On the left hand side is a "speed bar" to enable faster selection of the options from the menus. The speed bar will be dock-able or allowed to float on top of the main window, this will be selected via an options/preferences window not shown here.

There is a tab-bar along the top of the main window viewing area, this allows fast switching between the different database tables at the click of the mouse button.

The screen shot of Figure 2.5.4 shows the database information in what might be termed "view all" mode, another mode namely "single" mode is also provided. A user can single step through the records one at a time

A user will also be able to specify a search for record or records. This might include :

Search on	Date
	Name
	Location
	Number

The search is not limited to the four shown above, indeed the user will be able to select any field within that current database table to search on. This search facility will also allow a user to select a "logical join" type such as AND, OR, NOT e.g

Search for *xxx* where *yyy* AND *zzz* = *nnn*

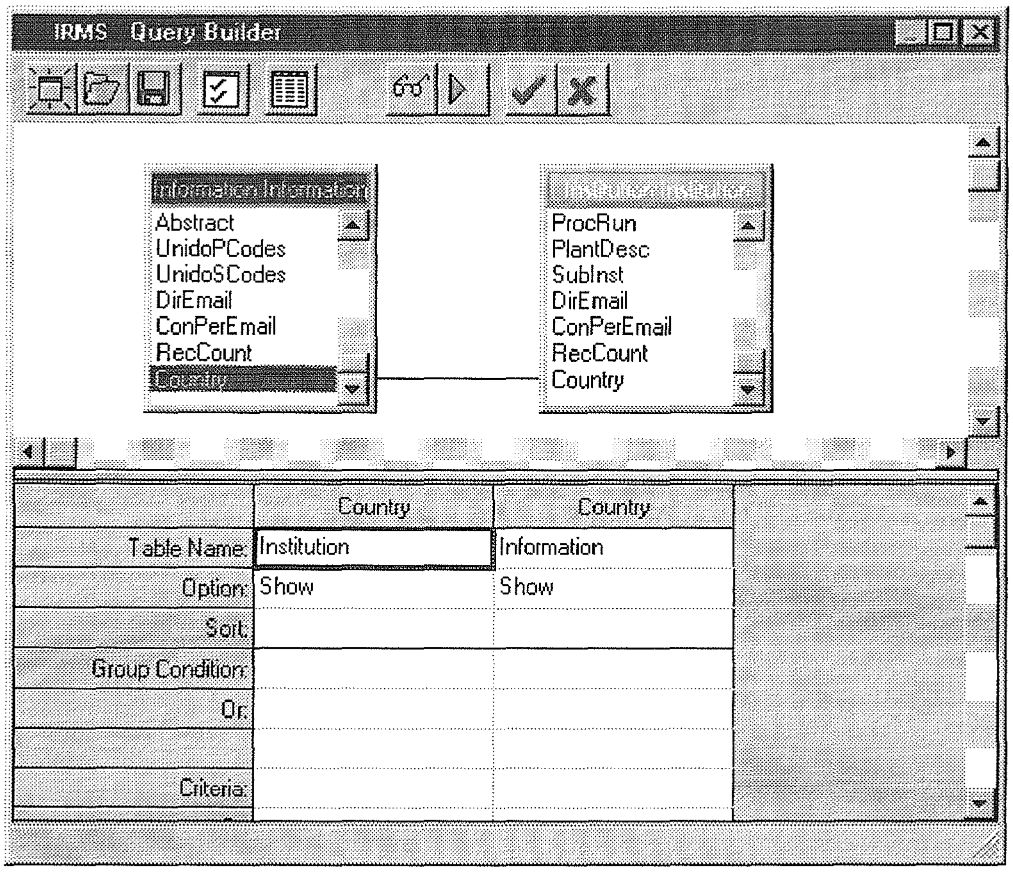
A visual query builder will also be included. This allows a user to graphically construct a database query across more than one table at a time, referred to as Joining. Figure 2.5.5 outlines the initial graphical query builder screen at the time of writing this document.

Figure 2.5.5 shows a simple visual query connecting both the *Information* and *Institution* tables. Executing this query will result in IFW displaying all records from both tables that have a country entered into their respective Country field.

This is a simple example, more complex queries will be allowed.

Note: this is still under development as of the time of writing this document.

Figure 2.5.5 The Visual Query Builder



	Country	Country
Table Name:	Institution	Information
Option:	Show	Show
Sort:		
Group Condition:		
Or:		
Criteria:		

5.5 Email Facility

Mail Application Programming Interface (MAPI). It is an interrelated set of mail services in an application. It exists at the function level, i.e. MAPI is implemented as a dynamic-link library (DLL).

MAPI is often referred to as Simple MAPI. Simple MAPI contains all the functionality needed to implement useful mail-based applications within a typical Microsoft Office environment. The components of MAPI will be included into IFW enabling email interaction with the underlying architecture of Window 95 and NT.

Simple MAPI provides the following general categories of services:

- Logon and Logoff
- Creating, Addressing, and Sending Messages

- Receiving and Reading Messages
- Deleting Messages

Note: At the time of writing this document the MAPI sub-system of IFW has not yet been written.

Conclusions

IRMS^{Plus} represents a major upgrade of the existing IRMS system. Presently IRMS is a paper based system with all the problems associated with such a system. By utilising the Internet coupled with the features of Windows NT and SQL-Server an advanced and widely accessible information system can be deployed. IRMS^{Plus} can disseminate information over the internet to other industrial organisations plus the World Wide Web site can be used to generate revenue for UNIDO through advertising.