ENTERPRISE MANAGEMENT SYSTEM

A DISSERTATION

Submitted in partial fulfilment of the
requirements for the award of the degree
of
MASTER OF COMPUTER APPLICATIONS

By

LOKESH GANDHI

CENTRAL LIBRARY
G11001
Acc. No.........
Date 13.9..02
I.I.T. ROORKEE

INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
श्रमं विना न किमपि साध्यम्

DEPARTMENT OF MATHEMATICS
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE-247 667 (INDIA)

MAY, 2002

CANDIDATE`S DECLARATION

I hereby certify that the work which is being presented in this dissertation entitled "**ENTERPRISE MANAGEMENT SYSTEM**" in partial fulfillment of the requirement for the award of the degree of Master of Computer Applications, submitted in the Department of Mathematics of the Indian Institute of Technology, Roorkee, is an authentic record of my work carried out in the period from Jan 2002 to May 2002,under the supervision and guidance of Dr. Mohan Lal and Mr. Rajagopalan Srinivasan

The matter embodied in this project has not been submitted by me for the award of any other degree.

Lokesh Gandhi

Date: May 29 , 2002
Place :IIT,Roorkee.

(Lokesh Gandhi)

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

(Mr Rajagopalan Srinivasan)
Senior Development Manager
Oracle India Pvt. Ltd.

30/05/2002
(Dr. Mohan Lal)
Asst. Professor,
Computer Center,
I I T Roorkee

H. G. Sharma
7.6.02
Professor & Head
Department of Mathematics
I.I.T. Roorkee-247 667

ORACLE®

ORACLE INDIA PRIVATE LIMITED
India Development Center
Oracle Technology Park
No. 3, Bannerghatta Road
Bangalore - 560 029, India.
Telephone : + 91 (80) 552 8335/40/45
+ 91 (80) 552 8350/55
Fax : + 91 (80) 552 6124

May 21, 2002

TO WHOMSOEVER IT MAY CONCERN

This is to certify that Mr. Lokesh Gandhi student of IIT, Roorkee has completed his project with Oracle India Private Limited.

The project was undertaken for a period of four months from January 15, 2002 to May 31, 2002. He worked on a project 'ENTERPRISE MANAGEMENT SYSTEM'.

We wish him all the best in his future endeavours.

Yours sincerely
for ORACLE INDIA PRIVATE LTD.

Satish Venkatachaliah
Senior Manager - Human Resources

ACKNOWLEDGEMENT

I would like to express my deep sense of gratitude and sincere thanks to my guides **Dr. Mohan Lal (Asst. Prof. , Computer Center)** and **Mr. Rajagopalan Srinivasan(Senior Development Manager, Oracle India Development Center)** for their valuable and ever willing precious guidance, technical support and constant encouragement during the course of this dissertation work under their guidance.

I am grateful to **Mr. Kiran Kulkarni** (Senior Applications Engineer, Oracle India Development Center) and other staff members for providing the necessary facilities for the successful completion of this work.I would like to mention the name of **Prof. R.C Mittal**, Department of Mathematics for their able guidance and kind support.

Lokesh Gandhi

Lokesh Gandhi
MCA 3rd yr
IIT Roorkee

ABSTRACT

From the initial stages, there was a need felt for the use of Computer Systems to automate end to end daily processes of Organizations. These Systems will have unified information architecture, reduce the paper work and redundancies in operations. With the globalization of economies and rapid expansion of corporates across the globe, the information is needed to be accessed instantly at any part, any time in the world. Also there is a consistent requirement to keep all the stored data in sync with each other. Several Function specific applications were made by different vendors for separate modules which attempted to serve this purpose by fulfilling some of the objectives, but there was an urgent need of integrating these applications, so that data is consistent as well as retrievable across all the departments around the globe.

ERP Applications attempts to integrate all departments and functions across a company onto a single computer system that serves all departments' particular needs. It combines them all together into a single, integrated software program that runs off a single database. So, various departments can more easily share information and communicate with each other. Oracle Applications are such kind of integrated applications which are packaged in one EBusiness suite called Oracle 11i.

This report describes building of some of the modules of such an Application suite called ORACLE 11i E Business suite namely the Purchasing and Payables modules, which automates an organization's purchasing, and payables process, is well integrated with other modules of Oracle 11i suite. This application like other Oracle Applications can be accessed through a browser using a three tier architecture and supports multiple languages and multiple platforms.

The Enterprise Management System is built on Oracle Applications platform .The Operating System used are Solaris and Microsoft Windows NT 4. The tools used are Developer 6i, Sql*plus and Oracle Flint. The language used is pl/sql.

Table of Contents

CHAPTER – 1

INTRODUCTION AND STATEMENT OF PROBLEM

1.1. What is ERP?

Enterprise resource planning software, or ERP attempts to integrate all departments and functions across a company onto a single computer system that can serve all those different departments' particular needs. It is like building a single software program that serves the needs of people in finance as well as it does the people in human resources and in the warehouse.

Each of those departments typically has its own computer system optimized for the particular ways that the department does its work. But ERP combines them all together into a single, integrated software program that runs off a single database so that the various departments can more easily share information and communicate with each other. That integrated approach can have a tremendous payback if companies install the software correctly. Take a customer order, for example. Typically, when a customer places an order, that order begins a mostly paper-based journey from in-basket to in-basket around the company, often being keyed and rekeyed into different departments' computer systems along the way. All that lounging around in in-baskets causes delays and lost orders, and all the keying into different computer systems invites errors. Meanwhile, no one in the company truly knows what the status of the order is at any given point because there is no way for the finance department, for example, to get into the warehouse's computer system to see whether the item has been shipped. "You'll have to call the warehouse" is the familiar refrain heard by frustrated customers.

ERP vanquishes the old standalone computer systems in finance, HR, manufacturing and the warehouse, and replaces them with a single unified software program divided into software modules that roughly approximate the old standalone systems. Finance, manufacturing and the warehouse all still get their own software, except now the software is linked together so that someone in finance can look into the warehouse software to see if an order has been shipped. Most vendors' ERP software is flexible enough that an organization can install some modules without buying the whole package. Many companies, for example, will just install an ERP finance or HR module and leave the rest of the functions for another day.

Every application in ERP Suite can run in one global instance of a single database. All the applications work together and share the same information. The result is a complete suite of applications that provides data for information that is displayed in customizable portals, revealing critical information such as sales positions, inventory levels, headcount, revenue, and expenses—across all organizations, lines of business, products, and geographies. And because the business intelligence systems and the data sit in the same system, the user won't wait for data to pass through a separate data aggregation and analysis system. The company's executives get daily business intelligence that reveals the state of the business every day, relative to past, present, and projected performance metrics. This allows the decision makers to reach more informed conclusions, make daily corrections, and drive the business towards achieving its goals for profitability—daily, not quarterly or even monthly. More precisely monitor performance to make more informed business decisions. Performance that does not meet targets can trigger immediate workflow notifications along with other corrective actions. Associated reports explaining the cause and effect of the information are only a link away, encouraging a cycle of continuous process improvement.

1.2 STATEMENT OF PROBLEM

ABC Trading Corporation is a trading firm, which takes orders for kits from various customers and procurès the components required, from different vendors. They also sell individual components in the Spares Market. Due to major expansion in business they are now feeling the need to automate the major activities of the company by implementing ERP. For this they have approached Oracle Corporation with some specs from their IT Implementation Team.

My project is to automate the following areas of the Corporation.

- Purchasing
- Payables

They want their system to be developed on Oracle Applications Platform as they have already implemented Oracle Applications in some parts of their business activities.

1.3 ORGANIZATION PROFILE

ORACLE Corporation, a $11 billion company, head quartered in Redwood Shores, California, is the world's second-largest software company, and the world's largest supplier of software for information management.

ORACLE was founded in 1977 with a vision of finding faster, easier, less expensive, and more powerful ways to manage and access information, and that vision has become a reality. ORACLE built the first commercial relational database system. ORACLE sold the first products employing SQL (structured query language), now the industry standard. ORACLE saw the value of low-cost, client/server systems over proprietary mainframes. ORACLE pioneered portable software that today runs on practically all hardware, from PCs to mainframes. In recent years, ORACLE championed parallel software as the breakthrough that will power very large database applications like data warehousing and information-on-demand. And recently, ORACLE introduced ORACLE® Universal Server, an extremely powerful software platform with the ability to integrate and consolidate all types of data for thousands of users over any network, including the World Wide Web.

For nearly two decades, ORACLE Corporation has been solving complex, critical information management challenges for companies of all types and sizes. In fact, ORACLE is the world's largest independent provider of software and services for managing information, with more than 42,000 dedicated software professionals, and operations in more than 90 countries.

ORACLE is the only company offering a 100% pure Internet suite of products to build and deploy e-business solutions on a global scale. Matter of fact, every one of the top 20 E-Commerce.coms and over

90% of all public '.com' companies work on ORACLE. ORACLE Corporation is the world's second largest software company and the leading supplier of software for enterprise information management. With annual revenues exceeding US$ 10.1 billion, ORACLE offers database, tools and application products along with consulting, education and support services in more than 145 countries worldwide.

ORACLE technology innovations have driven the computer industry, and more importantly, have enabled their customers to be more productive and more competitive using computers that cost less, but do more. This focus on software innovation explains why their information management software has emerged as the technology backbone for the Information Age, making possible tasks ranging from managing huge amounts of corporate information to delivering a favourite movie to your living room.

1.4 ORGANIZATION OF THE REPORT

The organization of report is as follows.

Chapter 2 gives the overview of ORACLE Applications and Application development.

Chapter 3 gives the analysis of the problem

Chapter 4 gives design details

Chapter 5 gives software implementation details and testing.

Chapter 6 gives the conclusion and suggestions for the future work followed by the Bibliography.

CHAPTER – 2

GENERAL CONSIDERATIONS

ORACLE Applications

2.1 Internet Computing Architecture

Internet Computing Architecture is a framework for three-tiered, distributed computing that supports ORACLE Applications products. Internet Computing Architecture distributes services among as many nodes on a network as are required to support the processing load. Each node is a machine on the network. Services are processes that run in the background, listening for requests and processing these requests. The HTTP service, for example, is a process that listens for and processes HTTP requests, and the Forms service is a process that listens for and processes requests for ORACLE Forms. The three tiers are the database tier, which manages ORACLE8i database; the application tier, which manages ORACLE Applications and other tools; and the desktop tier, which provides the user interface display. With Internet Computing Architecture, only the presentation layer of ORACLE Applications is on the desktop tier in the form of a plug-in to a standard Internet browser.

ORACLE Applications software and other tools are deployed on a middle tier of servers known as the application tier. This tier eliminates the need to install and maintain application software on each desktop client. The software on the application tier also enables ORACLE Applications to scale with load and to keep network traffic low.

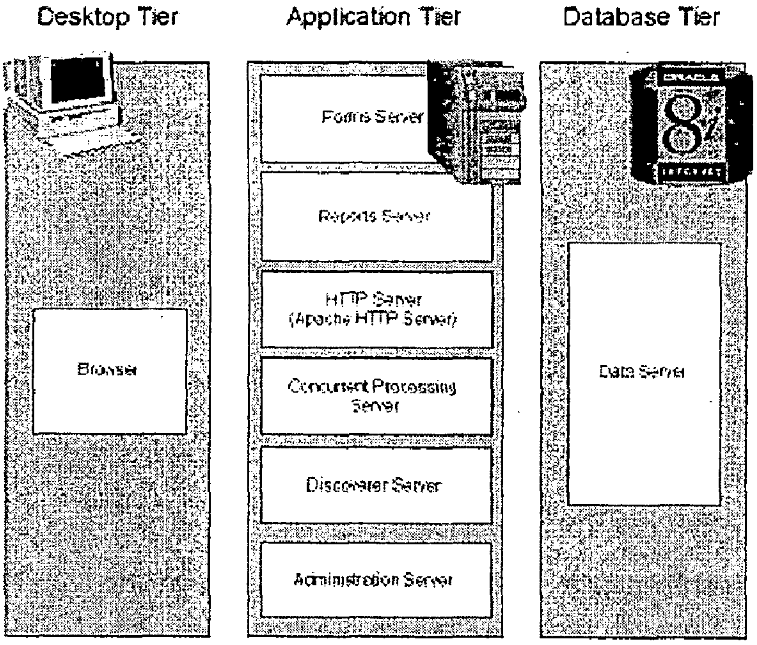


Figure 2.1 Internet Computing Architecture

The application tier servers operate very effectively over a WAN. The desktop client and application server send a minimum amount of information, such as field value comparison differences, but do not exchange graphical information such as screen painting. In a global operation with users at diverse locations, less network traffic also means less telecommunications expense.

Forms-based Products

Release 11i includes two principal product suites: Enterprise Resource Planning (ERP) products, and Customer Relationship Management (CRM) products.

Enterprise Resource Planning (ERP) Products

The ERP products are the "back office" products familiar to users of earlier ORACLE Applications releases. There are more than 156 ERP products that help the business manage important operations, including

product planning, purchasing, inventory management, interacting with suppliers, order tracking, human resources, financial
planning, and accounting. The ERP products are divided into several product families, such as Financials, Human Resources, Manufacturing and Distribution and Process Manufacturing.

Customer Relationship Management (CRM) Products

Customer Relationship Management products provide the "front office" functions such as call center management, e-commerce, and internet sales and marketing. CRM products help the business build lasting customer relationships and increase customer satisfaction and loyalty.

Forms Server and Forms Client

The application tier software used in most ERP and CRM products is the Forms server. The Forms server mediates between the Forms client, a Java applet running on the desktop, and the ORACLE8i database server on the back end. The Forms server produces the effects a user sees on the desktop screen and causes changes to database records based on user actions. Both the Forms server and Forms client are components of ORACLE Forms. The two exchange messages across a standard network connection, which may be either TCP/IP, or HTTP with or without SSL (Secure Sockets Layer).The Forms client can display any ORACLE Applications screen, and provides field-level validation, multiple coordinated windows, and data entry aids such as list of values. A Java-enabled Web browser manages the downloading, start-up, and execution of the Forms client on the desktop. Another software component, the HTTP server, helps start a client session over the internal or external Web. The HTTP server in Release 11i is the Apache HTTP Server. In installations that have multiple Forms servers, only one of the Forms servers runs the HTTP server software. If we use more than one Forms server, ORACLE Forms also provides a CGI script that distributes the processing load among the servers.

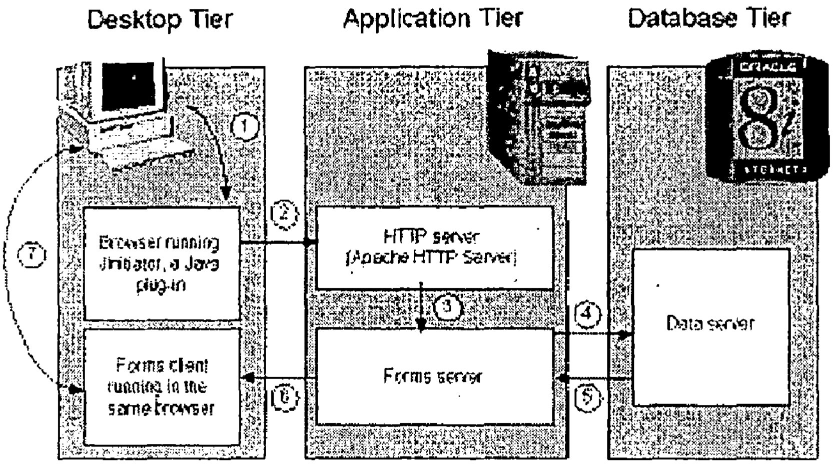

Desktop Tier
Application Tier
Database Tier
1
2
3
4
5
6
7
Browser running Jinitiator, a Java plug-in
HTTP server (Apache HTTP Server)
Data server
Forms client running in the same browser
Forms server
ORACLE
8i

Figure 2.2 Forms-based Architecture

HTML-based Products

In addition to Forms-based products, Release 11i includes other products that are not Forms-based, such as the ORACLE Self-Service Web Applications products, ORACLE Workflow, and the ORACLE Business Intelligence System (BIS) products. These products do not use the Forms server as the application tier software or the Forms client on the desktop, but rely on HTTP-based servers on the application tier and a Java-enabled Web browser on the desktop.

ORACLE Self-Service Web Applications and ORACLE Workflow

Self-Service Web Applications provide a fast and cost-effective way to get information to and from people within an organization or business. For example, Self-Service Web Applications allow customers to enter their own orders without involving the sales staff, or employees to enter their own change of address without involving the Human Resources staff. The interface is familiar to Web users, easy to work with, and doesn't require any training. Many ORACLE Applications products use ORACLE Workflow to automatically enforce business rules and policies and to provide a common notification system. The ORACLE Workflow monitors business processes, collects process data, and provides an e-mail and web page notification system. For example, when an employee uses ORACLE Internet Procurement (an ORACLE Self-Service Web product) to enter a requisition, ORACLE Workflow automatically validates the requisition and routes it to the appropriate manager for approval. Release 11i includes the full ORACLE Workflow product and the license to customize any ORACLE Applications embedded workflow. Most ORACLE Self-Service Web Applications and ORACLE Workflow are designed in HTML-based tools such as HTML, XML, and JavaScript. They operate by direct connection to the Apache HTTP server. Logic is controlled through stored procedures executed by the PL/SQL cartridge and by Java servlets and Java Server Pages (JSP) executed by the Apache JServ module. Apache communicates with the database using JDBC (Java Data Base Connectivity). The Apache HTTP Server can be the same machine used by ORACLE Forms.

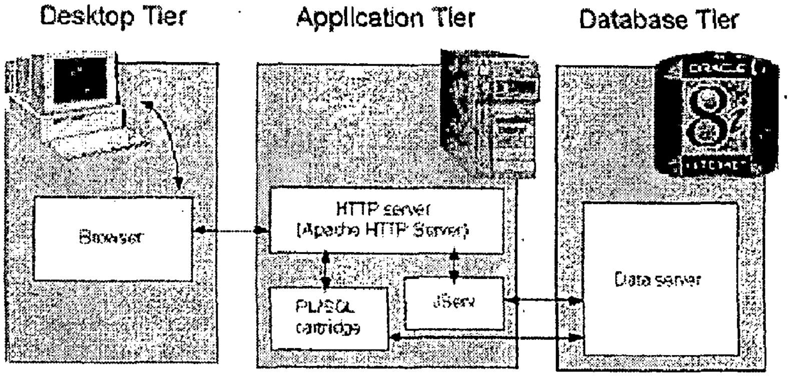


Fig 2.3 Self Service and Work Flow Architecture

2.2 The Network Computing Architecture

Place most user interface processing on a Java-enabled desktop, forms processing on application servers, and all data-oriented processing on database servers. It maximize the use of client and server resources while minimizing administration effort and network use.

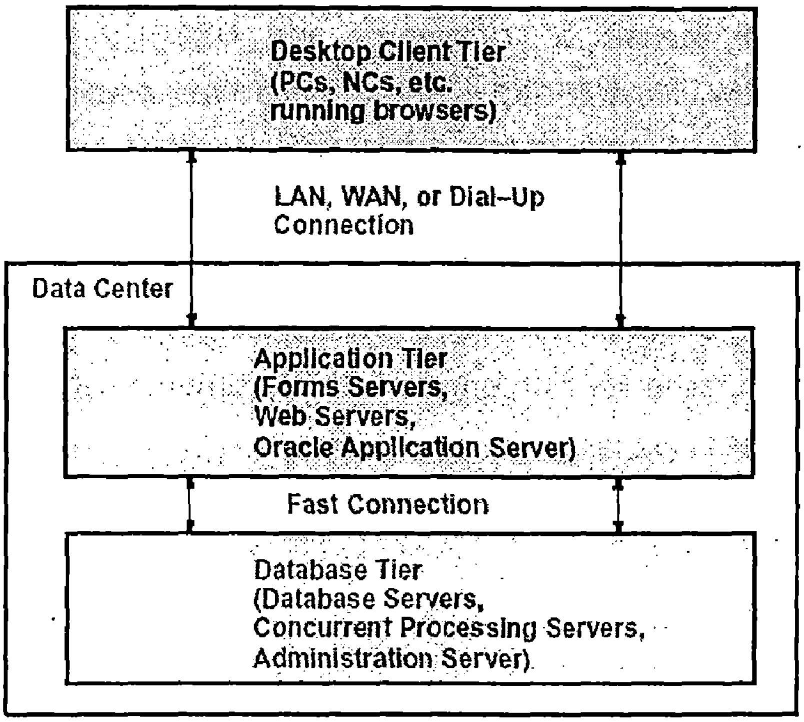


Fig 2.4 Network Connections in ORACLE Applications

Desktop Client

- Full graphical user interface, with user interface display handled by the browser or appletviewer

- Deploy on any PC, network computer, or other desktop on which Java is available
- Closely integrated windows present an entire business flow

Middle Tier (Forms Server)

- User interface logic happens on the forms server
- Key reference data cached locally
- Few network calls to database server needed

Database Server

- Remote Procedure Calls (RPCs) communicate with the database server when necessary
- Server handles data-oriented applications processing (for example, calculating tax on a sales order)
- Single RPC to stored procedures can initiate multiple database actions (SQL statements)
- Stored procedures are in the database, so communications between stored procedures and the database occur in memory, not across the network.

2.3 The Centralized Development Setup

In a "centralized" development setup, each developer has only the ORACLE Forms Designer and the browser on the developer's desktop machine. The middle tier resides on a shared server.

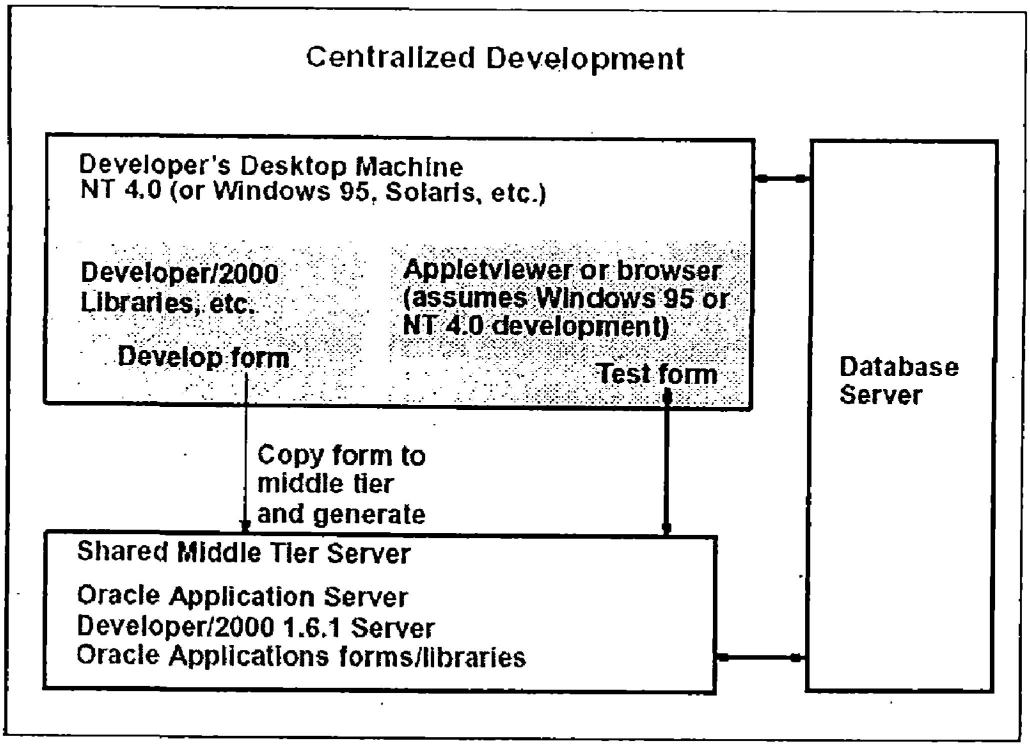
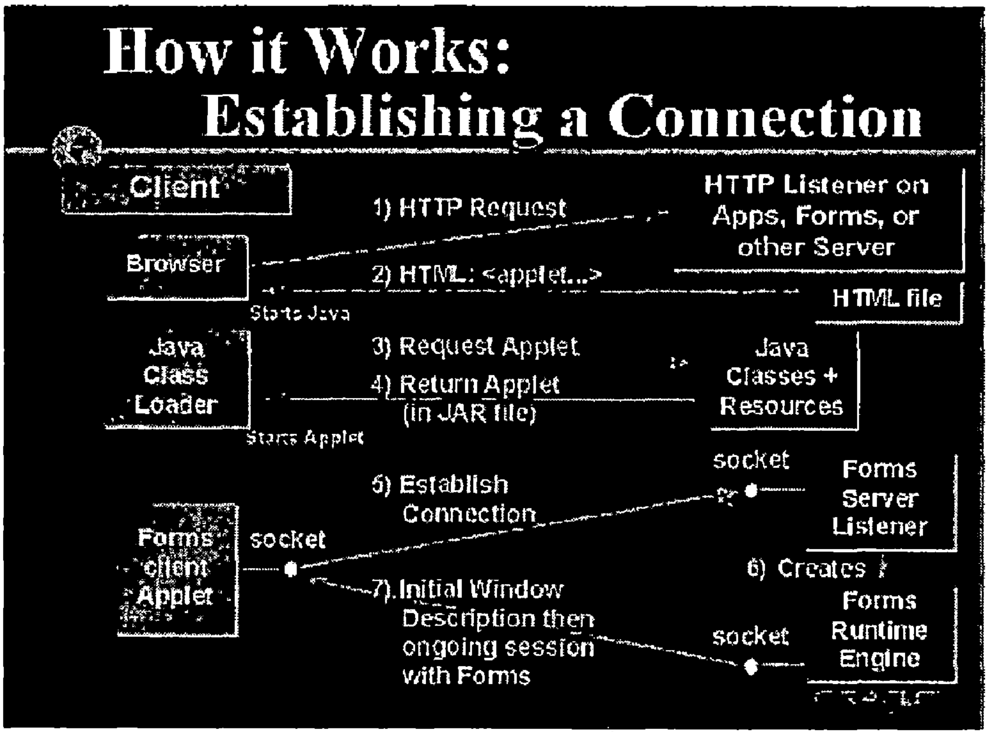


Fig 2.5 : Centralized development in ORACLE Applications

2.4 Running ORACLE Applications

When a user starts the appletviewer or browser and requests an appropriate URL. These are the connections between the desktop client and the applications tier.



How it Works:
Establishing a Connection

Client

Browser

1) HTTP Request

HTTP Listener on Apps, Forms, or other Server

2) HTML: <applet...>

HTML file

Starts Java

Java Class Loader

3) Request Applet

4) Return Applet (in JAR file)

Java Classes + Resources

Starts Applet

5) Establish Connection

socket

Forms Server Listener

Forms client Applet

socket

6) Creates

7) Initial Window Description then ongoing session with Forms

socket

Forms Runtime Engine

Figure 2.6 : How Connection Establishes

- The user starts up the appletviewer or browser and provides a URL (HTTP request) that requests an HTML page
- The request may be sent to the forms cartridge handler (ORACLE Application Server) to fill in any variables in the HTML file
- User or developer may include variables in the URL, depending on setup
- The HTML page contains a request for the Forms Client applet
- The Forms Client applet establishes a connection with the Forms Server
- The user sees a form (usually the ORACLE Applications Sign On window)

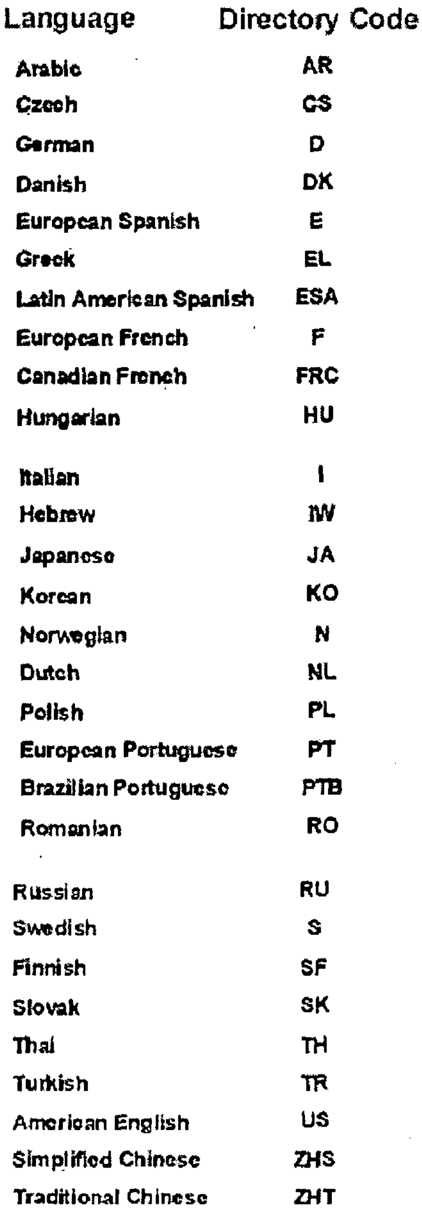
2.5 Platforms Supported By Oracle Applications

A *platform* is a particular computer hardware and software (operating system) combination. The Oracle8 Server, Tools, and Application Object Library layers help make Applications portable to many platforms:

- Sun SPARC Solaris
- IBM AIX RS/6000
- Digital UNIX
- Digital VMS (VAX, Alpha)
- HP/UX 98xx
- Intel-based UNIX
- Sequent DYNIX/ptx
- Solaris x86
- MIPS-based Unix
- SGI Irix
- Pyramid DC/OSx
- Windows NT server (Intel, Alpha)
- Siemens Nixdorf (SNI) RM600/800

Languages Supported by Oracle Applications

Oracle Applications can be run in languages other than American English (referred to as National Language Support or NLS), or it can be run in multiple languages simultaneously (referred to as Multi-Lingual Support or MLS).



Language	Directory Code
Arabic	AR
Czech	CS
German	D
Danish	DK
European Spanish	E
Greek	EL
Latin American Spanish	ESA
European French	F
Canadian French	FRC
Hungarian	HU
Italian	I
Hebrew	IW
Japanese	JA
Korean	KO
Norwegian	N
Dutch	NL
Polish	PL
European Portuguese	PT
Brazilian Portuguese	PTB
Romanian	RO
Russian	RU
Swedish	S
Finnish	SF
Slovak	SK
Thai	TH
Turkish	TR
American English	US
Simplified Chinese	ZHS
Traditional Chinese	ZHT

Figure 2.7 Languages Supported by Oracle Applications

File Character Set

Character sets are sets of encoded binary values that represent the letters, numerals, and punctuation marks of a language, or of a group of languages that use similar written symbols. For example, the WE8ISO8859P1 character set can be used by English and many other languages that use a Latin-based alphabet and Arabic numerals.

Terminals and printers handle text data by converting these encoded values to characters. A character set may also be called a code set.

A character set supports one or more languages. In Release 11*i*, support for the Unicode UTF8 character set removes the imitation on the number of supported languages that can be run in a single instance. The Unicode character set supports all characters in common use in all of the world's modern languages.

Multiple Reporting Currencies

Multiple Reporting Currencies (MRC) is a set of unique features embedded in Oracle Applications that permits an organization to report and maintain accounting records at the transaction level in more than one functional currency. MRC is intended for use by organizations that must regularly and routinely support statutory and legal reporting of both transactions and General Ledger account balances in multiple currencies, other than the primary functional currency.

2.6 Function Security

Function security restrict application functionality to authorized users.

Basic Function Security

- Group the forms and functionality of an application into logical menu structures .
- Assign a menu to one or more responsibilities
- Assign one or more responsibilities to one or more users
- Forms can be secured on a responsibility basis (that is, included or excluded from a responsibility)

Advanced Function Security

- ORACLE Applications GUI-based architecture aggregates several related business functions into a single form.
- Not all users should have access to every business function in a form.

- ORACLE Applications provides the ability to identify pieces of application logic as functions.
- Functions can be secured on a responsibility basis (that is, included or excluded from a responsibility).

Function Security extends the definitions of these existing terms.

Menu

A menu is a hierarchical arrangement of functions and menus of functions.

Menu Entry

A menu entry is a menu component that identifies a function or a menu of functions.

In some cases, both a function and a menu of functions correspond to the same menu entry. For example, a form and its menu of subfunctions can occupy the same menu entry.

Responsibility

When application users sign on, they select a responsibility that determines, among other things, the functions they may access.

Available functions are determined by the menu assigned to the current responsibility.

Forms

An ORACLE Forms .fmx file.

Forms are located in their application Basepath/forms/US (or appropriate language) directory.

Function Security introduces the following new terms:

Function

- A function is a part of an application's functionality, registered under a unique name, that can be assigned to or excluded from a responsibility.
- There are two types of functions: form functions (forms), and non-form functions (subfunctions).

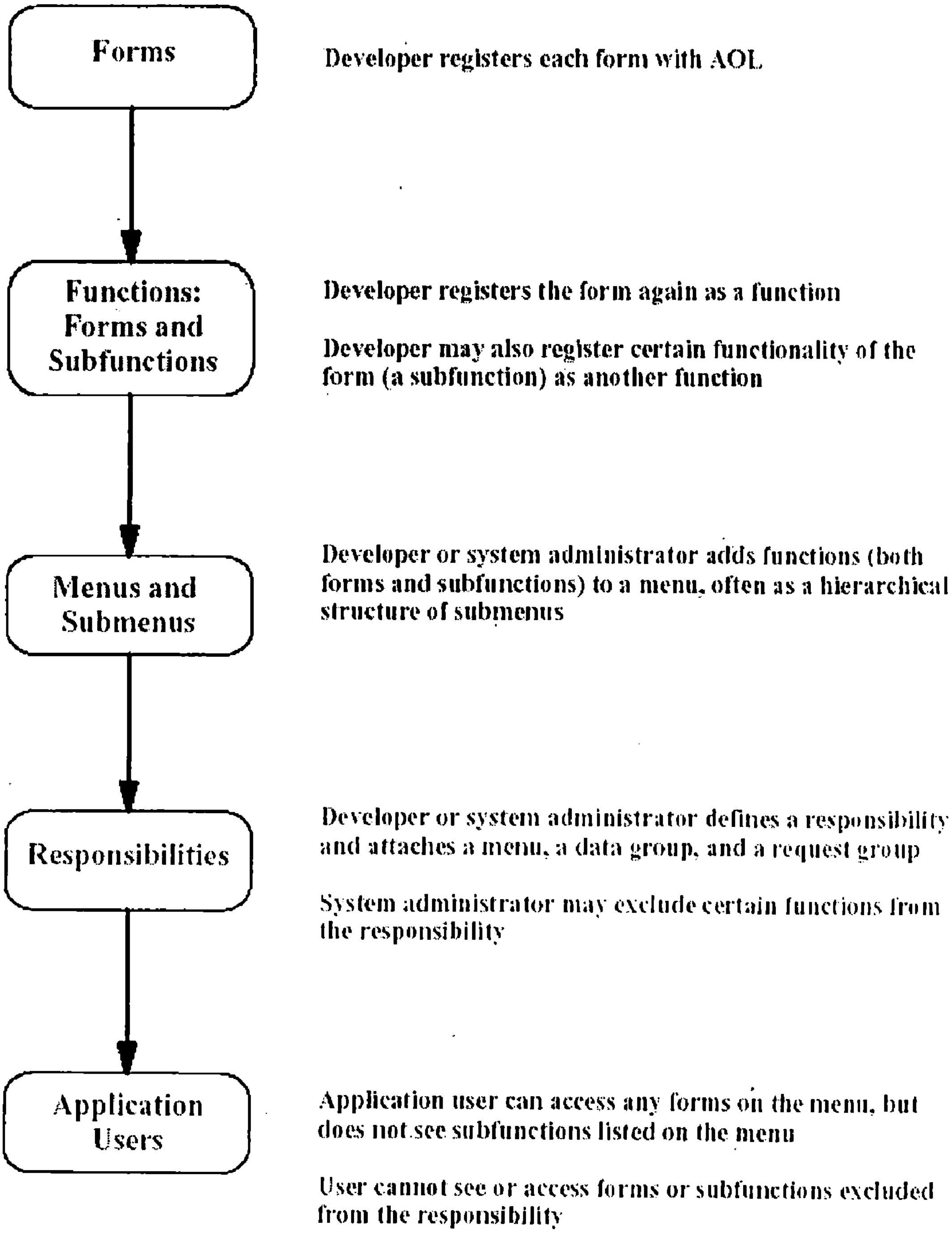
Form Function

- A form (form function) invokes an ORACLE Forms form.
- A form has the unique property that users may navigate to it from the Navigate window.

Subfunction

- A subfunction (non-form function) is a securable subset of a form's functionality
- A developer can write logic to test the availability of a subfunction in the current responsibility, then take some action based on whether the subfunction is available
- A subfunction is frequently associated with a button or an entry on the Special menu. When such a subfunction is enabled, the corresponding button or menu entry is enabled
- A subfunction may correspond to a form procedure not associated with a graphical element, and its availability may not be obvious to the end user

SETTING UP FUNCTION SECURITY



Developer registers each form with AOL

Developer registers the form again as a function

Developer may also register certain functionality of the form (a subfunction) as another function

Developer or system administrator adds functions (both forms and subfunctions) to a menu, often as a hierarchical structure of submenus

Developer or system administrator defines a responsibility and attaches a menu, a data group, and a request group

System administrator may exclude certain functions from the responsibility

Application user can access any forms on the menu, but does not see subfunctions listed on the menu

User cannot see or access forms or subfunctions excluded from the responsibility

Figure 2.8 : Setting up Function Security

Developers Create Functions and Menus of Functions

- Developers can require parts of their form code to test the availability of a particular function, then take some action based on whether the function is available.
- Developers register each function they create.
- For form functions, developers can register parameters that pass values to a function. For example, a form may support data entry only when a function parameter is passed to it.
- Developers define a menu including all the functions available in an application (that is, all the forms and their securable subfunctions).
- For some applications, developers may define additional menus that restrict the application's functionality by omitting certain forms and subfunctions.

System Administrators Exclude Functions from Menus

- Each ORACLE Applications product is delivered with one or more predefined menus.
- System Administrators can assign a predefined menu to a responsibility.
- To tailor a responsibility, a System Administrator can exclude functions or menus of functions using exclusion rules.
- When a menu is excluded, all of the functions and menus of functions that it selects are excluded.
- When a function is excluded, all occurrences of that function throughout the responsibility's menu are excluded.

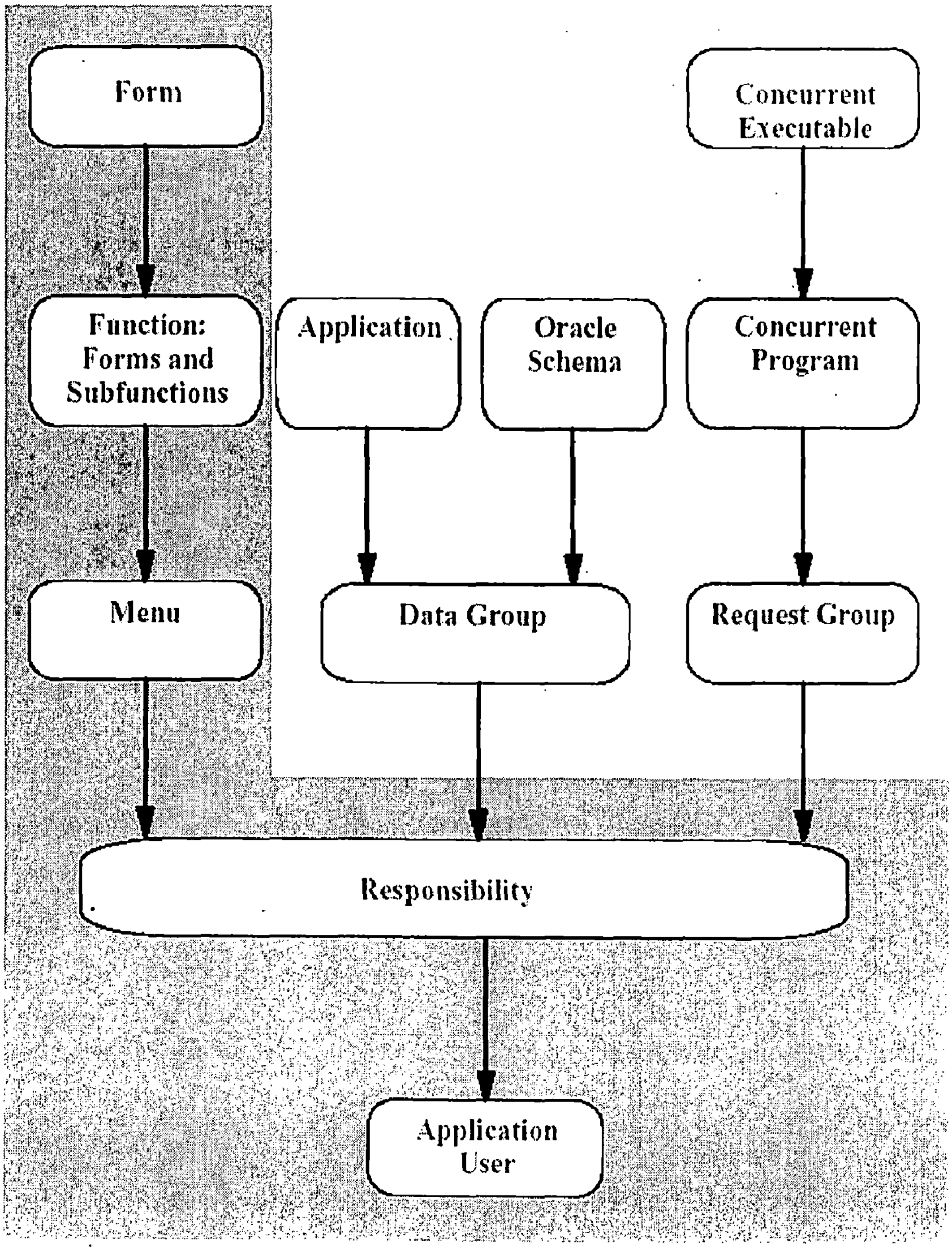


Figure 2.9 : Applications Development Process

2.7 Handlers

- Handlers provide clear standards for writing forms.
- They are centralized pieces of code that deal with a specific event, item or table.
- They are packaged procedures called from triggers

Use of Handlers :

- Easy to work with self-contained code
- All the code that affects an entity appears in one place for easy development and maintenance

Types of Handlers

- Item handlers – for each item in a block
- Event handlers – for complex events
- Table handlers – for base tables

ITEM HANDLERS VALIDATE ITEMS

Item handlers take an EVENT parameter that identifies the trigger calling the item handler.

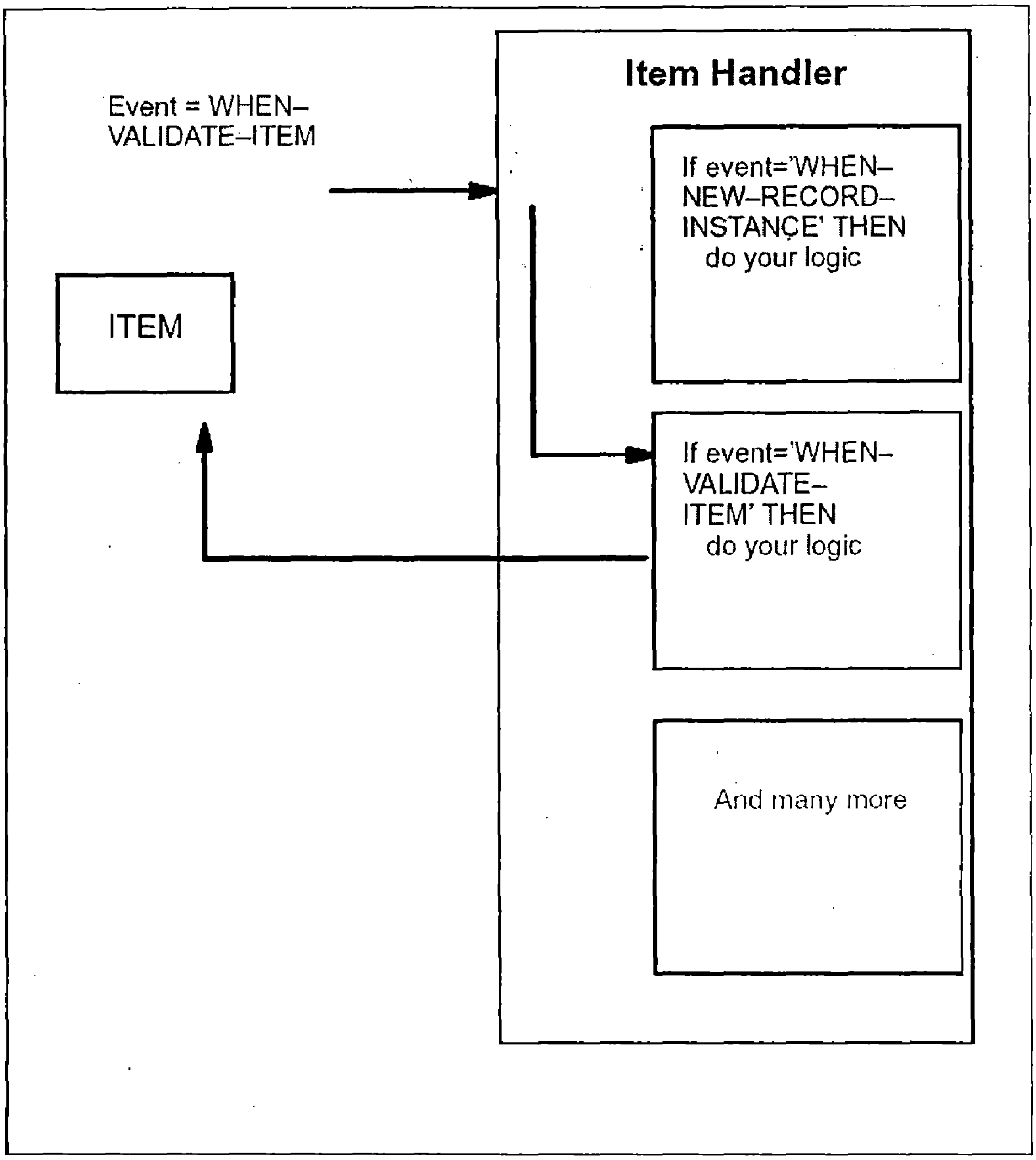

Event = WHEN–VALIDATE–ITEM
ITEM
Item Handler
If event='WHEN–NEW–RECORD–INSTANCE' THEN do your logic
If event='WHEN–VALIDATE–ITEM' THEN do your logic
And many more

Figure 2.10 : Item Handlers

Common item handler events include:

WHEN-VALIDATE-ITEM

Call an item handler to validate and set the dynamic

item attributes.

WHEN-NEW-RECORD-INSTANCE

Reset the item attributes to the default for a new record.

INIT

Examine current conditions and reset defaults and

dynamic attributes as necessary. Usually called by other

handlers that affect this item.

```
PROCEDURE example (EVENT VARCHAR2) IS

BEGIN

IF (EVENT = 'INIT') THEN

... /* code here */

END IF;

END example;
```

One Package Per Block

_ Named after the block (or form in the single block case).

_ Procedures named after their particular item.

Call Item Handlers from Triggers

_ Pass the trigger name (event) as the argument to the handlers.

- Always code to expect event name, even if we only have one event; we

may later want to add more events.

_ Grouping the code into a single package simplifies maintenance and

debugging.

EVENT HANDLERS CONTROL EVENTS

Some logic pertains to multiple items when a single event occurs.

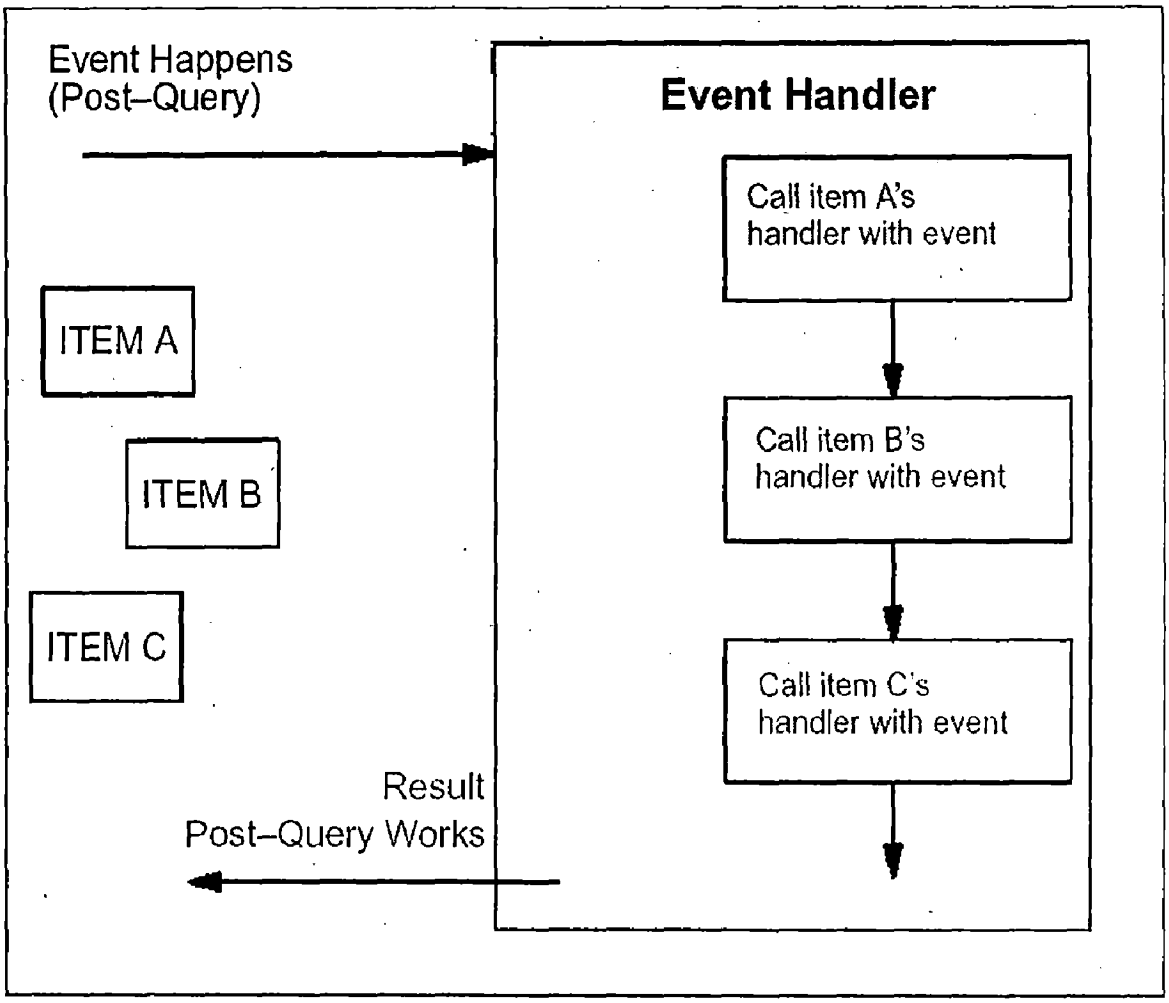

Event Happens
(Post–Query)
Event Handler
Call item A's handler with event
Call item B's handler with event
Call item C's handler with event
ITEM A
ITEM B
ITEM C
Result
Post–Query Works

Figure 2.11 : Event Handler Controlling Events

Table handlers support insert, update, delete and locks for the block-level views.

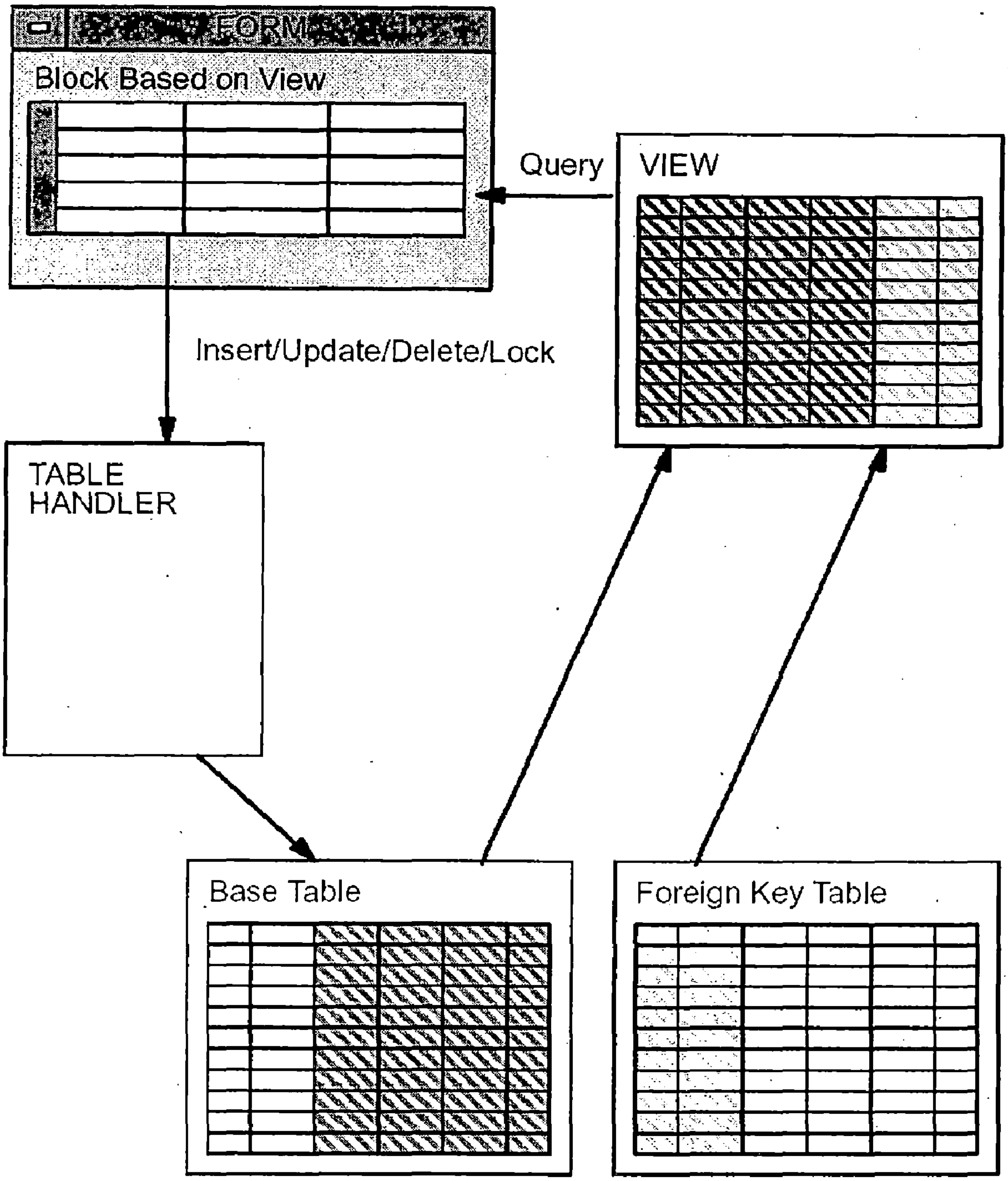


Figure 2.12 : Table Handler Controlling Database Upadation

2.8 User Profiles

User profiles lets us code logic based on a user's Site, Application, Responsibility, User-level information.

Overview of User Profiles

- A user profile is a set of changeable options that affects how the application looks and behaves.
- Developer can define user profile options whenever he wants the application to react in different ways for different users, depending on specific user attributes.

Four Different Levels for Maximum Flexibility :

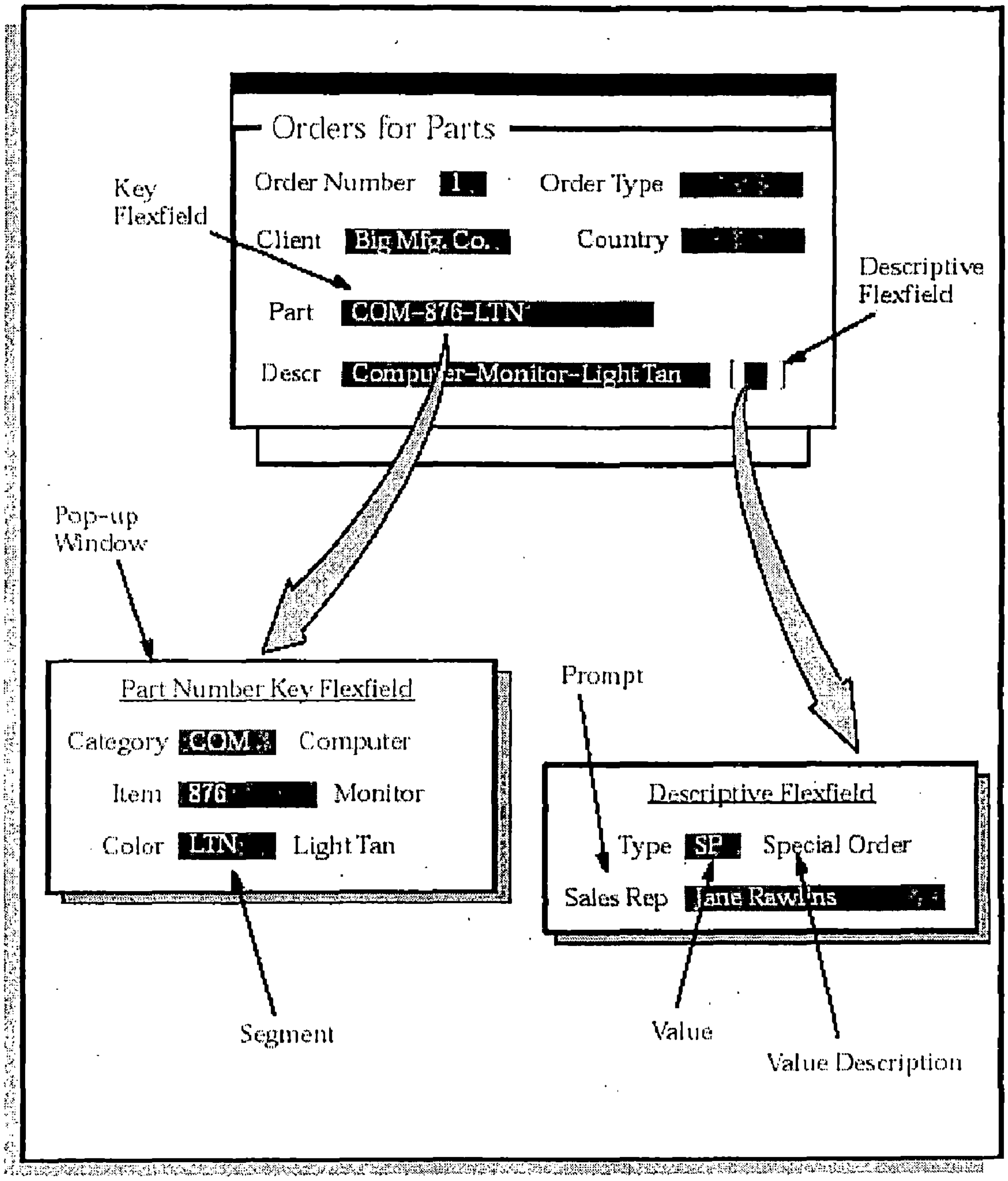
- Site: values pertain to all users at an installation site.
- Application: values affect all users of any responsibility associated with the application.
- Responsibility: values affect all users currently signed on under the responsibility.
- User: values affect an individual applications user.
- Site-level setting overridden by application, then responsibility, with user-level having the highest priority.

Profile Option Values Derived at Runtime

- ORACLE Application Object Library establishes values when the user logs on or changes responsibility.
- If a user changes a user-changeable profile option value, the new value takes effect immediately.
- If a system administrator changes a user's profile option value, the change takes effect when the user logs on again or changes responsibility

2.9 FLEXFIELDS

A flexfield is a field made up of sub-fields, or segments. A flexfield appears on the form as a pop-up window that contains a prompt for each segment. Each segment has a name and a set of valid values. There are two types of flexfields: key flexfields and descriptive flexfields.



Orders for Parts

Order Number 1 Order Type

Client Big Mfg. Co. Country

Part COM-876-LTN

Descr Computer-Monitor-Light Tan

Part Number Key Flexfield

Category COM Computer

Item 876 Monitor

Color LTN Light Tan

Descriptive Flexfield

Type SP Special Order

Sales Rep Jane Rawlins

Figure 2.13 : Flex fields

2.9.1 Key Flexfields

Most organizations use "codes" made up of meaningful segments (intelligent keys) to identify general ledger accounts, part numbers, and other business entities. Each segment of the code can represent a characteristic of the entity. For example, an organization might use the part number PAD-NR-YEL-8 1/2x14" to represent a notepad that is narrow-ruled, yellow, and 8 1/2" by 14". Another organization may identify the same notepad with the part number "PD-8x14-Y-NR". Both of these part numbers are codes whose segments describe a characteristic of the part. Although these codes represent the same part, they each have a different segment structure that is meaningful only to the organization using those codes.

The ORACLE Applications store these "codes" in key flexfields. Key

flexfields are flexible enough to let any organization use the code scheme they want, without programming. When an organization initially installs ORACLE Applications, the organization's implementation team customize the key flexfields to incorporate code segments that are meaningful to the business. It decides what each segment means, what values each segment can have, and what the segment values mean. The organization can define rules to specify which segment values can be combined to make a valid complete code (also called a combination). They can also define relationships among the segments. The result is that the

organization can use the codes it wants rather than changing the codes to .meet ORACLE Applications' requirements .For example, consider the codes an organization uses to identify general ledger accounts. ORACLE Applications represent these codes using a particular key flexfield called the Accounting Flexfield. One organization might choose to customize the Accounting Flexfield to include five segments: company, division, department, account, and project. Another

organization, however, might structure their general ledger account segments differently, perhaps using twelve segments instead of five. The Accounting Flexfield lets ORACLE General Ledger application accommodate the needs of different organizations
by allowing them to customize that key flexfield to their particular business usage.

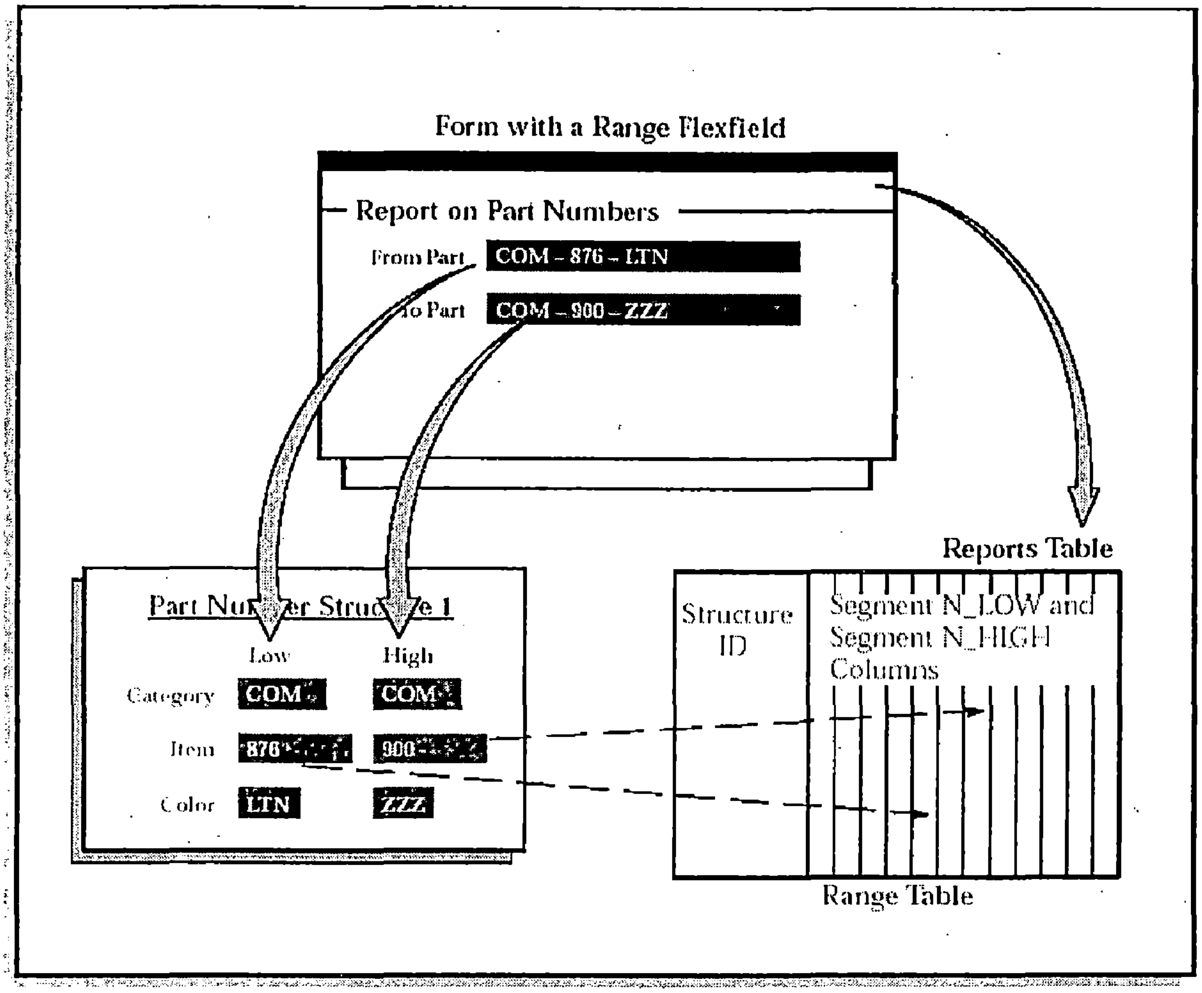


Figure 2.10 Storage of key Flex field

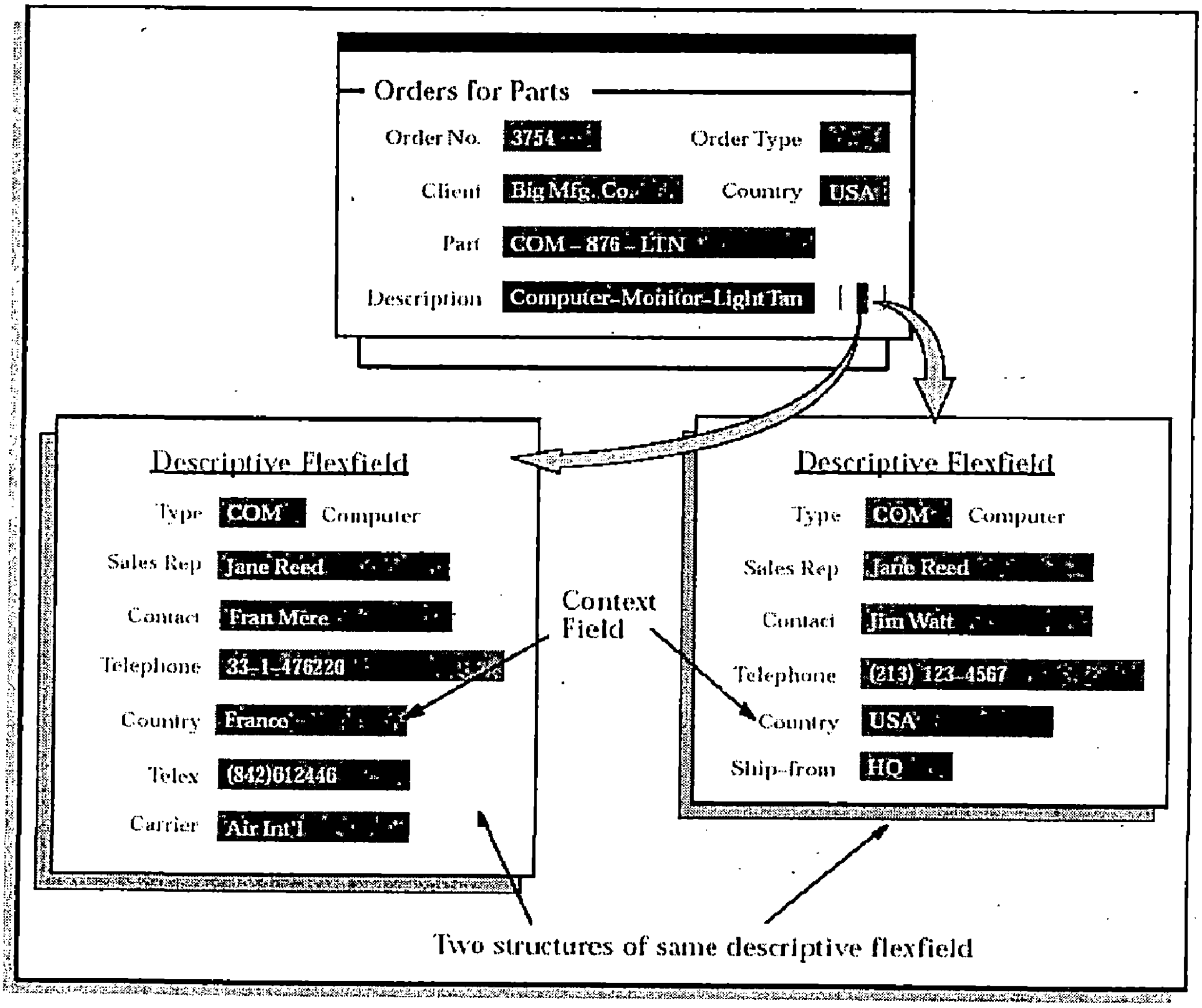
2.9.2 Descriptive Flexfields

Descriptive flexfields provide customizable "expansion space" on the forms. We can use descriptive flexfields to track additional information, important and unique to the business, that would not otherwise be captured by the form. Descriptive flexfields can be context sensitive, where the information the application stores depends on other values it's users enter in other parts of the form. A descriptive flexfield appears on a form as a single-character, unnamed field enclosed in brackets. Just like in a key flexfield, a

pop-up window appears when the user move the cursor into a customized descriptive flexfield. And like a key flexfield, the pop-up window has as many fields as that organization needs. Each field or segment in a descriptive flexfield has a prompt, just like ordinary fields, and can have a set of valid values. An organization can define dependencies among the segments or customize a descriptive flexfield to display context-sensitive segments, so that different segments or additional pop-up windows appear depending

on the values the user enters in other fields or segments. For example, consider the Additions form we use to define an asset in ORACLE Assets application. This form contains fields to capture the "normal" information about an asset, such as the type of asset and an asset number. However, the form does not contain specific fields for each detail about a given asset, such as amount of memory in a computer or lifting capacity of a forklift. In this case, having all the potentially-needed fields actually built into the form is not only difficult, it is undesirable. Because while one organization may have

computers and forklifts as assets, another organization may have only computers and luxury automobiles (and no forklifts) as assets. If the form contained built-in fields for each attribute of a forklift, for example, an organization with no forklifts would find those fields to be both unnecessary and a nuisance because a user must skip them to enter information about another type of asset. In fact, fields for forklift information would be cumbersome whenever a user in any organization tries to enter any asset that is not a forklift. Instead of trying to contain all possible fields for assets information, the Additions form has a descriptive flexfield that it can customize to capture just the information the organization needs about its assets. The flexfield structure can depend on the value of the Asset Category field and display only those fields (segments) that apply to the particular type of asset. For example, if the asset category were "desk, wood", the descriptive flexfield could prompt for style, size and wood type. If the asset category were "computer, hardware", the flexfield could prompt for CPU chip and memory size. They can even add to the descriptive flexfield later as they acquire new categories of assets .The Enter Journals window in the ORACLE General Ledger applications is another example of a form that includes descriptive flexfields to allow organizations to capture additional information of their own choosing. Each block contains a descriptive flexfield as its last field. It might use these to store additional information about each journal entry, such as a source document number or the name of the person who prepared the entry.



Orders for Parts

Order No. 3754
Order Type
Client Big Mfg. Co.
Country USA
Part COM - 876 - LTN
Description Computer-Monitor-Light Tan

Descriptive Flexfield

Type COM Computer
Sales Rep Jane Reed
Contact Fran Mère
Telephone 33-1-476220
Country France
Telex (842)612446
Carrier Air Int'l

Descriptive Flexfield

Type COM Computer
Sales Rep Jane Reed
Contact Jim Watt
Telephone (213) 123-4567
Country USA
Ship-from HQ

Fig 2.11 Descriptive Flexfield representations

Benefits of Flexfields

Flexfields provide the features an organization needs to satisfy the following business needs:

- Customize the applications to conform to the current business practice for accounting codes, product codes, and other codes.
- Customize the applications to capture data that would not otherwise be tracked by the application.
- Have "intelligent fields" that are fields comprised of one or more segments, where each segment has both a value and a meaning.
- Rely upon the application to validate the values and the combination of values that the user enters in intelligent fields.
- Have the structure of an intelligent field change depending on data in the form or application data.
- Customize data fields to meet the business needs without programming.

2.10 CONCURRENT PROCESSING

Concurrent processing simultaneously executes programs running in the background with online operations.

Purpose of Concurrent Processing

_ Take advantage of multitasking, parallel processing and distributed processing

_ Give system administrators flexibility to set up concurrent processing to fit the needs of their site and users

Benefits of Concurrent Processing

_ Consistent response time

_ Use all the capacity of hardware by executing many processes at once

_ Control the number of processes that run on each node

_ On-line file review

Portable interface to concurrent programs

CONCURRENT PROCESSING DEFINITIONS

Concurrent Program

An instance of an execution file, along with parameter definitions and incompatibilities. Several concurrent programs may use the same execution file to perform their specific tasks, each having different parameter defaults and incompatibilities.

Concurrent Program Executable

An executable file that performs a specific task. The file may be a program written in a standard language, a reporting tool or an operating system language

Concurrent Request

A request to run a concurrent program as a concurrent process

Concurrent Process

An instance of a running concurrent program that runs simultaneously with other concurrent processes

Concurrent Manager

A program that processes user's requests and runs concurrent programs.

System Administrators define concurrent managers to run different kinds of requests

Concurrent Queue

List of concurrent requests awaiting processing by a concurrent manager.

CHAPTER – 3

REQUIREMENT ANALYSIS

The first phase of Software Engineering Approach is the Requirement Analysis Phase. Here we analyse the requirement of the client.

The specifications given to me for each functional module were:

ABC Trading Corporation is planning to centralize its Purchasing Department eliminating the concept of paper requisitions, purchase orders and receipts.

Under this scenario, the system should cater to the following need of the user at ABC TC.

Requisitions

1. The user at ABC TC should be able to create, edit and view requisitions for items.
2. The user at ABC TC should be able to raise requisitions for multiple items.
3. The requisitions should contain the Preparer's Name and description.
4. For each of the items, the requisition should have the following information viz., the item number, description, unit price, the quantity, the date of requirement, destination and supplier details. The following are the destination details viz., the destination type, the requester and the warehouse. The following are the supplier information viz., supplier name, supplier site, supplier contact and phone number.

5. ABC TC has supplier information loaded in his already existing legacy system. He should be able to use the same.
6. The user should be able to specify the unit in which the item can be specified. He should also be able to change this.
7. The system should automatically calculate the total amount for the items as well as for the requisition.
8. Some users at ABC TC would want to enter the Requisition number manually, while others would want automatic numbering of Requisition. Hence, the system should provide for either manual or automatic numbering of requisitions.
9. Check for unique requisition number should be done in this case and a message should be displayed if the number entered is not unique.
10. The date of requirement entered should be always on or after the current date. On entering a date prior to the current date a message should be displayed and he should be prevented from doing so.
11. The user should be able to select the item, supplier details, destination details from a set of pre-defined values.
12. As far as the supplier is concerned, the user should be able to perform one of the
    - select from a set of pre-defined values
    - enter a new supplier name that does not exist in the set of pre-defined values
    - leave it blank.
13. ABC TC does not adhere to a rigid structure as far as the details that appear on requisitions are concerned. Its requirement is largely varied and hence his largely changing/growing business needs should be catered to with minimal effort. For instance in addition to the information stated earlier the user might want to enter the reason for requesting the item, the supplier item name, supplier item code, the supplier item description etc.

14.There should exist a control by which, the user would be allowed to raise multiple requests for the same item in a single requisition or would not be allowed to perform the same.

15.The user should be able to query requisitions based on the item., preparer 's name, supplier, status of requisition and destination details.

Querying

User frequently queries on the following attributes in different screens,

(i) For Requisitions it's done on Requisition Number , Item name, Preparer 's name , Date of preparation ,Supplier's name.

Purchase Orders

1. The user at ABC TC should be able to create, edit and view purchase orders for items.
2. The user should be able to create a Purchase Order based on an existing approved requisition.
3. The user should be able to raise Purchase Orders for multiple items.
4. The Purchase Order should contain the following information, supplier name, supplier site, contact, ship-to and bill-to locations, the buyer name and the amount for which the purchase order is raised.
5. The user should be able to define the following terms and conditions for the Purchase Order viz:, payment terms, freight terms and freight carrier.
6. The Purchase order should contain the following information pertaining to the item viz., the item number, item type, description, unit of measure, quantity, the unit price, promised date of receipt and date of requirement.

7. The Purchase Order should contain the information regarding the requisition against which it was prepared (if one exists).
8. Some users at ABC TC would want to enter the Purchase Order number manually, while others would want automatic numbering of Purchase Orders. Hence, the system should provide for either manual or automatic numbering of Purchase Orders.
9. Check for unique Purchase Order number should be done in this case and a message should be displayed if the number entered is not unique.
10. The date of requirement and promised date of receipt entered should be always on or after the current date. On entering a date prior to the current date a message should be displayed and he should be prevented from doing so.
11. The user should be able to select the following from a set of pre-defined values viz., supplier name, supplier site, supplier contact, ship-to and bill-to locations and the item.
12. There should exist a control by which, the user would be allowed to raise multiple requests for the same item in a single Purchase Orders or would not be allowed to perform the same.
13. The user should be able to define the ship-to location(warehouse), quantity, promised and need by dates for each item. The user should be able to see the status of the quantities i.e., the ordered quantity and received quantity.
14. The Purchase Order should define the following, the maximum acceptable number of days early/late to accept receipts. The user should also be able to define the tolerance for the quantity that can be received.

Querying

User frequently queries on the following attributes in different screens,

(i) For Purchase Orders it's done on Order Number, Supplier, Order Date..

(ii) For Purchase Order lines it is done on Item name and Requestor

Receipts

1. The user at ABC TC should be able to enter receipts against approved Purchase Orders, edit and view them.
2. The user should be able to enter multiple receipts against a single Purchase Order as well as enter a single receipt against multiple Purchase Orders.
3. The user should be able to find expected receipts (the Purchase Orders yet to be received) based on the following criteria, Purchase Order Number, Requisition Number, supplier name, warehouse and item.
4. The user should be able to enter the following for the receipts viz., the receipt number, shipped date, packing slip, waybill or airbill number, freight carrier and container information.
5. The user should be able to receive the items either fully or partially.
6. The receipt should have the following information about the item for each transaction viz., the quantity received, the unit in which it was received, the item, the item description, the warehouse and the requester.
7. On receiving the on-hand quantity at the warehouse level should be maintained.

Querying

User frequently queries on the following attributes in different screens,

(i) For Receipts it's done on Receipt Number, Supplier, Receiving Date.

Report

1. User at ABC TC should be able to take a report of Purchase Orders
2. These reports should be launched either from the menu as well as a separate concurrent program
3. The user should be able to obtain a report based on the following values the PO number, Item , Creation Date (From/To) and supplier.

Invoices

1. The user at KLS TC should be able to create, edit, cancel and view Invoices that are created against PO/Receipt or unmatched Invoices.
2. The user should be able to create an Invoice against an Approved PO.
3. The Invoice should contain foll. Information, Supplier Name, Site Name, Payment Terms, Payment Type, Pay group details, paid or unpaid, Approved or unapproved.
4. The user should be able to match multiple PO's against the same Invoice making sure that the Supplier Information is the same.
5. The Invoice should contain info. Regarding to which PO the Invoice was matched to.
6. The Invoice numbering should be automatic or manual. The Invoice number should be unique for a supplier.
7. Invoices can also be created for a negative amount.

Payments

1. Only the approved Invoices should be allowed to be paid.
2. Invoices can't be overpaid.
3. Payment documents need to be routed thru' a bank & must have a payment document.

4. Invoices can be paid thru' a batch or can be paid alone. If there are negative payments the amount should be adjusted.

Report

1. User should be able to take a report of all the Invoices
2. User should be able to take a report of all the Payments made against Invoices
3. User should at any instant find out the outstanding balances/credit against a Supplier.

Generic

1. Through out the application audit trail is to be maintained. There should be four fields that take care of this. Created by, Creation date, Last Updated by, Last Update date.
2. Where ever required give error messages like invalid item, item not in price list etc.
3. For all the date fields, a user should be able to see a list of value that gives all valid dates.
4. In all screen, for doing query provide find option that allows user to enter multiple select criterion for executing query

After doing Analysis I arrived on following proposals
The project should consist of 2 major parts ;

- Purchasing
- Payables

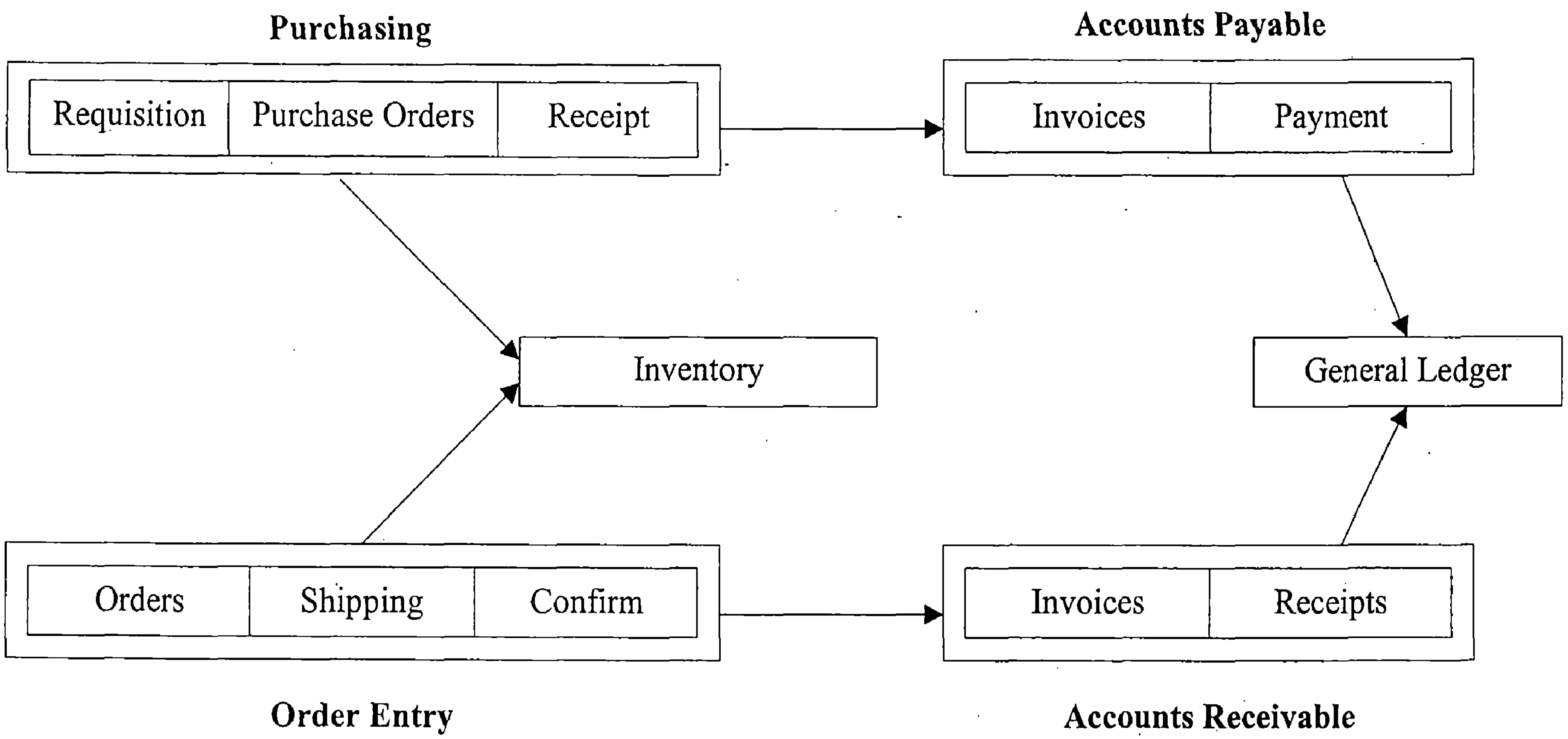
CHAPTER – 4

DESIGN

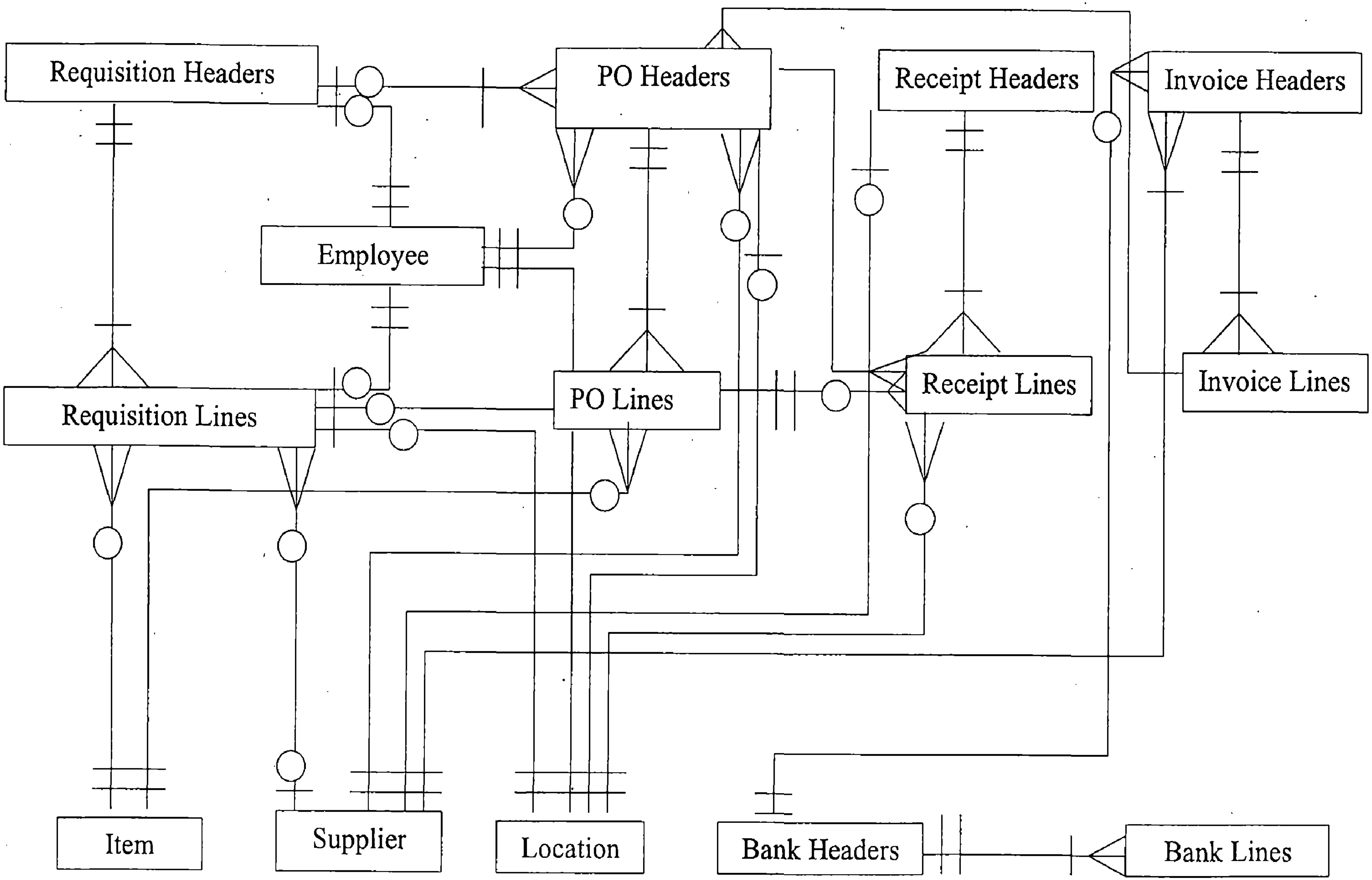
4.1 System Modeling

It is here the Data Flow Diagram(DFD) and the Entity Relationship Diagram(ER Diagram) are presented. The reason for doing this designs are - After System Analysis an English narrative of the system is often too vague. The system specifications are often redundant. To find information about one part of the system, one has to search through the entire document. Because of these drawbacks, structured tools such as Data Flow Diagram are used in designing a System.

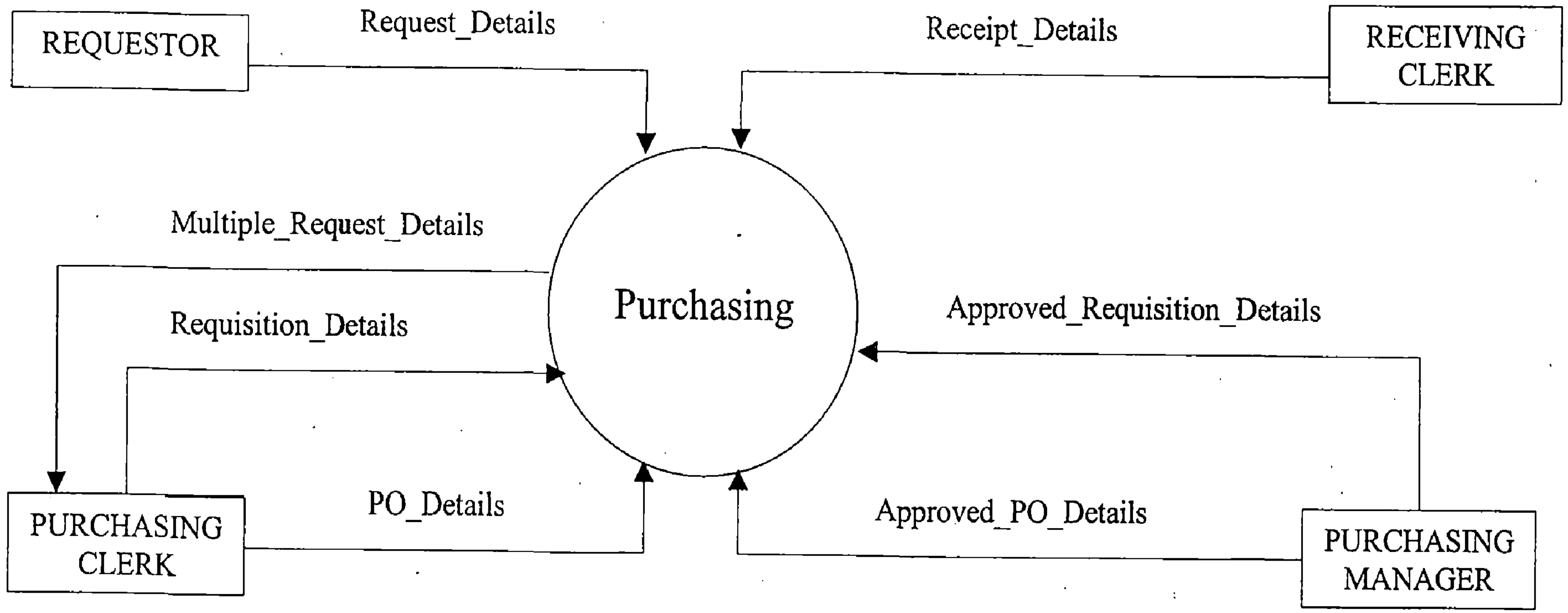
Enterprise Management System



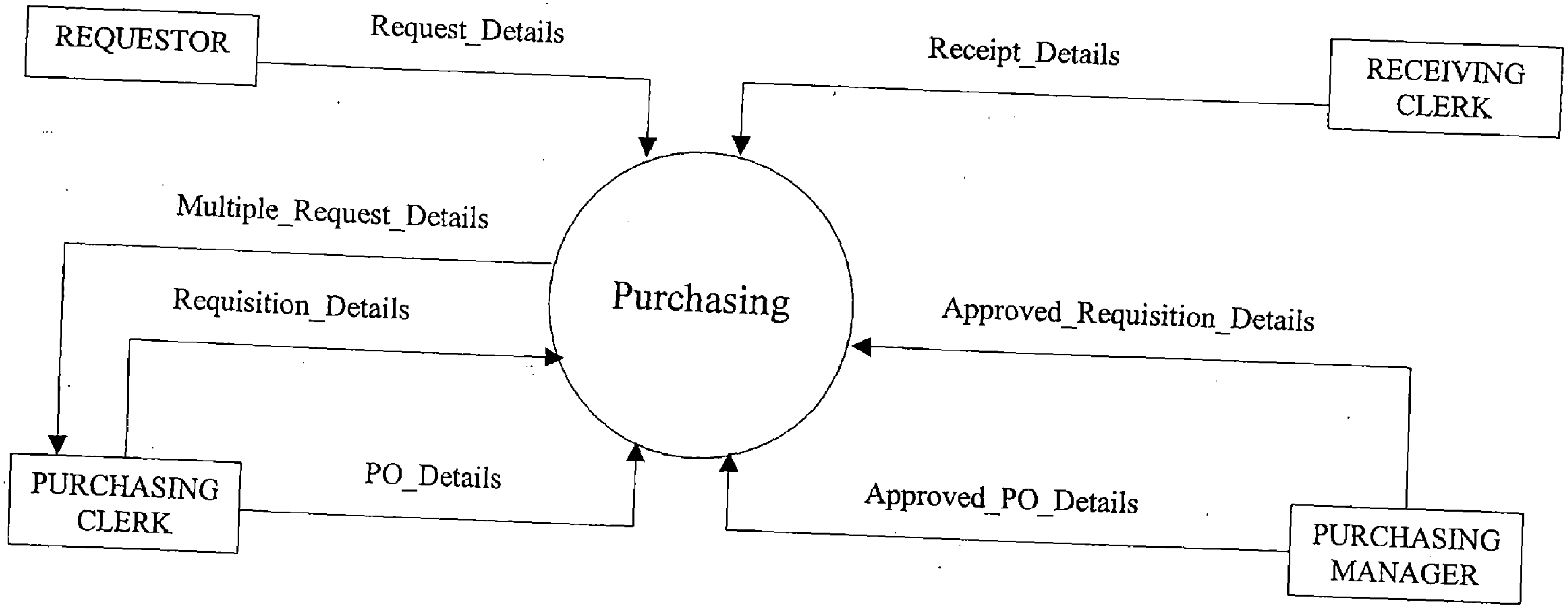
MODULE INTER DEPENDECNCIES


Requisition Headers
PO Headers
Receipt Headers
Invoice Headers
Employee
Receipt Lines
Invoice Lines
Requisition Lines
PO Lines
Item
Supplier
Location
Bank Headers
Bank Lines

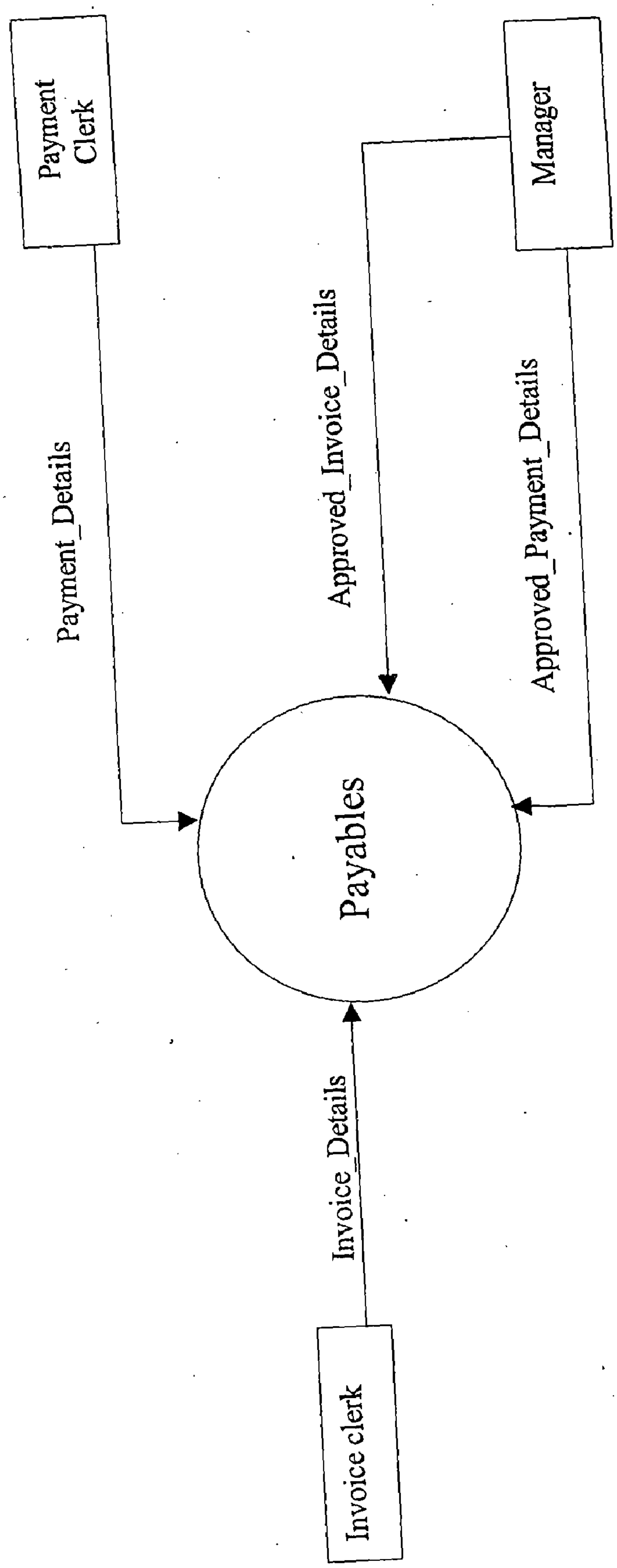
E.R. DIAGRAM



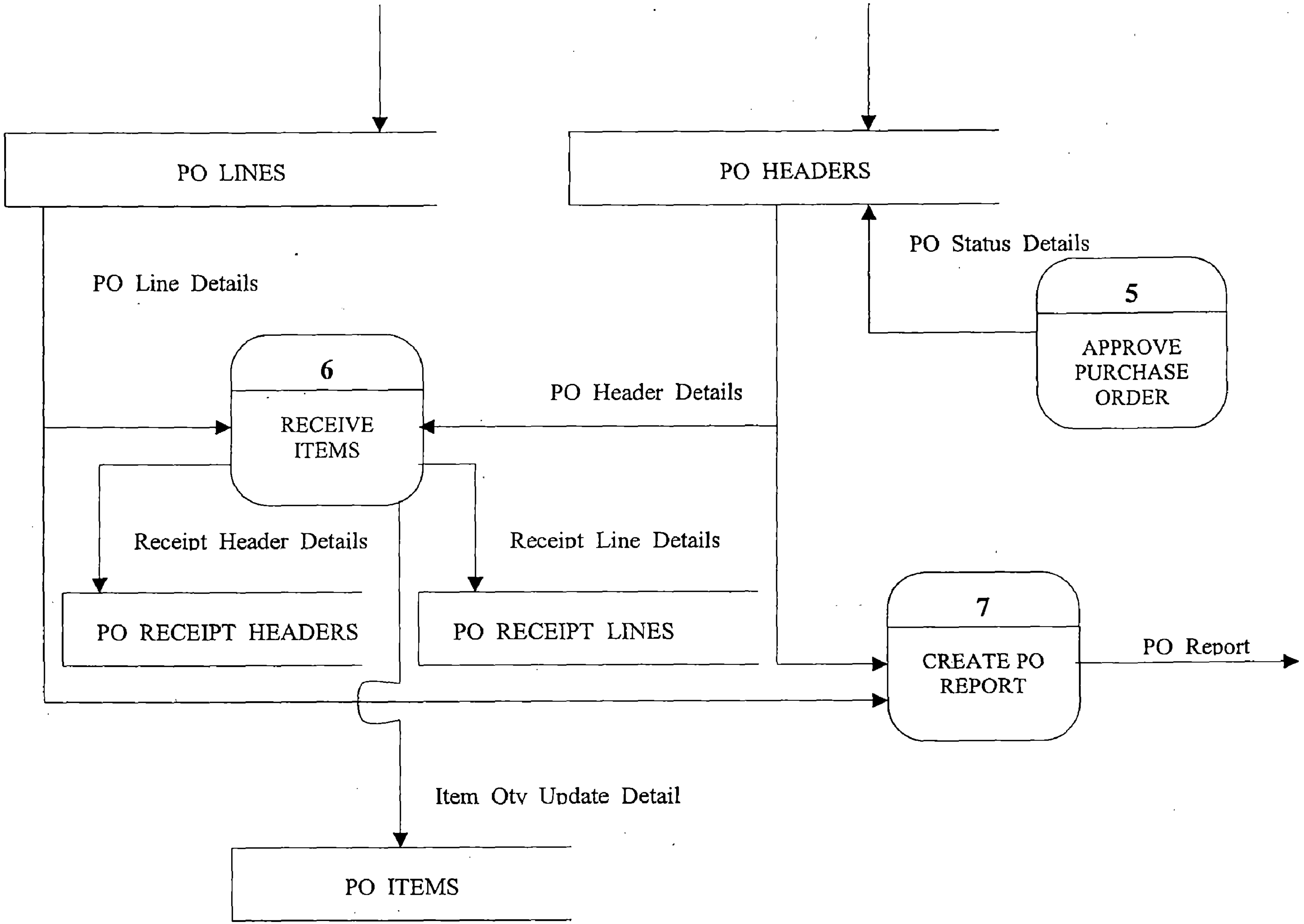
Context level 0 Data Flow Diagrams

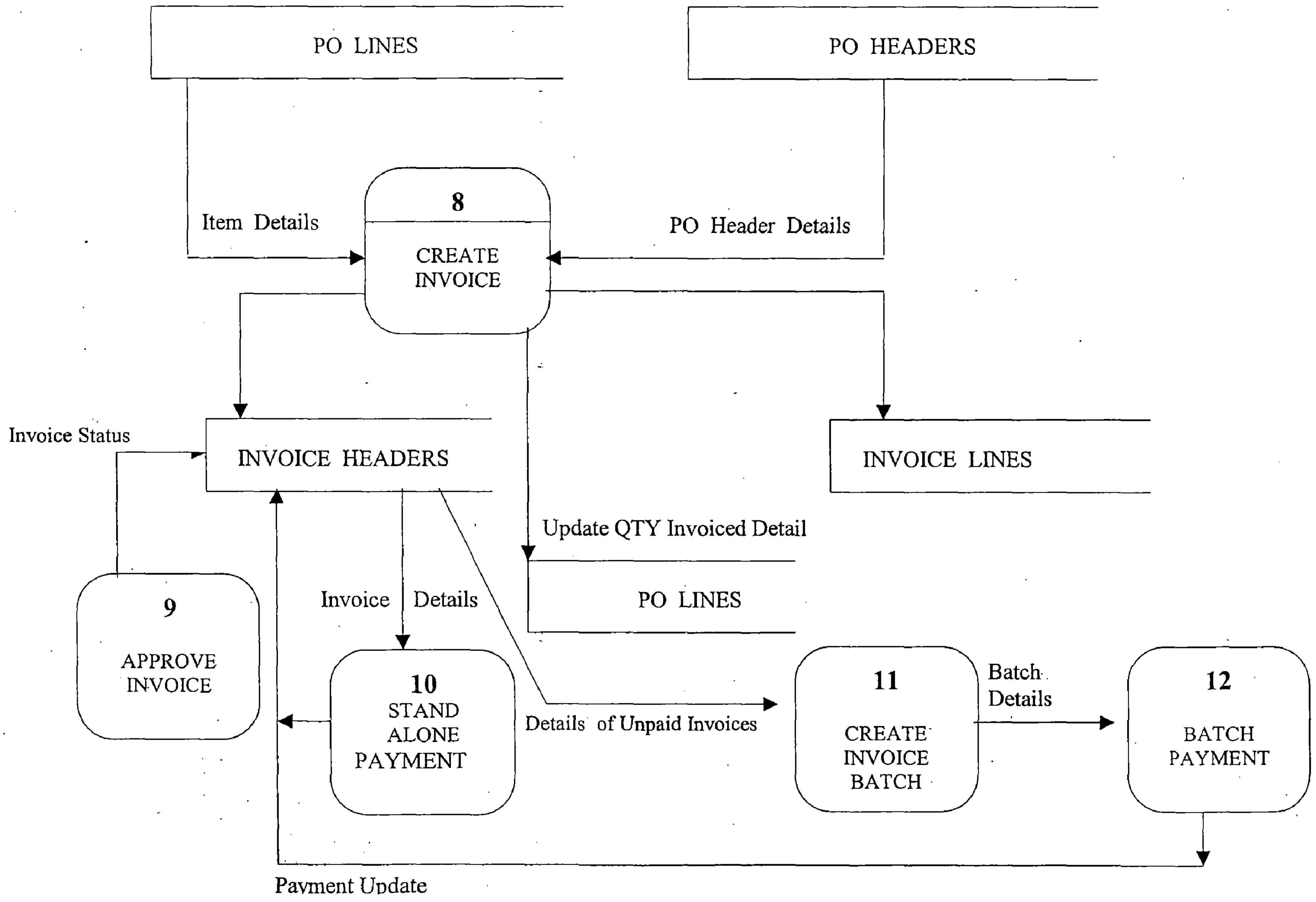


Context level 0 Data Flow Diagrams



G11001
CENTRAL LIBRARY
Acc. No.........
Date..............
I.I.T. ROORKEE


PO LINES
PO HEADERS
PO Status Details
5
APPROVE PURCHASE ORDER
PO Line Details
6
RECEIVE ITEMS
PO Header Details
Receipt Header Details
Receipt Line Details
PO RECEIPT HEADERS
PO RECEIPT LINES
7
CREATE PO REPORT
PO Report
Item Qty Update Detail
PO ITEMS


PO LINES
PO HEADERS
Item Details
8
CREATE INVOICE
PO Header Details
Invoice Status
INVOICE HEADERS
INVOICE LINES
Update QTY Invoiced Detail
9
APPROVE INVOICE
Invoice Details
PO LINES
10
STAND ALONE PAYMENT
Details of Unpaid Invoices
11
CREATE INVOICE BATCH
Batch Details
12
BATCH PAYMENT
Payment Update

4.5 Justification Of Development Methodology

The methodology used for the development of this project is Bottom-Up Approach. In this approach, the design starts from the bottom, with an implementation strategy laid out first, and the capabilities of the system depend heavily on this strategy.

The Requirement Specification was given first and then based on the Requirements , I decided the number of Forms needed and the fields to be used in the form. For this specification the Bottom-Up approach would be the most suitable.

4.6 Data Base Design

The results of the extensive system study conducted were used as the input for the database design. A well-designed database is essential for the performance of the system. Several tables are manipulated for varying purpose. The table, also known as relation, gives the information of attributes regarding the specific entities. Normalizing of tables is done to the extent possible. While normalizing tables, care is taken to see that the number of tables is limited to an optimum level so that table maintenance is convenient and efficient.

DATA BASE TABLES

PO_REQUISITION_HEADERS

PO_REQUISITION_HEADER_ID	NUMBER(38)	
REQUISITION_NUMBER	VARCHAR2(20)	NOT NULL
PREPARER_ID	NUMBER(38)	
REQUISITION_DESCRIPTION	VARCHAR2(250)	
STATUS	VARCHAR2(25)	NOT NULL,
STATUS_DATE	DATE	
REQUISITION_DATE	DATE	NOT NULL
ATTRIBUTE_CATEGORY	VARCHAR2(30)	
ATTRIBUTE1-15	VARCHAR2(150)	
CREATED_BY	NUMBER(15)	
CREATION_DATE	DATE	
LAST_UPDATE_BY	NUMBER(15)	NOT NULL
LAST_UPDATE_DATE	DATE	NOT NULL
LAST_UPDATE_LOGIN	NUMBER(15)	

PO_REQUISITION_LINES

PO_REQUISITION_HEADER_ID	NUMBER(38)	
PO_REQUISITION_LINE_ID	NUMBER(38)	NOT NULL
LINE_NUMBER	NUMBER(20)	NOT NULL
ITEM_ID	NUMBER(38)	NOT NULL
UOM	VARCHAR2(10)	
UNIT_PRICE	NUMBER(25,2)	
SUPPLIER_ID	NUMBER(38),	
REQUESTOR_ID	NUMBER(38)	NOT NULL
DATE_REQUIRED	DATE	
QUANTITY	NUMBER(22)	NOT NULL
PO_QUANTITY_ORDERED	NUMBER(22)	
REQ_QUANTITY_CANCELLED	NUMBER(22)	
AMOUNT	NUMBER(25,2)	
STATUS	VARCHAR2(25)	
DESTINATION_TYPE	VARCHAR2(10)	
WAREHOUSE_ID	NUMBER(38)	
ATTRIBUTE_CATEGORY	VARCHAR2(30)	
ATTRIBUTE1-15	VARCHAR2(150)	
CREATED_BY	NUMBER(15)	
CREATION_DATE	DATE	
LAST_UPDATE_BY	NUMBER(15)	NOT NULL
LAST_UPDATE_DATE	DATE	NOT NULL
LAST_UPDATE_LOGIN	NUMBER(15)	

PO_HEADERS

PO_HEADER_ID	NUMBER(38)	NOT NULL
PO_DATE	DATE	NOT NULL
PO_NUMBER	VARCHAR2(20)	NOT NULL
PO_REQUISITION_HEADER_ID	NUMBER(38)	
BILL_TO_LOCATION_ID	NUMBER(38)	NOT NULL
BUYER_ID	NUMBER(38)	NOT NULL
SUPPLIER_ID	NUMBER(38)	NOT NULL
PAYMENT_TERMS	VARCHAR2(250)	
FRIEGHT_TERMS	VARCHAR2(250)	
FRIEGHT_CARRIER	VARCHAR2(100)	
STATUS	VARCHAR2(25)	NOT NULL
STATUS_DATE	DATE	
ATTRIBUTE_CATEGORY	VARCHAR2(30)	
ATTRIBUTE1-15	VARCHAR2(150)	
CREATED_BY	NUMBER(15)	
CREATION_DATE	DATE	
LAST_UPDATE_BY	NUMBER(15)	NOT NULL
LAST_UPDATE_DATE	DATE	NOT NULL
LAST_UPDATE_LOGIN	NUMBER(15)	

PO_LINES

PO_LINE_ID	NUMBER(38)	NOT NULL
LINE_NUMBER	NUMBER(20)	NOT NULL
PO_HEADER_ID	NUMBER(38)	NOT NULL
PO_REQUISITION_HEADER_ID	NUMBER(38)	
PO_REQUISITION_LINE_ID	NUMBER(38)	
ITEM_ID	NUMBER(38)	NOT NULL
UOM	VARCHAR2(10)	
UNIT_PRICE	NUMBER(25,2)	
AMOUNT	NUMBER(25,2)	
REQUESTOR_ID	NUMBER(38)	NOT NULL
SHIP_TO_LOCATION_ID	NUMBER(38)	NOT NULL
QUANTITY_ORDERED	NUMBER(22)	
QUANTITY_RECEIVED	NUMBER(22)	
QUANTITY_CANCELLED	NUMBER(22)	
STATUS	VARCHAR2(25)	
DATE_PROMISED	DATE	
DATE_REQUIRED	DATE	
TOLERANCE_QTY_ABOVE	NUMBER(6)	
TOLERANCE_QTY_BELOW	NUMBER(6)	
TOLERANCE_DAYS_ABOVE	NUMBER(3)	
TOLERANCE_DAYS_BELOW	NUMBER(3)	
ATTRIBUTE_CATEGORY	VARCHAR2(30)	
ATTRIBUTE1-15	VARCHAR2(150)	
CREATED_BY	NUMBER(15)	
CREATION_DATE	DATE,	
LAST_UPDATE_BY	NUMBER(15)	NOT NULL

LAST_UPDATE_DATE	DATE	NOT NULL
LAST_UPDATE_LOGIN	NUMBER(15)	

PO_RECEIPT_HEADERS

PO_RECEIPT_HEADER_ID	NUMBER(38)	NOT NULL
RECEIPT_NUMBER	NUMBER(20)	NOT NULL
RECEIPT_DATE	DATE	NOT NULL
SUPPLIER_ID	NUMBER(38)	NOT NULL
SHIPPED_DATE	DATE	
FRIEGHT_CARRIER	VARCHAR2(100)	
PACKING_SLIP	VARCHAR2(70)	
WAYBILL	VARCHAR2(70)	
CONTAINER_INFORMATION	VARCHAR2(100)	
ATTRIBUTE_CATEGORY	VARCHAR2(30)	
ATTRIBUTE1-15	VARCHAR2(150)	
CREATED_BY	NUMBER(15)	
CREATION_DATE	DATE	
LAST_UPDATE_BY	NUMBER(15)	NOT NULL
LAST_UPDATE_DATE	DATE	NOT NULL
LAST_UPDATE_LOGIN	NUMBER(15)	

PO_RECEIPTI_LINES

PO_RECEIPT_LINE_ID	NUMBER(38)	NOT NULL
LINE_NUMBER	NUMBER(20)	NOT NULL
PO_RECEIPT_HEADER_ID	NUMBER(38)	NOT NULL
PO_HEADER_ID	NUMBER(38)	NOT NULL
PO_LINE_ID	NUMBER(38)	NOT NULL
ITEM_ID	NUMBER(38)	NOT NULL
UOM	VARCHAR2(10)	
REQUESTOR_ID	NUMBER(38)	NOT NULL
SHIP_TO_LOCATION_ID	NUMBER(38)	NOT NULL
QUANTITY_RECEIVED	NUMBER(22)	
QUANTITY_ACCEPTED	NUMBER(22)	
QUANTITY_REJECTED	NUMBER(22)	
ATTRIBUTE_CATEGORY	VARCHAR2(30)	
ATTRIBUTE1-15	VARCHAR2(150)	
CREATED_BY	NUMBER(15)	
CREATION_DATE	DATE	
LAST_UPDATE_BY	NUMBER(15)	NOT NULL
LAST_UPDATE_DATE	DATE	NOT NULL
LAST_UPDATE_LOGIN	NUMBER(15)	

ITEM_LIST

ITEM_DESCRIPTION	VARCHAR2(250)
ITEM_TYPE	VARCHAR2(10)
UNIT_PRICE	NUMBER(10,2)
UOM	VARCHAR2(10)

ATTRIBUTE_CATEGORY	VARCHAR2(30)	
ATTRIBUTE1 - 15	VARCHAR2(150)	
CREATED_BY	NUMBER(15)	
CREATION_DATE	DATE	
LAST_UPDATE_BY	NUMBER(15)	NOT NULL
LAST_UPDATE_DATE	DATE	NOT NULL
LAST_UPDATE_LOGIN	NUMBER(15)	

SUPPLIER

SUPPLIER_ID	NUMBER(6)	NOT NULL
SUPPLIER_NAME	VARCHAR2(30)	
SUPPLIER_SITE	VARCHAR2(30)	
CONTACT	VARCHAR2(30)	
PHONE	VARCHAR2(30)	
CREATED_BY	NUMBER(15)	
CREATION_DATE	DATE	
LAST_UPDATE_BY	NUMBER(15)	NOT NULL
LAST_UPDATE_DATE	DATE	NOT NULL
LAST_UPDATE_LOGIN	NUMBER(15)	

PO_SUPPLIER_ITEM

SUPPLIER_ID	NUMBER(6)	NOT NULL
ITEM_ID	NUMBER(38)	NOT NULL
UNIT_PRICE	NUMBER(10,2)	
SUPPLIER_ITEM_NAME	VARCHAR2(20)	
SUPPLIER_ITEM_CODE	VARCHAR2(10)	
SUPP LIER_ITEM_DESCRIPTION	VARCHAR2(50)	
CREATED_BY	NUMBER(15)	
CREATION_DATE	DATE	
LAST_UPDATE_BY	NUMBER(15)	NOT NULL
LAST_UPDATE_DATE	DATE	NOT NULL
LAST_UPDATE_LOGIN	NUMBER(15)	

LOCATION_LIST

LOCATION_ID	NUMBER(6)	NOT NULL
LOCATION_NAME	VARCHAR2(20)	NOT NULL
LOCATION_TYPE	VARCHAR2(20)	
CREATED_BY	NUMBER(15)	
CREATION_DATE	DATE	
LAST_UPDATE_BY	NUMBER(15)	NOT NULL
LAST_UPDATE_DATE	DATE	NOT NULL
LAST_UPDATE_LOGIN	NUMBER(15)	

EMPLOYEE

EMP_ID	NUMBER(6)	NOT NULL
ENAME	VARCHAR2(30)	
DESIGNATION	VARCHAR2(20)	

ATTRIBUTE_CATEGORY	VARCHAR2(30)	
ATTRIBUTE1 - 15	VARCHAR2(150)	
CREATED_BY	NUMBER(15)	
CREATION_DATE	DATE	
LAST_UPDATE_BY	NUMBER(15)	NOT NULL
LAST_UPDATE_DATE	DATE	NOT NULL
LAST_UPDATE_LOGIN	NUMBER(15)	

PO_EMPLOYEE_ROLE

EMP_ID	NUMBER(6)	NOT NULL
ROLE_NAME	VARCHAR2(20)	NOT NULL
CREATED_BY	NUMBER(15)	
CREATION_DATE	DATE	
LAST_UPDATE_BY	NUMBER(15)	NOT NULL
LAST_UPDATE_DATE	DATE	NOT NULL
LAST_UPDATE_LOGIN	NUMBER(15)	

PO_INVOICE_HEADERS

INVOICE_ID	NUMBER(38)
INVOICE_NUMBER	VARCHAR2(20)
INVOICE_DATE	DATE
SUPPLIER_ID	NUMBER(38)
INVOICE_AMOUNT	NUMBER(25,2)
TERMS	VARCHAR2(250)
STATUS	VARCHAR2(20)
BANK _ID	NUMBER(38)
ATTRIBUTE_CATEGORY	VARCHAR2(30)
ATTRIBUTE1-15	VARCHAR2(150)
CREATED_BY	NUMBER(15)
CREATION_DATE	DATE
LAST_UPDATE_BY	NUMBER(15)
LAST_UPDATE_DATE	DATE
LAST_UPDATE_LOGIN	NUMBER(15)

PO INVOICE LINES

INVOICE_LINE_ID	NUMBER(38)
LINE_NUMBER	NUMBER(20)
INVOICE_HEADER_ID	NUMBER(38)
SPILT_UP_AMOUNT	NUMBER(25,2)
PO_HEADER_ID	NUMBER(38)
PO_LINE_ID	NUMBER(38)
QUANTITY_INVOICED	NUMBER(22)
CREATED_BY	NUMBER(15)
CREATION_DATE	DATE
LAST_UPDATE_BY	NUMBER(15)
LAST_UPDATE_DATE	DATE
LAST_UPDATE_LOGIN	NUMBER(15)

PO BANK HEADERS

BANK_ID	NUMBER (38)
BANK_NAME	VARCHAR2(30)
ADDRESS	VARCHAR2(30)
CREATED_BY	NUMBER(15)
CREATION_DATE	DATE
LAST_UPDATE_BY	NUMBER(15)
LAST_UPDATE_DATE	DATE
LAST_UPDATE_LOGIN	NUMBER(15)

PO BANK LINES

BANK_ID	NUMBER(38)
BANK_LINE_ID	NUMBER(38)
LINE_NUMBER	NUMBER(20)
PAYMENT_MODE	VARCHAR2(20)
PAYMENT_DOCUMENT	VARCHAR2(30)
MIN_CHEQUE_NUMBER	NUMBER(10)
MAX_CHEQUE_NUMBER	NUMBER(10)
CURRENT_CHEQUE_NUMBER	NUMBER(10)
CREATED_BY	NUMBER(15)
CREATION_DATE	DATE
LAST_UPDATE_BY	NUMBER(15)
LAST_UPDATE_DATE	DATE
LAST_UPDATE_LOGIN	NUMBER(15)

PO PAYGROUPS

PAYGROUP_ID	NUMBER(38)
PAYGROUP_NAME	VARCHAR2(20)
DESCRITION	VARCHAR2(250)
CREATED_BY	NUMBER(15)
CREATION_DATE	DATE
LAST_UPDATE_BY	NUMBER(15)

LAST_UPDATE_DATE	DATE
LAST_UPDATE_LOGIN	NUMBER(15)

VIEWS

Views are necessary to aggregate information from various tables. All Forms are based on Views. Since, I have ID's in most tables which are like foreign keys to other tables so, to retrieve the complete information we need Views.

So for most of the above mentioned tables Views were created :

PO_REQUISITION_HEADER_V

```
 SELECT

      REQUISITION_ID
      REQUISITION_NUMBER
      E.EMP_ID            EMP_ID
      E.ENAME             ENAME
      REQUISITION_DESCRIPTION
      STATUS
      REQUI SITION_DATE
      PRH.ATTRIBUTE_CATEGORY          ATTRIBUTE_CATEGORY
      PRH.ATTRIBUTE1 - 15             ATTRIBUTE1 - 15
      PRH.CREATED_BY                  CREATED_BY
      PRH.CREATION_DATE                CREATION_DATE
      PRH.LAST_UPDATE_BY              LAST_UPDATE_BY
      PRH.LAST_UPDATE_DATE            LAST_UPDATE_DATE
      PRH.LAST_UPDATE_LOGIN           LAST_UPDATE_LOGIN

 FROM

      PO_REQUISITION_HEADER   PRH,
      EMPLOYEE   E

WHERE
      PRH.EMP_ID = E.EMP_ID
```

PO REQUISITION LINE V

```
    SELECT
            PRH.REQUISITION_ID      REQUISITION_ID
            SI.ITEM_ID              ITEM_ID
            SI.ITEM_DESCRIPTION     ITEM_DESCRIPTION
            LINE_NUMBER
            S.SUPPLIER_ID           SUPPLIER_ID
            S.SUPPLIER_NAME         SUPPLIER_NAME
            S.SUPPLIER_SITE         SUPPLIER_SITE
            S.CONTACT               CONTACT
            S.PHONE                 PHONE
            E.EMP_ID                EMP_ID
            E.ENAME                 ENAME
            DATE_REQUIRED
            QUANTITY
            DESTINATION_TYPE
            LL.LOCATION_ID(WAREHOUSE)
            LOCATION_NAME
            SI.UNIT_PRICE                UNIT_PRICE
            IRES.ATTRIBUTE_CATEGORY      ATTRIBUTE_CATEGORY
            IRES.ATTRIBUTE1 - 15         ATTRIBUTE1 - 15
            IRES.CREATED_BY               CREATED_BY
            IRES.CREATION_DATE           CREATION_DATE
            IRES.LAST_UPDATE_BY           LAST_UPDATE_BY
            IRES.LAST_UPDATE_DATE         LAST_UPDATE_DATE
            IRES.LAST_UPDATE_LOGIN        LAST_UPDATE_LOGIN

            FROM
            PO_REQUISITION_HEADER  PRH,
            PO_REQUISITION_LINES  IRES,
            SUPPLIER  S,
            PO_SUPPLIER_ITEM  SI,
            EMPLOYEE  E,
            ITEM_LIST  IL,
            LOCATION_LIST  LL

      WHERE
            PRH.REQUISITION_ID = IRES. REQUISITION_ID  AND
            IRES.ITEM_ID  =  IL.ITEM_ID  AND
            IRES.EMP_ID  =  E.EMP_ID  AND
            IRES.SUPPLIER_ID  =  S.SUPPLIER_ID  AND
            IRES.SUPPLIER_ID  = SI.SUPPLIER_ID  AND
            IRES.ITEM_ID  =  SI.ITEM_ID  AND
            IRES.LOCATION_ID  =  LL.LOCATION_ID
```

PO_HEADER_V

```
 SELECT

      PO_ID
      PO_DATE
      PO_NUMBER
      LL1.SHIP_TO_LOCATION_ID  SHIP_TO_LOCATION_ID
      LL1.LOCATION_NAME        SHIP_TO_ LOCATION_NAME
      LL2.BILL_TO_LOCATION_ID  BILL_TO_LOCATION_ID
      LL2.LOCATION_NAME        BILL_TO_ LOCATION_NAME
      E.EMP_ID
      E.ENAME
      PAYMENT_TERMS
      FRIEGHT_TERMS
      FRIEGHT_CARRIER
      PH.ATTRIBUTE_CATEGORY    ATTRIBUTE_CATEGORY
      PH.ATTRIBUTE1 - 15       ATTRIBUTE1 - 15
      PH.CREATED_BY            CREATED_BY
      PH.CREATION_DATE         CREATION_DATE
      PH.LAST_UPDATE_BY        LAST_UPDATE_BY
      PH.LAST_UPDATE_DATE      LAST_UPDATE_DATE
      PH.LAST_UPDATE_LOGIN     LAST_UPDATE_LOGIN

FROM
      PO_HEADERS   PH,
      EMPLOYEE   E,
      LOCATION_LIST  LL1,
      LOCATION_LIST  LL2

WHERE
      PH.EMP_ID = E.EMP_ID   AND
      PH.SHIP_TO_LOCATION_ID = LL1.LOCATION_ID  AND
      PH.BILL_TO_LOCATION_ID = LL2.LOCATION_ID
```

PO_LINE_V

```
      SELECT
      PH.PO_ID
      IL.ITEM_ID               ITEM_ID
      IL.ITEM_DESCRIPTION      ITEM_DESCRIPTION
      IL.ITEM_TYPE             ITEM_TYPE
      IL.UOM                   UOM
      E.EMP_ID                 EMP_ID
      E.ENAME                  ENAME
```

```
    QUANTITY_ORDERED
    QUANTITY_RECEIVED
    DATE_PROMISED
    DATE_REQUIRED
    TOLERANCE_QTY_ABOVE
    TOLERANCE_QTY_BELOW
    TOLERANCE_DAYS_ABOVE
    TOLERANCE_DAYS_BELOW
    PIE.ATTRIBUTE_CATEGORY      ATTRIBUTE_CATEGORY
    PIE.ATTRIBUTE1 - 15         ATTRIBUTE1 - 15
    PIE.CREATED_BY              CREATED_BY
    PIE.CREATION_DATE           CREATION_DATE
    PIE.LAST_UPDATE_BY          LAST_UPDATE_BY
    PIE.LAST_UPDATE_DATE        LAST_UPDATE_DATE
    PIE.LAST_UPDATE_LOGIN       LAST_UPDATE_LOGIN

    FROM
         PO_LINES   PIE,
         PO_HEADERS  PH,
         EMPLOYEE    E,
         ITEM_LIST     IL

    WHERE
         PH.PO_ID  =  PIE.PO_ID   AND
         PH.EMP_ID  = E. EMP_ID  AND
         PH.ITEM_ID = IL.ITEM_ID
```

PO_RECEIPT_HEADER_V

```
SELECT
     PRH.ROWID                              ROW_ID,
    PO_RECEIPT_HEADER_ID,
    RECEIPT_NUMBER,
    RECEIPT_DATE,
    SHIPPED_DATE,
    S.SUPPLIER_ID,
    S.SUPPLIER_NAME,
    FRIEGHT_CARRIER,
    PACKING_SLIP,
    WAYBILL,
    CONTAINER_INFORMATION,
    ATTRIBUTE_CATEGORY,
    ATTRIBUTE1                              ATTRIBUTE1,
    ATTRIBUTE2                              ATTRIBUTE2,
    ATTRIBUTE3                              ATTRIBUTE3,
    ATTRIBUTE4                              ATTRIBUTE4,
    ATTRIBUTE5                              ATTRIBUTE5,
          ATTRIBUTE6                              ATTRIBUTE6,
          ATTRIBUTE7                              ATTRIBUTE7,
          ATTRIBUTE8                              ATTRIBUTE8,
          ATTRIBUTE9                              ATTRIBUTE9,
          ATTRIBUTE10                             ATTRIBUTE10,
          ATTRIBUTE11                             ATTRIBUTE11,
          ATTRIBUTE12                             ATTRIBUTE12,
          ATTRIBUTE13                             ATTRIBUTE13,
          ATTRIBUTE14                             ATTRIBUTE14,
          ATTRIBUTE15                             ATTRIBUTE15,
          PRH.CREATED_BY                          CREATED_BY,
          PRH.CREATION_DATE                       CREATION_DATE,
          PRH.LAST_UPDATE_BY                      LAST_UPDATE_BY,
          PRH.LAST_UPDATE_DATE                    LAST_UPDATE_DATE,
          PRH.LAST_UPDATE_LOGIN                   LAST_UPDATE_LOGIN
FROM
        PO_RECEIPT_HEADERS  PRH,
        SUPPLIER        S

WHERE
       PRH.SUPPLIER_ID = S.SUPPLIER_ID
```

PO_RECEIPT_LINE_V

```
SELECT
         PO_RECEIPT_LINE_ID                       PO_RECEIPT_LINE_ID,
         PRL.ROWID                                ROW_ID,
         IL.ITEM_ID                               ITEM_ID,
         PRH.PO_RECEIPT_HEADER_ID                 RECEIPT_HEADER_ID,
         PH.PO_HEADER_ID                          PO_HEADER_ID,
         PH.PO_NUMBER                             PO_NUMBER,
         PL.PO_LINE_ID                            PO_LINE_ID,
         PL.LINE_NUMBER                           PO_LINE_NUMBER,
         IL.ITEM_DESCRIPTION                      ITEM_DESCRIPTION,

         IL.UOM                                   UOM,
         E.EMP_ID                                 REQUESTOR_ID,
         E.ENAME                                  REQUESTOR_NAME,

         LL.LOCATION_ID                           WAREHOUSE_ID,
          LL.LOCATION_NAME                        WAREHOUSE,
         PRL.QUANTITY_RECEIVED                    QUANTITY_RECEIVED,
         PRL.QUANTITY_ACCEPTED                    QUANTITY_ACCEPTED,
         PRL.QUANTITY_REJECTED                    QUANTITY_REJECTED,
         PRL.ATTRIBUTE_CATEGORY                   ATTRIBUTE_CATEGORY,
         PRL.ATTRIBUTE1                           ATTRIBUTE1,
         PRL.ATTRIBUTE2                           ATTRIBUTE2,
        PRL.ATTRIBUTE3                  ATTRIBUTE3,
        PRL.ATTRIBUTE4                  ATTRIBUTE4,
        PRL.ATTRIBUTE5                  ATTRIBUTE5,
        PRL.ATTRIBUTE6                  ATTRIBUTE6,
        PRL.ATTRIBUTE7                  ATTRIBUTE7,
        PRL.ATTRIBUTE8                  ATTRIBUTE8,
        PRL.ATTRIBUTE9                  ATTRIBUTE9,
        PRL.ATTRIBUTE10                 ATTRIBUTE10,
        PRL.ATTRIBUTE11                 ATTRIBUTE11,
        PRL.ATTRIBUTE12                 ATTRIBUTE12,
        PRL.ATTRIBUTE13                 ATTRIBUTE13,
        PRL.ATTRIBUTE14                 ATTRIBUTE14,
        PRL.ATTRIBUTE15                 ATTRIBUTE15,
        PRL.CREATED_BY                  CREATED_BY,
        PRL.CREATION_DATE               CREATION_DATE,
        PRL.LAST_UPDATE_BY              LAST_UPDATE_BY,
        PRL.LAST_UPDATE_DATE            LAST_UPDATE_DATE,
        PRL.LAST_UPDATE_LOGIN           LAST_UPDATE_LOGIN,
        PRL.PO_REQUISITION_HEADER_ID    PO_REQUISITION_HEADER_ID,
        PRL.PO_REQUISITION_LINE_ID      PO_REQUISITION_LINE_ID
FROM
        PO_RECEIPT_HEADERS  PRH,
        PO_RECEIPT_LINES  PRL,
        PO_HEADERS   PH,
        PO_LINES    PL,
        EMPLOYEE     E,
        ABC_ITEMS       IL,
        LOCATION_LIST  LL
WHERE
        PH.PO_HEADER_ID  =  PRL.PO_HEADER_ID   AND
        PRH.PO_RECEIPT_HEADER_ID  =  PRL.PO_RECEIPT_HEADER_ID  AND
        PL.REQUESTOR_ID  = E. EMP_ID  AND
        PL.ITEM_ID = IL.ITEM_ID  AND
        LL.LOCATION_ID  =  PL.SHIP_TO_LOCATION_ID  AND
        PRL.PO_LINE_ID =  PL. PO_LINE_ID
```

INVOICE_HEADER_V

```
SELECT
        IH.ROWID ROW_ID,
        INVOICE_ID,
        INVOICE_NUMBER,
        INVOICE_DATE,
        IH.SUPPLIER_ID                  SUPPLIER_ID,
        S.SUPPLIER_NAME                 SUPPLIER_NAME,
        S.SUPPLIER_SITE                 SUPPLIER_SITE,
        S.CONTACT                       CONTACT,
        S.PHONE                        PHONE,
        S.PAYGROUP_ID                   PAYGROUP_ID,
        PG PAYGROUP_NAME               PAYGROUP_NAME,
        PG.DESCRIPTION                 DESCRIPTION,
        IH.INVOICE_AMOUNT              INVOICE_AMOUNT,
        IH.BANK_ID                     BANK_ID,
        IH.TERMS                       TERMS,
        IH.STATUS                      STATUS,
        IH.ATTRIBUTE_CATEGORY          ATTRIBUTE_CATEGORY,
        IH.ATTRIBUTE1                  ATTRIBUTE1 ,
        IH.ATTRIBUTE2                  ATTRIBUTE2 ,
        IH.ATTRIBUTE3                  ATTRIBUTE3 ,
        IH.ATTRIBUTE4                  ATTRIBUTE4 ,
        IH.ATTRIBUTE5                  ATTRIBUTE5 ,
        IH.ATTRIBUTE6                  ATTRIBUTE6 ,
        IH.ATTRIBUTE7                  ATTRIBUTE7 ,
        IH.ATTRIBUTE8                  ATTRIBUTE8 ,
        IH.ATTRIBUTE9                  ATTRIBUTE9 ,
        IH.ATTRIBUTE10                 ATTRIBUTE10 ,
        IH.ATTRIBUTE11                 ATTRIBUTE11 ,
        IH.ATTRIBUTE12                 ATTRIBUTE12 ,
        IH.ATTRIBUTE13                 ATTRIBUTE13 ,
        IH.ATTRIBUTE14                 ATTRIBUTE14 ,
        IH.ATTRIBUTE15                 ATTRIBUTE15 ,
        IH.CREATED_BY                  CREATED_BY,
        IH.CREATION_DATE               CREATION_DATE,
        IH.LAST_UPDATE_BY              LAST_UPDATE_BY,
        IH.LAST_UPDATE_DATE            LAST_UPDATE_DATE,
        IH.LAST_UPDATE_LOGIN           LAST_UPDATE_LOGIN
FROM
        INVOICE_HEADERS IH,
        SUPPLIER S,
        PAYGROUP PG
WHERE
        IH.SUPPLIER_ID = S.SUPPLIER_ID
         AND S.PAYGROUP_ID = PG.PAYGROUP_ID
```

INVOICE_LINE_V

```
SELECT
        IL.ROWID  ROW_ID,
        INVOICE_HEADER_ID,
        INVOICE_LINE_ID,
        IL.LINE_NUMBER              LINE_NUMBER,
        IH.INVOICE_NUMBER           INVOICE_NUMBER,
        SPLIT_AMOUT,
        PL.PO_HEADER_ID            PO_HEADER_ID,
        PH.PO_NUMBER                PO_NUMBER,
        PH.PO_DATE               PO_DATE,
        PL.PO_LINE_ID            PO_LINE_ID,
        PL.LINE_NUMBER           PO_LINE_NUMBER,
        PL.ITEM_ID               ITEM_ID,
        PL.UOM                   UOM,
        PL.UNIT_PRICE            UNIT_PRICE,
        PL.REQUESTOR_ID          REQUESTOR_ID,
        E.ENAME                  REQUESTOR_NAME,
        IL.QUANTITY_INVOICED QUANTITY_INVOICED,
        IL.CREATED_BY            CREATED_BY,
        IL.CREATION_DATE         CREATION_DATE,
        IL.LAST_UPDATE_BY         LAST_UPDATE_BY,
        IL.LAST_UPDATE_DATE        LAST_UPDATE_DATE,
        IL.LAST_UPDATE_LOGIN        LAST_UPDATE_LOGIN
    FROM
        INVOICE_LINES IL,
        INVOICE_HEADERS IH,
         PO_LINES PL,
         PO_HEADERS PH,
         EMPLOYEE E
   WHERE
            IL.PO_HEADER_ID = PH.PO_HEADER_ID AND
            IL.PO_HEADER_ID = PL.PO_HEADER_ID AND
            IL.PO_LINE_ID   = PL.PO_LINE_ID AND
            IL.INVOICE_HEADER_ID = IH.INVOICE_ID AND
            PL.REQUESTOR_ID = E.EMP_ID
```

SOFTWARE IMPLEMENTATION AND TESTING

The Application Development Process includes :

- **Forms Coding**
- **Setting Up the Directory Structure :**

  Create separate directory structures on the forms and the database server.

  Define environment variables so the applications recognize our directories.

- **Registering the Application**

  Defining the application's user–friendly name and short name.

  Providing the base directory path for the application.

- **Registering the ORACLE Schema**

  Provide the application with a database password.

  Integrate the schema with ORACLE Applications APPS schema.

- **Registering tables with ORACLE Applications**
- **Creating Users**
- **Creating Profiles**
- **Creating responsibilities**

- **Defining Flexfields**
- **Registering Tables**
- **Set Up Concurrent Managers**
- **Registering Forms**
- **Defining Functions**
- **Defining Menus**
- **Defining concurrent managers to run immediate programs**

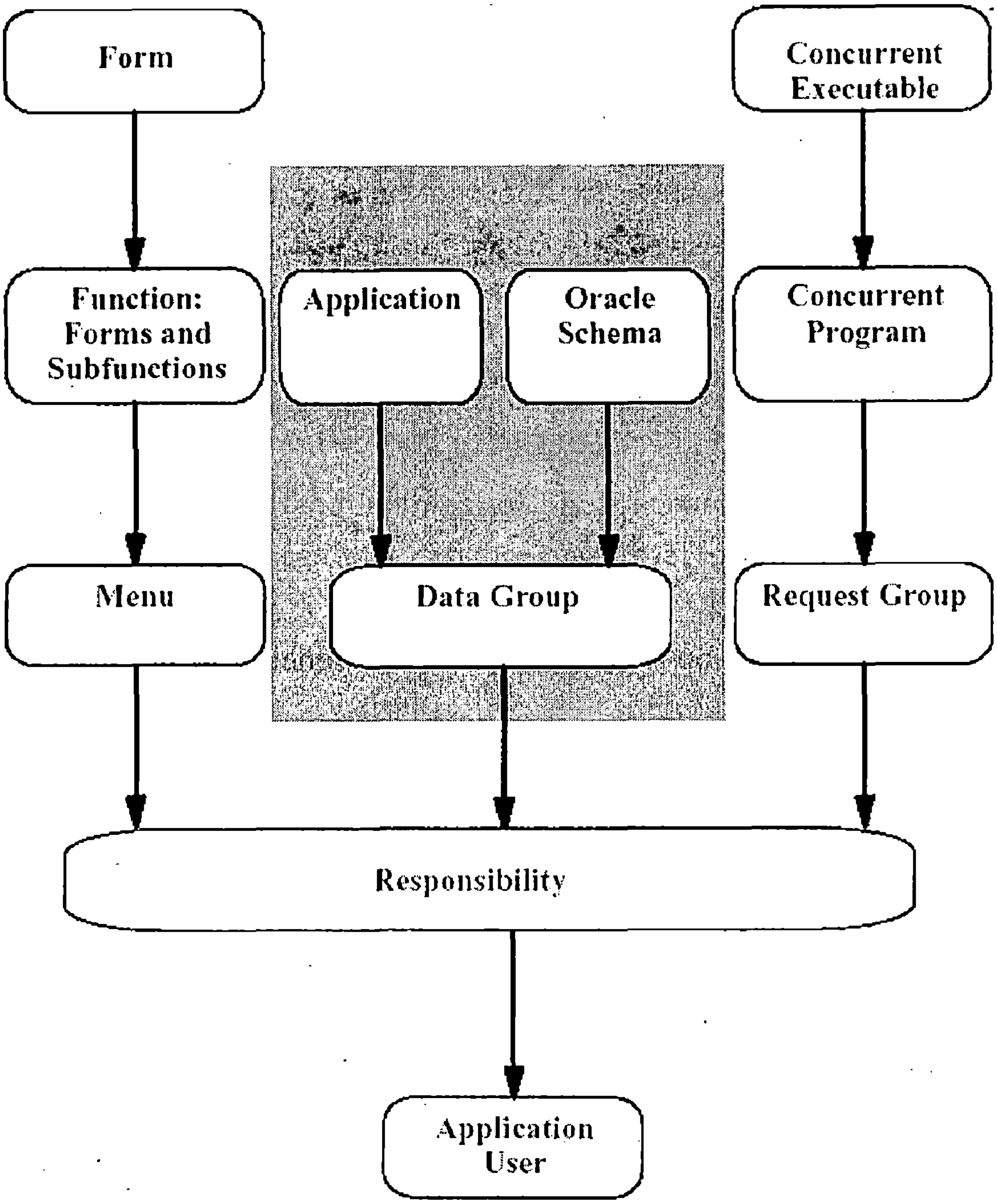


Figure 5.1 APPLICATION DEVELOPMENT PROCESS

I'll discuss each of these development steps in detail :

5.1 Development of Forms:

Forms were developed using Developer 2000.

Developer/2000 is an Oracle tool that helps to create forms and reports based on the tables that are created using Designer/2000. Developer 2K can be used to :

- Design and customize forms and reports.
- Add various functionality, like radio buttons, combo boxes, and list of values to make forms and reports more user friendly.
- Write triggers on objects to add functionality to them and capture errors.

Tools Provided By Oracle Developer/2000

Oracle Developer/2000 provides four tools:

- Object Navigator:

  In this tool we can view all the objects, add new objects and name/rename the objects.

- Layout Editor:

  This tool helps to design forms and reports and add various objects to them like push buttons and list boxes.

- PL-/SQL Editor:

  This is the tool that is used to write all the codes for the triggers, procedures or functions.

IMPLEMENTATION DETAILS

The two modules are :

1.Purchasing

2.Payables

Forms were built for each class of functionality. Coding is done only in triggers .

Triggers are at 3 levels:

Form level trigger ->like pre-form , post-form, when-new-form-instance.

Block level trigger ->like pre-block,when-new-block-intance,when -new record - instance ,when-block-change,when-validate-record etc

Item level trigger-> like when-new-item-instance,when-validate-item,key-next-item etc.

The listings of forms are:

1. Requisit.fmb -> Used to build Requisition interface, the snap shot is shown in later section.
2. Reqsum.fmb -> Used to build Requisition Summary interface, the user can search based on header details or line details.
3. PO.fmb -> Used to build PO interface ,the snap shot is shown in later section .
4. Posum.fmb -> Used to build PO Summary interface , the user can search based on header details or line details.
5. Receipt.fmb -> Used to build Receipt interface ,the snap shot is shown in later section .

6. Invoice.fmb -> Used to build Invoice interface , the snap shot is shown in later section .

7. Invoisum.fmb-> Used to build Invoice summary interface , the user can search based on header details or line details.

8. Payment.fmb -> Used to build Batch payment interface , all the unpayed invoices are listed there as soon as the user selects a paygroup to pay.

9. Bank.fmb -> Used to build Bank entry interface , there are 3 options of payment document: check, wire and EFT.

SQL scripts were used to build various schema objects like tables, views and indexes.

These SQL scripts were used to build the tables at the time of installation, these are included in database design. All the scripts are ended with commit as they are executing at the background. All the columns of the tables are given in database design.

Reqheader.sql -> This script creates Requisition headers table.
Reqline.sql -> This script creates Requisition lines table
Poheader.sql -> This script creates PO headers table
Poline.sql -> This script creates PO lines table
Recheader.sql -> This script creates Receipts headers table
Recline.sql -> This script creates Receipts lines table
Invheader.sql -> This script creates Invoice headers table
Invline.sql -> This script creates Invoice lines table

Bank.sql	->	This script creates Banks table
Reqheader.sql	->	This script creates Requisition headers table
Reqline.sql	->	This script creates Requisition lines table
Poheader.sql	->	This script creates PO headers table
Poline.sql	->	This script creates PO lines table
Recheader.sql	->	This script creates Receipts headers table
Recline.sql	->	This script creates Receipts lines table
Invheader.sql	->	This script creates Invoice headers table
Invline.sql	->	This script creates Invoice lines table
Bank.sql	->	This script Creates Banks table

I have based all the forms on views for security and performance reasons. The Views created were based on all the tables

Reqheader_v.sql	->	This script creates Requisition headers view
Reqline_v.sql	->	This script creates Requisition lines view
Poheader_v.sql	->	This script creates PO headers view
Poline_v.sql	->	This script creates PO lines view
Recheader_v.sql	->	This script creates Receipts headers view
Recline_v.sql	->	This script creates Receipts lines view
Invheader_v.sql	->	This script creates Invoice headers view
Invline_v.sql	->	This script creates Invoice lines view
Bank_v.sql	->	This script creates Banks view
Reqheader_v.sql	->	This script creates Requisition headers view
Reqline_v.sql	->	This script creates Requisition lines view
Poheader_v.sql	->	This script creates PO headers view
Poline_v.sql	->	This script creates PO lines view
Recheader_v.sql	->	This script creates Receipts headers view
Recline_v.sql	->	This script creates Receipts lines view
Invheader_v.sql	->	This script creates Invoice headers view

Invline_v.sql	->	This script creates Invoice lines view
Bank_v.sql	->	This script Creates Banks view

All these scripts are included in database design.

I also created Indexes based on each table to optimize time utilization.

The report files were created for each module. The reports were run on concurrent manager and the .out files were stored in /gldev/gl/rtout/findv115 directory.
The log of all the process was stored in
/gldev/gl/rtlog/findv115 directory. As .log file.

There is also a provision to make Trace ON, that will enable a trace file (.trc) to be build on the database. All the operation performed by the database server are recorded in the file. It helps in debugging. Each table has five 'WHO' columns namely

Created_by
Creation_date
Last_updated_by
Last_updation_date
Last_update_login

These five columns are automatically filled by Oracle Applications whenever any record in the database is modified or added. It is useful for security reasons.

- Menu Editor:

This tool helps create a customized menu that can be attached to form or report.

The back end was ORACLE 8i.

The Oracle8 Database Server

Benefits of the Oracle8 Database Server :

Oracle Applications are Fully Scalable on all Oracle Servers

- Uses state-of-the-art Oracle Parallel Server technology
- Parallelism at other layers of the architecture includes query processing, batch processing, transaction processing, and application module processing

Uses Multiple Nodes to Achieve Higher Performance

- Multiple database instances and their embedded applications servers are distributed across multiple nodes of a cluster or a massively parallel (MPP) system
- Having multiple nodes also provides better reliability
- Using only the Oracle Parallel Server allows for optimization and use of all features rather than using the lowest common denominator

The Oracle8*i* database provides following extended features:

- Multilingual operation
- High availability
    - CBO
    - Partitioned Tables
    - Materialized Views
    - Index Organized Tables

- Resource Manager

- Extreme scalability
- High performance

Oracle Applications Release 11*i* utilizes Cost-based optimization (CBO). Cost based optimization dynamically determines the most efficient access paths and join methods for query execution by taking into account statistics such as the size of each table and the selectivity of each query condition.

A transition to CBO improves performance and enables other database features that depend on cost-based optimization such as Partitioned Tables, Materialized Views, Index-Organized tables, and Resource Manager.

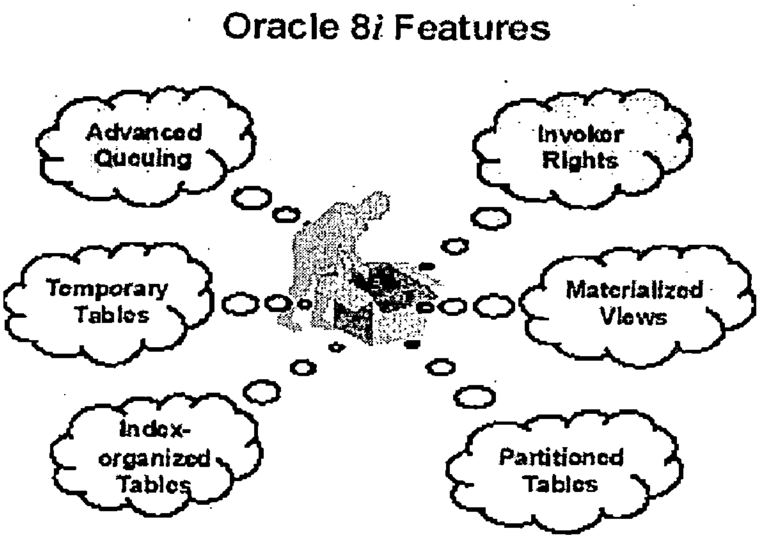


Figure 5.3 Features of Oracle8i Used By Oracle Applications

The Oracle8*i* features utilized by Oracle Applications 11*i* include:

Oracle Advanced Queuing

Oracle Advanced Queuing (Oracle AQ) integrates a message queuing system with the Oracle database. This allows storing messages into queues for deferred retrieval and processing by the Oracle 8*i* Server.

Temporary Tables

A temporary table is a table with session-specific or transaction-specific data. It is empty when the session or transaction begins, and the data are discarded at the end of the session or transaction. Temporary tables are useful for saving intermediate results that can be merged back into another table. In prior Oracle Applications releases, a new table was created to store intermediate data and the table was dropped when the transaction completed. With temporary tables, creating and dropping of tables is no longer necessary, thus improving performance of the process. As Temporary tables use temporary segments, access performance is increased significantly.

Index-organized table

An index-organized table differs from an ordinary table in that the data for the table is held in its associated index. Changes to the table data, such as adding new rows, updating rows, or deleting rows, result in updating only the index. Because data rows are stored in the index, index-organized tables provide faster key-based access to table data for queries that involve exact matches or range searches or both. The storage requirements are reduced because key columns are not duplicated as they are in an ordinary table and its index.

Partitioned tables

Partitioned tables allow data to be broken down into smaller, more manageable pieces called partitions, or even subpartitions. Partitioned tables are customizable to the specific needs of individual customers. Each partition can be managed individually, and can be used independently of the other partitions, thus providing a structure that can be better tuned for availability and performance.

Materialized Views

Materialized views are schema objects that can be used to summarize, precompute, replicate, and distribute data. They are suitable for various computing environments such as data warehousing, decision support, and distributed or mobile computing. For Oracle Applications, materialized views are created and owned by the APPS schema. The associated objects are stored in the respective product tablespace. Cost-based optimization makes use of materialized views to improve query performance by automatically recognizing when a materialized view can and should be used to satisfy a SQL request. The optimizer transparently rewrites the request to use the materialized view. Queries are then directed to the materialized view and not to the underlying detail tables or views. In distributed environments, materialized views are used to replicate data at distributed sites and synchronize updates done at several sites with conflict resolution methods. The replicated materialized views provide local access to data which otherwise would have to be accessed from remote sites.

Invoker Rights

The Invoker Rights model, introduced in Release 11*i*, allows PL/SQL packages to be executed with the privileges of the calling user. Prior releases used a definer rights model wherein PL/SQL packages execute with the privileges of the creating user (defining schema). An invoker-rights package executes with all of the invoker's privileges. Roles are enabled unless the invoker-rights procedure was called directly or indirectly by a definer-rights procedure. Invoker Rights eliminates the need to duplicate packages in other APPS schemas (for example, APPS_MRC). Therefore, maintenance of Multiple Reporting Currencies (MRC) is much quicker, less complicated, and less expensive.

5.2 Application Directory Structure

The directory tree to store application files looks like this :

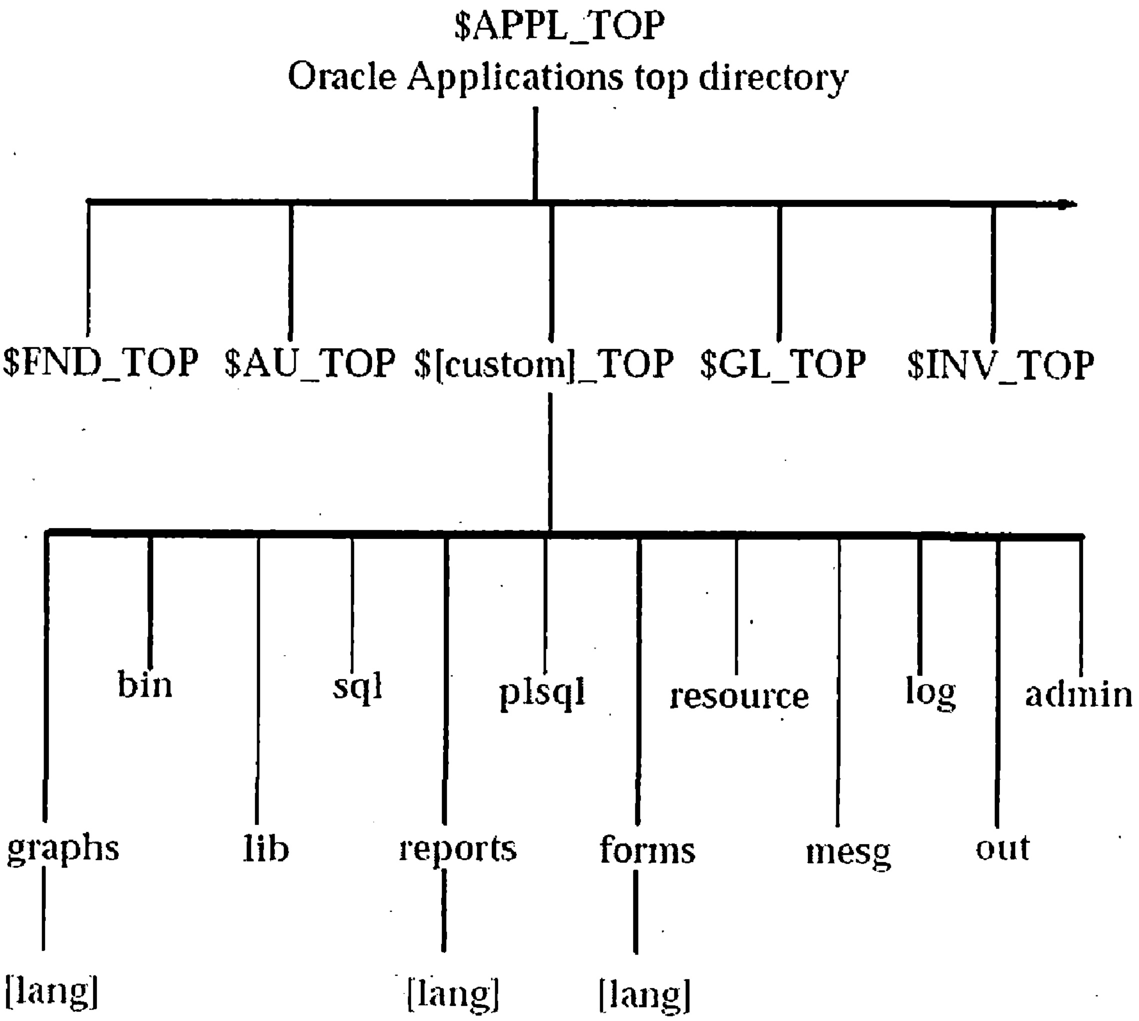


Figure 5.2 ORACLE Application Physical Structure

TOP Directories

$APPL_TOP, $FND_TOP, and so on are environment variables that point to the application base path (use of environment variables depends on the operating system)

BIN

Contains executable code of concurrent programs written in a programming language such as C, Pro*C, Fortran, or an operating system script

LIB

Contains compiled object code of concurrent programs

SQL

Contains concurrent programs written in SQL*Plus and PL/SQL scripts

RESOURCE

Contains PL/SQL libraries used with ORACLE Forms, which must be copied to $AU_TOP for forms generation

GRAPHS

Contains ORACLE Graphics files

FORMS/[LANGUAGE]

The FORMS directory contains .fmx files (and .fmb files) under language subdirectories.

REPORTS

Contains concurrent programs written with ORACLE Reports

May contain language subdirectories

PLSQL

Contains PL/SQL libraries used with ORACLE Reports

LOG

Contains log files from concurrent programs

OUT

Contains output files from concurrent programs

MESG

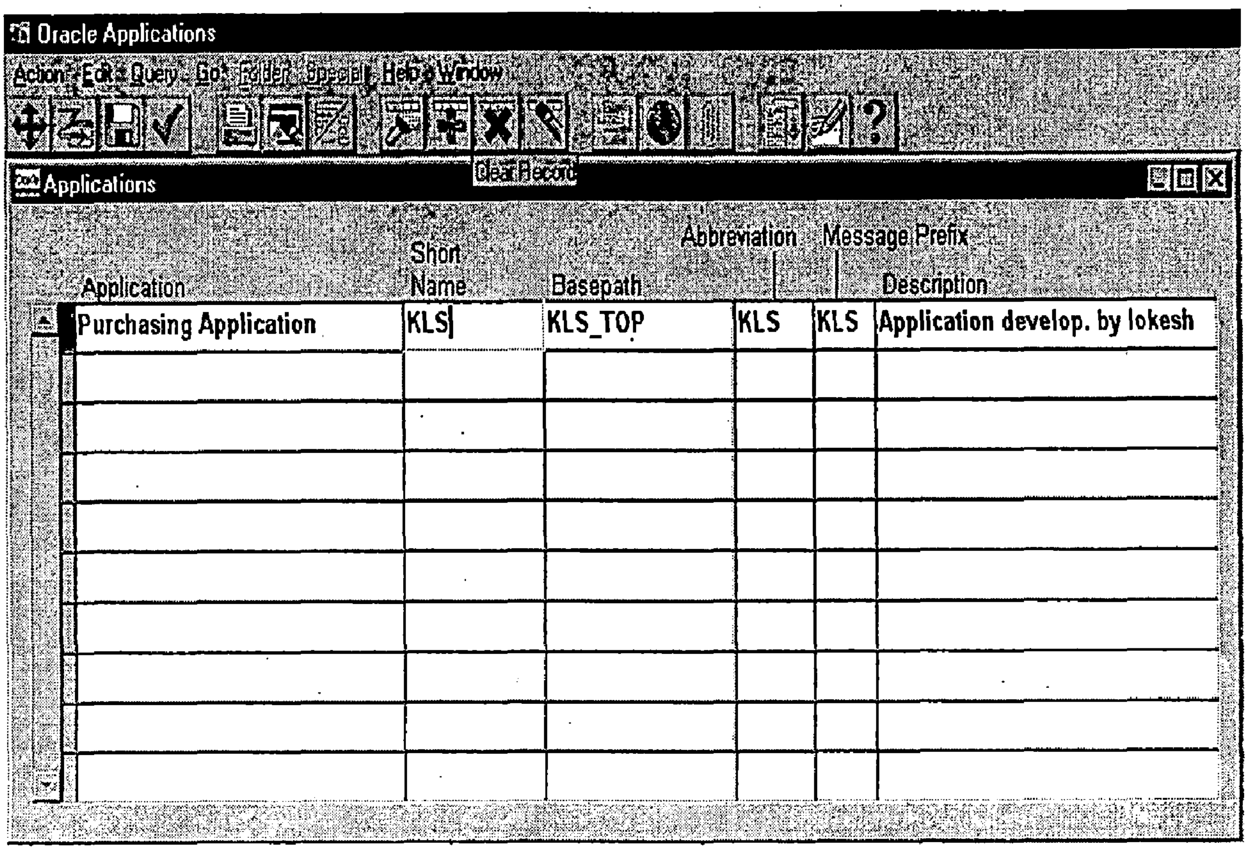
Holds application message files for Message Dictionary

Messages files are generated by the Generate Messages program and reside in a file designated by language names (such as US.msb)

5.3 Registering the Application :

Under System Administrator or Application Developer responsibility:

There's a function called Application Register



Application	Short Name	Basepath	Abbreviation	Message Prefix	Description
Purchasing Application	KLS	KLS_TOP	KLS	KLS	Application develop. by lokesh

Figure 5.3 ORACLE Application Physical Structure

Application Name

This user-friendly name appears in lists seen by application users

Short Name

ORACLE Applications use the application short name when identifying forms , menus, concurrent programs and other application components

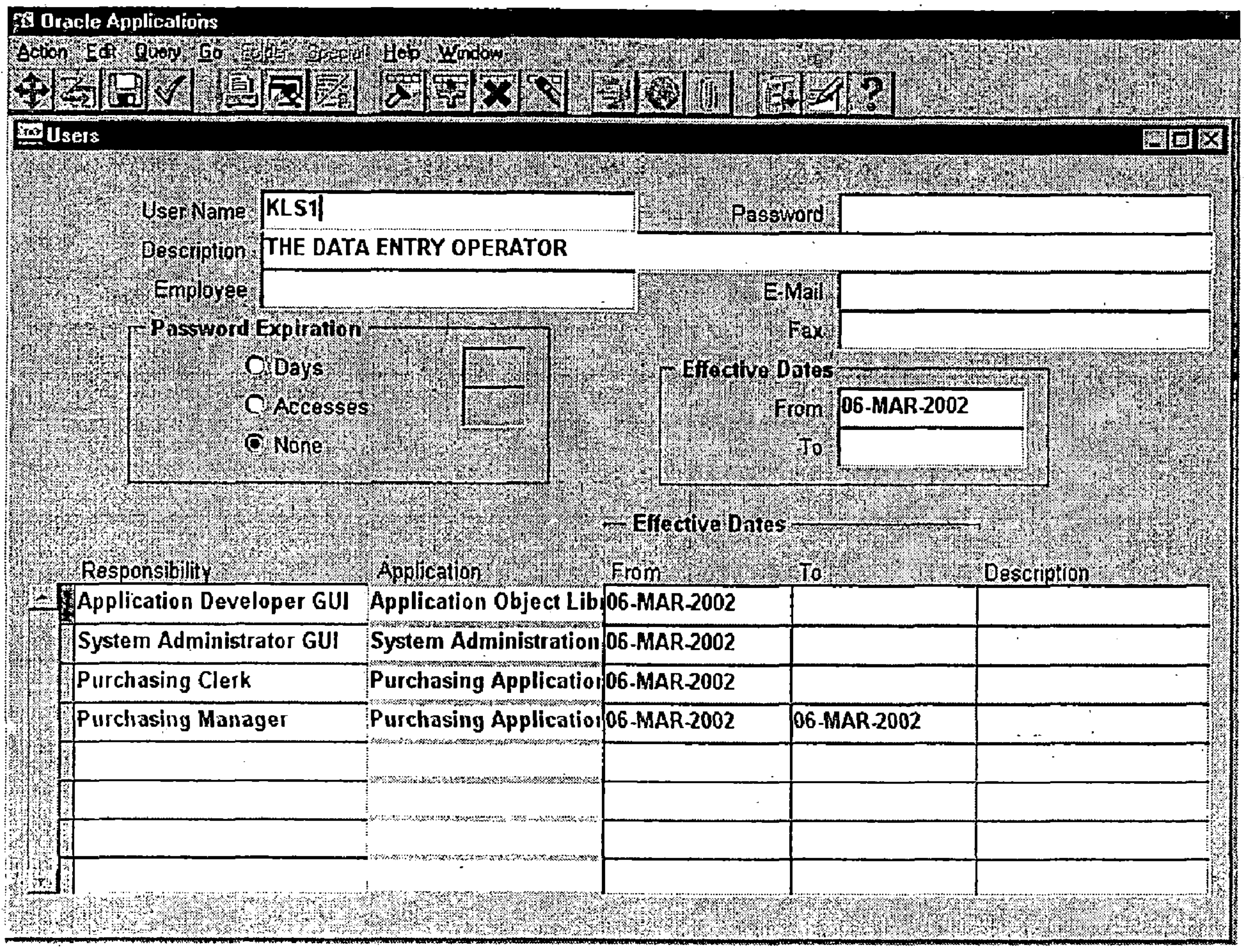
Base path

This is the name of an environment variable which translates into the top directory of application's directory tree (on the applications server)

The base path variable corresponds to the PRODUCT_TOP directory

5.4 Creating Users for the Application :

This is a function under System Administrator responsibility



User Name: KLS1

Password:

Description: THE DATA ENTRY OPERATOR

Employee:

E-Mail:

Fax:

Password Expiration

○ Days
○ Accesses
● None

Effective Dates

From: 06-MAR-2002

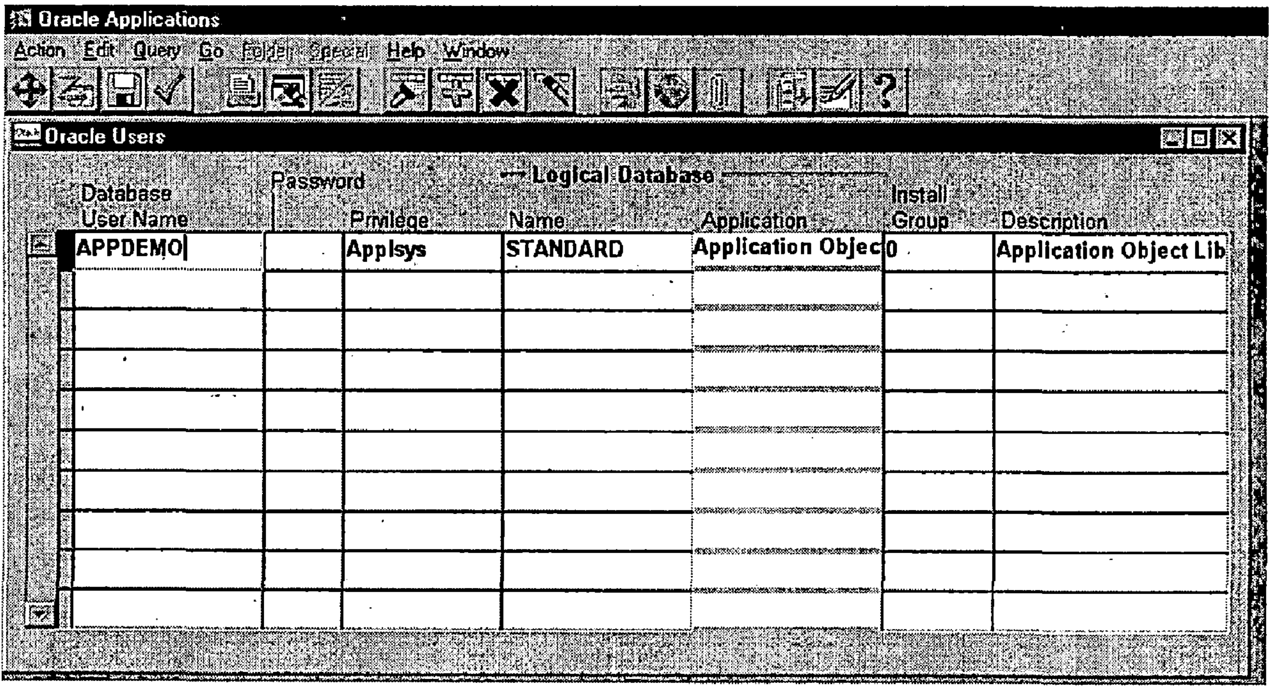
To:

		Effective Dates		
Responsibility	Application	From	To	Description
Application Developer GUI	Application Object Libi	06-MAR-2002		
System Administrator GUI	System Administration	06-MAR-2002		
Purchasing Clerk	Purchasing Applicatioi	06-MAR-2002		
Purchasing Manager	Purchasing Applicatioi	06-MAR-2002	06-MAR-2002	

Figure 5.4 Creating users for ORACLE Application

5.5 Creating ORACLE Users :

This is a function under System Administrator responsibility



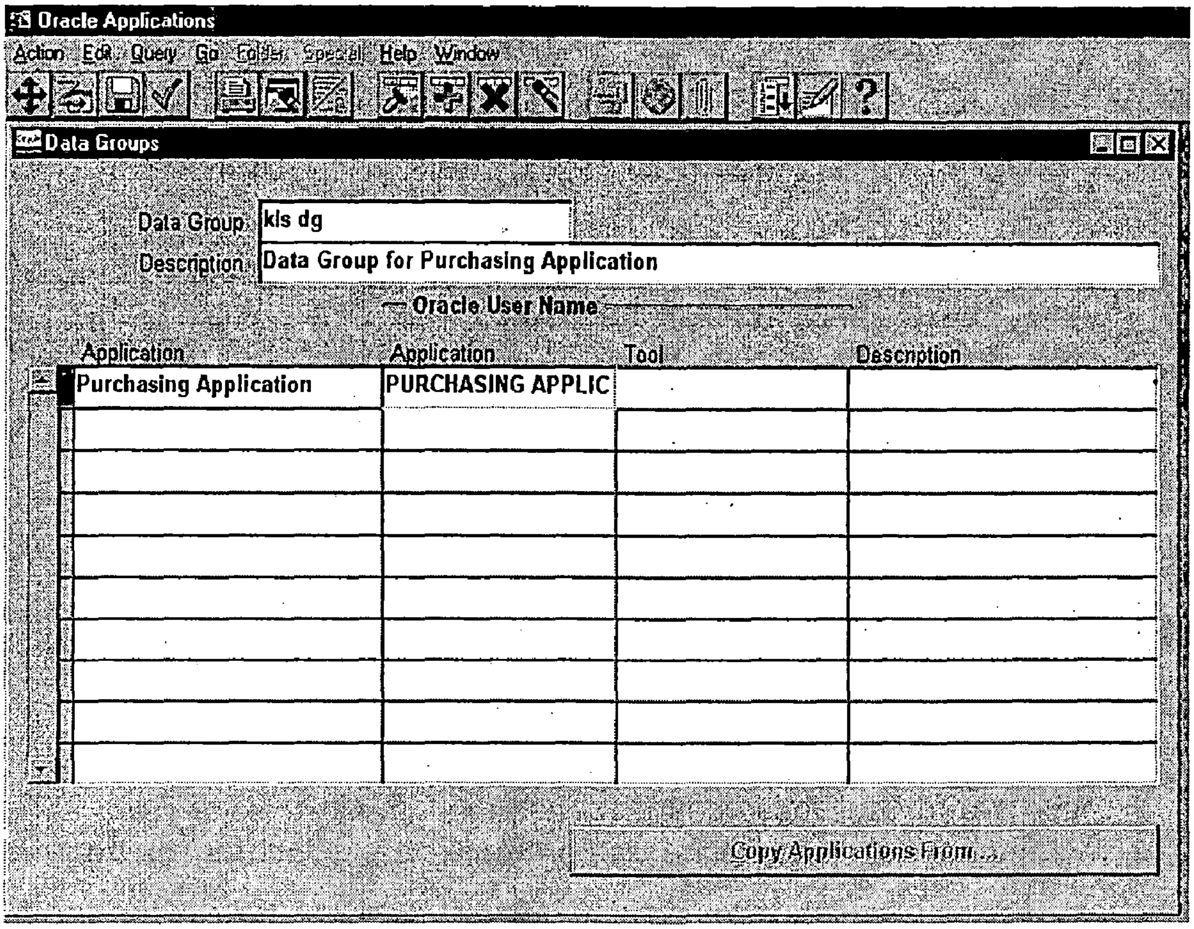
Database User Name	Password	Privilege	Logical Database Name	Application	Install Group	Description
APPDEMO		Applsys	STANDARD	Application Objec	0	Application Object Lib

Figure 5.5 Creating ORACLE Database users

These are my database connections.

Install Group is used for upgrading multiple product installations .

5.6 Defining the Data Group :


Oracle Applications
Action Edit Query Go Folder Special Help Window
Data Groups
Copy Applications From...

Data Group: kls dg

Description: Data Group for Purchasing Application

	Oracle User Name		
Application	Application	Tool	Description
Purchasing Application	PURCHASING APPLIC		

Figure 5.6 ORACLE Applications Data Group

We Typically make a copy of the Standard data group and add custom applications to the copy.

Specifying ORACLE User Names

Here, we specify which ORACLE user (schema) contains our application tables (the ORACLE User Name for the Application)

5.7 Function Security :

Function Security extends the definitions of these existing terms.

Menu

A menu is a hierarchical arrangement of functions and menus of functions

Menu Entry

A menu entry is a menu component that identifies a function or a menu of

Functions

In some cases, both a function and a menu of functions correspond to the same menu entry. For example, a form and its menu of subfunctions can occupy the same menu entry

Responsibility

When application users sign on, they select a responsibility that determines, among other things, the functions they may access.

Available functions are determined by the menu assigned to the current responsibility

Form

It's an ORACLE Forms .fmx file

Forms are located in their application basepath/forms/US (or appropriate language) directory

Function

A function is a part of an application's functionality, registered under a unique name, that can be assigned to or excluded from a responsibility There are two types of functions: form functions (forms), and non-form Functions (subfunctions)

Form Function

A form (form function) invokes an ORACLE Forms form.

A form has the unique property that users may navigate to it from the Navigate window.

Sub function

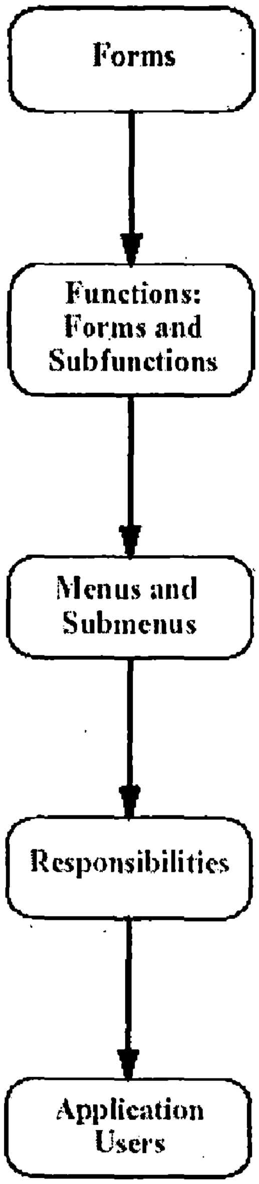
A subfunction (non-form function) is a securable subset of a form's Functionality.

A developer can write logic to test the availability of a subfunction in the current responsibility, then take some action based on whether the subfunction is available

A subfunction is frequently associated with a button or an entry on the Special menu. When such a subfunction is enabled, the corresponding button or menu entry is enabled

A subfunction may correspond to a form procedure not associated with a graphical element, and its availability may not be obvious to the end user

SETTING UP FUNCTION SECURITY



Developer registers each form with AOL

Developer registers the form again as a function

Developer may also register certain functionality of the form (a subfunction) as another function

Developer or system administrator adds functions (both forms and subfunctions) to a menu, often as a hierarchical structure of submenus

Developer or system administrator defines a responsibility and attaches a menu, a data group, and a request group

System administrator may exclude certain functions from the responsibility

Application user can access any forms on the menu, but does not see subfunctions listed on the menu

User cannot see or access forms or subfunctions excluded from the responsibility

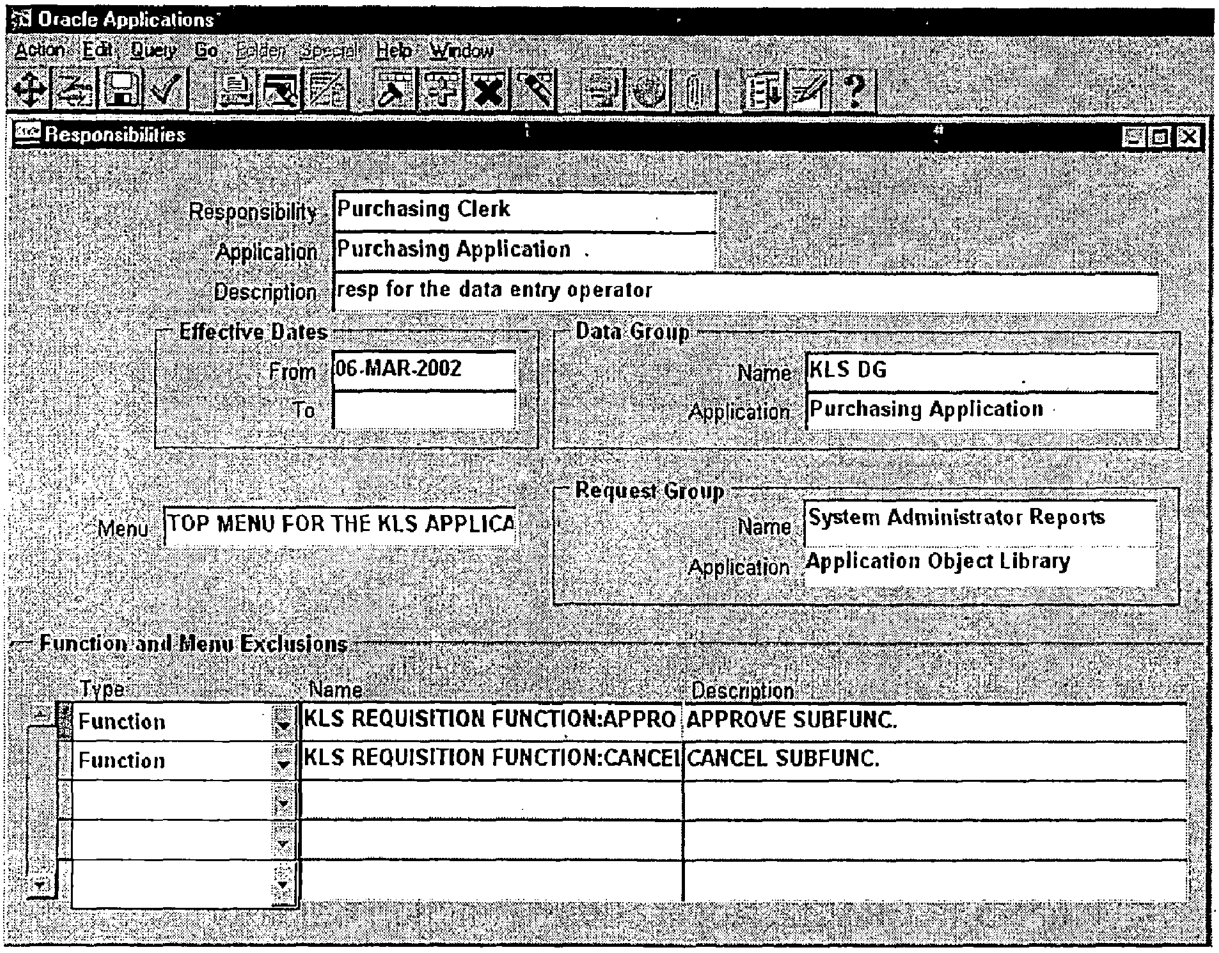
Figure 5.7 ORACLE Applications Function Security

- Functions and Menus of Functions are required , because in forms code I can test the availability of a particular function, then take some action based on whether the function is available

- Each function is then registered
- For form functions, I registered parameters that pass values to a function. For example, a form may support data entry only when a function parameter is passed to it.
- Then I need to define a menu including all the functions available in an application (that is, all the forms and their securable subfunctions)
- For some responsibilities ,I defined additional menus that restrict the application's functionality by omitting certain forms and subfunctions

5.8 Creating the Responsibility :

A responsibility is basically like a role defined with some security permissions.



Responsibility: Purchasing Clerk
Application: Purchasing Application .
Description: resp for the data entry operator

Effective Dates
From: 06-MAR-2002
To:

Data Group
Name: KLS DG
Application: Purchasing Application

Menu: TOP MENU FOR THE KLS APPLICA

Request Group
Name: System Administrator Reports
Application: Application Object Library

Function and Menu Exclusions

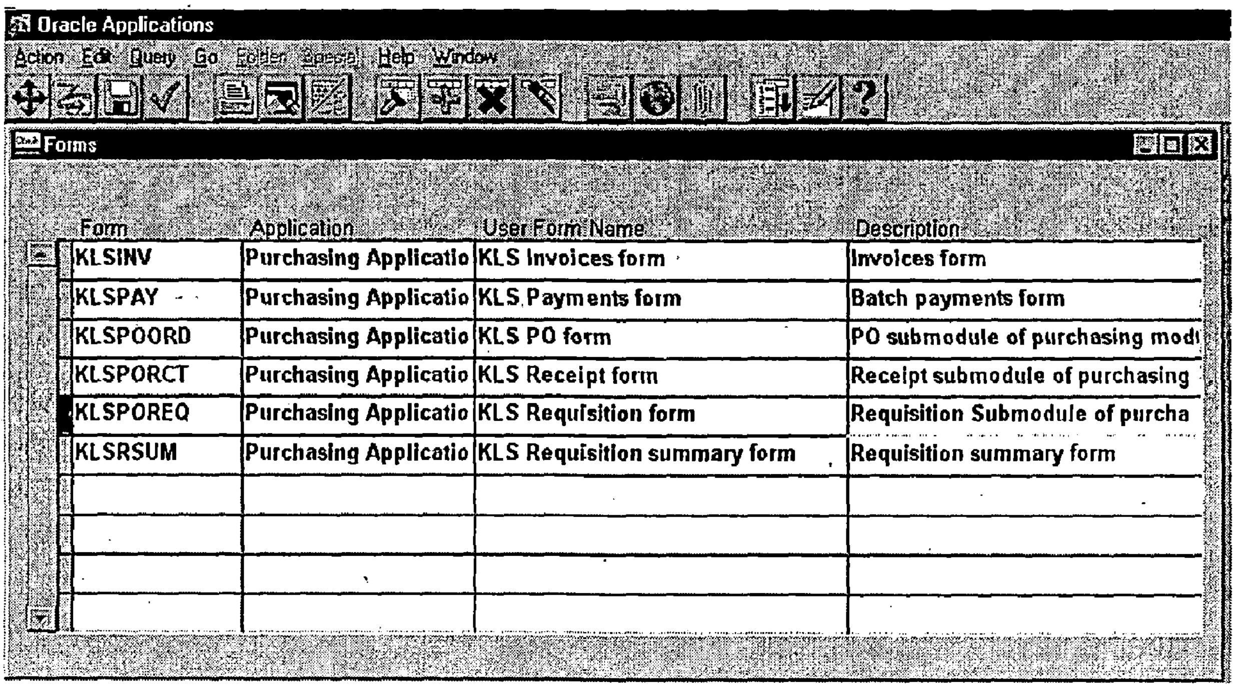
Type	Name	Description
Function	KLS REQUISITION FUNCTION:APPRO	APPROVE SUBFUNC.
Function	KLS REQUISITION FUNCTION:CANCEI	CANCEL SUBFUNC.

Figure 5.8 Creating a Responsibility

In this Application , there are four Responsibilities :

- Purchasing Manager
- Purchasing Clerk
- System Administrator
- Application Developer

5.9 Registering the form with ORACLE Application



Form	Application	User Form Name	Description
KLSINV	Purchasing Applicatio	KLS Invoices form	Invoices form
KLSPAY	Purchasing Applicatio	KLS Payments form	Batch payments form
KLSPOORD	Purchasing Applicatio	KLS PO form	PO submodule of purchasing modi
KLSPORCT	Purchasing Applicatio	KLS Receipt form	Receipt submodule of purchasing
KLSPOREQ	Purchasing Applicatio	KLS Requisition form	Requisition Submodule of purcha
KLSRSUM	Purchasing Applicatio	KLS Requisition summary form	Requisition summary form

Figure 5.9 Registration of Forms

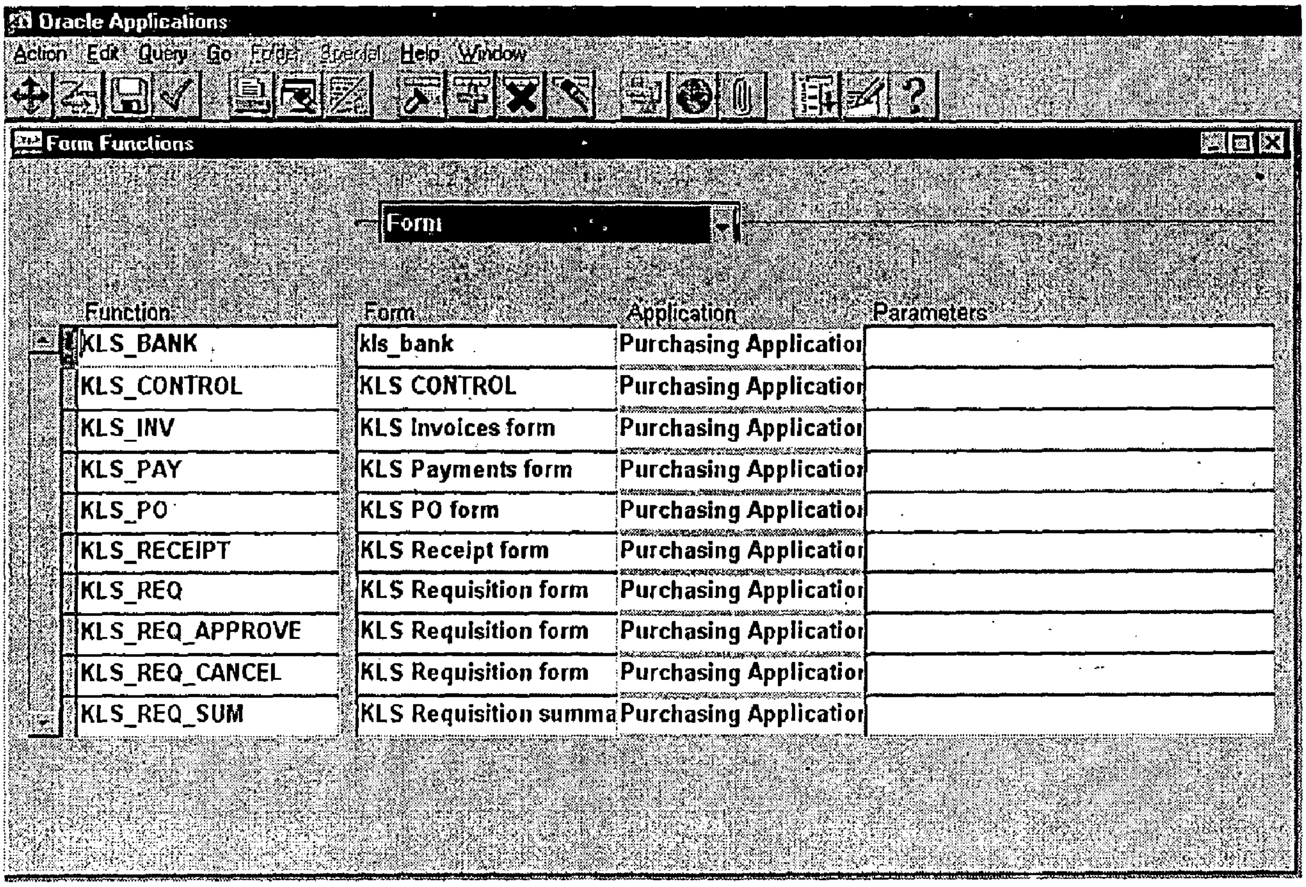
Form: This is the filename of my form (without an extension). The form filename must be all uppercase, and its .fmx file should be located in my application directory structure.

Application: This is the application that owns my form. The application tells ORACLE Application Object Library where to find the form file

User Form Name: This is the form name we see when selecting a form using the Functions window.

5.10 Registering Form Functions and Subfunctions

We register the form functions and sub functions on the Form Functions window



Function	Form	Application	Parameters
KLS_BANK	kls_bank	Purchasing Applicatio	
KLS_CONTROL	KLS CONTROL	Purchasing Applicatio	
KLS_INV	KLS Invoices form	Purchasing Applicatio	
KLS_PAY	KLS Payments form	Purchasing Applicatio	
KLS_PO	KLS PO form	Purchasing Applicatio	
KLS_RECEIPT	KLS Receipt form	Purchasing Applicatio	
KLS_REQ	KLS Requisition form	Purchasing Applicatio	
KLS_REQ_APPROVE	KLS Requisition form	Purchasing Applicatio	
KLS_REQ_CANCEL	KLS Requisition form	Purchasing Applicatio	
KLS_REQ_SUM	KLS Requisition summa	Purchasing Applicatio	

Figure 5.10 Registering Functions and Subfunctions

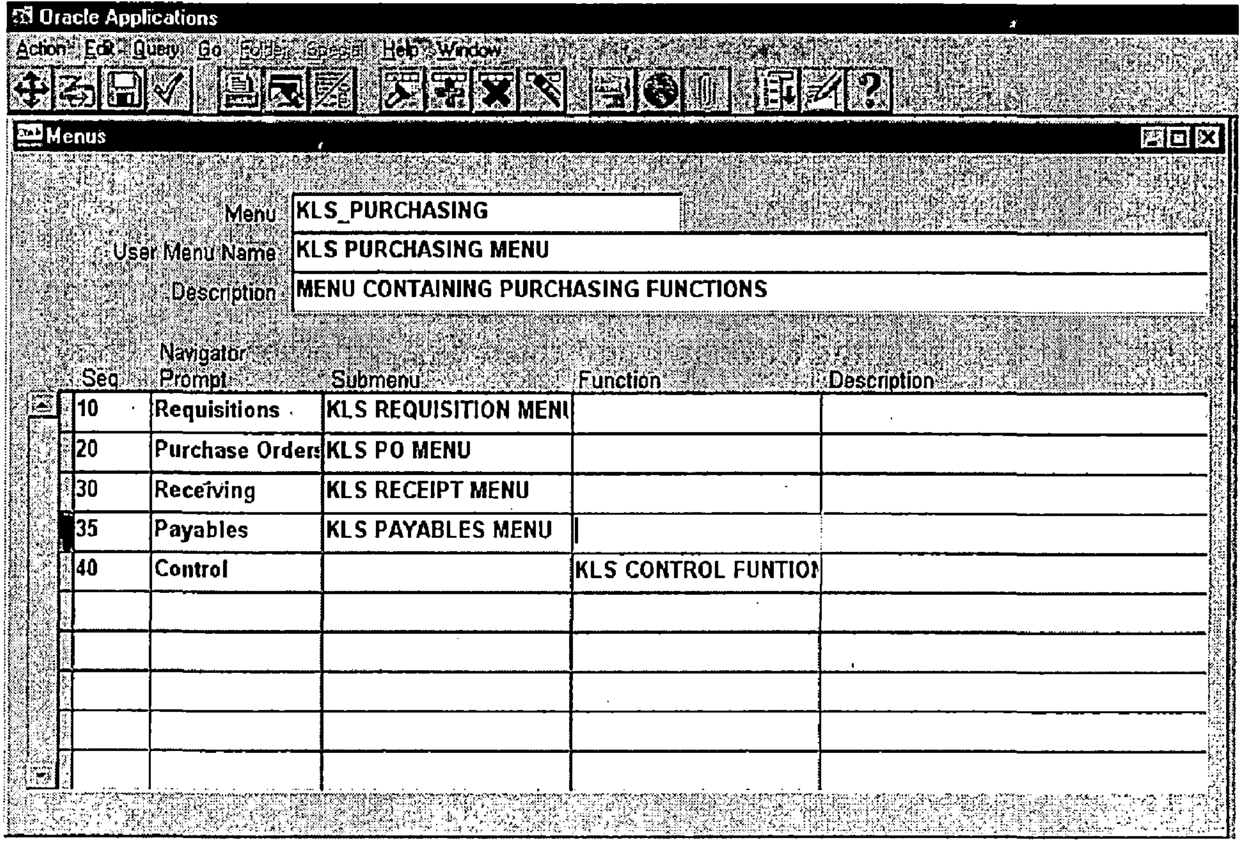
Function: Users do not see this unique function name, but we use it in our code when starting a form using function security routines or testing for function availability

Form /Application: If we are defining a form function, select the user name and application of our form.

Parameters: Enter the parameters to pass to the function (assuming the form is built to accept them). Separate parameters with a space

5.11 Creating A Menu Of Functions

Now , we need to add the functions to a menu. The functions in a menu determines the access privileges of a user.



Menu: KLS_PURCHASING

User Menu Name: KLS PURCHASING MENU

Description: MENU CONTAINING PURCHASING FUNCTIONS

Seq	Navigator Prompt	Submenu	Function	Description
10	Requisitions	KLS REQUISITION MENU		
20	Purchase Orders	KLS PO MENU		
30	Receiving	KLS RECEIPT MENU		
35	Payables	KLS PAYABLES MENU		
40	Control		KLS CONTROL FUNTIOI	

Figure 5.2 Creating a Menu

Here we include any forms the user should have access to, including forms that are opened programmatically from another form (such as by pressing a button) .

Menu: Choose a name that describes the purpose of the menu

User Menu Name: We use a menu name when a responsibility calls a main menu or when one menu calls another

Sequence: Enter a sequence number to specify where a menu entry appears relative to other menu entries in a menu

Navigator Prompt: Enter a user-friendly, intuitive prompt the menu displays for this menu entry. The user sees this menu prompt in the Navigate window, leave the prompt blank for subfunctions that should not appear in the Navigator menu listing even though they are on the menu

Submenu: Call another menu, which allows the user to select menu entries from that menu

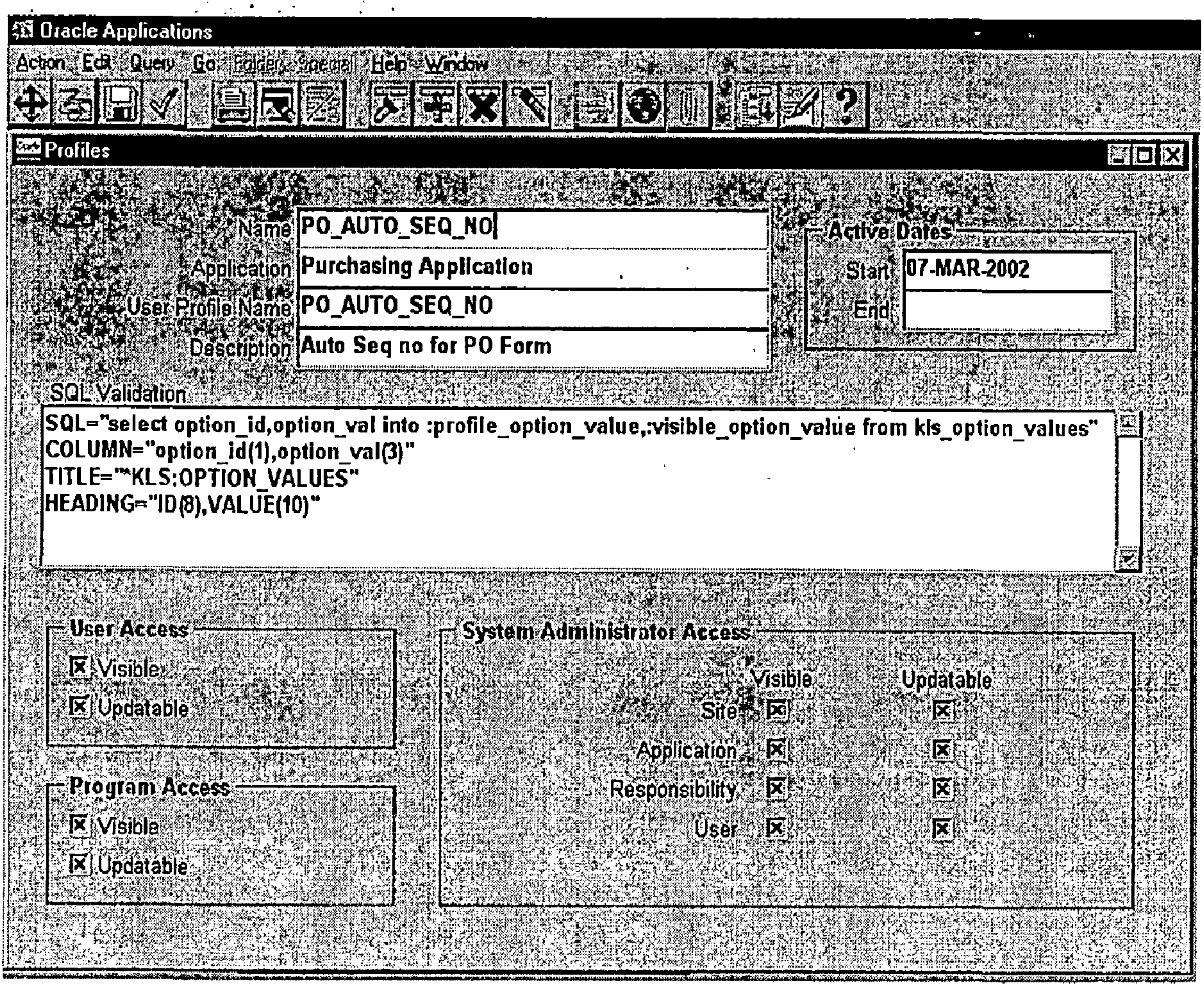
Function: Call a function we wish to include in the menu. A form function (form) appears in the Navigate window and allows access to that form. Other non-form functions (subfunctions) allow access to a particular subset of form functionality from this menu

Functions and submenus are not mutually exclusive—we can have both a submenu and a function as a single menu entry, though the function is invisible to the user in the Navigator and could be a separate menu entry

Description: Enter a description of the menu choice. This description appears in the Description field under the menu path in the Navigator

5.12 Creating User Profiles :

User profiles can be used for global preferences and security:



Name: PO_AUTO_SEQ_NO
Application: Purchasing Application
User Profile Name: PO_AUTO_SEQ_NO
Description: Auto Seq no for PO Form

Active Dates
Start: 07-MAR-2002
End:

SQL Validation

```
SQL="select option_id,option_val into :profile_option_value,:visible_option_value from kls_option_values"
COLUMN="option_id(1),option_val(3)"
TITLE="*KLS:OPTION_VALUES"
HEADING="ID(8),VALUE(10)"
```

User Access
☒ Visible
☒ Updatable

Program Access
☒ Visible
☒ Updatable

System Administrator Access

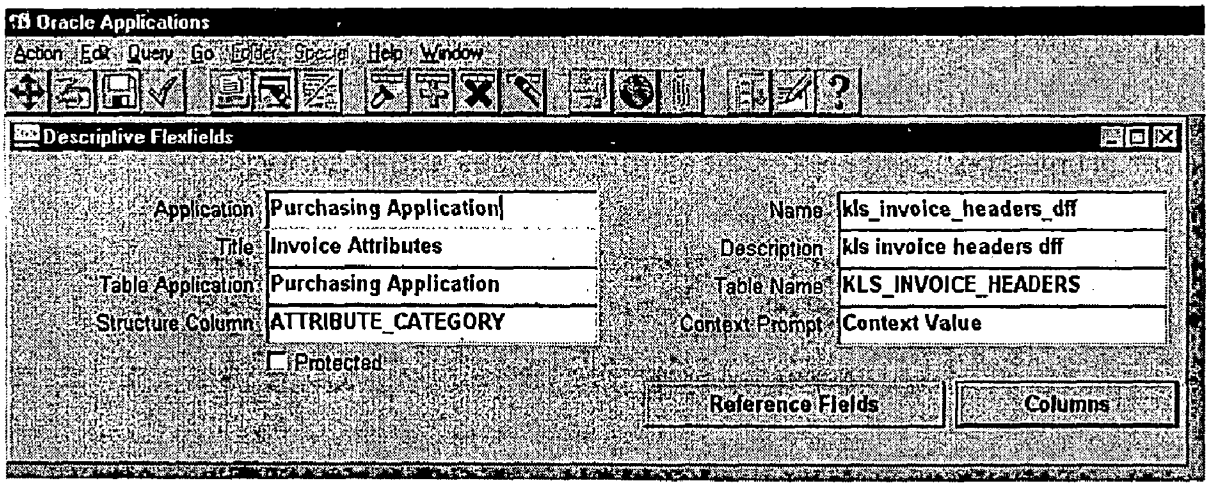
	Visible	Updatable
Site	☒	☒
Application	☒	☒
Responsibility	☒	☒
User	☒	☒

Figure 5.12 Creating User Profiles

They are basically user preferences which can be changed dynamically.

5.13 Registering Descriptive Flex Fields :

Descriptive flexfields provide customizable "expansion space" on the forms. I used descriptive flexfields to track additional information, important and unique to the business, that would not otherwise be captured by the form.



Application: Purchasing Application

Title: Invoice Attributes

Table Application: Purchasing Application

Structure Column: ATTRIBUTE_CATEGORY

□ Protected

Name: kls_invoice_headers_dff

Description: kls invoice headers dff

Table Name: KLS_INVOICE_HEADERS

Context Prompt: Context Value

Figure 5.13 Creating Descriptive Flex fields

5.14 Registration of Tables:

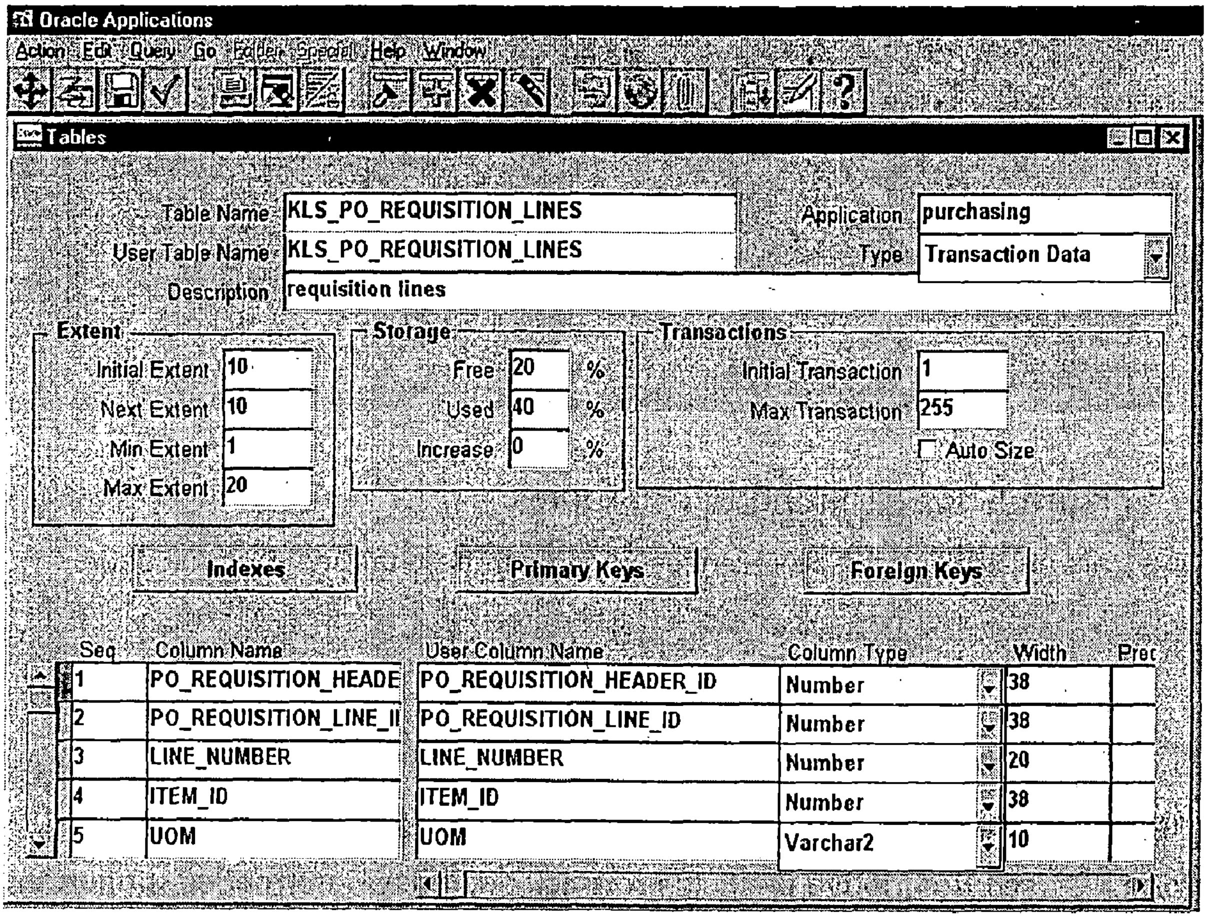


Table Name: KLS_PO_REQUISITION_LINES

User Table Name: KLS_PO_REQUISITION_LINES

Description: requisition lines

Application: purchasing

Type: Transaction Data

Extent

Initial Extent: 10

Next Extent: 10

Min Extent: 1

Max Extent: 20

Storage

Free: 20 %

Used: 40 %

Increase: 0 %

Transactions

Initial Transaction: 1

Max Transaction: 255

□ Auto Size

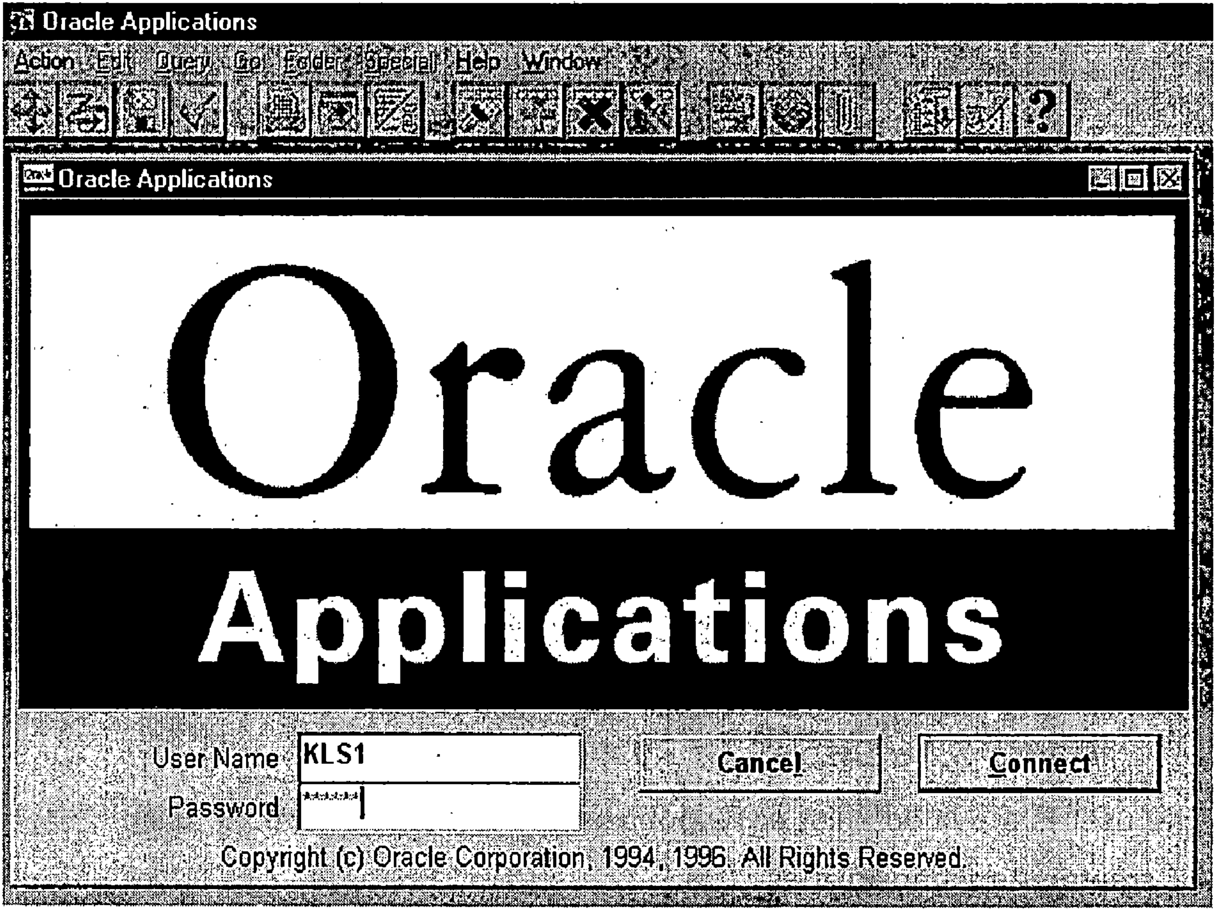
Seq	Column Name	User Column Name	Column Type	Width	Prec
1	PO_REQUISITION_HEADE	PO_REQUISITION_HEADER_ID	Number	38	
2	PO_REQUISITION_LINE_II	PO_REQUISITION_LINE_ID	Number	38	
3	LINE_NUMBER	LINE_NUMBER	Number	20	
4	ITEM_ID	ITEM_ID	Number	38	
5	UOM	UOM	Varchar2	10	

Figure 5.14 Registering a table with ORACLE Applications

We need to register those tables in which we intend to use Flexfields.

5.15 User Interface Of Custom Application

My Application Sign On screen :



User Name KLS1

Password ******

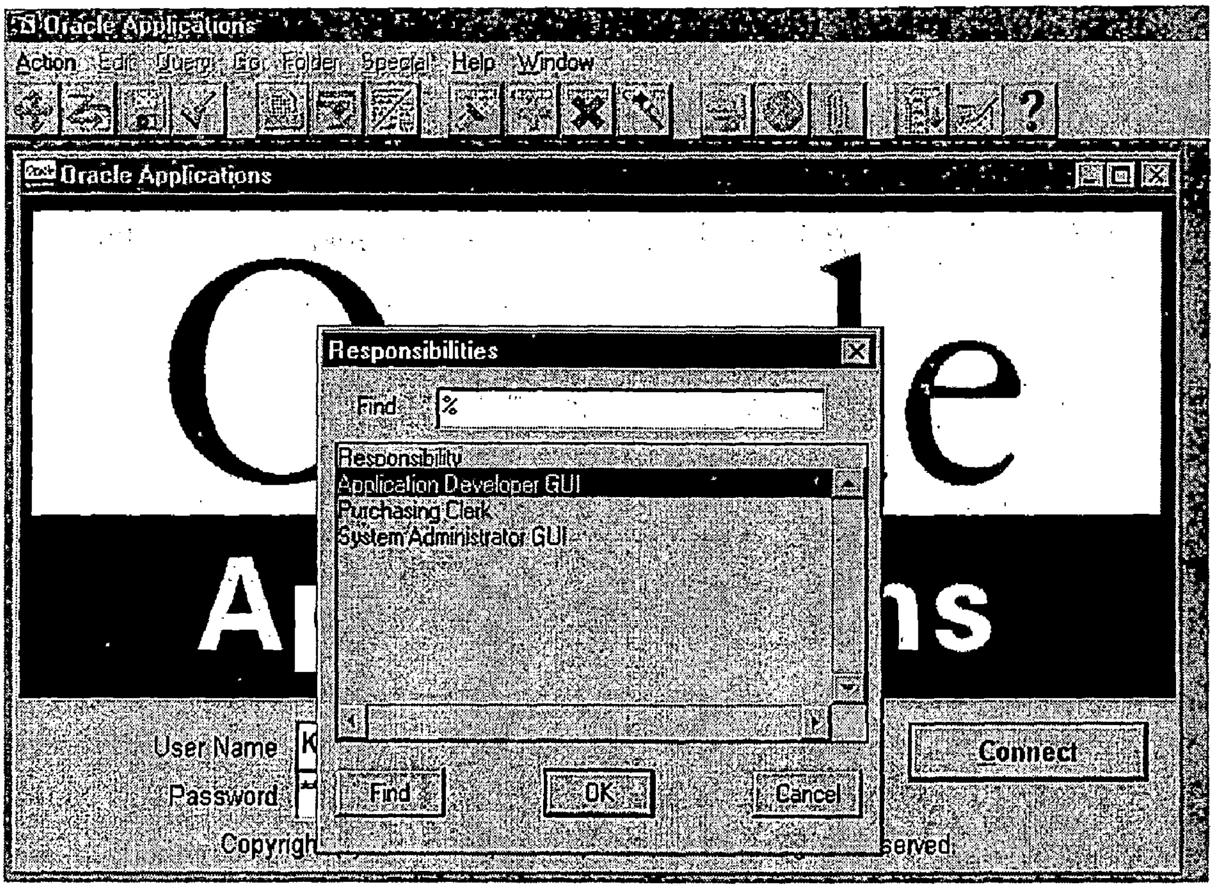
Copyright (c) Oracle Corporation, 1994, 1996. All Rights Reserved.

Figure 5.15 Sign on Screen

Here I created 2 users : KLS1 and KLS2.

Depending on the user corresponding responsibilities are listed to choose from.

After Sign On, depending upon the user, certain responsibilities appear in the menu to chose from :



Find %

Responsibility
Application Developer GUI
Purchasing Clerk
System Administrator GUI

Figure 5.16 Display Of Responsibilities

The user is provided a list of responsibilities to choose from.

I created 2 responsibilities :

- **Purchasing Manager**
- **Purchasing Clerk.**

The purchasing Manager has some additional powers along with powers of Purchasing Clerk. These are :

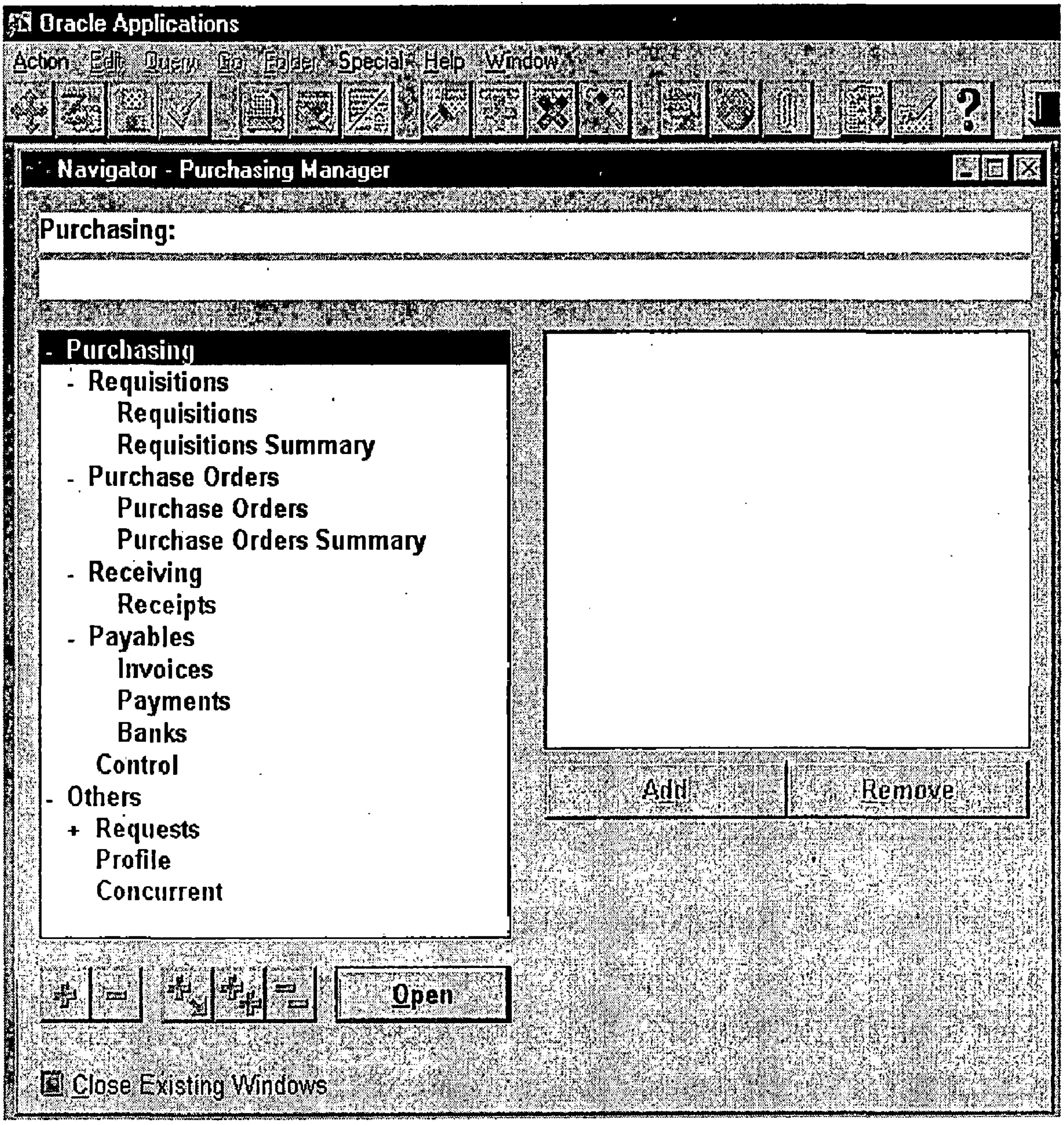
He can approve or cancel an unapproved Requisitions, Purchase Orders, Receipts and Invoices.

He can make Payments to the suppliers against the approved Invoices.

In contrast a clerk can only prepare a document Manager is responsible for further progress.

Purchasing Application Menu :

Each responsibility is associated with a menu.


Oracle Applications
Action Edit Query Go Folder Special Help Window
Navigator - Purchasing Manager
Add
Remove
Open
Close Existing Windows

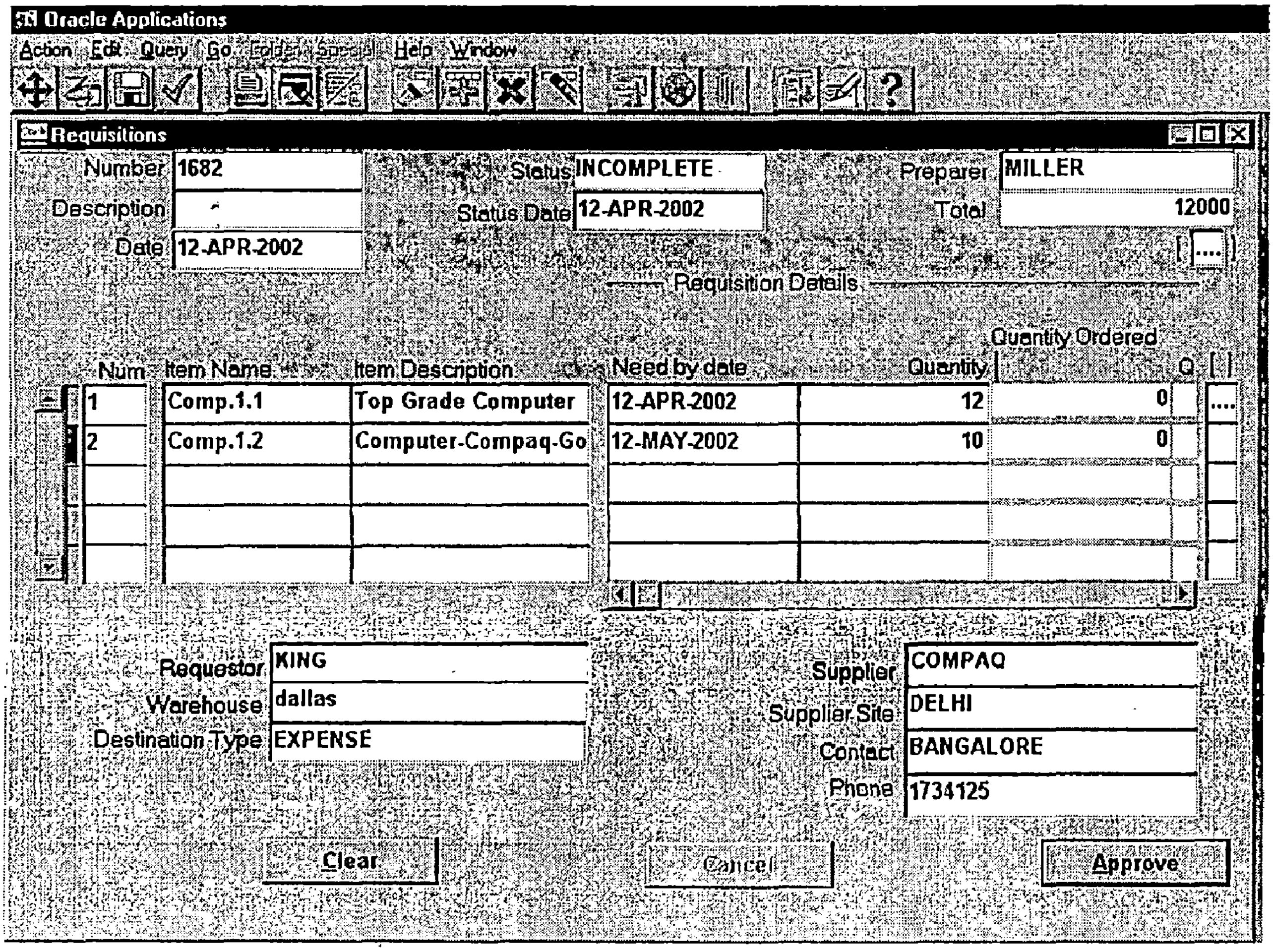
Purchasing:

- Purchasing
  - Requisitions
    - Requisitions
    - Requisitions Summary
  - Purchase Orders
    - Purchase Orders
    - Purchase Orders Summary
  - Receiving
    - Receipts
  - Payables
    - Invoices
    - Payments
    - Banks
  - Control
- Others
  - + Requests
  - Profile
  - Concurrent

Figure 5.17 Display of Menu

The menu displays all the functions and subfunctions available. We can hide or unhide some functions from certain responsibilities.

Requisition screen :



Number 1682
Status INCOMPLETE
Preparer MILLER
Description
Status Date 12-APR-2002
Total 12000
Date 12-APR-2002

Num	Item Name	Item Description	Need by date	Quantity	Quantity Ordered	Q
1	Comp.1.1	Top Grade Computer	12-APR-2002	12	0	
2	Comp.1.2	Computer-Compaq-Go	12-MAY-2002	10	0	

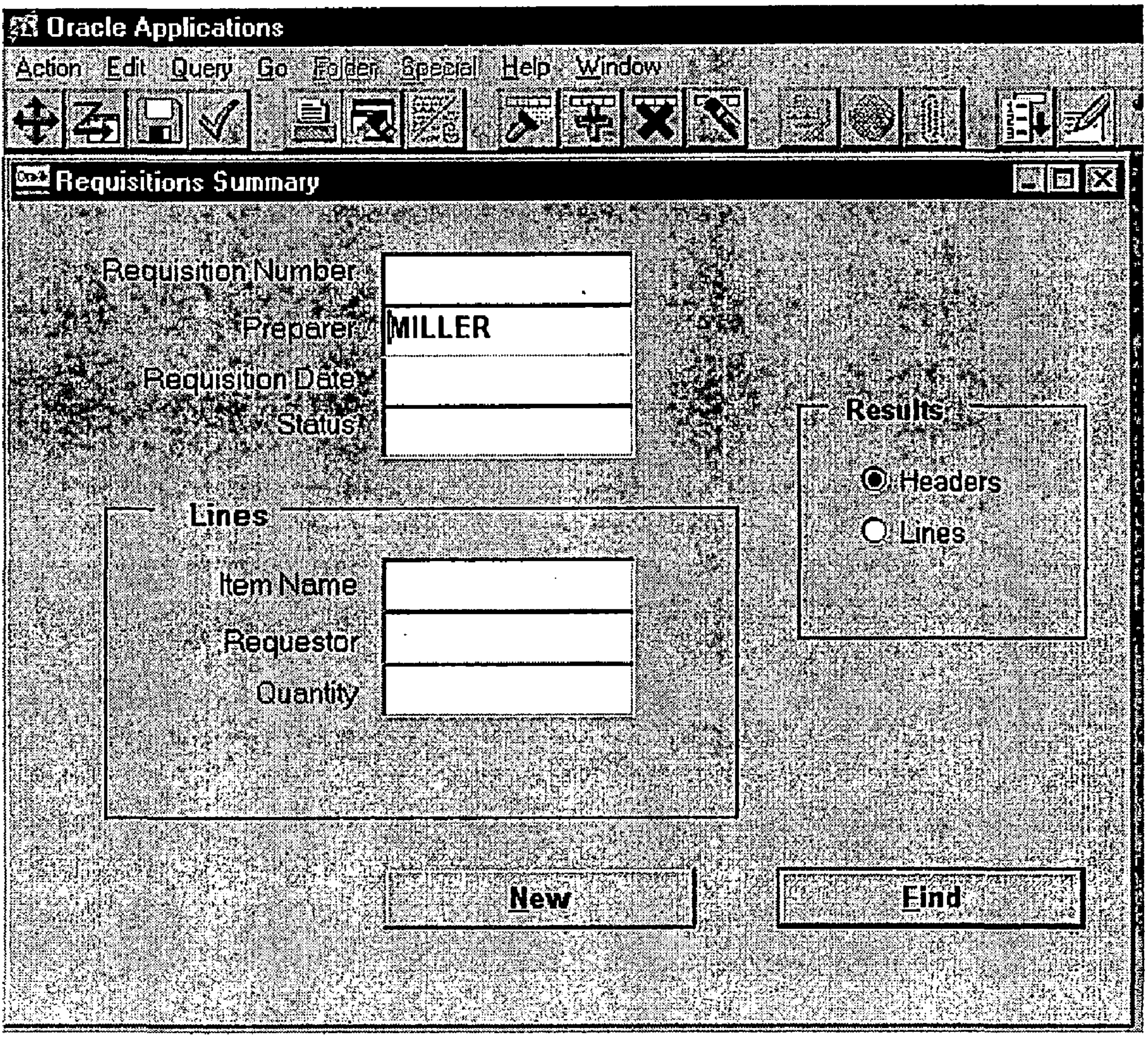
Requestor KING
Warehouse dallas
Destination Type EXPENSE
Supplier COMPAQ
Supplier Site DELHI
Contact BANGALORE
Phone 1734125

Figure 5.18 Requisition Screen

To starts the process a Requisition should be prepared. The Requisition status is initially INCOMPLETE. The user has LOV for most of the fields. The REQ number value can come automatically or is user enterable depending on the user profile "REQ_AUTO_SEQ_NUMBER".The user can alter the profile from the control from All fields are duly validated. Only a Manager can Approve a Requisition. We can't make Purchase Orders of Unapproved Requisitions. A Manager can also cancel an approved Requisition.

A Requisition can have any number of preferred suppliers ,one for each line item.

Requisition Summary:



Requisition Number
Preparer MILLER
Requisition Date
Status

Lines
Item Name
Requestor
Quantity

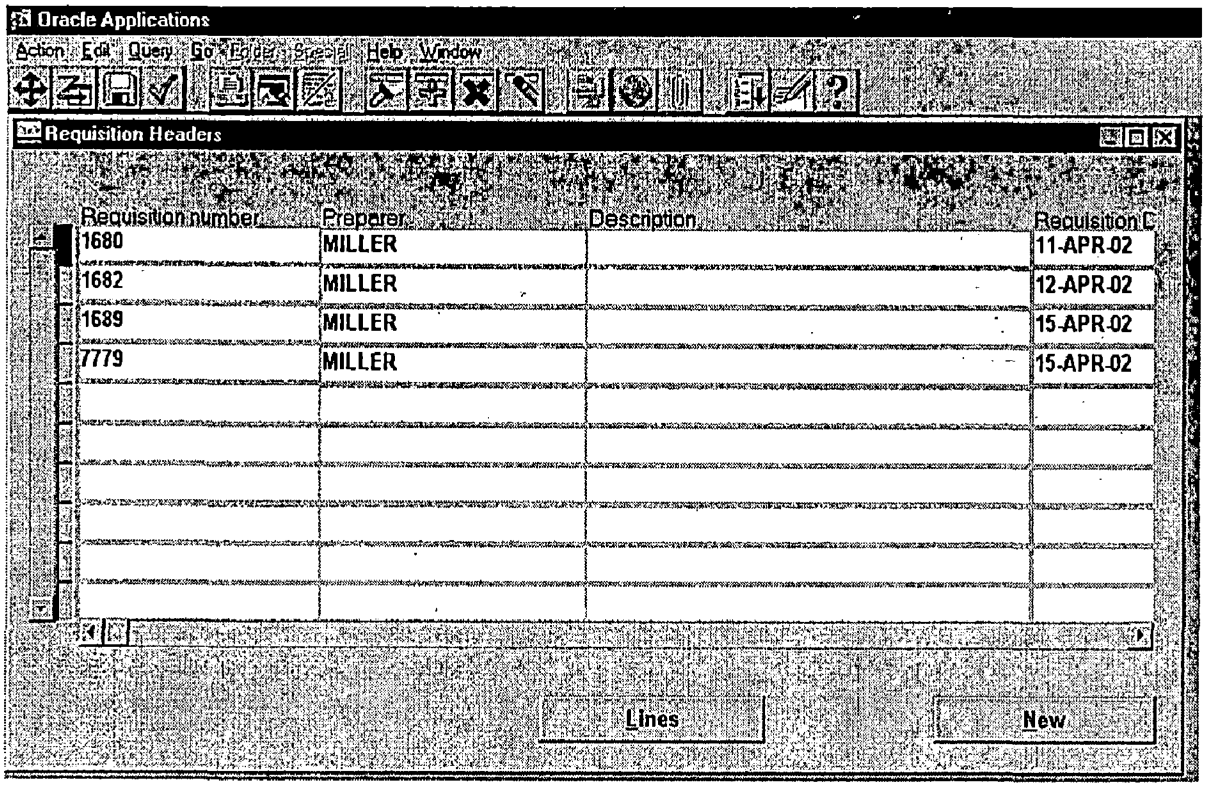
Results
● Headers
○ Lines

Figure 5.19 Requisition Summary

As specified in specs a user can query the existing Requisitions . He can based his search on basis of Requisition Number, Preparer , Requisition date, Status.

He can also query the Requisition lines item on basis of Item name, requestor and quantity.

Summary Header Result:

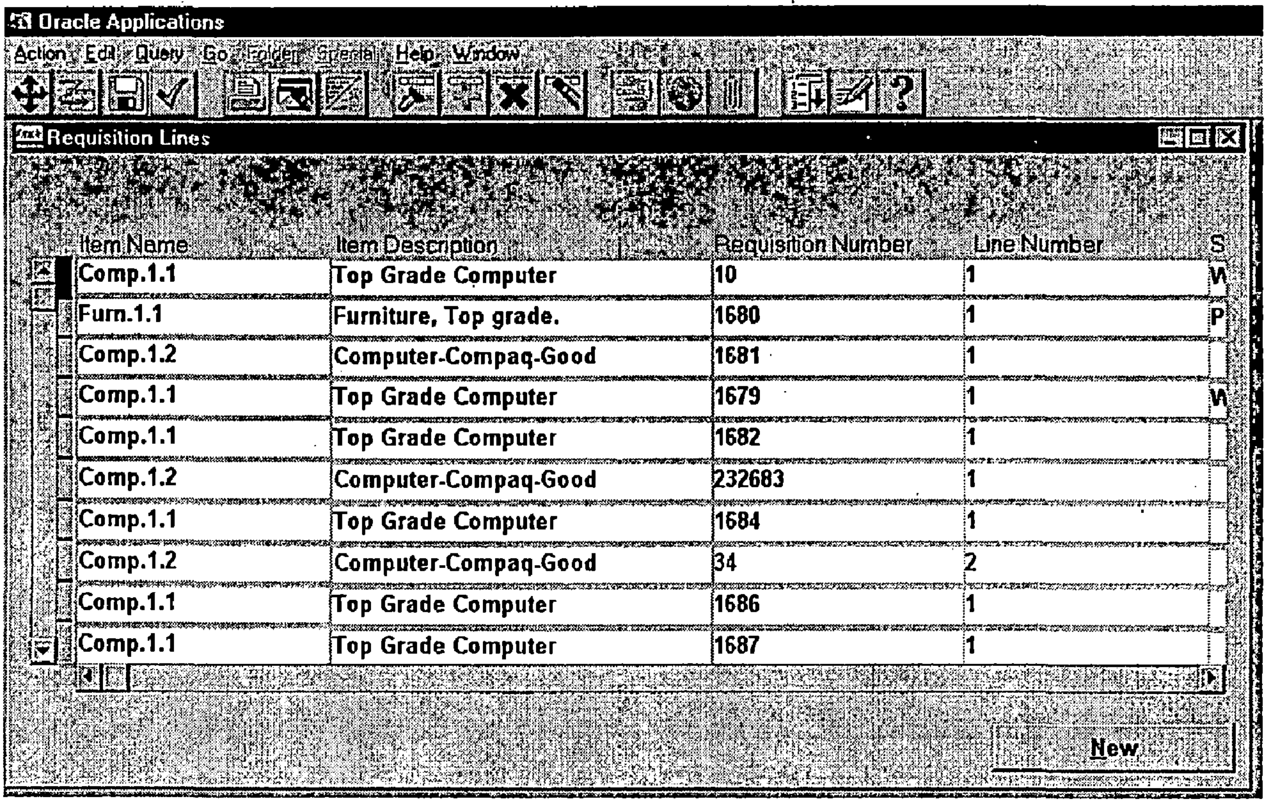


Requisition number	Preparer	Description	Requisition C
1680	MILLER		11-APR-02
1682	MILLER		12-APR-02
1689	MILLER		15-APR-02
7779	MILLER		15-APR-02

Figure 5.20 Summary Header Screen

Figure 5.20 shows the query result. We can directly view the corresponding lines by clicking on Lines button.

Summary Lines Result:

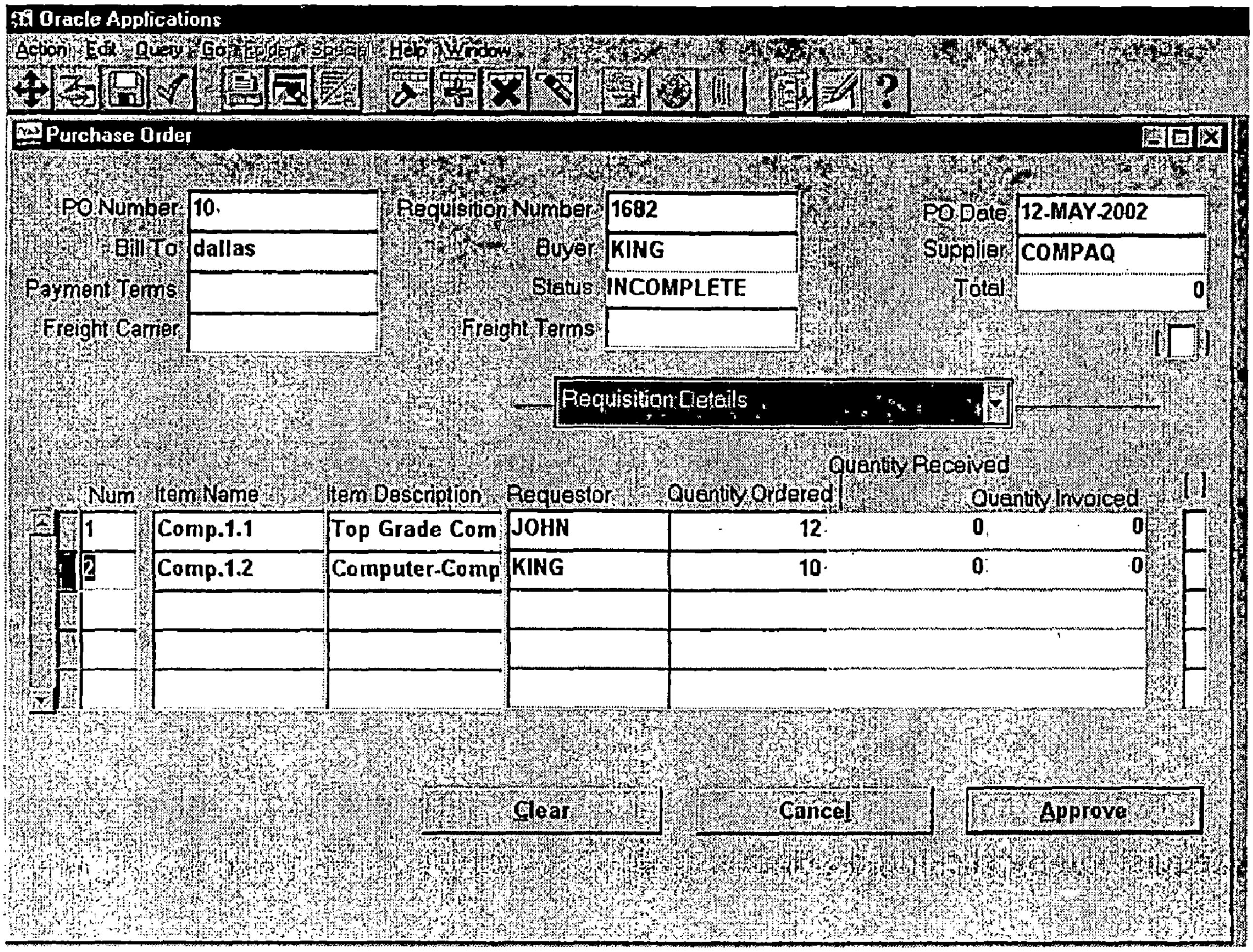


Item Name	Item Description	Requisition Number	Line Number	S
Comp.1.1	Top Grade Computer	10	1	W
Furn.1.1	Furniture, Top grade.	1680	1	P
Comp.1.2	Computer-Compaq-Good	1681	1	
Comp.1.1	Top Grade Computer	1679	1	W
Comp.1.1	Top Grade Computer	1682	1	
Comp.1.2	Computer-Compaq-Good	232683	1	
Comp.1.1	Top Grade Computer	1684	1	
Comp.1.2	Computer-Compaq-Good	34	2	
Comp.1.1	Top Grade Computer	1686	1	
Comp.1.1	Top Grade Computer	1687	1	

Figure 5.21 Summary Lines Display

Figure 5.21 shows corresponding Lines details. We can make a new Requisition by clicking on New button.

Purchase Order :

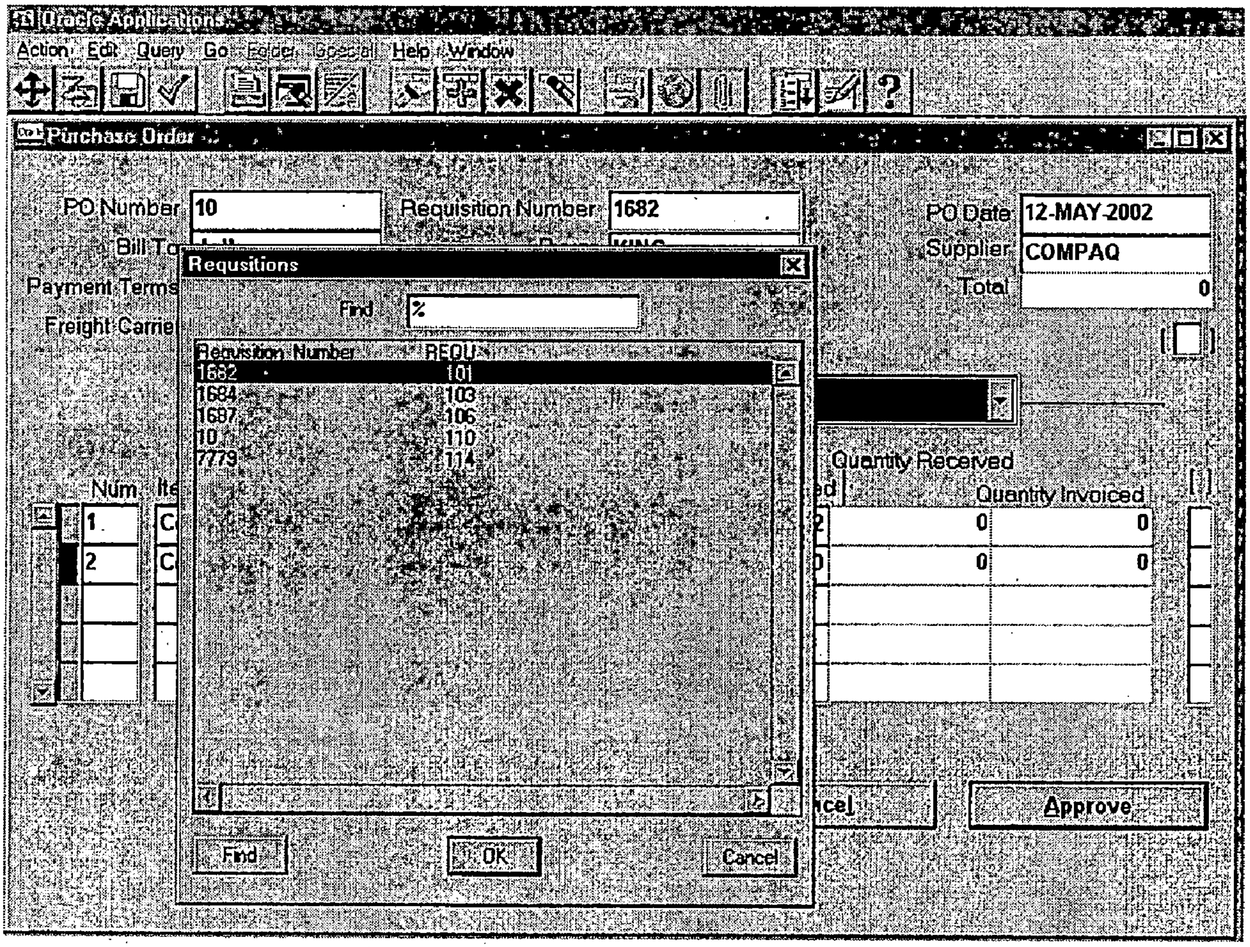


PO Number: 10
Requisition Number: 1682
PO Date: 12-MAY-2002
Bill To: dallas
Buyer: KING
Supplier: COMPAQ
Payment Terms:
Status: INCOMPLETE
Total: 0
Freight Carrier:
Freight Terms:

Num	Item Name	Item Description	Requestor	Quantity Ordered	Quantity Received	Quantity Invoiced
1	Comp.1.1	Top Grade Com	JOHN	12	0	0
2	Comp.1.2	Computer-Comp	KING	10	0	0

Figure 5.22 Purchase Order Screen

The user can either make a P.O. based on a Requisition or with out a Requisition also. The LOV shows all the Approved Requisitions which are unordered. The PO number value can come automatically or is user enterable depending on the user profile "PO_AUTO_SEQ_NUMBER".The user can alter the profile from the control from.



PO Number 10
Requisition Number 1682
PO Date 12-MAY-2002
Supplier COMPAQ
Total 0
Bill To
Payment Terms
Freight Carrie

Find %

Requisition Number	REQU
1682	101
1684	103
1687	106
10	110
7779	114

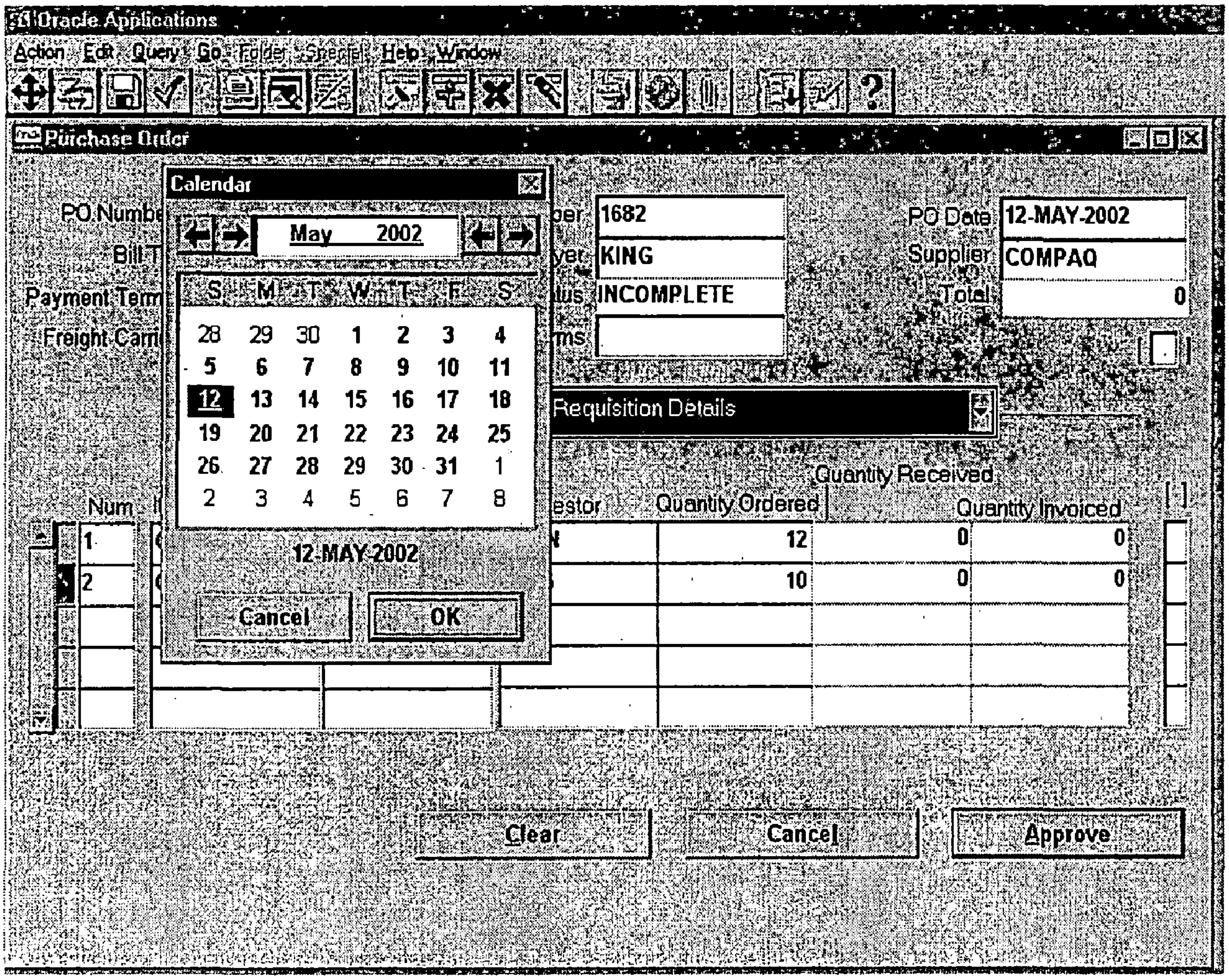
Num	Quantity Received	Quantity Invoiced
1	0	0
2	0	0

Figure 5.23 Requisition Number LOV

So, The user can select on which Requisition this PO is to be prepared.

And also as soon as the user chooses a particular supplier in P.O., all the requisition lines items which have that supplier as the preferred supplier will automatically populate the PO lines regions. As shown in Figure 5.24.The user can add or remove the items from PO lines.

Date Window pops up :



PO Numbe ber 1682

Bill T

Payment Term tus INCOMPLETE

Freight Carri ms

yer KING

PO Date 12-MAY-2002

Supplier COMPAQ

Total 0

May 2002

S	M	T	W	T	F	S
28	29	30	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

12-MAY-2002

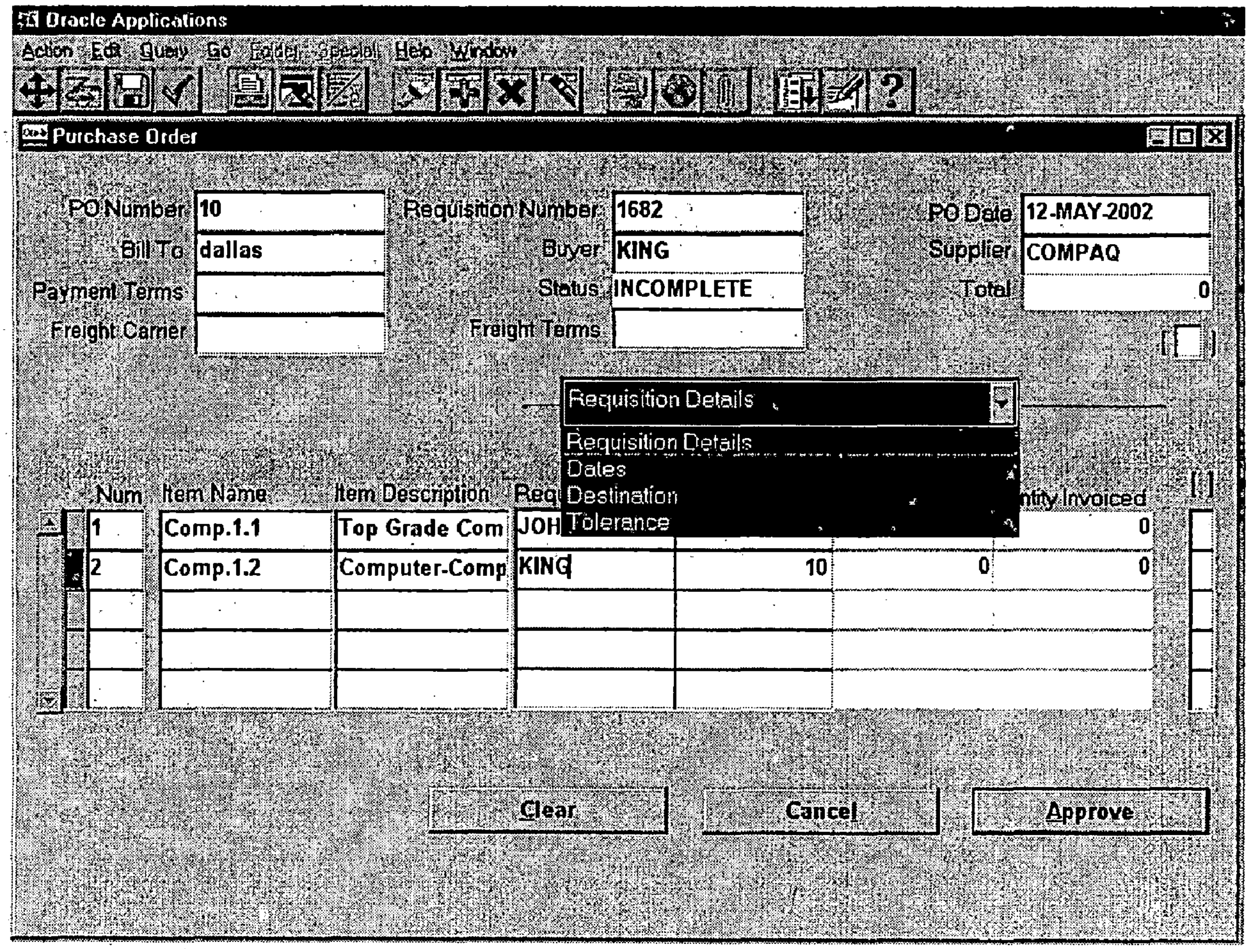
Num	estor	Quantity Ordered	Quantity Received	Quantity Invoiced
1		12	0	0
2		10	0	0

Figure 5.24 Date Selection

The user can choose a valid date of PO preparation from the LOV.

Alternate Regions :

Alternate Regions helps in arranging the items on the form in a logical way.



PO Number	10	Requisition Number	1682	PO Date	12-MAY-2002
Bill To	dallas	Buyer	KING	Supplier	COMPAQ
Payment Terms		Status	INCOMPLETE	Total	0
Freight Carrier		Freight Terms			

Requisition Details

- Requisition Details
- Dates
- Destination
- Tolerance

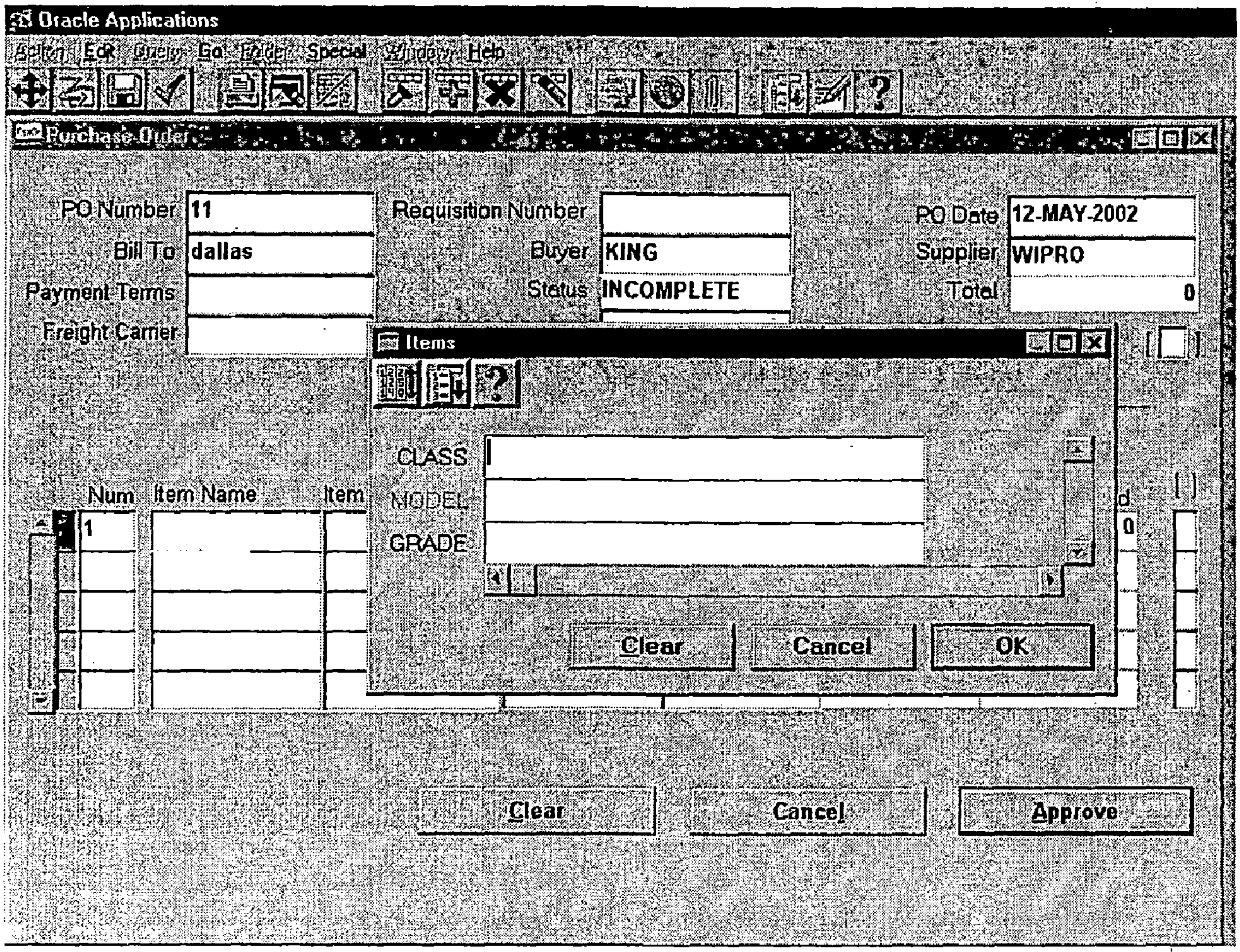
Num	Item Name	Item Description	Req			ntity Invoiced
1	Comp.1.1	Top Grade Com	JOH			0
2	Comp.1.2	Computer-Comp	KING	10	0	0

Figure 5.25 Alternate Regions Display

The alternate regions provides a clean interface to the user. In this PO ,I have about 16 items in the PO lines region. So, to arrange them in one form , alternate regions is an effective tool.

Item key flex field:

Figure 1 & 2 shows the flexibility provided in entering the Item Name(Provided by using Key Flex Field concept.)



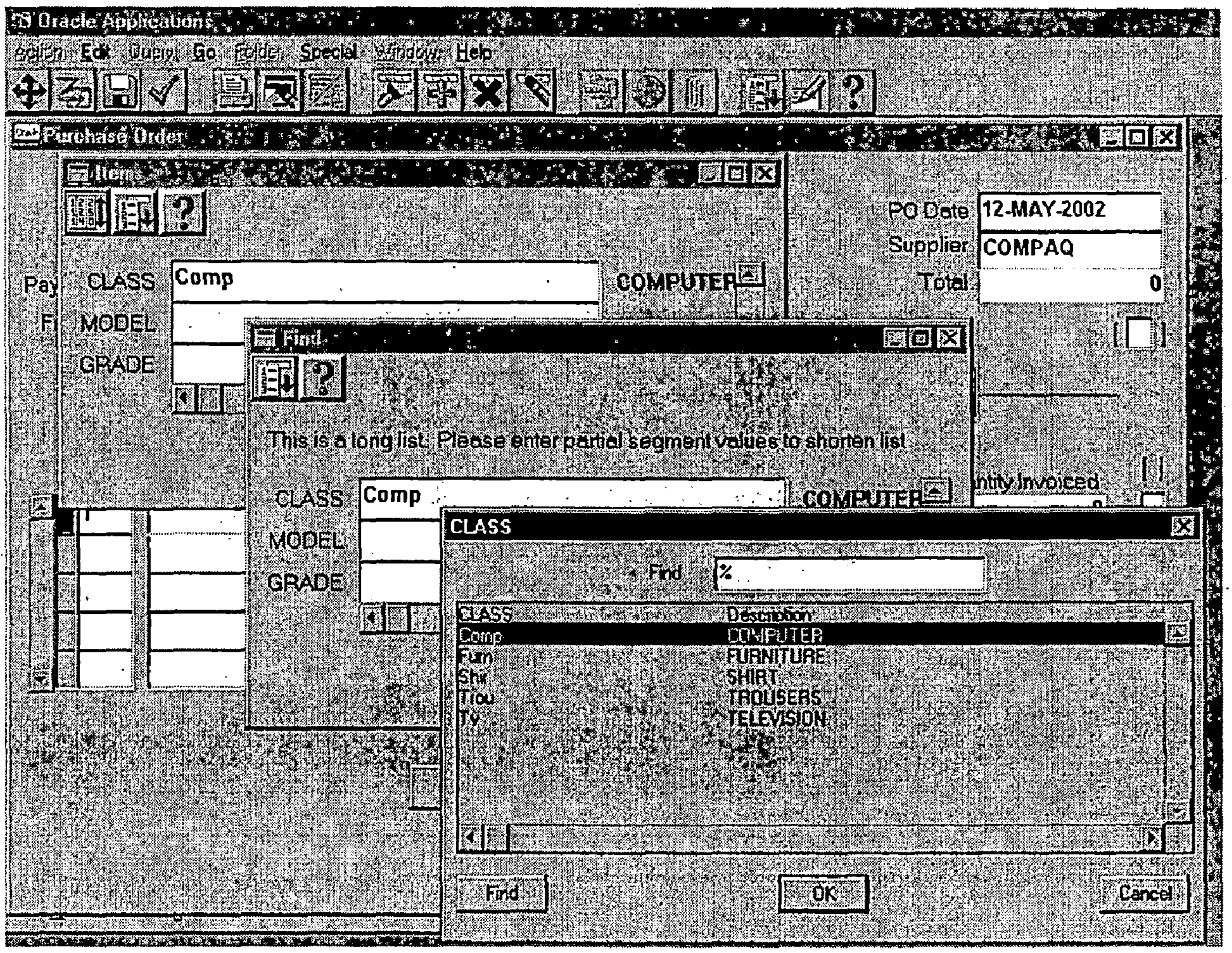
PO Number	11	Requisition Number		PO Date	12-MAY-2002
Bill To	dallas	Buyer	KING	Supplier	WIPRO
Payment Terms		Status	INCOMPLETE	Total	0
Freight Carrier					

CLASS
MODEL
GRADE

Num	Item Name	Item
1		

Figure 5.26 Display of Item Key flex field

As we navigate into a flexfield item, it pops up an LOV and asks us to enter a valid combination as defined in the table.



PO Date 12-MAY-2002
Supplier COMPAQ
Total 0

CLASS Comp COMPUTER
MODEL
GRADE

This is a long list. Please enter partial segment values to shorten list

CLASS Comp COMPUTER
MODEL
GRADE

Find %

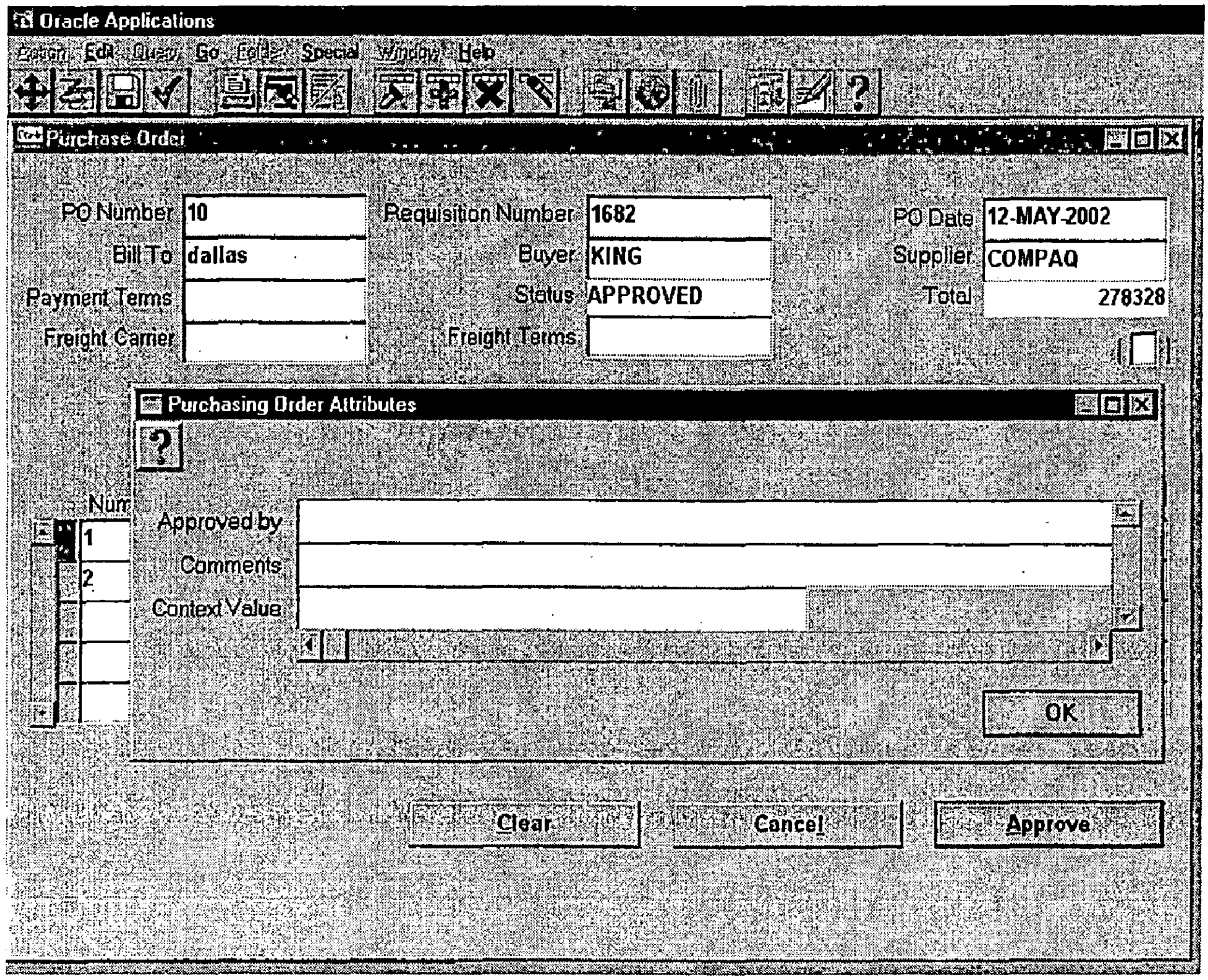
CLASS	Description
Comp	COMPUTER
Furn	FURNITURE
Shir	SHIRT
Trou	TROUSERS
Tv	TELEVISION

Figure 5.27 Display of Item Key flex field

We can select a valid combination from the LOV.

Descriptive Flex Field for PO Headers :

So, User can give context related information in the Descriptive Flex field.


Oracle Applications
Action Edit Query Go Folder Special Window Help
Purchase Order
Purchasing Order Attributes
?
Num
1
2
OK
Clear
Cancel
Approve

PO Number	10	Requisition Number	1682	PO Date	12-MAY-2002
Bill To	dallas	Buyer	KING	Supplier	COMPAQ
Payment Terms		Status	APPROVED	Total	278328
Freight Carrier		Freight Terms			

Approved by

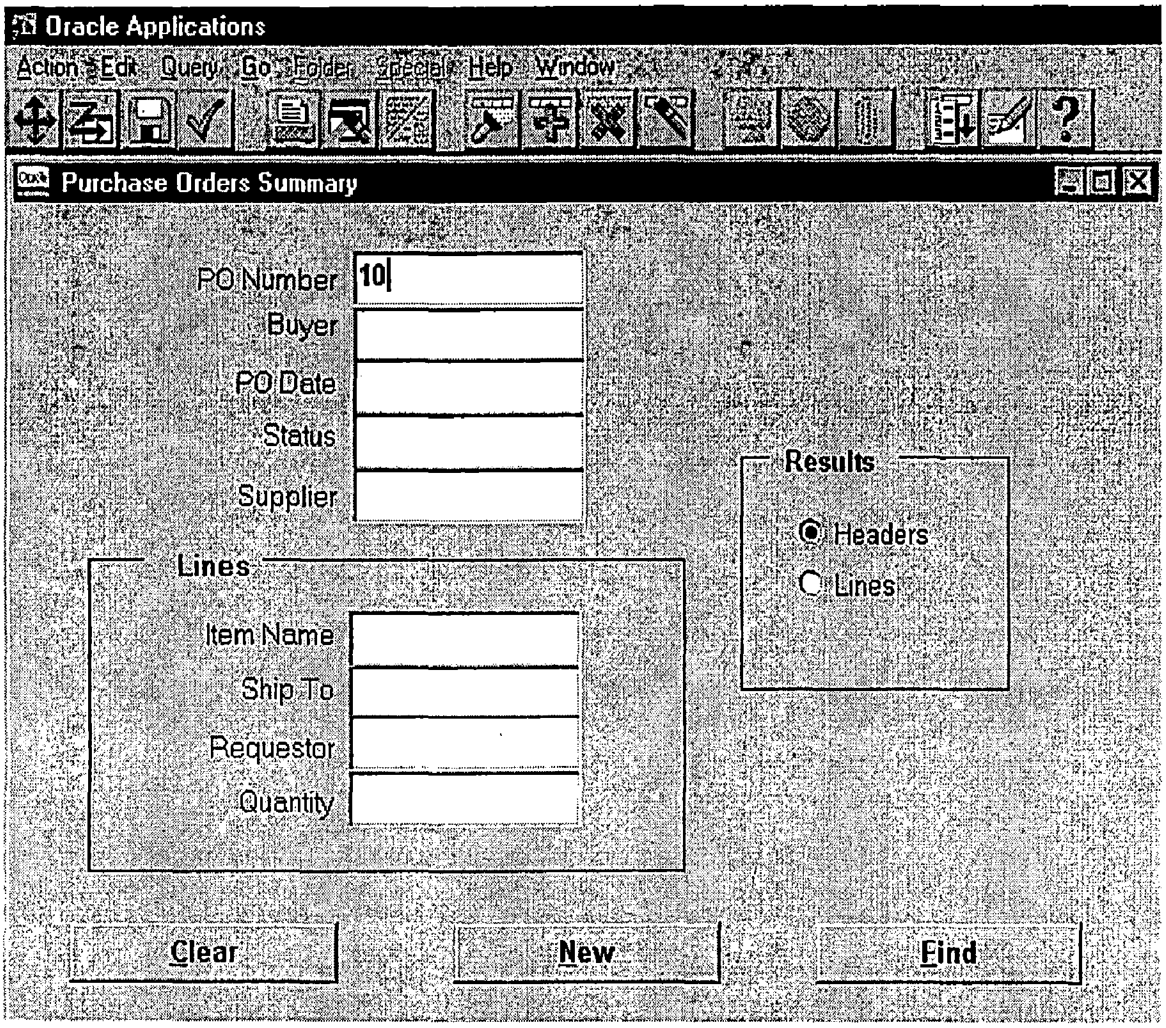
Comments

Context Value

Figure 5.28 Display of Descriptive flex field Window

Descriptive flexfields can display variable items in the window depending upon the context information. They can be optional or mandatory, depends upon the configuration.

Purchase Order Summary :



PO Number 10
Buyer
PO Date
Status
Supplier

Lines

Item Name
Ship To
Requestor
Quantity

Results

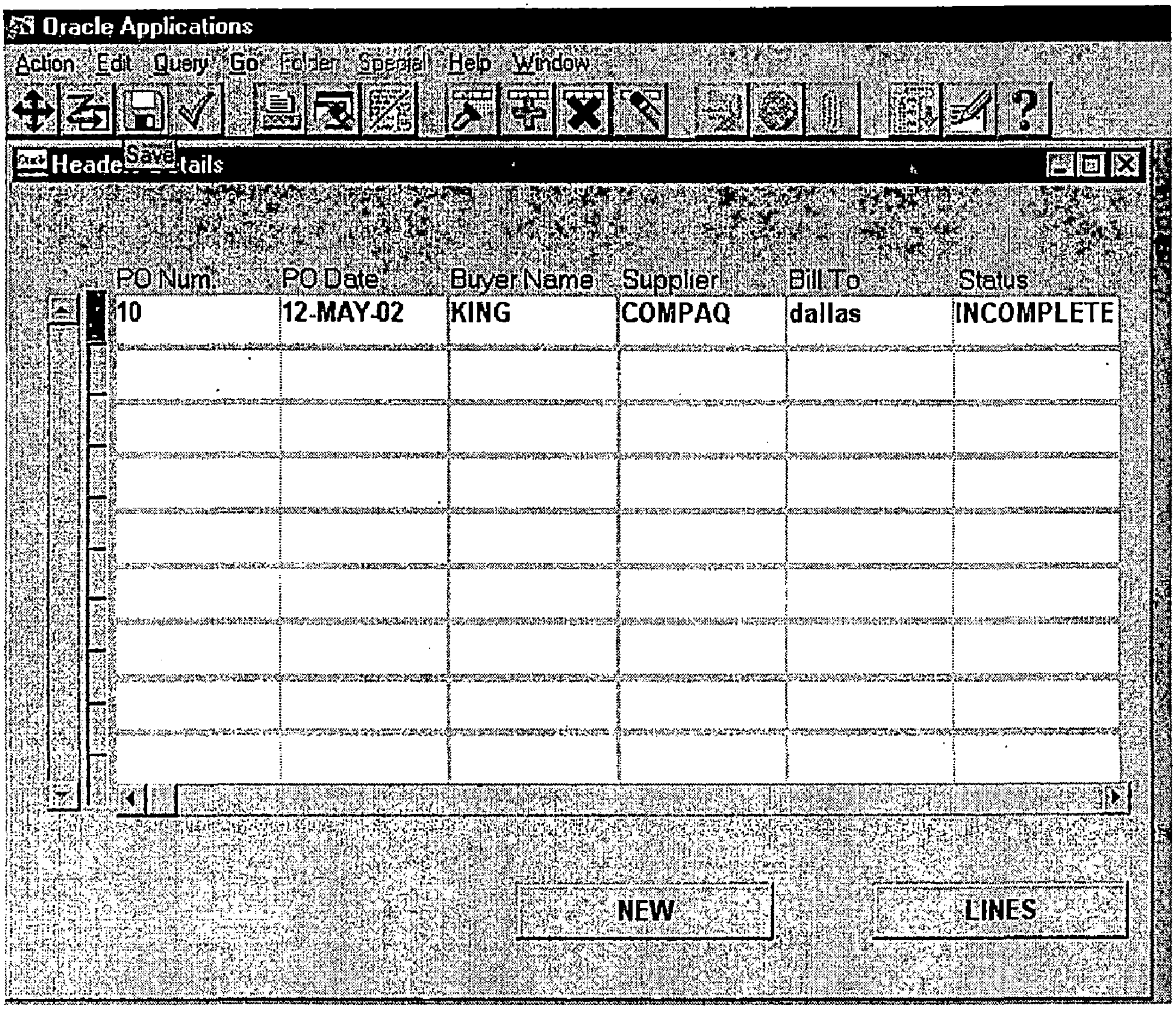
● Headers
○ Lines

Figure 5.29 Purchase Order Summary Window

As specified in specs a user can query the existing POs . He can base his search on basis of PO Number, Buyer , PO date, Status, Supplier.

He can also query the PO lines item on basis of Item name, Ship to ,Requestor and quantity.

PO Summary Headers Result :

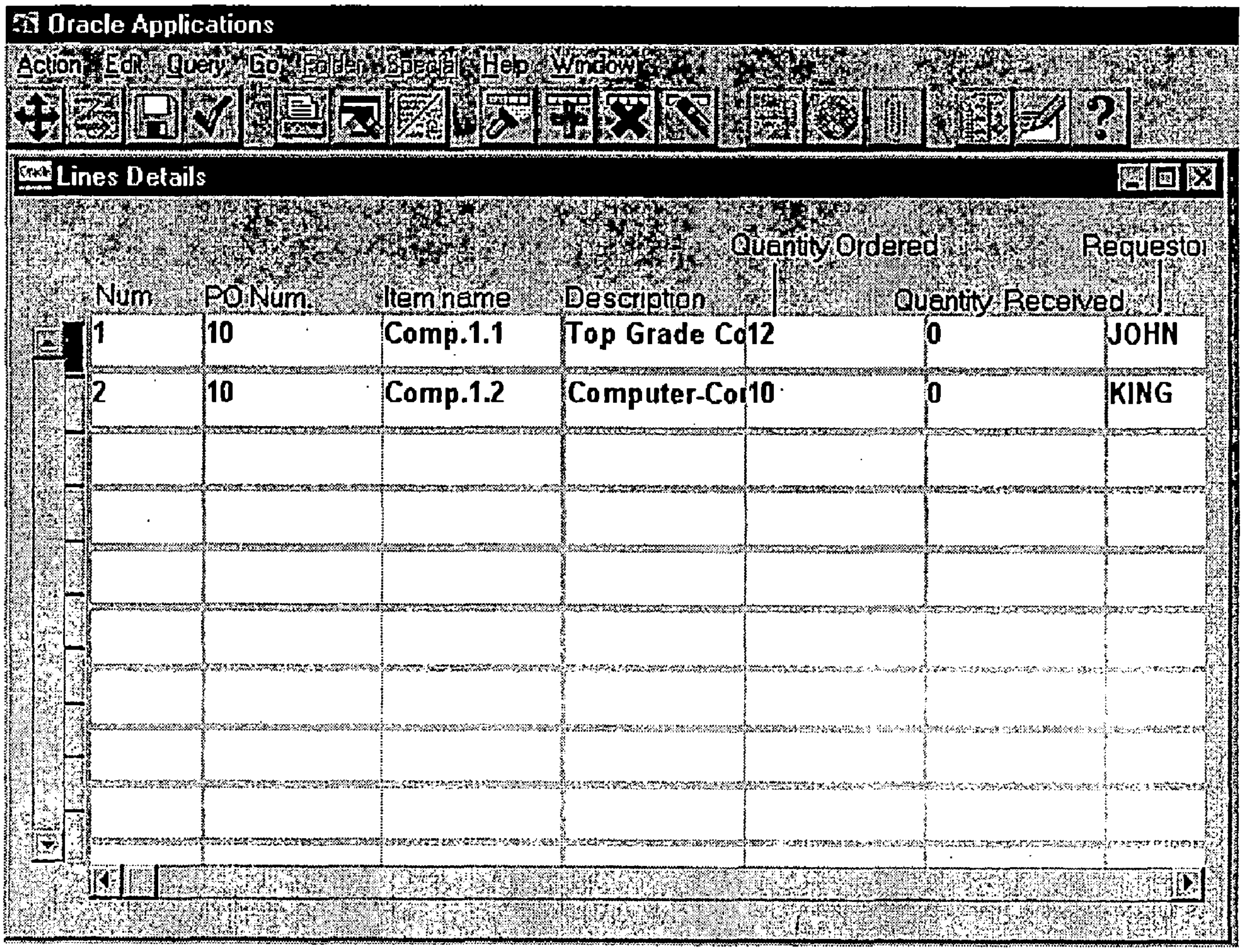


PO Num	PO Date	Buyer Name	Supplier	Bill To	Status
10	12-MAY-02	KING	COMPAQ	dallas	INCOMPLETE

Figure 5.30 Purchase Order Summary Result Window

Figure 5.30 shows the query result. We can directly view the corresponding lines by clicking on Lines button.

PO Summary Lines Result :

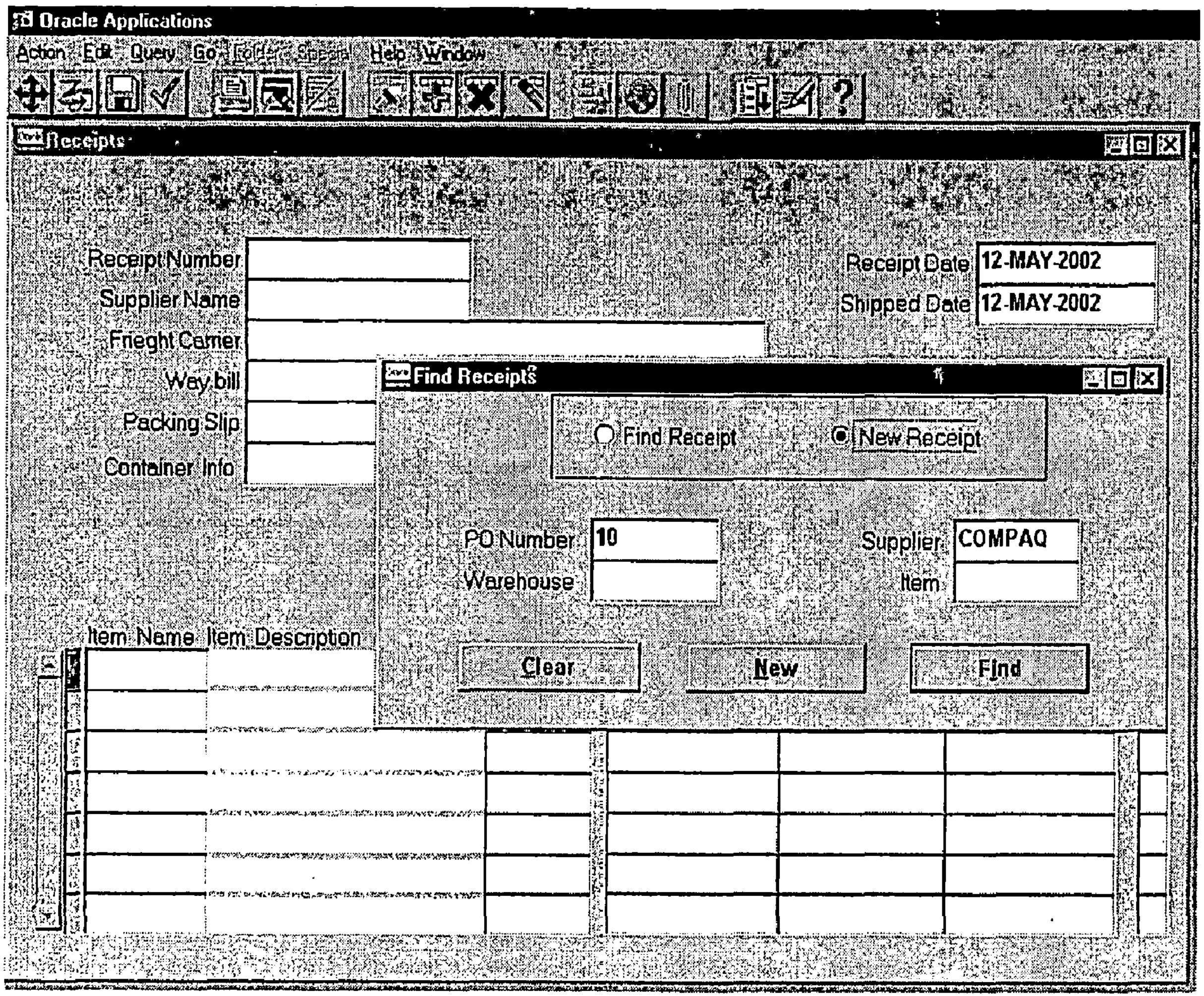


Num	PO Num.	Item name	Description	Quantity Ordered	Quantity Received	Requestor
1	10	Comp.1.1	Top Grade Co	12	0	JOHN
2	10	Comp.1.2	Computer-Co	10	0	KING

Figure 5.31 PO Summary Lines Window

Figure 5.31 shows corresponding Lines details. We can make a new PO by clicking on New button.

Receipt window :



Receipt Number

Supplier Name

Frieght Carrier

Way bill

Packing Slip

Container Info

Receipt Date 12-MAY-2002

Shipped Date 12-MAY-2002

○ Find Receipt ● New Receipt

PO Number 10

Warehouse

Supplier COMPAQ

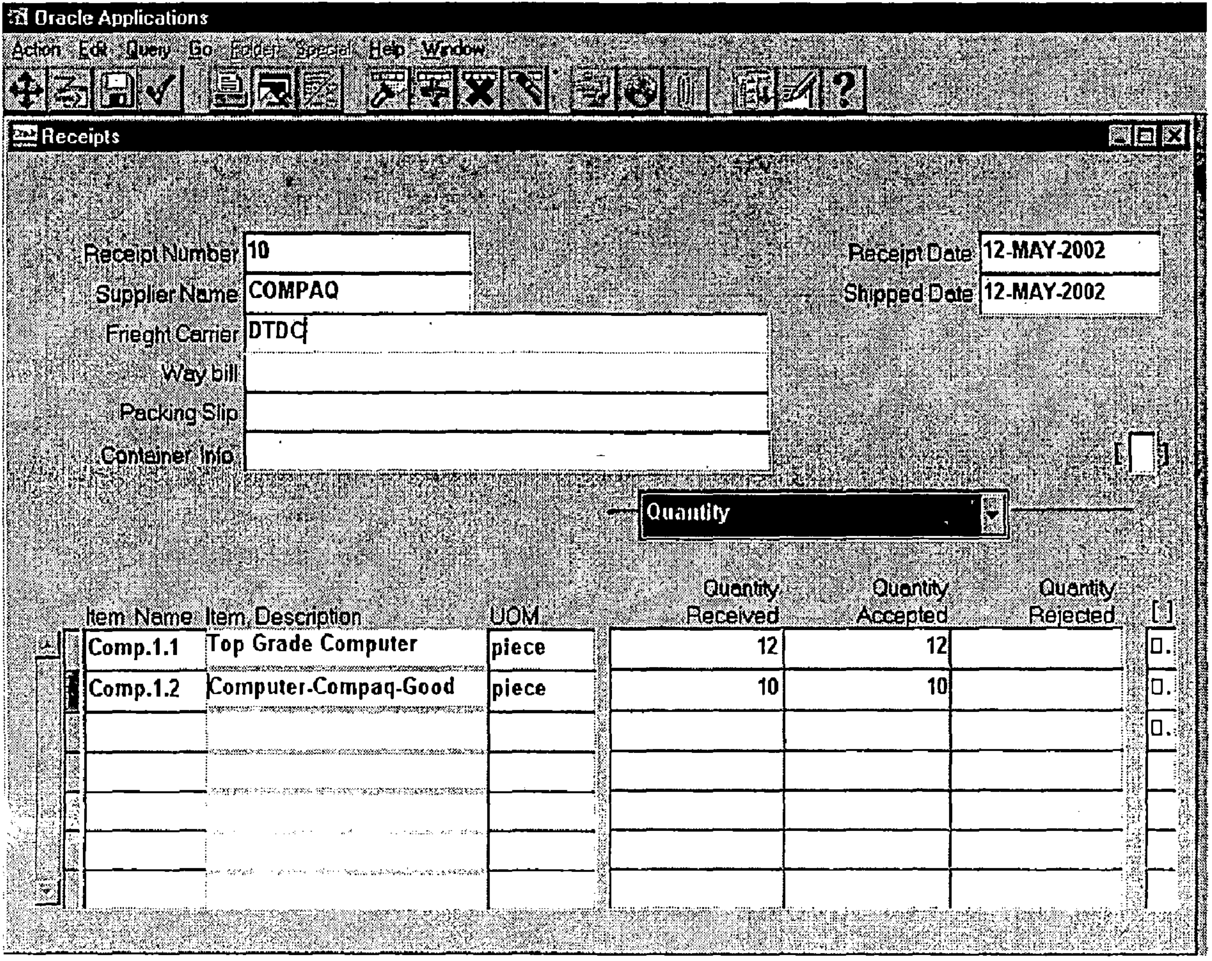
Item

Item Name Item Description

Figure 5.32 Receipts Entry Window

A user can enter the received items, he will choose a supplier to receive from, he can also choose against which PO the items are received. After then all the ordered items which fit this condition will populate the receipt lines field along with their ordered quantity and other details. Now the user can change the quantity received and quantity rejected. Then user can also receive some unordered items if he wants.

Received Items :



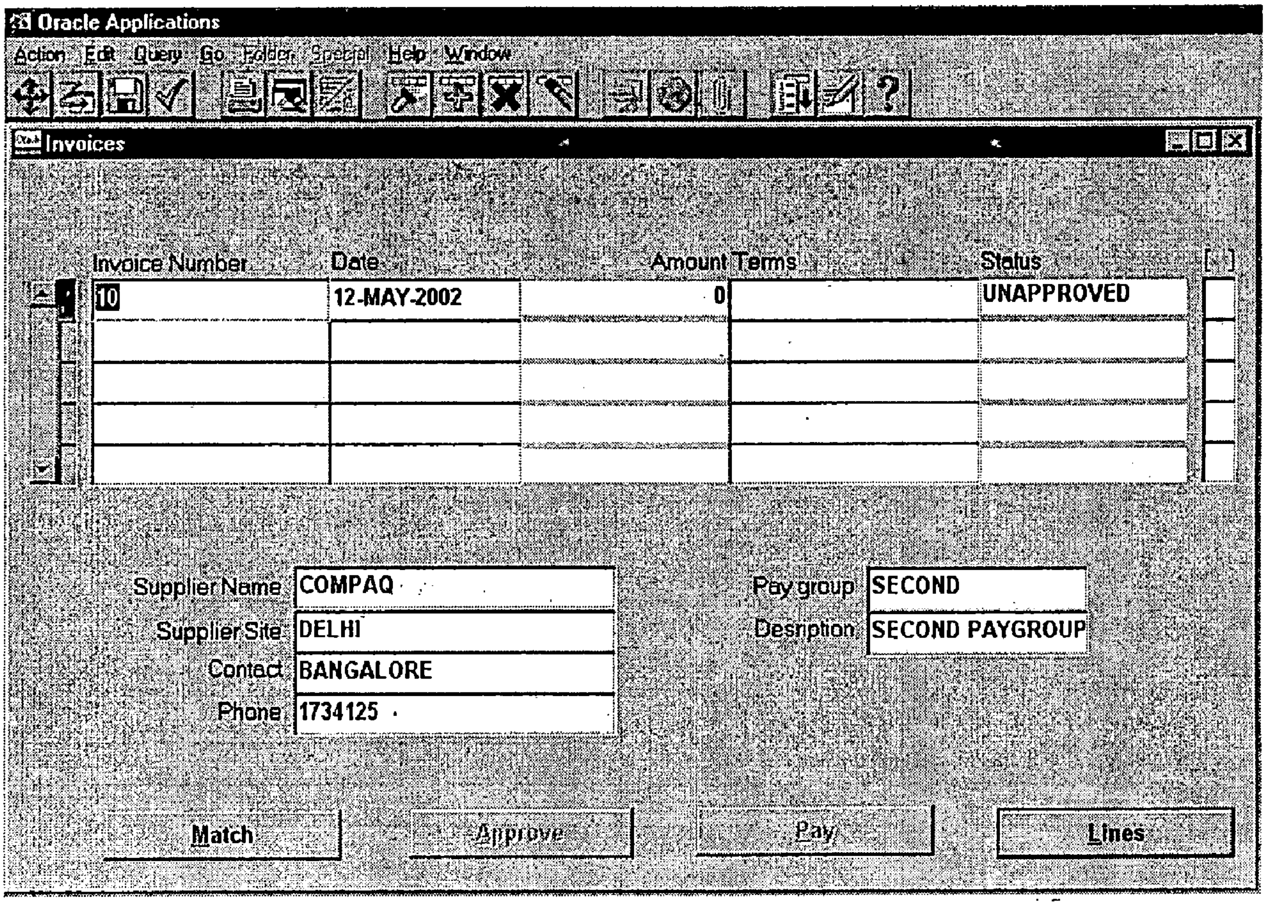
Receipt Number	10
Supplier Name	COMPAQ
Frieght Carrier	DTDC
Way bill	
Packing Slip	
Container Info	
Receipt Date	12-MAY-2002
Shipped Date	12-MAY-2002

Item Name	Item Description	UOM	Quantity Received	Quantity Accepted	Quantity Rejected
Comp.1.1	Top Grade Computer	piece	12	12	
Comp.1.2	Computer-Compaq-Good	piece	10	10	

Figure 5.33 Receipt Details Window

The user can now enter information like way bill no, packing slip, container info etc. He can lessen the quantity received but can't increment it. Figure 5.33 shows the Received items.

Invoice Window :



Invoice Number	Date	Amount	Terms	Status
10	12-MAY-2002	0		UNAPPROVED

Supplier Name COMPAQ

Supplier Site DELHI

Contact BANGALORE

Phone 1734125

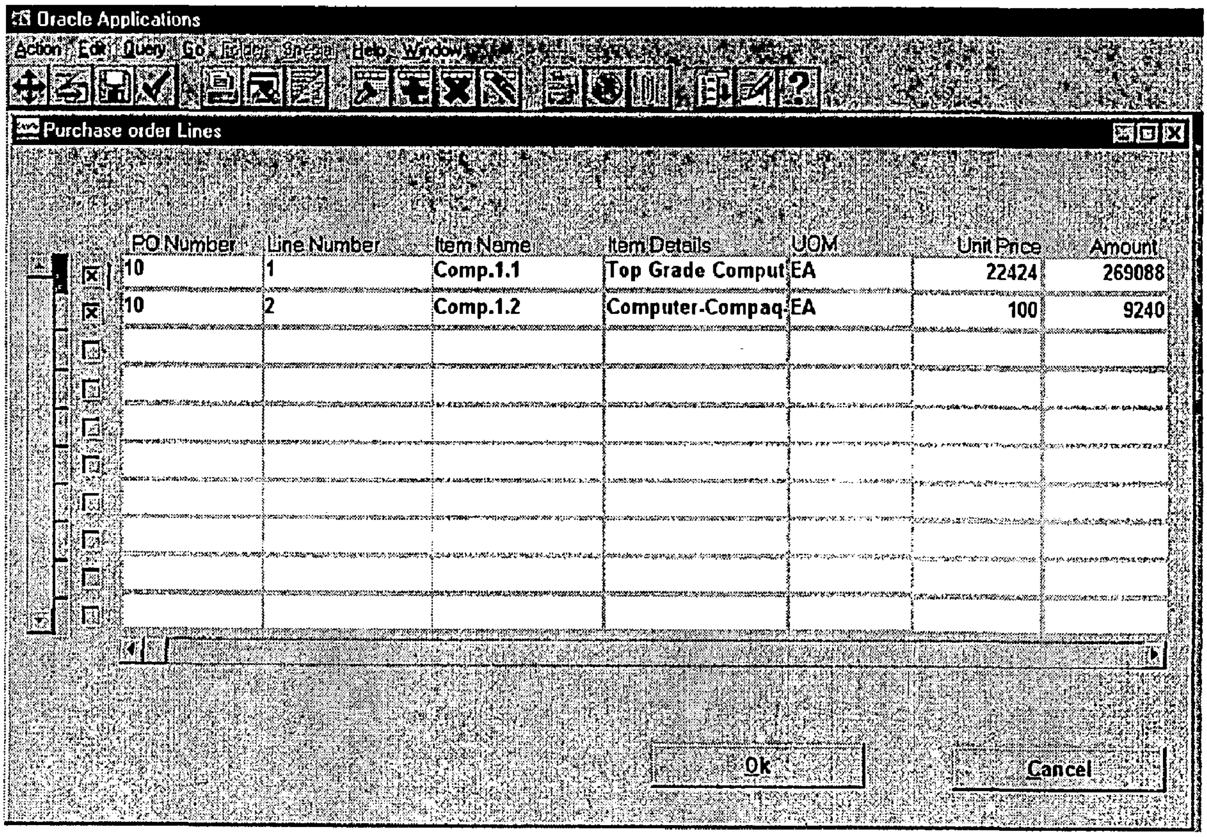
Pay group SECOND

Desription SECOND PAYGROUP

Figure 5.34 Invoice Window

The user simply enters supplier name. All the ordered item details against that supplier will populate invoice lines block. Figure 5.34 shows entry of supplier "COMPAQ". As soon as the user presses match button all the items ordered to Compaq will be listed as shown in next figure.

Invoice Matching Window :

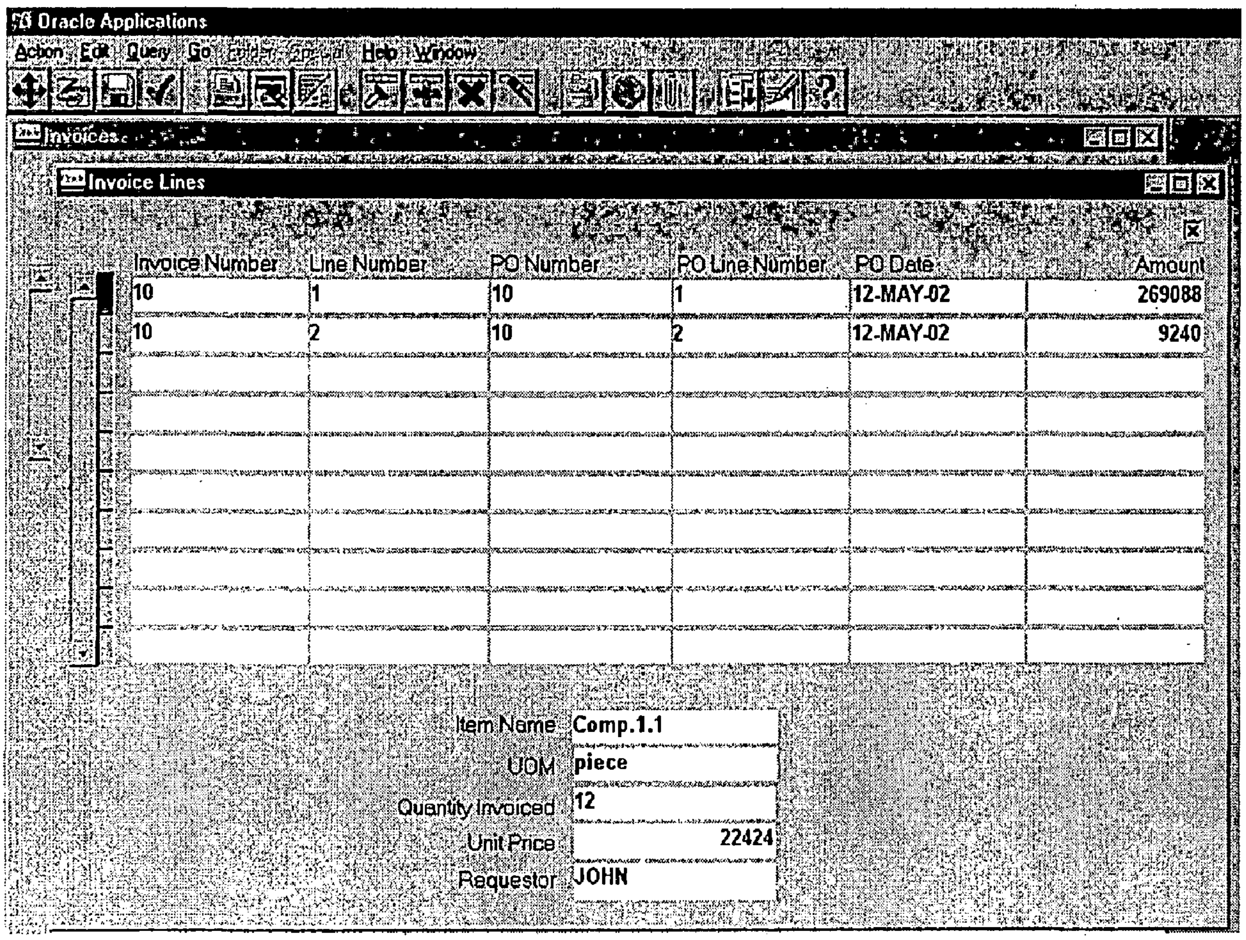


	PO Number	Line Number	Item Name	Item Details	UOM	Unit Price	Amount
☒	10	1	Comp.1.1	Top Grade Comput	EA	22424	269088
☒	10	2	Comp.1.2	Computer-Compaq-	EA	100	9240
☐							
☐							
☐							
☐							
☐							
☐							
☐							
☐							

Figure 5.35 Invoice Matching Window

The user can include or exclude a PO line from that invoice by clicking on the check box as shown in Figure 5.35

Invoice Lines Window :



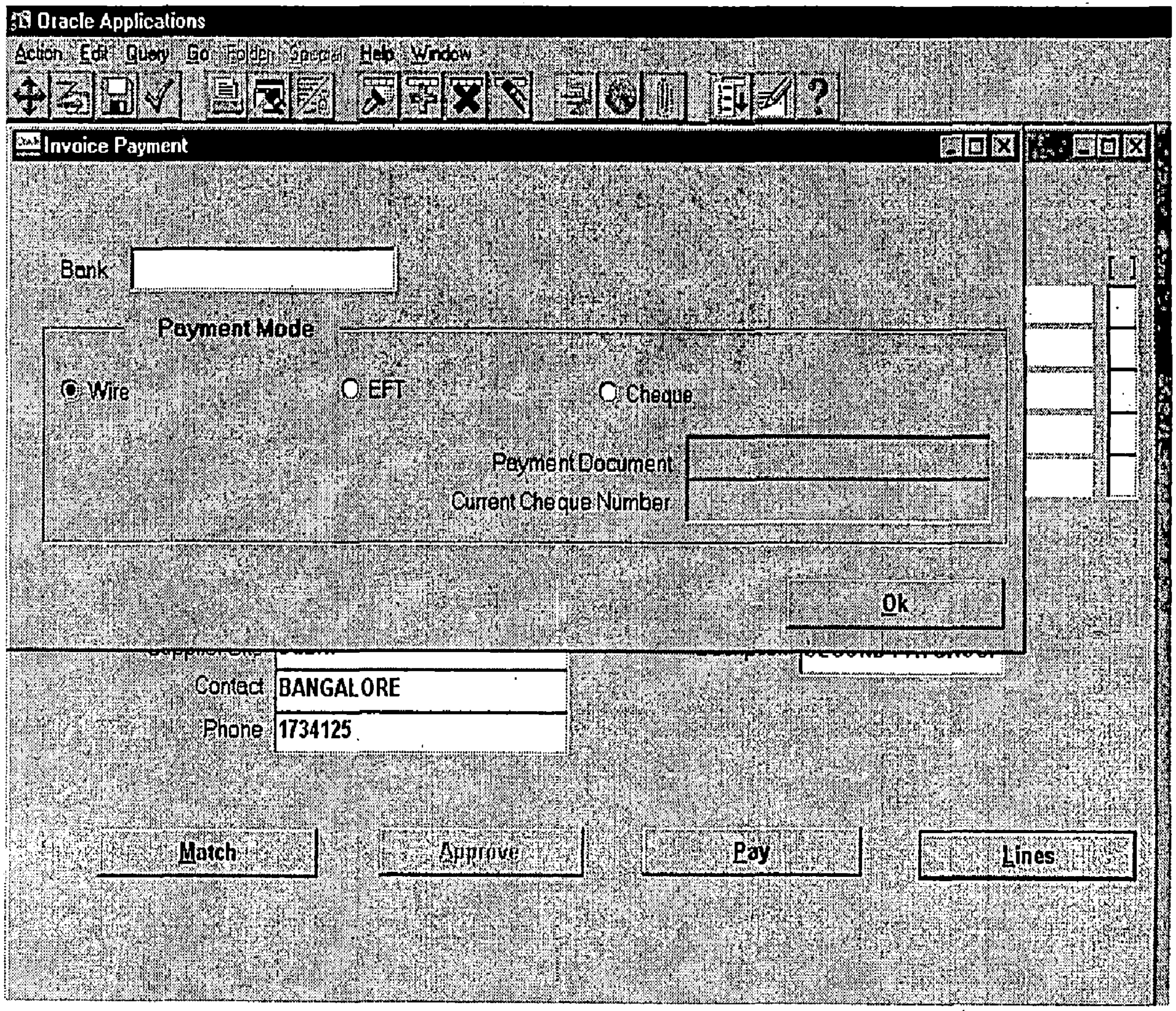
Invoice Number	Line Number	PO Number	PO Line Number	PO Date	Amount
10	1	10	1	12-MAY-02	269088
10	2	10	2	12-MAY-02	9240

Item Name	Comp.1.1
UOM	piece
Quantity Invoiced	12
Unit Price	22424
Requestor	JOHN

Figure 5.36 Invoice Lines Window

So, Figure 5.36 displays the invoice lines window, these are PO lines included in the invoice lines.

Individual Payment :



Bank

Payment Mode

● Wire ○ EFT ○ Cheque

Payment Document

Current Cheque Number

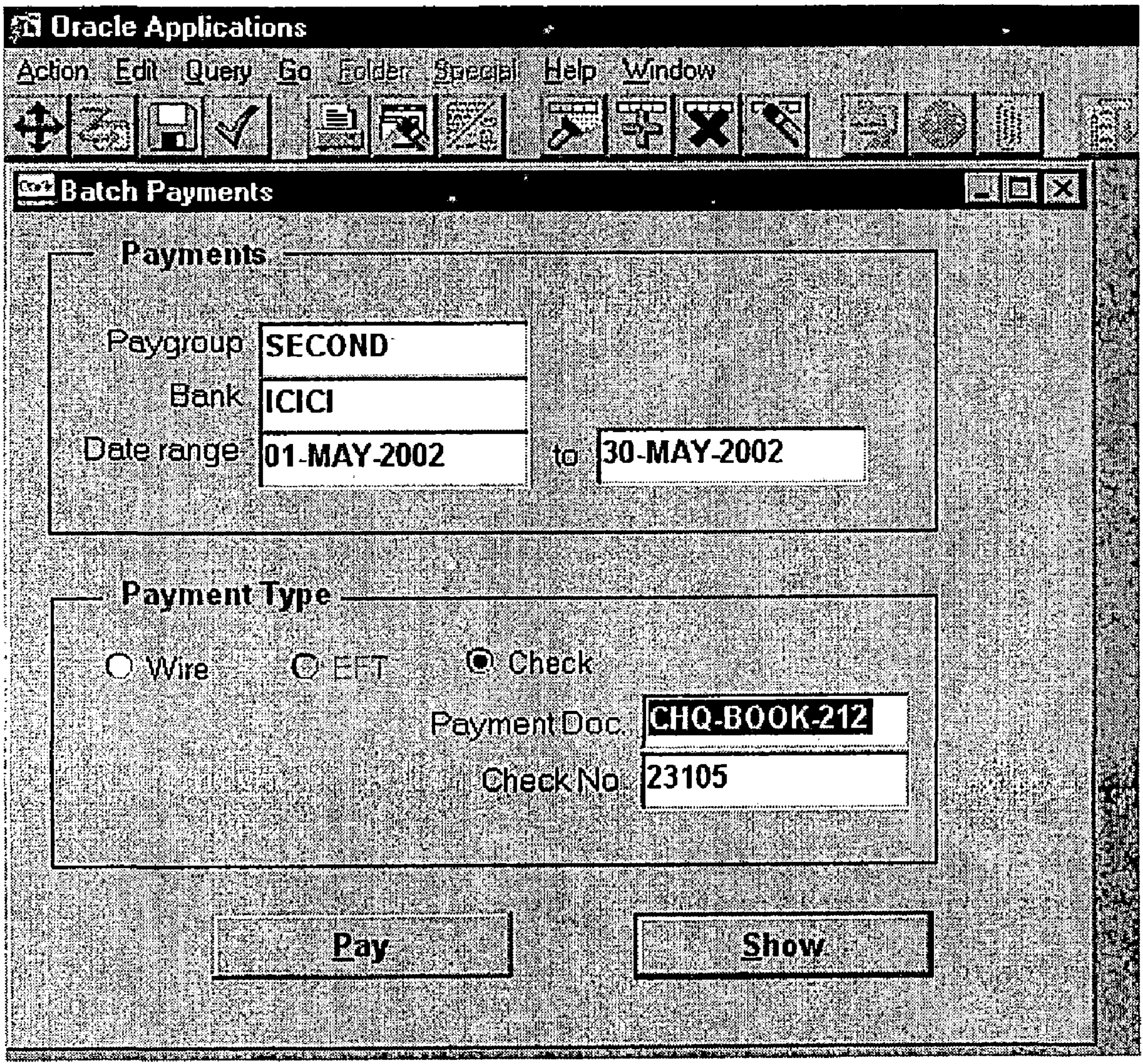
Contact BANGALORE

Phone 1734125

Figure 5.37 Individual Payment Window

The user can do payment instantly for that invoice from the form itself, or can do batch payment for al the invoice of suppliers of that paygroup.

Batch Payments :



Payments

Paygroup SECOND

Bank ICICI

Date range 01-MAY-2002 to 30-MAY-2002

Payment Type

○ Wire ○ EFT ● Check

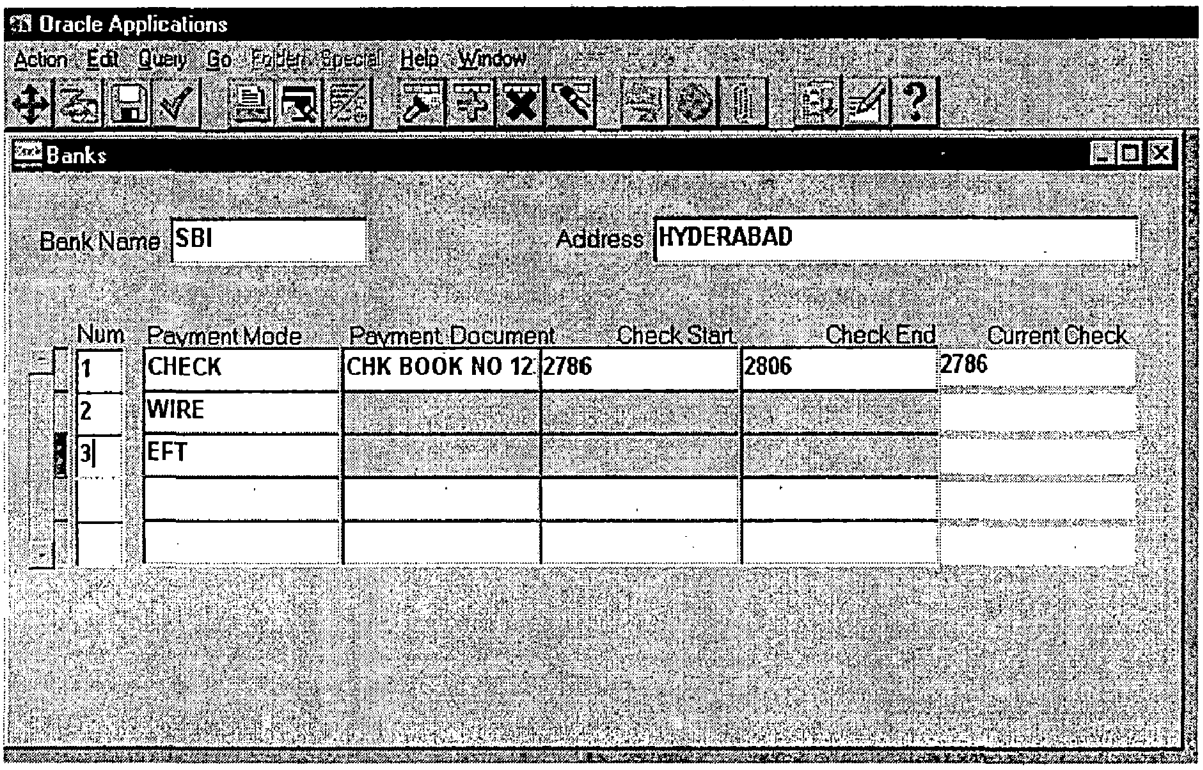
Payment Doc. CHQ-BOOK-212

Check No. 23105

Figure 5.38 Batch Payment Window

The user can do batch i.e. bulk payment by selecting the paygroup and bank name. All suppliers pertaining to that pay groupwill be paid against their Invoices.When the user enters 'Pay' button , it gets paid to all those supplier's invoices which match this criteria.If user presses 'Show ', it will show all the invoices matching that criteria.

Banks Window :



Bank Name SBI

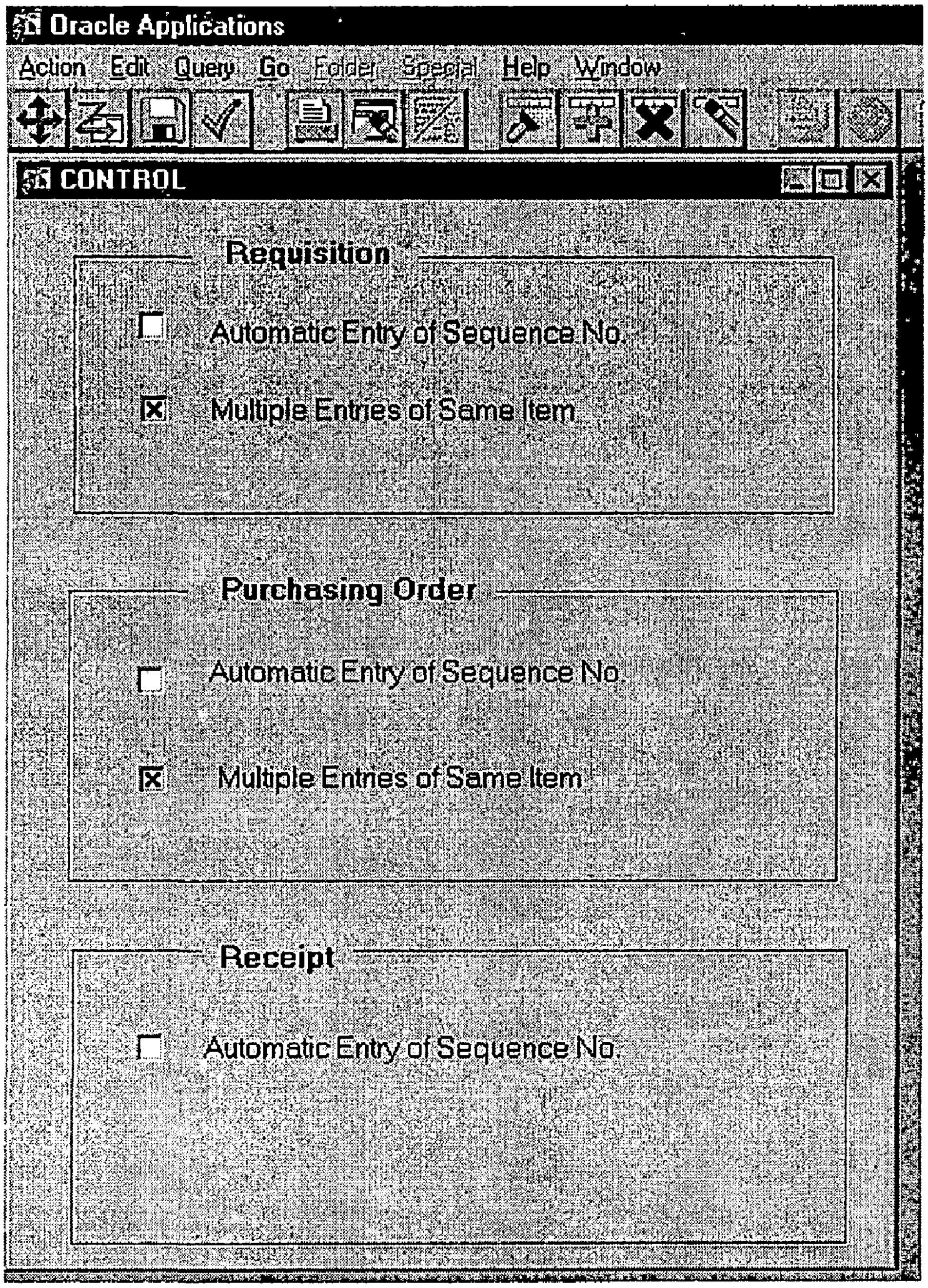
Address HYDERABAD

Num	Payment Mode	Payment Document	Check Start	Check End	Current Check
1	CHECK	CHK BOOK NO 12	2786	2806	2786
2	WIRE				
3	EFT				

Figure 5.39 Bank Entry Window

Apart from seeded banks data , the user can enter any no of banks here and their method of transaction.

Control Form :



Requisition

□ Automatic Entry of Sequence No.

☒ Multiple Entries of Same Item

Purchasing Order

□ Automatic Entry of Sequence No.

☒ Multiple Entries of Same Item

Receipt

□ Automatic Entry of Sequence No.

Figure 5.40 Control form Window

Here we can choose to change the profile settings, We can here specify whether Requisition Number ordering is automatic or not, or we can specify whether we can have multiple lines for the same item or not.

5.16 **Testing and TestPlan**

Testing

Developing a complete system consumes sufficient quantities of both time and money. A number of problems are likely to be encountered in the entire process. The designer finally lands up with the product, which in turn conforms, to all specification.

Test Plan

The various kinds of testing done in order to eliminate all the possible bugs and errors in the system are a) Unit Testing b) Integration Testing and c) System Testing.

The various modules are:

- Purchasing
- Payables

Unit testing is done for each of the above modules, which took a time duration of 2 to 3 days. Integration testing was done with combining the various modules of the system to see if they are compatible. After the successful completion of these two tests the system in its whole is tested and this is known as system testing.

5.17 Testing Methods

5.17.1 Unit Testing

At the lowest level, the function of the basic unit of software was tested in isolation. This is where the most detailed investigation of the internal workings of individual units was carried out. The purpose of unit testing was to find errors in the individual units, which could be data or logic related errors.

Activities followed in Unit Testing were

- Ensuring that instructions related to the test cases were executed properly
- Verification of operation at normal value range
- Verification of operation outside range values
- Verification of program execution at boundary conditions
- Ensuring that all loops terminated normally
- Identification and removal of abnormal termination of all loops
- Ensuring that all errors were trapped

5.17.2 Integration testing

The individual units were linked as per the design specification. After the completion of the integration errors were traced in two ways

- In the interfaces between the units
- The functions, which could not be tested during unit testing

5.17.3 System testing

After integration testing was completed the entire system is tested as whole. In this process search for errors in the end-to-end functionality of the system and security features was conducted. The working of the code as per the Function Specification was tested in this process.

CHAPTER – 6

CONCLUSION AND SUGGESTIONS FOR FUTURE WORK

6.1 CONCLUSION

The Project Work "ENTERPRISE MANAGEMENT SYSTEM USING ORACLE APPLICATIONS" has the objectives of computerizing the Purchasing and Payables parts of a Trading Company. All the manipulations are carried out in a efficient manner.

The system is developed in such a way that further modifications can be made very easily with minor changes. It is very much compatible with ORACLE applications platform .This project work is extensible i.e. we can add more modules to it as it is highly modular in nature.

6.2 SUGGESTIONS FOR FUTURE WORK

Although care has been taken while building the project so that all the specification can be met ,but there is always a room for improvement. The suggestions for later use are as follows:

1. We can't have multiple requisitions for a single P.O. , so this change can be incorporated

2. We can have only one bill to location for a PO , we can't have multiple bill to locations, one for each PO line.

ORACLE APPLICATIONS GLOSSARY

APPLETVIEWER A program residing on a client machine that runs and displays a Java applet or application.

BUGFIX is a correction to functionality. Bug fixes are generally tagged with the BUGDB bug number of the issue that they fix.

FEATURE is an increase in advertised functionality. Features must have names. FEATURES can be tagged with the BUGDB bug number of the enhancement they implement.

Java applet is a program, typically small in size, written in the Java programming language that is downloaded and run by a web browser or appletviewer.

MINIPACK is a merged collection of all STANDARD UPDATES related to a product at a given point in time.

ORACLE An Oracle Server database. This generally refers to a database and the objects it contains, not to the Oracle Server executable files.

Oracle Applications System Administrator The person responsible for administering Oracle Applications security and tailoring system operation.

Oracle Server The database management system sold by Oracle Corporation. The term refers in general to the product executable files and/or the ORACLE databases created through those files.

ORACLE_SID An environment variable that identifies an ORACLE database.

PATCHSET is a merged collection of individual UPDATES.

Patch driver A file read by AutoPatch that lists the actions required to apply a patch or release update. Examples of actions include copying a file, generating a form, or running a SQL script.

Platform Any individual operating system. Although most Oracle Applications procedures are the same cross platforms, some procedures vary. The latter procedures are called *platform-specific*.

Product Family Represents a group of related products. Examples of Product families are Financials, Manufacturing, Human Resources.

Product group A set of Oracle Applications products that uses a single installation of Oracle Application Object Library tables. Each product group can contain any number of Applications products..

ROLLUP is a merged collection of all STANDARD UPDATES related to a component at a given point in time.

UPDATE is any change to software. It can be a FEATURE, BUGFIX, PATCHSET, ROLLUP, or MINIPACK. UPDATE is synonymous with PATCH, although PATCH tends to imply a BUGFIX. STANDARD UPDATES are meant for general consumption by all customers. STANDALONE UPDATES have some special conditions or purpose that restrict their use.

Uniform Resource Locator An address used to uniquely identify a document on the World Wide Web. An example of a URL is **http://www.oracle.com.**

World Wide Web (WWW) A network of machines running web servers that provide access to hypertext documents. The network may consist of machines on the Internet, a corporate *intranet*, or a combination of both. Also called simply "*the Web*."

[1] 11i Oracle Applications Architecture, Internal Document, Edition 3.0, September 2001

[2] 11i Oracle Applications Developer Guide, Internal Document, Edition 3.0, September 2001

[3] 11i Oracle Applications Concepts, Internal Document, Edition 3.0, July 2001

[4] 11i Oracle Applications Architecture, Internal Document, Edition 2.0, September 2001

[5] Roger .S .Pressman ,"Software Engineering - A Practitioner's Approach", Published by : Mcgraw Hills , 5th Edition, New York, 2001

[6] SQL/PL SQL User Guide - Oracle Corporation

[7] Oracle Applications User Guide ,Internal Document, Edition 3.0, July 2001

Oracle websites:

[8] Online Documentation
htttp:\\www.otn.oracle.com

[9] Online Learning
htttp:\\www.ilearning.com

[10] Online Documentation
http:\\st-docs.oracle.com

[11] Online Documentation
htttp:\\www.etrm.us.oracle.com