

Oracle 9iAS Forms and Reports Architecture

Garry Chan and Jeff Wong

Apr 1, 2003

Table of Contents

1	INTRODUCTION	3
2	ENVIRONMENT	4
2.1	Hardware	4
2.2	Software	5
2.3	Development Tools	7
3	USER INTERFACE APPROACH	9
3.1	Forms	9
3.2	Application Security	14
3.3	User Authentication	15
4	AUDIT TRAIL	17
4.1	Oracle Designer Journaling	17
4.2	Application Maintained Audit Trail	17
4.3	Viewing Audit Trail Information	18
5	REPORTING AND BATCH SUBMISSION	19
5.1	Concurrent Processing – Reports	19
5.2	Reports Security	20
5.3	Concurrent Processing – Jobs	21
5.4	Jobs / Reports Submission Control	21
6	BACKUP AND RECOVERY	22
6.1	Approach	22
6.2	Backup Strategy	22
7	APPENDIX	24
7.1	References	24

1 Introduction

The purpose of this document is to describe the hardware and software architecture of an Oracle 9iAS architecture for an application.

2 Environment

2.1 Hardware

The following diagram represents hardware aspect of the technical architecture.

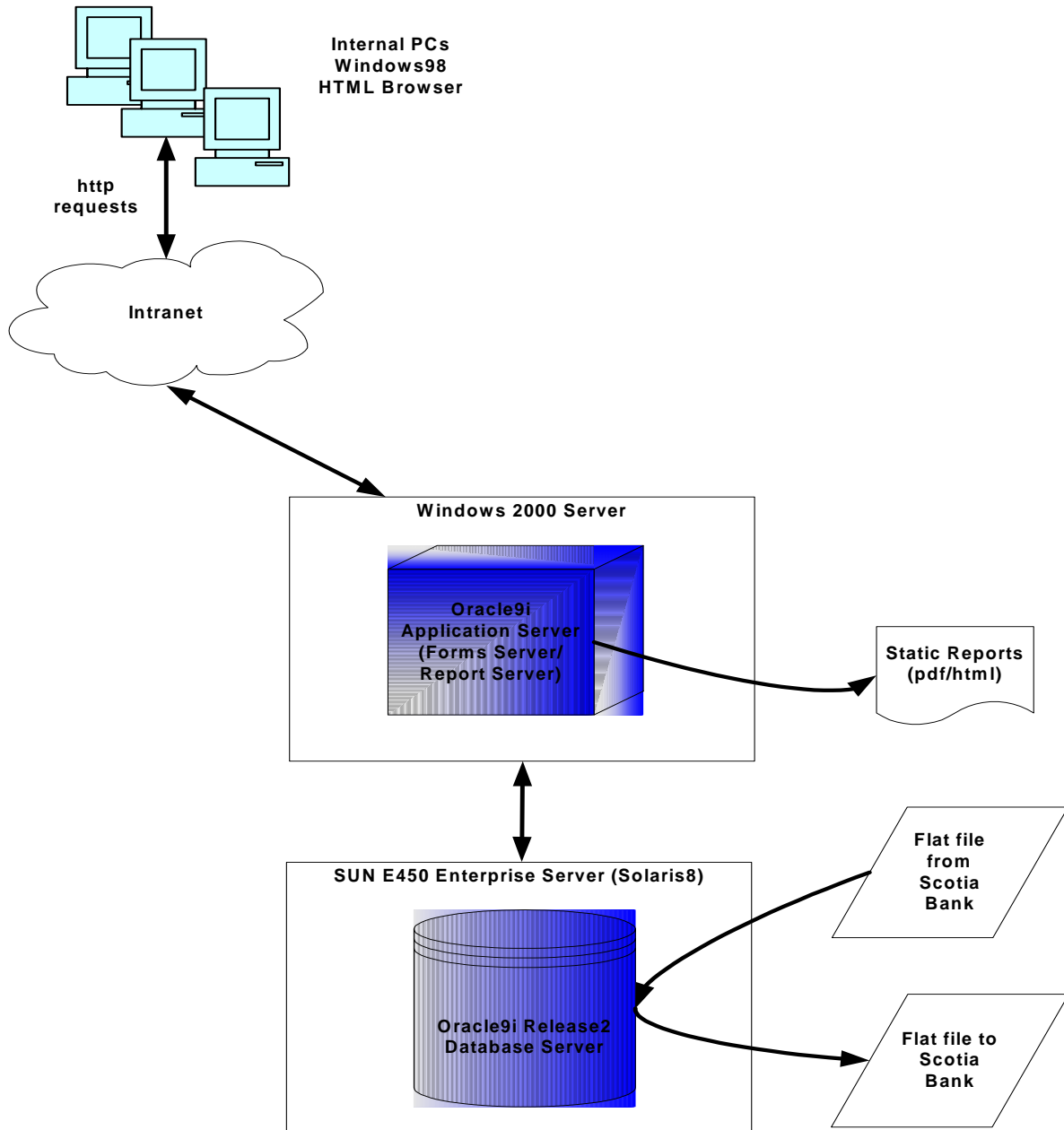


Figure 1: Architecture Overview

2.2 Software

Software includes:

1. Database tier: Sun Solaris 8 and Oracle Database 9i Release 2, Standard Edition.
2. Application tier: Microsoft Windows 2000; Oracle9i Application Server Release 2, which includes Oracle Forms Services and Oracle Report Services.
3. Client tier: Windows 98 or Windows XP with Microsoft Internet Explorer 6 (screen resolution 1024x768), and Adobe Acrobat Reader 5.1.

This application is designed for Intranet.. If the staff wishes to access the application from outside, the IT group needs to configure the firewall accordingly. The following figure depicts the software components and detailed flow of control:

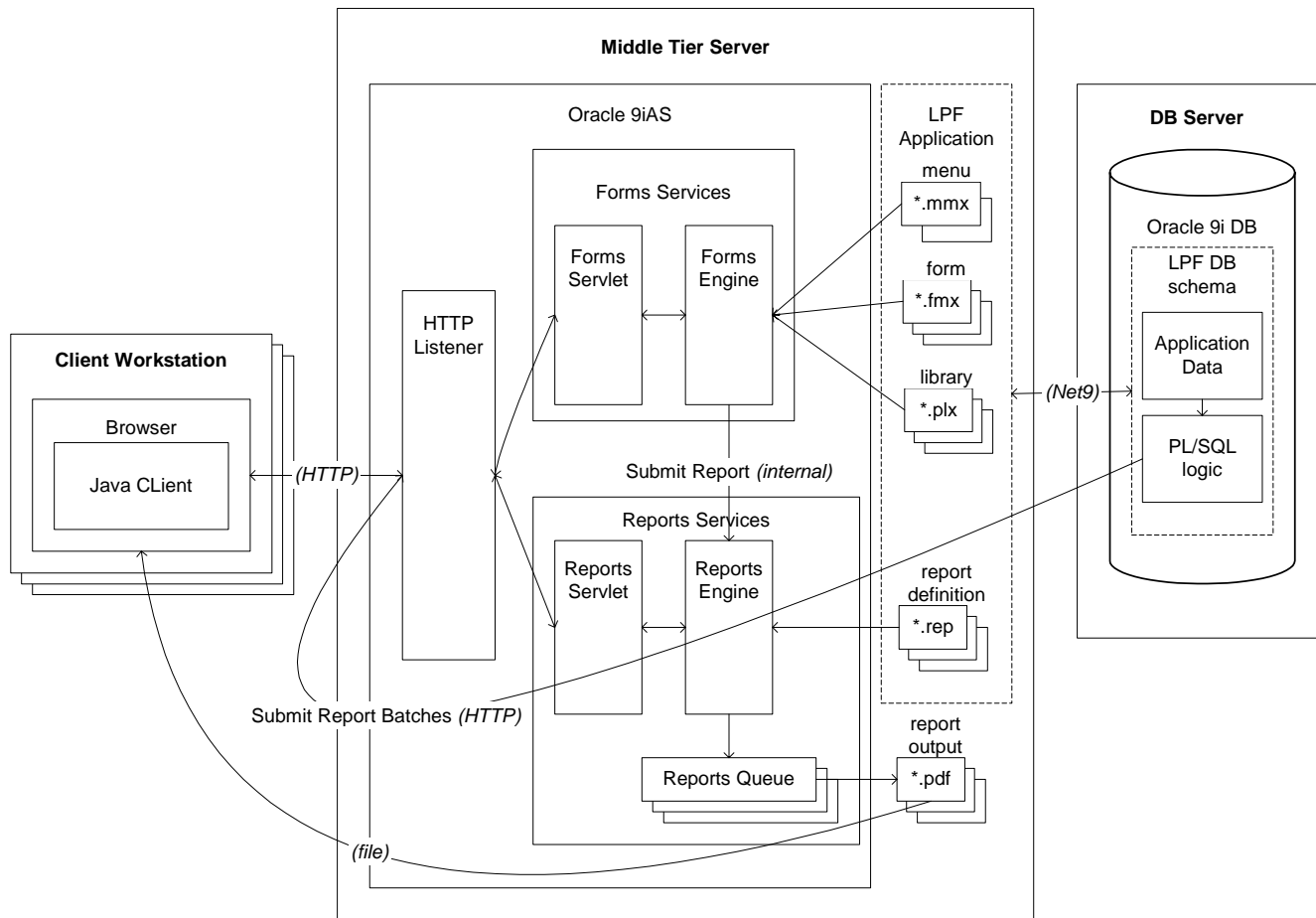


Figure 2: Software components and flow control

Component	Description
Java Client	The Forms Java Client consists of a set of generic Java classes that are downloaded to the client the first time a user calls a Forms application. Once loaded, the generic Java classes of the Forms client are cached permanently on the user's browser, to be renewed only when a new Forms client version is detected on the Application Server. The supported client is Microsoft Internet Explorer 6.
HTTP Listener	HTTP Listener is the HTTP listening entry point to Oracle9iAS. In Oracle9iAS Release 2, it dispatches requests to invoke program logic written in PL/SQL via a standard Apache Web Server mod architecture. Module Mod_oc4j acts as a connector to route requests to and responses from Oracle Forms and Reports servlets.
Forms Service	This is used to dispatch Oracle Forms modules.
Forms Servlet	The Forms Servlet is part of the Forms Service. It interacts with a user's web browser. The Forms Servlet simplifies the administration of different Forms applications handled by one Forms Services installation. The Forms Java Client communicates back to the middle tier, accessing the Forms Servlet to start up a dedicated Forms Web Runtime process for each user session. The Forms Web Runtime is started in an environment defined by a configuration file for all custom applications.
Forms Engine (Forms Web Runtime)	The business logic itself is saved in Oracle Forms executables (.fmx files) and PL/SQL library .plx files. The Forms Web Runtime Engine executes both file types and connects to the database using Oracle Net9. While it executes the business logic, the Forms Engine sends metadata messages to the client instructing it to render the user interface.
Reports Service	This is used to execute Oracle Reports executables.
Reports Servlet	The Reports Servlet is part of the Reports Servlet. It translates and delivers information between HTTP and the Reports Engine.
Reports Engine	The Reports Engine processes client requests, including steps such as authentication, scheduling, caching, and distribution e.g. PDF or HTML format. Reports Engine fetches requested data from the data source, formats the report, sends and fetches completed reports from the Reports cache, and notifies the client that the job is ready.
Reports Queue	Reports Services Queue keeps track of all jobs submitted to the server, including jobs that are running, scheduled to run, finished, or failed.
Net9	Net9 is the connectivity protocol provided by Oracle to connect the applications to the Oracle database server, and returning data from the database server back to the application.
Oracle 9i Database	Stores and manages the application data and logic.

2.3 Development Tools

The following development tools will be used for the project:

- a. Oracle9i Designer
- b. Oracle9i Forms Builder
- c. Oracle9i Reports Builder

The core development tool is Oracle9i Designer, which is a CASE (Computer Aided Software Engineering) tool. Oracle Designer contains a *Central Repository* that stores the definition of all database objects (Server Model), definitions of the Oracle Forms and Reports modules and Generator Preferences to control the process of generation.

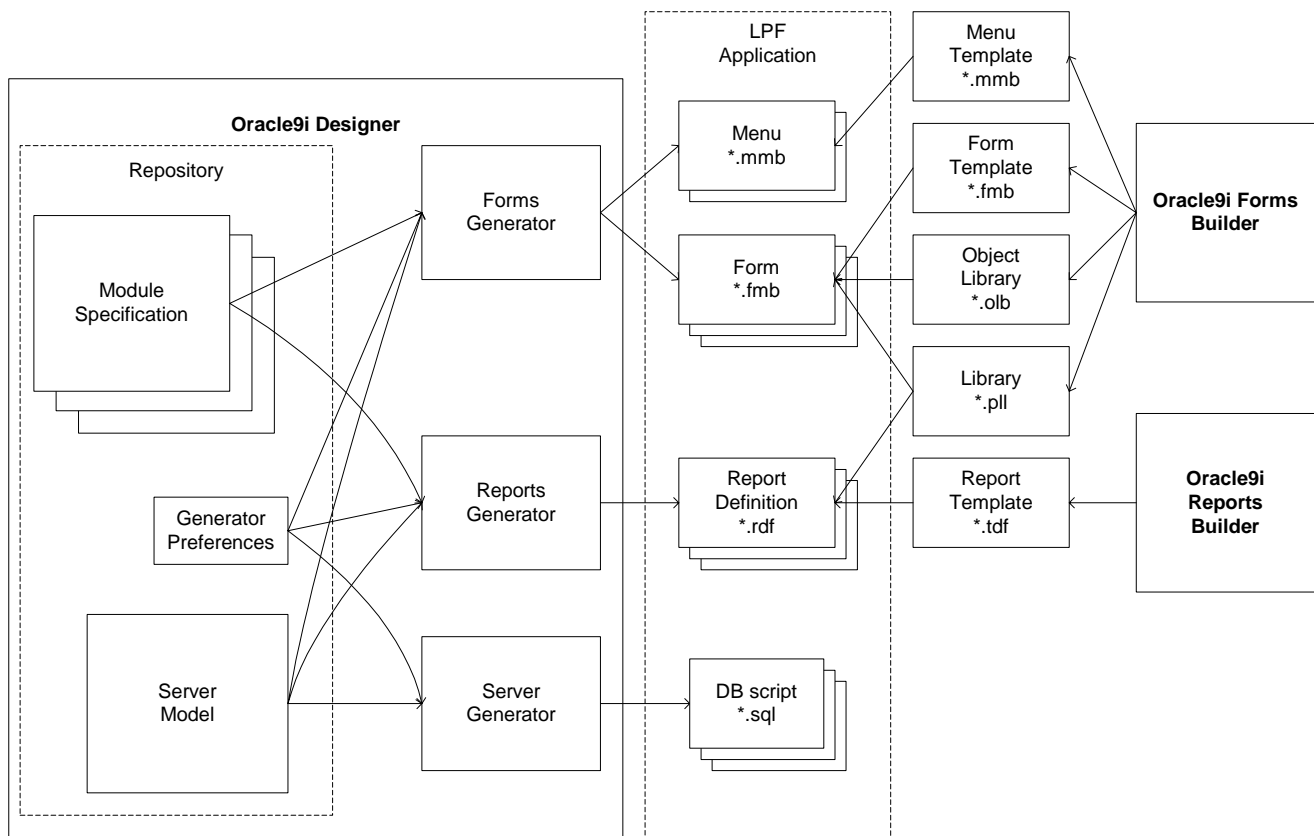


Figure 3: Development Tools

We will take advantage of the following features of Oracle Designer:

- a. Server Generator – generate all Data Definition Language (DDL) scripts for creation of the database structure.
- b. Forms Generator –capture and design Oracle Forms modules for the pension system with the goal of minimizing Forms customization outside Oracle9i Designer. This approach is in compliance with the Oracle methodology and simplifies maintenance. Oracle9i Forms Builder will be used to create and maintain templates and client libraries needed for the generation process as well as the runtime environment. In the event when it is difficult to generate a particular form without extensive customization, we may choose to simply develop the entire form outside Designer, and bring its definitions back to Designer for documentation purposes.
- c. Reports Generator – generate reports from Oracle9i Designer. Once generated, these reports may require further customization using Oracle9i Reports Builder. Once again, the intent is to keep customization outside Designer to a minimum. In the event when it is difficult to generate a particular report without extensive customization, we may choose to simply develop the entire report outside Designer, and bring its definitions back to Designer for documentation purposes.

3 User Interface Approach

This section describes the “building blocks” of the user interface, which will define the look and feel of the system. These building blocks specify the required terminology for discussing and documenting the user interface (e.g., “multiple-record form”, “tabular report” etc.)

In summary, Oracle Forms runs in a Java environment. As a result, the end users should note the following:

1. The user interface look-and-feel is different from the Microsoft conventions;
2. Conventions that are native to Microsoft may not be easy to replicate in Oracle Forms;
3. Further, Oracle Forms is extremely robust on data manipulation, transaction processing and database integration; however, it may not be as flexible as the other GUI (graphical user interface) tools in terms of user-friendliness.

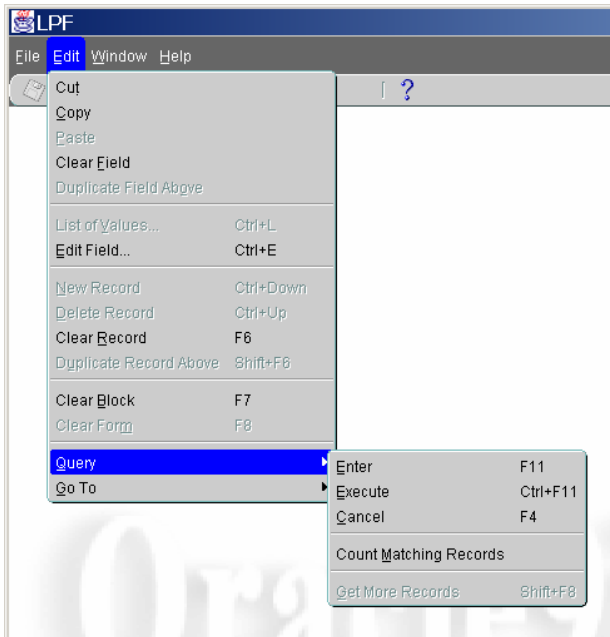
3.1 Forms

“Form” is a term used by Oracle Forms product to denote the building blocks of the user interface. Although, strictly speaking, one “form” can consist of multiple windows, for this project, we will use the term “form” interchangeably with “screen” and “window”.

3.1.1 Menu

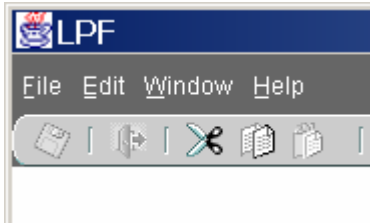
Each form has an associated drop-down menu. The menu can invoke three types of actions:

1. MS Windows standard GUI menu options (File, Edit, Window, etc. – on the high level and then Cut, Copy, Edit, etc.)
2. Form specific operations e.g. query database, create record, clear the form, next block, etc.
3. Form navigation, which defines the modules that can be invoked directly from the menu..



3.1.2 Toolbar

Each form also contains a toolbar, positioned directly below the form menu. The toolbar can be viewed as a subset of the menu, providing direct access to frequently used actions with a simple point-and-click. It typically contains buttons for the following standard Oracle Forms functions, such as “Save”, “Print”, “Clear Form”, “Delete Record” etc.



3.1.3 Blocks, Records, Items

A form consists of one or more blocks, each block consists of one or more records, and each record consists of one or more items.

A block usually represents a database table, so users can query the database data in the underlying table and, depending on the business logic, perform operations such as insert, update and delete. Blocks in a form can be organized in different layouts – see the section below on form layouts. The menu and the toolbar provide all the required functionality to navigate between multiple blocks in the form.

Each form record typically corresponds to a record in database (the table information is defined at the block level). Items can be columns in the underlying database table or “non-base table” items such buttons, display labels etc. Records and items have properties such as mandatory or optional, editable or non-editable and so on. Sometimes items can be combined into logical item groups to make block layout more understandable and intuitive (e.g. tab canvas – see below).

3.1.4 Interface Items

There are different types of representation for data items:

- Text item – used for entry of text; can be multiple-line text
- Display item – display-only text field; can be mapped to an underlying database column or a derived field
- Pick list (or pop list) – provides a list of allowable values for the field, can be hard-coded into the form or dynamically selected at runtime from a special reference codes table
- Combo box (suggested list) – a hybrid of a text item and a pop list. It allows a user to select a value from the list, or type a new entry directly into the field, with no validation against the list
- Radio group – suitable for a data item which has mutually exclusive, allowable values which are static throughout the life of the system e.g. gender codes can be displayed as radio buttons
- Check box – suitable for data items when only one value is applicable in a yes/no situation
- Buttons – for navigation or to execute a piece of functionality that performs actions on objects
- List of values (LOV) – a lookup window that shows all valid values for an item and allows to select from the list. This is similar to a pick list, except that LOV provides additional functionality as such wild card search.
- Alert – request confirmation from operators, warn operators of problems, or ask operators to make a decision.

3.1.5 Form Layout Components

Placing information in a form invariably requires a compromise between the desire to see many records of a table on one page, and to see all of the detail about each record at once. A summary view has the advantage of showing more records of an entity, but at the loss of detailed information about each record. Below are different presentation models, which can all be used in developing the UI.

Note that the purpose of these screen shots is to convey the presentation models; the detail layout such as colours and text alignment will be finalized during detailed form design.

3.1.5.1 Single Record Form

Single-record format allows the maximum number of fields for a single record to be displayed at once. In general, single-record formats should be used for the following cases:

- Detailed editing of a record e.g. member beneficiary data.
- The user must see many attributes as well as related information of one record at the same time e.g. display contribution details, but also display batch and receipts information.

3.1.5.2 Multi-record Form with Scrolling Region

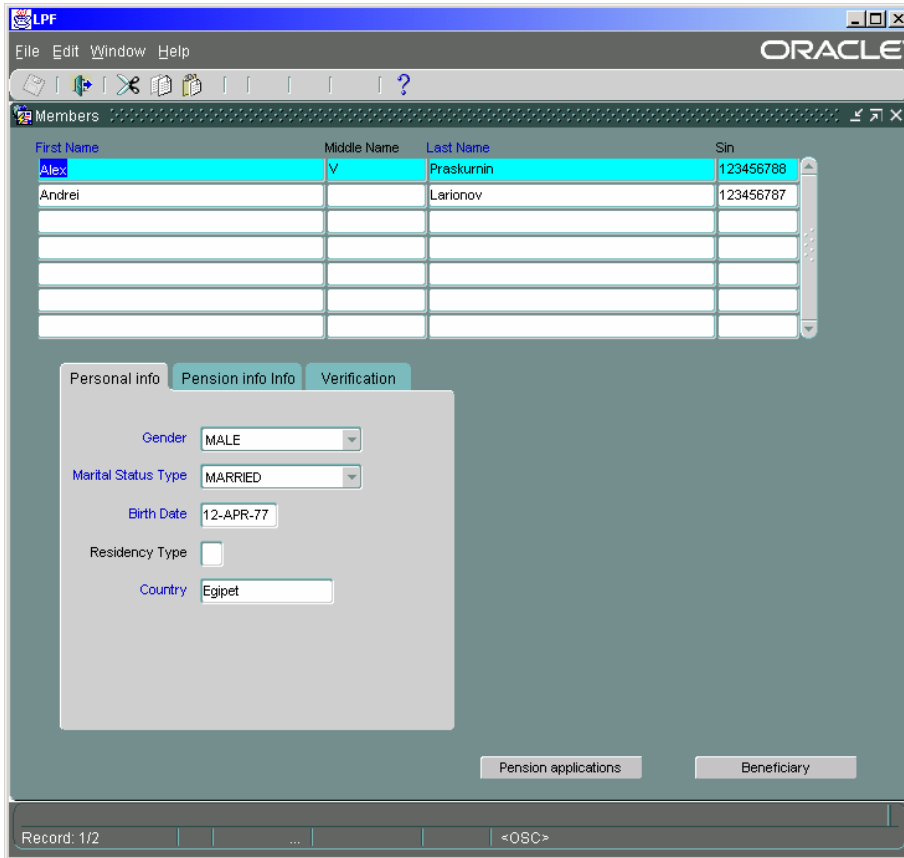
Multi-record format allows numerous records for a single table to be displayed at once. In general, multi-record formats should be used for any of the following cases:

- Displaying summary information for user to drill down into a single-record form e.g. display multiple member records as a result of a search.
- When the record that needs to be edited can be displayed in this tabular format.

If the record has more items than can be displayed on one line, a scrolling region or *spread table* can be used to make the additional fields visible, as shown below:

3.1.5.4 Tab Canvas Form

Presenting a table with a large number of fields is always a challenge. In such case the fields should be grouped to allow users to navigate easily and quickly find the needed information. A tab canvas can be used to group and display related fields of a block. This layout can be used for presenting multi-record form as well as master-detail forms – to be discussed in the next section.

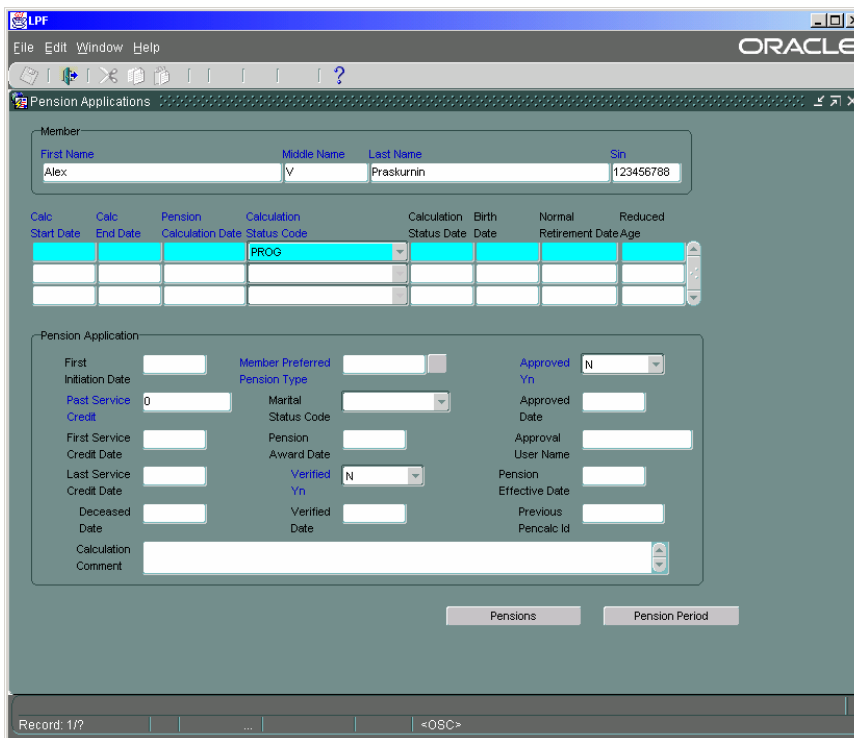


3.1.5.5 Master Detail Form (Multiple Blocks Layout)

Probably the most common layout for the new pension system will be the master detail form, which presents multiple blocks of data in the form window. This form is similar to the multi-record form with overflow area in terms of layout; however, the key difference is that the data for a master detail form comes from multiple database tables, with each block in the form corresponding to a database table. Further, the contents of the data blocks in a master detail form are synchronized i.e. the detail block depends on what is displayed in the master block. Typical usage of this form includes the following:

- Display member records as the master block, and beneficiary records as the detail block.
- Display pension calculation as the master block, and the pension records as the detail.

Note that the layout below shows three blocks of data: the top block is the master block, and the subsequent two blocks are the data records from the detail table.



The screenshot shows the Oracle LPF Pension Applications form. It is divided into three main sections:

- Member Information:** Fields for First Name (Alex), Middle Name (V), Last Name (Praskurnin), and Sin (123456788).
- Calculation Table:** A table with columns for Calc Start Date, Calc End Date, Pension Calculation Date, Calculation Status Code (PROG), Calculation Status Date, Birth Date, Normal Retirement Date, and Reduced Retirement Date.
- Pension Application Details:** Fields for First Initiation Date, Past Service Credit (0), First Service Credit Date, Last Service Credit Date, Deceased Date, Calculation Comment, Member Preferred Pension Type, Marital Status Code, Pension Award Date, Verified Yn (N), Member Preferred Pension Type, Approved Yn (N), Approved Date, Approval User Name, Pension Effective Date, Previous Pension Id, and Calculation Comment.

Buttons for 'Pensions' and 'Pension Period' are located at the bottom of the form.

3.2 Application Security

Once logged in, a pension application user may not access all available functionality in the application. The access control is implemented using application roles. (Note that this is different from database roles.)

3.2.1 Application Resources and Roles

We use the term “*application resource*” to denote all user interface objects that can be assigned separate (and independent) access privileges.

The new pension system supports the following types of application resource:

- Forms (windows)
- Data blocks

- Fields (data items)
- Reports
- Background job

Instead of granting access on each application resource directly to each user, we have introduced application or function roles to simplify the process. We will assign the application resource access to *functional (or application) roles* and then assign these roles to the application users. Functional roles should be defined to reflect the finest level of access restrictions that need to be set-up in the system. For example, rather than using roles “Pension analyst”, “Accounting entry person”, we would like to implement roles like “Create pension application”, “Approve pension application”, and “Verify the PSC”. The reason is that the “pension analyst” role is too broad a definition and whose privileges may overlap many other roles e.g. “pension supervisor” or “senior pension analyst”. We need to administer access privileges at a more granular level of details. Hence the use of functional roles will simplify privilege administration. Further, the use of functional roles implies that a user can be assigned more than one role.

Functional roles should be carefully defined during the design phase of the system.

Note: although access control within the form can be set-up at the data block or even data item level, such granularity should be avoided whenever possible because it will increase the complexity of the system, from both development and maintenance standpoints. For UI, we will try to set-up access at the form level and not to go down to the data block or item level, unless absolutely necessary.

3.2.2 Forms / Reports Security

After assigning access privilege for application resources to various functional roles, the new pension system will enforce application security accordingly. Upon invocation, each module (form or report) will check if a user trying to access it has the rights to invoke the module, and if so, the user can view or update a certain item.

For example, if user wants to open the Member Maintenance form, the form will first go through the list of roles assigned to the user and then check if access privilege on the form has been granted to one of those roles. If so, the form execution will continue; otherwise, the user will receive an error message and the form will be closed. A similar approach will be used for reports.

3.3 User Authentication

Because the system will have separate database and application tiers, there are two possible ways of implementing user authorization:

- Database user account: each user of the pension system as a separate database account in the Oracle database. The user credentials for the application will be passed to the database at the time of login.
- Application user account: each user is set up as an application user and will be maintained inside the pension application as a custom application user table. All application users who access the pension application will log into the Oracle database under the same database account.

We have investigated pros and cons of each approach, and here is the summary:

Metric	Pros and Cons
Licensing	Both options are considered to be the same by Oracle. Oracle counts “named users” when defining the license price. “Named user” is the same as the physical user who is allowed to access the system. Oracle does not offer “concurrent user” licensing option for database any longer.
Maintenance	Application users will require setting up additional tables and screens to maintain users; however, this could be maintained from within the application level.

Metric	Pros and Cons
	For database users, it would be enough to use standard tools and / or control commands provided by Oracle; however, this means that the database administrator must be involved every time a new user needs to be set up, distracting the administrator from more critical tasks such as monitoring database performance, and backup and recovery.
Scalability	The application users approach is a more scalable.
Implementation Efforts	Application users will require more custom code to be developed for forms, table triggers etc.

In spite of the additional efforts, we decided to adopt the “application user” approach due to the added application level control as well as scalability that this approach offers.

The following scenarios depict what user accounts need to be set up:

1. Deploy the pension system for production
 - a. Create a database account that will serve as the application schema. All front-end users will access the data using this database account, although each user will have his/her own application user account.
2. An end user that wants to access to the pension application
 - a. A new application user account needs to be set up by the authorized user (probably the IT staff).
3. An end user that needs to run reports via the pension application
 - a. Same as scenario 2.
4. An IT staff member who needs to access the database directly
 - a. Connect to the database using the database account and be granted the appropriate database roles.

4 Audit Trail

Audit trail, depending on the functional requirement, will be implemented by either turning on the journaling feature of Oracle Designer, or by using custom application logic and associated application tables.

Note that the audit trail facility tracks changes to the relevant application content, not the structure of the database.

Also note that all pension application tables have 4 audit columns to provide basic auditing capability: record creation user and time, and last modified user and time.

Note that the pension data cannot be protected from direct database updates by the authorized accounts, whether intentionally or by accident. Further audit trail can only be done if the updates are done via the proper channels. Hence, proper database backup and recovery may be required to recover the database under these circumstances.

4.1 Oracle Designer Journaling

Oracle Designer provides support for journaling data content changes. In essence, it creates the journal as well as the application logic required to populate these tables.

A journal table (or commonly known as a “shadow” table) is a database table that records details about each row that is inserted, updated or deleted in the main table. When the journaling option for a particular table is enabled (e.g. for table CONTRIBUTION), Oracle Designer creates a journal table (e.g. CONTRIBUTION_JN) during the database generation step. The journal table contains all of the columns of the main table, plus the following audit columns:

JN_OPERATION	Type of transaction performed: INSERT, UPDATE or DELETE.
JN_ORACLEUSER	Name of the Oracle user who performed the transaction. Since the system uses the application user accounts, this field will be the same for all updates initiated via the pension front-end. Nevertheless, per the database design, each table will have four standard audit columns, capturing the application user ID that created and last modified the record. As a result, the audit trail will provide information for both types of user accounts.
JN_DATETIME	Date and time the transaction was performed.
JN_APPLN	Name of the application in which the transaction was performed.
JN_NOTES	Notes associated with the transaction.
JN_SESSION	Identifying number of the auditing session for that user.

Oracle Designer will also generate all necessary stored procedure and database triggers to synchronize the journal table with the master.

In terms of identifying the user who issued the insert, update, or delete operation, the JN_ORACLEUSER column stores the database account, whereas the 4 audit columns in the table will store the application user information.

4.2 Application Maintained Audit Trail

Some audit trail requirement will be taken care of by the business rules implemented in the system. In other words, the history tracking capability of certain pension application tables already provide audit trail capability. For example, for employer contribution adjustment, the modification history should be reflected in the contribution allocation table. Details of such implementation have been discussed in the data model document.

4.3 Viewing Audit Trail Information

Audit data is available in the database tables (4 audit columns), the history tracking tables, and the journal tables. This is considered an internal database administration function. Access is enabled through tools like SQL/Plus or TOAD.

5 Reporting and Batch Submission

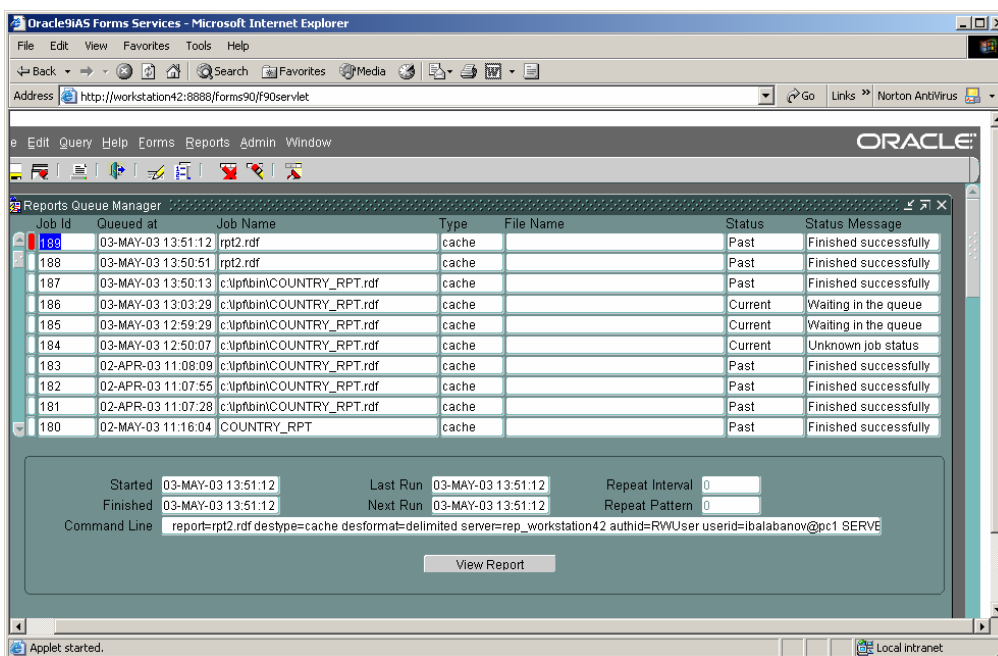
5.1 Concurrent Processing – Reports

In the new pension system, reports will be run in asynchronous mode. This means that a user does not need to wait on-line for the execution of the report, and can thus perform other tasks on the client PC. Once submitted, the report will be run in the background, on the application server.

In summary report execution consists of the following steps for a user:

1. Log into the pension system.
2. Invoke the report submission form.
3. Display the report from a list of available reports. Note that application security is implemented to ensure that only the valid reports are displayed for the currently connected user.
4. Select the report for execution.
5. Enter the appropriate report parameters e.g. member ID, printer ID etc.
6. Submit the request for immediate execution or schedule it for execution at a future date and time.
7. Monitor report execution as required.
8. View report completion status.
9. View output report as required.
10. Print report from Adobe Acrobat Reader. At this point a user can select the appropriate printer for a particular document. Further a user can print multiple copies as required.

A separate form will be developed to review the report submission, display reports that are queued up for execution, and retrieve report output. An example of such a screen is shown below:



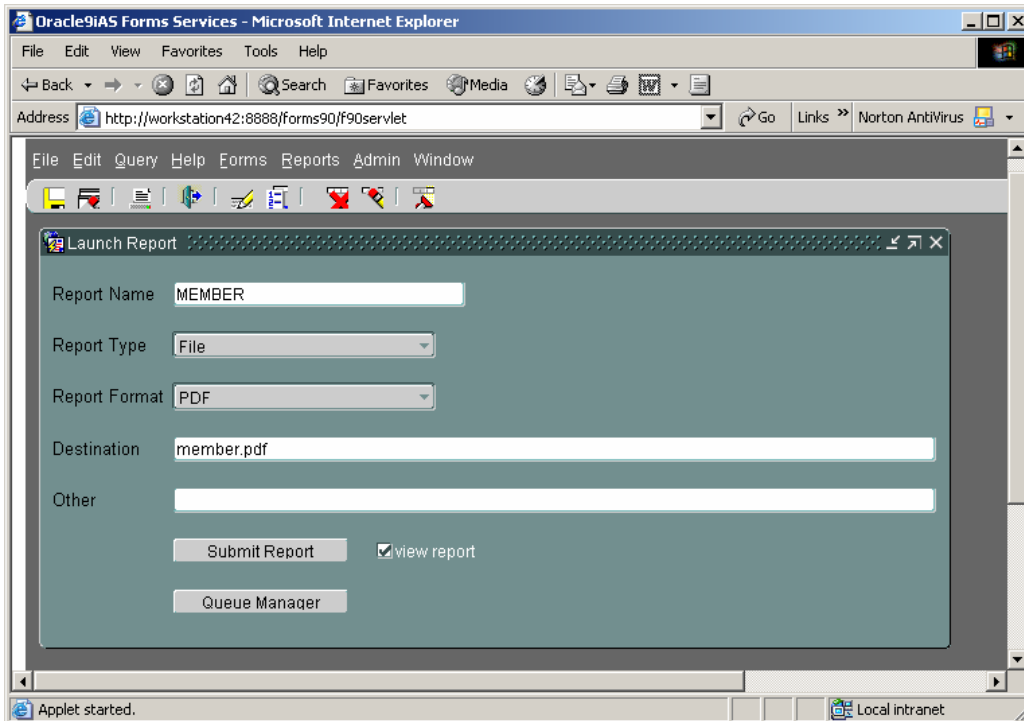
Reports can be set for immediate or delayed execution with different schedule parameters. Users can set the day, time, and frequency a report should be run.

Users can delete a report request that is still pending.

Here are some examples of the scheduling parameters:

SCHEDULE="every first fri of month from 15:53 Apr 23, 2003 retry 3 after 1 hour"
SCHEDULE="last weekday before 15 from 15:53 Apr 23, 2003 retry after 1 hour"

A generic form will be developed to support report submission. That form will provide a list of all available reports, and parameters for report type, format, destination name and scheduling information. Below is an example of such a form:



5.2 Reports Security

Under the Oracle 9iAS environment, reports are processed by the report services. User submits a report to the report server queue. Report server will process the request and put the output into its cache. Then, the user will be able to view the output through the browser using the URL such as the one shown below:

<http://www.lpfserver.com/reports90/repserv?server=server1&jobid=123>

Notice that the "jobid" in the URL uniquely identifies an output file. There may be a security problem here because the output files of all requests are placed in the same cache, with each file uniquely identified by the job ID. It is possible that a user can alter the URL to something like the following:

<http://www.lpfserver.com/reports90/repserv?server=server1&jobid=111>

In essence, the user can retrieve other output files and potentially view sensitive reports requested by other users. Special arrangement is needed to overcome this potential security problem.

To address this issue, the report server will create the output report files with long, encrypted file names; hence, it will be close to impossible for someone to guess the correct file name and view reports generated for others. In this scenario only the pension application's reporting infrastructure will know how to access these files.

5.3 Concurrent Processing – Jobs

Similar to report submission, the new pension system will provide a job submission mechanism for batch jobs. The difference between “reports” and “jobs” is that a report does not update the data in the database while a job does, and report is run by Oracle Reports services on the application server tier, while a job is run directly on the database server. An example of a “job” is a process that finds employees with 23 months with no contributions and creates termination records.

We will use the built-in feature of the Oracle database, the DBMS_JOB package, to implement the job queue and job scheduling capability. Further, we will augment DBMS_JOB with a job submission form, and the job execution management screen.

Once again, the new pension system is not intended to provide best-of-breed, generic batch job submission capability.

5.4 Jobs / Reports Submission Control

Some reports or background jobs can be very resource intensive and the system should prevent the users from submitting such requests during the normal office hours, or otherwise the online performance will be degraded.

Since the users submit the request through a custom built report / job submission form, we can program this form to restrict the submission time of certain reports or jobs. To define the allowable submission time period, we can put in additional columns in the APPLICATION_RESOURCE table to store such information. By referring to such information, the submission form can now either submit or reject a request from the user.

6 Backup and Recovery

6.1 Approach

Oracle has powerful and reliable features that, when implemented to the fullest possible extent, can facilitate 24-hour availability of the database with the possibility to restore its status to any given point of time.

It is understandable that the most sophisticated levels of backup functionality come at the price of higher volume requirements, lower database performance, and increased maintenance costs. Therefore, to make informed choice and avoid unnecessary implications, any backup considerations have to be based on the specific requirements, such as:

- Availability – Is it acceptable to shut down the system for backups, and what is the time and acceptable duration of such downtime?
- Acceptable level of the data loss. – Does the backup strategy allow for the loss of data e.g. 1 day's worth of data, or should the data be never lost (zero-tolerance)? Loss of the current day's data is acceptable.
- What is the acceptable system recovery time in case of failure?

Upon careful investigation of those requirements, we will develop the following strategies:

- Backup strategy – describing approach to backing up the database, application, and client tiers of the system.
- Disaster recovery and fallback strategy – approach to recovering the system from software or hardware failures. This section should list all possible scenarios and provide response to each of them. However cumbersome, this planning process will make sure that system can be restored during predictable time in case of failure.

6.2 Backup Strategy

The backup and recovery strategy is divided into two main areas: database backup and application backup, which correspond to backing-up the database and application tiers of the system (Sun Solaris and Windows 200 boxes) respectively. We do not see the need to address backing up of the client tier, as it will only hold the Oracle Forms client that can easily be downloaded from application server.

There are several database backup and recovery options available for this section as outlined below:

1. **Hot database backup using scripts (using the standard ARCHIVELOG mode and without RMAN).** Additional maintenance of the archived redo log files needs to be performed and a "manual" database restore (using native commands of Oracle and operating system,) would be required in the event of media failure. This method would take a full backup of database and archived redo log files while keeping the database online.

2. **Online database backup using Oracle Recovery Manager (RMAN).** RMAN enables complete online backup and restore of the database and archived redo log files, thus facilitating a 24X7 operational environment. The backups and restores can be run in parallel mode, thus enabling online backups to be taken with multiple channels defined. This method would improve the performance of the backups on multi-processor and multi-channelled backup systems and reduce the time taken for the backup process. The RMAN utility would be configured using a recovery catalogue. An additional database on the same or remote server would be required to support operation of RMAN. Since the RMAN catalogue holds all the vital backup information, the RMAN user export and cold database backup of the RMAN database would be also taken daily. In terms of recovery, the database can be restored to any point in time and up to the point of failure (depending on the type of disaster). This option however may require additional Oracle licences.
3. **“Cold” database backups.** This would require an offline backup (the database would have to be shut down). Usually, it is done every night. However, any failures during the day may result in the loss of up to one day’s worth of data since the recovery process would bring back the previous night’s backup.

The application backup options are outlined below:

1. **Nightly backup of the entire hard drive** for the Application Server (Windows 2000) and storing them on tape.
2. **Nightly backup of the *entire application*** (i.e. Oracle Forms executables, Oracle Reports executables) for the Application Server (Windows2000) and storing it on tape.
3. **Create a duplicate copy of the production application** (i.e. Oracle Forms executables, Oracle Reports executables) for the Application Server (Windows2000) on a *different* Windows2000 machine for backup purposes.
4. **Create a redundant Application Server that serves as a standby.** This option will have serious impact on the architecture of production system, as additional hardware will be required.

Also, all application codes are stored in the Central Repository in Oracle Designer, which enables re-generation of application codes.

7 Appendix

7.1 References

All of the documents referred here are accessible for free via "Oracle Technology Network" web site <http://otn.oracle.com> (user registration is required)

1. Oracle9i Application Server Concepts
http://download-east.oracle.com/docs/cd/A97329_03/core.902/a95926
2. Forms Services Deployment Guide
http://download-east.oracle.com/docs/cd/A97329_03/web.902/a92175
3. Reports Services Publishing Reports to the Web
http://download-east.oracle.com/docs/cd/A91773_01/ids902dl/bi.902
4. Backup and Recovery Concepts
http://otn.oracle.com/pls/db92/db92.to_pdf?partno=a96519&remark=docindex
5. Application Developer's Guide
http://download-east.oracle.com/docs/cd/A97329_03/core.902/a95101