NASA AMES RESEARCH CENTER

Web-Based
Hazard Reporting
System

Oracle Maintenance
for
System Administrators

Table of Contents

Preface 1
Conventions 1
Oracle RDBMS Management 2
  Introduction 2
  Startup 2
  Shutdown 2
  File Maintenance 2
SQL Language 3
  Datatypes 3
  Operators 3
  Functions 4
  Basic Commands 4
    SELECT 4
    INSERT 6
    UPDATE 6
    COMMIT 6
SQL*Plus user interface 7
  Introduction 7
  Startup 7
  Shutdown 7
  SQL Command Syntax 7
  Manipulating Commands 9
  Code Compilation 10
Oracle7 Server Utilities 11
  Export 11
  Import 12
  SQL*Loader 13
Oracle WebServer2.0 Administration 14
  Startup 14
  Shutdown 14
  Web-Based Administration 14
    Listener Configuration 15

Preface

WebServer uses procedures and functions stored in the database to generate the dynamic home pages. These stored procedures are written in a language called PL/SQL. PL/SQL is Oracle's procedural language based on the standard SQL database commands. These procedures can be initially written in any ASCII text editor. The procedures are then compiled and saved in the database through Oracle's command line interface called SQL*Plus.

Conventions

Convention	Explanation
Query	Button
Hazard Report Query	Menu option / Hypertext link
`Start`	User enter text or commands
`User_name`	Text to be replaced by user value

Oracle RDBMS Management

Introduction

The Oracle RDBMS is the heart of the dynamic web page generator. All procedures and functions are stored in the database.

Startup

To start oracle from a command line, type at the prompt:

```
SVRMGRL
```

This will invoke the command line server manger application. Now, you must connect to the database by typing the following at the prompt:

```
CONNECT INTERNAL
```

To start the database, type the following at the prompt:

```
STARTUP
```

Once the database has successfully started, exit the server manager by typing

```
EXIT
```

Shutdown

To stop oracle from a command line, type at the prompt:

```
SVRMGRL
```

This will invoke the command line server manger application. Now, you must connect to the database by typing the following at the prompt:

```
CONNECT INTERNAL
```

To start the database, type the following at the prompt:

```
SHUTDOWN IMMEDIATE
```

The “immediate” option will force the database to shutdown without waiting for everyone to log off. Once the database has successfully shutdown, exit the server manager by typing

```
EXIT
```

File Maintenance

There are several critical files that are required for the proper operation of the Oracle RDBMS.

SQL Language

Refer to the *SQL Language Reference Manual* for greater detail of the following.

Datatypes

Code	Internal Datatype	Description
1	VARCHAR2(*size*)	Variable character string of length *size*. Maximum *size* is 2000.
2	NUMBER(*p*, *s*)	Number having precision *p* and scale *s*.
8	LONG	Character data up to 2 Gb.
13	DATE	Valid dates between 4712BC and 4712AD.
23	RAW(*size*)	Raw binary data of *size* bytes. Maximum *size* is 255
24	LONG RAW	Raw binary data up to 2 Gb.
69	ROWID	Hexadecimal string representing a unique row address.
96	CHAR(*size*)	Fixed character data of length *size*. Maximum *size* of 255
106	MLSLABEL	Binary format of an operating system label.

Operators

Operator	Purpose	Example
Arithmetic		
`+ -`	Denotes a positive or negative expression	`SELECT * FROM ORDERS WHERE QTY = -1`
`* / + -`	Binary operators for basic math expressions	`SELECT SAL+(SAL*COMM) FROM EMP WHERE SYS_DATE – HIREDATE > 365`
Character		
`||`	Concatenates character strings	`SELECT 'Name is '||ename FROM EMP`
Comparison		
`=`	Equality test	`SELECT * FROM ORDERS WHERE QTY = 1`
`!=, <>`	Inequality test.	`SELECT * FROM ORDERS WHERE QTY != 1`
`>` `<`	"Greater than" "Less than"	`SELECT * FROM EMP WHERE SAL > 2500` `SELECT * FROM EMP WHERE SAL < 2500`
`>=` `<=`	"Greater than or equal to" "Less than or equal to"	`SELECT * FROM EMP WHERE SAL >= 2500` `SELECT * FROM EMP WHERE SAL <= 2500`
`IN`	"Equal to any member of" test.	`SELECT * FROM EMP WHERE ENAME IN ('CLERK','SALES')`
`NOT IN`	Equivalent to "!= ALL"	`SELECT * FROM EMP WHERE ENAME NOT IN ('CLERK','SALES')`
`[NOT] BETWEEN x AND y`	[Not} greater than or equal to *x* and less than or equal to *y*.	`SELECT * FROM EMP WHERE SAL BETWEEN 2000 AND 3000`
`EXISTS`	TRUE if a subquery returns at least one row.	`SELECT DNAME FROM EMP WHERE EXISTS (SELECT * FROM EMP WHERE DEPTNO=10)`
`X [NOT] LIKE y`	TRUE is *x* does [not] match the pattern of *y*.	`SELECT ENAME FROM EMP WHERE ENAME LIKE 'A%'`
`IS [NOT] NULL`	Tests for nulls.	`SELECT * FROM EMP WHERE COMM IS NULL`

Operator	Purpose	Example
Logical		
NOT	Returns TRUE if the following condition is false.	SELECT * FROM EMP WHERE NOT (JOB IS NULL)
AND	Returns TRUE if both components are TRUE.	SELECT * FROM EMP WHERE DEPTNO=10 AND JOB='CLERK'
OR	Returns TRUE if either component is TRUE.	SELECT * FROM EMP WHERE DEPTNO=10 OR JOB='CLERK'
Set		
UNION	All distinct rows selected by either query.	SELECT PART FROM ORDERS1 UNION SELECT PART FROM ORDERS2
UNION ALL	All rows selected by either query, including duplicates.	SELECT PART FROM ORDERS1 UNION ALL SELECT PART FROM ORDERS2
INTERSECT	All distinct rows selected by both queries.	SELECT PART FROM ORDERS1 INTERSECT SELECT PART FROM ORDERS2
MINUS	All distinct selected by the first query but not the second.	SELECT PART FROM ORDERS1 MINUS SELECT PART FROM ORDERS2
Other		
(+)	Indicates that the preceding column is the outer join column in a join.	SELECT ENAME FROM EMP, DEPT WHERE DEPT.DEPTNO=EMP.DEPTNO (+)
PRIOR	Evaluates the following expression for the parent row of the current row in a hierarchical query.	SELECT ENAME FROM EMP CONNECT BY PRIOR EMPNO=MGR

Functions

There are many common functions that can be used in SQL statements. Refer to the *SQL Language Reference Manual* for more details.

Basic Commands

There are several basic commands for getting information in and out of a SQL-compliant database: SELECT, INSERT, UPDATE, DELETE, and COMMIT.

<u>SELECT</u>
The SELECT command is used to retrieve data from one or more tables, views, or snapshots. You retrieve rows from base tables that match the search criteria, and which you have privilege to access.

Basic syntax:

```
SELECT column|table.column|expresion|*
FROM table|view|alias
WHERE condition
GROUP BY column
ORDER BY column
```

The SELECT clause determines the format of the output. The FROM clause determines the source of the output. The WHERE clause determines the type of output. The GROUP BY clause determines the organization of the output. The ORDER BY clause determines the sort order of the output.

Hierarchical Queries
Oracle RDBMS has the ability to generate queries with hierarchical or tree-like structure. This is done with the CONNECT BY clause. The CONNECT BY clause establishes the relationship between a record and its parent. The START WITH clause establishes the root row(s). The following command returns all employees in a hierarchical order, starting with the 'PRESIDENT'.

```
SELECT ename, empno, mgr, job FROM emp
  START WITH job = 'PRESIDENT'
  CONNECT BY PRIOR empno = mgr
```

The output generated:

```
ENAME         EMPNO    MGR JOB
------------ ------ ------ ---------
KING           7839        PRESIDENT
JONES          7566   7839 MANAGER
SCOTT          7788   7566 ANALYST
ADAMS          7876   7788 CLERK
```

Joins
A join is a SELECT statement that combines rows from two or more tables, views, or snapshots. The following example returns a list of employees with their corresponding department name. Then employee name comes from the EMP table and the department name comes from the DEPT table.

```
SELECT ename, dname FROM emp, dept WHERE dept.deptno = emp.deptno
```

The output generated:

```
ENAME      DNAME
---------- ----------
JONES      ACCOUNTING
KING       OPERATIONS
SCOTT      RESEARCH
```

Outer Joins
The outer join extends the result of a simple join. The outer join returns all the rows returned by the simple join as well as those rows from one table that do not match any row from the other table. The following example is an outer join of the previous example.

```
SELECT ename, dname FROM emp, dept
  WHERE dept.deptno = emp.deptno(+)
```

The output generated:

```
ENAME      DNAME
---------- ----------
ADAMS
JONES      ACCOUNTING
KING       OPERATIONS
SCOTT      RESEARCH
```

Subqueries
A subquery is a form of the SELECT command that appears inside another SQL statement. A subquery is often used to answer a multiple part question. The following example generates a list of employees would work in the same department as Taylor.

```
SELECT ENAME, DEPTNO FROM EMP WHERE DEPTNO =
  (SELECT DEPTNO FROM EMP WHERE ENAME='TAYLOR')
```

INSERT

The INSERT command allows the user to add rows to a table.

Syntax:

INSERT INTO table|view|alias [(column,…)]
VALUES (value|expression, …) | subquery

The following example inserts a row into the DEPT table:

```
INSERT INTO dept VALUES(50, 'PRODUCTION', 'DETROIT');
```

The following example copies all managers and presidents whose commission exceeds 25% into a BONUS table:

```
INSERT INTO bonus
  SELECT ename, job, sal, comm FROM emp
  WHERE comm > .25 * sal and job IN ('MANAGER', 'PRESIDENT');
```

DELETE

The purpose of the DELETE command is to remove rows from a table.

Syntax:

DELETE FROM table|view|alias
WHERE condition

To delete all rows from the table EMP, type:

```
DELETE FROM EMP
```

To delete the employees who are in department 10, type:

```
DELETE FROM EMP WHERE DEPTNO = 10
```

UPDATE

The UPDATE command allows the user to change existing values in a table.

Syntax:

UPDATE table|view|alias
SET column|table.column = expression|value|subquery
WHERE condition

The following example gives all trainees a null commission:

```
UPDATE emp SET comm = NULL WHERE job = 'TRAINEE'
```

The following example promotes JONES to manager of department 20:

```
UPDATE emp SET job = 'MANAGER', deptno=20
  WHERE ENAME = 'JONES'
```

COMMIT

The COMMIT command ends the current transaction and makes all changes permanent. The command also releases all savepoints and releases the locks.

Syntax:

COMMIT

SQL*Plus user interface

Introduction

You can use SQL*Plus to access databases through the SQL database language.

Startup

Double click on the icon, or from the command line prompt, type:

```
SQLPLUS
```

This will launch the application and ask the user for a username, password, and system. The system name is critical for accessing databases on a remote computer. When everything has started properly, the following prompt will appear:

```
SQL>
```

Shutdown

To exit SQL*Plus, enter the EXIT command at the prompt:

```
SQL> EXIT
```

SQL Command Syntax

Just as spoken language has syntax rules that govern the way we assemble words into sentences, SQL*Plus has syntax rules that govern how you assemble word into commands. You must follow these rules if you want SQL*Plus to accept and execute your commands.

Dividing a SQL command into separate lines. You can divide your SQL command into separate lines at any point, as long as words are not broken. The example below shows 2 ways to enter a command:

```
SQL> SELECT EMPNO, ENAME, JOB FROM EMP WHERE SAL < 2500;
```

Or,

```
SQL> SELECT
SQL> EMPNO, ENAME, JOB
SQL> FROM EMP
SQL> WHERE SAL < 2500;
```

Ending a SQL command. You can end a SQL command in one of three ways:

- With a semicolon (;)
- With a slash (/) on a line by itself
- With a blank line

The semicolon and slash tells SQL*Plus that you want to run the command. SQL*Plus will process the command and store it in the buffer. The blank line tells SQL*Plus that you have finished the command, but do not wish to run it.

Creating Store procedures. Stored procedures are PL/SQL functions, packages, or procedures generated through the SQL CREATE commands. End the subprogram with RUN or a slash (/). To display any error messages, use the SHOW ERRORS command. A stored procedure has three major parts. The first part is the declaration of the procedure. The second part is the declaration of variables. The third part is the body of the procedure.

The following is a simple example of stored procedure code that demonstrates the three basic parts.

```
CREATE OR REPLACE PROCEDURE hello_world IS
 my_date VARCHAR2(15);
BEGIN
  SELECT TO_CHAR(sysdate,'MONTH DD, YYYY') INTO my_date
    FROM DUAL;
END hello_world;
/
SHOW ERRORS
```

The procedure declaration establishes the type (function or procedure), the procedure name, and the parameters. In the example, the name of the procedure is HELLO_WORLD. There are no parameters.

```
CREATE OR REPLACE PROCEDURE hello_world IS
```

The variable declaration for the example only contains one variable. The variable MY_DATE has been defined as a variable character string of 15 characters.

```
my_date VARCHAR2(15);
```

The body of the procedure is a simple select statement that assigns the current date from the dummy table to the variable, MY_DATE.

```
BEGIN
  SELECT TO_CHAR(sysdate,'MONTH DD, YYYY') INTO my_date
    FROM DUAL;
END hello_world;
```

For more information, see *PL/SQL User's Guide and Reference*, *SQL Language Reference Manual*, and *SQL*Plus User's Guide and Reference*.

Manipulating Commands

Editing Commands
SQL*Plus stores SQL commands in a buffer. The following commands allow the user to edit the SQL command.

Command	Abbreviation	Purpose
`APPEND`	`A` *`text`*	add *text* at the end of a line
`CHANGE` *`/old/new`*	`C` *`/old/new`*	change *old* to *new* in a line
`CHANGE` *`/text`*	`C` *`/text`*	delete *text* from a line
`CLEAR BUFFER`	`CL BUFF`	delete all lines
`DEL`	`(none)`	delete a line
`INPUT`	`I`	add one or more lines
`INPUT` *`text`*	`I` *`text`*	add a line consisting of *text*
`LIST` *`n`*	`L` *`n`*	list one line
`LIST *`	`L *`	list the current line
`LIST LAST`	`L LAST`	list the last line
`LIST` *`m n`*	`L` *`m n`*	list a range of lines (*m* to *n*)

Saving Commands
To save the current SQL command, enter the following command and filename:

```
SQL> SAVE file_name
```

SQL*Plus adds the extension SQL to the file to identify it as a SQL query file.

Retrieving Commands
To retrieve a saved command into the buffer, use the GET command:

```
SQL> GET file_name
```

If the extension is something other than SQL, it must be included on the command line.

You can also use the "at" sign (@) to run a command file:

```
SQL> @file_name
```

The @ command lists and runs the commands in the file.

Setting Up Your Environment
There are commands that allow you to change the environment of the current session.

`SET PAGESIZE n`	Sets the number of lines per page.
`SET LINE n`	Sets the length of a line to *n* characters.
`SET PAUSE on|off`	Sets the pause function on or off. ON forces the output to hold each page until a key is pressed.

Using Substitution Variables
Substitution variables allow the user to place/replace values in a command at the time of execution. The variable is preceded by one or two ampersands (&). When SQL*Plus encounters a substitution variable, the user is prompted to enter a value to be used in place of it. For example:

```
SQL> SELECT EMPNO, ENAME, JOB FROM EMP WHERE SAL < &sal;
Enter value for sal: 2500
old 1: SQL> SELECT EMPNO, ENAME, JOB FROM EMP WHERE SAL < &sal
new 1: SQL> SELECT EMPNO, ENAME, JOB FROM EMP WHERE SAL < 2500
```

SQL*Plus displays the following output:

```
EMPNO ENAME      JOB
----- ---------- ----------
12345 JOHNSON    MANAGER
45234 DAVIS      SALES
93201 CARTER     CLERK
```

IMPORTANT: If the & is part of the command, and not a substitution variable, enter the & and the string it is asking you to replace. For example:

```
SQL> SELECT EMPNO, ENAME, JOB FROM EMP WHERE SAL < &sal;
Enter value for sal: &sal
old 1: SQL> SELECT EMPNO, ENAME, JOB FROM EMP WHERE SAL < &sal
new 1: SQL> SELECT EMPNO, ENAME, JOB FROM EMP WHERE SAL < &sal
```

Code Compilation

Compiling procedural code is like running a saved query command. Use the @ command to run the code through the system. At the end of the procedure, there should be two commands, each on separate lines. The first command is a slash (/). This will start the compilation. The second command is SHOW ERRORS. If the code has errors, this command will display them.

The following is a simple example of stored procedure code stored in the file hello.sql in the TEST directory:

```
CREATE OR REPLACE PROCEDURE hello_world IS
 my_date VARCHAR2(15);
BEGIN
  SELECT TO_CHAR(sysdate,'MONTH DD, YYYY') INTO my_date
    FROM DUAL;
END hello_world;
/
SHOW ERRORS
```

The following is the result of the compilation:

```
SQL> @/test/hello

Procedure created.

No errors.
SQL>
```

For more information, see *PL/SQL User's Guide and Reference*, *SQL Language Reference Manual*, and *SQL*Plus User's Guide and Reference*.

Oracle7 Server Utilities

For more detailed information about these utilities, see *Oracle7 Server Utility User's Guide*.

Export and Import are used primarily for the following tasks:

- Data archival
- Upgrading to new releases of Oracle
- Backing up Oracle databases
- Moving data between Oracle databases

Export

Export's basic function is to extract the objects of a database and put them in a flat file. Export writes data from an Oracle database to operating system files called export files. The export files are named during the export process. It is a good idea to give each export file a name that indicates its contents and the date it was written. Export files can only be read by Import, and no other Oracle7 utilities.

Export Modes

Table	Exports specified tables that the user has privileges to access.
User	Exports all objects in a user's schema.
Full Database.	Exports all objects in a database, including indexes and stored procedures.

Using Export

There are three ways to invoke export:

- Enter the command EXP *username*/*password* to begin an interactive session and let Export prompt you for the information needed. Functionality is limited in the mode.
- Enter the command EXP *username*/*password* followed by various parameters that control how Export runs. The number of parameters may not exceed the maximum line length.
- Enter the command EXP *username*/*password* PARFILE=*filename* with the remaining export parameters contained in the specified file.

The following is an example of the interactive export of a full database export:

```
Exp system/manager
…
Enter array fetch buffer size: 4096 > [Return]
Export file: EXPADT.DMP> dba.dmp
E(ntire database), U(sers), T(ables): U> e
Export grants (Y/N): Y> y
Export table data (Y/N): Y> y
Compress extents (Y/N): Y> y
```

Following the interactive exchange, export messages are displaying showing the status of the export.

The following is an example of the interactive export of a user:

```
exp scott/tiger
…
Enter array fetch buffer size: 4096 > [Return]
Export file: EXPADT.DMP> scott.dmp
E(ntire database), U(sers), T(ables): U> u
Export grants (Y/N): Y> y
Export table data (Y/N): Y> y
```

```
Compress extents (Y/N): Y> y
```

Following the interactive exchange, export messages are displaying showing the status of the export of scott's schema.

The following is an example of the interactive export of a user:

```
exp system/manager
…
Enter array fetch buffer size: 4096 > [Return]
Export file: EXPADT.DMP> mytabs.dmp
E(ntire database), U(sers), T(ables): U> t
Export grants (Y/N): Y> y
Export table data (Y/N): Y> y
Compress extents (Y/N): Y> y
About to export specified tables…
Table name(press RETURN when done): bar
. exporting table             BAR  2355 rows exported
…
```

Export messages are displaying showing the status of the table export immediately following the interactive exchange for each table.

Import

Import extracts objects from an export file and puts them into a database. A table is imported into its original tablespace using the original storage parameters. Table objects are imported from the export file in the following order:

1. Table definitions
2. Table data
3. Table indexes
4. Integrity constraints and triggers

When user definitions are imported into Oracle7, they are created with the CREATE USER, and may lose privileges from previous versions.

When a stored procedure, function, or package is imported, it retains its original timestamp. If the timestamp of the original is the same as the current version in the database, it is not marked for recompilation. If the timestamps don't match, the procedure will be recompiled.

Import Modes

Table	Imports specified tables that the user has privileges to access.
User	Imports all objects in a user's schema.
Full Database.	Imports all objects in a database, including indexes and stored procedures.

Using Import

There are three ways to invoke Import:

- Enter the command IMP *username*/*password* to begin an interactive session and let Import prompt you for the information needed. Functionality is limited in the mode.
- Enter the command IMP *username*/*password* followed by various parameters that control how import runs. The number of parameters may not exceed the maximum line length.
- Enter the command IMP *username*/*password* PARFILE=*filename* with the remaining import parameters contained in the specified file.

*SQL*Loader*

SQL*Loader is a tool for loading data from external text files into Oracle database tables. It loads a variety of formats, performs filtering, and loads multiple tables simultaneously.

The Control File

You must provide a file called the control file as an input to SQL*Loader. The control file tells how to interpret the datafile. SQL*Loader uses this information to convert the data into the corresponding datatypes specified by the table definition.

The Data

The data can be loaded in various formats. It is usually read from one or more datafiles. However, the data may also be placed in the control file information. The data can be in a fixed or variable format. With a fixed format, the data is contained in records that all have the same format. The data fields all have the same length, type, and position in each record.

In variable format, each record is only as long as necessary to contain the data. The data is divided into fields. The fields can have changing lengths and positions, separated by a delimiter. A delimiter can be a character, like a comma, or quotation marks.

The following is an example is an example of variable length data.
The control file:

```
LOAD DATA
INFILE ‘test.dat’
BADFILE ‘test.bad’
INTO TABLE dept
FIELDS TERMINATED BY ‘,’ OPTIONALLY ENCLOSED BY ‘”’
(deptno, dname, loc)
```

The datafile, TEST.DAT:

```
12, RESEARCH, “SARATOGA”
10,”ACCOUNTING”, CLEVELAND
11, ART, SALEM
```

To run this example, type the following command:

```
Sqlldr userid=scott/tiger control=test.ctl log=test.log
```

SQL*Loader will create a log file with the name TEST.LOG. Any rejected records will be copied into TEST.BAD.

For more detailed information about these utilities, see *Oracle7 Server Utility User’s Guide*.

Oracle WebServer2.1 Administration

The Oracle WebServer is a group of Internet engines and services that translates HTML and Java functions and requests into database commands. The WebServer can be configured to have many "ears" to receive these web commands. These ears are called listeners. Each listener receives commands on one or more ports. The port number is often indicated in a URL by a colon and a number immediately following the base address. For example: `dqa1.gsfc.nasa.gov:10000` indicates that the port to be used on this server is 10000.

The Listeners translate the URL into the appropriate directory alias and the security established for them. The Listener then passes on the information to the Web Request Broker (WRB). The WRB then determines which application to use in the interpretation of the request. If the request is a dynamic request to the database, the information is passed on to the Database Connector Descriptor (DCD). The DCD then translates the request into a database query. The response is then based back through the change to the requesting user.

These listeners can be remotely manipulated through an administration module over the web. The administration pages allow a user to: start and stop the database; start, stop, and configure the listeners; define listener types (WRBs); create and modify DCDs. Before any of this can be done, the administration listener must be started.

Startup

To use the Administration module through the web, the administration listener must be running. Make sure that the database is running before the administration listener is started.

At the prompt:

```
WLCTL21 START ADMIN
```

The other processes can be start by this means as well. The syntax is as follows:

```
WLCTL21 [START|STOP] <listener>
```

Replace *listener* with the name of the agent to be started.

Shutdown

First, shutdown the other listeners from the command line, type:

```
WLCTL21 STOP <listener>
```

Then shutdown the administration listener from the command line, type:

```
WLCTL21 STOP ADMIN
```

Web-Based Administration

Startup, shutdown, and configuration of the listeners are controlled through a web browser. To access this administration module, enter the following address in a web browser with the appropriate substitutions:

```
<server:port>/ows-adoc/Intro.html
```

A user name and password will be required to access this page. This will open Oracle WebServer Manager Home Page. The Manager menu has four options: Oracle7 Server, Oracle Web Listener, PL/SQL Agent, and Web Request Broker.

The **Oracle7 Server** page allows the user to start and stop the selected databases. To be able to do this, the database must be on the same computer as the Oracle Web Listener. Stopping a database will not disable the web listeners that are running at that time. This means that if another user tries to access the database over the web, an error will be generated.

The **Oracle Web Listener** page allows the user to start, stop, create, delete, and configure the web listeners currently on the server. At the bottom of the page is the list of listeners on the server. The blinking word "Running!" appears next to allow the listeners that are working. To start, stop, or delete a listener, click on the corresponding button next to the listener name. Clicking the **Configure** button opens the configuration pages.

The **PL/SQL Agent** page allows the user to create and modify the Database Connection Descriptor (DCD). These parameters are such things as the user to connect as and other oracle RDBMS information.

The **Web Request Broker** page allows the user to define the application to be used for interpretation for each directory alias. Application security is also handled here.

Listener Configuration

The Listener Configuration pages allows the privileged user to establish listener defaults, create directory mappings and aliases, and setup users and security.

Most of the parameters that can be changed in the first Web Listener Configuration page are defaults. The defaults include log file locations, language, cache directories, MIME types, connections, startup page, and directory mapping. Be sure to read the instructions carefully before making any changes.

NOTE: The directory mapping configured in the Listener must match the scheme in the WRB.

Security for the Listener is handled on a separate page that is accessed from the menu near the top of the Web Listener Configuration page. Oracle Web Listener provides 4 different forms of security:

- Basic Authentication
- Digest Authentication
- IP-based Authentication
- Domain-based Authentication

Basic and Digest security are the simplest: users and passwords are specified; groups contain users; realms contain groups. IP-based security allows groups of IP addresses to be granted or denied access to various directories. Domain-based security acts very similar to IP-based. Groups of domains are established.

The security page allows for the setup of Secure Sockets Layers through the use of certificates.

For any change to take effect, the Listener must be restarted.