R S I

ORACLE

DATA BASE ADMINISTRATOR'S GUIDE

Oracle Programmer's Guide - Version 2.3

O R A C L E

DATABASE ADMINISTRATOR'S GUIDE

TABLE OF CONTENTS

I. INTRODUCTION

Within an ORACLE environment there are a number of functions which should be under control of a centralized database administration. These functions, summarized below, are described in detail in this manual and in the ORACLE Installation Guide.

- ORACLE Installation
  The ORACLE Installation facilities are described in the ORACLE Installation Guide.

- ORACLE Initialization
  The ORACLE Initialization procedures are described in the ORACLE Installation Guide.

- Database Creation & Directory Functions
  Each ORACLE system contains a database directory which describes existing databases and the operating system files utilized for them. The DBF utility provides functions for creating databases, and for adding to, removing from, and listing information in the database directory.

- Database Backup & Recovery Procedures
  Standard procedures should be established for periodically saving the files comprising databases, through use of operating system utilities. The ORACLE Journal provides a means to recover databases following hardware failures. The JNL utility provides functions for controlling the Journal and for applying the Journal in the recovery process.

- Database Loader
  The ORACLE Loader Utility provides facilities for loading raw data from one or more files into a database table.

- Table Unload & Reload
  The Unload Reload Utility provides functions for unloading tables from a database and subsequently reloading those tables into the same database or another database.

D B F

DATABASE FILE UTILITY

I. INTRODUCTION

DBF is the ORACLE utility supporting the functions of establishing a database and mapping that database onto operating system files. ORACLE maintains a directory of databases under its control. The DBF utility is used to establish, modify, and delete entries in this directory. When performing initialization functions on an existing database, any data in the existing database will be deleted. The directory contains information on the databases, their extents, and the physical files used. DBF provides functions to:

- Establish a database as an existing or new file;
- Add an existing or new file as an extent to an existing database;
- Remove an existing database from the database directory;
- Enter the files of an existing database into the database directory;
- Reallocate an existing database extent to another file;
- Identify the system database;
- List database directory information.

When ORACLE is installed, the ORACLE system database is the first database to be created. THE DBF utility is used to create the system database. Subsequently, whenever the operating system is initialized, the DBF utility must be used to identify the system database to ORACLE, before normal ORACLE processing can proceed.

II. OPERATION

When an ORACLE database is opened, ORACLE obtains the information about that database from the database directory. From that point on the information is maintained until the last active user of that database issues a CLOSE. Thus, if DBF is used to modify information about a database when that database is opened, ORACLE will not use the new information until that database is subsequently closed by all active users of that database, and then reopened.

When a database is created or initialized by DBF, that database is established as a secure or nonsecure database, through the user-name parameter. Once a database has been created as a secure database, access to it is restricted to the user who created it and to any users specified by him (see DEFINE USER statement in SQL). For a secure database, all DBF functions (except SYSTEM DATABASE, LIST, and ENTER) require specification of the user-name/password, and may only be performed by the creator of that database.

DBF is invoked with the following command:

```
DBF  dbf-command   [parameter-list]
```

where

dbf-command	is a code identifying the function to be performed;
parameter-list	is an optional list of one or more parameters used for the function.

Parameters must adhere to the following conventions:

1. Database-names consist of any printable characters and can be up to 16 characters long.

2. Database-names must be unique.

3. File-names must follow the conventions of the host operating system for naming files.

4. The database-names "ORACLE" and "ORAWRK" are reserved for the system database and the ORACLE Work database.

5. The user-name consists of any printable characters and can be up to 20 characters long.

6. The password consists of any printable characters and can be up to 20 characters long.

7. Database-names, user-names, and passwords are automatically converted to upper-case characters.

8. For the CREATE and EXTEND functions, the size parameter refers to the size of the operating system file being created. It may be expressed as a number of 512-byte blocks, or as a number followed by the letter "K", representing the number of 1024-block units, as follows:

   500 means 500 blocks (256,000 bytes)
   3K means 3x1024 = 3072 blocks (1,572,864 bytes)

   The minimum size of a database is 1024 blocks. The system database must be a single extent (single file) and its size must be at least 4096 blocks.

III. DBF FUNCTIONS

Create Database

This function creates a file under the host file system, initializes the created file as a database, and enters the new database into the database directory.

```
      C
DBF   CREATE  db-name  file-name  size  [user-name/paswrd]
```

where:

db-name	is the name of the newly created database;
file-name	is the name of the file to be created and initialized as a database;
size	is the size of the file to be created, expressed as a number of 512-byte blocks, or as a number followed by the letter "K" representing a multiple of 1024-block units;
user-name/paswrd	is the optional parameter used to identify the creator of the database, which must be provided if the database is to be secure (protected with the GRANT/REVOKE privileges). If only a user-name is specified, DBF will prompt the user for a password, which when entered will not be displayed on the screen.

Extend Database

This function creates a file under the host file system, initializes the created file as an extent to the specified database, and enters the new extent into the database directory.

```
      E
DBF   EXTEND  db-name  file-name  size [user-name/paswrd]
```

where:

db-name	is the name of an existing database which is to be extended;
file-name	is the name of the file to be created and initialized as an extent;
size	is the size of the file to be created, expressed as a number of 512-byte blocks, or as a number followed by the letter "K" representing a multiple of 1024-block units;
user-name/paswrd	is the optional parameter identifying the user, required for any secure database. If only a user-name is specified, DBF will prompt the user for a password, which when entered will not be displayed on the screen.

Initialize Database

This function initializes an existing file as a database and enters the new database into the database directory.

```
          I
  DBF     INIT    db-name    file-name    [user-name/paswrd]
```

where:

db-name	is the name of the newly initialized database;
file-name	is the name of an existing file to be initialized as a database.
user-name/paswrd	is the optional parameter used to identify the creator of the database, which must be provided if the database is to be secure. If only a user-name is specified, DBF will prompt the user for a password, which when entered will not be displayed on the screen.

Initialize Database Extent

This function initializes an existing file as an extent to the specified database and enters the new extent into the database directory.

```
        IE
  DBF   INITEXTENT   db-name   file-name   [user-name/paswrd]
```

where:

db-name	is the name of an existing database which is to be extended;
file-name	is the name of an existing file to be initialized as an extent.
user-name/paswrd	is the optional parameter identifying the user, required for any secure database. If only a user-name is specified, DBF will prompt the user for a password, which when entered will not be displayed on the screen.

Remove Database

This function removes a database from the database directory. Note that the files belonging to the specified database are **not deleted** or in any way removed from the host file system.

Removing a database from a directory would be required when performing the following functions:

- Moving a database from one ORACLE system to another;
- Renaming a database;
- Taking a database off-line.

```
        R
DBF     REMOVE    db-name    [user-name/paswrd]
```

where:

db-name	is the name of the database to be removed.
user-name/paswrd	is the optional parameter identifying the user, required for any secure database. If only a user-name is specified, DBF will prompt the user for a password, which when entered will not be displayed on the screen.

Enter Database

This function enters a set of existing database extent files (which have previously been initialized) as a single database into the database directory. The Enter Database function is also used to re-enter a previously removed (see REMOVE function) database, into the database directory.

```
        ED
DBF     ENTER     db-name     fn1 ... fnn
```

where:

db-name	is the name of the database to be entered into the database directory;
fn1 .. fnn	is the list of file-names of the files which make-up the database. These files must be entered in extent number order. If the file-name list is not entered, DBF will prompt for each file-name as follows: enter extent file n: Responding to a prompt with a CR or ^Z indicates that all files to be entered have been specified.

Move Extent

This function logically moves an extent of a database from one file to another and updates the database directory to reflect the change. Note that no files are copied or deleted. The file being moved to, must be an exact duplicate of the specified extent file. When extents are added to an existing database, the extents are assigned sequential numbers by ORACLE. The MOVE function requires identification of the extent to be moved, by extent number.

```
      ME
DBF   MOVE  db-name  file-name  ext-num  [user-name/paswrd]
```

where:

db-name	is the name of an existing database to be operated on;
file-name	is the name of an existing file which is an exact duplicate of the specified extent file;
ext-num	is number which specifies the extent to be moved.
user-name/paswrd	is the optional parameter identifying the user, required for any secure database. If only a user-name is specified, DBF will prompt the user for a password, which when entered will not be displayed on the screen.

Reinitialize Database

This function re-initializes an existing database. All data in the database is lost and the database is like a newly initialized database.

CAUTION! This function deletes the data in an existing database and should be used with extreme care.

```
        RI
DBF     REINIT    db-name     [user-name/paswrd]
```

where:

db-name	is the name of the database to be reinitialized.
user-name/paswrd	is the optional parameter used to identify the creator of the database, which must be provided if the database is to be secure (protected with the GRANT/REVOKE privileges). If only a user-name is specified, DBF will prompt the user for a password, which when entered will not be displayed on the screen.

System Database

This function identifies the system database to ORACLE. The specified file must already exist. When the system database is created or initialized, this function is automatically performed. This function is required when bringing up ORACLE (such as in a newly initialized operating system).

```
        SD
DBF     SYSTEMDB    file-name
```

where:

file-name	is the name of the file to be known to ORACLE as the system database.

List Database Information

This function displays information from the database directory. The display contains information about a database and its extents. If a database name is not specified, information will be displayed for all databases in the directory.

For each database the following information is provided:

(1) Database Name;
(2) User Name (Displayed only for secure databases);
(2) Security/Journal Flags
    ('S' for Secure database, 'J' for Journaling);
(3) For Each Extent:
    (a) Extent Number;
    (b) Number of Blocks;
    (c) File Name.
(4) If LU is specified, the number of unused blocks in each database is also displayed.

```
        L
DBF     LIST    [db-name]
        LU
```

where:

db-name	is the optional parameter specifying the name of the database to be listed.

SAMPLE DISPLAY

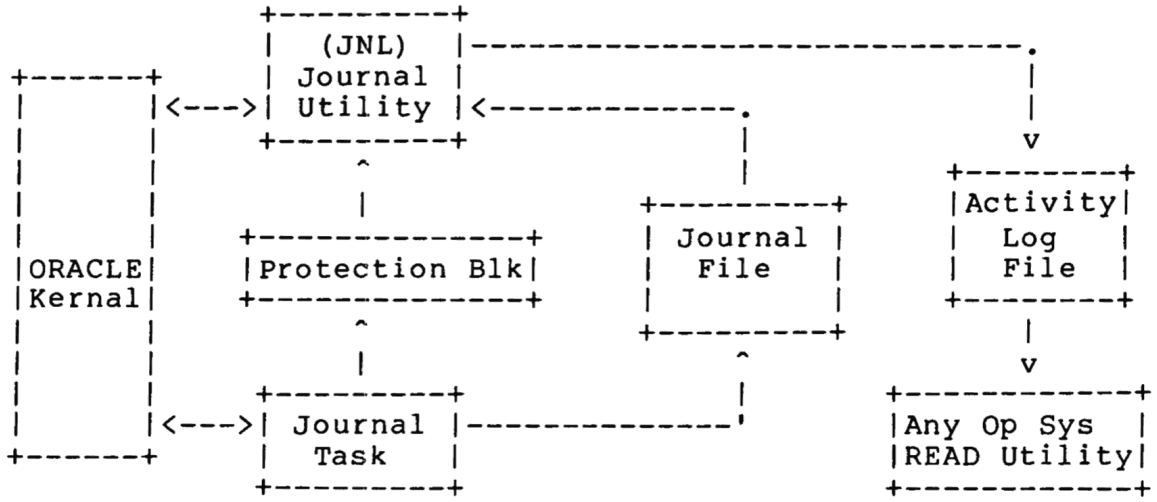
DATABASE NAME USER NAME	FLAGS	EXT #	BLOCKS	FILE
demo	SJ	1	1024	DR0:[1,1]DEMO.DBS;1
			1024	
kcm	S	1	4096	DR0:[1,1]KCM.DBS;1
qa			4096	
oracle		1	4096	DR0:[1,1]ORACLE.DBS;1
ora5555			4096	

I. INTRODUCTION

An ORACLE system includes a Journal Task which controls all writing to the databases. A database may optionally be journaled, in which case the Journal Task will write afterimages of the modified blocks, to a Journal file for use in forward recovery. The Journal Task uses a single Journal file for the one or more databases being journaled. A Journal Utility provides functions for:

- o Initiating and stopping the Journal during an ORACLE session;
- o Initiating and stopping the Journal for a database;
- o Applying the Journal to one or more databases;
- o Querying the status of the Journal activity.

The ORACLE Task also maintains a special single-block Data Protection file on disk. The relationship is as follows:



Application of the Journal is used to restore the status of a database in the following situations:

- Disk Failure

  A disk hardware problem can result in the loss of one or more database extents (files). This can affect one or more databases, requiring those databases to be restored. Note that the LIST function in the DBF utility can be used to determine the extents which comprise each database.

- System Failure

  A system failure can be the result of an Operating System crash, a power failure, or an ORACLE problem. When ORACLE terminates as a result of an ORACLE problem, an ORACLE cleanup task assures that the database is left in a valid state. During an Operating System crash or power failure, the integrity of the database will be compromised only in the event that the incident occurs during the actual writing of blocks to that database. At system start up time (such as following a crash or power failure), the ORACLE initialization routine should include the DISPLAY STATUS function of the JNL utility to indicate which databases, if any, were affected. Note that since the ORACLE initialization procedure initializes the Data Protection Block, it is **imperative** that the DISPLAY STATUS function be performed prior to initialization, in order to ascertain the status of the databases.

The application of the Journal can only be used to restore database integrity. The determination as to whether or not specific SQL Data Manipulation (DML) statements (such as a global updates) were successfully executed, requires SQL queries to determine if the update activity was completed.

Note that for the special case of Data Definition (DDL) statements (CREATE TABLE, EXPAND TABLE, DROP TABLE, DEFINE VIEW, and DROP VIEW), integrity is automatically guaranteed. Thus if a DDL statement was in process, but not yet completed at the time of the failure, ORACLE will automatically back out any completed portion of the DDL statement the first time that database is reopened.

The files involved in the Journal functions are named as follows:

	Protection Blk Block File	Activity Log File	Journal File
VMS	SYS$ORACLE:PARAM.JNL	SYS$ORACLE:LOG.JNL	user-name
RSX-11M,IAS	LB:[1,1]PARM.JNL	LB:[1,1]LOG.JNL	user-name
UNIX	/oracle/param.jnl	/oracle/log.jnl	user-name

II. JOURNAL PROCEDURES

Proper use of the Recovery Functions in an ORACLE system requires the establishment of certain operational procedures. The most important of these involve the periodic saving of the database (or extents of the database). This is accomplished with any of a number of facilities available in a given operating system. Some of the available facilities and their characteristics are summarized in Table 1. These back-up procedures should be performed together with the other operational procedures instituted for a given computer configuration.

Another important procedure involves the saving and naming conventions for the files of the Journal. The Journal consists of a series of disc files. When the Journal is initialized (at ORACLE system start up), the name (and size) is provided for the Journal File. When this file is full, ORACLE will start another file and automatically name the next file by incrementing a number in the name (indicated by a special character). Thus during an ORACLE session, there will probably be multiple files (depending on the file size and the update activity), all having a common name with an increment. If the periodic SAVE spans multiple ORACLE sessions, a naming convention should be used to be able to easily identify the many files making up the Journal since the last periodic SAVE. It might also be appropiate to archive disc Journal files to tape.

The Journal Utility maintains an Activity Log which is useful in monitoring the historical series of Journal activity. This log contains date and time entries for the following:

- Journal Start and Journal File name
- Journal Start for particular database
- Journal Stopped
- Journal Stopped for particular database

This Activity Log file can be read with any convenient system utility for reading files. The following is an sample of information on the log.

```
Tue Mar 24 13:00:44 1981  Journal started on database pers
Tue Mar 24 13:00:48 1981  Journal started on file PER?.JNL
Tue Mar 24 18:06:50 1981  Journal stopped
Tue Mar 24 18:06:56 1981  Journal stopped on database pers
```

OPERATING SYSTEM	UTIL PGM	DISK LEVEL	FILE LEVEL	DISK TO DISK	DISK TO TAPE	SPEED
VMS	BCK	X	X		X	HIGH
	COPY	X	X	X	X	LOW
	DSC	X		X	X	MEDIUM
	FLX		X		X	LOW
RSX-11M,M+	BRU	X	X	X	X	HIGH
	PIP		X	X		LOW
RSX-11M,M+ and IAS	DSC	X		X	X	MEDIUM
	FLX		X		X	LOW
IAS	COPY		X	X	X	LOW
	BCK		X	X	X	HIGH
UNIX Ver6 and UNIX PWB	DUMP/ RESTORE		X		X	MEDIUM
	CP	X	X	X	X	LOW
	DD	X	X	X	X	MEDIUM
	TP	X	X		X	LOW
UNIX PWB only	VC	X		X	X	HIGH
	CPIO		X	X	X	LOW

TABLE 1

OPERATING SYSTEM BACK-UP UTILITIES

III. JNL FUNCTIONS

JNL is invoked with the following type of command:

```
JNL    jnl-command    [parameter-list]
```

where

jnl-command	is a term specifying the function to be performed. The jnl-command may be abbreviated to any unique subset of the statement verb (ie, STAR for START or STAT for STATUS).
parameter-list	is an optional list of one or more parameters used for the function.

Parameters must adhere to the following conventions:

1. File names must follow the conventions of the host operating system for naming files.

2. For the START function, the size parameter refers to the size of the operating system file being created. It may be expressed as a size of 512-byte blocks, or as a number followed by the letter "K", representing the number of 1024-block units, as follows:

   500 means 500 blocks (256,000 bytes)
   3K means 3x1024 blocks = 3072 blocks (1,572,864 bytes)

Start Journal

This function starts the ORACLE Journal. The Start Journal function would normally be performed as part of the ORACLE start up procedure.

```
JNL    START    filename    size
```

where:

filename	is the name of the operating system file to be used as the Journal. The name must contain a "?" in some character position. This special character will be replaced by the 3-digit number "001". successive files of the Journal during this ORACLE session will receive names with incremented numbers.
size	is the threshhold size used in determining when to terminate a physical file in the Journal and start another one. The actual size of each file will be slightly larger than this value. The size is expressed as a number of 512-byte blocks, or as a number followed by the letter "K" representing a multiple of 1024-block units.

The following example will start a Journal on a file called FEB01001. Each file of the Journal will be about 3,072 blocks in size.

```
JNL    START    FEB01?    3K
```

Stop Journal

This function stops the ORACLE Journal. It would normally be performed as part of ending an ORACLE session.

```
JNL    STOP
```

Start Database Journal

This function starts Journal activity for a database. Once Journal activity for a database has been initiated in an ORACLE session, Journal activity for that database will continue in future ORACLE sessions unless specifically turned off with the DBSTOP function.

```
JNL    DBSTART    dbname
```

where:

dbname	is the name of the ORACLE database.

Stop Database Journal

This function stops Journal activity for a database. This function would only be used after it was determined that there was no need to ever have to recover a particular database.

```
JNL    DBSTOP     dbname
```

where:

dbname	is the name of the ORACLE database.

Apply Journal Block Images

This is the function which applies the afterimages from the Journal to a saved copy of the databases in an ORACLE environment. When it is required to recover one or more databases in an ORACLE environment, those databases will be established by copying the saved versions to a disk and then performing the JNL APPLY function. Note that this function will apply the Journal images to all databases for which the Journal was activated, and that during the APPLY function those databases will be locked. If it is not necessary to involve all of those databases, the databases not being recovered **must** temporarily removed (DBF REMOVE function) from the database directory. The temporarily removed database could still be active by temporarily entering it (DBF ENTER function) with a different name.

The JNL APPLY function will automatically process the multiple files of the Journal for an ORACLE session in sequence. However if a Journal spans multiple ORACLE sessions, the JNL APPLY function must be executed for each session and in the proper sequence (note importance for effective file naming conventions). If, while processing the multiple Journal files of an ORACLE session, a file is required which is not available, a message will be displayed indicating the name of the file and processing will terminate. The APPLY function may be continued when the missing file is made available by invoking the JNL CONTINUE function.

```
JNL    APPLY    filename
```

where:

filename	is the name of the Journal file for an ORACLE session. The name must contain a "?" to indicate the position of the 3-digit number, incremented across the multiple files constituting the Journal for that session. On PDP-11 cpu's, the file may exist on disk or tape files. On VAX-11 cpu's, it must be a disk file.

Continue Applying Journal Images

The CONTINUE function is used when an APPLY function stops because a file is not available. This function permits continuation of the APPLY once the missing file is made available.

```
JNL   CONTINUE
```

Display Journal Status

JNL STATUS

This function causes a display of the status of the Journal Activity within the ORACLE environment. The Display function would always be used following an abnormal termination in order to determine the status of any database update activity. The Display function is also used to determine the size (in blocks) of the current Journal file and the total size of all Journal files of the current ORACLE session. The output format of the Display function is as follows:

```
+-----------------------------------------------------------+
|                                                           |
|  DATABASE          DDL BAD JNL                            |
|  ----------------- --- --- ---                            |
|                                                           |
|  ORACLE            NO  NO  NO                             |
|                                                           |
|  PERSONNEL         NO  YES YES                            |
|                                                           |
|  DR1:[1,70]004.JNL is the current Journal file.           |
|  358 blocks have been written to the current file.        |
|  1969 blocks have been written to the Journal.            |
|                                                           |
|                                                           |
+-----------------------------------------------------------+
```

where:

DATABASE	is the name of the database. All open databases will be included in the report
DDL	indicates that a DDL statement was in process during the failure, and that any portion of that DDL statement which was completed, will automatically be backed out the first time that database is reopened. Since DDL statements are single-threaded, if a DDL statement was interrupted by the failure, it is impossible for other activity to have also been interrupted.
BAD	indicates that the failure interrupted a database write and that the Journal must be applied to restore integrity to that database.

JNL	indicates that the Journal is active for that database. If integrity is lost in a database for which the Journal is not active, that database cannot continue to be used. A back-up copy must be restored before it can successfully be returned online.

A "YES" in the DDL column is only to indicate that the DDL statement in process at the time of the failure will automatically be backed out. Since the ORACLE system database is not journaled, a failure on it requires the use of the DBF utility to re-create it and enter the user databases.

IV JNL. Messages

There are 4 types of messages produced by the JNL utility:

1. Successful Completion
2. Task Failure Reason
3. General Error
4. Function-Specific Error

The messages are generally self-explanatory. The <item> in the message examples are replaced by an appropiate data value when the message is displayed. Those messages notated with the term "(ORACLE-error)", will be followed by the appropriate ORACLE error message (see Messages & Codes Manual).

Successful Completion Messages:

Journal processing complete
Journal started on database <database-name>
Journal stopped on database <database-name>
Journal started
Status report complete
Journal stopped

Task Failure Reason Messages:

Journal task canceled because
of bad ORACLE buffer cache
of missing ORACLE buffers
of missing KDA
of a database write failure
of a journal write failure
of a journal read failure

General Error Messages:

Not enough arguments
Ambiguous journal command
Invalid journal command
Cannot log on to ORACLE (ORACLE-error)
Cannot open the system database (ORACLE-error)
Journal utility in use
Cannot create journal log file

Function-Specific Error Messages:

APPLY Function:

Read failure on journal file <file-name>
Sequence number failure on journal file <file-name>
Incomplete ddl operation found on database <database-name>
Invalid journal block on journal file <file-name>
Journal file <file-name> not available
Type 'jnl continue' when file is available
Cannot create apply context file
Database <database-name> does not exist
Cannot open database <database-name>
Sequence number check
Question mark missing from file name
Invalid file name

DBSTART Function:

Invalid database name

DBSTOP Function:

Invalid database name

CONTINUE Function:

Cannot open apply context file
Cannot read apply context file

START Function:

Invalid size parameter
Question mark missing from file name
Invalid file name
Journal is already active

STATUS Function:

Cannot open parameter file <file-name>
Cannot read parameter file <file-name>

STOP Function:

Journaled databases are active
Journal is not active

DATABASE LOADER UTILITY

O D L

1.0 INTRODUCTION

ODL is the ORACLE utility for loading raw data from operating system files into an ORACLE database. ODL may be used to initially load a database table or to add records to an existing table. The ODL program functionally will:

1. read the user's input data;
2. perform the mapping from the raw data to a database table;
3. load the data records into a database table;

These functions are accomplished by using a description language as input to the ODL program. The general processing of the ODL program is as follows:

1. read the description language;
2. create the necessary SQL commands;
3. load the data using SQL INSERT commands.

The database table being loaded must have been created with the SQL "CREATE TABLE" statement prior to use of the ODL utility for that table.

2.0 RAW DATA REQUIREMENTS

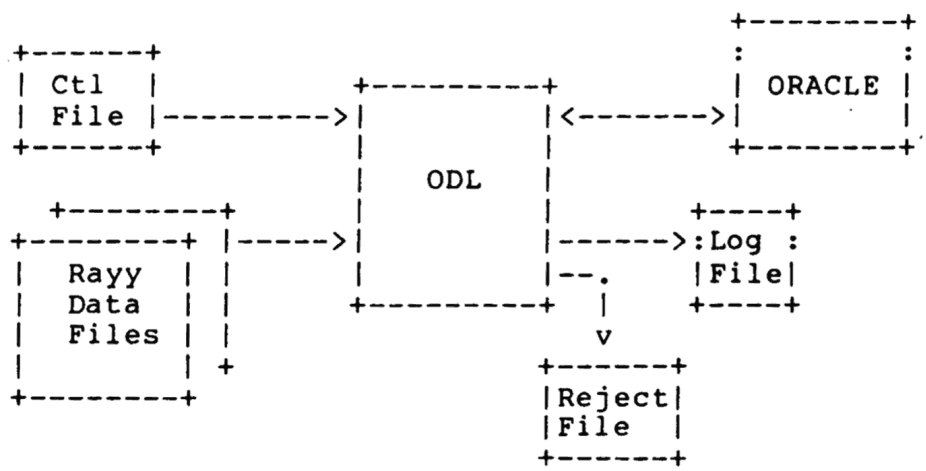
ODL requires the raw data to consist of one or more files of fixed-length records of a single format. The fields must also be fixed-length and those fields being loaded (not all fields need be loaded) must be of one of the following data types:

- Character String
- Binary Integer Number
- Floating Point Number

Future releases of ODL will support multiple record types, variable length records, and all host-supported data types.

3.0 ODL OPERATION

ODL will normally operate as a batch process. Therefore the control statements for ODL are stored on a control file and used during execution. A log file is utilized for error messages and statistics resulting from the execution of ODL. The raw data files are specified in the control statements. A reject file is maintained for up to 50 rejected raw data records (input records are rejected when they can not be loaded into a table).



ODL is invoked with the following statement:

```
ODL  ctl-file  log-file  database  [user-id/pswrd]
```

where:

ctl-file is the name of the file containing the ODL Control Language Statements.

log-file is the name of the file for logging error messages and statistics of the load processor.

database is the name of the database on which the load is being performed.

user-id is the name of the user as required for a secure database.

pswrd is the password for that user.

In the following example ODL is being invoked to load data into the Personnel database (secure). The ODL Control statements are on a file called load1.ctl and logging will be to a file called log1.

```
ODL  LOAD1.CTL  LOG1  PERSONNEL SCOTT/TIGER
```

The reject file will be given the name of the log file (less any extensions) plus the extension <.bad>. For example, it the log file was named <msg.log> (or just <msq>), the reject file would receive the name <msg.bad>. In the above example, the reject file will be named <LOG1.BAD>

Although ODL will allow up to 50 rejected raw data records before it terminates, the first record read must be acceptable or ODL will terminate.

4.0 ODL CONTROL LANGUAGE

The language is composed of definition and action statements. Definition statements define the raw data formats and the source files for the raw data. Action statements define the mapping of the input values and constants into the database table rows.

4.1 DEFINE RECORD STATEMENT

The DEFINE RECORD statement defines a particular input record. A list of one or more field definitions separated by commas, defines the raw data fields to be loaded.

```
DEFINE RECORD  rec-name  AS  [fld-def,...] fld-def;
```

where:

rec-name is a name used to refer to the raw data record.

fld-def is the data field definition described as follows:

```
fld-def = fld-name ( fld-type [, fld-loc] )
```

where:

fld-name is a name used to refer to a raw data field.

fld-type defines the data type as follows:

```
| FLOAT   |
| INTEGER | [ (size) ]
| CHAR    |
```

corresponding to a floating point, binary integer, and character. The size in bytes (enclosed in parenthesis) is optional and is required for sizes other than defaults. Acceptable sizes and defaults are as follows:

data type	sizes	defaults PDP-11	defaults VAX-11
FLOAT	8, 4	8	8
INTEGER	4, 2, 1	2	4
CHAR	1 thru 254	1	1

fld-loc defines the position of the field in the raw data record as follows:

```
                | + number |
LOC   ( | - number | )
                |   number |
```

An integer number without any sign is an absolute location from the beginning of the record. A sign in front of the number means the position is relative to the end of the last field. The minus sign is positioned towards the beginning of the record and the plus sign is positioned towards the end. The default position is at the end of the previous field or the beginning of the record for the first field.

If a database column is defined as NUMBER type and the raw data field is an ASCII string representing a numerical quantity, then the data type of the DEFINE RECORD filed must be CHAR. ORACLE handles the conversion to numerical format.

The following is a sample DEFINE RECORD statement:

```
DEFINE RECORD REC1 AS
   FLD1 (CHAR(6)), FLD2 (CHAR(25)),
   FLD3 (FLOAT, LOC (+3))
;
```

4.2 DEFINE SOURCE STATEMENT

The DEFINE SOURCE statement defines the input mediums to ODL and the records contained within this source.

```
DEFINE SOURCE   src-name   src-parms
    CONTAINING rec-name ;
```

where:

src-name is a symbolic name for this source.

src-parms provide the information necessary to read the source as follows:

```
FROM file-name [,file-name]...
LENGTH number
```

where:

file-name is the operating system file name of a raw data file. Multiple file names (separated by commas) can be used to refer to concatenated files.

length is the raw data record length.

rec-name is the record name used in the DEFINE RECORD statement.

Future releases will provide for multiple record types within a source and for multiple sources during a load.

The following is an example of a DEFINE SOURCE statement.

```
DEFINE SOURCE SRC1
   FROM FILE1, FILE2, FILE3
   LENGTH 80
   CONTAINING REC1
;
```

4.3 FOR EACH STATEMENT

The FOR EACH statement is a loop mechanism that reads one raw data record for each iteration. The FOR EACH statement defines the SQL INSERT statement to be executed for each record read. Future releases will provide for the use of multiple INSERT statements involving multiple database tables, and nested FOR EACH statements for hierarchical data.

```
FOR EACH RECORD    in-stmt   NEXT RECORD
```

where:

(in-stmt) is the SQL INSERT statement as follows:

```
INSERT INTO tab-name
    (col-name,...)   :  < src-data,... >
```

where:

tab-name identifies the database table.

col-name is the list of one or more columns, enclosed in parenthesis for which values are being loaded.

src-data is the list of raw data field names and data values, enclosed in angle brackets and separated by commas. The list can contain any of the following:

- o raw data field-name
- o character string
- o number (string of digits or scientific notation)
- o NULL

Note that the "NEXT RECORD" phrase is required (to terminate the loop function).

The following is an example of a FOR EACH statement.

```
FOR EACH RECORD
    INSERT INTO EMP ( EMPNO, EMPNAME, SALARY, COMM ) :
     < FLD1, FLD2, FLD3, NULL >
NEXT RECORD
```

5.0 ODL MESSAGES

All ODL messages are written to the log file. The following statistical messages are written to the log file:

```
# records successfully read
# records successfully loaded
```

When an error occurs on processing a record, the following message is produced:

```
RECORD # REJECTED
```

where # is the relative number of the record being read.

There are three types of errors which can occur during ODL processing:

- o Control Statement Errors
- o ODL Fatal Errors
- o ORACLE Errors

When an error occurs during processing of a raw data record, that record is written to the reject file. ODL will terminate after 50 records have been written to the reject file. If there is a problem during processing of the first raw data record, ODL will terminate without loading any records.

1. Control Statement Errors

These messages include information to help locate the error in the statement. There are 3 formats used as follows:

1) message ON LINE # COLUMN #
   where message is one of the following:
   - Number too large
   - String to large
   - Bad number
   - Missing exponent
   - Identifier too large
   - Illegal character
   - Quoted literal not ended

2) SYNTAX ERROR LINE # ON INPUT symbol
   where symbol identifies where the error was detected

3) symbol : message ON OR ABOUT LINE #
   where symbol and message are as follows:

symbol	message
rec field	Field name is ambiguous
rec field	Field is improperly aligned
rec field	Inconsistent rec field definition
rec field	Field location too negative
rec field	Bad location field expression
rec field	Inconsistent location field
rec field	Unsupported field length
source	Source from clause is missing
source	Source length clause missing
source	Multiple sources not supported
source	Bad source length clause
rec name	Record too large for source
rec name	Multiple records not supported
table	Table col/field count mismatch
table	Multiple tables not supported
symbol	Not a record field
symbol	Undefined record field
symbol	Previously defined
symbol	Record name expected
symbol	Undefined record name

2. ODL Fatal Errors

These errors occur because of an internal problem or because of a resource, such as memory, being exhausted. The format is as follows:

```
ODL FATAL ERROR : message
```

where message is as follows:

```
Out of parse stack space
Out of heap space *
Out of table space *
```

Those messages which can be corrected by linking ODL with more memory are indicated with an asterisk.

There are a set of fatal messages which can occur if other errors exist before the fatal error. These errors will usually disappear when the original error is corrected. These errors are as follows:

```
excsub : invalid kind
Illegal insert field definition
defsyn : unknown type
unreachable code - optional length
unreachable code - default length
```

3. ORACLE Errors

ORACLE error messages have the following format:

```
call ERROR : message
```

where:

call	is the name of the ORACLE interface subroutine which failed;
message	is the ORACLE error message as listed in the ORACLE Messages and Codes Manual.

6.0 SAMPLE ODL CONTROL LANGUAGE INPUT

The following ODL control language sequence defines a record containing an employee name, grade and salary. the salary field is located 10 bytes away from the end of the grade field.

The input source has two files with a record length of 80.

The database table EMPMAIN is loaded with fields from the EMPLOYEE raw data records and with some constants.

```
DEFINE RECORD EMPLOYEE AS
     NAME   (CHAR (20) ),
     GRADE  (INTEGER(2) ),
     SALARY (FLOAT(4), LOC( +10) );


define source tape
     from dr0:[1,110]file1, file 2
     length 80
     containing EMPLOYEE ;

FOR EACH RECORD
     INSERT INTO EMPMAIN
     ( NAME, PERFORMANCE, GRADE, GROSS,
        NULCOL, SALARY, START) :
     <NAME, 87.6543, EMPLOYEE GRADE, .86e + 6,
        NULL, SALARY, 356 >
NEXT RECORD
```

O R A C L E

UNLOAD/RELOAD DATABASE UTILITY

This utility provides the function of unloading tables of an ORACLE database onto a sequential file and later reloading them into a database. This is useful for:

- o physically reordering rows of a table;
- o unloading a table, changing column size, nonull, or image specifications, and reloading that table;
- o moving a database between versions of ORACLE when the internal format of the database changes.

The following specific functions are supported:

- o Unload all tables and/or views of a database;
- o Unload selected tables and/or views of a database ;
- o Unload tables and/or views without GRANT privileges;
- o Unload only the table and view definitions;
- o Reload the previously unloaded tables and/or views;
- o Reload the tables and/or views without GRANT privileges;
- o Reload previously unloaded tables by inserting into existing tables;
- o Display the names of tables and views on an unloaded database.

Tables and views of a secure database may only be unloaded and reloaded by the owners of the data (creators of the tables or definers of the views). Therefore Unload/Reload operations on a secure database require the use of the user-name/password parameter.

Unloading a database is called EXPorting and reloading a database is called IMPorting.

EXPORT (Unload Tables)

EXP will unload one or more tables and/or views of a database. EXP will display the names of tables and views unloaded, and will display row counts for each table unloaded.

```
EXP db-name [user-name/paswrd] [dumpfile] [- T V G D Cn ]
```

where:	db-name	is the name the database being unloaded.
	dumpfile	is the optional name which if provided, will be used to name the output file. Otherwise, the file name will be <database-name>.
	user-name/ paswrd	is the user-name and password required when unloading tables from a secure database.
	T	is the option which causes EXP to prompt the user for the table-names identifying the tables to be unloaded. Entering a <return> in answer to the prompt signifies that no more tables are to be unloaded.
	V	is the option which causes EXP to prompt the user for the view-names identifying the views to be unloaded. Entering a <return> in answer to the prompt signifies that no more views are to be unloaded.

G	is the option which precludes the unloading of GRANT privileges associated with the tables and views being unloaded.
D	is the option to unload **only** the table/view definitions and GRANT privileges.
Cn	is the option which overrides the default value for the size of the SWA (SQL Work Area) in ORACLE. The size is expressed as a number (n=1 to 16) of KBytes. The default is 3.

EXAMPLE 1 : EXP PERSONNEL SCOTT/TIGER PERS.SAV -C5

EXAMPLE 2 : EXP CREDIT -TV

The first example will cause the entire PERSONNEL database (secure) to be unloaded to a file called PERSONNEL.SAV. In this example the SWA is being overridden and set to 5 KB.

The second example will cause selected tables and views of the CREDIT (non-secure) database to be unloaded to a file called CREDIT.EXP. The users will will be prompted for the tables and views to be unloaded.

IMPORT (Reload Tables)

IMP will reload the one or more tables and/or views of a previously unloaded database. If for some reason, a table cannot be created in the reloaded database, processing continues with the next table. If a record can not be accepted, that record will be displayed and processing will continue with the next record. If a view definition is rejected, it will be displayed.

IMP will display the names of tables and views being reloaded, and for each table reloaded, will display the rows inserted. If the number of rows actually loaded is different than the rows read, this is indicated with a count of rows read.

```
IMP  db-name  [user-name/paswrd]  [dumpfile]  [- T V G D I Cn ]
```

where:	db-name	is the name of the database being reloaded. This may or may not be the same as the name of the database which was unloaded.
	dumpfile	is the optional name which if provided, is used to reference the input file. Otherwise, the name of the input file will be assummed to be <database-name>.
	user-name/ paswrd	is the user-name and password required when reloading tables to a secure database.

T	is the option which **prevents** reloading of tables.
V	is the option which **prevents** reloading of views.
G	is the option which **prevents** the inclusion of GRANT privileges on the tables and/or views being reloaded.
D	is the option which overrides all load options and simply displays the names of tables and views in the dump file
I	is the option which specifies that the reload is taking place against an existing table and the rows are to be inserted into that table. If this parameter is not specified, a reload against an existing table would result in no rows being inserted in that table.
Cn	is the option which overrides the default value for the size of the SWA (SQL Work Area) in ORACLE. The size is expressed as a number (n=1 to 16) of KBytes. The default is 3.

```
EXAMPLE 1 :  IMP  PERSONNEL  SCOTT/TIGER  PERS.SAV  -C5

EXAMPLE 2 :  IMP  CREDIT  -VI
```

The first example will cause the PERSONNEL database (secure) to be reloaded from a file called PERS.SAV with a SQL Work Area of 5KB.

The second example will cause only the tables of the unloaded CREDIT database (non-secure) to be reloaded from a file called CREDIT, and if any tables being reloaded are already in the database, the rows will be inserted.