ODU User’s Guide

The First Edition

2011.4

目录

CHAPTER 1 – INTRODUCTION

WHAT IS ODU

The full name of ODU is Oracle Database Unloader, a tool developed by OracleODU. Similar to DUL (a famous Oracle database recovery utility), ODU can unload data from a complete crashed or corrupted database, even if the database cannot be started anymore or the database itself cannot access the data for whatever reason. We have used it in several real life situations already (customers with a crashed database and no backup), with 100% success rate. In some real cases, DUL can not unload all the important data, but ODU can.

In reality there is always a lot of accidents, the data is accidentally deleted, the database corruption caused by damaged hardware, the ASM disk is incorrectly formatted, etc., in the case of no backup, ODU can directly read the Oracle database datafiles or directly access to the ASM disks, recover all good data intact to avoid the loss of data.

ODU'S MAIN FEATURES

- Bypass Oracle's database engine, extracting data directly at the block level.
- Supports ASM - ODU can unload data directly from ASM disks even all the diskgroups are dismounted.
- Supports extract files of any type directly from ASM disks even all the diskgroups are dismounted, including datafile, redo log, archive log, etc.
- Supports Oracle RDBMS versions 7, 8i, 9i,10g and 11g.
- Supports multiple database platforms, including AIX, LINUX, HPUX, SOLARIS, WINDOWS and so on. Supports cross-platform unloading, for example unloading AIX based datafiles on a Windows host.
- Supported data types: NUMBER, CHAR, VARCHAR2, NCHAR, NVARHCAR2, LONG, DATE, RAW, LONG RAW, BLOB, CLOB, TIMESTAMP (9i +), BINARY FLOAT, BINARY DOUBLE (10g +).
- Fully support LOB:
    - Supports CLOB, NCLOB and BLOB
    - Supports CLOB big endian and little endian byte order
    - Supports partitioned and subpartioned LOBs
    - Supports different chunk sizes of different LOB columns in the same table
    - CLOB data can be exported to the same file with other columns, or stored in a separate file
    - LOBs are still be able to export even the SYSTEM tablespace is not available
    - LOBs are still be able to export even the associated lob index is corrupted
- Supports various types of tables, including ordinary HEAP table, IOT table, CLUSTER

table.

- Supports IOT, supported IOT types are:
  - Ordinary IOT
  - Compressed IOT
  - IOT with overflow segments
  - Partitioned and subpartitioned IOT
  - IOT's are only supported when the SYSTEM tablespace is available
- Supports compressed table.
- Supports data recovery of truncated table.
- Supports data recovery of dropped table.
- Automatic acquisition of data dictionary information if SYSTEM tablespace is not totally corrupted.
- Supports data recovery in the absence of SYSTEM tablespace and data dictionary corruption. If data dictionary is not available, ODU can automatically determine the data type of a data column.
- Supports BigFile tablespace in Oracle 10g and above.
- Fully support for 64-bit systems, supports more than 4G size of the datafiles.
- Supports bad file copy even the operating system command (for example, cp) can not copy successfully.
- Supports different block size of datafiles in the same database.
- Supports conversion between various character sets, can convert CLOB, NCLOB, NVARCHAR2 column type of data to the specified character set correctly.
- Auto detection of tablespace number, file number and block size of datafiles.
- Export data formats including both plain text and DMP file. When exporting to plain text, ODU can generate the necessary sql statements for creating table and control file used for SQL*Loader automatically.
- Simulated dump block function of the Oracle, can dump data blocks from datafiles.
- Supports DESC command to a table to display the column definition.
- Supports to list all table partitions and subpartions.

ODU'S OTHER FEATURES

In addition to these functional characteristics, ODU also has the following characteristics:

➢ **Faster recovery rate, less memory usage**

ODU is written in C language, which is the fastest and most efficient memory usage in all major languages. ODU can achieve the same speed of data analysis with the Oracle software itself. ODU also perform some functions in parallel to increase the speed of recovery. ODU supports command file, you can improve data recovery speed by using different command file and parallel execution. When conducting data recovery in the above way, you can greatly reduce the required time, thereby reducing data loss and damage due to the business stop time.

➢ **Very easy to use**

ODU is very easy to use, only needs simple configuration plus two or three commands to restore data. When the data is damaged or lost, you can use the ODU to recover it without the specialized training and long learning curve.

- **Stable operation**

ODU can automatically detect and skip the bad blocks in the database, to avoid the abnormal exit by corrupt block parsing and maintain the stability of data recovery.

FEATURES ODU CURRENTLY DOES NOT SUPPORT

- 11g Securefiles
- Encrypted data using the Oracle TDE
- BFILE
- NESTED TABLE, user-defined data types

Data recovery is not affected because these types of data are rarely used.

CHAPTER 2 – INSTALLATION AND USAGE

Just three simple steps to complete the installation of ODU.

DOWNLOAD ODU SOFTWARE

You can download the latest version of ODU from http://www.oracleodu.com/en/download. When you download ODU, you need to make sure the platform you want to run it. ODU software installation packages are named as follows:

odu_<version>_<platform>.zip

or

odu_<version>_<platform>.tar.gz

If the version number contains the keyword "trial", means it is the trial version.

For example：

- ✧ odu_413_hp_ia64.tar.gz means that ODU is enterprise version, run in HP-UX operating system on Itanium platform, the version number is 4.1.3.
- ✧ odu_trial_411_hp_ia64.tar.gz means that ODU is trial version, run in HP-UX operating system on Itanium platform, the version number is 4.1.1.
- ✧ odu_413_linux_x86.tar.gz means that ODU is enterprise version, run in Linux operating system on x86 platform, the version number is 4.1.3.
- ✧ odu_413_win32.zip means that ODU is enterprise version, run in 32-bit Windows operating system, the version number is 4.1.3.

Tips：

Most of the new version of the operating system are compatible with the software on low version, while the high number (64 bit) platform can be able to run the low number (32 bit) platform software. For example, odu_413_win32 is able to run on all Windows 2000/xp/2003/Win7 operating system, regardless of operating system is 32-bit or 64-bit. The odu_413_linux_x86 can also be able to run on the major Linux distributions, regardless of operating system is 32-bit or 64-bit.

Tips：

When using IE browser to download ODU, the extension of the ODU installation package may be automatically changed from tar.gz to tar.tar, for such a change, please change the file extension back to tar.gz manually.


Tips：

The difference between ODU Enterprise Version and ODU Trail Version——ODU trial version only works for testing, learning and validation, it can only unload the data in SYSTEM tablespace, for the data in other tablespaces, it only unloads a small amount of data to verify the data recoverability. The enterprise version can be able to unload all the data which can be recovered after you get license.

CREATE DIRECTORIES AND UPLOAD ODU SOFTWARE

Create the directory to install the ODU software on the target system. Download the ODU software installation package via ftp/sftp and upload it to the directory. Just download, unzip and copy the installation package to the directory if the target system is Windows.

Recommend to use the Oracle software owner (usually Oracle) to install the ODU software. If use other users, ODU may not be able to read the target datafiles.

ODU installation directory includes the subdirectory odu. If you plan to install the ODU to directory /oracle/odu, you need to create directory /oracle and copy the ODU software installation package to directory /oracle.

ODU software itself is not much hard disk space required, 10MB of space is sufficient. But the space required for ODU metadata (mainly the data dictionary) which is generated during the unload operation depends on the database size. However, 200MB is usually enough.

UNZIP ODU SOFTWARE

For ODU installation package in Windows, use winzip, winrar or 7zip to unzip the package into the specified directory. For the installation package in Linux/Unix, use the gunzip and tar command to unzip it.
For example, if you plan to install the ODU into directory /oracle/odu in Linux, the installation package has been already copied to directory /oracle, then you can use the following commands:

```
cd /oracle
gunzip xf odu_413_linux_x86.tar.gz
tar xf odu_413_linux_x86.tar
```

After decompression of ODU installation package, you will usually find the following files and directories:

- ✧ odu # ODU executable file, which is the main program.
- ✧ config.txt # ODU configuration file.
- ✧ control.txt # ODU control file, used to specify Oracle datafiles.
- ✧ asmdisk.txt # ASM disk configuration file, used to specify ASM diskgroups and ASM disks.
- ✧ lib # ODU library directory, do not have this directory in some platforms.
- ✧ data # ODU default data directory, used to store recovered data.

After decompression of ODU installation package, the ODU main program may not be able to execute due to the lack of correct permission, in this case, use the following commands (assuming ODU is installed in /oracle/odu):

```
cd /oracle/odu
ls -l odu
-rwxr-xr-x 1 oracle oinstall 2677388 Apr  1 23:24 odu

If the above results show that the ODU does not have "x" attribute, you need to execute the following command:
chown u+x odu
```

USE ODU

After the installation is complete, enter into the ODU installation directory, for Windows, you can enter into the ODU command interface by running odu.exe directly; and for Linux/Unix system, execute the command ./odu for the same purpose. However, part of the operating system may report an error similar to the following:

```
./odu: error while loading shared libraries: libiconv.so.2: cannot open shared object file: No such file or directory
```

This is due to the library files needed to execute ODU is not in the search path. Execute the following command: export LD_LIBRARY_PATH = <ODU installation directory>/lib:$ LD_LIBRARY_PATH, and then run the command ./odu again (please replace the <ODU installation directory> by the actual ODU installation directory).

In Solaris and HP-UX, this environment variable is LD_LIBRARY_PATH, but in AIX, this environment variable is LIBPATH.

The library files which contains in ODU installation package may already exist in the system. To avoid problems caused by different versions of those library files, you can put the ODU lib directory in the front part of the environment variable (LIBPATH or LD_LIBRARY_PATH).

To avoid repeated set LD_LIBRARY_PATH and LIBPATH, you can add this environment variable to the user's profile file.

After entering into ODU, ODU will display the command prompt ODU>, so that you can enter commands to perform the operation:

```
ODU>
```

CHAPTER 3 – HOW TO RECOVER DATA

ODU DATA RECOVERY QUICK START

The following example uses ODU for Linux to recover data from the database using ASM diskgroups. Now assume that the database can not be opened, we need to unload important data from the table SYS.T1.

1、 Update ODU ASM disk configuration file (asmdisk.txt)

Enter into the ODU installation directory and use vi to modify asmdisk.txt:

```
cd /oracle/odu
vi asmdisk.txt
```

Adding all the ASM disk device files path and name to the asmdisk.txt:

```
0   /oradata/asm/disk1.dbf
0   /oradata/asm/disk2.dbf
0   /oradata/asm/disk3.dbf
```

Note: In this example, the above datafiles are used to simulate the ASM disks in a test environment.

2、 Modify ODU control file control.txt

Enter into the ODU installation directory and use vi to modify control.txt, Adding all the datafiles path and name to the control.txt:

```
0    0    0    +DGDATA/xty/datafile/system.260.745630773
0    0    0    +DGDATA/xty/datafile/undotbs1.261.745630805
0    0    0    +DGDATA/xty/datafile/sysaux.262.745630817
0    0    0    +DGDATA/xty/datafile/users.264.745630833
```

3、 Execute the command ./odu to enter into the ODU command interface:

```
$ ./odu

Oracle Data Unloader:Release 4.1.3

Copyright (c) 2008,2009,2010,2011 XiongJun. All rights reserved.

Web: http://www.oracleodu.com
```

```
Email: magic007cn@gmail.com
loading default config.....

byte_order little
block_size    8192
block_buffers 1024
db_timezone -7
client_timezone 8
asmfile_extract_path     /asmfile
data_path     data
lob_path      /odu/data/lob
charset_name US7ASCII
ncharset_name AL16UTF16
output_format text
lob_storage infile
clob_byte_order big
trace_level 1
delimiter |
unload_deleted no
file_header_offset 0
is_tru64 no
record_row_addr no
convert_clob_charset yes
use_scanned_lob    yes
trim_scanned_blob yes
lob_switch_dir_rows 20000
db_block_checksum yes
db_block_checking yes
rdba_file_bits 10
compatible 10
load config file 'config.txt' successful
loading default asm disk file ......


grp# dsk# bsize ausize disksize diskname             groupname           path
---- ---- ----- ------ -------- ---------------- ---------------- ----------------------
    1            1      4096      1024K             1024   DGDATA_0001             DGDATA
/oradata/asm/disk1.dbf
    1            0      4096      1024K             1024   DGDATA_0000             DGDATA
/oradata/asm/disk2.dbf
    1            2      4096      1024K             1024   DGDATA_0002             DGDATA
/oradata/asm/disk3.dbf

load asm disk file 'asmdisk.txt' successful
```

```
loading default control file ......


 ts#     fn    rfn bsize     blocks bf offset filename
---- ---- ---- ----- -------- -- ------ ----------------------------------------
    0               1               1        8192               44800    N                    0
+DGDATA/xty/datafile/system.260.745630773
    1               2               2        8192               25600    N                    0
+DGDATA/xty/datafile/undotbs1.261.745630805
    2               3               3        8192               15360    N                    0
+DGDATA/xty/datafile/sysaux.262.745630817
    4               4               4        8192                 800    N                    0
+DGDATA/xty/datafile/users.264.745630833
load control file 'control.txt' successful
loading dictionary data......done

loading scanned data......done
```

4、 Run command save control：

```
ODU> save control

The file write completed.
ODU> exit
```

The command "save control" will generate a file named oductl.txt in the ODU installation directory, then you can execute the exit command to exit the ODU command interface.

5、 Get license.

Send the file oductl.txt to the specified support mailbox in http://www.oracleodu.com/en/support , you will get the license file named oductl.dat, then copy this file to the ODU installation directory.

Tips：

One license can only recover the data in one database, while the license also has 30-day expiration period, which is to avoid the misuse of ODU to directly access the sensitive data in the Oracle database to ensure data security.

Before the recovery, you only need to get a license once, do not need to repeatedly obtain license for the same database.

For the ODU trial version, you don't need to implement the above steps 4 and 5.

6、 Get data dictionary information.

Re-enter into the ODU command interface, execute the command "unload dict" to get data dictionary information

```
ODU> unload dict
CLUSTER C_USER# file_no: 1 block_no: 89
TABLE OBJ$ file_no: 1 block_no: 121
CLUSTER C_OBJ# file_no: 1 block_no: 25
CLUSTER C_OBJ# file_no: 1 block_no: 25
found IND$'s obj# 19
found IND$'s dataobj#:2,ts#:0,file#:1,block#:25,tab#:3
found TABPART$'s obj# 266
found TABPART$'s dataobj#:266,ts#:0,file#:1,block#:2121,tab#:0
found INDPART$'s obj# 271
found INDPART$'s dataobj#:271,ts#:0,file#:1,block#:2161,tab#:0
found TABSUBPART$'s obj# 278
found TABSUBPART$'s dataobj#:278,ts#:0,file#:1,block#:2217,tab#:0
found INDSUBPART$'s obj# 283
found INDSUBPART$'s dataobj#:283,ts#:0,file#:1,block#:2257,tab#:0
found IND$'s obj# 19
found IND$'s dataobj#:2,ts#:0,file#:1,block#:25,tab#:3
found LOB$'s obj# 151
found LOB$'s dataobj#:2,ts#:0,file#:1,block#:25,tab#:6
found LOBFRAG$'s obj# 299
found LOBFRAG$'s dataobj#:299,ts#:0,file#:1,block#:2393,tab#:0
```

7、 Use the command unload to restore table data.

```
ODU> unload table sys.t1

Unloading table: T1,object ID: 42138 at 2011-04-07 13:58:41
Unloading segment,storage(Obj#=42138 DataObj#=42138 TS#=4 File#=4 Block#=11
Cluster=0)
41161 rows unloaded
At 2011-04-07 13:58:43
```

You can see that it only takes 2 seconds to restore 41161 rows of table T1 under SYS user.

8、 Import the data into the new database.

You can see 3 files located in subdirectory data in the ODU installation directory:

```
[oracle@xty data]$ ls -l
total 4800
-rw-r--r-- 1 oracle oinstall        597 Apr    7 13:58 SYS_T1.ctl
```

```
-rw-r--r-- 1 oracle oinstall       409 Apr   7 13:58 SYS_T1.sql
-rw-r--r-- 1 oracle oinstall 4893152 Apr   7 13:58 SYS_T1.txt

[oracle@xty data]$ cat SYS_T1.sql
CREATE TABLE "SYS"."T1"
(
      "OWNER" VARCHAR2(30) ,
      "OBJECT_NAME" VARCHAR2(128) ,
      "SUBOBJECT_NAME" VARCHAR2(30) ,
      "OBJECT_ID" NUMBER ,
      "DATA_OBJECT_ID" NUMBER ,
      "OBJECT_TYPE" VARCHAR2(19) ,
      "CREATED" DATE ,
      "LAST_DDL_TIME" DATE ,
      "TIMESTAMP" VARCHAR2(19) ,
      "STATUS" VARCHAR2(7) ,
      "TEMPORARY" VARCHAR2(1) ,
      "GENERATED" VARCHAR2(1) ,
      "SECONDARY" VARCHAR2(1)
);
[oracle@xty data]$ cat SYS_T1.ctl
--
--Generated by ODU,for table "SYS"."T1"
--
OPTIONS(BINDSIZE=8388608,READSIZE=8388608,ERRORS=-1,ROWS=50000)
LOAD DATA
INFILE 'SYS_T1.txt' "STR X'0a'"
APPEND INTO TABLE "SYS"."T1"
FIELDS TERMINATED BY X'7c' TRAILING NULLCOLS
(
      "OWNER" CHAR(30),
      "OBJECT_NAME" CHAR(128),
      "SUBOBJECT_NAME" CHAR(30),
      "OBJECT_ID" ,
      "DATA_OBJECT_ID" ,
      "OBJECT_TYPE" CHAR(19),
      "CREATED" DATE "yyyy-mm-dd hh24:mi:ss",
      "LAST_DDL_TIME" DATE "yyyy-mm-dd hh24:mi:ss",
      "TIMESTAMP" CHAR(19),
      "STATUS" CHAR(7),
      "TEMPORARY" CHAR(1),
      "GENERATED" CHAR(1),
      "SECONDARY" CHAR(1)
)
```

SYS_T1.sql is the SQL statements to create table SYS.T1, SYS_T1.ctl is the control file used by SQL*Loader for loading the restored table data into the database, SYS_T1.txt is the text file which stored actual data of table SYS.T1.
Using Oracle's SQL*Loader (sqlldr) to load the SYS_T1.txt into the database to complete the whole recovery process of table SYS.T1.

COMPLETE STEPS TO RECOVER DATA USING ODU

Although the use of ODU to restore the Oracle database data is relatively simple, but in order to complete the data recovery more smoothly and more quickly, we recommend the following steps to complete the recovery operation:

1) Determine the extent of data loss and recovery range.

Determine the extent of data loss before the recovery. Is the need to restore a small amount of table data due to the wrong DROP / TRUNCATE operation, or need to restore a small amount of table data because of the data corruption, or need to recover all the data under a user because the user is dropped, or the database can not be opened so you have to restore the whole database.

2) Determine the extent of database corruption.

You need to confirm whether the datafiles are all available, particularly the SYSTEM tablespace is intact. In the case of using ASM, whether the ASM disks are all available and the ASM diskgroups are able to be mounted. If the SYSTEM tablespace is missing or damaged, or the table/user is dropped, you need to use the particular recovery process which towards this situation. In this case, because of the missing data dictionary, the maintenance staff or developer which is very familiar with the system data and database/table structure is recommended to be involved in data recovery process.

3) Estimate the required storage space and prepare enough space to save the recovered data.

The recovery range has been determined in the first step, estimate the required storage space in such circumstances. If there is not enough storage space, you should seek the help of system administrators or storage administrators to allocate enough space.

4) Collect enough information.

The information includes the path of ASM disk, datafile name, database platform, database version, the existence of the datafile header offset in the case of using raw device.

5) Determine the format of saved recovery data.

ODU supports two types of saved format, one is text file format that can use SQL*Loader (sqlldr) to import it, and the other one is DMP file format that can use the IMP to import it. If the column data type of the recovered table is complicated, such as a LONG type, LOB type, the VARCHAR2 type with some special characters, the DMP format is recommended.

6) ODU Configuration.

After completing the steps mentioned above, modify ODU configuration parameters according to the collected information, modify ODU control file and ASM disk configuration file (if using ASM).

About ODU configuration parameters, see the following reference chapter on them.

About ODU control file and ASM disk configuration file, see the following reference chapter on them.

7) Get ODU enterprise version License.

Get ODU enterprise version License according to the steps in http://www.oracleodu.com/en/buy

8) Use ODU to recover data.

Use the command unload to recover data. About ODU's commands, see the following reference chapter on them.

9) Import and verify the recovered data.

Use the SQL*Loader (sqlldr) or IMP to import the recovered data into a new database. Not recommended to import into the original database, which is to protect the original data, to avoid reusing the original storage space by the imported data. As long as there is no problems after validating the data, then import the data into the original production database.

10) Create other objects.

After data recovery, you should also create other objects, then the database can be truly used in the production system. These objects include indexes, foreign key constraints and triggers. You also need to create stored procedures and other objects in the case of full database or user recovery.

The following sections provide a detailed description of how to use the ODU's functionality to recover data.

SEVERAL SCENARIOS OF ODU RECOVERY

This section describes several scenarios for data recovery and used ODU commands.

Scene 1. The database can not be opened, but the data dictionary information in the system tablespace is intact.

- Generate data dictionary information: unload dict
- List users: list user
- Lists all tables under a user: list table username
- Recover a table: unload table username.tablename
- Recover all tables under a user: unload user username

Scene 2. Table is truncated

- Offline the tablespace where the table belongs to
- Generate data dictionary information: unload dict
- Scan extent information: scan extent
- Recover the table: unload table username.tablename object truncate

If the above steps are unsuccessful, continue to perform the following steps:

- List the table's segment header: desc username.tablename
- Find the target data object id: dump datafile file# block block#
- Recover the table: unload table username.tablename object <data_object_id>

Scene 3. Table is dropped

- Offline the tablespace where the table belongs to
- Use logminer to excavate the redo log to get the data object id of the dropped table, or use the flashback query to query table SYS.OBJ$ for the data object id of the dropped table, if you can not get the data object id, according to the following scenario 4 for recovery.
- Scan extent information: scan extent
- If you don't have the table structure information, ODU can automatically determine the column type: unload object data_object_id sample
- Recover the table: unload object data_object_id column coltype coltype…

Scene 4. System tablespace is missing or corrupted

- Scan extent information: scan extent
- Get sample data: unload object all sample
- Find the required data from the result file sample.txt
- Recover the table: unload object data_object_id column coltype coltype…

Scene 5. The table data is accidentally deleted

- Set the parameter unload_deleted to YES
- Generate data dictionary information: unload dict
- Recover the table: unload table username.tablename

Scene 6. The table has some corrupted blocks

- Generate data dictionary information: unload dict
- Scan extent information: scan extent
- Recover the table: unload table username.tablename object scanned

USE ODU TO RECOVER TURUNCATED TABLE

Truncate table happened accidentally from time to time, ODU offers a convenient way to recover it. As long as the original space is not reused (ie, the original data is not overwritten), the data of the truncated table can be all recovered.

If you find a table is accidentally truncated, and needs immediate restoration. First thing to do is to shutdown the database, or offline the tablespace where the table belongs to, or close all the applications. Just for one purpose, to ensure that the space will not be reused, the original data will not be overwritten.

There are two ways to recover truncated table, one is to use the ODU command "unload table <schema.tablename> object truncate", the other is to manually find the table's data object id before truncate and then use the ODU command "unload table <schema.tablename> object <data_object_id> ".

The following is an example to use the first way to recover truncated table:

1) Create two test tables T1 and T2, the data are exactly the same in two tables. The purpose is to facilitate comparison and verify the data after the recovery. Then truncate table T1.

```
SQL> conn test/test
Connected.

SQL> create table t1 as select * from dba_objects where rownum<=1000;

Table created.

SQL> create table t2 as select * from t1;

Table created

SQL> truncate table t1;

Table truncated.
```

2) Offline the tablespace where the T1 belongs to (in the actual system, if there are some active transactions, the tablespace is not easily to be offline down). Then make a full checkpoint, so that ODU can read the latest data dictionary information.

```
SQL> select tablespace_name from user_tables where table_name='T1';

TABLESPACE_NAME
------------------------------
TBS_TEST

SQL> alter tablespace tbs_test offline;

SQL> alter system checkpoint;
```

3) Run the ODU, and unload the data dictionary:

```
ODU> unload dict
CLUSTER C_USER# file_no: 1 block_no: 89
TABLE OBJ$ file_no: 1 block_no: 121
CLUSTER C_OBJ# file_no: 1 block_no: 25
CLUSTER C_OBJ# file_no: 1 block_no: 25
found IND$'s obj# 19
found IND$'s dataobj#:2,ts#:0,file#:1,block#:25,tab#:3
found TABPART$'s obj# 266
found TABPART$'s dataobj#:266,ts#:0,file#:1,block#:2121,tab#:0
found INDPART$'s obj# 271
found INDPART$'s dataobj#:271,ts#:0,file#:1,block#:2161,tab#:0
found TABSUBPART$'s obj# 278
found TABSUBPART$'s dataobj#:278,ts#:0,file#:1,block#:2217,tab#:0
```

```
found INDSUBPART$'s obj# 283
found INDSUBPART$'s dataobj#:283,ts#:0,file#:1,block#:2257,tab#:0
found IND$'s obj# 19
found IND$'s dataobj#:2,ts#:0,file#:1,block#:25,tab#:3
found LOB$'s obj# 151
found LOB$'s dataobj#:2,ts#:0,file#:1,block#:25,tab#:6
found LOBFRAG$'s obj# 299
found LOBFRAG$'s dataobj#:299,ts#:0,file#:1,block#:2393,tab#:0
```

4) Get the information of table T1 under the user test:

```
ODU> desc test.t1


Object ID:42252
Storage(Obj#=42252 DataObj#=42255 TS#=6 File#=20 Block#=11 Cluster=0)

NO. SEG INT Column Name                    Null?      Type
--- --- --- ------------------------------ --------- ------------------------------
  1   1   1 OWNER                                     VARCHAR2(30)
  2   2   2 OBJECT_NAME                                VARCHAR2(128)
  3   3   3 SUBOBJECT_NAME                             VARCHAR2(30)
  4   4   4 OBJECT_ID                                 NUMBER
  5   5   5 DATA_OBJECT_ID                            NUMBER
  6   6   6 OBJECT_TYPE                               VARCHAR2(19)
  7   7   7 CREATED                                   DATE
  8   8   8 LAST_DDL_TIME                            DATE
  9   9   9 TIMESTAMP                                 VARCHAR2(19)
 10  10  10 STATUS                                    VARCHAR2(7)
 11  11  11 TEMPORARY                                 VARCHAR2(1)
 12  12  12 GENERATED                                 VARCHAR2(1)
 13  13  13 SECONDARY                                 VARCHAR2(1)
```

From the above output, we can see that the tablespace number of the tablespace where T1 belongs to is 6.

5) Scan the tablespace's extent with ODU:

```
ODU> scan extent tablespace 6

scan extent start: 2011-04-09 11:50:07
scanning extent...
scanning extent finished.
scan extent completed: 2011-04-09 11:50:25
```

6) Use ODU to unload the data:

```
ODU> unload table test.t1 object truncate
Auto mode truncated table.

Unloading table: T1,object ID: 42252 at 2011-04-09 11:51:02
Unloading segment,storage(Obj#=42252 DataObj#=42252 TS#=6 File#=20 Block#=11 Cluster=0)
1000 rows unloaded
At 2011-04-09 11:51:03
```

7) Online the tablespace tbs_test：

```
SQL> alter tablespace tbs_test online;
```

8) Use SQL*Loader (sqlldr) to import the recovered data:

```
[oracle@xty data]$ sqlldr test/test control=TEST_T1.ctl

SQL*Loader: Release 10.2.0.5.0 - Production on Sat Apr 9 11:51:56 2011

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

Commit point reached - logical record count 1000
```

At this point, all the steps to recover the data have been completed. Compare the recovered data and the data before truncate to see whether they are exactly the same:

```
SQL> select * from t2 minus select * from t1;

no rows selected

SQL> select * from t1 minus select * from t2;

no rows selected
```

You can see that the data has been completely restored.

The following is an example to use the second way to recover truncated table if the first way dose not work:

1) Create two test tables T1 and T2, the data are exactly the same in two tables. The purpose is to facilitate comparison and verify the data after the recovery. Then truncate

table T1.

```
SQL> connect test/test

SQL> create table t1 as select * from dba_objects;

SQL> create table t2 as select * from t1;

SQL> truncate table t1;
```

2) Offline the tablespace where the T1 belongs to (in the actual system, if there are some active transactions, the tablespace is not easily to be offline down). Then makes a full checkpoint, so that ODU can read the latest data dictionary information.

```
SQL> select tablespace_name from user_tables where table_name='T1';

TABLESPACE_NAME
------------------------------
TEST

SQL> alter tablespace test offline;

SQL> alter system checkpoint;
```

3) Run the ODU and unload the data dictionary:

```
ODU> unload dict
get_bootstrap_dba: compat header size:12
CLUSTER C_USER# file_no: 1 block_no: 177
TABLE OBJ$ file_no: 1 block_no: 241
CLUSTER C_OBJ# file_no: 1 block_no: 49
CLUSTER C_OBJ# file_no: 1 block_no: 49
found IND$'s obj# 19
found IND$'s dataobj#:2,ts#:0,file#:1,block#:49,tab#:3
found TABPART$'s obj# 230
found TABPART$'s dataobj#:230,ts#:0,file#:1,block#:3313,tab#:0
found INDPART$'s obj# 234
found INDPART$'s dataobj#:234,ts#:0,file#:1,block#:3377,tab#:0
found TABSUBPART$'s obj# 240
found TABSUBPART$'s dataobj#:240,ts#:0,file#:1,block#:3473,tab#:0
found INDSUBPART$'s obj# 245
found INDSUBPART$'s dataobj#:245,ts#:0,file#:1,block#:3553,tab#:0
found IND$'s obj# 19
found IND$'s dataobj#:2,ts#:0,file#:1,block#:49,tab#:3
```

```
found LOB$'s obj# 156
found LOB$'s dataobj#:2,ts#:0,file#:1,block#:49,tab#:6
found LOBFRAG$'s obj# 258
found LOBFRAG$'s dataobj#:258,ts#:0,file#:1,block#:3761,tab#:0
```

4) Get the information of table T1 under the user test:

```
ODU> desc test.t1

Object ID:33547
Storage(Obj#=33547 DataObj#=33549 TS#=11 File#=10 Block#=1400 Cluster=0)

NO. SEG INT Column Name                    Null?      Type
--- --- --- ------------------------------ --------- ------------------------------
  1   1   1 OWNER                                     VARCHAR2(30)
  2   2   2 OBJECT_NAME                               VARCHAR2(128)
  3   3   3 SUBOBJECT_NAME                            VARCHAR2(30)
  4   4   4 OBJECT_ID                                 NUMBER
  5   5   5 DATA_OBJECT_ID                           NUMBER
  6   6   6 OBJECT_TYPE                              VARCHAR2(18)
  7   7   7 CREATED                                  DATE
  8   8   8 LAST_DDL_TIME                           DATE
  9   9   9 TIMESTAMP                              VARCHAR2(19)
 10  10  10 STATUS                                   VARCHAR2(7)
 11  11  11 TEMPORARY                              VARCHAR2(1)
 12  12  12 GENERATED                              VARCHAR2(1)
 13  13  13 SECONDARY                              VARCHAR2(1)
```

From the above output, we can see that the tablespace number of the tablespace where T1 belongs to is 11 and the segment header block of table T1 is in datafile 10, block 1400.

5) Scan the tablespace's extent with ODU:

```
ODU> scan extent tablespace 11

scanning extent…
scanning extent finished.
```

6) We use ODU to determine the original data object id of the truncated table T1. In general, the valid data block is usually starts from the first block in the adjacent section after the segment header. We can dump the information in the header section to confirm:

```
ODU> dump datafile 10 block 1400
Block Header:
```

```
block type=0x23 (ASSM segment header block)
block format=0x02 (oracle 8 or 9)
block rdba=0x02800578 (file#=10, block#=1400)
scn=0x0000.00286f2d, seq=4, tail=0x6f2d2304
block checksum value=0x0=0, flag=0
Data Segment Header:
  Extent Control Header
  -------------------------------------------------------------
  Extent Header:: extents: 1    blocks: 5
                       last map: 0x00000000    #maps: 0    offset: 668
       Highwater:: 0x02800579    (rfile#=10,block#=1401)
                       ext#: 0    blk#: 3     ext size:5
       #blocks in seg. hdr's freelists: 0
       #blocks below: 0
       mapblk: 0x00000000     offset: 0
  -------------------------------------------------------------
  Low HighWater Mark :
       Highwater::    0x02800579    ext#: 0          blk#: 3          ext size: 5
  #blocks in seg. hdr's freelists: 0
  #blocks below: 0
  mapblk    0x00000000    offset: 0
  Level 1 BMB for High HWM block: 0x02800576
  Level 1 BMB for Low HWM block: 0x02800576
  -------------------------------------------------------------
  Segment Type: 1 nl2: 1          blksz: 2048     fbsz: 0
  L2 Array start offset:    0x00000434
  First Level 3 BMB:    0x00000000
  L2 Hint for inserts:    0x02800577
  Last Level 1 BMB:    0x02800576
  Last Level 1I BMB:    0x02800577
  Last Level 1II BMB:    0x00000000
     Map Header:: next    0x00000000    #extents: 1       obj#: 33549    flag: 0x220000000
  Extent Map
  -------------------------------------------------------------
   0x02800576    length: 5

  Auxillary Map
  -------------------------------------------------------------
   Extent 0         :    L1 dba:    0x02800576 Data dba:    0x02800579
  -------------------------------------------------------------

   Second Level Bitmap block DBAs
  -------------------------------------------------------------
   DBA 1:     0x02800577
```

From the above "Extent 0: L1 dba: 0x02800576 Data dba: 0x02800579", we can see the first data block's RDBA of this segment is 0x02800579, which is datafile 10, block 1401.

Dump the block 1401 of the datafile 10 to get the original data object id of the table T1:

```
ODU> dump datafile 10 block 1401 header
Block Header:
block type=0x06 (table/index/cluster segment data block)
block format=0x02 (oracle 8 or 9)
block rdba=0x02800579 (file#=10, block#=1401)
scn=0x0000.00285f2b, seq=2, tail=0x5f2b0602
block checksum value=0x0=0, flag=0
Data Block Header Dump:
 Object id on Block? Y
 seg/obj: 0x830b=33547   csc: 0x00.285f21   itc: 3   flg: E   typ: 1 (data)
       brn: 0   bdba: 0x2800576 ver: 0x01

 Itl              Xid                          Uba              Flag   Lck          Scn/Fsc
0x01    0xffff.000.00000000   0x00000000.0000.00   C---      0   scn 0x0000.00285f21
0x02    0x0000.000.00000000   0x00000000.0000.00   ----      0   fsc 0x0000.00000000
0x03    0x0000.000.00000000   0x00000000.0000.00   ----      0   fsc 0x0000.00000000
Data Block Dump:
================
flag=0x0 --------
ntab=1
nrow=16
frre=-1
fsbo=0x32
ffeo=0x145
avsp=0x113
tosp=0x113
```

You can see that the original data object id of T1 is 33547.

7) Use ODU to recover the data：

```
ODU> unload table test.t1 object 33547

Unloading table: T1,object ID: 33547
Unloading segment,storage(Obj#=33547 DataObj#=33547 TS#=11 File#=10 Block#=1400
Cluster=0)
```

8) Online the tablespace test where the T1 belongs to:

```
SQL> alter tablespace test online;
```

9) Use SQL*Loader (sqlldr) to import the recovered data:

```
E:\ODU\data>sqlldr test/test control=TEST_T1.ctl
```

At this point, all the steps to recover the data have been completed. Compare the recovered data and the data before truncate to see whether they are exactly the same:

```
SQL> select * from t2 minus select * from t1;

no rows selected

SQL> select * from t1 minus select * from t2;

no rows selected
```

You can see that the data has been completely restored.

USE ODU TO RECOVER DROPPED TABLE

The Oracle 10g and above version provides a recyclebin function, you can retrieve the dropped table using that function. But there are still many Oracle 8i, Oracle 9i and Oracle 10g databases who do not have or enable recyclebin function or the table is dropped with the purge option, etc. In such a situation, if you want to recover the accidentally dropped table, the conventional method is to use the backup to restore. If you do not have backup or valid backup, then there is no way to recover it. However, the corresponding function is provided by ODU, in the case of no backup or no valid backup, ODU can recover the data from the dropped table.

The following is an example using ODU to recover dropped table:

1) First create a test table：

```
SQL> create table odu_test ( a number,b varchar2(10),c nvarchar2(30),d varchar2(20),e date,f timestamp,g binary_float,h binary_double);

Table created.

SQL> insert into odu_test select rownum,lpad('x',10),'NC 测试' || rownum, 'ZHS 测试'|| rownum,sysdate+dbms_random.value(0,100),systimestamp+dbms_random.value(0,100),rownum+dbms_random.value(0,10000),rownum+dbms_random.value(0,10000) from dba_objects where rownum<=10000;
```

```
10000 rows created.

SQL> commit;

Commit complete.

SQL> create table t1 as select * from odu_test;

Table created.
SQL> drop table odu_test purge;

Table dropped.
```

2) Offline the tablespace where the dropped table belongs to.

When you find your important table is dropped accidentally, you should stop the application immediately, offline the tablespace where the dropped table belongs to or shutdown the database. In this example, table odu_test belongs to tablespace users, so we offline the tablespace users first.

```
SQL> alter tablespace users offline;

Tablespace altered.
```

3) Then use logminer to find the data object id of the dropped table:

```
SQL> select group#,status from v$log;

    GROUP# STATUS
---------- ----------------
         1 INACTIVE
         2 INACTIVE
         3 CURRENT

SQL> col member for a50
SQL> select member from v$logfile where group#=3;

MEMBER
--------------------------------------------------
/u01/app/oracle/oradata/xty/redo03.log

SQL> exec
sys.dbms_logmnr.add_logfile(logfilename=>'/u01/app/oracle/oradata/xty/redo03.log');
```

```
PL/SQL procedure successfully completed.

SQL>                                                                    exec
sys.dbms_logmnr.start_logmnr(options=>sys.dbms_logmnr.dict_from_online_catalog);

PL/SQL procedure successfully completed.

SQL> select scn,timestamp,sql_redo from v$logmnr_contents where operation='DDL' and
sql_redo like '%odu_test%' order by 2 ;

SCN TIMESTAMP SQL_REDO
---------- ------------------- --------------------------------------------------------------------
    681455 2009-05-08 11:20:50 create table odu_test ( a number,b varchar2(10),c
nvarchar2(30),d varc
                                    har2(20),e      date,f      timestamp,g      binary_float,h
binary_double);

    681521 2009-05-08 11:21:17 create table t1 as select * from odu_test;
    681567 2009-05-08 11:21:34 drop table odu_test purge;

SQL>      select      scn,timestamp,sql_redo      from      v$logmnr_contents      where
timestamp=to_date('2009-05-08 11:21:34','yyyy-mm-dd hh24:mi:ss') order by 1;

        SCN   SQL_REDO
----------   --------------------------------------------------------------------
    681566   set transaction read write;
    681567   drop table odu_test purge;
    681569   Unsupported
    681570   Unsupported
    681570
    681570
    681570
    681570   Unsupported
    681570
    681570
    681570
    681570   Unsupported
    681570
    681570
    681570
    681570   Unsupported
    681570
    681570
    681570
```

```
    681570
    681570
    681570
    681570  Unsupported
    681570  Unsupported
    681570
    681570
    681570
    681570  Unsupported
    681570
    681570
    681570
    681570  Unsupported
    681570
    681570
    681570
    681571  Unsupported
    681572
    681572  delete from "SYS"."OBJ$" where "OBJ#" = '52230' and "DATAOBJ#" = '5223
            0' and "OWNER#" = '57' and "NAME" = 'ODU_TEST' and "NAMESPACE" = '1' a
            nd "SUBNAME" IS NULL and "TYPE#" = '2' and "CTIME" = TO_DATE('2009-05-
            08 11:20:46', 'yyyy-mm-dd hh24:mi:ss') and "MTIME" = TO_DATE('2009-05-
            08 11:20:46', 'yyyy-mm-dd hh24:mi:ss') and "STIME" = TO_DATE('2009-05-
            08 11:20:46', 'yyyy-mm-dd hh24:mi:ss') and "STATUS" = '1' and "REMOTEO
            WNER" IS NULL and "LINKNAME" IS NULL and "FLAGS" = '0' and "OID$" IS N
            ULL and "SPARE1″ = '6' and "SPARE2″ = '1' and "SPARE3″ IS NULL and "SP
            ARE4″ IS NULL and "SPARE5″ IS NULL and "SPARE6″ IS NULL and ROWID =
'A
            AAAASAABAAAMzdAAS';

    681572
    681573  commit;
    681574  set transaction read write;
    681574  Unsupported
    681576  commit;
    681577  set transaction read write;
    681579  Unsupported
    681581  commit;

SQL> exec sys.dbms_logmnr.end_logmnr;

PL/SQL procedure successfully completed.
```

From the lines where the SCN is 681572, you can see the dropped table's data object id is

52230 from "delete from "SYS"."OBJ$" where "OBJ#" = '52230' and "DATAOBJ#" = '52230' ".

4) Use ODU to recover the dropped table:

```
[oracle@xty odu]$ ./odu

Oracle Data Unloader:Release 2.6.0

Copyright (c) 2008,2009 XiongJun. All rights reserved.

Web: http://www.laoxiong.net
Email: magic007cn@gmail.com

loading default config…….

 ts#     fn    rfn bsize     blocks bf offset filename
---- ---- ---- ----- -------- -- ------ ----------------------------------------------
    0      1      1   8192       62720 N           0 /u01/oradata/xty/system01.dbf
    1      2      2   8192       26240 N           0 /u01/oradata/xty/undotbs01.dbf
    2      3      3   8192       32000 N           0 /u01/oradata/xty/sysaux01.dbf
    4      4      4   8192         800 N           0 /u01/oradata/xty/users01.dbf
load control file 'control.txt' successful
loading dictionary data……
```

It is assumed that we don't know how many columns of this table and the data type of each column, ODU can automatically determine the column's type by its sampling function:

```
ODU> scan extent tablespace 4;

scanning extent…
scanning extent finished.

ODU> unload object 52230 sample

Unloading Object,object ID: 52230, Cluster: 0
output data is in file : 'data/ODU_ODU_0000052230.txt'

Sample result:
   object id: 52230
   tablespace no: 4
   sampled 1056 rows
   column count: 8
   column       1   type: NUMBER
   column       2   type: VARCHAR2
```

```
  column      3   type: NVARCHAR2
  column      4   type: VARCHAR2
  column      5   type: DATE
  column      6   type: DATE
  column      7   type: BINARY_FLOAT
  column      8   type: BINARY_DOUBLE

COMMAND:
unload object 52230 tablespace 4 column NUMBER VARCHAR2 NVARCHAR2 VARCHAR2
DATE DATE BINARY_FLOAT BINARY_DOUBLE
```

You can see that ODU has been determined the column types more accurately, even NVARCHAR type. Only because of the reasons for the test data, the TIMESTAMP type column which stores as DATE type (only 7 bytes long), so it is judged into DATE type by ODU, but the data here does not affect the recovery process. The content can be seen from the output, you can see the sample data from 'data/ODU_ODU_0000052230.txt', or you can see a more detailed sampling output from 'data/sample.txt'.

Now we can use ODU to recover data:

```
ODU> unload object 52230 tablespace 4 column NUMBER VARCHAR2 NVARCHAR2
VARCHAR2 DATE DATE BINARY_FLOAT BINARY_DOUBLE

Unloading Object,object ID: 52230, Cluster: 0
```

5) Online the tablespace users and import the recovered data.
Modify the generated sql text file 'ODU_ODU_0000052230.sql' to create the table:

```
SQL> CREATE TABLE "TEST"."T2"
  2  (
  3        "C0001" NUMBER ,
  4        "C0002" VARCHAR2(4000) ,
  5        "C0003" NVARCHAR2(2000) ,
  6        "C0004" VARCHAR2(4000) ,
  7        "C0005" DATE ,
  8        "C0006" DATE ,
  9        "C0007" BINARY_FLOAT ,
 10        "C0008" BINARY_DOUBLE
 11  );

Table created.
```

Modify the user and table name in the generated SQL*Loader control file 'ODU_ODU_0000052230.ctl' and use SQL*Loader (sqlldr) to import the recovered data:

```
export NLS_LANG=american_america.zhs16gbk
[oracle@xty data]$ sqlldr test/test control=ODU_ODU_0000052230.ctl

SQL*Loader: Release 10.2.0.4.0 - Production on Fri May 8 12:19:34 2009

Copyright (c) 1982, 2007, Oracle. All rights reserved.

Commit point reached - logical record count 630
Commit point reached - logical record count 1260
Commit point reached - logical record count 1890
Commit point reached - logical record count 2520
Commit point reached - logical record count 3150
Commit point reached - logical record count 3780
Commit point reached - logical record count 4410
Commit point reached - logical record count 5040
Commit point reached - logical record count 5670
Commit point reached - logical record count 6300
Commit point reached - logical record count 6930
Commit point reached - logical record count 7560
Commit point reached - logical record count 8190
Commit point reached - logical record count 8820
Commit point reached - logical record count 9450
Commit point reached - logical record count 10000
```

We find the data matches exactly by comparing data between recovered table T2 and backup table T1. If the data is exported in DMP file format, the data accuracy will not be affected. The data has been completely recovered so far.

CHAPTER 4 – COMMANDS REFERENCE

This chapter describes the commands supported by ODU and detailed description of the ODU commands format and usage.

- ODU using CLI (Command Line Interface) for various operations. ODU supports multiple different command modules, including ODU, BBED and ASMCMD, these three modules are shown in the command prompt as ODU>, BBED> and ASMCMD>. Most data recovery operation executes in the ODU> prompt, while other commands prompt are mainly used for some auxiliary functions.
- The end symbol of ODU's command is enter, rather than the semicolon used by sqlplus.
- ODU's commands are not case sensitive, the only exception is the user name or table name within double quotes when you use the unload command to restore the data, the user name and table name will be strictly matches the text within double quotes, including spaces. For example, unload table sys.t2 means the recovered table's name is T2 under the user SYS, that means if you do not use double quotes then all the user names and table names are converted to uppercase. On the contrary, unload table sys."t2" means the recovered table's name is t2 under the user SYS.
- ODU's command is composed by a command name plus zero or more keywords and parameters. In the command format, the contents in angle brackets indicate the parameters that need to input depends on the actual situation, the contents in square brackets indicate optional keywords and parameters. For example, unload table <schema.tablename> [object truncate] [partition <partition_name>], the <schema.tablename> is the parameter that must input, [object truncate] is the optional keywords, [partiion <partition_name>] is the optional keyword and parameter, the 'partition' is the keyword and the 'partition_name' is the parameter.

UNLOAD

ODU unload command is the most complex command in all ODU commands due to its powerful features and flexibility, but it is also the most critical command to recover data. The command format is as follows:

```
unload dict [block <bootstrap block#>]
unload table <schema.tablename> [object truncate] [partition <partition_name>]
unload table <schema.tablename> [object scanned] [partition <partition_name>]
unload table <schema.tablename> object <data_obj_id> [tablespace <ts_no>]
unload table <schema.tablename> datafile <rfile#> block <block#> [blocks <blocks>] [partition <partition_name>]
unload object <data_obj_id> [tablespace <ts_no>] [cluster <cluster_no>]  column <type [ type [ type.....]>
```

```
    type: VARCHAR2 VARCHAR CHAR NUMBER SKIP LONG RAW
          DATE LONG_RAW TIMESTAMP TIMESTAMP_TZ TIMESTAMP_LTZ
          BINARY_FLOAT BINARY_DOUBLE NVARCHAR2 NCHAR
          CLOB NCLOB BLOB
unload object <data_obj_id> [tablespace <ts_no>] [cluster <cluster_no>]   sample
unload object all [tablespace <ts_no>] sample
unload user <schema name>
```

The data recovered by unload command are stored in the directory which specified by ODU parameter DATA_PATH, the default value of DATA_PATH is the subdirectory data in ODU installation directory.

The data recovered by unload command are divided into two kinds of formats which specified by the value of the ODU parameter OUTPUT_FORMAT, when the value is "DMP", the file format of the recovered data is dmp which is usually generated by traditional Oracle exp tool, the version of the dmp file is Oracle 8.0, higher version of the imp command can import the dmp file which generated by lower version of the exp command. The file format of the recovered data is text file when the value is "TEXT", ODU will generate the necessary sql statements for creating table and control file used for SQL*Loader automatically. The data recovered from each table is saved as a separate file.

There are two cases which are "have data dictionary" and "no data dictionary" when you use unload command to recover data. In the case of "have data dictionary", the recovery data save file is named "<user name>_<table name>", and in case of "no data dictionary", the recovery data save file is named "ODU_<data_object_id>".

Command "unload table" and "unload user" are used for data recovery case which have data dictionary, while the command "unload object" is used for data recovery without data dictionary.

The following are detailed descriptions of these commands:

unload dict

The command "unload dict" is used to parse from the SYSTEM tablespace to get the data dictionary information which is needed by ODU, ODU will save the data dictionary information to user.odu, tab.odu, obj.odu, col.odu, ind.odu, lob.odu, lobfrag.odu and other files. ODU can use these data dictionary information to restore data, ODU can also generate the necessary sql statements for creating table and control file used for SQL*Loader automatically. For the case of "have data dictionary", ODU's recovery functionality can be maximized and can greatly reduce the workload of data recovery operation.

Example of this command is as follows:

```
ODU> unload dict
CLUSTER C_USER# file_no: 1 block_no: 89
TABLE OBJ$ file_no: 1 block_no: 121
```

```
CLUSTER C_OBJ# file_no: 1 block_no: 25
CLUSTER C_OBJ# file_no: 1 block_no: 25
found IND$'s obj# 19
found IND$'s dataobj#:2,ts#:0,file#:1,block#:25,tab#:3
found TABPART$'s obj# 266
found TABPART$'s dataobj#:266,ts#:0,file#:1,block#:2121,tab#:0
found INDPART$'s obj# 271
found INDPART$'s dataobj#:271,ts#:0,file#:1,block#:2161,tab#:0
found TABSUBPART$'s obj# 278
found TABSUBPART$'s dataobj#:278,ts#:0,file#:1,block#:2217,tab#:0
found INDSUBPART$'s obj# 283
found INDSUBPART$'s dataobj#:283,ts#:0,file#:1,block#:2257,tab#:0
found IND$'s obj# 19
found IND$'s dataobj#:2,ts#:0,file#:1,block#:25,tab#:3
found LOB$'s obj# 151
found LOB$'s dataobj#:2,ts#:0,file#:1,block#:25,tab#:6
found LOBFRAG$'s obj# 299
found LOBFRAG$'s dataobj#:299,ts#:0,file#:1,block#:2393,tab#:0
```

It should be noted that the bootstrap$ address which is needed for data dictionary parse is not get from the datafile 1 but from the file which is specified by the first line in ODU control file. Therefore, you should place the first file of the system tablespace (usually system01.dbf) at the first line in ODU control file control.txt. Otherwise you will get the following error:

```
can not get bootstrap$ address from SYSTEM tablespace
```

After you get the data dictionary information, you can use "list user", "list table" and other commands to see the users, tables, views and other objects. For example:

```
ODU> list user

      USER#     USERNAME
----------     ------------------------------
          17     GLOBAL_AQ_USER_ROLE
           6     SELECT_CATALOG_ROLE
          35     EXFSYS
           0     SYS
          11     OUTLN
          19     DIP
          25     ORACLE_OCM
          27     WM_ADMIN_ROLE
          34     JAVA_DEPLOY
          36     XDB
          41     TEST
           2     CONNECT
```

```
          38    XDBADMIN
          12    RECOVERY_CATALOG_OWNER
           3    RESOURCE
           1    PUBLIC
          37    ANONYMOUS
          20    HS_ADMIN_ROLE
          30    JAVASYSPRIV
          22    OEM_ADVISOR
           5    SYSTEM
          10    IMP_FULL_DATABASE
          40    XDBWEBSERVICES
          29    JAVAIDPRIV
          23    OEM_MONITOR
          18    SCHEDULER_ADMIN
           7    EXECUTE_CATALOG_ROLE
          33    JAVA_ADMIN
           4    DBA
          24    DBSNMP
          16    AQ_USER_ROLE
          28    JAVAUSERPRIV
          39    AUTHENTICATEDUSER
          32    EJBCLIENT
          21    TSMSYS
          15    AQ_ADMINISTRATOR_ROLE
          14    LOGSTDBY_ADMINISTRATOR
          42    _NEXT_USER
          13    GATHER_SYSTEM_STATISTICS
          31    JAVADEBUGPRIV
           9    EXP_FULL_DATABASE
          26    WMSYS
           8    DELETE_CATALOG_ROLE

ODU> list table outln

        OBJ#    OBJECT_NAME
----------    ------------------------------
         452    OL$
         453    OL$HINTS
         456    OL$NODES
```

unload table <schema .tablename> [partition <partition_name>]

This command is used to export certain tables under a user, if you specify a partition, we only unload the data from that partition. The keyword partition_name is the partition's name for a

simple partition table, and for composite partitioned tables, partition_name only stands for the sub-partition's name.

Example of this command is as follows:

```
ODU> unload table sys.t1

Unloading table: T1,object ID: 42138 at 2011-04-08 01:08:03
Unloading segment,storage(Obj#=42138 DataObj#=42138 TS#=4 File#=4 Block#=11 Cluster=0)
41161 rows unloaded
At 2011-04-08 01:08:05
```

When exporting table, ODU will display the table name, object ID, segment header information, start time, end time and other information.

ODU's commands are not case sensitive, the only exception is the user name or table name within double quotes when you use the unload command to restore data, the user name and table name will be strictly matches the text within double quotes, including spaces. For example, unload table sys.t2 means the recovered table's name is T2 under the user SYS, that means if you do not use double quotes then all the user names and table names are converted to uppercase. On the contrary, unload table sys."t2" means the recovered table's name is t2 under the user SYS. Here is an example to see the difference:

```
ODU> unload table sys.t2
table 'sys.t2' does not exist.

ODU> unload table sys."t2"

Unloading table: t2,object ID: 42149 at 2011-04-08 01:13:43
Unloading segment,storage(Obj#=42149 DataObj#=42149 TS#=0 File#=1 Block#=43001 Cluster=0)
100 rows unloaded
At 2011-04-08 01:13:43
```

You can see that ODU can not recover the data when using "unload table sys.t2", this is because the table T2 does not exist under the user SYS. The table's real name is t2 (please note the case difference).

unload table <schema.tablename> [object truncate] [partition <partition_name>]

This command has an extra option "object truncate" when compared with the previous

command. You need to execute the other command "scan extent" to get the required extent information before executing it. For example, the table SYS.T3 has been truncated, you need to recover the data from that table:

```
ODU> unload table sys.t3

Unloading table: T3,object ID: 42150 at 2011-04-08 01:28:22
Unloading segment,storage(Obj#=42150 DataObj#=42151 TS#=0 File#=1 Block#=43009 Cluster=0)
0 rows unloaded
At 2011-04-08 01:28:22
```

You can see the number of recovered rows is 0, that means the recovery operation is not successful.

```
ODU> scan extent tablespace 0

scan extent start: 2011-04-08 01:29:11
scanning extent...
scanning extent finished.
scan extent completed: 2011-04-08 01:29:34

ODU> unload table sys.t3 object truncate
Auto mode truncated table.

Unloading table: T3,object ID: 42150 at 2011-04-08 01:29:51
Unloading segment,storage(Obj#=42150 DataObj#=42150 TS#=0 File#=1 Block#=43009 Cluster=0)
5000 rows unloaded
At 2011-04-08 01:29:51
```

And this time we recover the data from the truncated table SYS.T3 successfully, the amount of recovered rows is 5000.

Tips：

When use this command to recover the truncated table, ODU will automatically search the first few data blocks in the first section of the table's segment header and try to determine whether it is the data before truncation, if so, ODU will recover the data from these and subsequent data blocks, otherwise, the table data will be considered partially reused. We need to use the next command to recover in that situation.

unload table <schema.tablename> object <data_obj_id> [tablespace <ts_no>]

The difference between this command and "unload table <schema.tablename> [object truncate] [partition <partition_name>]" is we specify the data object id after the table name, that is used for the discrepancy in data object id and the recorded data object id in data dictionary of that table. Simply put, when the command "unload table <schema.tablename> [object truncate] [partition <partition_name>]" is invalid for a truncated table, use this command instead. Execute "scan extent" before executing this command.

If you use the transport tablespace, the different segments' data object id may be the same in the same database, in which case you need to specify the tablespace number where the recovered table belongs to (please note that this is not the tablespace name but the tablespace number). It is impossible that different tables have the same data object id in the same tablespace.

Example of this command is as follows:

```
ODU> scan extent tablespace 0

scan extent start: 2011-04-08 01:44:12
scanning extent...
scanning extent finished.
scan extent completed: 2011-04-08 01:44:12

ODU> unload table sys.t3 object 42150

Unloading table: T3,object ID: 42150
Unloading segment,storage(Obj#=42150 DataObj#=42150 TS#=0 File#=1 Block#=43009 Cluster=0) at 2011-04-08 01:44:29
5000 rows unloaded
At 2011-04-08 01:44:29
```

Although this command and the command "unload table <schema.tablename> [object truncate] [partition <partition_name>]" can both recover the truncated table, but it is worth noting that, for a partition table, each partition or sub-partition has a data segment, that means each partition or sub-partition has a different data object id, if you use this command to restore the truncated partition table, you need to execute this command towards each partition or sub-partition and the data object id which specified in each command is different.

Tips：

When a table is truncated, the data of that table is not actually deleted, just the high water mark is shrunk to its segment header, the space is recycled and the data object id is increased.

So if you want to use this command to recover a truncated table, the key point it to know the data object id before truncation. If the table is not truncated or moved before, its data object id and object id should be the same, otherwise you need to get the data object id before truncation by flashback query or logminer.

unload table <schema.tablename> [object scanned] [partition <partition_name>]

Both this command and the command "unload table <schema.tablename> [partition <partition_name>]" are used to recover a table or a partition/sub-partition of a partition table. The difference between them is that this command is used for a table data recovery where exists key data block corruption. These key data blocks including the segment head block and extent map block. If these data blocks are damaged, Oracle can not scan all the data of that table properly.
Execute "scan extent" before executing this command.

Example of this command is as follows:
If the table SYS.T3's segment header or extent map block is corrupted and we need to recover this table:

```
ODU> scan extent tablespace 0

scan extent start: 2011-04-08 13:19:29
scanning extent...
scanning extent finished.
scan extent completed: 2011-04-08 13:19:53

ODU> unload table sys.t3 object scanned
Using scanned extent.

Unloading table: T3,object ID: 42158 at 2011-04-08 13:20:09
Unloading segment,storage(Obj#=42158 DataObj#=42158 TS#=0 File#=1 Block#=43009 Cluster=0)
20000 rows unloaded
At 2011-04-08 13:20:10
```

unload table <schema.tablename> datafile <rfile#> block <block#> [blocks <blocks>] [partition <partition_name>]

This command is used to recover the table data from the specified location and the specified

number of blocks, mainly used to recover the data from the specified corrupted data blocks.

datafile <rfile#> block <block#> specify the starting block to restore.
blocks <blocks> specify the number of blocks to be restored, if not specified, we use the default value 1.

For example：

```
ODU> unload table sys.t1 datafile 4 block 651 blocks 10
unload specific block mode.

Unloading table: T1,object ID: 42138 at 2011-04-08 18:01:23
Unloading segment,storage(Obj#=42138 DataObj#=42138 TS#=4 File#=4 Block#=11 Cluster=0)
752 rows unloaded
At 2011-04-08 18:01:23
```

unload object <data_obj_id> [tablespace <ts_no>] [cluster <cluster_no>] column <type [type [type.....]>

Simply put, this command is used for data recovery when there is no data dictionary or the system tablespace is corrupted, or used to recover data from a dropped table.

The column types in this command are as follows:

```
VARCHAR2 VARCHAR CHAR NUMBER SKIP LONG RAW
 DATE LONGRAW TIMESTAMP TIMESTAMPTZ TIMESTAMPLTZ
 BINARY_FLOAT BINARY_DOUBLE NVARCHAR2 NCHAR
 CLOB NCLOB BLOB
```

The keyword "SKIP" means do not restore the data from that column, execute "scan extent" before executing this command.

If you use the transport tablespace, the different segments' data object id may be the same in the same database, in which case you need to specify the tablespace number where the recovered table belongs to (please note that this is not the tablespace name but the tablespace number). It is impossible that different tables have the same data object id in the same tablespace.

The parameter cluster <cluster_no> is used to specify the table number of the cluster for a cluster table recovery.


Tips：
When use this command to recover data, the specified column order should be subject to the column order in the data blocks. The two are the same under normal circumstances, but in two cases there may be inconsistencies, the first is cluster table, the cluster key is always stored in the front of all the columns; the second is a table which have LONG or LONG RAW column, these two types of columns are always stored in the last of all the columns.

Example of this command is as follows:
First we create a test table and insert a row, and we drop that table finally:

```
SQL> create table t4 (a number, b long,c varchar2(100));

Table created.

SQL> insert into t4 values (1,'test long','test varchar2');

1 row created.

SQL> commit;
SQL> select object_name,data_object_id,object_id from dba_objects where owner=user and object_name='T4';
OBJECT_NAME                    DATA_OBJECT_ID    OBJECT_ID
------------------------------ -------------- ----------
T4                                      42168         42168
SQL> drop table t4;
```

We use ODU to recover the data from that table:

```
ODU> scan extent tablespace 0

scan extent start: 2011-04-08 17:24:52
scanning extent...
scanning extent finished.
scan extent completed: 2011-04-08 17:25:15

ODU> unload object 42168 column number varchar2 long

Unloading Object,object ID: 42168,    Cluster: 0 at 2011-04-08 17:26:12
1 rows unloaded
At 2011-04-08 17:26:12

[oracle@xty data]$ cat ODU_0000042168.txt
1|test varchar2|test long
```

Please note that the specified column order when we recover the data from that table is different with the column order when we create it. This is because the LONG column is always adjusted to store in the last of all the columns.


Tips：
You need to know the data object id and column type when use this command to recover a dropped table. Use logminer to excavate the redo log or use the flashback query to query table SYS.OBJ$ for the data object id of the dropped table. The column type can be obtained from the development or test database or from the cold backup of the production database. If not available, you can let ODU to determine the column types automatically by using the following command "unload object sample".

unload object <data_obj_id> [tablespace <ts_no>] [cluster <cluster_no>] sample

The format of this command is close to the previous command, which is mainly used to determine the table column type automatically through the sampling and analysis when there is no data dictionary information (such as the table is dropped accidentally). After getting the column type, then use the previous command to recover the data. This command is an auxiliary command, you need to execute "scan extent" before executing it.

Example of this command is as follows:

```
ODU> unload object 42168 sample

Unloading Object,object ID: 42168,    Cluster: 0
output data is in file :    'data/ODU_0000042168.txt'

Sample result:
   object id: 42168
   tablespace no: 0
   sampled 1 rows
   column count: 3
   column       1    type: NUMBER
   column       2    type: VARCHAR2
   column       3    type: VARCHAR2

COMMAND:
unload object 42168 tablespace 0 column NUMBER VARCHAR2 VARCHAR2
```

From the above output, you can see that this command produces the following results:

- ✧ The sample data of the recovered table is stored in the file ODU_0000042168.txt which is located in the sub-directory data in ODU installation directory. Check the contents of this

file can confirm whether the data is the right thing we need to restore, while also can recognize whether the column type identified automatically by ODU is correct.

- ✧ The obtained sampling data has one row, three columns. We also list these three column types respectively.
- ✧ We list the command which is used to recover the table data.
- ✧ The output shown above also stored in the file sample.txt which is located in the sub-directory data in ODU installation directory (this storage location is specified by the ODU parameter DATA_PATH). The sample.txt also records the first five generated sampling data in each table.

The type of the third column here is the LONG type actually, why ODU produces such a result is due to the length of the third column is less than 4000. If all the data length of the column is less than 4000 bytes, then use the VARCHAR2 type will still be able to recover the data. LONG type can be understood as a longer VARCHAR2 type.

unload object all [tablespace <ts_no>] sample

This command is used to sample all the data in the database to determine the number of columns in all tables and column types automatically in the situation that the system tablespace is missing or corrupted, that is, no data dictionary information. This is an auxiliary command, you need to execute "scan extent" before executing it.
The parameter "tablespace <ts_no>" limits the tablespace to sample.

Example of this command is as follows:

```
ODU> scan extent tablespace 4

scan extent start: 2011-04-08 17:50:25
scanning extent...
scanning extent finished.
scan extent completed: 2011-04-08 17:50:25

ODU> unload object all tablespace 4 sample

Unloading Object,object ID: 42169,    Cluster: 0
output data is in file :    'data/ODU_0000042169.txt'

Sample result:
   object id: 42169
   tablespace no: 4
   sampled 1056 rows
   column count: 13
   column       1    type: VARCHAR2
   column       2    type: VARCHAR2
   column       3    type: RAW
```

```
  column      4   type: NUMBER
  column      5   type: NUMBER
  column      6   type: VARCHAR2
  column      7   type: DATE
  column      8   type: DATE
  column      9   type: VARCHAR2
  column     10   type: VARCHAR2
  column     11   type: VARCHAR2
  column     12   type: VARCHAR2
  column     13   type: VARCHAR2

COMMAND:
unload object 42169 tablespace 4 column VARCHAR2 VARCHAR2 RAW NUMBER NUMBER
VARCHAR2 DATE DATE VARCHAR2 VARCHAR2 VARCHAR2 VARCHAR2 VARCHAR2



Unloading Object,object ID: 42170,    Cluster: 0
output data is in file :    'data/ODU_0000042170.txt'

Sample result:
  object id: 42170
  tablespace no: 4
  sampled 1053 rows
  column count: 13
  column      1   type: VARCHAR2
  column      2   type: VARCHAR2
  column      3   type: RAW
  column      4   type: NUMBER
  column      5   type: NUMBER
  column      6   type: VARCHAR2
  column      7   type: DATE
  column      8   type: DATE
  column      9   type: VARCHAR2
  column     10   type: VARCHAR2
  column     11   type: VARCHAR2
  column     12   type: VARCHAR2
  column     13   type: VARCHAR2

COMMAND:
unload object 42170 tablespace 4 column VARCHAR2 VARCHAR2 RAW NUMBER NUMBER
VARCHAR2 DATE DATE VARCHAR2 VARCHAR2 VARCHAR2 VARCHAR2 VARCHAR2
```

The output of this command is basically the same with the previous command, but the output sampling result will be more than one table.


Tips：
When use this command to automatically determine and find data in the database, the maintenance guy which is very familiar with the system or the developer may need to be involved in the recovery process, because there are a lot of junk data in the database, the real data needs to be identified by the maintenance personnel or the developer.

unload user <schema name>

This command is used to export all the tables under the specified user. It is only valid where the data dictionary information is intact. It simplifies the operation that is needed to export all the tables under the specified user.

HELP

The command "help" displays a list of commands that ODU supports, it also displays a brief description of each supported command:

```
ODU> help

help          ----     get command list
spool         ----     spool information to file
host          ----     enter os terminal
rowid         ----     decode rowid components
rdba          ----     decode RDBA to rfile# and block#
time          ----     convert number to timestamp
exit          ----     exit from odu
load config ----       load config information from file
open          ----     load database filename and asm disk list from file
hexdump       ----     dump file format hex
dump          ----     dump oracle datafile block
unload        ----     unload data
scan extent ----       scan extent
scan disk ----       scan asm disk or any disk or disk partition
list          ----     list schema object,partition,datafile
charset       ----     get or list supported charset name
```

In the execution of a command, if the command format you entered is not correct, you will be prompted the correct command format. For example:

```
ODU> unload sys.t1
unload dict [block <bootstrap block#>]
```

```
unload table <schema.tablename> [object truncate] [partition <partition_name>]
unload table <schema.tablename> [object scanned] [partition <partition_name>]
unload table <schema.tablename> object <data_obj_id> [tablespace <ts_no>]
unload table <schema.tablename> datafile <rfile#> block <block#> [blocks <blocks>] [partition <partition_name>]
unload object <data_obj_id> [tablespace <ts_no>] [cluster <cluster_no>]    column <type [ type [ type.....]>
      type: VARCHAR2 VARCHAR CHAR NUMBER SKIP LONG RAW
            DATE LONG_RAW TIMESTAMP TIMESTAMP_TZ TIMESTAMP_LTZ
            BINARY_FLOAT BINARY_DOUBLE NVARCHAR2 NCHAR
            CLOB NCLOB BLOB
unload object <data_obj_id> [tablespace <ts_no>] [cluster <cluster_no>]    sample
unload object all [tablespace <ts_no>] sample
unload user <schema name>
```

Different modules have different list of supported commands, such as in ASMCMD module, the results of using the command "help" are as follows:

```
ODU> asmcmd

Entering asmcmd module.

ASMCMD> help

help        ----    get command list
spool       ----    spool information to file
host        ----    enter os terminal
rowid       ----    decode rowid components
rdba        ----    decode RDBA to rfile# and block#
time        ----    convert number to timestamp
exit        ----    exit from asmcmd module
dump        ----    dump asm disk block
extract     ----    extract file from asm disk
```

LOAD CONFIG

The command "load config" is used to load the ODU configuration file, the command format is as follows:

```
load config [filename]
```

The default ODU configuration file's name is config.txt.
We will load the ODU configuration file config.txt automatically when you enter the ODU command interface. The later chapter will introduce the usage of this configuration file.

When ODU starts, you can reload the configuration file. After you reload the modified configuration file, the new configurations will take effect immediately.

Example of this command is as follows:

```
ODU> load config
byte_order little
block_size    8192
block_buffers 1024
db_timezone -7
client_timezone 8
asmfile_extract_path      /asmfile
data_path      data
lob_path       /odu/data/lob
charset_name US7ASCII
ncharset_name AL16UTF16
output_format text
lob_storage infile
clob_byte_order big
trace_level 1
delimiter |
unload_deleted no
file_header_offset 0
is_tru64 no
record_row_addr no
convert_clob_charset yes
use_scanned_lob    yes
trim_scanned_blob yes
lob_switch_dir_rows 20000
db_block_checksum yes
db_block_checking yes
rdba_file_bits 10
compatible 10
load config file 'config.txt' successful
```

Tips：

We suggest to execute the command "open" after the execution of the command "load config", otherwise it will affect the automatic recognition of Oracle's datafiles. For example, if the data block size of your database is 16384 (16K), then the ODU will not recognize the datafiles if it use the default data block size configuration 8192 (8K). When you modify the parameter block_size to 16384 and execute the command "load config" to let the modified configuration take effect immediately, you need to execute the command "open" to let the ODU to take advantage of the new configuration automatically.

OPEN

This command is used to open ODU control file and ASM configuration file, these two files are similar to Oracle's control file. ODU control file stores the Oracle database datafile name, data block size, datafile size, tablespace where a datafile belongs to, file number and other important information, while the ASM configuration file stores ASM disk device file path, ASM diskgroup name, AU size and other important information. So the ODU knows the disks or datafiles from which it needs to read data.
The default ODU control file names are control.txt and oductl.dat. The control.txt is used as old ODU control file before getting license, mainly used to provide data source which is needed to generate license. If you use the ODU enterprise version to recover data, we will use oductl.dat which contains the license information as the new ODU control file, the file name is fixed, you can not modify it.

The default ASM disk configuration file is asmdisk.txt.

The open command format is as follows:

```
open [control filename [asmdisk filename]]
```

The extra option "control filename" refers to the ODU control file name, the default value is control.txt when not specified.
The extra option "asmdisk filename" refers to the ASM disk configuration file, the default value is asmdisk.txt when not specified.


Tips：
For simplicity, you only need to use the open command.

For the ODU control file and ASM disk configuration file format, please see the following chapters.

We will load the ODU control files control.txt, oductl.dat and the default ASM disk configuration file asmdisk.txt automatically when you enter the ODU command interface. When ODU starts, you can reload these files if you modify them. You can use this command to let the modified control file or ASM disk configuration file to take effect immediately.

Example of this command is as follows:

```
ODU> open

grp# dsk# bsize ausize disksize diskname          groupname          path
---- ---- ----- ------ -------- -------------- -------------- ------------------------------
```

```
    1          1      4096      1024K              1024   DGDATA_0001              DGDATA
/oradata/asm/disk1.dbf
    1          0      4096      1024K              1024   DGDATA_0000              DGDATA
/oradata/asm/disk2.dbf
    1          2      4096      1024K              1024   DGDATA_0002              DGDATA
/oradata/asm/disk3.dbf

load asm disk file 'asmdisk.txt' successful

 ts#     fn    rfn bsize     blocks bf offset filename
---- ---- ---- ----- -------- -- ------ ------------------------------------------
    0      1      1   8192      44800 N          0 +DGDATA/xty/datafile/system.260.745630773
    1                2                2         8192                25600     N                              0
+DGDATA/xty/datafile/undotbs1.261.745630805
    2      3      3   8192      15360 N          0 +DGDATA/xty/datafile/sysaux.262.745630817
    4      4      4   8192        800 N          0 +DGDATA/xty/datafile/users.264.745630833
load control file 'oductl.dat' successful
```

- For the command "open", you can refer to the command "save control".
- For the problems may occur when you use the command "open", please refer to the following chapter "Trouble Shooting".

LIST

This command lists the objects in the database, including USER, TABLE, VIEW, table partitioning, etc., you can also use this command to list all datafiles used by ODU. The command format is as follows:

```
list user
list <table | view | procedure | function | index | package | sequence> <user_name>
list partition <user_name.table_name>
list datafile
```

For example：

1) List the users in the database:

```
ODU> list user

      USER#     USERNAME
----------     ------------------------------
          17     GLOBAL_AQ_USER_ROLE
           6     SELECT_CATALOG_ROLE
          35     EXFSYS
           0     SYS
```

```
        11    OUTLN
        19    DIP
        25    ORACLE_OCM
        27    WM_ADMIN_ROLE
        34    JAVA_DEPLOY
        36    XDB
        41    TEST
         2    CONNECT
        38    XDBADMIN
        12    RECOVERY_CATALOG_OWNER
         3    RESOURCE
         1    PUBLIC
        37    ANONYMOUS
        20    HS_ADMIN_ROLE
        30    JAVASYSPRIV
        22    OEM_ADVISOR
         5    SYSTEM
        10    IMP_FULL_DATABASE
        40    XDBWEBSERVICES
        29    JAVAIDPRIV
        23    OEM_MONITOR
        18    SCHEDULER_ADMIN
         7    EXECUTE_CATALOG_ROLE
        33    JAVA_ADMIN
         4    DBA
        24    DBSNMP
        16    AQ_USER_ROLE
        28    JAVAUSERPRIV
        39    AUTHENTICATEDUSER
        32    EJBCLIENT
        21    TSMSYS
        15    AQ_ADMINISTRATOR_ROLE
        14    LOGSTDBY_ADMINISTRATOR
        42    _NEXT_USER
        13    GATHER_SYSTEM_STATISTICS
        31    JAVADEBUGPRIV
         9    EXP_FULL_DATABASE
        26    WMSYS
         8    DELETE_CATALOG_ROLE
```

2) List all tables under the specified user TEST:

```
ODU> list table test

      OBJ#     OBJECT_NAME
```

```
----------     ------------------------------
       42252     T1
       42253     T2
```

3) List all datafiles used by ODU:

```
ODU> list datafile

 ts#     fn    rfn bsize     blocks bf offset filename
---- ---- ---- ----- -------- -- ------ ----------------------------------------
    0      1      1   8192      44800 N          0 +DGDATA/xty/datafile/system.260.745630773
    1                2                2         8192                25600     N                              0
+DGDATA/xty/datafile/undotbs1.261.745630805
    2      3      3   8192      16640 N          0 +DGDATA/xty/datafile/sysaux.262.745630817
    4      4      4   8192       1440 N          0 +DGDATA/xty/datafile/users.264.745630833
    6      5      5   8192        100 N          0 +DGDATA/xty/datafile/tbs_test.270.746469239
```

SCAN

This command is used to scan the segments and extents of datafiles. The main purpose of this command is for data recovery when the system tablespace is missing or corrupted. It can also be used to recover table data when the table is truncated or dropped accidentally. The command format is as follows:

```
scan extent [tablespace <ts#> [datafile <rfile#>] ] [object <data_object_id>] [parallel <parallel_degree>]
```

You can scan the specified tablespace (only tablespace number is accepted) or the specified datafile of the tablespace, you can also scan the specified data object id. We will generate ext.odu, lobpage.odu, lobind.odu and segment.txt in ODU installation directory when we finish the scan, the segment.txt contains the useful segment headers that we found during scan.

The extra option "parallel <parallel_degree>" is used to specify the degree of parallel scan. When the database is huge, use a parallel scan will greatly enhance the scanning speed. Please note that the current ODU windows version does not support parallel scanning.

Example of this command is as follows:

```
ODU> scan extent tablespace 4

scan extent start: 2011-04-08 19:03:41
scanning extent...
scanning extent finished.
scan extent completed: 2011-04-08 19:03:41
```

If you try to recover the data under the situation that the system tablespace is missing or corrupted, or you try to recover the table data that the table is truncated or dropped, you need to execute this command before executing other unload command.

ASMCMD

This command is used to enter into the ASMCMD module, after you enter into it, you can use the command "exit" to exit to the ODU main module.

```
ODU> asmcmd

Entering asmcmd module.

ASMCMD> exit

Exiting asmcmd module.
```

In ASMCMD module, you can also use the command "help" to display a list of supported commands you can use in this module.

EXTRACT

This command belongs to the ASMCMD module, used to extract the files in ASM disk group to file system. The command format is as follows:

```
extract asmfile <src filename> to <dst file_name> [force]
```

The extra option "force" means that the extracted file will overwrite the existing file.
Here the keyword "src filename" refers to any file in the ASM disk group. The format is "+ <diskgroup name>.<file number>" or "+ <diskgroup name/path/ filename>".

For example：

```
ASMCMD> extract asmfile +DGDATA.260 to /tmp/260.dbf

starting extract asm file '+DGDATA.260' to '/tmp/260.dbf',file size is 367009792
asm file extract completed.

ASMCMD> extract asmfile +DGDATA/xty/datafile/system.260.745630773 to /tmp/260.dbf

file exists. you can not overwrite it.

ASMCMD> extract asmfile +DGDATA/xty/datafile/system.260.745630773 to /tmp/260.dbf
```

```
force

starting extract asm file '+DGDATA/xty/datafile/system.260.745630773' to '/tmp/260.dbf',file size is 367009792
asm file extract completed.
```

You can use the dbv to check the extracted files:

```
[oracle@xty odu]$ dbv file=/tmp/260.dbf blocksize=8192

DBVERIFY: Release 10.2.0.5.0 - Production on Sat Apr 9 12:59:58 2011

Copyright (c) 1982, 2007, Oracle.    All rights reserved.

DBVERIFY - Verification starting : FILE = /tmp/260.dbf



DBVERIFY - Verification complete

Total Pages Examined             : 44800
Total Pages Processed (Data) : 28974
Total Pages Failing     (Data) : 0
Total Pages Processed (Index): 4639
Total Pages Failing     (Index): 0
Total Pages Processed (Other): 1820
Total Pages Processed (Seg)    : 1
Total Pages Failing     (Seg)    : 0
Total Pages Empty                 : 9367
Total Pages Marked Corrupt     : 0
Total Pages Influx                : 0
Highest block SCN                 : 348723 (0.348723)
```

DUMP

This command simulates the Oracle's command "alter system dump datafile XXX block XXX", used to parse and display the data block format of Oracle. We currently support datafile header block (only partial information), segment header block, table and index block. For other block types, we only display the block header information, this is enough to use in most cases. The future version of ODU will resolve and display the undo segment header and undo data block.

The dump command is used to help analyze the block format, used for the analysis of the block when the Oracle database can not be opened (such as the critical data dictionary object corruption due to the physical or logical corrupt blocks). The dump command format is as

follows:

```
dump datafile <file#> block <block#> [header]
```

The extra option "header" is used to dump block header only.

This command is similar to Oracle's command "alter system dump datafile XXX block XXX", where the datafile number is the absolute file number.
Here are some sample output:

Dump datafile header：

```
ODU> dump datafile 1 block 1
Block Header:
block type=0x0b (file header)
block format=0x02 (oracle 8 or 9)
block rdba=0x00400001 (file#=1, block#=1)
scn=0x0000.00000000, seq=1, tail=0x00000b01
block checksum value=0xcfe5=53221, flag=4
File Header:
Db Id=0xb0f1f85c=2968647772, Db Name=XJ, Root Dba=0x400341
Software vsn=0x9200000, Compatibility Vsn=0x8000000, File Size=0x1f400=128000 Blocks
File Type=0x3 (data file), File Number=1, Block Size=4096
Tablespace #0 - SYSTEM rel_fn:1
get_bootstrap_dba: compat header size:12
bootstrap rdba 0x004002f1 rfile#=1 block#=753
```

Dump segment header：

```
ODU> dump datafile 10 block 45
Block Header:
block type=0x23 (ASSM segment header block)
block format=0x02 (oracle 8 or 9)
block rdba=0x0280002d (file#=10, block#=45)
scn=0x0000.00209d2c, seq=3, tail=0x9d2c2303
block checksum value=0x7247=29255, flag=4
Data Segment Header:
  Extent Control Header
  -------------------------------------------------------------
  Extent Header:: extents: 1    blocks: 5
                        last map: 0x00000000    #maps: 0    offset: 668
       Highwater:: 0x02800030    (rfile#=10,block#=48)
                        ext#: 0    blk#: 5     ext size:5
       #blocks in seg. hdr's freelists: 0
       #blocks below: 2
       mapblk: 0x00000000     offset: 0
```

```
-------------------------------------------------------
Low HighWater Mark :
      Highwater::   0x02800030   ext#: 0         blk#: 5         ext size: 5
#blocks in seg. hdr's freelists: 0
#blocks below: 2
mapblk   0x00000000   offset: 0
Level 1 BMB for High HWM block: 0x0280002b
Level 1 BMB for Low HWM block: 0x0280002b
-------------------------------------------------------
Segment Type: 1 nl2: 1         blksz: 2048     fbsz: 0
L2 Array start offset:   0x00000434
First Level 3 BMB:   0x00000000
L2 Hint for inserts:   0x0280002c
Last Level 1 BMB:   0x0280002b
Last Level 1I BMB:   0x0280002c
Last Level 1II BMB:   0x00000000
    Map Header:: next   0x00000000   #extents: 1      obj#: 31208   flag: 0x220000000
Extent Map
-------------------------------------------------------------
 0x0280002b   length: 5

Auxillary Map
-------------------------------------------------------------
 Extent 0         :   L1 dba:   0x0280002b Data dba:   0x0280002e
-------------------------------------------------------------

 Second Level Bitmap block DBAs
-------------------------------------------------------------
 DBA 1:     0x0280002c
```

Dump data block of ordinary table：

```
ODU> dump datafile 10 block 47
Block Header:
block type=0x06 (table/index/cluster segment data block)
block format=0x02 (oracle 8 or 9)
block rdba=0x0280002f (file#=10, block#=47)
scn=0x0000.00209f21, seq=1, tail=0x9f210601
block checksum value=0xc8d=3213, flag=6
Data Block Header Dump:
 Object id on Block? Y
 seg/obj: 0x79e8=31208   csc: 0x00.209d2c   itc: 2   flg: E   typ: 1 (data)
      brn: 0   bdba: 0x280002b ver: 0x01

 Itl              Xid                  Uba            Flag   Lck        Scn/Fsc
```

```
0x01     0x000a.008.000006a4   0x008018c1.00e0.13   --U-       2   fsc 0x0000.00209f21
0x02     0x0000.000.00000000   0x00000000.0000.00   ----       0   fsc 0x0000.00000000
Data Block Dump:
================
flag=0x0 --------
ntab=1
nrow=2
frre=-1
fsbo=0x16
ffeo=0x6e7
avsp=0x6fc
tosp=0x6fc
0xe:pti[0] nrow=2    offs=0
0x12:pri[0]    offs=0x76d
0x14:pri[1]    offs=0x6e7
Block Rows Dump:
tab 0, row 0, @0x76d
fb: --H-FL-- lb: 0x1    cc: 2
col     0: [    2]    c1 02
col     1: [ 36]    00 54 00 01 01 0c 00 00 00 01 00 00 00 01 00 00 00 00 d9 95 00 10 09 00 00
00 00 00 00 00 00 00 00 00 00
tab 0, row 1, @0x6e7
fb: --H-FL-- lb: 0x1    cc: 2
col     0: [    2]    c1 03
col     1: [ 84]    00 54 00 01 01 0c 00 00 00 01 00 00 00 01 00 00 00 00 d9 96 00 40 05 00 00
00 03 b6 01 a1 00 00 00 00 00 03 02 80 00 a7 02 80 00 a6 02
  80 00 b1 02 80 00 ae 02 80 00 af 02 80 00 b0 02 80 00 b6 02 80 00 b3 02 80 00 b4 02 80 00
b5
```

HEXDUMP

This command is used to display block information in hexadecimal format and to help analyze the block format. The command format of "hexdump" is as follows:

```
hexdump datafile <file#> block <block#> [offset <offset>]
```

The extra option "offset" refers to the offset starting from the block header.
The following is a sample output:

```
ODU> hexdump datafile 10 block 47
                          -0--1--2--3--4--5--6--7--8--9--a--b--c--d--e--f-
0000000000017800     06 02 00 00 2f 00 80 02 21 9f 20 00 00 00 01 06
0000000000017810     8d 0c 00 00 01 00 00 00 e8 79 00 00 2c 9d 20 00
```

```
0000000000017820    00 00 00 00 02 00 32 00 2b 00 80 02 0a 00 08 00
0000000000017830    a4 06 00 00 c1 18 80 00 e0 00 13 00 02 20 00 00
0000000000017840    21 9f 20 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000000017850    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000000017860    00 00 00 00 00 01 02 00 ff ff 16 00 e7 06 fc 06
0000000000017870    fc 06 00 00 02 00 6d 07 e7 06 00 00 a3 00 80 02
0000000000017880    a3 00 80 02 00 00 00 00 00 00 00 00 00 00 00 00
0000000000017890    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000000000178a0    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000000000178b0    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000000000178c0    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000000000178d0    03 00 00 00 00 08 00 00 01 00 00 00 34 04 00 00
00000000000178e0    00 00 00 00 a4 00 80 02 01 00 00 00 a3 00 80 02
00000000000178f0    a4 00 80 02 00 00 00 00 00 00 00 00 00 00 00 00
0000000000017900    00 00 00 00 00 00 00 00 33 00 00 00 a8 01 80 02
0000000000017910    e4 79 00 00 00 00 00 20 a3 00 80 02 05 00 00 00
0000000000017920    ad 00 80 02 05 00 00 00 b2 00 80 02 05 00 00 00
0000000000017930    b7 00 80 02 05 00 00 00 bc 00 80 02 05 00 00 00
0000000000017940    c1 00 80 02 05 00 00 00 c6 00 80 02 05 00 00 00
0000000000017950    cb 00 80 02 05 00 00 00 d0 00 80 02 05 00 00 00
0000000000017960    d5 00 80 02 05 00 00 00 da 00 80 02 05 00 00 00
```

SPOOL

This command is similar to Oracle's sqlplus command "spool", used to store the display output to a file. The command format is as follows:

```
spool <on | off | filename>
```

The keyword "spool on" means to enable the above spool function and the output filename is odu_spool.txt.
The keyword "spool off" means to disable the above spool function.
The keyword "spool <filename>" is used to enable the above spool function and write the display output to a file which is specified by the "<filename>".

CHARSET

This command is used to list the character set supported by ODU. The command format is as follows:

```
charset list
charset name <charset name>
```

```
charset id <charset id> --id must greater than 0
```

The command "charset list" is used to list all the character sets supported by ODU:

```
ODU> charset list
CHARSET_NAME                                CHARSET_ID
----------------------------------- ----------
US7ASCII                                     1
ZHS16GBK                                   852
UTF8                                       871
AL16UTF16                                 2000
ZHS16CGB231280                             850
ISO2022-KR                                9997
ISO2022-CN                                9998
ISO2022-JP                                9999
...omit some output display...
```

The command "charset name" is used to display the character set id by its corresponding character set name:

```
ODU> charset name US7ASCII
CHARSET_NAME                                CHARSET_ID
----------------------------------- ----------
US7ASCII                                     1
```

The command "charset id" is used to display the character set name by its corresponding character set id:

```
ODU> charset id 2000
CHARSET_NAME                                CHARSET_ID
----------------------------------- ----------
AL16UTF16                                 2000
```

START

This command is similar to Oracle's sqlplus command "start", the comands in a file which is specified by the extra keyword "<filename>" can be executed sequentially, the command format is as follows:

```
@<filename>
start <filename>
```

The keyword "start" can be replaced by the symbol "@".
For example:

```
[oracle@xty odu]$ cat unloads.txt
unload table test.t1
unload table test.t2
```

You can execute the above file in a following way:

```
ODU> @unloads.txt
ODU> unload table test.t1

Unloading table: T1,object ID: 42252 at 2011-04-09 13:32:35
Unloading segment,storage(Obj#=42252 DataObj#=42255 TS#=6 File#=20 Block#=11
Cluster=0)
1000 rows unloaded
At 2011-04-09 13:32:35

ODU> unload table test.t2

Unloading table: T2,object ID: 42253 at 2011-04-09 13:32:35
Unloading segment,storage(Obj#=42253 DataObj#=42253 TS#=6 File#=20 Block#=27
Cluster=0)
1000 rows unloaded
At 2011-04-09 13:32:35
```

CHAPTER 5 – CONFIGURATION PARAMETERS REFERENCE

ODU can achieve its flexible, powerful, cross-platform data recovery features through its different configuration parameters. Some important parameters will affect the data recovery operation, and we will introduce the details of these configuration parameters in this chapter.

ODU's default configuration file is config.txt. ODU will load this configuration file automatically when it starts. After you enter into the ODU command interface, you can still use the command "load config [config filename]" to reload this configuration file. The extra option "[config filename]" is optional, if omitted, ODU will load the above default configuration file config.txt.

For ODU software on different platform, the parameters in the configuration file have been set a reasonable initial value for that platform in the ODU installation package. Thus, only a small number or even no parameters need to be set when we want to use ODU to recover data.

The configuration file is a plain text file, each line in this file is only for one parameter. The configuration format is "parameter name" and "parameter value", between them are a number of blank spaces. The parameter names are not case sensitive but path-related parameter values and parameter values related to the character set are case sensitive, the other parameter values are not case sensitive.

The following are default ODU configuration file contents running under Linux x86 platform:

```
byte_order little
block_size    8192
block_buffers 1024
db_timezone -7
client_timezone 8
asmfile_extract_path      /asmfile
data_path      data
lob_path       /odu/data/lob
charset_name US7ASCII
ncharset_name AL16UTF16
output_format text
lob_storage infile
clob_byte_order big
trace_level 1
delimiter |
unload_deleted no
file_header_offset 0
```

```
is_tru64 no
record_row_addr no
convert_clob_charset yes
use_scanned_lob    yes
trim_scanned_blob yes
lob_switch_dir_rows 20000
db_block_checksum yes
db_block_checking yes
rdba_file_bits 10
compatible 10
```

The detailed usage of each configuration parameter is as follows:

BYTE_ORDER

This parameter indicates the database platform byte order. Optional values are "LITTLE" and "BIG", the default is "LITTLE".

This parameter is independent of the platform on which the ODU is running, but only dependent on the platform where to restore the database. For example, if the database is installed on the AIX platform, this value is "BIG", and for database installed on the x86 platform, this value is "LITTLE". ODU can recover data across platforms, for example, you can use the ODU Windows version to recover the Oracle datafiles on AIX by copying these datafiles to Windows host and vice versa. As long as this parameter is properly instruct the database platform, you can achieve the cross-platform recovery purpose.

BLOCK_SIZE

This parameter sets the default block size of datafiles. ODU supports different block size of datafiles in the same database, if you do not specify the block size in the ODU control file (see the ODU control file reference on later section), we use the default value of BLOCK_SIZE. The optional values of this parameter are 2048, 4096, 8192, 16384 and 32768, the default value is 8192.

BLOCK_BUFFERS

This parameter specifies the ODU data block buffer cache number. For the purpose to improve data recovery performance, ODU will cache some types of data blocks, these types of data blocks must have the same size with the parameter BLOCK_SIZE and must be lob index blocks. Cache other types of data blocks has little effect on performance, so do not set this

value too large, the default value of 1024 can satisfy most requirements.

DATA_PATH

DATA_PATH specifies the directory where the recovery data to be restored, if the amount of the recovery data is very large, you can use this parameter to specify a different directory with the ODU default installation path. Please note that the directory which specified by this parameter must already exist, ODU does not automatically create this directory if it does not exist.
You can use the relative path, or you can also use the absolute path. The default value of this parameter is "data", means the recovery data stores in the subdirectory data in ODU installation directory.
You should estimate the required storage space first when you use ODU to recover data. We recommend you to set this parameter to a separate file system with large enough capacity.

LOB_PATH

LOB_PATH specifies the directory where the lob type recovery data to be restored. As the lob data generally consumes large storage space, you can set this parameter to let the lob recovery data to store in a different directory with the ordinary recovery data. The default value of this parameter is empty, indicating that the LOB data and common data are placed in the same directory which is specified by the parameter DATA_PATH. This parameter only works when you set the parameter OUTPUT_FORMAT to "TEXT". If you set the parameter OUTPUT_FORMAT to "DMP", then this parameter will not work. In addition, even you set the parameter OUTPUT_FORMAT to "TEXT", if the parameter LOB_STORAGE is set to "INFILE", then the CLOB type recovery data and the ordinary recovery data are stored in a single file, not stored separately. Please note that if the value of this parameter is not empty, its value should be set to absolute path, otherwise the Oracle SQL*Loader (sqlldr) will not find the lob recovery data when you use it to import the recovery data.

ASMFILE_EXTRACT_PATH

This parameter is used to specify the path when you use the command "extract asmfile" to extract ASM files from ASM diskgroup to file system, the default value of this parameter is empty, indicating that the extracted ASM files and common recovery data are placed in the same directory which is specified by the parameter DATA_PATH.

OUTPUT_FORMAT

OUTPUT_FORMAT specifies the file format of the recovered data, the optional values are "TEXT" and "DMP", when the value is "DMP", the file format of the recovered data is dmp which is usually generated by traditional Oracle exp tool, the version of the dmp file is Oracle 8.0, higher version of the imp command can import the dmp file which generated by lower version of the exp command. The file format of the recovered data is text file when the value is "TEXT", ODU will generate the necessary sql statements for creating table and control file used for SQL*Loader automatically. The data recovered from each table is saved as a separate file, the file format is either a plain text or a dmp. We do not support to restore multiple tables to a single dmp file and vice versa. The default value of this parameter is "TEXT".

LOB_STORAGE

The parameter LOB_STORAGE specifies the storage type of the lob type recovery data and whether to recover the lob data. The optional values of this parameter are "FILE", "INFILE" and "NONE", the default value is "INFILE". "NONE" means the lob data is not restored when you use ODU to recover data. If you set the parameter OUTPUT_FORMAT to "TEXT" and also set this parameter to "INFILE", then the CLOB type recovery data and the ordinary recovery data are stored in the same file, not stored separately. If you set the parameter OUTPUT_FORMAT to "TEXT" and also set this parameter to "FILE", the CLOB type recovery data is stored separately, ODU can identify these files because it stores the CLOB type recovery data file name with the ordinary recovery data. Blob type recovery data is always stored separately, not affected by this parameter. When LOB columns are stored as separate files, each line of each LOB column data will be stored as a single file.

CLOB_BYTE_ORDER

CLOB_BYTE_ORDER specifies the endianness of the CLOB column data. As CLOB column data in data blocks are stored in UNICODE character set, you need to set this parameter correctly to convert the CLOB column data into the client character set data. The Optional values of this parameter are "LITTLE" and "BIG", the default is "BIG". Starting from Oracle 10g, all the CLOB column data are stored in big endian byte order, in Oracle 10g and above, ODU will handle this case automatically. But there may be a case that your Oracle 10g database is upgrade from an Oracle 9i, for such a situation, you still need to set this parameter to the same endianness of your database platform. For Oracle 9i and below, you should also set this parameter to the same endianness of your database platform, for example, if your Oracle 9i database platform is x86, then this parameter should be set to "LITTLE". Please note that this parameter is just like the parameter BYTE_ORDER, independent of the platform on which the

ODU is running, but only dependent on the platform where to restore the database.

CONVERT_CLOB_CHARSET

If the database character set is multi-byte character set, then the CLOB column data in the database will be stored as UNICODE character set, but for a single-byte character set, CLOB column data will be stored with the same single-byte database character set. For such a database, the parameter CONVERT_CLOB_CHARSET should be set to FALSE, means do not convert CLOB column data to the client character set, the default value of this parameter is TRUE.

LOB_SWITCH_DIR_ROWS

If you set the parameter OUTPUT_FORMAT to "TEXT" and also set the parameter LOB_STORAGE to "FILE", the CLOB and BLOB type recovery data are stored separately. If these two parameters are set to the above values and the table data is quite large, then the recovery operation will generate a lot of lob recovery files, which may cause slow performance when operating these files. To avoid this situation, ODU will create subdirectory according to the value of the parameter LOB_SWITCH_DIR_ROWS. If the lob recovery files reaches the upper limit which is set by this parameter, ODU will generate a new subdirectory to store the LOB data. The default value of this parameter is 50000.

CHARSET_NAME AND NCHARSET_NAME

These two parameters are mainly used to identify the character set and national character set of the data to be restored, which is corresponding to the two Oracle database parameters NLS_CHARACTERSET and NLS_NCHAR_CHARACTERSET. When you use ODU to recover data, we will convert the NCHAR, NVARHCAR2 type of column data from the character set specified by NCHARSET_NAME to the character set specified by CHARSET_NAME and we will also convert the CLOB type recovery data to the character set specified by CHARSET_NAME. Incorrect setting will result in the disorder of these types of column data after recovery. The values of these two parameters are consistent with the values of the two Oracle database parameters NLS_CHARACTERSET and NLS_NCHAR_CHARACTERSET, the default value of the parameter CHARSET_NAME is “US7ASCII” and the default value of the parameter NCHARSET_NAME is “AL16UTF16”. You can use the command ” CHARSET LIST” to list all the character sets supported by ODU.

DELIMETER

DELIMITER specifies the separator between columns when the recovery data is in plain text format. The default is "|" (vertical). Since the character recovery data may contain this character, this parameter can be set to one of the other characters do not appear in the recovery data (such as a particular symbol). Delimiter can be specified as a string, not just a single character. For example, you can specify the "||" and so on.

UNLOAD_DELETED

UNLOAD_DELETED specifies whether to recover the rows that have been deleted. The optional values are “YES” and “NO”, the default value is “NO”. This parameter is used to recover the important rows that have been deleted accidentally in the case of flashback query, backup and logminer are all of no effect and the original space of these rows are not reused. As we can not determine the delete time, if you set this parameter to “YES”, we will unload all the rows that have been deleted, whether these rows are deleted accidentally or intentionally. That may lead to ODU exception in some cases. Therefore, this parameter should be set to "NO" in most cases.

COMPATIBLE

This parameter is used to specify the database version. The default value is 10, means the target database that ODU will unload data is Oracle 10g. The valid values for this parameter are consistent with Oracle's major version number, from 7 to 12.

FILE_HEADER_OFFSET

This parameter is used to set the default raw device datafile header offset. ODU supports different raw device datafile header offset within the same database. If you specify the raw device datafile header offset in ODU control file (see later section on the ODU control file reference), we use the value specified by this parameter as the offset value. The default value of this parameter is 0.

When using raw device as database datafiles, raw device header may be used by the operating system, the part used by Oracle is after the above raw device header. The sizes of these raw device headers are different on different operating systems. On AIX systems, the header size of the ordinary logical volume (lv is the raw device) is 4KB, we should set FILE_HEADER_OFFSET to 4096 in this case, and the logical volume on the new Scalable VG is no longer has the header, so the FILE_HEADER_OFFSET should be set to 0. In the HP

TRU64 system, FILE_HEADER_OFFSET should be set to 65536.

DB_BLOCK_CHECKSUM

This parameter is used to set whether the data block checksum value will be verified when the data block is read by ODU. The default value of this parameter is "TRUE".

DB_BLOCK_CHECKING

This parameter is used to set whether the data block will be verified logically when the data block is read by ODU. The default value of this parameter is "TRUE".

RDBA_FILE_BITS

This parameter is used to set the file number bits when the ODU is parsing the data block RDBA (relative data block address). Typically, the RDBA is composed of 10 bits file number and 22 bits data block number, so the default value of this parameter can meet the needs of the most cases. In certain versions of Oracle database and platforms, this value may be different, such as in Oracle 7, this parameter should be set to 8.

USE_SCANNED_LOB

This parameter is used to set whether the ODU will construct the lob index information and then try to recover the lob data when the data dictionary information of the lob recovery data is missing or the lob index is corrupted. The default value of this parameter is "YES".

TRIM_SCANNED_BLOB

This parameter is used to set whether the ODU will remove all the zero data at the final part of the BLOB recovery data when using scanned LOB information to recover BLOB data. Because this time the ODU can not get the exact length of the BLOB column.

CHAPTER6 – CONTROL FILE AND ASM CONFIGURATION FILE REFERENCE

ODU CONTROL FILE

ODU control file is a plain text (TEXT) file used to specify the datafiles that the ODU needs to use during the data recovery operation, the default ODU control file names are control.txt and oductl.dat. The control.txt is used as old ODU control file before getting license, mainly used to provide data source which is needed to generate license. If you use the ODU enterprise version to recover data, we will use oductl.dat which contains the license information as the new ODU control file. Each line of the ODU control file represents a datafile in the database, a line is a comment line if it begins with the symbol #. Each line consists of eight columns, these columns are separated by one or more spaces.

The detailed introductions of these eight columns are as follows:

- **TABLESPACE NUMBER（TS#)**

Used to specify the tablespace number where the datafile belongs to.

- **RELATIVE FILE NUMBER（RELATIVE FILE#)**

Used to specify the relative file number of the datafile.

- **ABSOLUTE FILE NUMBER（FILE ID)**

Used to specify the absolute file number of the datafile.

- **FILE NAME**

Used to specify the file name of the datafile.
For the use of ASM datafiles, you can use the standard ASM datafile name, such as "+DGDATA/xty/datafile/system.260.745630773", or you can use the simplified form, such as "+<DG name>.<DG file number>", so the above file name can be reduced to "+DGDATA.260".


Tips：
ODU does not directly support raw files under Linux, but ODU supports the corresponding actual device files, such as a database using /dev/raw/raw1, the corresponding actual device file is /dev/sdc1, you need to insert /dev/sdc1 to the corresponding column (datafile name) in ODU control file. If the user does not have the access privilege to execute ODU, then you should grant the read permission to that user, or to switch to an authorized user such as the root to run the ODU.

✧ **BLOCK SIZE**

When the datafile block size is inconsistent with the value specified by the parameter BLOCK_SIZE in ODU configuration file, you can fill this column with the actual datafile block size. Mainly used for the datafiles with different data block size are coexisted in the same database.

✧ **WHETHER IT IS BIG FILE（IS BIG FILE？)**

Used to specify whether the datafile is belongs to an Oracle 10g big file tablespace.

✧ **FILE HEAER OFFSET**

When the datafile header offset is inconsistent with the value specified by the parameter FILE_HEADER_OFFSET in ODU configuration file, you can fill this column with the actual datafile header offset. Mainly used for the datafiles with different header offset are coexisted in the same database. Such as the NORMAL VG and SCALABLE VG are coexisted in the same AIX system.

✧ **THE SIZE OF THE DATAFILE IN BLOCKS**

Used to specify the size of the datafile in blocks.

When to fill the ODU control file with the datafile information, it does not have to fill all the columns, if the datafile header is intact, ODU can identify the necessary datafile information from the datafile header automatically. In this case, you just only need to fill the datafile name.

When to fill the ODU control file with a column information, all the other columns information before that column must also be filled. Such as to fill the column FILE HEADER OFFSET, you must also fill the BLOCK SIZE and other columns, but do not have to fill the column BLOCKS.

Usually we only need to fill the data file name, but as mentioned above, the former three columns before the data file name must also be filled, if the datafile header is intact, simply fill the first three columns with zero is enough.

Here are the two ODU control file examples, the first one is the use of ASM datafiles, the second is the ordinary file system datafiles.

EXAMPLE 1：

```
0    0    0    +DGDATA/xty/datafile/system.260.745630773
0    0    0    +DGDATA/xty/datafile/undotbs1.261.745630805
0    0    0    +DGDATA/xty/datafile/sysaux.262.745630817
0    0    0    +DGDATA/xty/datafile/users.264.745630833
```

EXAMPLE 2：

```
0    0    0    D:\ORACLE\ORADATA\XJ\SYSTEM01.DBF
0    0    0    D:\ORACLE\ORADATA\XJ\UNDOTBS01.DBF
```

```
0    0    0    D:\ORACLE\ORADATA\XJ\INDX01.DBF
0    0    0    D:\ORACLE\ORADATA\XJ\TOOLS01.DBF
0    0    0    D:\ORACLE\ORADATA\XJ\USERS01.DBF
0    0    0    D:\ORACLE\ORADATA\XJ\TEST_8K.DBF
0    0    0    D:\ORACLE\ORADATA\XJ\SYSAUX01.DBF
0    0    0    D:\ORACLE\ORADATA\XJ\TT_TEST1.DBF
0    0    0    D:\ORACLE\ORADATA\XJ\TT_TEST2.DBF
0    0    0    D:\ORACLE\ORADATA\XJ\OEM.DBF
```

If the database can be mounted, you can use the following sql to get the necessary information to fill in the ODU control file:

```
SELECT 0,0,0,NAME FROM V$DATAFILE ORDER BY FILE#
```

ASM DISK CONFIGURATION FILE

ODU ASM configuration file is a plain text (TEXT) file used to specify the ASM disks information that the ODU needs to use during the data recovery operation, the default ODU ASM configuration file name is asmdisk.txt. Each line of the ODU ASM configuration file represents an ASM disk, a line is a comment line if it begins with the symbol #. Each line consists of seven columns, these columns are separated by one or more spaces.

The detailed introductions of these seven columns are as follows:

✧ **ASM DISK NUMBER（DISK NO)**

Used to specify the ASM disk number of the ASM disk group, usually the first ASM disk of the ASM disk group has the lowest ASM disk number. ASM disk number in the same ASM disk group can not be repeated.

✧ **ASM DISK PATH（DISK PATH)**

Used to specify the ASM disk path.

Tips：

ODU does not directly support raw disks (or raw partitions) under Linux, but ODU supports the corresponding actual device files, such as a database using /dev/raw/raw1, the corresponding actual device file is /dev/sdc1, you need to insert /dev/sdc1 to the corresponding column (ASM disk path) in ODU ASM disk configuration file. If the user does not have the access privilege to execute ODU, then you should grant the read permission to that user, or to switch to an authorized user such as the root to run the ODU.

The following are the specific contents of an Oracle 11gR2 ASM disk configuration file under Linux 6, the actual device file name corresponding to the raw disk /dev/raw/raw[i] is /dev/sda[i]:

If you use the raw disk name directly, ODU will not recognize the correct disk information:

```
[oracle@bspdev odu]$ cat asmdisk.txt
# disk_no    disk_path           group_name meta_block_size    ausize disk_size header_offset
0 /dev/raw/raw3 DATA 4096 1048576
1 /dev/raw/raw5 DATA 4096 1048576
2 /dev/raw/raw6 DATA 4096 1048576
0 /dev/raw/raw7 RECO 4096 1048576
1 /dev/raw/raw8 RECO 4096 1048576

[oracle@bspdev odu]$ ./odu

Oracle Data Unloader:Release 4.1.3

Copyright (c) 2008,2009,2010,2011 XiongJun. All rights reserved.

Web: http://www.oracleodu.com
Email: magic007cn@gmail.com

loading default config.....

byte_order little
block_size    8192
block_buffers 1024
db_timezone -7
client_timezone 8
asmfile_extract_path      /odu/asmfile
data_path      data
lob_path       /odu/data/lob
charset_name AL32UTF8
ncharset_name AL16UTF16
output_format text
lob_storage infile
clob_byte_order big
trace_level 1
delimiter |
unload_deleted no
file_header_offset 0
is_tru64 no
record_row_addr no
convert_clob_charset yes
use_scanned_lob    yes
trim_scanned_blob yes
lob_switch_dir_rows 20000
```

```
db_block_checksum yes
db_block_checking yes
rdba_file_bits 10
compatible 10
load config file 'config.txt' successful
loading default asm disk file ......

read data error from asm disk '/dev/raw/raw3'.error message:Invalid argument
read data error from asm disk '/dev/raw/raw5'.error message:Invalid argument
read data error from asm disk '/dev/raw/raw6'.error message:Invalid argument
read data error from asm disk '/dev/raw/raw7'.error message:Invalid argument
read data error from asm disk '/dev/raw/raw8'.error message:Invalid argument

grp# dsk# bsize ausize disksize diskname        groupname       path
---- ---- ----- ------ -------- --------------- --------------- ----------------------------------------

load asm disk file 'asmdisk.txt' successful
loading default control file ......

can not found diskgroup for file +DATA/ora11g/datafile/system.256.747310449.
can not found diskgroup for file +DATA/ora11g/datafile/sysaux.257.747310449.
can not found diskgroup for file +DATA/ora11g/datafile/undotbs1.258.747310451.
can not found diskgroup for file +DATA/ora11g/datafile/users.259.747310451.

 ts#     fn    rfn bsize     blocks bf offset filename
---- ---- ---- ----- -------- -- ------ --------------------------------------------
load control file 'oductl.dat' successful
loading dictionary data......done

loading scanned data......done
```

From the above output information we can see that the ODU can not recognize the raw disk directly.

At this point, you need to change the raw disk name /dev/raw/raw[i] to the corresponding actual device file name /dev/sda[i] in ODU ASM disk configuration file. If the user does not have the access privilege to execute ODU, then you should grant the read permission to that user, or to switch to an authorized user such as the root to run the ODU.

```
[oracle@bspdev odu]$ su
Password:

[root@bspdev odu]# cat asmdisk.txt
# disk_no   disk_path          group_name meta_block_size   ausize disk_size header_offset
0 /dev/sda3 DATA 4096 1048576
1 /dev/sda5 DATA 4096 1048576
```

2 **/dev/sda6** DATA 4096 1048576
0 **/dev/sda7** RECO 4096 1048576
1 **/dev/sda8** RECO 4096 1048576

[root@bspdev odu]# ./odu

Oracle Data Unloader:Release 4.1.3

Web: http://www.oracleodu.com
Email: magic007cn@gmail.com

loading default config.....

byte_order little
block_size 8192
block_buffers 1024
db_timezone -7
client_timezone 8
asmfile_extract_path /odu/asmfile
data_path data
lob_path /odu/data/lob
charset_name AL32UTF8
ncharset_name AL16UTF16
output_format text
lob_storage infile
clob_byte_order big
trace_level 1
delimiter |
unload_deleted no
file_header_offset 0
is_tru64 no
record_row_addr no
convert_clob_charset yes
use_scanned_lob yes
trim_scanned_blob yes
lob_switch_dir_rows 20000
db_block_checksum yes
db_block_checking yes
rdba_file_bits 10
compatible 10
load config file 'config.txt' successful
loading default asm disk file .....

```
grp# dsk# bsize ausize disksize diskname          groupname       path
---- ---- ----- ------ -------- --------------- --------------- ------------------------------------------
   1    0  4096  1024K     9000 DATA_0000       DATA            /dev/sda3
   1    1  4096  1024K     9000 DATA_0001       DATA            /dev/sda5
   1    2  4096  1024K     9000 DATA_0002       DATA            /dev/sda6
   2    0  4096  1024K     9000 RECO_0000       RECO            /dev/sda7
   2    1  4096  1024K     7288 RECO_0001       RECO            /dev/sda8

load asm disk file 'asmdisk.txt' successful
loading default control file ......


 ts#    fn   rfn bsize    blocks bf offset filename
---- ---- ---- ----- -------- -- ------ ------------------------------------------
   0    1    1  8192    88320 N         0 +DATA/ora11g/datafile/system.256.747310449
   1    2    2  8192    89600 N         0 +DATA/ora11g/datafile/sysaux.257.747310449
   2              3                3        8192                12160    N                           0
+DATA/ora11g/datafile/undotbs1.258.747310451
   4    4    4  8192      640 N         0 +DATA/ora11g/datafile/users.259.747310451
load control file 'oductl.dat' successful
loading dictionary data......done

loading scanned data......done
```

From the above output information, you can see that the ODU can recognize all the ASM disks information correctly now.

✧ **ASM DISK GROUP NAME（DISK GROUP NAME)**

Used to specify the ASM disk group name.

✧ **ASM METADATA BLOCK SIZE（META BLOCK SIZE)**

Used to specify the size of the ASM metadata block, usually 4KB.

✧ **ALLOCATION UNIT SIZE（AU SIZE)**

Used to specify the allocation unit (AU) size of the ASM disk group, usually 1MB.

✧ **DISK SIZE**

The size of the ASM disk in AU, if the AU size is 1MB and the DISK SIZE is set to 5000, means the size of the ASM disk is 5000MB.

✧ **DISK HEADER OFFSET（HEADER OFFSET)**

Used to specify the starting offset actually used by ASM disk, because the disk device header may be used by the operating system. When to fill the ODU ASM disk configuration file with

the ASM disk information, it does not have to fill all the columns, if the ASM disk header is intact, ODU can identify the necessary ASM disk information from the ASM disk header automatically. In this case, you just only need to fill the ASM disk path.

When to fill the ODU ASM disk configuration file with a column information, all the other columns information before that column must also be filled. Such as to fill the column DISK SIZE, you must also fill the AU SIZE and other columns, but do not have to fill the column HEADER OFFSET.

Usually we only need to fill the DISK PATH, but as mentioned above, the former column before the DISK PATH must also be filled, if the ASM disk header is intact, simply fill the first column with zero is enough.

asmdisk.txt example：

```
# disk#    path              group_name meta_block_size    ausize disk_size header_offset
0    /oradata/asm/disk1.dbf
0    /oradata/asm/disk2.dbf
0    /oradata/asm/disk3.dbf
```

Note: In this example, the above datafiles are used to simulate the ASM disks in a test environment.

CHAPTER7 – TROUBLE SHOOTING

This chapter describes the most frequently encountered problems when using the ODU to recover data.

ODU DOES NOT AUTOMATICALLY RECOGNIZE THE DATAFILE

After entering into the ODU command interface or use the command "open" to open ODU control file, if ODU can not recognize the datafiles information automatically, usually there are four reasons for that problem:

✧ INCORRECT BLOCK_SIZE

ODU will not be able to recognize the datafiles information correctly if you set incorrect value of the parameter BLOCK_SIZE. In that situation, you should change this parameter to a correct value and try again. If you can not determine the value of this parameter, you can try 2048, 4096, 8192, 16384 and 32768 respectively.

✧ INCORRECT BYTE_ORDER

ODU will not be able to recognize the datafiles information correctly if you set incorrect value of the parameter BYTE_ORDER. For HP, AIX and Solaris SPARC platform, this parameter should be set to BIG; for other platforms, this parameter should be set to LITTLE usually. If you can not determine the correct value of this parameter, you can try LITTLE and BIG respectively.

✧ INCORRECT DATAFILE HEADER OFFSET

ODU will not be able to recognize the datafiles information correctly if you set incorrect value of datafile header offset. This will only appear in the database that using raw device for datafiles and will not happen if the datafiles are located in file system. If you can not determine the correct value of the datafile header offset, you can try 0, 4096 and 65536 respectively.

✧ CORRUPT DATAFILE HEADER

If the datafile header is corrupt, the datafiles information can only be determined manually, you have to fill the ODU control file with these manually information.

CLOB DATA IS IN DISORDER

If the CLOB recovery data are in disorder, usually because the parameters associated with the CLOB are not correctly configured, these parameters are CLOB_BYTE_ORDER and CONVERT_CLOB_CHARSET.
CLOB_BYTE_ORDER should be set to a value that is consistent with the endianness of the database platform. That mean the parameter CLOB_BYTE_ORDER should be set to the same value of the ODU configuration parameter BYTE_ORDER. As for the parameter CONVERT_CLOB_CHARSET, If the database character set is single-byte character set, then this parameter should be set to FALSE; if the database character set is multi-byte character set, then this parameter should be set to TRUE.

OTHER PROBLEMS

For the problems not listed in this chapter, please visit the ODU technical support site http://www.oracleodu.com/en/support for support.