

人人都会 Oracle 异常数据恢复

AUL (MyDUL) Oracle 异常数据恢复指南

Ver: 1.0

作者: 楼方鑫

微信: anysql

邮件: anysql(at)126.com,

anysql(at)live.com

目录

最新版本	4
购买许可	4
恢复示例	5
字典信息	7
分配信息	8
字符集信息	9
数据恢复	10
INIT.SQL	15
选项设置	16
缺失 SYSTEM	19
扫描数据文件	20
猜测字段类型	20
执行恢复命令	21
表结构自动匹配	22
多份重复数据	24
恢复数据字典	25

从 2005 年开始, AUL (MyDUL) 已经为全球不同国家及地区的众多客户恢复了数十 TB 计的 Oracle 数据, 从损坏的 Oracle 8, Oracle 8i, Oracle 9i, Oracle 10g, Oracle 11g, Oracle 12c, Oracle 18c, Oracle 19c 及 Oracle ASM 上为客户快速恢复数据. AUL (MyDUL) 可以脱离 Oracle 运行环境, 直接从数据文件中读取记录, 与官方工具 Oracle DUL 具有同等功效并且功能更加丰富. 当你遇到下列极端情况, 并且没有有效备份 (客户有备份动作, 备份不起作用的情况也遇到过) 用来恢复数据时, AUL (MyDUL) 是往往是你最后的机会. 一直坚持 “拯救数据, 帮助客户” 的原则! 在最新版本 AUL 6 中, 可以直接访问 Oracle ASM 来恢复数据, 或从 Oracle ASM 中将数据文件拷贝出来.

针对以下场景, AUL 可以有效地进行数据恢复:

1. 丢失系统表空间。
2. 系统表空间有坏块, 无法启动 Oracle 数据库。
3. 表空间被删除, 但数据文件还在。
4. 表被删除 (Drop) 后, 马上被发现, 释放的空间还没有被其他表重用。
5. 表被截断 (Truncate) 后, 马上被发现, 释放空间未被其他表重用。
6. 一个表空间丢失部份文件或文件中的部份损坏, 导致表无法正常访问。
7. 数据文件头被勒索病毒破坏或加密。
8. Oracle ASM 存储损坏或磁盘损坏。
9. 其他无法正常打开数据库的情况。

AUL (MyDUL) 并不提供免费服务, 没有许可证的情况下最多允许同时打开 10 个数据文件, 并且只能访问文件的前 512MB 内容, 要支持更多的数据文件或更大的数据文件恢复, 你必须获得许可证并在使用前进行注册. 另外一个免费工具 AUL for Oracle ASM (下载) 可以将存放在 Oracle ASM 中的数据文件拷到文件系统, 在 Oracle ASM 损坏或磁盘不可用时, 进行文件级的数据恢复, 在 AUL (MyDUL 6) 中也集成了这个工具的所有功能, 并且免费使用, 最大支持 2028 块盘的 Oracle ASM 存贮。

最新版本

AUL 最新版本为 8.1，增强了对 Oracle 19c 版本的正式支持，从 2005 年第一版开始到现在已经有 15 年历史了。AUL 为命令行工具，启动 AUL 后会看到如下信息。

```
Register Code: FHBR-SFEX-TJXK-WPRL-HUHI
AUL : AnySQL UnLoader(MyDUL) for Oracle 19c and ASM, release 6.6.0
(C) Copyright Lou Fangxin 2005-2020 (AnySQL.net), all rights reserved.
Registered version, you are welcome!
AUL>
```

其中第一行为注册码，需要购买许可时将第一行信息给我；第二行和第三行为软件版本信息；第四行为注册信息，在这台机器上已经注册成功，所以显示的是注册的版本，没有“同时打开 10 个数据文件，并且只能访问文件的前 512MB 内容”的限制。

购买许可

如果没有许可（请体谅一下，做软件研发真是投入很大的事情，Oracle 数据库也是极其复杂的，研发第一版时差点累到吐血），则启动软件后显示如下：

```
Register Code: FHBR-SFEX-TJXK-WPRL-HUHI
AUL : AnySQL UnLoader(MyDUL) for Oracle 19c and ASM, release 6.6.0
(C) Copyright Lou Fangxin 2005-2020 (AnySQL.net), all rights reserved.
Unregistered version, with 512MB data file size limited!
AUL>
```

需要将第一行的 Register Code 给我，就是“FHBR-SFEX-TJXK-WPRL-HUHI”，

这个 Register Code 在同一台机器上会固定不变（不重装系统），因此只需要注册一次，就可以多次使用，属于非常划算的投资，国内不少做数据恢复的同行用的都是 AUL 软件。

在这里生成的许可码为“ABPHDYJ”，启动 AUL 软件，然后输入“SET LICENCE 许可码”命令就会生成许可文件“AULLIC.DAT”，如下图所示：

```
Register Code: FHBR-SFEX-TJXK-WPRL-HUHI
AUL : AnySQL UnLoader(MyDUL) for Oracle 19c and ASM, release 6.6.0
(C) Copyright Lou Fangxin 2005-2020 (AnySQL.net), all rights reserved.
Unregistered version, with 512MB data file size limited!
AUL> SET LICENCE ABPHDYJ
Registered, Elapsed: 484
AUL>
```

在启动 AUL 的目录（当前工作目录）中可以找到许可文件“AULLIC.DAT”，将许可文件拷到 Windows 系统目录（Windows 主机）或“/etc”目录（Linux/Unix 主机），就完成了软件注册。再次启动软件就会显示已注册。如下所示：

```
Register Code: FHBR-SFEX-TJXK-WPRL-HUHI
AUL : AnySQL UnLoader(MyDUL) for Oracle 19c and ASM, release 6.6.0
(C) Copyright Lou Fangxin 2005-2020 (AnySQL.net), all rights reserved.
Registered version, you are welcome!
AUL>
```

接下来就可以大显身手来进行数据恢复了。

恢复示例

在这里假定 SYSTEM 表空间没有被严重破坏，虽然 Oracle 数据库不能起动，但并不影响 AUL 从 SYSTEM 中读取必要的字典信息。首先需要有一个数据文件列表，需要包含系统表空间和用户表空间，不需要临时表空间文件、UNDO 表空间文件、

控制文件、联机日志和归档日志文件。如下所示：

1. SYSTEM01.DBF
2. PDSCI.DBF

用任何文本编辑工具，依次写入文件列表（包含路径，不能有空格），每一行代表一个数据文件，可以使用“#”开头来表示注释。需要将 SYSTEM 表空间文件写在最第一行，因为系统字典信息是扫描获取的，放在第一个可以加快扫描速度。假配置文件名字为“db.txt”，接下来在 AUL 软件中运行“open db.txt”命令，如下所示：

```
Register Code: FHBR-SFEX-TJXK-WPRL-HUHI
AUL : AnySQL UnLoader(MyDUL) for Oracle 19c and ASM, release 6.6.0
(C) Copyright Lou Fangxin 2005-2020 (AnySQL.net), all rights reserved.
Registered version, you are welcome!

AUL> open db.txt
* ts#  rfn ver bsize      blocks      sizemb hsize filename
-----
Y   0   1 a2  8192      35840        280      0 SYSTEM01.DBF
Y   3   61 a2  8192     2576640     20130      0 PDSCI.DBF
AUL>
```

可以看到数据文件被成功打开了，一些关键信息被自动读出来。每个列的含义如下：

1. *, 用来表示是否能自动识别到关键信息，“Y”表能，“N”表示不能。
2. ts#, 表空间编号，如果数据文件文件头破坏，这此值为 0，不影响使用。
3. rfn, 文件编号，数据文件在 Oracle 中的内部编号，非常关键的信息。
4. ver, 数据块格式的版本号，可以见到的值一般有“02”和“a2”。
5. bsize, 数据库大小，这个不是从文件中读取出来的，而是通过“set block_size 大小”来进行设置的，默认的设置 8192，刚好对上了。
AUL 只能处理同一种块大小的数据文件，如果不同的表空间用不同的块

大小，则需要按表空间来进行恢复。

6. blocks, 可以访问的数据块的数量, 也表示了数据文件大小。
7. sizemb, 数据文件的大小, 通过 blocks 换算而来的。
8. hsize, 头部保留空间, 仅用于 AIX 等机器的裸设备场景, 可能操作系统保留了前 4K 空间, 一般情况下都为 0。
9. filename, 数据文件名。

如果看不到这些信息, 则可能不是 Oracle 的数据文件, 或者块大小设错了, 或者数据文件已经完全损坏, 比如从磁盘阵列中恢复出来的文件错位了。

字典信息

字典信息只存在于系统表空中, 这一步最好只打开系统表空间数据文件, 而不要包含用户表空间文件, 以加快速度节约时间。在 AUL 软件中运行“UNLOAD TABLE DICT\$”命令 (命令后面加分号), 如下所示:

```
AUL> open db.txt
* ts# rfn ver bsize blocks sizemb hsize filename
-----
Y 0 1 a2 8192 35840 280 0 SYSTEM01.DBF
AUL> UNLOAD TABLE DICT$;
2020-07-08 20:24:21
2020-07-08 20:24:22
AUL>
```

系统表空间一般不会太大, 这一步会比较快, 这一步会在“AULDICT”目录中生成数据字典信息, 供后续的数据恢复使用。在这一步完成后, 就可以在 AUL 中使用“DESC 用户.表名”来查看表结构了。如下所示:

```
AUL> DESC SYS.PROPS$;
Storage(OBJ#=126 OBJD=126 TS=0 FILE=1 BLOCK=1096 CLUSTER=0)
No. SEQ INT Column Name Type
-----
1 1 1 NAME VARCHAR2(128) NOT NULL
2 2 2 VALUE$ VARCHAR2(4000)
3 3 3 COMMENT$ VARCHAR2(4000)
AUL>
```

如果能看到表结构, 则表示系统字典信息是好的。其中“Storage”这一行显示了目标表的存储信息, 各字段信息如下所示:

1. OBJ#, 对象编号
2. OBJD, 数据编号, 在数据文件中只有此编号和真实数据, 没有结构信息。
因表结构的信息都存放在 SYSTEM 表空间中, 如果没有 SYSTEM 表空间, 能过扫描数据文件, 则只能得到一个编号和真实数据, 无法精确知道数据是属于哪个表, 需要对应用非常熟悉了解的人来进行匹配。
3. TS, 目标表所在的表空间编号。
4. FILE, 目标表第一个数据块 (Segment Header) 所在数据文件编号。
5. BLOCK, 目标表第一个数据块 (Segment Header) 所在数据块位置。
6. CLUSTER, 如果多个表建在 Cluster 上, 则表示在 Cluster 上的位置, 如果为 0 表示此表不是建在 Cluster 上的, Oracle 系统表有很多是基于 Cluster 存放的。

接下来还需要来生成数据分配信息, 假设数据文件是坏的, 需要通过扫描所有数据文件的方式, 来知道不同表的存放信息, 以加速后续的数据恢复。

分配信息

这一步需要打开所有的用户数据文件, 如果要恢复系统表空间上的对象, 或有数据表建在系统表空间上, 也需要包含进来。然后运行“SCAN DATABASE”命令来扫描生成空间分配信息, 这一步需要访问所有的数据块, 会耗时比较长, 需要耐心等待。还好这一步和前面的字典信息, 针对同一个数据库仅需要执行一次, 而不是每次起动 AUL 都需要执行一次。如下所示:

```
AUL> open db.txt
*  ts#  rfn ver bsize      blocks      sizemb  hsize  filename
-----
Y   0   1 a2  8192      35840        280     0  SYSTEM01.DBF
Y   3   61 a2  8192     2576640     20130     0  PDSCI.DBF
AUL> scan database
2020-07-08 20:40:14
2020-07-08 20:45:15
AUL>
```

接下来就可以进行数据恢复了。

字符集信息

接下来需要获取数据库字符集信息，如果系统表空间是好的，可以从“SYS.PROPS\$”表中获取，就是前面我们查看过表结构的那张表，如下所示：

```
AUL> DESC SYS.PROPS$;

Storage(Obj#=126 ObjD=126 TS=0 File=1 Block=1096 Cluster=0)
No. SEQ INT Column Name                               Type
-----
  1   1   1 NAME                                             VARCHAR2(128) NOT NULL
  2   2   2 VALUE$                                           VARCHAR2(4000)
  3   3   3 COMMENT$                                           VARCHAR2(4000)

AUL>
```

接下来运行“UNLOAD TABLE SYS.PROPS\$”命令，就可以看到这个表的内容了，不同字段之间默认使用竖线分隔。如下所示：

```
AUL> UNLOAD TABLE SYS.PROPS$;
2020-07-08 20:49:23
Unload ObjD=126 File=1 Block=1096 Cluster=0 ...
DICT.BASE|2|dictionary base tables version #
DEFAULT_TEMP_TABLESPACE|TEMP|Name of default temporary tablespace
DEFAULT_PERMANENT_TABLESPACE|SYSTEM|Name of default permanent tablespace
DEFAULT_EDITION|ORA$BASE|Name of the database default edition
Flashback Timestamp TimeZone|GMT|Flashback timestamp created in GMT
TDE_MASTER_KEY_ID
DBTIMEZONE|-05:00|DB time zone
DEFAULT_TBS_TYPE|SMALLFILE|Default tablespace type
GLOBAL_DB_NAME|JSZY|Global database name
NLS_RDBMS_VERSION|12.1.0.2.0|RDBMS version for NLS parameters
NLS_NCHAR_CHARACTERSET|AL16UTF16|NCHAR Character set
NLS_NCHAR_CONV_EXCP|FALSE|NLS conversion exception
NLS_LENGTH_SEMANTICS|BYTE|NLS length semantics
NLS_COMP|BINARY|NLS comparison
NLS_DUAL_CURRENCY|$|Dual currency symbol
NLS_TIMESTAMP_TZ_FORMAT|DD-MON-RR HH.MI.SSXXFF AM TZR|Timestamp with time zone
NLS_TIME_TZ_FORMAT|HH.MI.SSXXFF AM TZR|Time with time zone format
NLS_TIMESTAMP_FORMAT|DD-MON-RR HH.MI.SSXXFF AM|Time stamp format
NLS_TIME_FORMAT|HH.MI.SSXXFF AM|Time format
NLS_SORT|BINARY|Linguistic definition
NLS_DATE_LANGUAGE|AMERICAN|Date language
NLS_DATE_FORMAT|DD-MON-RR|Date format
NLS_CALENDAR|GREGORIAN|Calendar system
NLS_CHARACTERSET|ZHS16GBK|Character set
```

找到第一列为“NLS_CHARACTERSET”的那一行，表示了数据库的字符集，在这里是“ZHS16GBK”；接下来找到“NLS_NCHAR_CHARACTERSET”开头的一行，表示了数据库的NCHAR字符集，在这里是“AL16UTF16”。这两个信息需要记住，每次重新启动 AUL，都需要重新设置一次，如下所示：

```
AUL> SET CHARSET ZHS16GBK
Current CHARSET is : 0x0354 (852)
AUL> SET NLSCHARSET AL16UTF16
Current NLSCHARSET is : 0x07d0 (2000)
AUL>
```

AUL 会将字符集的名字自动转化为字符集的内部编号，内部集成了 Oracle 19c 所支持的所有字符集，应当是非常齐全了。

数据恢复

在前面获取字符集信息时，已经相当于恢复了一个表了，就是使用“UNLOAD TABLE”命令来进行数据恢复。我们来恢复一下“SYSTEM”用户下的“HELP”表看看，只需要运行“UNLOAD TABLE SYSTEM.HELP TO HELP.TXT”命令，如下所示：

```
AUL> UNLOAD TABLE SYSTEM.HELP TO HELP.TXT;
2020-07-08 21:06:57
Unload OBJD=20369 FILE=1 BLOCK=18936 CLUSTER=0 ...
Sucessfully unload 938 rows ...
2020-07-08 21:06:57
AUL>
```

可以看到成功恢复了 938 条记录。默认的恢复格式是文本方式，将数据恢复成格式化文本文件，然后同步生成建表的 SQL 语句文件和用于 SQL*Loader 工具装载的控制文件，分别为：

1. HELP.TXT，数据文件
2. HELP_syntax.sql，基本建表语句，无分区、索引、约束等信息。

3. HELP_sqlldrctl, 用于 SQL*Loader 工具装载的控制文件。

有了这三个文件, 可以很方便地进行数据恢复。可以在数据文件中找到以下信息:

```
ARCHIVE LOG|1
ARCHIVE LOG|2| ARCHIVE LOG
ARCHIVE LOG|3| -----
ARCHIVE LOG|4
ARCHIVE LOG|5| Displays information about redo log files.
ARCHIVE LOG|6
ARCHIVE LOG|7| ARCHIVE LOG LIST
ARCHIVE LOG|8
```

可以在建表语句文件中看到以下内容:

```
CREATE TABLE "HELP" (
  "TOPIC" VARCHAR2(50) NOT NULL ,
  "SEQ" NUMBER NOT NULL ,
  "INFO" VARCHAR2(80)
);
exit;
```

可以在 SQL*Loader 工具装载的控制文件看到以下内容:

```
--
-- Generated by AUL/MyDUL, for table SYSTEM.HELP
--
OPTIONS(BINDSIZE=8388608, READSIZE=8388608, ERRORS=2147483647, ROWS=50000)
LOAD DATA
INFILE 'HELP.TXT' "STR X'0d0a'"
APPEND INTO TABLE HELP
FIELDS TERMINATED BY X'7c' TRAILING NULLCOLS
(
  TOPIC    CHAR(50) ,
  SEQ      CHAR ,
  INFO     CHAR(80)
)
```

接下来只需要运行 SQL*Loader 工具来进行数据恢复(请自行在目标用户下创建表结构), 在操作系统下(需要在机器上安装 Oracle 客户端及工具, 或者将

文件拷到 Oracle 服务器上运行，需要注意设置准确的字符集环境变量）运行命令“sqlldr user/password control=HELP_sqlldr.ct1”。先在“SCOTT”用户下创建“HELP”表，如下所示：

```
SQL> CREATE TABLE "HELP" (  
2  "TOPIC"  VARCHAR2(50) NOT NULL ,  
3  "SEQ"    NUMBER NOT NULL ,  
4  "INFO"   VARCHAR2(80)  
5  );  
表已创建。  
SQL> DESC HELP  
名称                是否为空? 类型  
-----  
TOPIC                NOT NULL  VARCHAR2(50)  
SEQ                  NOT NULL  NUMBER  
INFO                 VARCHAR2(80)  
SQL>
```

然后使用 SQL*Loader 工具来装载数据，如下所示：

```
D:\BaiduNetdiskDownload\jszy>sqlldr scott/tiger control=HELP_sqlldr.ct1  
SQL*Loader: Release 19.0.0.0.0 - Production on 星期三 7月 8 22:12:27 2020  
Version 19.3.0.0.0  
Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved.  
所用路径:          常规  
达到提交点 - 逻辑记录计数 938  
表 HELP:  
  已成功载入 938 行。  
查看日志文件:  
  HELP_sqlldr.log  
了解有关加载的详细信息。
```

可以看到全部 938 条记录装载成功，可以到 SQL*Plus 中去查一下记录数，如下所示：

```
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production  
Version 19.3.0.0.0  
SQL> select count(*) from help;  
  
COUNT(*)  
-----  
          938
```

可以在数据库中查到一致的记录数，说明数据恢复完全成功了。

将数据恢复成文本格式时，由于真实数据千变成化，需要注意设置准确的字

段与字段之间的分隔符号（“FIELD_TAG”）以及记录与记录之间的分隔符号（“RECORD_TAG”）。字段分隔符默认为竖线（“|”），记录分隔符默认为换行符，当表中数据本身会包含竖线或换行符时，就需要显式设置分隔符号了。可以设置任何不会出现在数据库的字符或字符串，或使有用“\xXX”格式来设置任何字符，如下所示：

```
AUL> set field_tag <field>
  Current FIELD_TAG :<field>
AUL> set record_tag <row>\x0a
  Current RECORD_TAG :<row>\n
AUL> set field_tag \x07
  Current FIELD_TAG :\x07
AUL> set record_tag \x06
  Current RECORD_TAG :\x06
AUL>
```

例如：

```
AUL> open db.txt
* ts# rfn ver bsize blocks sizemb hsize filename
-----
Y 0 1 a2 8192 35840 280 0 SYSTEM01.DBF
Y 3 61 a2 8192 2576640 20130 0 PDSCI.DBF
AUL> set field_tag <field>
  Current FIELD_TAG :<field>
AUL> set record_tag <row>\x0a
  Current RECORD_TAG :<row>\n
AUL> unload table sys.props$ limit 10;
2020-07-09 05:55:58
Unload OBJD=126 FILE=1 BLOCK=1096 CLUSTER=0 ...
DICT.BASE<field>2<field>dictionary base tables version #<row>
DEFAULT_TEMP_TABLESPACE<field>TEMP<field>Name of default temporary tablespace<ro
DEFAULT_PERMANENT_TABLESPACE<field>SYSTEM<field>Name of default permanent tables
DEFAULT_EDITION<field>ORA$BASE<field>Name of the database default edition<row>
Flashback Timestamp TimeZone<field>GMT<field>Flashback timestamp created in GMT<
TDE_MASTER_KEY_ID<row>
DBTIMEZONE<field>-05:00<field>DB time zone<row>
DEFAULT_TBS_TYPE<field>SMALLFILE<field>Default tablespace type<row>
GLOBAL_DB_NAME<field>JSZY<field>Global database name<row>
NLS_RDBMS_VERSION<field>12.1.0.2.0<field>RDBMS version for NLS parameters<row>
Successfully unload 10 rows ...
2020-07-09 05:55:58
```

可以看到恢复出的数据中，字段及记录分隔符不再是默认的竖线和换行了，对应生成的 SQL*Loader 控制文件也会相应变化，不影响数据载装。当然也可以将数据恢复成 Oracle Dump 格式（简称“DMP 格式”），就不需要这么小心地设置

字段与记录分隔符了。可以在“UNLOAD”命令之前执行“SET OUTPUT_STYLE {DMP | TXT}”来切换文格格式或 DMP 格式，如果输出的文件扩展名为“DMP”，则会自
动临时切换为 DMP 格式。如下所示：

```
AUL> open db.txt
*  ts#  rfn ver bsize      blocks    sizemb  hsize  filename
-----
Y   0   1 a2  8192      35840      280    0  SYSTEM01.DBF
Y   3   61 a2  8192     2576640    20130    0  PDSCI.DBF
AUL> SET CHARSET ZHS16GBK
Current CHARSET is : 0x0354 (852)
AUL> SET NLSCHARSET AL16UTF16
Current NLSCHARSET is : 0x07d0 (2000)
AUL> SET OUTPUT_STYLE DMP
Current OUTPUT_STYLE is : DMP
AUL> UNLOAD TABLE SYSTEM.HELP TO HELP.dmp;
2020-07-09 06:04:08
Unload OBJD=20369 FILE=1 BLOCK=18936 CLUSTER=0 ...
Sucessfully unload 938 rows ...
2020-07-09 06:04:08
AUL>
```

接下来使用 Import 工具进行导入，如下所示产：

```
D:\BaiduNetdiskDownload\jszy>imp system/oracle file=HELP.dmp fromuser=system touser=scott ignore=y
Import: Release 19.0.0.0.0 - Production on 星期四 7月 9 06:08:21 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved.

连接到: Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

经由常规路由由 EXPORT:V08.01.07 创建的导出文件
已经完成 ZHS16GBK 字符集和 AL16UTF16 NCHAR 字符集中的导入
IMP-00403:

警告: 此导入生成了单独的 SQL 文件 "import_sys", 其中包含了由于权限问题而失败的 DDL。
. 正在将 SYSTEM 的对象导入到 SCOTT
. 正在导入表 "HELP" 导入了 938 行
成功终止导入, 但出现警告。
```

可以看到已经成功导入 938 行记录,和前面文本方式的导入记录数完全一致。DMP 格式的生成需要完整的数据字典信息，即要求 SYSTEM 表空间相对 AUL 是完好的，比如一个月之前的系统表空间文件备份，对 Oracle 来讲可能时间差距过大，但对 AUL 来讲信息也是完整的（不包含最近一个月内创建或重建的表）。

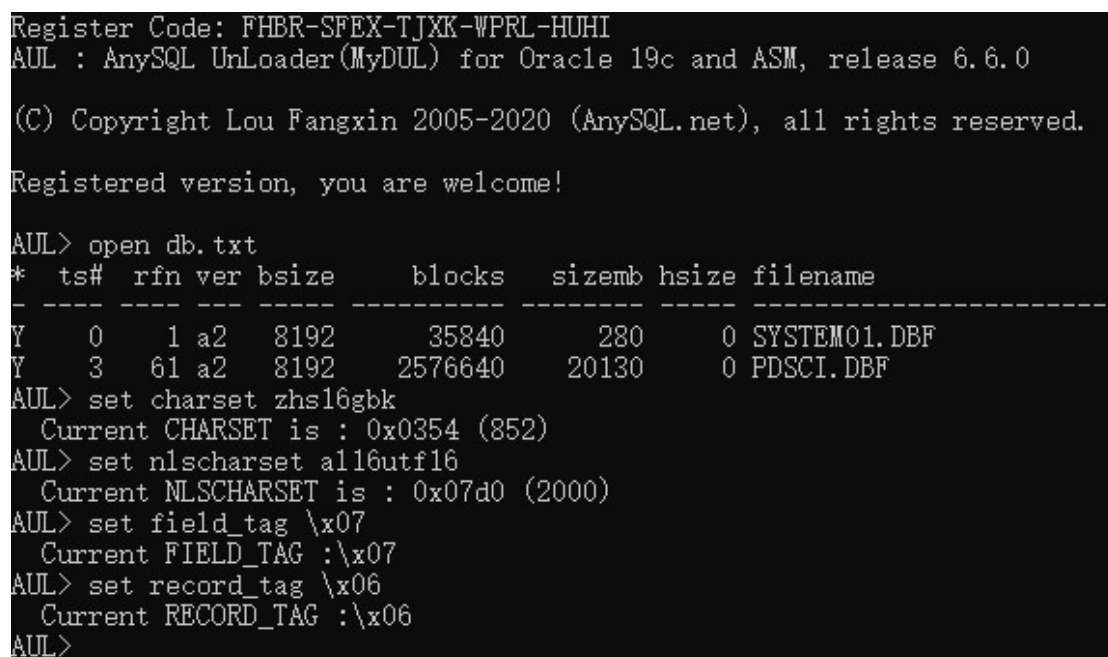
INIT.SQL

从前面的恢复步骤里可以看到，启动 AUL 后通常需要打开数据文件，并运行一些设置命令，然后才可以开始工作。为方便起见，可以将这些命令放到当前目录的“init.sql”文件中，好让 AUL 在启动时默认运行这些命令，为数据恢复做好准备，如下所示：



```
init - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
open db.txt
set charset zhs16gbk
set nlscharset all6utf16
set field_tag \x07
set record_tag \x06
|
```

接下来启动 AUL，就可以看到这些命令被自动执行了，如下所示：



```
Register Code: FHBR-SFEX-TJXK-WPRL-HUHI
AUL : AnySQL UnLoader(MyDUL) for Oracle 19c and ASM, release 6.6.0
(C) Copyright Lou Fangxin 2005-2020 (AnySQL.net), all rights reserved.
Registered version, you are welcome!

AUL> open db.txt
* ts#  rfn ver bsize      blocks   sizemb  hsize  filename
-----
Y    0   1 a2  8192      35840    280    0  SYSTEM01.DBF
Y    3   61 a2  8192    2576640   20130    0  PDSCI.DBF
AUL> set charset zhs16gbk
Current CHARSET is : 0x0354 (852)
AUL> set nlscharset all6utf16
Current NLSCHARSET is : 0x07d0 (2000)
AUL> set field_tag \x07
Current FIELD_TAG : \x07
AUL> set record_tag \x06
Current RECORD_TAG : \x06
AUL>
```

接下来只需要直接运行数据恢复的主要命令“UNLOAD TABLE …”就行了。

选项设置

AUL 中的“SET”命令用来进行选项设置，比较常用的选项有如下所示：

- BLOCK_SIZE
设定数据库大小，可选值为 2048、4096、8192、16384、32768，暂时不支持其他的块大小。AUL 同时只能处理一种块大小，如果一个数据库不同的表空间使用了不同的块大小，则按单个表空间恢复来进行处理。
- BYTE_ORDER
CPU 的字节对齐，一个 32 位整数有 4 个字节，在写到内存或存储时，可以先写高位字节（BIG）或先写低位字节（LITTLE）。一般 Intel CPU 是低位优先，而传统 UNIX 或安腾上则是高位优先。
- FIELD_TAG
字段分隔符，在导出成文档方式时，字段与字段之间会有此分隔符。需要选择不会出现在字段值中的字符串，可以用“\xXX”格式来设置表示任何字符。
- RECORD_TAG
记录分隔符，在导出成文本方式时，记录与记录之间会有此分隔符。需要选择不会出现在字段值中的字符串，可以用“\xXX”格式来设置表示任何字符。
- CHARSET
数据库基本字符集，内部用一整型数值表示，在 SET 时可以使用字符集名字。可以使用“LIST CHARSET”命令来查看所有支持的字符集。


```
AUL> list charset
US7ASCII           = 1 | WE8DEC           = 2
WE8EBCDIC37       = 5 | WE8EBCDIC500    = 6
WE8EBCDIC1146     = 9 | WE8PC850        = 10
S7DEC             = 13 | E7DEC           = 14
I7DEC            = 17 | NL7DEC          = 18
SF7DEC           = 21 | TR7DEC          = 22
```

- NLSCHARSET

数据库扩展字符集，内部用一整型数值表示，在 SET 时可以使用字符集名字。可以使用“LIST CHARSET”命令来查看所有支持的字符集。

```
AUL> list charset
US7ASCII           = 1 | WE8DEC           = 2
WE8EBCDIC37       = 5 | WE8EBCDIC500    = 6
WE8EBCDIC1146     = 9 | WE8PC850        = 10
S7DEC             = 13 | E7DEC           = 14
I7DEC            = 17 | NL7DEC          = 18
SF7DEC           = 21 | TR7DEC          = 22
```

- BIGFILE

是否为 BIGFILE 模式的表空间，默认情况下块地址的前 10 位表示文件号，但在 BIGFILE 模式的表空间中，整个表空间只有一个文件，所有位都用来表示块的地址。

- OUTPUT_STYLE

设置恢复文件的数据格式，支持将数据恢复成文本格式和 DMP 格式。此选项目前已经不再需要了，只需要将目标文件改成以“dmp”为后缀，则自动切换为 DMP 格式，否则为文本格式。

- VERBOSE

调试级别以输出更多的信息，比如数据库损坏时，有些块可能会导致 AUL 程序异常中止，这时需要调高 VERBOSE 的值，可以在处理数据块前打印出正在处理的数据块地址，以便后续处理。

```

AUL> set verbose 1
      Current VERBOSE is : 1
AUL> unload table sys.props$ to sys_props.dmp;
2020-08-26 06:48:04
Unload OBJD=126 FILE=1 BLOCK=1096 CLUSTER=0 ...
Recover rows from RDBA=4195401 ...
Sucessfully unload 38 rows ...
2020-08-26 06:48:04

```

AUL 中还有一些其他的选项设置，最基本的是以上几个。完整的 SET 选项如下所示：

```

AUL> SET
SET BLOCK_SIZE      {2048 | 4096 | 8192 | 16384 | 32768}
SET BYTE_ORDER      {BIG | LITTLE}
SET DELETED_ROW     {TRUE | FALSE}
SET COMMITTED_ONLY  {TRUE | FALSE}
SET FIELD_TAG       field_tag
SET RECORD_TAG      record_tag
SET CACHE_SIZE      kbytes (64 - 8192)
SET OUTPUT_STYLE    {TXT | DMP}
SET CHARSET         charsetid
SET NLSCHARSET      charsetid
SET FIXED_CHARSET   {true | false}
SET BLOCK_CHECK     {0 | 1}
SET HEAD_SIZE       header size (default 0)
SET VERBOSE         {0 | 1}
SET ROW_ADDRESS     {0 | 1}
SET CLOB_EDIAN      {BIG | LITTLE}
SET LOB_CONVERT     {0:NONE | 1:GBK | 2:UTF8}
SET LOB_STORAGE     {0:INLINE | 1:FILE | 2:NONE}
SET MAXLOBDIR       values between 100 and 2000
SET MAXCHAINS       integer value
SET BIGFILE         {Yes | NO}
SET ICONV_NCHAR     from_iconv_charset to_iconv_charset
SET ICONV_NCLOB     from_iconv_charset to_iconv_charset
SET ICONV_CLOB      from_iconv_charset to_iconv_charset
AUL>

```

使用不带值的 SET 命令可以查看当前的设置值，例如：

```

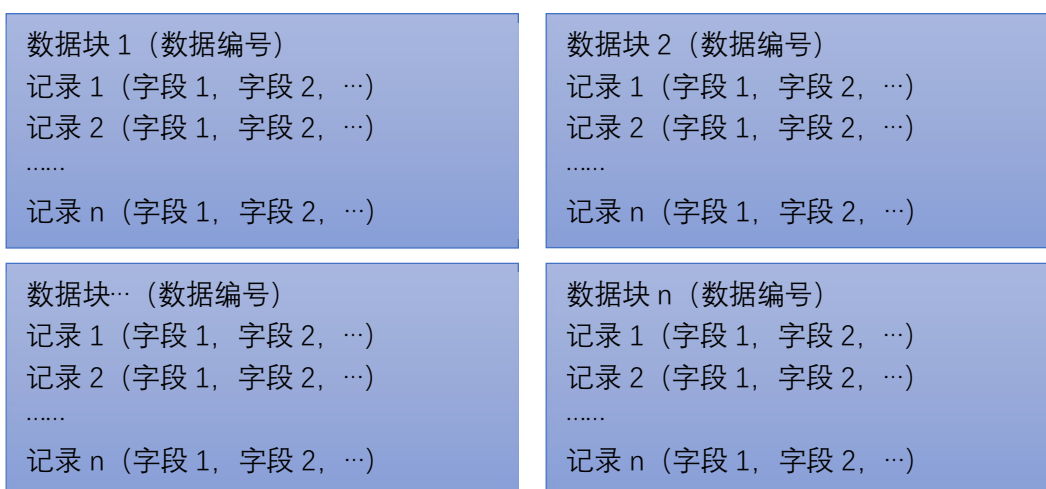
AUL> SET BLOCK_SIZE
      Current BLOCK_SIZE is : 8192
AUL> SET BIGFILE
      Current BIGFILE is : NO
AUL> SET BYTE_ORDER
      Current BYTE_ORDER is : LITTLE
AUL>

```

其中带 LOB 字样的选项和恢复 LOB 类型数据有关，会在 LOB 恢复中单独进行介绍。其他一些非常用选项，可以在用到时进行咨询。

缺失 SYSTEM

通常我们谈论数据时，会包含业务记录、表结构、视图定义、存储过程、触发器等等，但从数据恢复技术的角度出发，则只有记录。在 SYSTEM 表空间中有很多预定义格式（用户无法修改）的系统表，这些系统表描绘了用户表（指包含业务数据的用户创建的表）的结构。当丢失 SYSTEM 表空间时，在基他数据文件的数据块中，只能看到一个数字编号和按字段排的真实数据，如下图所示：



在没有 SYSTEM 的情况下做数据恢复，需要经过以下几个步骤：

- 第一步：扫描所有数据文件，分析数据文件中的数据分布情况，比如数据文件中有多少个不同的数据编号，以及每个数据编号的数据分布情况。
- 第二步：针对每个不同的数据编号，进行字段类型猜测，生成 UNLOAD 命令，生成的 UNLOAD 命令中列的类型（猜测不准）可能有误。
- 第三步：运行 UNLOAD 命令恢复数据，每个不同的数据编号会生成一个独立的文件，猜测的数据类型可能有误，需要检查字段类型的准确性。
- 第四步：对生成的文件进行分析，确定数据属于哪个表，建立数据编号和业务表的对应关系。
- 第五步：将恢复出来的数据导入到合适的业务表中，以完成数据恢复。

下面我们用用户表空间(包含一个数据文件: users01.dbf)做恢复例子, 来详细讲解在没有 SYSTEM 的情况下如何将数据恢复出来。

扫描数据文件

参照前面的步骤将“users01.dbf”文件添加到配置文件中, 并在 AUL 中打开配置文件, 然后运行“SCAN DATABASE”命令来完成数据文件扫描。如下图所示:

```
Register Code: FHBR-SFEX-TJXK-WPRL-HUHI
AUL : AnySQL UnLoader(MyDUL) for Oracle 19c and ASM, release 8.1.0
(C) Copyright Lou Fangxin 2005-2020 (AnySQL.net), all rights reserved.
Registered version, you are welcome!

AUL> open db.txt
*  ts#  rfn ver bsize      blocks      sizemb hsize filename
-----
Y   4   7 a2  8192         640           5      0 users01.DBF
AUL> scan database
2020-12-29 21:43:55
2020-12-29 21:43:58
```

扫描所需要的时间要看数据文件的大小、恢复所用机器的硬件条件, 如果 CPU 及 IO 不是问题, 大约一分钟能够扫描 4-6GB 的样子。这一步只需要做一次, 除非配置文件中的数据文件有增减。

猜测字段类型

在这一步只需要运行“SCAN TABLE”命令, 就会自动猜测各个列的数据类型, 并为你生成一个恢复数据用的 UNLOAD 命令。如下图所示:

```
AUL> scan table
2020-12-29 21:44:03
## Object = 63385.0, Columns = 4, Sample = 17, Blocks = 5, Size = 0.04 mb
UNLOAD OBJECT 63385 CLUSTER 0 COLUMN
    VARCHAR VARCHAR VARCHAR VARCHAR TO OBJD0000063385C000.txt;

## Object = 63387.0, Columns = 1, Sample = 4, Blocks = 5, Size = 0.04 mb
UNLOAD OBJECT 63387 CLUSTER 0 COLUMN
    VARCHAR TO OBJD0000063387C000.txt;

## Object = 63389.0, Columns = 2, Sample = 8, Blocks = 5, Size = 0.04 mb
UNLOAD OBJECT 63389 CLUSTER 0 COLUMN
    VARCHAR NUMBER TO OBJD0000063389C000.txt;
```

也可以在“SCAN TABLE”命令后面添加“TO 文件名”选项来输出到一个文件。如下图所示:

```
AUL> SCAN TABLE TO scan_table.log
2020-12-29 21:54:34
2020-12-29 21:54:34
AUL>
```

然后用文本编辑工具打开“scan_table.log”文件，就可以进行数据恢复了。

执行恢复命令

命令“SCAN TABLE”的输出中，“UNLOAD”字样开头的多行命令就是用来恢复数据的完整命令，从命令也可以看出没有字段名，只有字段类型和编号。下面我们来运行一个“UNLOAD”命令：

```
AUL> SCAN TABLE TO scan_table.log
2020-12-29 21:54:34
2020-12-29 21:54:34
AUL> UNLOAD OBJECT 63385 CLUSTER 0 COLUMN
      2      VARCHAR VARCHAR VARCHAR VARCHAR;
2020-12-29 21:58:41
BMPF |Y|Y|Microsoft Windows Bitmap
CAL5 |Y|Y|CAL5 Raster
FPIX |Y|N|Flashpix
GIF5 |Y|Y|Graphics Interchange Format
JFIF |Y|Y|JPEG File Interchange Format
PBMF |Y|Y|Portable Bitmap
PCXF |Y|N|PC Paintbrush File Format
```

从这个例子来看，字段类型猜的结果是完全准确的。我们再来看一个有NUMBER字段的数据的恢复。如下图所示：

```
AUL> UNLOAD OBJECT 63398 CLUSTER 0 COLUMN
      2      NUMBER VARCHAR VARCHAR NUMBER;
2020-12-29 22:00:48
21 |ordcmsd.xml|STANDARD_DICTIONARY|1
22 |ordcmpv.xml|PRIVATE_DICTIONARY|2
23 |ordcmp.xml|MAPPING|3
24 |ordcman.xml|ANONYMITY|4
25 |ordcmpf.xml|PREFERENCE|5
26 |ordcmui.xml|UID_DEFINITION|6
27 |ordcmcmc.xml|CONSTRAINT|7
27 |ordcmcmd.xml|CONSTRAINT|8
27 |ordcmct.xml|CONSTRAINT|9
Sucessfully unload 9 rows ...
2020-12-29 22:00:48
```

接下来就需要对数据十分熟悉的人员来进行数据和表的匹配工作了，后面我们还会讲更高效的方式，可以借用表结构信息，然后让AUL作自动匹配，来减轻表结构的匹配工作。

表结构自动匹配

如果数据文件有成百上千个表，匹配表结构会是一个十分复杂的事情。假设你还有重建表结构的 SQL 语句，或者你可以从其他数据库实例中获得同样的表结构，则可以借用这些表结构信息。首先需要用“UNLOAD TABLE DICT\$”命令来生成 AUL 可识别的表结构信息。如下所示：

```
AUL> open db.txt
*  ts#  rfn ver bsize      blocks      sizemb hsize filename
-----
Y   0   1 a2  8192      120320       940    0 system01.DBF
AUL> unload table dict$;
2020-12-29 22:09:41
2020-12-29 22:09:43
```

然后打开要恢复的数据文件，在运行“SCAN TABLE”命令前先使用“LOAD TABLE”命令将要恢复的表的结构信息装载到 AUL 中，再运行“SCAN TABLE”命令，就会自动进行表结构的匹配。如下图所示：

```
AUL> open db.txt
*  ts#  rfn ver bsize      blocks      sizemb hsize filename
-----
Y   6   5 a2  8192      2621440     20480    0 DEEPBLUE.DBF
AUL> load table dp;
Loaded Guess Table DP.ACRILLNESS
Loaded Guess Table DP.ACRPART
Loaded Guess Table DP.BACKUPMEDIAINFO
Loaded Guess Table DP.BOOKING_RECORDE
Loaded Guess Table DP.BOOKING_RULE
Loaded Guess Table DP.COSTVIEW
```

接下来运行“SCAN TABLE”命令就会得到不一样的结果。如下图所示：

```
AUL> SCAN TABLE
2020-12-29 22:14:12
## Object = 87145.0, Columns = 3, Sample = 8192, Blocks = 43, Size = 0.34 mb
## Guess : DP.ACRILLNESS , Match Count = 3/3, Score = 100%
## UNLOAD TABLE DP.ACRILLNESS OBJECT 87145 TO ACRILLNESS_87145.TXT;
UNLOAD OBJECT 87145 CLUSTER 0 COLUMN
    VARCHAR VARCHAR VARCHAR TO OBJD0000087145C000.txt;

## Object = 87147.0, Columns = 2, Sample = 732, Blocks = 5, Size = 0.04 mb
## Guess : DP.ACRPART , Match Count = 2/2, Score = 100%
## UNLOAD TABLE DP.ACRPART OBJECT 87147 TO ACRPART_87147.TXT;
UNLOAD OBJECT 87147 CLUSTER 0 COLUMN
    VARCHAR VARCHAR TO OBJD0000087147C000.txt;
```

可以看到输出中多了两行，多出的第一行显示的表名匹配的猜测结果，多出的第二列给出了使用表结构信息进行恢复的命令。“Match Count”表示多少个字段匹配上了，“Score”则表示有值的字段的匹配比例。当然这个表名的自动匹配

也不一定完全准确，如果你的每个表的表结构都很有特征，相互之前都不相同，则这个匹配程度还是很高的。要同时满足以下三个条件：

- 匹配的字段数最多。
- 尽力匹配字段数最少的表。
- 满足表结构 NOT NULL 限制。

“SCAN TABLE” 输出中各个字段的含义，如下所示：

- Object：数据编号+表号
- Columns：扫描出来的字段数，也是有数据的字段数
- Sample：用来分析字段类型的样本大小（记录数），样本大小为 8192
- Blocks：数据所占的数据块的数量，用来估算表的大小
- Size：根据块数乘以数据块大小进行计算的表大小，单位为兆
- Guess：自动匹配的表名
- Match Count：匹配的表名/表结构的字段数
- Score：匹配的字段数占有明确值的字段数的比重，最高为 100%

可以看到，在表结构自动匹配的功能下，缺失系统表空间的数据恢复在 AUL 中也不是一件非常困难的事情。但表结构匹配的结果仍然做不到 100%准确，在 Oracle 存储数据时，中间的 NULL 值（最后一个非空列之前的 NULL 值）只有占位标志可被识别成一个字段，但无法判断值的类型，而后面的 NULL 值则是连占位标识也没有，连字段数信息也丢失了。比如一个表定义有 10 个列，但只有前面两个列有值，后面 8 个列都为空值的话，在扫描这个表的数据块时，发现只有两个列，因此扫描结果中“Match Count”的两个值可能相差很大。如下图所示：

```
## Object = 87253.0, Columns = 25, Sample = 20000, Blocks = 494, Size = 3.86 mb
## Guess : DP.REQUISITION , Match Count = 14/26, Score = 100%
## UNLOAD TABLE DP.REQUISITION OBJECT 87253 TO REQUISITION_87253.TXT;
UNLOAD OBJECT 87253 CLUSTER 0 COLUMN
  VARCHAR VARCHAR DATE NUMBER UNKNOWN VARCHAR VARCHAR DATE VARCHAR UNKNOWN
  VARCHAR NUMBER VARCHAR VARCHAR VARCHAR UNKNOWN UNKNOWN UNKNOWN UNKNOWN UNKNOWN
  UNKNOWN UNKNOWN UNKNOWN UNKNOWN NUMBER
  TO OBJD0000087253C000.txt; |
```

在这里扫描出来的字段个数有 25 个（见“Columns”字段），其中有值的有 14 个（见“Match Count”的第一个数值），而匹配的表的字段数则有 26 个（见“Match Count”的第二个值），而匹配度（见“Score”字段）则是匹配的字段数与有值的字段数的比重，没有值的字段无法做任何比较。

多份重复数据

也会发现多份不同的数据会匹配到同一个表的情况，如下图所示：

```
## Object = 99457.1, Columns = 24, Sample = 2193, Blocks = 43, Size = 0.34 mb
## Guess : DP.PACS_STUDY , Match Count = 17/25, Score = 100%
## UNLOAD TABLE DP.PACS_STUDY OBJECT 99457 TO PACS_STUDY_99457.TXT;
UNLOAD OBJECT 99457 CLUSTER 1 COLUMN
    VARCHAR VARCHAR VARCHAR VARCHAR VARCHAR VARCHAR VARCHAR VARCHAR VARCHAR UNKNOWN
    DATE VARCHAR UNKNOWN NUMBER UNKNOWN UNKNOWN NUMBER DATE VARCHAR NUMBER
    UNKNOWN UNKNOWN UNKNOWN NUMBER TO OBJD0000099457C001.txt;

## Object = 99277.1, Columns = 24, Sample = 4860, Blocks = 96, Size = 0.75 mb
## Guess : DP.PACS_STUDY , Match Count = 17/25, Score = 100%
## UNLOAD TABLE DP.PACS_STUDY OBJECT 99277 TO PACS_STUDY_99277.TXT;
UNLOAD OBJECT 99277 CLUSTER 1 COLUMN
    VARCHAR VARCHAR VARCHAR VARCHAR VARCHAR VARCHAR VARCHAR VARCHAR VARCHAR UNKNOWN
    DATE VARCHAR UNKNOWN NUMBER UNKNOWN UNKNOWN NUMBER DATE VARCHAR NUMBER
    UNKNOWN UNKNOWN UNKNOWN NUMBER TO OBJD0000099277C001.txt;

## Object = 98962.1, Columns = 24, Sample = 462, Blocks = 10, Size = 0.08 mb
## Guess : DP.PACS_STUDY , Match Count = 17/25, Score = 100%
## UNLOAD TABLE DP.PACS_STUDY OBJECT 98962 TO PACS_STUDY_98962.TXT;
UNLOAD OBJECT 98962 CLUSTER 1 COLUMN
    VARCHAR VARCHAR VARCHAR VARCHAR VARCHAR VARCHAR VARCHAR VARCHAR VARCHAR UNKNOWN
    DATE VARCHAR UNKNOWN NUMBER UNKNOWN UNKNOWN NUMBER DATE VARCHAR NUMBER
    UNKNOWN UNKNOWN UNKNOWN NUMBER TO OBJD0000098962C001.txt;
```

出现这个情况可能有很多种原因，第一种是使用了分区表的原因，分区表的每一个分区在数据文件中都有独立的数据编号，第二种是表移动（MOVE）、截断（TRUNCATE）、删除重建（DROP）后留下的影子数据，为了加快这些操作，Oracle并不会将原有的数据区域全部清零，而是做了删除标志或修改了空间使用信息，这也是AUL能够恢复DROP/TRUNCATE操作的底层原理（需要这些操作后释放的空间还未被重复使用）。

出现这种情况时，仍然需要非常懂业务和数据的人来验证自动匹配表结构的准确性，以及数据的准确性。

恢复数据字典

Oracle 数据库的表结构、索引信息、函数、存储过程、Package、视图、Sequence 等对象的定义都放在 SYSTEM 表空间的众多系统表中。当丢失 SYSTEM 表空间文件时，这些信息随之丢失，只能恢复数据。当 SYSTEM 表空间文件未丢失时，可以恢复上述对象的定义，但过程比较复杂，需要经过以下几步：

- 导出关键的系统表
- 导入关键的系统表
- 写 SQL 从系统表中生成对象定义
- 在新数据库中创建对象

由于不同版本的系统表定义会有些差别，系统视图的定义最好从对应版本的库上去获得，可以方便从恢复的系统表中用 SQL 来重新生成对象创建脚本。

常见系统表

对恢复对象比较有用的系统表可以从常见的系统视图（DBA_/ALL_）的定义中去找，找到这些系统视图的定义后，找出里面的基础表。比较常见的有：

- USER\$: 用户表
- OBJ\$: 对象表
- SOURCE\$: 代码表 (Type、Function、Procedure、Package、Trigger 等)
- VIEW\$: 视图表
- SEQ\$: Sequence 表
- IND\$: 索引表
- INDCOL\$: 索引列定义表

导出系统表

```
UNLOAD TABLE SYS.USER$ AS SYS_USER TO SYS_USER.dmp;
UNLOAD TABLE SYS.OBJ$ AS SYS_OBJ TO SYS_OBJ.dmp;
UNLOAD TABLE SYS.SOURCE$ AS SYS_SOURCE TO SYS_SOURCE.dmp;
UNLOAD TABLE SYS.VIEW$ AS SYS_VIEW TO SYS_VIEW.dmp;
UNLOAD TABLE SYS.SEQ$ AS SYS_SEQ TO SYS_SEQ.dmp;
```

导入系统表

导出存储过程、函数、Package

```
user=gppls/gppls
record=
file=sys_source_all.txt
query=select
  decode(s.line, 1, '||chr(10)||'create or replace ', '') || s.source line
from
  sys_user u, sys_obj o, sys_source s
where
  s.obj# = o.obj# and u.user# = o.owner# and
  u.user# > 26 and u.name = 'DBUSRSYS'
order by o.name, o.type#, s.line
```

导出视图

```
user=gppls/gppls
field=
record=0x0d0x0a
file=VIEW_DBUSRSYS.txt
query=select
  'create or replace ' sql_cmd,
  o.name, ' as ', v.text, ';'
from
  sys_user u, sys_obj o, sys_view v
where
  v.obj# = o.obj# and u.user# = o.owner# and
  u.user# > 26 and u.name = 'DBUSRSYS'
order by o.name, o.type#
```

导出 SEQUENCE

```
user=gppls/gppls
```

```
field=
record=0x0d0x0a
file=SEQ_DBUSRPUBINT.txt
query=select
  'CREATE SEQUENCE ' sql_cmd,
  o.name,
  ' START WITH ', s.highwater,
  ' MINVALUE ', s.minvalue,
  ' MAXVALUE ', s.maxvalue,
  ' INCREMENT BY ', s.INCREMENT$,
  decode(s.cycle#, 1, ' CYCLE ', ''),
  decode(s.order$, 1, ' ORDER ', ''),
  ' CACHE ', s.cache, ';'
from
  sys_user   u, sys_obj   o,
  sys_seq    s
where
  s.obj# = o.obj# and u.user# = o.owner# and
  u.user# > 26 and u.name = 'DBUSRPUBINT'
order by o.name, o.type#
```