Data recovery UnLoader

Lunar

www.lunar2013.com

QQ: 5163721

内容介绍

DUL的开发

DUL的用途

DUL的发展

案例分析

参考信息

总结, 提问

DISCLAIMER

- DUL 是荷兰人Bernard van Duijnen用C写的，作者是荷兰的Oracle support
- DUL 不是 Oracle product,也不是Oracle支持的产品
- DUL 只被（Oracle Support等） internal use
- DUL 程序中暴露了Oracle的源代码，其使用受到严格的控制
- DUL v3开始支持export模式或者写一个Pro*c来加载LONG RAW数据

Life without DUL

常规恢复：

- **restore and rollforward**
- **export/import**
- **SQL*Loader 重新加载数据**
- **CTAS 和 PCTS(parallel create table as select)**
- **Transportable Tablespace**

Life without DUL

- 诊断工具
  - **orapatch**
  - **BBED (block browser/editor)**

- **Undocumented 参数**
  - **_corrupted_rollback_segments, _allow_resetlogs_corruption etc...**

使用限制

- **SYSTEM表空间(数据文件)损坏，且别无选择的概况下**
- **数据库不可能恢复，甚至使用了 undocumented 参数!)**
- **打patch是非常“麻烦”的，并且不保证能解决问题**
- **某些损坏是无法补救的（truncate, drop）**
- **底线 – 数据丢失!!**

其他的方法

- 克隆数据库，并导出(exp)数据，然后导入(imp)到需要恢复的数据库中

- 克隆一个数据库，然后使用传输表空间来恢复

解决方法

DUL 是可能的解决方案

- **DUL (?) - Bernard 说 ‘Life is DUL without it!’**
- **底线 – 尽可能拯救更多的数据**

警告 – 最后的恢复手段

- **DUL**是任何其他常规方法都不可用时的最后选择
- 尽量使用 **restore/rollforward, EXP, SQL*Plus** 等方法.
- 任何常规方法都不能满足要求的情况下， **DUL**是最后的恢复手段

注意: **DUL**不进行逻辑校验（脏读），不保证数据库的一致性

为什么用 DUL?

- 不需要数据库打开或者实例启动
- 不需要做 **recovery, archive logs** 等
- 不关心数据的一致性
  - 因此可以适应更多的数据损坏的情况**(truncate, drop**等等**)**
- 可以不需要 **SYTEM tablespace**

DUL 概览

DUL是一个从“严重损坏”的数据库中 unload数据 的工具

DUL可以扫描数据库文件，重组表头，获取extent的信息

DUL可以创建一个SQL*Loader 或者Export 格式的输出文件

- 相应的 SQL*Loader的 control file 也被自动生成

Overview - DUL will...

如果 **SYSTEM** 表空间文件可用，读数据字典分析所有的行

- **Columns的数量, column 数据类型， column 长度**

Overview - DUL

- DUL 所有的数据行类型
  - normal rows, migrated rows, chained rows, multiple extents, clustered tables等等.
- DUL执行, 只需要最小的（必要的）人工干预
- 支持跨平台的 unloading

DUL 概念

- 直接从Oracle 数据文件中恢复数据
  - 绕开数据库(RDBMS)的
- 脏读,这意味着它认为所有的transaction都是已经提交的
- 不检查是否做过介质恢复
- 数据库损坏，块损坏都可以
- 支持LMT(Locally Managed Tablespaces)
- 最新的DUL 10.2支持ASM

兼容性

- **Cross-unloading:** 数据文件可以被复制到其他主机做 **unloading**
- 支持所有的数据库结构:
  - **row chaining, row migration, hash/index clusters, longs, raws, rowids, dates, numbers, multiple free list groups, segment high water mark, NULLS, trailing NULL columns etc...**
- **DUL** 的版本 **6 , 7, 8 and 9, 10,** 分别使用于**ORACLE**不同版本的数据库

DUL支持什么？

- 支持行链接和行迁移
- 支持哈希和索引cluster
- 支持NULL列
- 支持LONG、RAW,DATE,NUMBER,ROWID
- 支持多free list group
- 支持多字节字符集
- 支持LOB类型列，不过需要注意的是，有LOB类型列的表在做DUL时输出需要为SQL*Loader格式。
- 从10.2.1.16开始支持ASM

DuL 92和Dul 10

- fix for problem with startup when db_block_size = 16K
- fix for scan database and Automatic Space Managed Segments
- fix for block type errors high block types; new max is 51
- Support for Automatic Space Managed Segments
- phase zero of new unformat command
- internal preparation for varray support
- Bug fix in the stricter checking of negative numbers
- Bug fix in the unloading of clustered tables
- 从10.2.1.16开始支持ASM

限制

数据库可以损坏，但是其中每个独立的数据块必须是完好的。

- **Unload期间会检查块，以确保他们是完好的且属于正确的segment**

DUL 只能unload表或者cluster 数据

- **它不能导出 triggers, constraints, stored procedures ，也不能生成建表或者view的语句**
  - **但数据字典表是可以unload的**
  - **如果dul在unload时遇到坏块，则会有一行提示信息显示出来，unload不会因此终止，而是继续进行**

限制

- 对多字节字符集的数据没有专门的支持

- **DUL 可以unload (LONG) RAWs, 但不能有效的使用SQL*Loader 导入数据，一般建议使用export模式**
  - **SQL*Loader 不能加载 LONG RAW**

FAQ

DUL 和大文件 (files > 2GB)

- 从DUL 8.0.6.7开始提示 是否可以支持32-bit i/o(不支持大文件)或者 64-bit i/o (支持大文件).

DUL 对裸设备的支持

- DUL 支持裸设备. 但是DUL关心是否为裸设备.
- 对于早期的raw device，DUL不能自动跳过头块（OS block），但是可以配置offset来跳过头块（比如早期的AIX 头块为4k，tru64为64k等等）
- DUL不会使用file header中的size，因此，DUL会读整个裸设备（包括文件尾部的空块）

配置 DUL

DUL需要两个配置文件

- **init.dul**
- **control.dul**

配置参数和平台相关

init.dul

- 包含了和数据文件格式相关的参数
- 包含如下信息：
  - **DUL cache size**
  - 块头的部署
  - **Oracle block** 大小
  - 输出文件的格式
  - **Sql*Loader** 格式和记录的大小.
  - 等等...

init.dul 的例子

```
# sample init.dul configuration parameters
# these must be big enough for the database in question
# the cache must hold all entries from the dollar tables.
dc_columns = 200000
dc_tables = 10000
dc_objects = 10000
dc_users = 40

# OS specific parameters
osd_big_endian_flag = false
osd_dba_file_bits = 6
osd_c_struct_alignment = 32
osd_file_leader_size = 1

# database parameters
db_block_size = 8192

# loader format definitions
LDR_ENCLOSE_CHAR = "
LDR_PHYS_REC_SIZE = 81

#ADD PARAMETERS
export_mode=true  # still needed with dul9
compatible=9
```

control.dul

用来转换文件号和文件名

- 第一列是file_number，
- 第二列是文件名（全路径和文件名）
- 第三列是可选的，表示偏移量，所有相关数据文件的fseek()操作都需要加上这个偏移量.

control.dul的例子1

1 /test/dul/lunar/system01.dbf

2 /test/dul/lunar/sysaux.dbf

3 /test/dul/lunar/user.dbf

4 /test/dul/lunar/index.dbf

5 /test/dul/lunar/test.dbf

control.dul的例子2

```
# AIX version 7 example with one file on raw device
   1 /usr/oracle/dbs/system.dbf
   8 /dev/rdsk/data.dbf 4096

   # Oracle8 example with a datafile split in multiple parts, each part smaller than 2GB
   0  1 /fs1/oradata/PMS/system.dbf
   1  2 /tmp/huge_file_part1 startblock 1 endblock 1000000
   1  2 /tmp/huge_file_part2 startblock 1000001 endblock 2000000
   1  2 /mnt3/huge_file_part3 startblock 2000001 endblock 2550000
```

使用 DUL

案例 1:

- SYSTEM 表空间可用

案例 2:

- SYSTEM 表空间不可用

案例1: 数据字典可用

- 简单、直接的方法
- 在OS上执行 ‘dul’，然后在DUL中执行 ‘bootstrap’
- 不需要了解应用的表结构，列类型，等等

案例2: SYSTEM表空间(文件)不可用

需要深入了解应用和业务表

- 如果你不了解应用和业务表相关信息，那么**unloaded** 出来的数据没有任何意义

- **DUL**可以猜出列的类型，但是**unload**出来的数据没有表名和列名
  - 列类型也可能被猜错
  - 还可以有类似下面的方法进行处理：
  - **insert into lost_tables select t.obj#, u.name, o.name, t.cols, NULL, NULLfrom sys.obj$ o, sys.tab$ t, sys.user$ uwhere o.obj# = t.obj# and o.owner# = u.user# and upper(u.name) like upper('&&USER_NAME') and upper(o.name) like upper('&&TABLE_NAME') and o.type = 2 and t.clu# IS NULL;**
  - 等等

案例2: SYSTEM表空间(文件)不可用(续)

DUL会丢失Trailing NULL columns数据

- 因为**Trailing NULL columns**（行数据末尾的字段如果是**null**）不存储在数据库中会导出已经被**drop**的表
- 当表被**drop,** 相关的信息就从数据字典中删除了空表不会被**unloader**出来

DUL 启动步骤

DUL的启动步骤:

- 读取"init.dul"
- 扫描DUL 的控制文件 (缺省为"control.dul")
- 尝试加载 USER$, OBJ$, TAB$ 和COL$的dump文件，如果这些文件可用（之前已经存在于该目录下），如果这些文件可用，DUL会加载他们到DUL的数据字典cache中
- 尝试加载seg.dat 和 col.dat.
- 获取DDL语句或者运行DDL脚本(启动的参数中可以指定)

数据字典可用时的DUL 步骤

- 配置 init.dul 和 control.dul
- 执行 DuL
- Bootstrap
- Unload database, user, table

数据字典不可用时的DUL 步骤

- 配置 init.dul 和 control.dul (control.dul中只包含需要恢复的数据文件的信息).
- 执行 DuL
- alter session set use_scanned_extent_map = true
- scan database
- scan tables
- 使用带有表结构定义的unload语句 :

```
unload table dul2.emp (EMPLOYEE_ID number(22), FIRST_NAME varchar2(20),
LAST_NAME varchar2(25),
EMAIL varchar2(25),PHONE_NUMBER varchar2(20), HIRE_DATE date, JOB_ID varchar2 (10),
SALARY number(22), COMMISSION_PCT number(22),MANAGER_ID number(22),
DEPARTMENT_ID number(22))
storage (dataobjno 28200);
```

Q & A