OBET 工具使用说明

软件介绍

OBET(Oracle Block Editor Tool)是惜分飞(www.xifenfei.com)开发的使用于 Oracle 数据块查看和编辑小工具

版本历史介绍

1. 2025 年 11 月 7 日 发布 2025.11.001 版本（第一版）实现数据块的查看,编辑,查看文件头基础结构,tailchk/checksum 修复，数据拷贝功能
2. 2025 年 11 月 15 日 发布 2025.11.002 版本（第二版）增加修复坏块，修复文件头 SCN，修复 Resetlogs，数据块拷贝功能，把第一版中的 copy 命令进行了补充和完善

主要功能

1. 16 进制格式查看 Oracle 数据块(可以指定 block/offset，不受 4G 大小限制)
2. 16 进制格式编辑 Oracle 数据块(可以指定 block/offset，不受 4G 大小限制)
3. 标记数据块为坏块功能
4. 自动修复 tailchk
5. 自动修复 checksum
6. 任意另个数据文件之间任意数据块内部任何大小数据的 copy(比如把 file 1 block 1 offset 32 count 128 拷贝到 file 10 block 2 offset 128 开始的位置)
7. 支持文件头 print/p 查看注意一级结构体
8. 坏块修复功能
9. 不同文件之前数据块拷贝功能
10. 增加文件头 checkpoint scn 修复功能
11. 增加文件头 resetlogs 信息修复功能

支持数据库版本

oracle 10g 到 oracle 26ai

软件版本

联系作者

QQ：107644445

邮箱：dba@xifenfei.com

微信/电话：17813235971

下载地址

百度网盘（https://pan.baidu.com/s/1Z7iqIoTO4ejhmBaLRhUUHw?pwd=27at）

启动软件

Linux 平台

```
chmod +x obet
./obet 进入 OBET> 界面
```

Windows 平台

```
双击 obet.exe 进入 OBET> 界面
```

启动软件之后进入界面

```
================================================
Welcome to Oracle Block Editor Tool (OBET)
================================================

***** !!! For Oracle Internal Use only !!! *****

[Software Function Description]
- View and edit data block in hexadecimal format
- Automatically repair tailchk and checksum
- Copy data between different data files
- Mark data block as corrupted block

[Developer Information]
- Name: XiFenFei
- Phone: +86-17813235971
- Email: dba@xifenfei
- Q Q: 107644445
- WeChat: 17813235971
- Website: https://www.xifenfei.com

[Version Details]
- Software Version: v2025.11.001
- Build Date: 2025.11.07

================================================
```

```
Type 'help' for command list | 'exit' to quit
==========================================


================================================
Software License Status: Not Registered
================================================
This software is not registered.
Please use 'license' command to register.
================================================


OBET>
```

使用帮助

输入 help 显示整体使用说明

```
OBET (Oracle Block Editor Tool) commands:
  open <config_file>    - Load file list from config file (format: <num> <path>)
  info                  - Show loaded file list (from open command)
  set filename <path>   - Set target file path (required)
  set file <num>        - Set filename using loaded file number (from open list)
  set blocksize <size>  - Set block size (2048,4096,[8192],16384,32768)
  set block <num>       - Set block number (starts from 0, default: 1)
  set offset <offset>   - Set offset within block (< blocksize, default: 0)
  set count <bytes>     - Set number of bytes to read (default: 32)
  set mode edit/browse  - Enable edit/browse mode
  d/dump [options]      - Display data (options: block X, offset Y, count N)
  m/modify <hex> [opts] - Modify data with hex (opts: block X, offset Y)
  undo                  - Undo last modification
  sum [block X]         - Calculate checksum for block (default: current block)
  sum apply [block X]   - Apply checksum: write calculated value to block
  tailchk [block X]     - Calculate tailchk for block (default: current block)
  tailchk apply [block X] - Apply tailchk: write calculated value to block
  copy <src> to <dest>  - Copy data between files
    <src> format: file,block,offset,count (e.g., 1,1,10,64)
    <dest> format: file[,block][,offset] (e.g., 3 or 3,1 or 3,1,128)
  corrupt [block X]     - Mark block as corrupted (default: current block)
  show                   - Display current settings (filename, blocksize, block, offset, count,
mode)
  license               - Show/manage software license (registration code required)
  version               - Show software version and developer information
  p/print <param>       - Print Oracle structure,Use the 'p/print' command to see details
  help                  - Show this help message
  quit/exit             - Exit OBET
```

输入 p/print 显示打印一级结构体具体信息

```
OBET> p
Error: Missing param.  The main commands include:

Supported structure display (one-level only):
  p tailchk                         - Block tail check
  p kcvfh                           - Complete KCVFH structure
  p kcvfh.kcvfhbfh                  - File header block
  p kcvfh.kcvfhhdr                  - File header descriptor
  p kcvfh.kcvfhrdb                  - Database block
  p kcvfh.kcvfhcrs                  - Creation SCN
  p kcvfh.kcvfhcrt                  - Creation time
  p kcvfh.kcvfhrlc                  - Reset log SCN
  p kcvfh.kcvfhrls                  - Reset logs SCN
```

```
  p kcvfh.kcvfhbti                - Backup table info
  p kcvfh.kcvfhbsc                - Basic SCN
  p kcvfh.kcvfhbth                - Backup table header
  p kcvfh.kcvfhsta                - Status
  p kcvfh.kcvfhbcp                - Checkpoint info
  p kcvfh.kcvfhbhz                - Block size header
  p kcvfh.kcvfhxcd                - Extended checkpoint
  p kcvfh.kcvfhtsn                - Tablespace number
  p kcvfh.kcvfhtln                - Tablespace name length
  p kcvfh.kcvfhtnm                - Tablespace name
  p kcvfh.kcvfhrfn                - Relative file number
  p kcvfh.kcvfhrfs                - Recovery info
  p kcvfh.kcvfhrft                - Recovery time
  p kcvfh.kcvfhafs                - Auxiliary recovery
  p kcvfh.kcvfhbcs                - Backup control
  p kcvfh.kcvfhofb                - Offset bytes
  p kcvfh.kcvfhnfb                - Next offset bytes
  p kcvfh.kcvfhprc                - Recovery related
  p kcvfh.kcvfhprs                - Recovery structure
  p kcvfh.kcvfhprfs               - Recovery finish
  p kcvfh.kcvfhcpc                - Checkpoint count
  p kcvfh.kcvfhrts                - Reset time
  p kcvfh.kcvfhccc                - Change count
  p kcvfh.kcvfhtrt                - Total reset time
  p kcvfh.kcvfhckp                - Complex checkpoint

Note: Only one-level structure access is supported.
Example: 'p kcvfh.kcvfhhdr' shows the complete kcvfhhdr structure.
No multi-level access like 'p kcvfh.kcvfhhdr.kccfhtyp' is supported.
```

打开数据文件

通过 set filename 数据文件路径

```
OBET> set filename h:/xifenfei/system01.dbf
filename set to: h:/xifenfei/system01.dbf

OBET> show

Current settings:
File: h:/xifenfei/system01.dbf
Blocksize: 8192 bytes
Block: 1
Offset in block: 0 (file offset: 0x00002000)
Count: 32 bytes
Mode: browse
```

Open 数据文件列表

```
OBET> open h:/xifenfei/datafile.txt
Loaded 3 files from config file 'h:/xifenfei/datafile.txt'.

OBET> info

Loaded files (3 total):
---------------------------------------
Number  Path
---------------------------------------
     1  H:\xifenfei\system01.dbf
     2  H:\xifenfei\sysaux01.dbf
```

```
     3  H:\xifenfei\undo01.dbf
----------------------------------------

OBET> set file 2
filename set to: H:\xifenfei\sysaux01.dbf (file#2)

OBET> show

Current settings:
File: H:\xifenfei\sysaux01.dbf
Blocksize: 8192 bytes
Block: 1
Offset in block: 0 (file offset: 0x00002000)
Count: 32 bytes
Mode: browse
Loaded files: 3 (use 'info' to list)
```

查看数据块内容（16 进制）

使用 d/dump 命令配合 block n offset m 查看 16 进制格式

d/dump <block n> <offset m>

```
OBET> d

File: H:\xifenfei\sysaux01.dbf
Block: 1               Offsets:     0 to    31
--------------------------------------------------------------------------------
00002000 0BA20000 01008000 00000000 00000104 DE840000 00000000 0004200B 9A794D59

OBET> set block 128
block set to: 128

OBET> dump

File: H:\xifenfei\sysaux01.dbf
Block: 128               Offsets:     0 to    31
--------------------------------------------------------------------------------
00100000 20A20000 80008000 7E090000 00000104 C1860000 00000000 00000000 00000000

<32 bytes read>

OBET> dump block 1 offset 484

File: H:\xifenfei\sysaux01.dbf
Block: 1               Offsets:   484 to   515
--------------------------------------------------------------------------------
000021E4 09BFDD1B 3D160000 FC618348 01000000 D3E30600 4BB40000 10000000 02000000

<32 bytes read>

OBET> set count 128
count set to: 128

OBET> dump block 1 offset 484

File: H:\xifenfei\sysaux01.dbf
Block: 1               Offsets:   484 to   611
--------------------------------------------------------------------------------
000021E4 09BFDD1B 3D160000 FC618348 01000000 D3E30600 4BB40000 10000000 02000000
00002204 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00002224 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00002244 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

```
<128 bytes read>
```

查看文件头结构体

使用 p/print 命令查看，具体输入 p 或者 print 会提示支持命令

本软件在这个功能上，修复了在 win 平台上 bbed 偏移量异常，和对于高版本的 big scn 的格式展示的支持

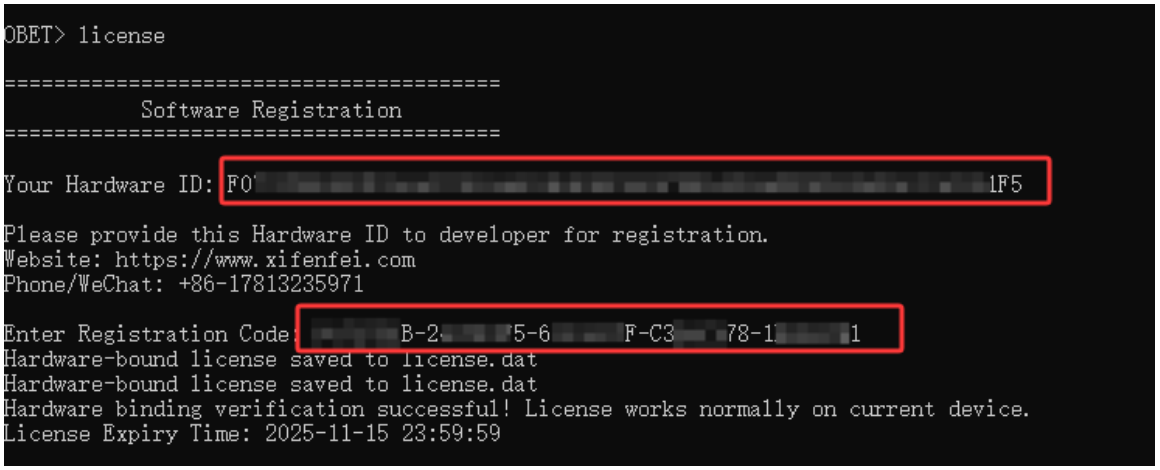
```
OBET> p kcvfh
Warning:kcvfh only supports block 1, force set block 1
File: H:\xifenfei\sysaux01.dbf
Size: 8192 bytes
Block: 1

struct kcvfh, 1272 bytes                    @0
   struct kcvfhbfh, 20 bytes                 @0
      ub1 type_kcbh                          @0        0x0B
      ub1 frmt_kcbh                          @1        0xA2
      ub2 wrp2_kcbh                          @2        0x0000
      ub4 rdba_kcbh                          @4        0x00800001
      ub4 bas_kcbh                           @8        0x00000000
      ub2 wrp_kcbh                           @12       0x0000
      ub1 seq_kcbh                           @14       0x01
      ub1 flg_kcbh                           @15       0x04
      ub2 chkval_kcbh                        @16       0x84DE
      ub2 spare3_kcbh                        @18       0x0000
   struct kcvfhhdr, 76 bytes                 @20
      ub4 kccfhswv                           @20       0x00000000
      ub4 kccfhcvn                           @24       0x0B200400
      ub4 kccfhdbi                           @28       0x594D799A
      text kccfhdbn[8]                       @32-39    SCMHK
      ub4 kccfhcsq                           @40       0x008B41EF
      ub4 kccfhfsz                           @44       0x00088400
      s_blkz kccfhbsz                        @48       0x00
      ub2 kccfhfno                           @52       0x0002
      ub2 kccfhtyp                           @54       0x0003
      ub4 kccfhacid                          @56       0x00000000
      ub4 kccfhcks                           @60       0x00000000
      text kccfhtag[32]                      @64-95
   ub4 kcvfhrdb                              @96       0x00000000
   struct kcvfhcrs, 8 bytes                   @100
      ub4 kscnbas                            @100      0x00000870
      ub2 kscnwrp                            @104      0x0000
      ub2 kscnwrp2                           @106      0x0000
   ub4 kcvfhcrt                              @108      0x2A9DFAF8
   ub4 kcvfhrlc                              @112      0x3215EC5C
   struct kcvfhrls, 8 bytes                   @116
      ub4 kscnbas                            @116      0x000E74FF
      ub2 kscnwrp                            @120      0x0000
      ub2 kscnwrp2                           @122      0x0000
   ub4 kcvfhbti                              @124      0x00000000
   struct kcvfhbsc, 8 bytes                   @128
      ub4 kscnbas                            @128      0x00000000
      ub2 kscnwrp                            @132      0x0000
      ub2 kscnwrp2                           @134      0x0000
   ub2 kcvfhbth                              @136      0x0000
   ub2 kcvfhsta                              @138      0x0000 (NONE)
   struct kcvfhckp, 36 bytes                  @484
      struct kcvcpscn, 8 bytes                @484
```

```
         ub4 kscnbas                          @484       0x1BDDBF09
         ub2 kscnwrp                          @488       0x163D
         ub2 kscnwrp2                         @490       0x0000
      ub4 kcvcptim                            @492       0x488361FC
      ub2 kcvcpthr                            @496       0x0001
      union u, 12 bytes                       @500
         struct kcvcprba, 12 bytes            @500
            ub4 kcrbaseq                      @500       0x0006E3D3
            ub4 kcrbabno                      @504       0x0000B44B
            ub2 kcrbabof                      @508       0x0010
      ub1 kcvcpetb[8]                         @512-519 02 00 00 00 00 00 00 00
   ub4 kcvfhcpc                               @140       0x0006E58C
   ub4 kcvfhrts                               @144       0x4881FDEE
   ub4 kcvfhccc                               @148       0x0006E58B
   struct kcvfhbcp, 36 bytes                   @152
      struct kcvcpscn, 8 bytes                 @152
         ub4 kscnbas                          @152       0x00000000
         ub2 kscnwrp                          @156       0x0000
         ub2 kscnwrp2                         @158       0x0000
      ub4 kcvcptim                            @160       0x00000000
      ub2 kcvcpthr                            @164       0x0000
      union u, 12 bytes                       @168
         struct kcvcprba, 12 bytes            @168
            ub4 kcrbaseq                      @168       0x00000000
            ub4 kcrbabno                      @172       0x00000000
            ub2 kcrbabof                      @176       0x0000
      ub1 kcvcpetb[8]                         @180-187 00 00 00 00 00 00 00 00
   ub4 kcvfhbhz                               @312       0x00000000
   struct kcvfhxcd, 16 bytes                   @316
      ub4 space_kcvmxcd[0]                    @316       0x00000000
      ub4 space_kcvmxcd[1]                    @320       0x00000000
      ub4 space_kcvmxcd[2]                    @324       0x00000000
      ub4 space_kcvmxcd[3]                    @328       0x00000000
   sword kcvfhtsn                             @332       1
   ub2 kcvfhtln                               @336       0x0006
   text kcvfhtnm[30]                          @338-367 SYSAUX
   ub4 kcvfhrfn                               @368       0x00000002
   struct kcvfhrfs, 8 bytes                    @372
      ub4 kscnbas                             @372       0x00000000
      ub2 kscnwrp                             @376       0x0000
      ub2 kscnwrp2                            @378       0x0000
   ub4 kcvfhrft                               @380       0x00000000
   struct kcvfhafs, 8 bytes                    @384
      ub4 kscnbas                             @384       0x00000000
      ub2 kscnwrp                             @388       0x0000
      ub2 kscnwrp2                            @390       0x0000
   ub4 kcvfhbbc                               @392       0x00000000
   ub4 kcvfhncb                               @396       0x00000000
   ub4 kcvfhmcb                               @400       0x00000000
   ub4 kcvfhlcb                               @404       0x00000000
   ub4 kcvfhbcs                               @408       0x00000000
   ub2 kcvfhofb                               @412       0x0000
   ub2 kcvfhnfb                               @414       0x0000
   ub4 kcvfhprc                               @416       0x2A9DFAF0
   struct kcvfhprs, 8 bytes                    @420
      ub4 kscnbas                             @420       0x00000001
      ub2 kscnwrp                             @424       0x0000
      ub2 kscnwrp2                            @426       0x0000
   struct kcvfhprfs, 8 bytes                   @428
      ub4 kscnbas                             @428       0x00000000
      ub2 kscnwrp                             @432       0x0000
      ub2 kscnwrp2                            @434       0x0000
   ub4 kcvfhtrt                               @444       0x00000000

<kcvfh structure printed successfully>
```

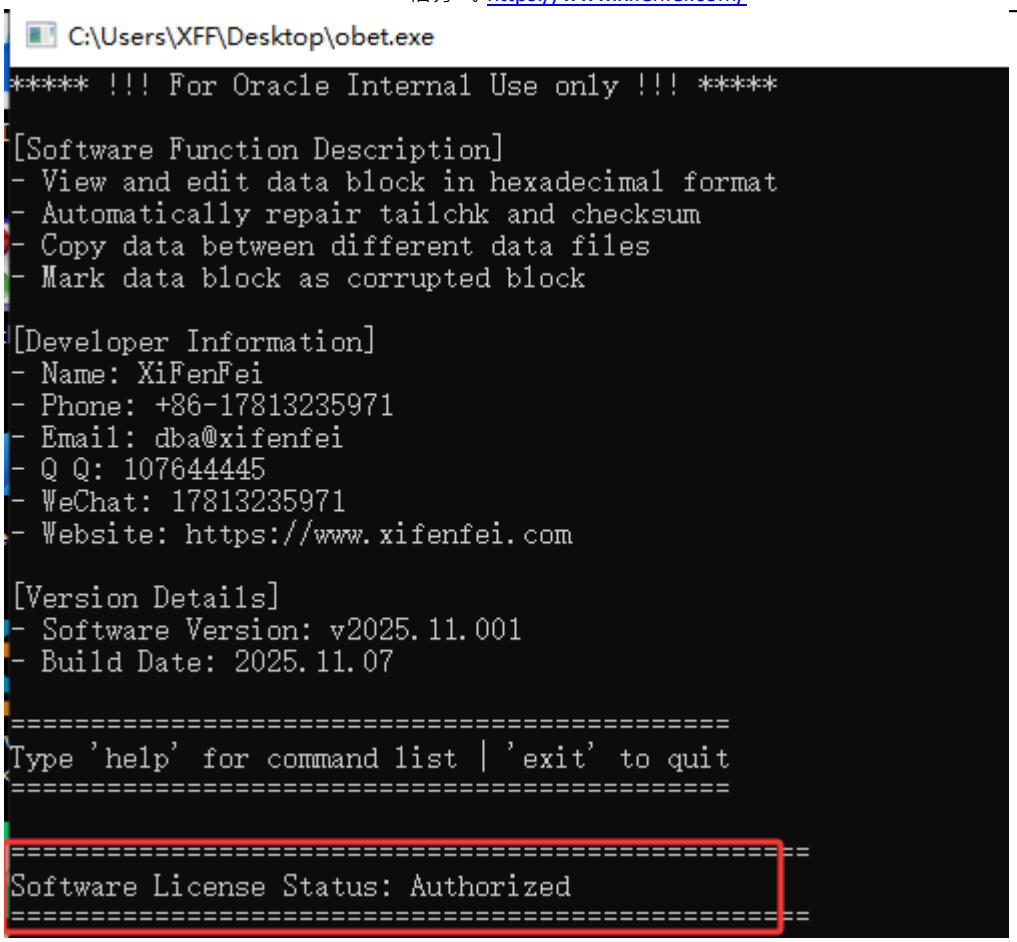
```
OBET> p kcvfh.kcvfhckp
File: H:\xifenfei\sysaux01.dbf
Size: 8192 bytes
Block: 1
Offset: 484

struct kcvfhckp, 36 bytes                   @484
   struct kcvcpscn, 8 bytes                 @484
      ub4 kscnbas                           @484      0x1BDDBF09
      ub2 kscnwrp                           @488      0x163D
      ub2 kscnwrp2                          @490      0x0000
   ub4 kcvcptim                             @492      0x488361FC
   ub2 kcvcpthr                             @496      0x0001
   union u, 12 bytes                        @500
      struct kcvcprba, 12 bytes             @500
         ub4 kcrbaseq                       @500      0x0006E3D3
         ub4 kcrbabno                       @504      0x0000B44B
         ub2 kcrbabof                       @508      0x0010
   ub1 kcvcpetb[8]                          @512-519 02 00 00 00 00 00 00 00

<kcvfh.kcvfhckp structure printed successfully>

OBET> p tailchk
File: H:\xifenfei\sysaux01.dbf
Size: 8192 bytes
Block: 1
Offset: 8188

ub4 tailchk                                 @8188     0x00000B01

<tailchk structure printed successfully>
```

软件授权

如果没授权输入 license 会得到硬件 ID



把这个 ID 发给惜分飞，获取授权码，进行授权，授权成功会提示有效期和在 obet 程序所在目录写入 license.dat 文件，授权成功即可正常使用,在授权期限内重启自动授权



编辑数据块内容（16 进制）

为了防止软件被乱用，恶意修改数据库，对于所有数据块修改操作需要注册之后才能使用，如果没授权进入编辑模式会被拒绝

```
OBET> set mode edit
Error: Software is not properly licensed. Please register first.
```

使用 m/modify 命令进行数据块修改，需要修改什么位置的数据，可以先通过 set block N，set offset M，进入到需要修改的位置，然后通过 m/modify 命令进行修改

m/modify 16 进制数据 <block n> <offset m>

```
OBET> info

Loaded files (3 total):
---------------------------------------
Number  Path
---------------------------------------
     1  H:\xifenfei\system01.dbf
     2  H:\xifenfei\sysaux01.dbf
     3  H:\xifenfei\undo01.dbf
---------------------------------------

OBET> set file 3
```

```
filename set to: H:\xifenfei\undo01.dbf (file#3)

OBET> set block 2
block set to: 2

OBET> d

File: H:\xifenfei\undo01.dbf
Block: 2               Offsets:     0 to    31
--------------------------------------------------------------------------------
00004000 1DA20000 02000002 FABCDD1B 3D160204 11EC0000 08000000 08000000 10130000

<32 bytes read>

OBET> set offset 8
offset set to: 8

OBET> d

File: H:\xifenfei\undo01.dbf
Block: 2               Offsets:     8 to    39
--------------------------------------------------------------------------------
00004008 FABCDD1B 3D160204 11EC0000 08000000 08000000 10130000 09000000 01000000

<32 bytes read>

OBET> set mode edit
mode set to: edit

OBET> m 1234

Confirm modification:
File: H:\xifenfei\undo01.dbf
Block: 2
Offset: 8 (file offset: 0x00004008)
Original value: FABC
New value:      1234
Confirm? (Y/YES to proceed): y
Verification successful: Data written correctly.
Modified 2 bytes at offset 0x00004008 successfully.

OBET> d

File: H:\xifenfei\undo01.dbf
Block: 2               Offsets:     8 to    39
--------------------------------------------------------------------------------
00004008 1234DD1B 3D160204 11EC0000 08000000 08000000 10130000 09000000 01000000

<32 bytes read>
```

tailchk 自动修复

使用 tailchk 计算当前的 tailchk 值是否正确，如果不正确直接使用 tailchk apply 命令进行修复（一种方法是进入到需要处理的 block，另外一种直接 在 tailchk apply block N）

tailchk <block n>

tailchk apply <block n>

```
OBET> tailchk
```

```
Check tailchk for File H:\xifenfei\undo01.dbf, Block 2:
current = 0x021DFABC, required = 0x1D021234

OBET> tailchk apply

Confirm applying tailchk:
File: H:\xifenfei\undo01.dbf
Block: 2
Offset in block: 8188 (file offset: 0x00005FFC)
Original value: 0x021DFABC
New value:      0x1D021234
Confirm? (Y/YES to proceed): y
Verification successful: Stored tailchk matches calculated value (0x1D021234).
Tailchk applied successfully.
```

checksum 自动修复

使用 sum 计算当前的 checksum 值是否正确，如果不正确直接使用 sum apply 命令进行修复（一种方法是进入到需要处理的 block，另外一种直接 在 sum apply block N）

sum <block n>

sum apply <block n>

```
OBET> sum
Check value for File H:\xifenfei\undo01.dbf, Block 2:
current = 0x11EC, required = 0x4250

OBET> sum apply

Confirm applying checksum:
File: H:\xifenfei\undo01.dbf
Block: 2
Offset in block: 16 (file offset: 0x00004010)
Original value: 0x11EC
New value:      0x4250
Confirm? (Y/YES to proceed): y
Verification successful: Stored checksum matches calculated value (0x4250).
Checksum applied successfully.
```

撤销最后一次操作

使用 undo 命令可以撤销最后一次对数据块的修改操作

```
OBET> d

File: H:\xifenfei\undo01.dbf
Block: 2               Offsets:     8 to    39
-------------------------------------------------------------------------------
00004008 1234DD1B 3D160204 42500000 08000000 08000000 10130000 09000000 01000000

<32 bytes read>

OBET> m abcd

Confirm modification:
```

```
File: H:\xifenfei\undo01.dbf
Block: 2
Offset: 8 (file offset: 0x00004008)
Original value: 1234
New value:      ABCD
Confirm? (Y/YES to proceed): y
Verification successful: Data written correctly.
Modified 2 bytes at offset 0x00004008 successfully.

OBET> d

File: H:\xifenfei\undo01.dbf
Block: 2               Offsets:     8 to    39
-------------------------------------------------------------------------------
00004008 ABCDDD1B 3D160204 42500000 08000000 08000000 10130000 09000000 01000000

<32 bytes read>

OBET> undo
Undo successful. Restored 2 bytes at offset 0x00004008.

OBET> d

File: H:\xifenfei\undo01.dbf
Block: 2               Offsets:     8 to    39
-------------------------------------------------------------------------------
00004008 1234DD1B 3D160204 42500000 08000000 08000000 10130000 09000000 01000000

<32 bytes read>
```

Copy data 数据操作

在有些情况下，需要把一个数据文件的某些数据拷贝到另外一个数据文件的某个位置,obet 可以通过 copy 命令来实现，命令格式:

copy data file,block,offset,count to file<,block><,offset>目标端 block 不写就是默认和源端一致，offset 不写默认和源端一致

```
OBET> copy data 1,1,0,128 to 2,2,8

Confirm copy:
Source: file#1 (H:\xifenfei\system01.dbf), block 1, offset 0, 128 bytes
Target: file#2 (H:\xifenfei\sysaux01.dbf), block 2, offset 8
Proceed? (Y/YES to confirm): y
Copy successful: 128 bytes copied from file #1 to file #2.

OBET> set file 2
filename set to: H:\xifenfei\sysaux01.dbf (file#2)

OBET> dump block 2 offset 8

File: H:\xifenfei\sysaux01.dbf
Block: 2               Offsets:     8 to   135
-------------------------------------------------------------------------------
00004008 0BA20000 01004000 00000000 00000104 497B0000 00000000 0004200B 9A794D59
00004028 53434D48 4B000000 EF418B00 00470900 00200000 01000300 00000000 00000000
00004048 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00004068 08024000 07000000 00000000 F4FA9D2A 5CEC1532 FF740E00 00000000 00000000

<128 bytes read>
```

```
OBET> set file 1
filename set to: H:\xifenfei\system01.dbf (file#1)

OBET> d block 1 offset 0

File: H:\xifenfei\system01.dbf
Block: 1               Offsets:     0 to   127
------------------------------------------------------------------------------
00002000 0BA20000 01004000 00000000 00000104 497B0000 00000000 0004200B 9A794D59
00002020 53434D48 4B000000 EF418B00 00470900 00200000 01000300 00000000 00000000
00002040 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00002060 08024000 07000000 00000000 F4FA9D2A 5CEC1532 FF740E00 00000000 00000000

<128 bytes read>
```

Copy chkscn 修改文件头 scn 信息

在有些情况下，需要把一个数据文件头的 scn 同步另外一个数据文件头上，命令格式:

copy chkscn file n to file m

```
OBET> p kcvfh.kcvfhckp
File: H:\xifenfei\system01.dbf
Size: 8192 bytes
Block: 1
Offset: 484

struct kcvfhckp, 36 bytes                  @484
   struct kcvcpscn, 8 bytes                @484
      ub4 kscnbas                          @484      0x1BDDBF09
      ub2 kscnwrp                          @488      0x163D
      ub2 kscnwrp2                         @490      0x0000
   ub4 kcvcptim                            @492      0x488361FC
   ub2 kcvcpthr                            @496      0x0001
   union u, 12 bytes                       @500
      struct kcvcprba, 12 bytes            @500
         ub4 kcrbaseq                      @500      0x0006E3D3
         ub4 kcrbabno                      @504      0x0000B44B
         ub2 kcrbabof                      @508      0x0010
   ub1 kcvcpetb[8]                         @512-519 02 00 00 00 00 00 00 00


OBET> p kcvfh.kcvfhckp
File: H:\xifenfei\sysaux01.dbf
Size: 8192 bytes
Block: 1
Offset: 484

struct kcvfhckp, 36 bytes                  @484
   struct kcvcpscn, 8 bytes                @484
      ub4 kscnbas                          @484      0x67452301
      ub2 kscnwrp                          @488      0x0000
      ub2 kscnwrp2                         @490      0x0000
   ub4 kcvcptim                            @492      0x00000000
   ub2 kcvcpthr                            @496      0x0000
   union u, 12 bytes                       @500
      struct kcvcprba, 12 bytes            @500
         ub4 kcrbaseq                      @500      0x0006E3D3
         ub4 kcrbabno                      @504      0x0000B44B
         ub2 kcrbabof                      @508      0x0010
   ub1 kcvcpetb[8]                         @512-519 02 00 00 00 00 00 00 00
```

```
<kcvfh.kcvfhckp structure printed successfully>



OBET> copy chkscn file 1 to file 2

Confirm Modify chkscn:
Source: file#1 (H:\xifenfei\system01.dbf)
Target: file#2 (H:\xifenfei\sysaux01.dbf)
Proceed? (Y/YES to confirm): y
Successfully copied checkpoint SCN information from file#1 to file#2.

OBET> p kcvfh.kcvfhckp
File: H:\xifenfei\sysaux01.dbf
Size: 8192 bytes
Block: 1
Offset: 484

struct kcvfhckp, 36 bytes                   @484
   struct kcvcpscn, 8 bytes                 @484
      ub4 kscnbas                           @484      0x1BDDBF09
      ub2 kscnwrp                           @488      0x163D
      ub2 kscnwrp2                          @490      0x0000
   ub4 kcvcptim                             @492      0x488361FC
   ub2 kcvcpthr                             @496      0x0001
   union u, 12 bytes                        @500
      struct kcvcprba, 12 bytes             @500
         ub4 kcrbaseq                       @500      0x0006E3D3
         ub4 kcrbabno                       @504      0x0000B44B
         ub2 kcrbabof                       @508      0x0010
   ub1 kcvcpetb[8]                          @512-519 02 00 00 00 00 00 00 00

<kcvfh.kcvfhckp structure printed successfully>
```

Copy resetlogscn 修改文件头 resetlogs 信息

在有些情况下，需要把一个数据文件头的 resetlogs 信息 同步另外一个数据文件头上，命令格式:

copy resetlogscn file n to file m

```
OBET> p kcvfh.kcvfhrlc
File: H:\xifenfei\system01.dbf
Size: 8192 bytes
Block: 1
Offset: 112

ub4 kcvfhrlc                            @112      0x3215EC5C

<kcvfh.kcvfhrlc structure printed successfully>

OBET> p kcvfh.kcvfhrls
File: H:\xifenfei\system01.dbf
Size: 8192 bytes
Block: 1
Offset: 116

struct kcvfhrls, 8 bytes                    @116
   ub4 kscnbas                              @116      0x000E74FF
   ub2 kscnwrp                              @120      0x0000
   ub2 kscnwrp2                             @122      0x0000
```

```
<kcvfh.kcvfhrls structure printed successfully>


OBET> p kcvfh.kcvfhrlc
File: H:\xifenfei\sysaux01.dbf
Size: 8192 bytes
Block: 1
Offset: 112

ub4 kcvfhrlc                            @112     0x67452301

<kcvfh.kcvfhrlc structure printed successfully>

OBET> p kcvfh.kcvfhrls
File: H:\xifenfei\sysaux01.dbf
Size: 8192 bytes
Block: 1
Offset: 116

struct kcvfhrls, 8 bytes                  @116
   ub4 kscnbas                           @116     0x43658709
   ub2 kscnwrp                           @120     0x0021
   ub2 kscnwrp2                          @122     0x0000

<kcvfh.kcvfhrls structure printed successfully>

OBET> copy resetlogscn file 1 to file 2

Confirm Modify resetlogscn:
Source: file#1 (H:\xifenfei\system01.dbf)
Target: file#2 (H:\xifenfei\sysaux01.dbf)
Proceed? (Y/YES to confirm): y
Successfully copied resetlog SCN information from file#1 to file#2.

OBET> p kcvfh.kcvfhrlc
File: H:\xifenfei\sysaux01.dbf
Size: 8192 bytes
Block: 1
Offset: 112

ub4 kcvfhrlc                            @112     0x3215EC5C

<kcvfh.kcvfhrlc structure printed successfully>

OBET> p kcvfh.kcvfhrls
File: H:\xifenfei\sysaux01.dbf
Size: 8192 bytes
Block: 1
Offset: 116

struct kcvfhrls, 8 bytes                  @116
   ub4 kscnbas                           @116     0x000E74FF
   ub2 kscnwrp                           @120     0x0000
   ub2 kscnwrp2                          @122     0x0000

<kcvfh.kcvfhrls structure printed successfully>
```

标记数据坏块

在某些情况下，由于某种原因需要把数据块标记为坏块,通过 corrupt 命令即可实现

corrupt <block n>

```
C:\Users\XFF>dbv file=h:/xifenfei/users01.dbf

DBVERIFY: Release 11.2.0.4.0 - Production on 星期三 11月 12 21:40:55 2025

Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.

DBVERIFY - 开始验证: FILE = H:\XIFENFEI\USERS01.DBF


DBVERIFY - 验证完成

检查的页总数: 26656
处理的页总数 (数据): 23569
失败的页总数 (数据): 0
处理的页总数 (索引): 309
失败的页总数 (索引): 0
处理的页总数 (其他): 720
处理的总页数 (段)  : 0
失败的总页数 (段)  : 0
空的页总数: 2058
标记为损坏的总页数: 0
流入的页总数: 0
加密的总页数         : 0
最高块 SCN            : 2255414715 (18.2255414715)


OBET> corrupt block 10

Confirm modification:
File: h:/xifenfei/users01.dbf
Block: 10
Offset: 14 (file offset: 0x0001400E)
Original value: 01
New value:      FF
Are you sure to set this block corrupted? (Y/YES to proceed): y
Verification successful: Block 10 marked as corrupted (offset 14 set to 0xFF).
Modification successful.

C:\Users\XFF>dbv file=h:/xifenfei/users01.dbf

DBVERIFY: Release 11.2.0.4.0 - Production on 星期三 11月 12 21:41:22 2025

Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.

DBVERIFY - 开始验证: FILE = H:\XIFENFEI\USERS01.DBF
页 10 流入 - 很可能是介质损坏
Corrupt block relative dba: 0x0100000a (file 4, block 10)
Fractured block found during dbv:
Data in bad block:
 type: 30 format: 2 rdba: 0x0100000a
 last change scn: 0x0000.00003e78 seq: 0xff flg: 0x04
 spare1: 0x0 spare2: 0x0 spare3: 0x0
 consistency value in tail: 0x3e781e01
 check value in block header: 0x1a6
 computed block checksum: 0xfe
```

```
DBVERIFY - 验证完成

检查的页总数: 26656
处理的页总数 (数据): 23569
失败的页总数 (数据): 0
处理的页总数 (索引): 309
失败的页总数 (索引): 0
处理的页总数 (其他): 719
处理的总页数 (段)  : 0
失败的总页数 (段)  : 0
空的页总数: 2058
标记为损坏的总页数: 1
流入的页总数: 1
加密的总页数         : 0
最高块 SCN            : 2255414715 (18.2255414715)
```

修复数据坏块

在某些情况下，可以通过 repair 命令数据块坏块自动修复

Repair <block n>

```
C:\Users\XFF>dbv file=h:/xifenfei/undo01.dbf

DBVERIFY: Release 11.2.0.4.0 - Production on 星期日 11 月 16 00:06:59 2025

Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.

DBVERIFY - 开始验证: FILE = H:\XIFENFEI\UNDO01.DBF
页 2 流入 - 很可能是介质损坏
Corrupt block relative dba: 0x02000002 (file 8, block 2)
Fractured block found during dbv:
Data in bad block:
 type: 29 format: 2 rdba: 0x02000002
 last change scn: 0x163d.1bddbcfa seq: 0xff flg: 0x04
 spare1: 0x0 spare2: 0x0 spare3: 0x0
 consistency value in tail: 0xbcfa1d02
 check value in block header: 0xec11
 computed block checksum: 0xfd



DBVERIFY - 验证完成

检查的页总数: 4880
处理的页总数 (数据): 0
失败的页总数 (数据): 0
处理的页总数 (索引): 0
失败的页总数 (索引): 0
处理的页总数 (其他): 4372
处理的总页数 (段)  : 11
失败的总页数 (段)  : 0
空的页总数: 507
标记为损坏的总页数: 1
流入的页总数: 1
加密的总页数         : 0
最高块 SCN            : 467517187 (5693.467517187)
```

```
OBET> repair

Repairing block 2 in file H:\xifenfei\undo01.dbf...

Repair analysis for block 2:
1. seq_kcbh check: 0xFF -> needs repair (0x01)
2. Tailchk check: 0x021DFABC -> needs repair (0x011DFABC)
3. Checksum check: 0x11EC -> OK

Confirm repair operations:
File: H:\xifenfei\undo01.dbf
Block: 2
Operations needed: fix offset14, fix tailchk
Confirm? (Y/YES to proceed): y

Verification after repair:
1. seq_kcbh: 0x01 OK
2. Tailchk: 0x011DFABC OK
3. Checksum: 0x11EC OK

Block 2 repair completed successfully.


C:\Users\XFF>dbv file=h:/xifenfei/undo01.dbf

DBVERIFY: Release 11.2.0.4.0 - Production on 星期日 11 月 16 00:07:29 2025

Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.

DBVERIFY - 开始验证: FILE = H:\XIFENFEI\UNDO01.DBF


DBVERIFY - 验证完成

检查的页总数: 4880
处理的页总数 (数据): 0
失败的页总数 (数据): 0
处理的页总数 (索引): 0
失败的页总数 (索引): 0
处理的页总数 (其他): 4373
处理的总页数 (段)  : 11
失败的总页数 (段)  : 0
空的页总数: 507
标记为损坏的总页数: 0
流入的页总数: 0
加密的总页数         : 0
最高块 SCN            : 467517187 (5693.467517187)
```