DUDE
Where's my other data ?

(or DUDE – Where's my data ? *The sequel*)

- Who am i
  - Kurt Van Meerbeeck
    - Engineer in electronics
    - Working with Java since 1996 (jdk 1.0.2)
    - Working with Oracle products since 1997 (Oracle 7.3.x, OAS 3.0)
  - Currently work for AXI NV/BV
    - Oracle rdbms & app server
    - IBM DB2 / Netezza / mysql
    - IAM & Security
  - Author of DUDE
    - Data Unloader tool (www.ora600.be)
  - Member of the Oaktable Network
    - www.oaktable.net
  - Co-Founder of the MDF Table Network
    - www.mdftable.net

Agenda

Data unloaders

What ?

How ?

Use Cases

Demo

Disclaimer

I don't do blockdumps in this presentation

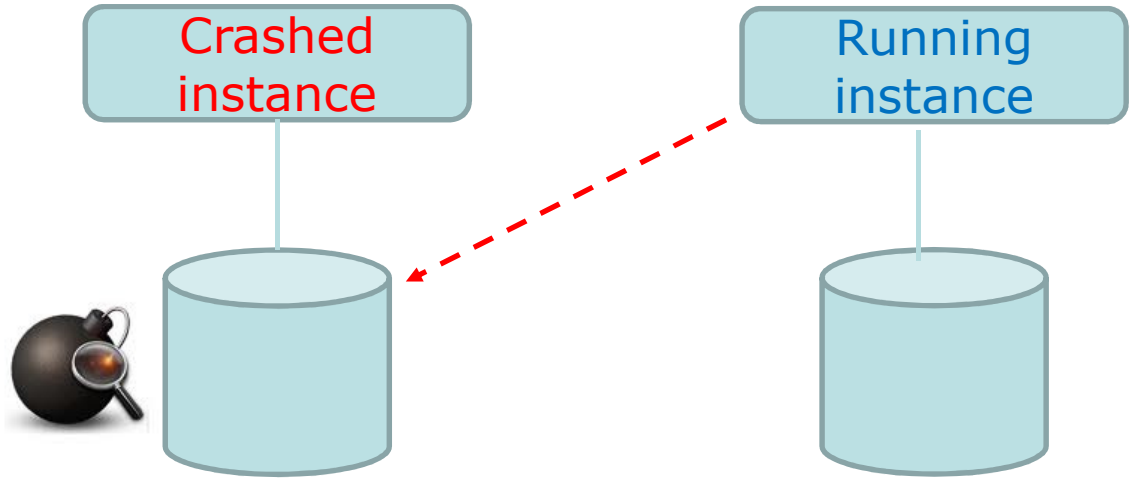
I will not open database files in a hexeditor

Disclaimer

But if you like that sort of thing,
Here's a tip

Alter system dump datafile '/x/y/z.dbf' block ...

It can dump blocks from datafiles from foreign Oracle databases !



What are data unloaders ?

- What are you talking about ...
  - unload data ?
  - not in the sense of ETL

- Imagine your production DB crashed
  - unrecoverable
  - corrupt
  - inconsistant datafiles
  - loss of system tablespace
  - missed something at Backup & Recovery course
  - and your backup scripts weren't as cool as you thought they were

You've tried everything ...
Database can't be opened ...
WHAT DO YOU DO ?!?

What are data unloaders ?

- Panic
- Cry
- Take up smoking again ...
- Call the wife – it'll be long night ... Again ...
- Oracle support
  - spend the next 30min trying to open a severity 1 SR
  - *call* them

What are data unloaders ?

Depending on support contract / country

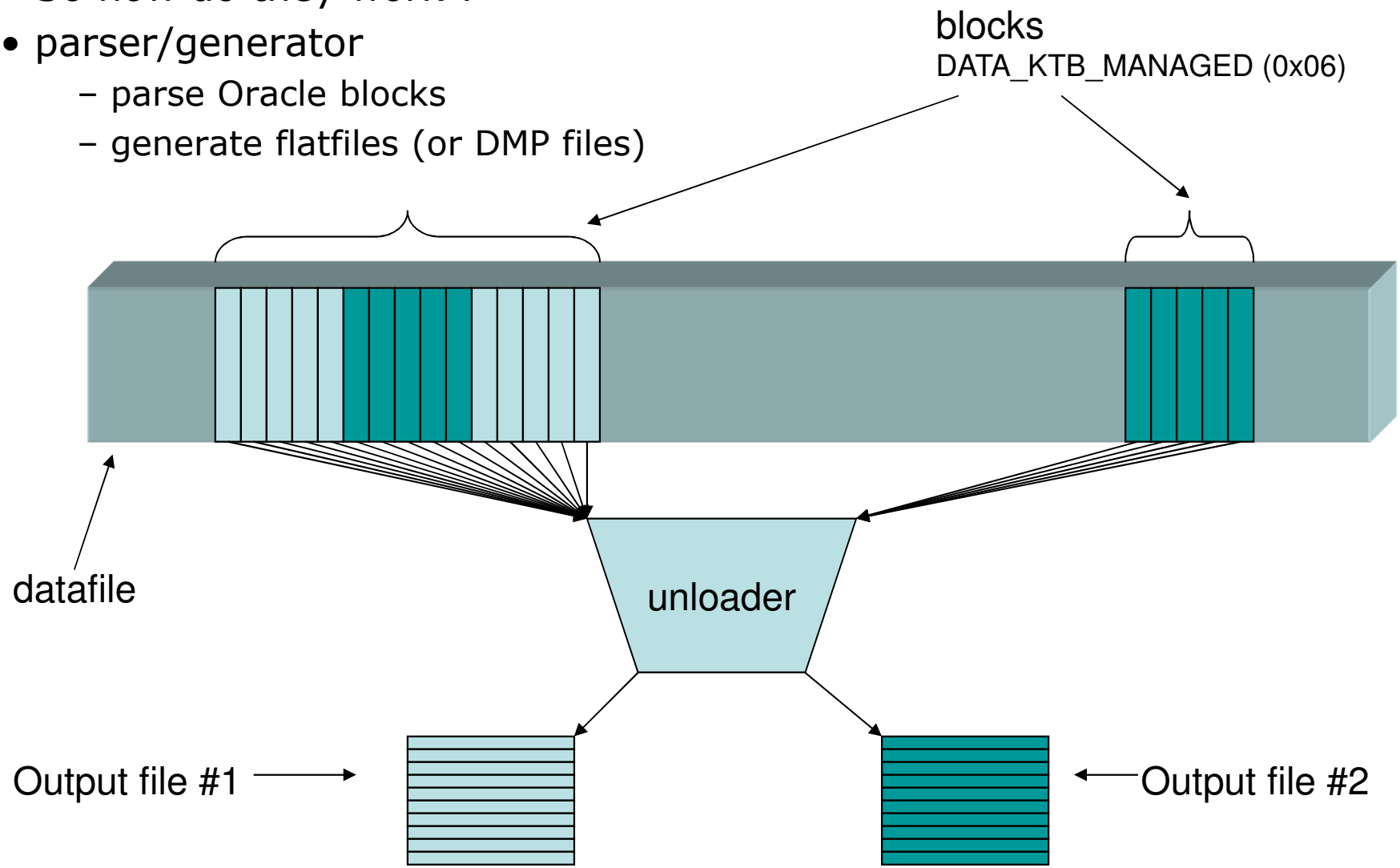
1. you’re screwed
2. You’re screwed – but maybe we can help you

What are data unloaders ?

- So what can support do what you can't ?
  - Hey – I've taken the Backup&Recovery course... Did I miss something ?
  - DUL (Data UnLoader)
    - the myth, the legend, the Holy Grale of Oracle data recovery
    - Extracts data without the instance being up
    - written by Bernard van Dujnen, Oracle, The Netherlands (1994)
    - not a public tool -> Oracle consulting
    - www.petefinnigan.com : links to DUL's user & config guide
  - DUDE (database unloading by data extraction)
    - Written in *java* around 2000
    - Intel, HPUX, IBM AIX, OpenVMS/alpha, Novell, Sun Solaris/SunOS, Tru64
    - Oracle 7-11

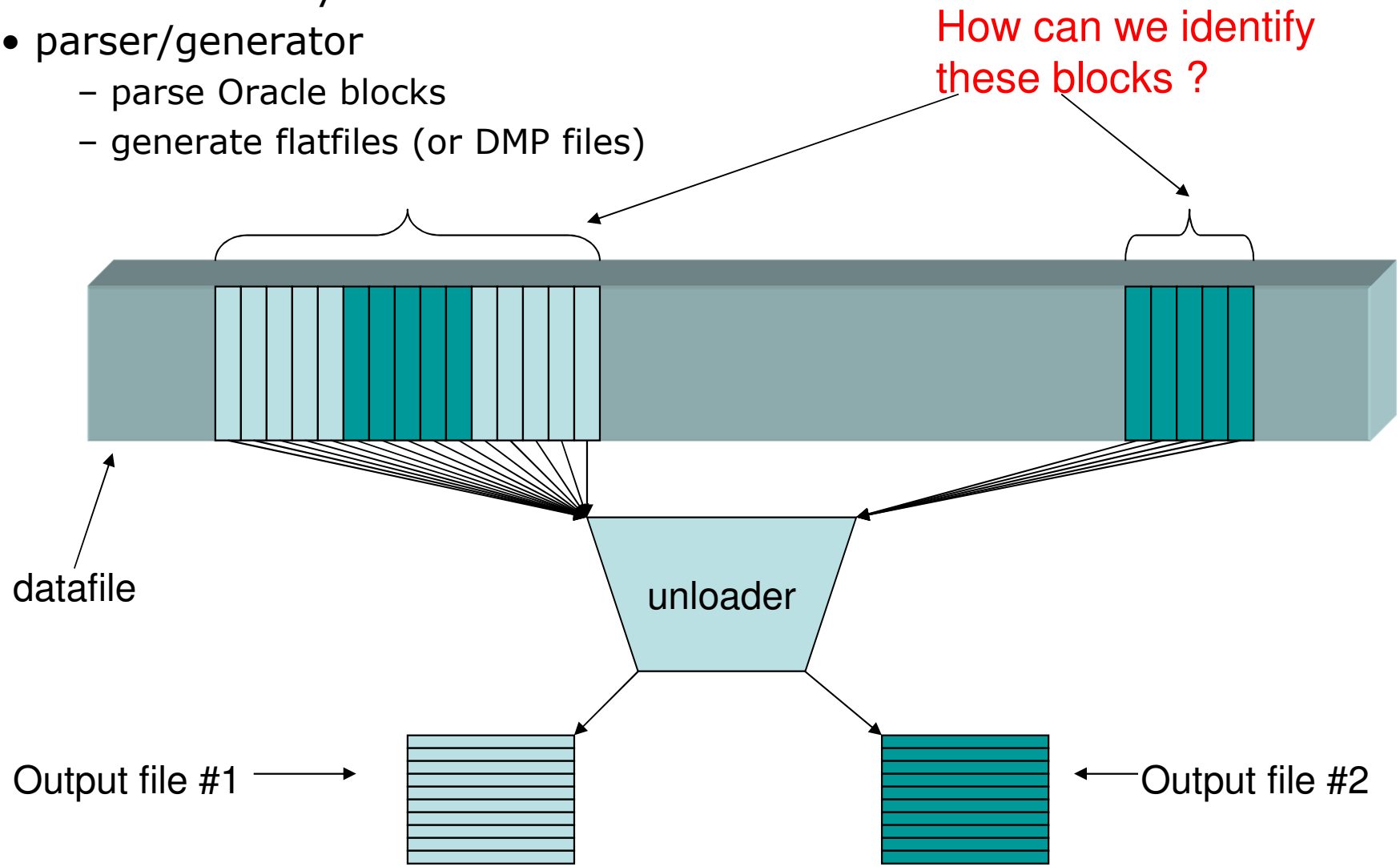
What are data unloaders ?

- So how do they work ?
- parser/generator
  - parse Oracle blocks
  - generate flatfiles (or DMP files)



What are data unloaders ?

- So how do they work ?
- parser/generator
  - parse Oracle blocks
  - generate flatfiles (or DMP files)



Identifying datablocks

- Easy - segment header !
- And then follow the pointers to the extent maps !

- *But how can we find the block containing the segment header ?*
- Look it up in SEG$ (file#, block#) part of c_file#_block# !

- *But how do we know the dataobjectid c_f#_b# and tablenumber of SEG$?*
- Look it up in bootstrap$!

- *And were can we find bootstrap$?*
- DBA can be found in SYSTEM fileheader

Identifying datablocks
Prepare for the worst !

What if

Fileheader is corrupt

Bootstrap$ is corrupt

Dictionary is corrupt

Segment header/extentmap is corrupt

Identifying datablocks
THE BLOCKMAPPER ROUTINE

"You Probably Don't need extentmaps"

1. **Scan everyblock**
2. **Note down file#, offset#**
3. **Group by dataobjectid**
4. **For select number of blocktypes**

Identifying datablocks
THE BLOCKMAPPER ROUTINE

```
DUDE> create blockmap for tablespace USERS ;
DUDE> ID := 0 BLOCKMAPPER for TABLESPACE NAME = USERS
OFFSET = 0
ASSM = true
BIGFILE = false
BLOCKSIZE = 2048
NUMBER = 11 FILENAME = c:\dudecase\users_01.dbf IBS = 2048


DUDE> Datafile c:\dudecase\users_01.dbf : start reading blocks ...
DUDE> Datafile c:\dudecase\users_01.dbf : reading blocks done !
DUDE> Block type 0 : 16492 blocks -> 32984Kb -> 9%
DUDE> Block type 6 : 151636 blocks -> 303272Kb -> 85%
DUDE> Block type 11 : 1 blocks -> 2Kb -> 0%
DUDE> Block type 29 : 1 blocks -> 2Kb -> 0%
DUDE> Block type 30 : 30 blocks -> 60Kb -> 0%
DUDE> Block type 32 : 2682 blocks -> 5364Kb -> 1%
DUDE> Block type 33 : 176 blocks -> 352Kb -> 0%
DUDE> Block type 35 : 173 blocks -> 346Kb -> 0%
DUDE> Block type 36 : 18 blocks -> 36Kb -> 0%
DUDE> Block type 40 : 7992 blocks -> 15984Kb -> 4%
DUDE> Total Blocks scanned = 179201
DUDE> Please wait for output stream to finish ...
DUDE> Please wait for chunk stream to finish ...
DUDE> Done !
```

Identifying datablocks
THE BLOCKMAPPER ROUTINE

Makes it possible to recover

- **Truncated tables**
- **Dropped tables**

Identifying datablocks

"You Probably Don't need BOOTSTRAP$"

The order of commands in sql.bsq
defines the dataobjectid's
for the base dictionary table
('mig' utility)

Scanning datablocks

"You Probably Don't need a SYSTEM tablespace"

Allthough it's mighty handy as it contains the base dictionary tables ☺

THE BLOCKSCAN ROUTINE

Scanning datablocks

THE BLOCKSCAN ROUTINE

1. **Scan every/most datablocks**
2. **Sample row**
3. **Note down 'nice numbers'**
4. **Note down 'possible date' datatypes**
5. **Note down '% printable chars'**
6. **Create an unload command**
7. **Maybe modify unload command (lobs, timestamp)**

Scanning datablocks

```
DUDE> scan objectid 75901 ;
DUDE> Reading blocks done !
DUDE> Please wait for output stream to finish ...
DUDE> Number of rows skipped : 0
DUDE>
Rows sampled : 306
COL 0 : LEN = 4 : NULLS = 0% : POSS. DATE = 0% : PRINTABLE = 33% : NICE NUM = 100%
COL 1 : LEN = 3 : NULLS = 0% : POSS. DATE = 0% : PRINTABLE = 100% : NICE NUM = 0%
COL 2 : LEN = 27 : NULLS = 0% : POSS. DATE = 0% : PRINTABLE = 100% : NICE NUM = 0%
COL 3 : LEN = 2 : NULLS = 0% : POSS. DATE = 0% : PRINTABLE = 0% : NICE NUM = 100%
COL 4 : LEN = 30 : NULLS = 0% : POSS. DATE = 0% : PRINTABLE = 95% : NICE NUM = 0%
COL 5 : LEN = 2 : NULLS = 0% : POSS. DATE = 0% : PRINTABLE = 0% : NICE NUM = 100%
COL 6 : LEN = 55 : NULLS = 0% : POSS. DATE = 61% : PRINTABLE = 100% : NICE NUM = 0%
COL 7 : LEN = 1 : NULLS = 100% : POSS. DATE = 0% : PRINTABLE = 0% : NICE NUM = 0%
COL 8 : LEN = 1 : NULLS = 100% : POSS. DATE = 0% : PRINTABLE = 0% : NICE NUM = 0%
COL 9 : LEN = 1 : NULLS = 0% : POSS. DATE = 0% : PRINTABLE = 0% : NICE NUM = 100%

DUDE> Done !
```

Scanning datablocks

```
dump SCANNED OBJECTID 75901 (   COL0 NUMBER ,
                                COL1 CHAR ,
                                COL2 CHAR ,
                                COL3 NUMBER ,
                                COL4 CHAR ,
                                COL5 NUMBER ,
                                COL6 CHAR ,
                                COL7 CHAR , → 100% NULL
                                COL8 CHAR , → 100% NULL
                                COL9 NUMBER ) ;

DUDE> show objectid 75901 ;
DUDE> OWNER = TECH NAME = TYPED_VIEW$
 OBJECTID = 75901 DATA OBJECTID = 75901
COL 1 [1] OBJ# NUMBER
COL 2 [2] TYPEOWNER VARCHAR2 [WE8MSWIN1252]
COL 3 [3] TYPENAME VARCHAR2 [WE8MSWIN1252]
COL 4 [4] TYPETEXTLENGTH NUMBER
COL 5 [5] TYPETEXT VARCHAR2 [WE8MSWIN1252]
COL 6 [6] OIDTEXTLENGTH NUMBER
COL 7 [7] OIDTEXT VARCHAR2 [WE8MSWIN1252]
COL 8 [8] TRANSTEXTLENGTH NUMBER
COL 9 [9] UNDERTEXT VARCHAR2 [WE8MSWIN1252]
COL 10 [10] UNDERTEXTLENGTH NUMBER
COL 11 [11] TRANSTEXT LONG ----------→ trailing NULL
Table Properties :
->pdml itl invariant
```

Scanning datablocks

Blockscan routine can also look for ****orphaned*** segments**

Quite handy when looking for dropped tables/partitions!

Other helpful routines when looking for dropped/deleted data :
INCLUDE_DROPPED=«true»
DELETED_ONLY=«true»

Scanning datablocks

```
dump SCANNED OBJECTID 75901 (    COL0 NUMBER ,
                                 COL1 CHAR ,
                                 COL2 CHAR ,
                                 COL3 NUMBER ,
                                 COL4 CHAR ,
                                 COL5 NUMBER ,
                                 COL6 CHAR ,
                                 COL7 CHAR ,
                                 COL8 CHAR ,
                                 COL9 NUMBER ) ;
```

"You probably don't need table and column names"

A developer with an elephant memory will do !

Unloading LOBs

"You probably don't need a LOB index"

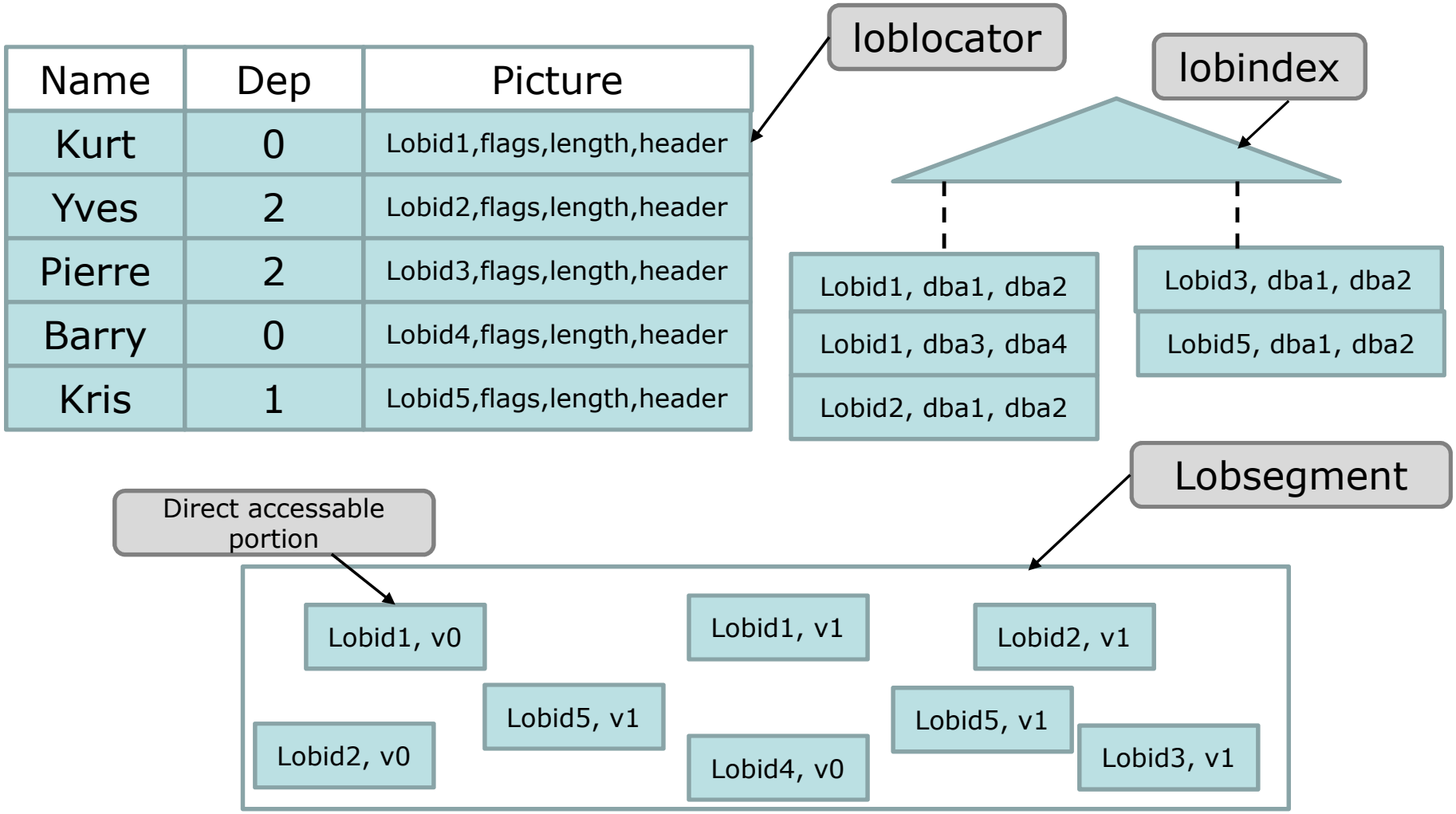
DUDE has 3 algorithm to unload LOBs

1. **LOB index**
2. **In-memory high version chunk hash**
3. **On-disk high version chunk hash**

DUDE also supports SECUREFILES (normal/compressed)

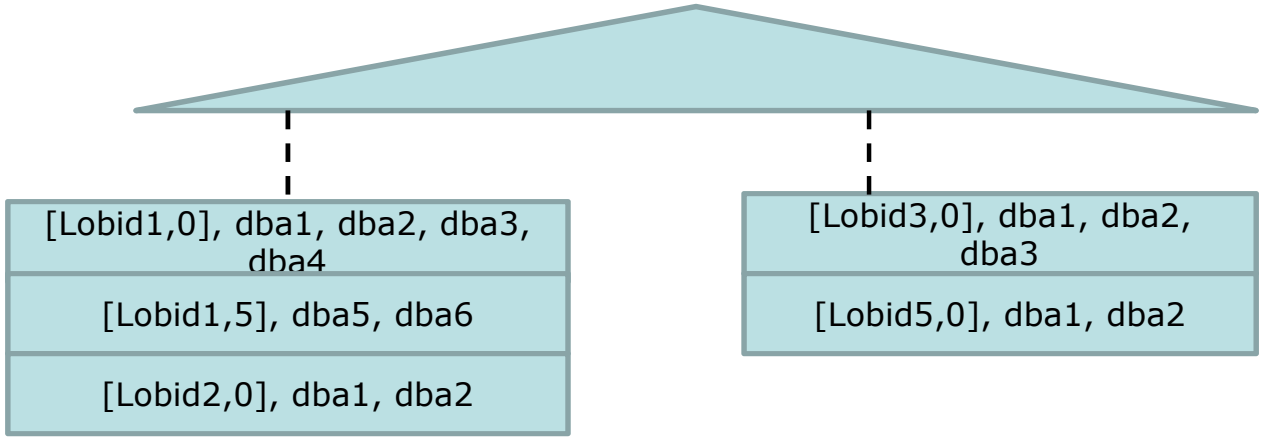
Unloading LOBs

Using the LOB index (if you insist)



Unloading LOBs

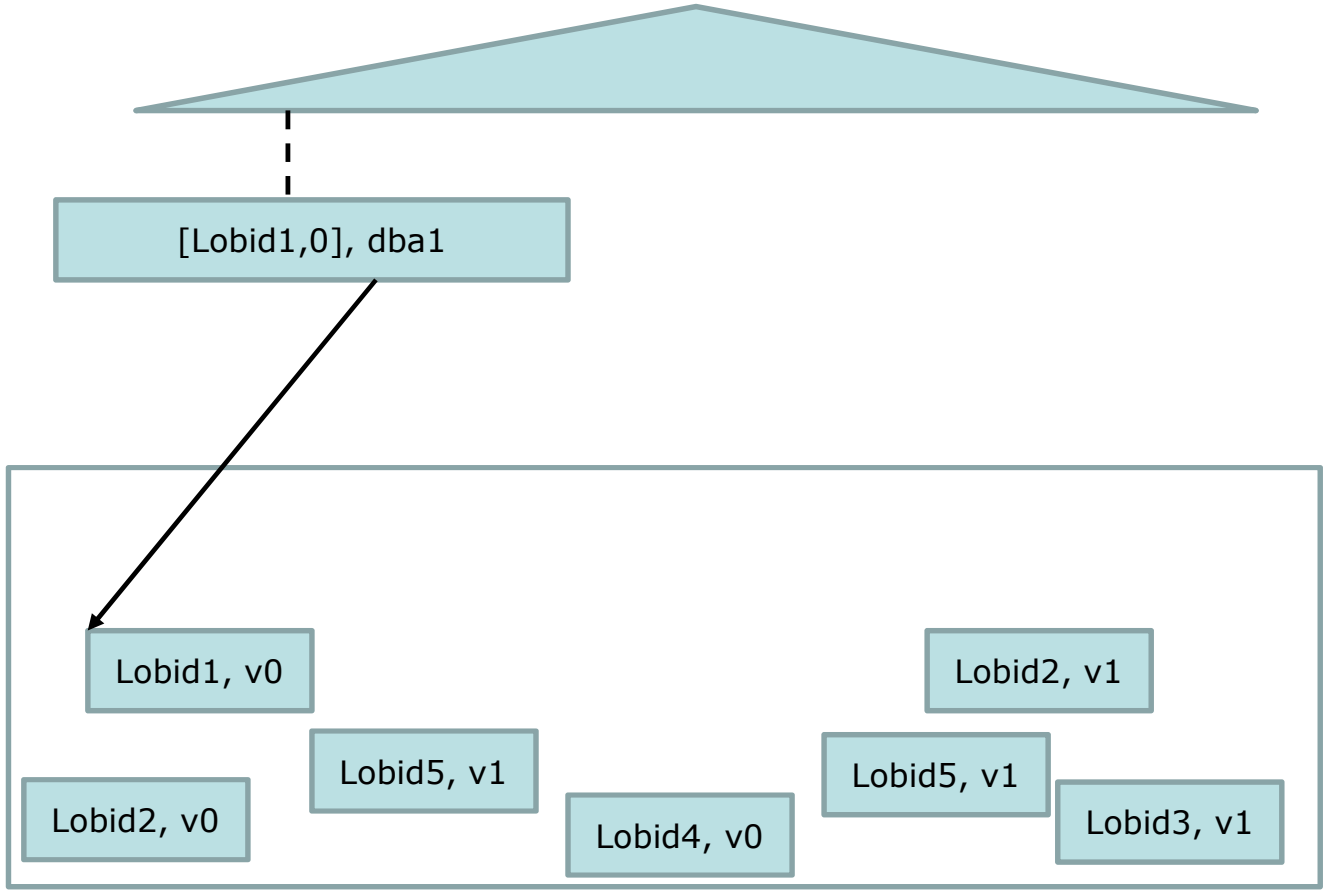
LOB index is special



1. **An entry can contain multiple pointers instead of one rowid**
2. **Index key is lobid+relative position in lob**
   Allows entries across multiple leaf blocks
3. **Maintains the reusable block list !**

Unloading LOBs

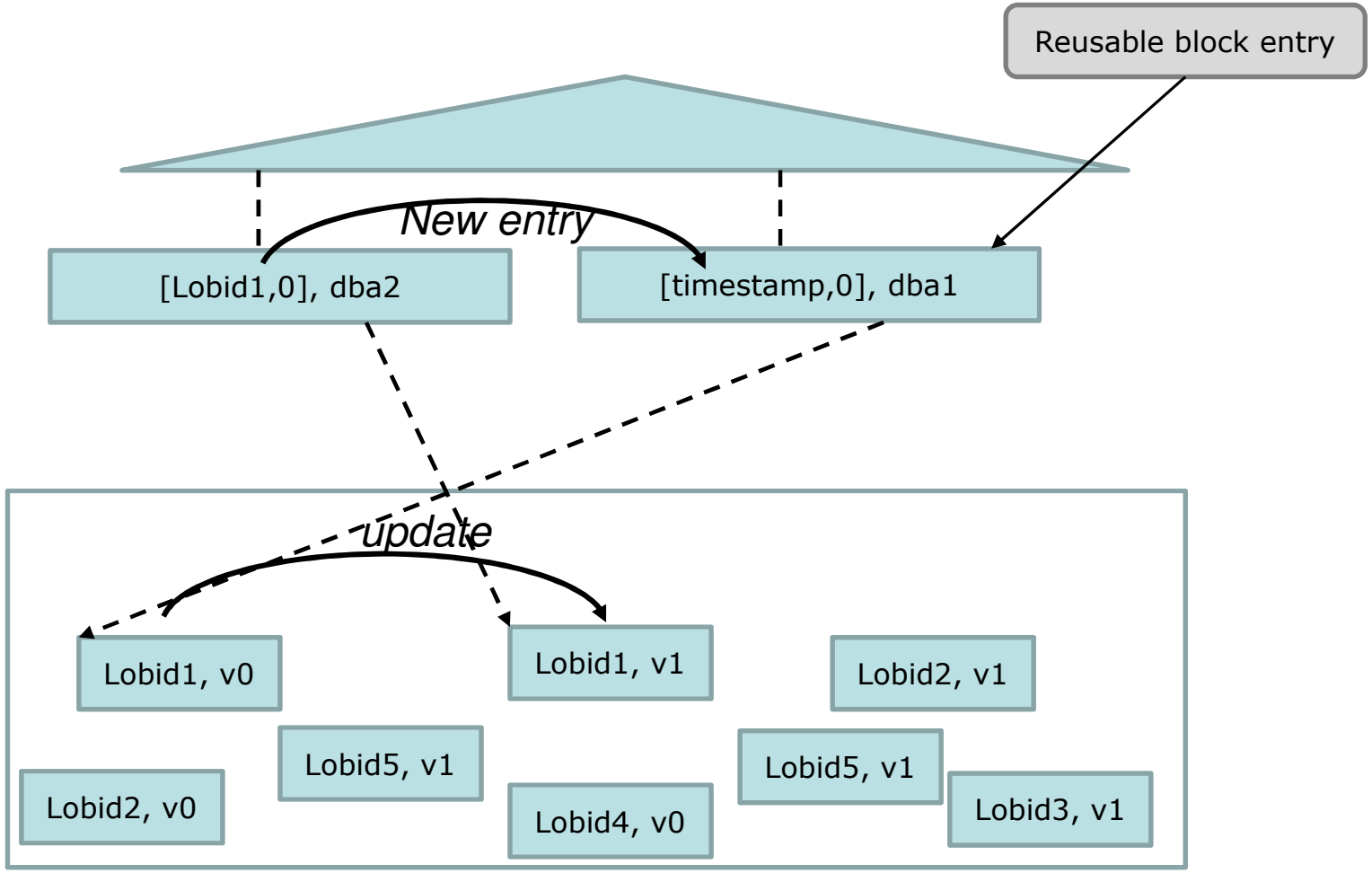
The reusable block/chunk list



Unloading LOBs

The reusable block/chunk list

[pctversion/retention]



Unloading LOBs

The reusable block/chunk list

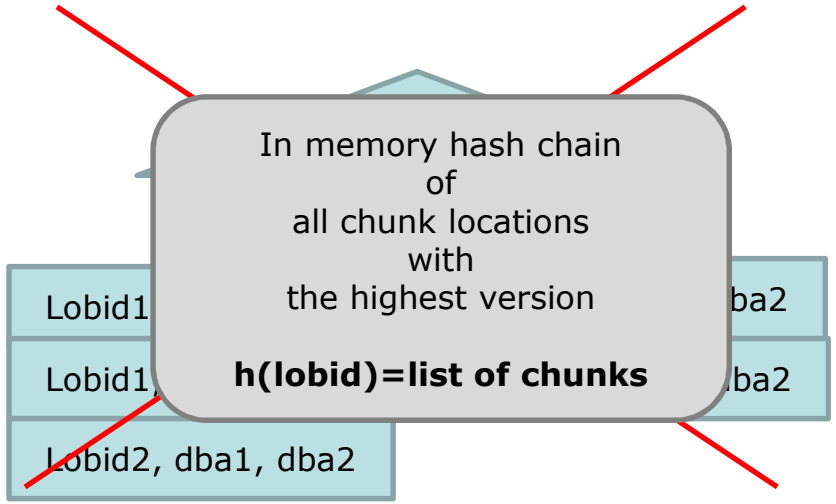
[pctversion/retention]

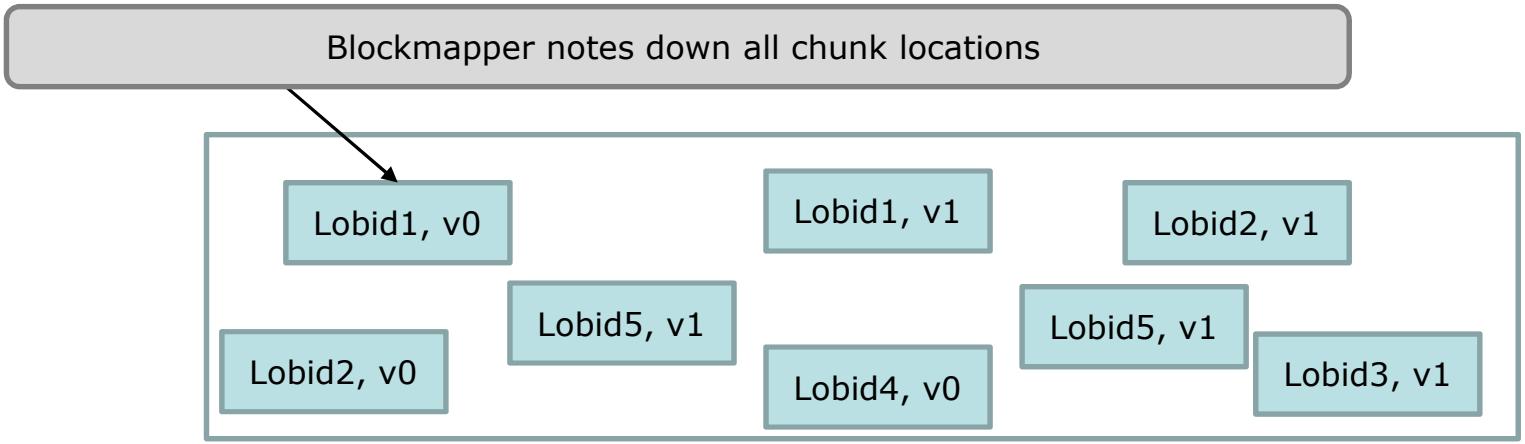
- **Reusable block entries are index entries with pointers to storage that may be reused**
- **On commit, a new reusable block entry is created where the LOBID key part is replaced by a relative timestamp**
- **ORA-22924 – snapshot too old … old chunkspace already reclaimed while still needed for read consistency**

Unloading LOBs

Avoiding LOB index (corruption)

Name	Dep	Picture
Kurt	0	Lobid1,flags,length,header
Yves	2	Lobid2,flags,length,header
Pierre	2	Lobid3,flags,length,header
Barry	0	Lobid4,flags,length,header
Kris	1	Lobid5,flags,length,header





Unloading LOBs

Avoiding LOB index (corruption)

Downsides of in-memory hash

1. **Time consuming**
   - *– blockmap all datafiles first*
   - *- Hashing*

2. **Resource consumption**
   - **Memory constraints**
   - **32bit java - ~2GB process memory**

Unloading LOBs

Avoiding LOB index (corruption)
On-disk hashing

To avoid the 2GB limit in 32bit environments

(human DNA strand in a CLOB ~1GB)
(hashing 70million fingerprints)

SLOWWWW!

Recovering from RAID5 Failure

RAID **recovery** Software - Win NT/2000/XP/2003/Vista
www.r-studio.com/
Download and try Free Demo

Raid **recovery** utility - **Recover** failed RAID 0 and 5
www.quetek.com/
Win 7, Vista, XP, 2008, 2003, 2000
Testimonials - Download - Buy - RAID/NAS recovery services

RAID 5 Data **Recovery** FAQ
www.vantagetech.com/faq/**raid-5-recovery**-faq.html
RAID 5 Data **Recovery** FAQ - Extensive information regarding data loss and **recovery** for **RAID 5** volumes from RAID **recovery** professionals. **... USA**, The Data **Recovery** Hotline: 1-800-ITS-LOST Toll Free (**US** & Canada): 800.487.5678 **...**

Raid 5 Data **Recovery**: Emergency **Raid 5** Server Repair by Secure **...**
www.securedatarecovery.com › ... › Raid Data Recovery
★★★★★ Rating: 5 - Review by Westside Union School District - Dec 2, 2011
We perform **RAID 5 Recovery** for all major RAID manufacturers, using specialized equipment, and **...** Call **us** to talk to a RAID **Recovery** Specialist Now **...**

RAID 5 Recovery Online 24/7
www.raid**recovery**online.com/**raid_5_recovery**/
Remote **RAID 5 Recovery** services and solutions 24/7, provided by RAID **... RAID 5 Recovery** (Striped Disks With Distributed Parity) **... USA**: (+1) 415-2870588 **...**

Recovering from RAID5 Failure

- **Customer case**
- ***Lost 2 drives in RAID5 array***
- ***Backup to disk ...*** *someone had taken the backup server !!!*
- ***Took array to specialized company and got data back ...***
- ***Database did not start*** ☺

Recovering from RAID5 Failure

- ***Data contained garbage data***
- ***Unfortunately – backup was zipped***
- ***Unload around garbage data***
  - *FLAG_FRACTURED_BLOCKS*
  - *SKIP_FRACTURED_BLOCKS*
  - *SKIP_OUT_OF_SYNC*
  - *SKIP_IF_NOMATCH_CC*
  - *_SKIP_MAX_ROWS_PER_BLOCK*
- ***Use sqlldr -> badfile***

Conclusion - unloaders

- a last resort to recover your data
  - missing archivelogs
  - corruption of data dictionary or bootstrap objects
  - orphaned datafiles (or loss of system tablespace)
  - dropped tablespaces
  - truncated/dropped tables
  - dropped columns
- no guarantee
  - there’s a reason why your DB doesn’t open
  - read consistency
  - zero’d out blocks

Questions

DEMO TIME