Three Types of Table Compression, Part 1

A tale about a room with two doors in plain view, and a hidden forgotten third door…

hrvatska udruga oracle korisnika

Tim Gorman

Delphix

Thursday, 16-Oct 2014

Agenda

- The story behind the story
- Overview of data compression
- Overview of table data compression in Oracle database
  - Review of related concepts within Oracle database
    - Internal block and row formats
    - Cluster tables, row-chaining, and direct-path loads
- Details of BASIC/OLTP and HCC table compression
  - De-duplication compression (basic and OLTP)
  - Hybrid Columnar Compression (HCC)
- Trailing NULL columns
  - The rest of the story

Today's agenda

- The story behind the story
- Overview of data compression
- Overview of table data compression in Oracle database
  - Review of related concepts within Oracle database
    - Internal block and row formats
    - Cluster tables, row-chaining, and direct-path loads
- Details of BASIC/OLTP and HCC table compression
  - De-duplication compression (basic and OLTP)
  - Hybrid Columnar Compression (HCC)
- Trailing NULL columns
  - The rest of the story

Tomorrow's agenda

- The story behind the story
- Overview of data compression
- Overview of table data compression in Oracle database
  - Review of related concepts within Oracle database
    - Internal block and row formats
    - Cluster tables, row-chaining, and direct-path loads
- **Details of BASIC/OLTP and HCC table compression**
  - **De-duplication compression (basic and OLTP)**
  - **Hybrid Columnar Compression (HCC)**
- **Trailing NULL columns**
  - **The rest of the story**

The Story Behind The Story

- This isn't a presentation about table compression
  - It ended up that way, however
- Instead, this began as a story about a solution to a specific problem
  - It was a lot of fun
  - I wanted to share it
  - But I had to fill in a lot of background before getting to the punch line
  - Which seems to make this a presentation about compression
  - Please bear with me for the next 59 minutes?

Data Compression

- White paper: *Introduction to Data Compression*
  - Guy E Blelloch, Carnegie-Mellon University, 25-Sep 2010
    - http://www.cs.cmu.edu/afs/cs/project/pscico-guyb/realworld/www/compression.pdf
- Lempel Ziv (LZ) lossless compression methods
  - Simplified generic LZ algorithm
    - Divides source into fixed-length (i.e. 10- or 12-bit) *patterns*
    - Stores distinct *patterns* in lookup table
    - Replaces *patterns* in output stream with lookup hash value
  - Variations on LZ methods
    - DEFLATE: focuses on speed (zip, gzip, LZO, ZLIB, etc)
    - Layered compression: focuses on compression ratio, relatively slow, uses several layers of compression techniques (BZIP2)

Compression in Oracle

- Index compression
- Table compression
  - Basic
  - OLTP*
- RMAN backup compression*
- SecureFile (LOB) compression*
- Data Pump export compression*
- Data Guard redo transport compression*
- Hybrid Columnar compression*

* Advanced Compression Option
* Exadata / ZFS / Pillar storage only

Oracle8i
Oracle9i
Oracle10g
Oracle11gR1
Oracle11gR2

Compression in Oracle

- Index compression
- Table compression
  - Basic
  - OLTP*
- RMAN backup compression*
- SecureFile (LOB) compression*
- Data Pump export compression*
- Data Guard redo transport compression*
- Hybrid Columnar compression*

* Advanced Compression Option
* Exadata / ZFS / Pillar storage only

Oracle8i
Oracle9i
Oracle10g
Oracle11gR1
Oracle11gR2

Table Compression

```
CREATE TABLE ...
   COMPRESS [ FOR DIRECT_LOAD OPERATIONS | BASIC ]
   COMPRESS FOR ALL OPERATIONS | COMPRESS FOR OLTP
   COMPRESS FOR QUERY [ LOW | HIGH ]
   COMPRESS FOR ARCHIVE [ LOW | HIGH ]
```

Key

- Oracle9i +
- Oracle11gR1
- Oracle11gR2 +

COMPRESS [BASIC]

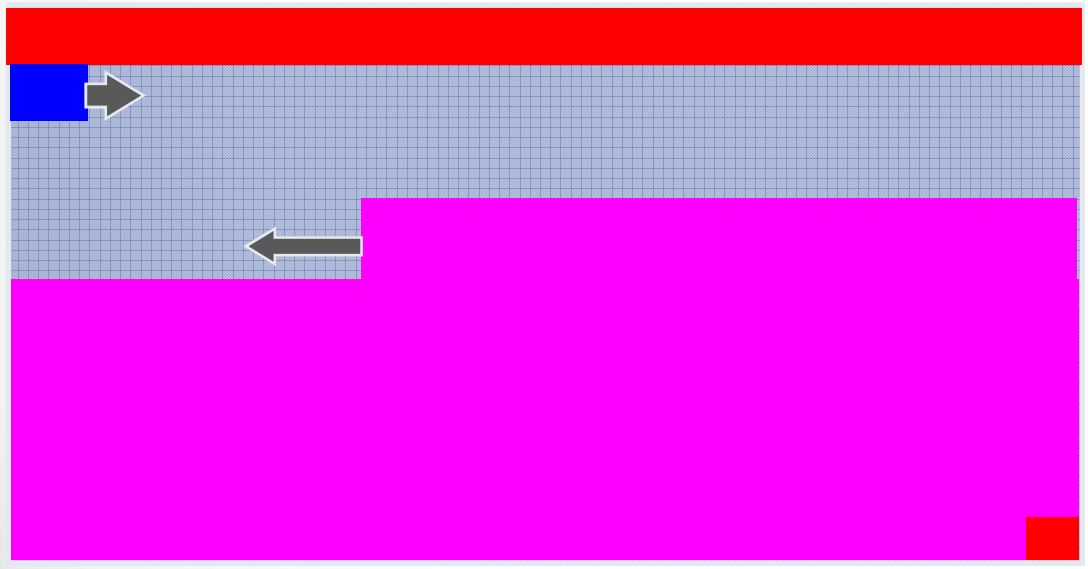
- Similar in concept to LZ algorithm
  - Distinct column values stored in symbol table within block
  - Column values replaced by offset value into symbol table
- Initial Oracle table compression implementation
  - No extra cost with Enterprise Edition, not available in Standard Edition
  - Enabled with `COMPRESS` in 9i and 10g, `COMPRESS [ FOR DIRECT_LOAD OPERATIONS ]` in 11gR1, `COMPRESS [ BASIC ]` from 11gR2 onward
  - Available only during *direct-path* bulk-loading operations
- Restrictions and limitations
  - Not supported for:
    - tables with more than 255 columns
    - index-organized tables (IOTs)
    - table clusters
  - ALTER TABLE .. DROP COLUMN not supported
    - Can only SET UNUSED

COMPRESS FOR OLTP

- Advanced compression option
  - Additional licensing required in addition to Enterprise Edition
  - Enabled with `COMPRESS FOR ALL OPERATIONS` added in 11gR1
    - Later renamed to `COMPRESS FOR OLTP` in 11gR2
  - Allows all types of conventional and direct-path DML
    - Compression triggered when block FULL encountered
- Restrictions and limitations
  - Not supported for:
    - tables with more than 255 columns
    - index-organized tables (IOTs)
    - table clusters
  - Migrated chained rows will be compressed
    - But rows chained due to row-length exceeding block size will not
  - Required List of Critical Patches
    - Support note #1061366.1

Block Format

- Database block layout illustration



Header & Tailer (*fixed size*)

Interested Txn List (ITL)

Free space

Uncompressed rows

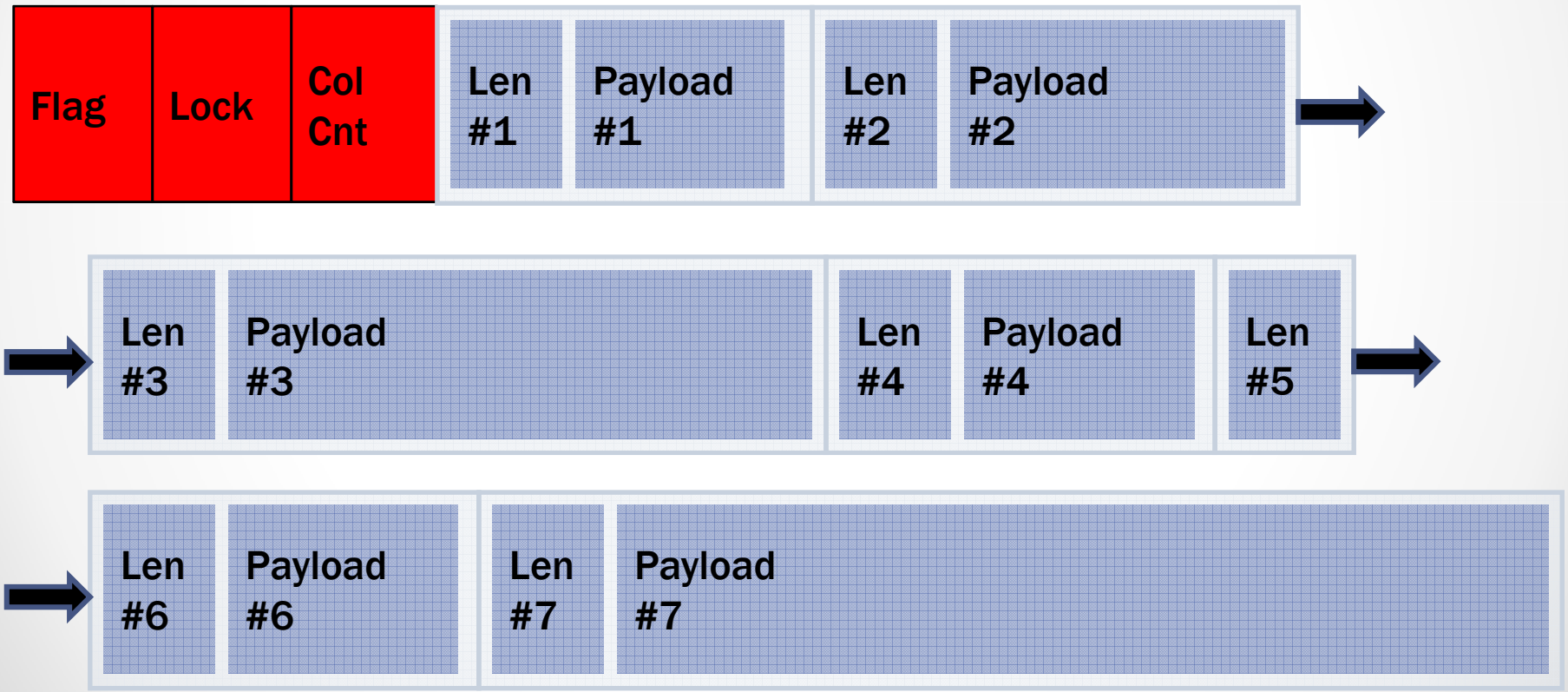
Block Format

- Header
  - Fixed header (110 bytes)
    - KCBH: Type, hdr, RDBA, SCN Base/Wrap, Seq, Flag, Chksum,(20 bytes)
    - KTBBH: Transaction Fixed Header (72 bytes)
    - KDBH: Data Header Structure (14 bytes)
    - KDBT: Table Directory Entry (4 bytes)
  - Interested Transaction List or ITL
    - XID, UBA, flag, lock, SCN Base/Wrap(23 bytes)
    - INITRANS <= number of entries <= MAXTRANS
- Free space
  - Header grows outward from beginning, row data grows inward from tail
- Tail
  - Check(4 bytes, fixed)
- Row entries

Row Format

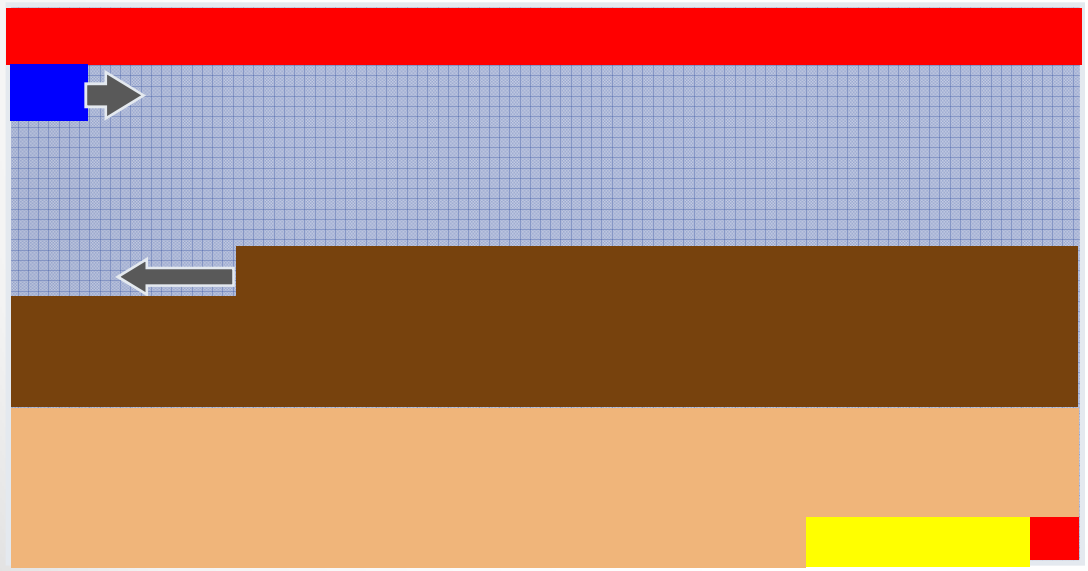
- Row-header
  - Flag :: Lock :: column-count [:: *cluster-key-ID* [:: *chained-ROWID*]]
    - Flag, Lock, column-count = 1 byte each
    - cluster-key-ID
    - chained-ROWID (6-8 bytes)
- Column-piece
  - Length :: data
    - Length <= 254 byres then 1-byte
    - Else length > 254 bytes, then 3-bytes
    - Data
      - DATE = 7 bytes
      - NUMBER = 1 byte exponent plus variable-length mantissa
      - VARCHAR2, CHAR = text
      - NULL data values
        - Non-trailing placeholder = 0xFF
        - Trailing NULLs are not stored

Row Format


Flag
Lock
Col Cnt
Len #1
Payload #1
Len #2
Payload #2
Len #3
Payload #3
Len #4
Payload #4
Len #5
Len #6
Payload #6
Len #7
Payload #7

Cluster Tables

- Tables which share one or more columns
  - Known as cluster key columns
- Rows from clustered tables reside within the same database block
  - Physically pre-joined relational tables



Header & Tailer (*fixed size*)

Interested Txn List (ITL)

Free space

Cluster keys (table 0)

(table 1)

(table 2)

Cluster Tables

```
tab 0, row 0, @0x3f87
tl: 25 fb: K-H-FL-- lb: 0x0  cc: 1
curc: 6 comc: 6 pk: 0x0040db0d.0 nk: 0x0040db0d.0
col  0: [ 5]  c4 04 04 50 24
tab 0, row 1, @0x3f6e
tl: 25 fb: K-H-FL-- lb: 0x0  cc: 1
curc: 18 comc: 18 pk: 0x0040db0d.1 nk: 0x0040db0d.1
col  0: [ 5]  c4 04 04 50 25
```

...several hundred lines edited out for brevity...

```
tab 1, row 0, @0x3a1b
tl: 65 fb: -CH-FL-- lb: 0x0  cc: 20 cki: 0
col  0: [ 4]  c3 05 45 2c
col  1: [ 2]  c1 02
col  2: [ 2]  c1 08
```

...several hundred lines edited out for brevity...

Flag byte showing "cluster key"

Key value

Reference back to cluster key

DUMP traces

- ALTER SYSTEM DUMP command

  DATAFILE [*file#* | '*file-name*']

  BLOCK [*block#* | MIN *block#* BLOCK MAX *block#*]

- Examples in SQL*Plus…

```
SHOW PARAMETER USER_DUMP_DEST
ALTER SESSION SET TRACEFILE_IDENTIFIER = DUMP_DBF;
ALTER SYSTEM DUMP DATAFILE 11 BLOCK 2378;
ALTER SYSTEM DUMP DATAFILE 741 BLOCK MIN 62078 BLOCK MAX 62085;
```

- Finding *file#* and *block#* for an object…
  - View DBA_EXTENTS columns FILE_ID, BLOCK_ID, and (BLOCKS-1)

```
select      'ALTER SYSTEM DUMP DATAFILE '||file_id||
            ' BLOCK MIN '||block_id||' BLOCK MAX '||(block_id-1)||';' txt
from        dba_extents
where       segment_name = 'T1_PK' and segment_type = 'INDEX'
order by    file_id, block_id;
```

Row Chaining

- Rows are chained for three reasons
  - Row migration
    - An UPDATE increases the length of the row so it can no longer fit
    - Only the *row header* is left behind, and *chain-ROWID* points to the location of the row in a different block
  - Row chaining across blocks
    - Row takes more space than database blocks can provide
    - Row is broken into pieces to fit, and chained across blocks
      - Chain-ROWID points to the location of the next chunk
  - Row chaining within blocks
    - Row has more than 255 columns
    - Row is broken into 255-column pieces, and chained within blocks
      - No Chain-ROWID used, row pieces are adjacent within block

Row Chaining

```
tab 0, row 0, @0x3c8a
tl: 766 fb: -----L-- lb: 0x1  cc: 255
col  0: [ 2]  c1 10
col  1: [ 2]  c1 11
col  2: [ 2]  c1 12
```

...several hundred lines edited out for brevity...

```
col 253: [ 2]  c1 13
col 254: [ 2]  c1 14
tab 0, row 1, @0x3bfb
tl: 143 fb: --H-F--- lb: 0x1  cc: 45
nrid:  0x06c1472e.0
col  0: [ 1]  80
col  1: [ 2]  c1 02
col  2: [ 2]  c1 03
```

...several dozen lines edited out for brevity...

```
col 43: [ 2]  c1 0e
col 44: [ 2]  c1 0f
```

- Dump of example table with 300 numeric columns

Direct-path loads

- Bulk loading feature first introduced in Oracle6 FASTLOAD utility on MVS only
  - Compete with DB2 on MVS
  - Incorporated into SQL*Loader DIRECT=TRUE in v7.0
  - Extended to parallel CREATE INDEX in v7.1
  - Extended to CREATE TABLE ... AS SELECT in v7.2
  - Extended to INSERT /*+ APPEND */ in v8.0
  - Enhanced in v8.1 to leave behind a direct-path log for use by MV “fast” refresh
  - Not much enhancement since...

Direct-path loads

- Direct-path operations are always INSERT
  - Never UPDATE or DELETE operations
- Loads data outside of "managed space"
  - During a serial direct-path load operation…
    - loads data above the "high-water mark" in the segment
    - After successful completion, high-water mark is raised to include newly-loaded rows in the table
  - During a parallel direct-path load operation…
    - Loads data into newly-created TEMPORARY segments
    - After successful completion, TEMPORARY segments are merged into the original target segment

Direct-path loads

- Formats new database blocks with inserted row data within private process memory (PGA)
  - Then writes the new and complete database blocks directly to the datafiles
- Largely bypasses many SGA mechanisms
  - Buffer Cache
  - Log Buffer
- Except for changes within data dictionary
  - Object creation and modification is fully recorded in undo and redo

Conventional-path loads

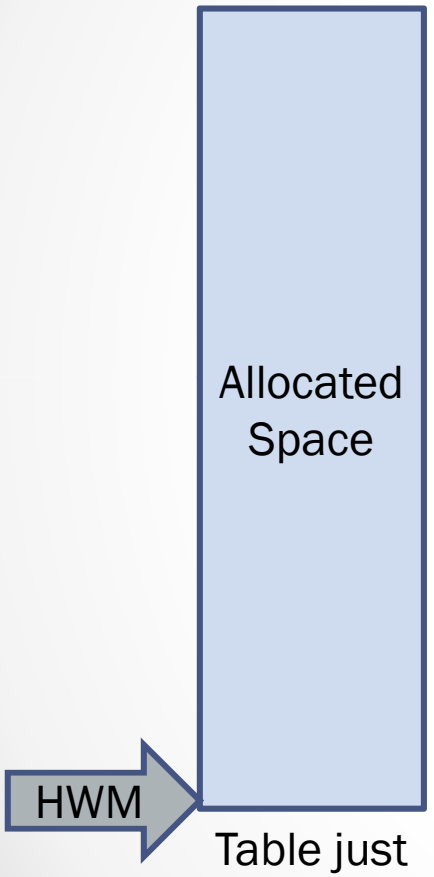
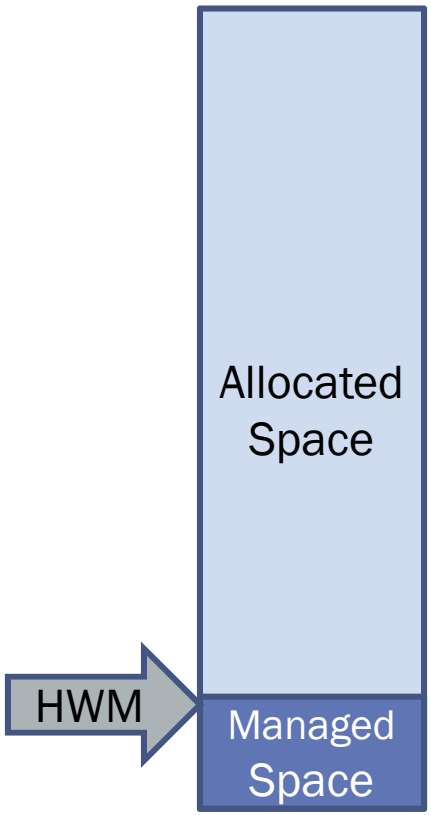
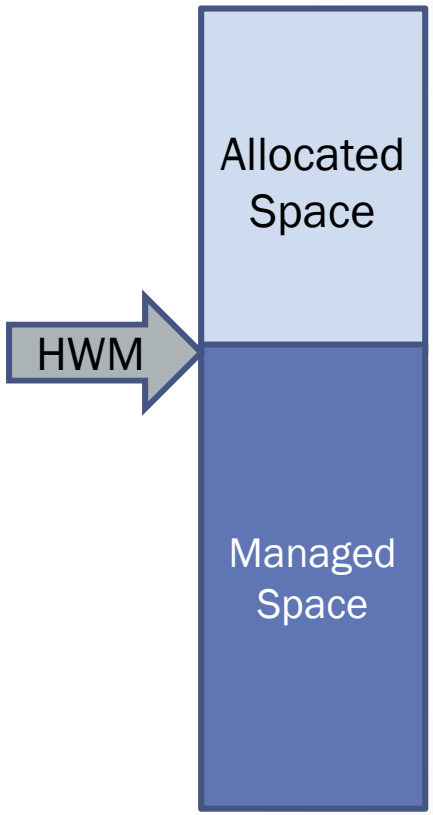


Table just created

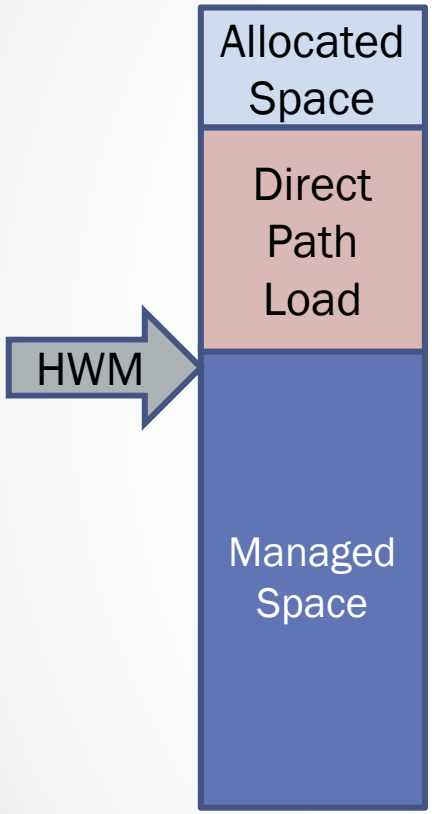


Rows inserted conventionally

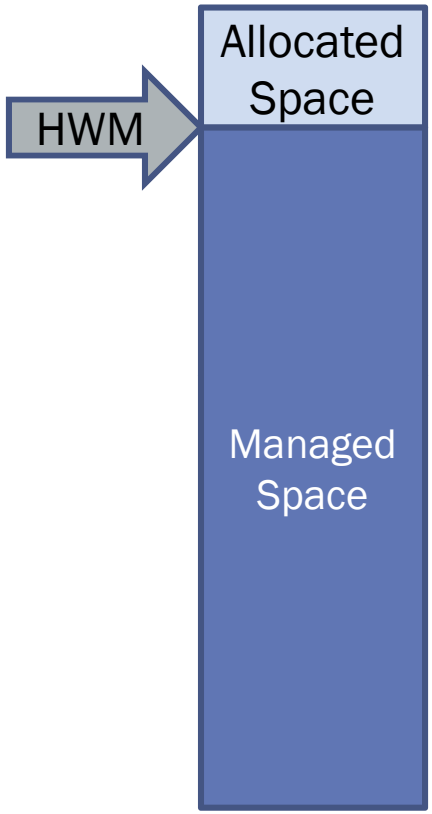


More rows inserted conventionally

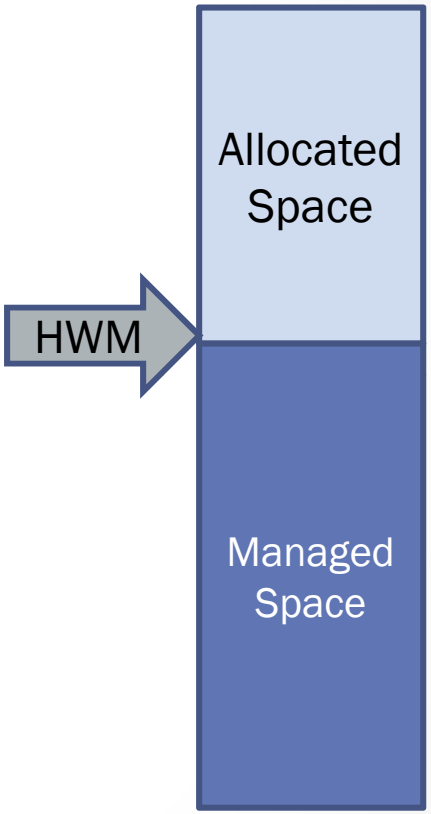
Serial direct-path loads



More rows being inserted using serial direct-path

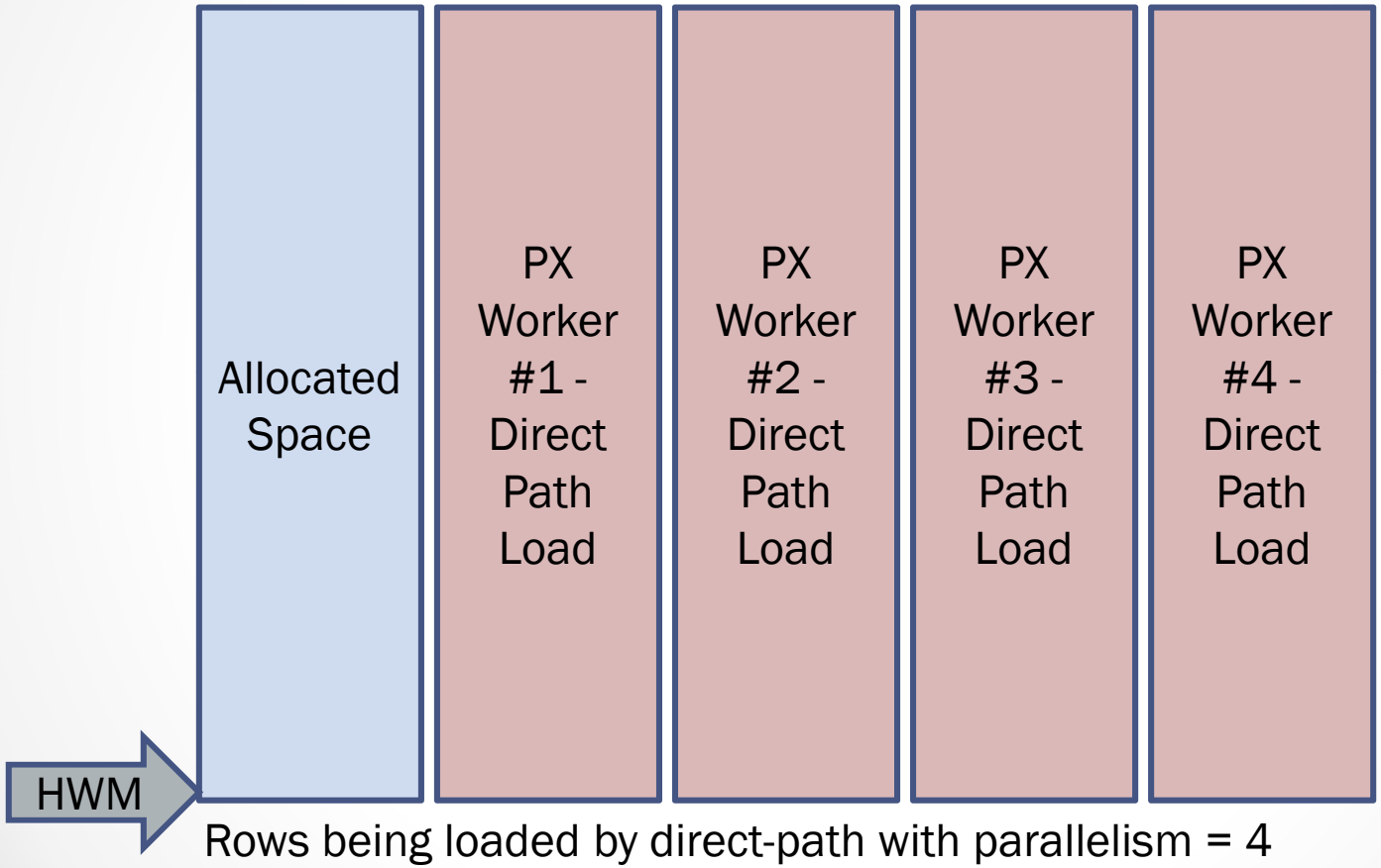


Serial direct-path load **after commit**



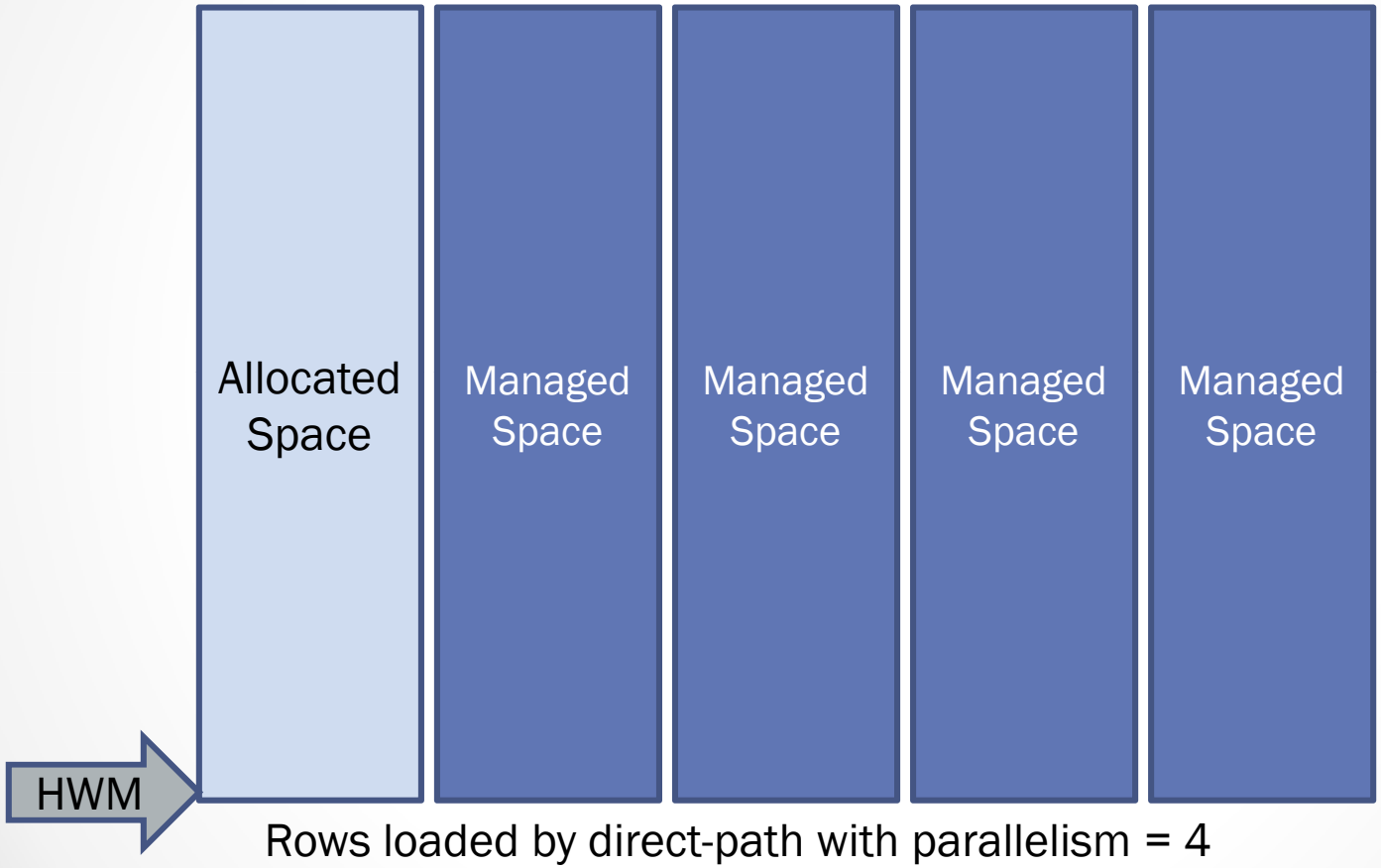
Serial direct-path load **after rollback**

Parallel direct-path loads



Rows being loaded by direct-path with parallelism = 4

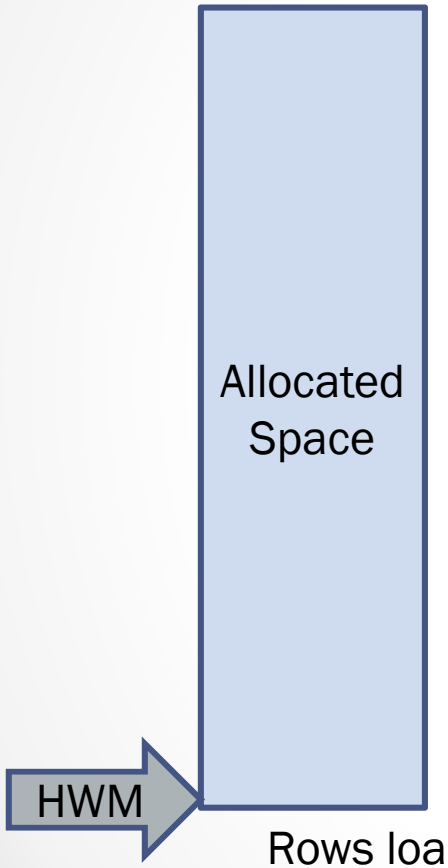
Parallel direct-path loads



Rows loaded by direct-path with parallelism = 4

after commit

Parallel direct-path loads



Rows loaded by direct-path with parallelism = 4
after rollback

Direct-path loads

- Must lock the table/index segment(s) against any other data modifications
  - Until COMMIT or ROLLBACK
- Parallel direct-path loads are very similar to a distributed-database transaction
  - Two-phase commit operation
  - Must COMMIT to resolve in-doubt transactions before the session can do anything else

Summary of "Part One"

- The story behind the story
- Overview of data compression
- Overview of table data compression in Oracle database
  - Review of related concepts within Oracle database
    - Internal block and row formats
    - Cluster tables, row-chaining, and direct-path loads
- Details of BASIC/OLTP and HCC table compression
  - De-duplication compression (basic and OLTP)
  - Hybrid Columnar Compression (HCC)
- Trailing NULL columns
  - The rest of the story

Building up to "Part Two"

- The story behind the story
- Overview of data compression
- Overview of table data compression in Oracle database
  - Review of related concepts within Oracle database
    - Internal block and row formats
    - Cluster tables, row-chaining, and direct-path loads
- **Details of BASIC/OLTP and HCC table compression**
  - **De-duplication compression (basic and OLTP)**
  - **Hybrid Columnar Compression (HCC)**
- **Trailing NULL columns**
  - **The rest of the story**

References

- Oracle11g *Concepts*, http://docs.oracle.com/cd/E14072_01/server.112/e10713/logical.htm#i4894
- Graham Thornton http://www.orafaq.com/papers/dissassembling_the_data_block.pdf

- Email: tim.gorman@delphix.com
- Blog: http://EvDBT.com/
  - Papers: http://EvDBT.com/papers/
  - Scripts: http://EvDBT.com/scripts/
- Twitter: @TimothyJGorman
- Mobile: +1 (303) 885-4526

See you tomorrow morning at 09:00!!!

hrvatska udruga oracle korisnika

- Email: tim.gorman@delphix.com
- Blog: http://EvDBT.com/
  - Papers: http://EvDBT.com/papers/
  - Scripts: http://EvDBT.com/scripts/
- Twitter: @TimothyJGorman
- Mobile: +1 (303) 885-4526

See you tomorrow morning at 09:00!!!