Chapter

11

Storage Management

Topics

- Multiple storage pools
- Data structures
- Storage allocation
- Storage algorithms

DC Multiple Storage Pools - Non-XA

- Up to 128 pools
- Storage pool 0 required
- Storage types (indicated in SMT's SMTTYPE):
  - Shared
  - Shared-kept
  - User
  - User-kept
  - Terminal
  - Database
  - System (storage pool 0 only)

DC Multiple Storage Pools - XA

- Up to 128 pools

  127 user defined (pool numbers 128-254)

  1 system defined (pool number 255)

- Storage types:

  Pools 128-254

  - Shared
  - Shared-kept
  - User
  - User-kept
  - Terminal
  - Database
  - System (pool 255 only)

DC Multiple Storage Pools - XA

A GET STORAGE request will allocate storage from XA storage (above 16M) if and only if all of the following are true:

- The GET STORAGE or #GETSTG request specifies (or defaults to) LOC=ANY
- The current program is running with AMODE(31)
- The task is defined as LOC=ANY

Data Structures

Overview

Storage management data structures are:

- Storage Management Table - SMT (#SMTDS)
- Storage Control Table - SCT (#SCTDS)
- Storage Control Map - SCM
- Storage Frame Prefix
- Storage Frame Suffix

Each data structure is discussed below.

Storage Management Table - SMT (#SMTDS)

The CSASMTA points to the SMT. The SMT contains one entry for each type of storage. Each SMT entry points to a list of valid SCTs for the storage type requested.

Storage Control Table header - SCT (#SCTDS)

One SCT controls the usage of one storage pool. SCTFLAG1 has storage types accommodated by pool. CSASCTA points to the SCT header for storage pool 0. SCTNSCTA points to the SCT for the next storage pool. Zeros in SCTNSCTA indicate there are no more SCTs.

Storage Control MAP - SCM

The SCM is a variable length extension of the SCT which contains a map denoting which 128-byte blocks of the storage pool have been allocated. The SCM immediately follows the SCT, each fullword maps a 4K block of storage. One fullword contains 32 bits.

```
32(bits) x 128(bytes) = 4096 The first bit in the first fullword of SCMs maps the
first 128 bytes of the storage pool; the second bit in the first fullword maps
the next 128 bytes.
```

0 = block is free.

1 = block is used.

Storage Frame Prefix

2 Fullwords at the Beginning of Each Block of Storage

- The first fullword is the address of the RCE for the block of storage.
- The second fullword is the length of the block of storage.

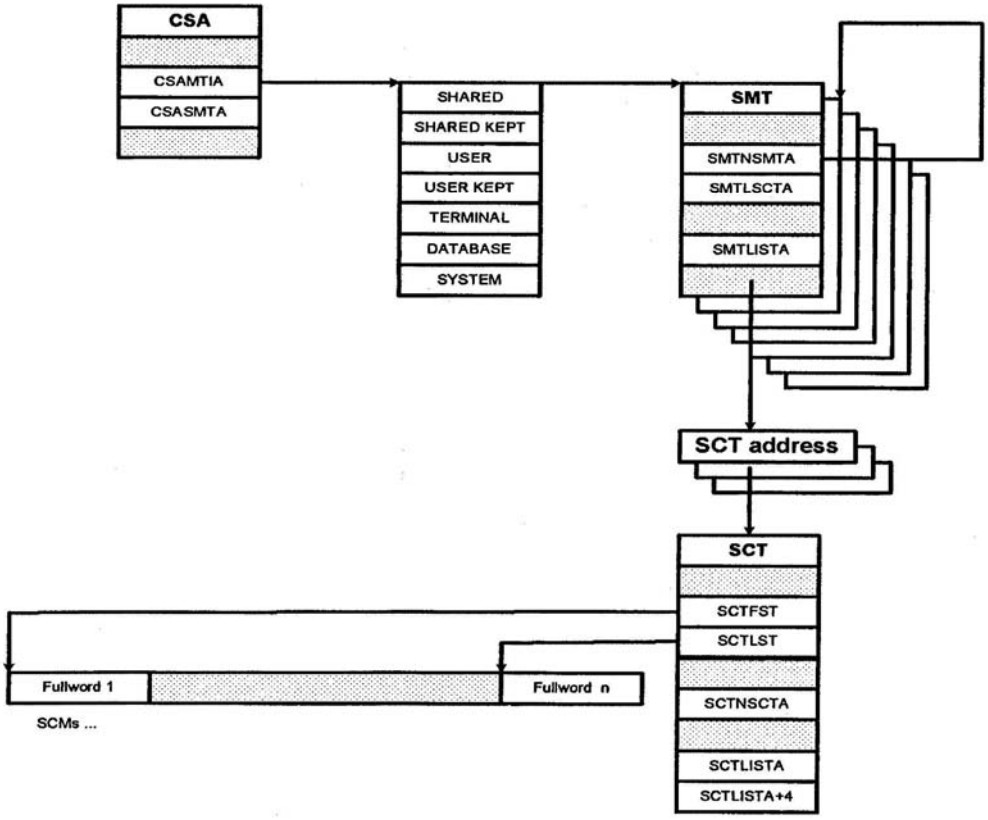
Storage Frame Suffix

1 Fullword at the End of the Block of Storage

- Contains the address of the RCE.

Note: DC016003 storage violated is detected when storage is freed. If the first fullword in the prefix (RCEADDR) does not equal the value in the suffix (RCEADDR)

Storage Management Control Blocks



Storage Frame

Storage Frame Prefix RCE address	Length	Requested Lenght (rounded up to 128 byte boundry if necessary)	Storage Frame suffix RCE address

Get Storage Request

RCE is allocated or extended (RCEMULT) for each block of storage allocated.

The RCE is anchored at:

```
LTE (LTEMRLEA)  If the GETSTG Request specified "LTEADDR="

TCE (TCEMRLEA)  If the GETSTG Request was for any of the
                following storage types:

                USER    TERMINAL        SYSTCE
                DATABASE

CSA (CSAMRLEA)  SYSCSA and SHARED
```

Only storage RCEs are anchored on these chains.

Multiple pieces of storage may be represented by a single RCE. The RCEMULT flag, X'80', at offset x'18' into the RCE will be on. When storage is allocated adjacent to an existing piece of storage and both pieces of storage have the same attributes, such as Type, Relocation, etc, the existing RCE is updated to account for the new storage allocation.

Storage Management

TCE Before Storage Allocations Have Occurred

The TCEMRLEA at offset X'38' within the TCE points to itself. This is a two fullword field. The first fullword is a forward pointer to an RLE. The second fullword is a prior pointer to an RLE.

The high order bit at address 00111440 indicates the end of the chain.

```
00111408  E3C3C55C 00000031 80111410 00111410 *TCE*............*
00111418  87D6CC54 07D6CCF0 87D6CC48 07D6CC48 *GO...O.0GO...O..*
00111428  80111428 00111428 08A6B808 07D7B2E0 *.........W...P..*
00111438  08F41E88 0010CB10 80111440 00111440 *.4.H....GO...O.U*
                              ^         ^
                              |         |
                           Forward     Prior
                                TCEMRLEA
```

TCE with no storage currently allocated.

RLE pointers set to the address of the anchor.

TCEMRLEA at offset X'38' in TCE X'80' bit at 00111440 denotes end of chain.

TCE and Control Blocks During Storage Allocations

The following sections demonstrate a TCE and associated control blocks during storage allocations.

TCE After First Storage Allocation

The RLE at address 7D6CDE0 was allocated.

The RCE at address 7D7B220 was allocated.

The storage requested was reserved at address 8F41E80.

Storage Frame Prefix placed at the beginning of the allocated storage (Address 8F41E80). Prefix contains the RCE address, 7D7B220, and the length of the storage allocation.

Storage Frame Suffix placed at the end of the allocated storage (Address 8F44D7C). Suffix contains the RCE address, 7D7B220.

The TCEMRLEA anchor was updated to point to the newly allocated RLE. Since this is the first storage allocation the Forward and Prior anchors both point to the same RLE.

The first two fullwords of the RLE are forward and prior pointers. Both the forward and prior pointers point back at the TCE anchor since this is the only RLE allocated at this time.

The third fullword of the RLE points to the RCE.

The fourth fullword of the RCE points to the allocated storage.

TCE

```
00111408  E3C3C55C 00000031 80111410 00111410 *TCE*............*
00111418  87D6CC54 07D6CCF0 87D6CC48 07D6CC48 *GO...O.0GO...O..*
00111428  80111428 00111428 08A6B808 07D7B2E0 *.........W...P..*
00111438  08F41E88 0010CB10 87D6CDE0 07D6CDE0 *.4.H....GO...O.U*
```

RLE

```
07D6CDE0  00111440 00111440 07D7B220 07D6CDF8 *.O..... .P...O.8*
             ^         ^        ^
             |         |        |
          Forward      Prior    RCE
```

RCE

```
07D7B220  01440001 00000031 00002F00 08F41E80 *.............4..*
07D7B230  0006E2C8 0006E1C0 00000000 00000000 *..SH............*
                            LENGTH   ADDRESS OF ALLOCATED STORAGE
```

Storage Area

```
          STORAGE FRAME PREFIX
08F41E80  07D7B220 00002F00 08F41088 00000000 *.P.......4.H....*
08F41E90  08F42C28 00234288 002342D8 00234208 *.4.....H...Q....*
08F41EA0  00000000 00000000 07D7B160 00000000 *.........P.-....*
    .         .        .        .        .
    .         .        .        .        .
08F44D50  00000000 4D010004 00000000 00000000 *....(............*
08F44D60  00000000 00000000 00000000 00000000 *................*
08F44D70  00000000 00000000 00000000 07D7B220 *................*
                                     STORAGE FRAME SUFFIX
```

TCE After Fifth Storage Allocation

Four additional storage allocations have occurred since the initial get storage. Currently there are five storage allocations yet only 2 RLEs and 2 RCEs. Notice the X'80' flag at offset X'18' in the second RCE, this is the RCEMULT Flag.

When the second get storage request occurred, a second RLE and RCE were allocated and chained into the TCEMRLEA list.

The TCEMRLEA forward pointer now points to the RLE at address 7D6CDE0, this RLEs forward pointer points to the RLE at address 7D6CD68, and this RLE's forward pointer points back at the anchor in the TCE. The prior chain has been connected as well. The third fullword of the RLE at address 7D6CD68 points to the RCE at address 7D7B580.

The RCEMULT flag was not on after the second get storage request. The third fullword of the RCE contains the length of the storage controlled by the RCE. Again, after the second get storage request this RCE length field contained a value of X'00000180' (see Storage Frame Prefix for the storage pointed to by the RCE, the second get storage request was for a length of X'00000180' bytes).

The following 3 get storage requests were for storage that was identical to the second get storage request and were placed in adjacent storage. Rather then allocate a new RLE and RCE and increase the lengths of these chains for each get storage request, the existing RCE was used, the RCEMULT flag was turned on, the RCE length field was increased by the length of each new piece of storage. Each storage allocation gets its own storage frame prefix and suffix.

TCE

```
00111408  E3C3C55C 00000031 80111410 00111410 *TCE*............*
00111418  87D6CC54 07D6CCF0 87D6CC48 07D6CC48 *GO...O.0GO...O..*
00111428  80111428 00111428 08A6B808 07D7B2E0 *.........W...P..*
00111438  08F41E88 0010CB10 87D6CDE0 07D6CD68 *.4.H....GO...O.U*
                              ^        ^
                              |        |
                           Forward     Prior
                                 TCEMRLEA
```

RLEs

```
          Forward      Prior    RCE
07D6CDE0  07D6CD68 00111440 07D7B220 07D6CDF8 *.O..... .P...O.8*

07D6CD68  00111440 07D6CDE0 07D7B580 07D6CD8C *.O...O...P...O..*
```

RCEs:

```
                              LENGTH   ADDRESS OF ALLOCATED STORAGE
07D7B220  01440001 00000031 00002F00 08F41E80 *.............4..*
07D7B230  0006E2C8 0006E1C0 00000000 00000000 *..SH............*

07D7B580  01400001 00000031 00000400 00234000 *. ............ .*
07D7B590  0011B504 0011B3E0 80000000 00000000 *................*
                             ^
                             |
                            RCEMUCT
```

Storage Areas

First Get Storage Request

```
08F41E80  07D7B220 00002F00 08F41088 00000000 *.P.......4.H....*
08F41E90  08F42C28 00234288 002342D8 00234208 *.4.....H...Q....*
08F41EA0  00000000 00000000 07D7B160 00000000 *.........P.-....*
    .         .        .        .        .
    .         .        .        .        .
08F44D70  00000000 00000000 00000000 07D7B220 *................*
```

Second Get Storage Request

```
STORAGE FRAME PREFIX
00234000  07D7B580 00000180 E4E2C5D9 D9C5C7E2 *.P.....USERREGS*
00234010  0000000E 00000000 000000A0 08A668FE *.............W..*
00234020  07D01504 08A6B808 8809A21A 00000000 *.....W..H.S.....*
     .         .        .        .        .
     .         .        .        .        .
00234170  00000000 00000000 00000000 07D7B580 *.............P..*
STORAGE FRAME SUFFIX
00234180  07D7B580 00000080 00000000 00000000 *.P..............*
00234190  00000000 00000000 00000000 00000000 *................*
     .         .        .        .        .
     .         .        .        .        .

002341F0  00000000 00000000 00000000 07D7B580 *.............P..*

00234200  07D7B580 00000080 88A6B808 00000000 *................*
00234210  00000000 00000000 00000000 00000000 *................*
     .         .        .        .        .
     .         .        .        .        .
00234270  00000000 00000000 00000000 07D7B580 *.............P..*

00234280  07D7B580 00000180 00000000 00000000 *.P..............*
00234290  00000000 00000000 00000000 00000000 *................*
002342A0  00000000 00000000 00000000 00000000 *................*
     .         .        .        .        .
     .         .        .        .        .
002343F0  00000000 00000000 00000000 07D7B580 *.............P..*
```

RCE Control Block Storage Type

The second byte of the RCE is the storage type

(Non Kept Storage)

40,41 User 42,43 Shared 44,45 SYSTCE

46,47 Terminal 48,49 Database 4A,4B SYSCSA

(Kept Storage)

C0,C1 User C2,C3 Shared C4,C5 SYSTCE

C6,C7 Terminal C8,C9 Database CA,CB SYSCSA

Storage Algorithms

Two values used are based on storage amount requested:

Blocks needed:

1 Block = 128 Bytes (including Prefix and Suffix)

Blocks are rounded up to 128 bytes if necessary

Pages needed:

1 Page = 32 Blocks

Pages are rounded up to 4096 bytes if necessary

Blocks

(Bytes Requested+SFP+SFS+127) / 128

SFP (Storage Frame Prefix) 8 Bytes

SFS (Storage Frame Suffix) 4 Bytes

Examples:

Bytes Requested = 116

(116 + 8 + 4 + 127) / 128

255 / 128

Blocks needed = 1

Bytes Requested = 117

(117 + 8 + 4 + 127) / 128

256 / 128

Blocks needed = 2

Pages

(Blocks requested+31) / 32

Block = 128 Bytes

Examples:

Blocks = 16

(16 + 31) / 32

47 / 32

Pages needed = 1

Storage Passes

Pass 1 storage allocation

Conditions:

- Need 32K or less
- Not UserShared,Protect,MT

Scan the storage RCE's already on the selected anchor to see if this storage request can be satisfied on a page in an existing allocation.

The RCE is deemed compatible if:

- The request::RCE XA/nonXA attribute matches
- The request::RCE poolcode matches or the RCE's SCT allows the type we want and
  - CSAPROT is off (all types of storage may coexist on the same page)
  - CSAPROT is on and the request::RCE both are for System storage or the request::RCE both are for User storage

If an RCE is not compatible, try next RCE. If no more RCEs, proceed to Pass 2. Once an eligible RCE is found, perform the following test.

1. Look for N free bits adjacent to the rightmost bit in the allocation.
2. Look for N free bits adjacent to the leftmost bit in the allocation.
3. If this RCE cannot be used, proceed to Pass 2.

Pass 2 Storage allocation

Scan SMT->SCT->SCMs for available space.

Starting with the selected SMT, find an SCT with sufficient free pages to satisfy the request..

Scan the SCMs to find N contigous free bits

(N = number of 128 byte blocks)

If Optional Apar bit 193 is used or
Protect is on at the System Level and the
uest is for System Type Storage and the
selected SCT only allows System type storage)

Begin search using first empty SCM pointed to by SCMFSTEP
Otherwise
Begin search using first SCM with available space pointed to by SCMFSTAV

• DC Storage Protection

- Enabled with PROTECT in the sysgen SYSTEM statement
- There are two storage protect keys:
  - Primary storage protect key
  - Alternate storage protect key
    - Alternate key 9 has a special meaning

Primary storage key

When CA-IDMS/DC or a nonprotected program has control:

- The PSW is set to the primary protect key
- All storage pages in the region are set to the primary storage protect key

Alternate storage protect key

Applies only when a program is both defined with the PROTECT option on the sysgen PROGRAM statement and the "Protect" option is specified on the SYSTEM statement

When a user mode program has control:

Alternate storage protect key other than 9 (10 – 15)

- The PSW is set to the alternate storage protect key
- The following pages are set to the alternate storage protect key:
  - Task pages in the TCEMRLEA list
  - Shared pages in the CSAMRLEA list
  - User program's pages in the nonreentrant program pool
- User/system boundary
- Keys are switched back each time the boundary is crossed
- Storage violations appear as protection exceptions
- SVC overhead - Storage protection requires additional system overhead: two calls for system services for each made by programs executing in storage protect mode

Alternate storage protect key of 9

- During startup, all of the pages in the "user" defined storage pools are set to the alternate storage protect key 9

- Only the PSW key is switched between the primary and alternate storage protect keys.
- Storage violations appear as protection exceptions
- There is virtually no CPU overhead
- Protects user programs from overlaying system storage
- Does not protect user programs from overlaying other user programs storage

Storage Management Process

1. Get storage requests may request storage be waited for if the request cannot be serviced.
   - The wait is on and internal ECBLIST consisting of the ECB within the SMT(s) SMTSSECB.
   - The wait can be on an XA pool on non-XA pool or both.
2. As each pool reaches short-on-storage (SOS), a flag is set in the SCT.
3. Global SOS condition flag is set if Pool 0 goes SOS.

Relocatable Storage

The relocatable threshold determines the point at which DC writes to scratch. Relocatable storage is written to scratch only when the storage pool exceeds the relocatable threshold.

The system generation ADSO statement tells ADS whether to issue #GETSTG with or without RELOC=YES. The DBA can control whether and when relocation takes place by specifying a relocation threshold of ALWAYS, NEVER, or BASED ON STORAGE POOL USAGE (THRESHOLD).

Exercise 11-1

Answer these questions using the memory displays on the next page.

1. How many SMT entries are shown?
2. Which SMT entry represents below the line storage ability?
3. What is the address of the first SCM for the SCT shown?
4. How many 128 bytes blocks in the first SCM are available?
5. What is the address of the first SCM with available space?
6. How many 128 byte blocks are available in the first SCM with available space?
7. True or false? The RCE shown describes a piece of storage the is X'220' bytes long.
8. True or false? The first fullword in the Storage area points to the RLE anchor in the TCE.

Memory Displays for Exercise 11-1

SMT

```
 <Addr>   <Offset>               <Hex>                   <Character>
0008EC60  00000000  80400001 0008EC84 00000000 00000000  *. .....d........*
0008EC70  00000010  400000FF 0AC50318 18000000 0008EE58  * ....E..........*
0008EC80  00000020  00000000 40400001 0008ECA8 0008E3C0  *....  .....y..T.*
0008EC90  00000030  00000002 400000FF 0AC50330 18000000  *.... ....E......*
0008ECA0  00000040  0008EE58 00000000 20400001 0008ECCC  *........ ......*
0008ECB0  00000050  0008E3C0 00003349 400000FF 0AC50348  *..T..... ....E..*
0008ECC0  00000060  18000000 0008EE58 00000000 10400001  *............. ..*
0008ECD0  00000070  0008ECF0 00000000 00000002 400000FF  *...0........ ...*
0008ECE0  00000080  0AC50360 18000000 0008EE58 00000000  *.E.-.....E.....*
0008ECF0  00000090  08400001 0008ED14 00000000 00000000  *. .............*
0008ED00  000000A0  400000FF 0AC53A18 18000000 0008EE58  * ....E.........*
0008ED10  000000B0  00000000 04400001 0008ED38 0008E3C0  *..... ........T.*
0008ED20  000000C0  0000001C 400000FF 0AC5D3E0 18000000  *.... ....EL.....*
0008ED30  000000D0  0008EE58 00000000 02400001 0008ED5C  *........ .....**
0008ED40  000000E0  0008E3C0 000000C4 40000000 00000000  *..T....D .......*
0008ED50  000000F0  18000000 0008EE58 00000000
```

SCT

```
 <Addr>   <Offset>               <Hex>                   <Character>
0008E3C0  00000000  0035E000 00561000 0008E440 0008EC48  *.........U ....*
0008E3D0  00000010  0008E480 00066BE0 00000203 00000203  *..U...,.........*
0008E3E0  00000020  0000001B 00000054 00000000 00000008  *................*
0008E3F0  00000030  00FE0000 0000344B 00003402 000032D7  *..............P*
0008E400  00000040  00000174 00000000 00000000 00000000  *................*
0008E410  00000050  00000000 00000000 00000000 0008EE68  *................*
0008E420  00000060  0008EE80 00000000 00000000 00000000  *................*
0008E430  00000070  0008E4AC 0008E58C 00000000 00000000  *..U...V.........*
0008E440  00000080  FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF  *................*
0008E450  00000090  FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF  *................*
0008E460  000000A0  FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF  *................*
0008E470  000000B0  FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF  *................*
0008E480  000000C0  FFFFFF80 80000000 FFFFFFFF FFFFFFFF  *................*
0008E490  000000D0  FFFFFFFF FFFFF000 FFF00000 FFFFFFFF  *......0..0......*
0008E4A0  000000E0  FFFFE000 FFF00000 80000000 00000000  *.....0.........*
0008E4B0  000000F0  00000000 00000000 00000000 00000000  *................*
0008E4C0  00000100  00000000 00000000 00000000 00000000  *................*
0008E4D0  00000110  00000000 00000000 00000000 00000000  *................*
0008E4E0  00000120  00000000 00000000 00000000           *............    *
```

RCE

```
0AC51498  01440001 00000220 00002B80 0AC74880 *.............G..*
0AC514A8  00070AF0 00070940 00000000 00000000 *...0... ........*
```

Storage Area

```
0AC74880  0AC51498 00002B80 0AC65F88 00000000 *.E.Q.....F.H....*
0AC74890  0AC75628 00377288 003772D8 00377208 *.G.....H...Q....*
0AC748A0  00000000 00000000 0AC7FC58 00000000 *.........G......*
0AC748B0  00000000 4D010004 00000000 00000000 *....(...........*
```