What's So Special About DEDBs

Overview of Fast Path DEDBs

KAREN TISCHER

ABSTRACT

What's so special about Fast Path Data Entry Data Bases (DEDBs) – an introduction.

DEDBs came into existence long before HALDBs and provided users with feature such as partitioning a database into independently managed AREAs, providing large database, high availability, online reorganization and recovery, fast data collection, among other features.

First designed for the banking industry, DEDB features appealed to other industries and grew in capabilities and functions to support those industries. Today you can find DEDBs in industries such as financial institutions, package shipping/tracking, airline scheduling/reservations, manufacturing, telephone companies, among others.

Are you a user?

What is your industry?

Why/how do you use DEDBs?

Join me to find out how great DEDBs are.

DEDB Topics

- Overview
- Characteristics
- Unique Features
- Summary

DEDB Overview

- Delivered with IMS DB Manager
- Available to ONLINE systems only
  - DB/DC
  - CICS-DBCTL
- Designed to provide:
  - High Availability
  - Large database
  - High volume processing
  - High speed processing

DEDB Characteristics

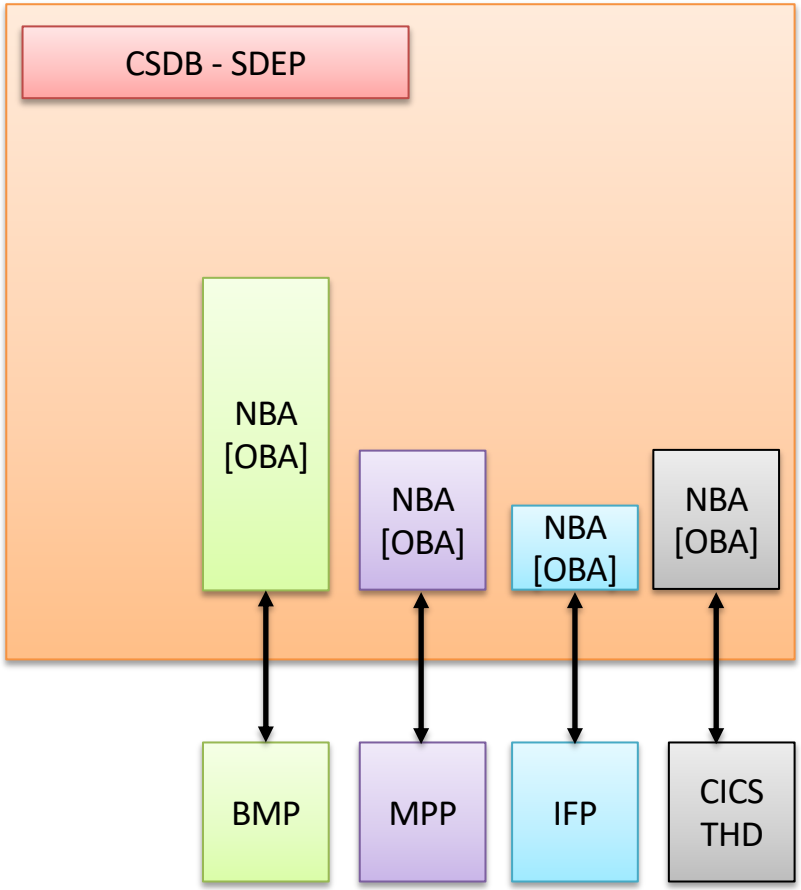
- HDAM like: Roots randomize to RAP
- Partitioned into up to 9999* AREAs (was 2048)
  - Area data sets: VSAM ESDS, max 4GB
    - Managed by Media Manager
    - Must be preformatted
    - No data set extensions allowed
  - Supports large database (9999 * 4GB = 39,996GB)
- Utilities run ONLINE at the AREA level
- Commands operate at the AREA level
- No logical relationships
- Some unique features

DEDB Unique Features

- System Functions
  - Buffers
  - Locking
  - Logging
  - DASD updates
- Hierarchic Structure
- Database record storage
- Randomizing
- Sequential Dependents
- Subset Pointers
- Multiple Area Data Sets
- High Speed Sequential Processing
- Virtual Storage Option
- Online Utilities

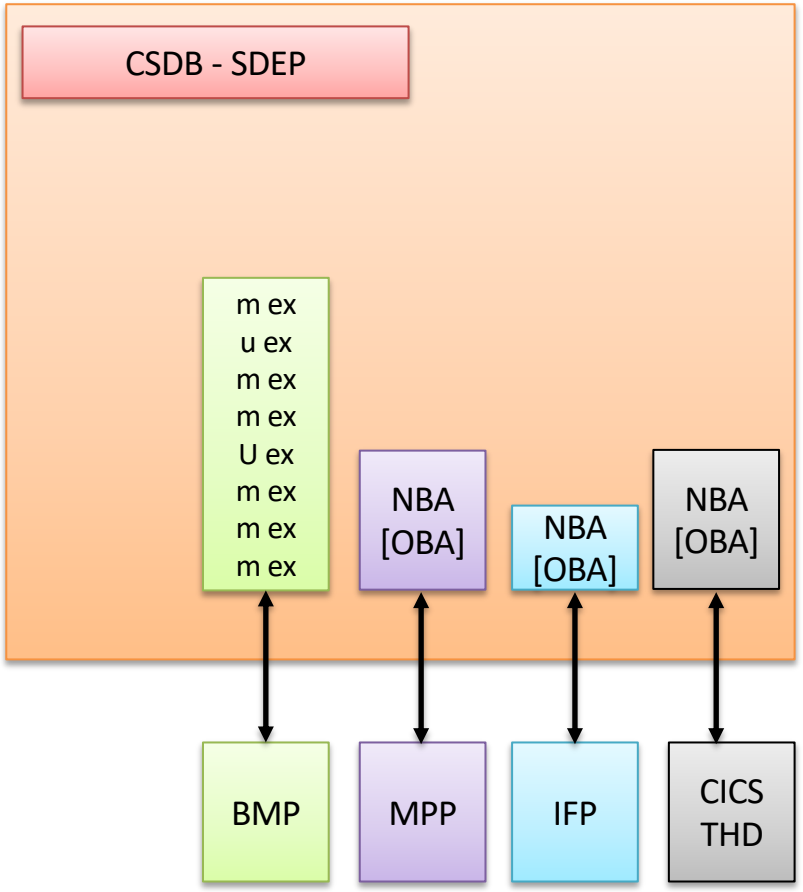
DEDB System Functions: Buffers

- Multiple buffer pools in 64-bit storage
  - dynamically allocated by appropriate size based on CISIZEs
  - Self adjusting based on demand
- Each scheduled PSB (thread/region) acquires its 'own' buffers which are not shared with others
    - NBA (FPBUF)
    - OBA (FPBOF)
- No buffer look-aside
- Special buffers for collecting SDEPS: written only when full

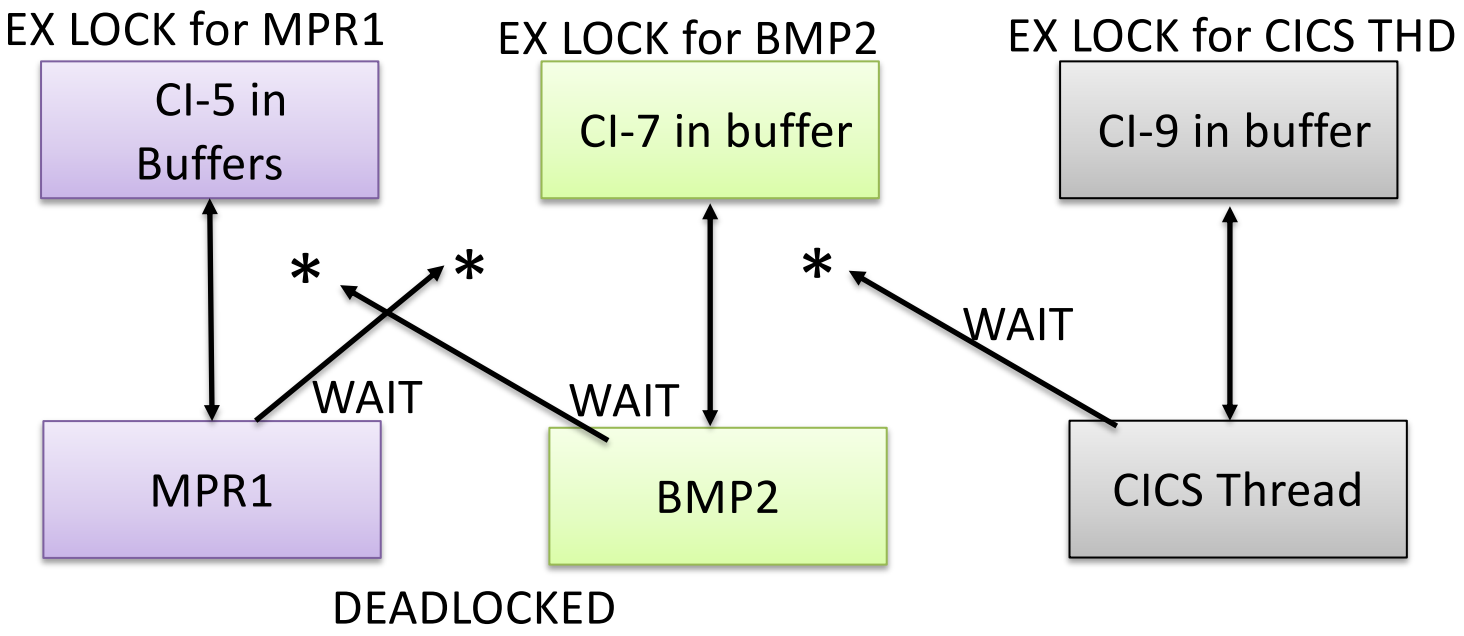


DEDB System Functions: LOCKING

- Locks on CI, based on PROCOPT in PCB
  - Exclusive
  - Read with integrity
  - No lock
- PI or IRLM used only for deadlock detection

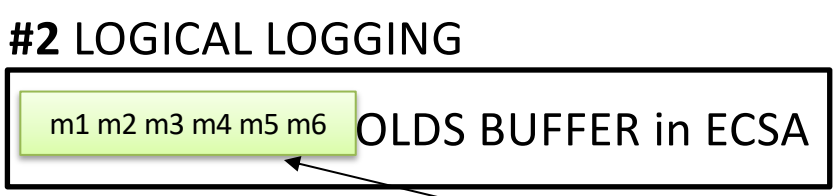


DEDB System Functions: DEADLOCKS

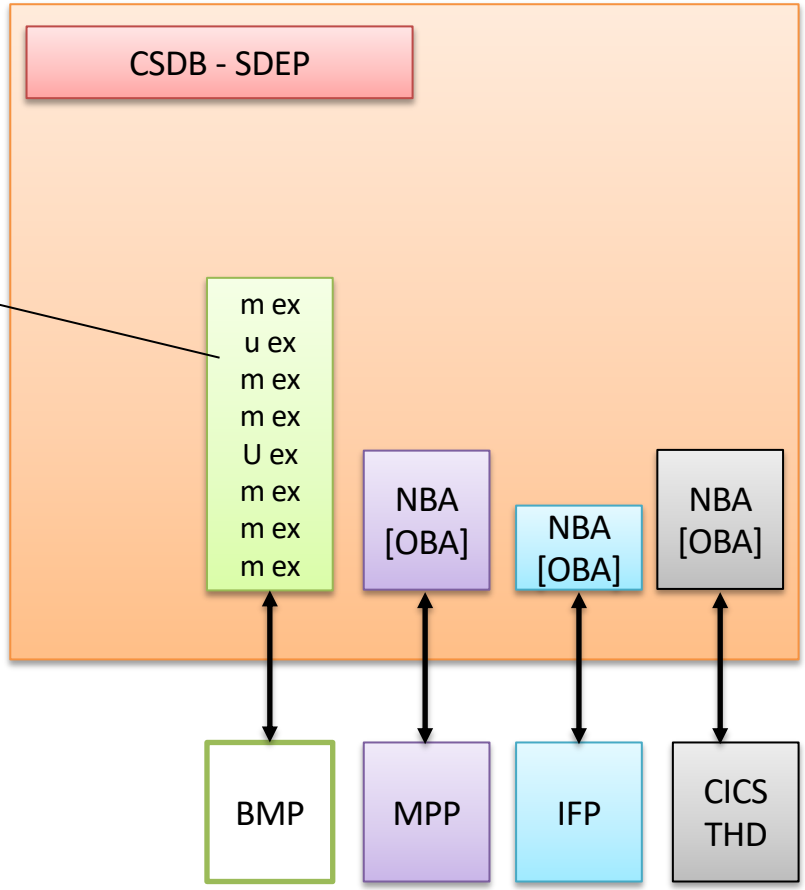


DEDB System Functions: COMMIT

#2 LOGICAL LOGGING



- Updates LOGICALLY logged at COMMIT time
  - Build AFTER images only
  - Move into OLDS buffer
- No backout needed nor possible
  - Database not physically updated during commit process but AFTER physical logging

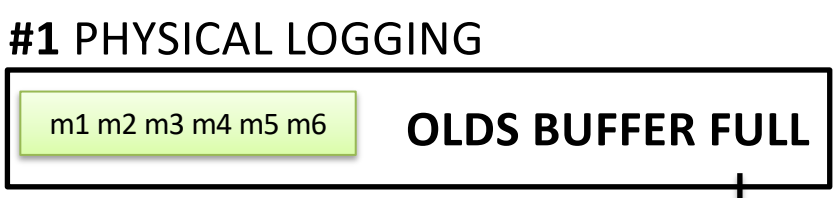


#1 BMP COMMITS

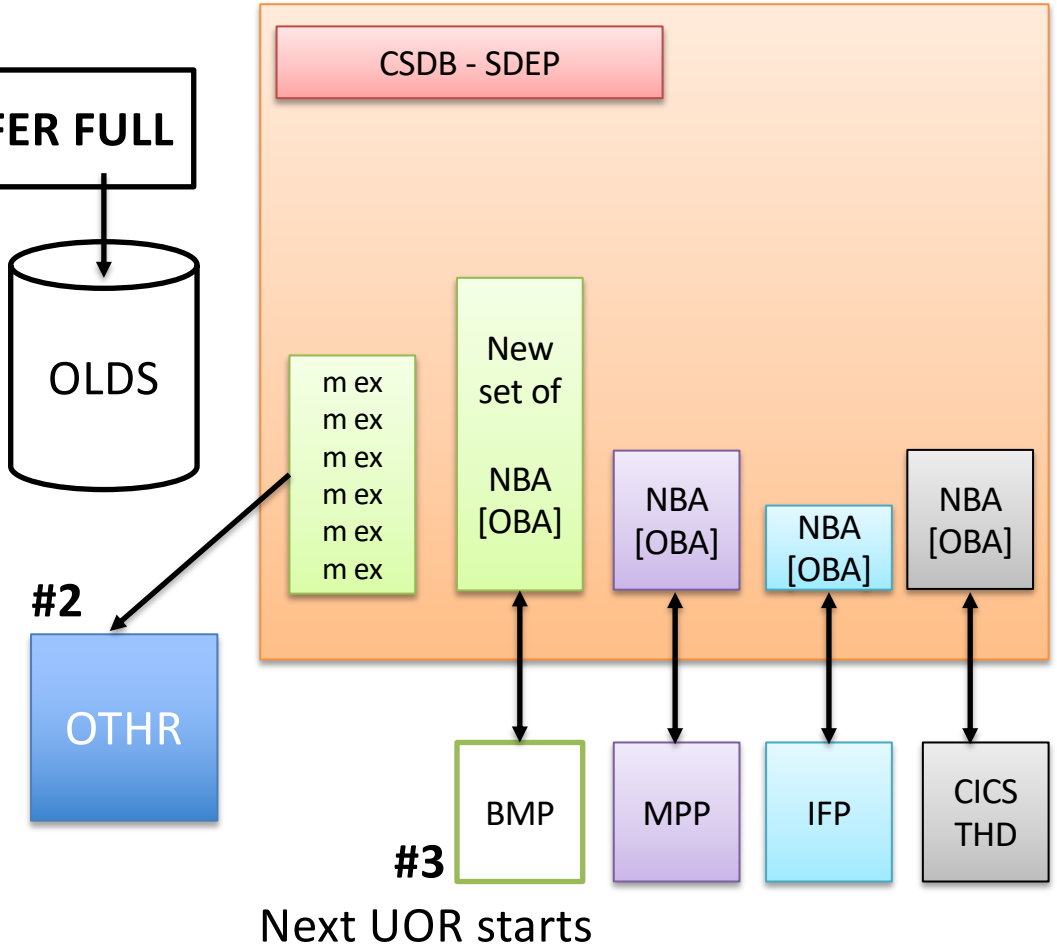
#3 WAITS until log records written to DASD

DEDB System Functions: LOGGING

#1 PHYSICAL LOGGING

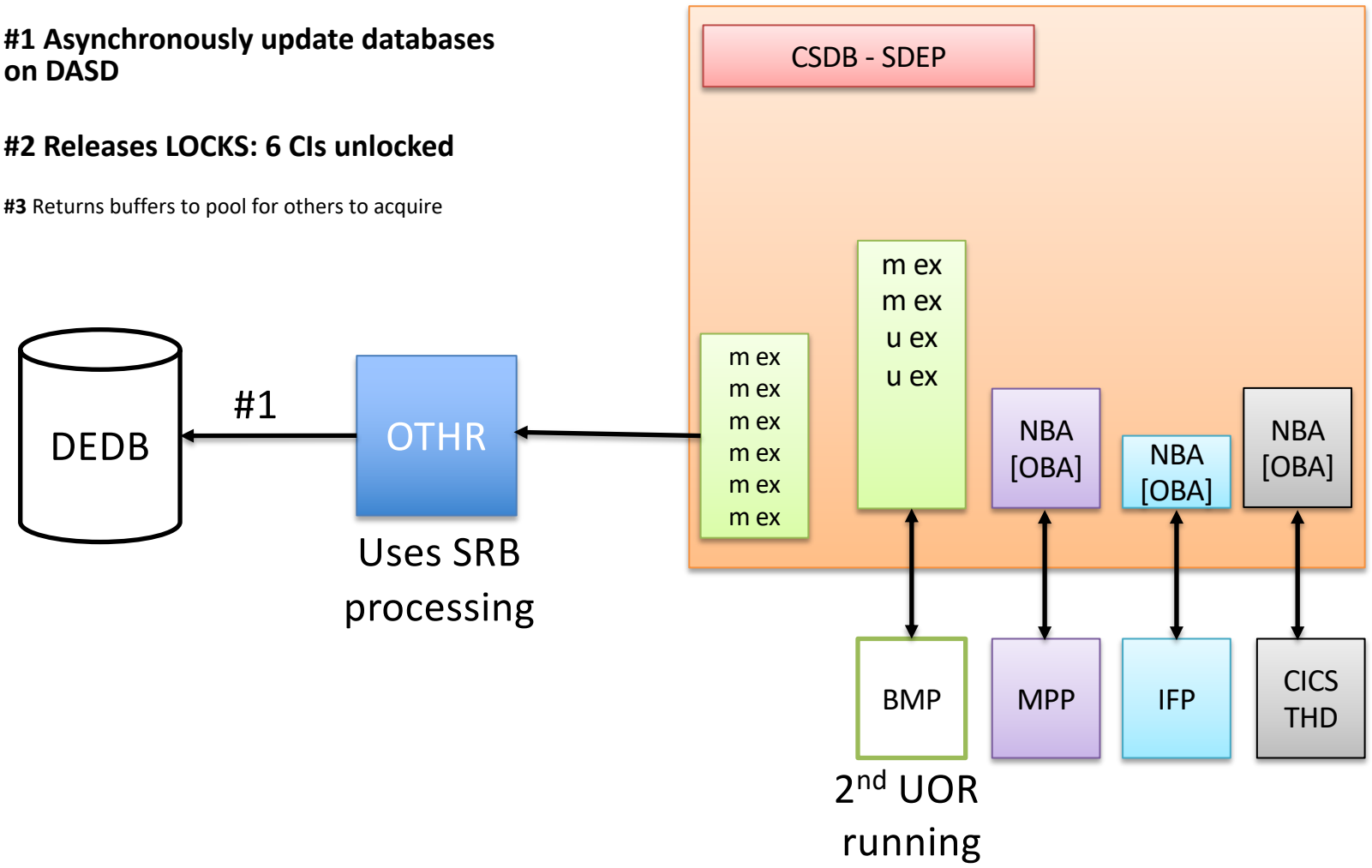


- When OLDS BUFFER is full
- BMP:
  - Passes Modified buffers to OTHREAD
  - Locks still held but now owned by OTHREAD
- BMP acquires a new set of buffers and proceed processing next UOR



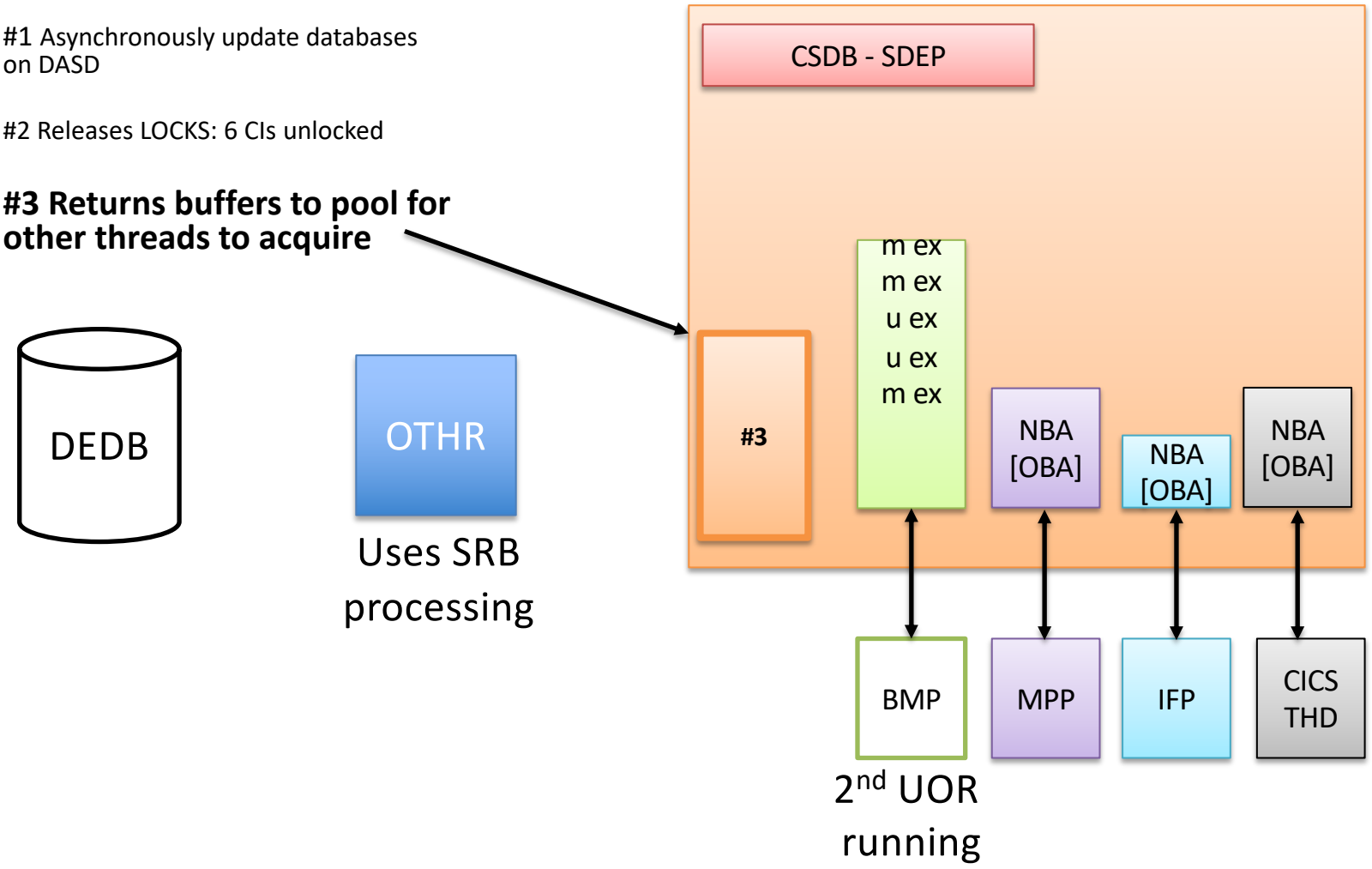
DEDB System Functions: OUTPUT THREAD

- **#1 Asynchronously update databases on DASD**
- **#2 Releases LOCKS: 6 CIs unlocked**
- **#3** Returns buffers to pool for others to acquire



DEDB System Functions: Buffers Returned

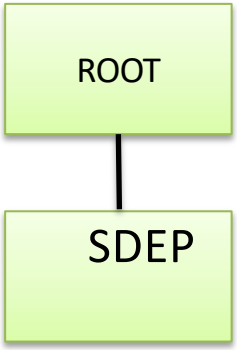
- #1 Asynchronously update databases on DASD
- #2 Releases LOCKS: 6 CIs unlocked
- **#3 Returns buffers to pool for other threads to acquire**



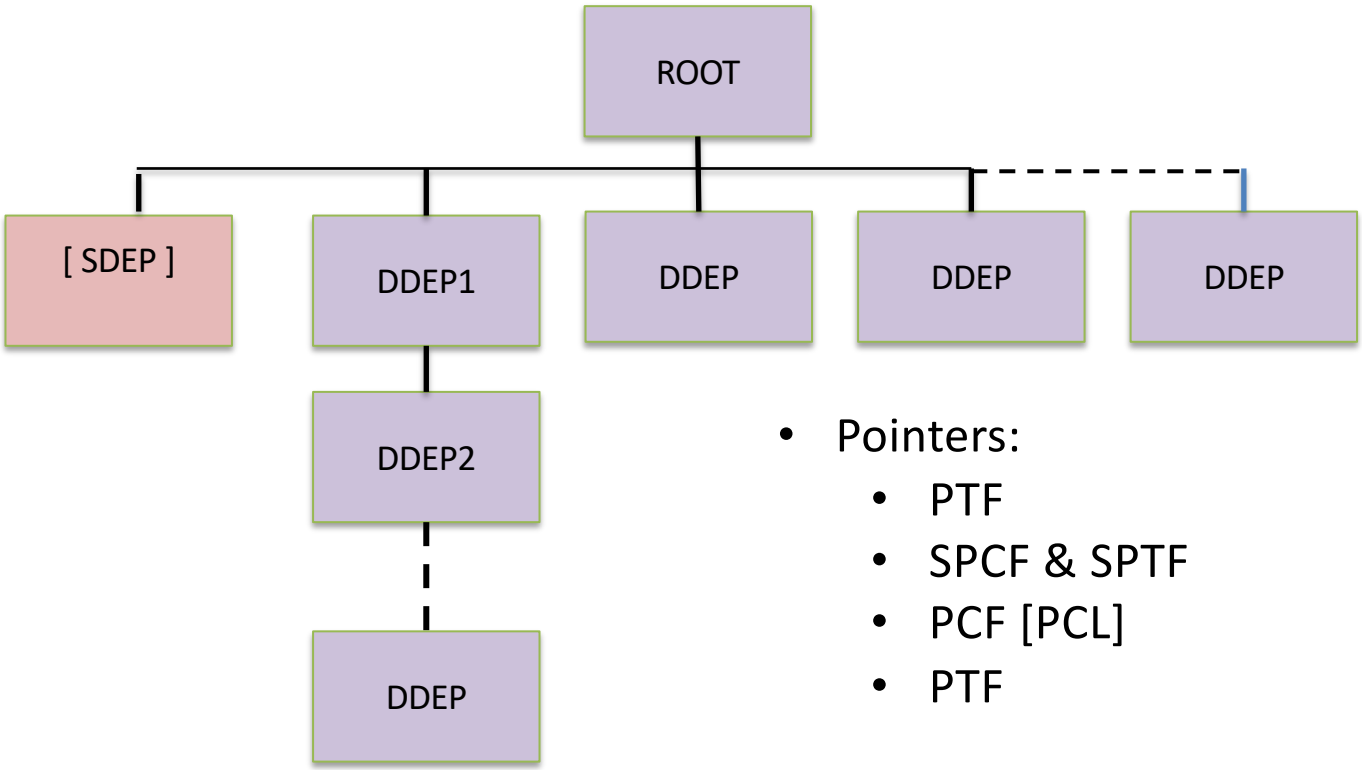
DEDB HIERARCHIC STRUCTURE 1 of 2
In the Beginning

1. Punch data into 80 character card
2. Put JCL in front of cards
3. Run program
   - Read card
   - Post account

1. Trade keypunch and cards for an online transaction
2. Invoke a transaction
3. Enter requested data
4. Program collects data in SDEP
5. SDEPs extracted and processed by program that posts accounts



DEDB RECORD STRUCTURE 2 of 2

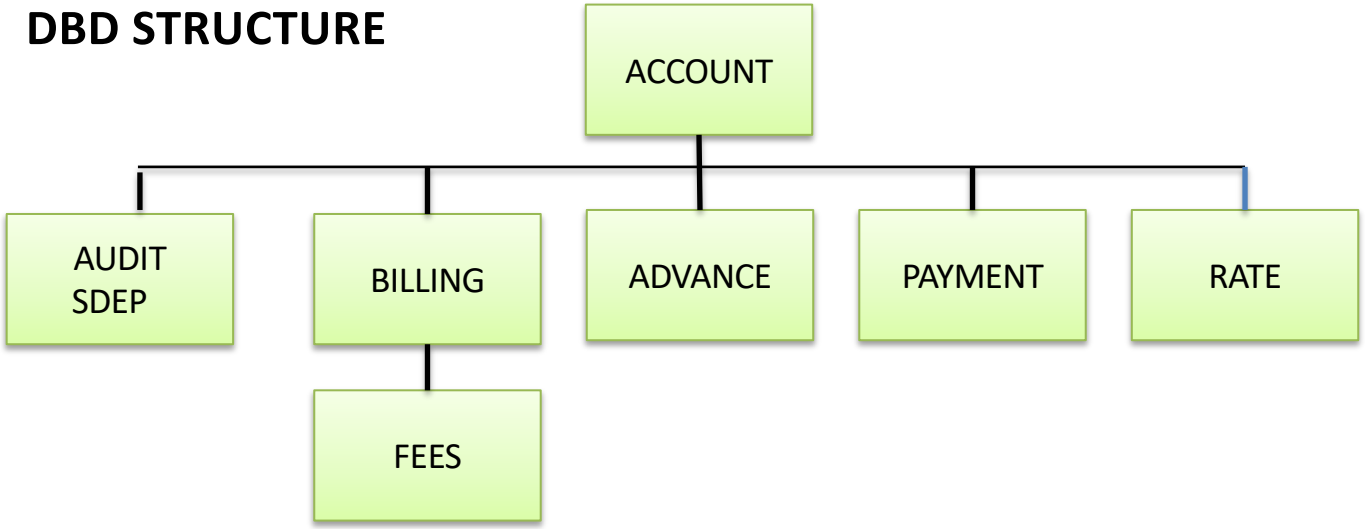


- Pointers:
  - PTF
  - SPCF & SPTF
  - PCF [PCL]
  - PTF

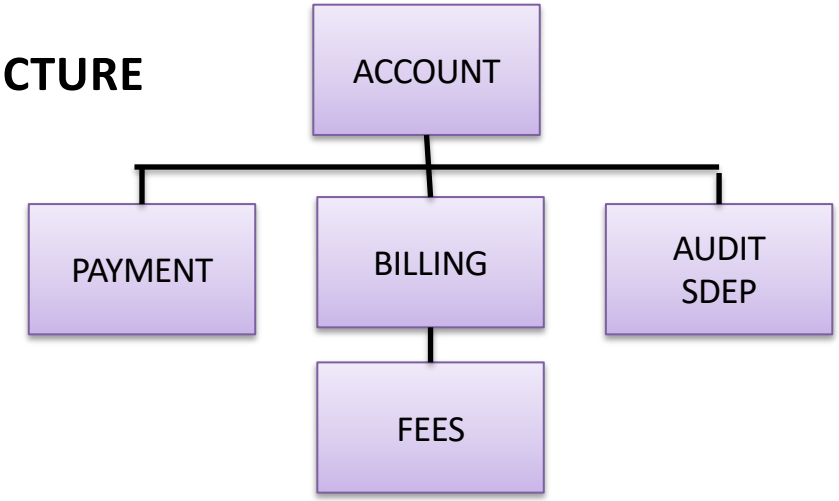
- 15 Levels
- 127 segment types
- SDEP optional – always leftmost @ level 2
- Stored on DASD as variable length (with LL)
- May be defined a fixed length for apps

DEDB DBD vs PSB: Multiple Positioning is the DEFAULT

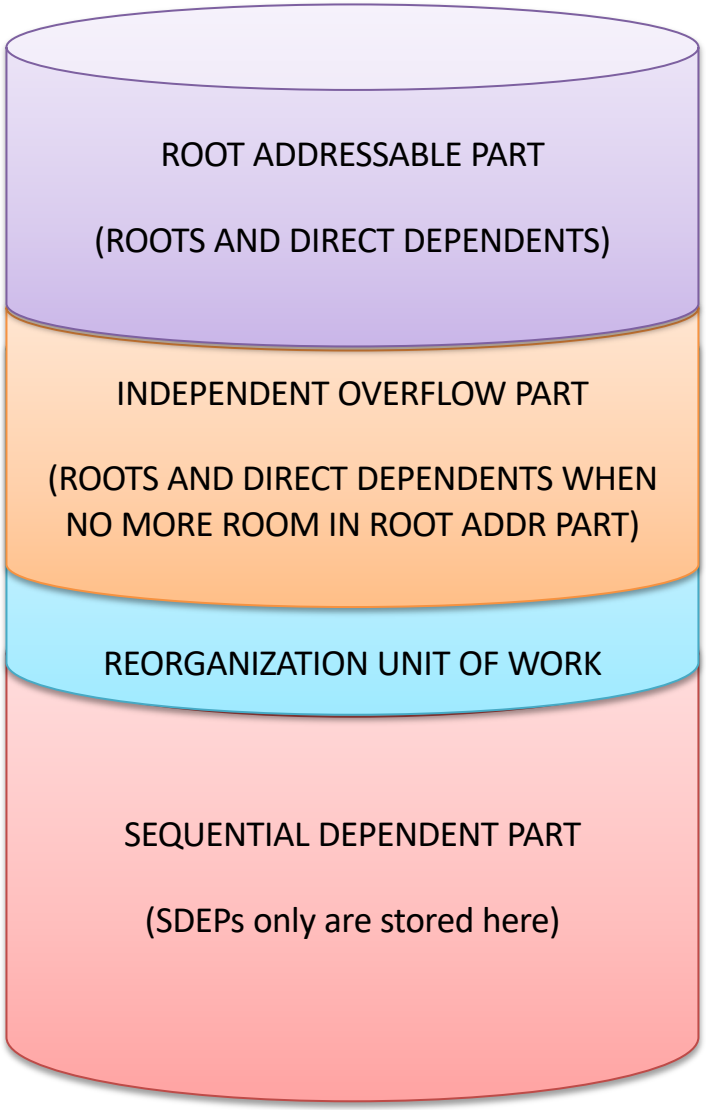
DBD STRUCTURE



PSB STRUCTURE

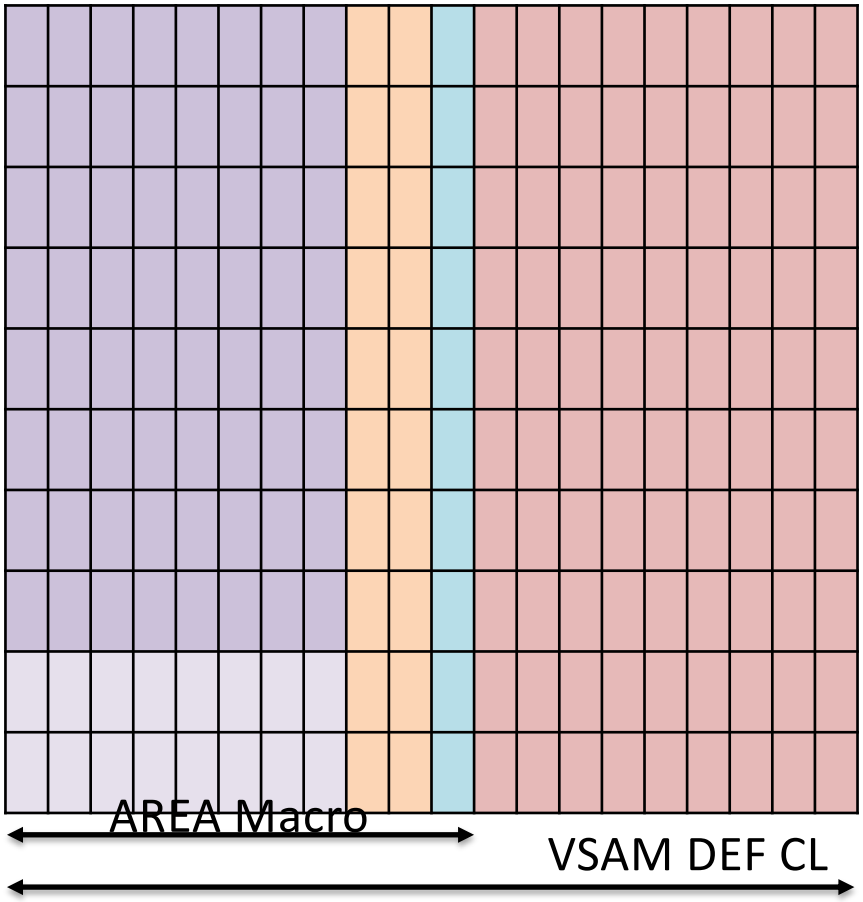


DEDB AREA must be FORMATTED



DBD DBDNAME=DEDB,ACCESS=DEDB

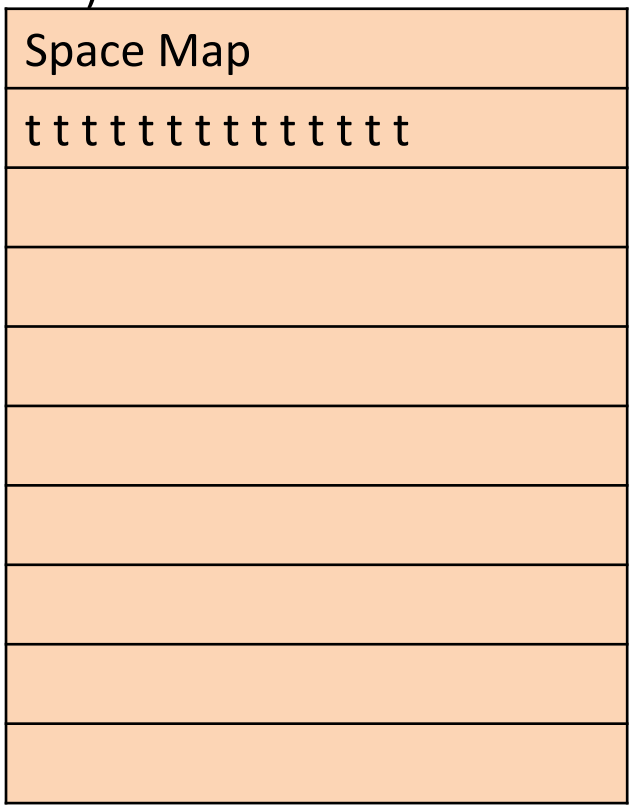
AREA UOW=(10,2),ROOT=(10,2),SIZE=8192



How data is stored in an AREA's UOW

UOW like a mini-HDAM DB

IOVF an imbedded UOW (data set) extension

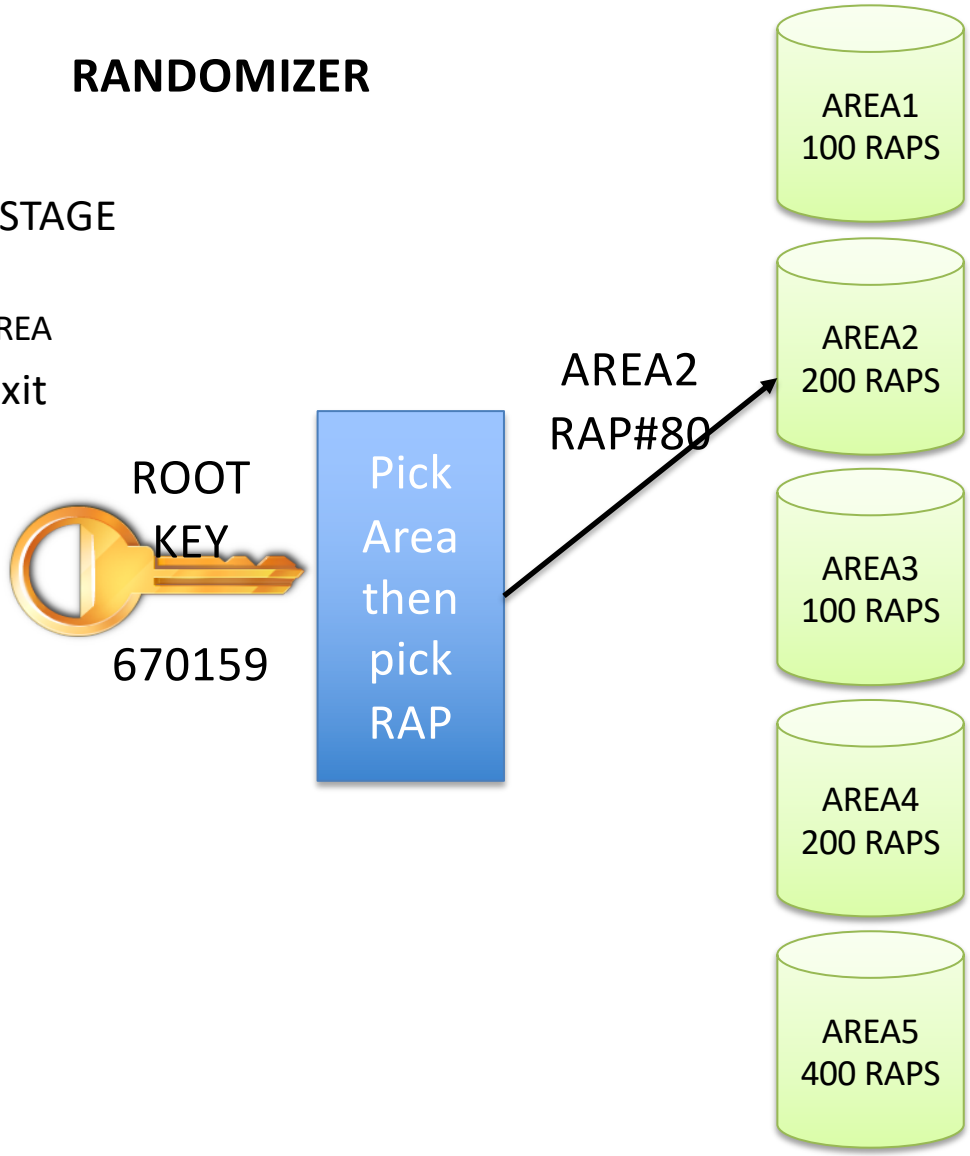


ROOT K randomized to RAP7 and has dependents. Record size = 12K
ROOT N randomized to RAP4 with dependents. Record size = 16K
ROOT T randomized to RAP7 with dependents. Record size = 10K

AREA UOW=(10,2),ROOT=(10,2),SIZE=8192

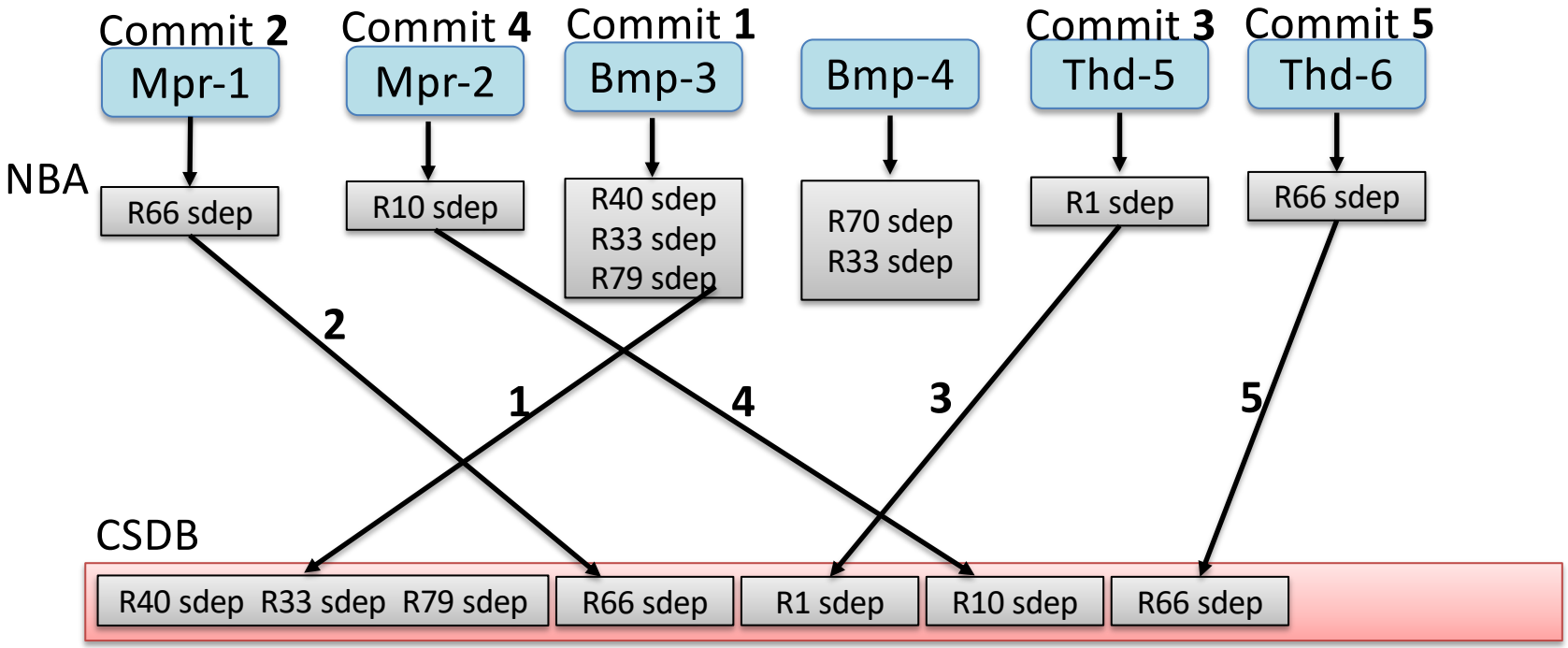
RANDOMIZER

- DBFHDC44: modify to TWO STAGE
  - Pick AREA first, then
  - Randomize to RAP with that AREA
- DBFPSE00: AREA selection exit routine
- Stage 1 based on:
  - Branch office #
  - ATM #
  - Time Zone
  - Division
  - Account #
  - Vehicle model #
  - Area code
  - Zipcode
  - Employee #
  - Other . . .



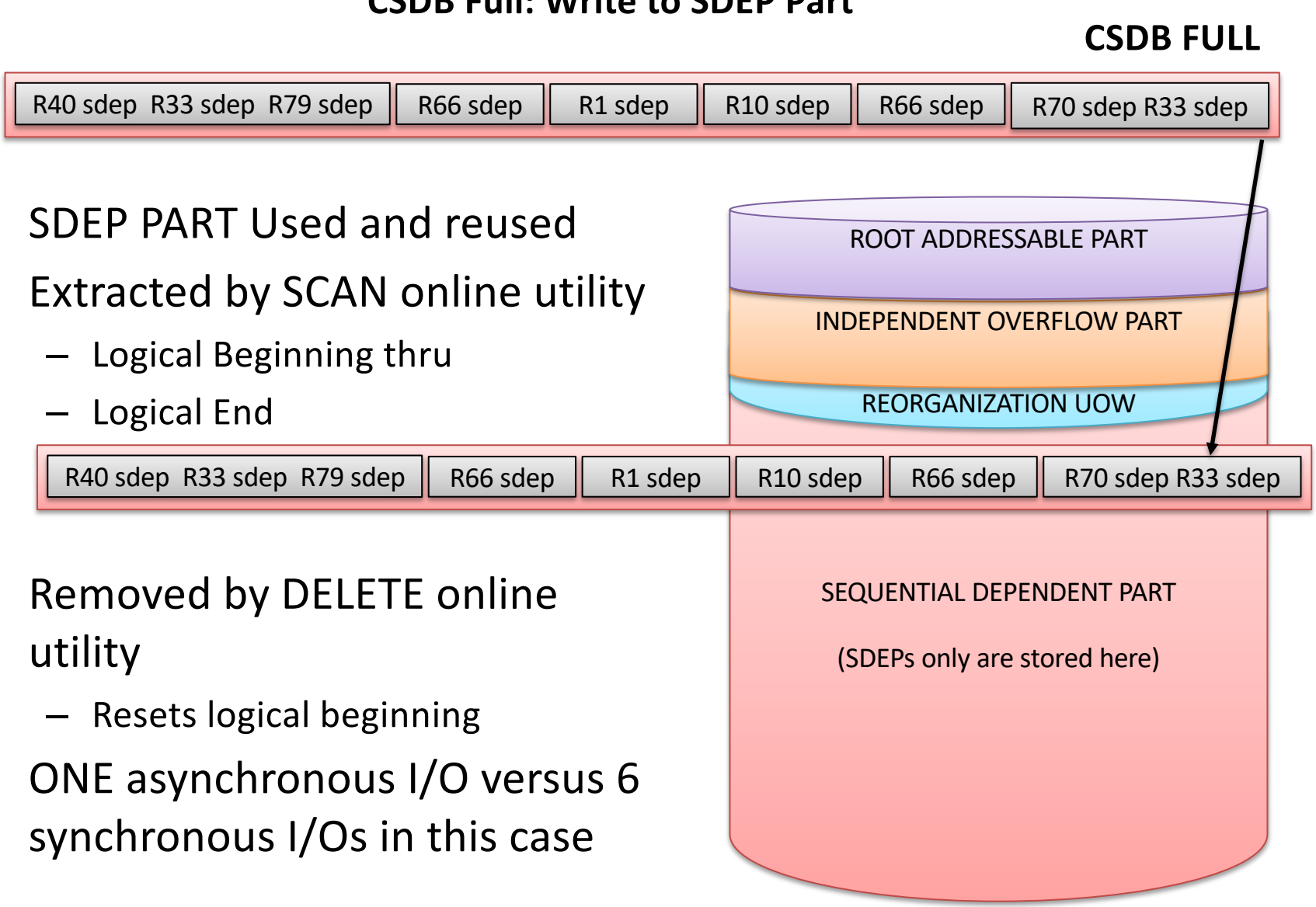
Sequential Dependents

- The following threads are ISRTing ‘sdeps’ into NBA buffers.
- At thread commit, the SDEPs are moved from NBA to CSDB



- **At commit,** move into next available place in CSDB in arrival time order – **no I/O to area**
- **CSDB written only when full or AREA closed**

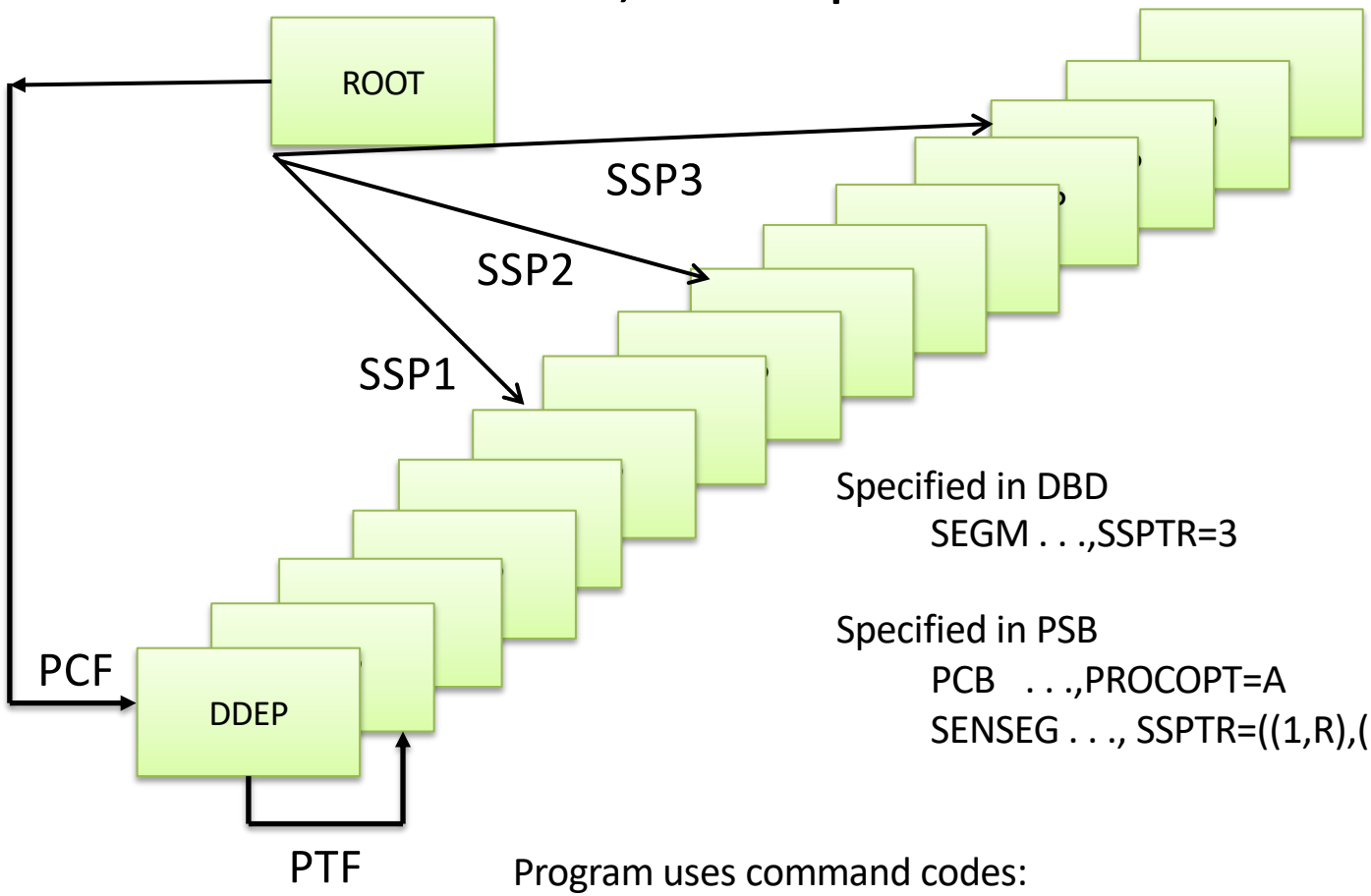
CSDB Full: Write to SDEP Part



- SDEP PART Used and reused
- Extracted by SCAN online utility
  - Logical Beginning thru
  - Logical End
- Removed by DELETE online utility
  - Resets logical beginning
- ONE asynchronous I/O versus 6 synchronous I/Os in this case

SUBSET POINTERS – Maximum of 8:

GU Root, GNP ddep*R3M3



Specified in DBD

SEGM . . .,SSPTR=3

Specified in PSB

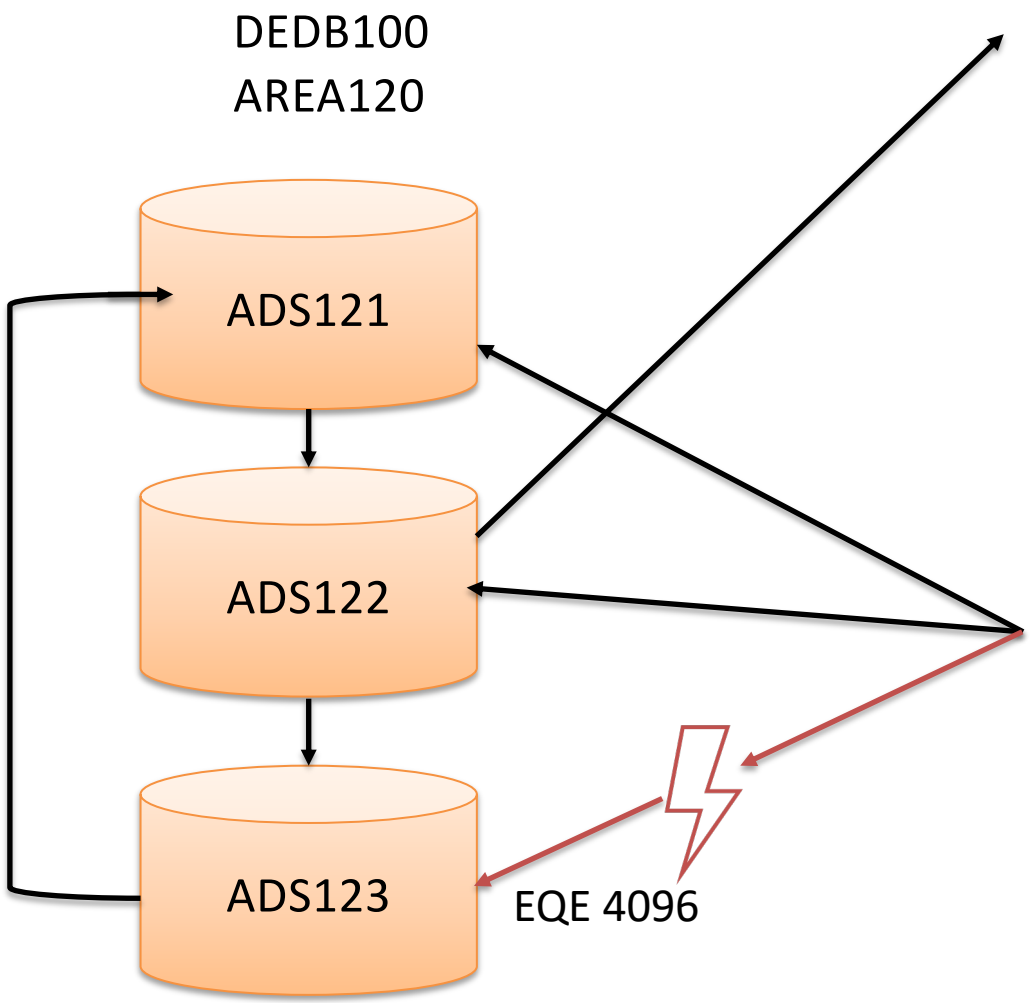
PCB . . .,PROCOPT=A

SENSEG . . ., SSPTR=((1,R),(3,U))

Program uses command codes:

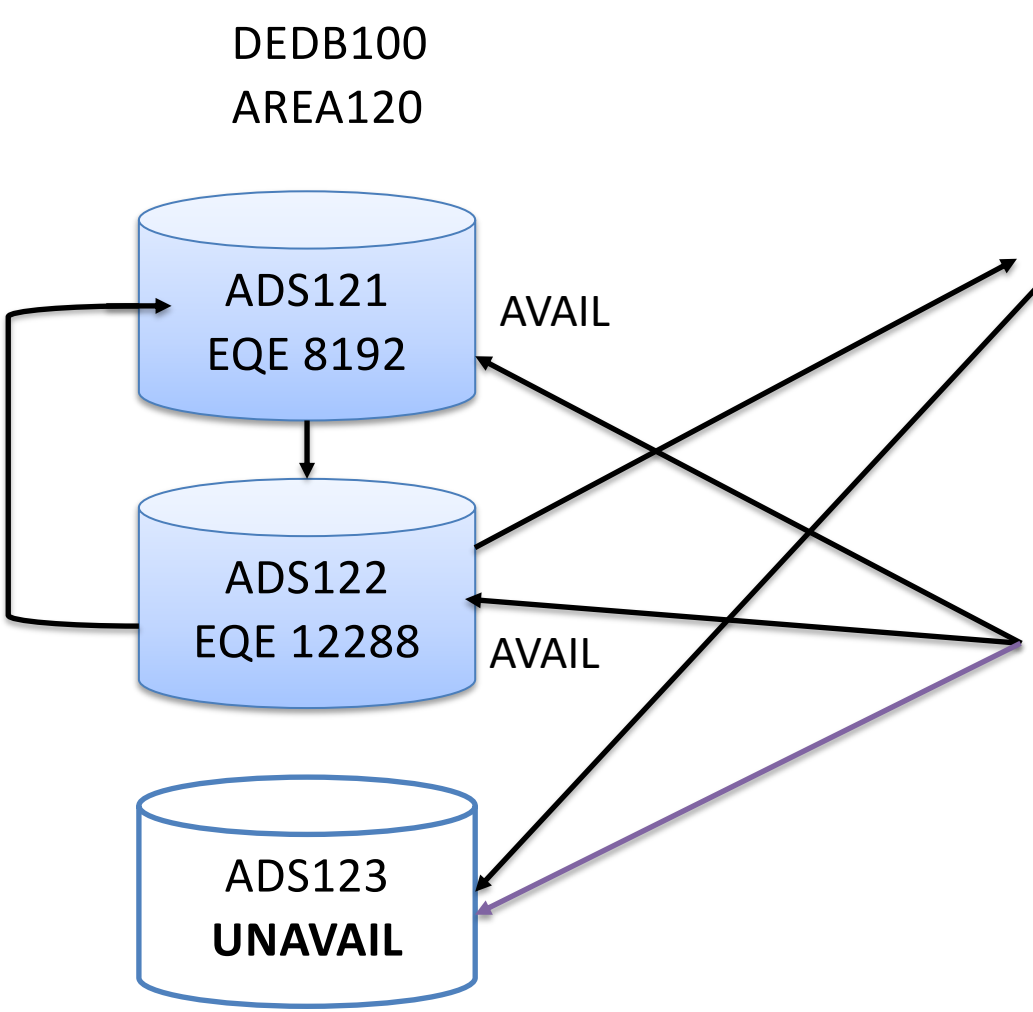
- Set a given ptr (S3 | W3)
- Move ptr to next segment (M3)
- Zero ptr (Z3)
- Use ptr to get/isrt/dlet (R1)

MULTIPLE AREA DATA SETS – UP TO 7



1. READ ANY CI 4096 from 121 | 122 | 123
2. UPDATE CI 4096 IN BUFFER
3. COMMIT UPDATE
4. WRITE ALL - WRITE ERROR ON ADS123
5. CI2 in 123 updated = EQE 4096
6. READ ANY CI 4096 **now from 121 or 122**

MADS CREATE UTILITY: ONLINE RECOVERY

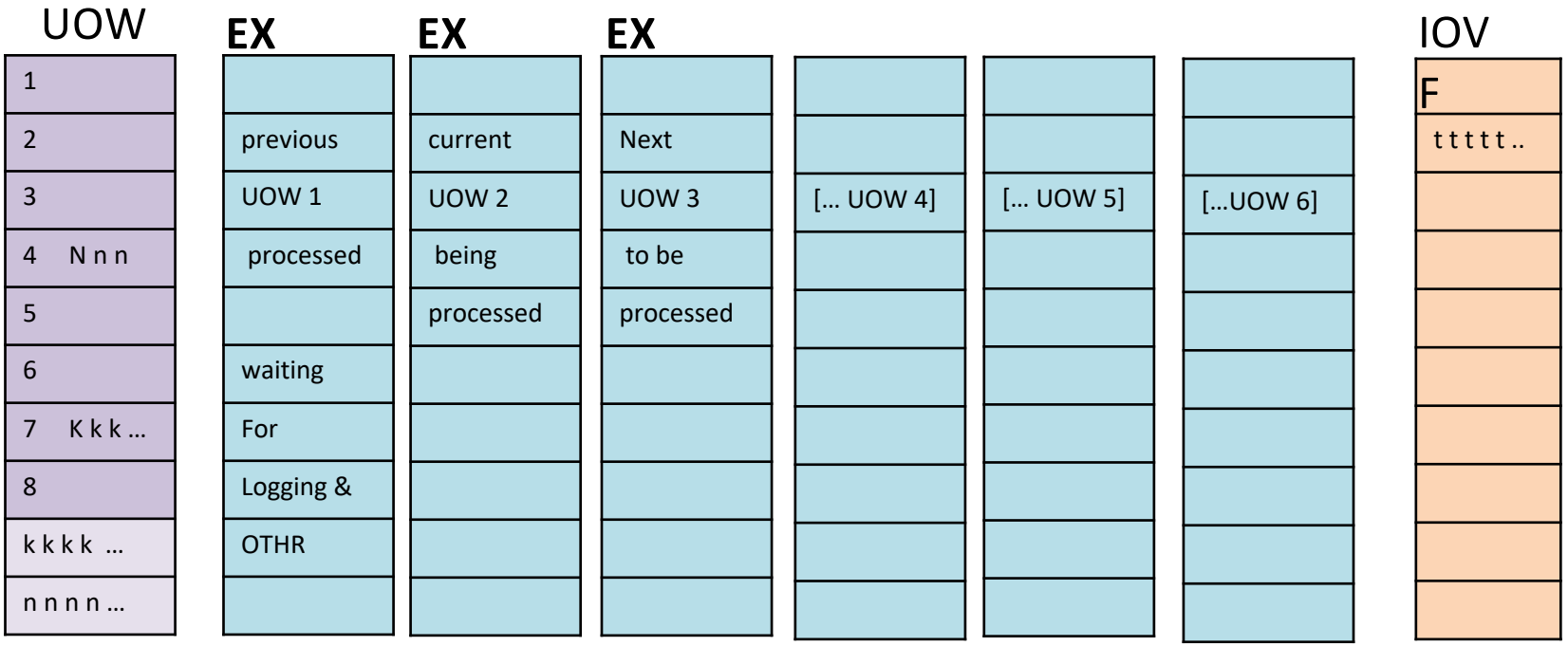


CREATE ADS123

1. INIT ADS123
2. READ ANY / **WRITE SPECIFIC ADS123**
3. ONLINE WORK IFP/MPP/BMP WRITES ALL AT COMMIT
4. CREATE COMPLETION - SET ADS123 'AVAIL'

HIGH SPEED SEQUENTIAL PROCESSING: HSSP

3 up to 6 Buffer SETs = # CIs / UOW



1. BMP PCB has PROCOPT=H
2. Control data set can specify NORDAH
3. EXCLUSIVE LOCK on UOW
4. Conditional LOCK request on next UOW
5. LOCK release after
   a. Physical logging
   b. OTHR processing

Uses NBA / OBA BUFFERS

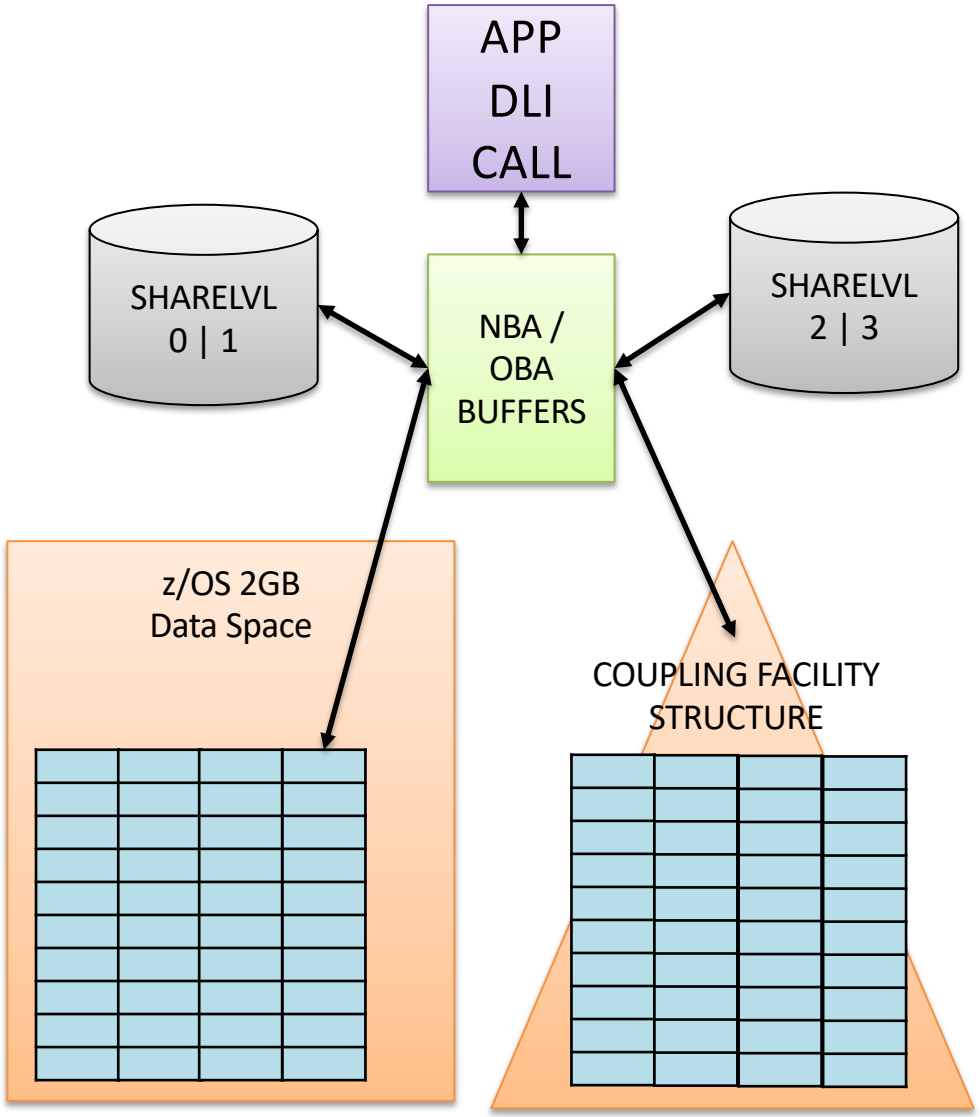
VIRTUAL STORAGE OPTION: VSO

OBJECTIVES:

- Reduces READ I/O
  - Data in memory
  - Moved into NBA/OBA for APP
- Reduces WRITE I/O
  - Output logged
  - DS | STR updated
  - Lock released
  - Batch output from DS | STR to ADS
- Decreases LOCK contention
- Alternative to MSDB

RESTRICTIONS

- Registered with DBRC
- Uses 2GB z/OS Data Spaces
  - SHARELVL (0 | 1)
- Uses Coupling Facility Structure
  - SHARELVL (2 | 3)



VSO DBRC PARAMETERS

INIT.DB | CHANGE.DB -

VSO | NOVSO -
PREOPEN | NOPREO -
PRELOAD | NOPREL –

/* VSO SHARELVL(2 | 3) only */

CFSTR1(cfstructure1name) -
CFSTR2(cfstructure2name) -
MAS | NOMAS -
LKASID | NOLKASID

DEDB ONLINE UTILITIES

- Execute in dependent regions (FPU)
  - ✓ Increases data availability
  - ✓ One utility at a time against an AREA
  - ✓ Do not use Fast Path database buffer pool
- DBFUMSC0: Sequential Dependent SCAN
- DBFUMDL0: Sequential Dependent DELETE
- DBFUMMH0: MADS Area Data Set Compare
- DBFUMRI0: Area Data Set Create
- DBFUHDR0: High Speed DEDB Direct Reorganization
- ALTER: Dynamically change DEDB specifications

ALTER UTILITY FUNCTIONS

- **ALTERAREA**: Alter a DEDB area's UOW, SIZE, ROOT
- **REPLRAND**: Replace the two-stage randomizer
- **ADDAREA**: Add areas to DEDBs
  - ✓ Ability to add Segment/Compression Exit Routine
  - ✓ *Reads or updates new DEDBs from IMS catalog if IMS Managed ACBs*
- **ALTERDB:**
  - ✓ Adding new fields in unmapped space of one or more existing DEDB segments
  - ✓ Increasing segment length of one or more variable-length fields in a DEDB segments without outage to users

ALTER UTILITY

Introduced in IMS V13: For DEDBs *without SDEPs*

Enhanced in IMS V14: For DEDBs with SDEPs

Enhanced with IMS V14 Continuous Delivery (SPE): IMS Catalog environment support added

Enhanced in IMS V15: ALTERDB function added

Summary: DEDBs ARE UNIQUE

- System Functions
  - Buffers
  - Locking
  - Logging
  - DASD updates
- Hierarchic Structure
- Database record storage
- Randomizing
- Sequential Dependents
- Subset Pointers
- Multiple Area Data Sets
- High Speed Sequential Processing
- Virtual Storage Option
- Online Utilities

What did I forget?

Overview of Fast Path DEDBs

Thank you . . .

KAREN TISCHER
IMS Education & Consulting
ktischer@mac.com