B08

Fast Path Data Entry Database Fundamentals

Bill Stillwell, Dallas Systems Center
stillwel@us.ibm.com

IMS Technical Conference

Miami Beach, FL October 22-25, 2001

RETURN TO INDEX

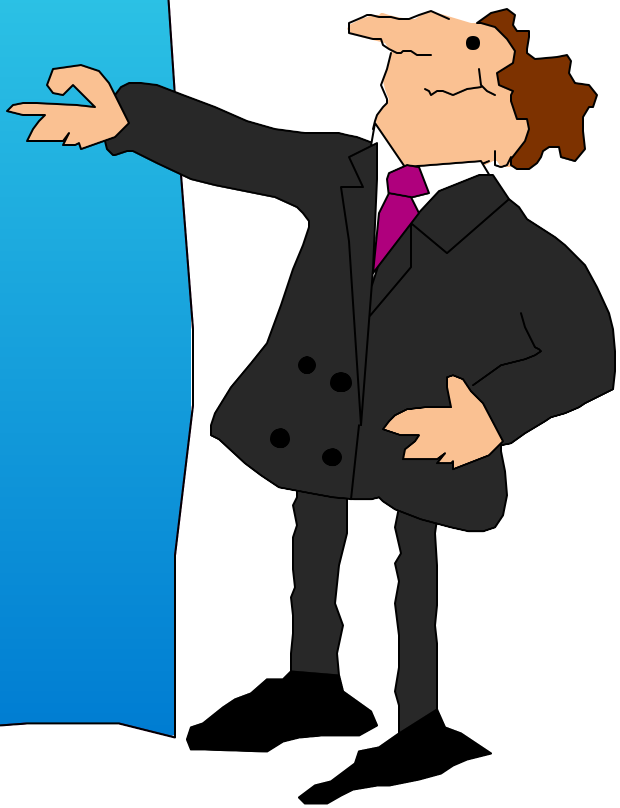
Agenda

Audience

- ★ ***New users*** of the Data Entry Database

Topics

- ★ Fast Path Components
  - ✓ EMH
  - ✓ MSDB
  - ✓ DEDB
- ★ DEDB Fundamentals
  - ✓ Partitioning
  - ✓ Structure and Topology
  - ✓ Space Management
  - ✓ DEDB System Functions



Fast Path Components (EMH)

Expedited Message Handler

- Delivered with IMS Transaction Manager

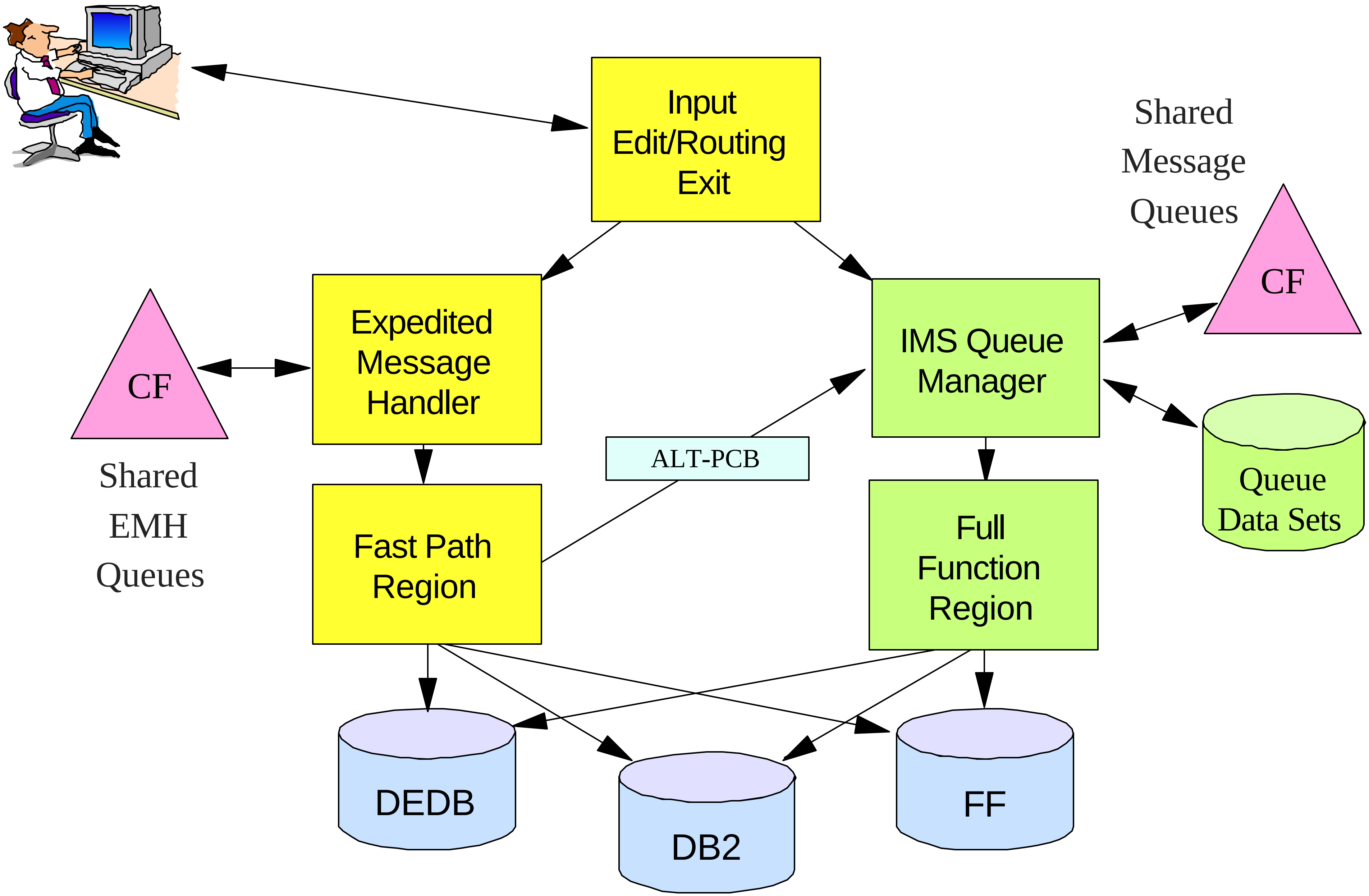
EMH Characteristics

- Alternative to full function message *queuing and scheduling*
- Handles fast path transactions
  - Defined as FPATH in SYSGEN
- Very high performance
  - 1000s per second
  - Scheduled in IFP regions (WFI)
  - Simplified queuing and scheduling
    - No priorities, classes, or other options
- Supports shared queues
  - Shared EMH structure

```
APPLCTN
    PSB=FPPSB1,...,
    FPATH=YES

TRANSACT
    CODE=FP1,...,
    FPATH=YES
```

Fast Path Components (EMH) ...



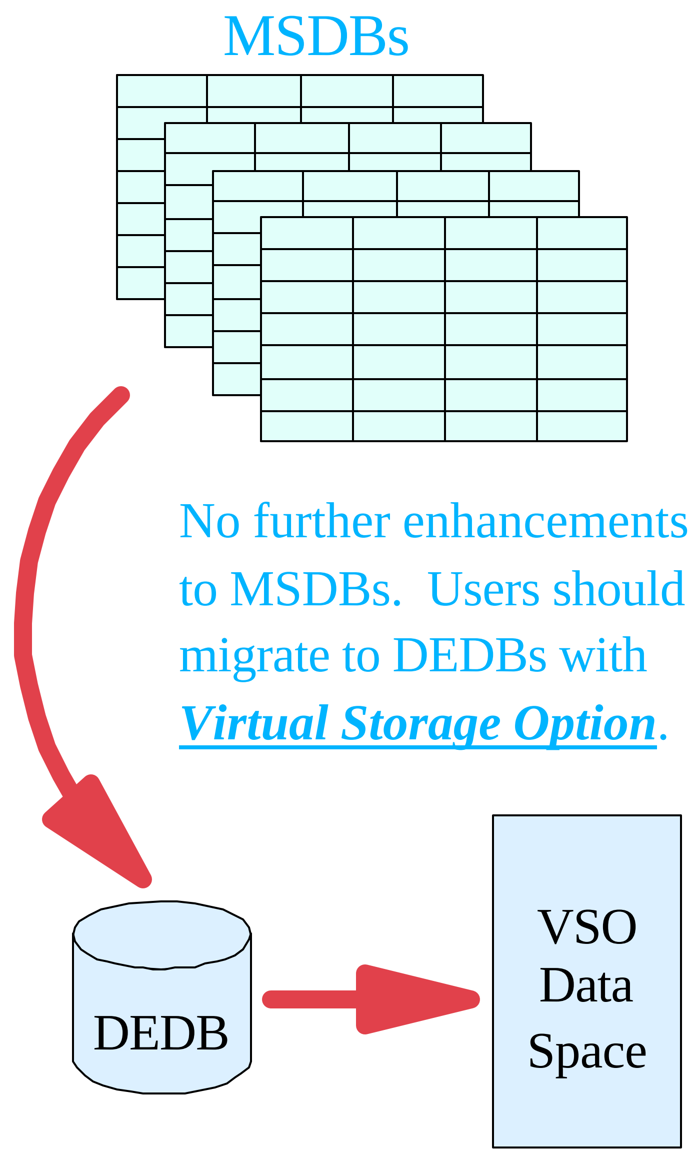
Fast Path Components (MSDB)

Main Storage Database

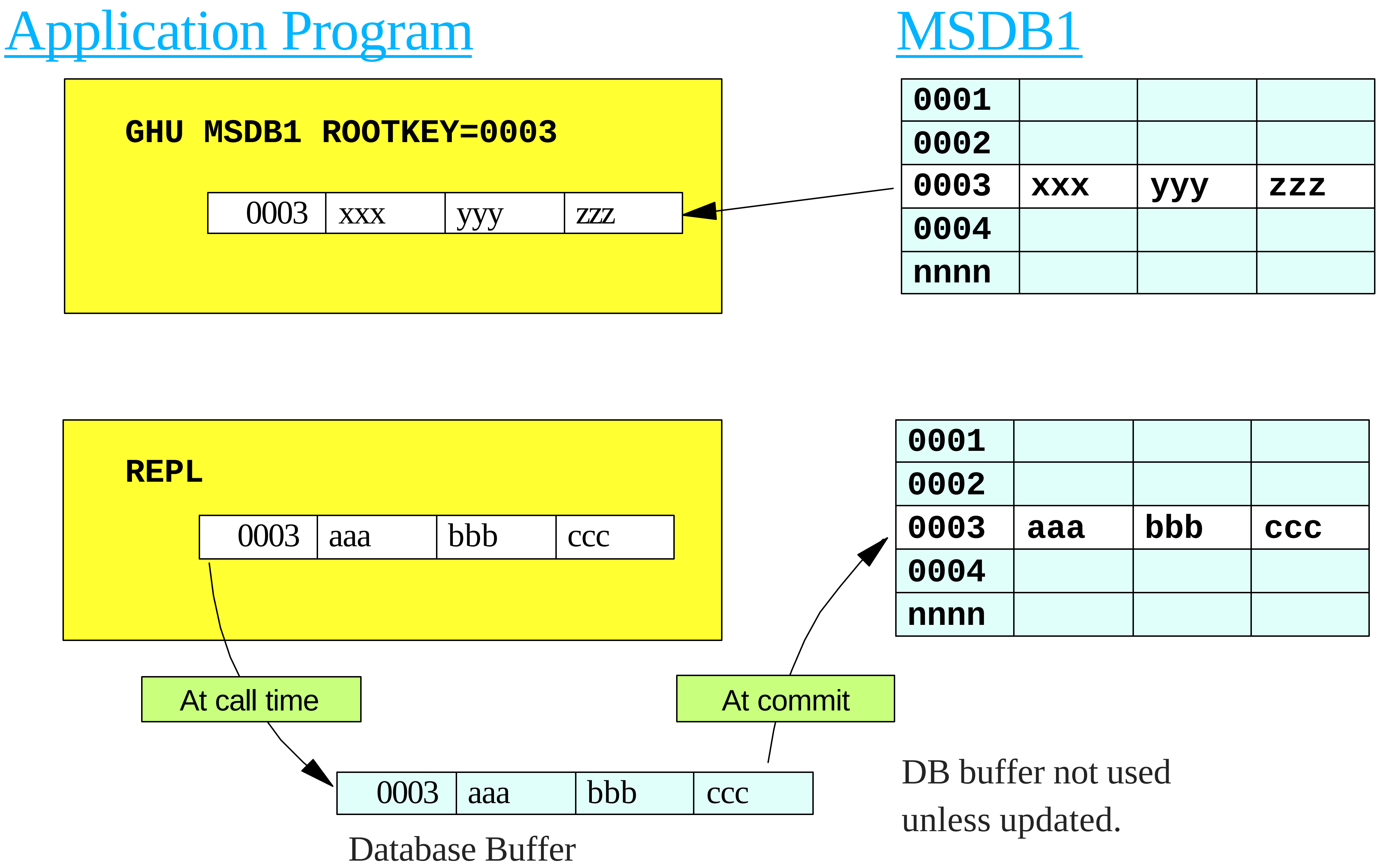
- Delivered with Database Manager
- Available only in IMS TM environment

MSDB Characteristics

- Resident in main storage (ECSA)
- Root-only fixed length segments
- Full database integrity
  - Non-standard backup and recovery
- Limited update capability
  - No online ISRT/DLET
  - No update by APPC/OTMA transactions
  - No data sharing support



Fast Path Components (MSDB) ...



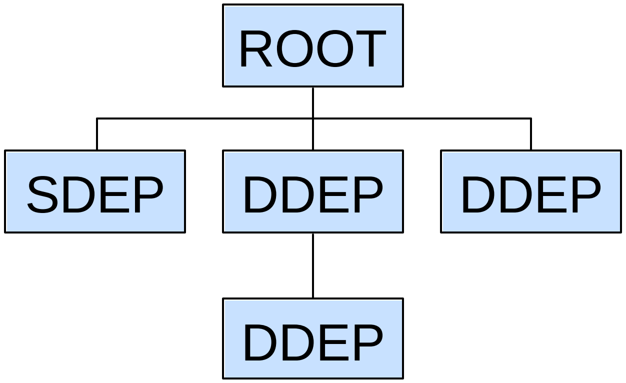
Fast Path Components (DEDB)

Data Entry Database

- Delivered with IMS/ESA DB Manager

- Available to ...
  - IMS MPPs, IFPs, and BMPs
  - CICS/DBCTL
  - Open Database Access (ODBA)

- Not available to ...
  - IMS Batch



Fast Path Components (DEDB) ...

DEDB Characteristics

- DASD-resident
  - With Virtual Storage Option (VSO)
- Direct access database (HDAM-like)
  - Uses randomizer
- Partitioned
  - Up to 240 Areas
- Sequential dependent segment type
  - Unique to DEDBs
- Supports large databases (960 GB)
- Replicated data (up to 7 copies)
- Online utilities (reorganization, ...)

DEDB Partitioning

A DEDB can be divided into multiple partitions

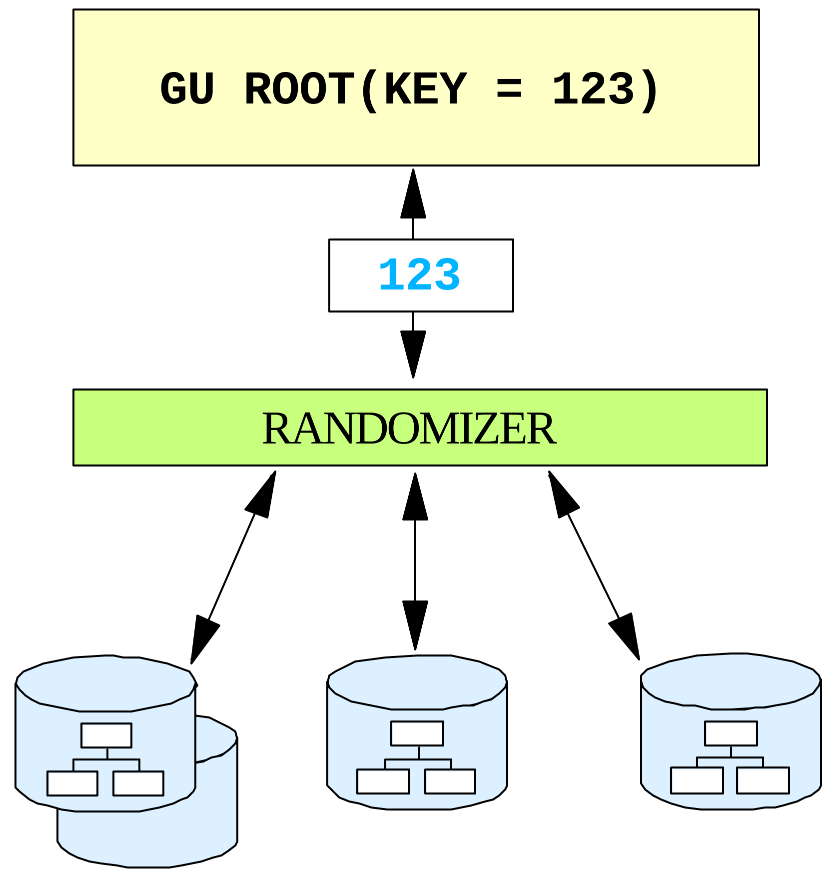
- Up to 240 "Areas"

Each AREA resides in an Area Data Set (ADS)

- VSAM ESDS
- 4 GB per Area
  - 4 GB x 240 = 960 GB

Single database image

- Single PCB for database
- Area transparent to application
- Area determined by randomizer
  - Based on root key



DEDB Partitioning ...

Define Areas and Randomizer in DBD

```
DBD
NAME=ACCTDB,ACCESS=DEDB,RMNAME=ACCTRAND
AREA   ...
AREA   ...
AREA   ...
```

KEY ⟶ Randomizer ⟶ AREA + RAP

Root key may be designed to identify

- Customer set (commercial, individual, ...)
- Geography (east, west, central, US, Europe, ...)
- Date/time (yyyymm)
- Billing cycle (1, 2, 3, ..., 12)
- Record size (small, normal, very large)

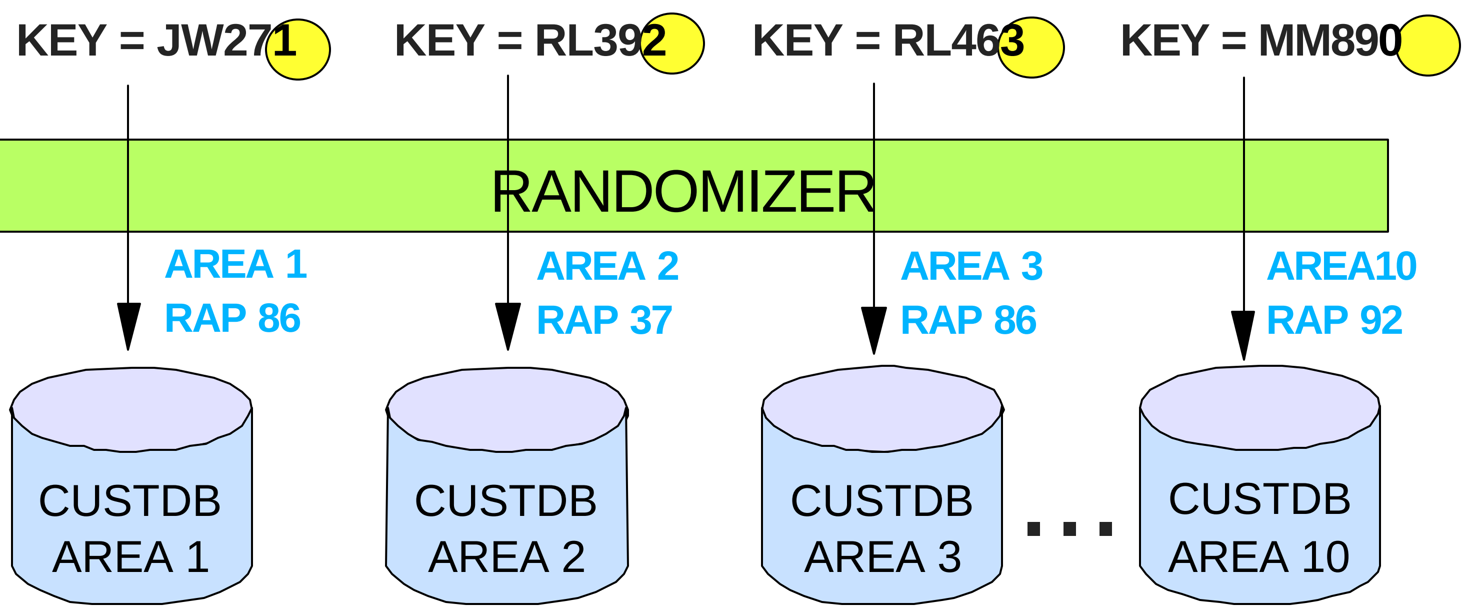
DEDB Partitioning ...

Or, if key can't be "designed"

- Use key range (e.g. 1-12500, 12501-25000, ...)
- Use arbitrary value (e.g. last digit)

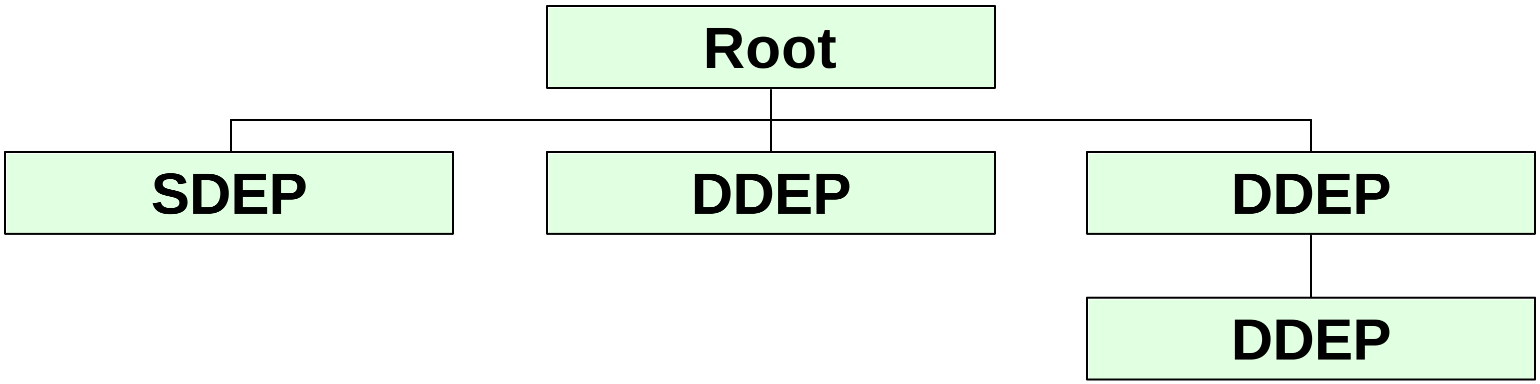
Example

- Randomizer assigns area based on last digit of key



Record Structure

Record Structure



Similar to HDAM

- Direct access (uses randomizer)
- 15 levels, 127 segment types

Segment length

- Fixed or variable length segments

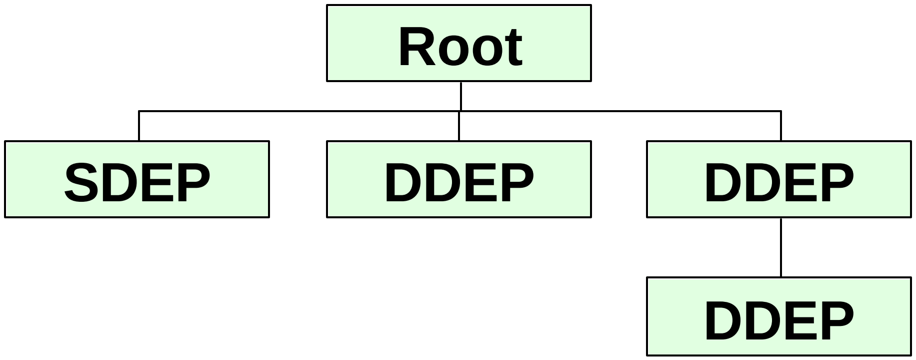
Record Structure ...

Root segment

- Unique key only

Direct Dependents (DDEP)

- Like HDAM dependents
- Unique key or no key



Sequential Dependent (SDEP)

- No key
- No HDAM equivalent

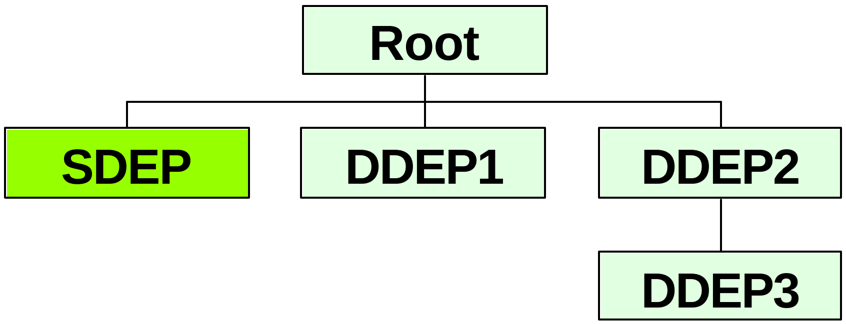
Not supported

- Secondary indexes
- Logical relationships

Sequential Dependents

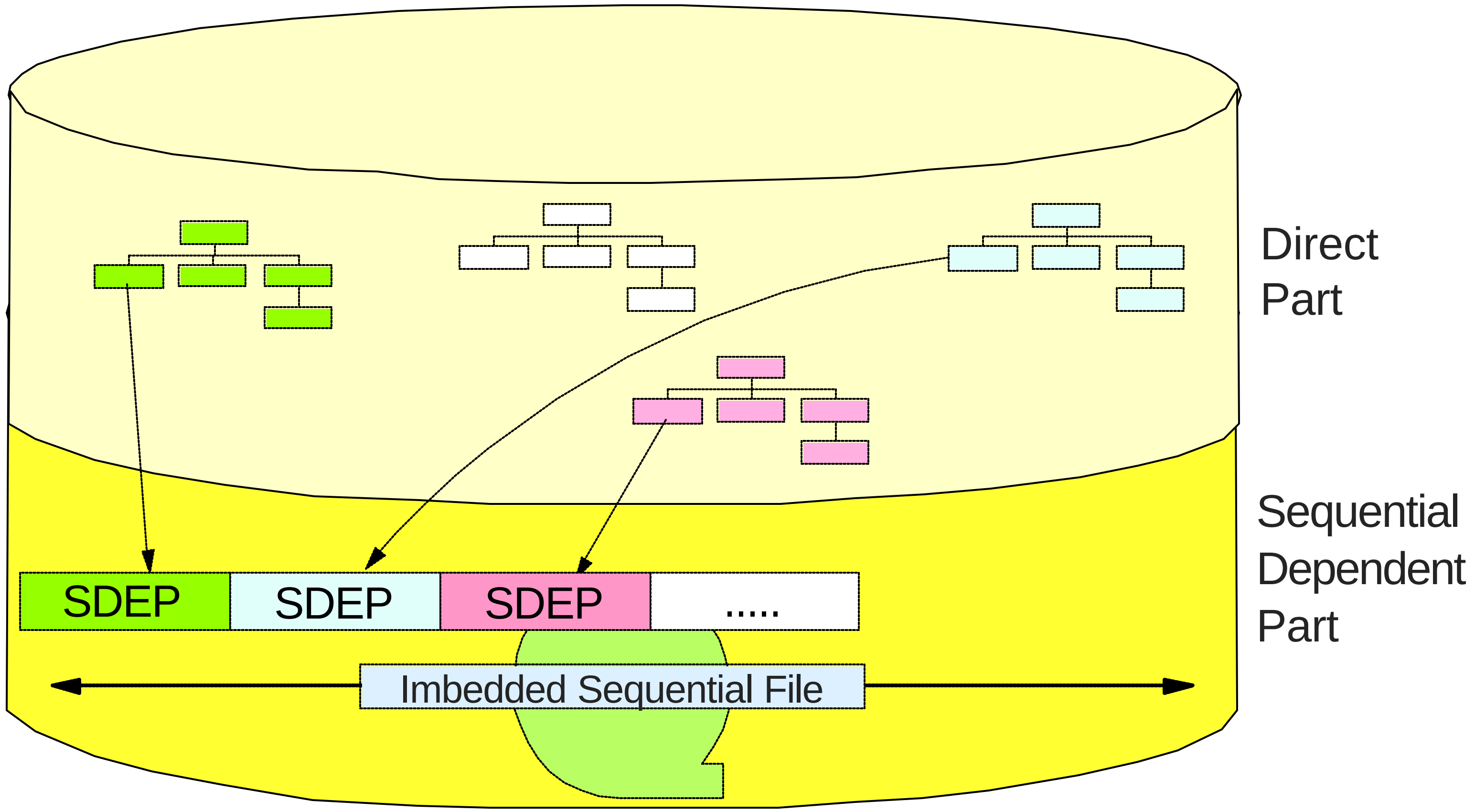
One *optional* Sequential Dependent Segment type

- May have twins but no children
- Must be defined as first dependent
- No sequence field
  - Segments inserted LIFO under root
- Stored at *end of area data set*
  - Not with ROOT and DDEPs




```
DBD    NAME=ACCTDB,ACCESS=DEDB,RMNAME=ACCTRAND
AREA   ...
SEGM   NAME=ROOT,PARENT=0,...
SEGM   NAME=SDEP,PARENT=ROOT,...,TYPE=SEQ
SEGM   NAME=DDEP1,PARENT=ROOT,...,TYPE=DIR
```

Sequential Dependents ...

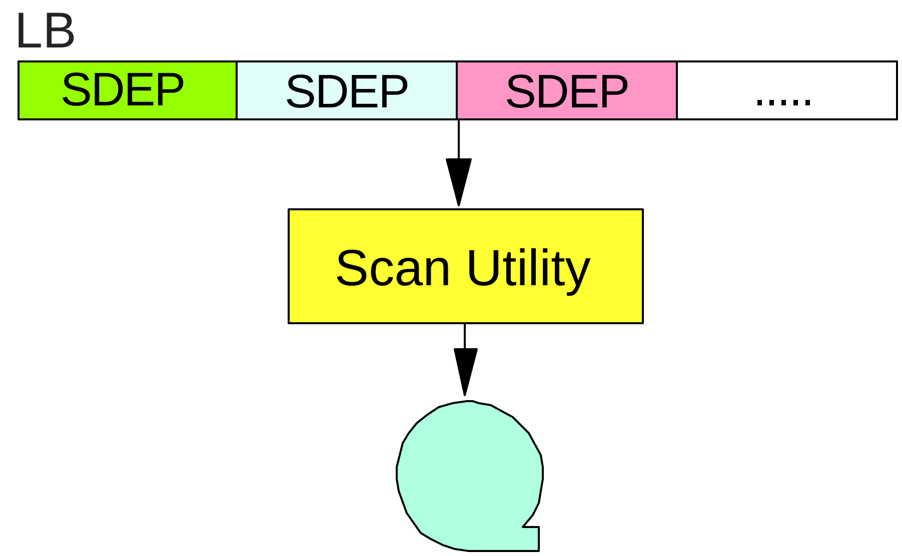


When SDEP is inserted, it occupies the **next available position** in the SDEP part. Segments are in **chronological** sequence.

Sequential Dependents ...

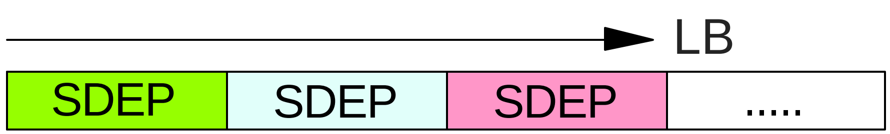
Application programming

- GU, GN, GNP, and ISRT only
- Application cannot REPL or DLET

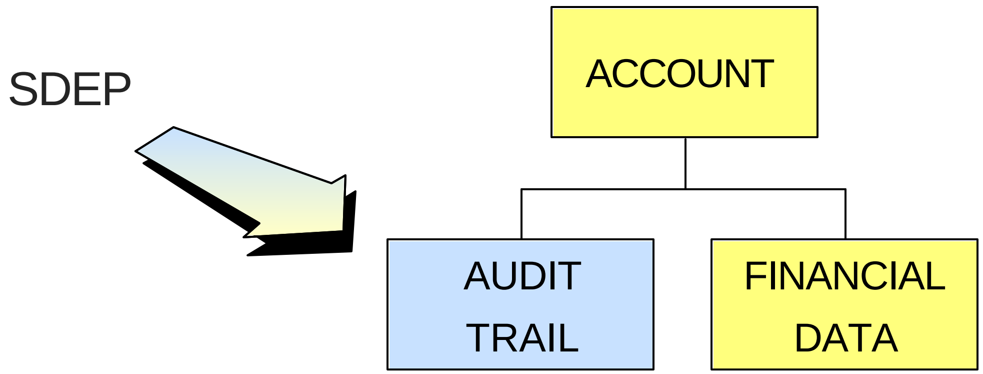


SDEP Online Utilities

- SDEP Scan Utility
  - Reads SDEPs in chronological sequence
  - Generates sequential output file
- SDEP Delete Utility
  - Deletes SDEP segments
  - Moves LB pointer in DMAC forward



SEP Audit Trail Example



When FINANCIAL data is updated

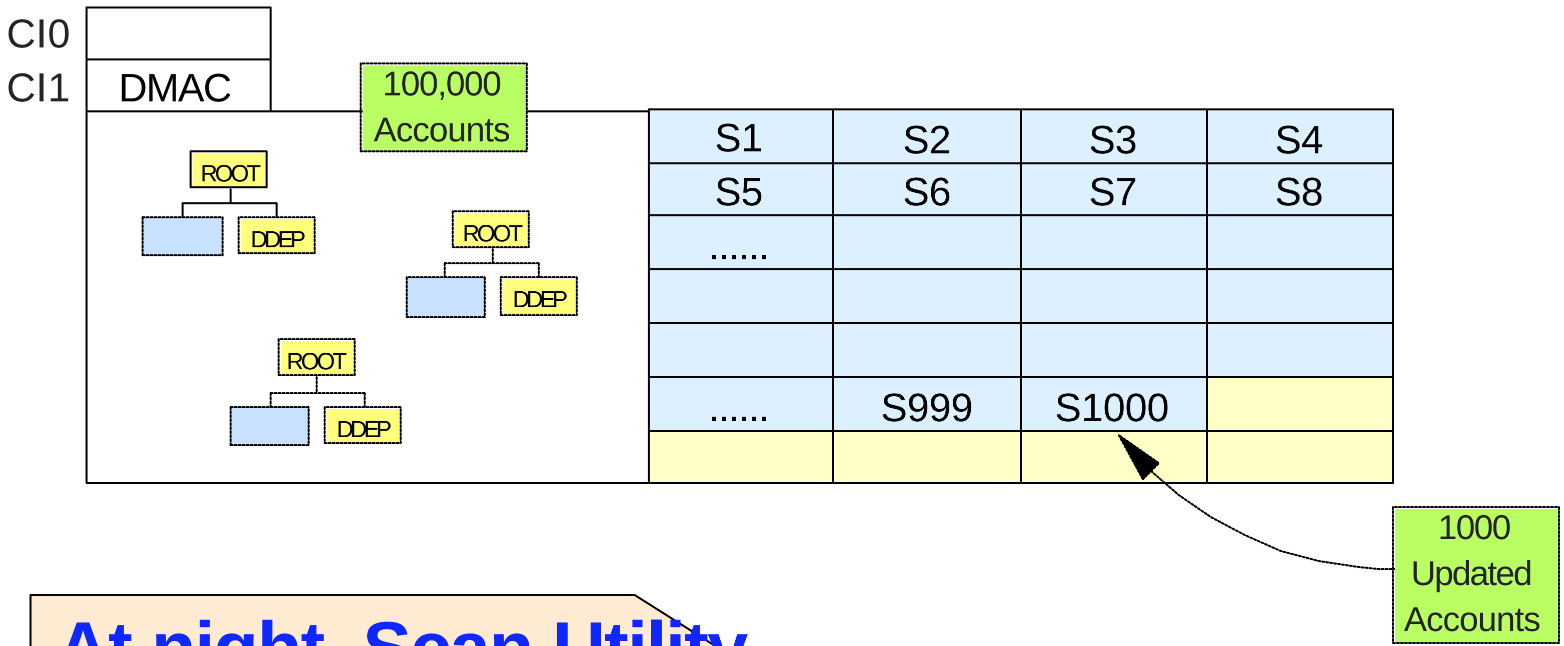
- Insert SDEP with audit information

Audit segments are inserted sequentially into sequential dependent part of Area

- When retrieved, they will be in chronological sequence, regardless of which root segment (Account) they were inserted under

Account Number	Date Time	USERID	Action	Amount ($)	ETC.

SEP Audit Trail Example ...



At night, Scan Utility

- Reads audit trail
- Creates sequential data set

User written batch program

- Produces report and/or history tape

More on SDEPs in Session A37

Area Topology

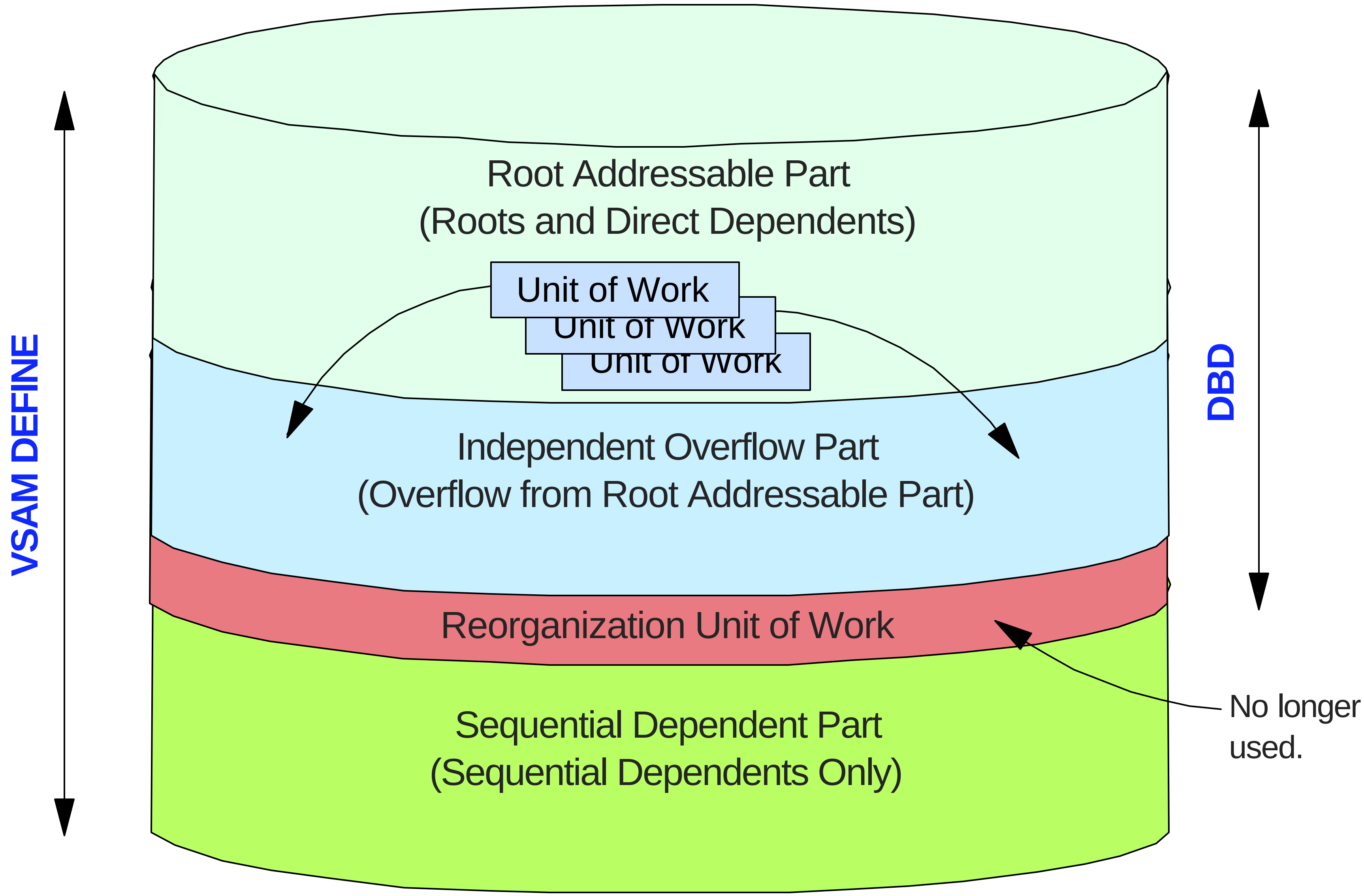
VSAM DEFINE determines ADS size

- Each Area Data Set (ADS) defined as VSAM ESDS
- I/O processed by DFP Media Manager

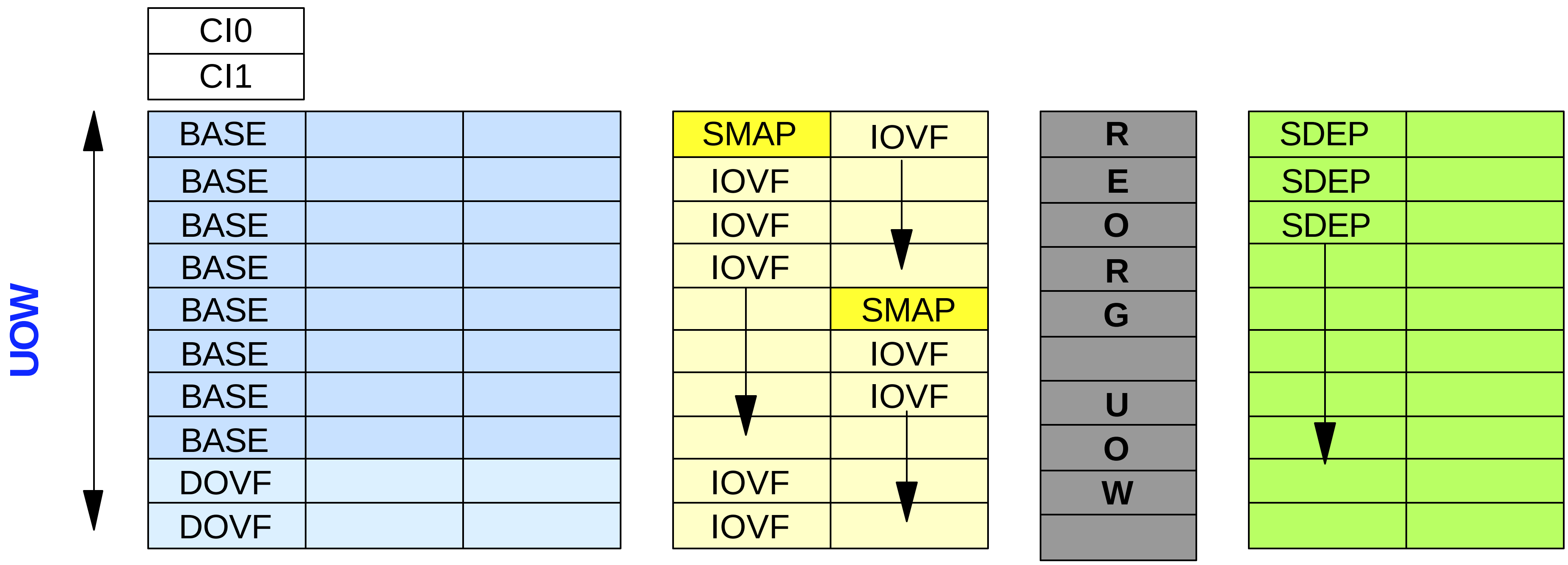
DBD determines ...

- Number of Areas
- Number of Root Anchor Points (RAPS)
- Allocation of space within Area

Area Topology ...



Area Terminology

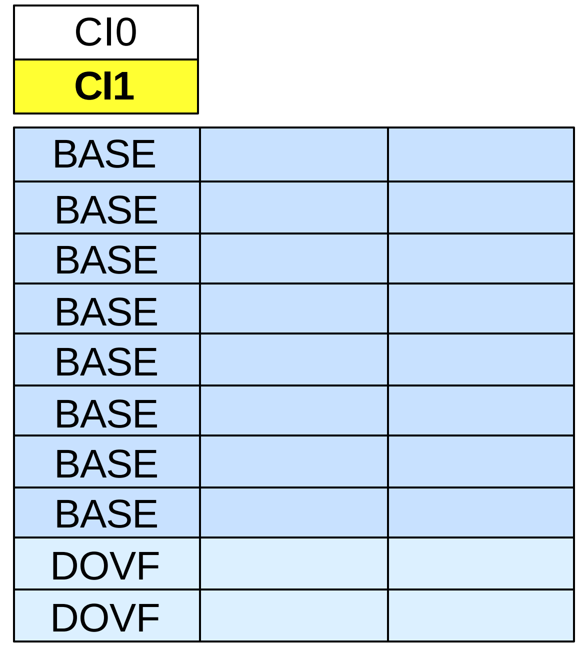


CI0	**Control Record**
CI1	**Contains Area Control Block (DMAC) and Error Queue Elements (EQEs)**
UOW	**Unit of Work (BASE and DOVF CIs)**
BASE	**Only CIs with RAPs (also called RAP CIs)**
DOVF	**Dependent Overflow (for UOW only)**
IOVF	**Independent Overflow (when DOVF is full)**
SMAP	**Space Map (monitors free space in IOVF)**
REORG	**Reorganization Unit of Work (no longer used, but still allocated in ADS)**
SDEP	**Sequential Dependents**

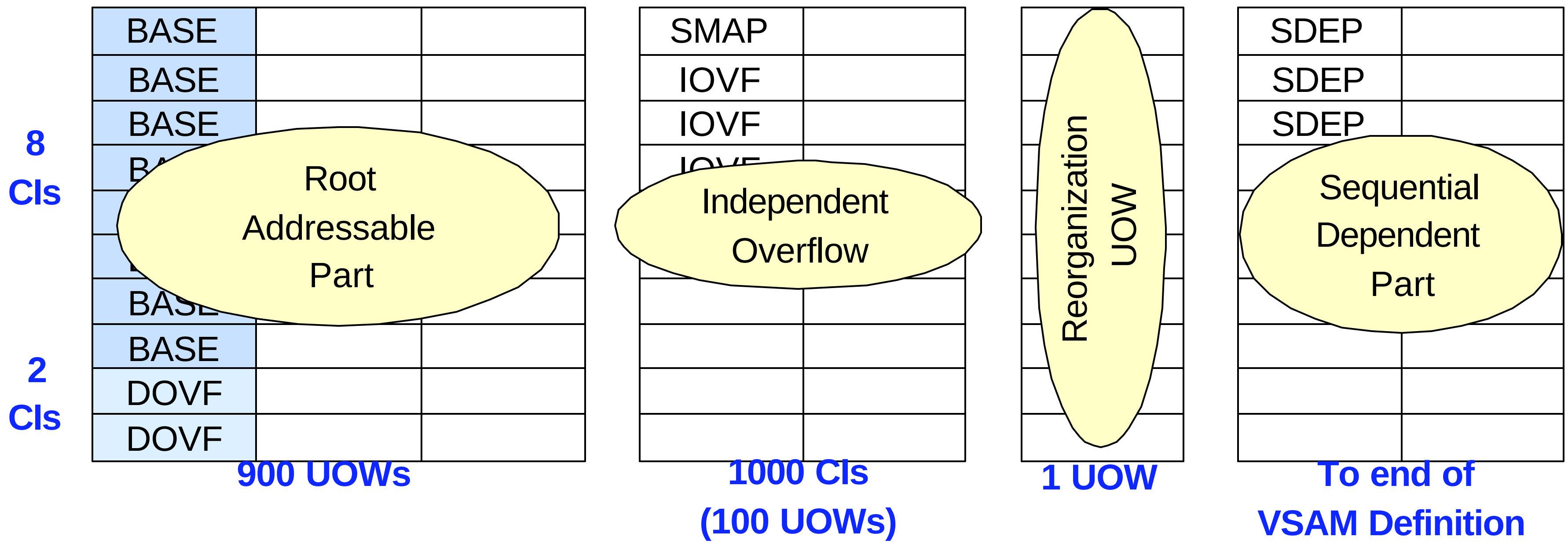
DMAC

DMAC - Area Control Block

- Kept in 2nd CI of each Area
- Physical description of Area
  - Information from DBD
- Sequential Dependent information
  - Physical beginning and physical end
  - Logical beginning and logical end
  - Cycle count
- Status of online utilities
  - What's currently running
  - Did anything fail?
- Miscellaneous headers and other fields



Area Space Allocation



DBD determines space allocation within Area

```
DBD     NAME=FPDEDB,ACCESS=DEDB,RMNAME=DEDBRAND

AREA    DD1=AREA1,...
AREA    DD1=AREA2,SIZE=4096,UOW=(10,2),ROOT=(1000,100)
AREA    DD1=AREA3,...

SEGM    (Same segment structure for all areas)
```

Units of Work (UOWs)

AREA ...,SIZE=1024,UOW=(10,3),ROOT=(6,2)

CI0					
CI1		UOW			
RAP0	RAP7	RAP14	RAP21	SMAP1	IOVF10
RAP1	RAP8	RAP15	RAP22	IOVF1	IOVF11
RAP2	RAP9	RAP16	RAP23	IOVF2	IOVF12
RAP3	RAP10	RAP17	RAP24	IOVF3	IOVF13
RAP4	RAP11	RAP18	RAP25	IOVF4	IOVF14
RAP5	RAP12	RAP19	RAP26	IOVF5	IOVF15
RAP6	RAP13	RAP20	RAP27	IOVF6	IOVF16
DOVF1	DOVF1	DOVF1	DOVF1	IOVF7	IOVF17
DOVF2	DOVF2	DOVF2	DOVF2	IOVF8	IOVF18
DOVF3	DOVF3	DOVF3	DOVF3	IOVF9	IOVF19

4 UOWs (6 - 2)

10 CIs per UOW

- **7 RAP CIs**
- **3 DOVF CIs**

20 IOVF CIs

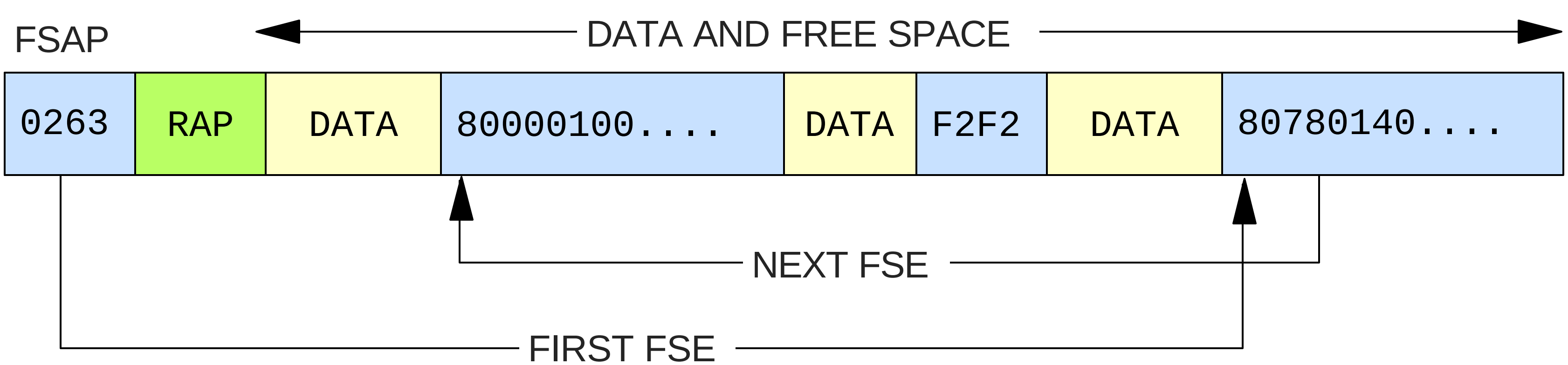
- **1 SMAP CI**
- **19 OVF CIs**

The concept of a **Unit of Work** is very important for both space management and application processing.

Space Management

All roots randomize to RAP CIs

- Never to any other type of CI
- No RAP CI contains data which randomized to another RAP CI
  - No "cascading" as with HDAM
- Space within RAP CI managed like HDAM
  - Free space anchor point (points to first free space element)
  - Free space elements (chained if necessary)
  - Root anchor point (only one per CI)
    - Roots chained off RAP
  - Data (roots and direct dependents)



Space Management ...

If RAP CI full

- ► Data overflows to DOVF

Dependent Overflow CI (DOVF)

- ► Allocated to Unit of Work (UOW)
- ► Contains overflow data from RAP CIs in that UOW only
  - – Never contains overflow data from another UOW
  - – May contain overflow data from multiple RAP CIs
- ► DOVF CIs filled in sequence
  - – 1st DOVF CI identifies current overflow CI

Space Management ...

RAP0	RAP7	RAP14	RAP21	SMAP1	IOVF10
RAP1	RAP8	RAP15	RAP22	IOVF1	IOVF11
RAP2	RAP9	RAP16	RAP23	IOVF2	IOVF12
RAP3	RAP10	RAP17	RAP24	IOVF3	IOVF13
RAP4	RAP11	RAP18	RAP25	IOVF4	IOVF14
RAP5	RAP12	RAP19	RAP26	IOVF5	IOVF15
RAP6	RAP13	RAP20	RAP27	IOVF6	IOVF16
DOVF1	DOVF1	DOVF1	DOVF1	IOVF7	IOVF17
DOVF2	DOVF2	DOVF2	DOVF2	IOVF8	IOVF18
DOVF3	DOVF3	DOVF3	DOVF3	IOVF9	IOVF19

DOVF1, DOVF2, and DOVF3 may contain

- **Overflow data from RAP0 through RAP6**

They will never contain

- **Overflow data from RAP7 through RAP20**

Space Management ...

Two types of Independent Overflow CIs

- Space Map CIs (SMAP)
  - Each SMAP CI keeps track of the next 119 IOVF CIs
    - Which ones are empty and which have been used
- Overflow CIs (IOVF)

RAP0	RAP7	RAP14	RAP21	SMAP1	IOVF10
RAP1	RAP8	RAP15	RAP22	IOVF1	IOVF11
RAP2	RAP9	RAP16	RAP23	IOVF2	IOVF12
RAP3	RAP10	RAP17	RAP24	IOVF3	IOVF13
RAP4	RAP11	RAP18	RAP25	IOVF4	IOVF14
RAP5	RAP12	RAP19	RAP26	IOVF5	IOVF15
RAP6	RAP13	RAP20	RAP27	IOVF6	IOVF16
DOVF1	DOVF1	DOVF1	DOVF1	IOVF7	IOVF17
DOVF2	DOVF2	DOVF2	DOVF2	IOVF8	IOVF18
DOVF3	DOVF3	DOVF3	DOVF3	IOVF9	IOVF19

Space Management ...

If RAP CI and all DOVF CIs are full

- Allocate a CI from independent overflow (IOVF)
  - Use Space Map CI to find empty IOVF CI
- Once IOVF selected, it "belongs" to that UOW
  - Will contain overflow data from only one UOW
- Process repeats as IOVF CIs fill up

RAP0	RAP7	RAP14	RAP21	SMAP1	IOVF10
RAP1	RAP8	RAP15	RAP22	IOVF1	IOVF11
RAP2	RAP9	RAP16	RAP23	IOVF2	IOVF12
RAP3	RAP10	RAP17	RAP24	IOVF3	IOVF13
RAP4	RAP11	RAP18	RAP25	IOVF4	IOVF14
RAP5	RAP12	RAP19	RAP26	IOVF5	IOVF15
RAP6	RAP13	RAP20	RAP27	IOVF6	IOVF16
DOVF1	DOVF1	DOVF1	DOVF1	IOVF7	IOVF17
DOVF2	DOVF2	DOVF2	DOVF2	IOVF8	IOVF18
DOVF3	DOVF3	DOVF3	DOVF3	IOVF9	IOVF19

Unit of Work Concept

So what does this do for us?

- Locking
  - Entire UOW can be locked
- High Speed Sequential Processing (HSSP)
  - For BMP which process Area sequentially
    - Lock UOW
    - Read entire UOW into buffers
    - Lock and read ahead next UOW
  - BMP notified when UOW boundary crossed
    - GN would get segment from next UOW
    - Program gets GC status code, must take checkpoint
    - Next UOW already in buffers
  - More on HSSP in Session A37 (DEDB Advanced Topics)
- Online Reorganization by UOW
  - Lock UOW, reorganize it, release lock, go on to next UOW

System Functions for DEDBs

Some system functions are handled differently for DEDBs than for full function databases

- Buffer management
- Locking
- Logging
- Commit processing
- Output thread processing

Buffer Management

DEDB global buffer pool

- Single pool
- Single buffer size
- Used by all DEDBs

FPCTRL DBBF=10000, BSIZ=8192,....

Dependent region (D/R) local buffer pool

- Defined in D/R JCL
- Carved out of global pool
- Normal buffer allocation (NBA)
  - Preallocated for D/R use
- Overflow buffer allocation (OBA)
  - Allocated to D/R when NBA not enough

//EXEC ...,NBA=100,OBA=200

Buffer Management ...

Buffer utilization by dependent regions

- D/R can NOT use more than NBA + OBA buffers
  - Else U1033 abend (or FR status for BMP)
- Buffers can be stolen and reused
  - If buffer unaltered by D/R
- If D/R needs more than NBA
  - Get OBA latch
  - Use OBA buffers

Buffers used by system

- Sequential dependent buffers

Altered buffers

- Not written to DASD until committed
- Committed buffers written to DASD by output threads

Locking

Non-block level data sharing environment

- Local lock manager may be
  - PI or IRLM for full function resources
  - Fast path *and* PI or IRLM for FP resources
- Fast path manages its own locks
- Invokes PI (or IRLM) only if WAIT condition occurs
  - WAIT may be part of deadlock
  - FP cannot detect deadlock between IMS and FP resources
  - PI (or IRLM) performs deadlock detection

Block level data sharing environment

- Global lock manager is IRLM
  - All locking for shared DEDBs managed by IRLM
- FP manages locks for non-shared DEDBs

Locking ...

DEDB resources locked

- Control Interval
  - When referenced by application
  - Held until buffer stolen or D/R reaches sync point
- UOW
  - All CIs in UOW locked with single lock
  - Used by HSSP and High Speed Reorganization Utility
- Miscellaneous other locks

Latches

- OBA latch
  - Acquired when D/R exceeds NBA
  - Serializes OBA usage
- Miscellaneous others

Logging

When DEDB updated

- Update made in buffers
- Location of changed data "remembered" in buffer header
  - Buffer Alteration Table
- Altered buffer cannot be stolen
  - Changes not yet logged - no backout possible

When application commits

- Log records built from information in Buffer Alteration Table

If application fails before commit

- All altered buffers discarded
  - No need to do backout

Fast path log records

- x'59xx'

Logging ...

Logical logging

- Creation of log record and placing log record in OLDS buffer
  - x'5950' - DEDB updates
  - Done during sync point processing
- Last log record is x'5937' sync record

Physical logging

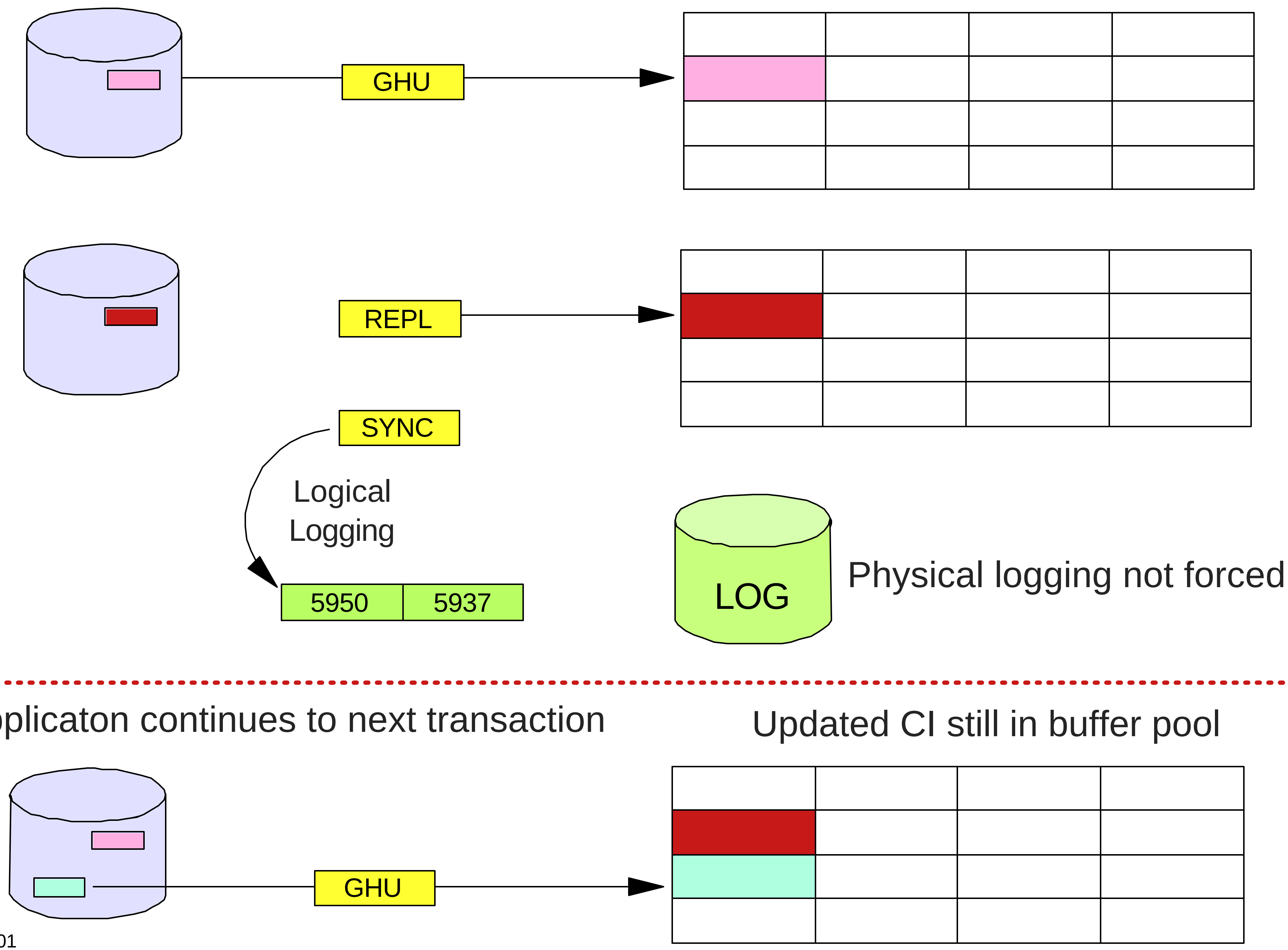
- Writing of log data to OLDS or WADS
- Fast path does NOT checkwrite log records
  - Waits until buffer full
- When physical logging of x'5937' does occur
  - Trigger ***Output Thread Processing*** for DEDB updates

Output Thread Processing

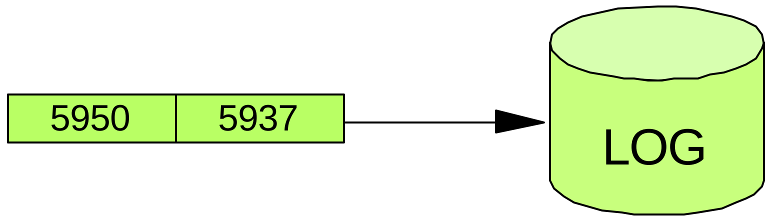
Output threads write DEDB updates to DASD

- Done after sync point processing
  - Triggered by physical logging of x'5937'
- Asynchronous to dependent region processing
  - D/R may be processing next transaction
- One output thread per Area
  - Parallel updates of ADSs
- Chained writes
  - Similar to OSAM queued write

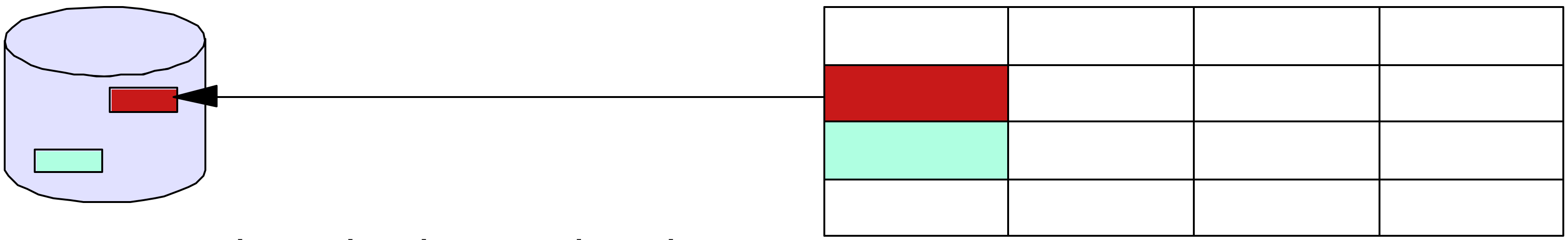
Output Thread Processing ...



Output Thread Processing ...



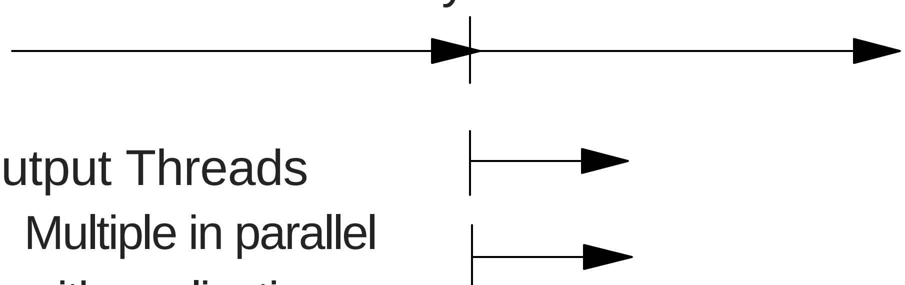
When physical logging does occur to OLDS or WADS



Output thread writes updated CI to Area Data Set

Sync

Application Processing



Output Threads

Multiple in parallel with application

DEDB vs HALDB (Highlights)

HALDB

- 1001 Partitions with 1-10 data sets @ 4GB
  - 1001 x 10 x 4 GB = 40 terabytes
  - 1001 x 4 GB = 4 terabytes (with single data set per partition)
- PHDAM and PHIDAM equivalents
  - Exactly the same API
- Primary and secondary indexes
- Logical relationships
- Look-aside buffering
- Sequential buffering and queued write
  - OSAM only

DEDB vs HALDB (Highlights) ...

DEDB

- 240 Areas @ 4 GB = 960 GB
- High Speed Sequential Processing
- High Speed Online Reorganization
- Virtual Storage Option
- Output thread processing
- Sequential dependents
- Shorter pathlengths

Summary

DEDBs are different

- Partitioned
  - Conceptually similar but implemented differently from HALDB
- Sequential dependent segment type
  - Excellent for fast data capture or for audit trail
- Space management
  - UOW concept enables HSSP and Online Reorganization
- System functions
  - Buffer management, locking, logging, output thread processing, ...

Summary ...

DEDBs provide many benefits

- Large database support
  - 960 GB
- High availability
  - Partitioning, online utilities
- Performance
  - Shorter pathlengths
  - Output thread processing
  - Sequential dependent processing
  - High Speed Sequential Processing (HSSP)
  - Virtual storage option (VSO)

Other DEDB Sessions

A37 - DEDB Advanced Topics

- SDEP processing
- High Speed Sequential Processing (HSSP)
- Virtual Storage Option (VSO)
- Data sharing

S60 - DEDB Performance Considerations

- DEDB characteristics that impact performance
- A look at some of the performance monitoring tools

RETURN TO INDEX