



# JES PROCESSORS

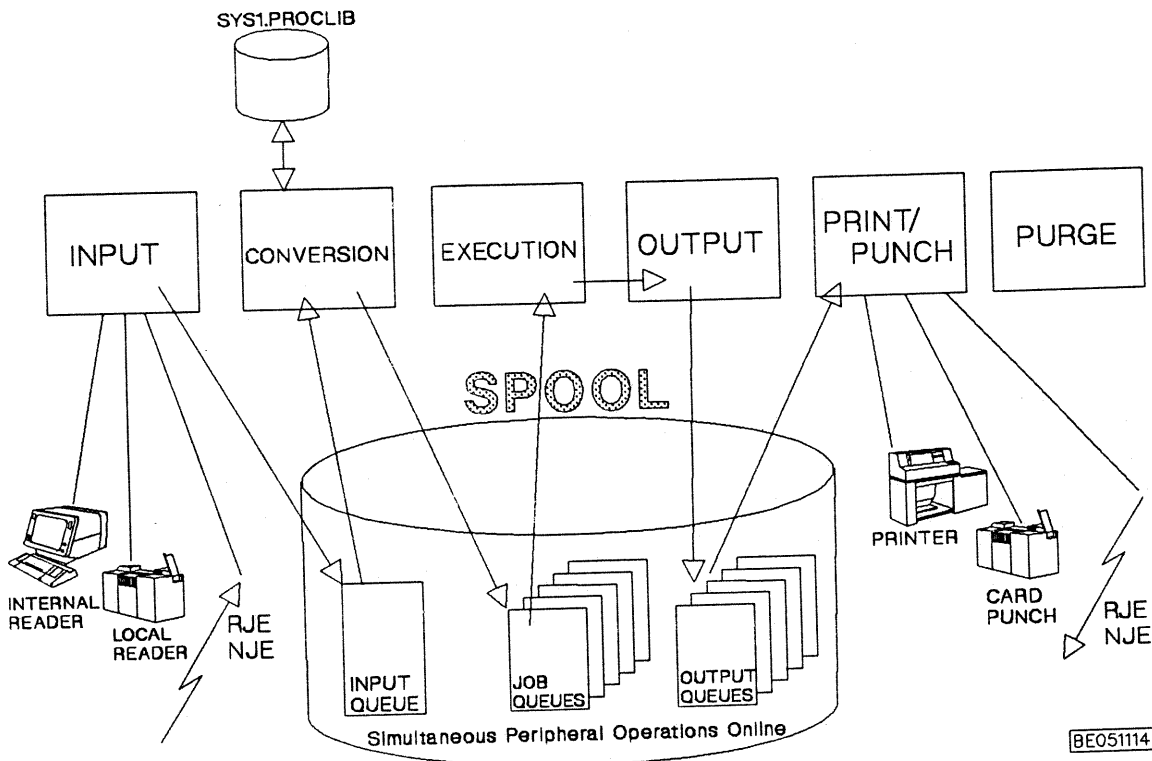


Figure 1-5. JES PROCESSING

## JES processing

- **INPUT**  
This is the process where a job will enter and it is given a number, THE JOB ID number which will be the ID of the job as long as it is in the system. The accounting information in the job card is tested and any JES commands are executed. The JCL statements are put on the spool dataset in the input queue.
- **CONVERSION**  
In this process the JCL information from the input queue is transformed to internal text and it is tested for syntax failures. If a procedure was called it will get it from one of the dataset in the JES PROCLIB concatenation and the JCL will be expand with this procedure. The result is put on the JOB QUEUE.
- **EXECUTION**  
The Internal text for the job are read into a Initiator an the job will execute.
- **OUTPUT**  
Here will the result of the job be formatted according to the job class or the statements in this job. The result is put on the decided output queue.
- **PRINT/PUNCH**  
Here you have the process that helps you to get hold of the output. It could be printed on a local printer or sent to a remote printer or even to another system.
- **PURGE**  
When all of the result of the job are printed it is time to clean up. The purge process will help us doing that and it will release the JOB ID that was received when entering the system.



# JOB SELECTION

## CHECK POINT

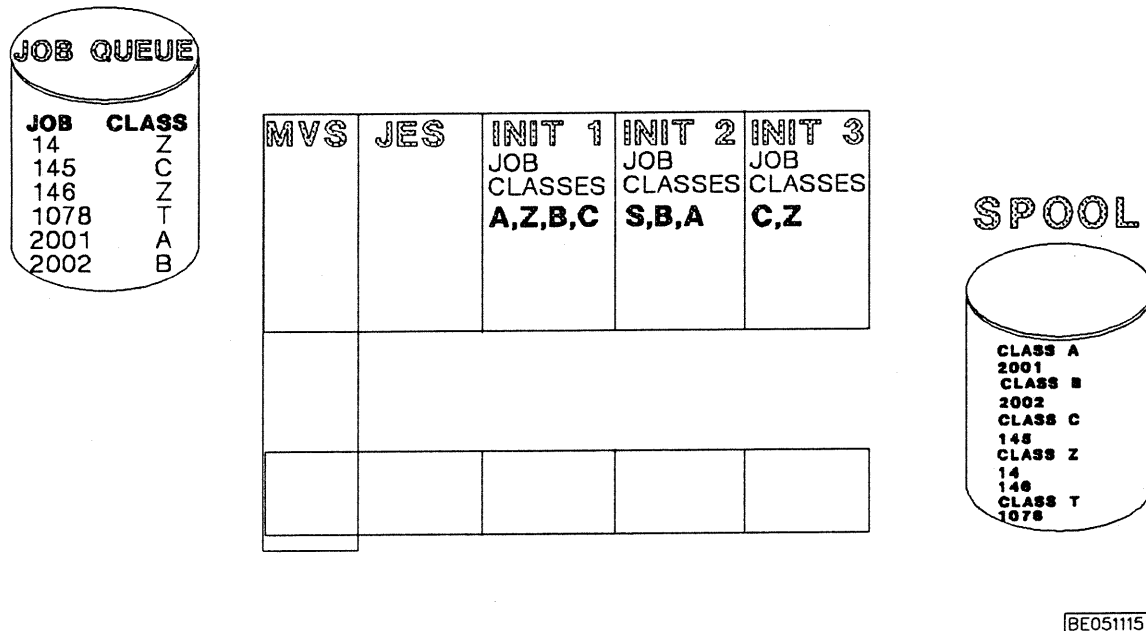


Figure 1-6. JOB SELECTION

### Job selection

- Batch Job's are running in INITIATOR Address Spaces and those can be started to serve different Job Classes.
- One Initiator could be started with several job classes and they're selected in priority order according to the order it had when the Initiator was started.
- When a Initiator is out of work it requests work from the JES A/S. JES then will look at the Job queue and select the highest priority job for this Initiator.

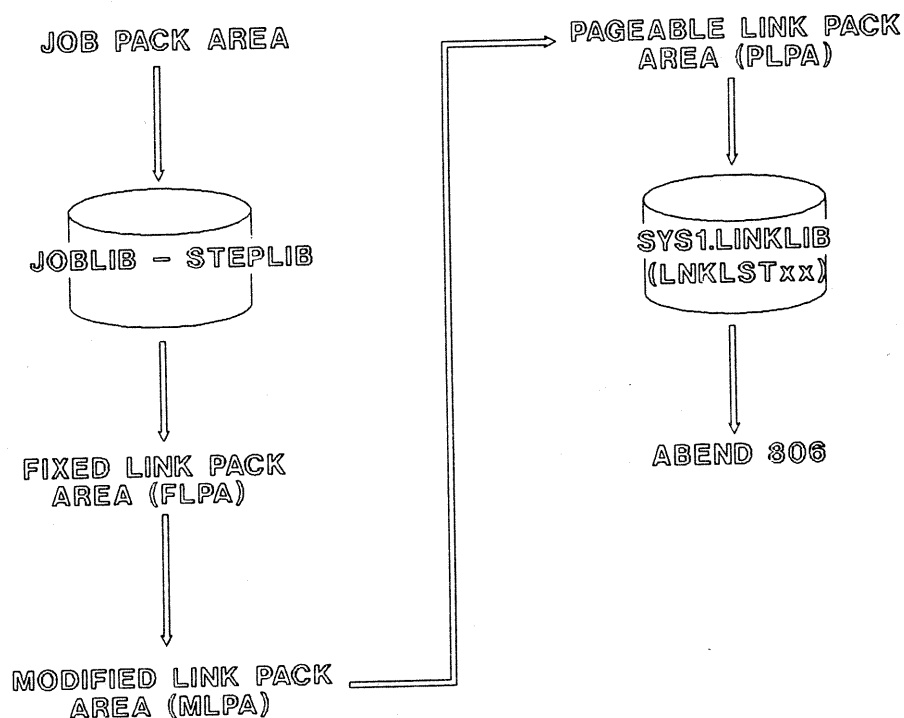
### Example

If looking at the figure Init 1 is ready to receive a new Job, it will first select one Job from the A class. The JOB 2001 will be found and started in that Init. When this Job is finished, the Initiator will request a new Job. If there are no more jobs in Class A, we have to look in the second highest Job Class which is Z and the JOB 14 will be selected for Init 1.

Both Init 2 and 3 are busy during this time.



## PROGRAM MANAGEMENT – SEARCH SEQUENCE



BE051116

Figure 1-7. PROGRAM SEARCH SEQUENCE

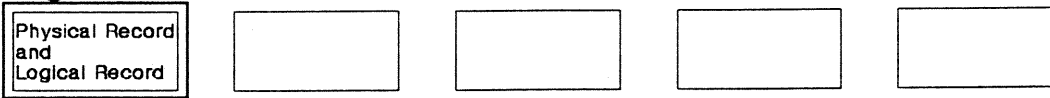
### Program search sequence

- When started the program selected in the EXEC Statement will be searched.
- First it starts looking within the Job itself in the JOB PACK AREA.
- Then it goes to see if there are some JOBLIB or STEPLIB JCL statements for this Job.
- If not it will look in the different LINK PACK AREAS.
- Finally it will look in the library's in the LINK LIST.
- When a program cannot be found, the job will abend with an 806 abend code.

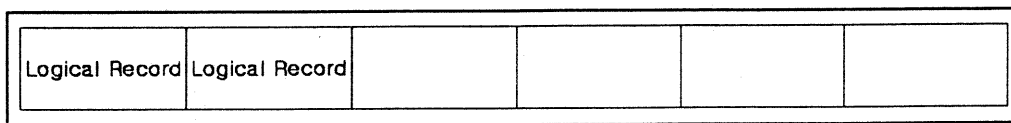
**Contains Restricted Materials of IBM - IBM Internal Use Only**

# LOGICAL AND PHYSICAL RECORDS

Physical Record  
and  
Logical Record



Physical Record



BE051211

Figure 2-3. Logical and Physical Records

## Logical and Physical Records

Datasets used by a program may have different characteristics in accordance to specific demands by the programmer. Not only may datasets differ from one another as a whole (different types of dataset) but also records within a dataset may differ between similar types of dataset.

### *Logical and Physical Records*

These are two distinct types of records:

#### A logical record

- has a size determined by what is natural and comfortable for the processing program to handle. Datasets used by text-processing programs (e.g. compilers) are given a logical record size equal to one line of text. A program processes one logical record at a time.

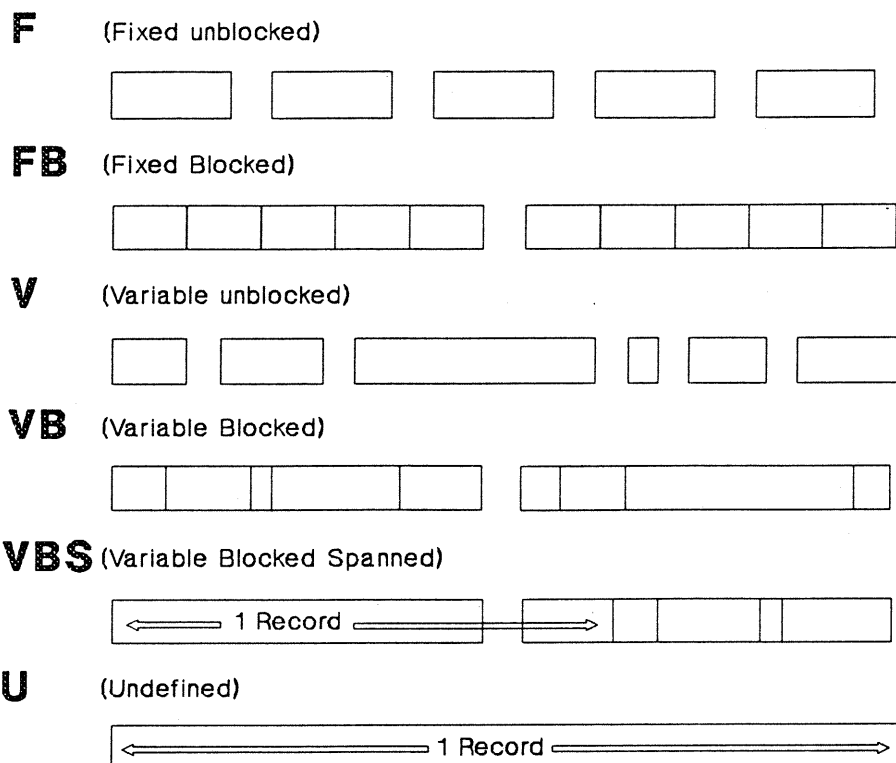
#### A physical record

- on the other hand, is given the size that yields the best performance for storage devices and I/O operations. A program accesses one physical record at a time.



**Contains Restricted Materials of IBM - IBM Internal Use Only**

## RECORD FORMAT AND BLOCKING



BE051212

Figure 2-4. Record Format and Blocking

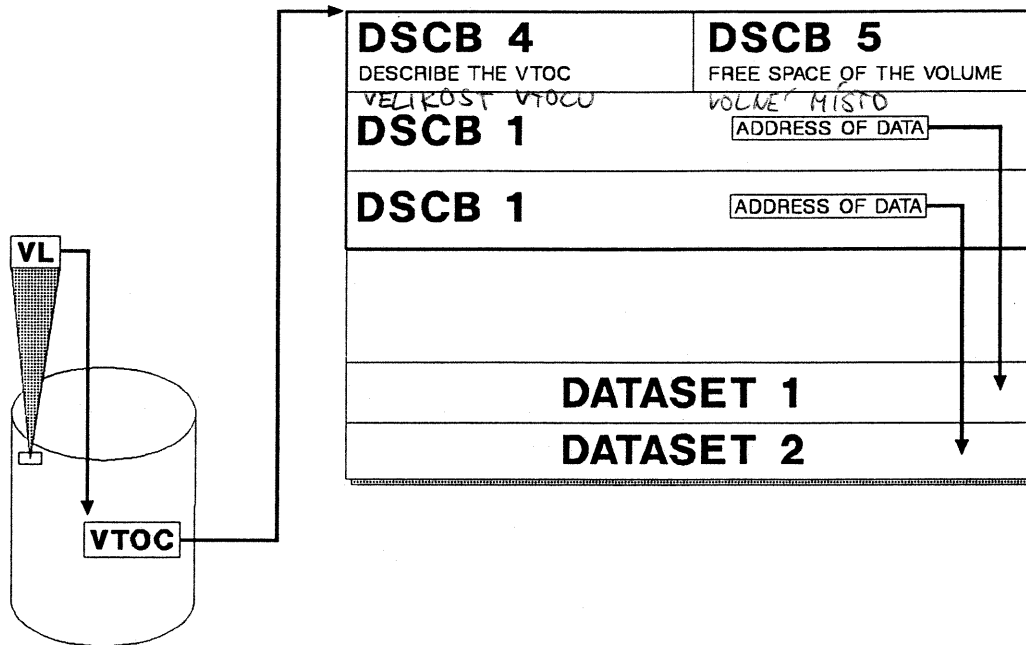
### Record Format and Blocking

Different types of datasets may, as stated above, have different logical record lengths. The format of the records may also differ. Datasets may have one of the following characteristics:

- **Fixed (F)** All records have the same length. MVS can check that a record read is of the same length as the record once written.
- **Variable (V)** The records are of variable length. Length is specified within the record. MVS can check that the record read is of the same length as the record once written.
- **Blocked (B)** Physical and logical records differ in length. MVS can check that the record read is of the same length as the record once written.
- **Spanned (S)** Logical records may span several physical records. MVS can check that the record read is of the same length as the record once written.
- **Undefined (U)** The records are of variable length. No length is specified within the record. MVS can **not** check that the record read is of the same length as the record once written.
- **ASA (A)** The 1. Byte of each record is an ISO/ANSI printer control character. **Machine (M)** The 1. Byte of each record is a machine control character.



## VOLUME TABLE OF CONTENT



BE051215

Figure 2-7. Volume Table Of Content

### Volume-Labels on DASD

Volume-label on disk will be found on track 0 record 3. It contains the name of the disk and information of where to find the Volume Table Of Content (VTOC).

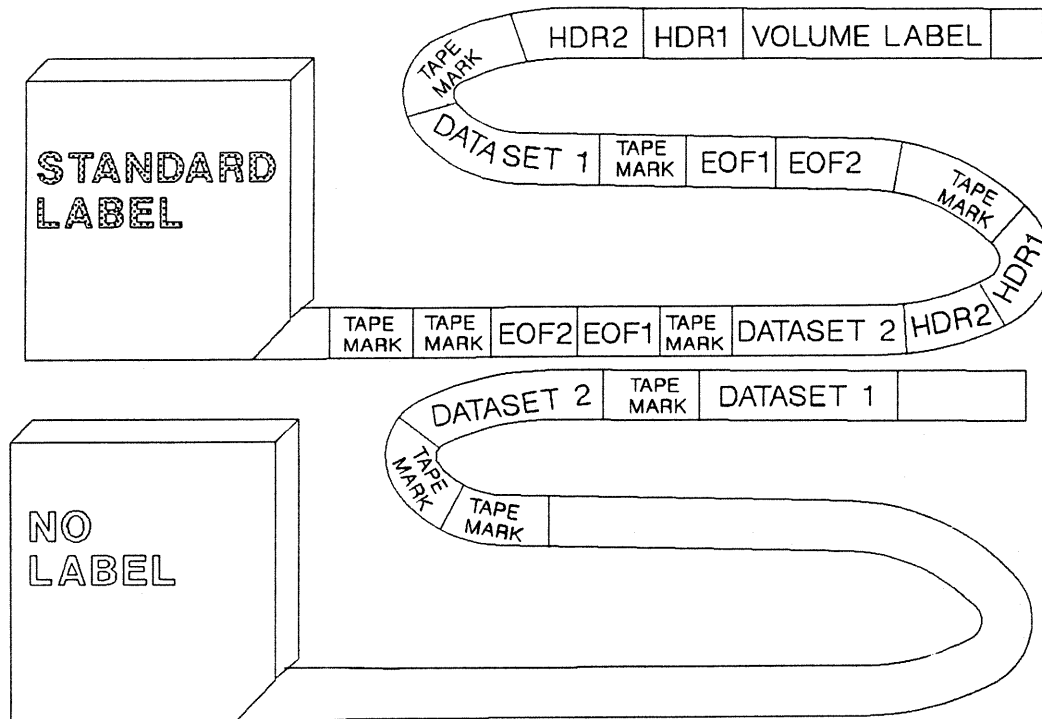
The VTOC contains several 140 bytes physical records called Data-Set Control Block (DSCB). There are several types of DSCB's containing different type of information about the disk and its data-sets.

Let's look at the different types and what they describes:

- Type 0     Free space in the VTOC.
- Type 1     Data-set information describing a data set.  
There must be one type 1 for every data set and it can contain up to 3 extents (non-contiguous space on the volume).
- Type 2     Describe data set using Index-sequential organization.
- Type 3     Holds up to 13 extents and are connected to a type 1 DSCB.
- Type 4     This DSCB is the first in the VTOC and describes the VTOC itself.
- Type 5     Describes free space on the volume.

**Contains Restricted Materials of IBM - IBM Internal Use Only**

## VOLUME LABEL ON TAPE



BE051216

Figure 2-8. Volume-Labels on Tape

### Volume-Labels on TAPE

A standard-label on tape has the same format as the volume label on DASD, but the pointer (address) to the VTOC is 0.

Instead of DSCB's there are Header- and Trailer-records on a tape with standard-labels.

#### HDR1 (HEADER 1)

- Contains of data set name and creation and expiration dates.

#### HDR2

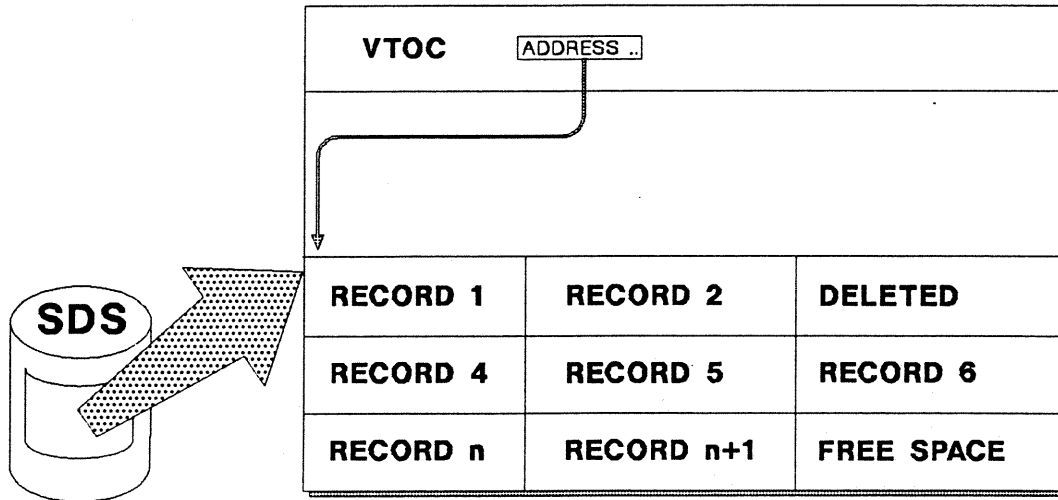
- Contains of DCB-parameters (Data Control Block) ex. logical record size and physical-blocksize.

#### EOF1 OCH EOF2 (End Of File)

- Has the same format and holds the same information as HDR1 and HDR2. EOF1 holds a count of how many blocks there are written in the data-set.



## SEQUENTIAL DATA SET



BE051217

Figure 2-9. Data Organization

### Data Organization

The organization of a dataset determines the ways in which separate records in it can be accessed. Since there must be different access methods for different organizations, the organization of the input and output data of a program is generally determined when the program is written.

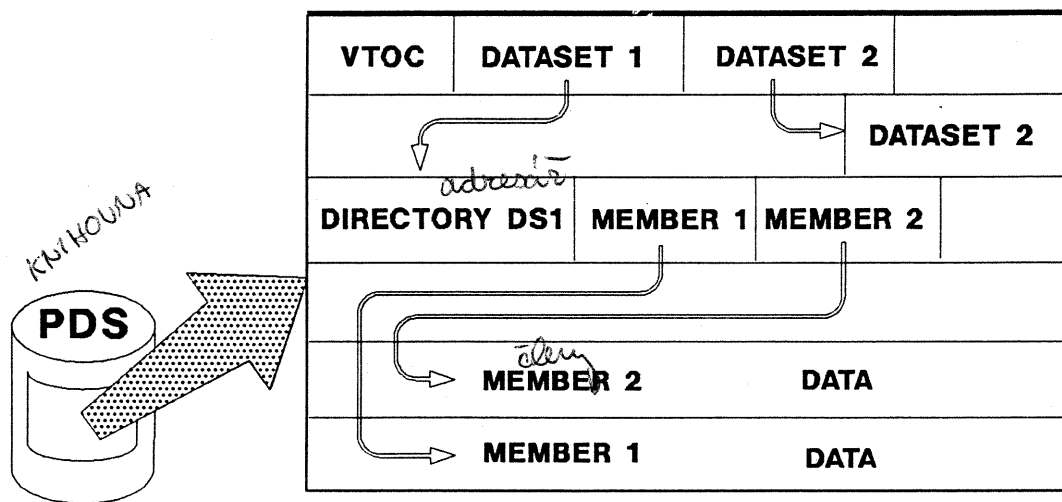
### *Sequential Organization*

The records of sequentially organized datasets are stored and retrieved sequentially, i.e. when reading them, programs start from the beginning and get access to them in the order they were written and when a new record is written, it is appended at the end of the dataset. Datasets residing on tape are sequential by physical necessity. Datasets on DASD may, but don't have to be sequential.





## PARTITIONED ORGANISATION



BE051218

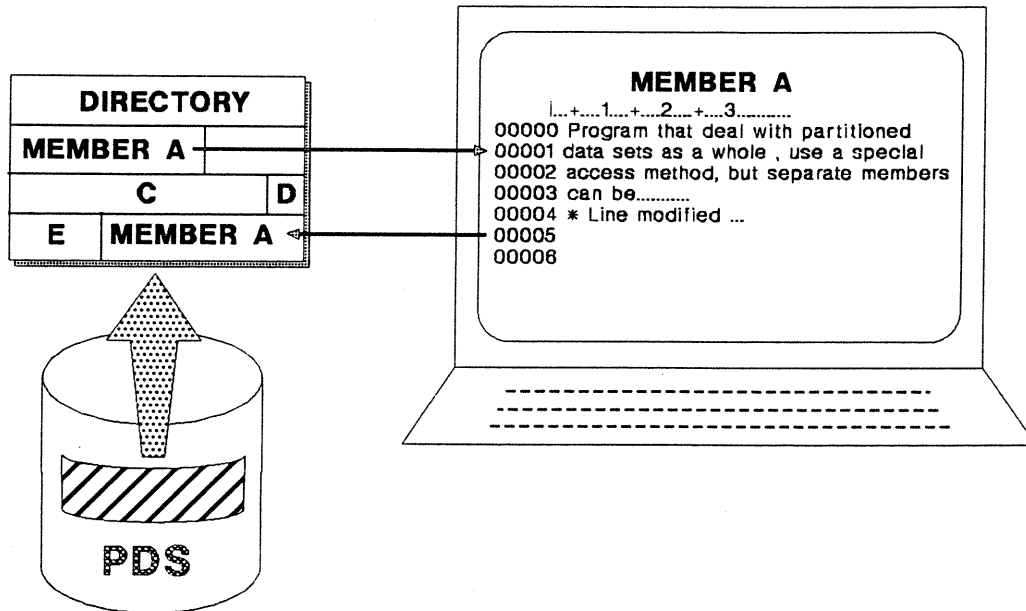
Figure 2-10. Partitioned Data Set (PDS)

### Partitioned Data Set (PDS)

- A partitioned data set is stored only on a direct access storage device. It is divided into sequentially organized members, each can have one or more records.
- Each member has a unique name stored in a directory that is part of the data set. The records of a given member are written or retrieved sequentially.
- A member can be added or deleted as required. When a member is deleted, the member name is removed from the directory, but the space used by the member cannot be reused until the data set is reorganized. (By compress using the IEBCOPY utility through ISPF)
- The directory, a series of 256 byte records at the beginning of the data set, contains an entry for each member. Each directory entry contains the member name and the starting location of the member within the data set.
- If there is not sufficient space available in the directory for an additional entry, or not enough space available within the data set for an additional member, or no room on the volume for additional extents, no new members can be stored. A directory cannot be extended and a partitioned data set cannot cross a volume boundary.



## UPDATING PDS MEMBER



BE05121A

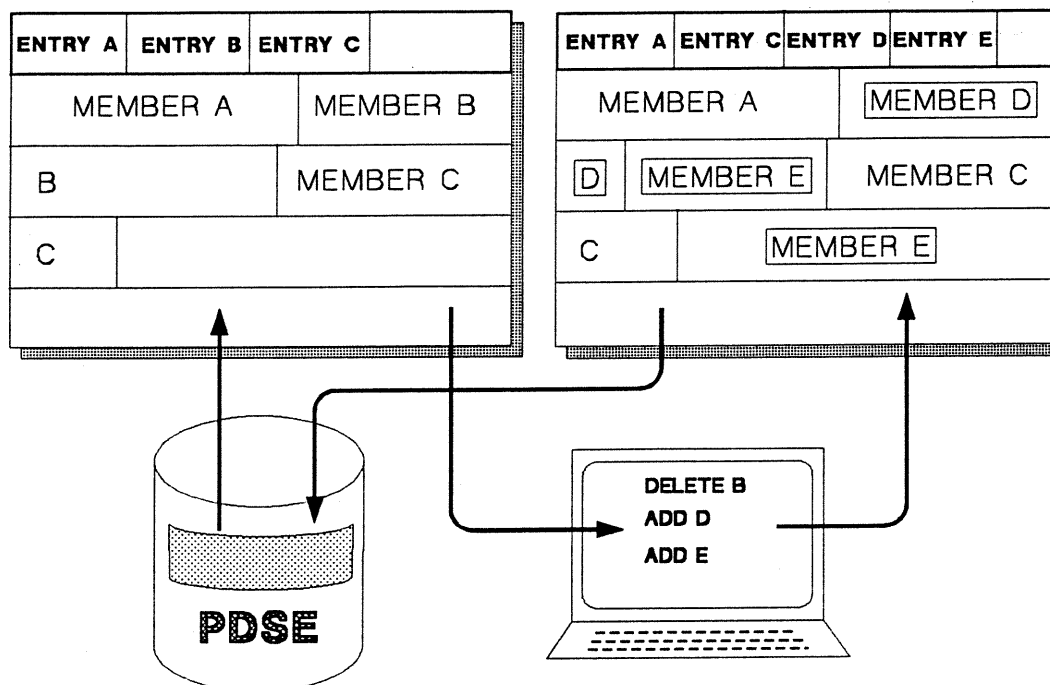
Figure 2-11. Updating a member of a (PDS)

### Updating a member of a PDS

If you want to extend or to add a record within a member, you must rewrite the complete member in another area of the data set. Because space is allocated when the data set is created, there is no need to request additional space.

**Contains Restricted Materials of IBM - IBM Internal Use Only**

## PARTITIONED DATA SET EXTENDED



BE05121B

Figure 2-12. Partitioned Data Set Extended (PDSE)

### Partitioned Data Set Extended (PDSE)

A partitioned data set extended is a system-managed data set divided into multiple members, each described by one or more directory entries.

In appearance, a PDSE is similar to a partitioned data set. For accessing a partitioned data set directory or member, most PDSEs interfaces are indistinguishable from partitioned data set interfaces, however, PDSEs have a different internal format which gives them increased usability.

You can use PDSE in place of a PDS to store data, but not load modules. PDSEs and PDSs are processed using the same access methods (BSAM, QSAM, BPAM) and macros.

### Advantage of using PDSE

PDSEs have several features that improve both, your productivity and system performance.

The main advantage of using a PDSE over a PDS is that PDSEs use DASD space much more efficiently.

- The size of partitioned data set directory is fixed regardless of number of members in it, while the size of the PDSE directory is flexible and expands to fit the members stored in it.
- The system reclaims the space automatically whenever a member is deleted or replaced, and returns it to the pool of space available for allocation to other members of the same PDSE.
- The space can be reused without having to do an compress.
- PDSE members can be shared. This makes it easier to maintain the integrity of the PDSE when modifying separate members at the same time.
- Reduced directory search time. The PDSE directory, which is indexed, is searched using that index.
- Creation of multiple members at the same time. For example, you can open two DCBs to the same PDSE and write two members at the same time.
- PDSEs contain up to 123 extents. An extent is a continuous area of space on a DASD storage volume occupied by or reserved for a specific data set.
- The PDSE directory is expandable, you can keep adding entries up to the directory's size limit or until the data set runs out of space.

**Note:** For a PDS, the size of the directory is determined when the data set is initially allocated and when the directory space is full, the PDS must be copied to a new data set with a larger directory or with more directory space before new members can be added.







## PDSE AND PDS DIFFERENCES

PDSE	PDS
Data set has 123 extent limit	Data set has 16 extent limit
Directory is open-ended and indexed by member name faster to search directory	Fixed size directory is searched sequentially
PDSEs are device independent: Records are reblockable and the TTR is simulated as a system key	For PDSs, TTR addressing and block sizes are device-dependent
Uses dynamic space allocation and reclaim	Must use IEBCOPY compress to reclaim space
You can create and update multiple members at the same time	You can create one member at a time

BE05121C

Figure 2-13. PDSE and PDS Differences

### PDSE and PDS Differences

Figure shows the significant differences between partitioned data sets extended and partitioned data sets.

**Note:** The significant similarities between PDSEs and PDSs are as follow:

- The same access methods and macros are used, with minor incompatibilities in some case.
- Records are stored in members and members are described in the directory.

**Contains Restricted Materials of IBM - IBM Internal Use Only**

## CREATING A MEMBER IN A PDS

```
//CREAPDS EXEC PGM=IEFBR14
//NEWPDS DD DSN=BE08.DEMO.PDS
//          DISP=(NEW,CATLG,DELETE),
//          SPACE=(TRK,(1,1,2)),
//          UNIT=SYSDA,
//          DCB=(LRECL=80,RECFM=FB,BLKSIZE=6320)
//*
//CREAMEM EXEC PGM=MYPROG
//OUTFILE DD DSN=BE08.DEMO.PDS(NEWMEM),
//          DISP=SHR
```

BE051353

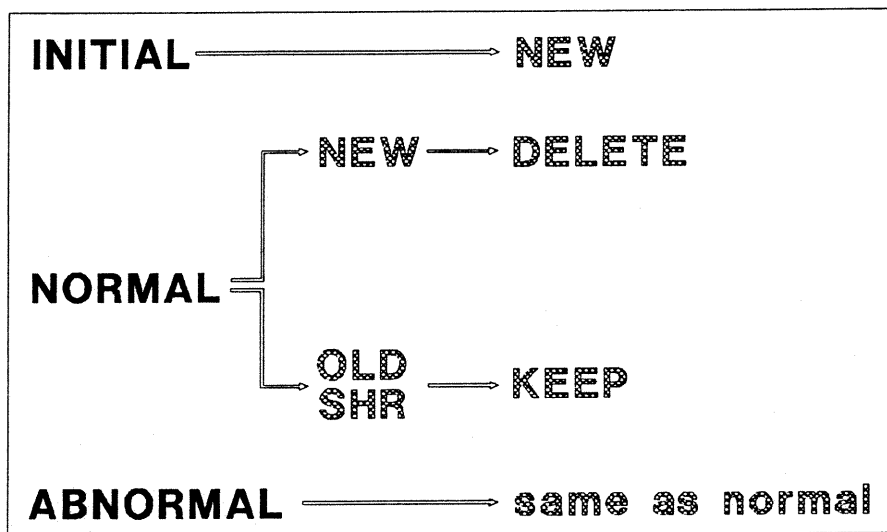
Figure 3-32. DD statement - DISP and the PDS

### DISP and the PDS

- The DISP parameter is applied on the complete data set.
- When members of a PDS are addressed, the PDS has to exist already.
- A PDS and a members of that PDS can not be created in the same job step.
- Only DISP=SHR or DISP=OLD is valid when we address a member of a PDS
- If a specified member name does not yet exist in the PDS, the member will be created.
- There is no way to delete a member from a PDS via the DISP parameter One has to use a utility.



## DISP DEFAULTS



BE051354

Figure 3-33. DD statement - DISP parameter defaults

### DISP parameter defaults

The system uses the following defaults if one of the subparameters is omitted:

- The default for INITIAL is NEW
- The default for NORMAL depends on INITIAL and is
  - DELETE when INITIAL is NEW
  - KEEP when INITIAL is SHR or OLD
- The default for ABNORMAL is the value of NORMAL unless PASS was specified, then default ABNORMAL is DELETE for new, KEEP for existing.



## KEYWORD PARAMETERS ON THE DD SYSOUT STATEMENT

```
//OUT2    DD  SYSOUT=A ,  
  
//                SPIN=UNALLOC | NO  
  
//                FREE=CLOSE | END
```

BE0513X1

Figure 3-42. SYSOUT STATEMENT - SPIN Parameter

### SPIN Parameter

SPIN parameter specifies that the output for the sysout data set is to be made available for printing.

#### UNALLOC

- Data set is available for printing immediately when data set is unallocated.
- If the system closes data set at end-of-step, the output will be available for printing.

#### NO

- No SYSOUT data set will be available for printing only at end-of-job.

#### Note:

If FREE = CLOSE is specified, the following default apply:

A data set that is closed by the application program is available

A data set that is closed as part of end-of-step is available for printing at the end-of-job.

If FREE = END is specified, the default is that the data set is available for printing at the end-of-job.





## SYSTEM UTILITY PROGRAMS

- IEHATLAS
- IEHINITT
- IEHLIST
- IEHMOVE
- IEHPROGM
- IFHSTATR

## DATA SET UTILITY PROGRAMS

- IEBCOMPR
- IEBCOPY
- IEBDG
- IEBEDIT
- IEBGENER
- IEBIMAGE
- IEBISAM
- IEBPTPCH
- IEBUPDTE

## INDEPENDENT UTILITY PROGRAMS

- ICAPRTBL

BE051413

Figure 4-3. IBM UTILITY PROGRAMS

### System UTILITY programs

- Used to Maintain and Manipulate System and User Data.
  - Maintain libraries and catalog entries
  - Initiate volumes
  - Volume and data set backup
  - List VTOC, Directories, and Catalog

### Data set utility programs

- Used to Reorganize , Change , or Compare data.
  - Data Set or Record Level
- Allow you to Manipulate PDS/SEQ Data Sets.
  - Fields within a logical record to entire Data Sets.
- Controlled by JCL and Utility Statements.
- The Programs are Executed as Jobs.

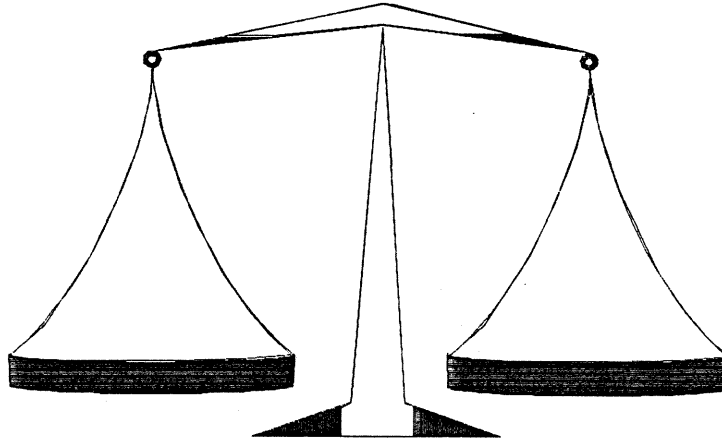
**Independent utility program**

- Used when the Operating System is **NOT AVAILABLE**.
  - Load the Universal Character Set buffer and the Forms Control Buffer for a printer.
- Controlled **ONLY** by Utility Statements.
- The Programs are **NOT INVOKED** by Calling a Program.





## IEBCOMPR



BE051411

Figure 4-14. IEBCOMPR

### IEBCOMPR

- Compare one PDS to another PDS
- Compare one sequential data set to another sequential data set

#### Job Control Statements

```
//STP001 EXEC PGM=IEBCOMPR  
//SYSUT1 DD DSN=INPDS1,DISP=SHR  
//SYSUT2 DD DSN=INPDS2,DISP=SHR  
//SYSPRINT DD ...  
//SYSIN DD ...
```

#### Control Statements for IEBCOMPR

COMPARE TYPORG={PS|PO}

IEBCOMPR sets a condition code of 8 if the files are not equal.



## JCL STATEMENTS FOR DFSORT

```
//jobname      JOB
//stepname     EXEC   PGM=SORT
//SYSOUT       DD     SYSOUT=*
//SORTIN       DD     ....
//SORTINnn     DD     ....
//SORTOUT      DD     ....
//SORTWKnn     DD     UNIT=SYSDA,
//              SPACE=(CYL,(1,1))
//SYSIN        DD     *
DFSORT control statements
/*
```

BE05141K

Figure 4-15. DFSORT program (SORT)

### DFSORT program (SORT)

#### Function

- SORT sequential data sets
- MERGE sequential data sets

#### Job Control Statements

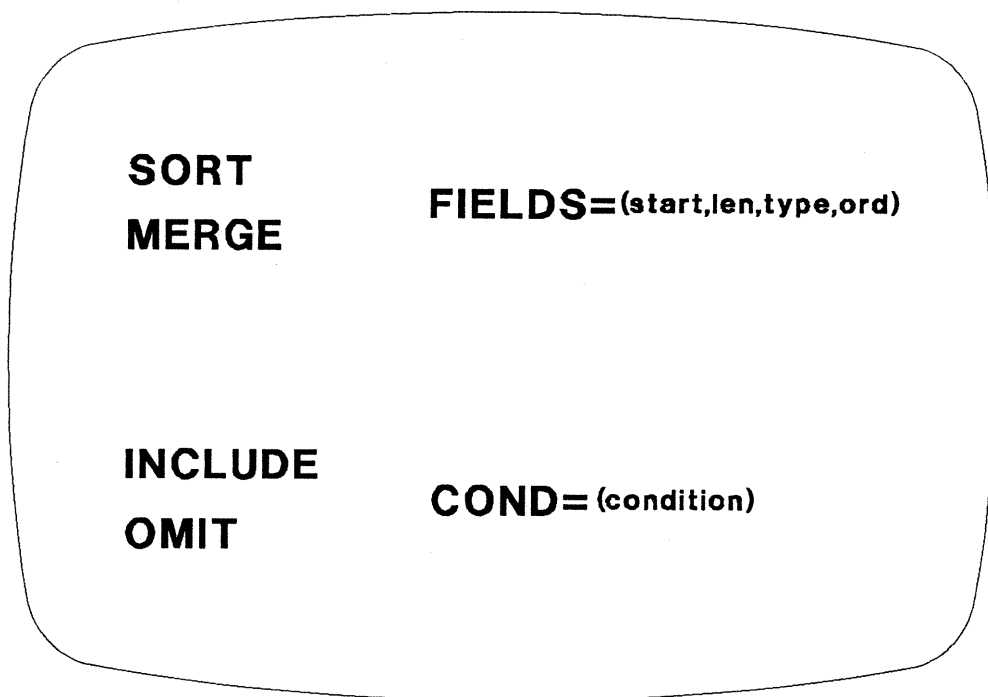
```
//stepname     EXEC PGM=SORT
//SORTWKnn     DD ...
//SORTIN       DD ...
//SORTINnn     DD ...
//SORTOUT      DD ...
//SYSIN        DD *
DFSORT control statements
/*
//SYSOUT       DD SYSOUT=*
```

The SORT program uses the ddname SYSIN for control statements. The ddname SYSOUT is used for messages from the program. The ddname SORTOUT always references the OUTPUT data set that SORT will be using. Large sorts may require sort work data sets. they use the ddnames SORTWKnn, where nn stands for 01,02,03, etc. Normally, only the UNIT and the SPACE parameter are supplied for these work data sets.



Contains Restricted Materials of IBM - IBM Internal Use Only

## DFSORT CONTROL STATEMENTS



BE05141L

Figure 4-16. DFSORT control statements

### DFSORT control statements

- **SORT** sequential data sets. **SORT** a sequential (or several concatenated) data set with another sequential data set. Input data set is referenced by ddname **SORTIN**.
- **MERGE** sequential data sets. **MERGE** several (concatenated) data set with another sequential data set. Input data sets are referenced by ddname **SORTINnn**, where **nn** stands for 01,02,03, etc. (max. 16). When **MERGING** data sets, no work data sets are required.
- **FIELDS=** Select the fields to be sorted or merged upon. Fields are specified via their starting position and length. The sort manner is specified via the order (Ascending or Descending) and the type of field (CHARacter of BINary).
- **OMIT** certain records that meet a specified condition.
- **INCLUDE** only those records that meet a specified condition.
- **COND=** Specifies the fields to be tested for the **INCLUDE** or **OMIT** operation.

It is also possible to summarize or reformat records.

**Examples:**

SORT FIELDS=(70,10,CH,A,30,5,BI,D)

MERGE FIELDS=(110,5,CH,D)

INCLUDE COND=(166,5,CH,GT,C'DEMO')

SORT FIELDS=(110,5,CH,D)

OMIT COND=(30,5,BI,EQ,60,5,BI,OR,40,3,CH,NE,45,3,CH)

MERGE FIELDS=(20,10,CH,D)

---

## EXERCISE 7

Copy the member JOB1 into the Member JOB2. Add the two sortsteps referred to the following flow charts.

- The sort order for the first SORT is:
  1. Position 44, 7 Bytes long
  2. Position 6, 8 Bytes long
  3. Position 14, 2 Bytes long
  4. Position 16, 4 Bytes long
  5. The data are character and ascending
- The sort order for the second SORT is:
  1. Position 6, 4 Bytes long
  2. Position 34, 4 Bytes long
  3. Position 10, 7 Bytes long
  4. The data are character and ascending



## JCL STATEMENTS FOR DFDSS

```
//jobname      JOB
//stepname     EXEC PGM=ADRDSSU
//SYSPRINT     DD      SYSOUT=*
//SYSIN        DD      *
```

### DFDSS control statements

```
/*
//input        DD      ....
//output       DD      ....
//filter       DD      ....
```

BE05141M

Figure 4-18. DFDSS program

### DFDSS program

#### Function.

Data Facility Data Set Services (DFDSS) is a direct access storage device (DASD) data space management tool. It often works on a volume basis. It is not a real end-user tool, but some purposes it can come in handy. It is used to:

- Copy and move data sets between volumes.
- Dump and restore data sets.
- Compress PDS's.
- Release unused space in data sets.

DFDSS is applicable to:

- SAM data sets.
- PDS's.
- VSAM data sets.
- etc.

DFDSS can be invoked via JCL or via ISMF.



# DFDSS CONTROL STATEMENTS

## COMPRESS

## COPY

## DUMP

## RELEASE

## RESTORE

BE05141N

Figure 4-19. DFDSS examples

### DFDSS examples

- DFDSS uses a simple command language to process data sets.
- DFDSS can process one data set at the time or a whole group of data sets using advanced filtering. Filter characters are \*\*, \* and %.

### Examples:

Compress a PDS on a specified volume:

```
//COMPRESS          EXEC PGM=ADDRSSU
//SYSPRINT          DD SYSOUT=*
//VOLUME            DD VOL=SER=0C0031,UNIT=SYSDA,DISP=SHR
//SYSIN             DD *
COMPRESS INCLUDE(BE05I.DEMO.PDS) DDNAME(VOLUME)
/*
```

Move a series of data sets to another volume:



Contains Restricted Materials of IBM - IBM Internal Use Only

```
//MOVE                                EXEC PGM=ADRDSSU
//SYSPRINT                            DD SYSOUT=*
//TARGET                             DD VOL=SER=0C0031,UNIT=SYSDA,DISP=SHR
//SYSIN                               DD *
COPY DATASET(INCLUDE(BE05I.**)) -
    OUTDDNAME(TARGET) DELETE CATALOG
/*
```

Copy a data set to tape:

```
//COPY                                EXEC PGM=ADRDSSU
//SYSPRINT                            DD SYSOUT=*
//TAPE                               DD VOL=SER=MYTAPE,UNIT=TAPE,
// DISP=(NEW,KEEP),DSN=PDS.COPY
//SYSIN                               DD *
COPY DATASET(INCLUDE(BE05I.DEMO.PDS)) -
    OUTDDNAME(TAPE)
/*
```

Free unused space for a series of data sets on a volume

```
//RELEASE                             EXEC PGM=ADRDSSU
//VOLUME                             DD VOL=SER=0C0031,
//                                  UNIT=SYSDA,DISP=OLD
//SYSPRINT                            DD SYSOUT=*
//SYSIN                               DD *
RELEASE INCLUDE(BE05I.*.PDS%) DDNAME(VOLUME)
/*
```



---

## TOPIC 5: ADVANCED JCL STATEMENTS

### Topic objectives

The purpose of this unit is to introduce the student to the advanced use of JCL statements. It will provide basic information about the use of some of these statement and how they are executed.

### References

- *Manuals:*
  - GC28-1653 JCL User's Guide
  - GC28-1654 JCL Reference
  - GC28-1656 MVS/ESA System Messages Vol.1
  - GC28-1657 MVS/ESA System Messages Vol.2
  - GC28-1658 MVS/ESA System Messages Vol.3
  - GC28-1664 MVS/ESA System Codes
  - SC26-4559 Utilities

## ADVANCED JCL STATEMENTS

### Introduction

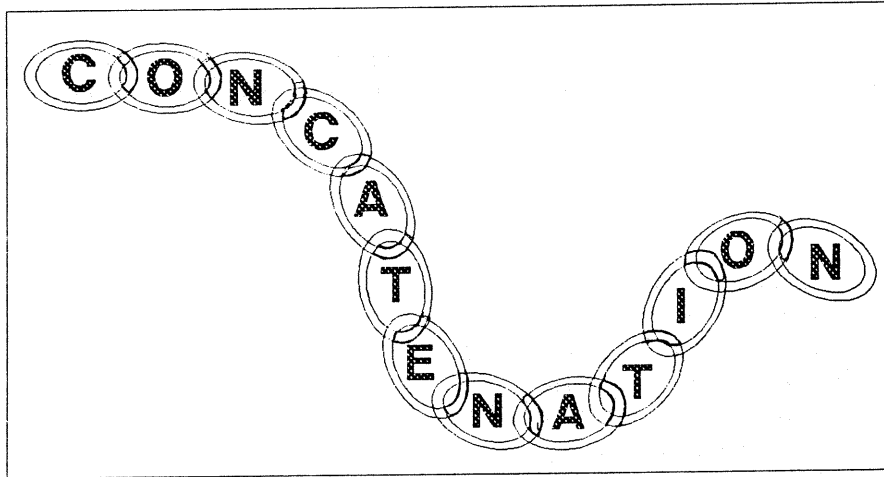
The purpose of this unit is to introduce some advanced JCL topics

### Subtopic objectives

Upon completion of this unit the student should be able to:

- Define concatenation and code concatenated data sets in JCL.
- Define referbacks and use them.
- Discuss restarting JCL and its implications.
- Code conditional JCL.
- Know how to look up the necessary JES2/JES3 statements needed to run a job.

## DEFINITION OF CONCATENATION



BE051511

Figure 5-1. Definition Of Concatenation

### Definition Of Concatenation

- Grouping a set of separate data sets into a single logical data set.

**Contains Restricted Materials of IBM - IBM Internal Use Only**

# RULES OF CONCATENATION



BE051512

Figure 5-2. Rules Of Concatenation

## Rules Of Concatenation

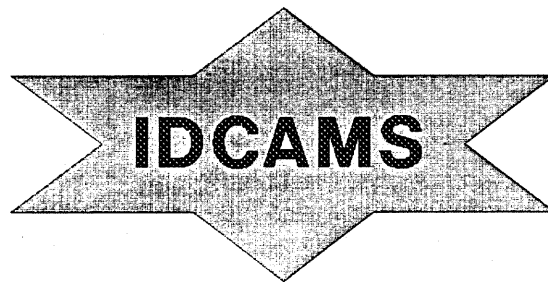
- Maximum of 255 sequential data sets
- Maximum of 16 partitioned data sets
- Partitioned and sequential data sets cannot be mixed
- Members of a partitioned data sets are treated as sequential data sets.
- RECFM must be the same on each data set being concatenated.
- BLKSIZE need not be the same for each data set, but the data set with the largest block size should be specified first.
- LRECL need not be the same for each data set, but the data set with the largest logical record length should be specified first.



---

# THE VSAM UTILITY

## ACCESS METHOD SERVICES



BE05161J

---

Figure 6-19. The VSAM Utility, IDCAMS

### The VSAM Utility, IDCAMS

- AMS - Access Method Services is a group of utility functions to establish and maintain catalogs and data sets for VSAM
- IDCAMS - is the name of the module that invokes AMS when executed
- Utility Functions Supported ...
  - Define
  - Copy
  - List
  - Print
  - Verify
  - Delete





## Invoking IDCAMS

```
//stepname      EXEC PGM=IDCAMS
//SYSPRINT      DD SYSOUT=A
//ddname        DD DSN=clustername
//SYSIN         DD *
                AMS statements
/*
```

BE05161K

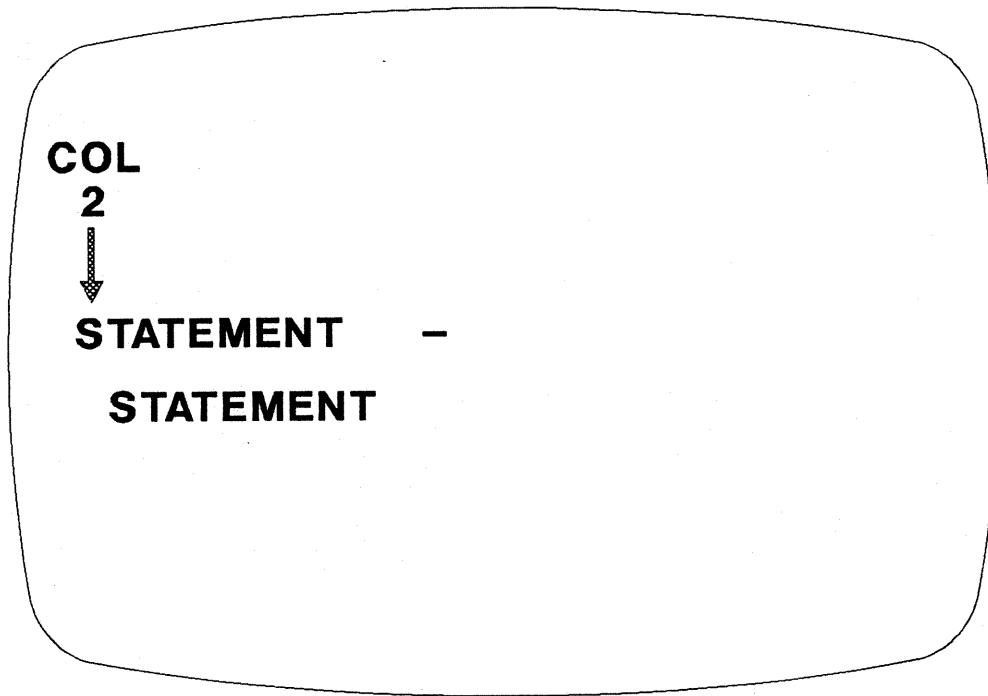
Figure 6-20. Invoking IDCAMS

### Invoking IDCAMS

- Invoke with JCL statements and AMS control statements



## CONTROL STATEMENTS



BE05161L

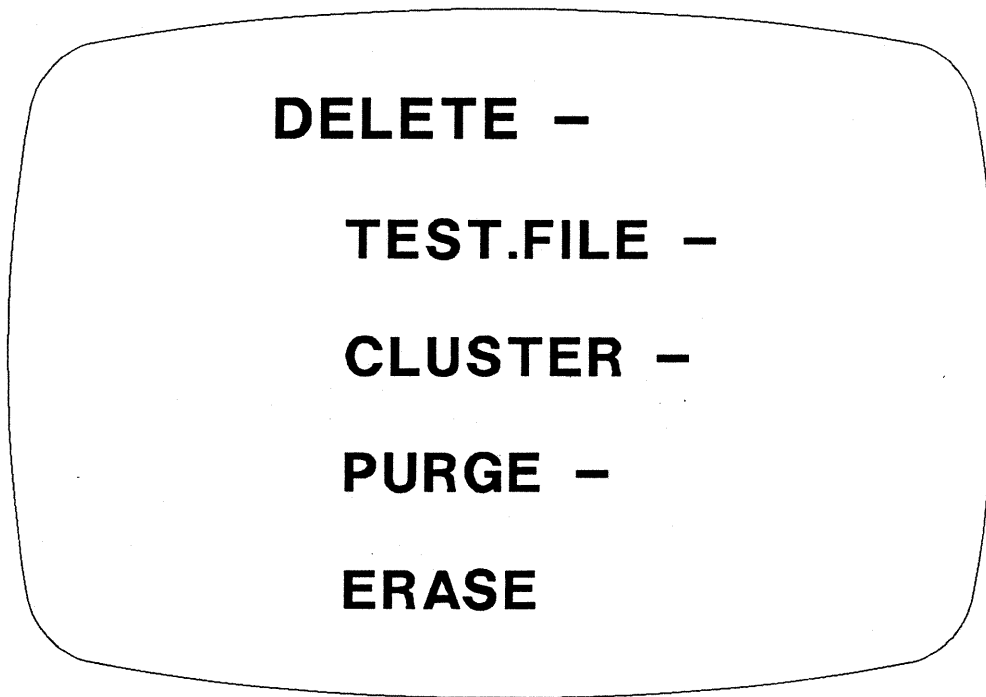
Figure 6-21. Control Statements

### Control Statements

- AMS Control Statements
  - Column 1 must be blank
  - Dash (-) used for continuation
  - Multiple AMS statements allowed within one 'EXEC PGM = IDCAMS'



## DELETING DATA SETS



BE05161R

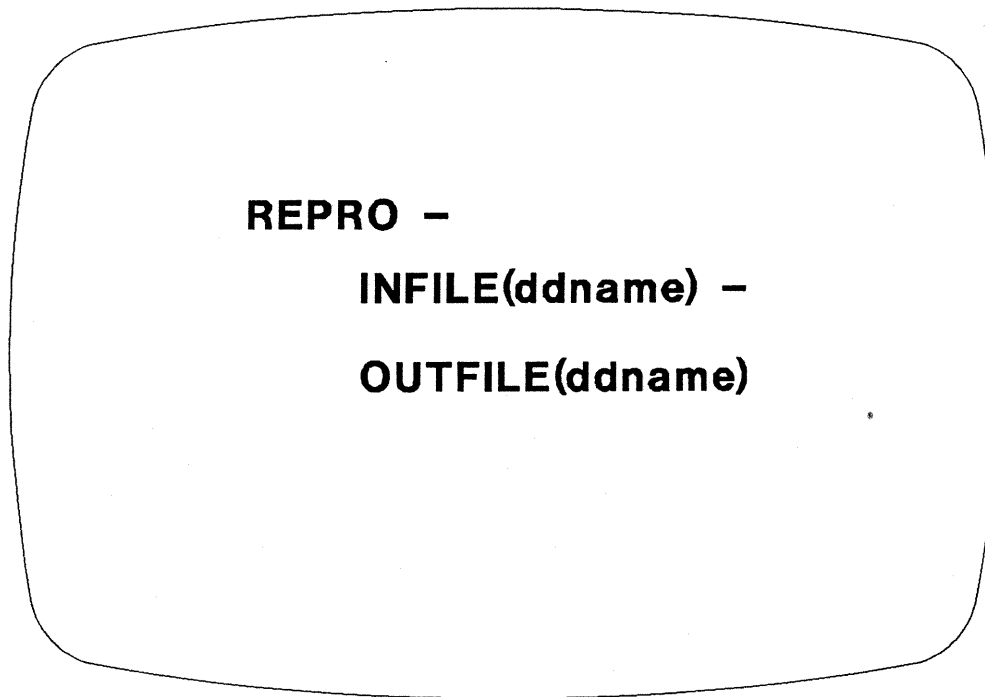
Figure 6-27. Deleting Data Sets

### Deleting Data Sets

- **TEST.FILE** is a suballocated data set residing on volume EDPAXx
- **CLUSTER** indicates that the object to be deleted is a VSAM cluster
- **PURGE** allows the data set to be deleted even though its retention period has not expired.
- **ERASE** indicates that the data component of the data set is to be erased during the deletion operation by overwriting it with binary zeros.



## REPRO STATEMENT



BE05161S

Figure 6-28. Repro Statement

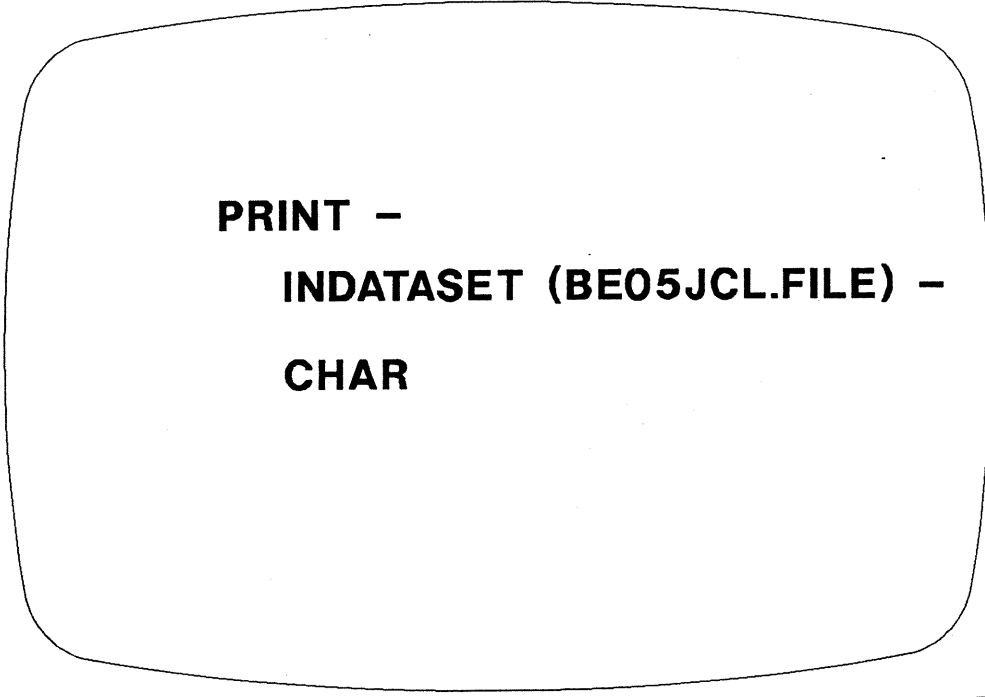
### Repro Statement

- Copies VSAM and non-VSAM data set
- **INFILE & OUTFILE** identifies ddnames for the input and output data sets
- **INDATASET** and **OUTDATASET** can optionally be specified instead of **INFILE** and **OUTFILE** to specify the input and output data sets
- **FROMKEY** and **TOKEY** identifies beginning and ending keys or generic keys (KSDS only)
- **FROMADDRESS** and **TOADDRESS** identifies beginning and ending relative byte address or RBA (KSDS or ESDS)
- **FROMNUMBER** and **TONUMBER** identifies beginning and ending relative record number (RRDS only)
- **SKIP** tells number of records to skip before copying
- **COUNT** tells number of records to copying





## PRINTING VSAM DATA SETS



**PRINT -  
INDATASET (BE05JCL.FILE) -  
CHAR**

BE05161T

Figure 6-29. Printing VSAM Data Sets

### Printing VSAM Data Sets

- **INFILE** or **INDATASET** must be specified as in **REPRO**
- **CHAR** specifies that the data set is to be printed record by record in readable format. If **CHAR** is omitted, the data set is printed in dump format.

Contains Restricted Materials of IBM - IBM Internal Use Only

---

# LISTCAT STATEMENT

**LISTCAT -**

**ENTRIES(BE05JCL.FILE)**

---

Figure 6-30. Listcat Statement

## Listcat Statement

- The listing can be tailored to meet your needs by ...
  - Limiting the number of entries to be printed
    - **Level** - all entries with high level qualifier
    - **Entries** - all entries for the one file name
  - Limiting the amount of information to be printed for each entry
    - The **ALL** parameter indicates that all catalog information is to be printed.
    - Other types of parameters ...
      - Name
      - Allocation
      - Volume
- Examples ...
  - LISTCAT ENTRIES(BE05JCL.FILE) ALL
  - LISTCAT LEVEL(BE05JCL)
  - LISTCAT ENTRIES(BE05JCL.FILE) VOLUME



# SET STATEMENT

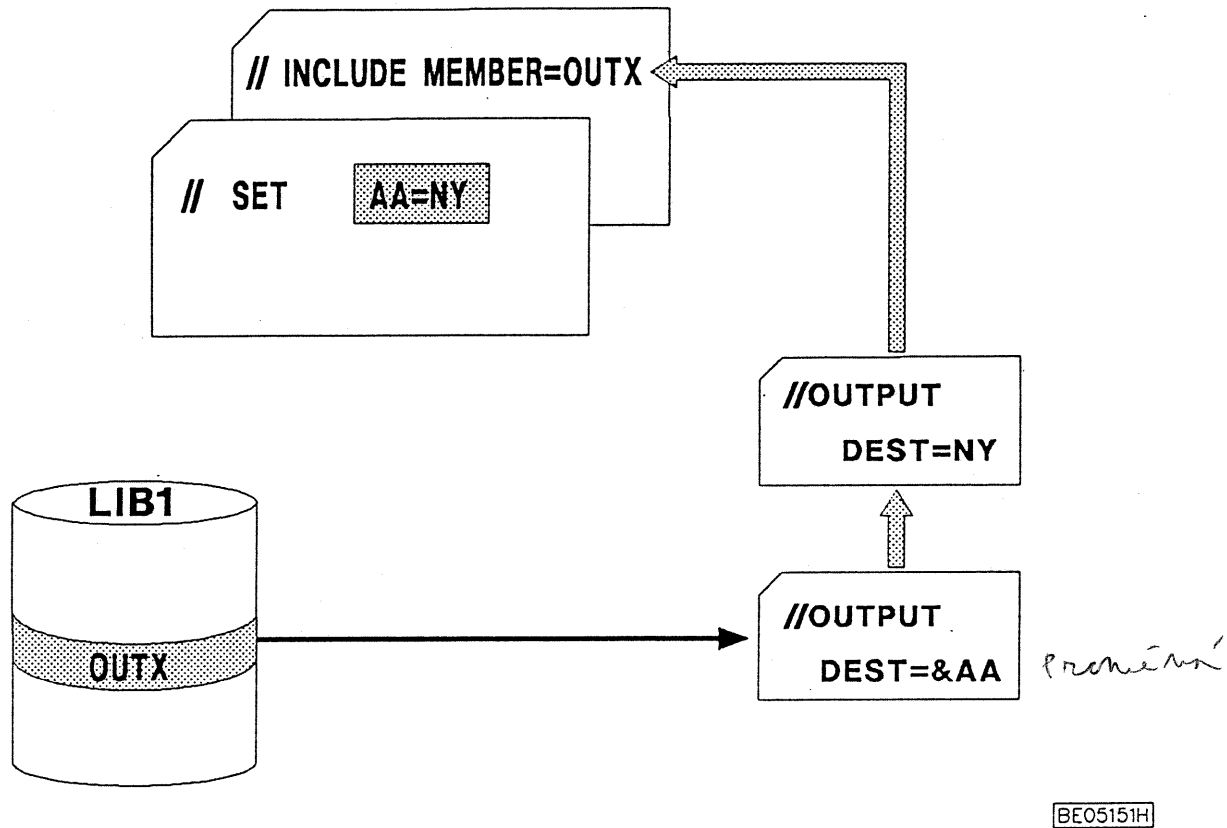


Figure 8-12. The SET statement

## The SET statement

- Specifies the initial values to symbolic parameters that are to be used when processing JCL statements.
- Changes or nullifies the values of defined symbolic parameters (those that are defined on previous SET statements) by assigning new values or nullifying current values.
- consists of // in columns 1 and 2 and may have four fields:

- Name (optional)
- Operation (SET)
- Parameter, the SET statement contains one or more parameters:
  - symbolic-parameter = value, symbolic-parameter = value

Defines a symbolic parameter and specifies the initial value to be assigned to the symbolic parameter appearing in subsequent JCL statements, separate each assignment of a value to a symbolic parameter by commas.

- To nullify a symbolic parameter, specify symbolic-parameter = ,
- Comments (optional)

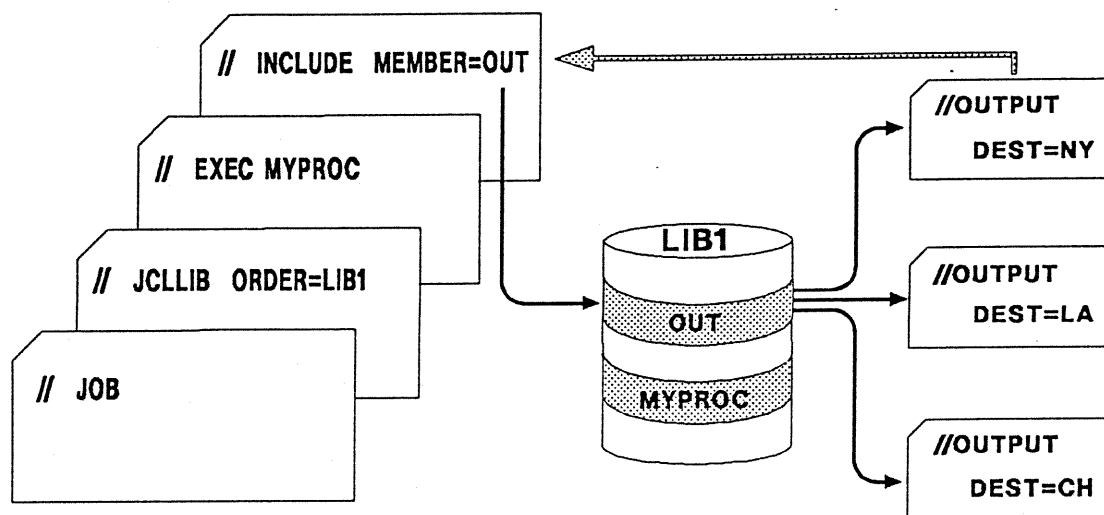
- A SET statement can appear anywhere in the JOB after the JOB statement, but must come before the intended use of the symbolic parameter.
- The SET statement is not executed conditionally. For example , if the SET statement appears in an IF/THEN/ELSE/ENDIF statement construct, the value is assigned to the symbolic parameter regardless of the logic of the construct.







# INCLUDE STATEMENT



BE05151F

Figure 8-13. The INCLUDE Statement

## INCLUDE Statement

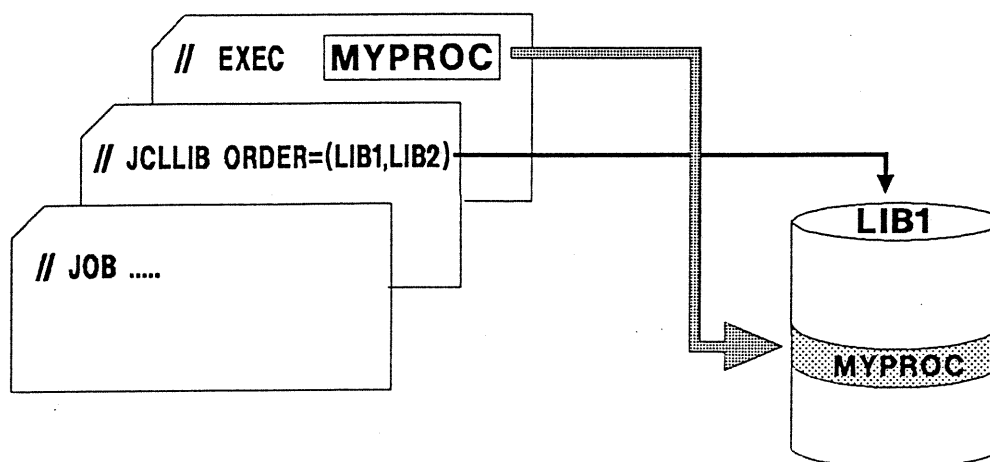
- Specifies the name of a member of a partitioned data set (PDS) that contains a set of JCL statements (such as DD and OUTPUT JCL statements), called an INCLUDE group.
- Can appear anywhere in the job after the JOB statement.
- Can appear within an INCLUDE group. INCLUDE groups can contain INCLUDE statements and can be nested up to a maximum of 15 levels.
- Consists of // in columns 1 and 2 and may have four fields:
  - Name (optional)
  - Operation (INCLUDE)
  - Parameter (MEMBER = name). Where name is either:
    - A member of a system procedure library (such as SYS1.PROCLIB), or
    - A member of an installation-defined procedure library, or
    - A member of a private library that you must specify on a JCLLIB statement appearing earlier in the job.
  - Comments (optional)

RECEIVED 11/11/11

NOV 11 11 11

11/11/11

# JCLLIB STATEMENT



BE05151G

Figure 8-14. The JCLLIB statement

## JCLLIB statement

- Specifies the names of the private libraries the system is to search for:
  - Procedures named on any EXEC statements.
  - Groups of JCL statements (called INCLUDE groups) named on any INCLUDE statements.
  - Specifies the concatenated order of search of the defined libraries PRIOR to searching any system default libraries.
- consists of // in columns 1 and 2 and may have four fields:
  - Name (optional)
  - Operation (JCLLIB)
  - Parameter (ORDER = (library,library...))
  - Comments (optional)
- The JCLLIB statement if used must appear after the JOB statement and before the first EXEC statement in the job.
- Must not appear within an INCLUDE group.
- All libraries defined must have the same data set attributes, which are:
  - logical record length of 80 bytes (LRECL=80)
  - Fixed length records (RECFM=F, or RECFM=FB)

## **STORAGE MANAGEMENT SUBSYSTEM**

### **Subtopic objectives**

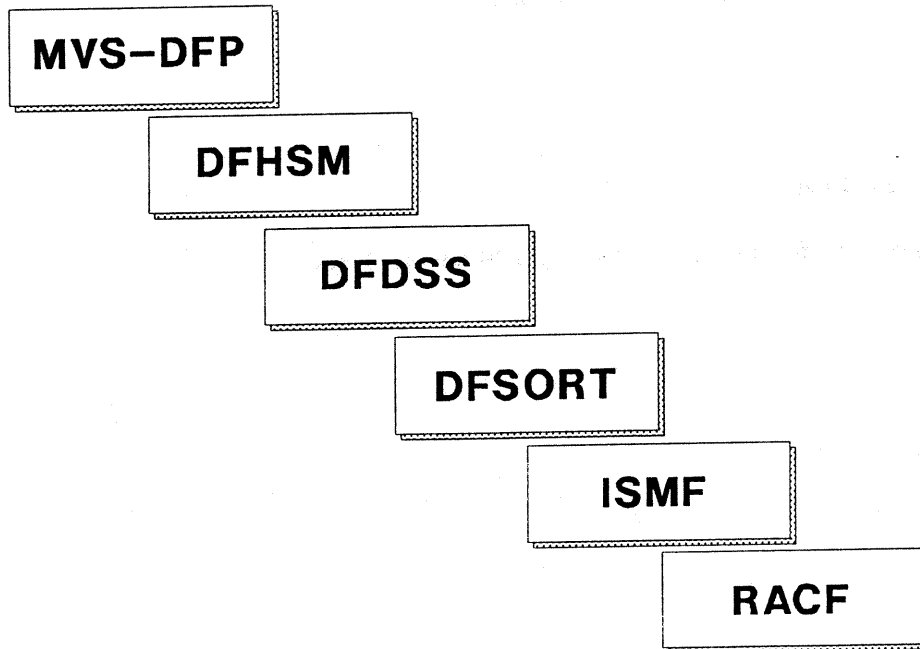
*The purpose of this unit is to introduce SMS concepts.*

### **Subtopic timing**

This topic contains **24** visuals.

The lecture time should be **50 minutes** followed by a **10 minutes** break.

## DATA FACILITY FAMILY



BE05191X

Figure 9-1. What is DFSMS

### What is DFSMS?

The Data Facility Family products (MVS/DFP, DFHSM, DFDSS, DFSORT) work together to perform storage management tasks. In addition, RACF provides security and authorisation control.

The Interactive Storage Management Facility (ISMF) for MVS/DFP provides an interactive interface to the Storage Management Subsystem, and to DFHSM and DFDSS.



# STORAGE MANAGEMENT SUBSYSTEM

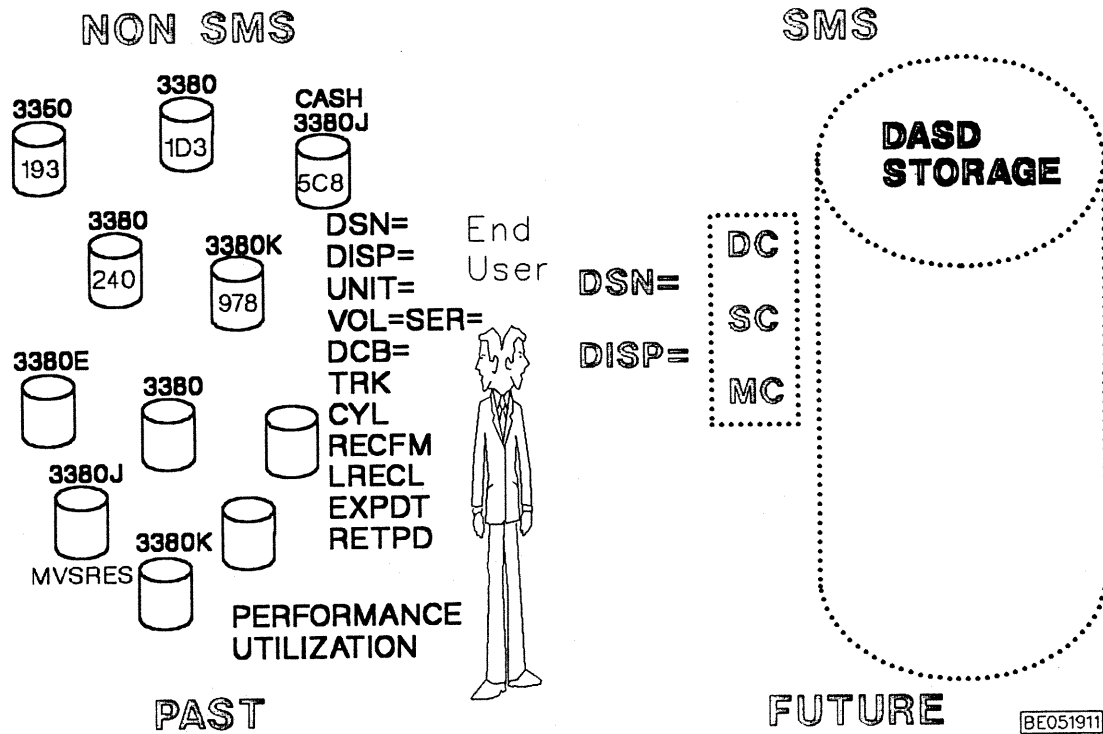


Figure 9-2. Why Management Storage Resources

## Why Management Storage Resources

The general meaning of system-managed storage is to have a system that performs the management of the major storage tasks.

The key to storage being system-managed is that the functions are performed automatically, allowing users to concentrate on logical data management.

It also allows the separation of the physical view from the logical view for the application user.





## MAJOR FUNCTIONS OF SMS

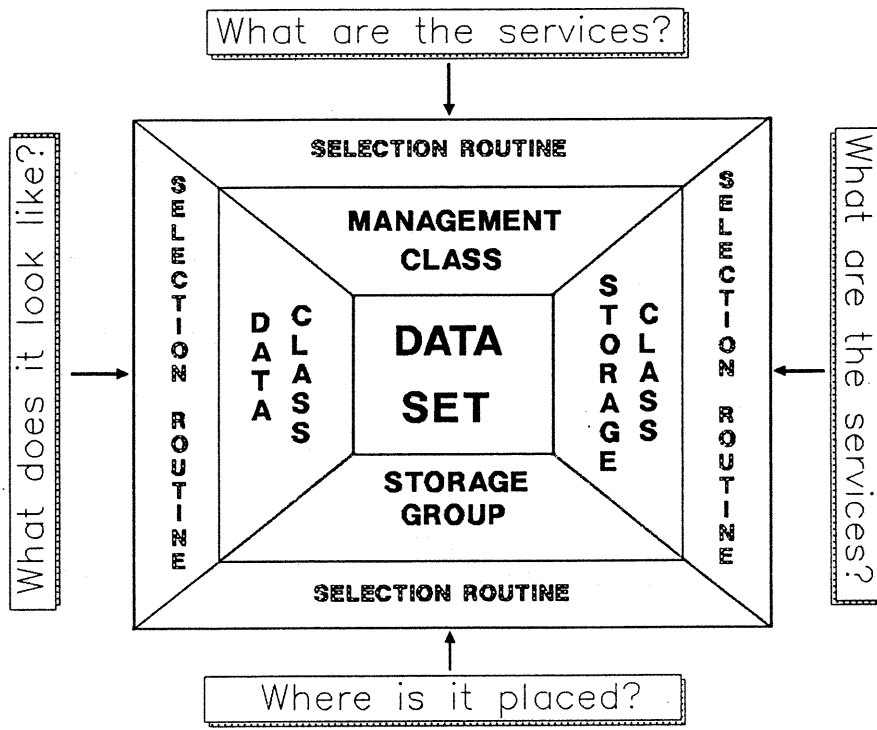


Figure 9-3. Major functions of SMS

### Major functions of SMS

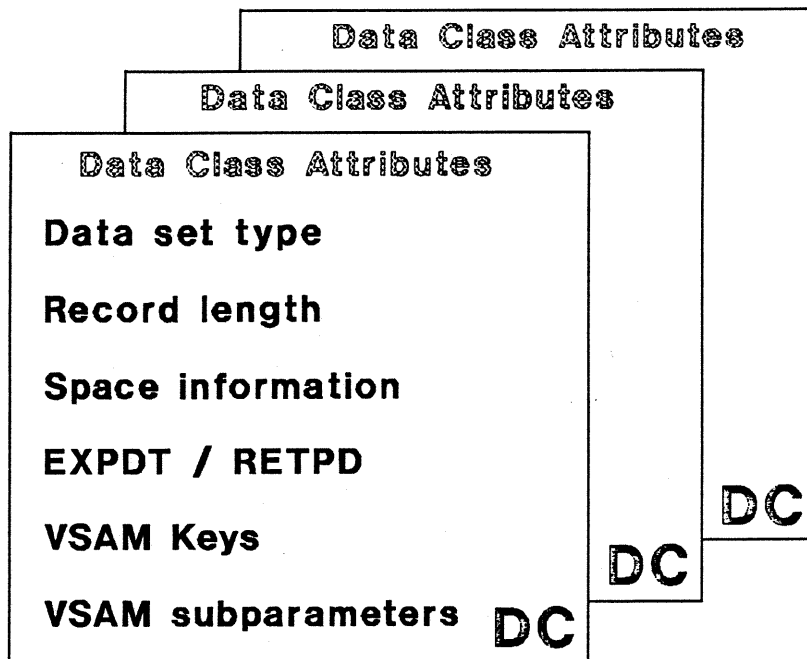
The major functions in SMS that affect the view of the life cycle of data are:

- Data Class
- Storage Class
- Management Class
- Storage Group

Since these functions describe installation policies and practices, they must be defined or constructed by the customer. As such the definitions of the four areas are called **constructs**.



## DATA CLASS



BE051913

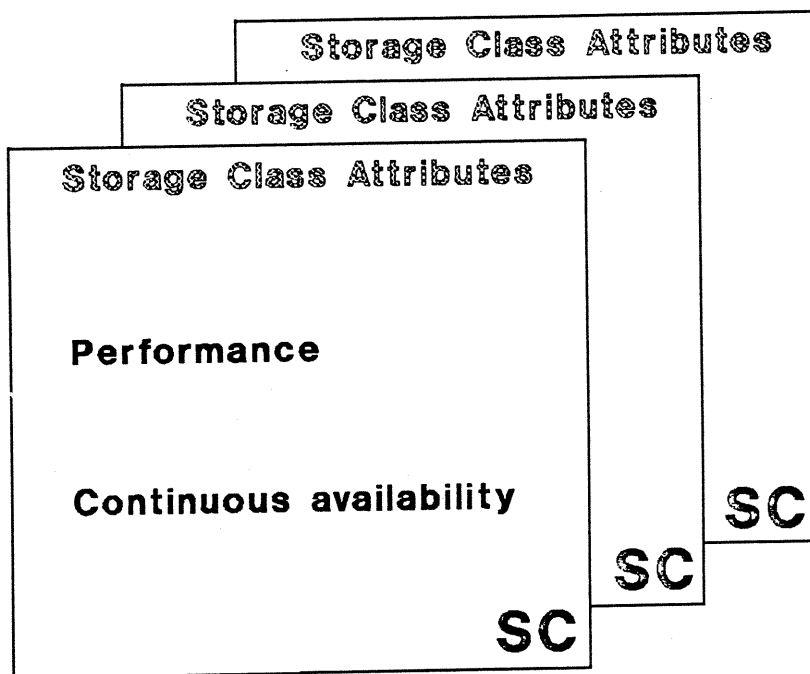
Figure 9-4. Data Class

### Data Class

Data Class provides data set modeling capability that allows system standards to be established and available for all users. This allows the installation to reduce the amount of JCL coding required by application programmers.



## STORAGE CLASS



BE051914

Figure 9-5. Storage Class

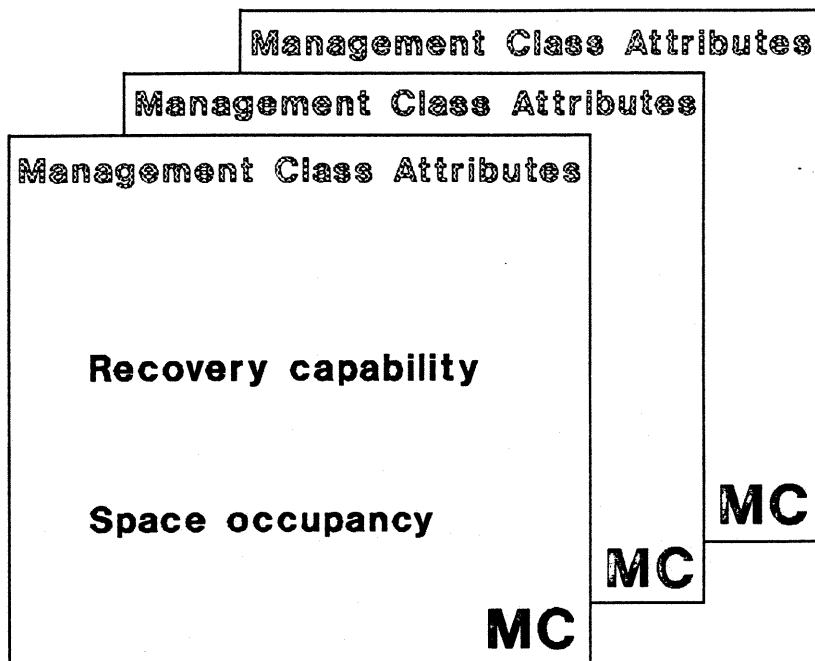
### Storage Class

Data sets may have performance and/or availability requirements. This information must be known before a data set is placed upon a volume. If a duplicate copy is required, then the data set must be placed on a volume that the 3990 knows as a dual copy volume so that duplication is performed. For performance requirements, if a service level is required, then the system needs to know this in order to try to select a volume that is capable of providing the requested level (for example, a cached volume).

Storage class applies only to system-managed data sets. If a data set is assigned a storage class, it is then managed by DFSMS. This means that it may also have a management class assigned and will be assigned to a system-managed volume.



## MANAGEMENT CLASS



BE051915

Figure 9-6. Management Class

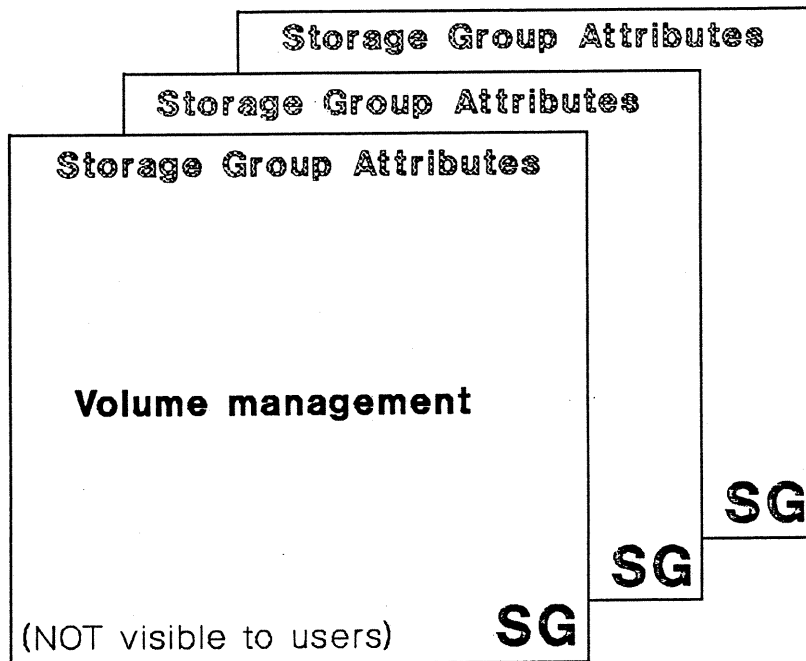
### Management Class

Management class addresses the issues as to what can or should be done to or for a data set after it has been placed upon a volume.





# STORAGE GROUP



BE051916

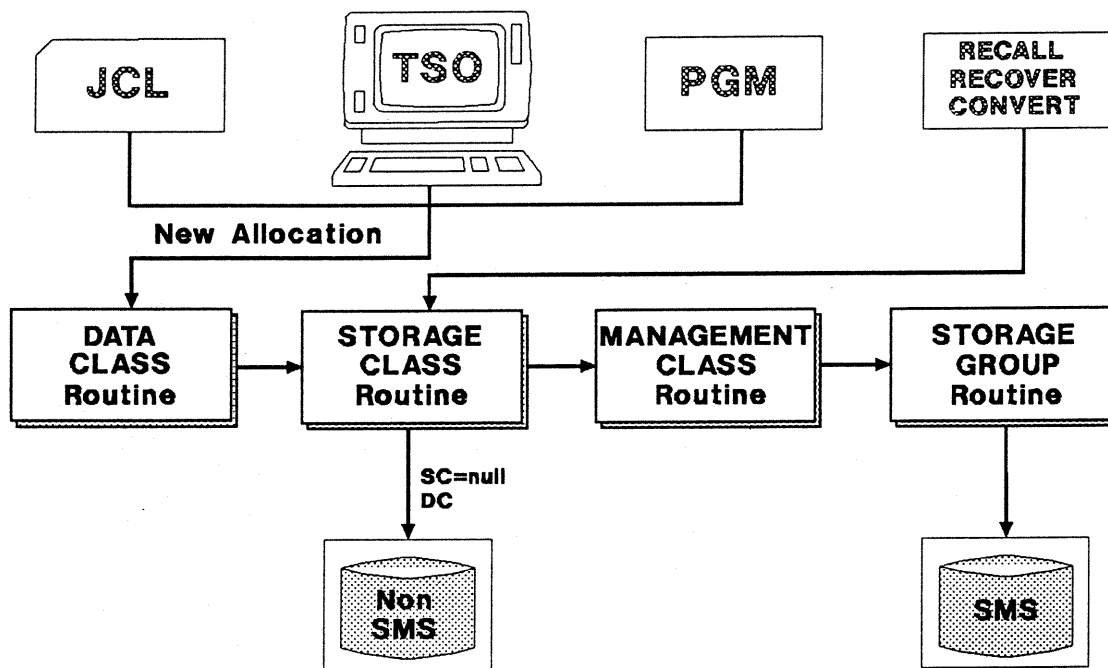
Figure 9-7. Storage Group

## Storage Group

Volume selection is now truly under the control of the system! There is no external (JCL, TSO Allocate, etc.) parameter available to specify a storage group.



## ACS PROCESSING SEQUENCE



BE051917

Figure 9-8. Automatic Class Selection Environment

### Automatic Class Selection Environment

Not all data sets will be managed by the storage management subsystem. This is especially true as you begin to implement DFSMS. The decision point is in the storage class selection routine. If a storage class is assigned, then the data set is system-managed. If not, it will be directed to a volume via the same methods as are currently in use (UNIT=, etc.).



# ***LET'S GET LOGICAL !***

```
//F1 DD DSN=SOME.DS.DATA,DISP=(,CATLG),  
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=9040),  
// SPACE=(9040,(2200,220)),  
// UNIT=SYSDA,  
// VOL=SER=PVT032
```

BE051918

Figure 9-9. Physical to Logical

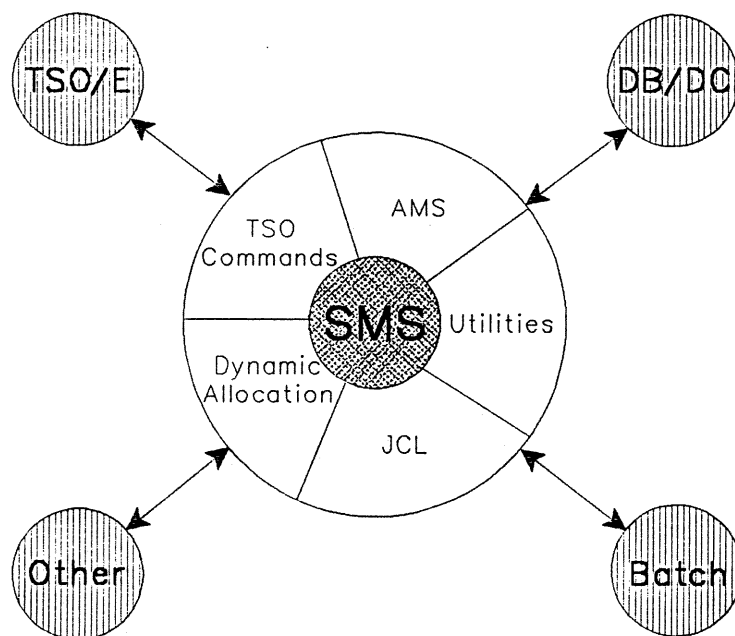
## **Physical to logical**

Many existing parameters are physically related:

- VOL =
- UNIT =
- SPACE = (tracks/cylinders/blocksize)
- DCB = (BLKSIZE = ...)



## Users and Access to Data



BE051919

Figure 9-10. Users accessing their Data

### Users accessing their data

**SMS User Interface** The objectives of SMS external changes are

- Simplify externals required to define data to the system
- Separate logical data requirements from physical device characteristics
- Use common keywords in all of the above shown user areas
- Provide enhanced functions for
  - Definition of VSAM data sets
  - Allocation of temporary VSAM data sets
  - Determination of blocksize by the system





---

## SMS can manage

- Physical sequential
- Partitioned
- VSAM
- BDAM
- Generation data groups (GDG)
- Temporary
- VIO

BE05191A

---

Figure 9-11. SMS Abilities

### SMS abilities

SMS, typically, manages only data sets residing on DASD volumes.



## SMS does not manage

- Uncataloged/cataloged in non-ICF ctlgs
- Non-ICF VSAM catalogs
- Tapes
- CVOLS
- ISAM
- VSAM data spaces
- Unmovable
- Absolute track allocated
- Pattern DSCBs for GDGs
- SYSIN and SYSOUT
- MVS resident data sets

BE05191B

Figure 9-12. Data Sets not managed

### Data Sets not managed

- IBM recommends ICF catalogs as the strategic catalog type. Therefore SMS relies on ICF catalogs only.
- ISAM data sets and VSAM data spaces are not strategic data set types. They should be replaced by VSAM clusters.
- Unmovable data sets and absolute track allocation are a contradiction per se to optimal space management and thus not supported by SMS.
- Pattern DSCBs are non-cataloged data sets and not supported by SMS.
- SYSIN and SYSOUT data sets remain to be handled by the JES subsystems.



## JCL Changes

---

### Overview JCL Changes

- JOBCAT/STEP CAT elimination
- Promoted keywords
- Support of SMS specifications
- New ways to reference 'model'
- VSAM definitions via JCL

BE05191C

---

Figure 9-13. Overview JCL Changes

#### Overview JCL changes



---

## JOB CAT/STEP CAT Elimination

- **SMS requires the STANDARD SEARCH ORDER for all jobs referencing SMS managed data sets.**
- **If JOB CAT/STEP CAT specified**
  - **JCL error occurs**
  - **true for all catalog functions**

BE05191D

---

Figure 9-14. JOB CAT/STEP CAT Elimination

### JOB CAT/STEP CAT elimination

- Step/job will also fail, if the referenced catalog is managed by SMS.

**Recommendation**    Remove JOB CAT/STEP CAT and use

- Standard search order
- DFDSS INCAT parameter (if authorized)
- AMS CATALOG parameter (if authorized)





## Current JCL Keywords

### Strategic

DSNAME  
DISP  
SPACE

### Promoted

RECFM  
LRECL  
KEYLEN  
RETPD  
EXPDT

### Nonstrategic

~~UNIT~~      ~~VOLUME~~  
~~DCB~~      ~~AMP~~  
~~TRK~~      ~~CYL~~

BE05191E

Figure 9-15. Current JCL Keywords

### Current JCL keywords

#### Example:

#### Current JCL

```
//NEWDATA DD DSN=MY.NEW.DATA,DISP=(NEW,CATLG),UNIT=SYSDA,
//          VOL=SER=123456,SPACE=(TRK,(45,45)),
//          DCB=(LRECL=180,BLKSIZE=3600,RECFM=FB)
```

#### New JCL

```
//NEWDATA DD DSN=MY.NEW.DATA,DISP=(NEW,CATLG),LRECL=180
```



## New JCL Keywords

### All Data Sets

DATACLAS  
RECORD  
AVGREC  
KEYOFF  
LIKE  
REFDD  
SECMODEL

### SMS Managed

STORCLAS  
MGMTCLAS

BE05191F

Figure 9-16. New JCL Keywords

### New JCL keywords

- Storage administrator defines class names and inherent attributes.
- User can browse existing data classes for information and reference using ISMF.
- Use of storage and management classes can be protected by RACF.
- ACS routines can override class names.
- SMS capable systems ignore parameters if specified for existing data sets.
- If SMS not active, system checks for correct syntax and ignores parameters.
- If parameters specified on pre-SMS system, JCL errors result.



## JCL Data Class

- Specifies data class for new system- or non-system-managed DASD data sets
- Data class is a named list of default allocation attributes

RECORD  
RECFM  
LRECL  
KEYLEN

KEYOFF  
RETPD/EXPDT  
SPACE  
Volume count

VSAM options (IMBED, REPLICATE, CISE, FREESPACE, SHAREOPTIONS)

BE05191G

Figure 9-17. JCL Data Class

### JCL Data Class

The idea of predefined data classes is convenience to the user. The space administrator defines a set of named JCL lists, that the user may choose from.

- Attributes specified on DD statement override corresponding attributes in selected data class.
- ACS cannot override DD specified attributes.
- ACS can override data class name specified on DATACLAS parameter.
- Data class ACS routine entered whether data set system-managed or not.
- Using a data class guarantees DSORG/RECORD assigned. If DSORG/RECORD not specified, but data class specified or ACS assigned, then DSORG defaults to PS/PO depending upon SPACE specification.



## JCL Storage Class

- **Specifies storage class for new system-managed data sets**
- **Storage class is a named list of objectives**  
**Performance**  
**Availability**
- **Request for service, not for resource**

BE05191H

Figure 9-18. JCL Storage Class

### JCL Storage Class

- This service request may be asked for explicitly by specifying the `STORCLAS =` parameter within JCL or
- The service request is derived implicitly by the ACS (Automatic Class Selection) routines based on the data set name.
- A request for performance will be satisfied if the appropriate type of device is available. If not, the allocation will not fail, the next best fitting device will be chosen.
- A request for *availability* will fail, if there is no device to realize the request.





---

## **JCL Management Class**

- **Specifies management class for new system-managed data sets**
- **Named list of availability and space management attributes**
  - ▬ **Partial release**
  - ▬ **Backup**
  - ▬ **Retention**
  - ▬ **Migration**
- **Specifications direct DFHSM activities**

BE05191I

---

Figure 9-19. JCL Management Class

### **JCL Management Class**

- JCL RETPD/EXPDT takes precedence over management class expiration attributes



## JCL Space Information

- **SPACE=(,.,dir))**  
Override number of directory blocks specified in data class
- **AVGREC=U|K|M**  
Specifies scale modification to average record length request,  
**SPACE=(reclgth,(prim-qty,sec-qty))**
  - U - multiply by 1
  - K - multiply by 1024
  - M - multiply by 1048576

BE05191J

Figure 9-20. JCL Space Information

### JCL space information

- If block size not specified
  - If RECFM specified, calls DASD calculation service (DCS) to calculate optimal block size based on allocated device type, LRECL and RECFM.
  - If RECFM not specified, uses 4096 for block size.
- Allocates space based on block size, SPACE, AVGREC (if specified) and device type
  - If generic device type specified (e.g. 3380) actual device geometry used for space allocation.
  - If esoteric device type specified (e.g. SYSDA) or no UNIT parameter specified, default device geometry is used for space allocation.



## New JCL LIKE Parameter

- Specifies properties of an existing cataloged data set to be used to allocate a new data set

REORG  
RECFM  
SPACE

LRECL  
KEYLEN  
KEYOFF

- Replaces DCB=dsname
- Inconsistencies between JCL and TSO LIKE
  - BLKSIZE, EXPDT copied for TSO LIKE, not so for JCL LIKE
  - JCL LIKE copies directory size, TSO prompts for it

BE05191K

Figure 9-21. JCL LIKE Parameter

### JCL LIKE parameter

There is a similar reference statement to LIKE

```
REFDD=*.ddname
      *.stepname.ddname
      *.stepname.procstepname.ddname
```

- Data set properties defined in a previous DD statement and associated data class are used to allocate new data set
- Same properties as LIKE
- Replaces: DCB = \*.ddname
  - \*.stepname.ddname
  - \*.stepname.procstepname.ddname
- REFDD and LIKE are mutually exclusive keywords
- Allocation attributes for a data set are assembled
  1. Explicit attributes from own JCL
  2. Attributes from referenced data set or JCL
  3. Defined attributes from specified data class



## VSAM via JCL

### VSAM Data Set via JCL

```
//NEWVSAM DD DSN=MY.NEW.VSAM,DISP=(NEW,CATLG),  
//          LRECL=120,  
//          SPACE=(120,(1000,500)),  
//          AVGREC=K,  
//          RECOrg=KS,  
//          KEYLEN=17,  
//          KEYOFF=6
```

BE05191L

Figure 9-22. VSAM Data Set via JCL

#### VSAM data set via JCL

- JCL may be used to allocate temporary or permanent VSAM data sets.
- VOLSER no longer has to be specified.
- RECOrg specifies data set organization
  - KS: VSAM Keyed Sequenced
  - ES: VSAM Entry Sequenced
  - RR: VSAM Relative Record
  - LS: VSAM Linear Space
- KEYOFF is equivalent to ACB subparameter RKP
- KEYLEN specifies length of the key and replaces ACB subparameter KEYLEN
- Not all AMS DEFINE CLUSTER attributes can be specified via JCL or data class, e.g. KEYRANGES
- Not all data class attributes can be specified via JCL
- If JCL allocates VSAM data set DISP=(OLD,DELETE)
  - Without SMS DELETE is ignored
  - With SMS data set will be deleted
- With SMS VSAM data sets can be passed within a job. System replaces PASS with KEEP for non-temporary VSAM data sets





## Cataloging Permanent SMS Data Sets

- All permanent SMS data sets are
  - ✦ cataloged at time of space allocation
  - ✦ uncataloged when data set is scratched
- Allocation of non-VSAM data sets causes
  - ✦ the creation of a BCS entry
  - ✦ the creation of a VVDS entry
- Uncataloged data sets cannot be SMS managed

BE05191M

Figure 9-23. Cataloging Permanent SMS Data Sets

### Cataloging permanent SMS data sets

- Disp KEEP will default to CATLG
- DISP = (NEW,PASS)
  - Data set cataloged at creation
  - Data set deleted at end of step specifying DISP = (OLD,DELETE) or at end of last step in job
- DISP = (NEW,DELETE)
  - Data set cataloged at creation
  - Data set deleted at end of step
- Potential incompatibilities for DISP = (NEW,PASS) or DISP = (NEW,DELETE)
  - Allocation may fail, because data set already cataloged by that name
  - Data set may orient towards catalog for which user not authorized



## Cataloging Temporary VSAM Data Sets

- **Cataloging temporary VSAM data sets**
  - **No BCS entry is created**
  - **Only VVR(s) are built**
  - **Standard temporary data set name**
- **Limitations**
  - **No AIX or PATH support**
  - **SHAREOPTIONS(1) only**
  - **Single volume only**
- **Temporary non-VSAM data sets remain uncataloged**

BE05192N

Figure 9-24. Cataloging Temporary VSAM Data Sets

### Cataloging Temporary VSAM data sets

- Reside on volumes managed by SMS.
- Specified same as non-system-managed temporary data sets, except storage class explicitly specified or ACS assigned
- Requirements
  - DSNAMES start with &, && or omitted
  - Explicit or ACS assigned storage class
  - DISP *status* must be NEW
  - DISP *normal-term-disp* must be PASS or DELETE
- System-managed non-VSAM temporary data sets are candidates for VIO



## GDG Processing

---

### GDG Processing

- SMS-managed GDG consists of cataloged sequential and/or direct data sets residing on DASD volumes
- SMS-managed GDG may contain both SMS- and non-SMS-managed GDSs
- GDSs in a GDG can have like or unlike data set attributes and data set organisations
- SMS affects GDG processing due to its implementation of mandatory cataloging. Changes are:
  - ▢ Allocation without pattern DSCB
  - ▢ Allocation of new generation
  - ▢ AMS ALTER command usage
  - ▢ Disposition processing of new generations

BE051910

---

Figure 9-25. GDG Processing

#### GDG processing

#### Generation Data Groups

- Pattern DSCBs are not used for allocation of SMS-managed GDSs, because they are uncataloged data sets.
- Pattern DSCBs are not allowed on SMS volumes, because their names conflict with the name of the GDG base.
- Pattern DSCBs are not supported for allocation of non-SMS-managed GDS, when catalog is on SMS-managed volume.

#### Defining a GDG

- Before creating the first GDS a GDG base entry in an ICF catalog must be built using the AMS DEFINE GDG command.
- Requirement for SMS-managed GDG is the association of a storage class for a new GDS.
- Attributes defined in data class and storage class are used when creating a new GDS.

*Example:* GDS allocation JCL

```
//DD1 DD DSN=GDG(+1),DISP=(NEW,CATLG),AVGREC=K,
//    SPACE=(133,(40,5)),STORCLAS=PRIMARY,
//    DATACLAS=FIXLIST
```