UNIT 8

GDG

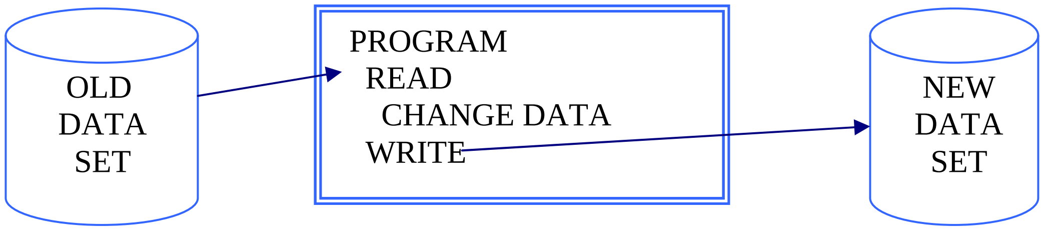
GDG (Generation Data groups)

- Generation Data Sets
- Relative Generation Numbers
- Building a GDG Base Entry
- Defining Attributes for Generation Data Sets
- Creating a Generation Data Set
- Retrieving a Generation Data Set

Objectives

- Understand the need for the GDGs
- Ways of Coding GDGs

Generation Data Groups



A Generation Data Group (GDG) consists of like-named data sets that are chronologically or functionally related. A data set in a GDG is called a generation.

Why use Generation Data Groups (GDGs)?

You can only catalog one entry of a particular dataset name. In other words, once a dataset (ABC.DATA for example) is cataloged, you cannot catalog a new version of that dataset (another ABC.DATA).

Case1: You attempt to catalog a new data set with the same name as an existing data set.
CATALOG ERROR1 NAME ALREADY IN THE CATALOG.

Case2: You keep the new data set with the same name.
JCL ERROR! DUPLICATE NAME ON VTOC.

Case3: You catalog or keep the data set with different names.
JCL STATEMENT MUST BE CHANGED TO THE NEW NAMES

PURPOSE OF GDG's:

For datasets that get created on a daily basis, GDG's allow you to create new versions of a dataset where the base name of the dataset remains the same, and a new *generation* number gets tagged onto the end of each new version of the dataset name.

One example of GDG use might be a daily transaction log. Each day a new file is created showing the transactions of that day. The base name of the transaction log might be TRANS.LOG, where the actual generations would be called TRANS.LOG.G0001V00, TRANS.LOG.G0002V00, TRANS.LOG.G0003V00, etc.

GDG- DSNAME specification

Generation data sets (generations) have two types of names: Relative and Absolute

```
DSN=ABD.DAILY.TRANS(+n)                DSN=ABC.DAILY.TRANS.GxxxxVyy

(+n) Add a new generation              GxxxxVyy:
(+0) Use current generation number       xxxx   Generation number
(-n) Use an old generation               yy     Version number

EXAMPLE                                EXAMPLE
DSN=ABC.DAILY.TRANS(+1)                DSN=ABC.DAILY.TRANS.G0053V00
```

Absolute Data Set Name:

The absolute data set name is the true data set name you would see on the volume table of contents (VTOC).

The Format is:

```
DSN = NAME.GDG.G0052V00

                    VERSION NUMBER
               GENERATION NUMBER
```

Relative Data Set Name:

The relative name is the most common form of GDG name used in JCL coding. It refers to a generation number relative to the most current generation in the catalog.

The format is:

```
DSN=NAME.GDG(+n)    or    DSN = NAME.GDG(-n)
```

where 'n' is the relative generation number of the GDG.

Possible Formats

`(+n)`	Create a new generation by adding 'n' to the current generation in the catalog **`DSN=NAME.GDG(+1)`**
`(0)`	Use the current generation in the catalog **`DSN=NAME.GDG(0)`**
`(-n)`	Use the generation that is 'n' less than the current generation **`DSN=NAME.GDG(-1)`**

Example

Current Catalog Entries
```
ABC.TRANS.DAILY.G0053V00
ABC.TRANS.DAILY.G0052V00
ABC.TRANS.DAILY.G0051V00
```

With the current catalog entries shown above, the following GDG's would be translated as shown:

Dataset	Translates to
`DSN=ABC.TRANS.DAILY(-1),DISP=OLD`	**`ABC.TRANS.DAILY.G0052V00`**

The current generation in the catalog is G0053V00. Since 53 – 1 = 52, G0052V00 is the generation requested. This is common when reading the previous cycle's file.

`DSN=ABC.TRANS.DAILY(0),DISP=OLD`	**`ABC.TRANS.DAILY.G0053V00`**

The current generation in the catalog is G0053V00. This is used to read the most current file available.

`DSN=ABC.TRANS.DAILY(+1),DISP=(NEW,CATLG)`	**`ABC.TRANS.DAILY.G0054V00`**

The current generation in the catalog is G0053V00. Since 53 + 1 = 54, G0054V00 is the generation requested. This is used when creating a new file for the current cycle.

GDG- DSNAME specification (Continued)

The generation number starts with G0001 for the first generation and is incremented by the value coded in the relative data set name.

For Example:

RELATIVE NAME	ABSOLUTE NAME	
`NAME.GDG(+0)`	`NAME.GDG.G0003V00`	current generation
`NAME.GDG(-1)`	`NAME.GDG.G0002V00`	current generation -1
`NAME.GDG(-2)`	`NAME.GDG.G0001V00`	current generation -2

IBM does not use the V00 or Version number. The version number is set to V00 with the first generation. If a generation is damaged and needs to be replaced, an installation can code the absolute name with a new version number, to replace the damaged generation, such as

```
     //INDD      DD DSN=NAME.GDG.G0052V00,DISP=OLD
     //OUTDD     DD DSN=NAME.GDG.G0052V01,DISP=(,CATLG)
```

GDG DSNAME – BASE CATALOG ENTRY

Before a generation data set can be created by a job, a GDG base Catalog entry and model DSCB (or pattern DSCB) must be defined to the catalog. (SMS does not support model DSCB's).

This is a sample IDCAMS job stream, which can be used to create both the catalog entry and build a model DSCB.

```
//JOBNAME   JOB, ,……
//STEP1     EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//DSCB      DD DSN=TEST.MODELDSCB.GDG1,SPACE=(TRK,0),
            DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
            UNIT=SYSDA,DISP=(,KEEP)
//SYSIN     DD *
      DEFINE GENERATIONDATAGROUP  -
      (NAME (TEST.GDG)            -
      NOSCRATCH                   -
      NOEMPTY                     -
      LIMIT (3))
```

There are five control statement parameters which are used in creating a GDG base catalog entry. They are:

LIMIT	Maximum number of generations allowed for this GDG entry
EMPTY	When limit is exceeded, uncatalog all generations.
NOEMPTY	When limit is exceeded, uncatalog oldest entry only
SCRATCH	Delete any uncataloged generation.
NOSCRATCH	Do not delete any uncataloged generation.

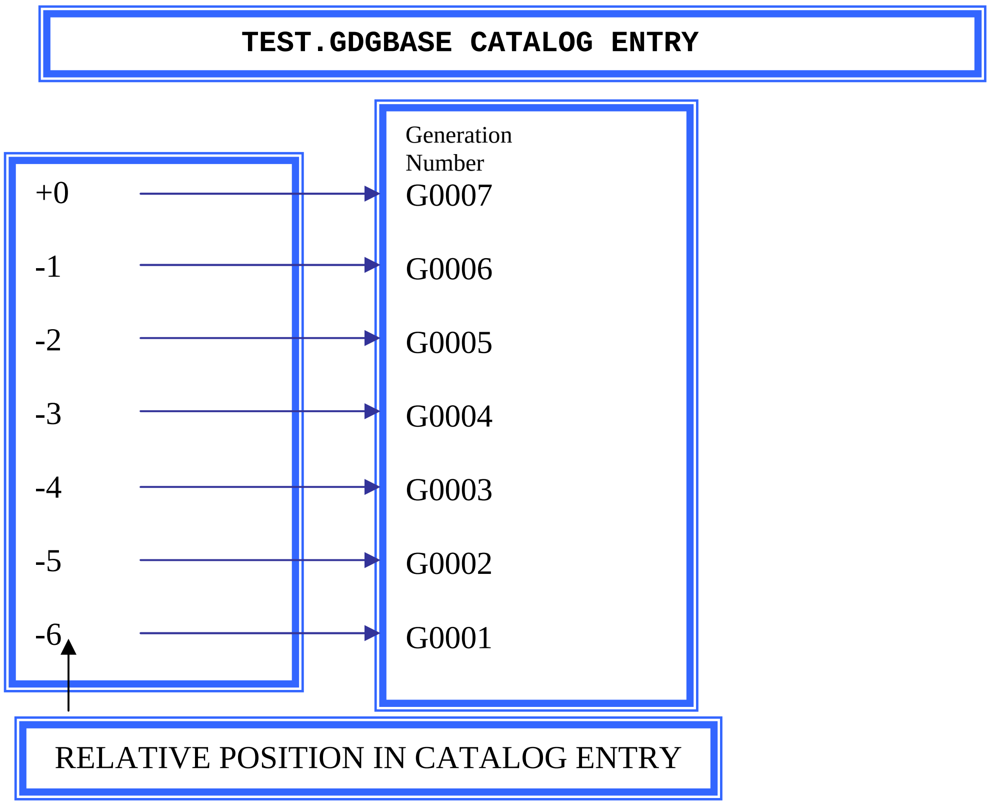
The model DSCB (//DSCB in the example above) ensures that the same DCB and EXPDT information is used to create all generations. This ensures greater consistency among generations.

The model DSCB must be allocated with Zero space, and it cannot be cataloged.

The model DSCB must also lie on the same volume where the base catalog entry is cataloged.

GDG DSNAME – CATALOG ENTRY

The following example shows a view of a GDG Catalog entry.



EXAMPLES:

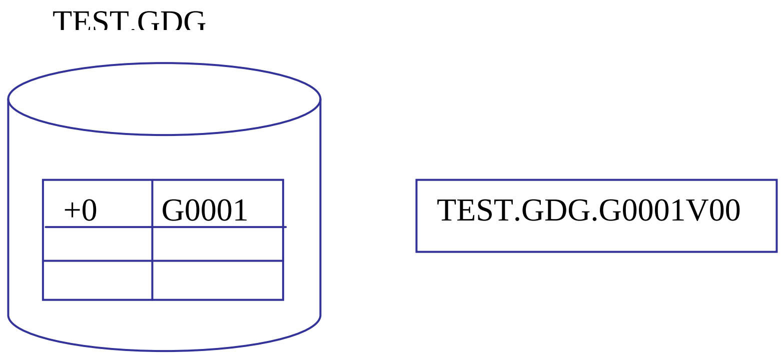
`//DD1 DD DSN=TEST.GDG(+0),DISP=OLD`	locates current(top) entry
`//DD2 DD DSN=TEST.GDG(-6),DISP=OLD`	locates oldest entry

Notice that the catalog entry operates like a pushdown stack. As new generations get created, the relative numbers for existing generations changes.

GDG EXAMPLE - FIRST GENERATION

The following example shows what happens when the first generation of a GDG gets created.

```
//EXAMPLE  JOB  378,SMITH,CLASS=T
//STEP1    EXEC PGM=USERPGM1
//FIRST    DD   DSN=TEST.GDG.(+1),DISP=(,CATLG),
//INPUT    DD   DSN=INITIAL.DATA,DISP=OLD
```



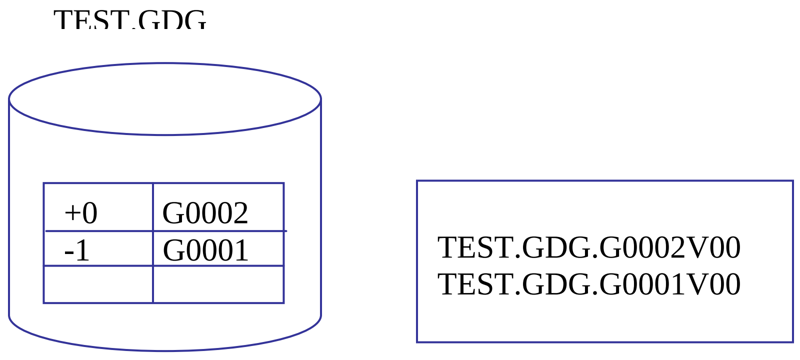
As the first generation of the generation data group named TEST.GDG is created:

- Note that the DISP=(,CATLG) is required when creating a new generation.
- Relative generation (+0) is generation G0001V00.
- The GDG Base must be created prior to creating any generation of the dataset.

GDG EXAMPLE – SECOND GENERATION

The following example shows what happens when the second generation of a GDG gets created.

```
//EXAMPLE  JOB   378,SMITH, CLASS=G
//STEPX    EXEC  PGM=MAINLINE
//GDGIN    DD    DSN=TEST.GDG.(+0),DISP= OLD
//GDGOUT   DD    DSN=TEST.GDG(+1),DISP=(NEW, CATLG),
//
```



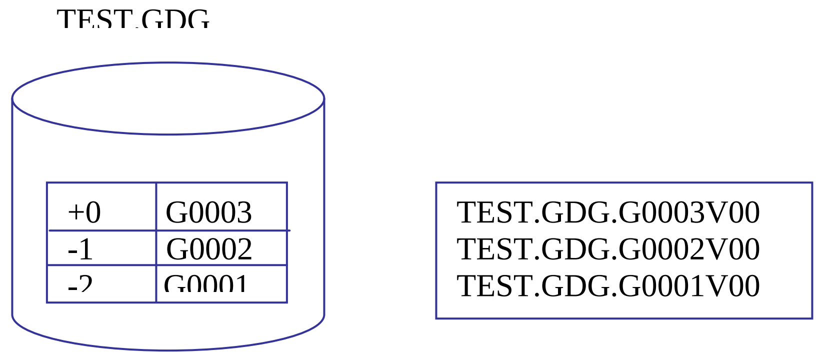
As the second generation of the generation data group named TEST.GDG is created:

- The current generation (0) is used as input. You are not required to code the plus sign with the zero (+0), a zero without the plus sign (0) is acceptable and commonly used.
- A new generation (+1) is created.
- Notice that the catalog entry operates like a push-down stack. Generation G0001V00 is now the (-1) generation.

GDG EXAMPLE – THIRD GENERATION

The following example shows what happens when the third generation of a GDG gets created.

```
//EXAMPLE  JOB   378,SMITH, CLASS=G
//STEPX    EXEC  PGM=MAINLINE
//GDGIN    DD    DSN=TEST.GDG(+0),DISP= OLD
//GDGOUT   DD    DSN=TEST.GDG(+1),DISP=(NEW, CATLG),
//
```

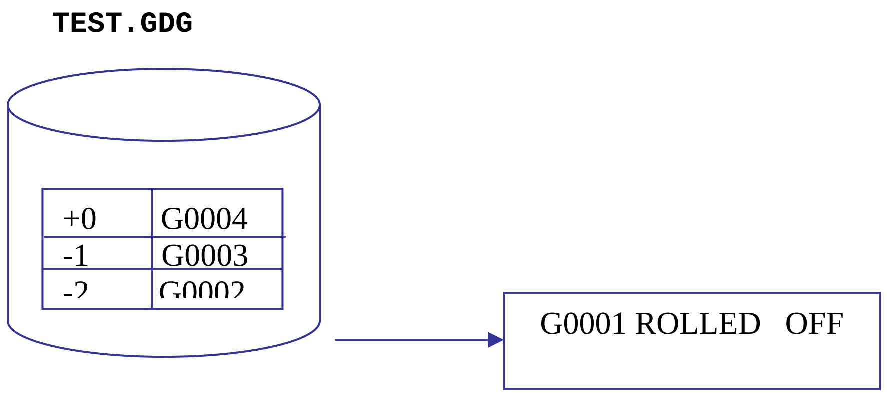


As the third generation of the generation data group named TEST.GDG is created:

- The current generation (0) is used as input.
- A new generation (+1) is created.
- Notice that the catalog entry operates like a pushdown stack.

GDG EXAMPLE - LIMIT EXCEEDED

Once a GDG reaches the number of generations defined in its LIMIT, the oldest generation "rolls off" the catalog as a new generation gets created. The old generation may automatically be kept or scratched, depending on the option you chose when you defined the GDG base.



```
//EXAMPLE  JOB   378,SMITH, CLASS=G
//STEPX    EXEC  PGM=MAINLINE
//GDGIN    DD    DSN=TEST.GDG(+0),DISP= OLD
//GDGOUT   DD    DSN=TEST.GDG(+1),DISP=(NEW, CATLG),
//
```

As the fourth generation of the generation data group named TEST.GDG is created:

- The current generation (0) is used as input.
- A new generation (+1) is created
- The first generation, **`TEST.GDG.G0001V00`**, will be removed from the list of cataloged entries, since the LIMIT was defined as 3.
- Since NOSCRATCH was specified, this generation will be kept, though it will no longer be cataloged

Other Notes

- If the generations are SMS managed, this generation will remain in the catalog.
- If the generations were NN-SMS managed, the generation would have been uncataloged.
- The maximum no. of generations that MVS can manage for a data set is 255.

GDG’s IN A MULTISTEP JOB

```
//STEP1      EXEC       PGM=ONE
//INBIG      DD         DSN=A.AIR,DISP=OLD
//OUT1       DD         DSN=A.AIR(+1),DISP=(,CATLG,DELETE) …….

--------------

//STEP2     EXEC        PGM=TWO
//IN2       DD          DSN=PROD.DATA,DISP=OLD
//OUT2      DD          DSN=A.AIR(+2),DISP=(,CATLG,DELETE) …


----------

//STEP3     EXEC        PGM=THREE
//IN3       DD          DSN=A.AIR(+1),DISP=OLD
//          DD          DSN=A.AIR(+2),DISP=OLD
```

STEP1

- A GDGALL request (DSN=A.AIR) is IBM’s terminology for automatic concatenation of ALL generations currently in the catalog.
- The (+1) generation in DD ‘OUT1’ creates a new generation.

STEP3

- Notice that the (+1) generation is now OLD. Even though that generation was cataloged in STEP1, and the catalog pushed down, you do not adjust the relative generation numbers within the same job. The system does that internally.

- As additional generations are specified in this step, increment the relative generation number. An increment of 1 is shown, but any increment can be used.

Notes

- Relative GDG numbers are not updated until end of job, therefore the relative numbers are held constant throughout a single job.

- Use DISP=OLD when processing a GDG to prevent other jobs (on the same system) from processing the same GDG at the same time.

Creating a Generation Data Set (Non-SMS)

When creating a new non-SMS-managed generation data set, always code the parameters

- DSNAME
- DISP
- UNIT

Optionally, code the parameters

- VOLUME
- SPACE
- LABEL
- DCB

In the DSNAME parameter, code the name of the GDG followed by a number, +1 to +255, in parentheses.

If this is the first dataset being added to a GDG in the job, code +1 in parentheses. Each time in the job you add a dataset to the same GDG, increase the number by one.

When referring to this data set in a subsequent job step, code the relative generation number used to create it on the DSNAME parameter. You cannot refer to the dataset in the same step in which it was created.

At the end of the job, the system updates the relative generation numbers of all generations in the group to reflect the additions.

Deleting a Generation Data Set

Use the DELETE parameter of the IDCAMS utility to delete a GDG from the catalog, such as:

```
//MODAL2    JOB  ‘0.2AMIP’
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=*
//SYSIN     DD    *
     DELETE   ABC.DATA.MONTHLY  FORCE
/*
//
```

In this example the GDG called **ABC.DATA.MONTHLY** is deleted. The FORCE parameter physically purges all entries related to GDG.

Unit 8 Exercises

1. The dataset `ABC.GDG.SAMPLE(+1)` refers to the ____________________ generation of the GDG `ABC.GDG.SAMPLE`.

2. The dataset `ABC.GDG.SAMPLE.G0023V00` refers to the ______________________ generation of the GDG `ABC.GDG.SAMPLE.`

3. In IDCAMS, the_______________________ option indicates that when older generations roll off the catalog, MVS should remove the generation and uncatalog it, but the dataset is not deleted.

4. In IDCAMS, the ______________________ parameter is used to delete a GDG base from the catalog.

5. Which of the following will read the current generation of `XYZ.TRANS.DAILY?` *`(circle the correct answer)`*

   a. `XYZ.TRANS.DAILY(-1)`

   b. `XYZ.TRANS.DAILY(0)`

   c. `XYZ.TRANS.DAILY(+1)`

   d. `None of the above`

Unit 8 Lab Exercises

Logon to TSO/ISPF and perform the following exercises. Wherever you see "userid" in lower case, substitute your valid security userid.

JOB 1:

STEP 1:

Create a GDG called userid.GDG.CARBASE that will contain the output from a weekly run. Therefore, the group should be large enough to contain a rolling 52 weeks of data.

A Model DSCB called USERID.TEST.GDG. should be used.
Create and catalogue the Model DSCB with the following specifications:

Logical Record Length	**- 80**
Block size	**- 800**
Record Format	**- Fixed Block**

STEP 2, STEP 3 & STEP 4:

Create three GDG's in three separate steps using the model DSCB attributes.

- **The first GDG should contain the data: THIS IS THE DATA FOR WEEK 1.**
- **The second GDG should contain the data: THIS IS THE DATA FOR WEEK 2.**
- **The third GDG should contain the data: THIS IS THE DATA FOR WEEK 3.**

JOB 2:

Read the three GDS's concurrently & print them to SYSOUT=X in reverse order so that they appear as follows:

THIS IS THE DATA FOR WEEK THREE.
THIS IS THE DATA FOR WEEK TWO.
THIS IS THE DATA FOR WEEK ONE.

Example

```
//JOBNAME   JOB, ,……
//STEP1     EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//DSCB      DD DSN=USERID.GDG.MODEL,SPACE=(TRK,0),
            DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
            UNIT=SYSDA,DISP=(,KEEP)
//SYSIN     DD *
      DEFINE GENERATIONDATAGROUP  -
      (NAME (USERID.GDG.MODEL)            -
      NOSCRATCH                   -
      NOEMPTY                     -
      LIMIT (53))

//EXAMPLE   JOB  378,SMITH,CLASS=T
//*FIRST GENARATION(FIRST WEEK DATA)
//STEP1     EXEC  PGM=USERPGM1
//FIRST     DD    DSN=USERID.TEST.GDG.(+1),DISP=(,CATLG),
//INPUT     DD    DSN=USREID.INITIAL.DATA,DISP=OLD
//

//EXAMPLE   JOB  378,SMITH,CLASS=T
//*SECOND GENARATION(SECOND WEEK DATA)
//STEP1     EXEC  PGM=USERPGM1
//GDGIN     DD    DSN=USERID.TEST.GDG.(+0),DISP=OLD
//GDGOUT    DD    DSN=USERID.TEST.GDG(+1),DISP=(,CATLG)
//

//EXAMPLE   JOB  378,SMITH,CLASS=T
//*THIRD GENARATION(THIRD WEEK DATA)
//STEP1     EXEC  PGM=USERPGM1
//GDGIN     DD    DSN=USERID.TEST.GDG.(+0),DISP=OLD
//GDGOUT    DD    DSN=USERID.TEST.GDG(+1),DISP=(,CATLG)
//
```

Notes