***IMS DB***

**IMS DB IMP POINTS**

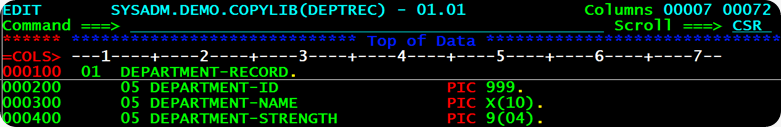
-----------------------------------------------------------------------------------------------------------------------------

->Go to **FAI** and press **sc** before the root to view all data present in db

**Questions and their Explanation**

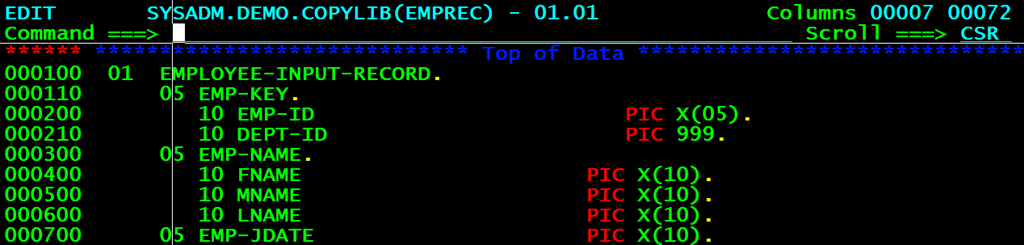
**Q. What is IMS/DB? (http://www.mainframes360.com/2010/11/introduction-to-imsdb.html)**

ANS. IMS(Information Management System) is the Hierarchical-Data Management System from IBM. When hierarchical data-structures are implemented using Flat-Files, it is the responsibility of Application-Programmers to ensure that it works flawlessly. Why storing huge-volumes of data having hierarchical-relationship in Flat-Files is a bad idea? Let's say, a list of Employees and the Departments in which they work – this Data is stored in Flat Files.

[](http://lh6.ggpht.com/_sQvdFWqMlMg/TM_sM5zD1wI/AAAAAAAADsE/Bntu5psZd_Q/s1600-h/image4.png)

The Departments Data is stored in DEPARTMENT File. The record-layout of the DEPARTMENT File is shown in the picture above. Every Department has a DEPARTMENT-ID, that uniquely identifies the Department. Moreover, a Department has a DEPARMENT-NAME and the total number of employees working under that department are recorded in DEPARTMENT-STRENGTH.

Every Department of the company has several Employees working under it. The Employees data is stored in EMPLOYEE File. The snap above shows its record-layout. Each Employee is identified by Unique EMP-KEY, consisting of EMP-ID and DEPT-ID. The Employee's name EMP-NAME is further broken down in FNAME, MNAME and LNAME. The Employee’s Joining Date is stored in EMP-JDATE. This hierarchical relationship between DEPARTMENT File and EMPLOYEE File is depicted in the picture below.

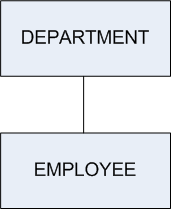


However, there are certain problems with this design.

1. Maintaining Data Consistency - For instance, if the Program adds a new Employee-Record, that has no DEPT-ID, no corresponding department, the DEPARTMENT and EMPLOYEE Files will no longer be synchronized. And the same is true, if a Department-Record is deleted, when active Employees work for that Department. These Employee-Records are orphaned out.

2. COBOL Program and Data Dependency – Because the structure of the Data(File Layout) is hard-coded or embedded in the COBOL Program accessing the data, whenever the data's structure changes(say you add one more field to the DEPARTMENT File), the COBOL Program has to be re-written and re-compiled all over again. **The Program heavily depends on the Data.**

When you use IMS/DB for storing data, difficulties like these don't come up. That's because the hierarchical relationships between data is now taken care of by IMS. IMS would use a single file for storing DEPARTMENT and its EMPLOYEE's data. The structural information is stored in IMS.

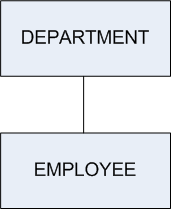


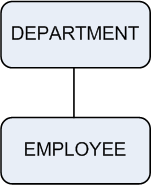
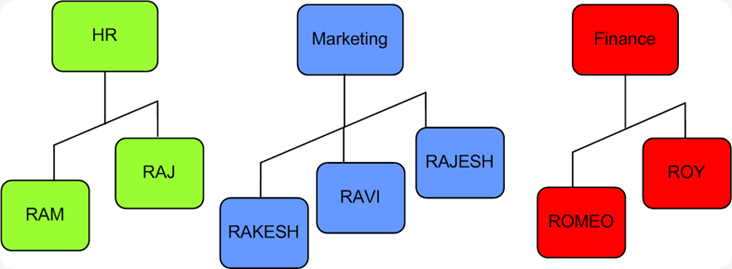
**Q.What is DBMS in IMS?**

ANS.A Database Management System is used to define and maintain the structure of the database.

**Q. What is meant by Segment-Type, Segment-Occurrence and Levels in IMS?**

In IMS/DB, when data is stored about any real-world thing, you call it aSegment. For example, if you record information about Employees, Employee is one-Segment. Because you would be storing  Departments is another segment. In IMS Jargon, I would call them DEPARTMENT Segment and EMPLOYEE Segment.   
  
I'll introduce two new terms as well – Segment Type and Segment Occurrence. Asegment-type is one category or class of data. For example, DEPARTMENT Segment-Type. Each instance of a Department is called a Segment-Occurrence. For example, you may have 3 segment-occurrences HR, MARKETING and FINANCE of theSegment-Type DEPARTMENT. Bear in mind, there can be only one segment-type of a particular kind, but infinite occurrences of it.   
  
An IMS Hierarchical-Database can be looked upon as an inverted-tree. The Height of the Tree tells you the number-of-levels in the Tree. 

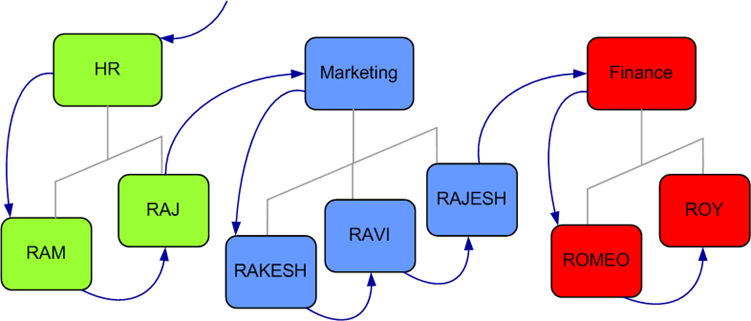
[](http://lh4.ggpht.com/_sQvdFWqMlMg/TM_sR8vNrKI/AAAAAAAADsc/HKUj29rrImo/s1600-h/image17.png)   
  
This Hierarchical-database has two Levels. **Q. What is meant by Root-Segment and Database Record?**

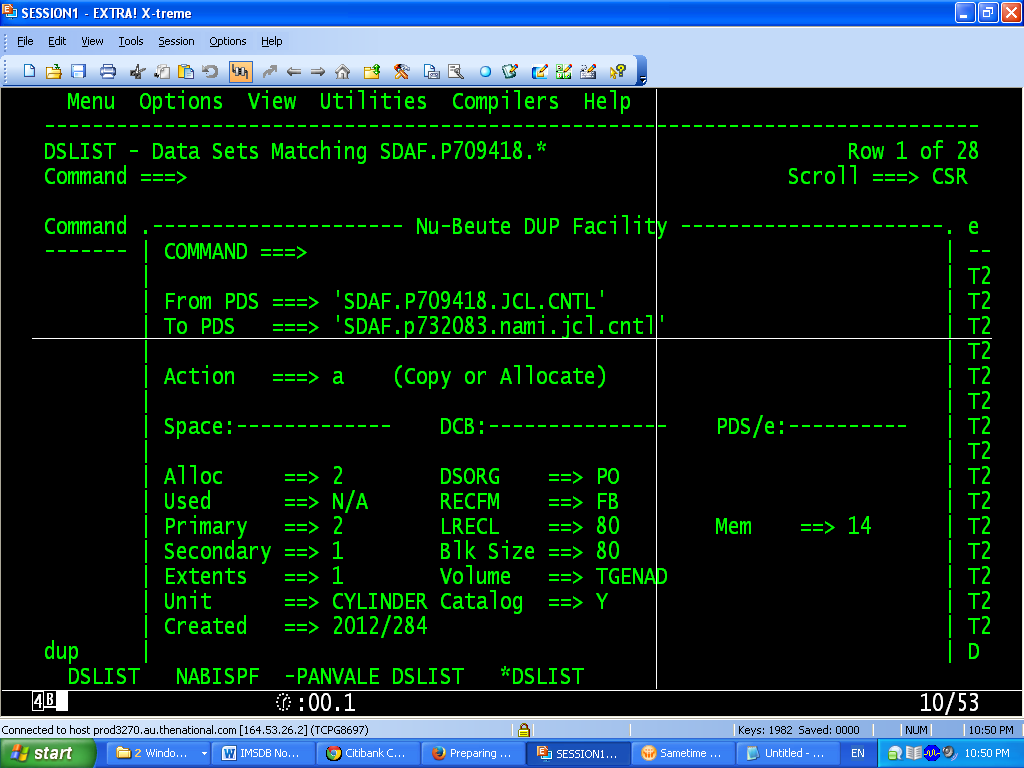
In an IMS Hierarchical-Database, the segment-type at the top of the hierarchy-tree is called the Root-Segment. For example, in the below database, theDEPARTMENT-Segment is called the Root-Segment.   
  
      [](http://lh6.ggpht.com/_sQvdFWqMlMg/TM_sTrg0C4I/AAAAAAAADsk/r-xpSJK0MJE/s1600-h/image27.png)   
Suppose the DEPARTMENT Segment-type has 3 segment-occurrences HR, Marketing and Finance. Each of these Department segment-occurrences have Employee Segment-occurrences subordinate to them. HR Department has 2 sub-ordinate Employee Segment-Occurrences RAM and RAJ. Marketing Department has 3 sub-ordinate Employee-Segment occurrences RAKESH, RAVI and RAJESH. Finance Department has 2 sub-ordinate Employee Segment-occurrences ROMEO and ROY.   
  
  [](http://lh6.ggpht.com/_sQvdFWqMlMg/TM_sVTSV0pI/AAAAAAAADss/ZL5i7kzvo9U/s1600-h/image23.png)   
One root segment-occurrence e.g. HR plus all the segment-occurrences sub-ordinate to it for example RAM and RAJ, together constitute one   
Database-Record. Thus, here there are 3 database-records, one for each root segment-occurrence. In all, there are 10 segment-occurrences in this IMS Database.

**Q. What are dependent, parent and Child and twin Segments?**

All the segments in an IMS Hierarchical DB, other than the Root-Segment are called Dependent-Segments. DEPARTMENT Segment is called Parent-Segment and EMPLOYEE Segment is its Child-Segment.   
  
The term Twin segments is applicable to Segment-Occurrences. Two children with same parent segment-occurrence are called twins. For example, RAKESH and RAVI are child of the same parent segment occurrence Marketing Department, so they are called Twin-Segments.

**Q. How data is accessed in an IMS Hierarchical Database?**

Data can be accessed in two-ways from an IMS-Database – **Sequential** and **Randommode** of access. When you access data from IMS DB Sequentially, you read it record-by-record. Within each record, the segments are accessed in the following general pattern : Top-to-Bottom, Left-to-Right.   
  
First, a root segment-occurrence is retrieved. Then, IMS DB digs as far deep as possible, right at the bottom, until it reaches the lowest level Segment-Occurrence. It then retrieves all the twin segment-occurrences at that level. Once, all the Twin Segments have been fetched, IMS moves back up in the hierarchy and retrieves any twins of the Parent. This process continues, till all the Database Records have been retrieved. So, the sequence in which the Data is fetched is HR-RAM-RAJ, MARKETING-RAKESH-RAVI-RAJESH and FINANCE-ROMEO-ROY.   
  
  [](http://lh6.ggpht.com/_sQvdFWqMlMg/TM_sXoJk46I/AAAAAAAADs0/CrZHjg-Gnyw/s1600-h/image39.png)   
  
Random-Access to Data in a Hierarchical-Database is possible by supplying **aconcatenated-Key**. Every Segment can have a **Key-Field(or Sequence Field).** For example, DEPARTMENT-ID may be a Key-Field of DEPARTMENT Segment. EMPLOYEE-ID would be Key-Field of Employee-Segment. A Concatenated-Key is formed by combining the Key-Fields of different segment-types  e.g.(DEPARTMENT-ID,EMPLOYEE-ID). This way you can jump or skip  directly to any particular Segment-Occurrence under any Database Record.



**Q.What are the objectives of DBMS in IMS?**

**ANS.**

1.Reduces data redundancy

2.Provides data communication facilities

3.Reduces data maintenance

4.Provides data integrity and security

5.Provides indexing capabilities

**Q.What is IMS (DB/DC)?**

ANS.IMS (Information Management System) is IBM’s hierarchical database management system. It has mainly two components: IMS DB and IMS DC (also know as IMS TM)

IMS DB - IMS/Database Manager as the name implies manages the IMS databases. It is used for physical storage creation and management and data retrieval.

IMS DC / IMS TM - IMS/Data Communications or IMS/Transaction Manager handles online transaction processing system.

**Q.What do you mean by Hierarchical Database?**

**ANS.**

1.Follows inverted tree structure

2.Data relationship is predefined by it's structure

3.Program accesses the data through predefined paths.

**Q. What is meant by Segment-Type, Segment-Occurrence and Levels in IMS?**

**ANS.**

In IMS/DB, when data is stored about any real-world thing, you call it a Segment. For example, if you record information about Employees, Employee is one-Segment. Because you would be storing Departments is another segment. In IMS Jargon, I would call them DEPARTMENT Segment and EMPLOYEE Segment.

I'll introduce two new terms as well – Segment Type and Segment Occurrence. A segment-type is one category or class of data. For example, DEPARTMENT Segment-Type. Each instance of a Department is called a Segment-Occurrence. For example, you may have 3 segment-occurrences HR, MARKETING and FINANCE of the Segment-Type DEPARTMENT. Bear in mind, there can be only one segment-type of a particular kind, but infinite occurrences of it.

An IMS Hierarchical-Database can be looked upon as an inverted-tree. The Height of the Tree tells you the number-of-levels in the Tree.

This Hierarchical-database has two Levels.

***IMS DB Notes***

*IMS is an IBM program product that provides* ***transaction management*** *and* ***database management*** *functions for large commercial application systems. It was* ***originally introduced in 1968****.*

*There are two major parts to IMS,*

* *a* ***Data Communication manager (DC)*** *and*
* *a* ***Database Manager (DB).***

***Application programming Languages***

* Applications written to use IMS functions can be written in a number of programming languages.
* Programming languages currently supported are **Assembler, C, COBOL, Pascal, PL/I and REXX.**
* The IMS resources are accessed by the application by calling a number of standard IMS functions.
* Applications access these functions through **a standard application programming interface (API)** for both the Transaction Manager and Database Manager components.
* This interface is normally *referred to as* ***Data Language I (DL/I).***

***Control Blocks***

* ***Physical structure of a DL/I data base isn’t specified in an application program***
* ***DL/I uses a set of control blocks(DBDs and PSBs) to define a data base’s structure***
* ***Data Base Descriptor (DBD)***
  + *Describes the complete structure of a data base*
  + *A unique DBD for each DL/I data base*
* ***Program Specification Block (PSB)***
  + *Application program’s view of the Database*
  + *PSB Specifies* 
    - *Data bases (one or more) a program can access,*
    - *Data elements a program can “see” in those data bases*
    - *The processing a program can do with the data elements*
  + *Application programs that have similar data base processing requirements can share a PSB*
* *Data Base Administrator (DBA) has to create DL/I control blocks*
* *DBDGEN and PSBGEN Control Statements*

**SAMPLE DBDGEN (Explained)**

**STMT SOURCE STATEMENT**

1 **PRINT** NOGEN

2 **DBD** **NAME**=INDBD,**ACCESS**=HIDAM

3 **DATASET** DD1=IN,DEVICE=3380

4 \*\*/ 3380 DISK STORAGE

5 \*

6 **SEGM** NAME=INVENSEG, PARENT=0,**POINTER**=TB,BYTES=131

7 **LCHILD** NAME=(INPXPNTR,INPXDBD),**POINTER**=INDX

8 **FIELD** **NAME**=(INVENCOD,**SEQ**),**BYTES**=3,**START**=1,**TYPE**=C

9 FIELD NAME=INVENNAM,BYTES=30,START=4,TYPE=C

10 FIELD NAME=INVENADR,BYTES=30,START=34,TYPE=C

11 FIELD NAME=INVENCIT,BYTES=17,START=64,TYPE=C

12 FIELD NAME=INVENSTA,BYTES=2,START=81,TYPE=C

13 FIELD NAME=INVENZIP,BYTES=9,START=83,TYPE=C

14 FIELD NAME=INVENTEL,BYTES=10,START=92,TYPE=C

15 FIELD NAME=INVENCON,BYTES=30,START=102,TYPE=C

16 \*

17 SEGM NAME=INITMSEG,PARENT=INVENSEG,BYTES=48

18 FIELD NAME=(INITMNUM,SEQ),BYTES=5,START=1,TYPE=C

19 FIELD NAME=INITMDES,BYTES=35,START=6,TYPE=C

20 FIELD NAME=INITMPRC,BYTES=4,START=41,TYPE=P

21 FIELD NAME=INITMCST,BYTES=4,START=45,TYPE=P

22 \*

23 SEGM NAME=INLOCSEG, PARENT=INITMSEG,BYTES=21

24 FIELD NAME=(INLOCLOC,SEQ),BYTES=3,START=1,TYPE=C

25 FIELD NAME=INLOCONH,BYTES=4,START=4,TYPE=P

26 FIELD NAME=INLOCROP,BYTES=4,START=8,TYPE=P

27 FIELD NAME=INLOCONO,BYTES=4,START=12,TYPE=P

28 FIELD NAME=INLOCDAT,BYTES=6,START=16,TYPE=C

29 \*

30 **DBDGEN**

72 \*\*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

73 \*\*/ RECOMMENDED VSAM DEFINE CLUSTER PARAMETERS

74 \*\*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

75 \*\*/\* \*NOTE2

76 \*\*/\* DEFINE CLUSTER (NAME(IN) NONINDEXED -

77 \*\*/\* RECORDSIZE (2041,2041) -

78 \*\*/\* COUNTERINTERVALSIZE (2048))

79 \*\*/\* \*NOTE2 - SHOULD SPECIFY DSNNAME FOR DD IN

80 \*\*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

162 \*\*/\*\*\*\*\*\*\*\*\*\*\*SEQUENCE FIELD\*\*\*\*\*\*\*\*\*\*\*\*\*

211 \*\*/\*\*\*\*\*\*\*\*\*\*\*SEQUENCE FIELD\*\*\*\*\*\*\*\*\*\*\*\*\*

325 FINISH

326 END

***Explanation***

* + ***First macro – DBD*** *– identifies the data base and specifies the DL/I access method*
  + ***Second macro – DATASET****- identifies the file that would contain the data base*
  + ***Symbolic name (IN)*** *identifies the data set in the JCL at execution time*
  + *Segment types are defined using the* ***SEGM macro***
  + *Segment hierarchical relationships are specified by the* ***PARENT*** *parameter on a SEGM macro*
    - ***PARENT= 0 or absence of PARENT parameter specifies root segment***
  + ***POINTER parameter and LCHILD macro are needed for HIDAM Databases***
  + *Only search fields need be specified in the DB*
  + ***FIELD macro*** *defines a field in the DB*
    - **START** ⇨ *position of field within segment*
    - ***NAME***⇨ *name of the field*
    - ***LENGTH***⇨ *length of the field*
    - ***TYPE***⇨ *data type of the field*

|  |  |
| --- | --- |
| ***FIELD Macro TYPE Codes*** | ***Data Type*** |
| *C* | *Character* |
| *P* | *Packed decimal* |
| *Z* | *Zoned decimal* |
| *X* | *Hexadecimal* |
| *H* | *Half word Binary* |
| *F* | *Full word Binary* |

* + ***SEQ*** *parameter specifies a sequence field*
    - ***segment occurrences are added in sequence by values in these fields***

**SAMPLE PSBGEN**

**STMT SOURCE STATEMENT**

1 PRINT NOGEN

2 **PCB** **TYPE**=DB,**DBDNAME**=INDBD,PROCOPT=LS

3 **SENSEG** NAME=INVENSEG

4 SENSEG NAME=INITMSEG,PARENT=INVENSEG

5 SENSEG NAME=INLOCSEG,PARENT=INITMSEG

6 **PSBGEN** **PSBNAME**=INLOAD,LANG=COBOL

87 END

***Explanation***

* + ***PCB*** *(Program Communication Block) refers to one data base.*
  + *One PCB macro for each database accessed*
  + ***Segment Level Sensitivity***
    - *A program’s access to parts of the data base identified at the segment level*
    - *Within sensitive segments, the program has access to all fields*
  + ***Field level sensitivity***
    - *When the program accesses that segment, only sensitive fields are presented*
  + ***DBDNAME*** *parameter on the PCB macro specifies the name of the DBD*
  + ***KEYLEN*** *parameter specifies the length of the longest concatenated key the program can process in the data base*
  + ***PROCOPT*** *parameter specifies the program’s processing options*
  + *For each PCB macro, subordinate* ***SENSEG*** *macros identify the sensitive segments in the data base*
  + ***Names specified in the SENSEG macros must be segment names from the DBDGEN*** *for the data base named in the DBDNAME parameter of the PCB macro*
  + ***PSBGEN*** *macro*
    - *Indicates that there are no more statements in the PSBGEN job*
    - ***PSBNAME*** *parameter specifies the name to be given to the output PSB module*

***LANG*** *parameter specifies the language in which the related application program will be written.*

**IMS Processing Options**

* *Indicates to IMS the type of access allowed for a sensitive segment (SENSEG)*
* *Commonly used Processing Options*
  + ***PROCOPT=G*** *means only read-only access*
  + ***PROCOPT=R*** *means read/replace access*
  + ***PROCOPT=I*** *means insert access allowed*
  + ***PROCOPT=D*** *means Read/Delete access*
  + ***PROCOPT=A*** *means all the above options present*
  + ***For GSAM DBs PROCOPT=LS for output and GS (Get Sequential) for input***
  + ***PROCOPT=L allows a 'load' into the DB. If VSAM DB, it should be empty prior to the load***
* ***The PROCOPT given for a Sensitive segment would override the one given for the DB***
  + *Example : -*

*PCB TYPE=DB,NAME=LDB42F,PROCOPT=G, KEYLEN=200*

*SENSEG NAME=SEGL4201, PARENT=0,PROCOPT=A*

* + *WARNING : Indiscriminate use of PROCOPTS can lead to inexplicable results !*

**ACB & ACBGEN**

* ***ACB(Application Control Blocks)*** *: It is created by merging and expanding PSB’s and DBD’s into an IMS internal format when an application program is scheduled for execution.*
* ***ACBGEN :*** *The process of building ACB is called Block Building and is done by means of ACBGEN.*
* *IMS can build ACB’s either dynamically or it can prebuild them using ACB maintenance utility.*
* ***ACB’s cannot be prebuilt for GSAM DBD’s.***
* ***ACB’s can be prebuild for PSB’s that reference GSAM databases.***
* ***ACB’s save instruction, execution and direct-access wait time and improves performance in application scheduling.***
* *ACB’s are maintained in* ***IMS.ACBLIB*** *library.*

**Running an application program under DL/I**

* ***Batch program does not access IMS directly***
* ***JCL invokes the DL/I ‘batch initialization module’ DFSRRC00 which loads the application program and the required DL/I modules***
* ***The program and DL/I modules execute together***
* ***Sample JCL :***

//JOBNAME JOB (ACCT),'PGMR NAME',

// CLASS=J,

// MSGCLASS=Z,

// NOTIFY=&SYSUID

//JOBLIB DD DSN=YOUR.PROGRAM.LOAD.LIBRARY,

// DISP=SHR

// DD DSN=YOUR.SYSTEM.RESLIB.LIBRARY,

// DISP=SHR

//PROC EXEC PROCNAME, SYMBOLIC PARAMETERS

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//PROCNAME PROC

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\* THIS PROC LOADS AN IMS VSAM DATABASE

//\* A PROGRAM 'LOAD' IS USED FOR THIS PURPOSE

//\* THE PSB USED FOR LOADING IS LOADPSB

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//LOAD EXEC PGM=**DFSRRC00**,

// **PARM**='DLI,LOAD,LOADPSB'

//**DFSRESLB** DD DSN=YOUR.DFRESLIB.LIBRARY,

// DISP=SHR

//**IMS** DD DSN=YOUR.DBD.LIBRARY,

// DISP=SHR

// DD DSN=YOUR.PSB.LIBRARY,

// DISP=SHR

//**IMSLOGR** DD DSN=YOUR.IMSRLOG.DATASET,

// DISP=SHR

//**IEFRDER** DD DSN=YOUR.IEFRDER.DATASET,

// DISP=OLD

//\* DD NAMES ARE AS SPECIFIED IN THE DATABASE

//**DATA** DD DSN=VSAMDB.DATA.PART,DISP=SHR

//**INDEX** DD DSN=VSAMDB.INDEX.PART,DISP=SHR

//**INPUT** DD DSN=FILE.USED.FOR.LOADING,

// DISP=SHR

//**DFSVSAMP** DD DSN=IMSVS.PROCLIB(DFSVSAMP),

// DISP=SHR

//**CPXMOPTS** DD DSN=PARMLIB.LIBRARY(LOAD),

// DISP=SHR

//**CPXMRPTS** DD SYSOUT=\*

//**SYSOUT** DD SYSOUT=\*

//**SYSPRINT** DD SYSOUT=\*

//**SYSUDUMP** DD SYSOUT=\*

//**IMSERR** DD SYSOUT=\*

//**IMSPRINT** DD SYSOUT=\*

**COBOL Basics for Processing a DL/I Data Base**

**ENTRY and GO BACK Statements**

* ***Format of the DL/I ENTRY Statement***

**ENTRY ‘DLITCBL’** **USING** PCB-name1

[PCB-name2...]

* *Application program is invoked under the control of the batch initialization module*
* ***DLITCBL => ‘DL/I to COBOL’ is the entry point to the program***
* *DL/I supplies the address of each PCB defined in the program’s PSB*
* ***PCBs must be defined in the Linkage Section***
* *Linkage Section definition of a PCB is called a ‘****PCB Mask’***
* *Addressability to PCBs established by listing the PCB Masks on the ENTRY statement*
* ***PCB masks should be listed on the ENTRY statement in the same sequence as they appear in your program’s PSBGEN***
* ***GO BACK Statement***
  + *When a program ends, it passes control back to the DL/I*
  + *DL/I reallocates resources and closes the data base data sets*

*Use GO BACK and not a STOP RUN statement*

**The DL/I Call**

* ***CALL statements are used to request DL/I services***
* ***Format of the DL/I call***

*Parameters you code on the CALL statement specify, among other things, the operation you want DL/I to perform*

**CALL ‘CBLTDLI’** USING DLI-function

PCB-mask

segment-io-area

[segment-search-argument(s)]

* ***CBLTDLI => ‘COBOL to DL/I’, is an interface module that is link edited with your program’s object module***

*PLITDLI, ASMTDLI are other options*

* ***The DL/I Function***
  + ***First parameter coded on any DL/I call***
  + ***Four character working storage field containing the function code***

***DL/I function codes***

**01 DLI-FUNCTIONS.**

05 DLI-GU PIC X(4) VALUE ‘GU ’.

05 DLI-GHU PIC X(4) VALUE ‘GHU ’.

05 DLI-GN PIC X(4) VALUE ‘GN ’.

05 DLI-GHN PIC X(4) VALUE ‘GHN ’.

05 DLI-GNP PIC X(4) VALUE ‘GNP ’.

05 DLI-GHNP PIC X(4) VALUE ‘GHNP’.

05 DLI-ISRT PIC X(4) VALUE ‘ISRT’.

05 DLI-DLET PIC X(4) VALUE ‘DLET’.

05 DLI-REPL PIC X(4) VALUE ‘REPL’.

05 DLI-CHKP PIC X(4) VALUE ‘CHKP’.

05 DLI-XRST PIC X(4) VALUE ‘XRST’.

05 DLI-PCB PIC X(4) VALUE ‘PCB ’.

* ***COBOL doesn’t allow coding literals in a CALL statement***

***Get functions***

* *First six 05-level items above*
* *Used to retrieve segments from a DL/I data base*
* ***GU***⇨ *‘get unique’ function causes DL/I to retrieve a specific segment occurrence based on field values that you specify*
* ***GN***⇨ *‘get next’ function used to retrieve segment occurrences in sequence*
* ***GNP***⇨ *‘get next within parent’ function lets you retrieve segment occurrences in sequence, but only subordinate to an established parent segment*
* *The three get function codes that contain an H are ‘****get hold functions’*** *and are used to specify an intent to update a segment after you retrieve it*
* ***GHU*** *or the ‘get hold unique’ function corresponds to GU*
* ***GHN*** *or the ‘get hold next’ function corresponds to GN*
* ***GHNP*** *or the ‘get hold next within parent’ function corresponds to GNP*

***Update functions***

* *Used to change data in the data base*
* *ISRT or the ‘insert’ function is used to add a new segment occurrence to a data base– whether it be change an existing data base or to load a new one*
* *DLET or the ‘delete’ function is used to remove a segment from a data base*
* *REPL or the ‘replace’ function is used to replace a segment occurrence* ***Other functions***
* *Functions* ***CHKP*** *(the ‘checkpoint’ function) and XRST (the ‘restart’ function) are used in programs to take advantage of IMS’s recovery and restart features*
* *Function* ***PCB*** *is used in CICS programs*
* *Function* ***SYNC*** *is used for releasing resources that IMS has locked for the program (applicable only in a BMP)*
* *Function* ***INIT*** *allows an application to receive status codes regarding deadlock and data availability (from DB PCBs)*
* ***PCB mask***
  + *Second parameter on the DL/I call*
  + *The name of the PCB mask defined in the program’s Linkage Section*
  + *ENTRY statement establishes a correspondence between PCB masks in the Linkage Section and the PCBs within the program’s PSB*
  + ***After each DL/I call, DL/I stores a status code in the PCB mask, which the programmer can use to determine whether the call succeeded or failed***
* ***Segment I/O Area***
  + *Third parameter on the DL/I call*
  + ***Name of the working storage field into which DL/I will return retrieved data or from which it will get data for an update operation***
* ***Segment search argument***
  + *Optional parameter on the DL/I call*
  + *Identifies the segment occurrence you wish to access*
  + *Multiple SSAs on a single DL/I call*
  + *Two kinds of SSAs– unqualified and qualified*
  + ***An unqualified SSA***
    - *Supplies the name of the next segment type that you want to operate on*
    - *If you issue a GN call with an unqualified SSA, DL/I will return the next occurrence of the segment type you specify*
  + ***A qualified SSA***
    - *Combines a segment name with additional information that specifies the segment occurrence to be processed*
    - *A GU call with a qualified SSA might request a particular occurrence of a named segment type by providing a key value*

**The PCB Mask**

* ***For each data base your program accesses, DL/I maintains an area of storage called the program communication block (PCB)***
* ***Masks are defined for those areas of storage in the Linkage Section of your program***

01 INVENTORY-PCB-MASK.

05 **IPCB-DBD-NAME PIC X(8).**

05 **IPCB-SEGMENT-LEVEL PIC XX.**

05 **IPCB-STATUS-CODE PIC XX.**

05 **IPCB-PROC-OPTIONS PIC X(4).**

05 FILLER PIC S9(5) COMP.

05 **IPCB-SEGMENT-NAME PIC X(8).**

05 **IPCB-KEY-LENGTH PIC S9(5) COMP.**

05 **IPCB-NUMB-SENS-SEGS PIC S9(5) COMP.**

05 **IPCB-KEY PIC X(11).**

* ***Data base name***
  + *The name of the data base being processed*
* ***Segment level***
  + *Specifies the current segment level in the data base*
  + *After a successful call, DL/I stores the level of the segment just processed in this field*
* ***Status code***
  + *Contains the DL/I status code*
  + *When* ***DL/I successfully completes the processing you request in a call, it indicates that to your program by moving spaces to the status code field in the PCB***
  + ***If a call is unsuccessful or raises some condition that isn’t normal, DL/I moves some non-blank value to the status code field***
  + *It is good programming practice to evaluate the status code after you issue a DL/I call*
* ***Processing options*** 
  + *Indicates the processing a program is allowed to do on the data base*
* ***Segment name feedback area***
  + *The name of the segment is stored by DL/I in this field after each DL/I call.*
* ***Key length feedback area***
  + *The field DL/I uses to report the length of the concatenated key of the lowest level segment processed during the previous call*
  + *Used with the key feedback area*
* ***Number of sensitive segments***
  + *Contains the number of SENSEG macros subordinate to the PCB macro for this data base*
* ***Key feedback area***
  + *Varies in length from one PCB to another*
  + *As long as the longest possible concatenated key that can be used with the program’s view of the data base*
  + *After a data base operation, DL/I returns the concatenated key of he lowest level segment processed in this field, and it returns the key’s length in the key length feedback area*