LEARN IMS DB

information management system

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

IMS stands for Information Management System. IMS was developed by IBM with Rockwell and Caterpillar in year 1966 for the Apollo program to send a man to the moon. It started the database management system revolution and still continues to evolve to meet data processing requirements.

IMS provides an easy-to-use, reliable, and standard environment for executing high-performance transactions. IMS database is used by high-level programming languages like COBOL to store data in hierarchical arrangement and access it.

Audience

This tutorial is designed for software programmers who are interested in understanding the concepts of IMS database starting from scratch. This tutorial gives enough understanding on IMS data management from where you can take yourself to higher levels of expertise.

Prerequisites

Before proceeding with this tutorial, you should have a basic understanding COBOL programming skills. A basic knowledge of database concepts will help you understand the IMS database management system.

Copyright & Disclaimer

© Copyright 2018 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

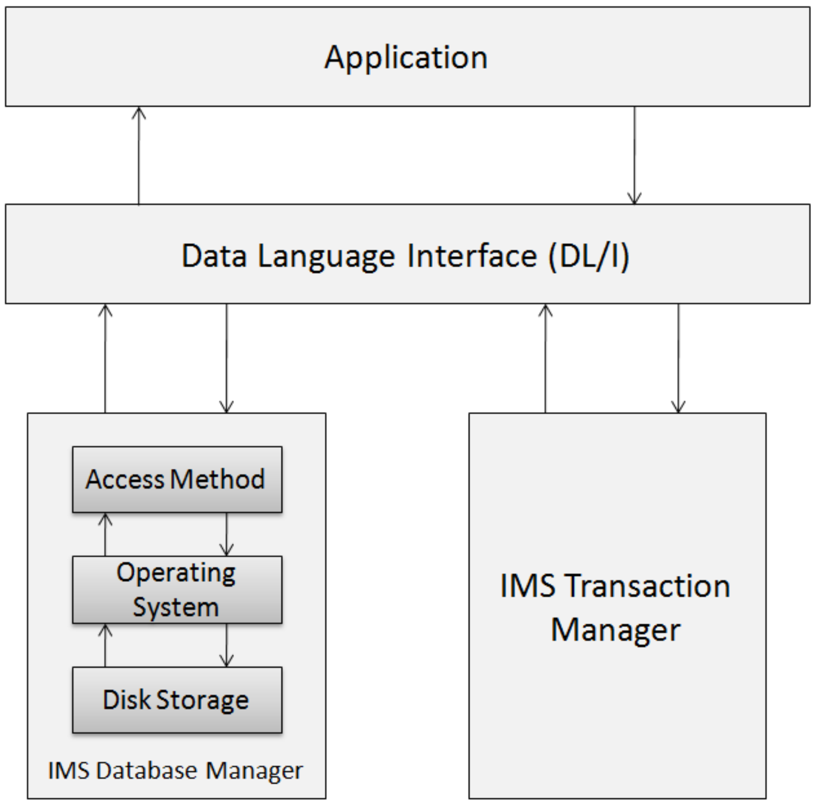
We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com.

Table of Contents

1. IMS DB – Overview

A Brief Overview

Database is a collection of correlated data items. These data items are organized and stored in a manner to provide fast and easy access. IMS database is a hierarchical database where data is stored at different levels and each entity is dependent on higher level entities. The physical elements on an application system that use IMS are shown in the following figure.



Database Management

A Database Management system is a set of application programs used for storing, accessing, and managing data in the database. IMS database management system maintains integrity and allows fast recovery of data by organizing it in such a way that it is easy to retrieve. IMS maintains a large amount of world's corporate data with the help of its database management system.

Transaction Manager

The function of transaction manager is to provide a communication platform between the database and the application programs. IMS acts as a transaction manager. A transaction manager deals with the end-user to store and retrieve data from the database. IMS can use IMS DB or DB2 as its back-end database to store the data.

DL/I – Data Language Interface

DL/I comprises of application programs that grant access to the data stored in the database. IMS DB uses DL/I which serves as the interface language that programmers use for accessing the database in an application program. We will discuss this in more detail in the upcoming chapters.

Characteristics of IMS

Points to note:

- IMS supports applications from different languages such as Java and XML.
- IMS applications and data can be accessed over any platform.
- IMS DB processing is very fast as compared to DB2.

Limitations of IMS

Points to note:

- Implementation of IMS DB is very complex.
- IMS predefined tree structure reduces flexibility.
- IMS DB is difficult to manage.

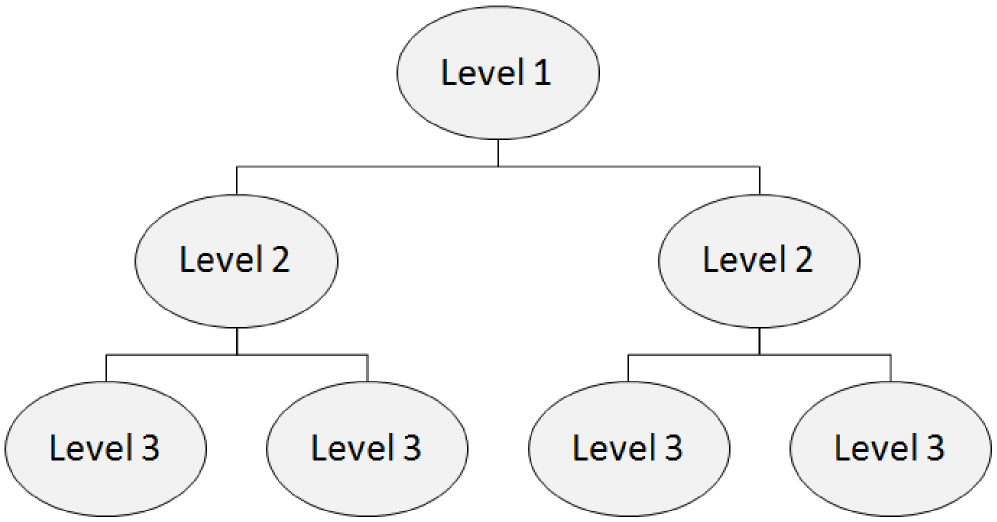
tutorialspoint
SIMPLYEASYLEARNING

2. IMS DB – Structure

Hierarchical Structure

An IMS database is a ccollection of data accommodating physical files. In a hierarchical database, the topmost level contains the general information about the entity. As we proceed from the top level to the bottom levels in the hierarchy, we get more and more information about the entity.

Each level in the hierarchy contains segments. In standard files, it is difficult to implement hierarchies but DL/I supports hierarchies. The following figure depicts the structure of IMS DB.

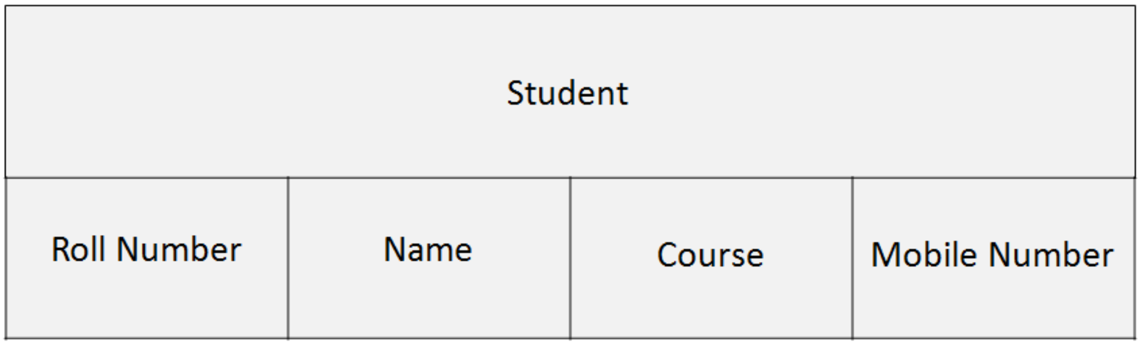


Segment

Points to note:

- A segment is created by grouping of similar data together.
- It is the smallest unit of information that DL/I transfers to and from an application program during any input-output operation.
- A segment can have one or more data fields grouped together.

In the following example, the segment Student has four data fields.



Field

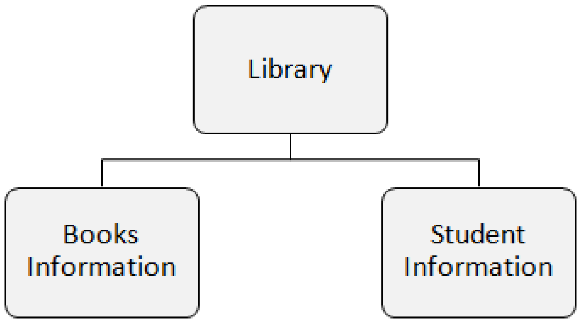
Points to note:

- A field is a single piece of data in a segment. For example, Roll Number, Name, Course, and Mobile Number are single fields in the Student segment.
- A segment consists of related fields to collect the information of an entity.
- Fields can be used as a key for ordering the segments.
- Fields can be used as a qualifier for searching information about a particular segment.

Segment Type

Points to note:

- Segment Type is a category of data in a segment.
- A DL/I database can have 255 different segment types and 15 levels of hierarchy.
- In the following figure, there are three segments namely, Library, Books Information, and Student Information.



Segment Occurrence

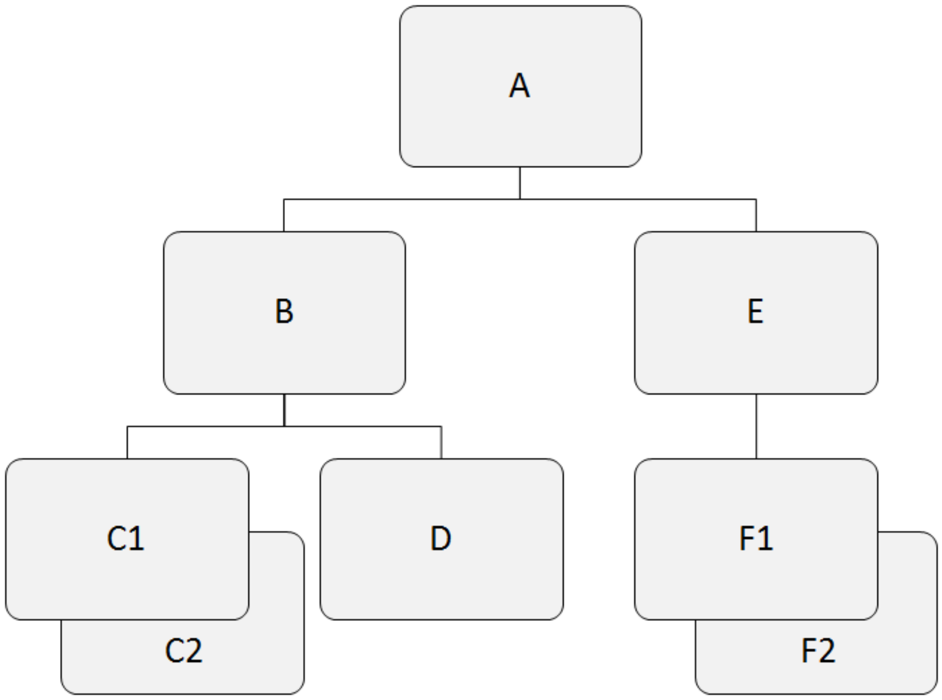
Points to note:

- A segment occurrence is an individual segment of a particular type containing user data. In the above example, Books Information is one segment type and there can any number of occurrences of it, as it can store the information about any number of books.

- Within the IMS Database, there is only one occurrence of each segment type, but there can be an unlimited number of occurrences of each segment type.

3. IMS DB – DL/I Terminology

Hierarchical databases work on the relationships between two or more segments. The following example shows how segments are related to each other in the IMS database structure.



Root Segment

Points to note:

- The segment that lies at the top of the hierarchy is called the root segment.
- The root segment is the only segment through which all dependent segments are accessed.
- The root segment is the only segment in the database which is never a child segment.
- There can be only one root segment in the IMS database structure.
- For example, **'A'** is the root segment in the above example.

Parent Segment

Points to note:

- A parent segment has one or more dependent segments directly below it.
- For example, **'A'**, **'B'**, and **'E'** are the parent segments in the above example.

Dependent Segment

Points to note:

- All segments other than the root segment are known as dependent segments.
- Dependent segments depend on one or more segments to present complete meaning.
- For example, **'B'**, **'C1'**, **'C2'**, **'D'**, **'E'**, **'F1'** and **'F2'** are dependent segments in our example.

Child Segment

Points to note:

- Any segment having a segment directly above it in the hierarchy is known as a child segment.
- Each dependent segment in the structure is a child segment.
- For example, **'B'**, **'C1'**, **'C2'**, **'D'**, **'E'**, **'F1'** and **'F2'** are child segments.

Twin Segments

Points to note:

- Two or more segment occurrences of a particular segment type under a single parent segment are called twin segments.
- For example, **'C1'** and **'C2'** are twin segments, so do **'F1'** and **'F2'** are.

Sibling Segment

Points to note:

- Sibling segments are the segments of different types and the same parent.
- For example, **'B'** and **'E'** are sibling segments. Similarly, **'C1'**, **'C2'**, and **'D'** are sibling segments.

Database Record

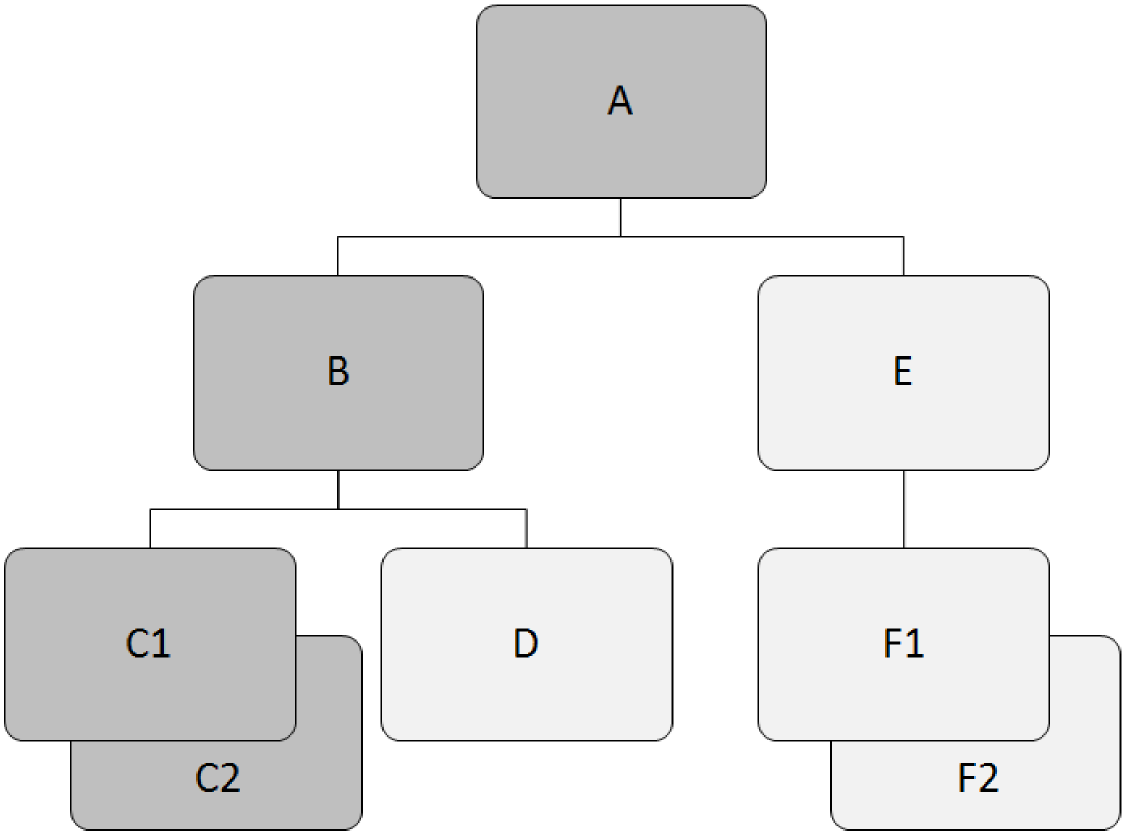
Points to note:

- Each occurrence of the root segment, plus all the subordinate segment occurrences make one database record.
- Every database record has only one root segment but it may have any number of segment occurrences.
- In standard file processing, a record is a unit of data that an application program uses for certain operations. In DL/I, that unit of data is known as a segment. A single database record has many segment occurrences.

Database Path

Points to note:

- A path is the series of segments that starts from the root segment of a database record to any specific segment occurrence.
- A path in the hierarchy structure need not be complete to the lowest level. It depends on how much information we require about an entity.
- A path must be continuous and we cannot skip intermediate levels in the structure.
- In the following figure, the child records in dark grey color show a path which starts from **'A'** and goes through **'C2'**.


A
B
E
C1
C2
D
F1
F2

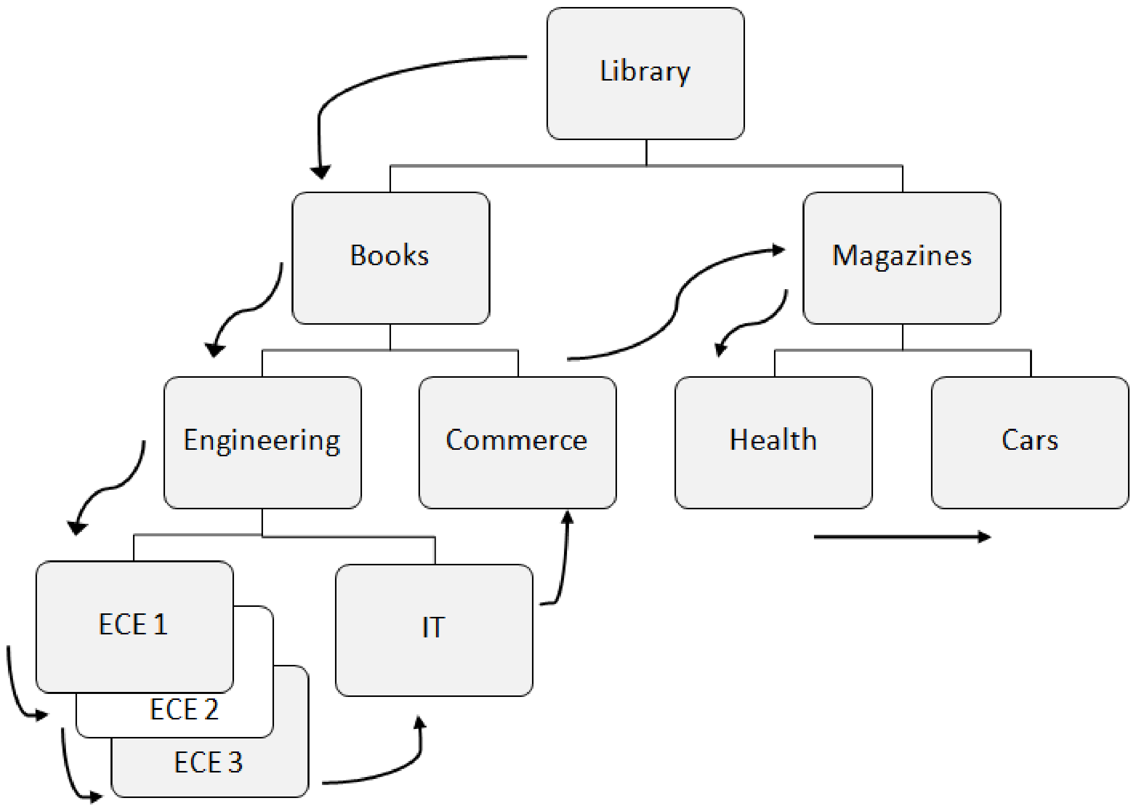
4. IMS DB – DL/I Processing

IMS DB stores data at different levels. Data is retrieved and inserted by issuing DL/I calls from an application program. We will discuss about DL/I calls in detail in the upcoming chapters. Data can be processed in the following two ways:

- Sequential Processing
- Random Processing

Sequential Processing

When segments are retrieved sequentially from the database, DL/I follows a predefined pattern. Let us understand the sequential processing of IMS-DB.

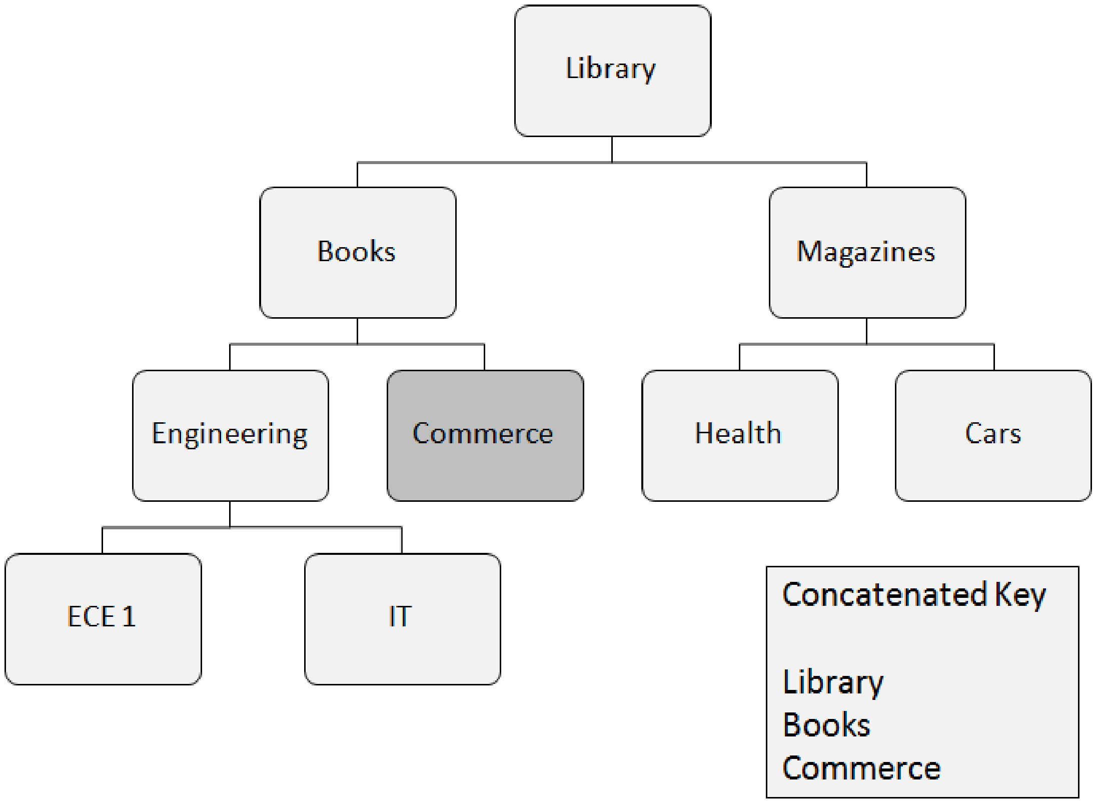


Listed below are the points to note about sequential processing:

- Predefined pattern for accessing data in DL/I is first down the hierarchy, then left to right.
- The root segment is retrieved first, then DL/I moves to the first left child and it goes down till the lowest level. At the lowest level, it retrieves all the occurrences of twin segments. Then it goes to the right segment.
- To understand better, observe the arrows in the above figure that show the flow for accessing the segments. Library is the root segment and the flow starts from there and goes till cars to access a single record. The same process is repeated for all occurrences to get all the data records.
- While accessing data, the program uses the **position** in the database which helps to retrieve and insert segments.

Random Processing

Random processing is also known as direct processing of data in IMS-DB. Let us take an example to understand random processing in IMS-DB:



Listed below are the points to note about random processing:

- Segment occurrence that needs to be retrieved randomly requires key fields of all the segments it depends upon. These key fields are supplied by the application program.
- A concatenated key completely identifies the path from the root segment to the segment which you want to retrieve.
- Suppose you want to retrieve an occurrence of the Commerce segment, then you need to supply the concatenated key field values of the segments it depends upon, such as Library, Books, and Commerce.
- Random processing is faster than sequential processing. In real-world scenario, the applications combine both sequential and random processing methods together to achieve best results.

Key Field

Points to note:

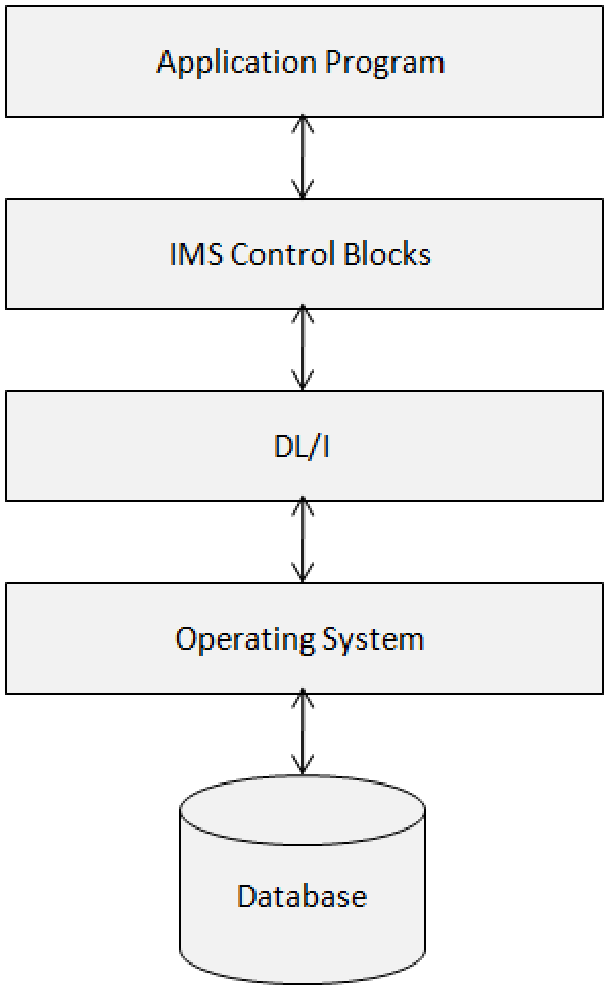
- A key field is also known as a sequence field.
- A key field is present within a segment and it is used to retrieve the segment occurrence.
- A key field manages the segment occurrence in ascending order.
- In each segment, only a single field can be used as a key field or sequence field.

Search Field

As mentioned, only a single field can be used as a key field. If you want to search for the contents of other segment fields which are not key fields, then the field which is used to retrieve the data is known as a search field.

5. IMS DB – Control Blocks

IMS Control Blocks define the structure of the IMS database and a program's access to them. The following diagram shows the structure of IMS control blocks.



DL/I uses the following three types of Control Blocks:

- Database Descriptor (DBD)
- Program Specification Block (PSB)

- Access Control Block (ACB)

Database Descriptor (DBD)

Points to note:

- DBD describes the complete physical structure of the database once all the segments have been defined.
- While installing a DL/I database, one DBD must be created as it is required to access the IMS database.
- Applications can use different views of the DBD. They are called Application Data Structures and they are specified in the Program Specification Block.
- The Database Administrator creates a DBD by coding **DBDGEN** control statements.

DBDGEN

DBDGEN is a Database Descriptor Generator. Creating control blocks is the responsibility of the Database Administrator. All the load modules are stored in the IMS library. Assembly Language macro statements are used to create control blocks. Given below is a sample code that shows how to create a DBD using DBDGEN control statements:

```
PRINT    NOGEN

DBD      NAME=LIBRARY,ACCESS=HIDAM

DATASET  DD1=LIB,DEVICE=3380

SEGM     NAME=LIBSEG,PARENT=0,BYTES=10

FIELD    NAME=(LIBRARY,SEQ,U),BYTES=10,START=1,TYPE=C

SEGM     NAME=BOOKSEG,PARENT=LIBSEG,BYTES=5

FIELD    NAME=(BOOKS,SEQ,U),BYTES=10,START=1,TYPE=C

SEGM     NAME=MAGSEG,PARENT=LIBSEG,BYTES=9

FIELD    NAME=(MAGZINES,SEQ),BYTES=8,START=1,TYPE=C

DBDGEN

FINISH

END
```

Let us understand the terms used in the above DBDGEN:

- When you execute the above control statements in **JCL**, it creates a physical structure where LIBRARY is the root segment, and BOOKS and MAGZINES are its child segments.
- The first DBD macro statement identifies the database. Here, we need to mention the NAME and ACCESS which is used by DL/I to access this database.
- The second DATASET macro statement identifies the file that contains the database.
- The segment types are defined using the SEGM macro statement. We need to specify the PARENT of that segment. If it is a Root segment, then mention PARENT=0.

The following table shows parameters used in FIELD macro statement:

Parameters	**Description**
Name	Name of the field, typically 1 to 8 characters long
Bytes	Length of the field
Start	Position of field within segment
Type	Data type of the field
Type C	Character data type
Type P	Packed decimal data type
Type Z	Zoned decimal data type
Type X	Hexadecimal data type
Type H	Half word binary data type
Type F	Full word binary data type

Program Specification Block (PSB)

The fundamentals of PSB are as given below:

- A database has a single physical structure defined by a DBD but the application programs that process it can have different views of the database. These views are called application data structure and are defined in the PSB.
- No program can use more than one PSB in a single execution.
- Application programs have their own PSB and it is common for application programs that have similar database processing requirements to share a PSB.
- PSB consists of one or more control blocks called Program Communication Blocks (PCBs). The PSB contains one PCB for each DL/I database the application program will access. We will discuss more about PCBs in the upcoming modules.

- PSBGEN must be performed to create a PSB for the program.

PSBGEN

PSBGEN is known as Program Specification Block Generator. The following example creates a PSB using PSBGEN:

```
PRINT   NOGEN

PCB     TYPE=DB,DBDNAME=LIBRARY,KEYLEN=10,PROCOPT=LS

SENSEG  NAME=LIBSEG

SENSEG  NAME=BOOKSEG,PARENT=LIBSEG

SENSEG  NAME=MAGSEG,PARENT=LIBSEG

PSBGEN  PSBNAME=LIBPSB,LANG=COBOL

END
```

Let us understand the terms used in the above DBDGEN:

- The first macro statement is the Program Communication Block (PCB) that describes the database Type, Name, Key-Length, and Processing Option.
- DBDNAME parameter on the PCB macro specifies the name of the DBD. KEYLEN specifies the length of the longest concatenated key. The program can process in the database. PROCOPT parameter specifies the program's processing options. For example, LS means only LOAD Operations.
- SENSEG is known as Segment Level Sensitivity. It defines the program's access to parts of the database and it is identified at the segment level. The program has access to all the fields within the segments to which it is sensitive. A program can also have field-level sensitivity. In this, we define a segment name and the parent name of the segment.
- The last macro statement is PCBGEN. PSBGEN is the last statement telling there are no more statements to process. PSBNAME defines the name given to the output PSB module. The LANG parameter specifies the language in which the application program is written, e.g., COBOL.

Access Control Block (ACB)

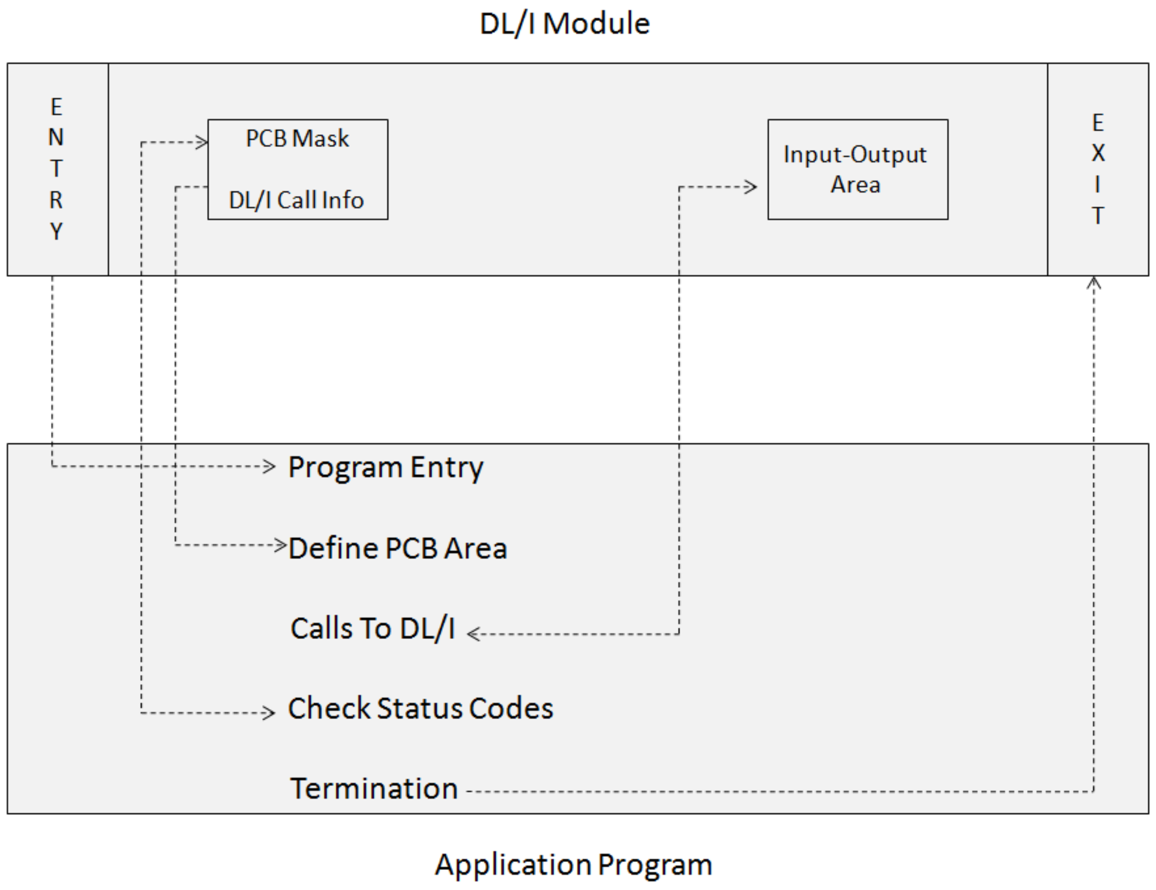
Listed below are the points to note about access control blocks:

- Access Control Blocks for an application program combines the Database Descriptor and the Program Specification Block into an executable form.
- ACBGEN is known as Access Control Blocks Generator. It is used to generate ACBs.

- For online programs, we need to pre-build ACBs. Hence the ACBGEN utility is executed before executing the application program.
- For batch programs, ACBs can be generated at execution time too.

6. IMS DB – Programming

An application program which includes DL/I calls cannot execute directly. Instead, a JCL is required to trigger the IMS DL/I batch module. The batch initialization module in IMS is DFSRRC00. The application program and the DL/I module execute together. The following diagram shows the structure of an application program which includes DL/I calls to access a database.



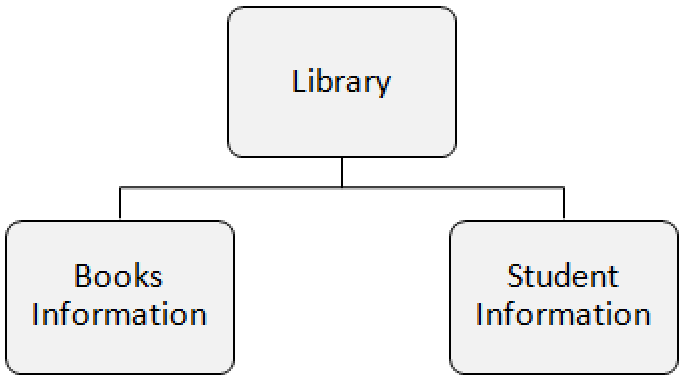
The application program interfaces with IMS DL/I modules via the following program elements:

- An ENTRY statement specifies that the PCBs are utilized by the program.
- A PCB-mask co-relates with the information preserved in the pre-constructed PCB which receives return information from the IMS.

- An Input-Output Area is used for passing data segments to and from the IMS database.
- Calls to DL/I specify the processing functions such as fetch, insert, delete, replace, etc.
- Check Status Codes is used to check the SQL return code of the processing option specified to inform whether the operation was successful or not.
- A Terminate statement is used to end the processing of the application program which includes the DL/I.

Segments Layout

As of now, we learnt that the IMS consists of segments which are used in high-level programming languages to access data. Consider the following IMS database structure of a Library which we have seen earlier and here we see the layout of its segments in COBOL:



```
01 LIBRARY-SEGMENT.
         05 BOOK-ID         PIC X(5).
         05 ISSUE-DATE      PIC X(10).
         05 RETURN-DATE     PIC X(10).
         05 STUDENT-ID      PIC A(25).

01 BOOK-SEGMENT.
         05 BOOK-ID         PIC X(5).
         05 BOOK-NAME       PIC A(30).
         05 AUTHOR         PIC A(25).


01 STUDENT-SEGMENT.
         05 STUDENT-ID     PIC X(5).
         05 STUDENT-NAME   PIC A(25).
         05 DIVISION       PIC X(10).
```

Application Program Overview

The structure of an IMS application program is different from that of a Non-IMS application program. An IMS program cannot be executed directly; rather it is always called as a subroutine. An IMS application program consists of Program Specification Blocks to provide a view of the IMS database.

The application program and the PSBs linked to that program are loaded when we execute an application program which includes IMS DL/I modules. Then the CALL requests triggered by the application programs are executed by the IMS module.

IMS Services

The following IMS services are used by the application program:

- Accessing database records
- Issuing IMS commands
- Issuing IMS service calls
- Checkpoint calls
- Sync calls
- Sending or receiving messages from online user terminals

End of ebook preview
If you liked what you saw…
Buy it from our store @ **https://store.tutorialspoint.com**