IMS/DB Introduction and Structure

Introduction

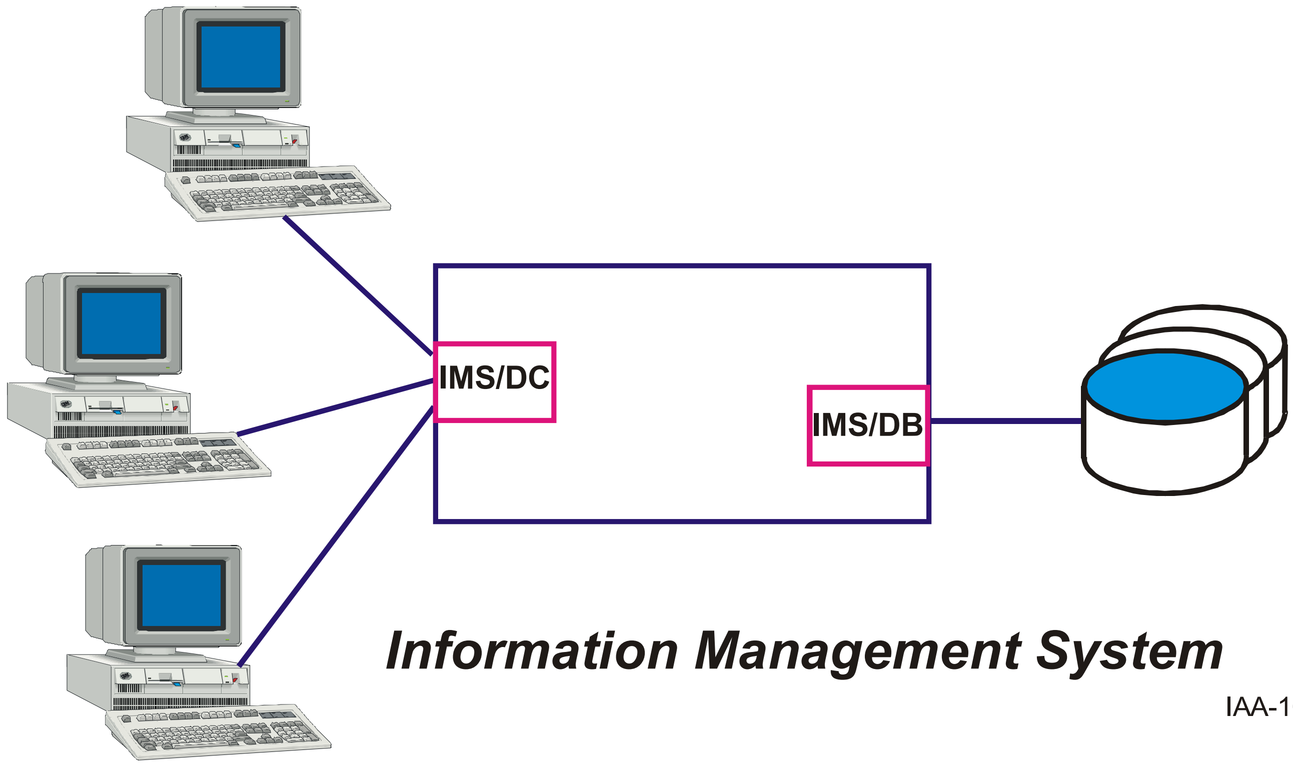
IMS is IBM's Information Management System, which provides a hierarchical database management system for large MVS installations. There are two main components:

• IMS/DB	manages the data (Data Base)
• IMS/DC	provides a transaction processing system (Data Communications)

IMS/DC is now known as IMS/TM (Transaction Manager). Most MVS installations use CICS instead of IMS/DC. This section is concerned with the way in which IMS/DB is structured and controlled.

A database, in its simplest form is an organised collection of information. The IMS data is accessed via a language called DL/1 (Data Language/1); a program will issue a 'call' to DL/1 when it performs an activity against the database. (DL/1 is also the name of IBM's hierarchical database management system for users of the VSE operating system.)

There are few differences in using IMS from the operations point of view when compared with a batch system; these are mainly in the area of data base recovery.



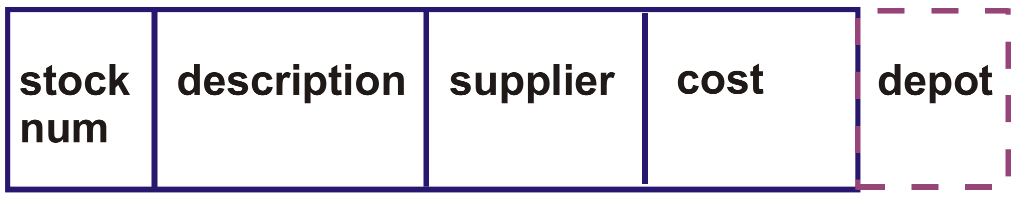
Information Management System

IAA-10

Before databases

Before databases became available, installations stored their data in conventional files such as VSAM datasets or less flexible (and older) QSAM or ISAM datasets.

One drawback with VSAM was the use of alternate indexes to access the data in different ways. Also, if an extra field needed to be added to the record, all programs which previously accessed the file would need to be changed to reflect the new record structure.

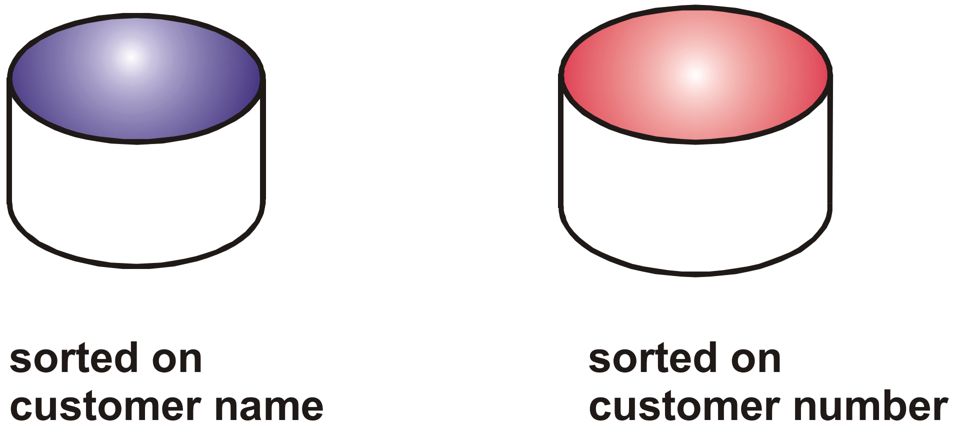


IIA-20

Fields in records of conventional files (e.g. VSAM)

Databases such as IMS were introduced to overcome the problems which many installations encountered with VSAM datasets. These problems included lack of flexibility of access to data.

To illustrate this, we can consider the problems faced by a site which has a large VSAM file which is keyed on the *customer number*. If the data in the file needs to be accessed also by a key based on the *customer name*, then several problems arise. One way of supplying the data in the customer name format would be to create a second version of the file.



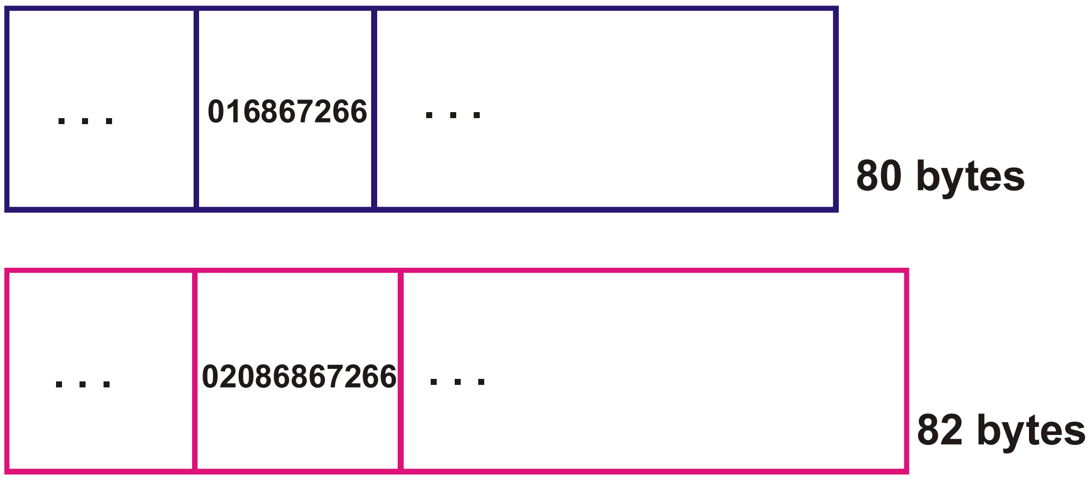
IIA-25

This leads to the following problems:

- the data would be duplicated leading to extra use of disk space.
- when a new record is added to the file, both copies must be updated. This leads to extra processing, and big problems occur if one of the files becomes out-of-step.
- there may be more than one customer with a particular name (e.g. Smith). VSAM keys must be unique.

A possible solution to these problems would be to use VSAM alternate indexes. However, they require the alternate keys to be contiguous parts of the record, so it is difficult to design files so that composite keys can be used easily.

A further problem with alternate indexes and VSAM files is that the record structure cannot be easily changed. This means that if an extra field was needed or a field needed to be extended in a particular record (perhaps a longer telephone number), all the programs which use that file would have to be modified - even if they did not use the additional information. This consumes a great deal of time and effort.



IIA-30

Ordinary files also have problems with confidential information. A program accessing a conventional file will be able to access the entire record - including details such as salary.

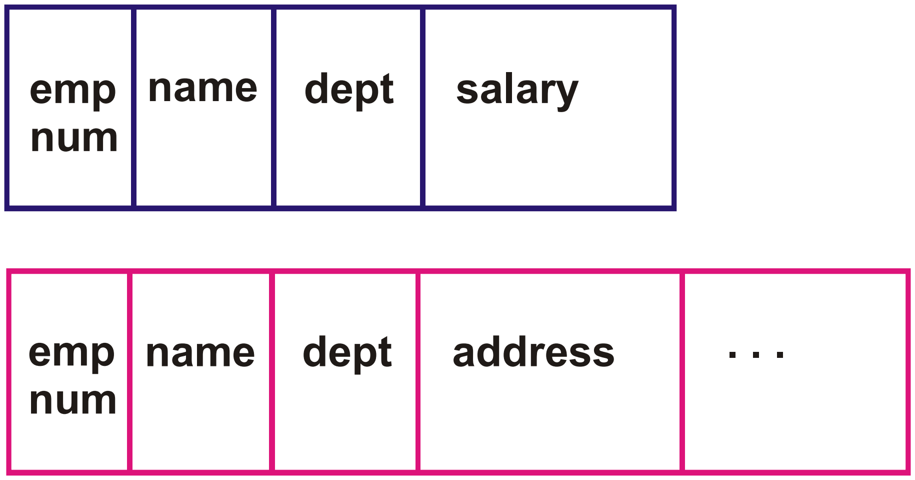
By using a data base management system such as IMS, these problems are overcome. This is because programs do not normally access all the parts of a record. Programs are given a *view* of the record; data fields which are not in the view are inaccessible to the program.

Database Requirements

- **Duplication of data** must be eliminated.
- All of the information should be held under a central control.
- Any item of data should be held only once.

One of the main reasons that data becomes duplicated is that different users want the information in different formats suitable to their systems. A example of this could be in an accounting and billing system. Some users would want to access the customer master file both by **account number** and by **customer name**. This means that the file must be sorted and a new file created.

Another cause of data duplication comes from the need to protect confidential information, for example salary details or medical records. Using conventional file organisations, there is no way of restricting access to part of a record. This means that confidential information must be held separately. However, certain information used for sequencing the data must now be held on the new file. It is almost certain that this data will be duplicated elsewhere.



IIA-40

The data could be corrupted by a failing program. Great care must be taken to ensure its **integrity**. The data must be isolated from potential dangers. In particular, it must be made independent of the programs accessing it. All changes to the data must be logged so that if a problem occurs, it will be possible to perform recovery.

IMS Objectives

So far we have looked at general requirements for a database system. The objectives specific to IMS can be listed as follows:

- **Data independence**
  Data should not be tailored to any particular application. This enables you to design more applications which will access the same data in a database.

- **Reduce program maintenance**
  Existing programs should not need modification due to database expansion or modification.

- **Minimise duplicate data**
  Data from various applications should be integrated to avoid duplication.

- **Phased implementation**
  Databases must be flexible enough to allow for new data and new applications to be added without adversely affecting existing applications and data..

- **High Level Language support**
  Conventional commercial programming languages should be usable.

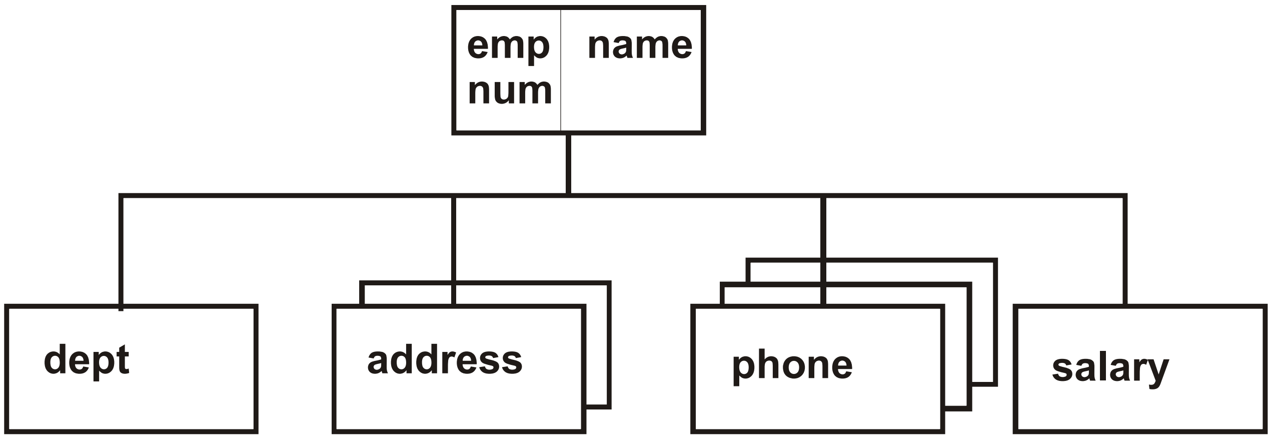
- **Reduce data maintenance**
  Data security and integrity should be maintained automatically e.g. by logging.
  Recovery and reorganisation utilities should be easy to use.

- **Data should be available online**
  CICS or IMS/DC provides the online access
  Data retrieval and update facilities should be available to batch and online programs.

- **Fast response with large data volumes**

IMS features

IMS data bases are organized in a **hierarchical structure**. Each record is made up of several **segments**. The segments can be duplicated and are related to others by a parent-child relationship.



IIA-60

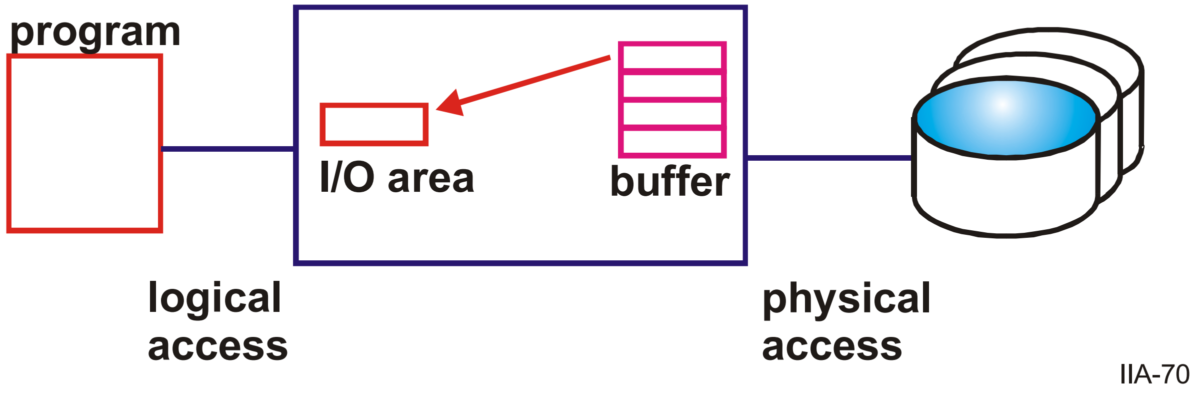
Segments are arranged in a hierarchical order

Extra segments can be added to the data base without causing problems for programs which do not use that information. The top segment is called the **root** segment.

The data base administrator can choose which of the segments will be visible to programs; only those marked as **sensitive** will be accessed by a program. Those segments which are to be protected or are irrelevant would therefore be deemed **non-sensitive** segments.

Hierarchical databases such as IMS allow access using multiple keys without the same overhead as VSAM.

IMS protects the program from having to consider the physical characteristics of the data storage. The program's I/O request is a logical access. The physical access is handled by IMS and is entirely independent of the program.



IMS provides data independence for programs

Converting from VSAM to IMS

In a conventional non-database system, applications often use several files containing items of **duplicate** data. Such duplication wastes disk space and also causes overheads when the duplicated items are to be updated.

Customer Master File

Account Number	Name	Address	Meter 1	Meter 2

Unpaid Bills File

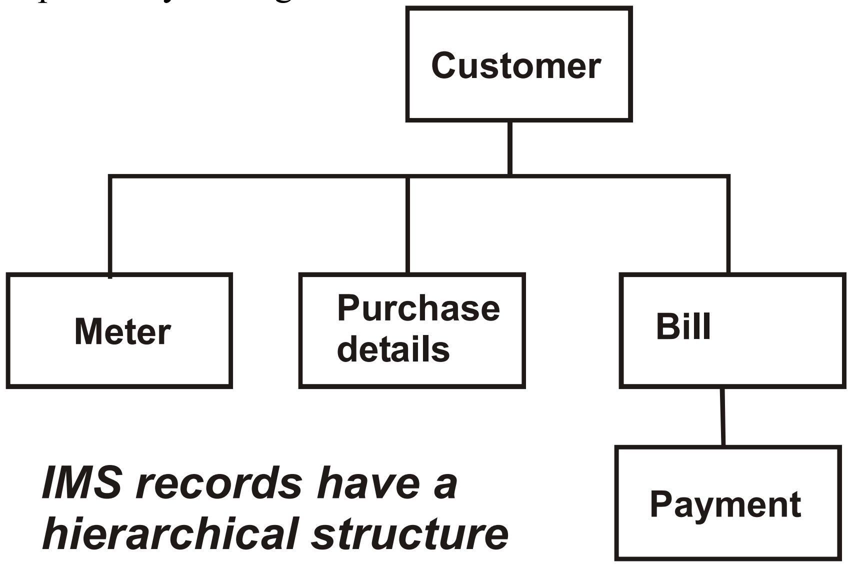
Account	Total Due	Name	Address

If we integrate these two files we can eliminate the duplication however, we must still reserve more space than we may need for the meters.

Account	Name	Address	Total Due	Meter 1	Meter 2

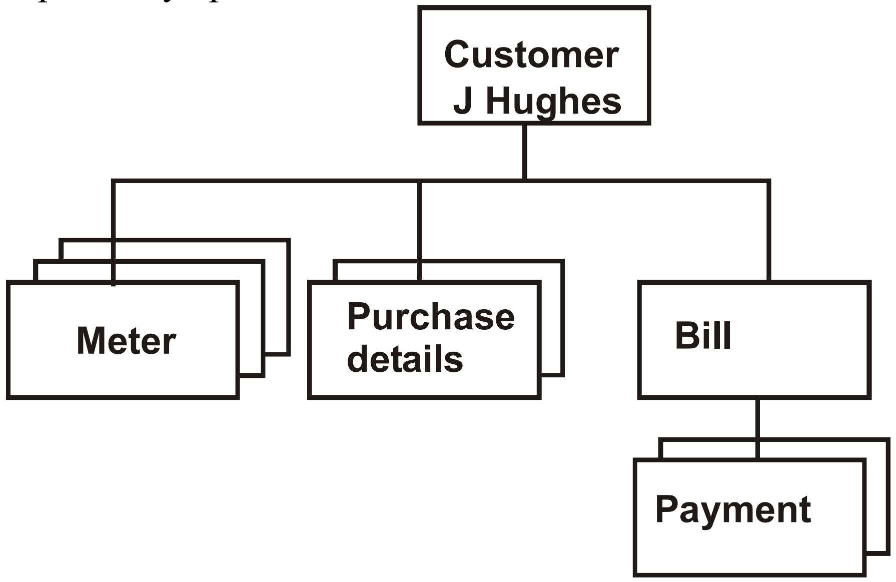
Wasted space will increase if we wish to write the contents of a **table** to a file. The record size must be big enough to contain the maximum number of elements in the table - even though only a few elements may contain valid data. The file designer may have reserved enough space for 12 monthly payments. If there are only three active payments, the space occupied by the other nine payments will be wasted.

IMS overcomes the space duplication problem by relating items of information according to dependency. In order to put a meter on file, we must first have a customer. Similarly, before recording a payment, the bill must already exist on file. We can represent this dependency in diagram form as follows:



IIA-100

This type of diagram is known as an **Hierarchical Structure**. Each box in the diagram is known as a **segment**, e.g. the Customer Segment. The hierarchical structure is a map of the database. It shows which segment types exist and their relationship or dependency upon each other.



IIA-110

This is the Database Record for customer J. Hughes. J. Hughes has three meters, two hire-purchase agreements and has made two payments to the company. We have only reserved enough space to hold this data, i.e. 3 meters, not 5.

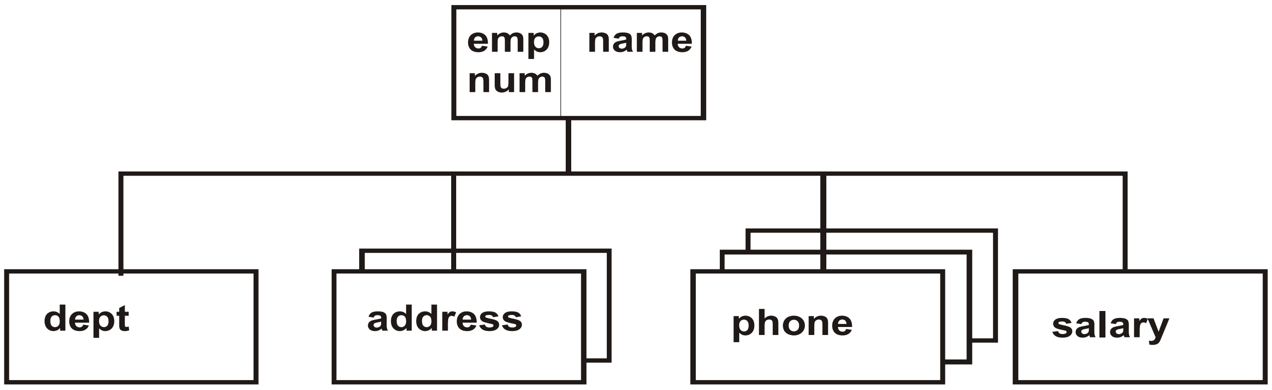
How is the database created?

The IMS data bases are set up and maintained by a **Data Base Administrator**. This person determines how the data can be processed by different programs, and also specifies which programs can access which parts of the data bases.

There are three activities which the data base administrator must perform before a program can access a IMS data base. These are to create a DBD, a PCB, and a PSB.

A **DBD** is a Data Base Description. This defines the contents and structure of the data base in terms of segments and their characteristics e.g. length of each segment; data type; key details (if any). The DBD is held in a library which is made available at execution time. The type of information supplied in a DBD includes:

- Database Name
- Segment names and lengths
- Segment dependency information
- Field names and lengths
- Access Method



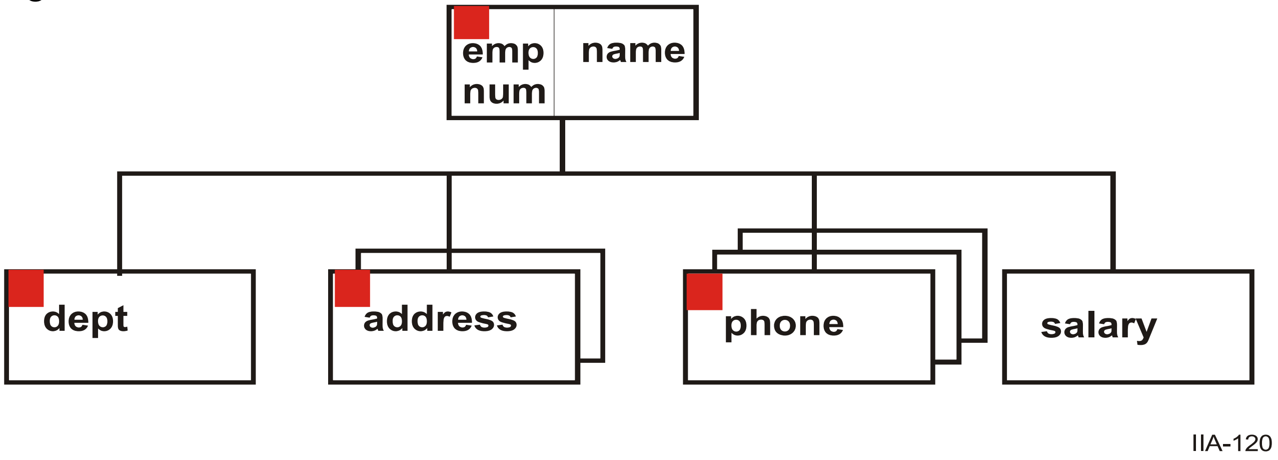
IIA-60

Segments are arranged in a hierarchical order

PSBs and PCBs

A program may not need to access all the segments in a DBD. This raises the question of how we can prevent a program or user from accessing the parts of a database record which contain irrelevant or inappropriate data. IMS allows us to protect against unauthorized access by allowing us to specify exactly which segments a program can access and the type of access that can be carried out i.e. Read Only, Update etc. The DBA will create a sub-view of the segments which a program may access, known as a **PCB** or Program Communication Block.

A PCB specifies which segments will be **sensitive** in a data base: in other words, which segments can be viewed by the program. Different programs may have different views of a given data base.



PCBs specify the segments which can be accessed

Segment sensitivity can be thought of as having three stages:

Non-sensitivity

An omitted segment all its dependent segments will be invisible to the program.

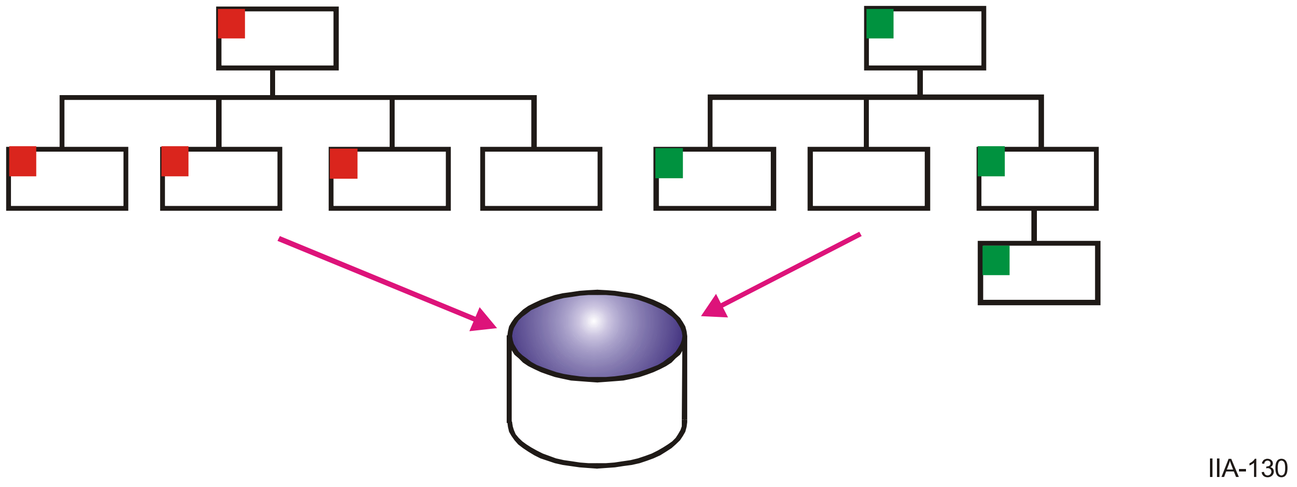
Data sensitivity

All of the data contained in the segment is accessible to the program.

Field sensitivity

This allows access to certain fields of a segment. For example, if a segment contains confidential information, we may still need to access its dependent segments. To do this, we define the segment as having Field Sensitivity for the *key only;* the data portion is protected

The PCB therefore contains details of all the segments in a database which a program will be able to access. If a program accesses two data bases, it will need two PCBs: one for each database. These PCBs will be are combined to form a PSB (Program Specification Block). The PSBs are held in a library which will be allocated at run time.



PCB details are used to form a PSB

If a new segment is added to a database, that database's DBD would be updated, but only the PSBs which need the new segment would have to be modified.

Each PCB provides the following information:

PCB statement giving	Database name Processing Options Concatenated Key Length
SENSEG statement giving	Name of the sensitive segment

There is one SENSEG statement for each segment defined. The SENSEG statements must be placed in hierarchical order.

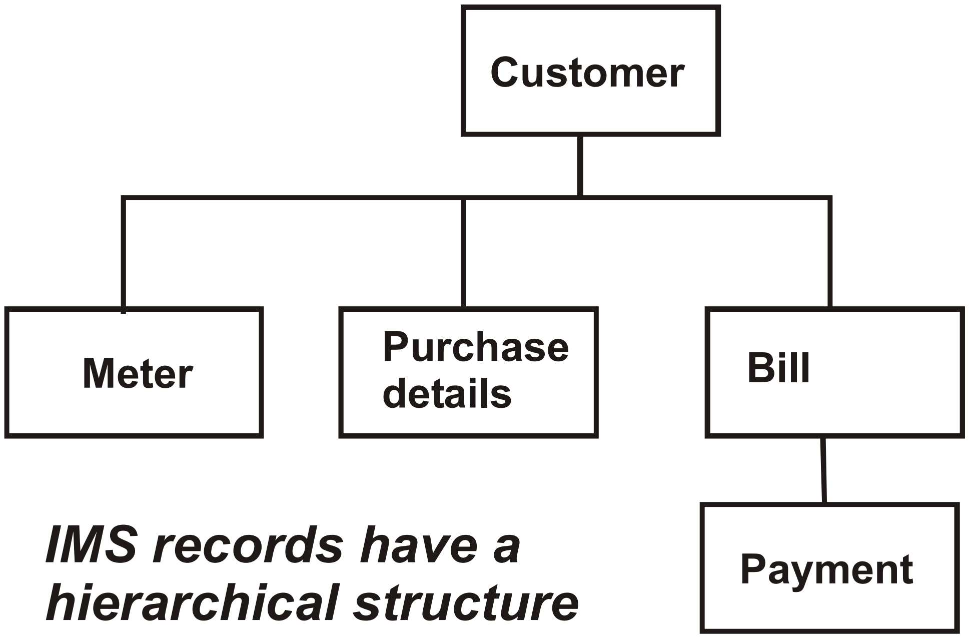
Therefore, for each IMS database, there will be:

- 1 DBD per database
- 1 PSB per application program
- 1 or more PCB per database used in the program

Database Structuring Rules

IMS is flexible in the ways it will allow us to organise our information; the structuring characteristics are:

1. no limit to the number of database records
2. each record may contain any number of segments within
3. there may be up to 255 different segment types
4. there is one root segment per database record
5. there may be any number of dependent segments per parent
6. a database record may have up to 15 hierarchical levels



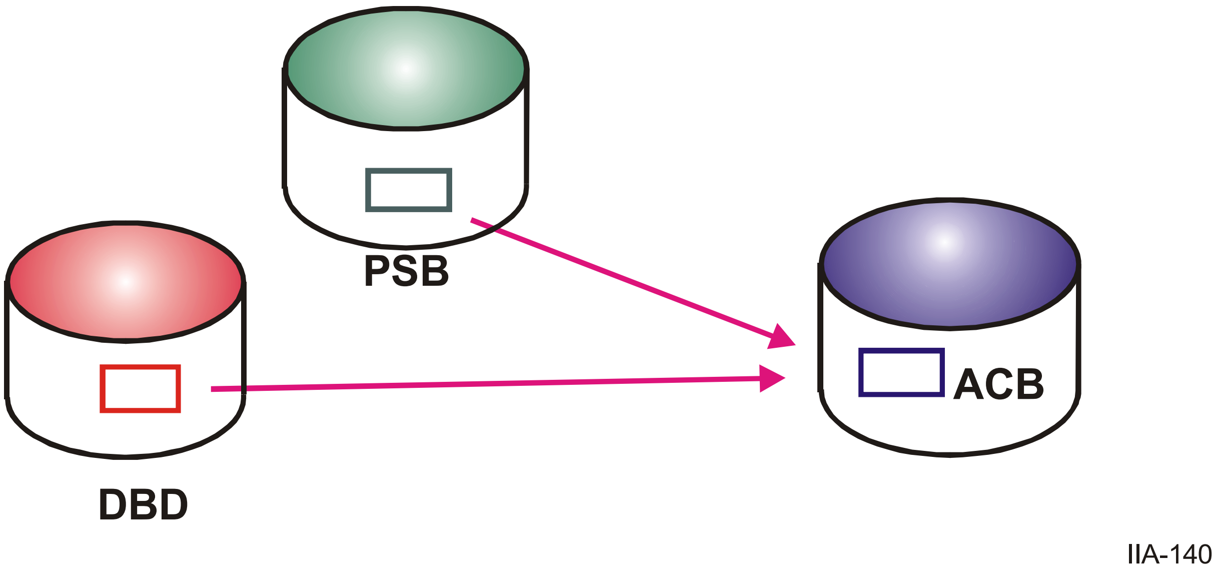
IMS records have a hierarchical structure

IIA-100

Application Control Blocks

The DBD and PSB information needs to be merged (into control blocks) before a program access a database. This process will take some time; this is not a problem for batch programs, but is an unwanted overhead for online execution.

In order to avoid this online overhead, the DBA will construct an ACBGEN. This builds a number of Application Control Blocks which must then be linked into the load library before the IMS program can be run.



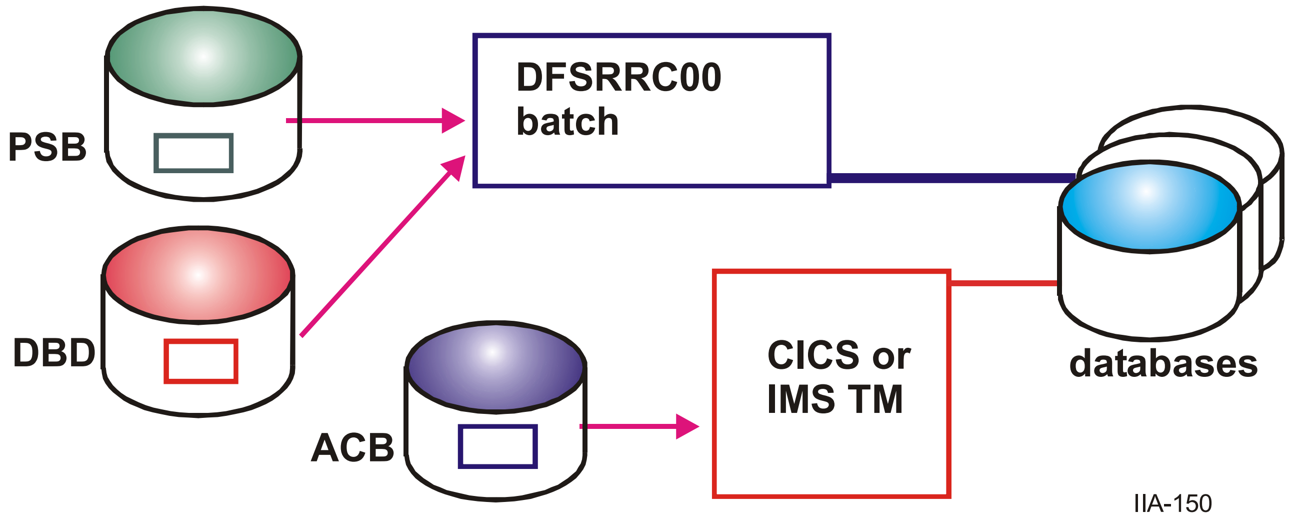
ACBs are used by online systems

Database definitions

Before your database program runs, several events must have taken place to created the environment in which the database can operate. These include:

• DBDGEN	define the database segments
• PSBGEN	define the PSB details
• ACBGEN	[needed for online]

- translate, compile, and link edit the application program
- execute DFSRRC00 for batch initialisation



Database definitions are accessed at run time

IMS Terminology

Segment
A unit of data. When we read or write data, we in fact read or write one or more segments.

Root
The segment at the top of the hierarchy on which all the other segment types depend e.g. Customer.

Database Record
All the segments which are related to one root segment.

Database
A collection of database records.

Parent
A segment which has a dependent segment e.g. Bill.

Child
A segment which is dependent on another segment e.g. Meter

Twin
More than one occurrence of the same segment type under a single occurrence of the parent e.g. Meter 1, Meter 2, etc.

Hierarchical Path
The series of segments, starting at the root which have to be passed in order to access a particular segment.

Hierarchical Order
The structure is always defined from Top-to-Bottom, Front-to-Back, Left-to-Right.

Hierarchical Level
All segments at the same level in the structure e.g. Meter and Bill are level 2.

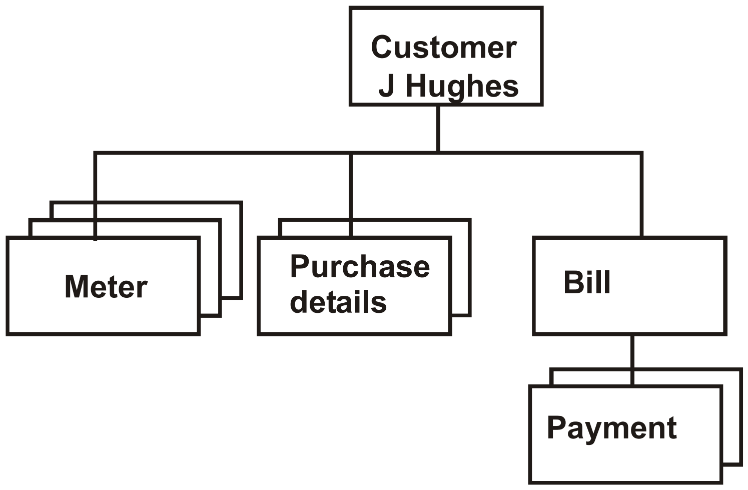
Search Field
A named part of a segment known to IMS. Any number of fields may be named and used in processing.

Key Field
The field used to determine the sequencing of twins under their parent. Multiple or unique key fields may be used.

Concatenated Key
The key field values of the segments in an hierarchical path, placed side by side.

Current Position
The starting position in the database; used by IMS to satisfy calls which are not directed elsewhere.

Parentage Position
An established position at a specified segment which is treated as a parent (even though it might be at a higher level).



IIA-110

DL/1 calls

The DL/I calls for database management are:

CLSE	GSAM Close	DEQ	Dequeue
DLET	Delete	FLD	Field
GHN	Get Hold Next	GHNP	Get Hold Next in Parent
GHU	Get Hold Unique	GN	Get Next
GNP	Get Next in Parent	GU	Get Unique
ISRT	Insert	OPEN	GSAM Open
POS	Position	REPL	Replace

The system service calls for DL/1 are:

APSB	Allocate PSB
CHKP	Basic Checkpoint
CHKP	Symbolic Checkpoint
CIMS	ODBA Function
DPSB	Deallocate PSB
GMSG	Get Message
GSCD	Get Address of System Contents Directory
ICMD	Issue Command
INIT	Initialize
INQY	Inquiry
LOG	Log
PCB	Specify and Schedule a PCB
RCMD	Retrieve Command
ROLB	Roll Back
ROLL	Roll
ROLS	Roll Back to SETS
SETS	Set a Backout Point
SETU	SET Unconditional
SNAP	Collects diagnostic information
STAT	Statistics
SYNC	Synchronization
TERM	Terminate
XRST	Extended Restart