GH20-1246-9
File No. S370/4300-50

Program Product

Data Language/I Disk Operating System/ Virtual Storage (DL/I DOS/VS) General Information

Program Number 5746-XX1

IBM

Tenth Edition (June 1983)

This edition, GH20-1246-9 applies to Data Language/I Disk Operating System/Virtual Storage (DL/I DOS/VS) Program Number 5746-XX1 and to all subsequent versions and modifications until otherwise indicated in new editions. It is a major revision of GH20-1246-8. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the *IBM System/370 and 4300 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

Changes or additions are indicated by a vertical line to the left of the change.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM program product in this publication is not intended to state or imply that only IBM's program product may be used. Any functionally equivalent program may be used instead.

Publications are not stocked at the addresses given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed either to:

IBM Corporation
Dept 812 BP
1133 Westchester Avenue
White Plains, NY, U.S.A. 10604

or to:

IBM Deutschland GmbH
Dept 3248
Schoenaicher Strasse 220
D-7030 Boeblingen, Federal Republic of Germany

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Preface

This publication presents a fundamental description of DL/I DOS/VS (Data Language/I Disk Operating System/Virtual Storage). It is intended primarily for persons interested in learning about DL/I DOS/VS at the introductory level so they may evaluate the applicability of DL/I DOS/VS to their installation. The various DL/I DOS/VS facilities are presented to help with that evaluation.

This publication is divided into six chapters.

- Chapter 1 provides an introduction to the data base concept, citing its advantages. Also discussed are the four functional elements that make up a data system: data base management, data communication, data administration, and data delivery.

- Chapter 2 presents DL/I highlights, covering data management, productivity, and the operating environments. Data management includes discussions on data redundancy, security, and recovery. Productivity discusses the impact on programmer productivity in relation to data and device independence.

- Chapter 3 presents the characteristics of DL/I, including hierarchical data structure, data organizations, and the access methods used. It also discusses the definition of data bases and the functions provided through DL/I utility programs.

- Chapter 4 provides sample applications for the manufacturing, financial, health, and process industries to indicate how DL/I might be put to use in those areas.

- Chapter 5 presents those responsibilities the user must assume to ensure a successful implementation of application programs with DL/I.

- Chapter 6 outlines the minimum configuration for executing DL/I. Two environments are presented: the batch environment and the online environment, which operates in conjunction with CICS/DOS/VS (Customer Information Control System/Disk Operating System/Virtual Storage).

A complete list of the DL/I publications can be found in the *DL/I DOS/VS Library Guide and Master Index*, GH24-5008. This manual is divided into two sections. The first section provides DL/I library information, including a description of each book, user tasks and user roles in a data base environment. The second section is a master index that consolidates the indexes of all the DL/I reference manuals to assist you in locating specific information.

References are made in this publication to CICS/DOS/VS. More information about CICS/DOS/VS can be found in the *Customer Information Control System/Disk Operating System/Virtual Storage (CICS/DOS/VS) General Information*, GC33-0066.

Reference is also made in this publication to three IBM licensed programs: the Data Base Design Aid, the Data Dictionary, and the Structured Query Language/Data System. More information about these programs can be found in:

- *Data Base Design Aid General Information*, GH20-1626
- *DB/DC Data Dictionary General Information*, GH20-9104
- *SQL/Data System General Information*, GH24-5012.

Contents

Chapter 1. Introduction

The growth of data processing systems generally parallels the growth of the corporate world. As the complexity of most corporations increases, the need for improved data handling becomes increasingly important. One solution to this growing problem of data handling is Data Language/I Disk Operating System/Virtual Storage (DL/I DOS/VS), a data base management system.

DL/I provides many features that facilitate implementation, change, and expansion of application and information files. DL/I also helps to reduce data processing costs by providing a capability to:

- Reduce application program maintenance.
- Reduce application programmer time required to implement new applications.
- Reduce the cost of converting to new direct access storage devices.
- Reduce the number of files in which data is repeated.

Some applications that have successfully been implemented using DL/I include:

- Payroll and personnel
- Manufacturing bill of material
- Inventory control
- Accounts receivable
- Hospital records
- Student records
- Petroleum well records
- Demand deposit accounts systems.

To better understand data bases, let's review the concept in more detail.

The Concept of Data Bases

The primary objective of a data base management system is to ensure the timely availability of current information. The effectiveness of a data base management system can be illustrated by comparing a non-data base environment with a data base environment.

In a non-data base environment, data files are usually designed to serve individual applications, such as inventory control, payroll, accounts receivable, purchasing, and so on. Each data file is specifically designed for its own application and stored separately on tape or disk. Quite often, the data files of different applications contain much of the same data. This duplication makes it difficult to keep the data consistent and current.

Furthermore, when modifying existing applications or adding new applications, your programming staff faces some of the following situations:

- Duplicate data exists on multiple files with different formats for different applications.
- Programmers spend a significant amount of time updating existing application programs to handle changes to the record layouts or the I/O device characteristics. Program function is usually not affected by these changes.
- Changing applications makes it desirable to move data files from one storage device to another (tape or disk), or to change from one access method (sequential) to another (direct).
- Programmer productivity is hindered by a limited knowledge of specific device characteristics or specific access methods.
- Batch applications have to be expanded to teleprocessing applications.

In short, every time a new application is planned, programmers must evaluate the impact of data changes on existing programs. As a result, application programs are often in a state of change, adding appreciably to the overall cost of data processing.

The solution is to:

1. Organize the data into a collection of interrelated data elements that can be processed by one or more applications. This collection is called a *data base*.
2. Provide a method for handling the data that can readily adapt to changes without impacting your present programs.

The most practical way to do this is to make the application program *independent* of the data it uses. The data base concept, as implemented by DL/I, provides this independence by removing the direct association between the application program and the physical storage of data (see Figure 1).

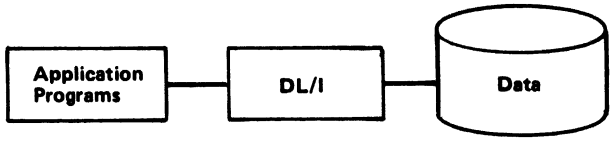


Figure 1. The Data Base Manager (DL/I)

A data base provides for the integration, sharing, and control of common data. As an example, a company may first integrate the data for a parts application with the data for a purchase order application (Figure 2). Later, data for order processing and accounts receivable may also be integrated.

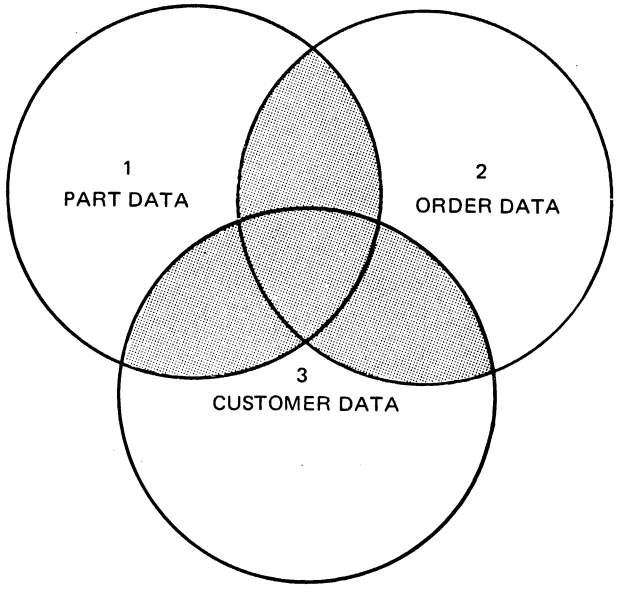


Figure 2. Application Data Integration — Data Base Concepts

Each application program views only the data that it uses, and accesses that data through the data base management system. For example, in Figure 3, assume that one application requires name and address information, and a second requires name and salary information. The applications share common data (employee name), but only the first accesses the address and only the second accesses the salary. To each application, data uniquely used by other applications does not exist.

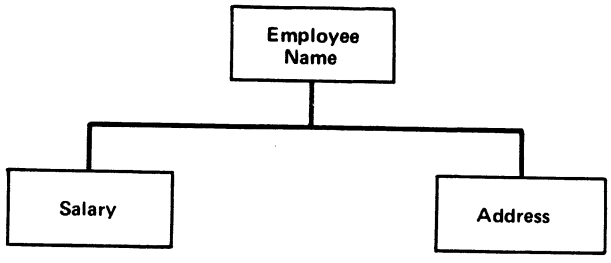
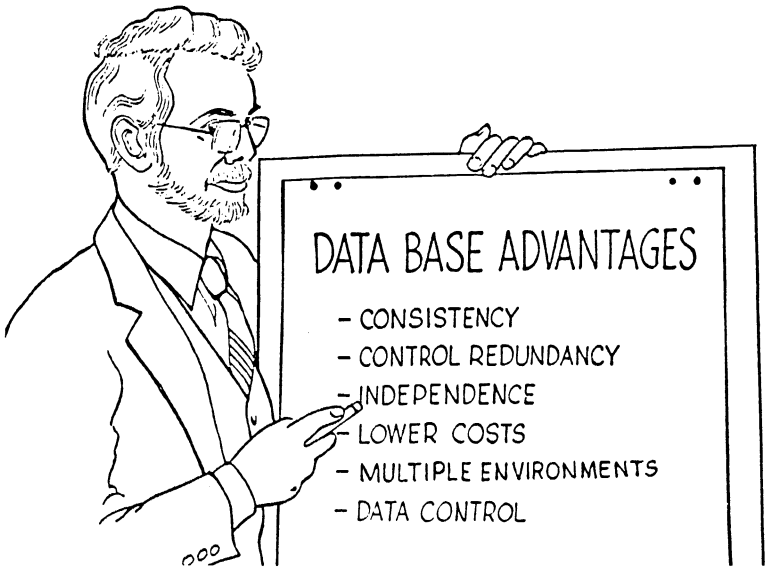


Figure 3. Hierarchical Data Base Concept

A data base provides flexibility of data organization. By removing the direct association between the application program and the physical storage of data, you can add data to an existing data base without modifying existing programs. This is called *data independence.*

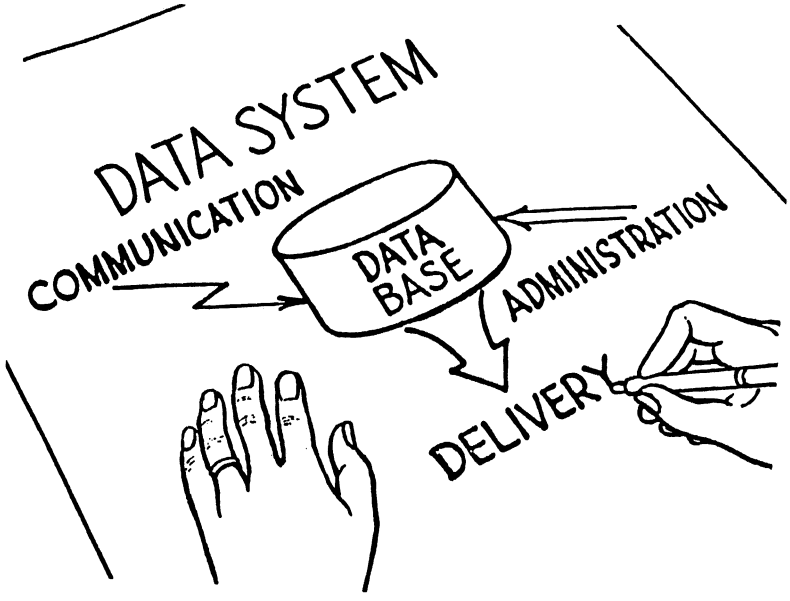
Advantages of Data Bases



The use of the data base concept can help provide several advantages summarized below.

- Consistency through the use of the same data by all groups in the company.
- Control of data redundancy and reduction of duplicate data maintenance.
- Application program independence from physical storage organizations and access methods.
- Help to reduce overall application costs.
- Data designs for multiple environments (batch and online processing).
- A focal point for the control of data.

The Data System Environment



Four functional elements make up the data system environment. These elements cover the management, the availability, the control, and the use of data for the day-to-day operation of your business and its future planning. The four elements are:

1. *Data base management* for the storage and retrieval of data.
2. *Data communication* for the efficient management of the terminal network functions that make data available to users on a timely and convenient basis.
3. *Data administration* for the control required to provide adequate protection and proper usage of all data.
4. *Data delivery* for the programming required to convert data into useful information for end users.

Data Base Management through DL/I

DL/I is a data base manager — a program that coordinates your programs and data.

By assuming the management of the physical environment (such as disks), DL/I eliminates the need for each application program to contain its own data descriptions. This can eliminate the multiple programming modifications ordinarily needed when changing data, device types, or the retrieval method.

DL/I also allows data, formerly maintained on separate files, to be integrated into a single data base. Thus, programs with slightly different data formats can share and update common data. (See Figure 4.)

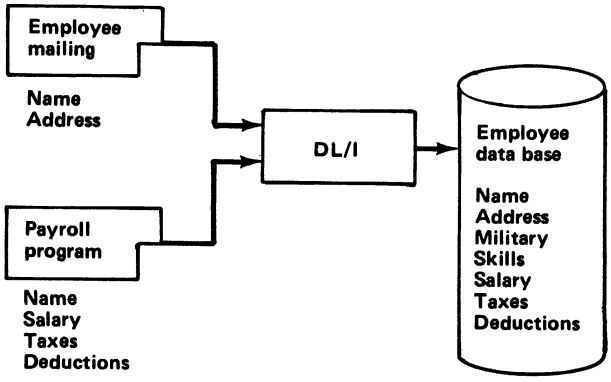


Figure 4. Integration of the Data Base

Each application program sees a logical picture of its data. Unique data, therefore, can be restricted to authorized programs to help improve security.

Some type of failure (whether it be human, system, or environmental) could occur to damage the data being maintained. DL/I, therefore, provides several mechanisms to assist in handling these data recovery problems.

Data Communications

The value of most data depends on its timeliness. A customer's order can be lost because an inventory system is slow in reporting the availability of a new shipment. Credit ratings, balances, and cash positions are only valuable for operations and decision making when they are current.

Effective data communications helps to improve the currency of data. A data communications manager (see Figure 5) can enhance programming productivity by providing network-handling functions that would otherwise have to be written for each online program. These functions provide access to current information by terminal users.

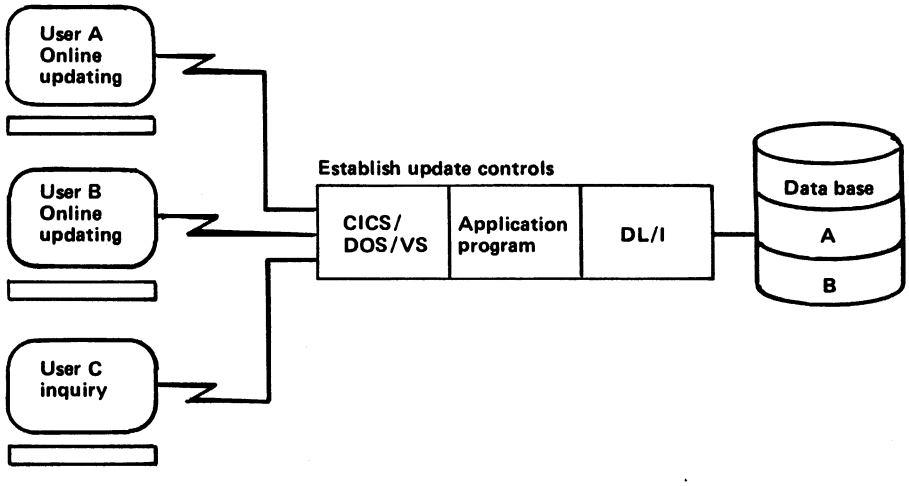


Figure 5. Data Communications Manager

Moreover, in a multiple-user environment, the data communications manager can help users to control data from their terminals.

The data communications manager can be effective when used with the data base manager to keep data consistent and to protect it from misuse. Together, these two software products can help reduce much of the maintenance otherwise required. Your time can then be applied to the development of new applications.

CICS/DOS/VS (Customer Information Control System/Disk Operating System/Virtual Storage) works with DL/I to provide data communications control.

Data Administration

Data administration is the key element of a data base environment and requires an understanding of such data attributes as content, description, and usage. In the course of developing this understanding, these questions could be raised about your data: What is it? Who accesses it...and which programs use it?

The centralization of data and control of access to this data is essential to a data base management system. One of the advantages of centralization is the availability of consistent data for more than one application. However, this requires a tighter control of that data and its usage.

Because the actual implementation of this function is largely dependent on your organization, this discussion is limited to the general characteristics of data administration. The responsibilities of data administration are to:

- Provide standards for, and control of, the administration of the data bases and their use.
- Guide, review, and approve the design of the data bases.
- Determine the rules of access to the data bases and monitor their security.
- Control data base integrity and availability, monitoring the necessary activities for reorganization, back-up, and recovery.
- Enforce procedures for accurate, complete, and timely updates to the data bases.
- Approve the operation of new programs with existing production data bases.

A tool that can be used to assist in performing the data administration function is a *data dictionary.* IBM provides a data dictionary: the *DOS/VS DB/DC Data Dictionary.*

A data dictionary allows you to establish a single authoritative source where all data definitions and cross-relationships are stored and controlled.

Once you have a data dictionary, it's easy to find out what data elements are used by each program and which users will be affected by a proposed change in a data base. The data dictionary can also identify and help eliminate existing naming inconsistencies. This, in turn, can simplify the use of established naming standards and reduce the need to rewrite definitions.

Data Delivery

Data delivery is concerned with processing and arranging the data into immediately usable information for end users. Traditionally this requires the creation of application programs by DP professionals in your installation.

Software products are available that can reduce the conventional program development effort. In addition, there are pre-written, ready-to-install IBM application offerings built around our data systems software. The range of existing software packages covers such application areas as personnel, health care, manufacturing, and finance.

Chapter 2. Highlights of DL/I

The value of DL/I can best be determined through an understanding of:

1. The way information in a data base is managed.
2. The effects on productivity in your installation.
3. The operating environments supported.

From this standpoint, let's review some of the highlights of DL/I and how it might benefit you.

Data Management

Reduced Data Redundancy

With DL/I, the data base can be expanded as your needs grow. The important benefit of this approach is that once a change is made to a data base record, it is effective for all programs using the data base. One data source for all programs can eliminate *data redundancy* and keep data consistent and current.

Data Integrity

DL/I provides the mechanisms necessary to help ensure the integrity of data so that users of a data base can access information that is current. This is accomplished through a locking procedure that prevents multiple users from updating the same data at the same time. The exclusive control of any piece of information in a data base thus ensures that accurate information is available to the next user.

DL/I also provides a logging facility that records all modifications to a data base. The log may be used in conjunction with the DL/I data base recovery utilities to rebuild the data base in the event a system failure occurs.

Data Security

With DL/I, access to confidential information in a data base can be restricted. If the data is not defined as accessible for a specific program, that program cannot access the data. This allows centralized control over who can access which portion of the data base. This concept is called *segment sensitivity*. Each application program is sensitive to only the segments it needs. (A segment, which is made up of one or more fields, is the basic unit of data in a DL/I data base that can be accessed by an application program.)

In addition to restricting access to a segment, access can be further restricted to fields within the segment. This enables different application programs to have different views of the same segment. This concept is called *field level sensitivity*.

The processing actions, as well as the data, of each application program can be restricted. Certain programs may update a data base while other programs can be restricted to data retrieval only.

Because DL/I allows your installation to control which programs can access data, and what processing they are permitted to do, confidential data can be protected.

Data Recovery

An effective data base management system must provide recovery procedures or programs to avoid damage to your data should some type of failure occur. These failures could include power blackouts, physical damage to equipment that make data unreadable, application programming errors, system malfunctions, and so on.

DL/I provides recovery facilities to help overcome these problems, including:

- A logging facility to capture all changes made to the data base.
- An abnormal termination routine to help minimize data base damage from a condition that prevents a normal end of job.
- A set of utility programs to correct errors or to reconstruct data bases after damage.
- A checkpoint and limited restart facility to minimize the time required to resume processing.

Productivity

Application Programmer Productivity

Your installation probably has an application that requires access to more than one file and a complex program to handle the interactions. With DL/I, you can expand the data base to allow a wider range of information processing to be accomplished. No matter how complex the data base may become, to application programmers it remains a simple, logical series of segments. This is because they are dealing with

only that portion of the full data base needed for the application. Very few application programmers see the full data base. They actually see only the portion they need for their particular application.

Programming with DL/I is standardized and simplified so application programmers no longer have to spend time describing the data and the environment to the operating system, as in non-data base environments.

Application Programming Languages

DL/I supports application programs written in:

- COBOL
- PL/I
- RPG II
- Assembler

Data and Device Independence

DL/I allows application programs to be independent of devices and data.

Data independence, for example, allows you to:

- add new types of information to existing data bases,
- optimize system performance by varying the record size, blocking factor, space allocation, and the access method, and
- change existing data forms

with no need to recompile the application programs.

DL/I provides *data independence* by separating the logic of the application programs from the organization of the data. If existing programs do not reference newly defined data, there is no need to recompile the programs.

For example, suppose you would like to develop a new system involving job incentives for your employees. In the non-data base environment, you must either merge the new information required into the existing employee master file (and force modification of all programs using it), or you must create another file. You already know the problems involved in expanding the file, so let's look at the alternative. You decide to construct a separate file for the job incentives system, using pertinent control information from the employee master file.

The problems arise, however, as changes, additions, and deletions occur. You now have two files to be updated. This creates the possibility of developing inconsistent data (one file is not updated while the other is).

With DL/I, you no longer need to create separate files. The file can be expanded as your needs grow. The important benefit of this approach is that once a change is made to a data base record, it is effective for all programs using the data base.

Device independence allows you to:

- change access methods,
- change devices within the disk family, and
- move data bases from tape to disk

with no need to recompile the application programs.

With DL/I, the physical characteristics of the data base are maintained separately from application programs. All programs are shielded from these changes and become *device independent.* All changes required for new device support are made through the use of DL/I utility programs and alterations to the descriptions of the data base.

Device independence can be illustrated with this example. A new storage device becomes available that has the capacity to greatly expand your capability to maintain data online. Since several applications have a need for additional space, you decide to convert to this new storage device.

In the non-data base environment, this conversion would require a modification, recompilation, and re-testing of every program using the data just to describe the new device and to change blocking factors.

In addition to changing the physical device, you can also change DL/I access methods by which data is retrieved for processing without changing the application programs. This change can be made in DL/I in the same way as changing a device; that is, by altering the description of the data base and using DL/I utility programs to unload and reload the file.

User Aids

Two important functions associated with DL/I and the data base concept are the design of the data base for effective and efficient use of information and the administration and control of that information. These IBM program products are available to assist you:

- Data Base Design Aid (DBDA), for the design process, and
- Data Dictionary, for data administration and control.

DB Design Aids

DOS/VS DBDA (Data Base Design Aid) - Program number 5746-XXQ

The DBDA is a collection of programs that assist in performing a major portion of the data base design process. The DBDA uses your installation's input that describes the data base requirements and produces a structural model of the data base by mapping the data elements into segments. It also produces a hierarchical structure that shows the minimum set of relationships required by an integrated data base (one which serves many application programs).

The DBDA assists the data base designer by:

1. Identifying inconsistencies or omissions in the input requirement specifications.
2. Listing redundant data.
3. Producing a series of diagnostic and design reports showing various errors and design alternatives.

More information about the IBM DBDA can be found in the *DBDA General Information Manual* listed in the Preface.

Data Dictionary

DOS/VS DB/DC Data Dictionary - Program Number 5746-XXC.

The concept of a data dictionary - a single consistent source of information about data in a data base management system - can be a valuable aid in controlling your data. Definitions in a data dictionary can include many different kinds of information such as:

- Characteristics about each item of data, its size, position, and content.
- Relationships that data may have to other data such as hierarchical locations, pointers, and indexes.
- "Where used" information about what programs and transactions use the data.
- "How used" information, indicating if the data can be changed, or merely accessed, and if there are accessing limitations.

The data dictionary can automatically accept existing DL/I definitions as well as COBOL program data definitions as input.

The data dictionary includes functions that allow you to:

- Generate standard reports about your data.
- Change data base structures.
- Generate new program data structures.

In addition, data dictionary capabilities enable production of:

- Standard documentation for program, data sets, segments, and data element definitions.
- Structured source data definition statements for inclusion in source code by programmers for COBOL, PL/I, or Assembler.
- Control tables used by DL/I.

Because the data dictionary enables data to be cross-referenced between programs, errors can be reduced, and there may be a reduction in time lost developing

applications that use conflicting data definitions. Recovery and restart procedures can be improved: if a program is discovered to generate bad data, the dictionary can help to determine what other data bases may have been affected.

In addition, the dictionary can include information about all files used by all programs in the installation. Because of this, it can be used as a DL/I conversion aid.

This centralized approach can also be beneficial when personnel changes occur. With the aid of data dictionary, new employees may become familiar with an application system more easily.

More information about the IBM Data Dictionary can be found in *Data Dictionary General Information Manual* listed in the Preface.

Operating Environments

DL/I supports three operating environments:

1. Batch
2. Online
3. MPS (Multiple Partition Support)

DL/I Batch Processing

The DL/I system operating in a batch environment controls a single application program's access to a data base(s). Through DL/I, the application program is able to read, add, delete, or replace information in a data base. In this environment, the DL/I system and the application program execute within the same batch partition.

DL/I Online Processing

The DL/I system operating in an online environment includes all the functions of a batch system. In addition multiple applications are allowed concurrent access to the data base(s). An additional set of service routines to support this capability are provided. In this environment, the DL/I system executes within the CICS/DOS/VS partition and uses CICS/DOS/VS services for such functions as task and storage management.

Distributed Data

Through CICS/DOS/VS intercommunication support, DL/I application programs can access a data base that is resident on another processor. The application programmer does not have to know where the data is located. This enables DL/I users to:

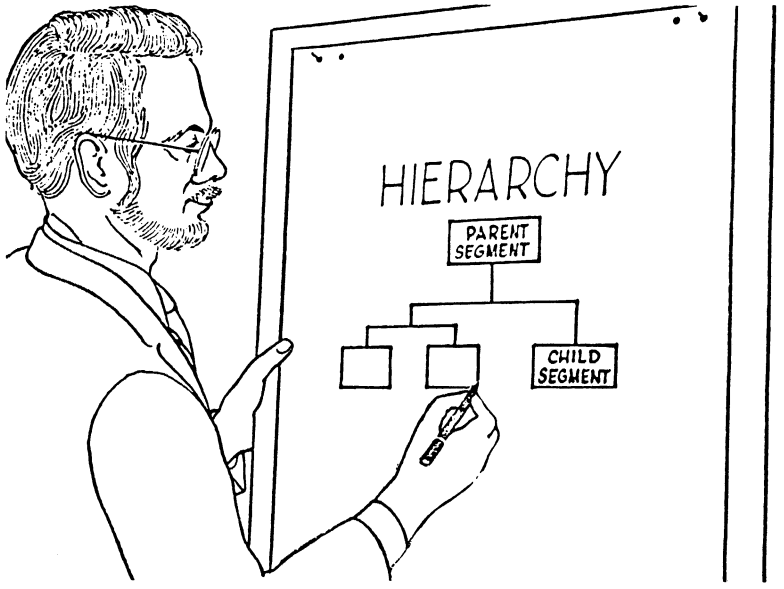
- Define and maintain data bases where most of the activity for that data base occurs while providing access to the data base from all interconnected systems.
- Centralize their data bases while distributing the processing of this data to other interconnected systems.

Multiple Partition Support (MPS)

The DL/I system operating in an MPS environment allows batch and online application programs to simultaneously access the same data base(s). This permits, for example, online applications to issue inquiries to a data base while a batch program updates that data base. MPS also uses CICS/DOS/VS services.

Chapter 3. Characteristics of DL/I

Implementing a DL/I data base management system requires a variety of technical decisions. These decisions, which are discussed in this chapter, range from the type of data organization you employ to the utilities that can be used to perform routine functions.



Hierarchical Data Structure

In the DL/I data base, data is represented as a hierarchical structure, where certain information in a data base is related to other subordinate types of information in a hierarchical manner. The most significant data resides on hierarchically higher levels while less significant but related data (dependent data) appears on subordinate levels. This hierarchy is called a *logical data structure.*

DL/I application programs deal with logical data structures. *Logical* refers to the manner in which the application program sees the data. A logical data structure is always a hierarchy of data elements called *segments.* Programs written to process logical data structures can be independent of the physical data structure. *Physical* refers to the manner in which data is stored on a tape or a direct access device. An application program in DL/I never deals directly with a physical data structure. Most data processing information, regardless of application, can be viewed as a logical data structure. Figure 6 shows an employee data base as it could be described in DL/I.

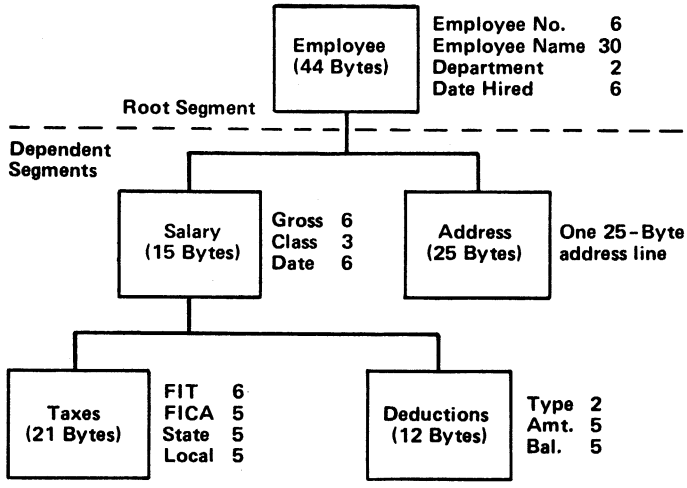


Figure 6. DL/I Logical Data Base Structure (Hierarchy)

Segments and Their Relationships

Each box shown in Figure 6 is known in DL/I terms as a segment. In this case, the structure depicts five different types of segments (Employee, Salary, Address, Taxes, and Deductions). Each segment consists of one or more data fields. Taken together, the segments constitute a data base record. The data base record, by definition, consists of one root segment and all of its of dependent segments.

The Employee segment, for example, has the employee number, name, department number, and date of hire. In this example, the Employee segment contains the identifying information for the data structure (name and number) and it is called the *root segment*. There can be only one root segment in a data hierarchy. All others are dependent segments. The Salary and Address segments relate to the Employee segment, and depend on the Employee segment for their full meaning (Whose salary? Whose address?).

Each type of segment can vary in length (for example, Employee is 44 bytes, Salary is 15 bytes, and so on). Segments of the same type may also vary in length. In this example, all segments of a specific type in the data base, such as Employee, are the same length.

There can also be many levels of dependency. For example, the Taxes and Deductions segments are subordinate to, and depend on, the Salary segment. The Salary segment, in turn, depends on Employee, the root segment.

The basic element of a data base is the segment. The basic building structure of the hierarchical data base is known as a *parent/child relationship* between segments of data. In Figure 7, the root segment (Employee) is also the *parent* of all the segments (that is, Salary and Address) that depend on it. The dependent segments, Salary and Address, are called *children* of the parent segment (Employee).

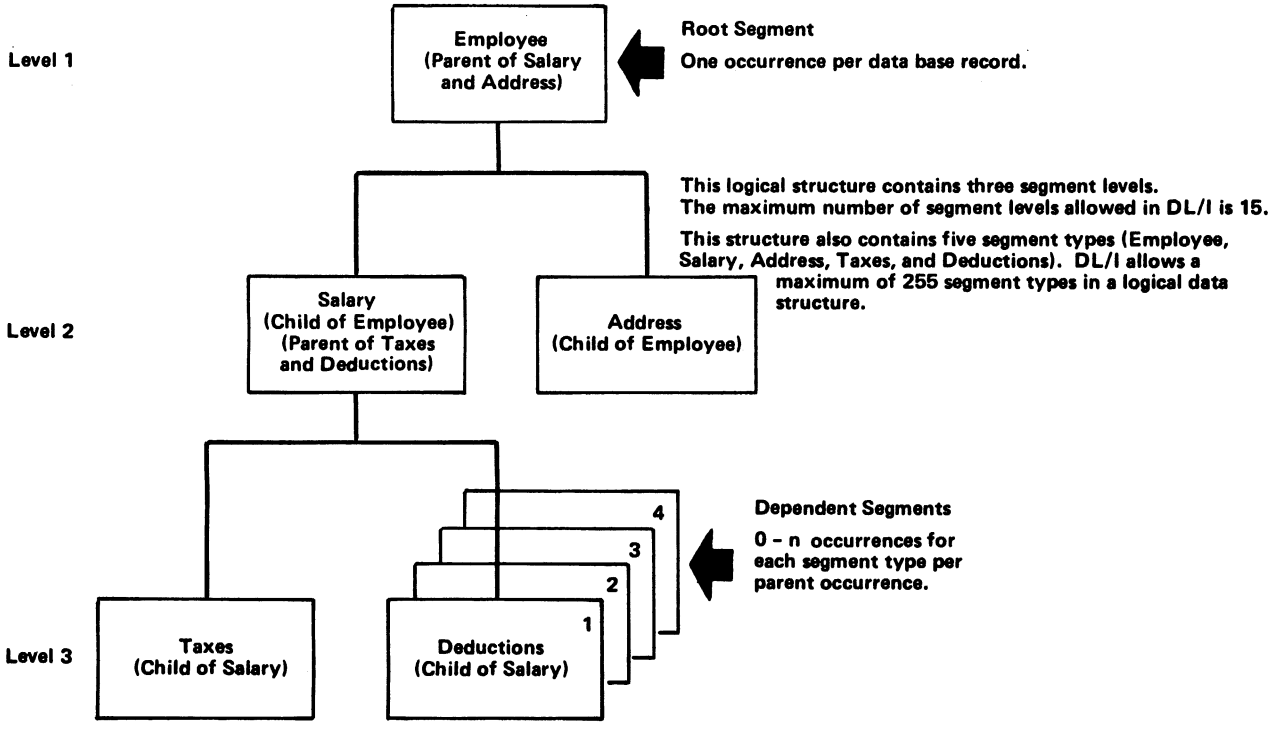


Figure 7. Logical Data Structure — The Programmer's View

The same relationship exists down the structure (Salary is the parent of Taxes and Deductions; Taxes and Deductions are children of Salary).

As shown in Figure 7, a parent (Employee) can have several child segment types (Salary and Address). Also, a child segment (Salary) can, at the same time, be a parent segment; that is, have children itself (Taxes and Deductions).

Because each dependent in the hierarchy has only one parent (immediate superior segment), the hierarchical data structure is sometimes called a tree structure. Each branch of the tree is called a *hierarchical path.* A hierarchical path to a segment contains all consecutive segments from the top of the structure down to that segment.

Data Organizations

DL/I supports two basic physical storage organizations. These organizations are *hierarchical direct* and *hierarchical sequential.* Each organization has certain advantages and should be selected based on your specific needs. The primary differences between the two organizations are the manner by which the segments are related and the techniques of data access.

Hierarchical Direct

Segments of one data base record in the hierarchical direct organization are stored in one or more blocks. These segments are related by direct addresses. Each segment in the data base record may be related to segments of the same type as well as dependent segment types through direct addresses.

Advantages of Hierarchical Direct

The hierarchical direct organization has the following advantages:

Performance

- Through the use of pointers, dependent segments in long data base records can be obtained with fewer physical access operations. This results in more rapid direct access to the segments within a data base record.
- Once segments are stored they are never moved; thus pointer maintenance is reduced.
- Data can be directly accessed through a randomizing routine or through an index. The index can be reorganized without reorganizing the indexed data.
- Logical relationships (special relationships between segments in the same or different data bases), secondary indexing (alternate paths into a data base), and variable length data can be employed.
- Multiple tasks can update the same segment type concurrently because locking is done at the segment occurrence level.

DASD Space

- When segments are deleted from the data base, space is freed and made available immediately. Because of the reuse of space, fewer reorganizations of the data base are required.
- It is possible, through user provided exit routines, to edit or compress segments before they are physically stored in the data base, and to reconstruct or expand them after retrieval from the data base. This feature can save additional direct access space and increase data security.

Direct Access Methods

The access methods implemented with the hierarchical direct organization are:

- HD (Hierarchical Direct) randomized access
- HDAM (Hierarchical Direct Access Method)
- HD (Hierarchical Direct) indexed access
- HIDAM (Hierarchical Indexed Direct Access Method)

The storage organization used for these access methods is essentially the same. The primary difference between them is that direct access to data is through a randomizing module for an HD randomized or HDAM data base and through an index for an HD or HIDAM indexed data base.

Hierarchical Sequential

Segments in the hierarchical sequential organization that represent one data base record are related by being physically adjacent. This requires that segments

representing one data base record be contained in a variable number of storage blocks unique to that data base record. When using hierarchical sequential access, both sequential and indexed sequential access are available.

Advantages of Hierarchical Sequential

The advantages of hierarchical sequential access are:

- Segments can be defined and distributed to make efficient use of the storage media. Little or no control information is required to describe each segment stored.
- Efficient space utilization results from having multiple logical records per physical record.

Sequential Access Methods

HSAM (Hierarchical Sequential Access Method), Simple HSAM, HISAM (Hierarchical Indexed Sequential Access Method), and Simple HISAM are the four access methods implemented with the hierarchical sequential organization.

HSAM and simple HSAM are used for sequential storage and access on tape and direct access storage. HISAM and simple HISAM are used for indexed sequential access to hierarchical sequential organization.

Simple HSAM and simple HISAM data bases contain only root segments.

Defining the Data Base

The logical and physical definitions of the data base are kept in two tables external to the programs in the DL/I system itself. These tables are called the DBD (data base description) and the PSB (program specification block). They are created and maintained by an individual responsible for data base administration.

Because both tables are created and maintained independently of your application program(s), they provide the basis for *data independence.*

When the DBDs and the PSBs have been created, all the attributes of the data bases contained in those DBDs and the PSBs can be referenced for effective administration of the data base definitions. This is accomplished with the *DL/I Documentation Aid* facility.

DBD (Data Base Description)

The DBD (data base description) describes most of the file characteristics you must put into every non-data base program. Each DBD is created from statements you provide. The statements define the hierarchical data structure and physical organization of the data base. The DBD contains a description of the contents of the data base, the names of the segments, their hierarchical relationship, and the physical organization and characteristics of the file. You can think of the DBD as the master description of everything that is in the data base.

The process of generating a DBD is called *data base description generation* (DBDGEN).

The DBD provides DL/I with the mapping from the data structure of the data base used in the application program to the physical organization of the data. The data

structure can be re-mapped into a different physical organization without program modification. This is done by changing the DBD. Other data can also be added to this data base and not require a change to the original programs. The concept of the DBD reduces program maintenance caused by changes in the data requirements of the application.

PSB (Program Specification Block)

The PSB (program specification block) defines the data structure for each application program. It is created from statements you provide for each of your programs. The PSB defines which segments of the data base a specific program requires (the data structure required by that application program).

A PSB contains one or more PCBs (program communication blocks), one for each hierarchical data structure the program intends to use. Each PCB defines the hierarchical (sub)structure the program sees of the data base. It specifies for each segment the kind of access allowed by the program (read only, update, insert, load, and delete). There is at least one PSB for every program that uses the data; more than one program may use the same PSB. You can think of the PSB as describing the logical data needed for the program (usually a subset of the entire data base).

The process of generating a PSB is called *program specification block generation* (PSBGEN).

DL/I Documentation Aid

The DL/I Documentation Aid provides tables containing the attributes of the data bases as described in the DBDs (data base descriptions) and the PSBs (program specification blocks). The contents of these tables enable more effective organization and documentation of DBD and PSB information.

These tables, which are provided by DL/I using the Structured Query Language/Data System (SQL/DS), present information on two levels: (1) the system view or the physical characteristics of the data bases, and (2) the application view or the logical structure of the data bases.

The first level includes such things as:

- The organization of the data bases.
- The type of access used for the data.
- The physical characteristics of the segment types and how they are related to other segments.
- The fields within the segments.

The second level includes such things as:

- The languages of the application programs using the data bases.
- The logical data structures and the segments within a data base to which the application programs are sensitive.
- Those fields in a segment to which an application program is sensitive.

The attributes of your data bases, as defined in the DBDs and the PSBs, can be quickly referenced through a 3270-type terminal using the interactive facility (ISQL) of the Structured Query Language/Data System (SQL/DS). Reports containing information about specific attributes can also be printed. A set of ISQL sample query and report routines are provided with DL/I to use with the DL/I Documentation Aid.

The DBD and PSB definitions are collected and stored in the tables by the DL/I Application Control Blocks Creation and Maintenance Utility program.

The DL/I Documentation Aid is implemented through the Structured Query Language/Data System, program number 5748-XXJ.

Interactive Macro Facility (IMF)

The DL/I Interactive Macro Facility (IMF) simplifies the task of creating, modifying, or deleting DL/I control blocks such as the DBD and the PSB. It provides formatted displays on a 3270-type terminal and prompts the user, who can then choose the appropriate activity and enter the required information interactively instead of using conventional coding methods.

IMF is implemented through the Interactive System Productivity Facility, program number 5668-960.

Application Programming with DL/I

The application programmer can request DL/I functions in one of two ways: through the High Level Programming Interface (HLPI) or through the lower-level program CALL interface.

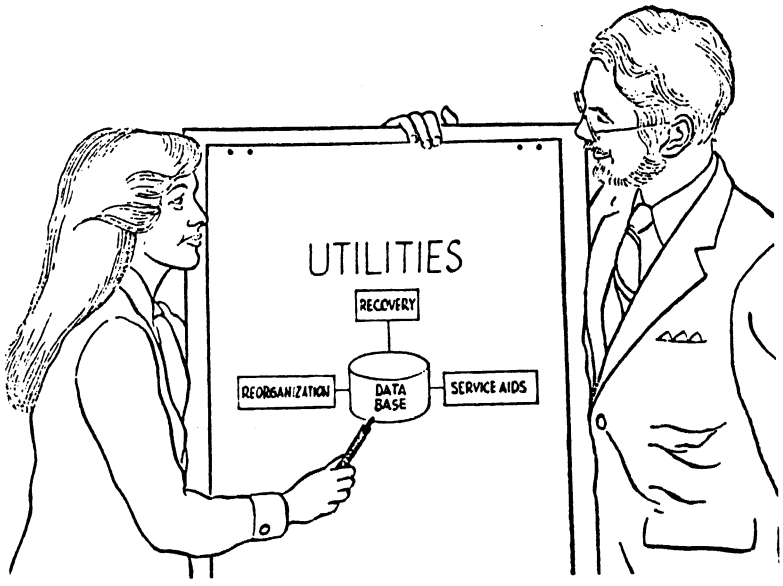
HLPI provides an easier way for application programmers to access DL/I data bases in a batch, MPS batch, or CICS/DOS/VS online environment. It is made up of a series of commands and represents a high-level approach to use the DL/I functions.

The program CALL is a lower-level method of requesting the same DL/I functions.

COBOL and PL/I application programmers can use either HLPI or the program CALL interface. Assembler application programmers can use only the program CALL. RPG II programmers can request DL/I functions by using an RQDLI (Request DL/I) command, which is translated into a program CALL by the RPG II Translator.

Utilities and Serviceability Aids

IBM-provided utilities and serviceability aids are available with DL/I. The Interactive Utility Generation (IUG) facility is provided to make it easier for users to create the job streams and the necessary parameters to run the DL/I utilities.



Interactive Utility Generation

The Interactive Utility Generation (IUG) facility makes its easier to generate the job control statements and parameters required to run each of the DL/I utility programs. It provides formatted displays on a 3270-type terminal and prompts the user for information to generate the job streams and parameters.

VSE job control statements are also created with IUG. The VSE/VSAM Space Management for SAM Feature, program number 5746-AM2, is used whenever possible for sequential access disk files to reduce the amount of data you must enter into the system to allocate disk space for the utilities.

IUG reduces the amount of information you must enter each time the utilities are run because certain information is saved and can be used later. Future job stream generations can also use this saved information, eliminating the need to reenter the data.

IUG is implemented through the Interactive System Productivity Facility, program number 5668-960.

Data Base Recovery Utilities

A group of DL/I utility programs supply the following functions:

- Backing out changes to the data bases.
- Copying data bases.
- Accumulating data base changes.
- Restoring a data base using prior copies and changes.
- Printing the contents of DL/I log files.

Data Base Reorganization/Load Processing Utilities

The DL/I reorganization utilities provide these basic functions:

- Reorganizing DL/I data bases, for example, changing DASD devices or reclaiming DASD space.
- Establishing logical relationship connections when initially loading data bases.
- Creating secondary index data bases when loading your data bases or when you are reorganizing them.

The reorganization utilities use the *DOS/VS Sort Merge*, 5746-SM2, or equivalent.

Automatic ISQL Extract Defines Utility

The DL/I Automatic ISQL Extract Defines utility is designed to eliminate the need to manually describe the organization and structure of the DL/I data necessary to retrieve information from DL/I data bases with the EXTRACT facility of SQL/DS.

This utility creates and stores an ISQL routine composed of ISQL Extract Define commands from data previously gathered and stored in tables with the DL/I Documentation Aid, which is described earlier in this book. Once the routine is created, it can be run under ISQL to define the necessary DL/I information to the EXTRACT facility.

The Automatic ISQL Extract Defines utility can reduce the possibility of errors and save time because the need to manually code the required DL/I information is removed and performed automatically.

Serviceability Aids

Serviceability aids are available to:

- List the most recent DL/I requests.
- Trace errors that occur in online operations.
- Print trace entries from tape or disk input files.
- Trace user selected points in the DL/I code.

Application Programming Support

Low-Level Code and Continuity Check

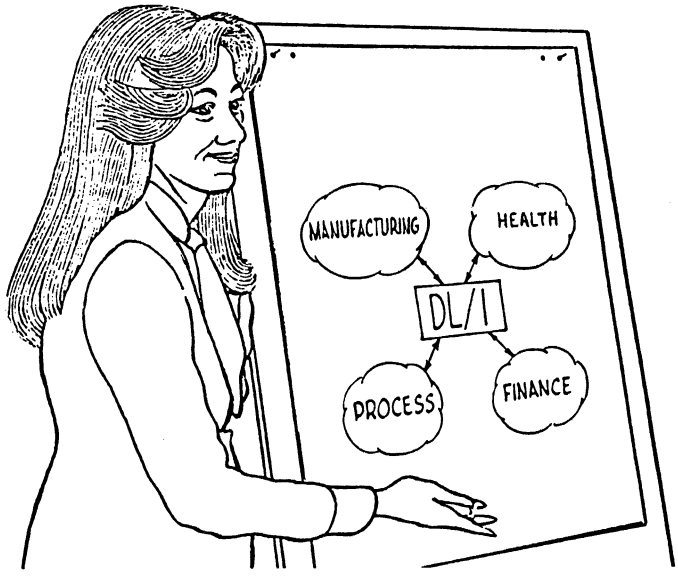
Low-level coding and continuity checking are established techniques for production planning and control in the manufacturing industry. DL/I includes an application program that provides these techniques.

Low-level coding is generally used to facilitate the planning of material requirements, from end users down to raw materials and purchased parts. Continuity checking assures the continuity of the related bills of material or, in other words, it prevents a part from being accidentally contained within itself.

The techniques of low-level coding and continuity checking can also be applied to non-manufacturing problems, which can be expressed as networks or as directed graphs.

Chapter 4. Sample Applications

This chapter demonstrates some of the environments in which DL/I is useful. They include financial, health, processing, and manufacturing industries. These selections are by no means all inclusive, but they show that a variety of relationships can be accommodated by DL/I in each area. DL/I provides the ability to add new information into existing data bases as well as the ability to create new data bases with minimal impact to the system user. This data base approach to system design promotes evolutionary system development in a growth environment.



Financial Industry

The following list represents typical requests for information in banking:

1. What are all the accounts associated with a particular individual? This information might be desired when a customer wants to make a deposit but does not know his account number.

2. What is the status of loans outstanding to a particular individual? This information might be useful when a bank officer is asked to accept a loan request from a customer.

3. Does the checking account of a known account number contain a balance adequate to cash a check?

4. What is the amount of money required to pay off the installment loan for a known loan account?

5. What is the property held and what is the par value for each property in a trust account.

6. What trust accounts hold a particular property, such as a particular stock? And what is the quantity in shares held of the stock in each trust account?

These questions and the subsequent data structures should stimulate you to consider other questions and additional data elements and structures. Both query and update operations against the data base are possible.

The first question, concerning all accounts associated with a given individual can be answered with the logical data structure of a customer information record in Figure 8.

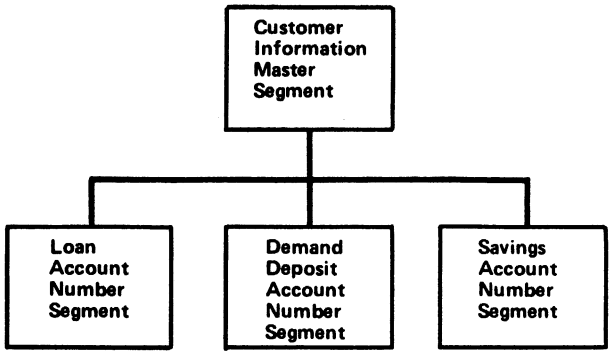


Figure 8. Logical Data Structure of a Customer Information Record — Financial

In this example, a customer information master segment exists for each of the bank's customers and contains name, address, dates, codes, and a customer-identifying key. The loan, demand deposit, and savings account number are keyed on account number and indicate account type and relationship to individual customers.

The second question, concerning the outstanding loans for an individual, can be answered with the same data structure.

The third question, concerning the current balance in the demand deposit account, can best be answered with a data structure organized by account number. However, we already have established a demand deposit account (number) subordinate to the customer information segment (Figure 8). DL/I enables you to enter the data base either by customer name or account number. This can be done by using the pointer-target concept as illustrated in Figure 9.

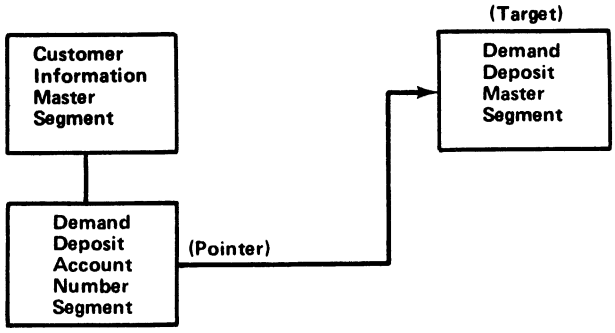


Figure 9. Data Structure with Pointer-Target Relationship

Through the DL/I logical relationship, a combined data structure can be created as shown in Figure 10.

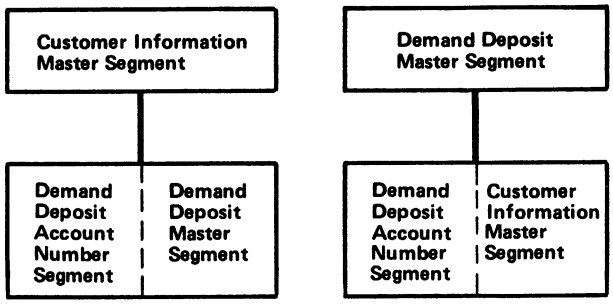


Figure 10. Logical Data Base Structures Showing Customer Information Specifications

The questions concerning the properties held in a particular trust account can be answered with the logical data structures, i.e., the application program's view of the data, as illustrated in Figure 11.

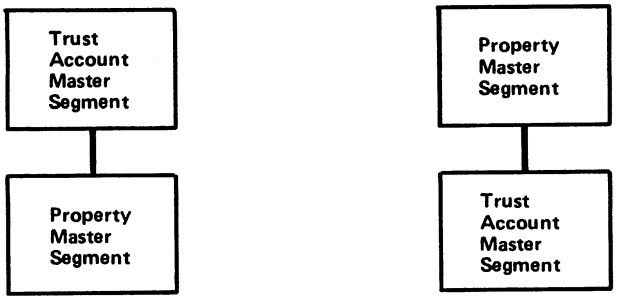


Figure 11. Logical Data Structures Showing Properties and Trust Information

The two questions about the properties in the trust account are answered with inverse data structures. They present a problem similar to the customer information-account master data structures. Again, the problem can be resolved using the pointer-target concept as illustrated in Figure 12.

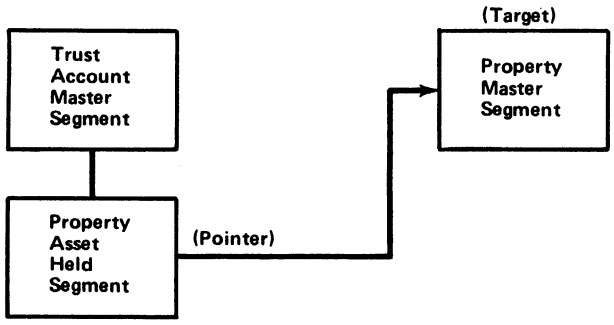


Figure 12. Logical Data Structure with Pointer — Target Relationship

Through the DL/I logical relationship, a combined data structure can be created as shown in Figure 13.

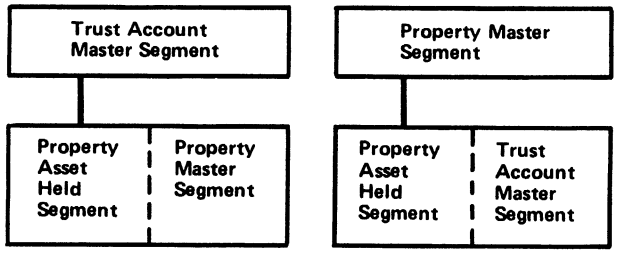


Figure 13. Logical Data Structure with Property and Trust Information

Health Care

In health care, a typical use of DL/I might be the storage of information about patients in a hospital or clinic. Associated with each patient is one record with a root segment containing basic information about the patient. This patient basic segment can be keyed on patient identification such as medical record number. In addition, it contains name, address, age, birth date, and sex (See Figure 14).

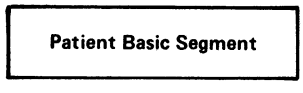


Figure 14. Health Data Base Record Root Segment — Patient Basic Segment

This data base record can be expanded to include information about the patient, including visits to the hospital or clinic, diagnostic information, prescriptions, surgery, laboratory test data, and so on. This information can be organized into a logical tree structure. Let's examine how this structure can be built as shown in Figure 15.

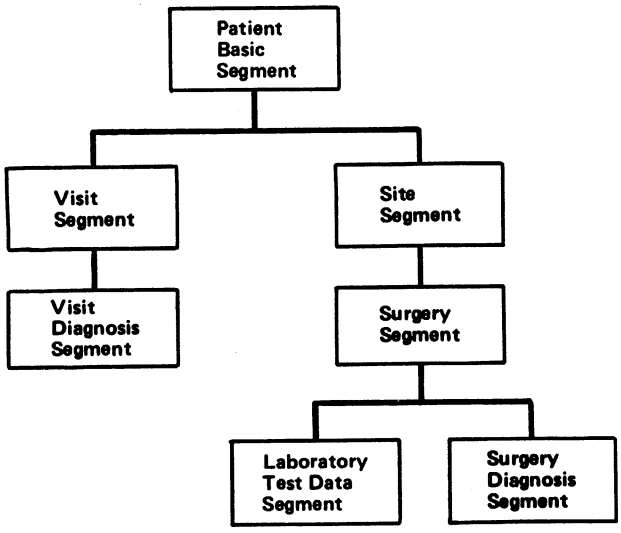


Figure 15. Logical Data Structure of Health Data Base Record or Data Base

For each visit to the hospital or clinic, a visit segment is added to the basic segment as a dependent. The visit segment contains the data and purpose of the visit as well as the attending physician's name. If information obtained during the visit results in a diagnosis of the patient's problem, it's in the diagnosis segment.

Certain visits might also involve diagnosis requiring surgery to be performed on a particular site of the human body. Different approaches may be taken to develop a logical view of the data base record, but one is to make the body's site of the surgery directly dependent to the patient basic segment and the surgery segment dependent to the site segment as illustrated in Figure 15.

It would also be appropriate to consider laboratory test data and diagnosis segments under the surgery segment since they would be pertinent to a particular operation.

The structure in Figure 15 can be expanded further. For example, if the visit to a clinic or doctor does not involve surgery, but involves X-ray or radioactive treatment, additional segment types can be added as dependents of the site segment. If a visit involves only the application of, or prescription for, a drug, a drug segment type can be added as a dependent of the visit or diagnosis segment.

Questions that might be asked of information contained within a data base of the structure depicted in Figure 15 can be of a simple or complex nature. The simple type of question to answer would be a request for information about a particular patient. The answers to questions of this simple type are facilitated because the data base is structured on patient identification sequence. Online terminal inquiry and update are quite practical.

A complex question might involve listing patients or information about patients who received a particular drug, contracted a particular disease, or satisfy a combination of such query criteria. The use of the pointer-segment concept can be used to answer these questions. Additional tree structures can be created within the patient tree structure. These tree structures can be associated with drugs, diseases, or other search arguments.

Process Industry

Online Order Entry and Production Control System

A total order entry and production control system has computer control of production from acceptance of orders through manufacturing, shipment, and delivery of the order. An implementation plan for these applications can be developed from the relationships that exist among these functions. The plan allows for a logical growth pattern through the implementation of online order entry, in-process inventory control, and plant balancing.

One of the major requirements in the installation of such a large-scale system is the ability to implement the individual program in modular increments. Each increment should gradually increase the functional scope of the system without the necessity of reprogramming previously written applications that use the same data base. The existing data bases should grow to service the additional applications.

A good beginning for a total system is an online order entry application that includes all of the processing operations necessary to accept orders from customers and provide the follow-up until the entire order has been shipped. The initial phase of online order entry includes acceptance of orders for stocked items. A subsequent expansion places additional information in the data bases, permitting acceptance of orders for items that call for a mill order to produce those items. When this application is added, it is necessary to add programs that check in-process inventory, operation routing, and facility loading information. It is also necessary to create mill order plan records. The organization of these required records is shown in the examples.

In addition to the functions performed by the online order entry, the in-process inventory control programs provides basic material control and order status. Programs to balance long-range scheduling objectives with short-range objectives can subsequently be added to the application.

If the data bases and programs described were implemented, the following types of questions could be answered:

- What is the availability of a product requested by a customer?
- What ship date can be promised for an item that requires manufacturing?
- What is the credit limit of a given customer and what is the total amount of unpaid invoices?
- What is the status of an existing customer order?

In addition, many changes to the information contained in the data bases can be entered from the communication terminals. This would include the following types of transactions:

- Enter receipt of new stock.
- Notification of low limits of inventory.
- Adding new items to an existing order.

- Changing the quantity previously entered in a customer order.
- Changing the ship-to location of an existing order.

The following logical data structure (Figure 16) can be used as a base to install the applications, permit the entry of data as listed, and answer the types of inquiries described.

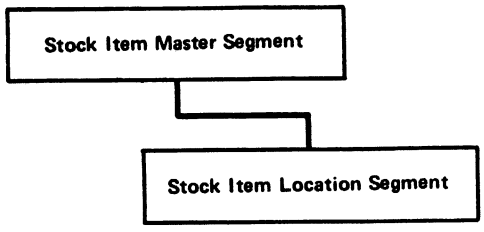


Figure 16. Logical Data Structure for Stock Item Data Base

Figure 16 illustrates a logical data structure in which a stock item master segment exists in a data base for each stocked item. Subordinate to a stock item master segment, one or more stock item location segments exist. These segments name the inventory locations where the items are stocked and give the quantities available.

The question concerning the ability to fill an order from stock can be answered with the data structure in Figure 16 by making inquiries against the described stock item records.

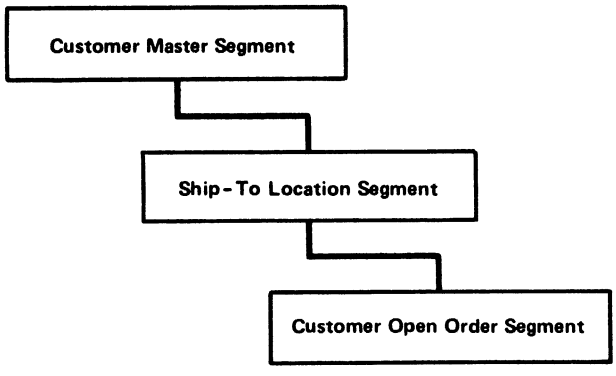


Figure 17. Logical Data Structure for Customer Master Data Base

To do a credit check, credit information must be available for the individual customer. Consider the logical data structure in Figure 17. The customer master segment is keyed upon a unique customer identifier and contains information such as credit clearance level, name and address, and total amount of unpaid invoices. This segment provides the answer to the credit check. The customer may also have one or more locations to which he wishes the orders to be shipped. The ship-to location segments provide this information. The customer open order segments

indicates all open orders for a particular customer and a particular location. These segments provide the pointers to the details of each open order. The open order data structure is shown in Figure 18.

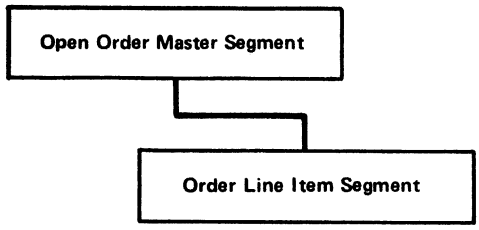


Figure 18. Logical Data Structure for Open Order Data Base

This logical data structure contains an open order master segment, which is keyed upon the order number. Subordinate to the open order master segment are one or more segments describing each line item in an order.

The open order master segment contains the status of the order, the due date, the customer name, and the customer's ship-to location for the order. Using this logical data structure and the stock item data structure previously discussed, order status inquiries can be answered.

If this application is expanded to include in-process inventory items as well as stocked items for handling orders, these additional questions might be asked:

1. What is the status of an open order that needed a mill order to produce the item?

2. What is the routing of operations performed in processing a given product?

3. What is the workload on a given plant facility? If a particular breakdown occurs, can another facility be used to complete the order?

4. If a plant facility breakdown occurs, what is the effect on an order's status?

In addition to answering these questions, data structures can allow data base update processing to assure maximum plant facility usage and minimal time until customer order availability.

Figure 19 describes the mill order plan and routing relationship. The mill order plan segment is keyed on mill order item number and includes a plan of manufacture and a time schedule. The in-process inventory segment includes the status of a mill order. Each operation routing segment describes an operation performed in the process of producing the mill order item and the facility at which the operation is to be performed.

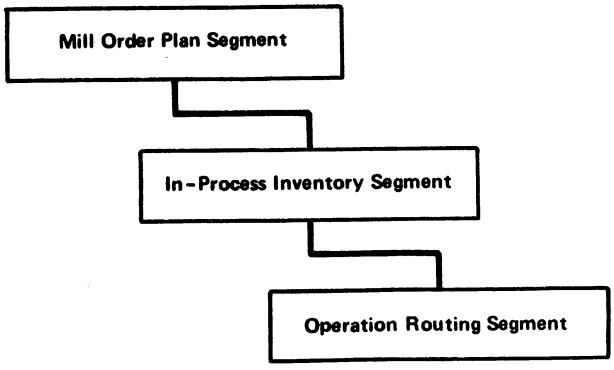


Figure 19. Logical Data Structure for Mill Order Planning

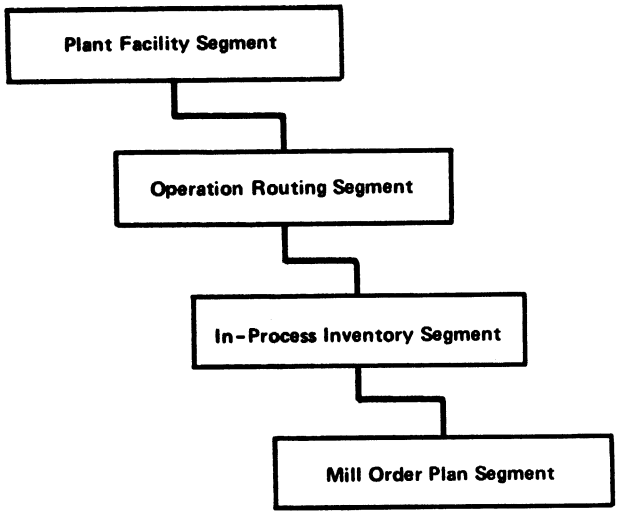


Figure 20. Logical Data Structure for Plant Facility

In Figure 20, the plant facility segment is keyed on facility number and contains facility loading data, such as total time scheduled, and so on. The operation routing segment contains the individual operations performed on a given mill order item at that facility. Figure 19 and Figure 20 are actually inverse data structures of each other and can be considered for physical storage using the pointer-target concept, which is illustrated in Figure 21.

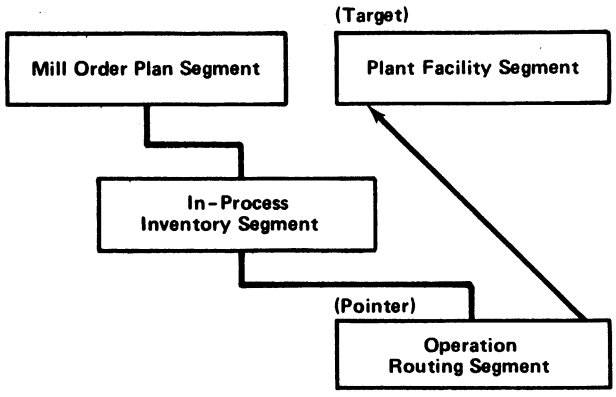


Figure 21. Logical Data Structure for Mill Order Planning and Plant Facility. This uses the pointer-target concept (combined Figure 19 and Figure 20).

Manufacturing Industry

The following list of questions represents typical requests for information in the manufacturing industry:

1. What are the components of an assembly, and in what quantity is each component required?
2. In what assemblies is a given part used, and in what quantities is the part used?
3. What is the inventory status of a part where multiple inventory sites exist?
4. What open purchase orders exist for a given part?
5. What are all the open purchase orders?

These questions can be answered from our sample application which assumes the physical data base records to be in three data bases as shown in Figure 22. These three data bases are interrelated through a logical relationship.

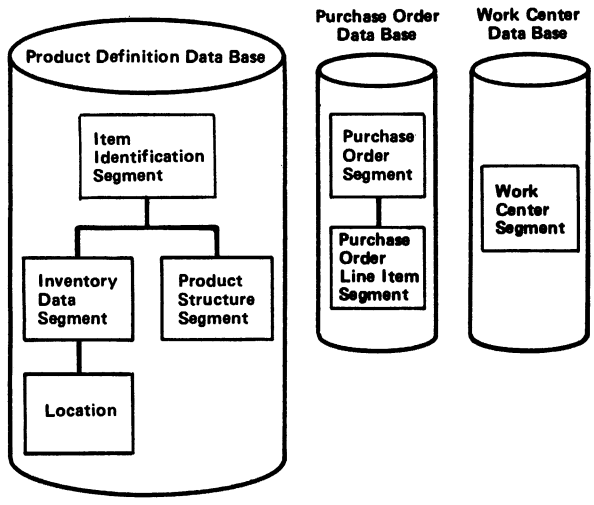


Figure 22. Physical Data Base Records

The first two questions relate to the structure of a product and can be asked for a particular part, a subassembly of a product composed of many parts, or an entire product assembly. One question, in reality, is the inverse of the other. The logical data structure for answering both questions is shown in Figure 23.

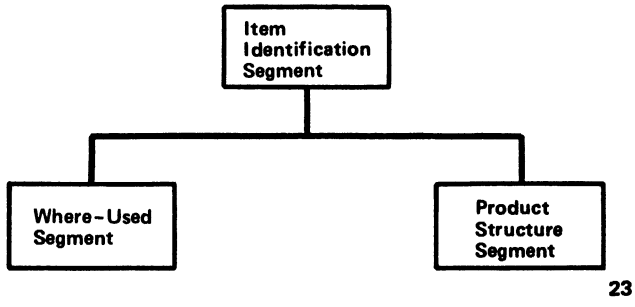


Figure 23. Logical Data Structure for Product Data Base

However, the where-used information for one part is the same as the component part information of the part where it is used. Actually, the where-used information for one part is redundant with the component part information of its assemblies. The functions of the where-used part segment and the component part segment can be achieved by one segment type. This segment type is called the product structure segment. A one-level bill of material is produced by proceeding from an item segment to its product structure segments by a parent-child relationship as shown in Figure 24.

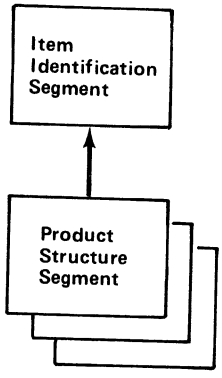


Figure 24. Parent-Child Relationship

The inventory status for a particular part could be supplied by the data structure in Figure 25.

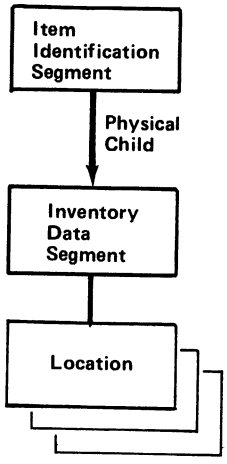


Figure 25. Logical Data Structure for Part Inventory Data

Each location segment for a particular part is a dependent segment under the inventory data segment. Each represents the inventory for the part at a particular location.

The open purchase orders and vendors assigned for a particular part could be supplied by the data structure in Figure 26.

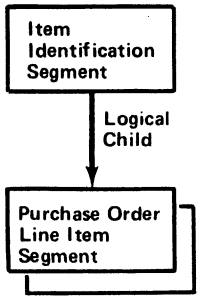


Figure 26. Logical Data Structure for Part Purchase Order

Chapter 5. Installation Responsibilities

To ensure successful implementation of your programs with DL/I, certain requirements are placed on you. These fall into the category of installation responsibilities, some of which are:

- You must ensure that personnel are sufficiently trained in DL/I operations.
- You must decide which applications are to run under DL/I. You may write your own applications or you may use available licensed programs, which may eliminate some of the other responsibilities.
- You must design and create the data bases. A data base description (DBD) generation must be performed for each data base.
- You must define how each application will access data bases through the program specification block (PSB).
- Your application programs must be written (or modified) to access DL/I data bases using DL/I requests. If segment edit/compression is planned, appropriate exit routines must be written. In addition, if you access data bases with the HD randomized or HDAM access methods, a randomizing routine must be used. You may provide your own randomizing routine or use one of four routines provided by DL/I.
- If you plan CICS/DOS/VS online operations, you must tailor your operations for, and generate, a CICS/DOS/VS system. You must also train your personnel in CICS/DOS/VS operations.

Chapter 6. Machine Configurations

Minimum DL/I Configuration

The minimum requirements for executing DL/I are outlined below. The real storage requirements of the DL/I application are not included. Two environments are shown, a batch environment and an online environment in conjunction with CICS/DOS/VS. In both cases, a limited and a full set of DL/I functions are considered to show the resulting difference for the real storage requirements.

The estimates of real storage requirements are derived from the size of the modules in the DL/I system necessary to handle normal execution, that is, modules handling exceptional conditions such as errors or open/close have not been included. Thus, if the specified real storage is available to DL/I and if no exceptional conditions arise, no paging will occur in the DL/I code. Of course, as for every system executing in a virtual storage environment, if performance is not critical, DL/I can execute in less real storage. Also, a trade-off may be considered between the available real storage in a central processing unit model and its instruction rate. For example, a 4341 may require less real storage than a 4331 to achieve the same performance, since its internal speed can make up for some additional paging overhead.

The minimum storage estimates provided here are based on the following application profile:

- One HISAM data base.
- Three VSAM key-sequenced-data set (KSDS) buffers and two VSAM entry-sequenced-data set (ESDS) buffers.
- VSAM Control Interval size of 512 bytes.
- No logging.

For the online system, the profile includes:

- Two PSBs in use.
- One active task.
- Program Isolation not selected.

System Function	Units Permitted
Processing Unit	Any IBM system using VSE (Virtual Storage Extended) with a minimum real storage of 256K for a batch system or 512K for an online system in conjunction with CICS/DOS/VS.
Minimum Real Storage (Including VSAM)	The practical amount of real storage is: Batch System • Basic retrieve operations only — 66K • All Functions — 84K Online system (excluding CICS/DOS/VS requirements[1]) • Retrieve operations only — 76K • All Functions — 94K Multiple Partition Support (MPS) The real storage requirements are the same for a system with MPS as they are for an online system plus 4 to 10K for each active MPS batch partition.
Minimum Virtual Storage	Virtual storage must be: Batch system — 512K Online system — 1024K
Tape Units	At least two 9-track 2400 or 3400 series tape units and control are required if tape logging is used.
Direct Access	For DL/I data base storage, within the capabilities and restrictions of VSE support by the virtual storage access method (VSAM) and sequential access method (SAM). 2314/2319 Direct Access Storage Facility 3310 Direct Access Storage 3330 Disk Storage 3340 Disk Storage 3350 Direct Access Storage 3370 Direct Access Storage 3375 Direct Access Storage
Telecommunications and Programmable Facilities Available	Various terminals, terminal control units, and special features are supported by CICS/DOS/VS.

[1] For information about the storage and CICS/DOS/VS-supported terminals and features, see the CICS/DOS/VS General Information Manual listed in the Preface.

Typical DL/I Real Storage Requirements

The following is an example of the real storage requirements for a typical DL/I production system. The figures are based on the following application profile:

- Five HDAM data bases
- One HIDAM data base
- Use of advanced functions
- Use of data base change logging
- One buffer subpool with 12 buffers of 2K each

- Five 1K buffers for each of two INDEX data bases (HIDAM primary index and one secondary index).

For the online system, the profile covers:

- Ten PSBs in use
- Three tasks running concurrently.

Real Storage

The typical amount of real storage is:

Batch System

- Retrieve operations only — 114K
- All functions — 150K.

Online system (excluding typical CICS/DOS/VS requirements)

- Retrieve operations only — 124K
- All functions — 160K.

Virtual Storage

Virtual storage (including the real storage) is:

- Batch System — 600K
- Online System — 1024K.

Data Language/I Disk Operating System/Virtual Storage
General Information
Order No. GH20-1246-9

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate. Comments may be written in your own language; English is not required.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

	Yes	*No*
• Does the publication meet your needs?	☐	☐
• Did you find the material:		
Easy to read and understand?	☐	☐
Organized for convenient use?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Written for your technical level?	☐	☐

• What is your occupation? _____

• How do you use this publication:

As an introduction to the subject?	☐	As an instructor in class?	☐
For advanced knowledge of the subject?	☐	As a student in class?	☐
To learn about operating procedures?	☐	As a reference manual?	☐

Your comments:

If you would like a reply, please supply your name and address on the reverse side of this form.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Cut or Fold Along Line

GH20-1246-9

Reader's Comment Form

Cut or Fold Along Line

Fold and Tape | Please Do Not Staple | Fold and Tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 812 BP
1133 Westchester Avenue
White Plains, New York 10604

Fold | Fold

If you would like a reply, *please print:*

Your Name _____

Company Name _____ *Department* _____

Street Address _____

City _____

State _____ *Zip Code* _____

IBM Branch Office serving you _____

IBM®

Data Language/I Disk Operating System/Virtual Storage
General Information
Order No. GH20-1246-9

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate. Comments may be written in your own language; English is not required.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

	Yes	*No*
• Does the publication meet your needs?	☐	☐
• Did you find the material:		
Easy to read and understand?	☐	☐
Organized for convenient use?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Written for your technical level?	☐	☐

• What is your occupation? _____

• How do you use this publication:

As an introduction to the subject?	☐	As an instructor in class?	☐
For advanced knowledge of the subject?	☐	As a student in class?	☐
To learn about operating procedures?	☐	As a reference manual?	☐

Your comments:

If you would like a reply, please supply your name and address on the reverse side of this form.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Cut or Fold Along Line

GH20-1246-9

Reader's Comment Form

Cut or Fold Along Line

Fold and Tape | Please Do Not Staple | Fold and Tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 812 BP
1133 Westchester Avenue
White Plains, New York 10604

Fold | Fold

If you would like a reply, *please print:*

Your Name _____

Company Name _____ *Department* _____

Street Address _____

City _____

State _____ *Zip Code* _____

IBM Branch Office serving you _____

IBM®

GH20-1246-09