GH12-5115-1

Program Product

Data Language/I-Entry DOS/VS (DL/I-Entry DOS/VS)

General Information Manual

Program Product 5746-XX7 (DOS/VS)

IBM

GH12-5115-1

Program Product

Data Language/I-Entry DOS/VS (DL/I-Entry DOS/VS)

General Information Manual

Program Product 5746-XX7 (DOS/VS)

This book introduces DL/I-Entry DOS/VS to the executive, system programmer, or application programmer and gives a general picture of what DL/I-Entry is.

The DL/I-Entry data base system makes data handling easier for application programs, and centralizes the data required by more than one application program. Both batch programs running under DOS/VS and online programs running under the the Customer Information Control System/DOS/VS (CICS/VS) can use DL/I-Entry.

This book explains the advantages of using data bases, gives a general description of DL/I-Entry data bases and of how application programs use them, and lists machine configuration requirements. Examples are provided of using DL/I-Entry to control the ordering and distribution of products, to produce a bill of materials in a manufacturing application, and to provide a complete system for order entry and production planning and control in the process industry.

Prerequisite to understanding this book is some familiarity with the use of computers to process data.

IBM

Preface

This book introduces the Data Language/I-Entry DOS/VS (DL/I-Entry DOS/VS or DL/I-Entry) data management control system. The book explains the advantages of using data bases, gives a general description of DL/I-Entry data bases and of how application programs use them, and lists machine configuration requirements. Examples are provided of using DL/I-Entry in distribution, bill of materials, and process industry applications.

This book is intended for the executive, system programmer, or application programmer. A more detailed description of DL/I-Entry can be found in the following publications.

- *DL/I-Entry DOS/VS Application Programming Reference Manual,* SH12-5415
- *DL/I-Entry DOS/VS Design and Implementation Guide,* SH12-5311
- *DL/I-Entry DOS/VS Program Logic Manual,* LY12-5017

Second Edition (February 1976)

This is a major revision of, and obsoletes, GH12-5115-0 and Technical Newsletters GN12-5049 and GN12-5055. Information on the hierarchical direct access method (HDAM) has been added to the section "Data Base Organization and Access Methods" and at other points throughout the book. Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

This edition applies to version 2, modification level 0, of Data Language/I-Entry DOS/VS (DL/I-Entry DOS/VS), Program Product 5746-XX7, and to all subsequent versions and modifications until otherwise indicated in new editions or Technical Newsletters. Changes are periodically made to the information herein; any such changes will be reported in subsequent revisions or Technical Newsletters.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form is provided at the back of this publication for reader's comments. If the form has been removed, comments may be addressed to IBM Program Product Center, 58 Schwertstrasse, D-7032 Sindelfingen, Germany. Comments become the property of IBM.

Contents

Figures

Introduction

Data Language/I-Entry DOS/VS (DL/I-Entry DOS/VS, hereafter referred to as DL/I-Entry) is a data management control system developed to aid the System/370 Model 115, 125, 135, or 145 user in implementing batch or teleprocessing applications. Teleprocessing applications may be implemented through the use of the Customer Information Control System/DOS/VS (hereafter referred to as CICS/VS), which provides the necessary interfaces to DL/I-Entry. DL/I-Entry runs as an application program in a virtual storage environment under DOS/VS.

Today companies evaluate computer systems not only in regard to programming systems and hardware but also in relation to the information needs of the total corporate environment. Increasing demands are made for applications that maintain and interrogate large centralized information files. DL/I-Entry provides a number of features to aid in establishing, changing, and expanding such applications and information files.

DL/I-Entry may be used to advantage with such applications as payroll and personnel, manufacturing bill of materials, inventory control, accounts receivable, hospital records, student records, petroleum well records, and demand deposit accounts systems. Using DL/I-Entry, a company can design its applications to interface with information files from remote terminals (using a CICS/VS interface), or to process information in the more conventional batch mode, or to use a combination of these methods.

These features of DL/I-Entry, together with its ability to respond to frequent and anticipated high-volume information requests, make it a powerful tool for the data processing user.

General Description

The DL/I-Entry system makes data base processing capabilities similar to those used on large systems available to System/370 DOS/VS users. It is a general-purpose data management system that satisfies a wide range of data base processing requirements. DL/I-Entry simplifies the user's task of creating and maintaining large common data bases, offers the user data independence, and frees the user from considerations of access methods and device characteristics.

DL/I-Entry can be used by batch-processing applications or by online data communication applications.

In batch processing, single data base transactions requested by applications are accumulated and then processed periodically against a data base. Because of this elapsed time, data in the data base is not always current. The use of batch processing should depend on how current the user's information must be, viewed in relation to the costs of other methods of processing data.

For data communication applications, DL/I-Entry provides an online processor, which utilizes an interface facility to CICS/VS. DL/I-Entry employs existing user options and exits provided by CICS/VS. CICS/VS is a transaction-oriented terminal management system. This type of system, as opposed to a batch system, responds to each transaction as it is requested. This eliminates the elapsed time inherent in batch processing systems and allows the user to maintain current data for his applications.

For a complete description of the CICS/VS system, refer to Customer Information Control System/DOS/VS General Information, GH20-7028.

What is a Data Base?

The traditional manner of organizing data used by application programs is into data files. Each data file is physically structured to present data in the physical sequence and format required by a particular application program. When the same data is shared by several applications, the data is duplicated on different data files so that it can be presented to each application program in the physical sequence and format required. This duplication uses additional storage space and results in increased maintenance time and cost, since the same data has to be maintained simultaneously in many locations.

The data management portion of the DL/I-Entry system uses data organization methods that free data processing application programs from their dependence on the physical organization of data and from the need to duplicate data. These undesirable attributes of data files have been eliminated with DL/I-Entry through use of the data base.

A data base is defined as "a nonredundant collection of interrelated data items processable by one or more applications."

All application data is stored in one or more data bases in a hierarchical manner; that is, the most significant data resides on hierarchically higher levels while less significant but related data (dependent data) appears on subordinate levels (see Figure 1). Through the use of a concept called "sensitivity," each application program views only that data in the structure which it uses, and accesses that data through DL/I-Entry calls. Assume that one application requires name

and address information, and a second requires name and payroll information. The applications share their common data (name), but only the first accesses address, and only the second accesses payroll. To each application, data used by other applications -- other than common data -- does not exist.

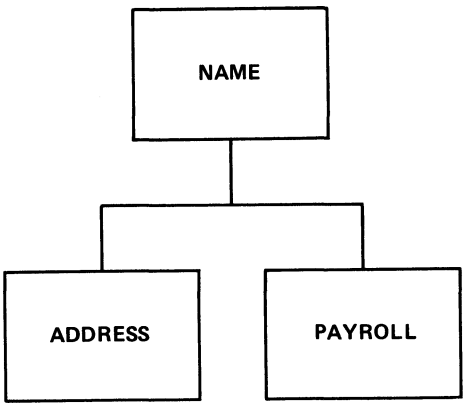


Figure 1. Hierarchical Data Base Concept

In practice, a system analyst or programmer reviews the data requirements of all applications, and then defines the data base or bases that can best serve those applications. To create a data base, the user defines to DL/I-Entry a common data structure and format that serve his applications, and then loads his application data into that structure. This definition is called a data base description (DBD); one DBD is required for each data base. The second definition required is the program specification block (PSB). The PSB defines to DL/I-Entry for each application program the data bases used, the type of data used within each data base, and the operations allowed on each data base. One PSB is required for each application program. Both of these control blocks -- the DBD and the PSB -- are used to link the application data in the data bases to the application program using the data. Through DL/I-Entry's use of the DBD and PSB, application programmers can write their programs without regard to the physical structure of data.

Batch System

The DL/I-Entry batch system (see Figure 2) contains the following functional parts:

- DL/I-Entry nucleus
- DL/I-Entry low-level code/continuity check feature
- DL/I-Entry language interface
- DL/I-Entry processor

Each of these parts is described below.

DL/I-ENTRY NUCLEUS

The DL/I-Entry nucleus is the controlling part of the DL/I-Entry system. It receives control from DOS/VS (step 1), initializes the DL/I-Entry system (step 2), and then passes control to the user's application program (step 3). After the application program has finished processing the data bases (steps 4 through 10), the data bases are disconnected from the DL/I-Entry system (step 11) and control is then returned to DOS/VS (step 12).

The DL/I-Entry low-level code/continuity check feature (step 4) can be used by application programs written in COBOL, PL/I, and Assembler language. In a manufacturing industry, low-level codes placed in the root segments of a parts data base identify the components or materials which make up a finished product. If used, the routine is called from the application program, and in turn issues DL/I-Entry calls like the application program.

DL/I-ENTRY LANGUAGE INTERFACE

The DL/I-Entry language interface is entered when a call to DL/I-Entry is issued by an application program (step 5). The programming languages supported are COBOL, PL/I, RPG II, and Assembler language. Each DL/I-Entry call is translated to a common format and is then passed to the DL/I-Entry processor (step 6).

DL/I-ENTRY PROCESSOR

The DL/I-Entry processor is the data management portion of the DL/I-Entry system. Through this facility, the user inserts, retrieves, deletes, or replaces the data in the data bases used by his application program (steps 7 through 10). As these operations are performed, the DL/I-Entry processor performs all the data maintenance tasks required on the data bases.

Logical relationships and secondary indexing are also handled by the DL/I-Entry processor. Logical relationships are a method of establishing a logical connection between segments unrelated physically -- the segments may be in different data base records or even in different data bases. Secondary indexing is a means of retrieving records based on information the records contain other than the key originally used to store them.

DL/I-Entry provides five access methods:

- Hierarchical sequential access method (HSAM)
- Simple hierarchical sequential access method (simple HSAM)
- Hierarchical indexed sequential access method (HISAM)
- Simple hierarchical indexed sequential access method (simple HISAM)
- Hierarchical direct access method (HDAM).

The virtual storage access method (VSAM) performs data management services for data bases which use HISAM, simple HISAM, or HDAM. The sequential access method (SAM) performs data management services for data bases which use HSAM or simple HSAM. For a description of the DL/I-Entry access methods, refer to "Data Base Organization and Access Methods" in the "System Concepts" section.

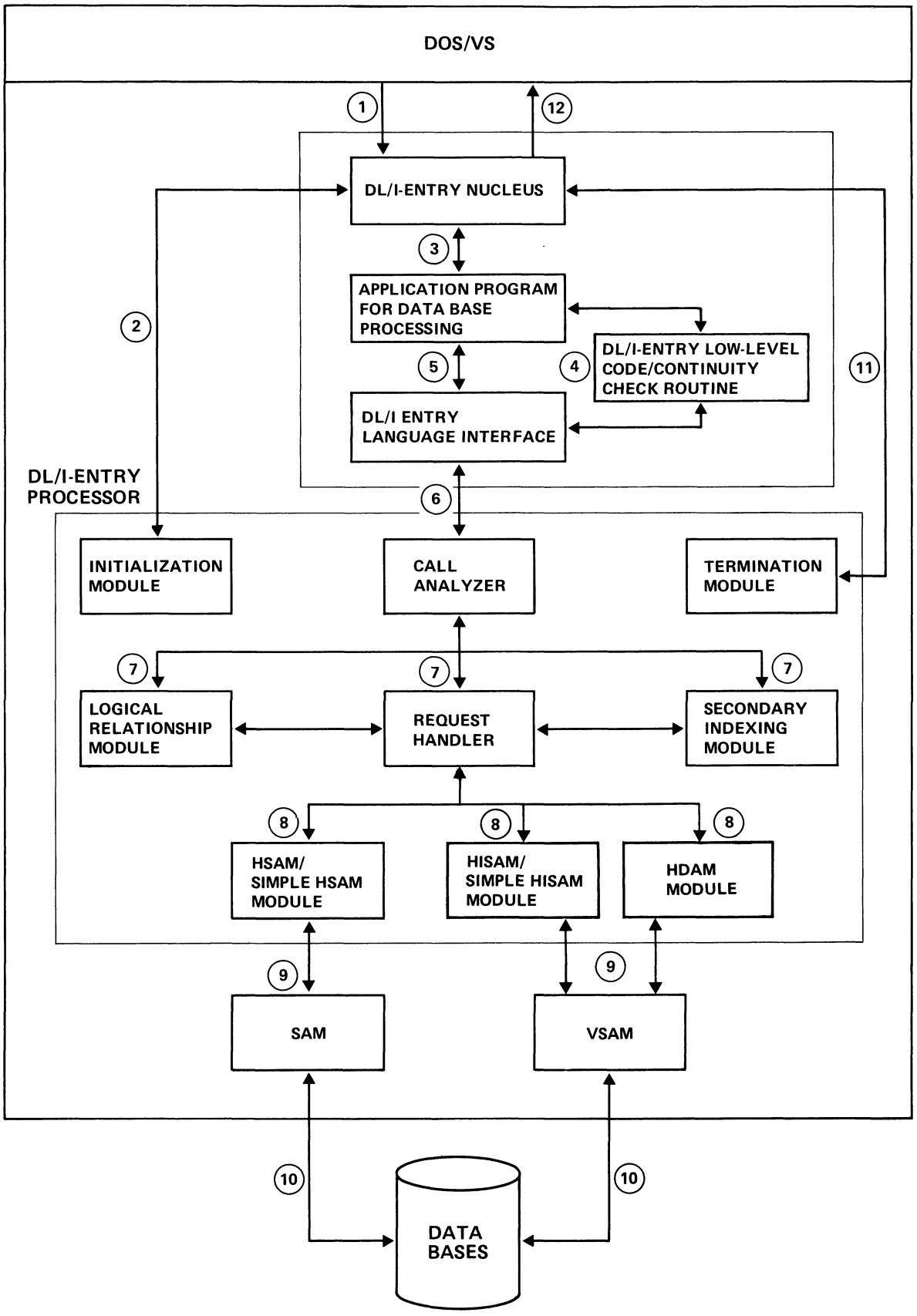


Figure 2. DL/I-Entry Batch System

Online System

The DL/I-Entry online system (see Figure 3) executes in an online environment under the control of CICS/VS. Since it is possible under CICS/VS for more than one program to run at the same time, it is possible for more than one application program to simultaneously work with the same DL/I-Entry data bases.

When a user request is entered at a terminal, CICS/VS activates and deactivates the application program (step 1). The DL/I-Entry online system contains the following functional parts:

- DL/I-Entry language interface
- DL/I-Entry processor.

DL/I-ENTRY LANGUAGE INTERFACE

The online DL/I-Entry language interface is entered when the application program issues a data call (step 2). Language interfaces are provided by the DL/I-Entry online system for COBOL, PL/I, and Assembler. Each DL/I-Entry call is translated to a common format and is then passed to the DL/I-Entry processor (step 3).

DL/I-ENTRY PROCESSOR

The online DL/I-Entry processor functions essentially the same as in the batch system. The processor performs all the tasks necessary to insert, retrieve, delete, or replace data in the data bases used by the application program (steps 5 through 10). The main differences of the online processor from the batch processor are:

- A special call is accepted for initiation purposes of the DL/I-Entry online system (step 4).
- An enqueue/dequeue module controls concurrent access requests to data bases by different tasks (step 8).
- The CICS/VS file control facility in conjunction with VSAM is used for data base access.
- In deadlock situations, a special call is accepted to release all data bases of a task (step 11).

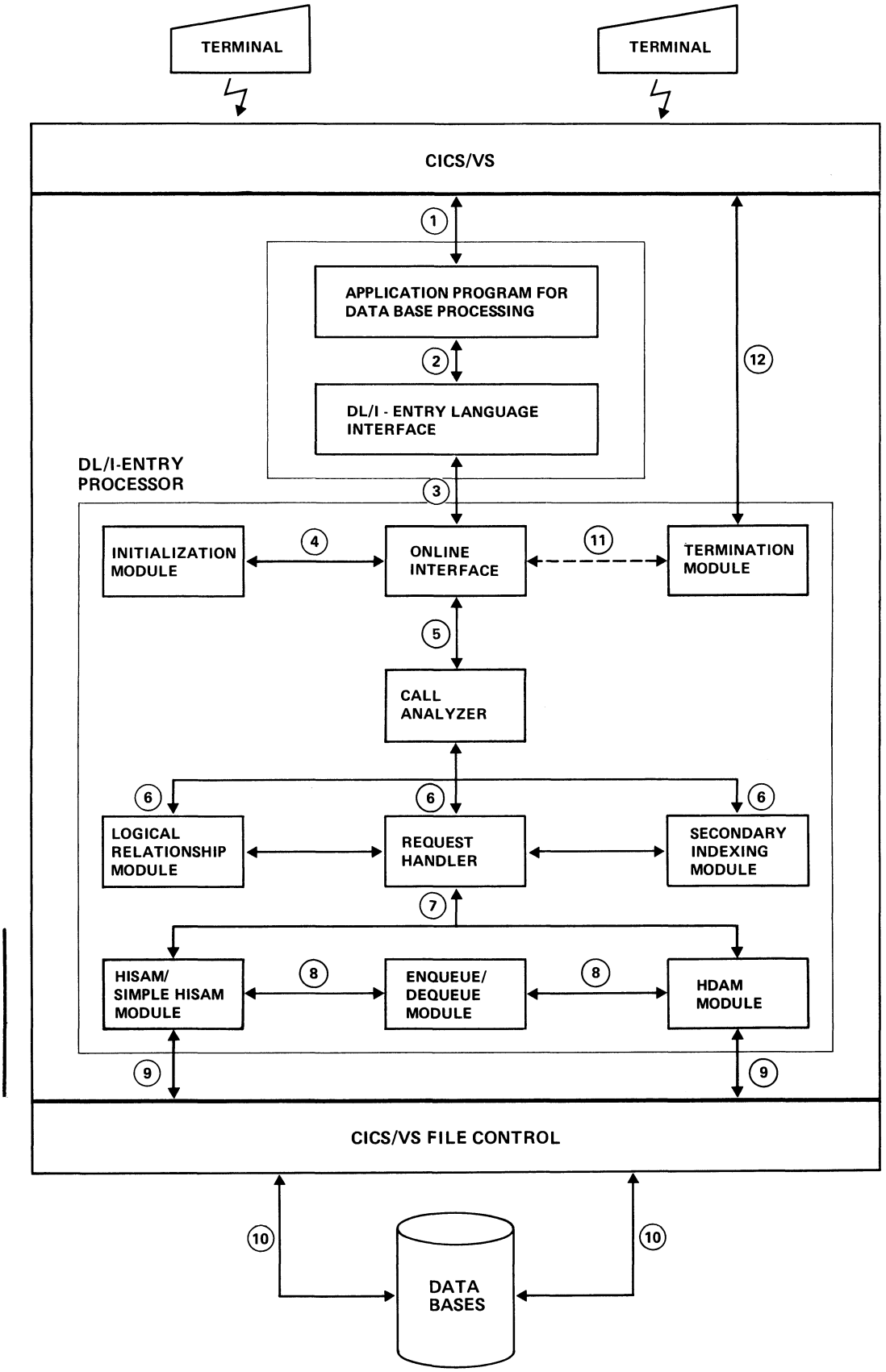


Figure 3. DL/I-Entry Online System

Utility Programs

Utility programs are provided with DL/I-Entry for generating data base descriptions (DBDs), for generating program specification blocks (PSBs), for producing backup copies of HISAM or HDAM data bases and reloading them, and for migrating from VANDL-1 to DL/I-Entry.

DATA BASE DESCRIPTION GENERATION

The DBD generation utility creates the control blocks that define to DL/I-Entry for each data base the data base name, its data structure, its data format, and the DL/I-Entry access method used.

PROGRAM SPECIFICATION BLOCK GENERATION

The PSB generation utility creates the control blocks that define to DL/I-Entry the data bases used, the type of data within each data base, and the operations allowed on each data base by a particular application program.

BACKUP/RELOAD

The HISAM backup utility unloads a HISAM data base into a sequential file. The HDAM backup utility unloads a HDAM data base into a sequential file. Both utilities also perform integrity checks on the data base and provide statistics about the data base structure. The sequential file can be kept for backup purposes, or it can be reloaded into a HISAM or HDAM data base using the reload utility. The reload utility reorganizes the data base which it reloads.

MIGRATION FROM VANDL-1 TO DL/I-ENTRY

Two migration utilities convert VANDL-1 (Vancouver Data Language One) data bases to DL/I-Entry data bases. The first migration utility runs as an application program under VANDL-1 and copies the VANDL-1 data base onto a sequential file. The second migration utility, which runs as a DL/I-Entry application program, loads the sequential file produced by the first migration utility into a DL/I-Entry data base.

This section goes beyond the general description of DL/I-Entry already given to discuss a number of technical considerations in additional detail. The information is of particular interest to persons responsible for planning the use of DL/I-Entry.

DL/I-Entry provides application program independence from access methods, physical storage organizations, and the characteristics of the devices on which the data of the application is stored. This independence is provided by a common symbolic program linkage and by data base descriptions external to the application program. Application program maintenance should thus be substantially reduced. See "Data Base User Interface" later in this section.

DL/I-Entry eliminates redundant data while assisting in the integration or sharing of common data. Most data has many interrelationships which lead to significant redundancy of data storage when conventional organizations and access methods are used. For example, manufacturing and engineering data is also useful to quality control. The storage organizations and access methods employed by DL/I-Entry facilitate data integration with a minimum of data redundancy. However, if analysis of a customer's data shows that all the data cannot be placed in a single common data base, DL/I-Entry allows the user the additional capability of physically structuring the data over more than one data base. Before DL/I-Entry, application programmers frequently were not able, nor did they have the time, to integrate other data with their own to eliminate redundancies without a major rewrite of the application programs involved.

An important capability of DL/I-Entry that protects each application of a multiapplication data base is the concept of data sensitivity. When making use of a DL/I-Entry data base, an application program may only access data which is predefined as sensitive. Each application using the data base can be sensitive to its unique subset of data. Where an application has defined sensitivity to a subset of the data within a data base, modification and addition of nonsensitive data do not affect the processing capability of the application. In addition, any application can be restricted as to the types of data base operations made upon its sensitive data.

Some Definitions

Here are some useful DL/I-Entry definitions.

- Segment. A data element of fixed length, containing one or more related data fields. A segment is the basic data element that interfaces between the application program and DL/I-Entry, and is the unit in terms of which the sensitivity of the application program is defined.

- Sensitivity. A means by which the user defines which subset of the data segments within the data base can be accessed by his application program and what operations (for instance, retrieve only) may be made upon the subset of the data base. This defines a logical view of a data base.

- Data base record. A set of hierarchically related, fixed-length segments of one or more segment types. As viewed by the application program, the data base record is always a hierarchical tree structure of segments.

- Data base. The major unit of data storage under DL/I-Entry. A set of data base records stored in a DI/I-Entry organization and for any one of the DL/I-Entry access methods.

No attempt is made at this point to relate the data base record or data base to a physical storage organization or access method technique. Such considerations are presented later in the section.

Data Independence

When processing traditional files instead of data bases, programs usually deal with a whole record at a time. Any change to a record structure involves changes to all the programs which use that record. Often this is true even for programs which do not actually handle the changed data. The programs are "data dependent."

DL/I-Entry, however, offers "data independence." This is achieved in two ways. First, programs handle only a segment at a time, rather than a whole record. This means that only changes to the information which a program actually handles necessitate changes in the program.

Second, programs are only aware of those segments to which they are sensitive. If a segment to which a program is not sensitive is added to a data base, deleted from it, moved within it, or changed to contain different data, no changes are needed for that particular program.

In addition, DL/I-Entry frees the user's data and program from considerations of access methods and organizations. The data base organization for DL/I-Entry is hierarchical sequential or hierarchical direct. To allow access to the hierarchical sequential organization, four DL/I-Entry access methods are provided: the hierarchical sequential access method (HSAM), the simple hierarchical sequential access method (simple HSAM), the hierarchical indexed sequential access method (HISAM), and the simple hierarchical indexed sequential access method (simple HISAM).

To allow access to the hierarchical direct organization, DL/I-Entry provides the hierarchical direct access method (HDAM). (Storage organization and access methods are discussed under "Data Base Organization and Access Methods" later in this section.) The application program interface with an organization type through the access methods is totally symbolic. The application program is typically insensitive to the particular organization, access method, or device.

Data Structures

Application programs written to use DL/I-Entry deal with data structures. A data structure is always a hierarchical structure of segments. Most data processing information, regardless of industry, can and should be viewed in a data structure. Personnel is chosen here for explanatory purposes because it is a familiar type of application. Additional examples are presented in the "Sample Applications" section.

The traditicnal manner of depicting data can be seen in Figure 4. This figure describes the physical makeup of the record as it might appear cn tape or on a direct access storage device. Each of the three divisions -- NAME, ADDRESS, and PAYROLL -- is called a field. These fields usually contain more basic data elements. For example, one of the data elements typically included in the PAYROLL field is rate of pay. In addition, the record might actually ccntain multiple ADDRESS and PAYROLL fields fcr a single NAME. This is typical if address and payroll histcry are desired.

Figure 4. Traditicnal Record Layout

This same data record appears in Figure 5 as a DL/I-Entry data structure. The NAME, ADDRESS, and PAYROLL fields are now considered segments cf information. Each segment of information is ccnsidered tc be made up of fields. Rate of pay would be a field within the PAYROLL segment.

The data structure in Figure 5 represents a hierarchical relationship. Data relaticnships described by this hierarchical picture have only cne segment at the first level in the hierarchy but may have multiple segments at subcrdinate levels in the hierarchy (for example, multiple ADDRESS and PAYROLL segments for cne NAME segment). Since each dependent segment in the hierarchy has only one parent or immediate superior segment, the representation is sometimes called a tree structure.

In Figure 5, the NAME segment with its associated ADDRESS and PAYRCLL segments constitutes a data base record.

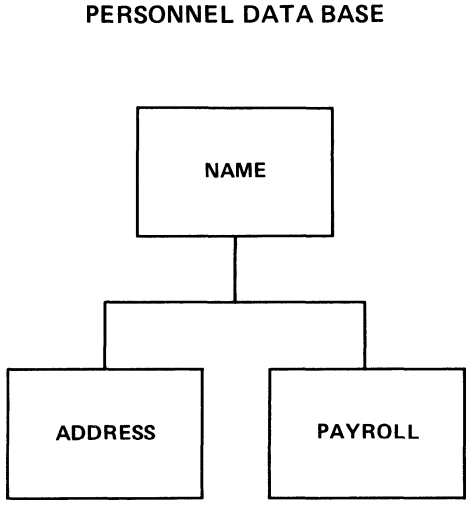


Figure 5. Hierarchical Record Laycut

Through the concept of program sensitivity, DL/I-Entry allows a program to be written in such a manner that it sees only those segments of information that are relevant to the processing being performed. For example, a payroll program could be written to see only the NAME and PAYROLL segments of the data base record shown in Figure 5. The program need not be aware of the existence of the ADDRESS segment.

The DL/I-Entry data base capabilities allow for handling hierarchically related data structures of considerable variance. The maximum number of segment types is limited to 63 per data base record. A maximum of 15 segment levels can be defined in a data base record.

Figure 6 represents an example of a data structure for a skill data base. The structure consists of two segment types, SKILL and NAME. The data structure shows that SKILL is the root segment, or highest segment in the hierarchy. The NAME segment is a dependent segment of SKILL. A dependent segment relies on some higher level segment for its full meaning or identification. Since the number of employees (NAME segments) may vary from one skill classification (SKILL segment) to the next, it is necessary for data base records to vary in size according to the number of segments occurring in the hierarchy for the data structure.

SKILL DATA BASE

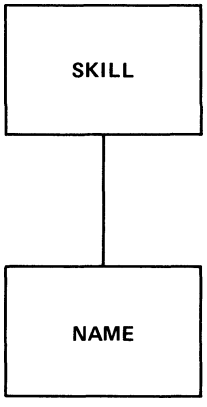


Figure 6. Data Structure

Figure 7 shows a specific data base record from the data structure described in Figure 6. The root segment contains the specific skill of artist. There are multiple NAME segments. In this case, there are three employees who have some capability as artists: Adams, Jones, and Smith. Although the data base can be described as having two segment types, this particular data base record consists of four segments because of the multiple occurrences of the NAME segment.

The segment types immediately above and below a given segment type are called the parent and child segment types, respectively. In Figure 7, the SKILL segment for artist is the parent of three NAME segments. Each NAME segment is a child segment of the SKILL segment. All occurrences of a particular segment type within a data base record are called twin segments. The three NAME segments for Jones, Smith, and Adams are twin segments.

SKILL DATA BASE

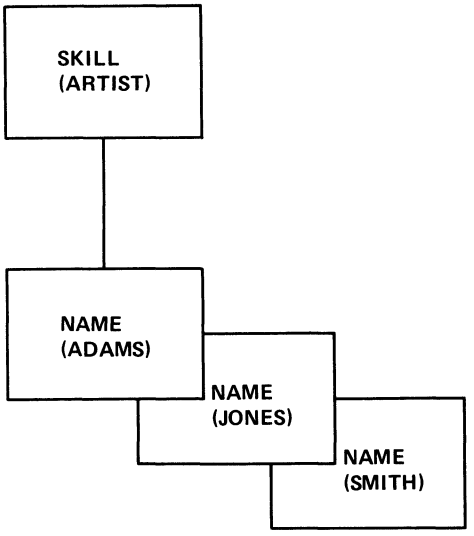


Figure 7. Data Base Record Structure (Derived from Figure 6)

Logical Relationships

Through the use of logical relationships, logical data structures may be constructed for use by applications where the logical data structure does not exist as a single physical data base record. A logical data structure may be composed of segments from one or more physical data base records connected by logical relationships. Logical relationships between segments in different physical data base records allow for separation of logical data structures from physical storage.

A logical relationship may exist between segments in one or more HISAM data bases, one or more HDAM data bases, or a combination of HISAM and HDAM data bases. Symbolic pointers are used to establish a logical relationship.

LOGICAL RELATIONSHIPS BETWEEN PHYSICAL DATA BASES

Logical relationships serve as the means by which DL/I-Entry can store a physical segment once and provide access to that data segment through multiple paths, thus avoiding redundant data.

If a logical relationship as shown in Figure 8 is established from the root segment SKILL in the skill data base to the root segment NAME in the personnel data base, the two physical data bases are interrelated. The logical relationship is actually constructed with the help of the dependent segment REFERENCE NAME. Rather than duplicating the NAME segment in both data bases, the skill data base now contains only a reference to the NAME segment stored in the personnel data base.

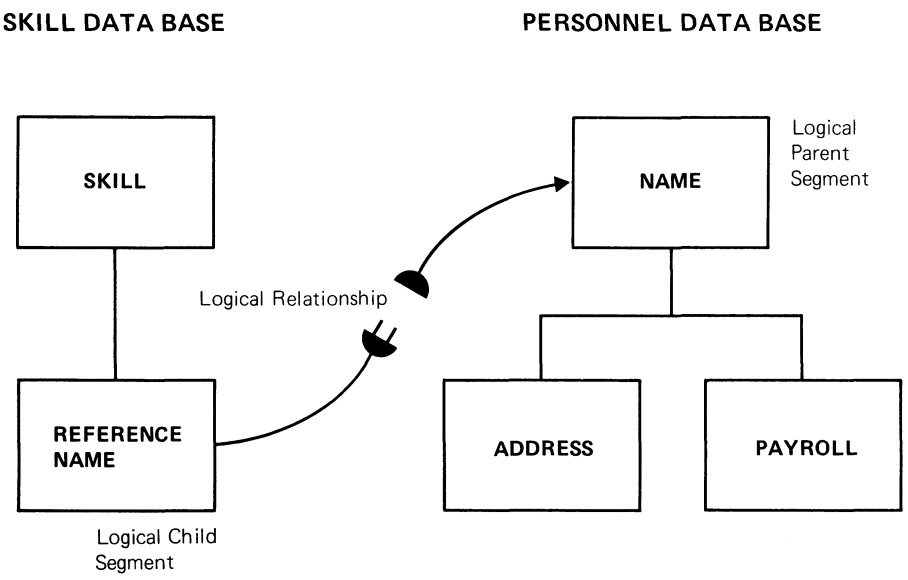


Figure 8. Unidirectional Logical Relationship Between Physical Data Bases

The segment REFERENCE NAME providing the logical relationship path from the SKILL segment to the NAME segment is called a logical child segment. The segment NAME which is related to the logical child segment REFERENCE NAME is called a logical parent segment. A logical parent segment must always be a root segment. In this example, the logical child segment REFERENCE NAME has a logical parent segment NAME and a physical parent segment SKILL.

The logical relationship path runs only in the direction from the logical child segment towards the logical parent segment, and is called a unidirectional logical relationship.

Several occurrences of the REFERENCE NAME segment under different SKILL segments may relate to the same logical parent segment NAME. Each REFERENCE NAME segment may provide data, called intersection data, unique to the relationship between a particular SKILL and NAME segment.

Having established a unidirectional logical relationship between the two physical data bases, a new logical data structure can be defined as shown in Figure 9. Note that the REFERENCE NAME segment and NAME segment are concatenated and presented to an application program as one segment.

Using the logical relationship, data in both physical data bases can be accessed and modified. For example, if the key field of a given SKILL segment is known, the segments REFERENCE NAME-NAME, ADDRESS, and PAYROLL can be examined.

All occurrences of REFERENCE NAME-NAME, ADDRESS, and PAYROLL segments under a particular SKILL segment become a logical data base record. The segments in this logical data base record may be accessed by an application program in the same manner as segments in a physical data base record.

As shown in Figure 8, the logical child segment REFERENCE NAME establishes a unidirectional logical relationship path from the SKILL segment to the NAME segment. If a logical relationship path also should

exist in the other direction -- from the NAME segment to the SKILL segment -- a second logical child segment must be introduced. In Figure 10 the dependent segment REFERENCE SKILL is used to construct the second logical relationship.

SKILL/PERSONNEL LOGICAL DATA BASE

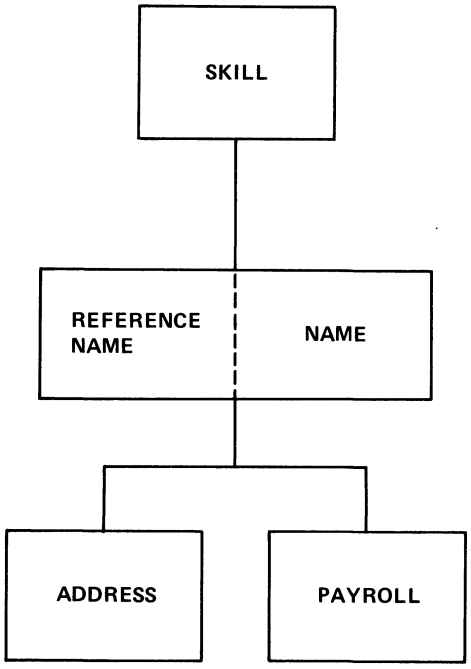


Figure 9. Combined Data Structure Viewed from the SKILL Segment

SKILL DATA BASE **PERSONNEL DATA BASE**

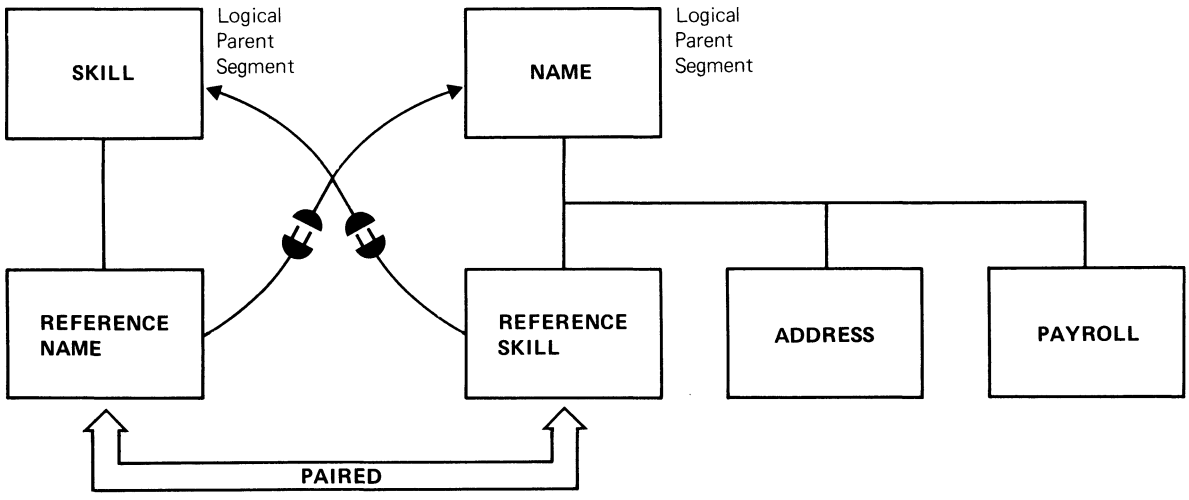


Figure 10. Bidirectional Logical Relationship Between Physical Data Bases

The logical relationship shown in Figure 10 is called a bidirectional logical relationship, whereby the REFERENCE NAME and REFERENCE SKILL segments are paired. Both logical child segments have a logical and a physical parent -- for instance, REFERENCE SKILL has the physical parent NAME and the logical parent SKILL.

The logical relationship shown in Figure 10 makes possible another logical structure -- in addition to that which has already been shown in Figure 9. The new structure is that as viewed from the NAME segment, shown in Figure 11. Here the REFERENCE SKILL and SKILL segments are concatenated and presented to an application program as one segment. For this example PAYROLL, not of interest, has been omitted.

PERSONNEL/SKILL LOGICAL DATA BASE

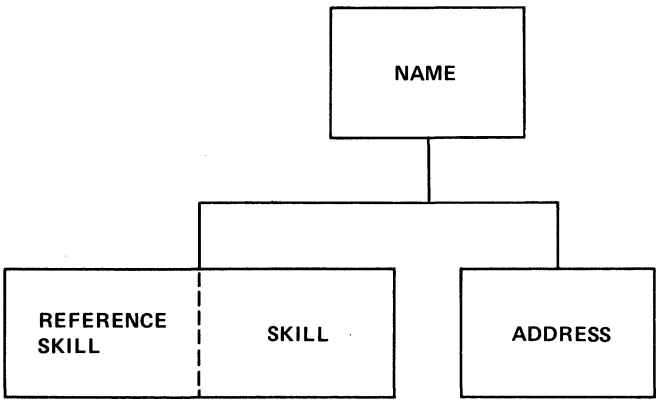


Figure 11. Combined Data Structure Viewed from the NAME Segment

Specific occurrences of REFERENCE NAME and REFERENCE SKILL are paired, whereby they have the same intersection data. Figure 12 shows an example of how a specific REFERENCE NAME occurrence for Jones is paired with REFERENCE SKILL for artist.

DL/I-Entry is responsible for ensuring the maintenance of this paired bidirectional logical relationship. If one direction of the relationship is altered by segment insertion or deletion, DL/I-Entry automatically maintains the other direction. For instance, if the segment REFERENCE SKILL for artist in Figure 12 were deleted, REFERENCE NAME for Jones would automatically be deleted.

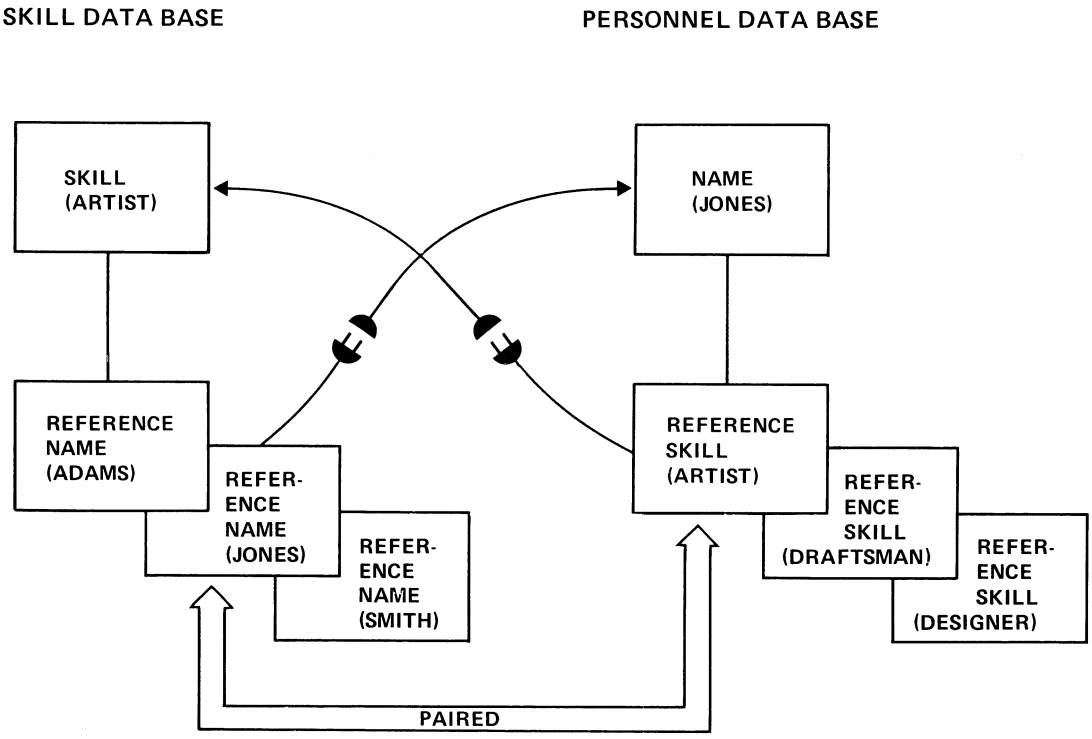


Figure 12. Pairing Specific Segment Occurrences

Some general rules regarding the segments which participate in a logical relationship are:

1. A logical child segment must always have a logical parent and a physical parent -- therefore it cannot exist as a root segment. However, it may exist at any dependent level in the physical hierarchy.

2. A logical child segment can have only one logical parent and one physical parent.

3. Only one logical child segment is permitted in a given branch of the tree-like hierarchical data structure.

4. A logical child segment can have no dependent segments.

5. A logical parent segment must be a root segment.

6. A logical parent segment may have one or multiple logical child segment types. Each logical child segment type depends on one logical relationship.

7. Only one logical relationship can be used to define a given branch of the tree-like hierarchical logical data structure.

Secondary Indexing

Secondary indexing is a special way of retrieving data base records (HISAM and HDAM only). Instead of retrieving them based on the sequential order of the key of the root segment, they are retrieved based on the sequential order of the secondary index.

Secondary indexing is made possible by creating a new data base called the index data base. For example, in Figure 13 the segment X (which might be a particular job title) would be the index_pointer_segment of the index data base. It contains the data used to index the index_target_segment, NAME. The index target segment must be the root segment of the indexed data base. The index_source_segment, DEPARTMENT, is the source of the search data from which a secondary index is created. The index source segment may be a root or a dependent segment (in the example it is a dependent segment). The secondary index has the sequential order of the search data values. An application program can thus retrieve occurrences of the index target segment based on the sequential order of information contained in the index source segment.

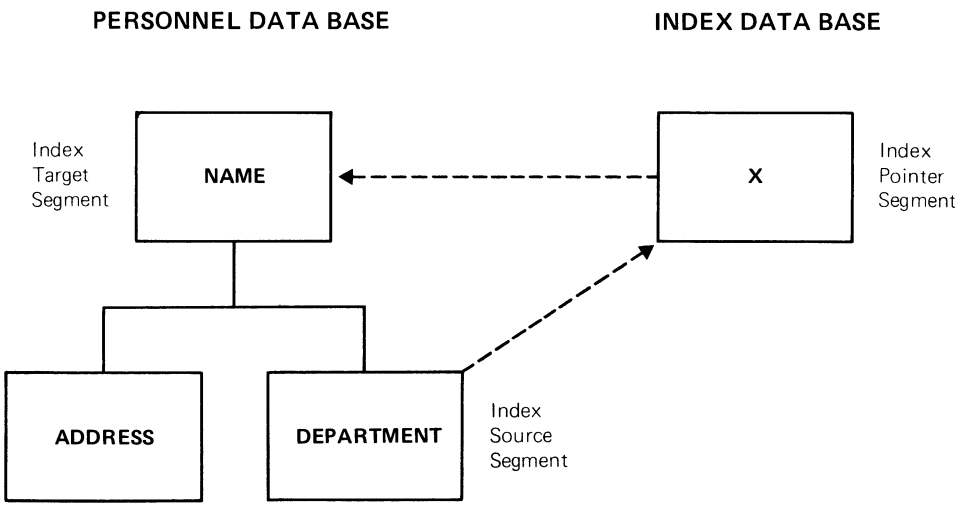


Figure 13. Secondary Index Data Base

For each occurrence of a DEPARTMENT segment, DL/I-Entry would automatically maintain the index pointer segment in the index data base, as shown in Figure 14.

More than one secondary index can be based on the same index source segment type, and the indexes may use search data which is different, overlapping, or the same.

The advantages offered by secondary indexing can be seen by considering, for instance, an application program which needs to retrieve all NAME segments within a single department. This is done much faster using the secondary processing sequence of the index data base rather than retrieving each record in sequence from the original data base in order to check for the desired department number.

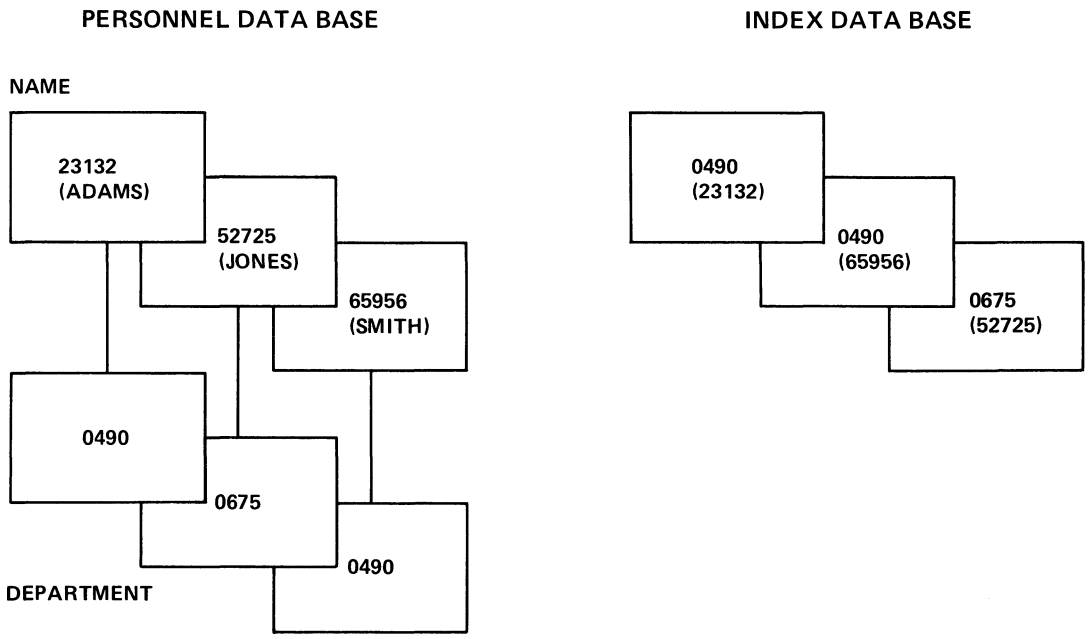


Figure 14. Secondary Index Data Base Example

Low-Level Codes

In a manufacturing industry, low-level codes placed in the root segments of a parts data base can be used to identify the components or materials which make up a finished product. DL/I-Entry offers a low-level code/continuity check feature which batch application programs written in COBOL, PL/I, or Assembler language can call to generate, check and update such low-level codes. Programs which use the feature are upwards source program compatible with DL/I DOS/VS Version 1.1 and subsequent versions.

Data Base User Interface

Application programs (except those written in RPG II) request DL/I-Entry services using a CALL statement. The DL/I-Entry CALL statement is compatible with VANDL-1 and DL/I DOS/VS. The CALL statement can request DL/I-Entry to:

- Retrieve a unique segment (GET UNIQUE).
- Retrieve the next sequential segment (GET NEXT).
- Retrieve the next sequential segment within an established parent (GET NEXT WITHIN PARENT).
- Replace the data in an existing segment (REPLACE).
- Delete an existing segment (DELETE).
- Insert a new segment (INSERT).

The RPG II programmer has two possibilities for requesting DL/I-Entry services. First, he can control them "implicitly," without making explicit requests. With this approach he can retrieve DL/I-Entry data

base records sequentially, and can update them or produce output to other files. He cannot, however, process segments non-sequentially, or add segments, or delete them.

By using "explicit" input and output, however, the RPG II programmer can process a data base both sequentially and nonsequentially, and can add and delete segments as well as being able to retrieve and update them. He requests services using RPG II READ and EXCPT statements. The DL/I-Entry data bases are defined in his program as special files.

A common symbolic program linkage handles the COBOL, RPG II, PL/I, and Assembler languages. The external data base description (DBD) gives the data structure and physical data organization of each data base to DL/I-Entry. Using these techniques, it is possible to physically reorganize established data bases in a timely manner without modifying application programs. Input/output operations and associated system control blocks are not compiled into the application programs.

Each DBD is created from user-provided statements which define the data structure and physical organization of the data base. These statements are read by the DL/I-Entry DBD generation utility, which creates and stores the DBD. The DBD then provides DL/I-Entry with a "map" from the structure of the data base used in the application program to the physical organization of the data used by DOS/VS data management. Other application data can be added to this data base without necessitating changes to the application programs which use the data. The concept of the DBD reduces application program maintenance caused by changes in the data requirements of the applications.

Data Base Organization and Access Methods

DL/I-Entry supports two basic physical storage organizations, hierarchical_sequential and hierarchical_direct. The hierarchical sequential organization provides the basis for four access methods:

- Hierarchical sequential access method (HSAM)
- Simple hierarchical sequential access method (simple HSAM)
- Hierarchical indexed sequential access method (HISAM)
- Simple hierarchical indexed sequential access method (simple HISAM).

Simple HSAM and simple HISAM process root-only data bases.

The hierarchical direct organization provides the basis for the hierarchical direct access method (HDAM).

Segments that represent one data_base_record (that is, a physical hierarchical tree structure) are related either by physical juxtaposition (HSAM or simple HSAM), by symbolic pointers (HISAM or simple HISAM), or by relative byte addresses (HDAM).

HSAM and simple HSAM are used for sequential storage and access on tape or direct access storage devices. The DOS/VS sequential access method (SAM) provides the data management services for these access methods.

HISAM and simple HISAM are used for indexed sequential access to the hierarchical sequential organization. The DOS/VS virtual storage access method (VSAM) provides the data management services. A HISAM or simple HISAM data base comprises one VSAM key sequenced file. Each data base record starts with a root segment contained in a VSAM record. As many dependent segments of that root segment as can be accommodated are

placed in the VSAM record (for HISAM only). Overflow dependent segments for that root segment are stored in additional VSAM records. Symbolic pointers relate all physical records for one HISAM data base record.

HDAM is used for direct storage and access. The DOS/VS virtual storage access method (VSAM) provides the data management services. The purpose of HDAM is to give fast, random access to data. A HDAM data base comprises a VSAM entry sequenced file split into a root addressable area and an overflow area. The root addressable area contains the beginnings (the root segments and some dependent segments) of data base records. In a well designed data base this is the information most often processed. The overflow area contains the continuations (the remaining dependent segments) of the data base records. This is the information less often processed.

Logical relationships and secondary indexing are possible only with HISAM and HDAM.

Segment Definition and Format

The foregoing discussion has introduced some of the concepts used in the physical storage of data. The term "data base record" was used. The data base record is represented as a simple hierarchical tree structure of related segments.

Each segment, regardless of the access method used, is composed of two parts -- the prefix and the data.

The format of the prefix for any segment type is unique and is determined by the data base organization. The use of the segment prefix is controlled entirely by DL/I-Entry. An application program need not be concerned about the presence or the format of the segment prefix.

The data portion of a segment may optionally begin with a key field to control the physical sequence of occurrences of that segment type. Only the data portion of a segment is passed between an application program and DL/I-Entry.

Online Processing Capability

DL/I-Entry functions may be extended to an online environment through CICS/VS. CICS/VS, a general-purpose data base/data communication system, functions as an interface between DL/I-Entry and application programs written in COBOL, PL/I, or Assembler language. CICS/VS makes it possible for more than one application program to simultaneously work with the same DL/I-Entry data bases.

In addition to serving as an interface with DL/I-Entry, CICS/VS offers the application programmer a user exit facility for optional processing routines, and macros for requesting CICS/VS services. For more information about CICS/VS, refer to Customer Information Control System/DOS/VS General Information, GH20-7028.

User Responsibilities

Installation personnel must be adequately trained in using DL/I-Entry. Training can take the form of classroom instruction or self-study using the DL/I-Entry documentation. In addition, if the user intends to run DL/I-Entry in an online environment, the installation personnel must be sufficiently trained in using CICS/VS.

The user must either have an existing VANDL-1 data base, or else have collected data from which he wants to build a new DL/I-Entry data base. The time required to convert an existing VANDL-1 data base to run on DL/I-Entry is approximately one day per data base. Users not coming from a VANDL-1 environment will require slightly more time to generate a data base, since the input for DBD and PSB generation must be written for the first time. Once this input is written, however, generation should require less than one day per data base.

Application programs written for VANDL-1 require no conversion. Other application programs require modification in order to use the DL/I-Entry CALL statement and to follow DL/I-Entry conventions.

The steps a system programmer must perform in order to migrate from VANDL-1 to DL/I-Entry are summarized in Figure 15. The steps the system and application programming personnel of a new DL/I-Entry installation must perform to install DL/I-Entry are summarized in Figure 16.

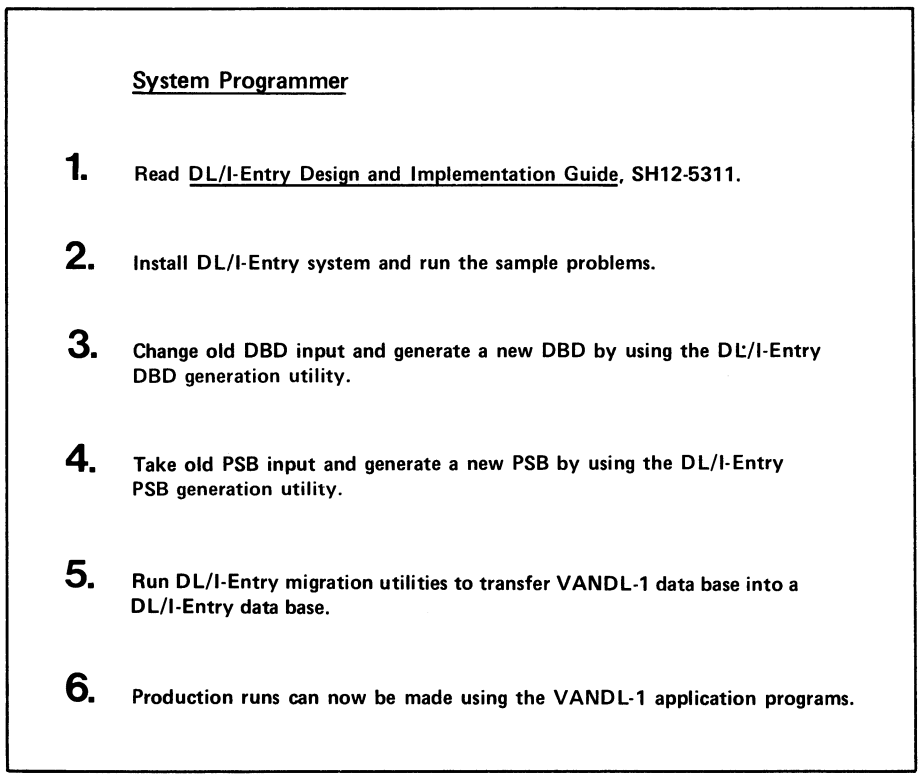


Figure 15. Migration from VANDL-1 to DL/I-Entry

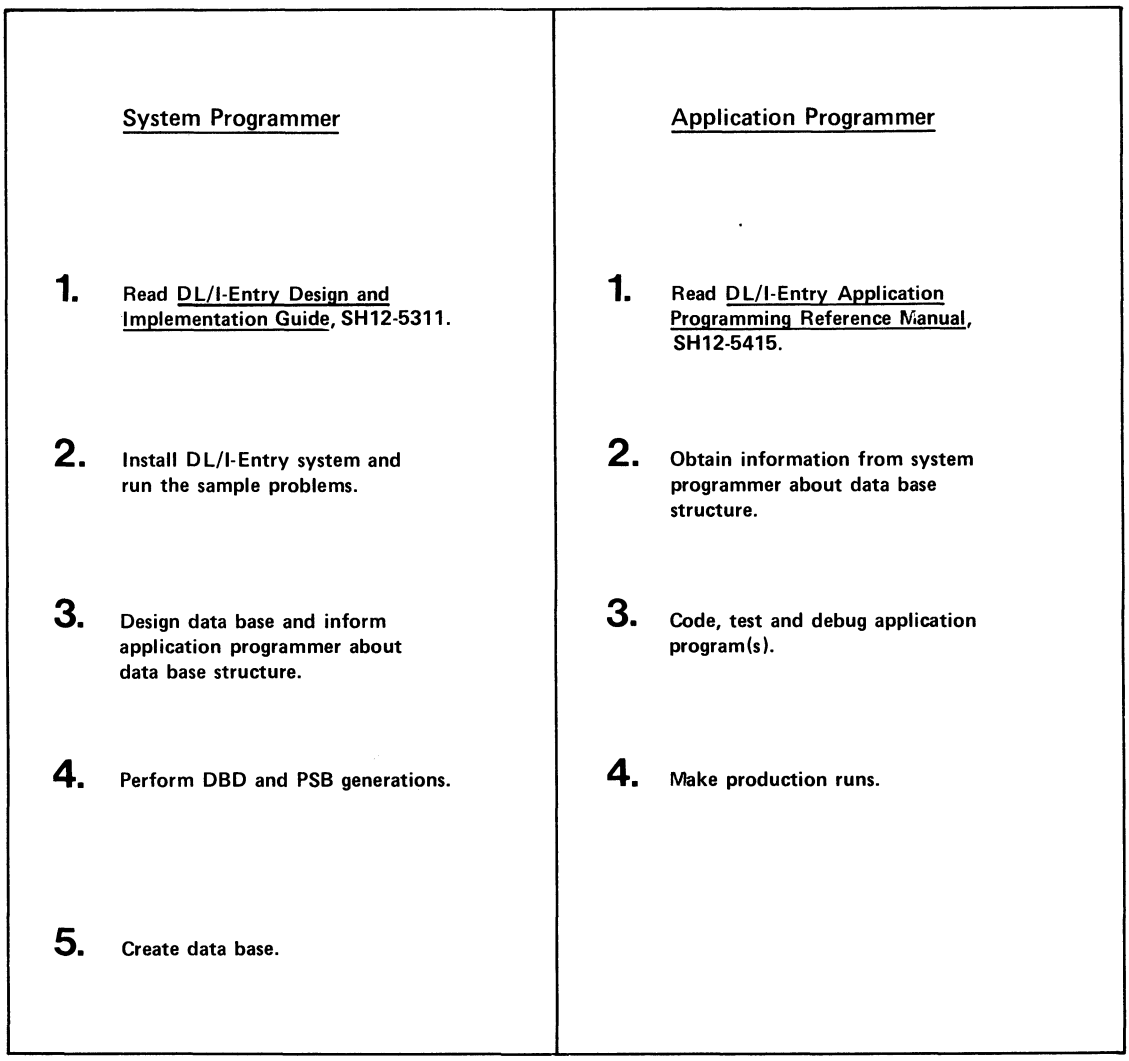


Figure 16. Installing DL/I-Entry at a New Installation

Programming Systems

DL/I-Entry will run exclusively on the System/370 under DOS/VS. DL/I-Entry is also supported under DOS/VS running under the control of VM/370. DL/I-Entry is written in Assembler language and uses the virtual storage access method (VSAM) and sequential access method (SAM) data management facilities. The following components of the DOS/VS SCP (5745-010) are required:

- Control and service programs
  - Initial program loader
  - Supervisor
  - Job control
  - Linkage editor
  - Relocating loader
  - Librarian
- Assembler
- Data management
  - Sequential access method (SAM)
  - Virtual storage access method (VSAM).

In addition to the above DOS/VS components, the user may require the following:

- DOS/VS COBOL Compiler and Library, 5746-CE1.
- Full ANS COBOL V3 Compiler, 5736-CE2, and Full ANS COBOL Library, 5736-LM2.
- PL/I Optimizing Compiler, 5736-PL/I.
- PL/I Resident Library, 5736-LM4.
- PL/I Transient Library, 5736-LM5.
- RPG II Compiler, 5736-RG1.

For online execution of DL/I-Entry, the Customer Information Control System/DOS/VS (CICS/VS), 5746-XX3, is required.

System Requirements

A minimum machine configuration for DL/I-Entry is outlined below. The differing requirements for a DL/I-Entry batch system and for a DL/I-Entry system having an online processing capability are identified.

System Function	Units
Processing Unit	An IBM System/370 Model 115, 125, 135, 145, or 158. The DL/I-Entry batch system requires 10K bytes of real storage and 32K bytes of virtual storage for execution. The DL/I-Entry online system requires 12K bytes of real storage and 34K bytes of virtual storage for execution.
Direct Access	For system libraries and working storage space: any direct access storage devices supported by DOS/VS. Minimum space for system use and maintenance: Batch System — 20 cyl IBM 2316 Disk Pack or equivalent Online System — 40 cyl IBM 2316 Disk Pack or equivalent For DL/I-Entry data base storage, within the capability and restrictions of DOS/VS support by the virtual storage access method and sequential access method: IBM 2314/2319 — Direct Access Storage Facility IBM 3330/3333 — Disk Storage IBM 3340 — Direct Access Storage Facility
Other I/O	One IBM 2560 Card Reader/Punch (or equivalent), one IBM 3203 Printer (or equivalent), and (required for online only) any terminal supported by CICS/VS.

Estimated Storage Requirements

The following is an example of the storage requirements for a DL/I-Entry batch application program which utilizes the basic DL/I-Entry functions and accesses two HISAM or HDAM data bases. The total virtual storage requirements are 94K bytes. The total real storage requirements are 20 pages (40K bytes). This is an estimate of the lowest number of page frames required for reasonable performance without "thrashing."

	Virtual (bytes)	Real (pages)	Comments
DL/I-Entry code	34K	5	Only 5 pages (10K) have heavy usage and therefore will always be in real storage.
DL/I-Entry working storage	2K	1	
DL/I-Entry control blocks	2K	1	Approximately 1K for each HISAM or HDAM data base.
VSAM modules: record management (OPEN/CLOSE processing 142K)	28K	7	Approximately 60% of VSAM is "once per job" code.
VSAM control blocks	4K	2	2K for each VSAM data set.
VSAM buffers and DL/I-Entry I/O areas	8K	2	4 buffers for each data base. Control interval size is 1024 bytes (1K).
Subtotal (execution mode)	78K	18	18 pages = 36K bytes.
Application program	16K	2	
Total (execution mode)	94K	20	20 pages = 40K bytes.

Sample Applications

This section presents several data base application examples. Each example shows the data base structures used for a specific application, and the types of inquiries which the data bases could be expected to handle. Although only three sample applications are given, these samples are constructed in a general way, so that the techniques they illustrate apply to a wide variety of other applications.

Distribution Example

A sample application of DL/I-Entry in distribution uses these three data bases:

- Product data base
- Customer index data base
- Supplier index data base.

These data bases are shown in Figure 17. In the following text, the individual fields which might be contained in the segments are described.

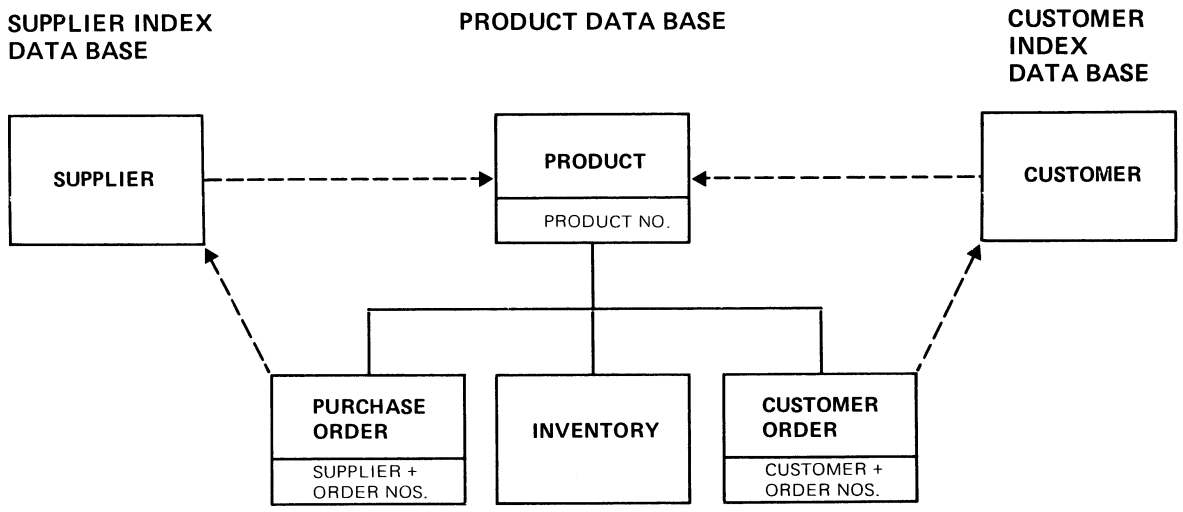


Figure 17. Distribution Data Bases

PRODUCT DATA BASE

The product data base is made up of four segment types on two hierarchical levels. The records are stored in ascending sequence based upon the product number.

The product number is the key of the root segment, the PRODUCT segment. In addition this segment contains:

- The product description.
- Overall sales information (such as sales values, sales periods, statistics).

The PURCHASE ORDER segment contains:

- The supplier number followed by the order number, used together as the segment key.
- The amount on order and the cost.
- The delivery date and address.
- Confirmation status.

The INVENTORY segment contains:

- The quantity on hand and stock location (in periods).
- The quantity in production (in periods).
- The quantity on order (in periods), broken down into the quantity on customer order and the quantity on purchase order.

The CUSTOMER ORDER segment contains:

- The customer number followed by the order number, used together as the segment key.
- The amount on order and the cost.
- The delivery date and address.
- Confirmation status.

SUPPLIER INDEX DATA BASE

The supplier index data base is created and maintained by DL/I-Entry. It is made up of a single segment type, the SUPPLIER segment. This segment contains information to relate, to a given supplier and order number, the corresponding product number(s).

CUSTOMER INDEX DATA BASE

The customer index data base is created and maintained by DL/I-Entry. It is made up of a single segment type, the CUSTOMER segment. This segment contains information to relate, to a given customer and order number, the corresponding product number(s).

INQUIRIES USING THESE DATA BASES

Any application program accessing the product data base may use one of three possible addressing schemes. That is, a particular PRODUCT segment may be accessed using either:

1. The product number (key of the root segment).

2. The supplier number plus the order number of one of the PURCHASE ORDER segments dependent on the PRODUCT segment.

3. The customer number plus the order number of one of the CUSTOMER ORDER segments dependent on the PRODUCT segment.

For sequential processing, each of these addressing schemes defines a different processing sequence for the product data base.

If the supplier index data base were used to provide the secondary processing sequence for the product data base, all products ordered from a particular supplier could be retrieved and updated for purchase planning. The quantities ordered from the supplier, and the delivery dates, could be determined and perhaps modified, and the purchase orders could be checked.

Another type of processing which could be performed with the product data base, based on the secondary processing sequence indicated by the supplier index data base, would be to control purchase orders. Production changes necessary because of late delivery could be determined, as could the effect on customer orders for a particular product.

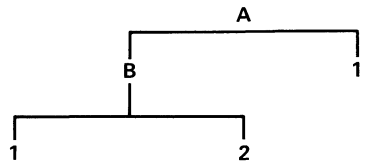
Using the other index data base, the customer index data base, to provide the secondary processing sequence for the product data base, the list of products for a particular customer could be obtained or modified, for use by sales or to aid in planning shipment.

In this way, a number of inquiries could be made using the index data bases. These inquiries will perform considerably faster than searching the product data base following its primary sequence established by the product number to select the desired information.

Bill of Materials Example

In manufacturing, a bill of materials for an assembly can be prepared when the component parts of each particular item of the assembly, and the location where the parts are used in the item, are known. For a sample item A, Figure 18 shows what might be the data structure for the bill of materials. In addition, this figure shows the DL/I-Entry data base which could represent the bill of materials.

BILL OF MATERIALS DATA STRUCTURE



ITEM DATA BASE

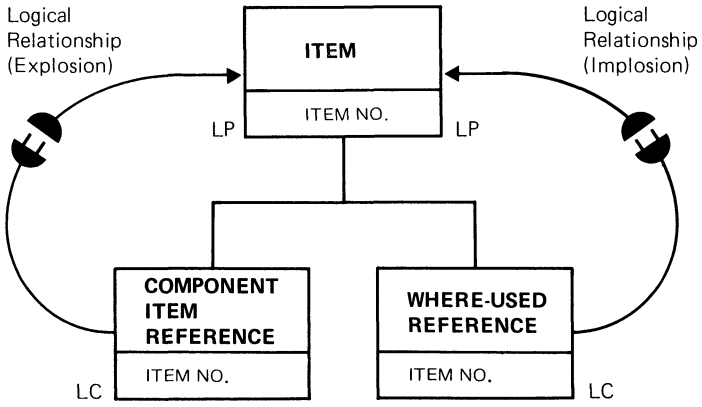


Figure 18. Bill of Materials Data Structure and Item Physical Data Base

The data base shown, the item data base, is made up of three segment types on two hierarchical levels. The ITEM segment contains the item number as segment key, the item name, and the item description. The COMPONENT ITEM REFERENCE segment contains only a key, the item number. The WHERE-USED REFERENCE segment also contains only a key, the item number. The latter two segments are used to establish logical relationships with the ITEM segment. They may optionally contain intersection data, that is, data unique to the relation of a certain occurrence of these segments and the corresponding occurrence of the ITEM segment.

The logical data base structure, which is derived from the data base shown in Figure 18 by using the logical relationships between the segments, is shown in Figure 19. This is the way an application program would view the data.

ITEM LOGICAL DATA BASE

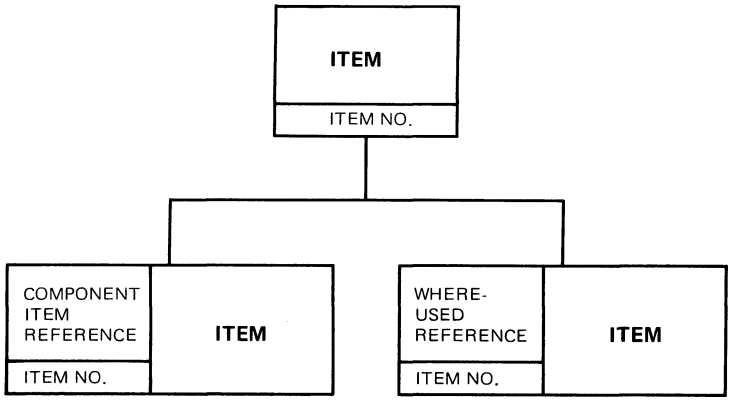


Figure 19. Item Logical Data Base

INQUIRIES USING THE LOGICAL DATA BASE

A one-level bill of materials (explosion) could be produced by proceding from an ITEM segment to its COMPONENT ITEM REFERENCE segment, which is logically related to the corresponding ITEM segment. By repeating this step for each item in an assembly, a complete explosion bill of materials for the entire assembly may be developed. An example is shown in Figure 20. In this example, letters are used to represent items which can be further subdivided, while numbers are used to represent items which cannot be further subdivided.

In step 1 of Figure 20, ITEM segments B and 1 are found as components of ITEM segment A, by means of the logical relationships from the COMPONENT ITEM REFERENCE segments B and 1 to the ITEM segments B and 1, respectively. And in step 2, ITEM segment B is then broken down into its ITEM segments 1 and 2 also through logical relationships.

One-level where-used information (implosion) could be produced by proceding from an ITEM segment, through its WHERE-USED REFERENCE segments, to the logically related larger ITEM segments. The process is similar to that for explosion, and is shown in Figure 21 for ITEM segments 1 and 2 in the assembly of ITEM A. ITEM 1 is used in both ITEM A and B of the assembly, and ITEM 2 is used in ITEM B. ITEM B is further used in ITEM A.

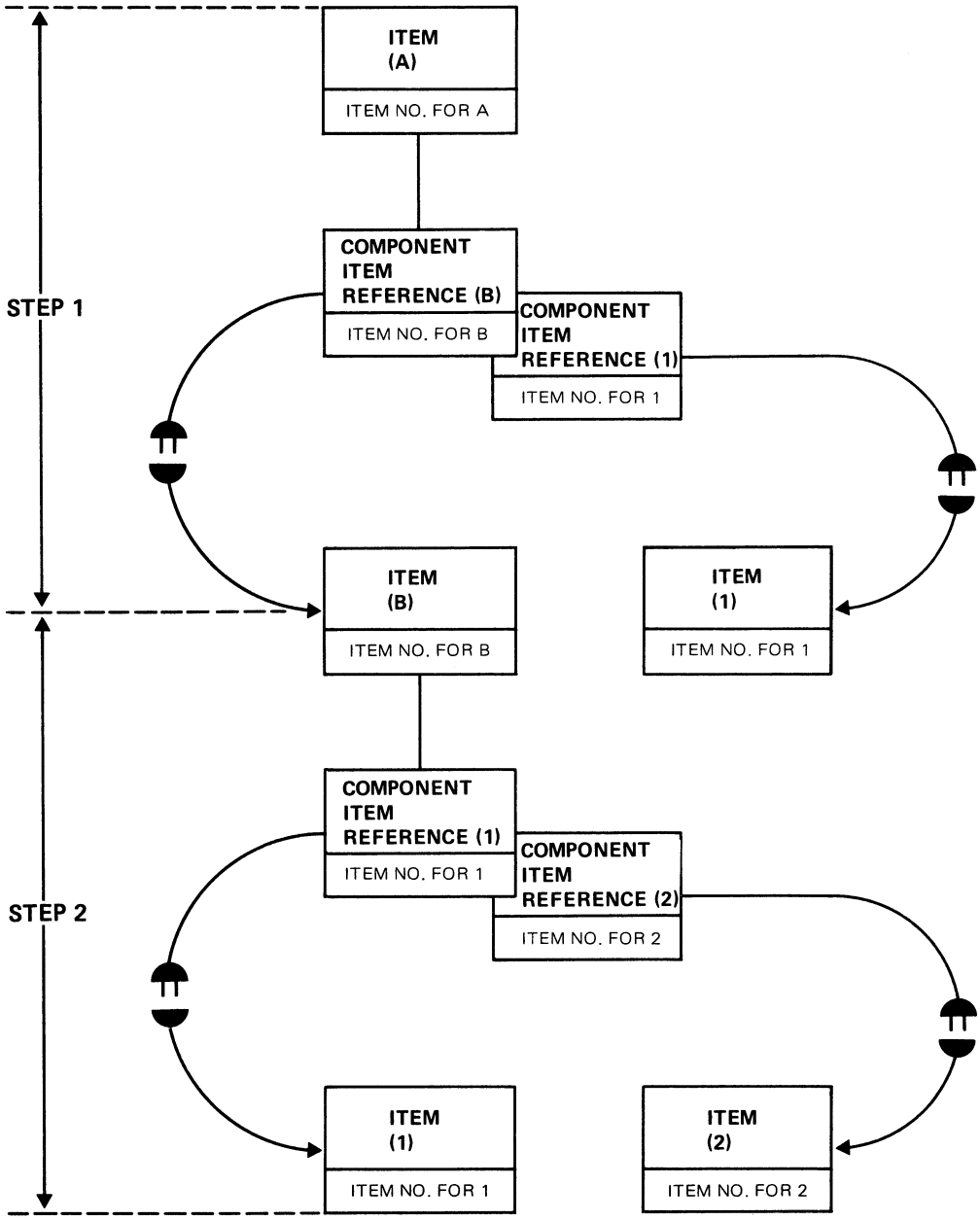


Figure 20. Explosion Bill of Materials Example

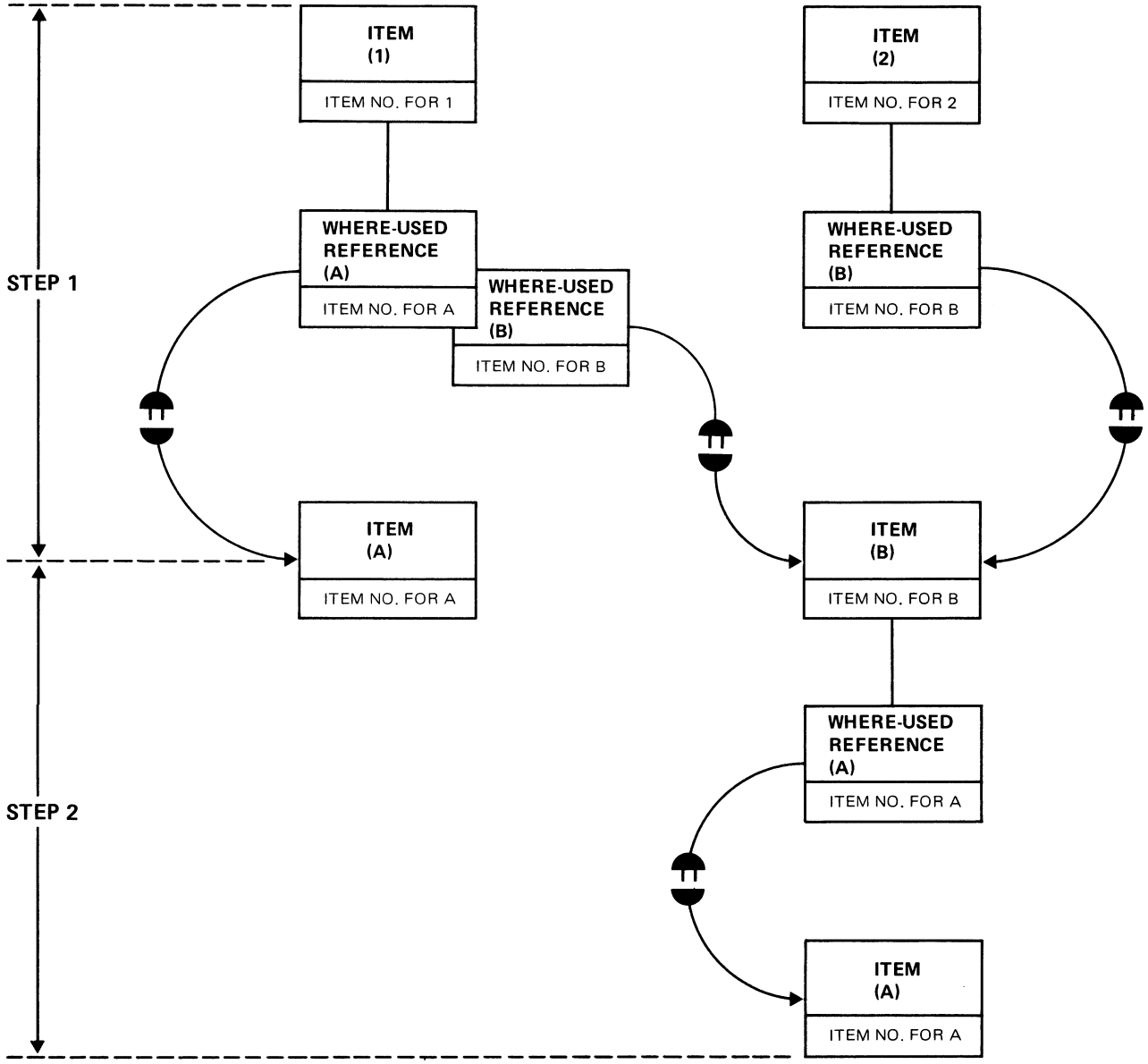


Figure 21. Implosion Bill of Materials Example

Process Industry Example

A sample application of DL/I-Entry in the process industry -- in a steel mill -- uses these four data bases:

- Customer data base
- Order line data base
- Product data base
- Production order data base.

These data bases contain all the information necessary for order entry applications, and for production planning and control applications. The data bases are shown in Figure 22. In the following text, the individual fields contained in the segments are described.

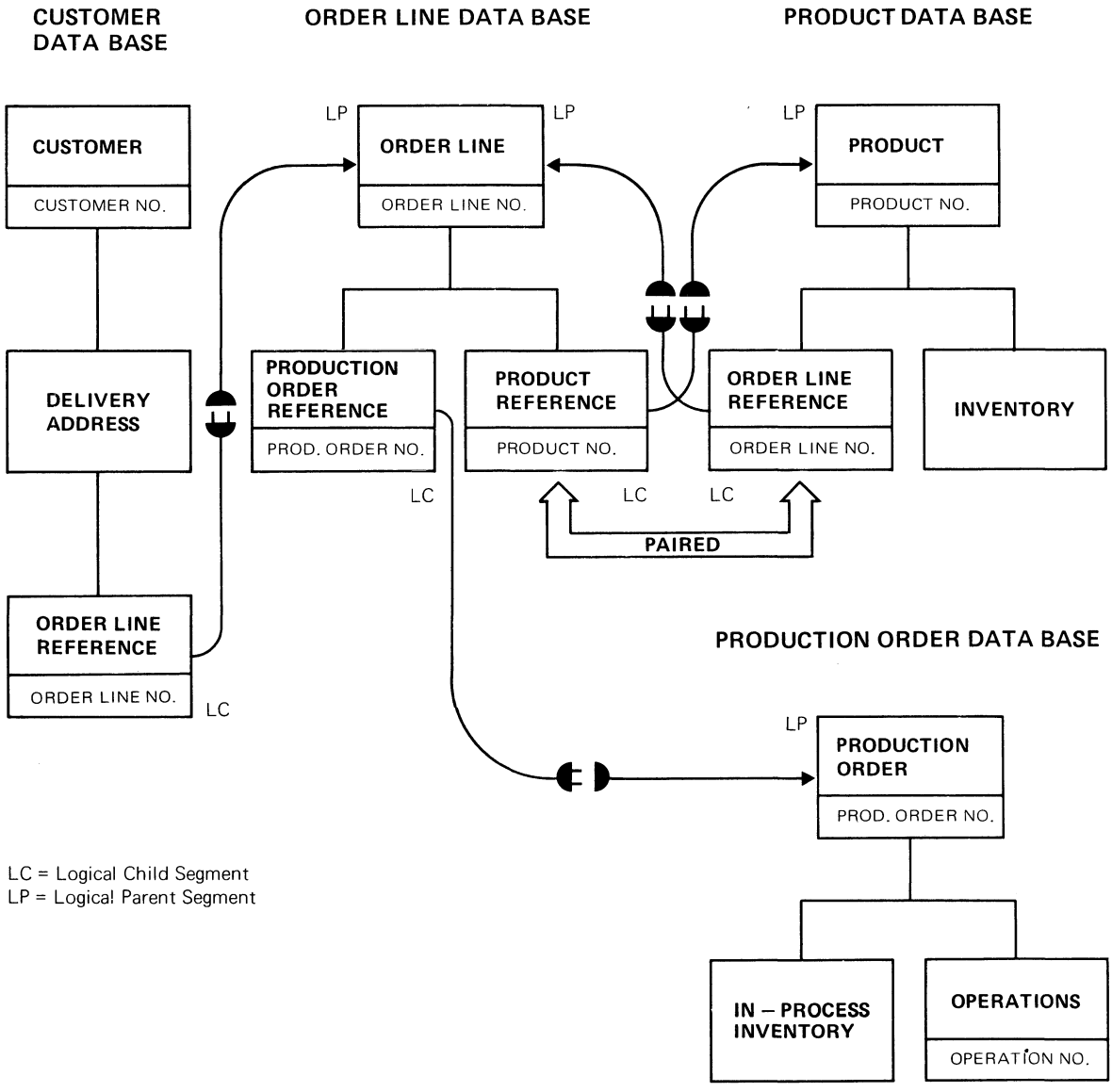


Figure 22. Process Industry Physical Data Bases

CUSTOMER DATA BASE

The customer data base is made up of three segment types on three hierarchical levels. The first segment, the CUSTOMER segment, contains:

- The customer number, used as the segment key.
- The customer name and address.
- Sales information (such as order values, invoice values, credit limits).

The DELIVERY ADDRESS segment gives the address to which orders should be shipped (not necessarily the same as the address contained in the CUSTOMER segment), and a location code.

The ORDER LINE REFERENCE segment establishes the logical relationship with the order line data base. This segment contains only the order line number, used as the segment key.

ORDER LINE DATA BASE

The order line data base is made up of three segment types on two hierarchical levels. The first segment, the ORDER LINE segment, contains:

- The order line number, used as segment key.
- Product identification.
- Quantities and values.
- The delivery date.
- Order line status, broken down into the quantity allocated from stock, the quantity to be produced, and the planned production date.

The PRODUCTION ORDER REFERENCE segment establishes the logical relationship with the production order data base. This segment contains only the production order number, used as the segment key.

The PRODUCT REFERENCE segment, containing only the product number as segment key, establishes the logical relationship with the product data base. The PRODUCT REFERENCE segment is paired with the ORDER LINE REFERENCE segment. The reason for this is to make DL/I-Entry responsible for maintaining both segments in the event one is changed.

PRODUCT DATA BASE

This data base contains three segment types on two hierarchical levels. The first segment, the PRODUCT segment, contains:

- The product number, used as segment key.
- The product description.
- Overall sales information (such as sales values, sales periods, statistics).

The ORDER LINE REFERENCE segment, containing only the order line number as segment key, establishes the logical relationship with the order line data base and is paired with the PRODUCT REFERENCE segment in that data base.

The INVENTORY segment is not involved in a logical relationship. It contains:

- The quantity on hand and stock location (in periods).
- The quantity in production (in periods).
- The quantity on order (in periods), broken down into the quantity on customer order and the quantity on purchase order.

PRODUCTION ORDER DATA BASE

The production order data base contains three segment types on two hierarchical levels. The main segment is the PRODUCTION ORDER segment; it describes both planned and in-process production. The segment contains:

- The product description
- The production begin and end dates, with quantities
- Material allocation information
- The checkpoint plan.

The IN-PROCESS INVENTORY segment contains the up-to-date inventory at various points in the processing sequence or at intermediate stores.

Occurrences of the OPERATIONS segment describe operations during processing. This segment contains:

- The operation description
- The operation duration
- The set-up time
- The operation status.

INQUIRIES USING THESE DATA BASES

The logical relationships established between the four physical data bases shown in Figure 22 make it possible to build three logical data bases:

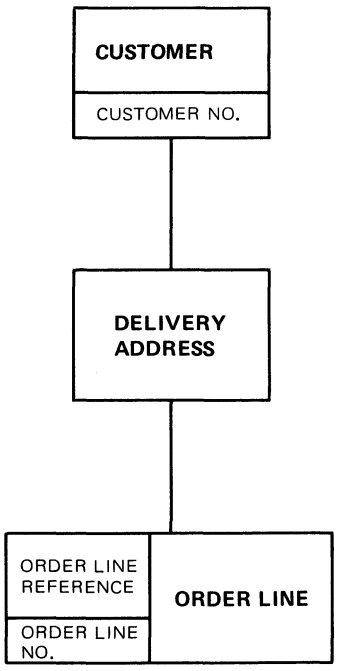
- Order entry data base
- Product control data base
- Production planning and control data base.

These logical data bases, shown in Figure 23, present the application programmer with new hierarchical structures of information, different from the structures contained in the physical data bases. The following text describes the types of processing that might be done with the logical data bases.

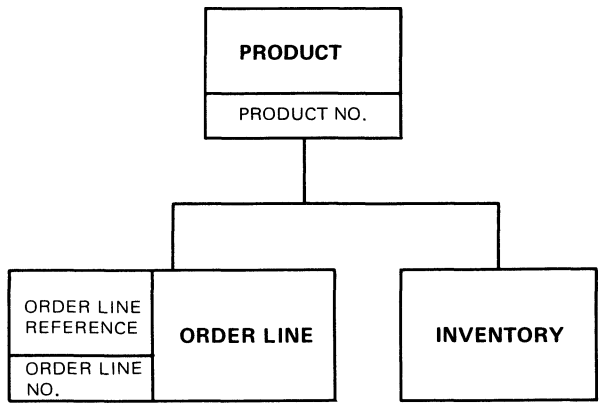
Order Entry Data Base

The order entry data base is made up of the customer data base and the logically related order line data base. Since, however, the two dependent segments of the order line data base are reference segments defining other logical relationships, they cannot be included in the order entry data base.

ORDER ENTRY DATA BASE



PRODUCT CONTROL DATA BASE



PRODUCTION PLANNING AND CONTROL DATA BASE

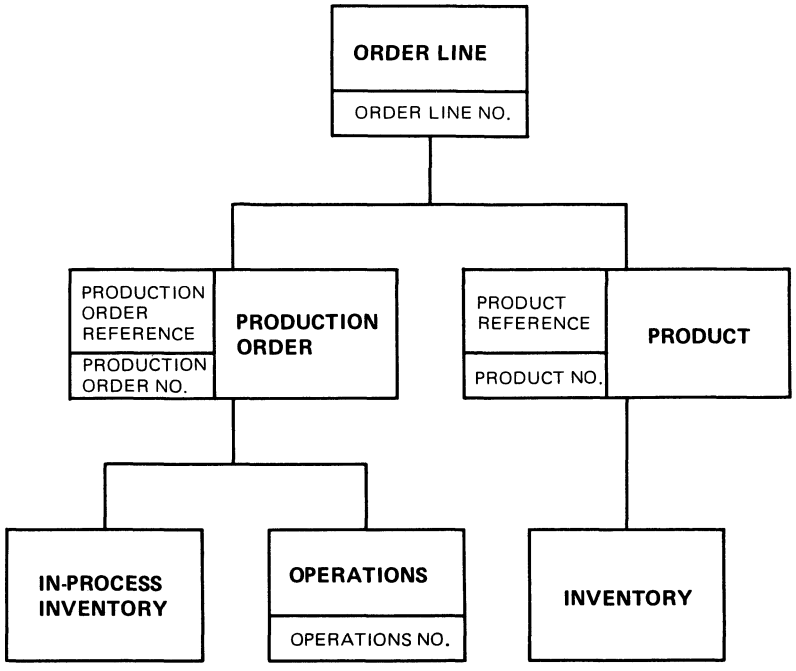


Figure 23. Process Industry Logical Data Bases

The order entry data base could be used to inquire about the existing orders for a particular customer, and to modify these orders. For instance, shipment dates for order lines could be confirmed and possibly changed, depending on their order line status. Or delivery dates, addresses, and quantities could be controlled. New customers and new order lines could be added.

Order processing would be another possible application for this data base, involving checking for valid customer numbers and addresses, controlling credit limits, and the like. Information might also be obtained for preparing invoices or other documents.

Product Control Data Base

The product control data base is made up of the product data base and the logically related order line data base.

Excluded from the product control data base are the two dependent segments of the order line data base, for they are reference segments defining other logical relationships. One of these segments, the PRODUCT REFERENCE segment, is nevertheless paired with the ORDER LINE REFERENCE segment, as shown in Figure 22. The reason for this is to make DL/I-Entry responsible for maintaining both segments in the event one is changed. For instance, if a new ORDER LINE REFERENCE segment were added to the product control data base, DL/I-Entry would add a corresponding ORDER LINE segment in the order line data base and also the appropriate PRODUCT REFERENCE segment in the order line data base.

The product control data base could be used to evaluate and modify production requirements. For instance, all order lines for a particular product could be retrieved, to establish delivery dates and quantities. Or decisions about production orders for a particular product could be made, such as the quantities needed to fill customer orders or the production dates necessary to meet specified delivery dates.

Production Planning and Control Data Base

The production planning and control data base is made up of the order line data base and the logically related production order and product data bases.

Since the ORDER LINE REFERENCE segment of the product data base is a logical child segment, it is excluded from the logical data base structure. This segment is nevertheless paired with the PRODUCT REFERENCE segment, as shown in Figure 22, in order to make DL/I-Entry responsible for maintaining both segments in the event that one is changed.

The production planning and control data base could be used to:

- Insert new production orders related to the appropriate order line, or update existing orders depending on decisions made during production requirements evaluation.
- Check order line status, production order status, in-process inventory, operation dates.
- Calculate the quantities not covered by allocated stock.
- Allocate available stock.
- Update inventories.

Index

Where more than one page reference is given, the major reference is first

READER'S COMMENT FORM

DL/I-Entry DOS/VS
General Information Manual

GH12-5115-1

Please comment on the usefulness and readability of this publication, suggest additions and deletions, and list specific errors and omissions (give page numbers). All comments and suggestions become the property of IBM. If you wish a reply, be sure to include your name and address.

COMMENTS

THANK YOU FOR YOUR COOPERATION
PLEASE FOLD ON TWO LINES, STAPLE AND MAIL

YOUR COMMENTS, PLEASE

Your comments on the other side of this form will help us improve future editions of this publication. Each reply will be carefully reviewed by the persons and department responsible for writing and publishing this material.

Please note that requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or the IBM branch office serving your locality.

IBM Germany
Program Product Center
58 Schwertstrasse
D-7032 Sindelfingen
Federal Republic of Germany

DL/I-Entry DOS/VS General Information Manual Printed in Denmark GH12-5115-1

IBM

IBM World Trade Corporation
821 United Nations Plaza
New York, New York 10017
U.S.A.

GH12-5115-1

DL/I-Entry DOS/VS General Information Manual Printed in Denmark GH12-5115-1

IBM

IBM World Trade Corporation
821 United Nations Plaza
New York, New York 10017
U.S.A.