Honeywell

LEVEL 66

SOFTWARE

DATA MANAGEMENT-IV SYSTEM OVERVIEW

SERIES 60 (LEVEL 66)

SOFTWARE
DATA MANAGEMENT-IV
SYSTEM OVERVIEW

SUBJECT

Overview of Data Management-IV Components

ORDER NUMBER

DF73, Rev. 0

August 1977

Honeywell

PREFACE

This publication provides an overview of Honeywell's Data Management-IV (DM-IV) system and the features of its individual components. It is directed toward both data processing personnel and other personnel who will be involved with application systems using DM-IV.

This document is the first and most general of a series of publications describing the administration and use of DM-IV systems. This system overview is divided into four sections:

- Section I briefly describes the DM-IV processing environment and the types of data processing problems which the DM-IV system helps provide solutions to.
- The salient technical features of each of the DM-IV components are described in Section II.
- Section III presents various scenarios which illustrate some ways in which each of the components might be used, thus describing types of requirements for which the components are most useful.
- Descriptions of some of the facilities which assist the data processing staff are contained in Section IV.

The following chart shows the relationships between the DM-IV documents. This is followed by an annotated listing of the documents.

File No.: 1P12

DF73

SERIES 60 (LEVEL 66)

DATA MANAGEMENT-IV SYSTEM OVERVIEW ADDENDUM B

SUBJECT

Additions and Changes to *Data Management-IV System Overview*

SPECIAL INSTRUCTIONS

This is the second addendum to DF73, Revision 0, dated August 1977. Insert the attached pages into the manual according to the collating instructions on the back of this cover. Change bars in the margins indicate technical additions and changes; asterisks indicate deletions. These changes will be incorporated into the next revision of the manual.

Note:

Insert this cover after the manual cover to indicate updating of the document with Addendum B.

ORDER NUMBER

DF73B, Rev. 0

July 1979

24562
1979
Printed in U.S.A.

Honeywell

COLLATING INSTRUCTIONS

To update the manual, remove old pages and insert new pages as follows:

Remove	Insert
v, vi	v, vi
2-9, blank	2-9, 2-10
i-1 through i-3	i-1 through i-3

File No.: 1P1c

SERIES 60 (LEVEL 66)

SOFTWARE

DATA MANAGEMENT-IV SYSTEM OVERVIEW ADDENDUM A

SUBJECT

Additions and Changes to *Data Management-IV System Overview*

SPECIAL INSTRUCTIONS

This is the first addendum to DF73, Revision 0, dated August 1977. Insert the attached pages into the manual according to the collating instructions on the back of this cover. Change bars in the margins indicate technical additions and changes; asterisks indicate deletions. These changes will be incorporated into the next revision of the manual.

Note:

Insert this cover after the manual cover to indicate updating of the document with Addendum A.

ORDER NUMBER

DF73A, Rev. 0

January 1978

20858
3578
Printed in U.S.A.

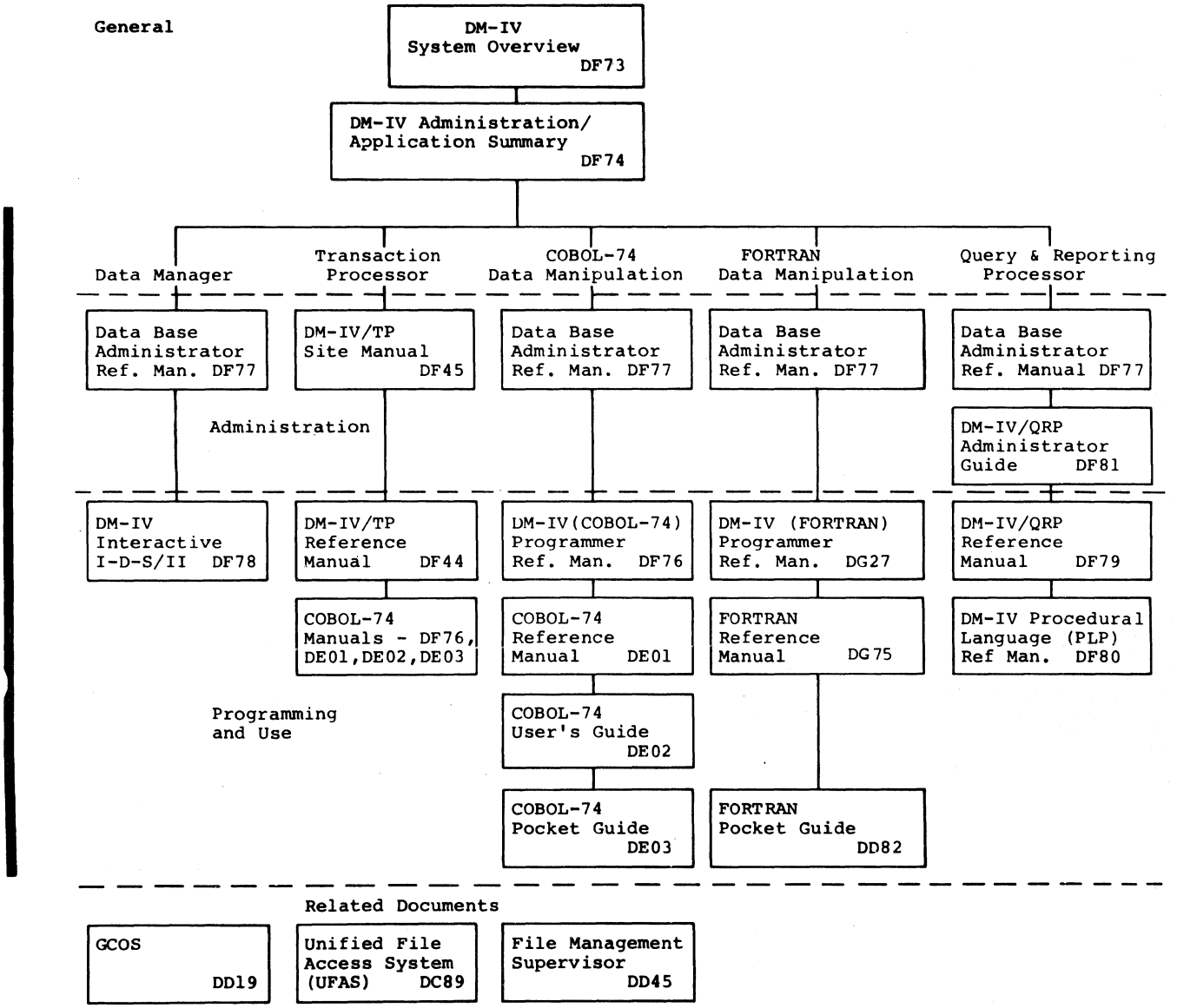
Honeywell

COLLATING INSTRUCTIONS

To update the manual, remove old pages and insert new pages as follows:

Remove	Insert
iii through vi	iii through vi
2-1, 2-2	2-1, 2-2
2-9, blank	2-9, blank
3-1, 3-2	3-1, 3-2
4-1, 4-2	4-1, 4-2

File No.: 1P12


General
DM-IV
System Overview
DF73
DM-IV Administration/
Application Summary
DF74
Data Manager
Transaction
Processor
COBOL-74
Data Manipulation
FORTRAN
Data Manipulation
Query & Reporting
Processor
Data Base
Administrator
Ref. Man. DF77
DM-IV/TP
Site Manual
DF45
Data Base
Administrator
Ref. Man. DF77
Data Base
Administrator
Ref. Man. DF77
Data Base
Administrator
Ref. Manual DF77
Administration
DM-IV/QRP
Administrator
Guide DF81
DM-IV
Interactive
I-D-S/II DF78
DM-IV/TP
Reference
Manual DF44
DM-IV(COBOL-74)
Programmer
Ref. Man. DF76
DM-IV (FORTRAN)
Programmer
Ref. Man. DG27
DM-IV/QRP
Reference
Manual DF79
COBOL-74
Manuals - DF76,
DE01,DE02,DE03
COBOL-74
Reference
Manual DE01
FORTRAN
Reference
Manual DG75
DM-IV Procedural
Language (PLP)
Ref Man. DF80
Programming
and Use
COBOL-74
User's Guide
DE02
COBOL-74
Pocket Guide
DE03
FORTRAN
Pocket Guide
DD82
Related Documents
GCOS
DD19
Unified File
Access System
(UFAS) DC89
File Management
Supervisor
DD45

Order No.	Title and Description

DF73 Data Management-IV System Overview

Basic introduction to the features incorporated in each of the DM-IV components.

DF74 Data Management-IV Administration/Application Summary

Description of the responsibilities of the Data Base Administrator and application programmers with respect to the data base and transaction processing. Serves to introduce both the Data Base Administrator (DBA) and programmers to the tools used by the DBA and to the data base management facilities of DM-IV.

DF76 Data Management-IV (COBOL-74) Programmer Reference Manual

Description of the Integrated Data Store/II (I-D-S/II) Data Manipulation Language used in COBOL-74. Also furnishes an introduction to the subschema Data Description Language (DDL) used by the DBA to describe data available to a COBOL-74 program.

DF77 Data Management-IV Data Base Administrator Reference Manual

Description of the languages used to describe the logical and physical structure of the data base, its records, and the relationships between records. Description of DBA utilities and samples of reports available to the DBA. Descriptions of the subschema DDL for COBOL-74 and FORTRAN.

DF78 Data Management-IV Interactive I-D-S/II

Description of the use of the Interactive I-D-S/II facility within DM-IV.

DF79 Data Management-IV Query and Reporting Reference Manual

Description of the programmer and end user interfaces provided in DM-IV/QRP. Instructions are given for the use of the DIALOG and DESCRIBE interactive facilities and for Query Language and the QRP Procedure Language.

DF80 Data Management-IV Procedural Language Reference Manual

Description of programming using the Procedural Language Processor, an option of QRP.

DF81 Data Management-IV Query and Reporting Administrator Guide

Description of the Data Base Administrator's tasks in defining the subschema and describing user requirements for DM-IV/QRP.

Order No.	Title and Description

DF44 Data Management-IV Transaction Processor Reference Manual

General description of DM-IV/TP and its components, with emphasis on the application programmer's view of the system. Description of the procedures for programming Transaction Processing Routines (TPR's) and implementing them to form an operational system. Also included are design considerations for TPR preparation and transaction terminal operation.

DF45 Data Management-IV Transaction Processor Site Manual

System generation, installation, and operating procedures for DM-IV/TP. Includes TP Data Base Manager Interface, site operation, and TP recovery/restart procedures.

DG27 Data Management-IV (FORTRAN) Programmer Reference Manual

Description of the Integrated Data Store/II (I-D-S/II) Data Manipulation Language used in FORTRAN. Also furnishes an introduction to the subschema Data Description Language (DDL) used by the Data Base Administrator to describe data available to a FORTRAN program.

CONTENTS

ILLUSTRATIONS

The DM-IV/TP system incorporates software necessary to manage the communications in a high-volume transaction oriented environment, control the concurrent execution and data base accesses of many transactions, and preserve system and data base integrity. This frees the application programmers to concentrate on the application, thus improving programmer productivity by avoiding the need to design and test the transaction management functions.

The programming of Transaction Processing Routines (TPR's) is done in COBOL-74; these TPR's are treated as subprograms of the transaction processing system. No special languages or preprocessors are needed, thus minimizing retraining requirements and allowing programmers to code in a standardized language and use the standard COBOL-74 compiler. Communication between TPR's and DM-IV/TP is via the Linkage Section, which uses a predefined interface format.

With these TPR's, transactions can be entered from remote locations and processed to update the data base or provide data to the transaction terminal operator. No knowledge of computers or the programs is required to enter transactions. The human interfaces are designed by the user, so they can be oriented to the requirements of the terminal operators. Transactions can be designed to serve for data entry, issuing messages only when errors are detected, or transactions can be designed so the system has a conversation with the terminal operator and guides the operator through the process. DM-IV allows the processing modes to vary among transaction types; the user is not restricted to using one processing mode or only a few transaction types.

Regardless of processing mode, DM-IV/TP makes it possible for many users to concurrently access the data base (for both retrieval and update) as if they were the only users online. DM-IV/TP prevents problems caused by simultaneous need for the same resources.

THE PROBLEM OF END USER ACCESS ON DEMAND

Not all information requirements can be preplanned; frequently, very little time is available to obtain the information needed to solve some problems and assist in making decisions. Even when the lead time for a request is sufficient to design and write a program, there might not be sufficient resources to obtain the desired results without impacting other projects. Very often, these are "spur of the moment" or one-time problems. Even without data base management systems, the data has often been available to programs -- albeit scattered and difficult to access; the more significant obstacle has been the time and other resources needed for programming.

Merely being able to access data (a capability achieved through the DM-IV Data Manager) is not sufficient by itself; to meet the user requirements, it is necessary to retrieve (and possibly update) specific data, manipulate it, and present it to the user in some desired format.

These problems are addressed by the DM-IV Query and Reporting Processor, and its several subsystems.

DIALOG is an interactive query system, allowing the user to directly interface with the system, eliminating the need to describe the requirements to a programmer or wait for programs to be written. By their very nature, many queries are heuristic processes in which it is difficult to state in advance exactly what is needed. New questions arise as the previous ones are answered. Through DIALOG, even the untrained user is prompted so the desired results can be obtained.

No knowledge of data structure is needed; the user specifies only which fields are needed and can be retrieved. Common queries do not need to be repeated; queries can be saved. In addition, precise or informal report formats can be specified.

In many cases, the users -- with little training -- can prepare the programs themselves, thus obviating the need to wait for the availability of programmers and freeing the programming staff for larger, more complex development efforts.

Oftentimes, it is necessary to develop programs on short notice or when limited resources are available. Other programs must be modified frequently to meet changing requirements. The QRP Procedural Language Processor (PLP) is well-suited for programs which must be developed quickly or frequently changed. Long reaction times caused by the time required to code and debug new programs or changes in other languages have often proven frustrating when information requirements change frequently. In PLP, programs are written at a higher level so the computer does more of the work. The programmer has less to code, plus has an online debugging capability, so programs can be developed (and changed) more quickly. Not only does this satisfy the new requirement sooner, but it also increases the productivity of the programming staff.

SECTION II

OVERVIEW OF DM-IV COMPONENTS

Data Management-IV is an evolutionary development integrating proven systems with new features designed to enhance ease of use and performance, resulting in a comprehensive approach toward meeting the dynamic requirements associated with the user's information resources.

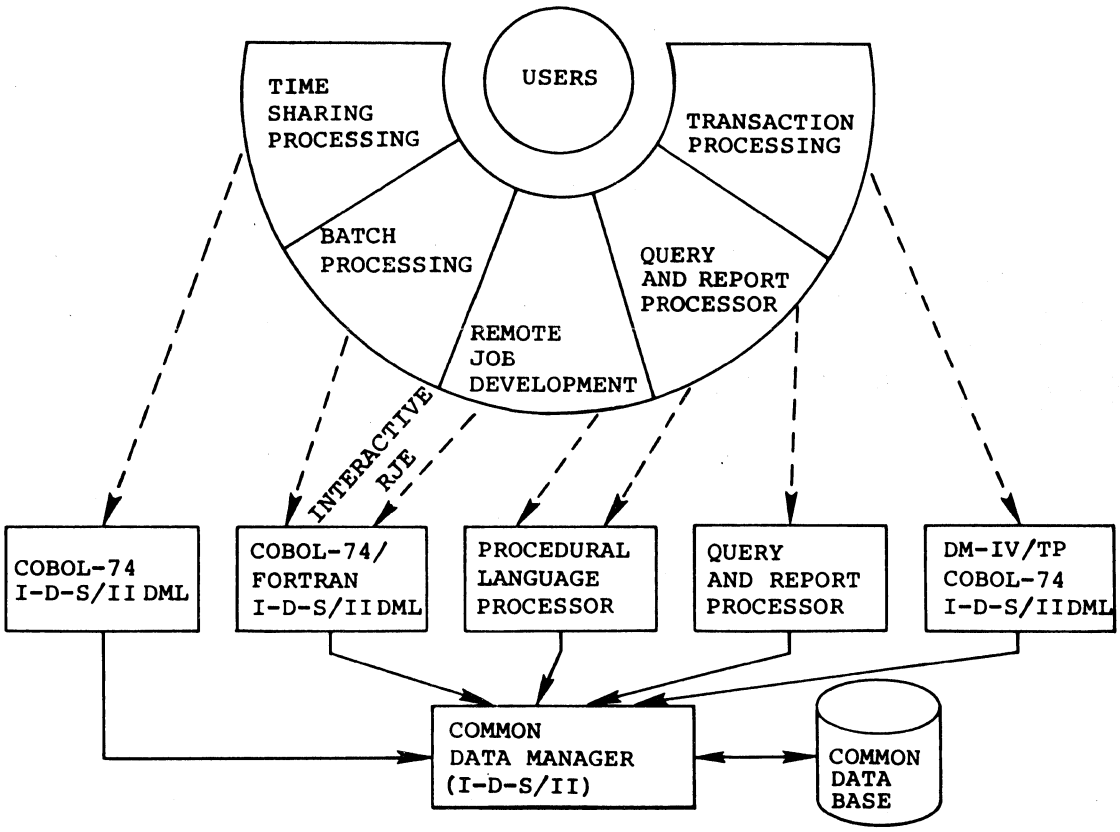


Figure 2-1. DM-IV Components and Processing Environments

Figure 2-1 illustrates the relationships between the various DM-IV components. These components serve the following processing environments:

- Application development
- Data base administration
- Transaction processing
- Interactive data access
- Batch production
- Time sharing

The heart of DM-IV is the Data Manager, for all other components process information by interfacing with the data base. The Data Manager administers the creation of the physical and logical structures of the data base (the schema) and controls the creation of the application-specific views (subschema) of the data base which are used in processing. It further serves as the interface between the data base and the various DM-IV processors which access the data base and perform operations upon it. The Data Manager also provides several facilities to support data base administration (see Section IV).

The DM-IV Data Manager includes Integrated Data Store/II (I-D-S/II), an advanced, comprehensive data base management system. Through I-D-S/II, the Data Manager has implemented the recommendations and specifications of the Committee on Data Systems Languages (CODASYL), as well as providing additional features to increase its flexibility.

The multiplicity of data structures available in I-D-S/II permits a wide variety of complex data relationships to be represented in the data base. In the past, the lack of suitable structures has hindered the development of efficient and effective data bases. With DM-IV, complex relationships can be represented without requiring that simple relationships be made complicated. Besides making it feasible to represent the complex relationships among records in a user's application, this flexibility also makes it possible to more easily structure the data for ease of access and fast retrieval, which is essential in online processing.

The basic element of the data base is the record, which is composed of data fields. This is somewhat analogous to a hardcopy record, such as an order form, personnel record, etc., which also has various fields. Data structures are created by relating records to each other, thus creating sets.

These sets, which can be likened to file folders, are defined by relating records of two or more types. The simplest relationship is one in which an "OWNER" record can be related to one or more "MEMBER" records of a single type. For example, just as a file folder for a vendor could contain purchase orders, a set for a vendor can be composed of purchase order records. Sets can also be composed of OWNER records with several MEMBER record types, MEMBER records having several OWNERS in different relationships (networks), OWNER records having several MEMBER types in different relationships (trees), multilevel relationships (hierarchies), and OWNER records having MEMBERS which can have the OWNER record type as members also (cycles). All of these basic structures can be used and combined in one or more integrated areas (i.e., GCOS files) in the same data base.

I-D-S/II also permits areas to use an indexed organization, with up to 64 record key types being used for the logical sequencing and indexing of records in the data base.

All data fields, records, sets, and areas in a data base are defined using a single independent language -- the schema Data Definition Language (DDL) -- which is common to all DM-IV components. Language specific descriptions -- using a subschema Data Definition Language -- are used to describe the data actually used by the programs. This approach affords flexibility by making it possible for any DM-IV component to access data in the data base, regardless of how it was created.

The centralized control which is a characteristic of the schema supports the concept of using a data base administrator (DBA), an individual or group, who is responsible for the overall administration and control of the data base.

By having a centralized point for the administration and control of data as a resource, together with a flexible, powerful processor for presenting this resource to the user, it is possible for each user to have a view of the data base without imposing that view on all other potential users and without being burdened with total responsibility for control.

Besides being responsible to the users for designing the data base to meet the user requirements, the DBA is also responsible for the physical design of how data is stored. The physical description of the storage structure is described using the schema Device Media Control Language (DMCL). The data structures (logical) and storage structures (physical) are independent; thus data structures are not constrained by physical limitations and improvements can be made to the storage structure without affecting the logical representation of data.

Besides the DMCL, the use of subschemas contributes to program/data independence. It avoids the problems caused by the lack of separation between data and how it is used. In the past, descriptions of data and the methods for accessing it were embedded within the programs using the data. If the data or its organization were changed, the programs had to be changed. Programs could not be written unless full details regarding data storage were available. Besides making it more difficult to use data, it might not have always been desirable (for security reasons) to give the programmer access to all the data when only part of the data was needed. With DM-IV, a program needs a description of only those data and relationships that are relevant to the application; even the existence of certain data items and relationships can easily be kept invisible to the program.

The subschema is a language dependent subset of the schema. Data descriptions in the subschema are described in a subschema DDL which is syntactically compatible with the programming language used for data manipulation. Thus, when the subschema is incorporated in a program, the definitions of the data base entities conform to the conventions of data definition in the language. In addition, the subschema can be used to reformat data fields by dividing them (e.g., splitting a date into year, month, day) or changing their representation (e.g., binary format to ASCII or changing the size of a field).

Freed from the mundane details of accessing the data, the programmer can concentrate on the application. In addition, this facilitates access by personnel with minimal technical training. Furthermore, the independence between data and programs provides much greater flexibility and adaptability to changing requirements. The data base can be changed to meet the requirements of some applications without requiring that all programs be recompiled. The transformation between the schema DDL/DMCL view of the data base and the subschema view is handled by the Data Base Control System (DBCS).

With many users needing to retrieve and update data, often simultaneously, true sharing of the data resource is a necessity. If the user is forced to wait beyond tolerable limits, the value of the information may deteriorate. The DM-IV Data Manager supports concurrent access by multiple users while protecting against potential problems which might occur if a record were changed while another user was reading it.

Concurrent access is controlled through the File Management Supervisor (FMS). Through FMS concurrent access control, it is possible to prevent other programs from read or read/write access while the record has been read (and is expected to be modified) by a program. The type of protection is specified by the DBA.

Records remain locked only as long as it is necessary to prevent conflict. Thus, a program may have to wait momentarily to preserve data base integrity, but the wait should be short, thus facilitating high throughput and fast response time. Additional concurrent access controls are implemented in the DM-IV Transaction Processor, so transactions can be processed simultaneously.

Data and storage structures can be designed for fast retrieval, which is especially necessary in transaction processing. The use of integrated structures is especially well-suited for applications that require many entry points into the data base, thus allowing shorter access paths.

Deletion of records can be deferred for throughput reasons by specifying logical (rather than physical) deletion of records. With logical deletion, the record still exists but there is no means of accessing it. The records can be physically deleted later by using a Data Manager utility. The utility (Q2DEL) also serves to compact the data base.

The capability for multi-user access demands precautions for data security. In DM-IV Data Manager, the privacy locks recommended by CODASYL are combined with the file permissions monitored by FMS. During execution of a program, privacy locks make it impossible to operate on specified fields, record types, set types, or areas unless the correct privacy key is supplied. Separate privacy locks can be specified for each Data Manipulation Language (DML) operation. In addition, privacy locks can be used to protect against compilation of a subschema, validation of a subschema, or any unauthorized access or modification of the schema or a subschema. Of course, by not defining elements or relationships in a subschema, an application has no way of even knowing that they exist.

Individual areas can be protected via FMS access permissions. Several types of permissions exist, with the basic ones being "read only" and "read/write". Permissions can be granted to all users or selectively to specific users (or everybody except specific users).

The DM-IV Data Manager, in combination with FMS, also provides for system integrity with the recovery/restart capability. As records are added and modified or set relationships are changed, these actions are journalized. When a process aborts, the data base can be quickly recovered and the program restarted.

Data integrity is also fostered by preventing bad data from entering the data base in the first place. Rather than requiring that each individual application incorporate data validation routines, user-coded procedures can be incorporated in the Data Manager. These procedures can be used to edit data, perform calculations, or reject data that should not be placed on the data base.

DM-IV TRANSACTION PROCESSOR

The DM-IV Transaction Processor (DM-IV/TP) is a communications-oriented executive which manages the transaction processing environment, including all use of data base/data communication facilities. DM-IV/TP provides the user with the capability of simultaneously processing a variety of transactions which may access the data base, allowing storage and retrieval of information on an online, real-time basis. The Transaction Processor's flexibility and capability for handling a high-volume of input and output facilitates obtaining information where and when it is needed and in a form most suitable to the end user.

Transaction processing is event-oriented; in an enterprise, the creation of transactions themselves is also event-oriented. Through DM-IV/TP, inquiries or data resulting from the normal flow of business can be entered from a remote terminal and immediately processed. Any pertinent updating of the data base is initiated and the transaction reply is immediately returned to the terminal user.

Because the Transaction Processor is based on events, actions are performed immediately upon receipt of new data or when current data is needed instead of waiting until there are enough transactions to process a batch or until the end of a day (or other period). The Executive Manager of DM-IV/TP includes the software necessary for controlling the simultaneous processing of transactions from many terminals.

The applications processing required for transactions is contained in user-written Transaction Processing Routines (TPR's). TPR's contain coding to read and process input data, access records on the data base, and build output messages to be returned to the terminal operator or other destinations.

Programming productivity is enhanced in several ways. The programmer deals with only a single transaction, without being concerned with the fact that other transactions are being processed at the same time. Because the TPR's are generally fairly short and straightforward, design and debugging time is reduced. Furthermore, no special language is needed; TPR's are coded in a standard subset of COBOL-74 (COBOL-74 input/output and associated statements are not used) and the standard I-D-S/II Data Manipulation Language (DML) for COBOL-74. Simple interfaces are used to interact with the Transaction Processor's executive routines. Thus, training time is minimized and the programmers are able to use a comprehensive subset of the COBOL-74 language.

From the application programmer's viewpoint, DM-IV/TP can be likened to batch processing wherein a driver program reads the input and determines which subroutines to call for processing the input record. Unlike batch processing, input records are independent in transaction processing. When additional data is needed to complete a transaction, the system can have a "conversation" with the terminal operator.

The form of input to be processed -- and also of the corresponding output messages (if any) -- is determined by the user's programming with the TPR's. Thus, the form of interaction is designed by the user to suit the application requirements and skill levels of the terminal operators. Although most transactions consist of a single input record (which might be several lines), conversational transactions are also supported.

Characteristically, the end user -- the person initiating the transaction from a remote terminal -- is not expected to understand programming or be computer-oriented. Rather, the end user understands the applications transactions. These could be orders, bank deposits, shipments of goods from inventory, requests for status, or any other kind of transaction. Entering a transaction is as simple as entering a user-defined code to identify the type of transaction, followed by the data, and waiting for the result. The Transaction Processor also supports special purpose terminals designed specifically for the application.

DM-IV/TP places no limit on the number of different transaction types which can be used in a transaction processing system. Separation of the processing requirements into different transaction types is based on the requirements of the application. If necessary, transactions can be serviced by several TPR's. Transactions of different types can use the same TPR's; this reduces the coding requirement and also ensures that common processes can be handled identically for several types of transactions.

Because each transaction performs only a limited number of specific functions, rather than a wide variety of functions that might occur in a batch program, security is enhanced. Each transaction can have a different subschema and different permissions for operating on the data base, thus restricting the actions of a TPR. In addition, a terminal user can be authorized to enter only certain types of transactions, thus limiting the types of functions which can be performed and, consequently, the types of data which can be made available.

Because each user's transaction processing requirements are different, the DM-IV/TP software is customized before it is used. "System generation" is used to define the environment in which the TPR's are to be processed; a system generation source program is processed by a special translation program which is used to generate the TP Executive needed for the user's application. This results in the DM-IV/TP routines being customized for the user's own hardware/software environment and application requirements.

The schema and subschema which the application uses as well as the system resources that will be required at run time are specified at system generation. For instance, it is at system generation time that hardware information such as maximum line length and screen size of terminals, and software information such as message identifiers is stated so that they can be available to DM-IV/TP to format and transmit messages.

When developing transaction processing applications, the programmer is actually using several interfacing "layers" of system software. In addition to the data base, the first level is the communications network and the operating system of the Front-end Network Processor (FNP); then the central computer unit, its operating system, GCOS, and the File Management Supervisor, FMS; and the DM-IV Transaction Processor, which operates as a privileged slave program.

The Transaction Processor software is logically divided into five major functional areas.

- Executive manager - Schedules and coordinates all Transaction Processor activities and allocates system resources for optimum performance. Enhances throughput by allowing parallel processing of similar transactions and their independent tasks.

- Message manager - Accepts and delivers transaction messages, validates terminals, transliterates and journalizes all incoming messages, activates transaction processing, and handles message exchange with the networking software of GCOS. Links an entire network of terminals to a centralized data base.

- Transaction manager - Controls and coordinates all activities while each transaction is being processed, including scheduling and allocating transaction resources and communicating with the user's customized application programs that perform the actual processing. These programs are in memory only during execution, reducing memory occupancy.

- Data base manager - Interfaces with the DM-IV Data Manager to allow concurrent access to the data base by all transaction programs in execution, while controlling these accesses and protecting the data base integrity. Journalizes all updates to the data base for recovery operations.

- System integrity manager - Provides the procedures to resume normal transaction processing after a hardware or software malfunction. Maintains data base integrity by rebuilding a file damaged during a malfunction from data journalized by the Data Base Manager.

DM-IV QUERY AND REPORTING PROCESSOR

The DM-IV Query and Reporting Processor is a group of subsystems designed to meet the requirements for quick reaction time, simplified programming, and direct access by end users. The driving force in the QRP is the Application Definition File (ADF), which defines the users' processing requirements and serves as a subschema. Because the ADF contains the description of the structure of the data base, it is not necessary for the user to be aware of the access paths used to retrieve records or know where fields are located. The user (or programmer) must know only the names of the fields to be referenced. Thus, by placing more information in the subschema, less information must be supplied when using QRP. Depending on the needs of the user, additional information can be supplied for more sophisticated retrievals or for obtaining reports in specific formats.

QRP includes two conversational query facilities. One, the DESCRIBE subsystem, aids the user who does not know the correct names for the fields in the data base. Using the ADF, DESCRIBE displays the names and descriptions of the fields which the user is allowed to access. The user is not given the names of fields which apply to other applications or which the user is not allowed to use. Thus, the ADF provides security and also gives a user a data base view specific to the user's application.

Through the DIALOG subsystem, QRP provides the casual user with a simple interactive capability for retrieving the values of selected data fields from the data base. No programming training is needed; DIALOG is designed for use by the person actually needing the information. DIALOG prompts the user with simple questions to aid in formulating the request and also incorporates a tutorial capability to describe all the possible options.

When the user has completed developing the query, QRP generates a program to execute the request and the program output is returned directly to the user's terminal. Executing the program in direct access mode (DAC) rather than time sharing improves the efficiency of the program while still retaining interactive responsiveness. For example, if the BROWSE capability is used, a program is generated to read the data base and display the contents of the desired data items at the terminal. Recurring queries can be saved so they need not be completely respecified each time they are needed.

Procedures can be written using the QRP Query Language to prepare informal or precise-format reports. The Query Language lets the user write formatted reports without writing statements to access the data base. Formats can include single-level reports with heading/footing lines, subtotals and totals, or multi-level reports with page layout and data element editing control. These reports can be directed to the originating terminal, central-site or remote printers, or time sharing report files. Parametric information can be supplied by the user at run time to tailor the procedure to the immediate processing needs.

Several types of common functions are performed automatically by QRP, thus allowing the user to specify only what needs to be done rather than how it is to be done. Selection criteria can be defined to restrict the collection of records to be used or records can be randomly sampled from the data base. If desired, these records can be sorted, using selected entries, data items, and/or variable values for up to fifty identifiers, in ascending or descending sequence. Besides the arithmetic calculations which can be performed by the user, the SUM, COUNT, STD DEV (standard deviation), AVERAGE (arithmetic mean), MIN (minimum), and MAX (maximum) of the values in the selected records can be automatically computed and displayed in heading or footing lines.

A more advanced capability, the QRP Procedure Language, provides the more experienced users with the option of writing their own retrieve statements, as well as other features. These extensions include a table lookup procedure and the ability to enter data interactively to an executing procedure.

To assure usability by the end user, it is necessary for a system to use familiar terminology. The Personal Language feature makes it possible for the QRP components to communicate with the user in application specific terminology or another language. The user can use native languages such as English, French, German, etc., or the user installation can define their own application languages or terminology.

PROCEDURAL LANGUAGE PROCESSOR

The Procedural Language Processor (PLP) an option of QRP, offers additional capabilities while still maintaining the ease of use of the basic QRP. Like QRP, PLP does not require the programmer to provide any descriptions of data -- merely the names of the data fields to be used. With PLP, there is a more powerful and extended data retrieval and processing capability, plus fields in the data base can be updated.

Programs written in the QRP Procedure Language are fully upward compatible with PLP. PLP extends the functionality of many QRP statement types as well as adding more types of statements. Thus, if processing requirements grow and the PLP features are needed, no conversion from QRP Procedure Language is necessary; the user merely builds upon the existing procedure.

PLP provides improved capabilities for performing retrievals from a data base and generating reports based on the content of the records and calculations. Criteria for selectively retrieving records can be combined or can be based on the results of previous operations. PLP can also operate on several data bases in the same program.

Unlike other QRP subsystems, PLP permits the user to update the data base. In addition, new data bases can be created by combining or splitting data bases or extracting a subset of another data base. Auxiliary sequential files can also be read or written. Sequential files created by other components can be processed, if certain conventions are followed in writing the file. For data integrity, checkpoints can be taken during update processing. Audit trails can also be produced to provide complete historical information of update activity against the data base.

PLP can increase programming productivity and improve reaction time since data base processing tasks can be accomplished with fewer language statements than other procedural languages, such as COBOL-74 and FORTRAN. PLP is also easier to modify, thus making PLP suitable for applications which are subject to frequent design changes.

BATCH PROCESSING

Even in environments where much of the processing is event-oriented and requires quick reaction times, there still may be applications which are more suited to batch processing. For applications with very long, complex reports or large volumes of input with less critical timeliness requirements (e.g., daily, weekly, etc.), batch processing may be preferred. DM-IV makes available the full functionality of COBOL-74 and FORTRAN. Programs written in COBOL-74 and FORTRAN can access the data base through I-D-S/II or can access conventional files through file input/output statements.

Programs written in COBOL-74 for batch processing can execute concurrently with DM-IV TP and QRP/PLP online processing. The data base and other resources are shared, so information is available to users regardless of the processing mode or DM-IV subsystems used.

TIME SHARING PROCESSING

DM-IV permits a COBOL-74 program that may contain I-D-S/II DML statements to be compiled as the result of the Time Sharing System CRN command and to access a data base under control of the Time Sharing Executive.

POINTER ARRAY SET

The pointer array set implementation supports the following functionalities:

1. Fast search of a non-VIA set.
2. Multimember sets.
3. Manual or automatic sets.
4. Efficient search techniques such as binary searches and multilevel pointer arrays.
5. Elimination of redundant data for duplicate keys in the Pointer Array record.
6. Local or global sets.

SECTION III

USING DATA MANAGEMENT-IV

This section illustrates, through use of a series of scenarios, the DM-IV components that can be used to process different types of information requirements. The interaction of these components is shown in Figure 3-1.

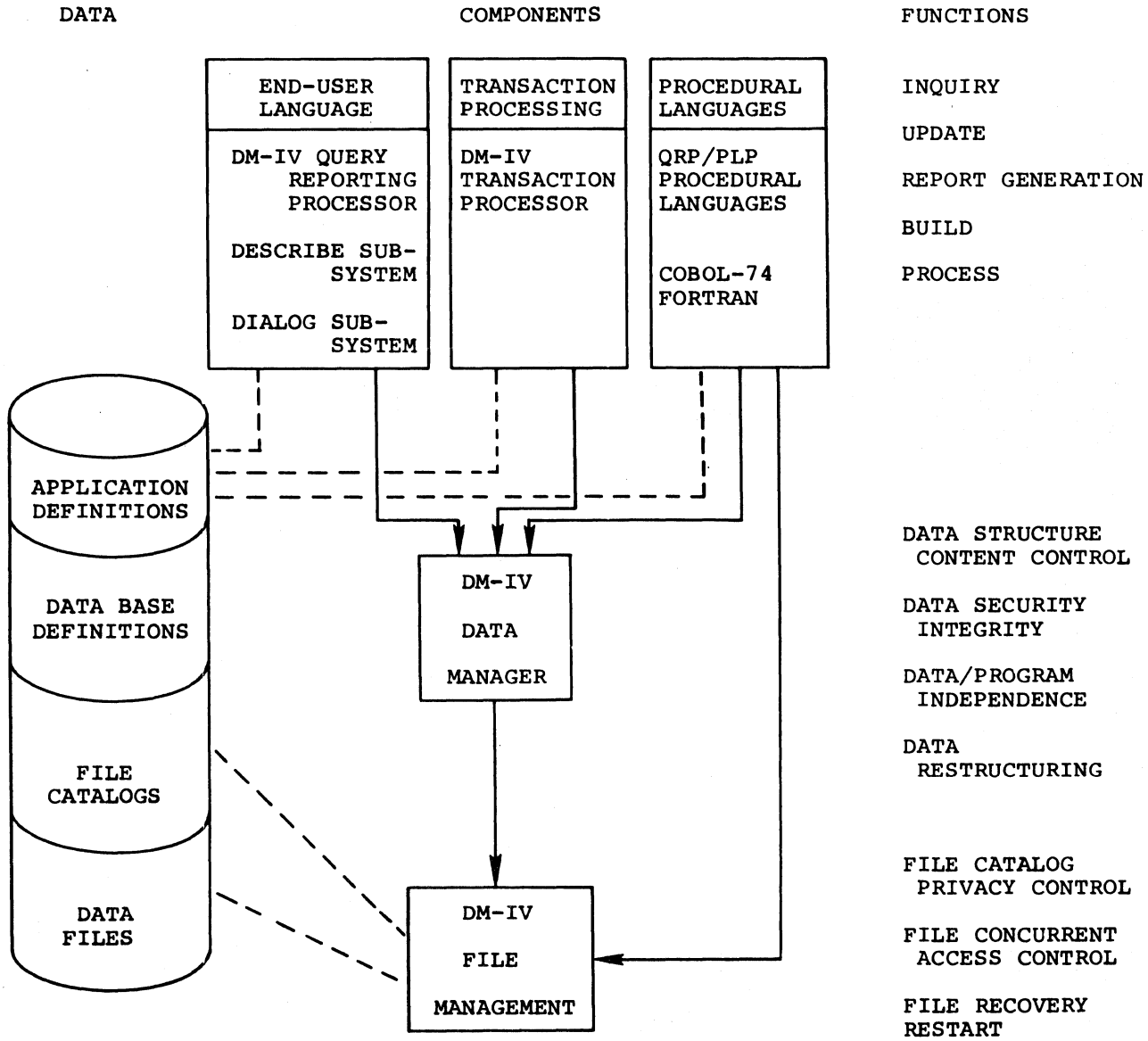


Figure 3-1. DM-IV Application Environment

The examples in this section are based on a data base supporting a hypothetical basketball league. The primary purpose of the data base is to serve as a repository of data on the performance of teams and individual players throughout a season, as well as maintaining certain business information. Information contained in the data base is of interest at both the league and individual team levels. The data base is stored at the league's centralized headquarters, but certain data is available to any team, while other data can be accessed by the team to which it pertains.

REQUIREMENTS OF THE DATA BASE

To support the information requirements of the hypothetical league, several types of records and relationships between records are defined. The full data base is described by the Data Base Administrator (DBA) in the schema. The logical relationships (data stucture) are described using the schema Data Description Language (DDL), while the physical placement of records (storage structure) is governed by the schema Device Media Control Language (DMCL). This permits either the data structure or storage structure to be changed without causing the other to change.

Each team in the league would have a record in the data base. This record could contain information on the team, its won-lost record, and ticket sales and gate receipts. Game records, which could contain schedule information and results from individual games, would relate teams to each other. For example, one team is related to a game record by its having a home game while for another team, the game record represents an away game. Through this record, a team's schedule and opponents can be found.

Records would also be created to store data on individual players. For example, this could include the player's name, season performance data, and other information. Player records would be associated to the team for which they play by being member records of a set owned by the team record.

There is a file of players' statistics by game, which can be used to determine a player's performance against any specific opponent.

Another relationship results from the need for an injury list. Players may optionally be members of a set consisting of all the injured players. This facilitates preparing reports which list injured players.

Through the Data Manager and the other DM-IV components, the data base becomes accessible to any of its users from remote locations. For security reasons, records for each team could be placed in separate data base areas (i.e., GCOS files). Using FMS file permissions, a team could be given read permission for all areas but only be able to write within its own area.

The data base administrator (DBA) can assign privacy locks for various entities (sets, records, etc.) with respect to each type of operation. These operations will not be permitted on the specified entity unless the correct privacy key is supplied at run time to satisfy the privacy lock. The data is thereby protected from unauthorized retrieval.

Several users can access the data base (or even the same records) concurrently. The File Management System (FMS) controls concurrent access, protecting from attempts by two users to simultaneously modify the record.

When records are modified or added, they can be journalized to afford system integrity and provide a recovery/restart capability.

In the following examples which include terminal input/output, text in capital letters represents messages from the system to the terminal. User responses are indicated in lower case. Where comments are necessary to explain the example, they are enclosed within parentheses.

ENTRY OF GAME RESULTS

One of the most critical requirements for timely information is that the initial entry of the data be quick and accurate. In this example, game results and player statistics must be reported for perhaps ten games in as many locations on any given day. Online access from these remote locations is a necessity.

Transaction Processing Routines (TPR's) can be written so the input can be processed as transactions by DM-IV/TP. The actual form of the transaction and its operator interface is designed by the user. One approach might be to use a "TSCORE" transaction for reporting team scores and a "PSCORE" transaction for entering totals for individual players. A session might look like:

```
tscore,770203,bulldogs,106,93
BULLDOGS 106 - GREYHOUNDS 93 ON 770203 (Verification message)
smith,12,7,6,3,9,11  (various statistics in predefined format)
SMITH SCORED 30 points
jones,6,3,5,5,4,4
FTA MUST BE GREATER THAN FT; REENTER LINE
jones,6,4,3,5,4,4
JONES SCORED 15 POINTS
   .
   .
   .
```

In the above example, the operator must be familiar with and experienced with the input format. Such an operator could quickly enter many transactions. However, the transaction can also provide prompting to assist the less experienced user. In addition, the transaction could be written as a single conversational transaction which processes all of a team's input, thus associating the input records to the team. This "SCORE" transaction might look like:

```
score
ENTER TEAM NAME, GAME DATE
greyhounds, 770102
ENTER PLAYER NAME
smith
ENTER FIELD GOALS, FREE THROWS MADE, ATTEMPTED
12, 7, 6
TOTAL POINTS SCORE WAS 30
ENTER FOULS
3
ENTER OFFENSIVE, DEFENSIVE REBOUNDS
9, 11
ENTER PLAYER NAME
   .
   .
   .
TEAM TOTALS -
    POINTS = 106    (provides a check that detailed input was correct)
       .
       .
       .
```

A dedicated user probably does not need that much prompting; too much time will be spent waiting for the prompting messages. Possibly, the input format may only be shown once (or repeated if operator makes an entry error):

```
ENTER TEAM NAME, GAME DATE
greyhounds, 770102
PLAYER, FG, FT, FTA, F, OR, DR
smith, 12, 7, 6, 3, 9, 11
SMITH SCORED 30 POINTS
jones, 6, 3, 5, 5, 4, 4
FTA MUST BE GREATER THAN FT: REENTER LINE
jones, 6, 4, 3, 5, 4, 4
JONES SCORED 15 POINTS
   .
   .
   .
TEAM TOTALS -
     POINTS = 106
        .
        .
        .
```

If the terminal supports forms mode, a form could be displayed on the terminal for the operator to "fill in the blanks." In this way, there would be no need for a different style of dialog for experienced and inexperienced users; the casual user would be guided by the form and the dedicated user would be able to use the system rapidly without having to wait for prompting messages.

ADD A PLAYER TO THE ROSTER

When a new player joins a team, it may be of importance to the team records to immediately add the record to the data base even though the complete data are not available. On the other hand, if it were decided that the complete set of data should be gathered and the record not created until all data became available, the addition could be handled using some of the same TPR's, but using other TPR's in addition.

With the conversational capabilities of transaction processing, it is possible to write a transaction to create a record and supply only part of the information necessary to complete the record. At one or more points during the processing, the transaction can, utilizing the conversational mode of TP, determine if the user wishes to continue building the record. If so, the transaction can store the data received up to this point, bypassing the remaining TPR's of the transaction, and terminate. At a later time, when the balance of the information becomes available, the user can execute the balance of the TPR's (which will have been made available by use of another transaction name; i.e., another entry point). This can also be done without conversational processing; the TPR's can process whatever data is available and then terminate. If the data is insufficient, the transaction can be aborted without creating a record.

Note that in both cases (either the full or partial supply of data), the same TPR's can be used. The difference in the processing is the early termination and the reentry at a different point. When reentered, the balance of the same set of TPR's (as would have been used if all the data had been entered at the same time) will be used to process the data.

TRADING A PLAYER

The trading of players between teams in the basketball league does not entail the addition of records to the data base because the players' records are already in the data base. All that is necessary in such a case is for the relationships of the players' records to be changed. A player record is removed from the roster set of one team and the same physical record is logically inserted into the roster set of another team. All other relationships in which the player record participated would remain the same.

REQUEST FOR A SINGLE DATA ITEM

At times, it may be necessary to check the value of only a single data item (or a very limited number of items). Unless this is a very common occurrence, it is probably unreasonable to write a program for it. Instead, the DIALOG subsystem of QRP or Interactive I-D-S/II can be used to perform the necessary retrievals.

For example, consider a situation in which it is necessary to check the total points made by a player in a specific game. The user can quickly log on to DM-IV and to the DIALOG subsystem. Presuming that the user knows the name of the Application Definition File (ADF) and the entry and item names, the user can perform the following ad hoc query:

```
APPLICATION FILE IS - statsadf
DATA-FILE-REFERENCE IS - playerstats
FUNCTION - browse
ENTRY IS stats
   .
   .
   .
DATA TO BE DISPLAYED - totpts
   .
   .
   .
SELECTION CRITERIA - team = "bulldog"
OR = PLAYER = "adams"      (AND is implied by testing different field)
OR = gamedate = "770308"
OR =      (carriage return)
ADDITIONAL CRITERIA -      (carriage return)
.....
SNUMB NNNNN      (Generated to facilitate identification of output)
NNNNN

TOTPTS = 14

ACTIVITY TERMINATED
```

If the user did not know the necessary names, he could have used the DESCRIBE subsystem to obtain the names and descriptions of the fields.

The degree of complexity of the data structure which must be traversed to locate given data determines the feasibility of using Interactive I-D-S/II for a query. The complete range of I-D-S/II Data Manipulation Language is available to the programmer.

PRODUCE INJURED PLAYERS REPORT

A special report containing the names, dates of injuries, and expected dates of return of the players on the injured list is wanted. The end user uses the QRP Query language to produce the report:

```
QUERY LEAGUERECORDADF
REPORT INJURED ON PRINTER
PAGE HEADING IS HEAD1
PAGE FOOTING IS FOOT1.
HEAD1.  LINE "INJURED PLAYERS LIST" COL 30. SPACE 2
        LINE "PLAYER", "DATE INJURED" COL 24, "TYPE INJURY" COL 40,
             "EXPECTED RETURN" COL 60
        SPACE 1
FOOT1.  SPACE 1
        LINE "DATE", %DATE, "PAGE NUMBER" COL 62, %PAGE NUMBER
DETAIL. LINE NAME, DATEINJURED COL 27, TYPEINJURY COL 40, ESTRETURN COL 64
PRINT INJURED
```

The Query statement identifies the Application Definition File to be used. The REPORT line names the procedure to produce the report (INJURED) and indicates the report is to be directed to the printer. The locations of items in the heading are indicated by column number with the first listed item automatically being started in column 1. The percent sign indicates the system is to supply the data, in this case, the current date (that the run is made), and the page number(s). The PRINT instruction calls for the report by the assigned name. The output for this query is as follows:

INJURED PLAYERS LIST

PLAYER		DATE INJURED	TYPE INJURY	EXPECTED RETURN
COLLINS	LEW	770407	SPRAINED ANKLE	770424
STEWART	BOB	770416	FRACTURE-WRIST	770630
ABBOT	NED	770310	NECK INJURY	770420
.				
.				
.				

DATE 770430 PAGE NUMBER 1

ANALYZE GATE RECEIPTS

The league wants a study of gate receipts for each arena of the league. The data to be included in the report for each arena is the game date, name of the visiting team, the team's standing, and the percent of the arena seats sold for that game. Because of the calculations involved, the QRP Procedural Language Processor option (PLP) can be used.

Using PLP, a report is described in the same manner as for Query. (See REPORT statement in Query example of Injured Players List.) If part of this procedure had originally been done using Query Language, the same REPORT statement could be used.

In setting up the calculations for the report, the arena capacity is available by retrieval of the home team's record. The game record for the first game at the specified arena is located and the game date is then available, as is the number of tickets sold. The game record serves as a relationship record for home and visiting teams, so the visiting team record is easily located using the relationship. The visiting team's standing must be calculated using the won-lost data from that team's record. The percentage of seats sold is calculated using fields retrieved from the home team record and game record (capacity and tickets sold).

The two calculated values (percent sold and visiting team's standing), and the information from the records (game date and visiting team name) are printed. The retrieval, calculation, and printing is done automatically for each game at that arena. The game record for the next game played at the arena is found and the procedure is repeated, etc. When all games at that arena have been found, "processed," and printed, the procedure terminates.

The QRP REPORT statement causes the appropriate page heading lines and detail lines to be produced via the PRINT statement.

If additional processing that entailed inserting data in the data base were required, the complete procedure as it was written (with the added coding) could be run. No rewriting would be necessary.

The report developed from the procedure would be as follows:

ICE PALACE GATE RECEIPT STUDY

DATE OF GAME	VISITING TEAM	STANDING OF TEAM	PERCENT OF ARENA SEATS SOLD
770410	BULLDOGS	.667	95
770412	TERRIERS	.500	95
770413	DASHERS	.333	90

PRODUCE A QUICK REPORT, SUBJECT TO CHANGE

Late in the week, a user requests a weekly report to be produced only for the remainder of the season. The first weekly report is to show each of the team's players total playing time and total score for the previous month for all out-of-town games, with the subsequent weekly reports to show accrued totals. However, the user may want to change the report to include home games, or incorporate other data.

The time frame for the availability of the reports is for the first report to be available after the following Sunday's results are entered, and the subsequent weekly reports to be available as soon as possible after each Sunday's game results have been entered.

Because the report is both required immediately and is subject to change, the programmer chooses QRP which eliminates the need for knowing data structure and permits the programmer's time to be used in writing the statements that will process the requested data for the report. Since all statistical data is permitted to all users, the program can use an existing ADF for the data descriptions.

REQUEST A PLAYER'S HISTORY

One report that will be used by all teams either on a regular basis for all players, or on a request basis for a particular player is a complete report of the player's history.

If the program were written as a transaction, the user could request the information for a particular player at any time from a remote terminal, and receive the listing at the terminal. The transaction would require input, and this could be listed with the transaction name if the user were familiar with the use of the transaction, or if not, the transaction could be written to prompt for the necessary information. This report could be requested for a period starting with the first game of the season, and carry through to include the most recently entered history.

In the example below, the system allows the user to determine a player's record for the season against all teams in the league or a specific team; for performance against a specific team, the record for an individual game can be obtained. Several variations of a "RECORD" transaction are shown:

```
recrd,paulson
ALL TEAMS OR ONE?
all
PAULSON HAS PLAYED 37 GAMES AND SCORED 406 POINTS
recrd,paulson
ALL TEAMS OR ONE?
greyhounds
ALL GAMES OR ONE?
all
PAULSON HAS PLAYED 3 GAMES AND SCORED 47 POINTS AGAINST - GREYHOUNDS
recrd,smith
ALL TEAMS OR ONE?
greyhounds
ALL GAMES OR ONE?
770412
TEAMS DID NOT PLAY THAT DAY; TRY AGAIN
770413
SMITH SCORED 12 POINTS AGAINST GREYHOUNDS ON 770413
```

Another example of a conversational type transaction is shown below. In this example, the TPR's are programmed to allow the user to browse through the data base while developing the query. In this example, the name of the transaction is BROWSE.

NOTE: 0 is entered when prompting is needed.

```
browse
TYPE OF INFO NEEDED? 0
     1 BY DIVISION
     2 BY TEAM
     3 BY PLAYER
TYPE OF INFO NEEDED? 3
ENTER PLAYER NAME? jones a
SPECIFY TEAM - PLAYER NAME JONES A IS LISTED FOR BULLDOGS, GREYHOUNDS
? bulldogs
WHICH STATISTICS? 0
     1 POINTS
     2 REBOUNDS
     3 FOULS
WHICH STATISTICS? 1
SEASON TOTAL IS 352 POINTS; DO YOU NEED MORE? yes
WHICH BREAKDOWN? 0
     1 AGAINST A TEAM
     2 INDIVIDUAL GAME
WHICH BREAKDOWN? 1
WHICH TEAM? greyhounds
SEASON TOTAL AGAINST GREYHOUNDS IS 43 POINTS IN 3 GAMES
OTHER STATISTICS NEEDED FOR THIS PLAYER? no
TYPE OF INFO NEEDED? 3, smith lr, retrievers, 2
SEASON TOTAL IS 105 REBOUNDS; DO YOU NEED MORE? no
```

In the QRP environment, procedures for producing these reports would be stored in the user's file to be available at any time without the necessity of generating a query each time information is required. Again, as in the transaction environment, the user would supply parametric information at execution time.

REQUEST AN UPDATED REPORT

After the day's game history information has been entered into the system, the new data can be used in calculating updates to all reports dependent on game history and players' statistics. One of these reports is the standings of the teams within the league. The manipulation of data that is necessary to produce the report is accomplished using the Transaction Processor -- treating the production of the report as a transaction.

To obtain the League Standing report, the user need only request the report (LSR transaction in the example below) and furnish the date for which he wants the report. In this example we assume the most current date's standings are wanted. (The histories from yesterday's games will have already been entered, but today's games have not yet been played.) The request, entered from a remote terminal, might progress as follows. (The user has signed onto the Transaction Processor system.)

```
THIS IS TP **HELLO**
LSR                          (user request for LSR transaction)
FOR WHAT DATE - (yesterday's date)
```

The report showing teams' standings by division and conference within the league can be printed just below the conversation shown above.

Actually when a transaction requires so little input and is, as in this example, a transaction that will be used frequently, it is much more realistic to assume the terminal operator requesting the report would prefer to immediately give the transaction all the information it needs rather than wait for a prompt. If the transaction were so programmed, the exchange at the terminal could be as follows:

```
THIS IS TP **HELLO**
LSR, (yesterday's date)
```

The user supplies the date at the same time the transaction is requested and the report can then follow immediately after. The League Standings report can be printed at many terminals automatically so that the information is immediately available to all interested parties. These parties could include not only the teams themselves, but also the news media. This listing, originating from the League office, could then be considered the current "official" Standings report.

USE DM-IV JOURNALS

Auditors are checking the league's books and during the audit request the financial office to supply a journal reflecting the accounts payable transactions for the second quarter so they can check those purchase order numbers that have not been justified against an account.

The office personnel realize that the auditors can use any complete list of transactions for that quarter instead of the actual (printed) journals that are kept in the office. Since DM-IV journalizes all transactions that are entered into the system as a precaution against system failure and a means of restoring the system after the failure, there already exists a complete list of transactions on magnetic tape.

The auditors are able to write a program to scan the tape and list those account names which carried the missing purchase order numbers, thereby providing them with the information they need in a fraction of the time it would have required to manually compare all the journal entries in the printed journals.

LOCATE DATA ITEM NAMES

Find Particular Data Names

If, in requesting a player's history, the user does not want a complete list of all statistics, the QRP DESCRIBE subsystem can be used to find the data names for items the user wants listed after the DATA ITEMS TO BE SUPPLIED - prompt. These items are listed by data names that are specified in the ADF entries, and which the user can list using DESCRIBE as follows:

```
*DMIV
*DESCRIBE
   .
   .
   .
ADF OBJECT FILE --statisticsadf
DATA-FILE-REFERENCE  ?                        (user does not know)
DATA-FILE-REFERENCES ARE
  PLAYERSTATS
  GAMESTATS
DATA-FILE-REFERENCE --playerstats
   .
   .
   .
ENTRY --playername                            (user knows this name)
ITEM --?                                      (but requests item names)
  MINPLAYED     2,N                           (two numeric characters)
  FIELDGOALS    5,A                           (five alphanumeric characters)
  FREETHROWS    5,A                           (etc.)
  REBOUNDS      2,N
  ASSISTS       2,N
  TOTPTS        3,N
   .
   .
   .
```

The DESCRIBE subsystem has now given the user the names and characteristics of the data items contained within the user's virtual record. These data items are available through all QRP/PLP components.

Find All Data Names

Had the user wanted to know names of all the data items in the entire file, DESCRIBE would have listed them if the user had responded to the DATA-FILE-REFERENCE request with the word ENTIRE instead of the interrogation mark (?) as shown above.

```
    .
    .
    .
ADF OBJECT FILE - statisticsadf
DATA-FILE-REFERENCE - entire

  (a complete listing of the file follows)
```

SECTION IV

FACILITIES FOR THE DATA PROCESSING STAFF

DM-IV provides a variety of programming and administrative aids to help the data processing staff to efficiently and effectively satisfy the information requirements of the user.

The responsibilities of the data base administrator (DBA) include the design, creation, implementation, modification, and maintenance of the data base. In describing the logical structure of the data base, the DBA uses the schema Data Description Language (DDL); the storage structure is described using the schema Device Media Control Language (DMCL). When these descriptions are translated, several reports are produced. Besides listing diagnostics from the translation process, the reports provide cross references and extracts by type of entity named, storage capacities, and distribution of records to the storage media.

Similarly, reports are produced when the subschema DDL is translated and validated to ensure that it is a proper subset of the schema.

To help ensure the security of the data base, some reports can only be produced after privacy locks are satisfied by privacy keys supplied by the DBA. Only the DBA is allowed to validate subschemas; the privacy locks also prevent unauthorized listing of the schema or subschema definitions or of the run-time privacy locks. Another report lists the names of all programs that have been compiled against each subschema.

The DM-IV Data Manager includes several utility programs. Using these utilities, data base files can be built, copied, and compressed. These utilities also provide analysis functions which permit the DBA to examine the distribution of records and available space on the data base.

The use of the DM-IV Data Manager instead of traditional file structures should reduce the programming effort. Definition and documentation of the data base are done by the DBA; the programmer concentrates on accessing the needed records and programming the application processing.

The use of QRP or PLP further reduces the programmer's level of concern with the data base. The information needed for accessing records and fields in the data base is contained in the Application Definition File, so it does not have to be supplied by the programmer. The programmer or end user can use the DESCRIBE facility to obtain a listing of the names and descriptions of the fields which the programmer is entitled to use.

The QRP DIALOG facility can be used for determining the content of fields in the data base. The DBA (or a programmer) can also use Interactive I-D-S/II to access the data base. In Interactive I-D-S/II, the programmer has available the full range of Data Manipulation Language (DML) statements, including the use of privacy keys. Besides serving as a vehicle for quick retrievals or updates, Interactive I-D-S/II can be used to test the logic of a program's usage of DML.

The DM-IV/QRP PERFORM subsystem can be used by the DBA to interactively generate the Job Control Language needed for executing any of the Data Manager or QRP/PLP utility programs.

Like the Data Manager, the Transaction Processor gives the DBA reports which can be used to monitor performance and workload. Through these reports, it is possible to determine which transactions represent the bulk of the workload, which terminals are most active, and various internal factors, including resource utilization.

INDEX

CUT ALONG

HONEYWELL INFORMATION SYSTEMS

Technical Publications Remarks Form

TITLE: SERIES 60(LEVEL 66) DATA MANAGEMENT-IV SYSTEM OVERVIEW, ADDENDUM B

ORDER NO. DF73B, REV. 0

DATED JULY 1979

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be promptly investigated by appropriate technical personnel and action will be taken as required. If you require a written reply, check here and furnish complete mailing address below. ☐

FROM: NAME _____ **DATE** _____

TITLE _____

COMPANY _____

ADDRESS _____

_____

PLEASE FOLD AND TAPE–
NOTE: U. S. Postal Service will not deliver stapled forms

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 39531 WALTHAM, MA02154

POSTAGE WILL BE PAID BY ADDRESSEE

HONEYWELL INFORMATION SYSTEMS
200 SMITH STREET
WALTHAM, MA 02154

ATTN: PUBLICATIONS, MS486

Honeywell

Honeywell

Honeywell Information Systems
In the U.S.A.: 200 Smith Street, MS 486, Waltham, Massachusetts 02154
In Canada: 2025 Sheppard Avenue East, Willowdale, Ontario M2J 1W5
In Mexico: Avenida Nuevo Leon 250, Mexico 11, D.F.

19212, 35877, Printed in U.S.A.

DF73, Rev. (