60482200

CONTROL DATA CORPORATION

DMS-170

FORTRAN DATA BASE FACILITY VERSION 1 REFERENCE MANUAL

CDC® OPERATING SYSTEMS:
NOS 1
NOS/BE 1

REVISION RECORD	
REVISION	**DESCRIPTION**
A	Original release.
(12-22-78)	
B	This revision documents version 1.1 of FDBF. Major changes include OPEN and CLOSE options for
(7-20-79)	relations, listing control directives in DDLF output, support of data base status block, support of
	library compaction facility, and checksum capability. Minor technical changes and corrections are
	included.
C	This revision documents version 1.2 of FDBF which provides an interface to FORTRAN 5 and
(12-17-79)	language extensions that include array declarations, CHARACTER and BOOLEAN data types,
	and error and end-of-file specifiers on DML statements. An enhanced capability for
	FORTRAN 4 to use a special long variable to correspond to character data in the schema is
	also documented.
D	Released at PSR level 528. This revision documents the DML START statement and the file position
(10-31-80)	field of the data base status block.
Publication No. 60482200	

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning this manual to:

CONTROL DATA CORPORATION
Publications and Graphics Division
215 MOFFETT PARK DRIVE
SUNNYVALE, CALIFORNIA 94086

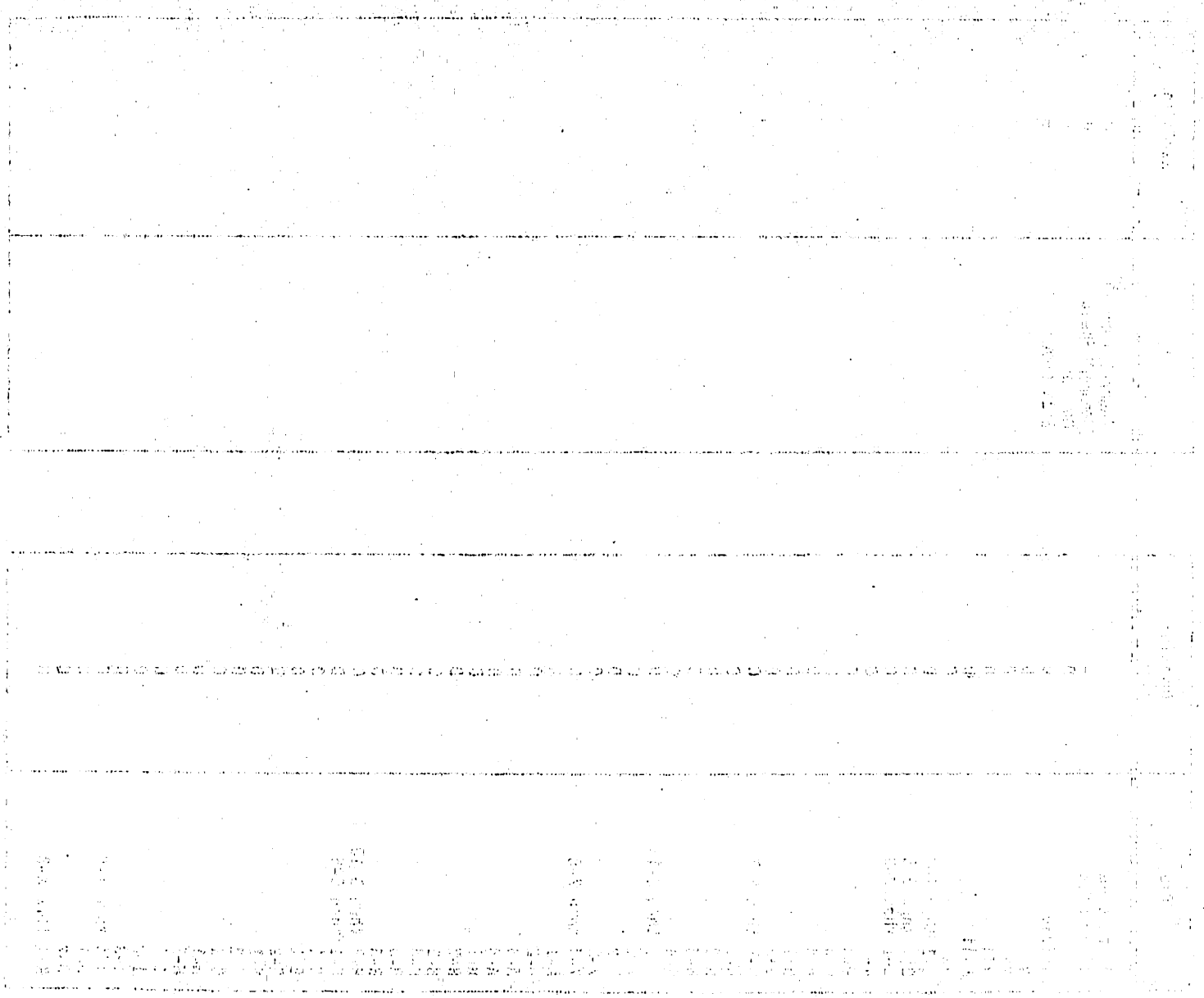
or use Comment Sheet in the back of this manual

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

Page	Revision
Front Cover	-
Title Page	-
ii	D
iii/iv	D
v	D
vi	D
vii	D
viii	D
ix	C
1-1 thru 1-7	D
2-1 thru 2-5	C
3-1 thru 3-3	C
4-1	C
4-2	C
4-3	D
4-4	D
5-1	D
5-2	D
5-3 thru 5-7	C
6-1	C
6-2	D
6-3	C
6-4	C
6-5 thru 6-15	D
7-1	C
7-2	D
7-3	D
7-4 thru 7-8	C
A-1	A
A-2	D
A-3	B
A-4	B
B-1	D
B-2	D
B-3	C
B-4	C
B-5	D
B-6	D
B-7 thru B-10	C
C-1 thru C-5	D
D-1	D
E-1	D
E-2	D
F-1	D
F-2	D
G-1	C
H-1	D
H-2	D
I-1	C
I-2	D
I-3	D
I-4 thru I-9	C
J-1	C
J-2	C
K-1 thru K-3	D

Page	Revision
Index-1 thru -3	D
Comment Sheet	D
Mailer	-
Back Cover	-



PREFACE

This manual describes the FORTRAN Data Base Facility (FDBF) within DMS-170, a data management system developed by Control Data Corporation. FDBF includes the Data Description Language for the FORTRAN Sub-Schema (FORTRAN/DDL) and the Data Manipulation Language (DML). FDBF enables FORTRAN Extended 4 and FORTRAN 5 programs to access a data base.

As described in this publication, FORTRAN Data Base Facility Version 1.2 operates under control of the following operating systems:

- NOS 1 for the CONTROL DATA® CYBER 170 Series; CYBER 70 Models 71, 72, 73, and 74; and 6000 Series Computer Systems
- NOS/BE 1 for the CDC® CYBER 170 Series; CYBER 70 Models 71, 72, 73, and 74; and 6000 Series Computer Systems

This manual is designed for use by the data administrator or the staff member responsible for describing FORTRAN sub-schemas and for use by the FORTRAN applications programmer writing programs that access the data base. It is assumed that the user is an experienced programmer and has used Control Data computers and software products. It is also assumed that the user is familiar with the FORTRAN programming language.

The FDBF user can find additional pertinent information in the Control Data Corporation publications.

The NOS Manual Abstracts and the NOS/BE Manual Abstracts are instant-sized manuals containing brief descriptions of the contents and intended audience of all NOS and NOS product set manuals, and NOS/BE and NOS/BE product set manuals, respectively. The abstracts manuals can be useful in determining which manuals are of greatest interest to a particular user. The Software Publications Release History serves as a guide in determining which revision level of software documentation corresponds to the Programming Systems Report (PSR) level of installed site software.

The publications are listed alphabetically in groupings that indicate relative importance to readers of this manual.

The following publications are of primary interest:

Publication	Publication Number
CYBER Database Control System Version 2 Reference Manual	60481800
DMS-170 DDL Version 3 Reference Manual Volume 1: Schema Definition for Use With: COBOL FORTRAN Query Update	60481900
FORTRAN Extended Version 4 Reference Manual	60497800
FORTRAN Version 5 Reference Manual	60481300

The following publications are of secondary interest:

Publication	Publication Number
CYBER Record Manager Advanced Access Methods Reference Manual	60499300
DMS-170 DDL Version 3 Reference Manual Volume 2: Sub-Schema Definition for CYBER Database Control System Use With: COBOL Query Update	60482000

Network Products Transaction Facility Version 1 Reference Manual	60455340
NOS Version 1 Manual Abstracts	84000420
NOS Version 1 Reference Manual, Volume 1 of 2	60435400
NOS/BE Version 1 Manual Abstracts	84000470
NOS/BE Version 1 Reference Manual	60493800
Software Publications Release History	60481000

CDC manuals can be ordered from Control Data Corporation, Literature and Distribution Services, 308 North Dale Street, St. Paul, Minnesota 55103.

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or parameters.

CONTENTS

APPENDIXES

INDEX

FIGURES

TABLES

NOTATIONS USED IN THIS MANUAL

The specifications for each FORTRAN/DDL and FORTRAN/DML statement are described in a reference format. The notations used in the reference formats are described as follows:

UPPERCASE	Words are reserved words and must appear exactly as shown. Reserved words can be used only as specified in the reference formats.
Lowercase	Words are generic terms that represent the words or symbols supplied by the user. When generic terms are repeated in a format, a number is appended to the term for identification.
[ ] Brackets	Enclose optional portions of a reference format. All of the format within the brackets can be omitted or included at the user's option. If items are stacked vertically within brackets, only one of the stacked items can be used.
{ } Braces	Enclose two or more vertically stacked items in a reference format when only one of the enclosed items must be used.
... Ellipses	Immediately follow a pair of brackets or braces to indicate that the enclosed material can be repeated at the user's option.

Punctuation symbols shown within the formats are required unless enclosed in brackets and specifically noted as optional. One or more spaces separate the elements in a reference format.

DATA BASE PROCESSING WITH DMS-170

The DMS-170 software package functions as the data management system for Control Data computer systems. Through this data management system, a data base can be defined, maintained, and controlled in an environment totally independent of the applications that are accessing it. Conventional files otherwise owned and processed by a number of distinct applications can be described through the data description language facilities of DMS-170. Consequently, the responsibility for tasks such as data description, data conversion, and validity checking is transferred from the applications programmer to the data administrator.

The DMS-170 data management system is composed of the following elements:

- Data Description Language (DDL), which creates the schema definition, as well as the COBOL and Query Update sub-schema definitions.
- CDC® CYBER Database Control System (CDCS), which controls, monitors, and interprets data base requests from COBOL, FORTRAN, and Query Update applications programs.
- CDC® CYBER Record Manager (CRM), which handles all input/output processing requests on a data base from an applications program.
- FORTRAN Data Base Facility, which is composed of a FORTRAN sub-schema definition language and a Data Manipulation Language (DML).

Each element of the DMS-170 system is used either in the definition or in the processing of a data base. The definition of the data base is accomplished through the capabilities of DDL and the master directory utility, one of the data base utilities provided by CDCS. Processing of the data base involves retrieval and updating of the data by applications programs through the facilities of CDCS.

The FORTRAN Data Base Facility allows a FORTRAN programmer to access a data base by inserting DML statements, which are similar to FORTRAN statements, into an applications program. Before the FORTRAN program is compiled, the DML preprocessor translates the special DML statements into FORTRAN statements and produces a modified version of the program for compilation. The FORTRAN Data Base Facility supports both FORTRAN Extended 4 and FORTRAN 5 applications programs.

DATA BASE DEFINITION

The responsibility for the definition of a data base lies with the data administrator. The data administrator is a person or group of persons who have the task of developing and defining the data base as well as monitoring and controlling the day-to-day processing of that data base. The relationship of the elements involved in defining a data base is shown in figure 1-1.

To define a data base, the data administrator uses the Data Description Language (DDL). Through this language, four types of data descriptions can be created: the schema, the COBOL sub-schema, the FORTRAN sub-schema, and the Query Update sub-schema. Each of these data descriptions follows specific structuring conventions, includes unique clauses and statements, and conforms to an individual set of rules. Once the schema and COBOL, FORTRAN, and Query Update sub-schema descriptions have been created and compiled, the data administrator creates the master directory through one of the data base utility routines provided as a part of the CDCS data management services.

SCHEMA DEFINITION

The schema is a detailed description in English-like syntax of the data in a data base. An installation can have many data bases, but only one schema is allowed for each data base. The schema description is generated by DDL statements that name the schema, organize the schema into files, describe each record type together with the characteristics of the data comprising the record, and describe relationships and constraints among files. The schema also includes access control locks that provide privacy at the file level. The DDL source statements describing the data are used as input to the DDL compiler and are compiled into an object schema, or schema directory. The data administrator then uses the schema description to define any number of sub-schemas.

SUB-SCHEMA DEFINITIONS

A sub-schema is a detailed description of selected portions of a data base to be used by applications programs. Although only one schema definition is allowed for each data base, any number of sub-schemas can be defined to meet the needs of different types of applications. Sub-schemas are defined by the data administrator for use by applications programs written in the COBOL, FORTRAN, and Query Update languages; the sub-schema descriptions are based on the schema definition.

COBOL Sub-Schemas

A COBOL sub-schema is defined through the capabilities of the DDL language. COBOL sub-schemas describe in COBOL-like syntax the parts of a data base that can be accessed by a COBOL program. Data descriptions in COBOL sub-schema source statements are written to correspond to data descriptions in the schema. Certain differences are allowed to exist; these differences are resolved by DDL and CDCS. The COBOL sub-schema description is generated by DDL source statements that identify the schema and sub-schema, specify files and the content and structure of records, identify relations among files to be used, specify record qualification for relation processing, and indicate any changes in data format required by the applications program.

The DDL source statements describing the sub-schema are compiled by the DDL compiler into an object sub-schema, or COBOL sub-schema directory. The schema must be compiled, however, before any sub-schemas using it can be compiled. A COBOL programmer then uses a listing of the sub-schema to learn the names and descriptions of the data to be referenced in the COBOL program.

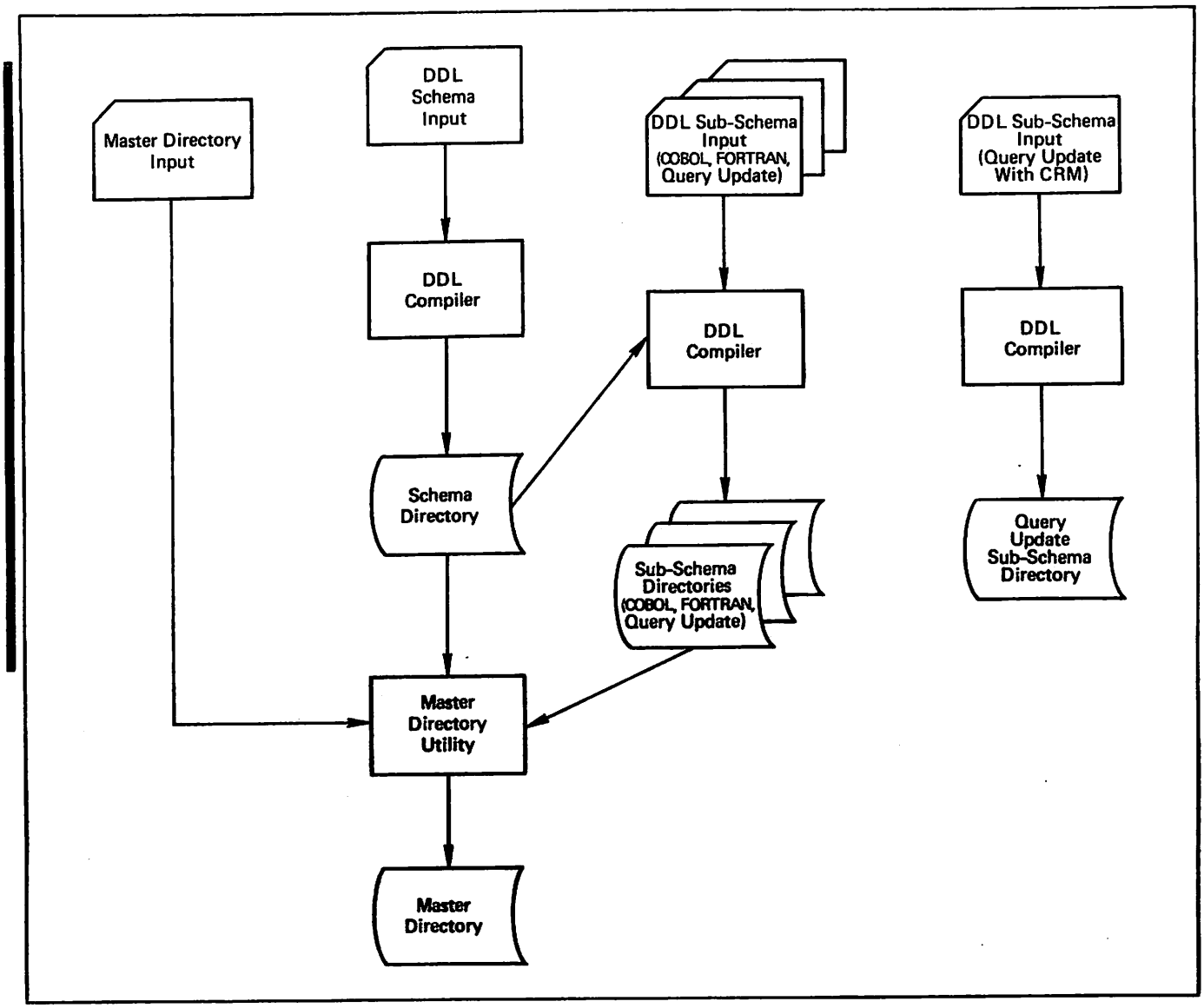


Figure 1-1. Data Base Definition With DMS-170

FORTRAN Sub-Schemas

A FORTRAN sub-schema is defined through the facilities of the DDL language. FORTRAN sub-schemas use statements similar to FORTRAN specification statements to describe the parts of a data base that can be accessed by a FORTRAN program. Data descriptions in FORTRAN sub-schema source statements are written to correspond to data descriptions in the schema. Certain differences are allowed to exist; these differences are resolved by DDL and CDCS. The FORTRAN sub-schema description is generated by DDL source statements that identify the schema and sub-schema, specify files and the content and structure of records, indicate changes in data format required by the applications program, identify relations among files to be used, and specify record qualification for relation processing.

FORTRAN sub-schemas, like COBOL sub-schemas, cannot be compiled until the schema being used has been compiled. Once the schema has been compiled, the DDL source statements describing each sub-schema are compiled by the DDL compiler into an object sub-schema, or FORTRAN sub-schema directory. A listing of the sub-schema is used by the FORTRAN programmer to obtain the names and descriptions of the data to be referenced in the FORTRAN program.

Query Update Sub-Schemas

Query Update sub-schemas are defined through the capabilities of the DDL language. A Query Update sub-schema describes the portion of a data base that can be accessed by a Query Update user in either CDCS data base access mode or CRM data base access mode.

CDCS Data Base Access Mode

The data descriptions in Query Update sub-schemas in CDCS data base access mode are written to correspond to data descriptions in the schema. Certain differences between the sub-schema and schema data descriptions are allowed to exist; these differences are resolved by DDL

and CDCS. For each sub-schema, the DDL source statements used as input to the DDL compiler name the schema and sub-schema, specify needed files and the content and structure of records, identify relations among files to be used, specify record qualification for relation processing, and indicate any changes in data format required by the Query Update program.

After the schema has been compiled, the DDL source statements describing the sub-schema are compiled by the DDL compiler into an object sub-schema, or sub-schema directory. The names and descriptions of data to be referenced in a Query Update program are obtained from a listing of the sub-schema.

CRM Data Base Access Mode

The data descriptions in Query Update sub-schemas in CRM data base access mode are not based on a schema definition. Schema-defined files can be accessed, however, providing the DDL source statements that define the sub-schema describe the data exactly as it is described in the schema. Appendix K contains a summary of data definition in DMS-170 to aid in accessing schema-defined files. For each sub-schema, the DDL source statements used as input to the DDL compiler name the sub-schema, specify needed files, define the content and structure of records, identify relations among files to be used, and specify record qualification for relation processing.

After the DDL source statements describing the sub-schema have been written, they are compiled into an object sub-schema, or sub-schema directory. The names and descriptions of data to be referenced in a Query Update program are obtained from a listing of the sub-schema.

MASTER DIRECTORY CREATION

The master directory must be constructed by the data administrator before any applications programs accessing data base files can be executed. The master directory is a file containing information relating to all data bases; schemas; and COBOL, FORTRAN, and Query Update sub-schemas known to CDCS. In addition to containing information about logging specifications and data base procedure libraries, the master directory functions as the source of all data base and media descriptions for CDCS. To create or update the master directory, the data administrator uses the DBMSTRD utility, one of the data base utilities provided through CDCS. Input to the utility for a creation run consists of three types of subentries. These subentries contain information relating to schemas, files, and sub-schemas. In specific clauses, the data administrator specifies information to be used by CDCS in attaching data base files and associated index files, as well as information regarding logging criteria and data base procedure libraries. After the master directory has been generated, it must be stored as a permanent file.

In the process of maintaining a data base environment, the data administrator might want to add information for one or more new schema definitions, delete or modify existing schema information, or modify permanent file information for data base files and procedure library files. Under any of these circumstances, appropriate changes must be made to the master directory through a modification run. A new data base cannot be accessed by applications programs until the appropriate information from the corresponding schema is added to the master directory. Similarly, when information pertaining to a schema is deleted from the master directory, the schema can no longer be used by an application. One other form of modification allows the addition or deletion of information pertaining to sub-schemas. No sub-schema can be referenced by a user during execution unless information about that sub-schema exists in the master directory. Modification of permanent file information for data base files and procedure library files is also allowed. If there are changes in the permanent file information that is required to attach a data base file or procedure library file, the information in the master directory must be modified so that CDCS has the correct information with which to attach the file.

DATA BASE PROCESSING

Once a data base has been defined by the data administrator, it can be accessed and modified by users of the COBOL, FORTRAN, and Query Update languages. The relationship of the elements involved in processing a data base is shown in figure 1-2.

Several special features of CDCS are involved in the processing of a data base. Two or more users can access the same data base file at the same time through the concurrency feature. Files can also be locked and unlocked during concurrent update operations. Data privacy controls can be specified at the file level through access control locks defined in the schema. User programs must supply appropriate key values to gain access to files having these locks.

An additional feature of CDCS processing is the relational data base facility, which allows data from several linked files to be retrieved with a single read request.

The CDCS constraint facility allows the data administrator to impose controls on update operations involving logically related files. The use of constraints protects the underlying relationship between files or between items within a file that might be altered as a consequence of an update.

CDCS provides a data base procedure linkage to allow special purpose subprograms written by the data administrator to be called when specific situations occur during CDCS processing. The procedures can perform functions supplemental to those provided by CDCS.

The input/output capabilities of CRM handle all data base processing requests from an applications program. Data base files are processed by CRM Advanced Access Methods (AAM) according to the requirements and restrictions for conventional files.

The logging and recovery facilities of CDCS provide an essential service within the data base environment. Through these facilities, the integrity of a data base can be preserved; that is, a destroyed data base can be reconstructed, and an invalid data base can be restored to a previous state.

APPLICATION LANGUAGES

The data in a data base can be accessed by the following application languages: COBOL 5, FORTRAN Extended 4 (called FORTRAN 4), FORTRAN 5, and Query Update. Processing of the data base by COBOL, FORTRAN, and Query Update programs is controlled and monitored by CDCS. These application languages can be used in either batch or interactive mode.

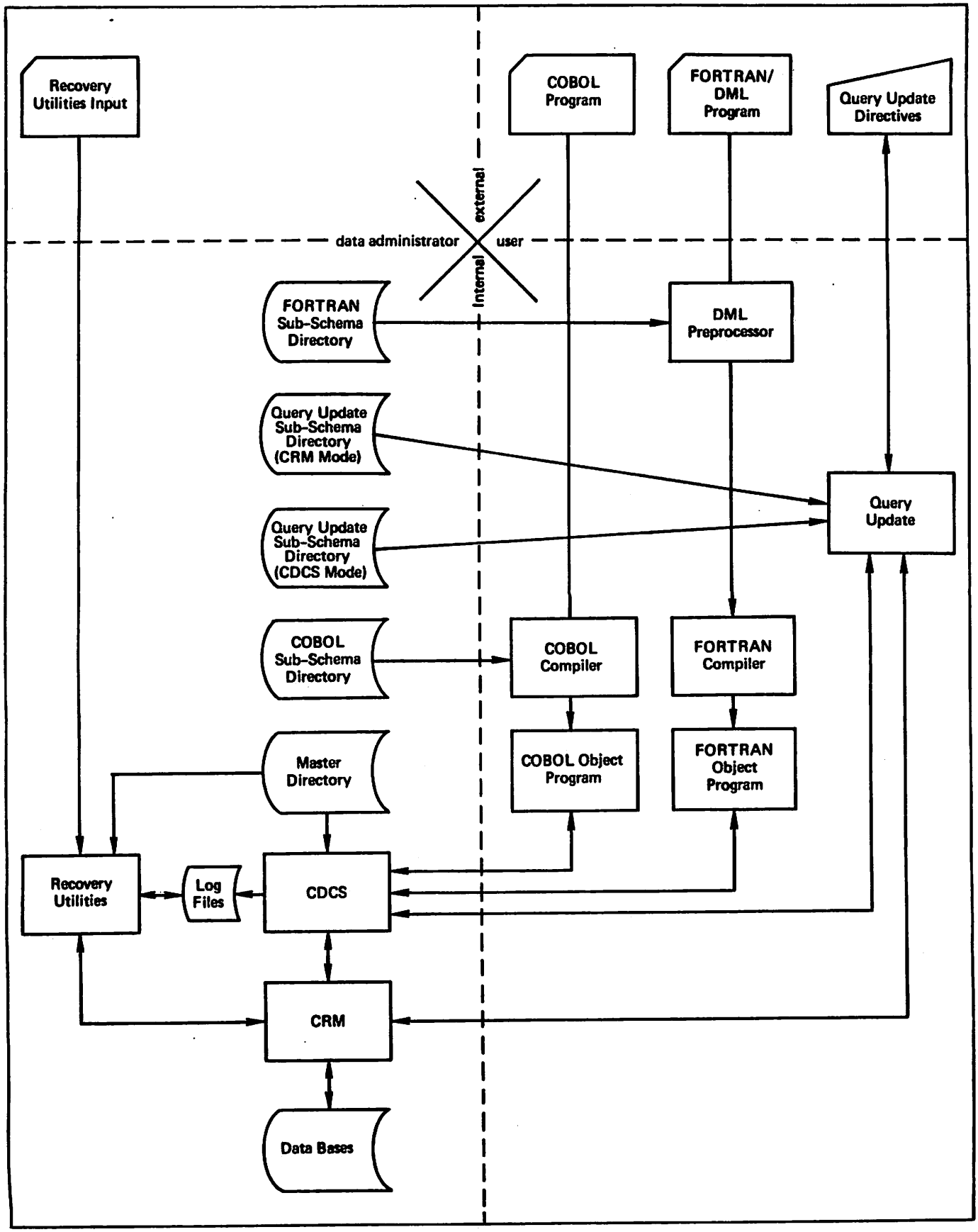


Figure 1-2. Data Base Processing With DMS-170

COBOL Processing

A COBOL program accesses data base files through conventional input/output statements. The files are opened and closed and records are read, written, deleted, and updated using the same means as for files that are not part of a data base. Relation processing is also accomplished by conventional COBOL statements. Data retrieved by the program is accessed in accordance with the way it is described in the COBOL sub-schema.

When a COBOL program utilizing CDCS is to be compiled, the file containing the sub-schema directory must first be attached. Once the program is compiled using the sub-schema, it can be executed later without reattaching the sub-schema directory.

Execution of an input/output statement for a data base file in a COBOL program causes the COBOL object-time routines to route I/O calls to CDCS. CDCS uses AAM for input/output processing.

FORTRAN Processing

A FORTRAN program accesses data base files through the FORTRAN Data Base Facility. The FORTRAN Data Base Facility consists of the FORTRAN sub-schema definitions (already discussed in this section) and the Data Manipulation Language (DML). The DML consists of a series of FORTRAN-like statements that are coded within the logic of a FORTRAN program. These statements allow the FORTRAN user to access and modify data base files.

Before a program containing FORTRAN DML statements is compiled, the DML preprocessor is called via a control statement to translate the DML statements into FORTRAN specification statements and CALL statements. Data descriptions are obtained from the FORTRAN sub-schema directory, which must be attached during the preprocessing phase. Following the preprocessing, compilation of the FORTRAN program proceeds as for a conventional FORTRAN program; the translated DML statements are compiled like other FORTRAN statements. Once the program is preprocessed using the sub-schema, it can be compiled and executed later without reattaching the sub-schema directory.

When a FORTRAN program utilizing the FORTRAN Data Base Facility is executed, CDCS controls all processing of data base files. CDCS in turn uses AAM for the input and output operations.

Query Update Processing

Query Update functions within the data base environment whenever a Query Update sub-schema is specified by a Query Update user. The Query Update language, which is a special nonprocedural, interactive language, can be used by both programmers and nonprogramming personnel to perform several functions. Through simple commands, search, retrieval, update, and display operations can be performed on data base files as well as on conventional files. In addition, a single Query Update command can be used in relation processing to display to the user data from more than one file. A comprehensive report writing capability is an integral part of Query Update.

A Query Update user can access data base files in CDCS data base access mode or in CRM data base access mode. In CDCS data base access mode, CDCS controls all file processing. CDCS in turn uses AAM for input and output operations. The concurrency, privacy checking, logging, and recovery features of CDCS are utilized by Query Update.

In CRM data base access mode, Query Update does not interface with CDCS to access data base files. Input/output processing requests on data base files are handled directly by AAM. To access schema-defined files, the sub-schema must describe the data exactly as it is described in the schema (see appendix K).

TRANSACTION PROCESSING

CDCS supports the Transaction Facility (TAF), which allows processing in transaction mode under NOS. Transaction processing allows high speed handling or repetitive executions of a relatively small number of jobs called tasks. The tasks can be executed by many different people from many locations. A task usually performs one of the following manipulations on a data base:

- Stores a new record
- Alters or deletes an existing record
- Produces formatted output

An on-line teller system is an example of transaction processing: tellers in many locations use terminals connected on-line to a central processor to make deposits or withdrawals for an account and to print confirmations. A task (a deposit or withdrawal) is initiated by a teller through the terminal; once initiated, the task is executed through TAF and CDCS. The task can communicate with the terminal through TAF and the Network Access Method (NAM) and can initiate subsequent tasks.

Figure 1-3 shows the CDCS/TAF interface. Access to the data base through TAF is concurrent with access in batch mode. All access to the data base is monitored by CDCS.

The CDCS/TAF interface supports tasks coded in the COBOL 5 and FORTRAN Extended 4 programming languages. The syntax used to code a task is the same as the syntax used to code a program for execution in batch mode through CDCS, with a few exceptions. The coding for a task must include TAF directives and must provide for a communication block. Also, TAF prohibits a task from making some requests that are allowed to be made by an applications program executing in batch mode. The Transaction Facility reference manual contains detailed information about data base processing through TAF.

CONCURRENCY

An important feature provided through CDCS is the concurrency feature. Concurrency means that two or more applications programs can access the same data base file at the same time. Programs can access a file concurrently for retrieval or update purposes. During concurrent update operations, CDCS provides a locking mechanism by which files and records can be locked and unlocked at appropriate times. Automatic locking and unlocking are performed by CDCS when certain input/output operations are specified. In addition, explicit lock and unlock requests can be issued from the applications program.

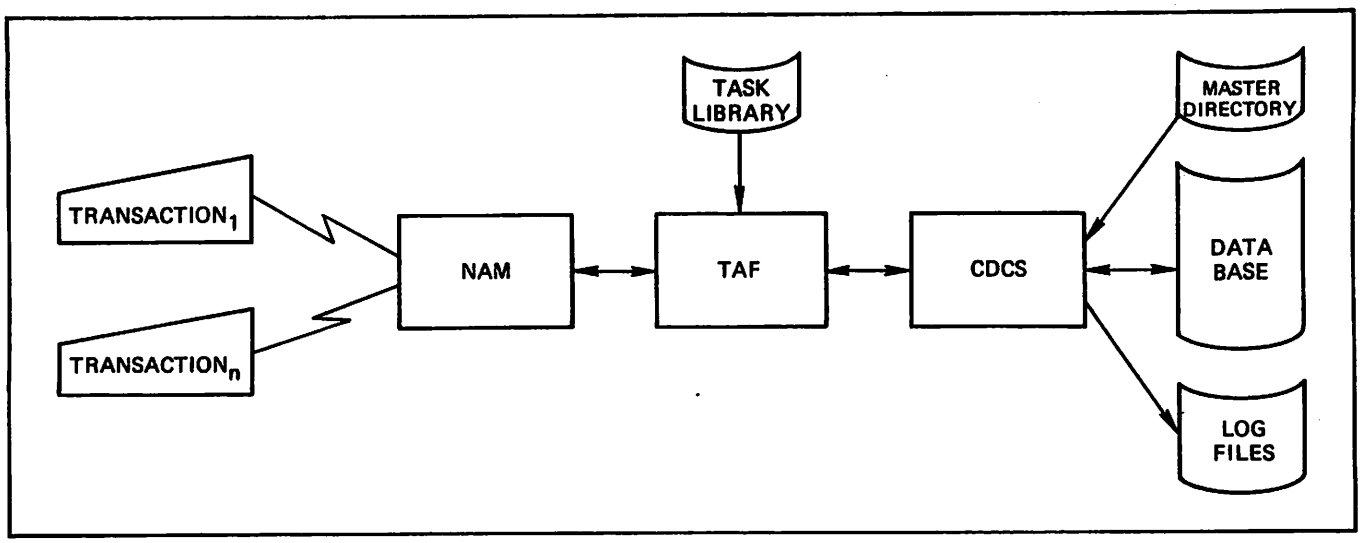


Figure 1-3. CDCS/TAF Interface

A deadlock situation can occur when two programs attempt to access files or records that have been locked by CDCS or by other programs. When this situation occurs, CDCS selects one of the contending programs and releases all locked resources held by that program. Appropriate code to handle recovery from a deadlock should be included in applications programs.

FILE PRIVACY

Another valuable function provided by CDCS is the privacy checking mechanism. Through this mechanism, access to data base files can be controlled on the basis of criteria specified in a data base procedure or on the basis of access control locks declared in the schema.

When a data base procedure is used for privacy checking, the procedure decides whether to allow the use of a data base file. The decision is based on the privacy key supplied by the applications program and on the job name of the program.

When access control locks are used for privacy checking, the ACCESS-CONTROL clause in the schema specifies the locks that apply to the use of a file. The applications programs must specify the appropriate privacy keys to gain access to the files at execution time.

RELATIONS

The relational data base facility of CDCS allows an applications program to access data from related files with a single read request. In the schema definition, the data administrator links files together into a logical, meaningful relationship, called a relation, by specifying a relation entry. The relation entry assigns a name to the relation and specifies the data items to be used to link the files.

The COBOL, FORTRAN, and Query Update users access relations based upon the relational information contained in the respective sub-schemas. The relations in the COBOL and FORTRAN sub-schemas are in turn based on the relations defined in the schema. The relations in the Query Update sub-schema in CDCS data base access mode are also based on the relations defined in the schema; the Query Update sub-schema in CRM data base access mode directly defines a relation. In addition to the relation definitions, the sub-schemas for all three application languages can specify qualification criteria for retrieving only certain records from the data base files joined in the relation.

An applications program accesses a relation by specifying a single read request with the name of the relation that is to be read. CDCS or Query Update processes the request and returns a record occurrence from each file in the relation to the user's work area for the file.

CONSTRAINTS

The constraint facility of CDCS is an independent feature that provides a means of protecting the integrity of data in a data base. Use of the facility prevents the possible introduction of inconsistent data into a data base as a consequence of update operations by application programs on records in logically related files, or on items within a single file.

In the schema definition, the data administrator establishes a dependent parent-child relationship between files or between items within a file by specifying a constraint entry. The constraint entry assigns a name to the constraint and specifies the data items involved in the dependent relationship. The files involved in the constraint must each contain a common data item, which is used to define the constraint. A parent record occurrence corresponds to a child record occurrence if both records contain the same value for the common item.

When a COBOL, FORTRAN, or Query Update applications program updating a data base is executed, CDCS enforces the constraints established in the schema. A write, delete, or rewrite request is permitted or rejected by CDCS on the basis of the effect of the proposed operation on the relationship in the applicable constraint.

DATA BASE PROCEDURES

Data base procedures are special subprograms written by the data administrator to perform a variety of supplemental operations not otherwise performed by CDCS. The procedures are called at execution time when specific situations occur during CDCS processing. The conditions under which data base procedures are to be executed are specified in the schema. The order of execution of the procedures and the names of the data base procedures are also indicated in the schema. When the schema is compiled, an alphabetic list of the data base procedures is printed at the end of the source program listing.

Some of the functions that can be performed by data base procedures are: data validation; data conversion not supported by CDCS; calculation of values for actual or virtual data items; compression and decompression of data; additional processing on creation, retrieval, or update of data base records; privacy checking; and special handling of error conditions detected within CDCS. The use of data base procedures to perform these functions provides a well-defined method of tailoring the CDCS system to meet the needs of a particular installation.

INPUT/OUTPUT PROCESSING

Execution-time processing of input/output statements that reference data base files is directed by CDCS to AAM. AAM handles all operations concerning the physical storage and access of data in a data base. All data base files supported by CDCS are conventional extended AAM files.

File Organization

File organization of data base files is specified in the operating system FILE control statement when the schema is compiled. The file organization information is stored in the schema directory. The only file organizations allowed for data base files that are to be accessed through CDCS are extended indexed sequential, extended direct access, and extended actual key.

Records in extended indexed sequential files are stored in ascending order by key. The records can be accessed either randomly by key or sequentially by position. This file organization should be used for files that are to be accessed both randomly and sequentially.

Records for extended direct access files are stored randomly in fixed length blocks. The number of the block to receive a record is determined by a calculation performed by the system on the record key. Records can be accessed randomly by key or sequentially. Extended direct access file organization is used most effectively when rapid random access is required.

Extended actual key files contain records whose key values are assigned by the system. The key value is a number that identifies the block and the position within the block in which the record is stored. Records can be accessed randomly by actual key; records also can be accessed sequentially. Extended actual key file organization is used most effectively for files when the user can keep track of system-assigned keys and when performance and file growth characteristics are of primary concern. It is also used when no unique key exists or when straight sequential access is desired.

Multiple-Index Processing

Multiple-index processing is performed when alternate keys are defined for extended indexed sequential, extended direct access, and extended actual key files. An index is created for each alternate key in a data file when the file is created. The indexes are updated automatically whenever the data file is updated. Records can then be retrieved by the primary key or by an alternate key. For detailed information refer to the Advanced Access Methods reference manual.

DATA BASE RECOVERY

The recovery facilities of CDCS supply the means to deal with a lost, partially destroyed, or invalid data base. Through the use of log files that record information about user interactions with a data base, along with the data base recovery utilities, the data administrator can reconstruct a destroyed data base or restore an invalid data base.

Log Files

Two types of log files are used in data base recovery operations. The journal log file contains a record of each occurrence of an update or write operation on a data base. In addition, a record is maintained on the journal log file of certain user requests and privacy breach attempts. The second type of log file, the quick recovery log file, is used internally by CDCS to record blocks of records before the data base is modified by AAM. Both types of logging are optional.

Both journal log file names and quick recovery log file names are specified as input to the master directory utility. Both are assigned on a per-schema basis. The journal log file, in addition to being input to a recovery run, can be processed by a program for statistical analysis at a later date. The quick recovery log file is processed before recovery or restoration operations take place.

Recovery Utilities

Four data base utilities are provided by CDCS to perform several functions required for restoration or reconstruction of a data base. All utilities are called into execution by the data administrator via control statements.

The DBRCN utility is used to reconstruct a data base when the data base has been destroyed through a disk or software failure. The DBRST utility restores a data base to a previous state when the data base has been erroneously updated. Prior to running both the DBRCN and the DBRST utilities, the DBQRFA utility must be executed to apply the contents of the quick recovery log file to the data base. The fourth utility, DBQRFI, is used to initialize the quick recovery log file for block logging.

SUB-SCHEMA ORGANIZATION 2

A FORTRAN sub-schema describes the portion of a data base to be used by one or more FORTRAN applications programs. Its descriptions link the descriptions found in the schema with the variables and arrays in the FORTRAN program. The sub-schema uses statements similar to FORTRAN specification statements to indicate the data type and dimensions of variables and arrays used in the applications program.

The output from compilation of the sub-schema is a sub-schema directory. When a FORTRAN program containing DML statements is processed, the file containing the sub-schema directory must be made available to the DML preprocessor.

The sub-schema definitions are inserted into the FORTRAN program by the DML preprocessor. Therefore, the variables and arrays defined in the sub-schema must not be specified in type statements in the applications program.

The sub-schema is based on the schema; the schema must be compiled before the sub-schema. The sub-schema can include all or part of the entities defined in the schema. A given schema can have any number of sub-schemas written for FORTRAN or COBOL. In general, the schema description of data can be changed in the sub-schema to meet the needs of the applications program; however, some restrictions have been imposed. These are outlined in the following paragraphs.

Terminology used for some entities differs among the contexts of the schema, the sub-schema, and the FORTRAN program. The most important of these is the term realm for a sub-schema, which is called an area in the context of a schema, and a file in a FORTRAN context.

SUB-SCHEMA STRUCTURE REQUIREMENTS

Table 2-1 shows the order in which groups of FORTRAN/DDL statements must be written in a sub-schema. In group 4, the type statements that apply to the variables and arrays belonging to a record defined by a RECORD statement appear immediately after the RECORD statement. In group 5, the RESTRICT statements that apply to a relation defined by the RELATION statement appear immediately after the RELATION statement. A FORTRAN sub-schema must contain at least one realm, one record, and one type statement.

DATA DESCRIPTION

A record description entry comprises a RECORD statement and the type statements immediately following it. The type statements following each RECORD statement specify the data items that are to be made available from the corresponding schema record. The ordering of type statements is independent of the ordering of items within the schema record.

TABLE 2-1. SUB-SCHEMA STATEMENT ORDERING

Group	Statements
1	SUBSCHEMA
2	ALIAS (optional)
3	REALM
4	RECORD and type
5	RELATION (optional) RESTRICT (optional)
6	END

Correspondence between schema and sub-schema items is based on the item name. Therefore, a data name in a type statement must be one of the following:

- A data name from the schema description
- An alias assigned in an ALIAS statement

Any data name in the schema longer than seven characters, or containing a hyphen, must be renamed in an ALIAS statement, because these data names are not legal variable names in FORTRAN. Once an alias is assigned to a schema name, the alias must be used in all DDL statements. Only elementary item names, from the schema, can be defined in a FORTRAN sub-schema. Each record type corresponds to a record type in the schema and must be within one of the realms specified in REALM statements. Each RECORD statement must be followed by at least one type statement.

All the data items defined in the sub-schema are included in the FORTRAN program by the DML preprocessor. Therefore, all data items named in the sub-schema are either variables or arrays in the FORTRAN applications program.

Character variables and arrays should be grouped together within a record in the sub-schema to minimize the number of common blocks that the DML preprocessor generates for a FORTRAN program. The variables and arrays are declared in common blocks in the same order as they are included in the sub-schema. One common block is generated for each realm. The name for that common block is in the form DBnnnn, where nnnn is the realm ordinal assigned by the DDL compiler. Realm ordinals are assigned incrementally starting with 1 for each realm named in the sub-schema. Character data items cannot share the same common block with noncharacter items; therefore, a new common block is generated each time a type statement is encountered that is not compatible with the previous type statement. The name for each additional common block required for a realm is in the

form Dnnnnxx, where nnnn is the realm ordinal and xx is a 2-letter identifier assigned incrementally from the series AA, AB, ..., ZZ.

VARIABLES

A variable declaration in a type statement associates a symbolic name of the specified type with a single data item. Because every data base data item that is to be referenced in the FORTRAN program must be declared in the sub-schema, implicit typing of variables is overridden. A variable defined in the sub-schema must correspond to a nonrepeating elementary item in the schema.

FORTRAN/DDL permits a special long variable only for FORTRAN 4 applications so that a FORTRAN 4 program can reference a schema item that is a nonrepeating alphanumeric item longer than 10 characters. In the sub-schema, this special long variable is declared as an array. The detailed description of this item is in the subsection Character Data. (For FORTRAN 5 applications, this same capability is provided by declaring a variable type CHARACTER.)

ARRAYS

The size, dimensions, and type of an array are defined in a type statement. Array declarations are identical in form and content to those in a FORTRAN program.

An array in a sub-schema corresponds to a repeating elementary item in the schema; that is, to an item containing an OCCURS clause. A repeating elementary item is called a vector. An array can correspond to either a fixed occurrence repeating item or a variable occurrence repeating item.

SCHEMA/SUB-SCHEMA CORRESPONDENCE

The sub-schema is created to accommodate the needs of a FORTRAN applications program. Some characteristics of the data in the data base are fixed by the schema and cannot be changed by the sub-schema; other characteristics specified in the schema can be different in the sub-schema. The following paragraphs outline the cases in which differences are allowed between the schema and the sub-schema, and the actions taken by the DDL compiler to resolve the differences in each case.

OMISSION OF DATA ITEMS

The sub-schema normally describes only a portion of the data base. Data items that are not required by the FORTRAN program are not included in the sub-schema description. Elementary items, complete records, and entire areas in the schema can be omitted from the sub-schema. When a record or an area is not included in the sub-schema description, all subordinate entries are automatically omitted and cannot be referenced in the sub-schema.

Unlike the COBOL sub-schema, the FORTRAN sub-schema has no mechanism for the description of group items. Therefore, these items must be omitted from the sub-schema. The elementary items making up the schema group item must also be omitted from the sub-schema.

Unlike the COBOL sub-schema, only one record type per realm is permitted in the FORTRAN sub-schema.

The primary key for a realm must be declared in the sub-schema; alternate keys are required only if they are actually used. A concatenated key is not supported in FDBF.

ORDERING OF DATA ITEMS

The order in which data items are specified within a record description in the sub-schema does not have to be the same as the order in the schema, as long as the names of the items match those in the schema or in the ALIAS statement.

DEFINITION OF DATA ITEMS

Data items can differ in size, type, and number of array elements from those in the schema.

Data Size and Type

The size and type of data items in the sub-schema are specified in the type statements. In the schema they are specified in either the TYPE or PICTURE clause. Since the schema and sub-schema statements differ in format, rules have been established for conversion between sub-schema and schema specifications. In some cases, the types specified in the schema and sub-schema match exactly; no conversion is required. In other cases, the types differ, but a meaningful conversion is established by DDL and carried out through mapping at execution time by CDCS. In still other cases, no conversion is possible and an error message is issued by the DDL compiler.

Complex and double precision variables occupy two words of storage each; all other variables occupy one word of storage. Complex and double precision arrays occupy two words of storage for each array element; all other arrays occupy one word of storage for each array element.

Table 2-2 shows the permissible correspondence between types of items in the sub-schema and data class specifications in the schema. (Refer to volume 1 of the DDL reference manual, for complete descriptions of these data classes.) For those cases where conversion is necessary, the table describes the method used.

If the schema specifies a CHECK IS PICTURE clause, the data description in the sub-schema must match the data description in the schema for both size and class. The CHECK IS PICTURE clause in the schema definition inhibits data conversion between the schema and the sub-schema. Table 2-2 indicates the data type required for a sub-schema item to correspond to a schema item defined with the CHECK IS PICTURE clause.

Character Data

In a sub-schema for a FORTRAN 5 program, an item must be declared as type CHARACTER to correspond to schema item of data class 0 (display alphanumeric) or class 1 (display alphabetic.) Because there is no character data type in FORTRAN 4, the combination of schema class 0 or 1 and sub-schema integer allows FORTRAN 4 programs to read and write character data without conversion. Therefore, any variable defined as integer in a FORTRAN 4 sub-schema and class 0 or 1 in the schema is treated as character (not numeric) data. Data assigned

TABLE 2-2. SCHEMA/SUB-SCHEMA MAPPING

Schema Class (type)	Sub-Schema Type: INTEGER FORTRAN 4	INTEGER FORTRAN 5	CHARACTER FORTRAN 5	BOOLEAN FORTRAN 5	LOGICAL FORTRAN 4 & 5	REAL FORTRAN 4 & 5	DOUBLE PRECISION FORTRAN 4 & 5	COMPLEX FORTRAN 4 & 5
0 Display alpha-numeric	No conversion required.	Not permitted.	No conversion required.†	Not permitted.	Not permitted.	Not permitted.	Not permitted.	Not permitted.
1 Display alphabetic	No conversion required. Error if not alphabetic.	Not permitted.	No conversion required. Error if not alphabetic.	Not permitted.	Not permitted.	Not permitted.	Not permitted.	Not permitted.
3 Display integer	Evaluate character string representing integer and convert to binary.	Evaluate character string representing integer and convert to binary.	No conversion required.	Evaluate character string representing integer and convert to binary.	Not permitted.	Evaluate character string representing value and float.	Same as for real; set least significant word to zero.	Not permitted.
4 Display fixed point	Evaluate character string representing integer and convert to binary.	Evaluate character string representing integer and convert to binary.	Not permitted.	Evaluate character string representing integer and convert to binary.	Not permitted.	Evaluate character string representing value and float.	Same as for real; set least significant word to zero.	Not permitted.
10 Coded fixed point	No conversion required.†	No conversion required.†	Not permitted.	No conversion required.	No conversion required.	Float.	Float; set least significant word to zero.	Float; set imaginary part to zero.
13 Coded floating point normalized	Truncate real value.	Truncate real value.	Not permitted.	No conversion required.	Not permitted.	No conversion required.†	Set least significant word to zero.	Set imaginary part to zero.
14 Coded double precision	Drop least significant word and truncate value of most significant word.	Drop least significant word and truncate value of most significant word.	Not permitted.	Drop least significant word.	Not permitted.	Drop least significant word.	No conversion required.†	Drop least significant word; set imaginary part to zero.
15 Coded complex	Drop imaginary part and truncate value of real part.	Drop imaginary part and truncate value of real part.	Not permitted.	Drop imaginary part.	Not permitted.	Drop imaginary part.	Drop imaginary part; set least significant word to zero.	No conversion required.

†Data type heading this column is required for a sub-schema item when the CHECK IS PICTURE clause is specified for the corresponding schema item.

to these variables in the application program should be Hollerith data, left-justified and blank filled.

In addition to providing a correspondence between a sub-schema FORTRAN 4 integer and a schema class 0 or 1 item, FORTRAN/DDL provides a special long variable to provide the capability for a program to reference a nonrepeating, schema item longer than 10 characters. In the schema, the item must be an elementary item and must not be part of a repeating group. In the sub-schema, the item must be declared as an integer array. DDL processing makes the schema item available to the FORTRAN 4 program by having the first word of the array reference characters 1 through 10, the second word of the array reference characters 11 through 20, and so forth for the length of the schema item providing the array is declared having sufficient length. (The maximum length allowed in the schema for character data is 32 767 characters.)

The way to reference the long variable depends on the context. In the DDL RESTRICT statement and as the key in the DML READ statement, the item must be referenced by the array-name with no subscripts. In a FORTRAN 4 statement in a program, the item must be referenced as an array or array element. See the FORTRAN 4 reference manual for the rules on array references.

Figure 2-1 shows a 30-character display alphabetic item in a schema and the corresponding item declared in a sub-schema. The sub-schema item MYKEY is used as if it were a 30-character variable when used in DDL and DML statements. In a FORTRAN 4 statement, the item must be used as a 3-word array.

```
Schema

    02 MYKEY PIC "X(30)".

Sub-Schema

    INTEGER MYKEY(3)
```

Figure 2-1. Long Variable for FORTRAN 4 Applications

Array Declaration

Arrays are declared in the sub-schema by type statements in the same format as those in a FORTRAN program. The number of elements in an array is the product of all its dimensions.

An array is defined in the schema as a vector, or a repeating elementary item. Elementary data items can be repeated either a fixed or varying number of times. Arrays in the sub-schema correspond with these repeating data items.

A vector in the schema can correspond to a sub-schema array with up to three dimensions for a FORTRAN 4 program or up to seven dimensions for a FORTRAN 5 program. The number of elements in an array must be less than or equal to the number of occurrences of the schema vector. A repeating elementary item is called a fixed occurrence repeating item if the OCCURS clause which describes it in the schema specifies the precise number of occurrences of the item.

Example:

Figure 2-2 shows the schema and sub-schema declarations for two arrays. Both are fixed occurrence elementary items in the schema.

The number of elements of the array declared in the sub-schema can fall short of (but cannot exceed) the number of occurrences of the item declared in the schema. When this happens, the initial elements of the array are matched to the initial occurrences of the item.

```
Schema

    RESULTS TYPE FLOAT

      OCCURS 40 TIMES.

    TESTNUM PICTURE 9(10)

      OCCURS 20 TIMES.

Sub-Schema

    REAL RESULTS(40)

    INTEGER TESTNUM(20)
```

Figure 2-2. Fixed Occurrence Elementary Items

For example, in figure 2-3, the arrays BAYNAME and QUANT have the same number of elements as their counterparts in the schema, even though they are defined as 2-dimensional arrays in the sub-schema. The array TOP10 has fewer elements than its counterpart in the schema.

```
Schema

    BAYNAME PICTURE "A(10)" OCCURS 16 TIMES

    QUANT PICTURE "999"

      OCCURS 2000 TIMES.

  TOP10 TYPE FIXED OCCURS 100 TIMES.

Sub-Schema

      INTEGER BAYNAME(2,8), QUANT(200,10),
    X   TOP10(10)
```

Figure 2-3. Schema/Sub-Schema Differences in Array Size and Dimension

A repeating elementary item in the schema is called a variable occurrence repeating item if a data name is used in the OCCURS clause. In the schema, the occurrences of a variable occurrence repeating item (which correspond to the elements of the array in the sub-schema) are indexed by the elementary item referenced in the OCCURS clause. In a FORTRAN sub-schema, this item must be declared whenever the repeating item is declared. In the DML statements in the applications program, the variable is used to specify the number of occurrences of a repeating item when a record is written. When a record is

read, CDCS sets the variable to the number of occurrences actually in the record.

If the sub-schema defines an item that is used in the schema to index the occurrences of a repeating item, the sub-schema must also define the elementary repeating item.

A sub-schema array that corresponds to a variable occurrence repeating item must be defined in the sub-schema to have the maximum number of elements possible (the upper limit of the CHECK IS VALUE clause).

Example:

In figure 2-4, the number of occurrences of PERCENT and SYMBOL can vary from 1 to 10, depending upon the value of ECOUNT. PERCENT and SYMBOL are variable occurrence repeating elementary items in the schema. PERCENT and SYMBOL are declared as arrays in the sub-schema with dimensions equal to the maximum occurs value. ECOUNT must also be declared in the sub-schema.

Schema

```
ECOUNT TYPE FIXED
CHECK VALUE 1 THRU 10.
SYMBOL PIC "A(10)", OCCURS ECOUNT TIMES.
FULL-NAME PIC "X(15)" OCCURS ECOUNT TIMES.
PERCENT TYPE FLOAT, OCCURS ECOUNT TIMES.
```

Sub-Schema

```
INTEGER ECOUNT, SYMBOL(10)
REAL PERCENT(10)
```

Figure 2-4. Variable Arrays in Schema and Sub-Schema

The FORTRAN/DDL sub-schema source program comprises a series of statements that describe a portion of a data base. The rules for coding FORTRAN/DDL statements are similar to those for the version of FORTRAN specified in the FORTRAN/DDL (DDLF) control statement. The statements themselves are described in section 4; this section describes the format of the statements. Appendix J summarizes required differences in coding a sub-schema for FORTRAN 4 or FORTRAN 5 applications programs.

LANGUAGE ELEMENTS

FORTRAN/DDL statements consist of keywords, user-defined names, constants, and operators. The operators are fully explained with the statements in section 4; the remainder of the elements are described here.

KEYWORDS

FORTRAN/DDL keywords identify statements and options within statements. Each statement begins with a specific keyword, and other keywords are used within statements. When a keyword is used, it must be specified exactly as shown in section 4. Keywords are shown in uppercase in this manual. For a complete list of keywords see appendix D.

USER-DEFINED NAMES

User-defined names identify the schema, sub-schema, realm, records, data items, and relations. They are indicated in the formats by lowercase words.

All names are taken from the schema except for schema entities that are renamed in the sub-schema by the ALIAS statement. The rules for forming names differ between data item names and all other names:

Data item names — Since these are used in the FORTRAN program as variable and array names, they must correspond to the FORTRAN rules. They must be from one to seven characters long, contain only letters and digits, and begin with a letter.

Other names — These are used only by DMS-170 and consequently follow the more lenient rules of the schema. They must contain from 1 to 30 letters, digits, or hyphens. The first character must be a letter, and hyphens cannot be used at the beginning or end or adjacent to each other.

Since each data item appears in a type statement, default types based on the first letter of the item name are not applicable in the sub-schema. Non-data base items (used in the RESTRICT statement) are not given types by DDL, and consequently must appear in type statements in the FORTRAN program if other than default types are desired for them.

A number of variables are generated in the FORTRAN program by the DML preprocessor; definition or declaration of these variables in a FORTRAN program can lead to invalid results. A complete list of these variables can be found in appendix G.

CONSTANTS

With a few restrictions, constants appearing in DDL statements follow the rules for FORTRAN constants. These rules are defined in the FORTRAN reference manual corresponding to the version of FORTRAN specified on the control statement used to compile the sub-schema. The constants allowed are restricted to integer or real constants without exponents, and to strings delimited by quotation marks or (for FORTRAN 5 programs only) to strings delimited by apostrophes. The following forms are not allowed: double precision, complex, logical, octal, and hexadecimal constants; Hollerith constants in H, L, or R format; and real constants with exponents. The following forms can be used:

- Integer constants consist of an optional sign (+ or -) followed by 1 to 18 digits.

  Examples:

  ```
  0
  -12345
  +2000
  000000000000000004
  ```

- Real constants consist of an optional sign (+ or -) followed by a string of digits containing exactly one decimal point. The maximum number of digits is 15. The decimal point can be anywhere within the string; that is, the number can consist of a fractional part, a whole number part, or both.

  Examples:

  ```
  0.
  -3.22
  +4000.
  -.5
  .0
  1.414
  ```

- Rules for nonnumeric constants depend on the version of FORTRAN. In a sub-schema for a FORTRAN 4 program, the constant must be a Hollerith constant (a string of from 1 to 255 characters delimited by quotation marks). In a sub-schema for a FORTRAN 5 program, the constant can be a character constant (a string of from 1 to 255 characters delimited by apostrophes) or a Hollerith constant (a string of from 1 to 10 characters delimited by quotation marks). If a delimiting character is to be used in the string, the character must be specified twice for each

occurrence. For example, "A""B" would yield the constant A"B; 'C'D' would yield the constant C'D.

Examples:

"MY KEY WORKS"	FORTRAN 4 constant	Hollerith
"MAX10CHARS"	FORTRAN 5 constant	Hollerith
'CATS & DOGS'	FORTRAN 5 constant	character

FORTRAN/DDL STATEMENT FORMAT

FORTRAN/DDL statements occupy from 1 to 100 character positions of a source line. A source line is a punch card or card image, and a character position corresponds to a card column. Table 3-1 shows the usage of the character positions within the source line.

TABLE 3-1. COLUMN USAGE IN FORTRAN/DDL STATEMENTS

Column	Usage
1	C or * indicates a comment line.
1-5	Optional statement label containing one through five digits.
6	Character other than blank or zero indicates a continuation line; does not apply to comment lines.
7-72	Text of FORTRAN/DDL statement.
73-100	Identification field, listed but not otherwise processed by DDL.

No statement can begin on a line that includes any part of the previous statement; the $ statement separator is not used.

The following paragraphs describe other aspects of the mechanics of coding FORTRAN/DDL statements.

CHARACTER SET

The FORTRAN/DDL character set is a subset of the FORTRAN character set. It consists of the letters A through Z, the digits 0 through 9, and the following special characters:

	Blank
=	Equal sign
,	Comma
(	Left parenthesis
)	Right parenthesis
.	Decimal point
" or ≠	Quotation mark
' or ↑	Apostrophe
+	Plus sign
-	Minus sign

In addition, any character (appendix A) can be used in character constants and in comments.

BLANKS

Unlike FORTRAN statements, in which blanks are ignored (except in character constants), DDL statements forbid or require blanks in specific cases. The rules are as follows:

- Blanks are significant in character constants.
- Keywords, user-defined names, relational operators, and constants cannot be interrupted by blanks.

  Permitted:

  ```
  RESTRICT RECA(ITEM1.EQ.5.0)
  ```

  Not permitted:

  ```
  REST RICT REC A(ITEM 1. EQ.5 .0)
  ```

- Keywords must be separated from adjacent names by at least one blank. Blanks are not required when a keyword is set off by other special characters.

  Permitted:

  ```
  DOUBLE PRECISION D1, D2(10,20),D3
  ```

  Not permitted:

  ```
  DOUBLEPRECISIOND1,D2(10,20),D3
  ```

  This example clarifies that DOUBLE and PRECISION are considered separate keywords, even though they are used together.

CONTINUATION

Statements can be continued for more than one line. A character other than a blank or zero in column 6 indicates that the line is a continuation line. Columns 1 through 5 of a continuation line must contain blanks. A line with a C or * in column 1 and any character in column 6 is a comment line, not a continuation line. The maximum number of continuation lines in one statement is 19.

The END statement cannot be continued.

STATEMENT LABELS

As in FORTRAN, any statement can contain a label in columns 1 through 5. A label is a one to five digit integer. Labels do not affect DDL processing, but can be included for documentary reasons. No diagnostic is issued if the same label is used more than once.

Labels on type statements become FORTRAN statement labels when the type statements are copied into the FORTRAN source program. If any label on the type statements is a duplicate of a label on another FORTRAN statement, a diagnostic is issued by the FORTRAN compiler.

COMMENT LINES

A line with a C or * in column 1 is a comment line. Comment lines are copied to the DDL output listing but are not otherwise processed. Comments can contain any character in the character set. Comments do not interrupt statement continuation.

BLANK LINES

Blank lines can be used freely between statements to produce blank lines on the source listing. Within statements, the interpretation of a blank line is consistent with the version of FORTRAN specified on the DDLF control statement. With FORTRAN 4, a blank line interrupts statement continuation, and the next line is interpreted as an initial line even if it has the form of a continuation line. With FORTRAN 5, a blank line is considered a comment line and does not break the continuation sequence.

This section describes the individual FORTRAN/DDL statements. The rules for coding the statements are in section 3, and table 2-1 shows the order in which statements must occur in a sub-schema. Appendix E contains a syntax summary of FORTRAN/DDL statements for FORTRAN 5 applications; appendix F contains the syntax summary for FORTRAN 4 applications.

SUBSCHEMA

The SUBSCHEMA statement (figure 4-1) must be the first statement in every FORTRAN/DDL program. It names the schema and the sub-schema. Only one SUBSCHEMA statement can appear.

```
SUBSCHEMA sub-schema-name, SCHEMA = schema-name
```

Figure 4-1. SUBSCHEMA Statement Format

The sub-schema named in the SUBSCHEMA statement is used whenever the sub-schema is referenced after it has been compiled and stored in the sub-schema library. The name must be unique among sub-schemas associated with the designated schema.

The schema-name identifies the schema to which the sub-schema belongs. The file containing the compiled schema must be attached when the sub-schema is compiled.

ALIAS

The ALIAS statement (figure 4-2) is used to change the name of an entity from the name used in the schema to the name used in the sub-schema. The name to be changed can be a realm, a record, or a data item.

The ALIAS statement is optional; if used it must immediately follow the SUBSCHEMA statement.

Assigning an alias in the sub-schema does not change the name in the schema; it is a substitute name that is used only in the sub-schema and in the FORTRAN programs referencing the sub-schema. The old name must appear in the schema; once an alias has been assigned, the new name, rather than the old name, must be used exclusively in the sub-schema and the FORTRAN program.

The name type specifier (REALM, RECORD, or ITEM) applies to all names in that ALIAS statement. Any number of ALIAS statements can be used, but no old name or new name can appear more than once.

The ALIAS statement must be used to change item names that are longer than seven characters, or that contain hyphens. Such names are valid in the schema but are not legal symbolic names in FORTRAN.

Example:

```
Schema:
01  LAST-IN-FIRST-OUT  PICTURE "X(10)"

Sub-Schema:
ALIAS (ITEM) LIFO = LAST-IN-FIRST-OUT
```

The form old-item-name.old-record-name must be used to provide unique names within the FORTRAN program when the same name appears in two different records in the schema.

Example:

```
Schema:
RECORD NAME IS TOTAL-REC
01  ACCOUNT         PICTURE "99999"
.
.
.
RECORD NAME IS PARTIAL-REC
01  ACCOUNT         PICTURE "99999"

Sub-Schema:
ALIAS (ITEM) ACNTTOT = ACCOUNT . TOTAL-REC,
1     ACNTPAR = ACCOUNT . PARTIAL-REC
```

REALM

The REALM statement (figure 4-3) identifies the specific schema areas that are used in the sub-schema and in the FORTRAN applications programs referencing the sub-schema. Realm and area are sub-schema and schema

```
ALIAS { (REALM)  }  new-name-1 = old-name-1  [, new-name-2 = old-name-2]  . . .
      { (RECORD) }
      { (ITEM)   }

Formats for old-name (with ITEM option only):

    old-item-name

    old-item-name.old-record-name
```

Figure 4-2. ALIAS Statement Format

terms, respectively, for what is known to the operating system, and to FORTRAN, as a file.

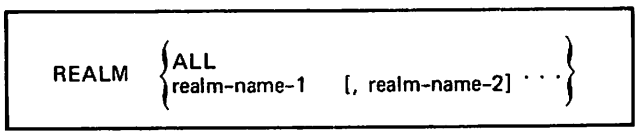


Figure 4-3. REALM Statement Format

At least one REALM statement must appear in the sub-schema. If ALIAS statements are used, the REALM statements must immediately follow the last ALIAS statement; otherwise, they must immediately follow the SUBSCHEMA statement.

Any number of REALM statements can be included, but no realm can be named more than once. All realms named must appear in the schema. The ALL option specifies that all the realms defined in the schema are to be available to the sub-schema. Unless the ALL option is used, only those realms named in a REALM statement are available to the sub-schema and to the FORTRAN programs referencing the sub-schema.

RECORD DEFINITION

A record definition comprises a RECORD statement followed by one or more type statements. The record definitions must immediately follow the REALM statements, and each sub-schema must contain at least one record definition. Each record definition corresponds to a record type defined in the schema.

The type statements following a RECORD statement specify those data items occurring within the record that are to be used in the sub-schema. These data items can be used as variables or arrays in the FORTRAN program. The type statements also define the FORTRAN data types of the items, as well as the number of dimensions and size of arrays. Data items not included in the sub-schema cannot be accessed by the FORTRAN program. The items defined for a record in the sub-schema can differ in order, number, and (in some cases) type from the equivalent definitions in the schema. Refer to section 2 for a discussion of schema/sub-schema compatibility.

RECORD STATEMENT

The RECORD statement (figure 4-4) identifies a schema record that is to be used in the sub-schema. Any number of RECORD statements, each followed by a group of type statements, can appear in the sub-schema, but no record name can be used more than once. Each record must appear in the schema. The schema links record types with areas; the area associated with a record in the sub-schema must appear in a REALM statement. Only one record type per realm in the sub-schema is permitted.

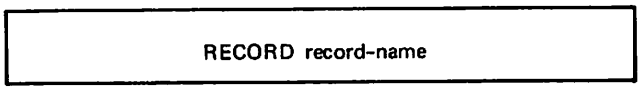


Figure 4-4. RECORD Statement Format

TYPE STATEMENTS

Type statements identify and describe the data items that are to be made available to the sub-schema and the applications program. FORTRAN/DDL recognizes the same type statements as the version of FORTRAN compiler specified in the DDLF control statement. The rules governing type statements and the amount of storage allocated to each variable are described in the FORTRAN Extended 4 reference manual or in the FORTRAN 5 reference manual. The formats of the type statements are shown in figure 4-5. FORTRAN 4 allows the following type statements: REAL, INTEGER, LOGICAL, COMPLEX, AND DOUBLE. FORTRAN 5 allows the following type statements: CHARACTER, BOOLEAN, REAL, INTEGER, LOGICAL, COMPLEX, and DOUBLE PRECISION. Unlike FORTRAN 4 syntax, FORTRAN 5 syntax requires that both the keywords DOUBLE and PRECISION be specified.

Each data item referenced in the sub-schema must appear in a type statement. If it is an array, its dimensions are given in the same type statement. No item can appear in more than one type statement. If an item occurs in more than one record type in the schema, it must be renamed by the ALIAS statement in the sub-schema. Since each item appears in a type statement, implicit typing according to the first letter of the item name is not recognized in FORTRAN/DDL.

Each item appearing in a type statement must be defined in the schema. (It could be defined under a different name if it appears in the ALIAS statement in the sub-schema.) Each item defined as an array must be defined in the schema with the OCCURS clause. The number of elements in the sub-schema array must not exceed the number of occurrences in the schema vector.

At least one type statement must be associated with each RECORD statement, even if no items are used by the FORTRAN program (for example, a program that counts the number of records in a file). The variable that is the primary key for the realm must be included in the sub-schema; alternate keys have to be included only if they are actually used in the program.

Example:

```
RECORD FIRST-REC
INTEGER A, B, C(12)
REAL X
RECORD SECOND-REC
INTEGER D, E
COMPLEX H(2,3,4)
CHARACTER*20 J
```

The variables A, B, and X and the array C are to be made available from the record type FIRST-REC; the variables D, E, and J, and the array H are to be made available from the record type SECOND-REC. The character variable J is allowed only for FORTRAN 5 applications. A similar data item for a FORTRAN 4 application would be a special long variable declared as INTEGER J(2).

For examples of correspondence between type statements in the sub-schema and item descriptions in the schema, see section 2.

RELATION DEFINITION

A relation definition comprises a RELATION statement optionally followed by any number of RESTRICT statements. Relation definitions are optional; if used, they must immediately follow the last group of RECORD and type statements.

FORTRAN 5 Format

```
{ CHARACTER [*default-length [,]] item-name-1 [*len-1] [,item-name-2 [*len-2]] . . .
{ {BOOLEAN  }                                                                    }
{ {REAL     }                                                                    }
{ {INTEGER  }  item-name-1 [,item-name-2] . . .                                  }
{ {LOGICAL  }                                                                    }
{ {COMPLEX  }                                                                    }
{ {DOUBLE PRECISION}                                                             }
```

In a CHARACTER type statement, default-length and len must be positive integer constants. If neither default-length nor len is specified, 1 is assumed.

In each type statement, the formats for item-name are:

item-name

item-name (d1 [,d2] . . .)

where d specifies the bounds of an array dimension. From one through seven bounds can be specified for an array.

The formats for d are:

upper-bound

lower-bound: upper-bound

where the bound specification must be an integer constant: positive, zero, or negative. The upper-bound must be greater than or equal to the lower-bound. (When the lower-bound is not specified, 1 is assumed.)

FORTRAN 4 Format

```
{REAL            }
{INTEGER         }
{LOGICAL         }  item-name-1 [,item-name-2] . . .
{COMPLEX         }
{DOUBLE [PRECISION]}
```

The formats for item-name are:

item-name

item-name (d1 [,d2 [, d3]])

where d is a dimension. Each dimension must be a positive integer constant. (The dimension specifies the upper-bound; 1 is always assumed for the lower-bound.)

Figure 4-5. Type Statement Formats

The RELATION statements specify the schema relations to be made available to the FORTRAN program. RESTRICT statements restrict records accessed by a relation to include only those which satisfy a logical expression. The RESTRICT statements immediately following a RELATION statement describe restrictions on that relation. Relations are explained in section 1.

```
RELATION relation-name
```

Figure 4-6. RELATION Statement Format

RELATION STATEMENT

The RELATION statement (figure 4-6) specifies that a relation defined in the schema is to be available to the FORTRAN program. Any number of RELATION statements can appear, but no relation can be named more than once.

The relation-name must be the same name specified in the RELATION NAME clause in the schema. The schema specifies which realms and record types apply to a relation. When a relation is declared in the sub-schema, the associated realms and record types must also be declared in the sub-schema.

RESTRICT STATEMENT

The RESTRICT statement (figure 4-7) limits a relation by specifying criteria that must be satisfied by a record occurrence on a read before it is made available to the FORTRAN program. RESTRICT statements are optional; any number can appear with a RELATION statement. However, only one RESTRICT statement can be included for a given record. All records referenced must be defined in the schema and must be in a realm that is joined by the relation named in the preceding RELATION statement.

The entities to be compared in the logical expression of a RESTRICT statement are data base items, non-data base items, and constants. A maximum of 1024 data base items, non-data base items, constants, and operators can appear in the restrictions on any one relation.

- A data base item must be defined in the sub-schema as part of the record. It must not be described in the schema through VIRTUAL RESULT or DECODING data base procedures. It must be a variable or an array element with a constant subscript. When two data base items are compared, they must be the same size and must be compatible as determined from the schema data class of each item. Items of schema data classes 0 through 4 (items stored as display code) are compatible. An item of any other schema data class is compatible only with an item of the same schema data class.

- Non-data base items are simple variables that appear in the FORTRAN program but are not defined in the sub-schema. Use of a non-data base item in a RESTRICT statement allows dynamic qualification of a relation. Non-data base items must agree in size and type with the sub-schema representation of the data base items to which they are being compared. Results will be unpredictable if the size and type are not identical.

- Constants are real, integer, Hollerith, or character constants. (Character constants are allowed only in FORTRAN 5.) The forms allowed for constants are shown in section 3. A constant must be compatible with the schema representation of the data base item to which it is being compared.

When a data base item is compared to another data base item or to a non-data base item, both items must have the same size and type. Complex and logical items are not allowed. Real, integer, and double precision items are compared by magnitude. If a data base item is described as an integer in the sub-schema and as a display item in the schema (schema classes 0 through 4), it is assumed to contain character data and can only be compared to a data base item similarly described, to a compatible non-data base item, or to a Hollerith constant. A real or integer constant can be compared to a data base item of any numeric type (real, integer, or double precision); appropriate conversion is performed before the magnitudes are compared.

Logical expressions are interpreted the same as in FORTRAN.

Example:

Valid logical expressions:

```
ITEM1 .GT. ITEM2
X .GE. 4.23 .AND.I .LT. 4
(FIRST .EQ. LAST) .AND. (SECOND .GT. THIRD)
I .OR. .NOT. FOURTH .GT. 0
```

Invalid logical expressions:

```
.NOT. ((I .GT. 2) .OR. (J .GT. 3))
    Two levels of nesting are not permitted.
LOGICAL L
A .GT. B .OR. L
    Logical variables are not allowed.
COMPLEX C,D
C .GT. D
    Complex variables are not allowed.
```

END

The END statement (figure 4-8) must be the last statement in every FORTRAN/DDL program. It cannot be continued. No statements or comment lines can appear after the END statement.

```
END
```

Figure 4-8. END Statement Format

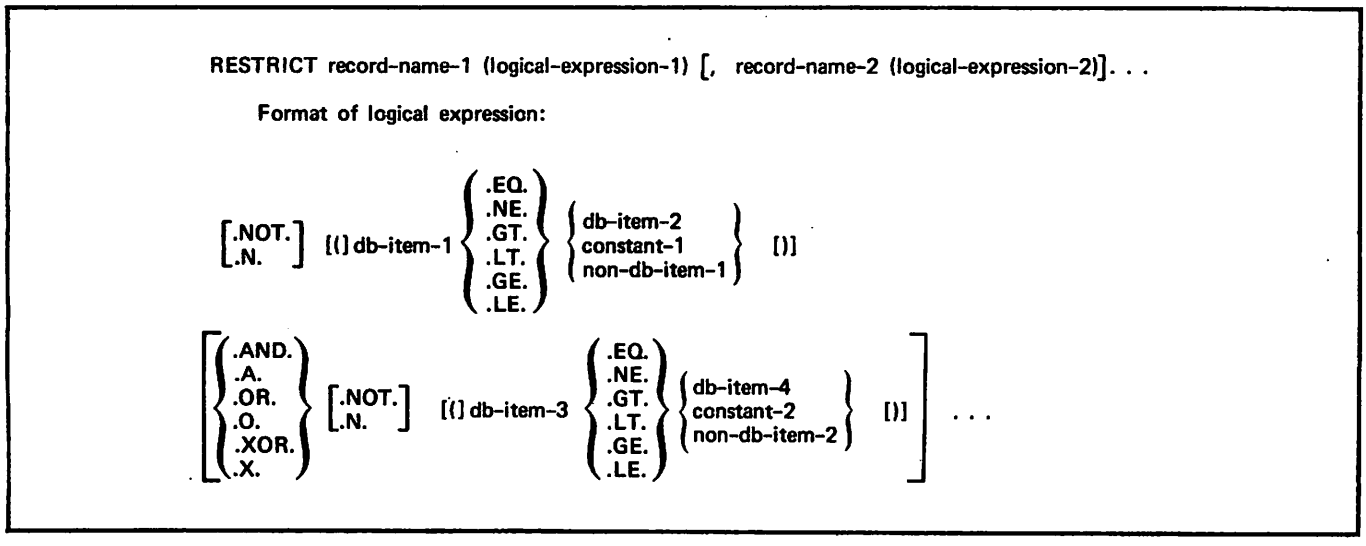


Figure 4-7. RESTRICT Statement Format

SUB-SCHEMA COMPILATION 5

The FORTRAN sub-schema is coded on a standard 80-column coding sheet according to the specifications in this manual. Coding information is punched on 80-column cards or entered through a terminal. The resulting source program and various control statements are input to the FORTRAN/DDL compiler to compile the sub-schema and to store it in the sub-schema library. The control statements provide information for the operating system and for the FORTRAN/DDL compiler. A set of control statements, beginning with a job statement and ending with a 7/8/9 card or its equivalent, precedes the FORTRAN/DDL source program.

SUB-SCHEMA LIBRARY

One or more compiled sub-schemas are stored in a permanent file called the sub-schema library. The library is created when the first sub-schema is stored in it. Subsequent sub-schemas are added to the library or replace existing sub-schemas in the library. A sub-schema can be deleted from the library through the use of the purge parameter on the DDLF control statement.

Data security can be maintained by creating more than one sub-schema library in order to control the availability of the sub-schemas to the FORTRAN programs. Sub-schemas providing access to data that is restricted to specific applications can be stored in one sub-schema library while sub-schemas for general use can be stored in a different library. Each library is identified by a unique permanent file name. When the applications program is compiled, a sub-schema library must be attached by specifying the permanent file name of the library. Sub-schemas stored in other libraries are not available to the program.

DDLF CONTROL STATEMENT

The DDLF control statement (figure 5-1) must be included in the set of control statements preceding the FORTRAN/DDL source program. It provides the FORTRAN/DDL compiler with information related to a specific sub-schema. The DDLF control statement is used to add, replace, or delete a sub-schema in the sub-schema library, or to create a compacted sub-schema library.

FORTRAN 4 and FORTRAN 5 applications programs can access areas in a data base if the sub-schema referenced by the program has been compiled for the appropriate version of FORTRAN. The DDLF control statement parameters F4 and F5 differentiate between the two versions.

DDLF($\begin{matrix}F4\\F5\end{matrix}$, SB=lfn,$p_1$,$p_2$,$p_3$,$p_4$,$p_5$,$p_6$,$p_7$,$p_8$,$p_9$)

DDLF		Identifies the FORTRAN/DDL control statement.
F4		An optional parameter that specifies language version. This parameter specifies a sub-schema for use with a FORTRAN 4 applications program. If F4 or F5 is not specified, F4 is assumed; however, this default can be changed at installation time.
F5		An optional parameter that specifies language version. This parameter specifies a sub-schema for use with a FORTRAN 5 applications program.
SB=lfn		Specifies a logical file name (lfn) of the sub-schema library; default is SBLFN.
p_1	I=lfn	An optional parameter that specifies a nondefault input file; lfn is the logical file name of the source input file; default is INPUT.
p_2	L=lfn	An optional parameter that specifies a nondefault output file; lfn is the logical file name of the source listing file; default is OUTPUT. If L=0 is specified, only diagnostics and associated statements are listed.
p_3	R	An optional parameter that replaces the existing sub-schema in the sub-schema library with the sub-schema compiled from the source program in the input deck. Replacement takes place only if no compilation errors other than informative diagnostics are encountered.
p_4	N	An optional parameter that compiles the sub-schema but does not add it to the sub-schema library.
p_5	P	An optional parameter that purges the specified sub-schemas from the sub-schema library; no compilation takes place.

Figure 5-1. FORTRAN/DDL Control Statement Format (Sheet 1 of 2)

p_6	SC=lfn	An optional parameter that specifies the logical file name of the schema; this parameter overrides the schema name specified in the SUBSCHEMA statement of the source program; default is the first seven characters of the name specified in the program.
p_7	A	An optional parameter that produces a list of sub-schemas and their corresponding schemas, together with their creation dates, from the library indicated by the SB parameter. No compilation takes place.
p_8	LO LO=op_1 LO=op_1/op_2	An optional parameter that selects the listing produced by the DDLF compiler. op_n must be one of the following: S source listing O object listing of data mapping modules and source listing If LO is not specified, or if LO only is specified, S only is assumed. Source and object listings are written to the file specified by the L parameter. If L=0 is specified, no listing is produced, except for error messages, regardless of LO specifications.
p9	NL=lfn	An optional parameter that invokes the sub-schema library compaction facility to create a compressed sub-schema library from an existing sub-schema library. Sub-schemas that have not been purged or replaced are copied to lfn and a revised index table is created. The default lfn is NEWLIB. No compilation takes place.

Figure 5-1. FORTRAN/DDL Control Statement Format (Sheet 2 of 2)

Only one library manipulation function can be performed in an execution of FORTRAN/DDL. If more than one function is specified, a control statement error is issued. If a library manipulation function is attempted on a nonempty file that does not contain a library, a diagnostic is issued and the job is aborted.

MULTIPLE SUB-SCHEMA COMPILATION

Several sub-schemas can be compiled with a single DDLF control statement. The sub-schemas must be combined into a single input file with no intervening end-of-records. Each sub-schema is compiled in turn until an end-of-record (end-of-information) is encountered. The parameters specified on the DDLF control statement apply to all sub-schemas in the input file.

NOS/BE CONTROL STATEMENTS

In addition to the DDLF control statement, standard NOS/BE control statements are included in the set of control statements preceding the FORTRAN/DDL source statements. Refer to the NOS/BE reference manual for complete descriptions of the control statements. Two control statements that must be used in specific instances are described in the following paragraphs.

REQUEST CONTROL STATEMENT

The REQUEST control statement (figure 5-2) is included in the set of control statements for the first sub-schema to be stored in the sub-schema library. This statement specifies the logical file name of the sub-schema library.

REQUEST(lfn,*PF)

lfn	Specifies the logical file name of the sub-schema library.

Figure 5-2. REQUEST Control Statement Format

CATALOG CONTROL STATEMENT

The CATALOG control statement (figure 5-3) is used to catalog the sub-schema library on a permanent file. This statement must be included in the set of control statements for the first sub-schema to be stored in the library. If more than one sub-schema is to be stored in the library, the extend (EX) and modify (MD) parameters must be specified.

CATALOG(lfn,pfn,ID=owner,EX=pw,MD=pw,CN=pw)

lfn	Specifies the logical file name of the sub-schema library. If lfn is omitted, the first seven characters of the pfn are used for the logical file name.
pfn	Specifies the permanent file name of the sub-schema library. If pfn is omitted, the lfn is used for the permanent file name.
ID=owner	Identifies the owner of the sub-schema library file being created.
EX=pw	Specifies the password required to add a sub-schema to the sub-schema library.
MD=pw	Specifies the password required to make a change to the sub-schema library.
CN=pw	Specifies the password required to purge the sub-schema library.

Figure 5-3. CATALOG Control Statement Format

NOS CONTROL STATEMENTS

The set of control statements preceding the FORTRAN/DDL source statements include the DDLF control statement and standard NOS control statements. Refer to the NOS reference manual for complete descriptions of the control statements. The DEFINE control statement (figure 5-4) must be included in specific instances.

DEFINE(lfn=pfn/PW=passwrd,CT=ct,M=m)

lfn=pfn	Specifies the logical and permanent file name of the sub-schema library. If pfn is omitted, lfn is the permanent file name.
PW=passwrd	Specifies the password required to add a sub-schema to the sub-schema library.
CT=ct	Specifies the method of access. Available entries include P (private), S(semiprivate), and PU(public).
M=m	Specifies the file or user permission mode. Available entries include W (write), M(modify), A(append), and R(read).

Figure 5-4. DEFINE Control Statement Format

SAMPLE DECK STRUCTURES

The following paragraphs describe the deck structures and control statements required to compile, store, and delete FORTRAN sub-schemas. Control statements for both the NOS/BE and NOS operating systems are described.

COMPILING A SUB-SCHEMA

The source program for a FORTRAN sub-schema can be compiled without storing it in the sub-schema library. This is accomplished with a deck structure similar to the one shown in figure 5-5. The DDLF control statement must be included in the control statements and the compile parameter (N) must be specified. The language version parameter (F4 or F5) should be specified, or a default value is assumed. The schema file must also be attached.

CREATING THE SUB-SCHEMA LIBRARY

The sub-schema library is created when the first sub-schema is stored in it. Figure 5-6 illustrates a deck structure used to compile a sub-schema and create a sub-schema library.

In this example, the schema file SCHPAY is attached. The REQUEST/CATALOG and DEFINE control statements specify the logical file name of the sub-schema library and assign it to a permanent file device. If only one sub-schema is to be stored in the library, the sub-schema name can be used for the library file name. The DDLF control statement names the sub-schema and schema files and specifies the language version.

ADDING TO THE SUB-SCHEMA LIBRARY

Once the sub-schema library has been created, new sub-schemas can be added to the library. The deck structure illustrated in figure 5-7 adds a sub-schema to the sub-schema library created by the example in figure 5-6.

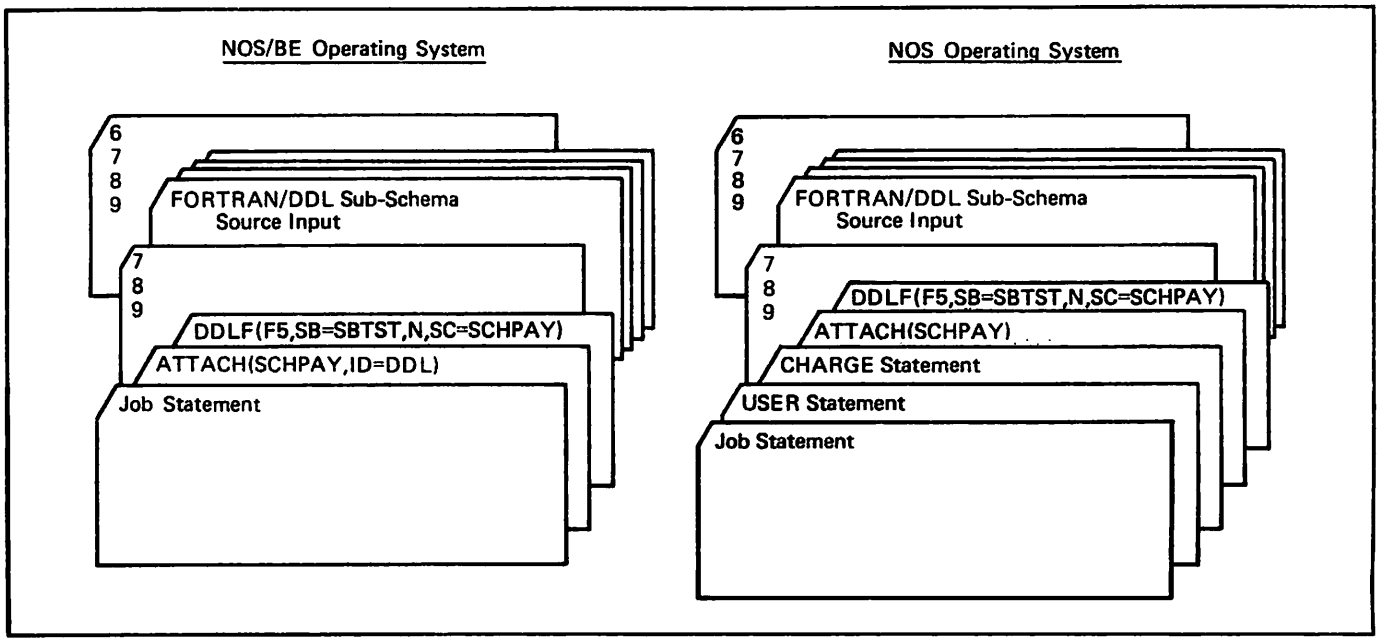


Figure 5-5. Compiling a Sub-Schema

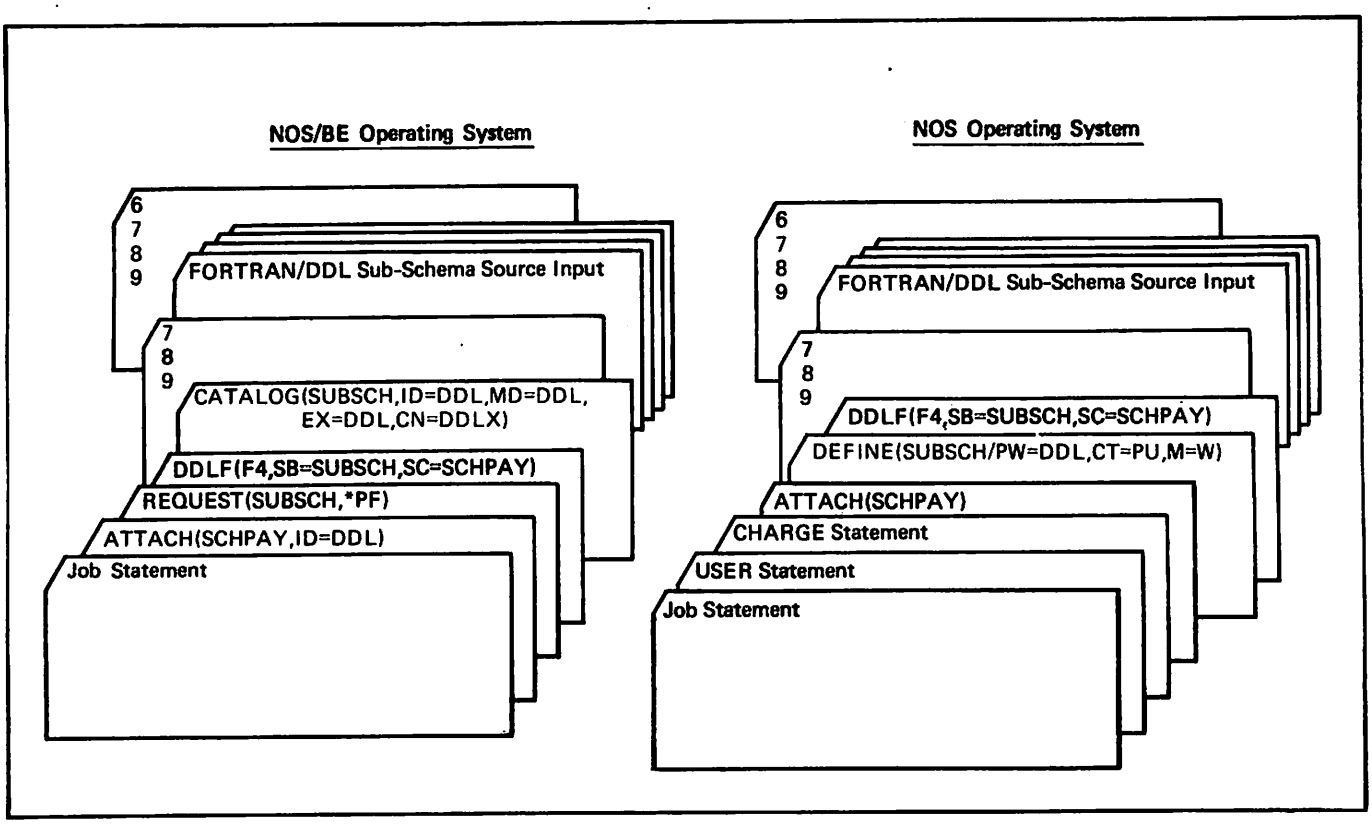


Figure 5-6. Creating a Sub-Schema Library

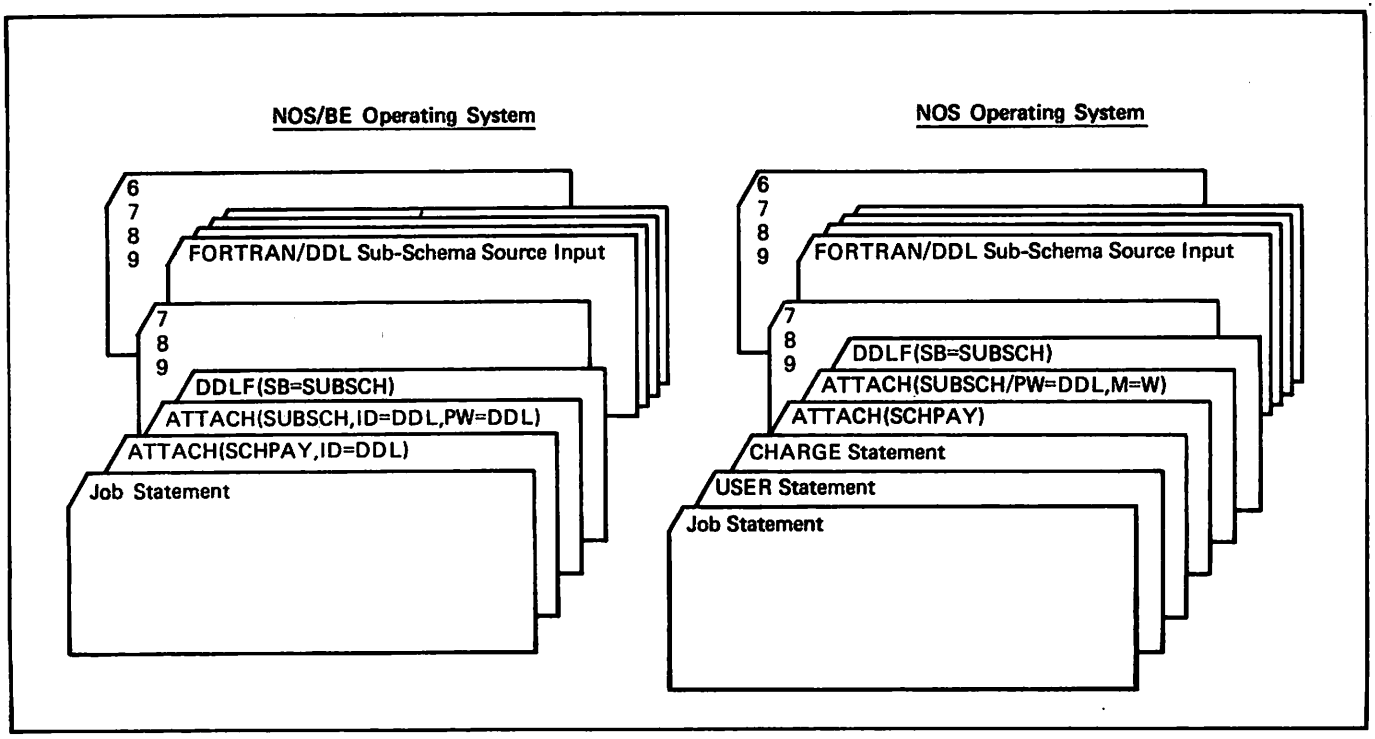


Figure 5-7. Adding a Sub-Schema to the Library

Two ATTACH control statements are required in this example. The schema file SCHPAY and the sub-schema library file SUBSCH are attached. The DDLF control statement requires only the SB parameter if the schema name and file name are the same and the appropriate language version is the default value.

Each sub-schema stored in the library must have a unique name. If the sub-schema being added to the library has the same name as a sub-schema already stored in the library, a diagnostic is issued and the job is aborted.

REPLACING A SUB-SCHEMA

A new sub-schema can replace one stored in the sub-schema library. Figure 5-8 illustrates a deck structure for replacing a sub-schema in the library.

The sub-schema library file is attached with the applicable password specified. The schema file, SCHPAY, must also be attached. In addition to the SB parameter in the DDLF control statement, the replacement parameter (R) must be specified. The DDLF control statement also specifies language version; in this figure F5 is specified. If the sub-schema to be replaced cannot be found in the sub-schema library, an informative diagnostic is issued and the new sub-schema is added to the library.

DELETING A SUB-SCHEMA

A sub-schema stored in the sub-schema library can be deleted from the library by specifying the purge parameter (P) in the DDLF control statement. A deck structure for deleting two sub-schemas is shown in figure 5-9.

The sub-schema file must be attached. The SB and P parameters are specified in the DDLF control statement. The 7/8/9 card or its equivalent designates the end of the control statements. This is followed by statements that specify the sub-schemas to be deleted. The sub-schema name is entered anywhere from column 7 through column 72. If more than one sub-schema name is entered, a comma must follow each sub-schema name.

COMPACTING A SUB-SCHEMA LIBRARY

When a sub-schema is purged or replaced in a sub-schema library, the index table is altered but the sub-schema is not physically removed from the library. This can result in much unused space in a sub-schema library. The sub-schema library compaction facility can be used to create a new, compacted sub-schema library from an existing sub-schema library, eliminating any purged or replaced sub-schemas. The sub-schema compaction facility is invoked by the NL parameter on the DDLF control statement. Figure 5-10 illustrates a deck structure for creating a compacted sub-schema library.

The existing sub-schema file is attached. The REQUEST/CATALOG and DEFINE control statements specify the logical file name of the new sub-schema file and assign it to a permanent file device. On the DDLF control statement, the SB parameter specifies the old sub-schema file name and the NL parameter specifies the new sub-schema file name and calls the library compaction facility. Active sub-schemas on the old library are transferred to the new library. The PURGE statement destroys the old library file.

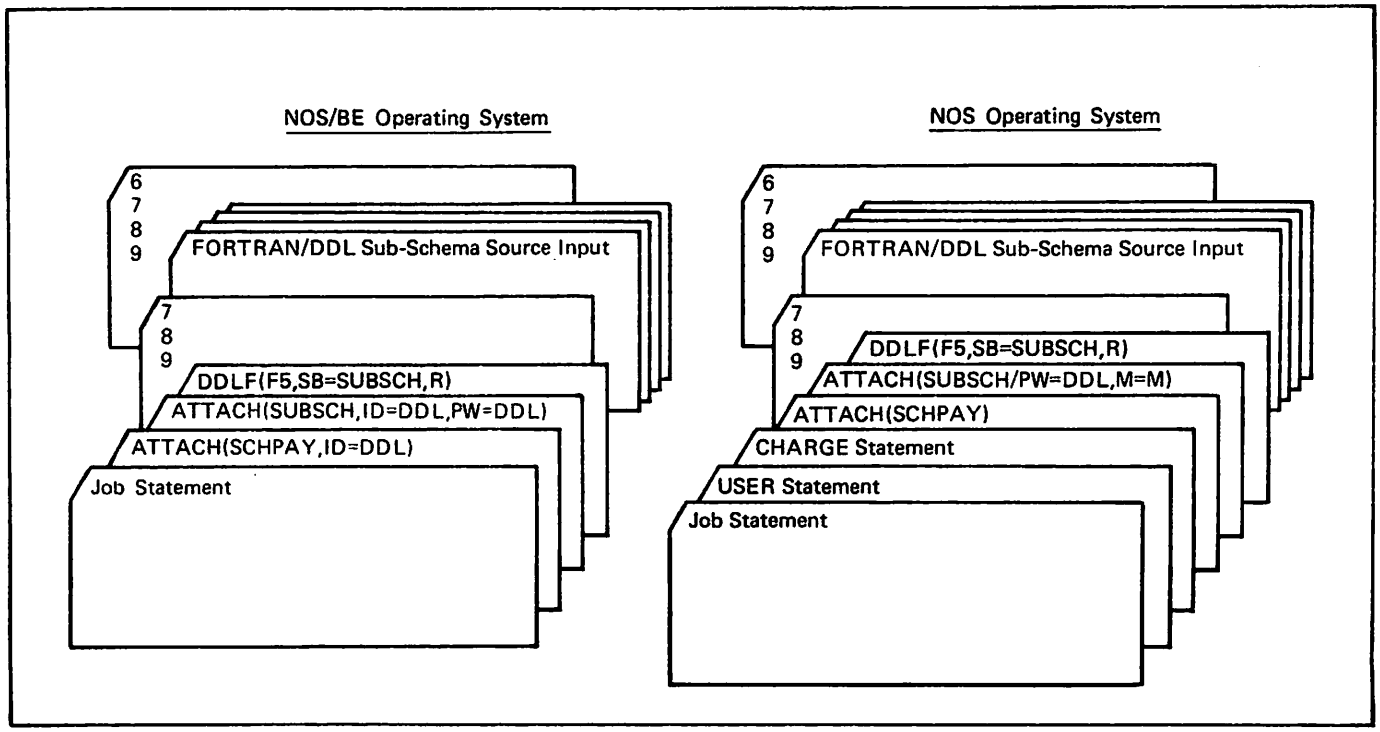


Figure 5-8. Replacing a Sub-Schema in the Library

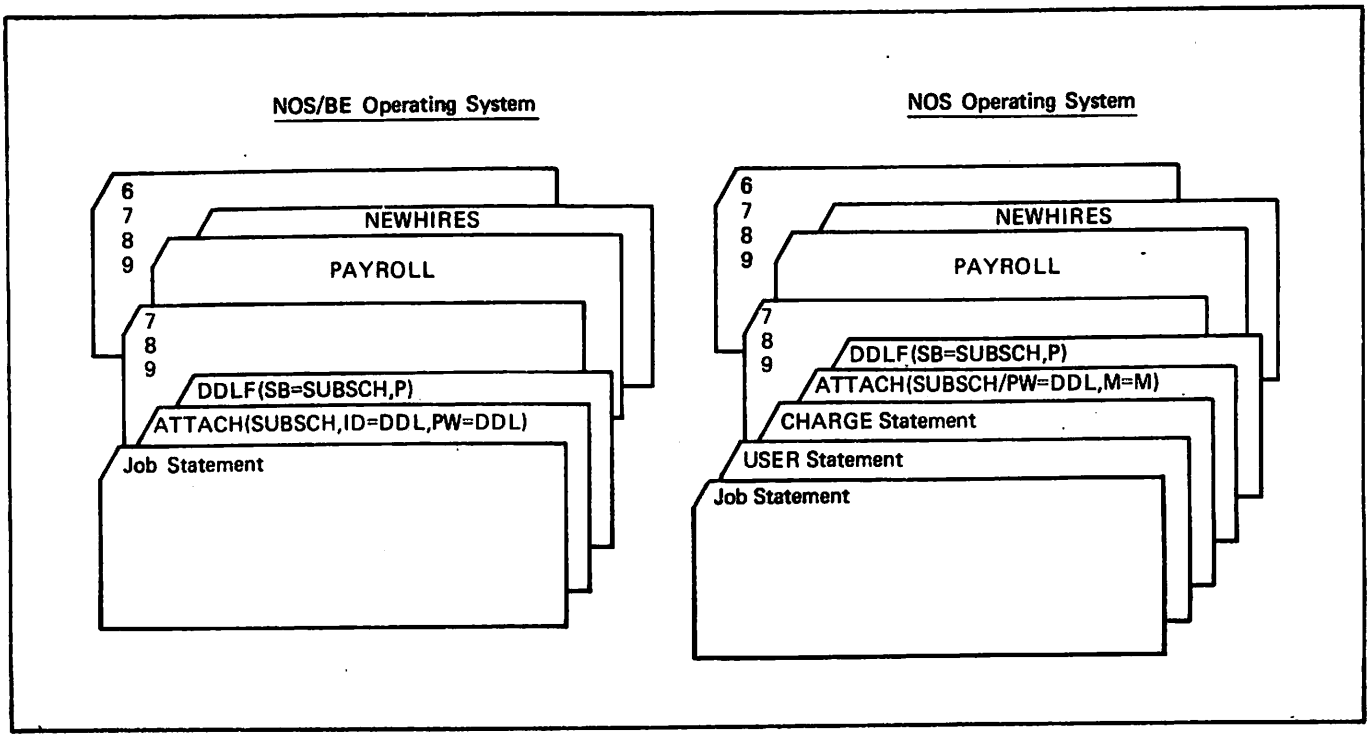


Figure 5-9. Deleting a Sub-Schema from the Library

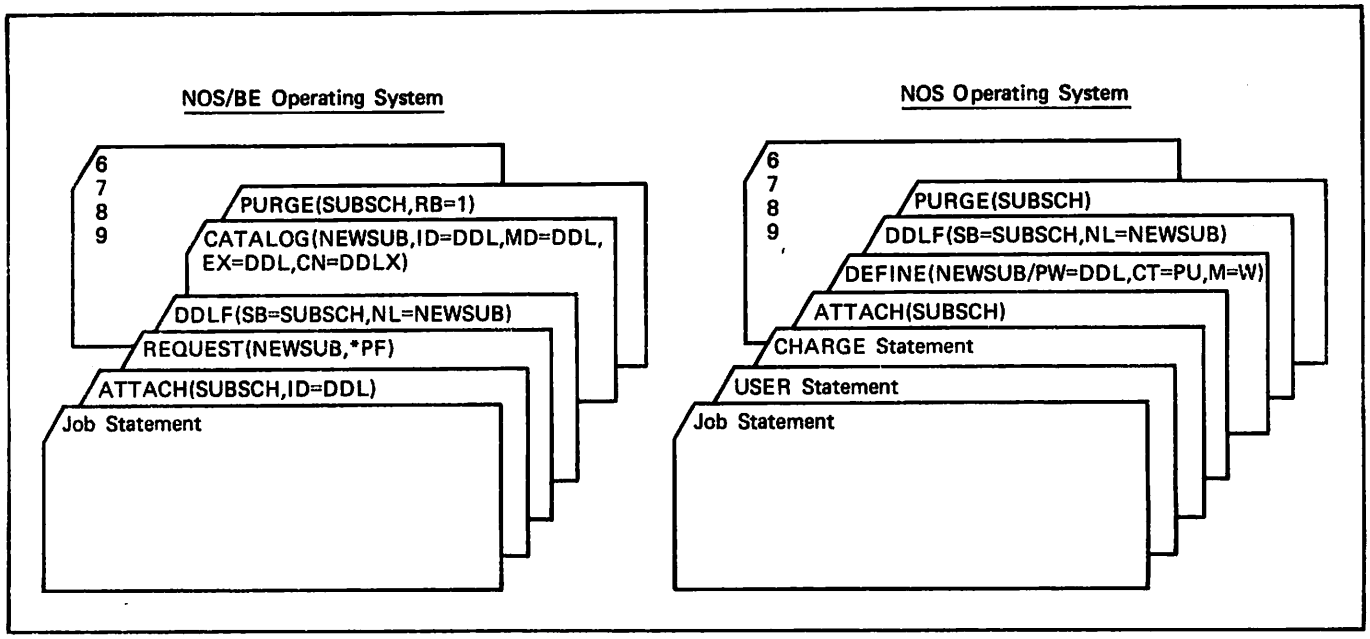


Figure 5-10 Compacting a Sub-Schema Library

COMPILATION OUTPUT

A listing of the FORTRAN/DDL source program is provided whenever a sub-schema is compiled. Each line of the listing corresponds to one source line or card image in the source program. The format and order of each line on the listing are identical to the format and order of the statements in the source program. The compilation output listings of the sub-schema used in the examples in this manual are included in appendix I.

The FORTRAN/DDL compiler assigns a line number to each input statement beginning with 00001. The line numbers are printed on the source listing starting in column 16. Diagnostic messages begin in column 3 of the listing. After the last input statement is listed, a compilation summary is printed. When relation statistics are applicable, relation names and their traversed areas are included.

The source listing can be suppressed by specifying L=0 on the FORTRAN/DDL control statement. Diagnostic messages and the compilation summary are the only listing that is printed.

A cross-reference list and a data map are not printed when the sub-schema is compiled. When the FORTRAN applications program containing DML statements is compiled, the variables and arrays defined in the sub-schema or generated by the DML preprocessor are not listed unless the DS parameter is specified on the DML control statement.

RECOMPILATION GUIDELINES

DDL generates a checksum (an identifying 1-word bit sum) for each area and relation in the schema. These checksums are the means for determining the need to recompile a sub-schema. If a checksum in a recompiled schema is different from the corresponding checksum in the previous schema, any sub-schema referencing the changed element must be recompiled. A sub-schema not referencing a changed element does not need to be recompiled.

DDL generates a single checksum for a sub-schema. This checksum is the means for determining the need to recompile a FORTRAN/DML program. If a checksum of a recompiled sub-schema differs from the previous checksum, all applications programs referencing that sub-schema must be processed again by the DML preprocessor and recompiled. When the FORTRAN/DML program is compiled, the checksum of the sub-schema it references is copied into the program binary output. When the program is executed, that checksum must be the same as a checksum of a sub-schema in the master directory. If the program references an invalid checksum, CDCS aborts the program and issues a diagnostic.

FIELD LENGTH REQUIREMENTS

Field length requirement for generating a minimum FORTRAN sub-schema is 60_8K. A minimum sub-schema contains one realm, one record, and one item. Field length requirements for a larger sub-schema can be estimated by using the formula:

60K + (n*12)

where n is the number of elements. An element is a realm, record, item, or relation.

The FORTRAN Data Manipulation Language (DML) is the execution time facility enabling data base access from a FORTRAN program. DML can be used with both FORTRAN 4 and FORTRAN 5 programs. DML consists of a series of statements, similar to FORTRAN statements, that are included in a FORTRAN program and processed by the DML preprocessor prior to compilation of the program. DML translates the statements into FORTRAN specification statements and CALL statements, which can then be compiled like other FORTRAN statements. At execution time, CDCS is called to access the data base.

This section describes the DML statements and the DML control statement, which causes the preprocessing of DML statements. The language version of the FORTRAN program is specified in the DML control statement.

DML STATEMENTS

The syntax requirements for DML statements are basically the same as for FORTRAN statements of the language version specified in the DML control statement. The syntax requirements are outlined in the FORTRAN Extended 4 reference manual and in the FORTRAN 5 reference manual. The exceptions to standard syntax are: a DML statement cannot be the object of a logical IF, a DML statement in a FORTRAN 4 program can contain a special long variable, and a DML statement cannot appear on the same line as another statement (that is, the $ statement separator cannot be used as is allowed in FORTRAN 4 statements).

For some DML statements, syntax requirements are different for FORTRAN 4 and FORTRAN 5 programs. In this section of the manual, these differences are indicated by two separate formats in the figure illustrating the statement format. Appendix E contains a syntax summary of DML statements for FORTRAN 5 applications programs. Appendix F contains a syntax summary of DML statements for FORTRAN 4 applications programs. The differences in FORTRAN 4 and FORTRAN 5 that affect syntax in DML statements are summarized in appendix J.

DML statements access files defined by the sub-schema. The sub-schema must be compiled before the DML preprocessor is called, and the sub-schema library file must be available to the DML preprocessor. (See section 5 for sub-schema compilation.) The files referenced in the DML statements should not be referenced elsewhere by conventional input/output statements, including the PROGRAM statement; they should be referenced exclusively by DML statements.

DML statements can appear both in the main program and in subprograms. Table 6-1 summarizes the DML statements and states where they are allowed to appear within the executable or nonexecutable statements of a program.

SUBSCHEMA STATEMENT

The SUBSCHEMA statement (figure 6-1) is required in every program unit accessing the data base defined by the sub-schema. It need not appear in a program unit that contains no DML statements. It is nonexecutable, and must appear after the specification statements and before the first DATA or NAMELIST statement, statement function, or executable statement.

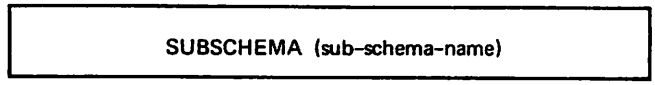


Figure 6-1. SUBSCHEMA Statement Format

Only one sub-schema can be used by a FORTRAN program. The sub-schema to be referenced must have been previously compiled with the same language version specified (F4 or F5) as specified in the DML control statement. The sub-schema must be available to the DML preprocessor.

At the point in a program unit where the SUBSCHEMA statement appears, the DML preprocessor copies the text of the type declarations and DATA statements resulting from the sub-schema compilation into the program. The variables and arrays appear in common blocks. In addition, several other variables and arrays are declared by DML. These variable and array names are reserved for use by DML, and should not be defined by the user. Appendix G contains a list of all the variable, array, and common block names created by the DML preprocessor.

The common block DB0000 is declared by DML in every FORTRAN/DML program. This common block contains variables used for error and status processing. The declarations for the block are as follows:

```
INTEGER DBREALM(3), DBTSTAT, . . . ,
    DBRnnnn(3), DBSnnnn, . . .
COMMON /DB0000/ DBREALM, DBSTAT, . . . ,
    DBRnnnn(3), DBSnnnn, . . .
```

These variables are used as follows:

DBREALM	Name of realm to which status code in DBSTAT applies.
DBSTAT	Status of DML request.
DBRnnnn	Name of realm whose sub-schema ordinal number is nnnn.
DBSnnnn	Current status of realm whose sub-schema ordinal number is nnnn.

INVOKE STATEMENT

The INVOKE statement (figure 6-2) must be executed before any other DML statement (except SUBSCHEMA, which is nonexecutable). It establishes connection between the applications program and CDCS. INVOKE must be executed in every program unit in which DML statements are executed.

TABLE 6-1. FORTRAN/DML STATEMENTS

Statement	Function	Position in Program
SUBSCHEMA	Identifies the sub-schema to be used by the program.	Must appear after specification statements and before any DATA or NAMELIST statements, statement function definitions, or executable statements of every program unit containing DML statements.
INVOKE	Establishes connection between the executing program and CDCS.	Must be executed before any other DML statement (except the SUBSCHEMA statement, which is nonexecutable); must appear in every program unit containing DML statements.
TERMINATE	Disconnects the executing program from CDCS.	Must be the last DML statement to be executed until a subsequent INVOKE statement.
PRIVACY	Establishes right of the program to access to a realm.	Optional; must be executed before the first execution of OPEN for a realm.
OPEN	Initiates processing of a realm.	Anywhere between INVOKE and TERMINATE; can only be executed when a realm is closed.
OPEN relation	Initiates processing of the realms joined in the relation.	Anywhere between INVOKE and TERMINATE; can only be executed if the realms are closed.
CLOSE	Ends processing of a realm.	Anywhere after OPEN and before TERMINATE; a realm can be opened and closed any number of times in a program.
CLOSE relation	Ends processing of the realms joined in the relation.	Anywhere after OPEN and before TERMINATE; realms in a relation can be opened and closed any number of times in a program.
START	Logically positions a realm for subsequent retrieval of records through a sequential DML READ statement.	Anywhere between OPEN and CLOSE for a realm.
START relation	Logically positions the root realm of the specified relation for subsequent retrieval of records through a DML READ relation statement.	Anywhere between OPEN and CLOSE for a relation.
READ	Transfers data from a record in the specified data base file to the variables included in the sub-schema description of the record.	Anywhere between OPEN and CLOSE for a realm.
READ relation	Transfers data from a record in each of the data base files comprising the relation to the variables included in the sub-schema descriptions of the records.	Anywhere between OPEN and CLOSE for a relation.
WRITE	Causes a record to be stored in a data base file consisting of the current values of the variables included in the sub-schema description of the record.	Anywhere between OPEN and CLOSE for a realm.
REWRITE	Logically replaces a record in a data base file.	Anywhere between OPEN and CLOSE for a realm, but following a READ or LOCK.
DELETE	Logically removes a record from a data base file.	Anywhere between OPEN and CLOSE for a realm, but following a READ or LOCK.
LOCK	Prevents other jobs from updating the specified realm.	Anywhere between OPEN and CLOSE for a realm.
UNLOCK	Releases a lock on a specified realm.	Anywhere between OPEN and CLOSE for a realm.

```
INVOKE
```

Figure 6-2. INVOKE Statement Format

The name of the program unit in which the first INVOKE statement is executed is used by CDCS as the run-time-id for log files. Subsequent INVOKE statements cause verification that identical sub-schemas are being referenced in each program unit.

TERMINATE STATEMENT

The TERMINATE statement (figure 6-3) disconnects the applications program from CDCS. After TERMINATE has been executed, no other DML statements can be executed until another INVOKE has been executed. The TERMINATE statement must be executed before the FORTRAN STOP or END statement.

```
TERMINATE
```

Figure 6-3. TERMINATE Statement Format

TERMINATE permits CDCS to close files and return resources used by the job. If the job terminates abnormally before TERMINATE is executed, CDCS automatically performs the same functions.

OPEN STATEMENT

The OPEN statement (figure 6-4) prepares a realm for processing. No other statement related to the realm (except PRIVACY) can be executed when the realm is not open. The realm must be among those described in the sub-schema.

FORTRAN 5 Format

```
OPEN ( { realm-name     } [ ,MODE = { I  } ] [,ERR=s] )
       { relation-name  }           { IO }
                                    { O  }
```

FORTRAN 4 Format

```
OPEN ( { realm-name     } [ ,MODE = { I  } ] )
       { relation-name  }           { IO }
                                    { O  }
```

I Open realm for input only.

IO Open realm for input and output; default when MODE is omitted.

O Open realm for output only (not valid for relations).

Figure 6-4. OPEN Statement Format

An OPEN relation statement prepares the realms joined in the relation for processing. The relation name must be included in the sub-schema. Relations are normally opened for input (MODE=I). The relation can be opened for input and output (MODE=IO) if the user wishes to have locking of records read for the relation occurrence, or if individual realms in the relation are to be updated.

If any of the realms included in the relation are already open, no action occurs for that realm. Any previous mode setting remains in effect.

When a realm is opened for input (MODE=I), only READ, LOCK, UNLOCK, and CLOSE can be executed. When a realm is opened for input/output (MODE=IO), READ, WRITE, DELETE, REWRITE, LOCK, UNLOCK, and CLOSE can be executed. Output mode (MODE=O) must be specified for creation of a new file; it is not valid for an existing file. When a realm is opened for output, only WRITE, LOCK, UNLOCK, and CLOSE can be executed.

Example:

```
OPEN (CUL-DE-SAC, MODE = I)
```

The realm named CUL-DE-SAC is opened in read-only mode.

For FORTRAN 5 programs only, the optional error specifier ERR=s is available to indicate the statement where execution is to continue if an error condition occurs on execution of the OPEN statement. In the specification ERR=s, s must be the statement label of an executable FORTRAN or DML statement within the same program unit that contains the specification. For more information, see the subsection ERR and END Specifiers.

CLOSE STATEMENT

The CLOSE statement (figure 6-5) ends processing of the specified realm or the realms joined in the specified relation. The only DML statement that can be executed when a realm is closed is another OPEN statement.

FORTRAN 5 Format

```
CLOSE ( { realm-name    } [,ERR=s] )
        { relation-name }
```

FORTRAN 4 Format

```
CLOSE ( { realm-name    } )
        { relation-name }
```

Figure 6-5. CLOSE Statement Format

Realms closed by a CLOSE relation statement should not be explicitly closed by a CLOSE realm statement. If a realm closed by a CLOSE relation statement is already closed, no action is taken for that realm.

For FORTRAN 5 programs only, the optional error specifier ERR=s is available to indicate the statement where execution is to continue if an error condition occurs on execution of the CLOSE statement. In the specification ERR=s, s must be the statement label of an executable FORTRAN or DML statement within the same program unit that contains the specification. For more information, see the subsection ERR and END Specifiers.

READ STATEMENT

The READ statement (figure 6-6) causes CDCS to read a record from the specified realm, disassemble it into the variables and arrays included in the sub-schema description of the record, and set these variables and

FORTRAN 5 Format

READ ({ realm-name / relation-name } [,KEY { = / .EQ. / .GT. / .GE. } item-name] [,ERR=s] [,END=s])

FORTRAN 4 Format

READ ({ realm-name / relation-name } [,KEY { = / .EQ. / .GT. / .GE. } item-name])

Figure 6-6. READ Statement Format

arrays in the FORTRAN program to their current values from the record. The record must be described in the sub-schema. If any type conversion is implied by the correspondence between the schema and sub-schema descriptions of a data item, it is performed at this time.

A relation read causes CDCS to read a relation occurrence. A relation occurrence consists of one record from each of the realms comprising the relation. The FORTRAN variables and arrays included in the sub-schema description of each record are set to their current values from the relation occurrence.

If the KEY option is omitted, the read is sequential; the record or relation occurrence located is the next record or relation occurrence from the current location of the realm or relation. The value 100_8 is returned in DBSTAT or in the data base status block to denote end-of-file. If the KEY option is used, the item referenced must be set before the read to the primary or alternate key value of the record desired. The item must have been defined in the sub-schema, and must refer to a primary or alternate key for the realm. For relations, the key must be in the root realm, or first realm named in the schema relation definition.

For FORTRAN 4 programs only, if the key is a special long variable, the array-name without subscripts must be specified in the KEY option.

Example:

```
MYKEY(1) = 10HABCDEFGHIJ
MYKEY(2) = 10HKLMNOPQRST
MYKEY(3) = 10HUVWXYZABCD
READ (REALMA,KEY=MYKEY)
```

The value ABCDEFGHIJKLMNOPQRSTUVWXYZABCD is established for the key that is referenced in the DML READ statement by the unsubscripted array name MYKEY. In the sub-schema, the item MYKEY was declared an integer array with three elements. (For more information about the special long variable, see the Character Data subsection.)

The data item used as the key value for the read can be of any data type except logical. If it is complex or double precision, it is treated as real. If it is complex, the imaginary part is discarded and it is treated as a real value. If it is double precision, the least significant part is discarded and it is treated as real.

To locate the desired record, the comparison specified by the KEY option is performed. The record returned is the first record in the realm (or in the root realm of the relation) that satisfies the specified comparison. If the comparison is .EQ. or =, the key of the record must exactly match the value in the data item (no conversion is performed, except as described for complex and double precision items). If the comparison is .GE., the key of the record must be greater than or equal to the value in the data item. Values are compared by numerical magnitude. If the comparison is .GT., the key in the record must be strictly greater than the value of the data item.

If the primary or alternate key is described in the schema as a character key, comparison by means of the ASCII or COBOL collating sequence is not available in a FORTRAN program; comparison is by display code only.

Example:

If the keys on the realm PONT-LEVEQUE are as follows:

```
.
.
.
21
31
34
36
37
.
.
.
```

then the following statements read the record whose key is 31:

```
MM = 31
READ (PONT-LEVEQUE, KEY = MM)
```

and the following statements read the record whose key is 36:

```
LAST = 35
READ (PONT-LEVEQUE, KEY .GE. LAST)
```

For FORTRAN 5 programs only, the optional specifiers END=s and ERR=s are available to indicate the statement where execution is to continue if an end-of-file condition or an error condition occurs on execution of the READ statement. The END specifier is applicable only for a sequential read. In the specification END=s or ERR=s, s must be the statement label of an executable FORTRAN or DML statement within the same program unit that contains the specification. For more information see the subsection ERR and END Specifiers.

START STATEMENT

The START statement (figure 6-7) positions a realm or relation for subsequent retrieval of records. Before the START statement is executed, the realm or relation must have been opened with MODE=I or MODE=IO specified.

Execution of the START statement causes CDCS to establish the current position of the realm or relation. If the KEY option is specified, the realm or relation is positioned at the first qualifying record occurrence or relation occurrence. In a START realm statement, the item name specified must be a data item defined in the sub-schema and must be a primary or alternate key for the realm. In a START relation statement, the item name specified must be a key that is defined in the root realm, the first realm named in the schema relation definition. For FORTRAN 4 programs, the item name can be a special long variable that is used as described in the READ Statement subsection.

If the KEY option is omitted, the relational operator is assumed to be an equals sign; the item name is assumed to be the primary key with a value equal to the current contents of that item.

For FORTRAN 5 programs only, the optional error specifier ERR=s is available to indicate the statement where execution is to continue if an error condition occurs on execution of the START statement. In the specification ERR=s, s must be the statement label of an executable FORTRAN or DML statement within the same program unit that contains the specification. For more information, see the subsection ERR and END Specifiers.

The START statement does not cause a record to be transferred to the program. The statement is normally followed by a sequential DML READ statement, which performs the read from the position established by the execution of the START statement. The START statement can be specified any number of times.

Under most circumstances, either a START statement or a READ statement that specifies the KEY option can be used to position a realm or relation. However, when a user wants to position a relation for subsequent sequential read operations and that particular relation is qualified by a RESTRICT statement, the START relation statement must be used. Under this one circumstance, positioning the relation with the READ statement (READ relation-name KEY) would cause unpredictable results.

WRITE, REWRITE, AND DELETE STATEMENTS

The formats of the WRITE, REWRITE, and DELETE statements are shown in figure 6-8.

FORTRAN 5 Format

{WRITE / REWRITE / DELETE} (realm-name [,ERR=s])

FORTRAN 4 Format

{WRITE / REWRITE / DELETE} (realm-name)

Figure 6-8. Formats of WRITE, REWRITE, and DELETE Statements

These statements all create or modify individual records in the named realm. The realm must have been defined in the sub-schema.

FORTRAN 5 Format

START ({realm-name / relation-name} [,KEY {= / .EQ. / .GT. / .GE.} item-name] [,ERR=s])

FORTRAN 4 Format

START ({realm-name / relation-name} [,KEY {= / .EQ. / .GT. / .GE.} item-name])

Figure 6-7. START Statement Format

WRITE uses the current values of all the variables defined for the record in the sub-schema to construct a record, and then writes the record to the named realm. All primary and alternate keys must be set appropriately before the record is written. Any data items in the schema record that are not defined in the sub-schema are given null values before the record is written to the data base. The actual null value for each data class is detailed in the CDCS 2 reference manual.

REWRITE replaces the last record read with a new record based on the current values of all the variables defined for the record in the sub-schema. The primary key must not have changed since the last read, or an error results. The only way to replace a record with a new record with a different primary key is to delete the old record and then write the new record.

DELETE removes the record most recently read from the realm. The primary key is checked to ensure that it has not changed since the last read.

For FORTRAN 5 programs only, the optional error specifier ERR=s is available to indicate the statement where execution is to continue if an error condition occurs on execution of the WRITE, REWRITE, or DELETE statement. In the specification ERR=s, s must be the statement label of an executable FORTRAN or DML statement within the same program unit that contains the specification. For more information, see the subsection ERR and END Specifiers.

Example:

The keys on the realm REBLOCHON are as follows:

```
.
.
.
70
120
190
550
663
664
.
.
.
```

If item PKEY is defined in the schema as the primary key, and the following statements are executed:

```
      PKEY = 662
C     CREATE A NEW RECORD
      WRITE (REBLOCHON)
      PKEY = 300
      READ (REBLOCHON, KEY .GT. PKEY)
C     THIS WILL DELETE RECORD WITH
        PRIMARY KEY OF 550
      DELETE (REBLOCHON)
      PKEY = 100
      READ (REBLOCHON, KEY .GE. PKEY)
      .
      .
      .
C     REWRITE RECORD WITH
        PRIMARY KEY OF 120
      REWRITE (REBLOCHON)
```

The keys on the file are then as follows:

```
.
.
.
70
120
190
662 (new record)
663
664
.
.
.
```

The CDCS locking mechanism requires that a REWRITE or DELETE statement be preceded by a DML READ statement or by a DML LOCK statement. The DML READ statement locks a record; the DML LOCK statement locks the realm.

Normally, statements intervening between the last READ and the REWRITE would alter items in the record (but not the primary key), since otherwise the identical record is rewritten.

LOCK AND UNLOCK STATEMENTS

The formats of the LOCK and UNLOCK statements are shown in figure 6-9.

FORTRAN 5 Format

{LOCK / UNLOCK} (realm-name [,ERR=s])

FORTRAN 4 Format

{LOCK / UNLOCK} (realm-name)

Figure 6-9. Formats of LOCK and UNLOCK Statements

The LOCK statement prevents other jobs from updating the specified realm until an UNLOCK statement is executed. Although CDCS always locks the current record whenever the realm is opened for I/O, LOCK locks the whole realm. The UNLOCK statement releases the lock on the specified realm. The realm must have been described in the sub-schema.

For FORTRAN 5 programs only, the optional error specifier ERR=s is available to indicate the statement where execution is to continue if an error condition occurs on execution of the LOCK or UNLOCK statement. In the specification ERR=s, s must be the statement label of an executable FORTRAN or DML statement within the same program unit that contains the specification. For more information, see the subsection ERR and END Specifiers.

PRIVACY STATEMENT

The PRIVACY statement (figure 6-10) establishes the right of a program to access a realm. It has no effect unless the realm was defined with an ACCESS-CONTROL

FORTRAN 5 Format

PRIVACY (realm-name, [MODE= { I / IO / O } ,] PRIVACY = { character-constant / variable-name / array-name-1 })

I	Restricts access to input operations.
IO	Allows access for both input and output; default when MODE is omitted.
O	Restricts access to output operations.
character-constant	A privacy key of 1 through 30 characters delimited by apostrophes.
variable-name	Name of a variable that is defined as type CHARACTER *30 and contains a 1- through 30-character privacy key that is left-justified and blank filled.
array-name-1	Unsubscripted name specifying a 3-word array containing a 1- through 30-character privacy key that is left-justified and blank filled in the field of the array.

FORTRAN 4 Format

PRIVACY (realm-name, [MODE= { I / IO / O } ,] PRIVACY = { Hollerith-constant / array-name-2 })

I	Restricts access to input operations.
IO	Allows access for both input and output; default when MODE is omitted.
O	Restricts access to output operations.
Hollerith-constant	A privacy key of 1 through 30 characters delimited by quotation marks.
array-name-2	Unsubscripted name specifying a 3-word array containing a 1- through 30-character privacy key that is left-justified and blank filled in the field of the array.

Figure 6-10. PRIVACY Statement Format

clause in the schema. If the realm was defined with an ACCESS-CONTROL clause in the schema, the PRIVACY statement must supply the privacy key before the realm can be opened. To access several realms, each requiring a privacy key, the FORTRAN program must contain a PRIVACY statement for each realm. Similarly, to open a relation that joins realms with each realm requiring a privacy key for access, a PRIVACY statement for each realm must be specified before the OPEN relation statement.

The PRIVACY statement specifies the privacy key. The language elements used to specify the privacy key depend on the version of FORTRAN; figure 6-10 details what each version allows. FORTRAN 5 allows a character-constant, variable-name, or an array-name to provide the privacy key. FORTRAN 4 allows a Hollerith-constant or an array-name to provide the key. For both versions of FORTRAN, the privacy key must be a string of from 1 to 30 characters made available through the PRIVACY statement. At execution time, CDCS compares the key specified in the PRIVACY statement with the lock specified in the schema ACCESS-CONTROL clause. If a program attempts to open a realm without supplying the correct privacy key, the program is terminated.

The type of access allowed when the character string matches depends on the setting of the MODE option. Table 6-2 details which mode option is used to satisfy the types of locks in the schema ACCESS-CONTROL clause. If no mode is specified, IO is assumed.

TABLE 6-2. SCHEMA ACCESS-CONTROL CLAUSE

Privacy Mode Option	Update	Retrieval
I		X
O	X	
IO	X	X

FORTRAN 4 example:

```
      PRIVACY(GORGONZOLA, MODE = IO, PRIVACY =
     1 "THIRTY CHARACTERS ARE ALLOWED.")
```

If the character string in the ACCESS-CONTROL clause in the schema is also THIRTY CHARACTERS ARE ALLOWED, then the realm GORGONZOLA can be opened in any mode.

FORTRAN 5 example:

```
CHARACTER*30 PKEY
DATA PKEY/'OPEN SESAME'/
.
PRIVACY(TALES,PRIVACY=PKEY)
```

In the FORTRAN 5 program, the character-variable PKEY is declared having a length of 30 characters and is given the value OPEN SESAME. The PRIVACY statement specifies the variable PKEY. If the character string in the ACCESS-CONTROL clause in the schema is also OPEN SESAME, then the realm TALES can be opened in any mode.

LISTING CONTROL DIRECTIVES

The DML preprocessor automatically generates listing control directives as part of the translated FORTRAN program. These directives have the form:

FORTRAN 4	FORTRAN 5
C/ LIST,NONE	C$ LIST(ALL=0)
C/ LIST,ALL	C$ LIST(ALL)

The first directive inhibits the listing of all succeeding FORTRAN statements. The second directive resumes listing of FORTRAN statements. The DML preprocessor inserts these directives immediately after the SUBSCHEMA and INVOKE statements, as in the following example:

```
.
.
.
**  SUBSCHEMA(BIRD)
C/  LIST,NONE
C/  LIST,ALL
C
**  INVOKE
C/  LIST,NONE
C/  LIST,ALL
.
.
.
```

The effect of the listing control directives is to inhibit the listing of the FORTRAN statements generated by FDBF. These directives can be suppressed by the DS parameter on the DML control statement, in which case all FORTRAN statements generated by FDBF appear on the FORTRAN source listing. Note that the CALL statements generated as a result of executable DML statements are not suppressed by the listing control directives.

ERROR PROCESSING

During execution of FORTRAN/DML processing, various error conditions are detected by CDCS and CRM. Some of these errors are fatal and cause immediate job step abort; others are nonfatal and allow the job to continue (while possibly leaving the current input/output operation incomplete). In the latter case, the user might wish to be informed of the error condition that was detected, possibly to print a message or to perform other operations. For this reason, several mechanisms have been established for communication between CDCS and a user program.

This subsection on error processing describes some of these mechanisms. Variables and arrays defined by CDCS are available for user access so that more information can be obtained when an error occurs. CDCS also can return error codes and status information in the data base status block if DML.DBST is called in a FORTRAN program. FORTRAN 5 programs can use ERR and END specifiers in DML statements to specify special processing when an error or end-of-file condition occurs. Some CDCS and CRM diagnostic codes returned in DBSTAT or in the data base status block do not necessarily represent error conditions; rather, they are informative diagnostics. A following subsection indicates some informative diagnostic codes.

RESERVED VARIABLES

Variables and arrays defined by FDBF are available for user access so that more information can be obtained when an error occurs. (See appendix G for a complete list of reserved variables.) The following variables are available:

DBREALM(3)	Name of the realm on which the most recent DML operation was performed. The name is a Hollerith constant, left-justified in the 3-word array, with blank fill.
DBSTAT	Error number in octal of the most recent error; either a CDCS error number or a CRM error number. Numbers between 600 and 677 are CDCS errors; all others are CRM errors. See the appropriate reference manual for a complete list of errors. If the value of DBSTAT is zero, no error occurred. DBSTAT should be printed using an O format.
DBRnnnn(3)	Realm name, in the same format as DBREALM. One variable in this form is reserved for each realm; the number nnnn is the realm ordinal. Realm ordinals are assigned in the same order as realms are declared in the sub-schema; the first ordinal is 1.
DBSnnnn	Error status for realm DBRnnnn. Same format as DBSTAT. If no error occurred on the last access to realm nnnn, DBSnnnn is zero; otherwise, it is set to the number of the error that occurred.

After each DML operation, the value in DBSTAT reflects the status of the input/output operation. If the statement has been successfully executed, the value in DBSTAT is 0. If an error has prevented successful execution of the statement, DBSTAT contains the octal value of the CRM or CDCS error or status (such as end-of-information). The status variable should be checked after each DML operation to maintain integrity of the data base.

For relation processing, several status or error codes might result from execution of a single DML READ. In this case, DBREALM contains the name of the last realm read that produced a nonzero status. DBSTAT contains the value of that status or error code. The status of each realm involved in the relation can be found in the DBSnnnn field for that realm. The CDCS 2 reference manual describes the status codes that can result from a READ relation.

With this scheme, the user can either determine the most recent error occurring in any realm (with DBSTAT) or the current status of a particular realm (with DBSnnnn).

DATA BASE STATUS BLOCK

The user can include an array, called a data base status block, in the FORTRAN program to which CDCS returns data base status information. CDCS updates the data base status block after every operation on a data base file or relation. Information returned to the data base status block includes the following:

- CRM or CDCS error code (or codes, for relations)
- Sub-schema item ordinal for item-level errors
- CRM code indicating file position of a realm
- Function being executed when an error occurred
- Rank in a relation of the realm on which either a CRM or CDCS error occurred or a special relation condition (null record or control break) occurred
- Name of the realm on which an error occurred

The user program communicates the location and length of the data base status block to CDCS by calling a routine. The length must be at least 1 word and should not be greater than 11 words. CDCS returns as much information as possible in the given length. The block should be of type INTEGER. The format of the call is as follows:

```
CALL DMLDBST(status-block,length)
```

where status-block is the name of the array to be used by CDCS and length is the length in words of the status block.

The information returned in the data base status block is as follows:

Word Number	Contents
1	CRM or CDCS error code in octal for the last data base operation on a realm or relation; zero if no error has occurred. Note that only error codes are returned. If the last operation was a relation retrieval for which a null record or control break occurred, the status codes for these conditions are not returned.
2	Sub-schema item ordinal for CDCS item-level errors. Item-level errors include data validation errors, record mapping errors, and item-level data base procedure errors. Value is zero if no errors have occurred. The item ordinal assigned by the FORTRAN DDL compiler is identified on the sub-schema source listing.
3	CRM code in octal indicating file position of the realm when the last data base operation was performed. A file position code is returned when open, close, read, and start operations are performed. For a relation operation, the file position code indicates the position of the root realm when the last operation was performed. The following list includes the file position codes that most commonly occur during data base processing. 10_8 End-of-key-list, which occurs when the last primary key value associated with a given alternate key has been returned during a read operation using an alternate key value. 20_8 End-of-record, which occurs when a record has been returned during a read operation. 100_8 End-of-information, which occurs when a sequential read operation is attempted after the previous read operation returned the last record in the file. This file position code can be used in a FORTRAN program to determine end-of-file. If a FORTRAN program uses information returned in the data base status block to determine end-of-file, the file position code should be used. The file position codes are CRM codes; further information on file position (the FP field of the file information table) is in the CYBER Record Manager Advanced Access Methods reference manual.
4	Not used (reserved for future use).
5	Function being performed when an error or relation condition occurred; one of the following character strings (left-justified and blank filled): UNDEFINED OPEN RAN-READ CLOSE SEQ-READ START WRITE RECOVR-PT REWRITE TIME DELETE PRIVACY REL-READ LOCK REL-NEXT UNLOCK REL-STRT END Value is undefined if no error has occurred.
6	For a relation operation, the rank of the realm on which a CRM or CDCS error occurred; zero if no error has occurred. The root realm of the relation has a rank of one. An error on a realm during a relation read terminates the operation. Consequently, there is never more than one rank in the relation which has a CRM or CDCS error.
7	For a relation operation, the lowest rank on which a control break occurred; zero if no control break has occurred. The root realm of the relation has a rank of one. All realms in the relation with a rank greater than the rank stored in this word also have control break status or null status (null status overrides control break status).
8	For a relation operation, the lowest rank for which there was a null record; zero if no null record.

The root realm of the relation has a rank of one. All realms in the relation with a rank greater than the rank stored in this word also have null record occurrences.

9,10,11 Name of the realm on which an error has occurred; stored as a left-justified, blank filled character string. Contains blanks if no error has occurred, or if the error has occurred on a non-I/O operation or an I/O operation not explicitly requested by the user.

An example of the declarations for the data base status block is as follows:

```
      INTEGER        STATUS(11),FUNCTN,RELSTAT(3)
      INTEGER        REALM(3),IER,IORD
      EQUIVALENCE    (STATUS(1),IER),
     *               (STATUS(2),IORD),
     *               (STATUS(3),IFP),
     *               (STATUS(5),FUNCTN),
     *               (STATUS(6),RELSTAT(1)),
     *               (STATUS(9),REALM)
        .
        .
        .
      CALL DMLDBST(STATUS,11)
        .
        .
        .
```

Note that the length of the status block is variable; CDCS updates only those words contained within the given length. If, for instance, the user program was not interested in the realm name, the last three words could be omitted. The array STATUS would then be eight words long. Only these eight words would be updated by CDCS after every data base operation.

Although the length of the data base status block is variable, the length provided must be sufficient to allow complete specification of each unit of status information, or that information will not be updated. Thus, three words are required for both the relation status (words 6, 7, and 8) and the realm name (words 9, 10, and 11). Additionally, CDCS updates words 2, 3 and 4 as a single unit that returns auxiliary status; for information to be returned in any one of these words, length must be provided for all three words. If, for example, the block length is six words, there is not enough space for CDCS to return all three words of the relation status, so no relation status is returned.

The data base status block consists of character and numeric information. Octal codes are returned in words 1 and 3. Decimal integers are returned in words 2, 6, 7, and 8. Character strings are returned as follows: a 10-character string in word 5 and a 30-character string in words 9, 10, and 11. A FORMAT statement that is used in printing words of the data base status block should appropriately specify the contents of each word that is being printed.

The routine DMLDBST can be called at any point in the FORTRAN program after the INVOKE. It need be called only once. The status block specified in the call is updated for any data base operation performed after the call. Each time DMLDBST is called, the status block is initialized to zeros or blanks, so it should not be called after execution of a DML statement if the status of that statement is desired. Only one status block can exist at a time for a FORTRAN program. If DMLDBST is called more than once in a program, the status block defined in the last call is the one that is updated by CDCS.

If DMLDBST is not called, the FORTRAN program can still reference the variable DBSTAT and the status words for each realm in the common blocks set up by FDBF (described under Error Processing).

ERR AND END SPECIFIERS

The optional error and end-of-file specifiers in the form ERR=s and END=s are allowed only in FORTRAN 5 programs. The s refers to the statement label of an executable FORTRAN or DML statement where execution is to continue if an error or end-of-file condition occurs during execution of the DML statement containing the specifier.

The ERR=s specifier can be added to the following DML statements: OPEN, CLOSE, READ, WRITE, DELETE, REWRITE, LOCK, and UNLOCK. If a CRM or CDCS error condition occurs during the execution of a DML statement containing this specifier, the following steps occur:

1. Execution of the DML statement terminates.
2. DML status variables are set to the CDCS or CRM error code.
3. The appropriate fields in the data base status block are set if DMLDBST has been called.
4. Execution continues with the statement labeled s.

Control is not transferred to the statement labeled s in the error specifier when a CDCS relation condition indicating a null record occurrence or control break (that is, a status code value of 627_8 and 632_8, respectively) occurs. Unlike execution of a FORTRAN 5 statement, execution of the program is not terminated when an error condition occurs on the execution of a DML statement that does not contain the ERR=s specifier; rather, execution continues at the next executable statement.

The END=s specifier can be added only to the DML READ statement and is applicable only for a sequential read. If a DML READ statement contains this specifier and an end-of-file condition is encountered, the following steps occur:

1. Execution of the READ statement terminates.
2. DML status variables are set to 100_8.
3. Execution continues with the statement labeled s.

For an explanation of the sequence in which the status variables are set, see the Informative Diagnostic Codes subsection.

Unlike execution of a FORTRAN 5 statement, execution of the program is not terminated when an end-of-file condition occurs on the execution of a DML READ statement that does not contain the END=s specifier; rather, execution continues at the next executable statement.

INFORMATIVE DIAGNOSTIC CODES

Some diagnostic codes returned in DBSTAT and in the data base status block do not represent error conditions; rather, the codes are informative diagnostics reporting a condition that has occurred. These conditions can be handled in the FORTRAN/DML program. Table 6-3 lists some of the informative diagnostics that are returned in the following status elements: the data base status block, DBSTAT, and DBSnnnn. Not all of the codes indicated are returned in all of the elements. The table indicates the component of DMS-170 recognizing the condition, the code returned, the message, and the status element to which the code is returned. Those informative codes that are returned in the data base status block are returned in the first word, with the exception of the code 100_8; exceptions concerning this code are indicated later in this subsection. Some conditions are recognized by CRM and others by CDCS; the user can refer to the CYBER Record Manager Advanced Access Methods reference manual or to the CDCS 2 reference manual for detailed information.

The following paragraphs indicate the significance of informative codes to the user.

The 100_8 diagnostic code indicates an attempt to read beyond end-of-information. This is an informative code that can be used by a FORTRAN program in determining end-of-file only when the code is returned to DBSTAT or DBSnnnn. When returned in the first word of the data base status block, this code indicates a real error. However, the data base status block provides a field (the third word) to which file position information is returned; the 100_8 file position code indicates end-of-information and can be used by a FORTRAN program in determining end-of-file. The timing of the writing of the 100_8 code to the status elements (DBSTAT, DBSnnnn, or the third word of the data base status block) can be important to the user who is considering placing count variables or other routines in association with a sequential DML READ statement. The status elements are set to 100_8 when the READ statement has been executed one additional time after the last record was read. For example, if there are 10 records being read, DBSTAT is set to 100_8 after the 11th time the READ statement is executed. Similarly, if a FORTRAN 5 applications program contains the READ statement with the END=s specifier, it would be after the 11th time the READ statement is executed that execution is transferred to the statement labeled s.

The 445_8 diagnostic code occurs when no record is found whose key matches the value of the key specified in the DML READ statement.

The 506_8 diagnostic code occurs when an alternate key value is not found. CRM issues this diagnostic when a read is attempted at the end of the alternate key file.

The 627_8 diagnostic code indicates a null record occurrence on a realm. The 632_8 code indicates a control break on a realm. These conditions can occur on a relation read. These codes are not returned in the data base status block. See the CDCS 2 reference manual for detailed information on relation processing.

The 652_8 diagnostic code indicates that an open was attempted on a realm already opened. The 654_8 diagnostic code indicates that a close was attempted on a realm already closed. These diagnostics can occur on the statements OPEN or CLOSE of a relation or a realm.

The 663_8 code indicates a deadlock condition; that is, a situation arising in concurrent data base access when two or more applications programs are contending for a resource that is locked by one of the other programs. The 663_8 code indicates that CDCS has unlocked all locks held by the FORTRAN program. The user should provide appropriate code to handle recovery from a deadlock. Refer to the CDCS 2 reference manual for more information.

TABLE 6-3. INFORMATIVE DIAGNOSTIC CODES

Code	Message	Returned to Data Base Status Block	Returned to DBSTAT and DBSnnnn	Diagnostic From
100_8	CANNOT SEQUENTIALLY POSITION BEYOND FILE BOUNDS		X	CRM
445_8	KEY NOT FOUND - FILE POSITION ALTERED - REQUEST IGNORED	X	X	CRM
506_8	ALTERNATE KEY NOT FOUND	X	X	CRM
627_8	No message. Indicates a null record occurrence was encountered on a file during relation processing.		X	CDCS
632_8	No message. Indicates a control break was encountered on a file during relation processing.		X	CDCS
652_8	AREA an ALREADY OPEN	X	X	CDCS
654_8	AREA an NOT OPEN	X	X	CDCS
663_8	DEADLOCK ON AREA an	X	X	CDCS

RECOVERY POINT DEFINITION

CDCS provides the data base functions of logging and recovery for use by the data administrator. Their use is transparent to the user program. The user, however, can define a recovery point to CDCS. This is a point to which the data base would be reset so that the program could be easily restarted should recovery of the data base be necessary.

Specification of recovery points is accomplished in the user program by issuing a call to the subroutine DMLRPT. The call marks the point for recovery purposes. The format of the call is:

```
CALL DMLRPT (num,comment)
```

where num is an integer containing the unique entry point number assigned by CDCS and comment is a 3-word array containing a Hollerith constant with blank fill. The user can retain the recovery point number for reference purposes if desired. The Hollerith constant is written to the journal log file along with the recovery point number.

Execution of the subroutine DMLRPT causes the following events to occur in the order given. The user program is suspended until these events have completed:

1. All I/O buffers for data base areas are flushed.
2. A recovery point log record is force written to the log file for the data base.
3. The quick recovery file for the data base is emptied.

On return from this subroutine, the user is assured that the data base can be recovered to its current state (barring such disasters as simultaneous destruction of both the data base and the log file).

The creation of a recovery point does incur overhead, since CDCS activity halts for all users until the preceding three events have been completed. To reduce this overhead, an application might choose to create a recovery point every fourth transaction. Judicious use of recovery points can aid in recovery, but misuse can severely impact throughput.

DML CONTROL STATEMENT

The DML preprocessor processes DML statements and translates them into FORTRAN statements. Therefore, the input to DML is a file containing a FORTRAN program with added DML statements, and the output from DML is a file containing the translated version of the input file. The output file can be compiled by the FORTRAN Extended 4 compiler or the FORTRAN 5 compiler as specified by the language version (LV) parameter in the DML control statement.

The format and options of the DML control statement are shown in figure 6-11. The parameters can appear in any order.

Example:

```
DML (LV=F4, SB=BRIE, I, E=ERRFILE, ET=W)
```

This control statement specifies that the DML preprocessor is to generate statements for the FORTRAN Extended 4 compiler, that the sub-schema library is on the file BRIE, that input is on the file COMPILE, that output is to be written to DMLOUT, that error messages are to be written to ERRFILE, and that the job step aborts if any errors of warning level or higher occur.

COMPILATION/EXECUTION

Following is a list of steps needed to execute a FORTRAN data base applications program.

1. Define and compile a schema.
2. Define and compile required sub-schemas.
3. Include the compiled sub-schemas in the master directory by executing the DBMSTRD utility.
4. Create a FORTRAN program containing DML statements and execute the DML preprocessor.
5. Compile the applications program, using the DML output file as input to the FORTRAN compiler.
6. Specify an LDSET(LIB=DMSLIB) control statement.
7. Execute the applications program.

Before execution of an applications program, the user must include the appropriate control statement to attach any independent file needed by the program. When the compiled program is executed, CDCS monitors and interprets all requests for action on relations and on data base files. All data base files and index files that are referenced in the sub-schema used by the applications program are attached automatically by CDCS. If any one of these files does not exist when the program is executed, the program is aborted. The master directory and log files are attached at CDCS initialization.

When CDCS is active, it resides at a system control point; an applications program resides at a user control point. CDCS and an applications program can reside at the same control point, however, when the CDCS Batch Test Facility is used. This facility is usually used to test and debug applications that require a sub-schema and master directory different from those active with CDCS at a system control point. To use the CDCS Batch Test Facility the same steps as mentioned previously must be performed with the exception that the following three steps are substituted for steps 6 and 7:

6. Attach the master directory file and, if applicable, attach log files.
7. Make the DMS-170 library available by specifying the LIBRARY(DMSLIB) control statement.
8. Execute the program by specifying the CDCSBTF control statement.

Refer to appendix H for more information about the CDCS Batch Test Facility.

DML (SB=lfn,p1,p2,p3,p4,p5,p6)

SB=	lfn	Name of file containing sub-schema library; default is SUBLFN. SB=0 is not allowed.
p1	LV=op	Language version. Specifies the version of FORTRAN for which the DML preprocessor is to generate statements. The value of op can be one of the following:
		F4 Specifies FORTRAN Extended 4.
		F5 Specifies FORTRAN 5.
	LV	Same as LV=F5.
	omitted	Same as LV=F4. (This default value can be changed at installation time.)
	LV=0	Not allowed.
p2	I=lfn	Name of file containing FORTRAN source program with added DML statements to be preprocessed by DML.
	I	Same as I=COMPILE.
	omitted	Same as I=INPUT.
	I=0	Not allowed.
p3	O=lfn	Name of file to which translated version of FORTRAN source program is to be written. DML statements appearing in FORTRAN program are translated into FORTRAN statements before being written to this file.
	O	Same as O=DMLOUT.
	omitted	Same as O=DMLOUT.
	O=0	No output is produced.
p4	E=lfn	Name of file to which error diagnostics are to be written.
	E	Same as E=ERRS.
	omitted	Same as E=OUTPUT.
p5	ET=op	Error termination code. Four levels of errors are defined; if an error of the specified level or higher takes place, the job is aborted to an EXIT(S) control statement (NOS/BE) or EXIT control statement (NOS). The abort does not take place until DML is finished. The possible values for op, in increasing order of severity, are as follows:
		T Trivial. The syntax of the usage is correct, but it is questionable.
		W Warning. The syntax is incorrect, but the processor has been able to recover by making an assumption about what was intended.
		F Fatal. An error that prevents DML from processing the statement in which it occurs. Unresolvable semantic errors also fall into this category. Processing continues with the next statement.
		C Catastrophic. Compilation cannot continue. However, DML advances to the end of the current program unit and attempts to process the next program unit.
	ET	Same as ET=F.
	omitted	Same as ET=0.
	ET=0	Do not abort the job step even if errors occur (except for control statement errors).

T and W errors do not invalidate the output file produced by DML (the file specified by the O option). The translated code on the file can still be compiled (barring any errors not related to DML), but the results might not be what the user intended. F and C errors, however, produce an output file that cannot be successfully compiled by FORTRAN.

p6	DS	Directive suppression. Listing control directives are not generated; all FORTRAN statements generated by FDBF appear on the FORTRAN source listing.
	omitted	Listing control directives are generated; FORTRAN statements generated by FDBF do not appear on the FORTRAN source listing.

FORTRAN CALL statements generated as a result of executable DML statements always appear on the FORTRAN source listing regardless of DS specification.

Figure 6-11. DML Control Statement

SAMPLE DECK STRUCTURES

Figure 6-12 illustrates the control statements needed to execute the DML preprocessor and to compile and execute a FORTRAN applications program. The ATTACH statement references the sub-schema library containing the compiled FORTRAN sub-schema referenced in the applications program. The referenced sub-schema must be included in the master directory and CDCS must be at a system control point. The LDSET(LIB=DMSLIB) control statement causes the DMS-170 library to be made available for execution of the program. The LGO control statement initiates execution of the relocatable binary program contained on that file.

The control statements necessary to compile, but not to execute, a program are all those indicated in figure 6-12 except that the LDSET(LIB=DMSLIB) and LGO statements are omitted. By removing these two control statements, the user can check complition of a program before specifying execution.

Figure 6-13 illustrates the control statements needed to execute the DML preprocessor, and to compile and execute a FORTRAN applications program using the CDCS Batch Test Facility. The first ATTACH statement references the sub-schema library containing the compiled FORTRAN sub-schema referenced in the applications program. The sub-schema library must be available to the DML preprocessor. The second ATTACH statement references the master directory file, which must contain the referenced sub-schema and must be available to execute the program. The LIBRARY(DMSLIB) statement makes the DMS-170 library available for program execution. The CDCSBTF statement causes execution of the program through the CDCS Batch Test Facility. The CDCSBTF statement must reference the file containing the relocatable binary program (for this application that file is the system default file, LGO).

The control statements necessary to compile, but not to execute, the FORTRAN/DML program are the same as those indicated in figure 6-13 except that the ATTACH of the master directory, the LIBRARY(DMSLIB), and the CDCSBTF(LGO) statements are omitted. By removing these three control statements, the user can check complition of a program before specifying execution.

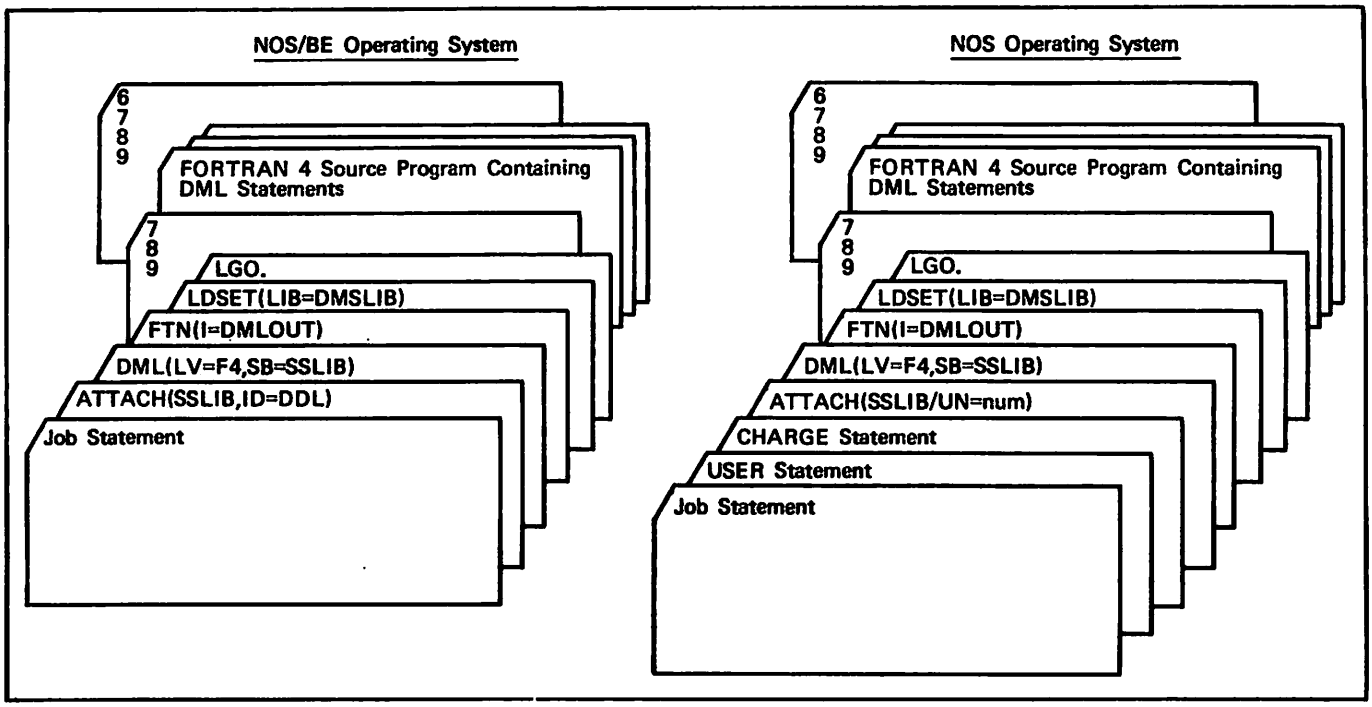


Figure 6-12. Program Compilation and Execution With CDCS at a System Control Point

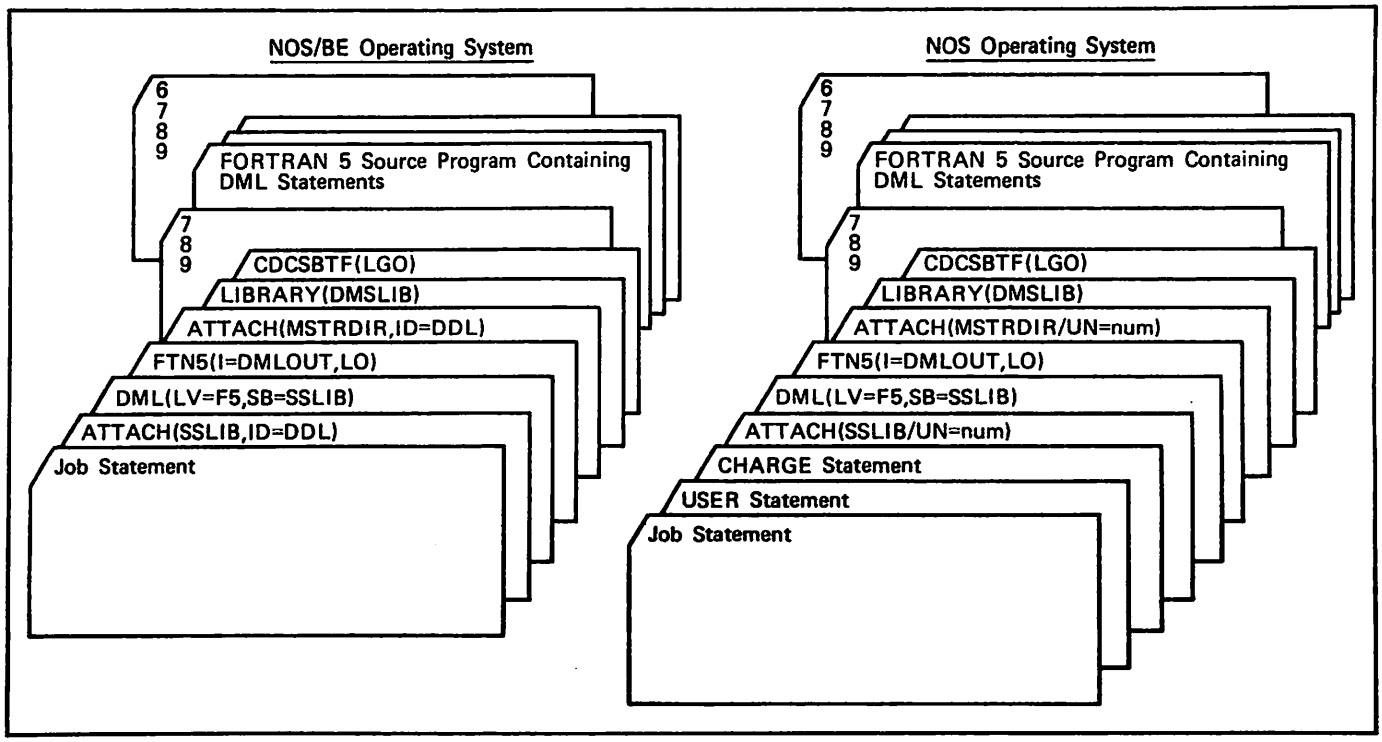


Figure 6-13. Program Compilation and Execution With CDCS Batch Test Facility

This section contains two examples using FORTRAN/DDL and the DML preprocessor. The first example is a relatively simple application using a schema, two sub-schemas, and a FORTRAN 5 program corresponding to each sub-schema. This example illustrates the use of an application with CDCS at a system control point. The second example shows the definition of a relation and use of the relation by a FORTRAN 4 program. This example illustrates the use of the CDCS Batch Test Facility.

The figures in this section show only the program source input of the examples. The compilation output listings of the example programs (that includes schemas, sub-schemas, master directories, and FORTRAN/DML programs) are shown in appendix I.

USING SUB-SCHEMAS

The first example, the university example, assumes the existence of a data base containing the test results for a series of tests administered to students. The schema for the data base, as well as a description of the items it contains, are shown in figure 7-1.

The first program of the example updates selected records by computing new probabilities based on new test scores. It also computes and prints the correlation coefficient, which is a measure of how closely the new data matches the old data. The sub-schema for this program is shown in figure 7-2. The only items required from the schema are the test number (the primary key, required in every sub-schema), the number of students, and the probability of a right answer for each question.

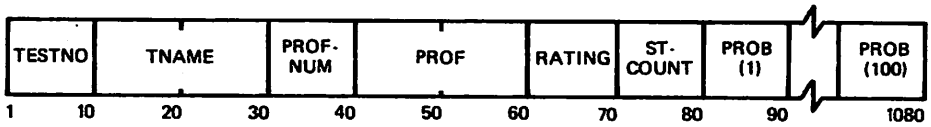
```
SUBSCHEMA PROBSS, SCHEMA = TEST-FILES
REALM TESTS
RECORD R1
INTEGER TESTNO,STCOUNT
REAL PROB(100)
END
```

Figure 7-2. First Sub-Schema for University Example

The program in figure 7-3 reads new records from a sequential file (NEWTPE) and uses the test number to locate the desired record in the data base. It then computes the correlation coefficient and updates the data base. The formula for the coefficient is shown in figure 7-4; in the program, the arrays PROB and NEWPROB are equivalenced to X and Y, respectively. The program calls DMLDBST and uses the array STATBLK (which is declared in the program) to return data base status information on error conditions. The error specifier is incorporated in both the DML READ and REWRITE statements. (See the Data Base Status Block subsection for more information.)

The second program uses the test number, the name and number of the professor, the professor's current rating (according to the students), and the probabilities of correct answers for each question. The sub-schema for the program is shown in figure 7-5. The program in figure 7-6 prints the name and rating of each professor, and then the name of each test the professor has given. For each test, the average number of correct responses per question is computed and printed. Since the data base

```
SCHEMA TEST-FILES.
AREA TESTS.
RECORD R1 WITHIN TESTS.
      TESTNO TYPE FIXED.
      TNAME TYPE CHARACTER 20.
      PROFNUM TYPE FIXED.
      PROF PICTURE "X(20)".
      RATING TYPE FLOAT.
      STCOUNT TYPE FIXED.
      PROB TYPE FLOAT OCCURS 100 TIMES,
          CHECK VALUE 0.0 THRU 1.0.
DATA CONTROL.
AREA TESTS KEY IS TESTNO, DUPLICATES ARE NOT ALLOWED,
      KEY IS ALTERNATE PROFNUM, DUPLICATES ARE INDEXED,
      SEQUENCE IS ASCII.
```



TESTNO	Test number	RATING	Rating of professor; from 1.0 to 10.0
TNAME	Test name	STCOUNT	Number of students taking test
PROFNUM	Identification number assigned to professor	PROB	For each question on test, probability that a student gets the right answer; from 0.0 to 1.0
PROF	Name of professor		

Figure 7-1. Schema for University Example

```
      PROGRAM NEWTEST
      INTEGER STATBLK(5)
      REAL NEWPROB(100),X(100),Y(100)
      EQUIVALENCE(PROB,X),(NEWPROB,Y),(CORCOEF,R)
      SUBSCHEMA(PROBSS)
      N = 100
      INVOKE
      CALL DMLDBST(STATBLK,5)
      OPEN(5,FILE='NEWTPE',STATUS='OLD',ACCESS='SEQUENTIAL')
      OPEN(TESTS)
C     MAIN LOOP
 10   READ(5,*,ERR=40,END=50) TESTNO,NEWST,NEWPROB
      READ(TESTS,KEY=TESTNO,ERR=45)
      SUMXY = SUMX = SUMY = SUMXSQ = SUMYSQ = 0.0
      DO 20 I=1,N
      SUMXY = SUMXY + X(I)*Y(I)
      SUMX = SUMX + X(I)
      SUMY = SUMY + Y(I)
      SUMXSQ = SUMXSQ + X(I)**2
 20   SUMYSQ = SUMYSQ + Y(I)**2
      R = (N*SUMXY - SUMX*SUMY)/
     1    (SQRT(N*SUMXSQ - SUMX**2) * SQRT(N*SUMYSQ - SUMY**2))
      PRINT *, ' TEST NO. = ', TESTNO,
     1         ' CORRELATION COEFFICIENT = ', CORCOEF
      NEWTOT = STCOUNT + NEWST
      DO 30 I=1,N
 30   PROB(I) = (PROB(I)*STCOUNT + NEWPROB(I)*NEWST)/NEWTOT
      STCOUNT = NEWTOT
      REWRITE(TESTS,ERR=45)
      GO TO 10
 40   PRINT *, ' ERROR ON FILE READ'
 45   PRINT 46, STATBLK
 46   FORMAT (1X,'STATUS BLOCK'/
     1        1X,O4,2X,I5,2X,O3,2X,I2,2X,A10)
 50   CLOSE(TESTS)
      CLOSE(5,STATUS='DELETE')
      TERMINATE
      STOP
      END
```

Figure 7-3. First FORTRAN 5 Program for University Example

$$R = \frac{N\sum_{i=1}^{N} X_i Y_i - \sum_{i=1}^{N} X_i \sum_{i=1}^{N} Y_i}{\sqrt{N\sum_{i=1}^{N} X_i^2 - \left(\sum_{i=1}^{N} X_i\right)^2}\ \sqrt{N\sum_{i=1}^{N} Y_i^2 - \left(\sum_{i=1}^{N} Y_i\right)^2}}$$

Figure 7-4. Formula for Correlation Coefficient

```
SUBSCHEMA RATE, SCHEMA = TEST-FILES
REALM TESTS
RECORD R1
INTEGER TESTNO, PROFNUM
CHARACTER *20 TNAME, PROF
REAL RATING, PROB(100)
END
```

Figure 7-5. Second Sub-Schema for University Example

file is being read by a key (an alternate key), the value of PROFNUM must be set before the data base can be read. Therefore, the sequential file PROFS, which contains selected values of PROFNUM, is read first. Then, with the value of PROFNUM established, PROFNUM is used to position the file on the data base read to the first record of the professor. Then the remaining records for that professor are read sequentially.

The input for the master directory for the university example is shown in figure 7-7. The master directory must contain permanent file information. For the NOS operating system, the permenent file identification must be specified in the form UN "usernum" as shown in figure 7-7. For the NOS/BE operating system, ID "file-id" must be specified.

Figure 7-8 indicates the control statements required for each step involved in setting up the data base, compiling the FORTRAN/DML program named NEWTEST, and executing it with CDCS at a system control point. The control statements for compiling the sub-schemas and creating the sub-schema library show both sub-schemas being compiled with one set of control statements. The data base administrator should perform steps 1, 2, and 3 and should provide the applications programmer with the necessary information to use the sub-schema. With the sub-schema information provided, the programmer can write the FORTRAN/DML program. The control

```
      PROGRAM RATER
      INTEGER ALTKEY
      INTEGER STATBLK(5)
      SUBSCHEMA (RATE)
      INVOKE
      CALL DMLDBST(STATBLK,5)
      OPEN(5,FILE='PROFS',STATUS='OLD',ACCESS='SEQUENTIAL')
      OPEN(TESTS)
100   READ (5,*,END=900) PROFNUM
      READ(TESTS,KEY=PROFNUM,ERR=800)
      ALTKEY = PROFNUM
      PRINT 12, PROF, RATING
200   SUM = 0.0
      DO 300 I=1,100
      SUM = SUM + PROB(I)
300   CONTINUE
      AVG = SUM/100.0
      PRINT 13, TNAME, AVG
      READ(TESTS,ERR=800,END=100)
      IF (PROFNUM .NE. ALTKEY) GO TO 100
      GO TO 200
800   PRINT 14, STATBLK
900   CLOSE(TESTS)
      TERMINATE
      CLOSE(5,STATUS='DELETE')
      STOP
12    FORMAT (' PROF = ', A20, ' RATING = ', F4.1)
13    FORMAT (1X,A20,4X,F4.3)
14    FORMAT (1X,'STATUS BLOCK'/
     1        1X,O4,2X,I5,2X,O3,2X,I2,2X,A10)
      END
```

Figure 7-6. Second FORTRAN 5 Program for University Example

```
SCHEMA NAME IS TEST-FILES
       FILE NAME IS TFILE.
       AREA NAME IS TESTS
           PFN IS "TESTS"
           UN IS "QUD0468"
       INDEX PFN IS "RATEIN"
           UN IS "QUD0468".
SUBSCHEMA NAME IS PROBSS
       FILE NAME IS SUBLIB2.
SUBSCHEMA NAME IS RATE
       FILE NAME IS SUBLIB2.
```

Figure 7-7. Input for Master Directory for University Example

statements to compile and execute the sample program NEWTEST are shown in step 4 of figure 7-8. The control statements indicate that the sub-schema library must be attached for DML preprocessing and that the independent file NEWTPE must be attached for execution; the data base files are automatically attached. The LDSET(LIB=DMSLIB) control statement and the statement specifying the program binary output file (for this example, the default file LGO) are necessary to execute the program.

USING RELATIONS

This second example illustrates building a data base, compiling a FORTRAN 4 program, and executing the program through the CDCS Batch Test Facility. After the data base is built, the data base is read by a FORTRAN 4 program using a read relation statement and is updated by the program with a REWRITE statement. The program is executed using the CDCS Batch Test Facility, which can be used for testing and debugging a program that requires a schema, sub-schema, and a master directory different from those active with CDCS at a system control point.

The following steps are included in the example:

1. Schema compilation
2. Sub-schema compilation
3. Master directory build
4. Preprocessing the FORTRAN/DML program by DML
5. Compilation of the FORTRAN program
6. Testing of the program with the CDCS Batch Test Facility

The control statements necessary to perform the preceding steps are illustrated in figure 7-9. Include also are control statements necessary to establish files as permanent files and to print CDCS output.

The data base is to contain three files (areas). A FILE statement is included for each area. These statements define the file characteristics to CYBER Record Manager.

The schema for the data base is illustrated in figure 7-10. The schema, PAYDATA, describes the variables included in each record, the record format for each area, the keys associated with each record, and any desired relations between records.

The sample schema contains three areas. The first area (PFILE) contains the name (last, first, and middle initial) of all employees, as well as the title, social security number, and pay code for each. The second area (EXMPT) contains the social security number, monthly salary, date

	Step	NOS/BE Operating System	NOS Operating System
1.	Schema Compilation	Job Statement REQUEST(TFILE,*PF) FILE(TESTS,FO=IS,RT=F,FL=1080,MRL=1080 MNR=1080,KT=I,KL=10,XN=RATEIN) DDL3(SC=TFILE) CATALOG(TFILE,ID=DDL,RP=999) 7/8/9 in Column 1 Schema Source Input 6/7/8/9 in Column 1	Job Statement USER Statement CHARGE Statement DEFINE(TFILE/CT=PU,M=R FILE(TESTS,FO=IS,RT=F,FL=1080,MRL=1080 MNR=1080,KT=I,KL=10,XN=RATEIN) DDL3(SC=TFILE) 7/8/9 in Column 1 Schema Source Input 6/7/8/9 in Column 1
2.	Sub-Schema Compilation and Library Creation	Job Statement REQUEST(SUBLIB2,*PF) ATTACH(TFILE,ID=DDL) DDLF(F5,SB=SUBLIB2,SC=TFILE) CATALOG(SUBLIB2,ID=DDL,RP=999) 7/8/9 in Column 1 1st Sub-Schema Source Deck 2nd Sub-Schema Source Deck 6/7/8/9 in Column 1	Job Statement USER Statement CHARGE Statement DEFINE(SUBLIB2/CT=PU,M=W) ATTACH(TFILE) DDLF(F5,SB=SUBLIB2,SC=TFILE) 7/8/9 in Column 1 1st Sub-Schema Source Deck 2nd Sub-Schema Source Deck 6/7/8/9 in Column 1
3.	Master Directory Creation	Job Statement REQUEST(MSTRDIR,*PF) ATTACH(TFILE,ID=DDL) ATTACH(SUBLIB2,ID=DDL) DBMSTRD(NMD=MSTRDIR,LD) CATALOG(MSTRDIR,ID=DDL,RP=999) 7/8/9 in Column 1 Master Directory Source Input 6/7/8/9 in Column 1	Job Statement USER Statement CHARGE Statement DEFINE(MSTRDIR/CT=PU,M=W) ATTACH(TFILE) ATTACH(SUBLIB2) DBMSTRD(NMD=MSTRDIR,LD) 7/8/9 in Column 1 Master Directory Source Input 6/7/8/9 in Column 1
4.	Program NEWTEST Compilation and Execution	Job Statement ATTACH(SUBLIB2,ID=DDL) DML(LV=F5,SB=SUBLIB2,ET=W) FTN5(I=DMLOUT,LO) ATTACH(NEWTPE,ID=MYID) LDSET(LIB=DMSLIB) LGO. 7/8/9 in Column 1 FORTRAN/DML Program 6/7/8/9 in Column 1	Job Statement USER Statement CHARGE Statement ATTACH(SUBLIB2/UN=num) DML(LV=F5,SB=SUBLIB2,ET=W) FTN5(I=DMLOUT,LO) ATTACH(NEWTPE) LDSET(LIB=DMSLIB) LGO. 7/8/9 in Column 1 FORTRAN/DML Program 6/7/8/9 in Column 1

Figure 7-8. Control Statements for University Example

of hire, and vacation and sick leave hours accrued by all exempt employees. The third area (NEXMPT) contains the same information for all hourly employees, and, additionally, any required union deductions.

The first relation, FIXUP1, establishes a link between areas PFILE and EXMPT. The second relation, FIXUP2, establishes a link between areas PFILE and NEXMPT. This allows the applications program to perform I/O operations on the joined areas with a single I/O statement.

A sample sub-schema is shown in figure 7-11. The sub-schema describes the portion of the data base to be accessed by the applications program. The ALIAS clause assigns a unique name, TSOCSEC, to the data item SOCSEC contained in record SREC1 to distinguish it from the data item of the same name contained in SREC2. The applications program must reference the data item by its alias.

After the schema and sub-schema are compiled, using the DDL3 and DDLF control statements shown in figure 7-9, they are bound together into a single master directory. The master directory also contains the operating system permanent file names and passwords. The master directory build is initiated by the DBMSTRD control statement. The program source input for the master directory build is shown in figure 7-12.

The preceding steps are performed by the data administrator. After these steps have been completed, the FORTRAN programmer uses DML to access the data base.

NOS/BE Operating System

```
JOB Statement
REQUEST(PFILE,*PF)
REWIND(PFILE)
CATALOG(PFILE,ID=MYID)
RETURN(PFILE)
REQUEST(PNDX,*PF)
REWIND(PNDX)
CATALOG(PNDX,ID=MYID)
RETURN(PNDX)
REQUEST(EXMPT,*PF)
REWIND(EXMPT)
CATALOG(EXMPT,ID=MYID)
RETURN(EXMPT)

FILE(EXMPT,FO=IS,RT=F,FL=27,MNR=27,
     KL=9,EMK=YES,EFC=3)
FILE(PFILE,FO=IS,RT=F,FL=64,MNR=64,
     XN=PNDX,EMK=YES,KL=9,EFC=3)
FILE(NEXMPT,FO=IS,RT=F,FL=29,MNR=29,
     KL=9,EMK=YES,EFC=3)

DDL3(DS,SC=PAYDATA)
DDLF(F4,SC=PAYDATA,SB=SUBLIB)
REQUEST(MSTRDIR,*PF)
DBMSTRD(NMD=MSTRDIR,LO)
CATALOG(MSTRDIR,ID=MYID)
DML(LV=F4,SB=SUBLIB,DS)
FTN(I=DMLOUT)
LIBRARY(DMSLIB)
CDCSBTF(LGO)
REWIND(CDCSOUT)
COPYSBF(CDCSOUT,OUTPUT)

7/8/9 in Column 1
Schema Source Statements

7/8/9 in Column 1
Sub-Schema Source Statements

7/8/9 in Column 1
Input for Master Directory Build

7/8/9 in Column 1
FORTRAN 4 Source Statements

6/7/8/9 in Column 1
```

NOS Operating System

```
JOB Statement
USER Statement
CHARGE Statement
DEFINE(PFILE,EXMPT,PNDX/PW=SECRET)
DEFINE(MSTRDIR)
PERMIT(PFILE,usernum1=W,usernum2=W)
PERMIT(EXMPT,usernum1=W,usernum2=W)
PERMIT(PNDX,usernum1=W,usernum2=W)
PERMIT(MSTRDIR,usernum1=W,usernum2=W)
RETURN(PFILE,EXMPT,PNDX)

FILE(EXMPT,FO=IS,RT=F,FL=27,MNR=27,
     KL=9,EMK=YES,EFC=3)
FILE(PFILE,FO=IS,RT=F,FL=64,MNR=64,
     XN=PNDX,EMK=YES,KL=9,EFC=3)
FILE(NEXMPT,FO=IS,RT=F,FL=29,MNR=29,
     KL=9,EMK=YES,EFC=3)
DDL3(DS,SC=PAYDATA)
DDLF(F4,SC=PAYDATA,SB=SUBLIB)
DBMSTRD(NMD=MSTRDIR,LO)
DML(LV=F4,SB=SUBLIB,DS)
FTN(I=DMLOUT)
LIBRARY(DMSLIB)
CDCSBTF(LGO)
REWIND(CDCSOUT)
COPYSBF(CDCSOUT,OUTPUT)

7/8/9 in Column 1
Schema Source Statements

7/8/9 in Column 1
Sub-Schema Source Statements

7/8/9 in Column 1
Input for Master Directory Build

7/8/9 in Column 1
FORTRAN 4 Source Statements

6/7/8/9 in Column 1
```

Figure 7-9. Sample Control Statements for Data Base Build

An applications program to update the data base and the output of the program are shown in figure 7-13. This program makes the following changes to the data base:

A new record is written to PFILE and to EXMPT. The record contains the values defined in the source program. DML statements open, write, and close the files. To create the file, the first open specifies MODE=O. Then data is written to the file. The file is then closed and reopened with MODE=IO so it can be read and updated. After each of these operations, the value of DBSTAT is printed to check the status of the operation.

A single record is updated. A relation read is used to read a record from each of the realms PFILE and EXMPT. The record key for the read is the current value of SOCSEC. The relation links records in PFILE and EXMPT; the record read from each area is the one in which the value of the social security number is equal to the key value. A new value is assigned to MSALARY and the record is rewritten to area EXMPT. In order to verify the change, the new record is read using a read relation, and selected values are printed.

```
       SCHEMA PAYDATA.
/*     SET UP A SCHEMA TO BE USED       */
/*     BY FORTRAN 4 APPLICATIONS        */
       AREA PFILE.
       RECORD IS REC WITHIN PFILE.
           SOCSEC TYPE CHARACTER 9.
           LNAME TYPE CHARACTER 10
               OCCURS 2 TIMES.
           FNAME TYPE CHARACTER 10.
           MI TYPE CHARACTER 1.
           PCODE TYPE CHARACTER 4.
           TITLE TYPE CHARACTER 10
               OCCURS 2 TIMES.
       AREA EXMPT.
       RECORD IS SREC1 WITHIN EXMPT.
           SOCSEC TYPE CHARACTER 9.
           MSALARY TYPE CHARACTER 4.
           VACHRS TYPE CHARACTER 4.
           SICKHRS TYPE CHARACTER 4.
           EMPDATE TYPE CHARACTER 6.
       AREA NEXMPT.
       RECORD IS SREC2 WITHIN NEXMPT.
           SOCSEC TYPE CHARACTER 9.
           HSALARY TYPE CHARACTER 4.
           UNION PICTURE "XX".
           EMPDATE PICTURE "X(6)".
           VACHRS PICTURE "X(4)".
           SICKHRS PICTURE "X(4)".
       DATA CONTROL.
/* KEYS ARE DEFINED HERE.  NOTE THAT   */
/* SOCSEC MUST BE QUALIFIED BY RECORD  */
/* SINCE IT IS NOT A UNIQUE NAME       */
           AREA NAME IS PFILE
               KEY IS SOCSEC OF REC
               DUPLICATES ARE NOT ALLOWED
               KEY IS ALTERNATE PCODE
               DUPLICATES ARE NOT ALLOWED.
           AREA NAME IS EXMPT
               KEY IS SOCSEC OF SREC1
               DUPLICATES ARE NOT ALLOWED.
           AREA NAME IS NEXMPT
               KEY IS SOCSEC OF SREC2
               DUPLICATES ARE NOT ALLOWED.
       RELATION NAME IS FIXUP1
           JOIN WHERE SOCSEC OF REC
               EQ
           SOCSEC OF SREC1.
       RELATION NAME IS FIXUP2
           JOIN WHERE SOCSEC OF REC
               EQ
           SOCSEC OF SREC2.
```

Figure 7-10. Schema for Payroll Example

```
      SUBSCHEMA EXAMP1, SCHEMA=PAYDATA
C
C     THIS COMMENT DEMONSTRATES THE
C     ABILITY OF FDBF TO CARRY SUB-SCHEMA
C     COMMENTS TO THE FORTRAN SOURCE
C     LISTING IF THE DS PARAMETER IS
C     SPECIFIED ON THE DML CONTROL
C     STATEMENT
C
      ALIAS (ITEM) TSOCSEC=SOCSEC.SREC1
      REALM PFILE,EXMPT
      RECORD REC
      INTEGER SOCSEC,LNAME(2),FNAME,MI,
     +        PCODE,TITLE(2)
      RECORD SREC1
      INTEGER TSOCSEC,MSALARY,VACHRS,
     +        SICKHRS,EMPDATE
      RELATION FIXUP1
      END
```

Figure 7-11. Sub-Schema for Payroll Example

```
SCHEMA NAME IS PAYDATA
    FILE NAME IS PAYDATA.
    AREA NAME IS PFILE
        PFN IS "PFILE"
        UN "QUD0468" PW "SECRET"
    INDEX FILE ASSIGNED
        PFN IS "PNDX"
        UN "QUD0468" PW "SECRET".
    AREA NAME IS EXMPT
        PFN IS "EXMPT"
        UN "QUD0468" PW "SECRET".
    AREA NAME IS NEXMPT
        PFN IS "NEXMPT"
        UN "QUD0468" PW "SECRET".
SUBSCHEMA NAME IS EXAMP1
    FILE NAME IS SUBLIB.
```

Figure 7-12. Input for Master Directory for Payroll Example

```
      PROGRAM FDBFTST(OUTPUT=64)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     A PROGRAM TO TEST SOME FEATURES OF THE FORTRAN DATA BASE FACILITY
C
C     DECLARE SUBSCHEMA APPLICATION CAN ACCESS
C
C     INVOKE CDCS AND OPEN FILES FOR PROCESSING
C
      SUBSCHEMA(EXAMP1)
      INVOKE
      OPEN(PFILE,MODE=0)
101     FORMAT (" DBSTAT= ", O4)
      WRITE 101, DBSTAT
      OPEN(EXMPT,MODE=0)
      WRITE 101, DBSTAT
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     DEFINE VALUES OF DATA ELEMENTS
C
      SOCSEC="123456789"
      FNAME="TOM"
      MI="X"
      LNAME(1)="JONES"
      LNAME(2)=" "
      PCODE="3000"
      TITLE(1)="PROGRAMMER"
      TITLE(2)=" "
      TSOCSEC="123456789"
      MSALARY="1750"
      VACHRS="80"
      SICKHRS="20"
      EMPDATE="010177"
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     WRITE VALUES TO DATABASE
C
      WRITE(PFILE)
      WRITE 101, DBSTAT
      WRITE(EXMPT)
      WRITE 101, DBSTAT
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     CLOSE FILES TO END CREATION MODE OF DATA BASE
C
      CLOSE(PFILE)
      WRITE 101, DBSTAT
      CLOSE(EXMPT)
      WRITE 101, DBSTAT
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     OPEN FILES AND BLANK FILL LAST NAME AND MONTHLY SALARY
C
      OPEN(PFILE)
      OPEN(EXMPT)
      LNAME(1)=10H
      MSALARY=10H
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     READ USING THE RELATION DEFINED IN THE SCHEMA/SUBSCHEMA
C
      READ(FIXUP1,KEY=SOCSEC)
      PRINT *,"READ BY RELATION RETURNS:"
      PRINT 100,LNAME,MSALARY
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     CHANGE THE VALUE OF MONTHLY SALARY AND REWRITE EXEMPT RECORD
      MSALARY="2000"
      REWRITE(EXMPT)
```

Figure 7-13. FORTRAN 4 Program for Payroll Example (Sheet 1 of 2)

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      READ AND PRINT THE VALUES TO VERIFY CHANGE TOOK PLACE
C
       READ(FIXUP1,KEY=SOCSEC)
       PRINT *,"READ BY RELATION AFTER REWRITE:"
       PRINT 100,LNAME,MSALARY
100    FORMAT(1X,3A10)
       CLOSE(EXMPT)
       CLOSE(PFILE)
       NNN=6LOUTPUT
       ENDFILE NNN
       TERMINATE
       STOP
       END
```

Program Output

```
DBSTAT= 0000
DBSTAT= 0000
DBSTAT= 0000
DBSTAT= 0000
DBSTAT= 0000
DBSTAT= 0000
READ BY RELATION RETURNS:
JONES                1750
READ BY RELATION AFTER REWRITE:
JONES                2000
```

Figure 7-13. FORTRAN 4 Program for Payroll Example (Sheet 2 of 2)

STANDARD CHARACTER SETS

CONTROL DATA operating systems offer the following variations of a basic character set:

CDC 64-character set

CDC 63-character set

ASCII 64-character set

ASCII 63-character set

The set in use at a particular installation was specified when the operating system was installed.

Depending on another installation option, the system assumes an input deck has been punched either in 026 or in 029 mode (regardless of the character set in use). Under NOS/BE, the alternate mode can be specified by a 26 or 29 punched in columns 79 or 80 of the job statement or any 7/8/9 card. The specified mode remains in effect through the end of the job unless it is reset by specification of the alternate mode on a subsequent 7/8/9 card.

Under NOS, the alternate mode can be specified by a 26 or 29 punched in columns 79 and 80 of any 6/7/9 card, as described above for a 7/8/9 card. In addition, 026 mode can be specified by a card with 5/7/9 multipunched in column 1, and 029 mode can be specified by a card with 5/7/9 multipunched in column 1, and a 9 punched in column 2.

Graphic character representation appearing at a terminal or printer depends on the installation character set and the terminal type. Characters shown in the CDC Graphic column of the standard character set table are applicable to BCD terminals; ASCII graphic characters are applicable to ASCII-CRT and ASCII-TTY terminals.

TABLE A-1. STANDARD CHARACTER SETS

Display Code (octal)	CDC			ASCII		
	Graphic	Hollerith Punch (026)	External BCD Code	Graphic Subset	Punch (029)	Code (octal)
00†	: (colon)††	8-2	00	: (colon) ††	8-2	072
01	A	12-1	61	A	12-1	101
02	B	12-2	62	B	12-2	102
03	C	12-3	63	C	12-3	103
04	D	12-4	64	D	12-4	104
05	E	12-5	65	E	12-5	105
06	F	12-6	66	F	12-6	106
07	G	12-7	67	G	12-7	107
10	H	12-8	70	H	12-8	110
11	I	12-9	71	I	12-9	111
12	J	11-1	41	J	11-1	112
13	K	11-2	42	K	11-2	113
14	L	11-3	43	L	11-3	114
15	M	11-4	44	M	11-4	115
16	N	11-5	45	N	11-5	116
17	O	11-6	46	O	11-6	117
20	P	11-7	47	P	11-7	120
21	Q	11-8	50	Q	11-8	121
22	R	11-9	51	R	11-9	122
23	S	0-2	22	S	0-2	123
24	T	0-3	23	T	0-3	124
25	U	0-4	24	U	0-4	125
26	V	0-5	25	V	0-5	126
27	W	0-6	26	W	0-6	127
30	X	0-7	27	X	0-7	130
31	Y	0-8	30	Y	0-8	131
32	Z	0-9	31	Z	0-9	132
33	0	0	12	0	0	060
34	1	1	01	1	1	061
35	2	2	02	2	2	062
36	3	3	03	3	3	063
37	4	4	04	4	4	064
40	5	5	05	5	5	065
41	6	6	06	6	6	066
42	7	7	07	7	7	067
43	8	8	10	8	8	070
44	9	9	11	9	9	071
45	+	12	60	+	12-8-6	053
46	-	11	40	-	11	055
47	*	11-8-4	54	*	11-8-4	052
50	/	0-1	21	/	0-1	057
51	(	0-8-4	34	(	12-8-5	050
52	)	12-8-4	74	)	11-8-5	051
53	$	11-8-3	53	$	11-8-3	044
54	=	8-3	13	=	8-6	075
55	blank	no punch	20	blank	no punch	040
56	, (comma)	0-8-3	33	, (comma)	0-8-3	054
57	. (period)	12-8-3	73	. (period)	12-8-3	056
60	≡	0-8-6	36	#	8-3	043
61	[	8-7	17	[	12-8-2	133
62	]	0-8-2	32	]	11-8-2	135
63	% ††	8-6	16	% ††	0-8-4	045
64	≠	8-4	14	" (quote)	8-7	042
65	←	0-8-5	35	_ (underline)	0-8-5	137
66	∨	11-0	52	!	12-8-7	041
67	∧	0-8-7	37	&	12	046
70	↑	11-8-5	55	' (apostrophe)	8-5	047
71	↓	11-8-6	56	?	0-8-7	077
72	<	12-0	72	<	12-8-4	074
73	>	11-8-7	57	>	0-8-6	076
74	≤	8-5	15	@	8-4	100
75	≥	12-8-5	75	\	0-8-2	134
76	¬	12-8-6	76	~ (circumflex)	11-8-7	136
77	; (semicolon)	12-8-7	77	; (semicolon)	11-8-6	073

†Twelve zero bits at the end of a 60-bit word in a zero byte record are an end-of-record mark rather than two colons.

††In installations using a 63-graphic set, display code 00 has no associated graphic or card code; display code 63 is the colon (8-2 punch). The % graphic and related card codes do not exist and translations yield a blank (55_8).

TABLE A-2. CDC CHARACTER SET COLLATING SEQUENCE

Collating Sequence Decimal	Collating Sequence Octal	CDC Graphic	Display Code	External BCD	Collating Sequence Decimal	Collating Sequence Octal	CDC Graphic	Display Code	External BCD
00	00	blank	55	20	32	40	H	10	70
01	01	≤	74	15	33	41	I	11	71
02	02	%	63 †	16 †	34	42	v	66	52
03	03	[	61	17	35	43	J	12	41
04	04	→	65	35	36	44	K	13	42
05	05	≡	60	36	37	45	L	14	43
06	06	^	67	37	38	46	M	15	44
07	07	↑	70	55	39	47	N	16	45
08	10	↓	71	56	40	50	O	17	46
09	11	>	73	57	41	51	P	20	47
10	12	≥	75	75	42	52	Q	21	50
11	13	¬	76	76	43	53	R	22	51
12	14	.	57	73	44	54	]	62	32
13	15	)	52	74	45	55	S	23	22
14	16	;	77	77	46	56	T	24	23
15	17	+	45	60	47	57	U	25	24
16	20	$	53	53	48	60	V	26	25
17	21	*	47	54	49	61	W	27	26
18	22	-	46	40	50	62	X	30	27
19	23	/	50	21	51	63	Y	31	30
20	24	,	56	33	52	64	Z	32	31
21	25	(	51	34	53	65	:	00 †	none†
22	26	=	54	13	54	66	0	33	12
23	27	≠	64	14	55	67	1	34	01
24	30	<	72	72	56	70	2	35	02
25	31	A	01	61	57	71	3	36	03
26	32	B	02	62	58	72	4	37	04
27	33	C	03	63	59	73	5	40	05
28	34	D	04	64	60	74	6	41	06
29	35	E	05	65	61	75	7	42	07
30	36	F	06	66	62	76	8	43	10
31	37	G	07	67	63	77	9	44	11

†In installations using the 63-graphic set, the % graphic does not exist. The : graphic is display code 63, External BCD code 16.

TABLE A-3. ASCII CHARACTER SET COLLATING SEQUENCE

Collating Sequence Decimal	Collating Sequence Octal	ASCII Graphic Subset	Display Code	ASCII Code	Collating Sequence Decimal	Collating Sequence Octal	ASCII Graphic Subset	Display Code	ASCII Code
00	00	blank	55	20	32	40	@	74	40
01	01	!	66	21	33	41	A	01	41
02	02	"	64	22	34	42	B	02	42
03	03	#	60	23	35	43	C	03	43
04	04	$	53	24	36	44	D	04	44
05	05	%	63†	25	37	45	E	05	45
06	06	&	67	26	38	46	F	06	46
07	07	'	70	27	39	47	G	07	47
08	10	(	51	28	40	50	H	10	48
09	11	)	52	29	41	51	I	11	49
10	12	*	47	2A	42	52	J	12	4A
11	13	+	45	2B	43	53	K	13	4B
12	14	,	56	2C	44	54	L	14	4C
13	15	-	46	2D	45	55	M	15	4D
14	16	.	57	2E	46	56	N	16	4E
15	17	/	50	2F	47	57	O	17	4F
16	20	0	33	30	48	60	P	20	50
17	21	1	34	31	49	61	Q	21	51
18	22	2	35	32	50	62	R	22	52
19	23	3	36	33	51	63	S	23	53
20	24	4	37	34	52	64	T	24	54
21	25	5	40	35	53	65	U	25	55
22	26	6	41	36	54	66	V	26	56
23	27	7	42	37	55	67	W	27	57
24	30	8	43	38	56	70	X	30	58
25	31	9	44	39	57	71	Y	31	59
26	32	:	00†	3A	58	72	Z	32	5A
27	33	;	77	3B	59	73	[	61	5B
28	34	<	72	3C	60	74	\	75	5C
29	35	=	54	3D	61	75	]	62	5D
30	36	>	73	3E	62	76	^	76	5E
31	37	?	71	3F	63	77	_	65	5F

†In installations using a 63-graphic set, the % graphic does not exist. The : graphic is display code 63.

Diagnostics are produced by the FORTRAN/DDL compiler and the DML preprocessor. These diagnostics are listed in the tables below.

FORTRAN/DDL DIAGNOSTICS

All diagnostics that can be issued during the compilation of a FORTRAN sub-schema are listed in table B-1. When a diagnostic message is printed on the source listing, it is preceded by a three-digit number enclosed in asterisks. The diagnostics are listed in order by this number.

Sub-schema library maintenance messages are listed in table B-2.

FORTRAN/DML DIAGNOSTICS

All diagnostics that can be issued by the DML preprocessor are listed in table B-3. These diagnostics are written to the file named by the E parameter of the DML control statement. Each diagnostic has one of the following severity codes associated with it:

T Trivial. The syntax of the usage is correct, but it is questionable.

W Warning. The syntax is incorrect, but the processor has been able to recover by making an assumption about what was intended.

F Fatal. An error which prevents DML from processing the statement in which it occurs. Unresolvable semantic errors also fall into this category. Processing continues with the next statement.

C Catastrophic. Compilation cannot continue; however, DML advances to the end of the current program unit and attempts to process the next program unit.

EXECUTION TIME DIAGNOSTICS

Execution time diagnostics are issued by CDCS 2; FORTRAN Data Base Facility has no execution diagnostics of its own. CDCS 2 diagnostics are listed in the CDCS 2 reference manual.

TABLE B-1. FORTRAN/DDL DIAGNOSTICS

Error Code	Message	Significance	Action
099	SOURCE WORD LONGER THAN 255 CHARACTERS, UNABLE TO CONTINUE COMPILATION - DDLF ABORTED	The compiler is unable to interpret the input stream.	Correct the error and recompile.
100	EMPTY INPUT FILE	The input file is empty and the compilation is terminated.	Create a new file and recompile.
107	INVALID NAME IN ALIAS STATEMENT	The specified name does not conform to the naming conventions.	Correct the error and recompile.
108	EQUAL SIGN MISSING	The syntax rules for this statement require an equal sign.	Correct the error and recompile.
111	INVALID QUALIFIER NAME	The specified qualifier name does not conform to the naming conventions.	Correct the error and recompile.
114	INVALID STATEMENT - REALM OR ALIAS STATEMENT EXPECTED	If ALIAS statements are used, they must immediately follow the SUBSCHEMA statement and precede any REALM statement. If ALIAS statements are not used, the REALM statements must immediately follow the SUBSCHEMA statement.	Correct the error and recompile.
115	INVALID REALM NAME	The specified name does not conform to the naming conventions.	Correct the error and recompile.

TABLE B-1. FORTRAN/DDL DIAGNOSTICS (Contd)

Error Code	Message	Significance	Action
116	DUPLICATE REALM NAME	All realm names must be unique.	Correct the error and recompile.
119	RECORD STATEMENT NOT SPECIFIED, UNABLE TO CONTINUE - COMPILATION ABORTED	The sub-schema must include at least one RECORD statement. Compilation is terminated.	Correct the error and recompile.
120	INVALID RECORD NAME	The specified record name does not conform to the naming conventions.	Correct the error and recompile.
121	DUPLICATE RECORD NAME	All record names must be unique.	Correct the error and recompile.
122	CANNOT LOCATE OWNER REALM IN THE SCHEMA	The schema area in which this record is defined is not specified in a sub-schema REALM statement.	Correct the error and recompile.
124	INVALID ITEM NAME	The specified item name does not conform to the naming conventions.	Correct the error and recompile.
125	ITEM NAME NOT UNIQUE	All item names must be unique.	Correct the error and recompile.
126	ITEM SIZE GREATER THAN MAXIMUM SIZE ALLOWED	The maximum size allowed for a CHARACTER item is 32 767 characters.	Correct the error and recompile.
127	INVALID ITEM LENGTH	The length for a CHARACTER item must be specified in the form *len, where len is a positive integer.	Correct the error and recompile.
128	LENGTH SPECIFIED FOR NON-CHARACTER ITEM	The specification for length, *len (where len is an integer), is allowed only for type CHARACTER items.	Correct the error and recompile.
130	INVALID DIMENSION BOUND	The specified dimension value is not an integer.	Correct the error and recompile.
131	THE UPPER DIMENSION BOUND IS LESS THAN THE LOWER BOUND	When the user specifies a dimension of an array, the value of the upper bound must be greater than or equal to the value of the lower bound. See the description of the type statement for more information.	Correct the error and recompile.
133	EXCEEDED THE MAXIMUM NUMBER OF DIMENSIONS ALLOWED	In a sub-schema for a FORTRAN 4 program, an array can have a maximum of three dimensions; in a sub-schema for a FORTRAN 5 program, an array can have a maximum of seven dimensions.	Correct the error and recompile.
138	DOUBLE PRECISION MUST BE SPECIFIED IN FORTRAN 5	The keyword PRECISION must be specified in the type statement for a double precision item when the F5 parameter is specified in the DDLF control statement.	Correct the error and recompile.

TABLE B-1. FORTRAN/DDL DIAGNOSTICS (Contd)

Error Code	Message	Significance	Action
142	ALIAS ENTRY TYPE NOT SPECIFIED	Alias entry type REALM, RECORD, or ITEM was not specified; the current entry is ignored.	Specify entry type and recompile.
143	AN ITEM IS THE ONLY ALIAS TYPE THAT CAN BE QUALIFIED	Record and realm names must be unique; therefore, there is no need to qualify them.	Correct the error and recompile.
144	UNKNOWN QUALIFIER NAME	The qualifier entry could not be located; the ALIAS statement is ignored.	Correct the error and recompile.
145	parm INVALID IN THE FOLLOWING STATEMENT	The indicated parameter is invalid.	Correct the error and recompile.
146	SUBSCHEMA STATEMENT NOT SPECIFIED - COMPILATION ABORTED	The SUBSCHEMA statement is required in every FORTRAN/DDL source program. Compilation is terminated.	Correct the error and recompile.
147	SCHEMA KEYWORD MISSING	The keyword SCHEMA is required in the SUBSCHEMA statement.	Correct the error and recompile.
148	INVALID SCHEMA OR SUBSCHEMA NAME	The specified schema or subschema name does not conform to the naming conventions.	Correct the error and recompile.
155	SCHEMA NAME SPECIFIED IN THE SUB-SCHEMA DOES NOT MATCH THE SCHEMA NAME IN THE SCHEMA	The schema name specified in the SUBSCHEMA statement must be the same as the name specified in the schema declaration in the schema.	Correct the error and recompile.
166	INVALID SUBSCRIPT IN CLASS MATRIX	Internal to DDL.	See the systems analyst.
169	EMPTY SCHEMA DIRECTORY	No information in the schema directory.	Recompile the schema.
171	INSUFFICIENT FIELD LENGTH - INCREASE YOUR FL --- DDL ABORTED	Not enough field length was specified to complete the compilation. The job is aborted.	Use the RFL control statement to increase field length.
179	UNABLE TO COPY AREA ENTRIES FROM THE SCHEMA, SCHEMA IN INVALID	The specified schema did not contain valid data, and area entries could not be read.	Correct and recompile the schema.
200	RECORD STRUCTURE CAUSES MAXIMUM NUMBER OF COMMON BLOCKS TO BE EXCEEDED	The maximum number of common blocks allowed is 500. FORTRAN/DDL generates common blocks sequentially according to the subschema source program: one common block is generated for each area; an additional common block is generated each time FORTRAN/DDL encounters a data type incompatible with the previous data type. (Character data is incompatible with all other data types.)	Group data items of the same type together within a record to minimize the number of common blocks generated, and recompile.
201	AN ALIASED NAME MUST BE REFERENCED BY THE ALIAS-NAME SPECIFIED IN THE ALIAS STATEMENT	When an alias is assigned, the alias and not the schema-assigned name must be referenced.	Correct the error and recompile.

TABLE B-1. FORTRAN/DDL DIAGNOSTICS (Contd)

Error Code	Message	Significance	Action
202	MAXIMUM AREA COUNT EXCEEDED	A maximum of 4095 areas is allowed.	Correct the error and recompile.
203	MAXIMUM RECORD COUNT EXCEEDED	A maximum of 4095 records is allowed.	Correct the error and recompile.
204	MAXIMUM ITEM COUNT EXCEEDED	A maximum of 4095 items can be declared.	Correct the error and recompile.
205	MAXIMUM ITEM COUNT FOR A RECORD EXCEEDED	A maximum of 4095 items per record is allowed.	Correct the error and recompile.
206	FATAL ERRORS OCCURRED -- FOLLOWING STATEMENTS IGNORED	Due to the occurrence of one or more fatal errors, subsequent statements are not processed by the compiler.	Correct the error and recompile.
210	A RECORD HAS BEEN PREVIOUSLY DEFINED FOR THIS REALM	Only one record can be defined for a realm.	Correct the error and recompile.
211	NO DATA ITEMS DEFINED FOR THIS RECORD	At least one data item must be defined for each record.	Correct the error and recompile.
212	INVALID STATEMENT - END STATEMENT EXPECTED	Only an END statement is allowed at this position in the input stream.	Correct the error and recompile.
213	INVALID STATEMENT FOLLOWING END STATEMENT	A SUBSCHEMA statement is the only statement that can follow an END statement.	Correct the error and recompile.
214	NO END STATEMENT	The last statement in a FORTRAN/ DDL program must be an END statement.	Correct the error and recompile.
215	COMMA OR EQUAL MISSING	The syntax rules for this statement require a comma or equal sign.	Correct the error and recompile.
216	STATEMENT CONTAINS EXTRANEOUS INFORMATION	Data was found beyond the end of the statement.	Correct the error and recompile.
217	INTERNAL DDLF ERROR	An internal DDL error occurred.	See the systems analyst.
218	TYPE CHARACTER IS NOT VALID IN FORTRAN 4	An item cannot be defined as type CHARACTER if the F4 parameter is specified on the DDLF control statement.	Change usage to one permitted in FORTRAN 4.
219	THE LOW:HIGH BOUNDS FEATURE IS NOT VALID IN FORTRAN 4	The low:high bounds feature is allowed only in FORTRAN 5. In FORTRAN 4, the low bound of an array dimension is assumed to be 1; the user specifies only the high bound.	Correct the error and recompile.
220	TYPE BOOLEAN IS NOT VALID IN FORTRAN 4	An item cannot be described as type BOOLEAN if F4 is specified on the DDLF control statement. Type BOOLEAN is allowed only in FORTRAN 5.	Correct the error and recompile.
300	RECORD parm NOT FOUND IN SCHEMA	The record named is not defined in the schema.	Correct the error and recompile.

TABLE B-1. FORTRAN/DDL DIAGNOSTICS (Contd)

Error Code	Message	Significance	Action
301	MAXIMUM ARRAY ELEMENTS IN SUB-SCHEMA GREATER THAN IN SCHEMA	The dimension value specified in the sub-schema must be less than or equal to the maximum occurs value in the schema.	Correct the error and recompile.
302	SUBSCHEMA ITEM DOES NOT APPEAR IN SCHEMA	The item specified in the sub-schema is not defined in the schema.	Correct the error and recompile.
304	KEY ITEM FOR REALM parm NOT FOUND IN SUBSCHEMA	The primary key must be defined for every realm in the sub-schema.	Correct the error and recompile.
306	TYPE OF REPEATING ITEM DOES NOT AGREE WITH TYPE OF REPEATING ITEM IN SCHEMA	An array must correspond to a vector in the schema. A variable must correspond to a nonrepeated data item in the schema.	Correct the error and recompile.
307	ILLEGAL ITEM CONVERSION	Data type specified in the schema cannot be converted to data type specified in the sub-schema.	Correct the error and recompile.
308	REALM parm NOT FOUND IN SCHEMA	The realm name specified in the sub-schema cannot be located in the schema.	Correct the error and recompile.
310	INSUFFICIENT FL FOR DDLF PASS 2	Not enough field length was specified to complete the compilation. The job is aborted.	Use the RFL control statement to increase field length.
312	HIERARCHY OF SUB-SCHEMA ITEM DIFFERS FROM CORRESPONDING ITEM IN SCHEMA	A sub-schema item corresponds to a schema item that is part of a repeating group. Items in a repeating group cannot be referenced in FORTRAN/DDL.	Convert schema repeating group into vectors or omit item from sub-schema.
313	NO RECORD ENTRIES WERE SPECIFIED FOR REALM parm	At least one record statement must be specified for each realm specified in the sub-schema.	Correct the error and recompile.
314	WARNING: SS SIZE GR SCHEMA SIZE - MAY CAUSE TRUNCATION ERRORS AT EXECUTION TIME	An execution error can result from nonzero or nonblank truncation.	Increase schema size of item, or ensure that the data is within the schema-defined limit.
315	SUBSCHEMA ITEM CANNOT DIFFER FROM SCHEMA ITEM WITH CHECK IS PICTURE OPTION	The CHECK IS PICTURE option in the schema disallows data conversion.	Refer to the CHECK IS PICTURE option for type requirements.
401	INVALID RELATION NAME	The specified relation name does not conform to the naming conventions.	Correct the error and recompile.
402	RELATION NAME NOT UNIQUE	The relation name must be unique among all relation and realm names in the sub-schema.	Correct the error and recompile.
403	UNABLE TO FIND CORRESPONDING RELATION ENTRY IN THE SCHEMA	The relation name specified in the sub-schema does not appear in the schema.	Correct the error and recompile.
404	AREA parm TRAVERSED IN THE SCHEMA MUST BE SPECIFIED IN A REALM STATEMENT	The indicated area is joined in a relationship within the schema, but is not specified in a REALM statement.	Correct the error and recompile.

TABLE B-1. FORTRAN/DDL DIAGNOSTICS (Contd)

Error Code	Message	Significance	Action
405	INVALID RECORD NAME	The record name specified in a RESTRICT statement does not conform to the naming conventions.	Correct the error and recompile.
406	UNABLE TO FIND RECORD ENTRY	A record referenced in a RESTRICT statement does not appear in the sub-schema.	Correct the error and recompile.
407	parm IS AN INVALID DATA-BASE-IDENTIFIER NAME	The indicated parameter is not a valid data base item in a RESTRICT statement.	Correct the error and recompile.
408	DATA-BASE-IDENTIFIER parm IS UNDEFINED	The indicated data base item in a RESTRICT statement does not appear in the sub-schema.	Correct the error and recompile.
409	DATA-BASE-IDENTIFIER parm IS UNDEFINED IN THE SCHEMA	The indicated identifier parameter does not appear in the schema.	Correct the error and recompile.
410	CONVERSION NOT POSSIBLE FOR LITERAL	Arithmetic operations cannot be performed on a literal.	Correct the error and recompile.
411	INVALID USE OF DBI parm - DOES NOT BELONG TO RESTRICTED RECORD	The indicated data base item does not apply to the record referenced in a RESTRICT statement.	Correct the error and recompile.
413	SOURCE DBI DOES NOT HAVE A COMPATIBLE SCHEMA DATA REPRESENTATION WITH THE TARGET DBI	Source and target data base items used in a RESTRICT statement must be compatible as determined from the schema data class of each item. Items of schema data classes 0 through 4 (items stored as display code) are compatible. An item of any other schema data class is compatible only with an item of the same schema data class.	Correct the error and recompile.
417	INVALID STATEMENT - END STATEMENT EXPECTED	Only an END statement is allowed at this position in the input stream.	Correct the error and recompile.
418	PERIOD MISSING	A period was expected in this statement and was not found.	Correct the error and recompile.
419	EXTRANEOUS DATA IN STATEMENT	Data was found beyond the end of the statement.	Correct the error and recompile.
420	NUMBER OF DIMENSIONS SPECIFIED DOES NOT EQUAL THE NUMBER OF DIMENSIONS DEFINED FOR ARRAY parm	The number of dimensions specified for the item in the RESTRICT clause does not conform to the sub-schema description of the item.	Correct the error and recompile.
421	SUBSCRIPT SPECIFIED FOR ITEM WHICH IS NOT AN ARRAY	A subscript is specified for an item that is not defined as an array in the sub-schema.	Correct the error and recompile.
422	DUE TO FATAL ERRORS, COMPILATION IS ABORTED	Too many fatal errors have occurred.	Correct as many errors as possible and recompile.
426	OWNER AREA OF THE RESTRICTED RECORD WAS NOT SPECIFIED IN THE RELATION ENTRY (IN THE SCHEMA)	The area associated with a record named in a RESTRICT statement is not referenced in the schema relation entry.	Correct the error and recompile.

TABLE B-1. FORTRAN/DDL DIAGNOSTICS (Contd)

Error Code	Message	Significance	Action
427	INVALID OR MISSING RELATIONAL OPERATOR	A relational operator is missing or is not one of the six valid operators.	Correct the error and recompile.
428	EXCEEDED FIELD LENGTH, INCREASE YOUR FIELD LENGTH	Not enough field length was specified to complete the compilation. The job is aborted.	Use the RFL control statement to increase field length.
429	RECORD ENTRY parm WAS PREVIOUSLY SPECIFIED IN A RESTRICT CLAUSE	Only one RESTRICT statement can be included for a given record. The indicated parameter is not allowed.	Correct the error and recompile.
430	INVALID LOGICAL OPERATOR	A logical operator is missing or is not one of the allowed operators.	Correct the error and recompile.
431	NO END STATEMENT	The last statement in a FORTRAN/ DDL must be an END statement.	Correct the error and recompile.
432	INVALID STATEMENT FOLLOWING END STATEMENT	A SUBSCHEMA statement is the only statement that can follow an END statement.	Correct the error and recompile.
433	SKIPPING TO THE NEXT RESTRICT STATEMENT	An invalid RESTRICT statement causes the system to scan for the next RESTRICT statement.	Correct the error and recompile.
434	RESTRICT STATEMENT FOUND	The scan following an invalid RESTRICT statement located another RESTRICT statement.	Correct the error and recompile.
435	COULD NOT FIND ANY MORE RESTRICT STATEMENTS FOR THE CURRENT RELATION ENTRY	The scan following an invalid RESTRICT statement did not locate another RESTRICT statement.	Correct the error and recompile.
436	parm IS AN INVALID SUBSCRIPT	Subscripts must be positive integer constants. The indicated parameter is not valid.	Correct the error and recompile.
438	TERMINATING DELIMITER FOR SUBSCRIPT IS MISSING	The closing parenthesis of a subscript is missing.	Correct the error and recompile.
439	THE SUBSCRIPT VALUE IS NOT WITHIN THE BOUNDS OF ARRAY DECLARATION	The subscript specified is greater than the declared upper bound or less than the declared lower bound.	Correct the error and recompile.
440	DID NOT SPECIFY SUBSCRIPT FOR ARRAY parm	The indicated parameter requires subscripting.	Correct the error and recompile.
441	MAXIMUM RELATION COUNT EXCEEDED	A maximum of 4095 relations can be declared.	Correct the error and recompile.
442	LEFT PARENTHESIS MISSING	Left-right parentheses must match.	Correct the error and recompile.
443	RIGHT PARENTHESIS MISSING	Left-right parentheses must match.	Correct the error and recompile.

TABLE B-2. SUB-SCHEMA LIBRARY MAINTENANCE MESSAGES

Message	Significance	Action
DID NOT LOCATE parm, PURGE NOT POSSIBLE	The sub-schema name specified for the purge could not be located in the sub-schema library.	Specify the correct sub-schema name and recompile.
DID NOT LOCATE SUB-SCHEMA TO BE REPLACED --- NEW SUB-SCHEMA HAS BEEN ADDED	The sub-schema to be replaced in the library could not be located; the current sub-schema has been added to the library.	Specify correct library or sub-schema and recompile.
ILL-FORMATTED LIBRARY -- NOT UPDATABLE, DDL ABORTED	The sub-schema library contains an error and cannot be updated. DDL terminates.	Check for empty library or file that is not a library. Correct the error and recompile.
SUB-SCHEMA WITH THE SAME NAME AS THE NEW SUB-SCHEMA ALREADY EXISTS --- FILE NOT UPDATED	The current sub-schema could not be added to the library since the library contains a sub-schema with the same name.	Change the sub-schema name and recompile.
WARNING --- EMPTY SUB-SCHEMA FILE	The sub-schema library contains no sub-schemas. The sub-schemas might have been purged prior to this compilation. The new sub-schema is added to the library and is the only sub-schema in the library.	If sub-schemas have been purged, create a new library.
EMPTY SUB-SCHEMA FILE, DDLF ABORTED	The sub-schema library contains no sub-schemas; issued during a sub-schema compaction, audit, or purge. The sub-schemas might have been purged prior to this compilation.	If the sub-schemas have been purged, create a new library.
OLD SUB-SCHEMA FILE BAD, SUB-SCHEMA LENGTH IS ZERO. DDLF ABORTED	The sub-schema library contains a sub-schema of length zero, indicating a bad library file; issued during a sub-schema compaction. DDLF terminates.	Re-create the library.
EMPTY INPUT FILE --- PURGE NOT POSSIBLE	The input file contains no sub-schemas. The input file might have been purged prior to this operation.	If the input file is in error, create a new file.

TABLE B-3. FORTRAN/DML DIAGNOSTICS

Error Code	Message	Severity	Significance	Action
100	EMPTY INPUT FILE	C	The input file is empty and processing is terminated.	Create a new file and resubmit.
101	LEFT PARENTHESIS MISSING	F	A left parenthesis was expected but not found.	Correct the error and resubmit.
102	INVALID SUBSCHEMA NAME	F	The specified sub-schema name does not conform to naming conventions.	Correct the error and resubmit.
103	RIGHT PARENTHESIS MISSING	F	A right parenthesis was expected but not found.	Correct the error and resubmit.
104	EXTRANEOUS DATA FOLLOWING RIGHT PARENTHESIS	F	Data was found beyond the end of a statement.	Correct the error and resubmit.

TABLE B-3. FORTRAN/DML DIAGNOSTICS (Contd)

Error Code	Message	Severity	Significance	Action
105	NO INVOKE STATEMENT WAS FOUND IN PRECEDING PROGRAM UNIT	F	An INVOKE statement must be included in each program unit that contains DML statements.	Correct the error and resubmit.
107	MODE=O INVALID ON OPEN RELATION	F	The mode option must be I or IO for an OPEN relation.	Correct the error and resubmit.
108	UNKNOWN SOURCE WORD	F	The DML preprocessor is unable to interpret the statement.	Correct the error and resubmit.
109	INVALID MODE	F	The mode must be I, IO, or O.	Correct the error and resubmit.
110	EQUAL SIGN MISSING	F	An equal sign was expected but not found.	Correct the error and resubmit.
111	SUBSCHEMA NOT AVAILABLE	F	The sub-schema could not be found in the specified library.	Specify the correct library and resubmit.
112	EXTRANEOUS DATA IN PARAMETER LIST	F	DML does not recognize the item following the equal sign in a keyword specification such as END=, ERR=, KEY=, or MODE=.	Correct the error and resubmit.
113	KEY PARAMETER MISSING	F	The keyword KEY was expected but not found.	Correct the error and resubmit.
114	INVALID RELATIONAL OPERATOR	F	The relational operator is missing or is invalid.	Correct the error and resubmit.
115	INVALID ITEM NAME	F	The item name does not conform to naming conventions.	Correct the error and resubmit.
116	ITEM IS NOT DEFINED AS A KEY FOR THIS REALM	F	The item is not defined as a key in the schema.	Specify the correct key or update the schema.
117	PERIOD MISSING	F	A period was expected but not found.	Correct the error and resubmit.
118	INTERNAL DML ERROR	C	An error internal to DML occurred.	Contact the systems analyst.
119	COMMA OR RIGHT PARENTHESIS NOT FOUND	F	A comma or right parenthesis was expected but not found.	Correct the error and resubmit.
121	PRIVACY TYPE MUST BE LITERAL OR ARRAY NAME	F	An invalid privacy type was specified.	Correct the error and resubmit.
122	INVALID PARAMETER	F	The specified parameter is not valid in this statement.	Correct the error and resubmit.
123	DUPLICATE PARAMETER	F	The same parameter cannot be specified more than once.	Correct the error and resubmit.
124	PRIVACY PARAMETER LITERAL IS GREATER THAN 30 CHARACTERS	F	The privacy literal must be 30 or fewer characters.	Correct the error and resubmit.
125	PRIVACY PARAMETER NOT SPECIFIED	F	The privacy parameter is required.	Correct the error and resubmit.
126	INSUFFICIENT FIELD LENGTH - DML ABORTED	C	Not enough field length was specified to complete preprocessing. The job is aborted.	Use the RFL control statement to increase field length.

Error Code	Message	Severity	Significance	Action
127	DML LANGUAGE VERSION (LV) DIFFERS FROM SUB-SCHEMA	F	A sub-schema and a FORTRAN/DML applications program must specify the same version of FORTRAN. For example, when a sub-schema is compiled with F4 specified in the DDLF control statement, applications programs that use that sub-schema must specify LV=F4 in the DML control statement. Similarly, for FORTRAN 5 applications, F5 must be substituted for F4 in the example above.	Correct the error and resubmit.
128	INVALID FORTRAN LABEL	F	The END=s and ERR=s parameters must specify a valid statement label (any 1- through 5-digit positive nonzero integer).	Correct the error and resubmit.
129	END= IS NOT VALID IN FORTRAN 4	F	The parameter END= is allowed only in FORTRAN 5.	Correct the error and resubmit.
130	ERR= IS NOT VALID IN FORTRAN 4	F	The parameter ERR= is allowed only in FORTRAN 5.	Correct the error and resubmit.

Access Control -
Protection of data from unauthorized access or modification; called privacy in the FORTRAN Data Base Facility.

Actual Key -
A file organization in which records are stored according to their key values.

Advanced Access Methods (AAM) -
A file manager that processes indexed sequential, direct access, and actual key file organizations and supports the Multiple-Index Processor. See CYBER Record Manager.

Alias -
A data name used in the sub-schema in place of a schema data name.

Alternate Key -
A data item for which the value can be used to randomly access a record in a CRM file.

Area -
A uniquely named schema data base subdivision that contains data records; identified in the sub-schema as a realm; a file.

Array -
A data item consisting of a set of elements of the same type that is identified by a single name; FORTRAN sub-schema data structure. See Elementary Item.

Attach -
The process of making a permanent file accessible to a job by specifying the proper permanent file identification and passwords.

Basic Access Methods (BAM) -
A file manager that processes sequential and word addressable file organizations. See CYBER Record Manager.

Beginning-of-Information (BOI) -
As defined by CRM, the start of the first user record in a file. System-supplied information, such as an index block or control word, does not affect beginning-of-information. Any label on a tape exists prior to beginning-of-information.

Block -
On tape, information between interrecord gaps on a tape. CRM defines several blocks depending on file organization as shown in table C-1.

Checksum -
A one-word sum generated by DDL for each area and relation in a schema and for each sub-schema. Checksums are stored in the schema and sub-schema directories, and in the master directory. CDCS references them to check the validity of using a previously compiled sub-schema with the current schema or of using a previously compiled applications program with a current sub-schema.

TABLE C-1. BLOCK TYPES

Organization	Blocks
Indexed sequential	Data block; index block
Direct access	Home block; overflow block
Actual key	Data block
Sequential	Block type I, C, K, E

Child Record Occurrence -
For relation processing, a record occurrence that has another record occurrence (the parent record occurrence) at the next lower rank in a hierarchical tree structure of the relation.

Concurrency -
Simultaneous access to the same data in a data base by two or more applications programs during a given span of time.

Constant -
A fixed value, explicitly written in a source statement. In a FORTRAN sub-schema, the term corresponds to a literal in the schema.

Control Break -
A condition occurring during a relation read; the condition signifies that a new record occurrence was read for the parent file.

Control Word -
A system-supplied word that precedes each W type record in storage.

Conversion -
The process of changing data characteristics between the schema and the sub-schema.

CYBER Database Control System (CDCS) -
The controlling module that provides the interface between the applications program and the data base.

CYBER Record Manager (CRM) -
A generic term relating to the common products BAM and AAM, which run under the NOS and NOS/BE operating systems and allow a variety of record types, blocking types, and file organizations to be created and accessed. The execution time input/output of COBOL, FORTRAN, Sort/Merge 4, ALGOL, and the DMS-170 products is implemented through CRM. Neither the input/output of the NOS and NOS/BE operating systems nor any of the system utilities such as COPY or SKIPF are implemented through CRM. All CRM file processing requests ultimately pass through the operating system input/output routines.

Data Administrator -
A person who defines the format and organization of the data base.

Data Base -
A systematically organized, central pool of information; organization is described by a schema.

Data Base Procedure -
A special-purpose routine that performs a predefined operation, specified in the schema and initiated by CDCS.

Data Description Language (DDL) -
The language used to structure the schema and the sub-schema.

Data Item -
A unit of data within a record; can be an elementary or group data item in the schema. A data item can be a variable or array in the FORTRAN sub-schema.

Data Manipulation Language (DML) -
The extensions to FORTRAN that provide access to a data base.

Deadlock -
A situation that arises in concurrent data base access when two or more applications programs are contending for a resource that is locked by one of the other applications programs, and none of the programs can proceed without that resource.

Direct Access -
In the context of CRM, one of the five file organizations. It is characterized by the system hashing of the unique key within each file record to distribute records randomly in blocks called home blocks of the file.

In the context of NOS permanent files, a direct access file is a file that is accessed and modified directly.

Directory -
A file that contains area and record attributes of the data base; created when the schema is compiled; an object schema.

Elementary Item -
A data item that is not subdivided into other data items in the schema data structure; an elementary item that is part of a group item has the highest level number in the group item. A nonrepeating elementary item in the schema corresponds to a variable in the FORTRAN sub-schema; a repeating elementary item corresponds to an array.

End-of-Information (EOI) -
Defined by CRM in terms of the file organization and file residence as shown in table C-2.

File -
A collection of records treated as a unit; an area in the schema; a realm in the sub-schema.

File Information Table (FIT) -
A table through which CDCS communicates with CRM.

Fixed Occurrence Data Item -
A data item that is repeated the same number of times in all records in the schema data structure.

TABLE C-2. END-OF-INFORMATION BOUNDARIES

File Organization	File Residence	Physical Position
Sequential	Mass storage	After the last user record.
	Labeled tape in SI, I, S, or L format	After the last user record and before any file trailer labels.
	Unlabeled tape in SI or I format	After the last user record and before any file trailer labels.
	Unlabeled tape in S or L format	Undefined.
Word Addressable	Mass storage	After the last word allocated to the file, which might be beyond the last user record.
Indexed Sequential, Actual Key	Mass storage	After the record with the highest key value.
Direct Access	Mass storage	After the last record in the most recently created overflow block or home block with the highest relative address.

Group Item -
A data item that is subdivided into other data items; a collection of data items in the schema data structure. Group items and elementary items within a group are not referenced in the FORTRAN sub-schema.

Hierarchical Tree Structure -
A representation that commonly illustrates record occurrences for files joined in a directed relation. The root of the tree is a record occurrence in the root file and each successive level represents the record occurrences in each joined file.

Home Block -
Mass storage allocated for a file with direct access organization at the time the file is created.

Indexed Sequential -
A file organization in which records are stored in ascending order by key.

Keyword -
A word that is required in a DDL source program statement.

Level -
For system-logical-records, an octal number 0 through 17 in the system-supplied 48-bit marker that terminates a short or zero-length PRU.

Level Number -
A number defining the structure of data within a record in the schema.

Literal -
A constant completely defined by its own identity in the schema.

Logging -
The facility of CDCS through which historical records are kept of operations performed by users on data base areas. Logging information is used in data base recovery and restoration operations.

Logical File Name (lfn) -
The one to seven display code alphabetic or numeric characters by which the operating system recognizes a file. Every lfn in a job must be unique and must begin with a letter.

Logical Record -
Under NOS, a data grouping that consists of one or more PRUs terminated by a short PRU or zero-length PRU. Equivalent to a system-logical-record under NOS/BE.

Nested Group Item -
A group item that is subordinate to another group item in the schema data structure.

Noise Record -
The number of characters the tape drivers discard as being extraneous noise rather than a valid record. The value depends on installation settings.

Null Record Occurrence -
A record occurrence composed of the display code right bracket symbol in each character position. It is used in a relation occurrence to denote that no record occurrence qualifies or that a record occurrence does not exist at a given level in the relation.

Operation -
A particular function performed on units of data; for instance, opening or closing an area, or storing or deleting a record.

Overflow Block -
Mass storage the system adds to a file with direct access organization when records cannot be accommodated in the home block.

Parent Record Occurrence -
For relation processing, a record occurrence that has another record occurrence (the child record occurence) at the next higher rank in a hierarchical tree structure of the relation.

Partition -
As defined by CRM, a division within a file with sequential organization. Generally, a partition contains several records or sections. Implementation of a partition boundary is affected by file structure and residence, as shown in table C-3.

Notice that in a file with W type records, a short PRU of level 0 terminates both a section and a partition.

TABLE C-3. PARTITION BOUNDARIES

Device	Record Type (RT)	Block Type (BT)	Physical Boundary
PRU device	W	I	A short PRU of level 0 containing a one-word deleted record pointing back to the last I block boundary, followed by a control word with a flag indicating a partition boundary.
	W	C	A short PRU of level 0 containing a control word with a flag indicating a partition boundary.
	D,F,R, T,U,Z	C	A short PRU of level 0 followed by a zero-length PRU of level 17_8.
	S	-	A zero-length PRU of level number 17_8.
S or L format tape	W	I	A separate tape block containing as many deleted records of record length 0 as required to exceed noise record size, followed by a deleted one-word record pointing back to the last I block boundary, followed by a control word with a flag indicating a partition boundary.
	W	C	A separate tape block containing as many deleted records of record length 0 as required to exceed noise record size, followed by a control word with a flag indicating a partition boundary.
	D,F,T, R,U,Z	C,K,E	A tapemark.
	S	-	A tapemark.
Any other tape format	-	-	Undefined.

Permanent File -
A file on a mass storage permanent file device that is protected against accidental destruction by the system and can be protected against unauthorized access or destruction.

Physical Record Unit (PRU) -
Under NOS and NOS/BE, the amount of information transmitted by a single physical operation of a specified device, as shown in table C-4.

TABLE C-4. PRU SIZES

Device	Size in Number of 60-Bit Words
Mass storage (NOS and NOS/BE only).	64
Tape in SI format with coded data (NOS/BE only).	128
Tape in SI format with binary data.	512
Tape in I format (NOS only).	512
Tape in any other format.	Undefined.

A PRU that is not full of user data is called a short PRU; a PRU that has a level terminator but no user data is called a zero-length PRU.

Primary Key -
A key that must be defined for random access of a record in an indexed sequential, direct access, or actual key file.

PRU Device -
Under NOS and NOS/BE, a mass storage device or a tape in SI or I format, so called because records on these devices are written in PRUs.

Random File -
In the context of CRM, a file with word addressable, indexed sequential, direct access, or actual key organization in which individual records can be accessed by the values of their keys; in the context of the NOS and NOS/BE operating systems, a file with the random bit set in the file environment table in which individual records are accessed by their relative PRU numbers.

Rank -
The rank of a file in a DMS-170 relation corresponds to the position of the file in the schema definition of the relation. The ranks of the files joined in a relation are numbered consecutively, with the root file having a rank of 1.

Realm -
A uniquely named sub-schema data base subdivision that contains data records; identified in the schema as an area; a file.

Realm Ordinal -
A unique identifier assigned to each realm in a sub-schema when the sub-schema is compiled. Realm ordinals for a FORTRAN sub-schema are used in conjunction with the FORTRAN status variables.

Record -
As defined by CRM, a group of related characters. A record or a portion thereof is the smallest collection of information passed between CRM and a user program. Eight different record types exist, as defined by the RT field of the file information table.

For processing through the FORTRAN Data Base Facility, a record is equivalent to a record occurrence.

Other parts of the operating systems and their products might have additional or different definitions of records.

Record Occurrence -
An actual data base record that conforms to a record description in the schema.

Record Type -
A term that can have one of several meanings, depending on the context. CRM defines eight record types established by an RT field in the file information table. Tables output by the loader are classified as record types such as text, relocatable, or absolute, depending on the first few words of the tables.

In FORTRAN/DDL, the description of the attributes of a record; record layout. The name of a record type is given in the RECORD statement.

Relation -
The logical structure formed by the joining of files for the purpose of allowing an applications program to retrieve data from more than one file at the same time. The structure is declared in the schema and is based on common identifiers in the files.

Relation Occurrence -
The logical concatenation of a record occurrence from each record type specified in the relation.

Repeating Group -
A collection of data items that occurs a number of times within a record; can consist of elementary items, group items, and vectors.

Root Realm -
The realm that ranks lowest in a relation; its record occurrences are pictured as the root of a tree in a hierarchical tree structure.

Schema -
A detailed description of the internal structure of the complete data base.

Section -
As defined by CRM, a division within a file with sequential organization. Generally, a section contains more than one record and is a division within a partition of a file. A section terminates with a physical representation of a section boundary, as shown in table C-5.

The NOS and NOS/BE operating systems equate a section with a system-logical-record of level 0 through 16$_8$.

Sequential -
A file organization in which records are stored in the order in which they are generated.

TABLE C-5. SECTION BOUNDARIES

Device	Record Type (RT)	Block Type (BT)	Physical Representation
PRU device	W	I	A deleted one-word record pointing back to the last I block boundary followed by a control word with flags indicating a section boundary. At least the control word is in a short PRU of level 0.
	W	C	A control word with flags indicating a section boundary. The control word is in a short PRU of level 0.
	D,F,R, T,U,Z	C	A short PRU with a level less than 17_8.
	S	-	Undefined.
S or L format tape	W	I	A separate tape block containing as many deleted records of record length 0 as required to exceed noise record size, followed by a deleted one-word record pointing back to the last I block boundary, followed by a control word with flags indicating a section boundary.
	W	C	A separate tape block containing as many deleted records of record length 0 as required to exceed noise record size, followed by a control word with flags indicating a section boundary.
	D,F,R, T,U,Z	C,K,E	Undefined.
	S	-	Undefined.
Any other tape format	-	-	Undefined.

Short PRU -
A PRU that does not contain as much user data as the PRU can hold and is terminated by a system terminator with a level number.

Under NOS, a short PRU defines EOR; under NOS/BE, a short PRU defines the end of a system-logical-record. In the CRM context, a short PRU can have several interpretations depending on the record and blocking types.

Sub-Schema -
A detailed description of the portion of the data base to be made available to one or more FORTRAN applications programs.

Sub-Schema Item Ordinal -
A unique identifier within a record assigned to each item in a sub-schema when the sub-schema is compiled. Sub-schema item ordinals are used in conjunction with the data base status block.

Sub-Schema Library -
A permanent file containing one or more sub-schemas.

System-Logical-Record -
Under NOS/BE, a data grouping that consists of one or more PRUs terminated by a short PRU or zero-length PRU. These records can be transferred between devices without loss of structure.

Equivalent to a logical record under NOS.

Equivalent to a CRM S type record.

Type -
The storage format of a data item which determines permitted values, length, and arithmetic meaning.

Variable -
A single named data item in the FORTRAN sub-schema data structure.

W Type Record -
One of the eight record types supported by CRM. Such records appear in storage preceded by a system-supplied control word. The existence of the control word allows files with sequential organization to have both partition and section boundaries.

Word Addressable -
A word addressable file is a mass storage file containing continuous data or space for data. Words within a word addressable file are numbered from 1 to n, each word containing 10 characters. Retrieving or writing of the data at any given word within the file is specified by the word number, called the word address.

Zero-Byte Terminator -
The 12 bits of zero in the low order position of a word; the 12 bits mark the end of the line to be displayed at a terminal or printed on a line printer. The image of cards input through the card reader or terminal also has such a terminator.

Zero-Length PRU -
A PRU that contains system information, but no user data. Under CRM, a zero-length PRU of level 17 is a partition boundary. Under NOS, a zero-length PRU defines EOF.

The keywords for FORTRAN/DDL and FORTRAN/DML are listed below.

FORTRAN/DDL KEYWORDS

.A.
ALIAS
ALL
.AND.

BOOLEAN

CHARACTER
COMPLEX

DOUBLE

END
.EQ.

.GE.
.GT.

INTEGER
ITEM

.LE.
LOGICAL
.LT.

.N.
.NE.
.NOT.

.O.
.OR.

PRECISION

REAL
REALM
RECORD
RELATION
RESTRICT

SCHEMA
SUBSCHEMA

.X.
.XOR.

FORTRAN/DML KEYWORDS

CLOSE

DELETE

END
.EQ.
ERR

.GE.
.GT.

I
INVOKE
IO

KEY

.LE.
LOCK
.LT.

MODE

.NOT.

O
OPEN

PRIVACY

READ
REWRITE

START
SUBSCHEMA

TERMINATE

UNLOCK

WRITE

The format specifications for all FORTRAN/DDL and FORTRAN/DML statements for use with FORTRAN 5 are summarized and listed in this appendix. Detailed information for each format is referenced by page number.

SUMMARY OF FORTRAN/DDL STATEMENTS - FORTRAN 5

Format of logical expression:

[.NOT. / .N.] [(] db-item-1 { .EQ. / .NE. / .GT. / .LT. / .GE. / .LE. } { db-item-2 / constant-1 / non-db-item-1 } [)]

[{ .AND. / .A. / .OR. / .O. / .XOR. / .X. } [.NOT. / .N.] [(] db-item-3 { .EQ. / .NE. / .GT. / .LT. / .GE. / .LE. } { db-item-4 / constant-2 / non-db-item-2 } [)]] . . .

SUMMARY OF FORTRAN/DML STATEMENTS - FORTRAN 5

	Page No.
CLOSE ({realm-name / relation-name}[,ERR=s])	6-3
DELETE (realm-name[,ERR=s])	6-5
INVOKE	6-1
LOCK (realm-name[,ERR=s])	6-6
OPEN ({realm-name / relation-name} [, MODE= {I / IO / O}] [,ERR=s])	6-3
PRIVACY (realm-name,[MODE= {I / IO / O} ,] PRIVACY = {character-constant / variable-name / array-name})	6-6
READ ({realm-name / relation-name} [, KEY {= / .EQ. / .GT. / .GE.} item-name] [,ERR=s][,END=s])	6-3
REWRITE (realm-name[,ERR=s])	6-5
START ({realm-name / relation-name} [, KEY {= / .EQ. / .GT. / .GE.} item-name] [,ERR=s])	6-5
SUBSCHEMA (sub-schema-name)	6-1
TERMINATE	6-3
UNLOCK (realm-name[,ERR=s])	6-6
WRITE (realm-name[,ERR=s])	6-5

SYNTAX SUMMARY - FORTRAN 4

The format specifications for all FORTRAN/DDL and FORTRAN/DML statements for use with FORTRAN 4 are summarized and listed in this appendix. Detailed information for each format is referenced by page number.

SUMMARY OF FORTRAN/DDL STATEMENTS - FORTRAN 4

Format of logical expression:

[.NOT. / .N.] [(]db-item-1 {.EQ. / .NE. / .GT. / .LT. / .GE. / .LE.} {db-item-2 / constant-1 / non-db-item-1} [)]

[{.AND. / .A. / .OR. / .O. / .XOR. / .X.} [.NOT. / .N.] [(]db-item-3 {.EQ. / .NE. / .GT. / .LT. / .GE. / .LE.} {db-item-4 / constant-2 / non-db-item-2} [)]] . . .

SUMMARY OF FORTRAN/DML STATEMENTS - FORTRAN 4

	Page No.
CLOSE ({ realm-name / relation-name })	6-3
DELETE (realm-name)	6-5
INVOKE	6-1
LOCK (realm-name)	6-6
OPEN ({ realm-name / relation-name } [, MODE= { I / IO / O }])	6-3
PRIVACY (realm-name, [MODE = { I / IO / O } ,] PRIVACY = { Hollerith-constant / array-name })	6-6
READ ({ realm-name / relation-name } [, KEY { = / .EQ. / .GT. / .GE. } item-name])	6-3
REWRITE (realm-name)	6-5
START ({ realm-name / relation-name } [, KEY { = / .EQ. / .GT. / .GE. } item-name])	6-5
SUBSCHEMA (sub-schema-name)	6-1
TERMINATE	6-3
UNLOCK (realm-name)	6-6
WRITE (realm-name)	6-5

NAMES OF VARIABLES AND COMMON BLOCKS GENERATED BY THE DML PREPROCESSOR

DBFnnnn	where nnnn is 0001 through 9999
DBInnnn	where nnnn is 0001 through 9999
DBNnnnn	where nnnn is 0001 through 9999
DBREALM	
DBRELST	
DBRUID	
DBRnnnn	where nnnn is 0001 through 9999
DBSCNAM	
DBSTAT	
DBSnnnn	where nnnn is 0001 through 9999
DBTEMP	
DBTnnnn	where nnnn is 0001 through 9999
DBnnnn	where nnnn is 0000 through 9999
Dnnnnxx	where nnnn is 0001 through 9999 and xx is AA through ZZ

The CDCS Batch Test Facility provides the capability of running CDCS along with one or more user jobs as a normal batch job at a control point. This facility is intended primarily for use when a program is being developed and tested, since data and file definitions are changing frequently during this stage. By running the Batch Test Facility, the user can attach new versions of the master directory file each time the job is run. Normally, when CDCS is running at a system control point, the control point must be dropped and reinitiated to attach a new master directory file.

The CDCS Batch Test Facility is an absolute program, called CDCSBTF, which resides on the system library. CDCSBTF consists of the normal CDCS system control point routines and tables plus a set of special routines that communicate with the user programs at the CDCS control point. These special routines load the user programs and simulate the interface between the user control point and a system control point.

Multiple copies of CDCSBTF can be run concurrently with each other and with a system control point version of CDCS. In addition, as many as 16 user programs can be run with each copy of CDCSBTF. Because user programs are loaded by a program-initiated load from CDCSBTF, the programs must be in relocatable binary format. Programs in absolute binary format, as well as segmented programs and overlays, cannot be run with CDCSBTF.

CDCSBTF EXECUTION

The CDCSBTF program is called into execution by a control statement. Before the program is executed, however, several file requirements and restrictions should be considered. In addition, load maps from the user call load operations can be obtained by setting switches prior to the execution of CDCSBTF. The allocation of field length and other required resources, such as magnetic tapes for log files, is handled as for a normal batch job.

CDCSBTF CONTROL STATEMENT

The CDCS Batch Test Facility is executed by the CDCSBTF control statement. As many as 16 user program file names can be specified in the statement. No control statement parameters for the programs can be included, however. The format of the CDCSBTF control statement is shown in figure H-1.

CDCSBTF(lfn-1,lfn-2, . . .)

lfn	Specifies the logical file name of a relocatable binary file containing a user program. Up to 16 files can be specified.

Figure H-1. CDCSBTF Control Statement Format

FILE REQUIREMENTS AND RESTRICTIONS

Certain files must be attached or requested before the CDCSBTF program can be executed. The master directory file must be attached using the file name MSTRDIR. All necessary log files must also be attached or requested. The names of the log files must correspond to the file names described in the master directory utility run. The job name recorded on a log file during execution of CDCSBTF is Cnnnnnn, where nnnnnn is the number associated with the particular user job. The user programs that are to be run with CDCSBTF must be present in relocatable binary format either as local files or as permanent files.

When CDCS is executing as the Batch Test Facility, the name of the output file produced by CDCS is CDCSOUT.

Care must be taken in assigning file names to non-CDCS files when the Batch Test Facility is being used. Because several user programs can be executed during one run of CDCSBTF, every non-CDCS file referenced by the user programs must have a unique name. This restriction is especially critical for the file names INPUT and OUTPUT; they can be used by only one of the programs. Since CDCS cannot enforce this restriction, the user must take particular care in using these names.

Programs running with CDCSBTF cannot use as file names the name MSTRDIR, CDCSOUT, or a name beginning with five Zs; moreover, a name consisting of P, X, or F followed by six digits cannot be used. In addition, no logical file name used for a log file can appear within a user program.

FORTRAN programs running with CDCSBTF must execute the DML TERMINATE statement before a STOP or END statement.

PROCESSING CONSIDERATIONS

The following paragraphs describe processing limitations that apply to CDCS executing as the Batch Test Facility.

In a FORTRAN program executing with the CDCS Batch Test Facility, the Data Manipulation Language (DML) TERMINATE statement must execute before a FORTRAN STOP or END statement. If execution of the FORTRAN program ends without a TERMINATE statement being executed, processing is discontinued for all programs specified in the CDCSBTF control statement that have not completed execution.

The CDCS Batch Test Facility has a limitation on the number of jobs for which abnormal end-of-job processing can be performed. For all the programs specified in the CDCSBTF control statement, the Batch Test Facility allows a total of two concurrent calls to the RECOVR routine. If each of three or more programs, executing with the CDCS Batch Test Facility, requires a call to the RECOVR routine, the CDCS Batch Test Facility aborts processing. The following diagnostic is issued:

RECOVR - TOO MANY RECOVERY REQUESTS

FORTRAN 5 provides a mechanism that can eliminate automatic calls to the RECOVR routine and can help prevent the problem of too many recovery requests. The DB parameter in the FTN5 control statement affects end-of-job processing. If the parameter DB=0 is specified in the FTN5 control statement, the RECOVR routine is not automatically called for abnormal end-of-job processing.

If the CDCS Batch Test Facility aborts processing, a close operation is performed on any open data base files.

LOAD MAPS

Maps of the program loading operations of a program-initiated load can be obtained by setting sense switches 1 through 4 prior to execution of the CDCSBTF control statement. Each sense switch setting corresponds to different information on the load map. The settings and the associated types of information are as follows:

Setting	Load Map Information
SWITCH,1.	Statistics (S)
SWITCH,2.	Block maps (B)
SWITCH,3.	Entry point maps (E)
SWITCH,4.	Entry point cross-reference maps (X)

The control statements in figure H-2 illustrate sample NOS and NOS/BE jobs in which the CDCS Batch Test Facility is executed for a FORTRAN program. Before the CDCSBTF program is called, files containing the sub-schema directory (needed for program compilation), the master directory, and a journal log file are attached. In addition, a second journal log file is requested and the desired portions of the load map are selected.

NOS/BE Operating System	NOS Operating System	
JOB,CMfl.	JOB,CMfl.	Specifies maximum field length.
ATTACH,SUBSC,ID=xxx.	ATTACH,SUBSC/UN=xxx.	Attaches the sub-schema.
DML,SB=SUBSC.	DML,SB=SUBSC.	Preprocesses the DML statements in the FORTRAN program and writes to DMLOUT.
FTN,I=DMLOUT.	FTN,I=DMLOUT.	Compiles the FORTRAN program on DMLOUT and places it on the LGO file.
ATTACH,MSTRDIR,ID=xxx.	ATTACH,MSTRDIR/UN=xxx.	Attaches the master directory.
ATTACH,LOG1,ID=xxx.	ATTACH,LOG1/UN=xxx.	Attaches first journal log file.
REQUEST,LOG2,MT,RING,...	REQUEST,LOG2,MT,PO=W,...	Requests second journal log file.
SWITCH,2.	SWITCH,2.	Requests block map on program-initiated load.
SWITCH,3.	SWITCH,3.	Requests entry point map on program-initiated load.
SWITCH,4.	SWITCH,4.	Requests entry point cross reference map on program-initiated load.
LIBRARY,DMSLIB.	LIBRARY,DMSLIB.	Specifies that library DMSLIB is to be used to satisfy externals.
CDCSBTF,LGO.	CDCSBTF,LGO.	Executes CDCSBTF.
REWIND,CDCSOUT.	REWIND,CDCSOUT.	Rewinds the CDCSBTF output file.
COPYSBF,CDCSOUT,OUTPUT.	COPYSBF,CDCSOUT,OUTPUT.	Prints the CDCSBTF output file.
EXIT.	EXIT.	Establishes processing if error occurs.
DMP.	DMP.	Dumps the exchange package.
DMP,177000.	DMP,177000.	Dumps the contents of the field length.
REWIND,CDCSOUT.	REWIND,CDCSOUT.	Rewinds the CDCSBTF output file.
COPYSBF,CDCSOUT,OUTPUT.	COPYSBF,CDCSOUT,OUTPUT.	Prints the CDCSBTF output file.

Figure H-2. Sample FORTRAN Execution of CDCS Batch Test Facility

COMPILATION OUTPUT LISTINGS OF EXAMPLES I

This appendix contains the compilation output listings of the programs used in the examples in this manual. The figure number in parentheses shown with each output listing matches the figure number of the corresponding program source listing that appears in section 7.

For the ouput listing of the FORTRAN/DML program for the payroll example, the DS parameter was specified in the DML control statement so that the listing contains all the FDBF-generated statements. The listing (figure I-10) shows that comments from the sub-schema can be inserted in the FORTRAN/DML program. See the Compilation/Execution subsection for more information on the DS parameter.

The following compilation output listings are from the programs in the university example on using sub-schemas. These programs illustrate a FORTRAN 5 application.

Figure I-1	Schema
Figure I-2	First sub-schema
Figure I-3	First FORTRAN/DML program
Figure I-4	Second sub-schema
Figure I-5	Second FORTRAN/DML program
Figure I-6	Master directory

```
                00001          SCHEMA TEST-FILES.
                00002          AREA TESTS.
                00003          RECORD R1 WITHIN TESTS.
                00004                TESTNO TYPE FIXED.
                00005                TNAME TYPE CHARACTER 20.
                00006                PROFNUM TYPE FIXED.
                00007                PROF PICTURE "X(20)".
                00008                RATING TYPE FLOAT.
                00009                STCOUNT TYPE FIXED.
                00010                PROB TYPE FLOAT OCCURS 100 TIMES,
                00011                    CHECK VALUE 0.0 THRU 1.0.
                00012          DATA CONTROL.
                00013          AREA TESTS KEY IS TESTNO, DUPLICATES ARE NOT ALLOWED,
                00014                KEY IS ALTERNATE PROFNUM, DUPLICATES ARE INDEXED,
                00015                SEQUENCE IS ASCII.
*** AREA CHECKSUMS ***
        AREA NAME                                  CHECKSUM
        TESTS                                      2330471564566635511
DDL COMPLETE.           0 DIAGNOSTICS.
  45300B CM USED.     0.230  CP SECS.
```

Figure I-1. Output Listing of the Schema for the University Example (Figure 7-1)

```
                00001                SUBSCHEMA PROBSS, SCHEMA = TEST-FILES
                00002                REALM TESTS
                00003                RECORD R1
** WITHIN TESTS
                00004                INTEGER TESTNO,STCOUNT
** ORDINAL    2
                00005                REAL PROB(100)
** ORDINAL    3
                00006                END
                *****       END OF SUB-SCHEMA SOURCE INPUT

 PRIMARY KEY    00004       TESTNO FOR AREA TESTS
                *****       RECORD MAPPING IS NEEDED FOR REALM -   TESTS


                -----       BEGIN SUB-SCHEMA FILE MAINTENANCE                -----

          SUBSCHEMA                                CHECKSUM
          PROBSS                                   14037043413701606733

                -----       END OF FILE MAINTENANCE                  -----
DDLF COMPLETE.          0 DIAGNOSTICS.
  50100B CM USED.     0.951  CP SECS.
```

Figure I-2. Output Listing of the First Sub-Schema for the University Example

```
 1        PROGRAM NEWTEST
 2        INTEGER STATBLK(5)
 3        REAL NEWPROB(100),X(100),Y(100)
 4        EQUIVALENCE(PROB,X),(NEWPROB,Y),(CORCOEF,R)
 5 **     SUBSCHEMA(PROBSS)
 6 C$     LIST(ALL=0)
34 C$     LIST(ALL)
35        N = 100
36 **     INVOKE
37 C$     LIST(ALL=0)
41 C$     LIST(ALL)
42        CALL DMLINV(0001,DBF0001,10HPROBSS     ,10H           ,
43       +10H           ,0"3327171321772015547 7")
44        CALL DMLDBST(STATBLK,5)
45        OPEN(5,FILE='NEWTPE',STATUS='OLD',ACCESS='SEQUENTIAL')
46 **     OPEN(TESTS)
47 C    MAIN LOOP
48        CALL DMLOPN(DBF0001,0001,2H10)
49  10    READ(5,*,ERR=40,END=50) TESTNO,NEWST,NEWPROB
50 **     READ(TESTS,KEY=TESTNO,ERR=45)
51        CALL DMLRDK(DBF0001,0001,00001,0001,1,0010,1,0000,00,
52       +TESTNO ,*45    )
53        SUMXY = SUMX = SUMY = SUMXSQ = SUMYSQ = 0.0
54        DO 20 I=1,N
55        SUMXY = SUMXY + X(I)*Y(I)
56        SUMX = SUMX + X(I)
57        SUMY = SUMY + Y(I)
58        SUMXSQ = SUMXSQ + X(I)**2
59  20    SUMYSQ = SUMYSQ + Y(I)**2
60        R = (N*SUMXY - SUMX*SUMY)/
61       1     (SQRT(N*SUMXSQ - SUMX**2) * SQRT(N*SUMYSQ - SUMY**2)
62        PRINT *, ' TEST NO. = ', TESTNO,
63       1          ' CORRELATION COEFFICIENT = ', CORCOEF
64        NEWTOT = STCOUNT + NEWST
65        DO 30 I=1,N
66  30    PROB(I) = (PROB(I)*STCOUNT + NEWPROB(I)*NEWST)/NEWTOT
67        STCOUNT = NEWTOT
68 **     REWRITE(TESTS,ERR=45)
69        CALL DMLREW(DBF0001,0,0001,00001,*45    )
70        GO TO 10
71  40    PRINT *, ' ERROR ON FILE READ'
72  45    PRINT 46, STATBLK
73  46    FORMAT (1X,'STATUS BLOCK'/
74       1        1X,04,2X,I5,2X,03,2X,I2,2X,A10)
75 **     CLOSE(TESTS)
76  50    CALL DMLCLS(DBF0001,0001)
77        CLOSE(5,STATUS='DELETE')
78 **     TERMINATE
79        CALL DMLEND
80        STOP
81        END
```

Figure I-3. Output Listing of the First FORTRAN/DML Program for the University Example (Figure 7-3)

```
                00001                SUBSCHEMA RATE, SCHEMA = TEST-FILES
                00002                REALM TESTS
                00003                RECORD R1
** WITHIN TESTS
                00004                INTEGER TESTNO, PROFNUM
** ORDINAL    2
                00005                CHARACTER *20 TNAME, PROF
** ORDINAL    4
                00006                REAL RATING, PROB(100)
** ORDINAL    6
                00007                END
                *****          END OF SUB-SCHEMA SOURCE INPUT

 PRIMARY KEY    00004          TESTNO FOR AREA TESTS
 ALTERNATE KEY  00004          PROFNUM FOR AREA TESTS
                *****          RECORD MAPPING IS NEEDED FOR REALM -   TESTS


                -----          BEGIN SUB-SCHEMA FILE MAINTENANCE                -----

          SUBSCHEMA                                    CHECKSUM
          RATE                                         7413057023341753027 3

                -----          END OF FILE MAINTENANCE                          -----
 DDLF COMPLETE.             0 DIAGNOSTICS.
   501008 CM USED.         1.169  CP SECS.
```

Figure I-4. Output Listing of the Second Sub-Schema for the University Example (Figure 7-5)

```
     1        PROGRAM RATER
     2        INTEGER ALTKEY
     3        INTEGER STATBLK(5)
     4 **     SUBSCHEMA (RATE)
     5 C$     LIST(ALL=0)
    36 C$     LIST(ALL)
    37 **     INVOKE
    38 C$     LIST(ALL=0)
    42 C$     LIST(ALL)
    43        CALL DMLINV(0001,DBF0001,10HRATE       ,10H           ,
    44       +10H           ,0"3224134454175112763 1")
    45        CALL DMLDBST(STATBLK,5)
    46        OPEN(5,FILE='PROFS',STATUS='OLD',ACCESS='SEQUENTIAL')
    47 **     OPEN(TESTS)
    48        CALL DMLOPN(DBF0001,0001,2HIO)
    49  100   READ (5,*,END=900) PROFNUM
    50 **     READ(TESTS,KEY=PROFNUM,ERR=800)
    51        CALL DMLRDK(DBF0001,0001,00002,0001,1,0010,1,0001,00,
    52       +PROFNUM,*800  )
    53        ALTKEY = PROFNUM
    54        PRINT 12, PROF, RATING
    55  200   SUM = 0.0
    56        DO 300 I=1,100
    57        SUM = SUM + PROB(I)
    58  300   CONTINUE
    59        AVG = SUM/100.0
    60        PRINT 13, TNAME, AVG
    61 **     READ(TESTS,ERR=800,END=100)
    62        CALL DMLRD(DBF0001,0001,1,1,*800  ,*100  )
    63        IF (PROFNUM .NE. ALTKEY) GO TO 100
    64        GO TO 200
    65  800   PRINT 14, STATBLK
    66 **     CLOSE(TESTS)
    67  900   CALL DMLCLS(DBF0001,0001)
    68 **     TERMINATE
    69        CALL DMLEND
    70        CLOSE(5,STATUS='DELETE')
    71        STOP
    72   12   FORMAT (' PROF = ', A20, ' RATING = ', F4.1)
    73   13   FORMAT (1X,A20,4X,F4.3)
    74   14   FORMAT (1X,'STATUS BLOCK'/
    75       1         1X,O4,2X,I5,2X,O3,2X,I2,2X,A10)
    76        END
```

Figure I-5. Output Listing of the Second FORTRAN/DML Program for the University Example (Figure 7-6)

```
 M A S T E R   D I R E C T O R Y   C O N T E N T S
0    (S=SCHEMA, A=AREA, R=RELATION, SS=SUB-SCHEMA)
0      NAME                                        ID
-
     S  TEST-FILES                                  1
        CREATION DATE -  79282  TIME -  10.32
        A  TESTS                                    1
             CHECKSUM -  23104715645666355511
          SS  PROBSS
             CHECKSUM -  14037043413701606733
             CREATION DATE - 79282  TIME -  10.32
          SS  RATE
             CHECKSUM -  74130570233417530273
             CREATION DATE - 79282  TIME -  10.32

     SUMMARY FOR TEST-FILES
        NUMBER OF AREAS                  1
        NUMBER OF RELATIONS              0
        NUMBER OF SUB-SCHEMAS            2
-
     OVERALL SUMMARY
        NUMBER OF SCHEMAS                1
        NUMBER OF SUB-SCHEMAS            2
        NUMBER OF RELATIONS              0
        NUMBER OF AREAS                  1
        DIRECTORY SIZE (WORDS)         725
```

Figure I-6. Output Listing of the Master Directory for the University Example (Figure 7-7)

The following compilation output listings are from the programs used in the payroll example on using relations. These programs illustrate a FORTRAN 4 application.

Figure I-7	Schema
Figure I-8	Sub-schema
Figure I-9	Master directory
Figure I-10	FORTRAN/DML program

```
00001                   SCHEMA PAYDATA.
00002           /*      SET UP A SCHEMA TO BE USED      */
00003           /*      BY FORTRAN 4 APPLICATIONS       */
00004                   AREA PFILE.
00005                   RECORD IS REC WITHIN PFILE.
00006                       SOCSEC TYPE CHARACTER 9.
00007                       LNAME TYPE CHARACTER 10
00008                           OCCURS 2 TIMES.
00009                       FNAME TYPE CHARACTER 10.
00010                       MI TYPE CHARACTER 1.
00011                       PCODE TYPE CHARACTER 4.
00012                       TITLE TYPE CHARACTER 10
00013                           OCCURS 2 TIMES.
00014                   AREA EXMPT.
00015                   RECORD IS SREC1 WITHIN EXMPT.
00016                       SOCSEC TYPE CHARACTER 9.
00017                       MSALARY TYPE CHARACTER 4.
00018                       VACHRS TYPE CHARACTER 4.
00019                       SICKHRS TYPE CHARACTER 4.
00020                       EMPDATE TYPE CHARACTER 6.
00021                   AREA NEXMPT.
00022                   RECORD IS SREC2 WITHIN NEXMPT.
00023                       SOCSEC TYPE CHARACTER 9.
00024                       MSALARY TYPE CHARACTER 4.
00025                       UNION PICTURE "XX".
00026                       EMPDATE PICTURE "X(6)".
00027                       VACHRS PICTURE "X(4)".
00028                       SICKHRS PICTURE "X(4)".
00029                   DATA CONTROL.
00030           /* KEYS ARE DEFINED HERE.  NOTE THAT    */
00031           /* SOCSEC MUST BE QUALIFIED BY RECORD   */
00032           /* SINCE IT IS NOT A UNIQUE NAME        */
00033                       AREA NAME IS PFILE
00034                           KEY IS SOCSEC OF REC
00035                           DUPLICATES ARE NOT ALLOWED
00036                           KEY IS ALTERNATE PCODE
00037                           DUPLICATES ARE NOT ALLOWED.
00038                       AREA NAME IS EXMPT
00039                           KEY IS SOCSEC OF SREC1
00040                           DUPLICATES ARE NOT ALLOWED.
00041                       AREA NAME IS NEXMPT
00042                           KEY IS SOCSEC OF SREC2
00043                           DUPLICATES ARE NOT ALLOWED.
00044                   RELATION NAME IS FIXUP1
00045                       JOIN WHERE SOCSEC OF REC
00046                           EQ
00047                       SOCSEC OF SREC1.
00048                   RELATION NAME IS FIXUP2
00049                       JOIN WHERE SOCSEC OF REC
00050                           EQ
00051                       SOCSEC OF SREC2.
*** AREA CHECKSUMS ***
        AREA NAME                           CHECKSUM
        PFILE                               4621455760357704246 4
        EXMPT                               6365312420004116233 1
        NEXMPT                              4211761540354613551 3
*** RELATION CHECKSUMS ***
        RELATION NAME                       CHECKSUM
        FIXUP1                              5246053606120007001 3
        FIXUP2                              5246053606320007001 3
DDL COMPLETE.          0 DIAGNOSTICS.
  453008 CM USED.     0.693  CP SECS.
```

Figure I-7. Output Listing of the Schema for the Payroll Example (Figure 7-10)

```
                00001                   SUBSCHEMA EXAMP1, SCHEMA=PAYDATA
                00002             C
                00003             C     THIS COMMENT DEMONSTRATES THE
                00004             C     ABILITY OF FDBF TO CARRY SUB-SCHEMA
                00005             C     COMMENTS TO THE FORTRAN SOURCE
                00006             C     LISTING IF THE DS PARAMETER IS
                00007             C     SPECIFIED ON THE DML CONTROL
                00008             C     STATEMENT
                00009             C
                00010                   ALIAS (ITEM) TSOCSEC=SOCSEC.SREC1
                00011                   REALM PFILE,EXMPT
                00012                   RECORD REC
** WITHIN PFILE
                00013                   INTEGER SOCSEC,LNAME(2),FNAME,MI,
                00014                  +        PCODE,TITLE(2)
** ORDINAL    6
                00015                   RECORD SREC1
** WITHIN EXMPT
                00016                   INTEGER TSOCSEC,MSALARY,VACHRS,
                00017                  +        SICKHRS,EMPDATE
** ORDINAL    5
                00018                   RELATION FIXUP1
  ***314*       00013         WARNING: ITEM -SOCSEC- SS SIZE GR SCHEMA SIZE - MAY CAUSE TRU
  ***314*       00013         WARNING: ITEM -MI- SS SIZE GR SCHEMA SIZE - MAY CAUSE TRUNCAT
  ***314*       00013         WARNING: ITEM -PCODE- SS SIZE GR SCHEMA SIZE - MAY CAUSE TRUN
  ***314*       00016         WARNING: ITEM -TSOCSEC- SS SIZE GR SCHEMA SIZE - MAY CAUSE TR
  ***314*       00016         WARNING: ITEM -MSALARY- SS SIZE GR SCHEMA SIZE - MAY CAUSE TI
  ***314*       00016         WARNING: ITEM -VACHRS- SS SIZE GR SCHEMA SIZE - MAY CAUSE TRI
  ***314*       00016         WARNING: ITEM -SICKHRS- SS SIZE GR SCHEMA SIZE - MAY CAUSE TR
  ***314*       00016         WARNING: ITEM -EMPDATE- SS SIZE GR SCHEMA SIZE - MAY CAUSE TR
 PRIMARY KEY    00013         SOCSEC FOR AREA PFILE
 ALTERNATE KEY  00013         PCODE FOR AREA PFILE
 PRIMARY KEY    00016         TSOCSEC FOR AREA EXMPT
                *****         RECORD MAPPING IS NEEDED FOR REALM -   PFILE
                *****         RECORD MAPPING IS NEEDED FOR REALM -   EXMPT
                00019                   END
                *****         END OF SUB-SCHEMA SOURCE INPUT

                *****                     RELATION   STATISTICS                    *****
 RELATION 001                 FIXUP1 JOINS              AREA - PFILE
                                                        AREA - EXMPT




                -----         BEGIN SUB-SCHEMA FILE MAINTENANCE                    -----

          SUBSCHEMA                                    CHECKSUM
          EXAMP1                                       66421650035257141055

                -----         END OF FILE MAINTENANCE                    -----
 DDLF COMPLETE.             8 DIAGNOSTICS.
   51000B CM USED.        1.783  CP SECS.
```

Figure I-8. Output Listing of the Sub-Schema for the Payroll Example (Figure 7-11)

```
          M A S T E R   D I R E C T O R Y   C O N T E N T S
0    (S=SCHEMA, A=AREA, R=RELATION, SS=SUB-SCHEMA)
0      NAME                                        ID
-
    S  PAYDATA                                     1
       CREATION DATE -  79285  TIME -  11.07
       A  PFILE                                    1
          CHECKSUM -  46214557603577042464
       A  EXMPT                                    2
          CHECKSUM -  63653124200041162331
       A  NEXMPT                                   3
          CHECKSUM -  42117615403546135513
       R  FIXUP1
          CHECKSUM -  52460536061200070013
       R  FIXUP2
          CHECKSUM -  52460536063200070013
      SS  EXAMP1
          CHECKSUM -  66421650035257141055
          CREATION DATE - 79285  TIME -  11.08

    SUMMARY FOR PAYDATA
       NUMBER OF AREAS                  3
       NUMBER OF RELATIONS              2
       NUMBER OF SUB-SCHEMAS            1
-
    OVERALL SUMMARY
       NUMBER OF SCHEMAS                1
       NUMBER OF SUB-SCHEMAS            1
       NUMBER OF RELATIONS              2
       NUMBER OF AREAS                  3
       DIRECTORY SIZE (WORDS)        1465
```

Figure I-9. Output Listing of the Master Directory for the Payroll Example (Figure 7-12)

```
      PROGRAM FDBFTST(OUTPUT=64)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     A PROGRAM TO TEST SOME FEATURES OF THE FORTRAN DATA BASE FACILITY
C
C     DECLARE SUBSCHEMA APPLICATION CAN ACCESS
C
C     INVOKE CDCS AND OPEN FILES FOR PROCESSING
C
**    SUBSCHEMA(EXAMP1)
**    SUBSCHEMA EXAMP1, SCHEMA=PAYDATA
C
C     THIS COMMENT DEMONSTRATES THE
C     ABILITY OF FDBF TO CARRY SUB-SCHEMA
C     COMMENTS TO THE FORTRAN SOURCE
C     LISTING IF THE DS PARAMETER IS
C     SPECIFIED ON THE DML CONTROL
C     STATEMENT
C
**    ALIAS (ITEM) TSOCSEC=SOCSEC.SREC1
**    REALM PFILE,EXMPT
**    RECORD REC
** WITHIN PFILE
      INTEGER SOCSEC,LNAME(2),FNAME,MI,
     +        PCODE,TITLE(2)
** ORDINAL    6
**    RECORD SREC1
** WITHIN EXMPT
      INTEGER TSOCSEC,MSALARY,VACHRS,
     +        SICKHRS,EMPDATE
** ORDINAL    5
**    RELATION FIXUP1
**    END
```

Figure I-10. Output Listing of the FORTRAN/DML Program for the Payroll Example (Figure 7-13) (Sheet 1 of 3)

```
      INTEGER DBI0001
      EQUIVALENCE (DBI0001,SOCSEC )
      COMMON/DB0001/ SOCSEC,LNAME,FNAME,MI,PCODE,TITLE
      INTEGER DBI0002
      EQUIVALENCE (DBI0002,TSOCSEC)
      COMMON/DB0002/ TSOCSEC,MSALARY,VACHRS,SICKHRS,EMPDATE
      INTEGER DBREALM(3),DBSTAT,DBSCNAM(3),DBRUID
      COMMON/DB0000/DBREALM,DBSTAT,DBSCNAM,DBRUID
     +,DBRELST
      INTEGER DBR0001(3),DBS0001,DBF0001(35),DBT0001(2)
      COMMON/DB0000/DBR0001,DBS0001,DBF0001,DBT0001
      INTEGER DBR0002(3),DBS0002,DBF0002(35),DBT0002(2)
      COMMON/DB0000/DBR0002,DBS0002,DBF0002,DBT0002
      INTEGER DBN0001(0003)
      COMMON/DB0000/DBN0001
      INTEGER DBA0001(0003)
      INTEGER DBRELST(0002)
      DATA DBN0001/0002*0,0/
      DATA DBA0001/
     +0001          ,
     +0002          ,
     +0/
      DATA DBRELST/
     +00001000000000000001B,
     +00000000000000000000B/
      DATA DBT0001/10HFT4       ,0/
      DATA DBREALM/3*1H /,DBSTAT/0/
     +,DBSCNAM/10HPAYDATA    ,10H          ,10H          /
      DATA DBR0001/10HPFILE     ,10H          ,10H          /,
     + DBS0001/0/,DBF0001/12*0,00001200000000000000B,11*0,
     +00000000000000000000B,6*0,00000002200000000000B,
     +00000000007000000000B,2*0/
      DATA DBR0002/10HEXMPT     ,10H          ,10H          /,
     + DBS0002/0/,DBF0002/12*0,00000620000000000000B,11*0,
     +00000000000000000000B,6*0,00000002200000000000B,
     +00000000007000000000B,2*0/
**    INVOKE
      DATA DBRUID /10HFDBFTST   /
      DBF0001(16)=LOCF(DBI0001)
      DBF0001(25)=DBF0001(25)+DBF0001(16)
      DBF0002(16)=LOCF(DBI0002)
      DBF0002(25)=DBF0002(25)+DBF0002(16)
      DBN0001(0001)=LOCF(DBF0001)
      DBN0001(0002)=LOCF(DBF0002)
      CALL DMLINV(0002,DBF0001,10HEXAMP1    ,10H          ,
     +10H          ,66421650035257141055B)
**    OPEN(PFILE,MODE=0)
      CALL DMLOPN(DBF0001,0001,2H0 )
101     FORMAT (" DBSTAT= ", O4)
      WRITE 101, DBSTAT
**    OPEN(EXMPT,MODE=0)
      CALL DMLOPN(DBF0002,0002,2H0 )
      WRITE 101, DBSTAT
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     DEFINE VALUES OF DATA ELEMENTS
C
      SOCSEC="123456789"
      FNAME="TOM"
      MI="X"
      LNAME(1)="JONES"
      LNAME(2)=" "
      PCODE="3000"
      TITLE(1)="PROGRAMMER"
      TITLE(2)=" "
      TSOCSEC="123456789"
      MSALARY="1750"
      VACHRS="80"
      SICKHRS="20"
      EMPDATE="010177"
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

Figure I-10. Output Listing of the FORTRAN/DML Program for the Payroll Example (Figure 7-13) (Sheet 2 of 3)

```
      C
      C     WRITE VALUES TO DATABASE
      C
      **    WRITE(PFILE)
            CALL DMLWRT(DBF0001,0,0001,00001)
            WRITE 101, DBSTAT
      **    WRITE(EXMPT)
            CALL DMLWRT(DBF0002,0,0002,00001)
            WRITE 101, DBSTAT
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      C
      C     CLOSE FILES TO END CREATION MODE OF DATA BASE
      C
      **    CLOSE(PFILE)
            CALL DMLCLS(DBF0001,0001)
            WRITE 101, DBSTAT
      **    CLOSE(EXMPT)
            CALL DMLCLS(DBF0002,0002)
            WRITE 101, DBSTAT
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      C
      C     OPEN FILES AND BLANK FILL LAST NAME AND MONTHLY SALARY
      C
      **    OPEN(PFILE)
            CALL DMLOPN(DBF0001,0001,2HIO)
      **    OPEN(EXMPT)
            CALL DMLOPN(DBF0002,0002,2HIO)
            LNAME(1)=10H
            MSALARY=10H
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      C
      C     READ USING THE RELATION DEFINED IN THE SCHEMA/SUBSCHEMA
      C
      **    READ(FIXUP1,KEY=SOCSEC)
            CALL DMLRLK(DBN0001,0001,00001,0001,1,0010,0,0000,00,SOCSEC ,
           + 0001)
            PRINT *,"READ BY RELATION RETURNS:"
            PRINT 100,LNAME,MSALARY
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      C
      C     CHANGE THE VALUE OF MONTHLY SALARY AND REWRITE EXEMPT RECORD
            MSALARY="2000"
      **    REWRITE(EXMPT)
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      C
      C     READ AND PRINT THE VALUES TO VERIFY CHANGE TOOK PLACE
      C
            CALL DMLREW(DBF0002,0,0002,00001)
      **    READ(FIXUP1,KEY=SOCSEC)
            CALL DMLRLK(DBN0001,0001,00001,0001,1,0010,0,0000,00,SOCSEC ,
           + 0001)
            PRINT *,"READ BY RELATION AFTER REWRITE:"
            PRINT 100,LNAME,MSALARY
      100   FORMAT(1X,3A10)
      **    CLOSE(EXMPT)
            CALL DMLCLS(DBF0002,0002)
      **    CLOSE(PFILE)
            CALL DMLCLS(DBF0001,0001)
            NNN=6LOUTPUT
            ENDFILE NNN
      **    TERMINATE
            CALL DMLEND
            STOP
            END
```

Figure I-10. Output Listing of the FORTRAN/DML Program for the Payroll Example (Figure 7-13) (Sheet 3 of 3)

This appendix summarizes the differences in FORTRAN 4 and FORTRAN 5 that are reflected in FORTRAN/DDL statements and in DML statements.

SPECIFYING LANGUAGE VERSION

Both the DDLF control statement (specifying sub-schema compilation) and the DML control statement (specifying DML preprocessing) require that language version be specified. If the language version is not actually specified, a default is assumed. The sub-schema referenced by a FORTRAN/DML program must have been compiled with the same language version specified as that specified for DML preprocessing.

In the DDLF control statement, specifying an F4 or F5 parameter designates compilation of a sub-schema for use by a FORTRAN 4 or FORTRAN 5 program, respectively.

Example:

```
DDLF(F5,SB=SUBSCHM)
```

The DDLF control statement designates that the sub-schema is compiled for use by a FORTRAN 5 program and that the sub-schema is to reside on the sub-schema library file SUBSCHM.

In the DML control statement, specifying the language version parameter LV=F4 or LV=F5 designates preprocessing for FORTRAN 4 or FORTRAN 5, respectively.

Example:

```
DML(LV=F5,SB=SUBSCHM)
```

The DML control statement designates that a program on file INPUT is preprocessed for the FORTRAN 5 compiler, that output is written to file DMLOUT, and that the sub-schema referenced by the program resides on the sub-schema library file SUBSCHM.

LANGUAGE VERSION DIFFERENCES

This section indicates the language elements used in FORTRAN/DDL and DML statements which are different in FORTRAN 4 and FORTRAN 5. For a detailed description of the syntax required in FORTRAN/DDL statements, see section 4 of this manual. For a detailed description of the syntax required in DML statements, see section 6. The syntax summary for FORTRAN 5 applications is in appendix E; the syntax summary for FORTRAN 4 applications is in appendix F.

The following two statements are general syntax rules that apply to the language elements described in subsequent subsections.

> The FORTRAN/DDL compiler requires that the syntax of the statements comprising the sub-schema be consistent with the syntax requirements of the version of FORTRAN specified on the DDLF control statement.
>
> The DML preprocesser requires that the syntax of the DML statements in the FORTRAN/DML program be consistent with the version of FORTRAN specified on the DML control statement.

BLANK LINES

FORTRAN 5 interprets a blank line as a comment line, which does not break a possible continuation sequence. FORTRAN 4 interprets a blank line as an initial statement; therefore, a blank line breaks a possible continuation sequence.

Blank lines can be used in the sub-schema source input and in the FORTRAN/DML program.

DATA TYPES

Data types allowed in the sub-schema type statement depend on the version of FORTRAN. In addition, the method of declaring character data depends on the version of FORTRAN.

Recognized Data Types

FORTRAN 5 allows two data types not allowed by FORTRAN 4; namely, type BOOLEAN and type CHARACTER.

Both versions of FORTRAN recognize the double precision data type. FORTRAN 5 allows the following specification:

```
DOUBLE PRECISION
```

FORTRAN 4 recognizes the following two specifications:

```
DOUBLE
DOUBLE PRECISION
```

These data types are declared in type statements in sub-schema source input.

Declaring Character Data

When a sub-schema is specified for FORTRAN 5, a type CHARACTER data item must be declared in the sub-schema to correspond to a schema data item that is display alphanumeric or display alphabetic (class 0 of 1).

When a sub-schema is specified for FORTRAN 4 and no conversion is desired, a type INTEGER data item must be declared to corrrespond to a schema data item that is display alphanumeric or display alphabetic. If the schema item is longer than 10 characters, FORTRAN/DDL allows a special long variable, which is an integer array declared in the sub-schema, to correspond to the schema item. When specifying this array in a DML READ statement or in a DDL RESTRICT statement, the array-name without subscripts must be specified. In a FORTRAN 4 statement the array is specified as a standard array. For more information about this special long variable, see section 2.

ARRAY DECLARATION

FORTRAN 5 allows an array to have a maximum of seven dimensions. Both the lower and upper bound of a dimension can be specified by a positive, zero, or negative integer constant; however, the lower bound must be less than the declared upper bound.

FORTRAN 4 allows an array to have a maximum of three dimensions. The lower bound is always assumed to be one; the upper bound must be specified by a positive integer constant.

Array declarations are made in type statements in sub-schema source input.

STRING DELIMITERS

FORTRAN 5 allows two delimiters: the apostrophe (in some character sets indicated as an up-arrow) and the quotation mark (in some character sets indicated as a not equals sign). A string of characters delimited by apostrophes is interpreted as a character constant. A string of characters delimited by quotation marks is interpreted as a Hollerith constant; FORTRAN 5 limits Hollerith constants to a length of 10 characters.

FORTRAN 4 allows one delimiter: the quotation mark. The string is interpreted as a Hollerith constant.

The strings can be used in two statements. Used in the FORTRAN/DDL RESTRICT statement, the string can contain up to 255 characters. Used in the DML PRIVACY statement, the string can contain up to 30 characters.

ERROR AND END-OF-FILE SPECIFIERS

FORTRAN 5 allows the ERR (error) and END (end-of-file) specifiers. The specifier indicates the statement where execution is to continue when an error or end-of-file condition occurs upon execution of the statement. The ERR specifier can be used in the following DML statements: CLOSE, DELETE, LOCK, OPEN, READ, REWRITE, UNLOCK, and WRITE. The END specifier can be used in a sequential DML READ statement.

FORTRAN 4 does not allow these specifiers.

These specifiers provide the capability for conditional processing in DML statements for the FORTRAN 5 user. FDBF provides other means for error and end-of-file processing. See the Error Processing subsection in section 6 for more information.

SUMMARY OF DATA DEFINITION IN DMS-170 K

This summary includes all the clauses or statements that can be used in defining data items. Table K-1 shows the schema definition required for data items of each schema data class and the sub-schema definitions that correspond to each schema data class.

For Query Update access to schema-defined data base files in CYBER Record Manager (CRM) data base access mode, the data base must be defined in the Query Update sub-schema exactly as the data base is defined in the schema. Therefore, every data item must be defined to correspond in size and class to the schema definition of the item.

For COBOL, FORTRAN, and Query Update access to schema-defined files in CYBER Database Control System (CDCS) data base access mode, the definition of data items in the sub-schemas does not have to correspond exactly to the schema definition of data items. Through mapping, CDCS can generate a record image conforming to the sub-schema format from a record in schema format, or can perform the conversion of data from sub-schema format to schema format. Detailed information about conversions allowed is included in this manual, and in the CDCS 2, the DDL 3 volume 1, and the DDL 3 volume 2 reference manuals.

TABLE K-1. DATA DEFINITION IN DMS-170

SCHEMA					COBOL Sub-Schema		FORTRAN 5 Sub-Schema Type Statement	FORTRAN Extended 4 Sub-Schema Type Statement	Query Update Sub-Schema in CDCS Data Base Access Mode		Query Update Sub-Schema in CRM Data Base Access Mode	
Data Class No.	Data Class Name	PICTURE Clause	TYPE Clause	Internal Representation	PICTURE Clause	USAGE Clause			PICTURE Clause	USAGE Clause	PICTURE Clause	USAGE Clause
0	Display alpha-numeric	Alpha-numeric (A X 9; not all As or 9s; mixed specifi-cation used as all Xs)	CHARACTER	Display code, alphanumeric	Alpha-numeric (A X 9; not all As or 9s; mixed specifi-cation used as all Xs)	DISPLAY (or none)	CHARACTER	INTEGER	Alpha-numeric (A X 9; not all As or 9s; mixed specifi-cation used as all Xs)	DISPLAY (or none)	Alpha-numeric (A X 9; not all As or 9s; mixed specifi-cation used as all Xs)	DISPLAY (or none)
1	Display alpha-betic	Alpha-betic (A)	None	Display code, alphabetic	Alpha-betic (A)	DISPLAY (or none)	CHARACTER	INTEGER	Alpha-betic (A)	DISPLAY (or none)	Alpha-betic (A) (or none)	DISPLAY (or none)
3	Display integer	Numeric (9 T)	None	Display code numeric, can have sign overpunch in last character position	Numeric (9 S)	DISPLAY COMP (or none)	None†	None†	Numeric (9 S and insertion and re-placement characters)	DISPLAY COMP (or none)	Numeric (9 S and insertion and re-placement characters)	DISPLAY COMP (or none)
4	Display fixed	Numeric (9 P T V .)	None	Display code numeric plus implicit or explicit deci-mal or scaling position	Numeric (9 S V P)	DISPLAY COMP (or none)	None†	None†	Numeric (9 S V P and inser-tion and replacement characters)	DISPLAY COMP (or none)	Numeric (9 S V P and inser-tion and replacement characters)	DISPLAY COMP (or none)
10	Coded binary integer	None	FIXED integer-1 integer-2 (where the inte-ger value is 1 thru 18)	Binary integer	Numeric (9 S V P)	COMP-1 INDEX††	INTEGER LOGICAL BOOLEAN	INTEGER LOGICAL	Numeric (9 S V P and inser-tion and replacement characters)	COMP-1 INDEX†† LOGICAL	Numeric (9 S V P and inser-tion and replacement characters)	COMP-1 INTEGER LOGICAL
13	Coded floating point normal-ized	None	FLOAT integer-1 (where the inte-ger value is 1 thru 14)	Signed, normalized floating point (1 word)	Numeric (9 S V P)	COMP-2	REAL BOOLEAN	REAL	Numeric (9 S V P and inser-tion and replacement characters)	COMP-2	Numeric (9 S V P and inser-tion and replacement characters)	COMP-2

TABLE K-1. DATA DEFINITION IN DMS-170 (Contd)

SCHEMA					COBOL Sub-Schema		FORTRAN 5 Sub-Schema Type Statement	FORTRAN Extended 4 Sub-Schema Type Statement	Query Update Sub-Schema in CDCS Data Base Access Mode		Query Update Sub-Schema in CRM Data Base Access Mode	
Data Class No.	Data Class Name	PICTURE Clause	TYPE Clause	Internal Representation	PICTURE Clause	USAGE Clause			PICTURE Clause	USAGE Clause	PICTURE Clause	USAGE Clause
14	Coded double precision	None	FLOAT integer-1 (where the integer value is 15 thru 29)	Signed, normalized floating point (2 words)	None†	None†	DOUBLE PRECISION	DOUBLE PRECISION	Numeric (9 S V P and insertion and replacement characters)	DOUBLE	Numeric (9 S V P and insertion and replacement characters)	DOUBLE
15	Coded complex	None	COMPLEX	Floating point with real part and imaginary part (2 words)	None†	None†	COMPLEX	COMPLEX	Numeric (9 S V P and insertion and replacement characters)	COMPLEX	Numeric (9 S V P and insertion and replacement characters)	COMPLEX

†No corresponding sub-schema type. Valid conversion is shown in the DDL3 volume 2 or FORTRAN Data Base Facility reference manual.
††No picture clause allowed.

INDEX

CONTROL DATA CORPORATION

COMMENT SHEET

MANUAL TITLE: FORTRAN Data Base Facility Version 1 Reference Manual

PUBLICATION NO.: 60482200 **REVISION:** D

NAME: _____

COMPANY: _____

STREET ADDRESS: _____

CITY: _____ **STATE:** _____ **ZIP CODE:** _____

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

☐ **Please Reply** ☐ **No Reply Necessary**

CUT ALONG LINE

AA3419 REV. 4/79 PRINTED IN U.S.A.

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES

TAPE

TAPE

FOLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Publications and Graphics Division
215 Moffett Park Drive
Sunnyvale, California 94086

CUT ALONG LINE

FOLD

FOLD

TAPE

TAPE