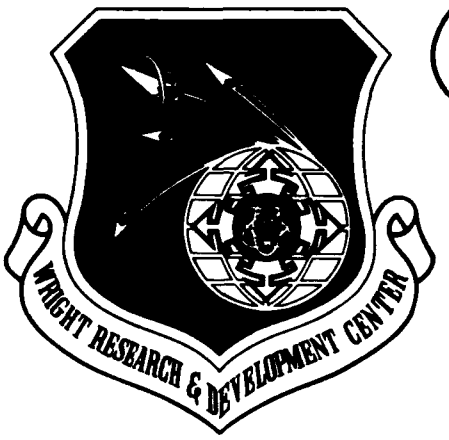
WRDC-TR-90-8007
Volume V
Part 9
Section 1



INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume V - Common Data Model Subsystem
Part 9 - Neutral Data Manipulation Language (NDML) Precompiler Development Specification
Section 1 of 5

J. Althoff, M. Apicella

Control Data Corporation
Integration Technology Services
2970 Presidential Drive
Fairborn, OH 45324-6209

September 1990

Final Report for Period 1 April 1987 - 31 December 1990

Approved for Public Release; Distribution is Unlimited

MANUFACTURING TECHNOLOGY DIRECTORATE
WRIGHT RESEARCH AND DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6533

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever, regardless whether or not the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data. It should not, therefore, be construed or implied by any person, persons, or organization that the Government is licensing or conveying any rights or permission to manufacture, use, or market any patented invention that may in any way be related thereto.

This technical report has been reviewed and is approved for publication.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations

DAVID L. JUDSON, Project Manager
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

FOR THE COMMANDER:

BRUCE A. RASMUSSEN, Chief
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

If your address has changed, if you wish to be removed form our mailing list, or if the addressee is no longer employed by your organization please notify WRDC/MTI, Wright-Patterson Air Force Base, OH 45433-6533 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE

FORM APPROVED
OMB NO. 0704-0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1990	Final Technical Report 1Apr87 - 31Dec90

4. TITLE AND SUBTITLE
INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume V - Common Data Model Subsystem
Part 9 - Neutral Data Manipulation Language (NDML) Precompiler Development Specification
Section 1 of 5

5. FUNDING NUMBERS
Contract No.: F33600-87-C-0464
PE: 78011F
Proj. No.: 595600
Task No.: P95600
WU: 20950607

6. AUTHOR(S)
J. Althoff, M. Apicella

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Controld Data Corporation
Integration Technology Services
2970 Presidential Drive
Fairborn, OH 45324-6209

8. PERFORMING ORGANIZATION REPORT NUMBER
DS 620341200

9. SPONSORING MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Manufacturing Technology Directorate (WRDC/MTI)
Wright-Patterson AFB, OH 45433-6533

10. SPONSORING/MONITORING AGENCY REP NUMBER
WRDC-TR-90-8007, Vol. V, Part 9
Section 1 of 5

11. SUPPLEMENTARY NOTES
WRDC/MTI Project Priority 6203

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release; Distribution is Unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT

This development Specification (DS) describes the functions, performance, environment, interfaces, and design requirements for the Neutral Data Manipulation Language (NDML) Precompiler. The NDML Precompiler is a component of the Common Data Model Processor (CDMP) and it is used to generate various programs (e.g., request processor or RP, RP drivers, CS-ES transformers, and local subroutine callers) tailored to satisfy the NDML requests in a specific application program.

This report is divided into five (5) sections.

14. SUBJECT TERMS	15. NUMBER OF PAGES	
	885	
	16. PRICE CODE	

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASS OF THIS PAGE.	19. SECURITY CLASS OF ABSTRACT	20. LIMITATION ABSTRACT
SAR	SAR	SAR	SAR

FOREWORD

This technical report covers work performed under Air Force Contract F33600-87-C-0464, DAPro Project. This contract is sponsored by the Manufacturing Technology Directorate, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio. It was administered under the technical direction of Mr. Bruce A. Rasmussen, Branch Chief, Integration Technology Division, Manufacturing Technology Directorate, through Mr. David L. Judson, Project Manager. The Prime Contractor was Integration Technology Services, Software Programs Division, of the Control Data Corporation, Dayton, Ohio, under the direction of Mr. W. A. Osborne. The DAPro Project Manager for Control Data Corporation was Mr. Jimmy P. Maxwell.

The DAPro project was created to continue the development, test, and demonstration of the Integrated Information Support System (IISS). The IISS technology work comprises enhancements to IISS software and the establishment and operation of IISS test bed hardware and communications for developers and users.

The following list names the Control Data Corporation subcontractors and their contributing activities:

SUBCONTRACTOR	ROLE
Control Data Corporation	Responsible for the overall Common Data Model design development and implementation, IISS integration and test, and technology transfer of IISS.
D. Appleton Company	Responsible for providing software information services for the Common Data Model and IDEF1X integration methodology.
ONTEK	Responsible for defining and testing a representative integrated system base in Artificial Intelligence techniques to establish fitness for use.

Simpact Corporation	Responsible for Communication development.
Structural Dynamics Research Corporation	Responsible for User Interfaces, Virtual Terminal Interface,and Network Transaction Manager design, development, implementation, and support.
Arizona State University	Responsible for test bed operations and support.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION 1

SCOPE

1.1 Identification

This specification establishes the performance, development, test, and qualification requirements of a collection of computer programs identified as Configuration Item "Precompiler."

This CI constitutes one of the major subsystems of the "Common Data Model Processor" (CDMP) which is described in the System Design Specification (SDS) for the ICAM Integrated Support System (IISS). The CDMP scope is based on a logical concept of subsystem modules that interface with other external systems of the IISS. The CDMP has been portrayed with three configuration items: the Precompiler, the Distributed Request Supervisor, and the Aggregator. The scope of the CDMP and its configuration items is described in the following narrative.

Common Data Model Processor (CDMP)

Input to the CDMP consists of user transactions in the form of neutral data manipulation language (NDML) commands embedded in COBOL or FORTRAN host programs. NDML commands phrased as stand-alone requests may be supported in future enhancements.

The Precompiler CI parses the application program source code, identifying NDML commands. It applies external-schema-to-conceptual-schema and conceptual-schema-to-internal-schema transforms on the NDML command, thereby decomposing the NDML command into single-database requests. These single-database requests are each transformed into generic DML commands. Programs are generated from the generic DML commands which can access the specific databases to retrieve the data required to evaluate the NDML command. These programs, referred to as Request Processors (RP), are stored at the appropriate host machines. The NDML commands in the application source program are replaced by function calls which when executed, will activate the run-time request evaluation processes associated with the particular NDML command.

The Precompiler also generates a CS/ES Transformer program which will take the final results of the request, stored in a file as a table with conceptual schema structure, and convert the data values into their external schema form.

Finally, the Precompiler generates a Join Query Graph and Result Field Table, which are used by the Distributed Request Supervisor (DRS) during the run-time evaluation of the request.

The DRS CI is responsible for coordination of the run-time activity associated with the evaluation of an NDML command. It is activated by the application program, which sends it the names and locations of the RPs to activate, along with run-time parameters which are to be sent to the RPs. The DRS activates the RPs, sending them the run-time parameters. The results of the RPs are stored as files, in the form of conceptual schema relations, on

the host which executed the RP. Using the Join Query Graph, transmission cost information, and data about intermediate results, the DRS determines the optimal strategy for combining the intermediate results of the NDML command. It issues the appropriate file transfer requests, activates aggregators to perform unions and joins, and activates the appropriate CS/ES Transformer program to transform the final results. Finally, the DRS notifies the application program that the request is completed, and sends it the name of the file which contains the results of the request.

The Aggregator CI is activated by the DRS. An instance of the Aggregator is executed for each join, each outer join, and each union operation performed. It is passed information describing the operation to be performed, and the file names containing the operands of the operation. The DRS ensures that these files already exist on the host which is executing the particular Aggregator program. The Aggregator performs the requested operation, storing the results in a file, whose name was specified by the DRS and which is located on the host executing the Aggregator.

The CDMP provides the application programmer with important capabilities to:

1. Request database accesses in a non-procedural data manipulation language (the NDML) that is independent of the data manipulation language (DML) of any particular Data Base Management System (DBMS),

2. Request database access using a DML that specifies accesses to a set of related records rather than to individual records, i.e., using a relational DML,

3. Request access to data that are distributed across multiple databases with a single DML command, without knowledge of data locations or distribution details.

   Information about external schemas, the conceptual schema, and internal schemas (including data locations) are provided by CDMP access to the Common Data Model (CDM) database. The CDM is a relational database of metadata pertaining to IISS. It is described by the CDM1 information model using IDEF1.

1.2 Functional Summary

The overall objective of this CI is to generate compilable code that will be activated at run-time to access distributed databases and to perform required internal-to-conceptual-to-external transforms. It also produces join query graphs that control the management of run-time transaction processing by the Distributed Request Supervisor CI. The Precompiler CI parses application program source code, identifies NDML commands, applies transformations from external schema form to conceptual schema form, locates requested data, decomposes the commands to

appropriate single-database requests, applies transformations [Bfrom conceptual schema form to internal schema forms, and selects appropriate access paths through the identified databases.

Major functions to be described in this document for this CI are:

Function PRE1	Parcel AP
Function PRE2	Parse Procedure Division
Function CDQCSTK	Control Precompilation Functions
Function PRE3	Parse NDML
Function CDTRANS	Translate "XOR" and "NOT"
Function PRE4	Transform ES/CS
Function PRE5	Decompose CS NDML
Function PRE5A	Distributed Logic Evaluator
Function PRE6	Select IS Access Path
Function PRE7	Transform IS Access Path/Generic DML
Function PRE8	Generate CS/ES Transform
Function PRE8C	Generate CS/CS Transform
Function PRE8D	Generate Referential Integrity CS/CS Transform
Function PRE9.1	Request Processor Support Routines
Function PRE9.2	Generate SQL Request Processor
Function PRE9.3	Generate CODASYL Request Processor
Function PRE9.4	Generate TOTAL Request Processor
Function PRE9.5	Generate IMS Request Processor
Function PRE10	Build Calls and Messages
Function PRE11	Build Source Code
Function PRE12	Control Precompilation
Function PRE13	Control Code Generation
Function PRE14	Generate Request Processor Driver
Function PRE15	Generate Local Subroutine Caller
Function PRE16	Remote Compile and Link

The specific requirements for these functions were identified in the Test Bed System Development Specifications.

SECTION 2

DOCUMENTS

2.1 Applicable Documents

Related ICAM Documents included:

UM620341001	CDM Administrator's Manual
CCS620341000	CDM1, An IDEF1 Model of the Common Data Model
UM620341100	Neutral Data Definition Language (NDDL) User's Guide
PRM620341200	Embedded NDML Programmer's Reference Manual
DS620341200	Development Specification for the IISS NDML Precompiler Configuration Item
DS620341310	Development Specification for the IISS Distributed Request Supervisor Configuration Item
DS620341320	Development Specification for the IISS Aggregator Configuration Item

Other references include:

Cardenas, A.F. and Pirahesh M.H., "Database Communication in a Heterogeneous Database Management System Network," Information Systems, Vol. 5, pp. 55-79, 1980.

Chamberlin, D.D., et. al., "Sequel 2: A Unified Approach to Data Definition, Manipulation, and Control," IBM Journal of Research and Development, Vol. 20, No. 6, November 1976, pp. 560-575.

Date, C.J., A Guide to DB2, Addison-Wesley Publ. Co., 1984.

General Electric Company, Test Bed System Development Specification, November 9, 1982.

Katz, R.H. and Wong, E., "Decompiling CODASYL DML into Relational Queries," ACM Transactions on Database Systems, Vol. 7, No. 1, pp. 1-23, May 1982.

2.2 Terms and Conditions

The following acronyms are used in this document:

APL	Attribute Pair List
AUC	Attribute Use Class
CDMP	Common Data Model Processor

CI	Configuration Item
CS	Conceptual Schema
DML	Data Manipulation Language
DRS	Distributed Request Supervisor (previously SS: Stager/Scheduler)
ES	External Schema
ICAM	Integrated Computer Aided Manufacturing
IS	Internal Schema
NDML	Neutral Data Manipulation Language
RFT	Result Field Table
RP	Request Processor (previously QP: Query Processor)
SDS	System Design Specification

SECTION 3

REQUIREMENTS

3.1 Computer Program Definition

3.1.1 System Capacities

The software for this CI must operate within the available capacity of the target host computer.

3.1.2 Interface Requirements

3.1.2.1 Interface Blocks

This CI generates code that will be executed to provide access to distributed Class II data. Its interfaces, illustrated in Figure 3-1, include input in the form of application source code containing Neutral Data Manipulation Language (NDML) statements and output in the form of generated code (referred to as Request Processors and CS/ES Transformers), modified application source code, and information to guide run-time scheduling of intermediate stages of request processing.

3.1.2.2 Detailed Interface Definition

The specific interface relationships of this CI to other CIs and modules are described in detail for appropriate functions in Section 3.2. The specific interface relationships between the functions of this CI are also described in detail in Section 3.2.

3.1.3 Design/Implementation Differences

This section describes the significant differences between the design of the NDML Precompiler that is documented in this Development Specification and the software that has been produced to implement the Precompiler. This section is not concerned with minor differences, such as the exact structure of tables that are passed from one module to another within the Precompiler.

The entire specification has been updated to reflect the "AS-BUILT" design.

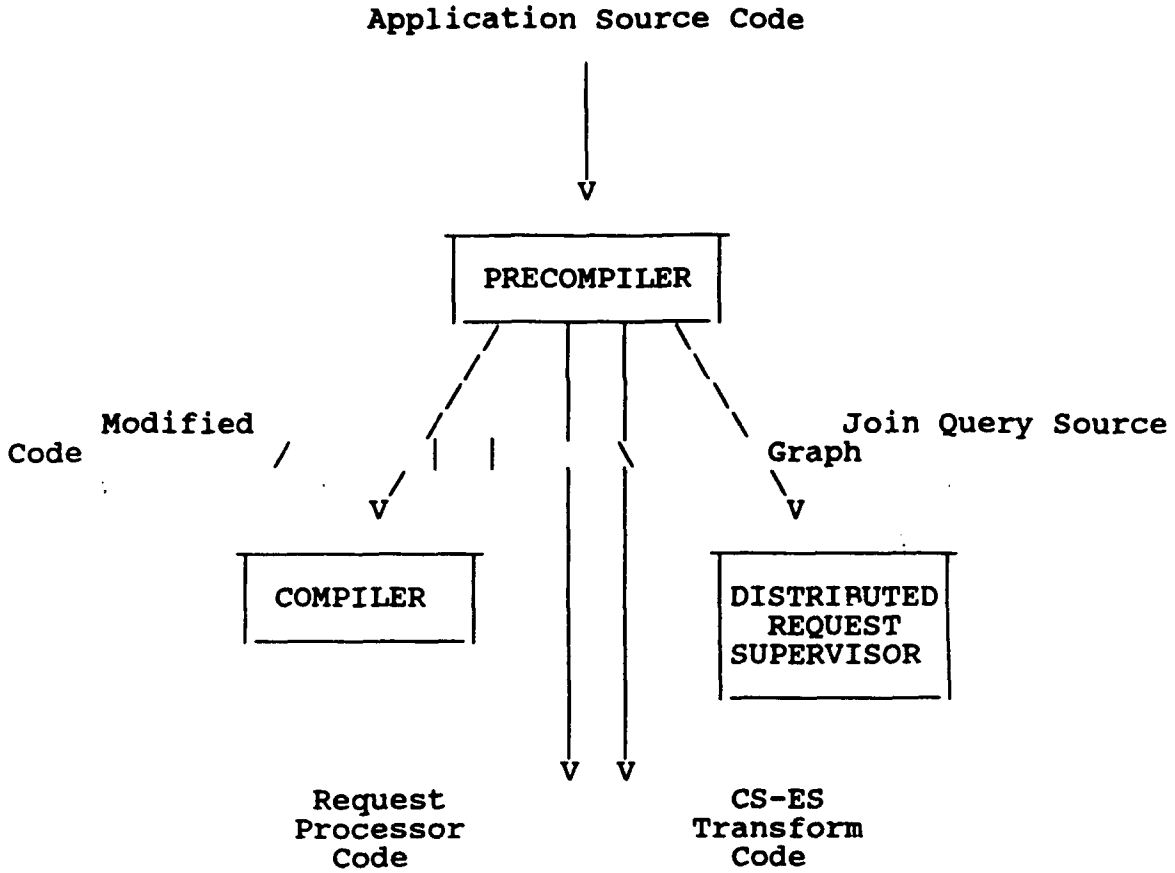


Figure 3-1. Precompiler Interfaces

3.2 Detailed Functional Requirements

The following sections, respectively, document each of the Precompiler's major functions identified in Section 1.2.

3.3 Special Requirements

Principles of structured design and programming will be adhered to.

3.4 Human Performance

Not applicable.

3.5 Database Requirements

The Precompiler programs require access to the CDM database.

3.6 Adaptation Requirements

The system will be implemented at the ICAM IISS Test Bed site located at Arizona State University, Tempe, Arizona. The first Precompiler processes will be implemented on the VAX VMS host.

SECTION 4

FUNCTION PRE1 - PARCEL AP

This function:

1. Extracts the Program ID from the application. This is done so that NDML requests in the AP can be verified for access permission.

2. Partitions an input AP into four parcels that will be added to by other Precompiler components.

   a. Identification parcel receives program description statements from the COBOL Identification and Environment Divisions.

   b. File parcel receives COBOL file declarations and layout statements for result files.

   c. Working storage parcel receives layouts to hold information used in message traffic, variables for received file names, and run-time transformation variables.

   d. Remaining statements comprise the procedure parcel which is handled by Parse Procedure (function PRE2).

4.1 Input

1. A flat file containing a single source program for the user AP. This file is output from PRE12. If the user AP consists of several source programs, they are placed in this file one at a time.

4.2 Processing

1. Create the identification parcel.

   Begin copying all statements from the beginning of the source program into the identification parcel.

   Search for the PROGRAM-ID statement. In addition to copying it into the identification parcel, copy it into the PROGRAM-ID parameter so it can be returned to PRE12.

   Search for any of the following statements to signal the end of the identification parcel and the beginning of the file parcel:

   ```
   INPUT-OUTPUT SECTION
   IO-CONTROL
   DATA DIVISION
   ```

   1.1 If INPUT-OUTPUT SECTION is found, continue copying into the identification parcel while searching for either of the other two statements. When either one is found, stop copying into the identification parcel

and proceed to Step 2. The IO-CONTROL or DATA DIVISION statement becomes the first in the file parcel.

1.2 If either IO-CONTROL or DATA DIVISION is found, stop copying into the identification parcel. Since an INPUT-OUTPUT SECTION statement was not found, generate one and a FILE-CONTROL statement, and place both in the identification parcel. Then proceed to Step 2. The IO-CONTROL or DATA DIVISION statement becomes the first in the file parcel.

2. Create the file parcel

Begin copying all statements from either the IO-CONTROL or DATA DIVISION statement into the file parcel.

Search for any of the following statements to signal the end of the file parcel and the beginning of the working-storage parcel:

```
FILE SECTION
WORKING-STORAGE SECTION
LINKAGE SECTION
PROCEDURE DIVISION
```

2.1 If FILE SECTION is found, continue copying into the file parcel while searching for any of the other three statements.

2.1.1 If WORKING-STORAGE SECTION is found, stop copying into the file parcel and proceed to Step 3. The WORKING-STORAGE SECTION statement becomes the first in the working-storage parcel.

2.1.2 If either LINKAGE SECTION or PROCEDURE DIVISION is found, stop copying into the file parcel. Since a WORKING-STORAGE SECTION was not found, generate one and place it as the only statement in the working-storage parcel. Then proceed to Step 4. The LINKAGE SECTION or PROCEDURE DIVISION statement becomes the first in the procedure parcel.

2.2 If WORKING-STORAGE SECTION, LINKAGE SECTION, or PROCEDURE DIVISION is found, stop copying into the file parcel. Since a FILE SECTION statement was not found, generate one and place it in the file parcel. Then continue processing depending on the following:

2.2.1 If WORKING-STORAGE SECTION was found, proceed to Step 3. The WORKING STORAGE SECTION statement becomes the first in the working-storage parcel.

2.2.2 If either LINKAGE SECTION or PROCEDURE DIVISION was found, generate a WORKING-STORAGE SECTION statement, since one was not found, and place it as the only statement in the working-storage parcel. Then proceed to Step 4. The LINKAGE SECTION or PROCEDURE DIVISION statement becomes the first in the procedure parcel.

3. Create the working-storage parcel.

   Begin copying all statements from the WORKING-STORAGE SECTION statement into the working-storage parcel.

   Search for either of the following statements to signal the end of the working-storage parcel and the beginning of the procedure parcel:

   ```
   LINKAGE SECTION
   PROCEDURE DIVISION
   ```

   When either is found, stop copying to the working-storage parcel and proceed to Step 4. The LINKAGE SECTION or PROCEDURE DIVISION statement becomes the first in the procedure parcel.

4. Create the procedure parcel.

   Copy the LINKAGE SECTION or PROCEDURE DIVISION statement into the procedure parcel. Then return to PRE12; the remainder of the source program will be processed in PRE2.

4.3 Output

1. Identification parcel, which is a flat file containing all the statements from the beginning of the source program until either the IO-CONTROL or DATA DIVISION statement, whichever comes first.

2. File Parcel, which is a flat file containing all the source statements from either the IO-CONTROL or DATA DIVISION statement, whichever comes first, until either the FILE SECTION, WORKING-STORAGE SECTION, LINKAGE SECTION, or PROCEDURE DIVISION statement, whichever comes first.

3. Working-storage parcel, which is a flat file containing all the source statements from the WORKING-STORAGE SECTION statement until either the LINKAGE SECTION or PROCEDURE DIVISION statement, whichever comes first.

4. Procedure parcel, which is a flat file containing only the LINKAGE SECTION or PROCEDURE DIVISION statement, whichever comes first.

5. PROGRAM-ID, which is a parameter for returning the identification of the source program to PRE12.

SECTION 5

FUNCTION PRE2 - PARSE PROCEDURE DIVISION

This function:

1. Appends source code to the procedure division parcel.

2. Extracts NDML clauses from the application source to send to the NDML parser.

   The possible NDML commands and associated clauses are:

Command	Clause
SELECT	SELECT FROM (only with SELECT) WHERE ORDER BY
INSERT	INSERT VALUES
MODIFY	MODIFY USING SET WHERE
DELETE	DELETE USING WHERE
BEGIN	BEGIN
COMMIT	COMMIT
ROLLBACK	ROLLBACK
UNDO	UNDO

3. Identifies the end of an NDML command and suspends operations until all other Precompiler activity on the command completes.

5.1 Input

1. Procedure Division of source program, which PRE1 began dividing into parcels.

2. The following tables, lists, and variables which PRE2 only receives from certain Precompiler modules and passes on to others:

NDML-COUNTER	from PRE12 to PRE4
CODE-GENERATOR-TABLE	from PRE12 to PRE13

5.2 Processing

1. Build the procedure parcel and locate NDML statements.

   Begin copying all remaining statements in the source program into the procedure parcel.

   Search for NDML statements, i.e. those with '*#' in columns 7:8. When one is found, in addition to copying it into the procedure parcel, remove the '*#' and any trailing blanks and process it as follows:

   1.1 If the NDML statement begins a new NDML clause and the NDML buffer is empty, place the NDML statement at the beginning of the buffer.

   1.2 If the NDML statement begins a new NDML clause and the NDML buffer is not empty, invoke the NDML Parser to parse the clause that is already in the buffer.

   When parsing is finished, if any errors were found in the clause, discontinue the precompilation; otherwise, clear the NDML buffer and place the new NDML statement at the beginning of it.

   1.3 If the NDML statement is the continuation of an NDML clause begun in a prior statement, append it to what is already in the NDML buffer.

   1.4 If the NDML statement contains ';' (the NDML terminator) or '(' (the looping construct initiator), invoke the NDML Parser to parse the clause that is already in the NDML buffer.

   When parsing is finished, if any errors were found in the clause, discontinue the precompilation; otherwise, invoke CDQCSTK to begin translating the NDML request and generating source code to satisfy it.

   When CDQCSTK is finished again, clear the NDML buffer and continue searching for NDML statements.

5.3 Output

1. NDML requests, which are sent to NDML Parser to be passed.

2. Procedure parcel, which is the flat file begun in PRE1 and to which the remainder of the source program has been appended.

SECTION 6

FUNCTION CDQCSTK

This function will control the processing of all precompile functions for an NDML command. It determines the type of NDML statement, either single or query combination and then will precompile a given statement or generate code that will perform the query combination (UNION, DIFFERENCE or INTERSECT) of sub-queries.

6.1 Inputs

1. Identification number of the NDML command

   COMMAND-NO

2. Application Program parcel names

   IDFILE-NAME
   FDFILE-NAME
   WORKFILE-NAME
   PROCFILE-NAME

3. Application Program error file name

   ERROR-FILE

4. Source language of the Application Program

   SOURCE-LANGUAGE

5. Host Information about Application Program

   PRECOMPILE-HOST
   AP-TARGET-HOST

6. Code Generator Table

   CODE-GENERATOR-TABLE

7. Last Case Number

   LAST-CASE-NO

8. Logical Unit of Work Name

   LUW-NAME

9. User Module Name

   USER-MODULE-NAME

10. IOS Indicator

    IOS-IND

6.2 CDM Requirements

None

6.3 Internal Requirements

1. External Schema representation of the data

THE EXTERNAL SCHEMA ACTION LIST

```
01  ES-ACTION-LIST.
    03  ES-MAX                          PIC 99 VALUE 50.
    03  ES-USED                         PIC 99 VALUE 0.
    03  ES-NDML-NO                      PIC 999.
    03  ES-ACTION                       PIC X.
        88  ES-MODIFY-ACTION            VALUE "M".
        88  ES-DELETE-ACTION            VALUE "D".
        88  ES-INSERT-ACTION            VALUE "I".
        88  ES-SELECT-ACTION            VALUE "S".
        88  ES-SELECT-COMB              VALUE "Q".
        88  ES-BEGIN-ACTION             VALUE "B".
        88  ES-COMMIT-ACTION            VALUE "C".
        88  ES-ROLLBACK-ACTION          VALUE "R".
        88  ES-NEXT-CONT-ACTION         VALUE "N".
        88  ES-END-CURLEY-ACTION        VALUE "E".
        88  ES-EXIT-BREAK-ACTION        VALUE "X".

    03  ES-DISTINCT-FLAG                PIC X.
        88  ES-DISTINCT                 VALUE "Y".
    03  ES-FILE-NAME                    PIC X(30).
    03  ES-STRUCTURE                    PIC X(30).
    03  ES-SEMI-CURLY-IND               PIC X.
    03  ES-LOCK                         PIC X.
        88  ES-SHARED-LOCK              VALUE "S".
        88  ES-EXCLUSIVE-LOCK           VALUE "X".
        88  ES-NO-LOCK                  VALUE "N".
    03  ES-TABLE-ROW     OCCURS 50 TIMES INDEXED BY ES-INDEX.
        05  ES-DELETE-FLAG              PIC 9.
            88 ES-DELETED               VALUE 1.
        05  ES-UV-ABBR                  PIC XX.
        05  ES-DATA-ITEM                PIC X(30).
        05  ES-VE-USED                  PIC 99.
        05  ES-VALUE-ENTRY OCCURS 5 TIMES.
            07  ES-LOCAL-VARIABLE  PIC X(64).
            07  ES-SUBSCRIPT OCCURS 3 TIMES PIC XXX.
            07  ES-VALUE                PIC X(30).
        05  ES-SORT-SEQUENCE            PIC 99.
        05  ES-SORT-DIRECTION           PIC X.
            88  UP-SORT                 VALUE "A".
            88  DOWN-SORT               VALUE "D".

*
*           NOTE:  A  =  ASCENDING
*                  D  =  DESCENDING
*
```

```
           05  ES-PROJECT-FLAG          PIC X.
               88  TO-BE-PROJECTED      VALUE "Y".
           05  ES-FCTN-NAME             PIC X(5).
               88  APPLY-DISTINCT       VALUE "Y".
           05  ES-CS-PTR                PIC 999.
           05  ES-SOURCE                PIC X.
               88  ES-GENERATED         VALUE "G".
               88  ES-USER              VALUE SPACE.
           05  ES-META.
               07  ES-UV-NO             PIC 9(6).
               07  ES-DI-NO             PIC 9(6).
               07  ES-TYPE              PIC X.
               07  ES-SIZE              PIC 999.
               07  ES-ND                PIC 99.
```

THE EXTERNAL SCHEMA QUALIFY LIST

```
01  ES-QUALIFY-LIST.
    03  ESQ-MAX                        PIC 99 VALUE 50.
    03  ESQ-USED                       PIC 99 VALUE 0.
    03  ES-QUAL-ITEM   OCCURS 50 TIMES INDEXED BY ESQ-INDEX.
        05  ESQ-OP                     PIC XX.
        05  ESQ-LOCAL-VARIABLE         PIC X(64).
        05  ESQ-VALUE                  PIC X(30).
        05  ESQ-SUBSCRIPT OCCURS 3 TIMES PIC XXX.
        05  ESQ-BOOLEAN                PIC X(7).
        05  ESQ-CS-PTR                 PIC 999.
        05  ESQ-FILLER.
            07  ESQ-UV-ABBRL           PIC XX.
            07  ESQ-DATA-ITEML         PIC (30).
            07  ESQ-L-UV-NO            PIC 9(6).
            07  ESQ-L-DI-NO            PIC 9(6).
            07  ESQ-L-TYPE             PIC X.
            07  ESQ-L-SIZE             PIC 999.
            07  ESQ-L-NO               PIC 99.
            07  ESQ-UV-ABBRR           PIC XX.
            07  ESQ-DATA-ITEMR         PIC X(30).
            07  ESQ-R-UV-NO            PIC 9(6).
            07  ESQ-R-DI-NO            PIC 9(6).
            07  ESQ-R-TYPE             PIC X.
            07  ESQ-R-SIZE             PIC 999.
            07  ESQ-R-ND               PIC 99.
```

2. Boolean operators, conditions and parenthesis from the NDML WHERE clause.

BOOLEAN LIST

```
01  BOOLEAN-LIST.
    03  BL-MAX                         PIC 999 VALUE 100.
    03  BL-USED                        PIC 999.
    03  BL-ENTRIES  OCCURS 100 TIMES
                    INDEXED BY BL-INDEX.

        05  BL-OP                      PIC XXX.
        05  BL-ESQ-PTR                 PIC 9(4)  COMP  SYNC.
        05  BL-CSQ-PTR                 PIC 9(4)  COMP  SYNC.
        05  BL-CS-PTR                  PIC 9(4)  COMP  SYNC.
```

```
        05  BL-EVAL-FLAG              PIC 9.
            88 BL-CANNOT-EVALUATE     VALUE 0.
            88 BL-CAN-EVALUATE        VALUE 1 2 3 4.
```

3. User view information

USER VIEW ABBREVIATION LIST

```
01  UV-ABBR-LIST.
    03  UV-MAX                       PIC 99  VALUE 25.
    03  UV-USED                      PIC 99  VALUE 0.
    03  UV-ABBREV-ENTRY  OCCURS 25 TIMES
                  INDEXED BY UV-INDEX
        05  UV-NAME                  PIC X(30).
        05  UV-ABBR                  PIC XX.
        05  UV-NO                    PIC 9(6).
```

4. NDML command nesting information

```
01  NDML-STACK.
    03  NDML-COUNT                   PIC S9(4)  COMP VALUE 0.
    03  STACK-MAX                    PIC S9(4)  VALUE 25.
    03  STACK-USED                   PIC S9(4)  COMP.
    03  STACK-NO  OCCURS 25 TIMES    PIC S9(4)  COMP.
```

5. NDML Query Combination command information

```
01  QUERY-RESULTS-STACK
    03  QRS-MAX                      PIC 99  VALUE 25.
    03  QRS-USED                     PIC 99  VALUE 0.
    03  STACK-TOP                    PIC S9(4) VALUE 0.
    03  FILE-ENTRY OCCURS 25 TIMES.
        05  ES-QUERY-RESULTS-ID      PIC X(4).
        05  CS-QUERY-RESULTS-ID      PIC X(6).

01  OPERATOR-STACK.
    03  OPERATOR-MAX                 PIC 99    VALUE 25.
    03  OPERATOR-USED                PIC 99    VALUE 0.
    03  OPERATOR-STACK-TOP           PIC S9(4) VALUE 0.
    03  OPERATOR-ENTRIES  OCCURS 25 TIMES.
        05  OPERATOR                 PIC X.
        05  OPERATOR-NO              PIC S9(4) COMP SYNC.

01  OPERAND-STACK.
    03  OPERAND-MAX                  PIC 99    VALUE 25.
    03  OPERAND-USED                 PIC 99    VALUE 0.
    03  OPERAND-STACK-TOP            PIC S9(4) VALUE 0.
    03  OPERAND-ENTRIES  OCCURS 25 TIMES.
        05  OPERAND                  PIC S9(4) COMP.
```

6.4 Processing

1. Determine the type of NDML statement to process, either single or query combination.

   Check the contents of "Query Operator List" by calling module "GET_FIRST_SYMB" with the following parameters:

QUERY-OPERATOR-LIST
SYMBOL
ATTRIBUTE
RET-CODE

If no data exists on this list (MODULE-STATUS NOT = 0), then it is a single NDML statement. If data exists on the list, it is a query combination command and processing continues at step 2.

1.1 Populate the external schema precompiler tables by executing function PRE3.

1.2 Translate any "XOR" or "NOT" operators in the "WHERE" clause of the NDML statement by executing function CDTRANS.

1.3 Start the precompilation process for a single NDML statement by executing function PRE4.

1.4 Exit processing from routine CDQCSTK.

2. Determine the type of data in the "Query Operator List". Check the contents of the variable "ATTRIBUTE". If ATTRIBUTE > 0, then "SYMBOL" contains an operator of the NDML query combination command ("(", ")", "JOIN", "UNION", "DIFFERENCE").

2.1 If ATTRIBUTE > 0
store the value of "ATTRIBUTE" in the OPERAND-STACK
continue processing at step 3.

2.2 If ATTRIBUTE < 0

2.2.1 Determine the precedence of the operator according to the following chart

Symbol	Precedence
(	4
INTERSECT	3
UNION, DIFFERENCE	2
)	1
*	5

2.2.2 Store the symbol and symbol precedence in the OPERATOR-STACK (changing the precedence of "4" to 0 before adding it to the stack).

3. Precompile the Query Combination Command by processing all remaining entries in the "QUERY-OPERATOR-LIST". Perform steps 3.1 - 3.3 for each entry in the list. When no more entries exist, continue processing at step 4.

3.1 Obtain the next "SYMBOL" and "ATTRIBUTE" from the "QUERY-OPERATOR-LIST" by calling module "GET_NEXT_SYMBOL" with the following parameters:

QUERY-OPERATOR-LIST
SYMBOL
ATTRIBUTE
RET-CODE

3.2 If ATTRIBUTE > 0
store the value of "ATTRIBUTE" in the OPERAND-STACK
continue processing at step 3.1

3.3 Obtain the precedence of the operator as in step 2.2.1.

3.3.1 If the current operator is greater than one on top of the stack, store the symbol and symbol precedence in the OPERATOR-STACK as in step 2.2.2. Continue processing at step 3.1.

3.3.2 If the current operator is less than one on top of the stack, process all entries in stack until stack is empty or current operator is greater than or on top of the stack.

3.3.2.1 If operand on top of OPERAND-STACK is greater than zero then process an inner SELECT.

Populate the external schema precompiler tables by executing function PRE3.

Translate any "XOR" or "NOT" operators in the "WHERE"clause by executing function CDTRANS.

Start precompilation process for an inner SELECT by executing function PRE4.

Save the ES-ACTION-LIST and CS-ACTION-LIST in temporary list for use during processing of the outer SELECT.

Store the ES-NDML-NO and CS-NDML-NO for this inner SELECT in the QUERY-RESULTS-STACK.

3.3.2.2 Process the next inner SELECT. Perform step 3.3.2.1.

3.3.2.3 Operate on the two previous inner SELECTS.

Pop the top two entries in the QUERY-RESULTS-STACK.

Generate code into the user's application to process the query operator by executing function CDP10S.

Store the identifier of the results in the QUERY-RESULTS-STACK.

Store intermediate results indicated by a negative value in the OPERAND-STACK.

3.2.3 If the current operator is "(" remove it from the opeator stack.

3.2.4 If the current operator does not equal an ")", add it to the OPERATOR-STACK.

4. Process all remaining entries in the OPERAND-STACK by executing steps 3.3.2.1 - 3.3.2.3.

5. Generate code onto the user's application program to process the outer (mapping) SELECT of the query combination command.

5.1 Populate the external schema precompiler tables by executing function PRE3.

5.2 Generate code to perform the final mapping of results from Query Combination command by executing function CDP10T.

6. Set the function status for CDQCSTK and exit processing.

6.5 Outputs

1. Function status indicating if any errors occurred

MODULE-STATUS PIC X(5)

3.1 Obtain the next "SYMBOL" and "ATTRIBUTE" from the "QUERY-OPERATOR-LIST" by calling module "GET_NEXT_SYMBOL" with the following parameters:

QUERY-OPERATOR-LIST
SYMBOL
ATTRIBUTE
RET-CODE

3.2 If ATTRIBUTE > 0
store the value of "ATTRIBUTE" in the OPERAND-STACK
continue processing at step 3.1

3.3 Obtain the precedence of the operator as in step 2.2.1.

3.3.1 If the current operator is greater than one on top of the stack, store the symbol and symbol precedence in the OPERATOR-STACK as in step 2.2.2. Continue processing at step 3.1.

3.3.2 If the current operator is less than one on top of the stack, process all entries in stack until stack is empty or current operator is greater than or on top of the stack.

3.3.2.1 If operand on top of OPERAND-STACK is greater than zero then process an inner SELECT.

Populate the external schema precompiler tables by executing function PRE3.

Translate any "XOR" or "NOT" operators in the "WHERE"clause by executing function CDTRANS.

Start precompilation process for an inner SELECT by executing function PRE4.

Save the ES-ACTION-LIST and CS-ACTION-LIST in temporary list for use during processing of the outer SELECT.

Store the ES-NDML-NO and CS-NDML-NO for this inner SELECT in the QUERY-RESULTS-STACK.

3.3.2.2 Process the next inner SELECT. Perform step 3.3.2.1.

3.3.2.3 Operate on the two previous inner SELECTS.

Pop the top two entries in the QUERY-RESULTS-STACK.

Generate code into the user's application to process the query operator by executing function CDP10S.

Store the identifier of the results in the QUERY-RESULTS-STACK.

Store intermediate results indicated by a negative value in the OPERAND-STACK.

3.2.3 If the current operator is "(" remove it from the opeator stack.

3.2.4 If the current operator does not equal an ")", add it to the OPERATOR-STACK.

4. Process all remaining entries in the OPERAND-STACK by executing steps 3.3.2.1 - 3.3.2.3.

5. Generate code onto the user's application program to process the outer (mapping) SELECT of the query combination command.

5.1 Populate the external schema precompiler tables by executing function PRE3.

5.2 Generate code to perform the final mapping of results from Query Combination command by executing function CDP10T.

6. Set the function status for CDQCSTK and exit processing.

6.5 Outputs

1. Function status indicating if any errors occurred

MODULE-STATUS PIC X(5)

SECTION 7

FUNCTION PRE3 - PARSE NDML

This function populates the external schema data structures from the tokenized items of the NDML clauses.

7.1 Input

1. A complete NDML text clause.

7.2 Processing

For SELECT and ORDER BY clauses the ES-ACTION-LIST is filled in as follows:

ES-NDML-NO	=	blank
ES-ACTION	=	'S'
ES-DISTINCT-FLAG	=	'Y' if SELECT DISTINCT is specified
	=	blank if no DISTINCT clause is specified
ES-FILE-NAME	=	file-name if INTO file-name is specified
	=	blank if INTO STRUCTURE or no INTO clause is specified
ES-STRUCTURE	=	variable-name if INTO STRUCTURE :variable-name is specified
	=	blank if INTO file-name or no INTO clause is specified
ES-LOCK	=	'S' if WITH SHARED LOCK is specified
	=	'X' if WITH EXCLUSIVE LOCK is specified
	=	'N' if WITH NO LOCK or no LOCK clause is specified

Note: One ES-ACTION-ENTRY, consisting of all the following, is filled in for each item in the SELECT list or the ORDER BY list. If the same item is in both lists, only one ES-ACTION-ENTRY is filled in.

ES-UV-ABBR	=	table-label if the item contains table-label.column-name or table-label.ALL.
	=	generated table-label if the item contains table-name.column-name, table-name.ALL, or just column-name or ALL
ES-DATA-ITEM	=	column-name or ALL

Note: Only the first ES-VALUE-ENTRY, consisting of the following three fields, is filled in. All other ES-VALUE-ENTRYs are left blank.

ES-LOCAL-VARIABLE(1)	=	variable-name if the SELECT item contains :variable-name
	=	blank if the SELECT item does not contain :variable-name or if the item is not in the SELECT list
ES-SUBSCRIPT(1,j)	=	integer in position j of the subscript-list if the SELECT item contains :variable-name (subscript-list)
	=	blank if the subscript-list has fewer than j integers or if the SELECT item does not contain a subscript-list or if the item is not in the SELECT list
ES-VALUE(1)	=	blank
ES-SORT-SEQUENCE	=	a positive number indicating the order of sorting, 1 being the most significant sort key field, if the item is in the ORDER BY list
	=	zero if the item is not in the ORDER BY list or if no ORDER BY clause is specified
ES-SORT-DIRECTION	=	'A' if an ascending sort on the ORDER BY item is specified
	=	'D' if a descending sort on the ORDER BY item is specified
	=	blank if the item is not in the ORDER BY list or if no ORDER BY clause is specified
ES-PROJECT-FLAG	=	'Y' if the item is in the SELECT list
	=	blank if the item is in the ORDER BY list but not in the SELECT list
ES-FCTN-NAME	=	func-name if the SELECT item contains a func-name
	=	blank if the SELECT item does not contain a func-name or if the item is not in the SELECT list
ES-FCTN-DISTINCT	=	'Y' if the SELECT item contains DISTINCT
	=	blank if the SELECT item does not contain DISTINCT or if the item is not in the SELECT list
ES-UV-NO	=	blank
ES-DI-NO	=	blank

For INSERT clauses the ES-ACTION-LIST is filled in as follows:

ES-NDML-NO = blank

ES-ACTION = 'I'

ES-DISTINCT-FLAG = blank

ES-FILE-NAME = file-name if FROM file-name is specified
= blank if FROM STRUCTURE or no FROM clause is specified

ES-STRUCTURE = variable-name if FROM STRUCTURE :variable-name is specified
= blank if FROM file-name or no FROM clause is specified.

ES-LOCK = blank

Note: One ES-ACTION-ENTRY, consisting of all the following, is filled in for each item in the INSERT list.

ES-UV-ABBR = same as for SELECT

ES-DATA-ITEM = same as for SELECT

Note: One ES-VALUE-ENTRY, consisting of all the next three fields, is filled in for each value in the VALUES list. The VALUES list may contain more than one set or row of values, each enclosed in parenthesis. In this case, one value from each set is associated with each item in the INSERT list. If the VALUES clause contains a FROM clause instead of a list of values, all ES-VALUE-ENTRYs are left blank.

ES-LOCAL-VARIABLE(i) = variable-name if the value for this INSERT item in set i contains :variable-name
= blank if the value for this INSERT item in set i is a number or a quoted-string

ES-SUBSCRIPT(i,j) = integer in position j of the subscript-list if the value for this INSERT item in set i is :variable-name (sub-script-list)
= blank if the subscript-list has fewer than j integers or if the value for this INSERT item in set i does not contain a subscript-list

ES-VALUE(i) = number if the value for this INSERT item in set i is a number
= string (without quotes) if the value for this INSERT item in set i is a quoted-string

= blank if the value for this INSERT item in set i contains :variable-name

ES-VE-USED = number of rows of insert values

ES-SORT-SEQUENCE = zero

ES-SORT-DIRECTION = blank

ES-PROJECT-FLAG = blank

ES-FCTN-NAME = blank

ES-FCTN-DISTINCT = blank

ES-UV-NO = blank

ES-DI-NO = blank

For MODIFY clauses the ES-ACTION-LIST is filled in as follows:

ES-NDML-NO = blank

ES-ACTION = 'M'

ES-DISTINCT-FLAG = blank

ES-FILE-NAME = blank

ES-STRUCTURE = blank

ES-LOCK = blank

Note: One ES-ACTION-ENTRY, consisting of all the following, is filled in for each item in the MODIFY list.

ES-UV-ABBR = same as for SELECT.

ES-DATA-ITEM = same as for SELECT.

Note: Only the first ES-VALUE-ENTRY, consisting of the following three fields, is filled in. All other ES-VALUE-ENTRYs are left blank.

ES-LOCAL-VARIABLE(1) = variable-name if the MODIFY item contains :variable-name
= blank if the MODIFY item contains a number or a quoted-string

ES-SUBSCRIPT(1,j) = integer in position j of the subscript-list if the MODIFY item contains :variable-name (subscript-list)
= blank if the subscript-list has

fewer than j integers or if the MODIFY item does not contain a subscript-list

ES-VALUE(1) = number if the MODIFY item contains a number
= string (without quotes) if the MODIFY item contains a quoted-string
= blank if the MODIFY item contains :variable-name

ES-SORT-SEQUENCE = zero

ES-SORT-DIRECTION = blank

ES-PROJECT-FLAG = blank

ES-FCTN-NAME = blank

ES-FCTN-DISTINCT = blank

ES-UV-NO = blank

ES-DI-NO = blank

For DELETE clauses the ES-ACTION-LIST is filled in as follows:

ES-NDML-NO = blank

ES-ACTION = 'D'

ES-DISTINCT-FLAG = blank

ES-FILE-NAME = blank

ES-STRUCTURE = blank

ES-LOCK = blank

Note: One ES-ACTION-ENTRY, consisting of all the following, is filled in. All other ES-ACTION-ENTRYs are left blank.

ES-UV-ABBR = same as for SELECT

ES-DATA-ITEM = blank

ES-LOCAL-VARIABLE = blank

ES-SUBSCRIPT = blank

ES-VALUE = blank

ES-SORT-SEQUENCE = zero

ES-SORT-DIRECTION = blank

ES-PROJECT-FLAG = blank

ES-FCTN-NAME = blank

ES-FCTN-DISTINCT = blank

ES-UV-NO = blank

ES-DI-NO = blank

For BEGIN, COMMIT, ROLLBACK, and UNDO clauses only one field in the ES-ACTION-LIST is filled in as follows; all others are left blank.

ES-ACTION = 'B' for a BEGIN clause.
= 'C' for a COMMIT clause.
= 'R' for a ROLLBACK clause or for an UNDO clause.

One UV-ABBR-ENTRY is filled in as follows for each table in the FROM clause of a SELECT command, for each table in the USING clause of a MODIFY command plus the table being modified, or for each table in the USING clause of a DELETE command plus the table from which rows are being deleted. If a table-label is not specified for a table in an NDML command, the parser generates one and records it in the appropriate ES-ACTION-ENTRYs, UV-ABBR-ENTRYs, and ES-QUALIFY-ENTRYs. If ALL is specified without a table-label or table-name in a SELECT list, the parser checks that only one table is included in the FROM clause of that SELECT command. If more than one is included, the parser issues an error for that command.

UV-NAME = table-name.

UV-ABBR = table-label if one is specified for the table.
= generated table-label if a table-label is not specified for the table.

UV-NO = blank.

One or more ES-QUALIFY-ENTRIES are filled in as follows for each column-predicate or join-predicate in a WHERE clause. If a column-predicate is in the form:

value operator column-spec

it is changed into the form:

column-spec operator value

with the operator changing as follows:

from < to >
from <= to >=
from > to <
from >= to <=

The = and != operators are not changed. If WHERE ALL is specified instead of column-predicates or join-predicates, all ES-QUALIFY-ENTRYs are left blank.

If the "BETWEEN" operator is used in the WHERE clause, two entries are added to the ES-QUALIFY-LIST using the following translation logic:

```
column-spec   BETWEEN value1 AND value2

                  becomes

     column-spec > = value1  AND
     column-spec < = value2


column-spec NOT BETWEEN  value1 AND value2

                  becomes

     column-spec < value1 OR
     column-spec > value2
```

ESQ-UV-ABBRL	=	table-label if the left side of the predicate is table-label.column-name
	=	generated table-label if the left side of the predicate is table-name.column-name or just column-name
ESQ-DATA-ITEML	=	column-name from the left side of the predicate
ESQ-L-UV-NO	=	blank
ESQ-L-DI-NO	=	blank
ESQ-OP	=	operator from the predicate, if the operator does not equal BEWTEEN, IS NULL, IS NOT NULL
	=	"NN" if operator from the predicate is IS NOT NULL "NL" if operator from the predicate is IS NULL
	=	operator from the BETWEEN translation logic, if the operator from the predicate is BETWEEN
ESQ-LOCAL-VARIABLE	=	variable-name if the right side of the column-predicate contains :variable-name
	=	blank if the right side of the column-predicate is a number or a quoted-string or if the predicate is a join-predicate

ESQ-SUBSCRIPT(i)	=	integer in position i of the subscript-list if the right side of the column-predicate is :variable-name (subscript-list)
	=	blank if the subscript-list has fewer than i integers or if the value on the right side of the column-predicate does not contain a subscript-list or if the predicate is a join-predicate
ESQ-VALUE	=	number if the right side of the column-predicate is a number
	=	string (without quotes) if the right side of the column-predicate is a quoted-string
	=	blank if the right side of the column-predicate contains :variable-name or if the predicate is a join-predicate
ESQ-UV-ABBRR	=	table-label if the right side of the join-predicate is table-label.column-name
	=	generated table-label if the right side of the join-predicate is table-name.column-name or just column-name
	=	blank if the predicate is a column-predicate
ESQ-DATA-ITEMR	=	column-name from the right side of the join-predicate
	=	blank if the predicate is a column-predicate
ESQ-R-UV-NO	=	blank
ESQ-R-DI-NO	=	blank
ESQ-BOOLEAN	=	blank

One BOOLEAN-ENTRY is filled in as follows for each external qualification criteria (all non-join criteria) which is in following format:

column-spec operator value

BL-OP	=	"(", ")", "AND", "OR", "XOR", "NOT" of the predicate
	=	blank, if BL-ESQ-PTR is filled in
BL-ESQ-PTR	=	entry in the ES-QUALIFY-LIST containing the predicate
	=	blank, if BL-OP is filled in

BL-CSQ-PTR = blank

BL-CS-PTR = blank

When the NDML clause has been parsed, the ES-ACTION-LIST, ES-QUALIFY-LIST, and UV-ABBR-LIST are returned to CDQCSTK.

Perform semantic checks on parsed NDML SELECT requests.

Every time PRE3 finishes parsing an entire SELECT request, ensure that the following statements regarding statistics, sorting, and the disposition of retrieval results are true:

A. Only one of the following is specified as the disposition for the SELECT results

   a file (ES-FILE-NAME not blank)
   a program structure (ES-STRUTURE-NAME not blank)
   program variables (ES-LOCAL-VARIABLE not blank)

B. If a program variable is specified for a column being retrieved (ES-LOCAL-VARIABLE not blank and (ES-PROJECT-FLAG = 'Y'), then one is specified for every such column.

C. If a statistics function is specified for a column (ES-FCTN-NAME not blank), then one is specified for every column.

D. If a statistics function is specified for a column (ES-FCTN-NAME not blank), then sorting is not specified for that column (ES-SORT-SEQUENCE = 0).

E. If a statistics function is specified for a column (ES-FCTN-NAME not blank), then SELECT DISTINCT is not specified for that column (ES-DISTINCT-FLAG not 'Y').

Perform semantic checks on parsed NDML WHERE clause

A. Join criteria (column-spec operator ccolumn-spec) must be ANDed in the WHERE clause

B. Join criteria may not be embedded inside parentheses with non-join criteria

7.3 Output

Parsed lists containing clause tokens:

THE EXTERNAL SCHEMA ACTION LIST

```
01  ES-ACTION-LIST.
    03  ES-MAX                    PIC 99 VALUE 50.
    03  ES-USED                   PIC 99 VALUE 0.
    03  ES-NDML-NO                PIC 999.
    03  ES-ACTION                 PIC X.
```

```
           88  ES-MODIFY-ACTION          VALUE "M".
           88  ES-DELETE-ACTION          VALUE "D".
           88  ES-INSERT-ACTION          VALUE "I".
           88  ES-SELECT-ACTION          VALUE "S".
           88  ES-SELECT-COMB            VALUE "Q".
           88  ES-BEGIN-ACTION           VALUE "B".
           88  ES-COMMIT-ACTION          VALUE "C".
           88  ES-ROLLBACK-ACTION        VALUE "R".
           88  ES-NEXT-CONT-ACTION       VALUE "N".
           88  ES-END-CURLEY-ACTION      VALUE "E".
           88  ES-EXIT-BREAK-ACTION      VALUE "X".
       03  ES-DISTINCT-FLAG              PIC X.
           88  ES-DISTINCT               VALUE "Y".
       03  ES-FILE-NAME                  PIC X(30).
       03  ES-STRUCTURE                  PIC X(30).
       03  ES-SEMI-CURLY-IND             PIC X.
       03  ES-LOCK                       PIC X.
           88  ES-SHARED-LOCK            VALUE "S".
           88  ES-EXCLUSIVE-LOCK         VALUE "X".
           88  ES-NO-LOCK                VALUE "N".
       03  ES-TABLE-ROW      OCCURS 50 TIMES INDEXED BY ES-INDEX.
           05  ES-DELETE-FLAG            PIC 9.
               88 ES-DELETED             VALUE 1.
           05  ES-UV-ABBR                PIC XX.
           05  ES-DATA-ITEM              PIC X(30).
           05  ES-VE-USED                PIC 99.
           05  ES-VALUE-ENTRY OCCURS 5 TIMES.
               07  ES-LOCAL-VARIABLE     PIC X(64).
               07  ES-SUBSCRIPT OCCURS 3 TIMES PIC XXX.
               07  ES-VALUE              PIC X(30).
           05  ES-SORT-SEQUENCE          PIC 99.
           05  ES-SORT-DIRECTION         PIC X.
               88  UP-SORT               VALUE "A".
               88  DOWN-SORT             VALUE "D".

*
*          NOTE:  A  =  ASCENDING
*                 D  =  DESCENDING
*
           05  ES-PROJECT-FLAG           PIC X.
               88  TO-BE-PROJECTED       VALUE "Y".
           05  ES-FCTN-NAME              PIC X(5).
               88  APPLY-DISTINCT        VALUE "Y".
           05  ES-CS-PTR                 PIC 999.
           05  ES-SOURCE                 PIC X.
               88  ES-GENERATED          VALUE "G".
               88  ES-USER               VALUE SPACE.
           05  ES-META.
               07  ES-UV-NO              PIC 9(6).
               07  ES-DI-NO              PIC 9(6).
               07  ES-TYPE               PIC X.
               07  ES-SIZE               PIC 999.
               07  ES-ND                 PIC 99.
```

THE EXTERNAL SCHEMA QUALIFY LIST

```
01  ES-QUALIFY-LIST.
    03  ESQ-MAX                        PIC 99 VALUE 50.
    03  ESQ-USED                       PIC 99 VALUE 0.
    03  ES-QUAL-ITEM   OCCURS 50 TIMES INDEXED BY ESQ-INDEX.
        05  ESQ-OP                     PIC XX.
        05  ESQ-LOCAL-VARIABLE         PIC X(64).
        05  ESQ-VALUE                  PIC X(30).
        05  ESQ-SUBSCRIPT OCCURS 3 TIMES PIC XXX.
        05  ESQ-BOOLEAN                PIC X(7).
        05  ESQ-CS-PTR                 PIC 999.
        05  ESQ-FILLER.
            07  ESQ-UV-ABBRL           PIC XX.
            07  ESQ-DATA-ITEML         PIC (30).
            07  ESQ-L-UV-NO            PIC 9(6).
            07  ESQ-L-DI-NO            PIC 9(6).
            07  ESQ-L-TYPE             PIC X.
            07  ESQ-L-SIZE             PIC 999.
            07  ESQ-L-NO               PIC 99.
            07  ESQ-UV-ABBRR           PIC XX.
            07  ESQ-DATA-ITEMR         PIC X(30).
            07  ESQ-R-UV-NO            PIC 9(6).
            07  ESQ-R-DI-NO            PIC 9(6).
            07  ESQ-R-TYPE             PIC X.
            07  ESQ-R-SIZE             PIC 999.
            07  ESQ-R-ND               PIC 99.
```

USER VIEW ABBREVIATION LIST

```
01   UV-ABBR-LIST.
     03  UV-MAX                  PIC 99 VALUE 25.
     03  UV-USED                 PIC 99 VALUE 0.
     03  UV-ABBREV-ENTRY         OCCURS 25 TIMES INDEXED BY
UV-INDEX.
         05  UV-NAME                      PIC X(30).
         05  UV-ABBR                      PIC XX.
         05  UV-NO                        PIC 9(6).


01   BOOLEAN-LIST.
     03  BL-MAX                           PIC 999 VALUE 100.
     03  BL-USED                          PIC 999.
     03  BL-ENTRIES              OCCURS 100 TIMES INDEXED BY
                                           BL-INDEX.
         05  BL-OP                        PIC XXX.
         05  BL-ESQ-PTR     PIC 9(4)      COMP     SYNC.
         05  BL-CSQ-PTR     PIC 9(4)      COMP     SYNC.
         05  BL-CS-PTR      PIC 9(4)      COMP     SYNC.
         05  BL-EVAL-FLAG   PIC 9.
             88 BL-CANNOT-EVALUATE        VALUE 0.
             88 BL-CAN-EVALUATE           VALUE 1 2 3 4.
```

7.4 Internal Data Requirements

NONE

SECTION 8

FUNCTION CDTRANS - TRANSLATE EXCLUSIVE OR (XOR) AND NOT OPERATORS.

This function will translate the "XOR" and "NOT" operators in the WHERE clause of the NDML statement. It will update both the ES-QUALIFY-LIST and BOOLEAN-LIST to reflect the translation of "XOR" and "NOT" to all "AND" and "OR" operators.

The exclusive OR (XOR) operator will be translated as follows:

```
X.A < 5  XOR X.B =12
```

will be translated to:

```
(X.A < 5 OR X.B = 12) AND (X.A >= 5 OR X.B != 12)
```

The NOT operator will be translated according to De Morgan's Law; operators are reversed, AND becomes OR and OR becomes AND.

8.1 Inputs

1. External schema representation of the WHERE clause.

THE EXTERNAL SCHEMA QUALIFY LIST

```
01  ES-QUALIFY-LIST.
    03  ESQ-MAX                        PIC 99 VALUE 50.
    03  ESQ-USED                       PIC 99 VALUE 0.
    03  ES-QUAL-ITEM   OCCURS 50 TIMES INDEXED BY ESQ-INDEX.
        05  ESQ-OP                     PIC XX.
        05  ESQ-LOCAL-VARIABLE         PIC X(64).
        05  ESQ-VALUE                  PIC X(30).
        05  ESQ-SUBSCRIPT OCCURS 3 TIMES PIC XXX.
        05  ESQ-BOOLEAN                PIC X(7).
        05  ESQ-CS-PTR                 PIC 999.
        05  ESQ-FILLER.
            07  ESQ-UV-ABBRL           PIC XX.
            07  ESQ-DATA-ITEML         PIC (30).
            07  ESQ-L-UV-NO            PIC 9(6).
            07  ESQ-L-DI-NO            PIC 9(6).
            07  ESQ-L-TYPE             PIC X.
            07  ESQ-L-SIZE             PIC 999.
            07  ESQ-L-NO               PIC 99.
            07  ESQ-UV-ABBRR           PIC XX.
            07  ESQ-DATA-ITEMR         PIC X(30).
            07  ESQ-R-UV-NO            PIC 9(6).
            07  ESQ-R-DI-NO            PIC 9(6).
            07  ESQ-R-TYPE             PIC X.
            07  ESQ-R-SIZE             PIC 999.
            07  ESQ-R-ND               PIC 99.
```

```
01   BOOLEAN-LIST.
     03  BL-MAX                              PIC 999 VALUE 100.
     03  BL-USED                             PIC 999.
     03  BL-ENTRIES                 OCCURS 100 TIMES INDEXED BY
                                                BL-INDEX.
         05  BL-OP                           PIC XXX.
         05  BL-ESQ-PTR      PIC 9(4)        COMP     SYNC.
         05  BL-CSQ-PTR      PIC 9(4)        COMP     SYNC.
         05  BL-CS-PTR       PIC 9(4)        COMP     SYNC.
         05  BL-EVAL-FLAG    PIC 9.
             88 BL-CANNOT-EVALUATE           VALUE 0.
             88 BL-CAN-EVALUATE              VALUE 1 2 3 4.
```

8.2 CDM Requirements

NONE

8.3 Internal Requirements

NONE

8.4 Processing

1. Initialize local variables

2. Translate all "XOR" entries in the BOOLEAN-LIST. When all entries have been processed (BL-INDEX > BL-USED), continue processing at step 3.

   2.1 If BL-OP (BL-INDEX) NOT = "XOR", continue processing at step 2 with the next BOOLEAN-LIST entry. If BL-OP (BL-INDEX) = "XOR":

      2.1.1 Position back to the beginning of the expression on the left of the "XOR" operator and get the size of the left expression. Save this position as the beginning position of "XOR" expression.

         2.1.1.1 Beginning of left expression can be indicated by one of three conditions:

            1) Beginning of the Boolean list

            2) If right parentheses are found, then finding matching left parentheses

            3) First non-"NOT" Boolean operator when RIGHT-PAREN-CNT equals LEFT-PAREN-CNT

            Note: Include all immediately preceding NOTs when left expression is found.

2.1.1.2 Size of left expression is equal to the position of "XOR" minus the beginning of left expression

2.1.2 Position forward to the end of the expression to the right of "XOR" operator and get the size of right expression. Save the position as the end position of "XOR" expression.

2.1.2.1 End of right expression can be indicated by one of three conditions:

1) End of Boolean list

2) If left parentheses are found, then finding matching right parentheses

3) First not-"NOT" Boolean operator when LEFT-PAREN-CNT equals RIGHT-PAREN-CNT

Note: Include all NOTs when searching for end of right expression

2.1.2.2 Size of right expression is equal to the end of left expression minus the position of "XOR".

2.1.3 Gap size = BL-MAX - BL-USED

2.1.4 Check if gap size >= expression size (left + right + 7). If not, signal BL-ENTRY overflowed error, set the return status and exit.

2.1.5 Open the gap in BL-ENTRY from the beginning position of expression, i.e. move all BL-ENTRYs starting with the beginning position of expression to the last (BL-USED) position to TEMP-BL-ENTRY.

2.1.6 Set and save the top position of gap = beginning position of expression

2.1.7 Set the beginning and end position of the "XOR" operator in TEMP-BL-ENTRY. Also set TEMP-BL-ENTRY-MAX.

```
beginning = 1
end       = previous end - previous
beginning + 1
TEMP-MAX  = BL-USED - previous
            beginning + 1
```

2.1.8 Put operator "(" to the top of gap; increment the top of gap

2.1.9 Copy each entry of BL-ENTRY from beginning position to end position of expression of operator "XOR" to the top of gap. Increment top of gap after each copy.

2.1.10 Put operator ")" to the top of gap; increment the top of gap

2.1.11 Set the position of "XOR" operator = saved top position of gap + size of left expression + 1

2.1.12 Change operator "XOR" to "OR"

2.1.13 Put operator "AND" to the top of gap; increment the top of gap

2.1.14 Put operator "NOT" to the top of gap; increment the top of gap

2.1.15 Save the top position of gap

2.1.16 Put operator "(" to the top of gap; increment the top of gap

2.1.17 Loop through each entry of BL-ENTRY from beginning position to end position of expression of operator "XOR" to gap

2.1.17.1 Copy each entry of ES-QUALIFY-LIST pointed to by BL-ESQ-PTR to the rear of ES-QUALIFY-LIST, incrementing ESQ-USED first.

2.1.17.2 Copy entry of BL-ENTRY to the top of gap, increment top of gap after each copy and reset the BL-ESQ-PTR to the new entry created at 2.1.17.1

2.1.18 Put operator ")" to the top of gap; increment the top of gap

2.1.19 Set the position of "XOR" operator = saved top position of gap + size of left expression + 1

2.1.20 Change operator "XOR" to "AND"

2.1.21 Move TEMP-BL-ENTRY from end of expression + 1 to TEMP-MAX to top of gap. After each move, increment the top of gap.

2.1.22 Update BL-USED = current top of gap - 1

2.1.23 Continue processing at step 2

3. Translate all "NOT" entries in the BOOLEAN-LIST. When all entries have been processed, (BL-INDEX > BL-USED) continue processing at step 4.

   3.1 If BL-OP (BL-INDEX) NOT = "NOT" continue processing at step 3 with the next BOOLEAN-LIST entry

      3.1.1 Delete "NOT" by removing it from the BOOLEAN LIST

      3.1.2 Convert "AND" to "OR" and "OR" to "AND" from beginning to the end of the "NOT" expression

      3.1.3 Convert every operator in ES-QUALIFY-LIST to the opposite operator, i.e. "=" becomes "!=", ">" becomes ">=" ... etc. from the beginning to the end of the "NOT" expression

      3.1.4 Continue processing at step 4

4. Exit CDTRANS

8.5 Outputs

1. Updated external schema representation of the WHERE clause with no "XOR" or "NOT" entries

   ES-QUALIFY-LIST
   BOOLEAN-LIST

SECTION 9

FUNCTION PRE4 - TRANSFORM ES/CS

The external-schema-to-conceptual-schema transformer converts an NDML request expressed in external schema terms into one or more NDML requests expressed in conceptual schema terms.

The conversion involves translating each user view into the corresponding entity classes and each data item into the corresponding attribute use classes.

It also involves identifying the relational join operations that are needed to construct each user view table from the entity class tables and identifying the integrity tests that will be employed with each NDML update request. This implementation will not support derived data items.

9.1 Inputs

1. CDM Metadata

   The entity classes needed are:

Alpha-Numeric Data Format	=	ANDF	(E234)
Attribute Class Data Description	=	ACDD	(E184)
Attribute Use Class	=	AUC	(E5)
AUC-DI Mapping	=	AUCDIM	(E64)
Data Format	=	DF	(E233)
Data Item	=	DI	(E16)
EC-UV Join	=	ECUVJ	(E79)
Inherited Attribute Class	=	IAC	(E7)
Key Class	=	KC	(E3)
Key Class Member	=	KCM	(E6)
Numeric Data Format	=	NDF	(E235)
Relation Class	=	RC	(E4)
User View	=	UV	(E15)

2. The NDML external schema request to be transformed

   This is output from the parser function PRE3 and includes:

   ES-ACTION-LIST
   UV-ABBR-LIST
   ESQ-QUALIFY-LIST

3. NDML-COUNTER

   This counter is used by PRE4 to create a unique case number for each NDML command in the user AP. It is supplied by the main function, which retrieved the last used NDML-COUNTER from the CDM for the logical unit of work being precomplied. PRE4 increments it

for each NDML command and returns it to CDQCSTK when PRE4 ends. CDQCSTK does not change it during the precompilation of a user AP.

4. Parcels 1 through 4

   PARCEL1, PARCEL2, PARCEL3 AND PARCEL4 contain names of files which contain the partitioned user module.

5. ERROR-FILE

   The file to which error messages are generated.

6. MY-HOST

   The host name upon whih CDPRE4 runs.

7. TARGET-HOST

   The host upon which the user application will run.

8. SOURCE-LANGUAGE

   Language in which the user application is written.

9. CODE-GENERATOR-TABLE

   Information regarding precompiler generated code.

10. IOS-IND

    Information regarding the presence or absence of an INPUT OUTPUT section in the user application.

11. USER-MOD-ID

    Name of the user's subroutine being precompiled, as identified by PRE1.

12. BOOLEAN-LIST

    Information regarding column versus literal or variable qualifications.

9.2 Processing

1. Fill in CURRENT-CS with the model number for the current version of the conceptual schema.

2. For each UV-ABBR-ENTRY fill in UV-NO with VIEW-NO from the UV (E15) entry that has VIEW-NAME = UV-NAME. If no such UV entry is found, reject the NDML statement (nonexistent user view). If DISTINCT_IND from USER_VIEW is "Y" but the user has not specified a "distinct" function, set the ES_DISTINCT-FLAG to "Y".

3. If ES-ACTION = 'S', transform each ES-ACTION-ENTRY by doing the following:

   3.1 Fill in ES-UV-NO with UV-NO from the UV-ABBR-ENTRY that has UV-ABBR = ES-UV-ABBR.

   3.2 If ES-DATA-ITEM = 'ALL', find all the DI (E16) entries that have VIEW-NO = ES-UV-NO, and fill in a new ES-ACTION-ENTRY for each as follows:

ES-UV-ABBR	=	ES-UV-ABBR in the ES-ACTION-ENTRY that contains 'ALL'
ES-DATA-ITEM	=	DI-NAME in the DI (E16) entry
ES-PROJECT-FLAG	=	'Y'

   All other fields in the new ES-ACTION-ENTRYs are left blank or zero. The existing entry containing 'ALL' is replaced by the first ES-DATA-ITEM. The remainder of Step 3 is done for each ES-ACTION-ENTRY that is filled in.

   3.3 Deposit external metadata for each external data item into the current ES-TYPE, ES-SIZE, ES-ND, and ES-DI-NO. If the data item does not exist, reject the NDML statement (non-existent data item).

   3.4 If ES-FCTN-NAME = 'SUM', 'AVG', or 'MEAN' and DT-CODE in the DI entry indicates a non-numeric data type, reject the NDML statement (function requires numeric data).

   3.5 If ES-FCTN-NAME = 'MIN' or 'MAX' and ES-FCTN-DISTINCT = 'Y', reset ES-FCTN-DISTINCT to blank.

   3.6 Verify that the data item is not derived and extract the tag number mapped to by the current data item.

   3.7 This step deleted.

   3.8 Extract conceptual metadata and entity class number given the tag number.

   3.9 This step was deleted. Its function is performed by CDPRE2.

   3.10 This step was deleted. The CE-WORK-LIST has been dropped.

   3.11 Fill in a CS-ACTION-ENTRY as follows:

CS-ECNO	=	EC-NO in the AUC entry
CS-AUC	=	AUC-NO in the AUC entry
CS-TYPE	=	type from 3.8
CS-SIZE	=	size from 3.8
CS-ND	=	ND from 3.8

```
CS-ES-PTR          =  ES-ACTION-LIST index
CS-LOCAL-VARIABLE  =  'ES-A-ndml-index' where:
                         ndml  = NDML-counter
                         index = ES-INDEX
CS-FCTN-NAME       =  ES-FCTN-NAME
CS-FCTN-DISTINCT   =  ES-FCTN-DISTINCT
CS-SOURCE          =  blank
CS-DELETE-FLAG     =  zero
```

3.12 This step was deleted.

4. If ES-ACTION = 'I', do the following:

4.1 Transform each ES-ACTION-ENTRY by doing the following:

4.1.1 Fill in ES-UV-NO with UV-NO from the UV-ABBR-ENTRY that has UV-ABBR = ES-UV-ABBR. Same as Step 3.1.

4.1.1a Verify that the view is not mapped to more than one entity. If so, reject the NDML statement. (User view maps to multiple entity classes.)

4.1.2 Populate external metadata and ES-DI-NO.

For each used ES-DATA-ITEM, extract external metadata and data item number by calling CDEMD with the following parameters.

INPUTS

USER-VIEW-NO
DATA-ITEM-NAME
ERROR-FILE

OUTPUTS

DI-NO
ETYPE
ESIZE
E-ND
RET-STATUS

Populate the current ES-DI-NO with DI-NO, the current ES-TYPE with ETYPE, the current ES-SIZE with ESIZE and the current ES-ND with E-ND.

4.1.3 For each ES-VALUE entry whose index is less than or equal to ES-VE-USED (ES-INDEX) which has the corresponding ES-LOCAL-VARIABLE equal to spaces and the corresponding ES-TYPE equal to I,F, N, P or S call CDVNV to insure that the value is numeric:

INPUTS

ES-VALUE
ES-DATA-ITEM
ERROR FILE

OUTPUT

ERROR-STATUS

4.1.4 Reject any derived data items. Same as Step 3.6.

4.1.5 This step deleted.

4.1.6 Extract conceptual metadata given the tag number. Same as Step 3.8.

4.1.7 Begin filling in a TEMP-XFORM-ENTRY as follows:

TEMP-EC-NO	=	EC-NO from 4.1.6
TEMP-AUC	=	TAG NUMBER from 4.1.4
TEMP-TYPE	=	TYPE from 4.1.6
TEMP-SIZE	=	SIZE from 4.1.6
TEMP-ND	=	ND from 4.1.6
TEMP-ES-PTR	=	ES-ACTION-LIST index
TEMP-LOCAL-VARIABLE	=	ES-A-ndml-esindex where: ndml = NDML-COUNTER esindex = ES-ACTION-LIST index

4.1.8 Find the owner tag and relation class, if any, given the current TEMP-AUC.

4.1.9 If an owner tag is found, finish filling in the TEMP-XFORM-ENTRY as follows:

TEMP-RC-NO	=	RC-NO from the previous step
TEMP-KCM-AUC-NO	=	owner tag from the previous step

Otherwise, finish filling in the entry by setting both of these to zero.

4.2 This step was moved to step 4.1.1a.

4.3 If there is an AUC (E5) entry that has MODEL-NO = CURRENT-CS and EC-NO = TEMP-EC-NO (1) but does not have AUC-NO = TEMP-AUC in any TEMP-XFORM-ENTRY, reject the NDML statement (user view maps to partial entity class).

4.4 For each RC (E4) entry that has MODEL-NO = CURRENT-CS and DEP-EC-NO = TEMP-EC-NO (1), if any, set up a Type 1 referential integrity test by doing the following:

4.4.1 Increment NDML-COUNTER.

4.4.2a For the first TEMP-XFORM-ENTRY that has TEMP-RC-NO = RC-NO in the RC entry, fill in a CS-ACTION-ENTRY as follows:

CS-LOCK	=	'S'
CS-NDML-NO	=	NDML-COUNTER
CS-ACTION	=	'1' (for Type 1 referential integrity test)
CS-ECNO	=	IND-EC-NO in the RC entry
CS-AUC	=	TEMP-KCM-AUC-NO
CS-ES-PTR	=	zero
CS-LOCAL-VARIABLE	=	blank
CS-FCTN-NAME	=	blank
CS-FCTN-DISTINCT	=	blank
CS-DELETE-FLAG	=	zero
CS-SOURCE	=	blank

4.4.2b Extract the conceptual metadata for the TEMP-KCM-TAG-NO.

CS-TYPE	=	type
CS-SIZE	=	size
CS-ND	=	ND

4.4.3a For each TEMP-XFORM-ENTRY that has TEMP-RC-NO = RC-NO in the RC entry, fill in a CS-QUALIFY-ENTRY as follows:

CSQ-NDML-NO	=	NDML-COUNTER
CSQ-ECNOL	=	IND-EC-NO in the RC entry
CSQ-AUCL	=	TEMP-KCM-AUC-NO
CSQ-OP	=	'='
CSQ-VARIABLE	=	TEMP-LOCAL-VARIABLE
CSQ-ECNOR	=	zero
CSQ-AUCR	=	zero
CSQ-BOOLEAN	=	'AND'
CSQ-ES-PTR	=	TEMP-ES-PTR
CSQ-R-TYPE	=	blank
CSQ-R-SIZE	=	zero
CSQ-R-ND	=	zero
CSQ-RCNOR	=	zero

4.4.3b Extract conceptual metadata for the CSQ-AUCL.

CSQ-L-TYPE	= type
CSQ-L-SIZE	= size
CSQ-L-ND	= ND

4.4.3c Eliminate duplicate CS-QUALIFY entries. Same as step 10.

4.4.4 Leave CSQ-BOOLEAN blank in the last CS-QUALIFY-ENTRY that is created for each RC entry.

4.4.4a Call CDMQAL to populate the CS-ACTION-LIST with any AUC's which are not already represented there to support conceptual evaluation of those data fields not internally evaluatable.

4.4.4b Call CDPBL to populate the LOCAL-BOOLEAN-LIST.

4.4.5 Invoke PRE5 to transform the Type 1 referential integrity test from CS to IS.

4.5 For each KC (E3) entry that has MODEL-NO = CURRENT-CS and EC-NO = TEMP-EC-NO (1), set up a key uniqueness test by doing the following:

4.5.1 Increment NDML-COUNTER.

4.5.2a Fill in a CS-ACTION-ENTRY as follows:

CS-LOCK	= 'S'
CS-NDML-NO	= NDML-COUNTER
CS-ACTION	= 'K' (for key uniqueness test)
CS-ECNO	= TEMP-EC-NO (1)
CS-AUC	= KCM-AUC-NO from the first KCM (E6) entry with the same MODEL-NO and KC-NO as the KC entry
CS-ES-PTR	= zero
CS-LOCAL-VARIABLE	= blank
CS-FCTN-NAME	= blank
CS-FCTN-DISTINCT	= blank
CS-DELETE-FLAG	= zero
CS-SOURCE	= blank

4.5.2b Extract conceptual metadata for the KCM-AUC-NO

CS-TYPE	= type
CS-SIZE	= size
CS-ND	= nd

4.5.3a For each KCM (E6) entry that has MODEL-NO = CURRENT-CS and the same KC-NO as the KC entry, fill in a CS-QUALIFY-ENTRY as follows:

CSQ-NDML-NO	=	NDML-COUNTER
CSQ-ECNOL	=	TEMP-EC-NO (1)
CSQ-AUCL	=	AUC-NO in the KCM entry
CSQ-OP	=	'='
CSQ-VARIABLE	=	TEMP-LOCAL-VARIABLE in the TEMP-XFORM-LIST entry that has TEMP-AUC = AUC-NO in the KCM entry
CSQ-ECNOR	=	zero
CSQ-AUCR	=	zero
CSQ-BOOLEAN	=	'AND'
CSQ-R-TYPE	=	blank
CSQ-R-SIZE	=	zero
CSQ-R-ND	=	zero
CSQ-RCNOR	=	zero
CSQ-SOURCE	=	space

4.5.3b Extract conceptual metadata for the CSQ-AUCL.

CSQ-L-TYPE	=	type
CSQ-L-SIZE	=	size
CSQ-L-ND	=	ND

4.5.3c Eliminate duplicate CS-QUALIFY entries. Same as step 10.

4.5.4 Leave CSQ-BOOLEAN blank in the last CS-QUALIFY-ENTRY for each KC entry.

4.5.4a Call CDMQAL to populate the CS-ACTION-LIST with any AUC's which are not already represented there to support conceptual evaluation of those data fields not internally evaluatable.

4.5.4b Call CDPBL to populate the LOCAL-BOOLEAN-LIST.

4.5.5 Invoke PRE5 to transform the key uniqueness test from CS to IS.

4.6 Set up the insertion by doing the following:

4.6.1 This step deleted.

4.6.2 For each TEMP-XFORM-ENTRY fill in a CS-ACTION-ENTRY as follows:

```
CS-ECNO                 =  TEMP-EC-NO
CS-AUC                  =  TEMP-AUC
CS-TYPE                 =  TEMP-TYPE
CS-SIZE                 =  TEMP-SIZE
CS-ND                   =  TEMP-ND
CS-ES-PTR               =  TEMP-ES-PTR
CS-LOCAL-VARIABLE       =  TEMP-LOCAL-
                           VARIABLE
CS-FCTN-NAME            =  blank
CS-FCTN-DISTINCT        =  blank
CS-SOURCE               =  blank
CS-DELETE-FLAG          =  zero
IF ES-ACTION not        =  delete
ES-CS-PTR (CS-ES-PTR (CS-INDEX)) =
    CS-INDEX
```

5. If ES-ACTION = 'M', do the following:

5.1 Transform each ES-ACTION-ENTRY by doing the following:

5.1.1 Fill in ES-UV-NO with the UV-NO from the UV-ABBR-ENTRY that has UV-ABBR = ES-UV-ABBR. Same as Step 3.1

5.1.1a Reject NDML statement if view maps to more than one entity class. Same as step 4.1.1a.

5.1.2 Populate external metadata and ES-DI-NO.

For each used ES-DATA-ITEM, extract external metadata and data item number by calling CDEMD with the following parameters:

INPUTS:

USER-VIEW-NO
DATA-ITEM-NAME
ERROR-FILE

OUTPUTS:

DI-NO
ETYPE
ESIZE
E-ND
RET-STATUS

Populate the current ES-DI-NO with DI-NO, the current ES-TYPE with ETYPE, the current ES-SIZE with ESIZE and the current ES-ND with E-ND.

5.1.3 For each ES-VALUE entry whose index is less than or equal to ES-VE-USED (ES-INDEX) which has the corresponding ES-LOCAL-VARIABLE equal to spaces and

which has the corresponding ES-TYPE equal to I, F, N, P or S, call CDVNV to insure that the value is numeric. Call CDVNV with the following parameters:

INPUTS:

ES-VALUE
ES-DATA-ITEM
ERROR-FILE

OUTPUTS:

ERROR-STATUS

5.1.4 Reject any derived data items. Same as Step 3.6.

5.1.5 This step deleted.

5.1.6 Extract conceptual metadata given the tag number. Same as Step 3.8

5.1.7 Begin filling in a TEMP-XFORM-ENTRY. Same as Step 4.1.7.

5.1.8 Find owner tag and relation class, if any. Same as Step 4.1.8.

5.1.9 Update the current TEMP-XFORM row. Same as Step 4.1.9.

5.2 This step moved to step 5.1.1a.

5.3 If there is any KCM (E6) entry that has MODEL-NO = CURRENT-CS and AUC-NO = TEMP-AUC in a TEMP-XFORM-ENTRY, reject the NDML statement (modification of key class member).

5.4 If there is any IAC (E7) entry that has MODEL-NO = CURRENT-CS and RC-NO = TEMP-RC-NO (other than zero) in a TEMP-XFORM-ENTRY but does not have AUC-NO = TEMP-AUC in that TEMP-XFORM-ENTRY, reject the NDML statement (modification of partial inherited key class).

5.5 For each RC (E4) entry that has MODEL-NO = CURRENT-CS and RC-NO = TEMP-RC-NO (other than zero) in a TEMP-XFORM-ENTRY, if any, set up a Type 1 referential integrity test by doing the following:

5.5.1 Increment NDML-COUNTER. Same as 4.4.1.

5.5.2a Fill in a CS-ACTION entry for the first TEMP-XFORM-ENTRY whose TEMP-RC-NO matches the RC-NO from 5.5. Same as 4.4.2a.

5.5.2b Extract conceptual metadata for the TEMP-KCM-TAG-NO. Same as 4.4.2b.

5.5.3a Fill in a CS-QUALIFY ENTRY for each TEMP-XFORM-ENTRY whose TEMP-RC-NO matches the RC-NO from 5.5. Same as 4.4.3a.

5.5.3b Extract conceptual metadata for the CSQ-AUCL. Same as 4.4.3b.

5.5.3c Eliminate duplicate CS-QUALIFY entries. Same as step 10.

5.5.4 Leave CSQ-BOOLEAN blank in the last CS-QUALIFY entry for each RC entry. Same as 4.4.4.

5.5.4a Move AUC's from the CS-QUALIFY list to the CS-ACTION list. Same as 4.4.4a.

5.5.4b Populate the local boolean list. Same as 4.4.4b.

5.5.5 Transform the Type 1 Referential Integrity Test from conceptual to internal. Same as 4.4.5.

5.6 Set up the modification by doing the following:

5.6.1 This step deleted.

5.6.2 Fill in a CS-ACTION entry for each TEMP-XFORM-ENTRY. Same as 4.6.2.

6. If ES-ACTION = 'D', do the following:

6.1 Fill in ES-UV-NO (1) from UV-NO (1).

6.2 Reject the NDML statement if it maps to multiple entity classes. Same as step 4.1.1a.

6.3 This step was moved to 6.2.

6.4 For each AUC that has MODEL-NO = CURRENT-CS and VIEW-NO = ES-UV-NO (1), do the following:

6.4.1 This step deleted.

6.4.2 Extract conceptual metadata and entity class number. Same as Step 3.8.

6.2.2a This step deleted.

6.4.3 Begin filling in a TEMP-XFORM-ENTRY. Same as Step 4.1.7.

6.4a If a "using" clause appeared in the Delete statement (UV-USED is greater than 1), call CDVJUV with the following parameters to verify that the target user view is joined.

INPUT:

UV-ABBR-LIST
ES-QUALIFY-LIST
ERROR-FILE

OUTPUT:

ERROR-STATUS

If the ERROR-STATUS returns with a non-zero value, exit.

6.4b Transform each ES-QUALIFY entry, if any, by doing the following:

6.4.b1 Fill in ESQ-L-UV-NO. Same as Step 8.1.

6.4.b2 Fill in ESQ-L-DI-NO and ESQ-L-TYPE, ESQ-L-SIZE and ESQ-L-ND from the CDM. Same as Step 8.2.

6.4.b3 Verify that the ESQ-VALUE is numeric if ESQ-L-TYPE is numeric. Same as Step 8.3.

6.4.b4 Verify that the data item is not derived and extract the tag number mapped to by the current data item. Same as Step 8.4.

6.4.b5 Extract conceptual metadata for the tag number extracted in the previous step. Same as Step 8.5.

6.4.b6 Begin filling in a CS-QUALIFY entry.

CSQ-ECNOL	=	EC-NO from Step 6.4.b5
CSQ-AUCL	=	TAG-NO from Step 6.4.b4
CSQ-OP	=	ESQ-OP
CSQ-ECNOR	=	ZERO
CSQ-AUCR	=	ZERO
CSQ-BOOLEAN	=	ESQ-BOOLEAN
CSQ-L-TYPE	=	TYPE from Step 6.4.b5
CSQ-L-SIZE	=	SIZE from Step 6.4.b5
CSQ-L-ND	=	ND from Step 6.4.b5
CSQ-R-TYPE	=	BLANK
CSQ-R-SIZE	=	ZERO
CSQ-R-ND	=	ZERO
ESQ-CS-PTR	=	CSQ-INDEX
CSQ-RCNOR	=	ZERO
CSQ-ES-PTR	=	ESQ-INDEX
ESQ-R-UV-NO	=	ZERO
ESQ-R-DI-NO	=	ZERO

ESQ-R-TYPE = BLANK
ESQ-R-SIZE = ZERO
ESQ-R-ND = ZERO

If ESQ-UV-ABBRR is blank, move

ES-Q-ndml-esqindex to CSQ-VARIABLE where ndml is the NDML-COUNTER and esqindex is the current ESQ-INDEX.

IF ESQ-UV-ABBRR is not blank, move spaces to CSQ-VARIABLE.

6.4.b7 If ESQ-UV-ABBRR is not blank, do the following.

6.4.b.7.1 Fill in ESQ-R-UV-NO. Same as Step 8.7.1.

6.4.b.7.2 Fill in ESQ-R-DI-NO and ESQ-R-TYPE, ESQ-R-SIZE and ESQ-R-ND from the CDM. Same as Step 8.7.2.

6.4.b.7.3 Verify that both or neither ESQ-L-TYPE and ESQ-R-TYPE are character; otherwise, reject the NDML statement (incompatable qualify data types). Same as Step 8.7.3.

6.4.b.7.4 Verify that ESQ-DATA-ITEMR is not derived. Same as 8.7.4.

6.4.b.7.5 Extract conceptual metadata for the tag mapped to by ESQ-DATA-ITEMR. Same as Step 8.7.5.

6.4.b.7.6 Continue filling in a CSQ-ENTRY.

CSQ-ECNOR = EC-NO from Step 6.4.b.7.5

CSQ-AUCR = TAG-NO from Step 6.4.b.7.4

CSQ-R-TYPE = TYPE from Step 6.4.b.7.5

CSQ-R-SIZE = SIZE from Step 6.4.b.7.5

CSQ-R-ND = ND from Step 6.4.b.7.5

CSQ-SOURCE = U (USER ENTERE

6.4.b.7.7 If CSQ-OP is U= (Outer join), extract the RC-NO from INHERITED_ATT_USE where the tag number equals CSQ-AUCR. Same as Step 8.7.7.

CSQ-RCNOR = RC-NO

6.4.c Remove duplicate CS-QUALIFY entries. Same as Step 10.

6.4.d Call CDGTV to add type 2 qualifications to the CS-QUALIFY and add to the BOOLEAN-LIST as follows:

6.4.d.1 Select type 2 qualifications from the USER_VIEW. (AUC OP VARIABLE)

6.4.d.2 Force the last CSQ-BOOLEAN entry to "AND".

6.4.d.3 Begin filling in CS-QUALIFY entry.

CSQ-ECNOR = ZERO
CSQ-AUCR = ZERO
CSQ-ES-PTR = ZERO
CSQ-RCNOR = ZERO
CSQ-R-SIZE = ZERO
CSQ-R-ND = ZERO
CSQ-R-TYPE = SPACE

6.4.d.4 If the type 2 qualificatiion selected in step 6.4.d.1 is a tag number:

6.4.d.4.1 Retrieve the entity number for the tag from Attribute_Use_Class.

6.4.d.4.2 Extract the conceptual metadata for the tag number.

6.4.d.4.3 Continue filling in the CS-QUALIFY:

CSQ-ECNOL = EC-NO from step 6.4.d.4.1
CSQ-AUCR = AUC from step 6.4.d.1
CSQ-LTYPE = Type from step 6.4.d.4.2
CSQ-L-SIZE = Size from step 6.4.d.4.2
CSQ-L-ND = Number decimals from step 6.4.d.4.2
CSQ-SOURCE = V 6.4.d.4.4
Fill in the BOOLEAN-LIST
BL-CSQ-PTR = CSQ-USED

BL-OP = SPACES
BL-EVAL-FLAG = SPACES
BL-CS-PTR = ZEROS
BL-ESQ-PTR = ZEROS

6.4.d.5 If the type 2 qualification selected in step 6.4.d.1 is a logical operator. Fill in the BOOLEAN-LIST as follows:

BL-OP = Logical operator selected in step 6.4.d.1
BL-ESQ-PTR = ZERO
BL-CSQ-PTR = ZERO
BL-CS-PTR = ZERO
BL-EVAL-FLAG = ZERO

6.4.d.5.a If the logical operator is an "AND" or "OR", fill in the CS-QUALIFY:
CSQ-BOOLEAN = Logical operator

6.4.d.6 If the type 2 qualification selected i step 6.4.d.1 is a comparison operator, continue filling in the CSQ-QUALIFY.

CSQ-OP = Comparison operator

6.4.d.7 If the type 2 qualification selected is a literal constant or a numeric constant, continue filling in the CS-QUALIFY.

CSQ-VARIABLE = Numeric or literal constant

6.5 For each RC (E4) entry that has MODEL-NO = CURRENT-CS and IND-EC-NO = TEMP-EC-NO (1), if any, set up a Type 2 referential integrity test by doing the following:

6.5.1 Increment NDML-COUNTER.

6.5.2 For the first IAC (E7) entry that has MODEL-NO = CURRENT-CS and the same RC-NO as the RC entry, fill in a CS-ACTION-ENTRY as follows:

CS-LOCK	=	'S'
CS-NDML-NO	=	NDML-COUNTER
CS-ACTION	=	'2' (for Type 2 referential integrity test)
CS-ECNO	=	DEP-EC-NO in the RC entry
CS-AUC	=	AUC-NO in the IAC entry
CS-TYPE	=	Type of CS-AUC

CS-SIZE	=	Size of CS-AUC
CS-ND	=	ND of CS-AUC
CS-ES-PTR	=	ZERO
CS-LOCAL-VARIABLE	=	blank
CS-FCTN-NAME	=	blank
CS-FCTN-DISTINCT	=	blank
CS-DELETE-FLAG	=	ZERO
CS-SOURCE	=	blank

6.5.3 For each IAC entry that has MODEL-NO = CURRENT-CS and the same RC-NO as the RC entry, fill in a new CS-QUALIFY-ENTRY as follows:

CSQ-NDML-NO	=	NDML-COUNTER
CSQ-ECNOL	=	DEP-EC-NO in the RC entry
CSQ-AUCL	=	AUC-NO in the IAC entry
CSQ-OP	=	'='
CSQ-VARIABLE	=	blank
CSQ-ECNOR	=	IND-EC-NO in the RC entry
CSQ-AUCR	=	KCM-AUC-NO in the IAC entry
CSQ-BOOLEAN	=	'AND'
CSQ-ES-PTR	=	ZERO
CSQ-RCNOR	=	ZERO
CSQ-L-TYPE	=	Type of CSQ-AUCL
CSQ-L-SIZE	=	Size of CSQ-AUCL
CSQ-L-ND	=	ND of CSQ-AUCL
CSQ-R-TYPE	=	Type of CSQ-AUCR
CSQ-R-SIZE	=	Size of CSQ-AUCR
CSQ-R-ND	=	ND of CSQ-AUCR
CSQ-SOURCE	=	'U'

6.5.4 Step moved to 6.4.b

6.5.5 Eliminate duplicate CS-QUALIFY entries. Same as Step 10.

6.5.6 Leave CSQ-BOOLEAN blank in the last CSQ-QUALIFY-ENTRY for each RC entry.

6.5.6A Call CDMQAL to populate the CS-ACTION-LIST with any AUC's which are not already represented there to support conceptual evaluation of those data fields not internally evaluatable.

6.5.6B Call CDPBL to populate the BOOLEAN-LIST.

6.5.7 Invoke PRE5 to transform the Type 2 referential integrity test from CS to IS.

6.6 Set up the deletion by doing the following:

6.6.1 This step deleted.

6.6.2 Fill in a CS-ACTION-entry for each TEMP-XFORM-ENTRY. Same as 4.6.2.

7. If a Select and no qualifications are entered, generate a warning to the user and continue (CPR 00093).

8. Transform each ES-QUALIFY entry, if Select or Modify.

8.01 If processing a Modify and a using clause appeared in the NDML statement, call CDJUV with the following parameters to verify the target user view is joined.

INPUTS:

UV-ABBR-LIST
ES-QUALIFY-LIST
ERROR-FILE

OUTPUTS:

ERROR-STATUS

If the ERROR-STATUS returns with a non-zero value, exit.

Perform the following steps for each ES-QUALIFY entry.

8.1 Fill in ESQ-L-UV-NO with UV-NO from the UV-ABBR-LIST entry that has UV-ABBR = ESQ-UV-ABBRL.

8.2 Fill in ESQ-L-DI-NO with DI-NO from the DI (E16) entry that has VIEW-NO = ESQ-L-UV-NO and DI-NAME = ESQ-DATA-ITEML. If no such DI entry is found, reject the NDML statement (nonexistent data item).

8.3 For each ESQ entry which has ESQ-L-TYPE equal to I, P, N, F or S and which has the corresponding ESQ-UV-ABBRR and ESQ-LOCAL-VARIABLE equal to spaces, call CDVNV to verify the ESQ-VALUE is numeric. Call CDVNV with the following parameters:

INPUTS:

ESQ-VALUE
ESQ-DATA-ITEML
ERROR-FILE

OUTPUTS:

ERROR-STATUS

8.4 Verify that ESQ-DATA-ITEML is not derived and extract the tag number which is mapped to.

8.5 Extract conceptual metadata and entity class number for the tag number in the previous step.

8.6 Begin filling in a CS-QUALIFY-ENTRY as follows:

```
CSQ-ECNOL       =  EC-NO from Step 8.5
CSQ-AUCL        =  TAG-NO from Step 8.4
CSQ-OP          =  ESQ-OP
CSQ-ECNOR       =  ZERO
CSQ-AUCR        =  ZERO
CSQ-BOOLEAN     =  ESQ-BOOLEAN
CSQ-L-TYPE      =  TYPE from Step 8.5
CSQ-L-SIZE      =  SIZE from Step 8.5
CSQ-L-ND        =  ND from Step 8.5
CSQ-R-TYPE      =  BLANK
CSQ-R-SIZE      =  ZERO
CSQ-R-ND        =  ZERO
ESQ-CS-PTR      =  CSQ-INDEX
CSQ-SOURCE      =  'U'
CSQ-RCNOR       =  ZERO
CSQ-ES-PTR      =  ESQ-INDEX
ESQ-R-UV-NO     =  ZERO
ESQ-R-DI-NO     =  ZERO
ESQ-R-TYPE      =  BLANK
ESQ-R-SIZE      =  ZERO
ESQ-R-ND        =  ZERO
```

If ESQ-UV-ABBRR is blank, move ES-Q-ndml-esqindex to CSQ-VARIABLE

Where ndml is the NDML-COUNTER and esqindex is the current ESQ-INDEX.

If ESQ-UV-ABBRR is not blank, move spaces to CSQ-VARIABLE.

8.7 If ESQ-UV-ABBRR is filled in, do the following:

8.7.1 Fill in ESQ-R-UV-NO with UV-NO from the UV-ABBR-ENTRY that has UV-ABBR = ESQ-UV-ABBRR.

8.7.2 Fill in conceptual metadata for ESQ-DATA-ITEMR and extract the data item number.

```
ESQ-R-DI-NO   =  DATA ITEM NUMER
ESQ-R-TYPE    =  TYPE
ESQ-R-SIZE    =  SIZE
ESQ-R-ND      =  ND
```

8.7.3 If either ESQ-L-TYPE is character and ESQ-R-TYPE is not character or ESQ-R-TYPE is character and ESQ-L-TYPE is not character, reject the NDML statement (incompatable qualify data types).

8.7.4 Verify that ESQ-DATA-ITEMR is not derived and extract the tag to which it maps.

8.7.5 Extract conceptual metadata and entity class number for the tag in the previous step.

8.7.6 Continue filling in the CS-QUALIFY-ENTRY as follows:

CSQ-ECNOR = EC-NO from Step 8.7.5

CSQ-AUCR = TAG-NO from Step 8.7.4

CSQ-R-TYPE = Type from Step 8.7.5

CSQ-R-SIZE = Size from Step 8.7.5

CSQ-R-ND = ND from Step 8.7.5

8.7.7 If CSQ-OP is U= (Outer join), extract the RC-NO from INHERITED_ATT_USE where the tag number equals CSQ-AUCR.

CSQ-RCNOR = RC-NO

8.7.8 Remove duplicate CS-QUALIFY entries. Same as Step 10.

8.02 If ES-ACTION = "M", "S" or "Q", call CDGTVW to select the type 2 qualifications from the USER-VIEW and build the CS-QUALIFY and BOOLEAN-LIST. Same as step 6.4.d.

9. If ES-ACTION = "S" or "Q" for each UV-ABBR-ENTRY set up any additional join operations that are needed to compose the user view referenced in that entry by doing the following:

9.1 Find all the ECUVJ (E79) entries that have MODEL-NO = CURRENT-CS and VIEW-NO = UV-NO. If no such ECUVJ entries are found, this user view is not the result of any join operations and can be ignored for the remainder of Step 9.

9.2 For each ECUVJ entry that is found do the following:

9.2.1 Select all type 3 qualifications from USER_VIEW on the CDM in the form of tag number operator tag number.

9.2.1.a For each pair of tag numbers selected in step 9.2.1, determine from INHERITED_ATT_USE

which is the independent tag number and which is the dependent tag number and retrieve the RC number (RCNO) for that combination.

9.2.1.b Retrieve the dependent entity and independent entity from RELATION_CLASS for that RCNO.

9.2.2 Fill in a TEMP-JOIN-ENTRY as follows:

```
TEMP-IND-EC  =  IND-EC-NO in the RC entry
TEMP-DEP-EC  =  DEP-EC-NO in the RC entry
TEMP-IND-TAG =  Independent tag from
                step 9.2.1.a
TEMP-DEP-TAG =  Dependent tag from
                step 9.2.1.a
TEMP-RC      =  RC-NO in the RC entry
TEMP-JOIN    =  Operator from step 9.2.1
```

Note: The TEMP-JOIN-LIST should not contain entries for more than one user view at a time. After it has been loaded from the RC entries for one user view, Steps 9.3 - 9.5 should be performed. Then it should be emptied before it is used for the next user view.

9.3 Find all the TEMP-JOIN-ENTRYs that reference "leaf" entity classes. A leaf entity class is one whose EC-NO appears in only one TEMP-JOIN-ENTRY, in either TEMP-IND-EC or TEMP-DEP-EC. It may not appear in TEMP-IND-EC in one entry and in TEMP-DEP-EC in another.

9.4 Eliminate each TEMP-JOIN-ENTRY from Step 9.3 whose leaf EC-NO does not appear in any of the following:

CS-ECNO in any CS-COLUMN-ENTRY
CSQ-ECNOL in any CS-QUALIFY-ENTRY
CSQ-ECNOR in any CS-QUALIFY-ENTRY

The elimination of a TEMP-JOIN-ENTRY may cause another entity class to now qualify as a leaf, so if any entries are eliminated, return to Step 9.3 to re-examine all the entries that remain.

9.5 (Having determined that each leaf entity class that is referenced in a remaining TEMP-JOIN-ENTRY is also referenced in at least one CS-ACTION-ENTRY or one CS-QUALIFY-ENTRY) for each remaining TEMP-JOIN-ENTRY,if any, do the following:

9.5.1 If the CS-QUALIFY-LIST is not empty and if its last entry has CSQ-BOOLEAN = blank, move 'AND' to CSQ-BOOLEAN in the last entry.

9.5.2 For each IAC entry that has MODEL-NO = CURRENT-CS and RC-NO = TEMP-RC, fill in a CS-QUALIFY-ENTRY as follows:

CSQ-ECNOL	=	TEMP-IND-EC
CSQ-AUCL	=	KCM-AUC-NO in the IAC entry
CSQ-OP	=	'='
CSQ-VARIABLE	=	blank
CSQ-ECNOR	=	TEMP-DEP-EC
CSQ-AUCR	=	AUC-NO in the IAC entry
CSQ-BOOLEAN	=	'AND'
CSQ-RCNOR	=	ZERO or RCNO
CSQ-SOURCE	=	"V" from step 9.2.1.a if this is an outer join

9.5.3 Extract conceptual metadata for CSQ-AUCL.

9.5.4 Extract conceptual metadata for CSQ-AUCR.

10. Eliminate any duplicate CS-QUALIFY entries satisfying the following requirements:

```
     CSQ-OP      (I)       =  '='                 and
     CSQ-OP      (I)       =  CSQ-OP       (J)  and
     CSQ-ECNOR   (I)  not  =  0                   and

either   (CSQ-ECNOL   (I)  =  CSQ-ECNOL   (J)  and
          CSQ-AUCL    (I)  =  CSQ-AUCL    (J)  and
          CSQ-ECNOR   (I)  =  CSQ-ECNOR   (J)  and
          CSQ-AUCR    (I)  =  CSQ-AUCR    (J))

or       (CSQ-ECNOL   (I)  =  CSQ-ECNOR   (J)  and
          CSQ-AUCL    (I)  =  CSQ-AUCR    (J)  and
          CSQ-ECNOR   (I)  =  CSQ-ECNOL   (J)  and
          CSQ-AUCR    (I)  =  CSQ-AUCL    (J))
```

11. Leave CSQ-BOOLEAN blank in the last CS-QUALIFY-ENTRY.

12. Transform the NDML request from CS to IS.

```
12.1  If   ES-ACTION    =  Q then
           CS-ACTION    =  S
      Else
           CS-ACTION    =  ES-ACTION

      NDML-COUNTER =  NDML-COUNTER +1
      CS-NDML-NO   =  NDML-COUNTER
```

If Select or Select combination or delete or modify,

Call CDMQAL to populate the ES and CS action lists with any data items or AUC's which aren't already represented there to support conceptual qualification of thoses data fields not internally evaluatable.

12.2 If Select or Select combination or delete or modify,

Call CDPBL to populate BL-CS-PTR and BL-CSQ-PTR.

12.3 Complete filling in CS ACTION and CS QUALIFY lists.

If CSQ-USED > 0 then
CSQ-NDML-NO = NDML-COUNTER

Insert the current NDML-COUNTER into each populated CSQ-VARIABLE and CS-LOCAL-VARIABLE.

If Select,

CS-LOCK = ES-LOCK

If INSERT, MODIFY or DELETE,

CS-LOCK = X

12.4 Invoke PRE5 to transform the NDML request from CS to IS.

13. Upon completion of precompilation by PRE5, update the CDM cross reference.

13.1 Delete Action

If the ES-ACTION is delete, store a new occurrence of VU (E280) by calling CDIDIU with the following parameters.

INPUTS:

USER-MOD-ID
DI-NO
VIEW-NO
USAGE-CODE

OUTPUTS:

RET-STATUS

The DI-NO should contain zero. The VIEW-NO should contain the value in UV-NO (1) from the UV-ABBR-LIST. The USAGE-CODE should contain D.

13.2 Insert, Modify, Select or Select Combination Actions

For ES-ACTIONs of insert, modify, select, or select combination store new occurences of DIU (E281).

For each ES-ACTION entry with ES-SOURCE not = 'G', call CDIDIU with the following parameter:

INPUTS:

USER-MOD-ID
DI-NO
VIEW-NO
USAGE-CODE

OUTPUTS:

RET-STATUS

The DI-NO should contain the current ES-DI-NO. The VIEW-NO should contain zero. The USAGE-CODE should contain the ES-ACTION.

13.3 Select, Modify, Delete, or Select Combination Actions

For each ESQ-L-DI-NO and ESQ-R-DI-NO not equal zero, call CDIDIU with the following parameters:

INPUTS:

USER-MOD-ID
DI-NO
VIEW-NO
USAGE-CODE

OUTPUTS:

RET-STATUS

The DI-NO should contain either the ESQ-L-DI-NO or ESQ-R-DI-NO. The VIEW-NO should contain zero. The USAGE-CODE should contain Q.

9.3 Outputs

1. The NDML conceptual schema request represented by the CS-ACTION-LIST and CS-QUALIFY-LIST.

   These will be input to the CS NDML Decomposer. The CS-ACTION-LIST is also input to the CS/ES Transform Generator and the Call and Message Builder.

2. Metadata describing the ES/CS transform.

These metadata are input to the CS/ES Transform Generator (PRE8) and Call and Message Builder (PRE10). They include:

a. ES-ACTION-LIST - PRE3 creates this list. PRE4 adds user view and data item numbers to it.

b. ES-QUALIFY-LIST - PRE3 creates this list. PRE4 adds user view and data item numbers to it.

3. NDML-COUNTER (returned to MAIN)

4. CDM Metadata

View Usage = VU (E280)
Data Item Usage = DIU (E281)

5. RET-STATUS

Completion Status.

9.4 Internal Data Requirements

The following table is used in Steps 4-6 to temporarily store metadata about an NDML update request expressed in conceptual schema terms.

```
01 TEMP-XFORM-LIST.
   03 TEMP-XFORM-ENTRY OCCURS ?? TIMES.
      05 TEMP-EC-NO           PIC 9(5).
      05 TEMP-AUC             PIC 9(6).
      05 TEMP-TYPE            PIC X.
      05 TEMP-SIZE            PIC 9(3).
      05 TEMP-ND              PIC 9(2).
      05 TEMP-ES-PTR          PIC 9(2).
      05 TEMP-LOCAL-VARIABLE  PIC X(30).
      05 TEMP-RC-NO           PIC 9(5).
      05 TEMP-KCM-AUC-NO      PIC 9(6).
```

The following table is used in Step 9 to temporarily store metadata about joins that may have to be performed.

```
01 TEMP-JOIN-LIST.
   03 TEMP-JOIN-ENTRY OCCURS ?? TIMES.
      05 TEMP-IND-EC          PIC 9(5).
      05 TEMP-IND-TAG         PIC S9(4) comp.
      05 TEMP-DEP-EC          PIC 9(5).
      05 TEMP-DEP-TAG         PIC S9(4) comp.
      05 TEMP-RC              PIC X(30).
      05 TEMP-JOIN            PIC XX.
```

Neither table is input to nor output from this function.

SECTION 10

FUNCTION CDVJUV - VERIFY JOIN TO TARGET USER VIEW

This routine verifies that there is at least one type 3 qualification referencing the target table. This routine should be called only for delete and modify actions which employ the using clause.

10.1 Inputs

1. UV-ABBR-LIST

   UV-ABBR-LIST contains information about the user views referenced in an NDML statement.

2. ES-QUALIFY-LIST

   ES-QUALIFY-LIST contains the external representation of the WHERE clause.

3. ERROR-FILE PIC X(30)

   ERROR-FILE contains the name of the file to which error messages are generated.

10.2 CDM Requirements

None

10.3 Internal Requirements

None

10.4 Processing

1. Return if no using clause in the NDML statement.

   If UV-USED equals 1, exit the program with ERROR-STATUS equal zero.

2. Search for a match between UV-ABBR (1) and any used ESQ-UV-ABBRL. If not found, go to step 3, otherwise perform the following steps:

2.1 If the corresponding ESQ-UV-ABBRR equals spaces, go back to step 2 and continue the search.

2.2 If the corresponding ESQ-UV-ABBRR equals ESQ-UV-ABBRL, go back to step 2 and continue the search.

2.3 Exit the program with ERROR-STATUS equal zero.

3. Search for a match between UV-ABBR(1) and any used ESQ-UV-ABBRR.

3.1 If a match is not found, set ERROR-STATUS to 1, generate the following error message using RPTERR, increment USER-ERROR-COUNT by 1 and exit the program.

uv-name NOT IN JOIN CRITERIA

where uv-name contains the value of UV-NAME (1).

3.2 If a match is found and the corresponding ESQ-UV-ABBRL equals ESQ-UV-ABBRR, return to step 3 and continue the search.

3.3 If a match is found and the corresponding ESQ-UV-ABBRL does not equal ESQ-UV-ABBRR, exit the program with ERROR-STATUS equal to zero.

10.5 Outputs

1. USER-ERROR-COUNT PIC 9(5)

USER-ERROR-COUNT contains the count of user errors encountered.

2. ERROR-STATUS PIC 9

ERROR-STATUS contains the return status for this module. Zero indicates success; 1 indicates failure.

SECTION 11

FUNCTION CDVNV - VERIFY NUMERIC VALUE

This routine validates that a character string contains a numeric value.

11.1 Inputs

1. VALUE-IN PIC X(30)

   VALUE-IN contains the value to be checked.

2. DATA-ITEM PIC X(30)

   DATA-ITEM contains the name of the dataitem to be compared against the value.

3. ERROR-FILE PIC X(30)

   ERROR-FILE contains the file name to which error messages will be generated.

11.2 CDM Requirements

None

11.3 Internal Requirements

None

11.4 Processing

1. Validate that VALUE-IN satisfies the following rules:

   1.1 A sign, if present, must be either "+" or "-" and must immediately preceed a decimal digit or decimal point.

   1.2 The number including sign, may begin at any character position, as long as there are no embedded blanks.

   1.3 There may be, at most, 1 decimal point.

   1.4 A decimal number must either preceed or follow the decimal point. A decimal number may both preceed and follow the decimal point.

   1.5 At least 1 decimal digit must appear in the number.

2. If any of the above rules are violated, set ERROR-STATUS to 1 and generate the following message:

   DATA-ITEM must be compared with a numeric value.

3. Terminate processing.

11.5 Outputs

1. ERROR-STATUS PIC 9

   ERROR-STATUS indicator contains zero if VALUE-IN is numeric and 1 if not numeric.

SECTION 12

FUNCTION CDMQAL - BUILD ES/CS ACTION LIST ENTRIES

This routine places ES-QUALIFY and/or CS-QUALIFY entries which are not represented in the ES-ACTION-LIST and CS-ACTION-LIST respectively on those lists in support of conceptual evaluation of those qualify entries not internally evaluated.

12.1 Inputs

1. ES-ACTION-LIST
2. ES-QUALIFY-LIST
3. CS-ACTION-LIST
4. CS-QUALIFY-LIST

12.2 CDM Requirements

None

12.3 Internal Requirements

None

12.4 Processing

1. If processing a select, query combination, type 1 referential integrity test, type 2 referential integrity test, modify, delete or a key uniqueness test, add new CS-ACTION entries.

   1.1 Scan the CS-QUALIFY-LIST. For each used CSQ-ECNOL/CSQ-AUCL combination which matches no used CS-ECNO/CS-AUC combination, add a new CS-ACTION entry.

   1.1.1 Add 1 to CS-USED.

   1.1.2 If CS-USED is greater than CS-MAX, generate a fatal error message and exit.

   1.1.3 Populate the following CS-ACTION items:

CS-DELETE-FLAG	= zero
CS-ECNO	= CSQ-ECNOL
CS-AUC	= CSQ-AUCL
CS-TYPE	= CSQ-L-TYPE
CS-SIZE	= CSQ-L-SIZE
CS-ND	= CSQ-L-ND
CS-LOCAL-VARIABLE	= blank
CS-FCTN-NAME	= blank

```
CS-FCTN-DISTINCT        = blank
CS-SOURCE               = G
CS-ES-PTR               = zero
```

1.2 Scan the CS-QUALIFY-LIST again. For each used non-zero CSQ-ECNOR/CSQ-AUCR combination which matches no used CS-ECNO/CS-AUC combination, add a new CS-ACTION entry.

1.2.1 Add 1 TO CS-USED.

1.2.2 If CS-USED is greater than CS-MAX, generate a fatal error message and exit.

1.2.3 Populate the following CS-ACTION items.

```
CS-DELETE-FLAG   = zero
CS-ECNO          = CSQ-ECNOR
CS-AUC           = CSQ-AUCR
CS-TYPE          = CSQ-R-TYPE
CS-SIZE          = CSQ-R-SIZE
CS-ND            = CSQ-R-ND
CS-LOCAL-VARIABLE = blank
CS-FCTN-NAME     = blank
CS-FCTN-DISTINCT  = blank
CS-SOURCE        = G
CS-ES-PTR        = zero
```

2. If processing a select or query combination, add new ES-ACTION entries.

2.1 Scan the ES-QUALIFY-LIST. For each used ESQ-L-UV-NO/ESQ-L-DI-NO combination which matches no used ES-UV-NO/ES-DI-NO combination, add a new ES-ACTION entry.

2.1.1 Add 1 TO ES-USED.

2.1.2 If ES-USED is greater than ES-MAX, generate a fatal error message and exit.

2.1.3 Populate the following ES-ACTION items.

```
ES-DELETE-FLAG              = zero
ES-UV-ABBR                  = ESQ-UV-ABBRL
ES-DATA-ITEM                = ESQ-DATA-ITEML
ES-VE-USED                   = zero
ES-VALUE-ENTRYs 1 through 5 = blank
ES-SORT-SEQUENCE             = zero
ES-SORT-DIRECTION           = blank
ES-PROJECT-FLAG             = N
ES-FCTN-NAME                = blank
ES-FCTN-DISTINCT            = blank
ES-UV-NO                    = ESQ-L-UV-NO
ES-DI-NO                    = ESQ-L-DI-NO
ES-TYPE                     = ESQ-L-TYPE
```

```
ES-SIZE                    = ESQ-L-SIZE
ES-ND                      = ESQ-L-ND
ES-SOURCE                  = G
```

2.1.4 Populate the current ES-CS-PTR and the corresponding CS-ES-PTR.

2.1.4.1 Look at the CSQ-ECNOL/CSQ-AUCL combination pointed to by the current ESQ-CS-PTR.

2.1.4.2 Scan the CS-ACTION-LIST looking for a match between the CSQ-ECNOL/CSQ-AUCL combination found in the previous step and a CS-ECNO/CS-AUC combination.

2.1.4.3 When a match is found, populate both pointers.

```
ES-CS-PTR  =  CS-INDEX
CS-ES-PTR  =  ES-INDEX
```

2.2 Scan the ES-QUALIFY-LIST again. For each used non-zero ESQ-R-UV-NO/ESQ-R-DI-NO combination which matches no used ES-UV-NO/ES-DI-NO combination, add a new ES-ACTION entry.

2.2.1 Add 1 to ES-USED.

2.2.2 If ES-USED is greater than ES-MAX, generate a fatal error message and exit.

2.2.3 Populate the following ES-ACTION items.

```
ES-DELETE-FLAG          = zero
ES-UV-ABBR              = ESQ-UV-ABBRR
ES-DATA-ITEM            = ESQ-DATA-ITEMR
ES-VE-USED              = zero
ES-VALUE-ENTRYs 1 through 5 = blank
ES-SORT-SEQUENCE        = zero
ES-SORT-DIRECTION       = zero
ES-PROJECT-FLAG         = N
ES-FCTN-NAME            = blank
ES-FCTN-DISTINCT        = blank
ES-UV-NO                = ESQ-R-UV-NO
ES-DI-NO                = ESQ-R-DI-NO
ES-TYPE                 = ESQ-R-TYPE
ES-SIZE                 = ESQ-R-SIZE
ES-ND                   = ESQ-R-ND
ES-SOURCE               = G
```

2.2.4 Populate the current ES-CS-PTR and the corresponding CS-ES-PTR.

2.2.4.1 Look at the CSQ-ECNOR/CSQ-AUCR combination pointed to by the current ESQ-CS-PTR.

2.2.4.2 Scan the CS-ACTION-LIST looking for a match between the CSQ-ECNOR/CSQ-AUCR combination found in the previous step and a CS-ECNO/CS-AUC combination.

2.2.4.3 When a match is found, populate both pointers.

ES-CS-PTR = CS-INDEX
CS-ES-PTR = ES-INDEX

12.5 Outputs

1. RET-STATUS

SECTION 13

FUNCTION CDGTN - RETRIEVE TAG NAME

This routine retrieves the entity class name and tag name given a tag number.

13.1 Inputs

1. TAG-NO PIC S9(4) COMP.
2. EC-NO PIC S9(4) COMP.

13.2 CDM Requirements

1. E5 ATTRIBUTE_USE_CL
2. E142 ENTITY_NAME

13.3 Internal Requirements

None

13.4 Processing

1. Process the following single row SQL select.

```
Select A. TAG_NAME,
       E. EC_NAME
From   ATTRIBUTE_USE_CL A,
        ENTITY_NAME E
Where  E. EC_NO = A. EC_NO AND
       E. EC_NAME_TYPE = 'PRIMARY' AND
       A. TAG_NO = :TAG-NO
```

2. If the select is unsuccessful, perform standard error processing.

13.5 Outputs

1. TAG-NAME PIC X(30)
2. EC-NAME PIC X(30)
3. RET-STATUS PIC X(5)

13.5 Outputs

1.	TAG-NAME	PIC X(30)
2.	EC-NAME	PIC X(30)
3.	RET-STATUS	PIC X(5)

SECTION 14

FUNCTION CDPBL - POPULATE BOOLEAN LIST

For type 1 referential integrity and key uniqueness tests, this routine builds a complete boolean list. For selects, modifys, deletes and type 2 referential integrity tests, this routine populates BL-CSQ-PTRs and BL-CS-PTRs.

14.1 Inputs

1. ES-QUALIFY-LIST
2. CS-QUALIFY-LIST
3. CS-ACTION-LIST

14.2 CDM Requirements

None

14.3 Internal Requirements

None

14.4 Processing

1. If processing a type 1 referential integrity test or a key uniqueness test (CS-ACTION equals 1 or K) perform the following steps:

   1.1 Set BL-USED to zero.

   1.2 Scan the CS-QUALIFY-LIST looking for all used entries which have CSQ-ECNOR equal zero.

   1.3 For each such CS-QUALIFY entry found, populate two new rows of the BOOLEAN-LIST.

      1.3.1 BL-USED = BL-USED + 1

      1.3.2 If BL-USED is greater than BL-MAX, generate an appropriate error message and exit.

      1.3.3 Set BL-INDEX to BL-USED.

      1.3.4 BL-OP = blank
      BL-ESQ-PTR = zero
      BL-CSQ-PTR = CSQ-INDEX
      BL-EVAL-FLAG = zero

      1.3.5 Search the CS-ACTION-LIST looking for a CS-ECNO/CS-AUC combination matching the current CSQ-ECNOL/CSQ-AUCL combination. When found:

BL-CS-PTR = CS-INDEX

1.3.6 BL-USED = BL-USED + 1

1.3.7 If BL-USED is greater than BL-MAX, generate an error message and exit.

1.3.8 Set BL-INDEX to BL-USED.

1.3.9 BL-ESQ-PTR = ZERO
BL-CSQ-PTR = ZERO
BL-CS-PTR = ZERO
BL-EVAL-FLAG = ZERO
BL-OP = "AND"

1.3.10 Exit the program.

2. If processing a select, modify or type 2 referential integrity test, fill in all used BL-CSQ-PTRs and BL-CS-PTRs with BL-ESQ-PTR not equal zero.

2.1 BL-CSQ-PTR = ESQ-CS-PTR (BL-ESQ-PTR)

2.2 Scan the CS-ACTION-LIST attempting to match a CS-ECNO/CS-AUC combination with the CSQ-ECNOL/CSQ-AUCL combination pointed to by BL-CSQ-PTR. When the match is found:

BL-CS-PTR = CS-INDEX

2.3 Exit the program.

3. If processing an insert (CS-ACTION equals I), exit the program.

14.5 Output

1. BOOLEAN-LIST

2. RET-STATUS

SECTION 15

FUNCTION PRE5 - DECOMPOSE CS NDML

The CS NDML Decomposer is a precompile-time module whose purpose is to break down a CS NDML Transaction into its various IS NDML Subtransactions. Each Subtransaction accesses only one database, managed by one DBMS, at one computer. If it is a retrieval request, its result is a single relation. Each Subtransaction to a non-relational database will be passed to the IS Access Path Selector (PRE6), which determines the necessary path to traverse the local Internal Schema.

The Decomposer maps from CS attribute use classes to IS counterparts, and passes CS/IS transform information to the Request Processor Generator (PRE9). It builds a SET-TABLE, which will be input to the IS Access Path Selector (PRE6). The SET-TABLE describes IS record sets that must be accessed in processing the request.

For retrieval requests, the Decomposer also generates a Join Query Graph (JQG) and Result Field Table (RFT) which will be input to the Distributed Request Supervisor configuration item to determine the best sequence of joins, unions, and OUTER JOIN operations to combine the results of the Subtransactions. The JQG and RFT are also input to the Call and Message Builder (PRE10).

The CS NDML Decomposer performs the following:

1. It maps all CS attribute use classes, entity classes, and relation classes referenced by the transaction to IS counterparts, determining the database locations of all attribute use classes involved.

   This includes identification of CS entities that participate in IS unions and IS horizontal partitions. An IS union occurs when multiple CS entity classes map to the same IS record type without being joined through a relation class. An IS horizontal partition occurs when a CS entity class maps to multiple IS record types, with the distribution of entity instances determined by values of one or more CS attribute classes.

2. It identifies all unions, intra-database joins, inter-database joins, and OUTER JOIN operations in the transaction.

3. It reformats the original transaction into single database Subtransactions, each of which accesses one database. If the Subtransaction is a retrieval, it results in a single relation (i.e., table). It provides a unique name for each result relation.

4. It creates a Join Query Graph (JQG) to record the joins, unions, and OUTER JOIN operations necessary to complete the transaction.

5. It creates a Result Field Table (RFT) to record the fields that will comprise the answer to a retrieval transaction.

15.1 Inputs

1. CDM Metadata

The entity classes needed are:

ENTITY_CLASS	CDM_TABLE	ENTITY_NUMBER
ATTRIBUTE_USE_CL	AUC	E5
AUC_IS_MAPPING	AUCISM	E108
AUC_ST_MAPPING	AUCSM	E135
COMPLEX_MAPPING_PARM	AUC_PARM	E254
	CON_PARM	E255
	RT_PARM	E256
	DF_PARM	E257
DATA_BASE	F/DB	E24
DATA_FIELD	DF	E67
	EDF	E119
	RDF	E106
	CDF	E9
	OCC_DEP_DF	E106
	INDEX_DF	E83
	FILLER_DF	E38
DATA_FIELD_USAGE	DFU	E300
DB_PASSWORD	PWD	E25
DBMS	DMS	E23
DBMS_ON_HOST	DBH	E20
DISTRIBUTED_RULES	DR	
ECRTUD	ECRTUD	E204, E205
ENTITY_CLASS	EC	E1
HORIZONTAL_PART	HP-FRAG	E212
INHERITED_ATT_USE	IAC	E7
KEY_CLASS_MEMBER	KCM	E6
MODULE_PARAMETER	PARM	E59
PROJECT_DATA_FIELD	AUCDF	E108
RC_BASED_REC_SET	RCSM	E109
RECORD_SET	RS	E72
RECORD_SET_USAGE	RSU	E299
RECORD_TYPE	RT	E66
SCHEMA_NAMES	SCH	E14
SET_TYPE_MEMBER	RSM	E72
USER_DEF_DATA_TYPE	UDDT	E233, E234, E235

2. The NDML conceptual schema request to be transformed.

This is output from the Transform ES/CS function PRE4 and includes:

CS-ACTION-LIST
CS-QUALIFY-LIST
BOOLEAN-LIST

3. The following tables and lists that are simply passed on to other modules:

ES-ACTION-LIST	from PRE4	to PRE8, PRE10
ES-QUALIFY-LIST	from PRE4	to PRE10
UV-ABBR-LIST	from PRE4	to PRE8
CODE-GENERATOR-TABLE	from PRE12	to PRE13
FORTRAN-VARIABLE-TABLE	from PRE2	to PRE10

15.2 Processing

1. Initialize all local tables and output lists. If CS-ACTION is a value other than 'S' or '1' or '2' or 'K' or 'I' or 'M' or 'D' or 'Q' go to Step 15. Otherwise, fill in IS-LOCK = CS-LOCK and IS-NDML-NO = CS-NDML-NO.

2. If CS-ACTION = 'S' or '1' or '2' or 'K' or 'Q' or 'M' or 'I' or 'D' transform each CS-ACTION-LIST entry by doing the following:

   2.a Retrieve information about the entity being mapped. Access the CDM table Distributed_Rules (DR) to determine the retrieval and update rules. Also access the CDM table HORIZONTAL_PART to determine if the entity is horizontally partitioned.

   Select the AUC-IS mappings to use based on the retrieval or update rules.

   2.a.1 If CS-ACTION = 'S' or 'Q' and RETRIEVAL-RULE = "AR", select the mappings to local databases with the same HOST where the AP will run. Find all the entries with AUCNO = CS-AUC and HOST-ID = host and MAP-CATEGORY = "ACTIVE" where the AP will run.

   Access CDM tables F/DB, RT and AUCISM to select an entry with the lowest preference number, i.e. closest to 1. If an entry is found, populate the AUCISM-LIST with the AUC-NO, DB-ID, RT-ID, PREF-NO and MAP-TYPE and proceed to Step 2.a.4. If no entry is found, proceed at Step 2.a.2 to select the first preference mapping.

   2.a.2 If CS-Action = 'S' or 'Q' and RETRIEVAL-RULE = "AR" and no mapping on host found in Step 2.a.1
   or CS-Action = 'S' or 'Q' and RETRIEVAL-RULE = "DR"
   or Cs-Action = '1' or '2' or 'K'
   or CS-Action = 'I' or 'M' or 'D' and UPDATE-RULE = "DU" select the primary AUC-IS Mapping. Find an entry with AUCNO = CS-AUC.

   Access CDM tables F/DB, RT and AUCISM to select this entry. If an entry is found with PREF-NO = 1,

populate the AUCISM-LIST with the AUC-NO, DB-ID, RT-ID, PREF-NO and MAP-TYPE, and continue at Step 2.a.4. If an entry is found with PREF-NO > 1, reject the NDML statement. (The AUC does not have a primary mapping and cannot be handled by this version of the precompiler). If an entry is not found, the AUC is a phantom and must be dealt with at Step 2.a.3, but if CS-Action = 'I' or 'M' or 'D', reject the NDML statement (update of phantoms not supported).

2.a.3 Transform phantoms which do not map directly to the internal schema. Find the parent AUC (KCM-TAG-NO) of this AUC, along with the relation class (RC-NO) thru which this AUC was inherited. Access CDM tables IAC and AUC to find an entry where CS-AUC = inherited tag (KCM-TAG-NO). If no entry is found, reject the NDML statement. The AUC does not have a corresponding IAC. This AUC is not mapped and cannot be handled by the precompiler.

If an entry is found,

a) Find any relation class to set mappings for the RC-NO found earlier and populate the SET-TABLE. Access CDM tables RCSM, RS and RSM to find entries with the same RC-NO found in Step 2.a.3. For each entry found, populate the SET-TABLE with no duplication, with the DB-ID, SET-ID, OWNER-RTID and MEMBER-RTID.

b) Using CS-AUC = KCM-TAG-NO found in Step 2.a.3, return to Step 2.a.1 to find the AUC-IS mapping for the owner of the phantom AUC.

2.a.4 Determine if the AUC-IS mapping selected is in a fragment of a horizontal partition. If the entity is horizontally partitioned, and CS-ACTION = 'I' or 'M' or 'D', reject the NDML statement. (Update actions are not supported for horizontally partitioned entities since the current version of the precompiler does not handle distributed update). If any other action, select the corresponding mapping of all other fragments also. Using the AUCISM-LIST entry populated in Step 2.a.1 or 2.a.2, access the CDM tables HP-FRAG, AUCISM and RT to select all the other fragments of the partition. For each entry found, populate the AUCISM-LIST with the TAG-NO, DB-ID, RT-ID, PREF-NO and MAP-TYPE. Exit Step 2a.

2.a.5 If CS-Action = 'I' or 'M' or 'D' and UPDATE-RULE = "AU" (attempt non-guaranteed distributed update) Select all preference AUC-IS Mappings where AUCNO = CS-AUC.

2.a.6 Continue populating the AUCISM list with all other preference mappings of this AUC, if any, that were not selected in Step 2.a.1 or 2.a.2 or 2.a.4.

These AUC-IS mappings would possibly be used to record inter-subtransaction joins. They will not be used to build IS-ACTION-ENTRYs. Flag these entries accordingly. Exit Step 2.a.

2.1 Transform AUCs to data fields.

For each AUCISM entry from Step 2.a.1 or 2.a.2 and 2.a.3, if MAP-TYPE = "FIELD":

2.1.1 Find the AUC to Data Field Mapping, and build an IS-ACTION-ENTRY. Access the CDM tables AUCDF and DF to retrieve information about the datafield. Using the DB-ID, RT-ID and TAG-NO of the AUCISM-LIST entry, find the DF-ID, DF-NO, NUM-OCCURS, DBMS-ACCESS, INDEX-IND, COMP-OF-DFNO, INDEX-BY-DFNO and DATA-TYPE-NAME.

2.1.2 Build an IS-ACTION-ENTRY as follows:

```
IS-ACTION              = CS-ACTION
IS-DBID                = DBID of AUCDF's key
IS-RTID                = RTID of AUCDF's key
IS-DFID                = DFID of AUCDF's key
IS-LOCAL-VARIABLE      = CS-LOCAL-VARIABLE
IS-MAPPED-TO-FLAG      = 'Y'
IS-CS-PTR              = CS-ACTION-LIST index
IS-FCTN-NAME           = CS-FCTN-NAME
IS-FCTN-DISTINCT       = CS-FCTN-DISTINCT
IS-SOURCE              = CS-SOURCE
IS-DF-KNOWN-FLAG       = DBMS-ACCESS
IS-DF-OCCURS           = NUM-OCCURS
```

2.1.3 If the DATA-TYPE-NAME retrieved is NULL, implying a group data field (i.e. not an elementary data field) populate the IS-ACTION-ENTRY with the CS-ACTION entry's data type, size and decimal specification.

```
IS-DATA-TYPE           = CS-TYPE
IS-SIZE                = CS-SIZE
IS-ND                  = CS-ND
```

If a DATA-TYPE-NAME is retrieved, access the CDM table UDDT to retrieve the DATA-TYPE, SIZE and ND.

```
IS-DATA-TYPE           = DATA-TYPE
IS-SIZE                = SIZE
IS-ND                  = ND
```

2.1.4 Check for mappings to repeating data fields, indexes or components of repeating data fields

2.1.4a Check for mappings to indexes for repeating data fields. IF INDEX-IND = "G" (implying it is a generated index) Set IS-DF-REPEAT-FLAG = 'I' Continue at Step 2.1.5 A generated index cannot be a repeating field.

IF INDEX-IND = 'Y' (implying it is an index)
Set IS-DF-REPEAT-FLAG = 'I'
Continue a Step 2.1.5 A user specified index cannot be a repeating field.

2.1.4b Check for mapping to repeating data fields and components of repeating data fields.

IF IS-DF-OCCURS = 1 and COMP-OF-DF = "NULL"
SET IS-DF-REPEAT-FLAG = N
Continue at Step 2.1.5

2.1.4c At this point, the field is a repeating field, a component of a repeating data field or a component of a non-repeating field.

Using a recursive search against CDM table DF, record the levels of repeating data fields in the TEMP-OCCURS-TABLE.

2.1.4d If it is determined that no field repeats in the recursive search:

SET IS-DF-REPEAT-FLAG = "C" stating the AUC is mapped to a component of a repeating or non-repeating data field.
Continue at Step 2.1.5

Else

SET IS-DF-REPEAT-FLAG = "Y" stating the AUC is mapped to a repeating data field or a component of a repeating data field. This component structure is stored in TEMP-OCCURS-TABLE and will be referenced later in Step 12.1.
Continue at Step 2.1.5

2.1.5 Record other AUC-DF mappings for possible use in inter-subtransaction joins. If CS-ACTION = 'I', ignore this step because insert action does not build an IS-QUALIFY list. Continue at Step 2.3. If the entity is horizontally partitioned, ignore this step since it is not meaningful to join across record types of a horizontally partitioned entity, because no rows of data would meet this qualification.

Find all the KCM (E6) entries with AUCNO = CS-AUC. If none are found, this AUC is not a key class member and therefore cannot be used for joining subtransaction results; ignore this step.

For each remaining AUCISM-LIST entry for which an IS-ACTION entry was NOT built and if one or more KCM entries are found, proceed as follows:

if IS-DF-REPEAT-FLAG = N
(NUM-OCCURS = 1 and
INDEX-IND = 'N' and
COMP-OF-DF = ZEROS)

Build the REPL-JOIN-LIST, TEMP-KEY-LIST and KEY-JOIN-LIST

Else

Continue at Step 2.2

2.1.5.1 Fill in a REPL-JOIN-LIST entry:

RJ-DBID = DBID of AUCDF's key
RJ-RTID = RTID of AUCDF's key
RJ-DFID = DFID of AUCDF's key
RJ-TYPE = DATATYPE of the DF
RJ-SIZE = SIZE of the DF
RJ-ND = ND of the DF
RJ-PTR = IS-ACTION-LIST index
for the entry filled
in Step 2.1.2
RJ-PTR-TYPE = 1
RJ-OK = 'Y'

2.1.5.2 Find the TEMP-KEY-LIST entries with TK-AUCNO = AUCNO of the KCM (E6). For each, set TK-REF-FLAG = 'Y'.

2.1.5.3 If no TEMP-KEY-LIST entries were found in Step 2.1.5.2, find the KCM (E6) entries with the same KCNO as the KCM (E6) entry found in Step 2.1.5.

Fill in a TEMP-KEY-LIST entry for each of these KCM (E6):

TK-KCNO = KCNO of KCM
TK-AUCNO = AUCNO of KCM
TK-REF-FLAG = 'N' for each of the KCM entries except the on found in Step 2.1.where TK-REF-FLAG = 'Y'

2.1.5.4 Fill in a KEY-JOIN-LIST entry for each possible pair of a REPL-JOIN-LIST entry created in Step 2.1.5.1 and a TEMP-KEY-LIST entry created in Step 2.1.5.3:

KJ-TK-PTR = TEMP-KEY-LIST index
KJ-RJ-PTR = REPL-JOIN-LIST index

2.2 Transform AUCs to record sets.

For each AUCISM entry from Step 2.a.1 or 2.a.2 and 2.a.3 if MAP-TYPE = "SET":

2.2.1 Access the CDM tables AUCSM, RS, RSM and RT to retrieve information about the set and set values. Using the DB-ID, RT-ID and TAG-NO of the AUCISM-LIST entry, find the SET-ID, RT-ID of member, RT-ID of owner, TOTAL-NUM-MEMBERS and AUC-VALUES.

2.2.2 Fill in an IS-ACTION-LIST entry for the AUCSM (E135) entry found in Step 2.2.1:

```
IS-ACTION        = CS-ACTION
IS-DBID          = DBID of AUCSM's key
IS-NUM-RS        = number of located AUCSMs
IS-RSNO (i)      = RSNO of i-th AUCSM's key
IS-RS-VALUE (i)  = EQUIVALENT-AUC-VALUE
                   in i-th AUCSM
IS-LOCAL-VARIABLE = CS-LOCAL-VARIABLE
IS-MAPPED-TO-FLAG = 'Y'
IS-CS-PTR        = CS-ACTION-LIST index
IS-FCTN-NAME     = CS-FCTN-NAME
IS-FCTN-DISTINCT = CS-FCTN-DISTINCT
IS-DF-REPEAT-FLAG = 'N'
IS-DF-OCCURS     = 1
```

2.2.3 Fill in a SET-TABLE entry for the AUCSM (E135) entry found in Step 2.2.1:

ST-DBID = DBID of AUCSM's key
ST-RSNO = RSNO of AUCSM's key

Note: If there is already a SET-TABLE entry with this ST-DBID and ST-RSNO, do not duplicate it.

RS-MEMBER (i) = RTNO of RSM's key

Set the value of ST-NUM-MEMBERS correctly in order to manage the repeating group.

ST-OWNER = OWNER-RTNO in RS
ST-TOTAL-MEMBERS = TOTAL-NUM-MEMBERS in RS

2.3 Transform AUCs using complex mapping algorithms.

For each AUCISM entry from Step 2.a.1 or 2.a.2 and 2.a.3 for MAP-TYPE = "COMPLEX" set up the algorithm direction:

IF CS-ACTION = "I" or "M"
CS-IS-ALG-USE-CODE = "U"
Else
CS-IS-ALG-USE-CODE = "R"

Determine if the algorithm transforms from AUC(s) to datafields(s) (DF-PARM) or from AUC(s) to an entire record (RT-PARM).

2.3.1 First find the AUC-PARM entry with the same AUC-NO as the AUCISM-LIST entry, and the CS-IS-ALG-USE-CODE. Using the MOD-ID, MOD-INSTANCE and ALG-USE-CODE retrieved from the AUC-PARM entry and the DB-ID, RT-ID of the AUCISM-LIST entry, access the CDM tables CMA and DF to determine if the AUC maps to datafield(s).

If no entry is found, access the CDM tables CMA and RT to determine if the AUC maps to a record type.

If the AUC does not map to either, reject the NDML statement (mapping not found). Exit Step 2.3.

2.3.2 Access the CDM table CMA using the MOD-ID, MOD-INSTANCE and ALG-USE-CODE to retrieve all the parameters of this complex mapping algorithm. Populate the CMA-ALG-ENTRY.

```
CMA-MOD-ID    = MOD-ID
CMA-MOD-INST  = MOD-INST
CMA-RETR-UPD  = ALG-USE-CODE
```

For each parameter retrieved:

2.3.3 Access the CDM tables PARM and UDDT to retrieve the parameters DATA-TYPE, SIZE and ND. Populate the CMA-PARM-ENTRY.

```
CMA-PARM-TYPE    = DATA TYPE
CMA-PARM-SIZE    = SIZE
CMA-PARM-ND      = ND
```

2.3.4 If the parameter is an AUC-PARM (attribute use class):
CMA-TAG-NO = AUCNO retrieved

If the parameter is a CONST-PARM (constant):
CMA-CONST-VAL = CONSTANT-VALUE retrieved

If the parameter is a RT-PARM (record):
CMA-RT-NO = RT-NO retrieved, also populate the CMA-DF-ENTRY with information about all the elementary datafields of this record, along with the type, size and decimal specification of each datafield. Access CDM tables DF and UDDT using the RT-NO parameter.

If the parameter is a DF-PARM (data field):
CMA-DF-NO = DF-NO retrieved
CMA-RT-NO = RT-NO of DF-NO retrieved
Also, retrieve the data type, size and decimal specification of the datafield by accessing the CDM tables DF and UDDT. If the datafield is not elementary, issue a warning and set

```
CMA-DF-TYPE = DATATYPE of PARM (CMA-PARM-TYPE)
CMA-DF-SIZE = SIZE     of PARM (CMA-PARM-SIZE)
CMA-DF-ND   = ND       of PARM (CMA-PARM-ND)
Else
CMA-DF-TYPE = TYPE of DF
CMA-DF-SIZE = SIZE of DF
CMA-DF-ND   = ND   of DF
```

Note: An AUC can map to either a record or to datafield(s) through a complex mapping algorithm, but not both.

2.3.5 Record the complex mapping algorithm that must be used.

Build an IS-ACTION-ENTRY for the AUC-PARM entry from Step 2.1.2 as follows:

```
IS-ACTION   =   CS-ACTION
IS-DBID     =   DBID in the AUC-PARM entry
IS-RTID     =   RTNO in the AUC-PARM entry
IS-DFNO     =   blank
IS-MAP-ALG-ID =   MOD-ID in the AUC-PARM entry
IS-MAP-ALG-PTR =   index value of the
CMA-ALGORITHM-ENTRY
from Step 2.3.3
IS-PARM-NO          =   PARM-NO from Step 2.3.1
IS-LOCAL-VARIABLE =   CS-LOCAL-VARIABLE
IS-MAPPED-TO-FLAG =   'Y'
IS-CS-PTR           =   CS-ACTION-LIST index
IS-FCTN-NAME        =   CS-FCTN-NAME
IS-FCTN-DISTINCT    =   CS-FCTN-DISTINCT
IS-DF-REPEAT-FLAG =   'N'
```

2.4 Fill in the TEMP-RECORD-TABLE as follows:

If CS-Action = "I" or "D"

Scan the IS-ACTION-LIST, making an TRT-LIST entry for each distinct IS-DBID or IS-RTNO encountered:

```
TRT-DBNO = IS-DBID
TRT-RTNO = IS-RTNO
TRT-ECNO = CS-ECNO
```

If there is already a TRT-LIST entry for a TRT-DBID or TRT-DBID or TRT-RTNO, do not duplicate the entry.

Note: If IS-ACTION = "DELETE", consider only those entries where CS-SOURCE not = "G" (generated). CS-SOURCE is "G" for CS-QUALIFY AUC entries from the "USING" clause of a DELETE statement, which were moved to the CS-ACTION.

3. If CS-ACTION = 'I' or 'D' or 'M' continue processing the IS-ACTION-LIST as follows.

3.1 Determine that if more than one tag participates in a complex mapping algorithm, all tags are mentioned in the CS-ACTION-LIST.

Search the CS-ACTION-LIST using all tags in the COMPLEX-MAPPING-ALG-TBL, looking for a match. If a match is not found, reject the NDML statement (not all parameters for the complex mapping algorithm are specified on the NDML statement).

3.2 Fetch all the elementary datafields for the record being inserted or deleted. These datafields will be marked as 'NOT-MAPPED-TO'. For an insert action, NULLS will be inserted for these fields. For a delete action, the record will be modified with nulls being placed in the 'MAPPED-TO' fields. For each entry in the TEMP-RECORD-TABLE, if CS-ACTION = INSERT or CS-ACTION = DELETE:

3.2.1 Find the elementary datafield (EDF) entries with the same DBID and RTNO as the TRT-LIST entry.

3.2.2 For each of the DF (E67) entries found in Step 3.2.1:

3.2.2.1 If this elementary field is not known to the DBMS, (DBMS-ACCESS-FLAG = Unknown), continue with the next iteration of Step 3.2.2.

3.2.2.2 If there exists an IS-ACTION-LIST entry with:

IS-DBID = DBID of the DF
IS-RTNO = RTNO of the DF
IS-DFNO = DFNO of the DF

or this field is a component of an insert or delete field (component data field number matches an entry in the IS-ACTION-LIST), or this field is a redefinition of an insert or delete field (redefines data field number matches an entry in the IS-ACTION-LIST), then continue with the next iteration of Step 3.2.2.

3.2.2.3 If there exists a CMA entry for this datafield with

CMA-DBID = DBID of the DF
CMA-RTNO = RTNO of the DF
CMA-DFNO = DFNO of the DF
(CMA-DFNO is filled in for a DF-PARM)
or
CMA-DBID = DBID of the DF
CMA-RTNO = RTNO of the DF
CMA-DFNO = ZERO (CMA-DFNO is zero for a RT-PARM)

or
this field is a component of an insert or delete field (redefines data field number matches an entry in the COMPLEX-MAPPING-TABLE)

or
this field is a redefinition of an insert or delete field (redefines data field number matches an entry in the COMPLEX-MAPPING-TABLE)

then continue with the next iteration of Step 3.2.2.

3.2.2.4 Fill in an IS-ACTION-LIST entry for this elementary data field as follows:

IS-ACTION = CS-ACTION used in Step 3.2
IS-DBID = DBID of the DF
IS-RTNO = RTNO of the DF
IS-DFNO = DFNO of the DF
IS-TYPE = DATA TYPE of the DF
IS-SIZE = SIZE of the DF
IS-ND = ND of the DF
IS-LOCAL-VARIABLE = blank
IS-MAPPED-TO-FLAG = 'N'

```
IS-CS-PTR          = zero
IS-FCTN-NAME       = blank
IS-FCTN-DISTINCT   = blank
IS-DF-KNOWN-FLAG   = DBMS accessible Data Field
                     indicator of the DF
IS-CS-PTR          = 0
IS-SOURCE          = 'G'
```

3.2.2.5 If this field does not repeat, nor is a component of another field: if NUM-OCURS = 0 and COMP-OF-DF = 0 then SET IS-DF-REPEAT-FLAG = 'N' and continue with the next iteration of Step 3.2.2.

3.2.2.6 Determine if this field repeats or is part of a repeating group or is a non-repeating component.

3.2.2.6.1 If the field repeats or is part of a repeating group, build a TEMP-OCCURS-TABLE recording the component chain. Also, set IS-DF-OCCURS-FLAG = 'Y'.

3.2.2.6.2 If the field is a non-repeating component set IS-DF-OCCURS-FLAG = 'C'

4. Provide union discriminator values for INSERTs.

If CS-ACTION = 'I', for each unique pair of IS-DBID and IS-RTNO find all the ECRTUD (E205) entries with:

```
ECNO        =   CS-ECNO in any CS-ACTION-ENTRY
                (all CS-ECNOs are identical for an

DBID        =   IS-DBID
RTNO        =   IS-RTNO
```

If no such ECRTUD entries are found, none of the record types result from the union of entity classes and the rest of Step 4 can be ignored.

For each ECRTUD entry found, determine the comparison operator. If any operator from the ECRTUD entry is not =, >=, or <=, reject the NDML statement (operator not supported for an insert of a record type resulting from the union of entity classes).

For each ECRTUD entry found:

4.1 Start an IS-ACTION-LIST entry:

```
IS-ACTION          = CS-ACTION
IS-DBID            = DBID from the ECRTUD entry
IS-RTNO            = RTNO from the ECRTUD entry
IS-DFNO            = DFNO from the ECRTUD entry
IS-LOCAL-VARIABLE  = generated local variable
                     containing UNION-VALUE
                     from the ECRTUD entry
IS-MAPPED-TO-FLAG  = 'Y'
IS-CS-PTR          = zero
IS-FCTN-NAME       = blank
```

```
IS-FCTN-DISTINCT  = blank
IS-DF-REPEAT-FLAG = 'N'
IS-DF-OCCURS      = 1
IS-DF-KNOWN       = 'Y'
IS-DF-REPEAT-FLAG = 'N'
IS-SOURCE         = 'G'
```

4.2 Access the CDM tables DF and UDDT to retrieve the DATATYPE, SIZE and ND of the datafield.

```
IS-DATA-TYPE = DATATYPE of DF
IS-SIZE      = SIZE     of DF
IS-ND        = ND       of DF
```

5. If CS-ACTION = 'S' or '1' or '2' or 'K' or 'Q' or 'M' or 'D', transform each CS-QUALIFY-LIST entry. Perform Steps 5.a thru 5.4 to transform the CS-QUALIFY left side.

Perform Steps 5.5 thru 5.9 to transform the CS-QUALIFY right side. Proceed as follows:

5.a Select the AUC-IS mapping(s) to use. For each CSQ-AUCL Set CSQ-AUC = CSQ-AUCL

5.a.1 Select all preference mappings for this AUC. Find an entry with AUCNO = CSQ-AUC.

Access CDM tables F/DB, RT and AUCISM to select all entries where MAP CATEGORY = ACTIVE, ordered by PREF-NO. For each entry found populate the AUCISM-LIST with the AUC-NO, DB-ID, RT-ID, PREF-NO and MAP-TYPE. Also, populate variable AUCISM-FLAG. For all preference one mappings or if IS-ACTION = "M" or "D", set the flag equal to "IS". Otherwise, set the flag equal to "RJ." If no entry is found with PREF-NO = 1, reject the NDML statement since the AUC does not have a primary mapping and cannot be handled by this version of the precompiler. If an entry is not found, the AUC is a phantom and must be dealt with at Step 5.a.2, but if CS-ACTION = 'I' or 'M' or 'D', reject the NDML statement since update of phantoms is not supported.

5.a.2 Transform phantoms which do not map directly to the internal schema. Find the parent AUC (KCM-TAG-NO) of this AUC, along with the relation class (RC-NO) through which this AUC was inherited. Access CDM tables IAC and AUC to find an entry where CSQ-AUC = inherited tag (KCM-TAG-NO). If no entry is found, reject the NDML statement since the AUC does not have a corresponding IAC. This AUC is not mapped and cannot be handled by the precompiler. If an entry is found:

a) Find any relation class to set mappings for the RC-NO found earlier and populate the SET-TABLE. Access CDM tables RCSM, RS and RSM to find entries with the same RC-NO found in Step 5.a.2. For each entry found, populate the SET-TABLE using no duplications, with the DB-ID, SET-ID, OWNER-RTID and MEMBER-RTID.

b) Using CSQ-AUC = KCM-TAG-NO found in Step 5.a.2, return to Step 5.a.1 to find the AUC-IS mapping for the owner of the phantom AUC.

5.1 Transform AUCs to data fields.

For each AUCISM entry from Step 5.a where MAP-TYPE = "FIELD" and AUCISM-FLAG = "IS", perform steps 5.1.1 through 5.1.4. For each entry where MAP-TYPE = "FIELD" and AUCISM-FLAG = "RJ", perform step 5.1.5.

5.1.1 Find the AUC to Data Field Mapping and build an IS-QUALIFY-ENTRY.

Access the CDM tables AUCDF and DF to retrieve information about the datafield. Using the DB-ID, RT-ID and TAG-NO of the AUCISM-LIST entry, find the DF-ID, DF-NO, NUM-OCCURS, DBMS-ACCESS, INDEX-IND, COMP-OF-DFNO, INDEX-BY-DFNO and DATA-TYPE-NAME.

5.1.2 Build an IS-QUALIFY-ENTRY. If this is the first entry in the AUCISM-LIST, set ISQ-INDEX-HOLD = isq-index of this IS-QUALIFY-ENTRY being built

```
ISQ-DBIDL           = DBID of AUCDF's key
ISQ-RTIDL           = RTID of AUCDF's key
ISQ-DFIDL           = DFID of AUCDF's key
ISQ-LOCAL-VARIABLE = CSQ-LOCAL-VARIABLE
ISQ-CSQ-PTR         = CS-QUALIFY-LIST index
ISQ-DFL-KNOWN-FLAG = DBMS-ACCESS
ISQ-BOOLEAN-PTR     = BOOLEAN-LIST index
where
BL-CSQ-PTR          = CSQ-INDEX
```

5.1.3 If the DATA-TYPE-NAME retrieved is NULL, implying a group data field, not an elementary data field, populate the IS-QUALIFY-ENTRY with the CS-QUALIFY entry's data type, size and decimal specification.

```
ISQ-TYPEL         = CSQ-TYPE
ISQ-SIZEL         = CSQ-SIZE
ISQ-NDL                 = CSQ-ND
```

If a DATA-TYPE-NAME is retrieved, access the CDM table UDDT to retrieve the DATA-TYPE, SIZE and ND.

```
ISQ-TYPEL         = DATA-TYPE
ISQ-SIZEL         = SIZE
ISQ-NDL                 = ND
```

5.1.4 Check for mappings to repeating data fields, indexes or components of repeating data fields

5.1.4a IF INDEX-IND = 'Y',
set ISQ-TYPE2-SOURCE = 'I'
Continue a Step 5.1.5
A user specified index cannot be a repeating field.

5.1.4b Check for mapping to repeating data fields and components of repeating data fields.

IF NUM-OCCURS = 1 and COMP-OF-DF = ZERO,
Continue at Step 5.1.5

5.1.4c At this point, the field is a repeating field or a component of a repeating data field or simply a component of a non-repeating field.

If it is determined that the field repeats or is part of a repeating group, issue an error message. The current precompiler does not support qualification of repeating fields.

5.1.5 Record other AUC-DF mappings for possible use in inter-subtransaction joins.

If the entity is horizontally partitioned, ignore this step. It is not meaningful to join across record types of a horizontally partitioned entity, since no rows of data would meet this qualification.

Find all the KCM (E6) entries with AUCNO = CSQ-AUCL. If none are found, this AUC is not a key class member and therefore cannot be used for joining subtransaction results. Ignore this step.

For each remaining AUCISM-LIST entry, if IS-QUALIFY entries were built for a first and second preference mapping and if one or more KCM entries are found, proceed as follows:

if (NUM-OCCURS = 1 and
INDEX-IND = 'N' and
COMP-OF-DF = NULL)

Build the REPL-JOIN-LIST, TEMP-KEY-LIST and KEY-JOIN-LIST

Else

Continue at Step 5.2

5.1.5.1 Fill in a REPL-JOIN-LIST entry:

```
RJ-DBID      = DBID of AUCDF's key
RJ-RTID      = RTID of AUCDF's key
RJ-DFID      = DFID of AUCDF's key
RJ-TYPE      = DATATYPE of the DF
RJ-SIZE      = SIZE of the DF
RJ-ND        = ND of the DF
RJ-PTR       = IS-QUALIFY-LIST index
               for the first entry filled in Step 5.1.2
               (ISQ-INDEX-HOLD)
RJ-PTR-TYPE  = 2
RJ-OK        = 'Y'
```

5.1.5.2 Find the TEMP-KEY-LIST entries with TK-AUCNO = AUCNO of the KCM (E6). For each, set TK-REF-FLAG = 'Y'.

5.1.5.3 If no TEMP-KEY-LIST entries were found in Step 5.1.5.2, find the KCM (E6) entries with the same KCNO as the KCM (E6) entry found in Step 5.1.5.

Fill in a TEMP-KEY-LIST entry for each of these KCM (E6):

```
TK-KCNO      = KCNO of KCM
TK-AUCNO     = AUCNO of KCM
TK-REF-FLAG = 'N' for each of the KCM
               entries except the one
               found in Step 5.1.5,
               where TK-REF-FLAG = 'Y'
```

5.1.5.4 Fill in a KEY-JOIN-LIST entry for each possible pair of one REPL-JOIN-LIST entry created in Step 5 1.5.1 and one TEMP-KEY-LIST entry created in Step 5.1.5.3:

```
KJ-TK-PTR = TEMP-KEY-LIST index
KJ-RJ-PTR = REPL-JOIN-LIST index
```

5.2 Transform AUCs to record sets.

For each AUCISM entry from Step 5.a if MAP-TYPE = "SET":

5.2.1 Access the CDM tables AUCISM, RS, RSM and RT to retrieve information about the set and set values. Using the DB-ID, RT-ID and TAG-NO of the AUCISM-LIST entry, find the SET-ID, RT-ID of number, RT-ID of owner, TOTAL-NUM-MEMBERS and AUC-VALUES.

5.2.2 Fill in an IS-QUALIFY-LIST entry for the AUCSM (E135) entry found in Step 5.2.1:

```
ISQ-DBIDL           = DBID of AUCISM's key
ISQ-NUM-RSL         = number of located AUCISMs
ISQ-RSNOL (i)       = RSNO of i-th AUCISM's key
ISQ-RSL-VALUE  (i) = EQUIVALENT-AUC-VALUE
                      in i-th AUCISM
ISQ-LOCAL-VARIABLE = CSQ-LOCAL-VARIABLE
ISQ-CSQ-PTR         = CS-QUALIFY-LIST index
ISQ-OP              = CSQ-OP
ISQ-BOOLEAN-PTR     = BOOLEAN-LIST index where
BL-CSQ-PTR          = CSQ-INDEX
```

5.2.3 Fill in a SET-TABLE entry for the AUCSM (E135) entry found in Step 5.2.1:

```
ST-DBID = DBID of AUCSM's key
ST-RSNO = RSNO of AUCSM's key
```

Note: If there is already a SET-TABLE entry with this ST-DBID, ST-RSNO, do not duplicate it.

```
RS-MEMBER (i) = RTNO of RSM's key
```

Set the value of ST-NUM-MEMBERS correctly to manage the repeating group.

ST-OWNER = OWNER-RTNO in RS
ST-TOTAL-MEMBERS = TOTAL-NUM-MEMBERS in RS

5.3 Transform AUCs using complex mapping algorithms.

For each AUCISM entry from Step 5.a for MAP-TYPE = "COMPLEX" set up the algorithm direction:

CS-IS-ALG-USE-CODE = "R"

Note: Since qualification conditions involving algorithms can only be evaluated at the CS level, the retrieval version of the algorithm is always obtained for ISQ entries.

Determine if the algorithm transforms from AUC(s) to datafield(s) (DF-PARM) or from AUC(s) to an entire record (RT-PARM).

5.3.1 First find the AUC-PARM entry with the same AUC-NO as the AUCISM-LIST entry, and the CS-IS-ALG-USE-CODE. Using the MOD-ID, MOD-INSTANCE and ALG-USE-CODE retrieved from the AUC-PARM entry and the DB-ID, RT-ID of the AUCISM-LIST entry, access the CDM tables CMA and DF to determine if the AUC maps to datafield(s).

If no entry is found, access the CDM tables CMA and RT to determine if the AUC maps to a record type.

If the AUC does not map to either, reject the NDML statement (mapping not found). Exit Step 5.3.

5.3.2 Access the CDM table CMA using the MOD-ID, MOD-INSTANCE and ALG-USE-CODE to retrieve all the parameters of this complex mapping algorithm. Populate the CMA-ALG-ENTRY.

CMA-MOD-ID = MOD-ID
CMA-MOD-INST = MOD-INST
CMA-RETR-UPD = ALG-USE-CODE

For each parameter retrieved:

5.3.3 Access the CDM tables PARM and UDDT to retrieve the parameters DATA-TYPE, SIZE and ND. Populate the CMA-PARM-ENTRY.

CMA-PARM-TYPE = DATA TYPE
CMA-PARM-SIZE = SIZE
CMA-PARM-ND = ND

5.3.4 If the parameter is an AUC-PARM:
CMA-TAG-NO = AUCNO retrieved

If the parameter is a CONST-PARM:
CMA-CONST-VAL = CONSTANT-VALUE retrieved

If the parameter is a RT-PARM:
CMA-RT-NO = RT-NO retrieved
Also, populate the CMA-DF-ENTRY with information about all the elementary datafields of this record, along with the type, size and decimal specification of each datafield. Access CDM tables DF and UDDT using the RT-NO parameter.

If the parameter is a DF-PARM:
CMA-DF-NO = DF-NO retrieved
CMA-RT-NO = RT-NO of DF-NO retrieved
Also, retrieve the data type, size and decimal specification of the datafield by accessing the CDM tables DF and UDDT. If the datafield is not elementary, issue a warning and set

```
CMA-DF-TYPE = DATATYPE of PARM (CMA-PARM-TYPE)
CMA-DF-SIZE = SIZE     of PARM (CMA-PARM-SIZE)
CMA-DF-ND   = ND       of PARM (CMA-PARM-ND)
Else
CMA-DF-TYPE = TYPE of DF
CMA-DF-SIZE = SIZE of DF
CMA-DF-ND   = ND   of DF
```

Note: An AUC can map to either a record or to datafield(s) but not both, through a complex mapping algorithm

5.3.5 Record the complex mapping algorithm that must be used.

Build an IS-QUALIFY-ENTRY for the AUC-PARM entry from Step 5.1.2 as follows:

ISQ-DBIDL	=	DBID in the AUC-PARM entry
ISQ-RTIDL	=	RTNO in the AUC-PARM entry
ISQ-DFNOL	=	blank
ISQ-MAP-ALG-IDL	=	MOD-ID in the AUC-PARM entry
ISQ-MAP-ALG-PTRL	=	index value of the CMA-ALGORITHM-ENTRY from Step 5.3.3
ISQ-PARM-NOL	=	PARM NO found in Step 5.3.1
ISQ-LOCAL-VARIABLE	=	CSQ-LOCAL-VARIABLE
ISQ-CSQ-PTR	=	CSQ-QUALIFY-LIST index
ISQ-OP	=	CSQ-OP

5.4 Fill in the TEMP-RECORD-TABLE as follows:

If IS-ACTION = 'S', '1', '2', OR 'K'

Scan the IS-QUALIFY-LIST, making a TRT-LIST entry for each distinct ISQ-DBIDL, ISQ-RTNOL encountered:

```
TRT-DBNO = ISQ-DBIDL
TRT-RTNO = ISQ-RTNOL
TRT-ECNO = CSQ-ECNOL
```

If there is already a TRT-LIST entry for a TRT-DBID or TRT-RTNO, do not duplicate the entry.
Return to Step 5.1 to process the next AUCISM-ENTRY.

5.5 Process the right side of the CS-QUALIFY-LIST.

For each non-zero CSQ-AUCR, proceed as follows:
Set CSQ-AUC = CSQ-AUCR
Select the AUC-IS mapping(s) to use for each CSQ-AUCR by performing Step 5.a.

5.6 Transform AUCs to data fields.

For each AUCISM entry from Step 5.a if MAP-TYPE = "FIELD"

5.6.1 Find the AUC to Data Field Mapping, and build an IS-QUALIFY-ENTRY. Access the CDM tables AUCDF and DF to retrieve information about the datafield. Using the DB-ID, RT-ID and TAG-NO of the AUCISM-LIST entry, find the DF-ID, DF-NO, NUM-OCCURS, DBMS-ACCESS, INDEX-IND, COMP-OF-DFNO, INDEX-BY-DFNO and DATA-TYPE-NAME.

5.6.2 Build an IS-QUALIFY-ENTRY as follows:

Fill an IS-QUALIFY-ENTRY with ISQ-CSQ-PTR the same as the entry's CS-QUALIFY-LIST index, creating a new entry (replicating the already filled in left side) if this CSQ-AUCR maps to more than one datafield.
If this is the first entry in the AUCISM-LIST, set ISQ-INDEX-HOLD = isq-index of this IS-QUALIFY-ENTRY being built

```
ISQ-DBIDR          = DBID of AUCDF's key
ISQ-RTIDR          = RTID of AUCDF's key
ISQ-DFIDR          = DFID of AUCDF's key
```

5.6.3 If the DATA-TYPE-NAME retrieved is NULL, implying a group data field, not an elementary data field, populate the IS-QUALIFY-ENTRY with the CS-QUALIFY entry's data type, size and decimal specification.

```
ISQ-TYPER                = CSQ-TYPE
ISQ-SIZER                = CSQ-SIZE
ISQ-NDR  = CSQ-ND
```

If a DATA-TYPE-NAME is retrieved, access the CDM table UDDT to retrieve the DATA-TYPE, SIZE and ND.

```
ISQ-TYPER                          = DATA-TYPE
ISQ-SIZER                          = SIZE
ISQ-NDR                  = ND
```

5.6.4 Check for mappings to repeating data fields, or components of repeating data fields.

5.6.4a IF NUM-OCCURS = 1 and COMP-OF-DF = NULL
Continue at Step 5.6.5

5.6.4b At this point, the field is a repeating field or a component of a repeating data field or simply a component of a non-repeating field.

If it is determined that the field repeats or is part of a repeating group, issue an error message. The current precompiler does not support qualification of repeating fields. Exit Step 5.

5.6.5 Record other AUC-DF mappings for possible use in inter-subtransaction joins. If the entity is horizontally partitioned, ignore this step. It is not meaningful to join across record types of a horizontally partitioned entity, since no rows of data would meet this qualification. Find all the KCM (E6) entries with AUCNO = CSQ-AUCR. If none are found, this AUC is not a key class member and therefore cannot be used for joining subtransaction results. Ignore this step.

For each remaining AUCISM-LIST entry, if IS-QUALIFY entries were built for a first and second preference mapping and if one or more KCM entries are found, proceed as follows:

if (NUM-OCCURS = 1 and
INDEX-IND = 'N' and
COMP-OF-DF = NULL)

Build the REPL-JOIN-LIST, TEMP-KEY-LIST and KEY-JOIN-LIST

Else

Continue at Step 5.7

5.6.5.1 Fill in a REPL-JOIN-LIST entry:

RJ-DBID	= DBID of AUCDF's key
RJ-RTID	= RTID of AUCDF's key
RJ-DFID	= DFID of AUCDF's key
RJ-TYPE	= DATATYPE of the DF
RJ-SIZE	= SIZE of the DF
RJ-ND	= ND of the DF
RJ-PTR	= IS-QUALIFY-LIST index for the first entry filled in Step 5.6.2 (ISQ-INDEX-HOLD)
RJ-PTR-TYPE	= 3
RJ-OK	= 'Y'

5.6.5.2 Find the TEMP-KEY-LIST entries with TK-AUCNO = AUCNO of the KCM (E6). For each, set TK-REF-FLAG = 'Y'.

5.6.5.3 If no TEMP-KEY-LIST entries were found in Step 5.6.5.2, find the KCM (E6) entries with the same KCNO as the KCM (E6) entry found in Step 5.6.5.

Fill in a TEMP-KEY-LIST entry for each of these KCM (E6):

TK-KCNO = KCNO of KCM
TK-AUCNO = AUCNO of KCM
TK-REF-FLAG = 'N' for each of the KCM entries except the one found in Step 5.6.5, where TK-REF-FLAG = 'Y'

5.6.5.4 Fill in a KEY-JOIN-LIST entry for each possible pair of one REPL-JOIN-LIST entry created in Step 5.6.5.1 and one TEMP-KEY-LIST entry created in Step 5.6.5.3:

KJ-TK-PTR = TEMP-KEY-LIST index
KJ-RJ-PTR = REPL-JOIN-LIST index

5.7 Transform AUCs to record sets.

For each AUCISM entry from Step 5.a if MAP-TYPE = "SET":

5.7.1 Access the CDM tables AUCISM, RS, RSM and RT to retrieve information about the set and set values. Using the DB-ID, RT-ID and TAG-NO of the AUCISM-LIST entry, find the SET-ID, RT-ID of member, RT-ID of owner, TOTAL-NUM-MEMBERS and AUC-VALUES.

5.7.2 Fill in an IS-QUALIFY-LIST entry for the AUCSM (E135) entry found in Step 5.7.1:

ISQ-DBIDR = DBID of AUCISM's key
ISQ-NUM-RSR = number of located AUCISMs
ISQ-RSNOR (i) = RSNO of i-th AUCISM's key
ISQ-RSR-VALUE (i) = EQUIVALENT-AUC-VALUE in i-th AUCISM

5.7.3 Fill in a SET-TABLE entry for the AUCSM (E135) entry found in Step 5.7.1:

ST-DBID = DBID of AUCSM's key
ST-RSNO = RSNO of AUCSM's key

Note: If there is already a SET-TABLE entry with this ST-DBID, ST-RSNO, do not duplicate it.

RS-MEMBER (i) = RTNO of RSM's key

Set the value of ST-NUM-MEMBERS correctly to manage the repeating group.

ST-OWNER = OWNER-RTNO in RS
ST-TOTAL-MEMBERS = TOTAL-NUM-MEMBERS in RS

5.8 Transform AUCs using complex mapping algorithms.

For each AUCISM entry from Step 5.a for MAP-TYPE = "COMPLEX" set up the algorithm direction.

CS-IS-ALG-USE-CODE = "R"

Note: Since qualification conditions involving algorithms can only be evaluated at the CS level, the retrieval version of the algorithm is always obtained for ISQ entries.

Determine if the algorithm transforms from AUC(s) to datafield(s) (DF-PARM) or from AUC(s) to an entire record (RT-PARM).

5.8.1 First find the AUC-PARM entry with the same AUC-NO as the AUCISM-LIST entry, and the CS-IS-ALG-USE-CODE. Using the MOD-ID, MOD-INSTANCE and ALG-USE-CODE retrieved from the AUC-PARM entry and the DB-ID, RT-ID of the AUCISM-LIST entry, access the CDM tables CMA and DF to determine if the AUC maps to datafield(s).

If no entry is found, access the CDM tables CMA and RT to determine if the AUC maps to a record type.

If the AUC does not map to either, reject the NDML statement (mapping not found). Exit Step 5.8.

5.8.2 Access the CDM table CMA using the MOD-ID, MOD-INSTANCE and ALG-USE-CODE to retrieve all the parameters of this complex mapping algorithm. Populate the CMA-ALG-ENTRY.

```
CMA-MOD-ID        = MOD-ID
CMA-MOD-INST      = MOD-INST
CMA-RETR-UPD      = ALG-USE-CODE
```

For each parameter retrieved:

5.8.3 Access the CDM tables PARM and UDDT to retrieve the parameters DATA-TYPE, SIZE and ND. Populate the CMA-PARM-ENTRY.

```
CMA-PARM-TYPE     = DATA TYPE
CMA-PARM-SIZE     = SIZE
CMA-PARM-ND       = ND
```

5.8.4 If the parameter is an AUC-PARM (attribute):
CMA-TAG-NO = AUCNO retrieved

If the parameter is a CONST-PARM (constant):
CMA-CONST-VAL = CONSTANT-VALUE retrieved
CMA-RT-NO = RT-NO retrieved, also

populate the CMA-DF-ENTRY with information about all the elementary datafields of this record, along with the type, size and decimal specification of each datafield. Access CDM tables DF and UDDT using the RT-NO parameter.

If the parameter is a DF-PARM (datafield):
CMA-DF-NO = DF-NO retrieved
CMA-RT-NO = RT-NO of DF-NO retrieved, also retrieve the data type, size and decimal

specification of the datafield by accessing the CDM tables DF and UDDT.

If the datafield is not elementary, issue a warning and set:

```
CMA-DF-TYPE              = DATATYPE of PARM
                           (CMA-PARM-TYPE)
CMA-DF-SIZE              = SIZE of PARM (CMA-PARM-SIZE)
CMA-DF-ND                = ND of PARM (CMA-PARM-ND)
Else
CMA-DF-TYPE              = TYPE of DF
CMA-DF-SIZE              = SIZE of DF
CMA-DF-ND                = ND   of DF
```

Note: An AUC can map to either a record or to datafield(s) but not both, through a complex mapping algorithm

5.8.5 Record the complex mapping algorithm that must be used.

Build an IS-QUALIFY-ENTRY for the AUC-PARM entry from Step 5.8.1 as follows:

```
ISQ-DBIDR        = DBID in the AUC-PARM entry
ISQ-RTIDR        = RTNO in the AUC-PARM entry
ISQ-DFNOR        = blank
ISQ-MAP-ALG-IDR  = MOD-ID in the AUC-PARM entry
ISQ-MAP-ALG-PTRR = index value of the
                   CMA-ALGORITHM-ENTRY
                   from Step 5.8.3
ISQ-PARM-NOR     = PARM NO found in Step 5.8.1
```

5.9 Fill in the TEMP-RECORD-TABLE as follows:

If IS-ACTION = 'S', '1', '2', OR 'K'

Scan the IS-QUALIFY-LIST, making a TRT-LIST entry for each distinct ISQ-DBIDR, ISQ-RTNOR encountered:

```
TRT-DBNO       = ISQ-DBIDR
TRT-RTNO       = ISQ-RTNOR
TRT-ECNO       = CSQ-ECNOR
```

If there is already a TRT-LIST entry for a TRT-DBID, TRT-RTNO, do not duplicate the entry.
Return to Step 5.6 to process the next AUCISM entry.

6a. Provide union discriminator values for record types that result from the union of entity classes.

If the CS-ACTION is 'S', '1', '2', or 'K', consider entries in both the IS-ACTION and IS-QUALIFY lists for the remainder of this step. If the CS-ACTION is 'M' or 'D' consider only the first entry in the IS-ACTION list for the remainder of this step. If the CS-ACTION is 'I' continue processing at Step 6.

For each IS-ACTION and IS-QUALIFY-LIST entry, find all the ECTRUD (E205) entries with:

```
ECNO                = CS-ECNO in the CS-QUALIFY-ENTRY
                      whose index = ISQ-CS-PTR or CS-ECNO
                      in the CS-ACTION-ENTRY whose index =
                      IS-CS-PTR and
either

DBID                = IS-DBID
RTNO                = IS-RTNO
or
DBID                = ISQ-DBIDL and
RTNO                = ISQ-RTNOL
or
DBID                = ISQ-DBIDR and
RTNO                = ISQ-RTNOR.
```

If no such ECRTUD entries are found, none of the record types result from unions and the rest of Step6a can be ignored.

Prepare an IS-QUALIFY-LIST entry from each ECRTUD entry found as follows:

```
ISQ-EC-NO           = EC-NO from the ECRTUD entry
ISQ-DBIDL           = DBID from the ECRTUD entry
ISQ-RTNOL           = RTNO from the ECRTUD entry
ISQ-DFNOL           = DFNO  from the ECRTUD entry
ISQ-OP              = COMPARISON_OP from the ECRTUD
ISQ-VARIABLE        = generated local variable
                      containing UNION-VALUE
                      from the ECRTUD entry
ISQ-DFL-KNOWN-FLAG  = DBMS access flag of ECRTUD entry
ISQ-TYPEL              = TYPE of ECRTUD entry
ISQ-SIZEL              = SIZE of ECRTUD entry
ISQ-NDL                = ND   of ECRTUD entry
ISQ-CSQ-PTR           = zero
ISQ-BOOLEAN            = Module Pre5A will handle the
                         'AND' and 'OR' logic for the
                         union discriminators
ISQ-TYPE2-SOURCE    = "U"
```

6. Determine additional joins that can be used to qualify data.

6.1 For each TK-KCNO in TEMP-KEY-LIST:

6.1.1 Find the TEMP-KEY-LIST entries with the TK-KCNO. In the rest of Step 6.1 consider this TK-KCNO only if all the entries just found have TK-REF-FLAG = 'Y'.

6.1.2 For each of the TEMP-KEY-LIST entries found in Step 6.1.1, modify the KEY-JOIN-LIST entry where KJ-TK-PTR = index of the TEMP-KEY-LIST entry:

Set KJ-RJ-OK = 'Y'

6.2 Eliminate any duplicate REPL-JOIN-LIST entries (entries that have exactly the same values).

For each remaining entry in the REPL-JOIN-LIST:

6.2.1 Find the KEY-JOIN-LIST entries with KJ-RJ-PTR = index of the REPL-JOIN-LIST entry. If all of these have KJ-RJ-OK = 'N', then modify the REPL-JOIN-LIST entry:

Set RJ-OK = 'N'

6.2.2 If RJ-DBNO and RJ-RTNO do not appear as the IS-DBNO or IS-RTNO of any entry in the IS-ACTION-LIST or as the ISQ-DBNOL or ISQ-RTNOL or the ISQ-DBNOR or ISQ-RTNOR of any entry in the IS-QUALIFY-LIST, then modify the REPL-JOIN-LIST entry:

Set RJ-OK = 'N'

6.2.3 If RJ-PTR-TYPE = 1 and
RJ-DBNO, RJ-RTNO and
IS-DBNO, IS-RTNO in
IS-ACTION-LIST (RJ-PTR)
equals
ISQ-DBNOL, ISQ-RTNOL and
ISQ-DBNOR, ISQ-RTNOR
(or vice versa) in (1-n)
IS-QUALIFY-LIST
then
modify the REPL-JOIN-LIST entry:

Set RJ-OK = 'N'

6.2.4 If RJ-OK = 'Y', build a corresponding IS-QUALIFY-LIST entry:

```
ISQ-NDML-NO = CSQ-NDML-NO for this
              request
ISQ-DBNOL   = RJ-DBNO
ISQ-RTNOL   = RJ-RTNO
ISQ-DFNOL   = RJ-DFNO
ISQ-TYPEL   = RJ-TYPE
ISQ-SIZEL   = RJ-SIZE
ISQ-NDL     = RJ-ND
ISQ-OP      = '='
ISQ-BOOLEAN = 'AND'
```

Use RJ-PTR-TYPE to determine the source for each of the following:

	= 1	= 2	= 3
ISQ-DBNOR =	IS-DBNO	ISQ-DBNOL	ISQ-DBNOR
ISQ-RTNOR =	IS-RTNO	ISQ-RTNOL	ISQ-RTNOR
ISQ-DFNOR =	IS-DFNO	ISQ-DFNOL	ISQ-DFNOR

using the entry in IS-ACTION-LIST (if RJ-PTR-TYPE = 1) or IS-QUALIFY-LIST (if RJ-PTR-TYPE = 2 or 3). Complete the right side data type, size and decimal specification.

7. Clean up the various tables and lists.

7.1 Determine if there are duplicate IS-ACTION-LIST

entries for a DELETE, MODIFY, or INSERT action. If this occurs, the NDML request because this indicates multiple update values for one data field which is not allowed. This is a warning message, until such time the CDM-1 as built model is modified, to prevent the same attribute being migrated from two separate independent entities to the same dependent entity.

Note:Do not eliminate any duplicate IS-ACTION-LIST entries for a SELECT action. The user is allowed to specify duplicate data items.

7.2 Eliminate any duplicate IS-QUALIFY-LIST entries, (any whose "left" and "right" sides match another's "left" and "right" sides), arbitrarily picking one to retain.

7.3 Eliminate any IS-QUALIFY-LIST entries whose "left" and "right" sides are another's "right" and "left" sides, and both have ISQ-OP equal to '=', arbitrarily picking one to retain.

8. Include type values in ISQ-TYPE and eliminate extraneous joins.

8.1 Modify the IS-ACTION-LIST and IS-QUALIFY-LIST to include type values for each phrase of the transaction.

Each entry in the IS-ACTION-LIST is designated Type 1.

Each entry in the IS-QUALIFY-LIST for which ISQ-VARIABLE is not blank is designated Type 2 (select-predicate).

Each Type 2 entry in the IS-QUALIFY is designated an ISQ-TYPE2-SOURCE where:

2I - source is index
2U - source is a union discriminator
2E - source is a user external schema qualification

For every Type 2 entry with ISQ-TYPE2-SOURCE equal blanks, assign ISQ-TYPE2-SOURCE = "E"

Each entry in the IS-QUALIFY-LIST for which ISQ-VARIABLE is blank and ISQ-OP is '=' is designated Type 3 (join predicate).

Each entry in the IS-QUALIFY-LIST for which ISQ-VARIABLE is blank and ISQ-OP is `U=' is designated Type 7 (outer join operator).

8.2 Eliminate any IS-QUALIFY-LIST Type 3 entries for which either side's DBNO or RTNO does not appear in another IS-ACTION-LIST or IS-QUALIFY-LIST entry. Note that the elimination of an entry may cause another entry to be removed that was previously considered to be needed.

Search all Type 3 IS-QUALIFY entries until all eliminations are completed. Proceed as follows:

8.2.1 Set ENTRY-CLEARED = 'N'

8.2.2 IF ENTRY-CLEARED = 'Y', search once again thru the IS-QUALITY entries, starting wih the first Type 3 entry.

8.2.3 If ENTRY-CLEARED = 'N', consider the next IS-QUALIFY entry.

Using the IS-QUALIFY, Type 3 entry selected:

8.2.4 If the ISQ left entry matches any IS-ACTION entry and the ISQ right entry matches any IS-ACTION entry, continue at Step 8.2.3.

8.2.5 If the ISQ left and right entry does not match any IS-ACTION entry, this qualification is unnecessary and should be deleted. Delete this IS-QUALIFY entry. Set ENTRY-CLEARED = 'Y'. Continue at Step 8.2.2.

8.2.6 If the ISQ left entry matches any IS-ACTION entry and the ISQ right entry matches any IS-QUALIFY LEFT or RIGHT ENTRY, continue at Step 8.2.3.

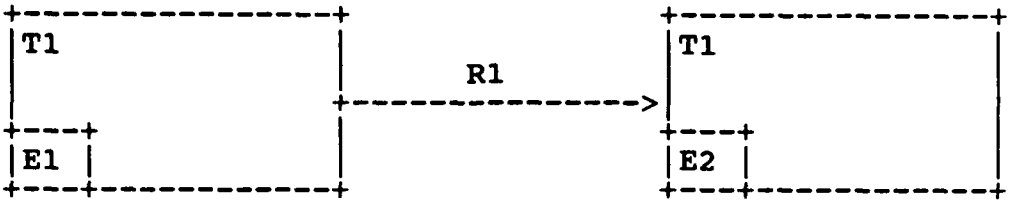
8.2.7 If the ISQ left entry matches any IS-QUALIFY left or right entry, and the ISQ right entry matches any IS-ACTION entry continue at Step 8.2.3.

8.2.8 Delete this IS-QUALIFY entry
Set ENTRY-CLEARED = 'Y'
Continue at Step 8.2.2

8.3 Eliminate a union discriminator Type 2 entry whose DBID does not appear in the IS-ACTION-LIST.

9. Identify the additional IS record sets that can be used in processing the request. These record sets map to relation classes for which the request specifies joins across owned and inherited key classes. Note that the intra-subtransaction joins for which the join fields are not owned/inherited key classes are handled by the Aggregator CI.

We are attempting to find the following case:



If the CSQ had the following qualification:

E1.T1 = E2.T1, which we will find by following the ISQ pointer back to the CSQ, find if the relation class (R1) maps to a record set.

Proceed as follows:

For each Type 3 entry in the IS-QUALIFY-LIST with ISQ-DBNOL = ISQ-DBNOR:

9.1 If either ISQ-RTNOL and ISQ-DFNOL or ISQ-RTNOR and ISQ-DFNOR are not filled in, eliminate the entry from further consideration in Step 9.

9.2 Follow ISQ-CSQ-PTR back to the CS-QUALIFY-LIST.

9.2.1 If it is not the case that:
there is an IAC (E7) with AUCNO = CSQ-AUCL and KC-MEMBER-AUC-NO = CSQ-AUCR or there is an IAC (E7) with AUCNO = CSQ-AUCR and KC-MEMBER-AUC-NO = CSQ-AUCL
then eliminate the entry from further consideration in Step 9.

9.2.2 Find the IAC (E7) entries with the same RCNO as the IAC (E7) found in Step 9.2.1. If any entry does not have a corresponding CS-QUALIFY-LIST entry with:

```
CSQ-AUCL or CSQ-AUCR = AUCNO
CSQ-VARIABLE         = blank
CSQ-OP               = '='
```

then eliminate the entry from further consideration in Step 9.

9.2.3 Find the RCSM (E109) entries with RCNO the same as the RCNO in the IAC (E7) found in Step 9.2.1.

Fill in the following fields in the IS-QUALIFY-LIST entry:

```
ISQ-NUM-RSL  = 1
ISQ-RSNOL(1) = RSNO of RCSM's key
```

9.2.4 Fill in a SET-TABLE entry for each of the RCSM (E109) entries found in Step 9.2.3:

```
ST-DBNO         = DBNO of RCSM's key
ST-RSNO         = RSNO of RCSM's key
ST-NUM-MEMBERS = 1
ST-MEMBER (1)   = MEMBER-RTNO of RCSM's
                  key
```

Note: If there already is a SET-TABLE entry with this ST-DBNO and ST-RSNO, do not duplicate it.

Find the RS (E72) entry with the same DBNO and RSNO as the RCSM. Continue to fill in the SET-TABLE entry:

```
ST-OWNER               = OWNER-RT-NO in RS
ST-TOTAL-NUM-MEMBERS = TOTAL-NUM-MEMBERS in RS
```

10. Group the pieces of the request into subtransactions, each of which represents a connected path through a database, by doing the following:

Initialize the GROUP-TABLE.
Perform until all IS-SUBTRANS-IDs in the IS-ACTION-LIST and all ISQ-SUBTRANS-IDLs in Type 2, 3, and 7 entries in the IS-QUALIFY-LIST and all ISQ-SUBTRANS-IDRs in Type 3 entries in the IS-QUALIFY-LIST are filled in:

10.1 Select an IS-DBID, IS-RTID from the IS-ACTION-LIST with the IS-SUBTRANS-ID not filled in. If there are none, go to Step 10.2, otherwise,

Set TEMP-DBID = DB-ID of IS-ACTION entry
TEMP-RTID = RT-ID of IS-ACTION entry
Go to Step 10.3 to begin grouping.

10.2 Select a ST-DBID, ST-RTID-OF-OWNER from the SET-TABLE with ST-SUBTRANS-ID not filled in. If there are none, go to Step 10.3 otherwise,

Set TEMP-DBID = DBID
TEMP-RTID = RTID-OF-OWNER

10.3 Start a new group by assigning it an identifier, which will be used as a SUBTRANS-ID.

10.4 Access the CDM tables DBH, SCH, PWD using the DBID of the current GROUP-TABLE entry to retrieve information about the database, dbms and host. Add an entry to the SUBTRANSACTION-PROCESS-ID-TABLE.

10.5 If the DBMS selected is NOT relational such as DB2 or ORACLE, continue at Step 10.6. A relational database is grouped into one simple subtransaction for a "Select" action.

For an update action, each unique record is grouped into a separate subtransaction.

10.5.1 Scan through the IS-ACTION-LIST. For each entry having IS-DBID = TEMP-DBID (IS-RTID = TEMP-RTID, if not update), assign the group-identifier to the IS-SUBTRANS-ID.

10.5.2 Scan through the IS-QUALIFY-LIST. For each entry having ISQ-DBIDL = TEMP-DBID (ISQ-RTIDL = TEMP-RTID, if not udate), assign the group-identifier to the ISQ-SUBTRANS-IDL. Also, for each entry having ISQ-DBIDR = TEMP-DBID (ISQ-RTIDR = TEMP-RTID, if not update), assign the group-identifier to the ISQ-SUBTRANS-IDR.

10.5.3 Return to Step 10.1 to select the next IS-ACTION entry.

10.6 Fill entries in the GROUP-TABLE with all related record types so they can be grouped into the same subtransaction.

10.6.1 Make a new entry in the group table
SET PIVOT-DBID = TEMP-DBID
SET PIVOT-RTID = TEMP-RTID
ADD 1 to GROUP-USED
SET GR-RTID = TEMP-RTID
SET GR-FLAG = 'N'

10.6.2 Scan through the SET-TABLE for an entry with ST-SUBTRANS-ID not filled in.

10.6.3 IF ST-DBID = PIVOT-DBID and
ST-RTID-OF-OWNER = PIVOT-RTID
Assign the group-identifier to ST-SUBTRANS-ID
SET TEMP-RTID = ST-RTID-OF-MEMBER
ADD TEMP-RTID to the GROUP-TABLE. Do not duplicate.
Set GR-FLAG = 'N'

10.6.4 IF ST-DBID = PIVOT-DBID and
ST-RTID-OF-MEMBER = PIVOT-RTID
SET TEMP-RTID = ST-RT-ID-OF-OWNER
ADD TEMP-RTID to the GROUP-TABLE.
Do not duplicate. Set GR-FLAG = 'N'
SET TEMP-RTID = ST-RTID-OF-MEMBER
ADD TEMP-RTID to the GROUP-TABLE
SET GR-FLAG = 'N'

10.6.5 Perform grouping for each entry in the GROUP-TABLE until grouping completed (all GR-FLAG = 'Y').
Set GR-INDEX to 1 and proceed as follows:

10.6.6 Scan the IS-ACTION list for each entry having IS-DBID = PIVOT-DBID and IS-RTID = PIVOT-RTID or if IS-RTID is not filled in the IS-STID = ST-SETID of the SET-TABLE having the owner or member record type as PIVOT-RTID.

If a match is found, assign the group-identifier to IS-SUBTRANS-ID.

10.6.7 Scan the left side of the IS-QUALIFY list for a match on PIVOT-DBID and PIVOT-RTID. If a match is found, assign the group-identifier to ISQ-SUBTRANS-IDL.

10.6.8 Scan the right side of the IS-QUALIFY-LIST for a match on PIVOT-DBID and PIVOT-RTID. If a match is found, assign the group-identifier to ISQ-SUBTRANS-IDR.

10.6.9 Set GR-FLAG = 'Y' for this GROUP-TABLE entry.

10.6.10 Check if the GROUP-TABLE is empty or all GR-FLAG = 'Y'.

10.6.11 If there are more entries to be processed:
Add 1 to GR-INDEX
Set PIVOT-RTID = GR-RTID
Perform Step 10.6.2, looking further for owner and member record matches in the SET-TABLE. Return to Step 10.6.6.

Otherwise, continue at Step 10.1.

10.7 For each Type 7 entry in the IS-QUALIFY-LIST:

If CSQ-RCNOR = zero in the CS-QUALIFY-ENTRY whose index = ISQ-CSQ-PTR, assign a new identifier to ISQ-SUBTRANS-IDR.

If that CSQ-RCNOR not = zero and not = CSQ-RCNOR for any prior IS-QUALIFY-ENTRY, assign a new identifier to ISQ-SUBTRANS-IDR.

If that CSQ-RCNOR not = zero but does = CSQ-RCNOR for a prior IS-QUALIFY-ENTRY, assign the identifier from that prior IS-QUALIFY-ENTRY to ISQ-SUBTRANS-IDR in this IS-QUALIFY-ENTRY.

Note that the "right" side of each Type 7 entry becomes a Subtransaction by itself unless it is a member of the same inherited key class as another. All members of the same inherited key class are placed in the same Subtransaction.

10.8 An Update action may result in multiple subtransactions regardless of whether the entity being updated "allows" or "disallows" update.

No special verification is peformed for an Insert NDML statement. All insert actions are considered updatable.

Validation needs to be performed for Delete and Modify Actions to verify that the records being updated contain the necessary qualification logic for each subtransaction. Proceed as follows: for TYPE-2 Qualifications.

10.8.1 Initialize local variables and internal tables.

10.8.2 Populate the table IS-ACTION-UPDATE-LIST (IAUL) by scanning the IS-Action for each unique database/record being updated.

10.8.3 Consider all IS-QUALIFY Type 2 enties resulting for each CS-QUALIFY entry.

10.8.4 There must exist a match where IAUL-SUBTRANS = ISQ-SUBRANS-IDL for each IAUL-entry populated.

10.8.5 If a match is not found, issue an error message that no qualification was found to update a particular record.

10.9 Proceed as follows for Type 3 Qualifications. Type 3 Qualifications will exist if a Delete or Modify Action has a "Using" clause.

10.9.1 Determine if a Using clause has been specified. If a "Using" entity is not found, exit Step 10.9.

10.9.2 Scan the IS-Action List for entity(s) specified on the using clause and the join qualifications across the using entity and entity bein updated. If this is a relational DBMS, populate the table USING-ENTITY-LIST (UEL) with information about the "using" entities.

10.9.3 If this is a non-relational subtrans, there must exist a set between the USING-ENTITY and the UPDATE-ENTITY, and will consequently result in a 1 subtransaction. Validate that the statement is true.

10.9.4 If this is a relational DBMS, extra processing is necessary. Each unique record has been grouped into its own subtransaction, i.e. the USING-ENTITY and UPDATE-ENTITY are in 2 distinct subtransactions. Validate that a join does exist between these two subtransactions and collapse them into the single subtransaction. Take care to duplicate IS-Action entries if necessary.

10.9.5 If the relevant join is not found in Step 10.9.3 or 10.9.4, issue an error stating that a legal join does not exist between the USING and UPDATE entities.

11. Distinguish between the inter- (Type 4) and intra-(Type 3) Subtransaction joins.

For each Type 3 entry in the IS-QUALIFY-LIST, if ISQ-SUBTRANS-IDL is not = ISQ-SUBTRANS-IDR, then change the entry's ISQ-TYPE to be Type 4.

12. IF IS-ACTION = 'S' or 'Q' or '1' or '2' or 'K', put all result fields into the IS-ACTION-LIST and RFT:

12.1 If it is not already there, add to the IS-ACTION-LIST each of the Type 4 and Type 7 halves.

Fill in the corresponding fields from the left or right half of the IS-QUALIFY-LIST entry. Also, set

```
IS-SOURCE          = blank
IS-ISQ-L-R         = "L" or "R" depending on whether
                     this entry is built from the
                     IS-QUALIFY Left or Right
IS-ISQ-PTR         = ISQ-INDEX
```

12.2 For each entry in the IS-ACTION-LIST with non-blank IS-CS-PTR, add an entry to the RFT.

Follow the IS-CS-PTR to find the corresponding entry in the CS-ACTION-LIST.

```
RFT-SUBTRANS = IS-SUBTRANS-ID
RFT-ATTR     = AUC
RFT-SIZE     = CS-SIZE
RFT-IS-PTR   = index of the IS-ACTION-LIST entry
RFT-TYPE     = CS-DATATYPE
RFT-ND       = CS-ND
```

12a. Check for multiple sets of nested repeating data fields in the same subtransaction.

Check if the following subcomponent datafield is referenced:

```
03  D1  OCCURS 10.
    04  E2  PIC X.
    04  E3.
        05  E4  PIC X
        05  E5  PIC X

03  F1.
    04  F2  PIC X
    04  F3  OCCURS 10,
        05  F4  PIC X
        05  F5  PIC X
```

Note: Selection of E2 and E4 and E5 is allowable. Selection of E2, E4 and F4 is not allowable. All fields selected per subtransaction must have the same parentage chain.

e.g. E2's parentage is D1
E4's parentage is D1 E3
E5's parentage is D1 E3
F4's parentage is F1 F3

For each IS-ACTION-ENTRY with IS-DF-REPEAT-FLAG not = 'N':

12a.1 Find the TOT-OCCURS-NEST entry with TOT-IS-PTR = index for the current IS-ACTION-ENTRY.

Set OT-INDEX-2 to zero.

Find an OT-SUBTRANS entry with OT-SUBTRANS-ID = IS-SUBTRANS-ID. If one is not found, go to Step 12a.3.

12a.2 Compare a TOT-OCCURS-STACK entry to an OT-OCCURS-LEVEL entry.

If this TOT-OCCURS-STACK is empty, go to Step 12a.1 to check the next IS-ACTION-ENTRY for a repeating data field.

Pop the TOT-OCCURS-STACK into TEMP-OCCURS-ENTRY.

Increment OT-INDEX-2. If OT-INDEX-2 is greater than OT-LEVELS-USED, go to Step 12a.4.

If TEMP-OCCURS-ENTRY = OT-OCCURS-LEVEL (OT-INDEX-1, OT-INDEX-2), go to Step 12a.2.

If this point is reached, two different sets of nested repeating data fields are being called for. Reject the NDML statement (cannot retrieve from more than one set of nested repeating data fields).

12a.3 Begin a new OT-SUBTRANS entry.

Set OT-SUBTRANS-ID = IS-SUBTRANS-ID.

Set OT-LEVELS-USED = zero.

Go to Step 12a.2.

12a.4 Build a new OT-OCCURS-LEVEL entry.

Set OT-OCCURS-LEVEL (OT-INDEX-1, OT-INDEX-2) to TEMP-OCCURS-ENTRY.

Increment OT-LEVELS-USED.

Go to Step 12a.2.

13. Build the JQG if CS-ACTION = 'S' or 'Q' or '1' or '2' or 'K'.

13.a For each Type 4 or Type 7 entry in the IS-QUALIFY-LIST, fill in a JQG entry:

13.1 If the set of CS-PTR values in the IS-ACTION-LIST entries corresponding to the left side of a Type 4 entry is exactly the same as the set of CS-PTR values in the IS-ACTION-LIST entries corresponding to the right side of the Type 4 entry

JQG-EDGE-TYPE = 5
else JQG-EDGE-TYPE = 4.

The CS-PTR values would be identical if we are attempting to qualify on two horizontally partitioned fragments of an entity. If the operator was left as an equal operator (=), no rows of data would be retrieved for the NDML request. This situation consequently should become a union.

13.2 For a Type 7 entry, JQG-EDGE-TYPE = 7.

13.3 JQG-SUBTRANS-IDL = ISQ-SUBTRANS-IDL
JQG-SUBTRANS-IDR = ISQ-SUBTRANS-IDR

13.4 Make an entry in the JQG-ATTRIBUTE-PAIR-LIST for each of the matching fields if the data types of the attributes of the corresponding CS-QUALIFY-LIST entry are compatible (i.e. a character type (C) must only be compared to a character type and a numeric type (N,S) must be compared to a numeric type).

If the data types are compatible, make an entry in the JQG-ATTRIBUTE-PAIR-LIST as follows:

JQG-ATTRL = CSQ-AUCL
JQG-ATTRR = CSQ-AUCR

13.b Create JQG union entries for horizontally-partitioned CS entity classes.

Scan the IS-ACTION-LIST for entries containing identical IS-CS-PTR values. For each set of entries found, fill in a JQG entry, such that each element of the set is "connected" to one other:

JQG-EDGE-TYPE = 5
JQG-SUBTRANS-IDL = IS-SUBTRANS-ID from one of the pair
JQG-SUBTRANS-IDR = IS-SUBTRANS-ID from the other of the pair

14. Clean up the JQG.

If there are JQG entries with the same edge type and identical pairs of JQG-SUBTRANS-IDL and JQG-SUBTRANS-IDR values, combine their JQG-ATTRIBUTE-PAIR-LIST entries and collapse the JQG entries into a single JQG entry.

15. If CS-ACTION not 'S' or 'Q' or '1' or '2' or 'K' or 'I' or 'M' or 'D', then fill in the first entry of the IS-ACTION-LIST as follows:

IS-ACTION = CS-ACTION
IS-NDML-NO = CS-NDML-NO

Leave all other fields in the entry blank and leave all other entries blank. Proceed to Step 20.

Steps 16 through 19 must be performed at the completion of function PRE5 to insure that each subtransaction identified is valid and that the join query graph will have enough information to combine the results of all subtransactions.

16. Check that each subtransaction appears in at least one entry in the IS-ACTION-LIST.

16.1 For each subtransaction identified for the request (SUB-USED), search the IS-ACTION-LIST for an entry having the same subtransaction identification.

IS-SUBTRANS-ID = current SUB-INDEX

16.2 If no matching entries are found, issue an error message and reject the NDML request.

17. Check that the join query graph built for this NDML request is complete and will be able to combine the results of all subtransactions.

17.1 For each subtransaction identified for the request, search the JQG for an entry having the same subtransaction identification.

JGQ-SUBTRANS-IDL = current SUB-INDEX
or
JQG-SUBTRANS-IDR = current SUB-INDEX

17.2 If no matching entries are found, issue an error message and reject the NDML request.

18. Check that an NDML update request did not result in distributed transactions.

If IS-ACTION = 'I', 'M', or 'D', and SUB-USED is greater than 1, issue a warning message for this NDML request.

19. Determine if an NDML MODIFY or DELETE action for a SQL DBMS has both a "USING" clause and any complex mapping algorithms in the WHERE clause.

19.1 Determine if the DBMS for a MODIFY or DELETE action is SQL based.

If the DBMS for the subtransaction is not "ORACLE" or "DB2", continue processing at Step 20.

19.2 Determine if the NDML transaction has a "USING" clause.

If a "USING" clause was not specified, continue processing at Step 20.

19.3 Determine if any qualification entries participate in complex mapping algorithms.

If each used left and right half entry in the IS-QUALIFY-LIST does not participate in a complex mapping algorithm, continue processing at Step 20.

ISQ-MAP-ALG-PTRL = 0 and
ISQ-MAP-ALG-PTRR = 0
This must be true for each half.

If one entry is found to participate in a complex mapping algorithm, issue an error message and reject the NDML request. This is true if any

ISQ-MAP-ALG-PTRL NOT = 0 or
ISQ-MAP-ALG-PTRR NOT = 0

20. Complete precompilation of this conceptual transaction.

20.1 Invoke function PRE5A to determine what qualifications are evaluatable at the internal schema level.

20.2 Invoke function PRE5B to remove retrieval entries from identified CDMP tables that are evaluatable at the internal schema level.

20.3 Invoke function PRE13 to control the generation of the code to satisfy the NDML request. When PRE13 is finished return to PRE4.

21. Upon successful precompilation of the conceptual transaction by PRE13, store all cross references from the generated software module to the internal schema objects.

21.1 For every entry in the IS-ACTION-LIST, store a DFU (E300) entry. The IS-ACTION-LIST contains the data field object number (IS-DFNO). The MOD-ID of (E300) is the generated module name. This may be found by using the CGT-MOD-NAME of the CGT where the IS-SUBTRANS-ID = CGT-SUBTRANS-ID and the CGT-CASE-NO = the NDML-COUNTER.

The DF-USAGE-CODE will be the value I, M, D, or S from the IS-ACTION.

21.2 For every data field entry (left or right) in the IS-QUALIFY-LIST, store a DFU (E300) entry. The IS-QUALIFY-LIST contains the data field object number (ISQ-DFNOL and ISQ-DFNOR). The MOD-ID of (E300) is the generated module name. This may be found by using the CGT table as in step 18.1, but with the ISQ-SUBTRANS-IDL or ISQ-SUBTRANS-IDR.

21.3 For every entry in the SET-TABLE, store an RSU (E299) entry. The SET-TABLE contains the record set object number (ST-RSNO). The MOD-ID of (E300) is the generated module name. This may be found by using the CGT table as in Step 17.1 and the ST-SUBTRANS-ID.

15.3 Constraints

This algorithm does not accommodate row-wise derivation of attributes in the CS-IS mapping. This type of derivation is not implemented in this release.

Replication is handled by designating in the CDM a primary source for each replicated data field.

For Retrieval

If Entity Retrieval Rule = "ALLOW", a secondary copy is considered if the requesting process is on the same host. If a copy is not available on the local host, the primary copy is retrieved.

If the Entity Retrieval Rule = "DISALLOW", only the primary copy is considered.

For Update

If the Entity Update Rule = "DISALLOW", only the primary copy is updated.

If the Entity Update Rule = "ALLOW", update transactions are generated for the primary and secondary copies or sources of data.

Note, that if a CS attribute use class maps to more than one IS record set, then all of those record sets must have the same owner record type and same member record types.

15.4 Outputs

1. IS NDML Subtransactions represented by the IS-ACTION-LIST and IS-QUALIFY-LIST. Each of these subtransactions is in NDML format and accesses one database. These will be input to function PRE6 - Select IS Access Path, and are in IS terms.

```
01 IS-ACTION-LIST.
   input to pre6
   03 IS-MAX                          PIC 99 VALUE 60.
   03 IS-USED                         PIC 99 VALUE 0.
   03 IS-ST-MAX                       PIC 99 VALUE 25.
   03 IS-LOCK                         PIC X.
      88 IS-SHARED-LOCK               VALUE "S".
      88 IS-EXCLUSIVE-LOCK            VALUE "X".
      88 IS-NO-LOCK                   VALUE "N".
   03 IS-ACTION                       PIC X.
      88 IS-MODIFY-ACTION             VALUE "M".
      88 IS-DELETE-ACTION             VALUE "D".
      88 IS-INSERT-ACTION             VALUE "I".
      88 IS-SELECT-ACTION             VALUE "S".
      88 IS-SELECT-COMB               VALUE "Q".
      88 IS-REF-INTEG-1               VALUE "1".
      88 IS-REF-INTEG-2               VALUE "2".
      88 IS-UNIQUE-KEY                VALUE "K".
      88 IS-BEGIN-ACTION              VALUE "B".
      88 IS-COMMIT-ACTION             VALUE "C".
      88 IS-ROLLBACK-ACTION           VALUE "R".
      88 IS-NEXT-CONT                 VALUE "N".
      88 IS-END-CURLY                 VALUE "E".
      88 IS-EXIT-BREAK                VALUE "X".
   03 IS-NDML-NO                      PIC 9(6).
      Segregator of CS NDML statements and controller
of case structues in generated request processors
   03 IS-RETR-ENTRY                   OCCURS 60 TIMES
                                      INDEXED BY
```

IS-INDEX
One entry per column on select and modify, giving primary data field.
One entry per data field in mapped-to record type for insert and delete, some without corresponding AUCs in CS/ES.

```
05 IS-SUBTRANS-ID                PIC 999.
05 IS-META.
   07 IS-DBID                    PIC 9(6).
   07 IS-RTID                    PIC X(30).
   07 IS-RTNO                    PIC 9(6).
   07 IS-DFID                    PIC X(30).
   07 IS-DFNO                    PIC 9(6).
   07 IS-DATA-TYPE               PIC X.
   07 IS-SIZE                    PIC 999.
   07 IS-ND                      PIC 99.
05 IS-LOCAL-VARIABLE             PIC X(64).
   source field for insert, modify target field
   for select
05 IS-UNION-DISC-VAR             REDEFINES
                                 IS-LOCAL-VARIABLE.
   07 IS-UNION-VALUE             PIC X(30).
   07 FILLER                     PIC X(34).
05 IS-MAPPED-TO-FLAG              PIC X.
   88 IS-MAPPED-TO               VALUE "Y".
   88 IS-NOT-MAPPED-TO           VALUE "N".
   distinguishes between datafields in record
   type that have AUC counterpats and those that
   do not
05 IS-CS-PTR                     PIC 999.
05 IS-ISQ-PTR                    PIC 999.
05 IS-ISQ-LR                     PIC X.
05 IS-FLAG                       PIC 9.
   88 HAS-ACCESS-SPEC            VALUE 1.
   filled in by pre6
05 IS-FLAG-CONV                  PIC X.
   88 ISCS-ALG                   VALUE "A" "K".
   88 PRE6-USED                  VALUE "*".
   88 UN-USED-ENTRY              VALUE " ".
   88 ALGORITHM-CONVERSION       VALUE "A".
   88 CONSTRAINT-CHECK           VALUE "C".
05 IS-TYPE                       PIC X.
   88 TARGET-VALUE               VALUE "1".
05 IS-RFT-PTR                    PIC 999.
05 IS-FCTN-NAME                  PIC X(5).
05 IS-FCTN-DISTINCT              PIC X.
   88 APPLY-DISTINCT             VALUE "Y".
05 IS-KEYFLAG                    PIC X.
   Filled in by pre6 to denote keys for direct
access
05 IS-ALG-ID                     PIC X(8).
05 IS-PARM-NO                    PIC 999.
05 IS-ALG-PTR                    PIC 999.
05 IS-DF-KNOWN-FLAG              PIC X.
   88 IS-DF-KNOWN                VALUE "K".
   88 IS-DF-UNKNOWN              VALUE "U".
05 IS-DF-REPEAT-FLAG             PIC X.
   88 IS-DF-DOESNT REPEAT        VALUE "N".
```

```
            88 IS-DF-REPEATS                VALUE "Y".
            88 IS-DF-RG-COMP                VALUE "C".
            88 IS-DF-USE-INDEX              VALUE "I".
         05 IS-SOURCE                       PIC X.
            88 IS-GENERATED                 VALUE "G".
            88 IS-USER                      VALUE SPACE.
         05 IS-DELETE-FLAG                  PIC 9.
            88 IS-DELETED                   VALUE 1.
         05 IS-ST-USED                      PIC 99.
      03 IS-RETR-ENTRY1                     OCCURS 25 TIMES
                                            INDEXED BY
                                            IS-INDEX1.
         05 IS-RSNO                         PIC 9(6).
         05 IS-STID                         PIC X(30).
         05 IS-ST-VALUE                     PIC X(30).
         05 IS-INDEX-PTR                    PIC 9(3).

01 IS-QUALIFY-LIST.
   input to pre6
      03 ISQ-MAX                          PIC 99 VALUE 40.
      03 ISQ-USED                         PIC 99 VALUE 0.
      03 ISQ-ST-MAX                       PIC 99 VALUE 25.
      03 ISQ-NDML-NO                      PIC 99.
      03 ISQ-ENTRY                        OCCURS 40 TIMES
                                          INDEXED BY
                                          ISQ-INDEX.
      One entry per WHERE clause entry + one entry per
          CS-ES join structure on select
      One entry per WHERE clause entry on modify, delete
      Not used on insert, begin, commit, or rollback
         05 ISQ-CSQ-PTR                   PIC 999.
         05 ISQ-RJ-PTR                    PIC 999.
         05 ISQ-KEYFLAG                   PIC X.
            88 IS-PRIMARY-KEY             VALUE "P".
            88 IS-SECONDARY-KEY           VALUE "S".
            88 IS-NOT-KEY                 VALUE SPACE.
               filled in by pre6 to find access ports
         05 ISQ-VARIABLE                  PIC X(64).
         05 ISQ-UNION-DISC-VAR            REDEFINES ISQ-VARIABLE.
            07 ISQ-UNION-VALUE            PIC X(30).
            07 FILLER                     PIC X(34).
         05 ISQ-OP                        PIC XX.
         05 ISQ-BOOLEAN                   PIC X(7).
         05 ISQ-TYPE                      PIC 9.
            88 SELECT-PREDICATE                VALUE 2.
            88 INTRASUBTRANS-JOIN-PREDICATE    VALUE 3.
            88 INTERSUBTRANS-JOIN-PREDICATE    VALUE 4.
            88 INTERSUBTRANS-UNION             VALUE 5.
            88 OUTER-JOIN-PREDICATE            VALUE 7.
         05 ISQ-TYPE2-SOURCE              PIC X.
            88 SOURCE-IS-EXTERNAL              VALUE "E".
            88 SOURCE-IS-UNION                 VALUE "U".
            88 SOURCE-IS-INDEX                 VALUE "I".
         05 ISQ-EVAL-FLAG                 PIC 9.
         05 ISQ-BOOLEAN-PTR               PIC 999.
         05 ISQ-LEFT-META.
            07 ISQ-DBIDL                  PIC 9(6).
            07 ISQ-RTIDL                  PIC X(30).
            07 ISQ-RTNOL                  PIC 9(6).
```

```
            07 ISQ-DFIDL                    PIC X(30).
            07 ISQ-DFNOL                    PIC 9(6).
            07 ISQ-TYPEL                    PIC X.
            07 ISQ-SIZEL                    PIC 999.
            07 ISQ-NDL                      PIC 99.
         05 ISQ-SUBTRANS-IDL                PIC 999.
         05 ISQ-LEFT                        PIC 999.
            88 HAS-ACCESS-SPEC-L            VALUE 1.
               filled in by pre6
         05 ISQ-ALG-IDL                     PIC X(8).
         05 ISQ-PARM-NOL                    PIC 999.
         05 ISQ-ALG-PTRL                    PIC 999.
         05 ISQ-DFL-KNOWN-FLAG              PIC X.
            88 IS-DFL-KNOWN                 VALUE "K".
            88 IS-DFL-UNKNOWN               VALUE "U".
         05 ISQ-EC-NO                       PIC 9(6).
         05 ISQ-RIGHT-META.
            07 ISQ-DBIDR                    PIC 9(6).
            07 ISQ-RTIDR                    PIC X(30).
            07 ISQ-RTNOR                    PIC 9(6).
            07 ISQ-DFIDR                    PIC X(30).
            07 ISQ-DFNOR                    PIC 9(6).
            07 ISQ-TYPER                    PIC X.
            07 ISQ-SIZER                    PIC 999.
            07 ISQ-NDR                      PIC 99.
         05 ISQ-SUBTRANS-IDR                PIC 999.
         05 ISQ-RIGHT                       PIC 999.
            88 HAS-ACCESS-SPEC-R            VALUE 1.
               filled in by pre6
         05 ISQ-ALG-IDR                     PIC X(8).
         05 ISQ-PARM-NOR                    PIC 999.
         05 ISQ-ALG-PTRR                    PIC 999.
         05 ISQ-DFR-KNOWN-FLAG              PIC X.
            88 IS-DFR-KNOWN                 VALUE "K".
            88 IS-DFR-UNKNOWN               VALUE "U".
         05 ISQ-STL-USED                    PIC 99.
         05 ISQ-STR-USED                    PIC 99.
      03 ISQ-ENTRY1 OCCURS 25 TIMES INDEXED BY ISQ-INDEX1.
            07 ISQ-STIDL                    PIC X(30).
            07 ISQ-RSNOL                    PIC 9(6).
            07 ISQ-STL-VALUE                PIC X(30).
            07 ISQ-INDEX-L                  PIC 9(3).
      03 ISQ-ENTRY2 OCCURS 25 TIMES INDEXED BY ISQ-INDEX2.
            07 ISQ-STIDR                    PIC X(30).
            07 ISQ-RSNOR                    PIC 9(6).
            07 ISQ-STR-VALUE                PIC X(30).
            07 ISQ-INDEX-R                  PIC 9(3).
```

2. Join Query Graph. Each node represents an intermediate relation that will result from a single Subtransaction. Each edge represents the action to be taken to match rows of the intermediate relations. The actions will be performed by the Aggregator CI. Their sequence will be scheduled by the Distributed Request Supervisor CI. The JQG and JQG-ATTRIBUTE-PAIR-LIST are input to the Distributed Request Supervisor at run-time and to function PRE10 - Build Calls and Messages at precompile-time.

```
01 JQG.
   input to DRS and PRE10
   03 JQG-MAX                          PIC 99 VALUE 30.
   03 JQG-USED                         PIC 99 VALUE 0.
   03 JQG-EDGE OCCURS 30 TIMES INDEXED BY JQG-INDEX.
      05 JQG-EDGE-TYPE                 PIC X.
         88 JQG-UNION                  VALUE "5".
         88 JQG-JOIN                   VALUE "4".
         88 JQG-NOT                    VALUE "6".
         88 JQG-OUTER-JOIN             VALUE "7".
         88 JQG-DELETED                VALUE "*".
      05 JQG-JOIN-PTR-TOP              PIC 999.
      05 JQG-SUBTRANS-IDL              PIC 999.
      05 JQG-SUBTRANS-IDR              PIC 999.

01 JQG-ATTRIBUTE-PAIR-LIST.
     accompanies JQG
   03 APL-MAX                          PIC 99 VALUE 60.
   03 APL-USED                         PIC 99 VALUE 0.
   03 APL-ROW-SIZE                     PIC 99 VALUE 22.
   03 APL-ROW OCCURS 60 TIMES INDEXED BY APL-INDEX.
      05 JQG-SUBTRANSL                 PIC 999.
      05 JQG-ATTRL                     PIC 9(6).
      05 JQG-SUBTRANSR                 PIC 999.
      05 JQG-ATTRR                     PIC 9(6).
      05 JQG-NEXT-PTR                  PIC 99.
      05 JQG-OP                        PIC XX.
```

3. Result Field Table. Each entry describes an attribute in an intermediate relation, with the identifier of the process that creates it.

   The RFT is input to the Distributed Request Supervisor CI at run-time and to function PRE10 - Build Calls and Messages at precompile-time.

```
01 RFT.
   Contains all result fields (anything to be
   transferred by the NTM) and their creating
   application process. Input to PRE10 and DRS.

   03 RFT-MAX                          PIC 999 VALUE 200.
   03 RFT-USED                         PIC 999 VALUE 0.
   03 RFT-ROW-SIZE                     PIC 999 VALUE 24.
   03 RFT-ENTRY OCCURS 200 TIMES INDEXED BY RFT-INDEX.
      05 RFT-PID                       PIC 9(6).
      05 RFT-SUBTRANS                  PIC 999.
      05 RFT-ATTR                      PIC 9(6).
      05 RFT-SIZE                      PIC 999.
      05 RFT-IS-PTR                    PIC 999.
      05 RFT-TYPE                      PIC X.
      05 RFT-ND                        PIC 99.
```

4. Set Table. Each entry describes a record set that must be traversed in processing a subtransaction. The SET-TABLE is input to function PRE6 - Select IS Access Path.

```
01 SET-TABLE.
   input to pre6
   03 SET-MAX                              PIC 99 VALUE 25.
   03 SET-USED                             PIC 99 VALUE 0.
   03 ST-ENTRY OCCURS 25 TIMES INDEXED BY ST-INDEX.
      05 ST-DBID                           PIC 9(6).
      05 ST-RSNO                           PIC 9(6).
      05 ST-OWNER-ID                       PIC 9(6).
      05 ST-SETID                          PIC X(30).
      05 ST-OWNER                          PIC X(30).
      05 ST-FLAG                           PIC X.
      05 ST-SUBTRANS-ID                    PIC 9(3).
      05 ST-MARK                           PIC X.
         88 HAS-ACCESS-SPEC-S             VALUE "Y".
            filled in by pre6
      05 ST-MEMBER                         PIC X(30).
      05 ST-MEMBER-ID                      PIC 9(6).
      05 ST-MAPPING                        PIC 9(6).
         88 AUC-SET-VALUE                  VALUE 1.
         88 RC-BASED-REC-SET               VALUE 2.
```

5. Subtransaction processes ID table.

   This table identifies the subtransactions used for this NDML statement.

```
01 SUBTRANS-PROCESS-ID-TABLE.
03 SUB-MAX                                PIC 99 VALUE 50.
03 SUB-USED                               PIC 99 VALUE 0.
03 SUBTRANS                               OCCURS 50 TIMES
                                          INDEXED BY
                                          SUB-INDEX.
   05 STR-PROCESS-ID                      PIC X(10).
   05 STR-DBMS-NAME                       PIC X(30).
   05 STR-HOST-ID                         PIC XXX.
   05 STR-DB-NAME                         PIC X(30).
   05 STR-LIBRARY-NAME                    PIC X(30).
   05 STR-DBID                            PIC 9(6).
   05 STR-PASSWORD                        PIC X(30).
   05 STR-DB-LOCATION                     PIC X(30).
   05 STR-SCHEMA                          PIC X(30).
   05 STR-SUBSCHEMA                       PIC X(30).
   05 STR-CHAR-NULL-VALUE                 PIC X(30).
   05 STR-INTG-NULL-VALUE                 PIC X(30).
   05 STR-LOCALITY                        PIC X.
   05 STR-NTM-DIRECT                      PIC XX.
```

6. IS OCCURS-TABLE. This table identifies the set of nested repeating data fields, if any, that are involved in each subtransaction.

```
01 OCCURS-TABLE.
   03 OT-SUBTRANS-USED                    PIC 99.
   03 OT-SUBTRANS-MAX                     PIC 99 VALUE 25.
   03 OT-STACK-MAX                        PIC 9 VALUE 4.
   03 OT-OCCURS-NEST OCCURS 25 TIMES      INDEXED BY
                                          OT-INDEX-1.
05 OT-SUBTRANS                           PIC 999.
      05 OT-RTNO                          PIC 9(6).
```

```
           05 OT-NEST-ID                    PIC 99.
           05 OT-MAPPED-TO                  PIC X.
           05 OT-INDEX-LEVELS               PIC 9.
           05 OT-STACK-USED                 PIC 9.
           05 OT-DFNO-STACK OCCURS 4 TIMES INDEXED BY
                                            OT-INDEX-2.
              07 OT-DFNO                    PIC 9(6).
              07 OT-COMP-DFNO               PIC 9(6).
              07 OT-OCCURS-DEP-DFNO         PIC 9(6).
              07 OT-INDEX-DFNO              PIC 9(6).
              07 OT-NUM-OCCURS              PIC 9(4).
              07 OT-LEVEL-NO                PIC 9.
```

7. Complex Mapping Algorithm Table. This table identifies the software modules and parameters that are needed to perform complex mappings between CS and IS formats.

```
01 COMPLEX-MAPPING-ALG-TABLE.
  03 CMA-MAX                              PIC 99 VALUE 10.
  03 CMA-USED                             PIC 99 VALUE 0.
  03 CMA-ALG-ENTRY                        OCCURS 10 TIMES
                                          INDEXED BY CMA-INDEX.
     05 CMA-MOD-ID                        PIC X(8).
     05 CMA-MOD-INST                      PIC 999.
     05 CMA-RETR-UPD                      PIC X.
     05 CMA-PARM-COUNT                    PIC 99.
     05 CMA-FLAG                          PIC X.
        88 PARM-GENERATED                 VALUE "Y".
        88 NO-PARM-GENERATED              VALUE "N".
     05 CMA-SUBTRANSACTION                PIC 999.
     05 CMA-PARM-ENTRY OCCURS 10 TIMES    INDEXED BY
                                          CMA-PARM-INDEX.
         07 CMA-PARM-NO                   PIC 99.
         07 CMA-TAG-NO                    PIC 9(6).
         07 CMA-RTID                      PIC X(30).
         07 CMA-RT-NO                     PIC 9(6).
         07 CMA-DFID                      PIC X(30)
         07 CMA-DF-NO                     PIC 9(6).
         07 CMA-CONST-VAL                 PIC X(30).
         07 CMA-PARM-TYPE                 PIC X.
         07 CMA-PARM-SIZE                 PIC 999.
         07 CMA-PARM-ND                   PIC 99.
         07 CMA-DF-TYPE                   PIC X.
         07 CMA-DF-SIZE                   PIC 999.
         07 CMA-DF-ND                     PIC 99.
         07 CMA-DBID                      PIC 9(6).
  03 CMA-DF-COUNT                         PIC 999 VALUE 100.
  03 CMA-DF-ENTRY                         OCCURS 100 TIMES
                                          INDEXED BY
                                          CMA-DF-INDEX.
     05 DF-DFNO                           PIC 9(6).
     05 DF-DFID                           PIC X(30).
     05 DF-TYPE                           PIC X.
     05 DF-SIZE                           PIC 999.
     05 DF-ND                             PIC 99.
     05 DF-MOD-PTR                        PIC 99.
     05 DF-PARM-PTR                       PIC 99.
```

8. SUBTRANS-BOOLEAN-LIST

Contains all the BOOLEAN operators, parentheses, and conditions which can be satisfied at the Internal Schema level, for each subtransaction.

```
01 SUBTRANS-BOOLEAN-LIST.
   03 SBL-MAX                          PIC 999 VALUE 300.
   03 SBL-USED                         PIC 999.
   03 SBL-ENTRY                        OCCURS 300 TIMES
                                       INDEXED BY SBL-INDEX.
      05 SBL-SUBTRANS                  PIC 999.
      05 SBL-OP                        PIC XXX.
      05 SBL-ISQ-PTR                   PIC 999.
      05 SBL-TYPE                      PIC XX.
         88 SBL-TYP2-QUAL              VALUE "2E".
         88 SBL-RECORD-UNION           VALUE "2U".
         88 SBL-TYPE3-QUAL             VALUE "3 ".
      05 SBL-RTNO                      PIC 9(6).
```

15.5 Internal Data Requirements

1. Temp-Record-Table

This table is used to temporarily store information about which record types are mapped to in an INSERT or DELETE request.

```
01 TEMP-RECORD-TABLE.
```

Used to ensure that inserts and deletes properly handle all data fields on records that an entity class partially maps to

```
   03 TRT-MAX                          PIC 999 VALUE 25.
   03 TRT-USED                         PIC 99.
   03 TRT-ENTRY                        OCCURS 25 TIMES
                                       INDEXED
                                       BY TRT-INDEX.
      05 TRT-ECNO                      PIC 9(6).
      05 TRT-DBID                      PIC 9(6).
      05 TRT-RTID                      PIC X(30).
      05 TRT-RTNO                      PIC 9(6).
```

2. Replication Tables

The following three tables are used to find replicated key fields that can be used for inter-subtransaction joins.

```
01 REPL-JOIN-LIST.
   Used to add joins taking advantage of key
   replication across databases
   03 REPL-MAX                         PIC 99 VALUE 25.
   03 REPL-USED                        PIC 99 VALUE 0.
   03 RJ-ENTRY OCCURS 25 TIMES INDEXED BY RJ-INDEX.
      05 RJ-DBID                       PIC 9(5).
      05 RJ-RTID                       PIC X(30).
      05 RJ-DFID                       PIC X(30).
      05 RJ-DFNO                       PIC 9(9).
```

```
           05 RJ-RTNO                    PIC 9(9).
           05 RJ-TYPE                    PIC X.
           05 RJ-SIZE                    PIC 999.
           05 RJ-ND                      PIC 99.
           05 RJ-PTR-TYPE                PIC 9.
              88 RJ-ACTION-LIST          VALUE 1.
              88 RJ-CSQ-L                VALUE 2.
              88 RJ-CSQ-R                VALUE 3.
           05 RJ-OK                      PIC X.
              88 OK-FOR-ADDED-JOIN       VALUE "Y".
              88 NOT-OK-FOR-JOIN         VALUE "N".
           05 RJ-IS-PTR                  PIC 99.

     01 TEMP-KEY-LIST.
```

Used to ensure that only whole keys are used in forming joins using key replication

```
        03 TKL-MAX                       PIC 999 VALUE 25.
        03 TKL-USED                      PIC 999.
        03 TKL-ENTRY                     OCCURS 25 TIMES
                                         INDEXED BY TKL-INDEX.
           05 TK-KCNO                    PIC 9(6).
           05 TK-TAGNO                   PIC 9(6).
           05 TK-REF-FLAG                PIC X.
              88 TAG-NOT-IN-RJL          VALUE "N".
              88 TAG-IN-RJL              VALUE "Y".

     01 KEY-JOIN-LIST.
```

Used in conjunction with TEMP-KEY-LIST to ensure that only whole keys are used in forming joins using key replication

```
        03 KJL-MAX                       PIC 999 VALUE 25.
        03 KJL-USED                      PIC 99.
        03 KJL-ENTRY                     OCCURS 25 TIMES
                                         INDEXED BY KJL-INDEX.
           05 KJ-TK-PTR                  PIC 99.
           05 KJ-RJ-PTR                  PIC 99.
           05 KJ-RJ-OK                   PIC X.
              88 KCNO-IN-RJL             VALUE "Y".
              88 KCNO-NOT-IN-RJL         VALUE "N".
```

3. Grouping Table

The group table is used to determine which parts of a request belong in each subtransaction.

```
     01 GROUP-TABLE.
          used to identify subtransactions
        03 GROUP-IDENTIFIER                PIC 9(3).
        03 PIVOT-DBID                      PIC 9(5).
        03 PIVOT-RTID                      PIC X(30).
        03 GROUP-MAX                       PIC 99 VALUE 25.
        03 GROUP-USED                      PIC 99 VALUE 0.
        03 GROUP-ROW-SIZE                  PIC 99 VALUE 31.
        03 GROUP-ENTRY OCCURS 25 TIMES INDEXED BY GR-INDEX.
           05 GR-RTID                      PIC X (30).
```

```
05 GR-FLAG                          PIC X.
   88 GR-FLAG-ON                    VALUE "1".
   88 GR-FLAG-OFF                   VALUE "0".
```

4. Temporary Occurrence Table. This table is used to identify all the sets of nested repeating data fields that are required by an NDML statement. It is the source for the entries that are placed in the OCCURS-TABLE.

```
01 TEMP-OCCURS-TABLE.
   03 TOT-MAX                        PIC 99 VALUE 25.
   03 TOT-STACK-MAX                  PIC 99 VALUE 25.
   03 TOT-USED                       PIC 99 VALUE 0.
   03 TOT-OCCURS-NEST OCCURS 25 TIMES INDEXED BY TOT-INDEX-1.
      05 TOT-SUBTRANS                PIC 9(6).
      05 TOT-DBID                    PIC 9(6).
      05 TOT-RTNO                    PIC 9(6).
      05 TOT-NEST-ID                 PIC 99.
      05 TOT-MAPPED-TO               PIC X.
      05 TOT-STACK-USED              PIC 99.
      05 TOT-OCCURS-STACK OCCURS 25 TIMES INDEXED BY
         TOT-INDEX-2.
         07 TOT-DFNO                 PIC 9(6).
         07 TOT-DFID                 PIC X(30).
         07 TOT-COMP-DFNO            PIC 9(6).
         07 TOT-DEP-DFNO             PIC 9(6).
         07 TOT-OCCURS               PIC 9(6).
         07 TOT-INDEX-DFNO           PIC 9(6).
```

None of these tables is input to or output from this function.

5. AUCISM-LIST contains the CS to IS mapings for a given tag.

```
01 AUCISM-LIST.
   03 AUCISM-MAX                     PIC 99 VALUE 20.
   03 AUCISM-USED                    PIC 99 VALUE 0.
   03 AUCISM-PRIM-CNT                PIC 99.
   03 AUCISM-SEC-CNT                 PIC 99.
   03 AUCISM-TAGNO                   PIC 9(6).
   03 AUCISM-ECNO                    PIC 9(6).
   03 AUCISM-ENTRY                   OCCURS 20 TIMES
                                     INDEXED
                                     BY AUCISM-INDEX.
      05 AUCISM-RTNO                 PIC 9(6)
      05 AUCISM-RTID                 PIC X(30).
      05 AUCISM-DBID                 PIC 9(6).
      05 AUCISM-PREF-NO              PIC 9(2).
      05 AUCISM-MAP-TYPE             PIC X(10).
```

6. ELEMENTARY-DATA-FIELD-TBL
Table to hold elementary record definition variables.

```
01 EDF-TABLE.
   03 EDF-MAX                        PIC 999 VALUE 256.
   03 EDF-USED                       PIC 999 VALUE 0.
   03 EDF-DBID                       PIC S9(6).
   03 EDF-RTID                       PIC X(30).
```

```
03 EDF-RTNO                          PIC 9(6).
03 EDF-ENTRY                         OCCURS 256 TIMES
                                     INDEXED
                                     BY EDF-INDEX.
   05 EDF-DFID                       PIC X(30).
   05 EDF-DFNO                       PIC S9(6) COMP.
   05 EDF-OCCURS                     PIC S9(6) COMP.
   05 EDF-REDEF-DF-NO                PIC S9(6) COMP.
   05 EDF-COMPONENT-DF               PIC S9(6) COMP.
   05 EDF-INDEX-IND                  PIC X.
   05 EDF-KNOWN-TO-DBMS              PIC X.
   05 EDF-TYPE                       PIC X.
   05 EDF-SIZE                       PIC 9(3).
   05 EDF-ND                         PIC 9(2).
```

7. The UEC-TABLE contains the union disciminator fields, the meta-data, and the union values for entries which participate in a record union.

```
01 UEC-TABLE.
   05 UEC-MAX                        PIC 99 VALUE 25.
   05 UEC-USED                       PIC 99 VALUE 0.
   05 UEC-ENTRY                      OCCURS 25 TIMES
                                     INDEXED
                                     BY UEC-INDEX.
      07 UEC-DBID                    PIC 9(6).
      07 UEC-RTID                    PIC X(30).
      07 UEC-RTNO                    PIC 9(6).
      07 UEC-DFID                    PIC X(30).
      07 UEC-DFNO                    PIC 9(6).
      07 UEC-TYPE                    PIC X.
      07 UEC-SIZE                    PIC 9(3).
      07 UEC-ND                      PIC 9(2).
      07 UEC-VALUE                   PIC X(30).
      07 UEC-OP                      PIC X(2).
      07 UEC-ECNO                    PIC 9(6).
```

U.S. Government Printing Office: 1992—648-127/62391